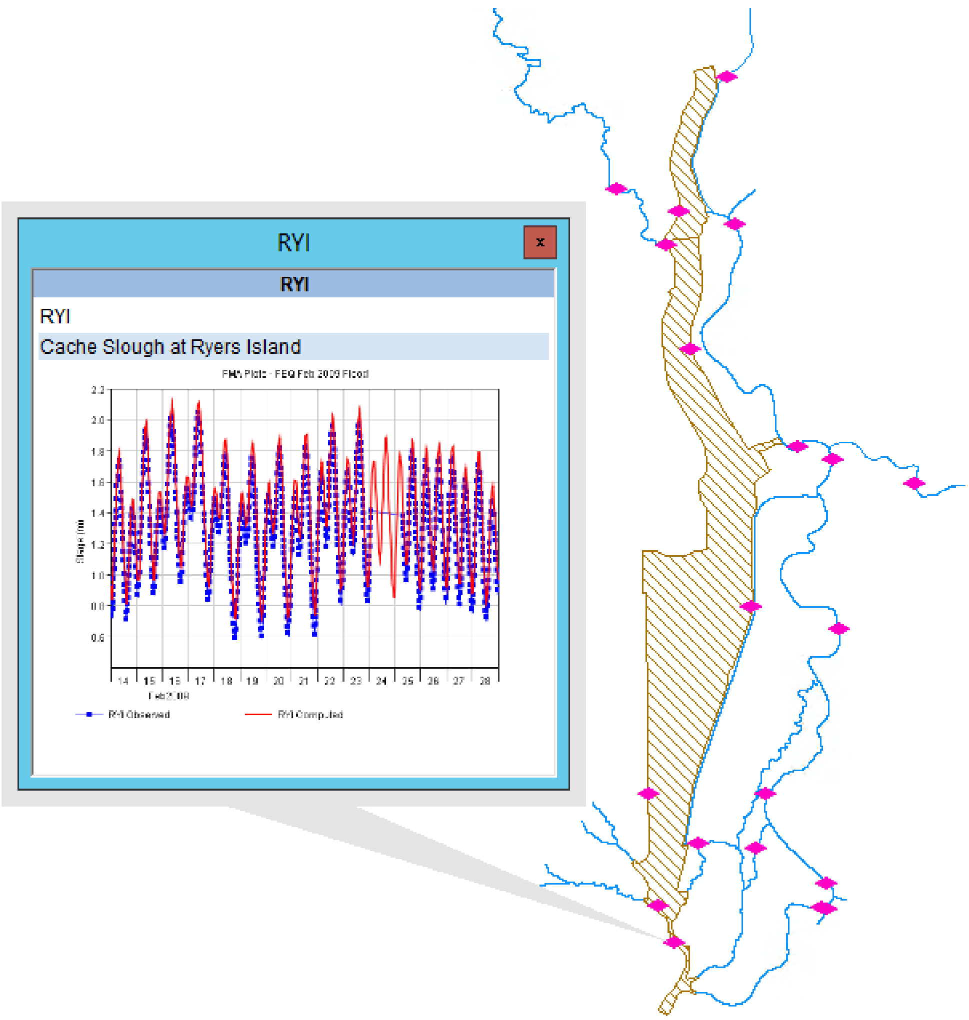


HaD to Py Users' Manual

Tool to get **H**DF **a**nd **D**SS file data in**to** a **Py**thon environment!

What Does HaD to Py Do?

HaD to Py extracts results from the output file from an HEC-RAS simulation and creates plots which can quickly be viewed spatially against observed data for calibration or validation. Extracted data can be saved.

All documents are available online at https://github.com/latomkovic/HaD-to-Py. The script and its dependent files are free to the public. Everything was written by Lily Ann Tomkovic, a graduate student researcher at the University of California Davis, working at the Center for Watershed Sciences.

What Does HaD to Py Need?

To use HaD to Py in the way it was intended, the user should have the following:

- Python
- HEC-RAS 5.0 or earlier
- HEC-DSSVue 2.0 or earlier
- Python 2.7 or earlier
  - Several Python Modules listed later

Further instruction is found within the Users' Manual.

Author: Lily Ann Tomkovic

Graduate Student Researcher

UC Davis, Center for Watershed Sciences

latomkovic@ucdavis.edu

TABLE OF CONTENTS

INTRODUCTION

HaD to Py extracts results from the output file from an HEC-RAS simulation and creates plots which can quickly be viewed spatially against observed data for calibration or validation. Extracted data can be saved.

All documents are available online at https://github.com/latomkovic/HaD-to-Py. The script and its dependent files are free to the public. Everything was written by Lily Ann Tomkovic, a graduate student researcher at the University of California Davis, working at the Center for Watershed Sciences.

Several people helped along the way to make this tool possible, including but not limited to: Bill Fleenor at UCD, and Joan Klipsch, Gary Brunner, Bill Charley, Mike Perryman, and Tom Evans at USACE-HEC.

For help understanding how to use the code properly you can contact Lily Ann Tomkovic at latomkovic@ucdavis.edu

REQUIREMENTS

To use HaD to Py in the way it was intended, the user should have the following installed on their machine:

- Python
- HEC-RAS 5.0 or earlier
- HEC-DSSVue 2.0 or earlier
- Python 2.7 or earlier
    - Several Python Modules listed later

The user will also need to designate a folder or working directory with all of the files included in the download package.

DOWNLOAD PACKAGE CONTENTS AND DESCRIPTION

FILE	DESCRIPTION
pyHDF_DSS.py	The main code which the user will run in order to get the plots/data
storeDSSdata.py	A Jython code which will be peripherally run by HEC-DSSVue from within the main code It can store records to a DSS file path
getDSSdata.py	A Jython code which will be peripherally run by HEC-DSSVue from within the main code It accesses DSS records and outputs the values and times to a temporary txt file found in the temp_files folder
obs_paths.txt	File needed to provide observed data DSS path to the script. See *obs_path Text File* section for correct formatting and more information on its use
one_dim_comp_paths	File needed to provide the computed one-dimensional data DSS path to the script. See *one_dim_comp_paths Text File* section for correct formatting and more information on its use
two_dim_coords.txt	File needed to provide the geographic coordinates of the gages found in RAS 2D Flow Areas. See *two_dim_coords* Text File section for correct formatting and more information on its use
temp_files • DSS_data_input.txt • DSS_data_output.txt • ras_2D_cells.txt	A folder containing temporary txt files • The temporary input file for storing DSS data • The temporary output file for getting DSS data • A temporary file which contains pertinent information on the gages found in 2D flow area

Tutorial
 	
• txtFiles
  ○ obs_paths
  ○ one_dim_comp_paths
  ○ two_dim_coords
• BaldCreek2D.dss
 	
• BaldCreekObservedData.dss
 	
• BaldEagleGages (.cpg, .dbf, .sbn, .sbx, .shp, .shx)
	The folder which contains all of the files necessary for the Tutorial Example
	• Where the completed txt files are located for the tutorial
	  ○ The observed data DSS pathnames
	  ○ The computed 1D RAS DSS paths
	  ○ The spatial coordinates of the gages within the 2D areas
	• An empty DSS file where the code can store 2D data that it harvests
	• A DSS file which has (fabricated for example) observed data to compare to the gages
	• A shapefile which has (fabricated for example) gages for the tutorial

INSTRUCTIONS

CONCEPT

The main idea is pretty simple, there are observed data entries in a DSS file, and an HEC simulation with one-dimensional and two-dimensional computed data entries in a separate DSS file and an HDF file, respectively. HaD to Py takes the data from the HDF and DSS files and handles them in Python.

HaD to Py 1.0 was initially developed to handle water surface elevation plots, but it would be rather simple to modify the code and allow other variables to be shown.

ITEM TO PLOT	LOCATION
Observed Data	An Observed Data DSS File
1D Computed Point	The HEC-RAS Output DSS File
2D Computed Point	The HEC-RAS Output HDF File

FILE REQUIREMENTS

Before you start using the code, you need to be sure that the required files and modules are downloaded/imported and in the right places.

It is best to take all of the files found in the *Download Package Contents and Description* section of the *Introduction* into the working directory folder, or the main folder where you intend to operate this code. This will become clearer in the *File Requirements* section.

PYTHON MODULES REQUIRED

os, numpy, scipy, Tkinter, tkMessageBox, subprocess, h5py, matplotlib.pyplot

All of the above modules need to be downloaded/available to your Python interpreter.

JYTHON FILES FOR HEC-DSS

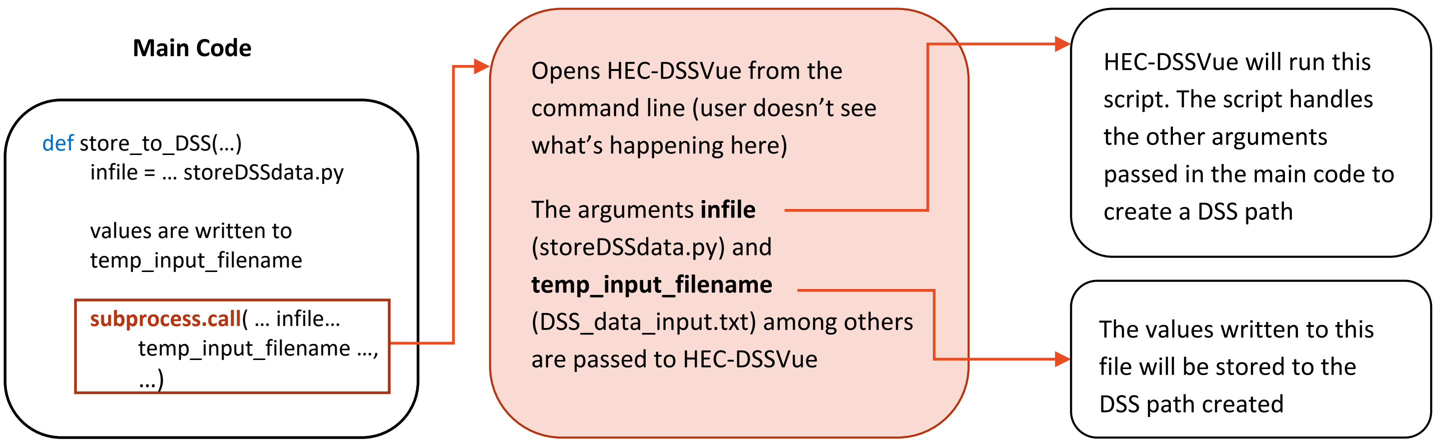
There are two files which HEC-DSSVue will use when the main Python code passes them through the command line. One file retrieves DSS data and the other writes a DSS record to a file.

Unless you know what you're doing, it is not recommended that you change or delete these files. For more information on what these files do, see the following sections for more information.

STOREDSSDATA.PY

In the main code (see *Main Code* section) there's a function, **store_to_DSS** which passes this script to HEC-DSSVue in the command line. In order to handle the arguments in the way that the HEC-DSSVue scripting environment enjoys, a txt file in the temp folder is also passed in which contains the actual data to be stored.

The storeDSSdata.py script handles the data given to it within the HEC-DSSVue environment and creates a DSS path and stores data to it, see below.

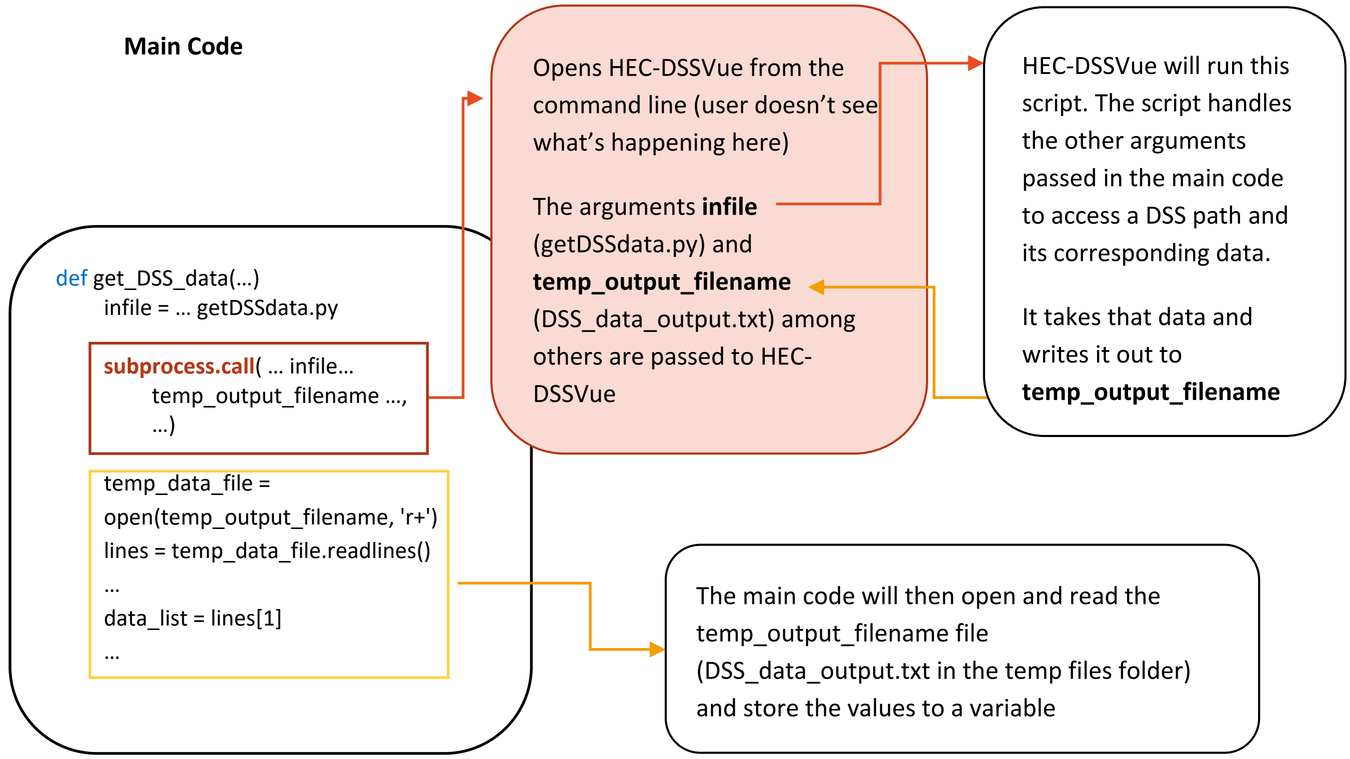


The functions found in storeDSSdata.py are necessary because when HEC-DSSVue reads the arguments passed to it, it treats spaces as a delimeter. For instance, a file path that looks like "C:/Users/Test/My Documents/Python" will be read as 2 strings: "C:/Users/Test/My" and "Documents/Python".

GETDSSDATA.PY

In the main code (see *Main Code* section) a function, **get_DSS_data** passes this script to HEC-DSSVue in the command line. The script is used by DSSVue to produce a temporary output file that the **get_DSS_data** function reads and stores to a variable in Python.

Similar to the storeDSSdata.py script, the getDSSdata.py script has functions that are necessary to compile DSS filenames, pathnames, and time windows in order to use the HEC scripting conventions to access and output path data. See the diagram below for more clarity.



OBS_PATH TEXT FILE

The obs_path text file contains a simple list of the observed data paths that the code will use to retrieve data from the simulation period.

In the main code (see *Main Code* section) a function opens the obs_path.txt file and reads the pathname corresponding to the gage argument which is passed through the function.

The following demonstrates what the file should look like so that the **get_obs_path** function works properly:

```
Gage: SUT Path: /SUTTER SLOUGH AT COURTLAND/SUT/STAGE/31DEC2005/IR-DAY/USGS/
Gage: VON Path: /SACRAMENTO RIVER AT VERONA/VON/STAGE/01FEB2005/1HOUR/USGS DWR/
Gage: YBY Path: /YOLO BYPASS NEAR WOODLAND/YBY/STAGE/28FEB2005/IR-DAY/USGS/
```

The important points of this file are highlighted with the boxes and arrows:

1. The words "Gage: " and "Path: " must only be used to indicate the gage name and pathname.
   a. i.e. the pathname shouldn't contain "Gage: " or "Path: " as they are used as delimiters to parse the data
2. There are no line breaks between entries
3. There is one space at the end of the file
   a. Again, the code uses a line break to delimit the pathnames

The user is required to correlate the observed data DSS path to a specified gage. See *Enter obs_paths.txt Data* Section for an example of entering data correctly.

ONE_DIM_COMP_PATHS TEXT FILE

In the main code (see *Main Code* section) a function, **get_all_plot_data,** uses the one_dim_comp_paths.txt file and reads the pathname corresponding to the gage argument which is passed through the function.

The one_dim_comp_paths text file has a similar appearance to the obs_path file, but has a small distinction.

If there are no 1D gages, then this file should be completely empty, but the file should still exist.

The following demonstrates what the file should look like so that the **get_all_plot_data** function works properly:

```
Gage: SSS Path: /STEAMBOAT SLOUGH SAC TO SUTTER/16027.84/STAGE/%s/%s/%s/
Gage: SUT Path: /SUTTER SLOUGH SAC TO ELK/10665.02/STAGE/%s/%s/%s/
Gage: VON Path: /SACRAMENTO NCC TO AMERICAN/106605.4/STAGE/%s/%s/%s/
```

The important points are the same as before.

The distinction between this and obs_path is that the last 3 fields of the pathname have %s instead of any value. These three values will change based on the output of the RAS simulation.

For instance, if the simulation time period is from 14FEB2009 10:00 – 23FEB2009 23:00, and the hydrograph interval was set to 15MIN, and the short ID of the plan was TESTING, then the last three fields will look like:

…/01FEB2009/15MIN/TESTING/

The **get_all_plot_data** function will assign variables to the pathnames which will then be used to access computed data.

Tthe user is required to correlate the observed gage location to a RAS 1D Cross section. See *Enter one_dim_comp_paths.txt Data* Section for an example of entering data correctly.

TWO_DIM_COORDS TEXT FILE

This contains the latitude and longitude (in ***meters***) for each gage that is found in a 2D area. Currently, the code will find the cells that correlate to the gage coordinates and store those cell indices for each version of the geometry you use.

This is particularly helpful for times where you are changing the cell size or arrangements of cells in a 2D area.

In the main code (see *Main Code* section) a function, **get_coordinates,** uses the two_dim_coords.txt file and stores the coordinate pairs corresponding to the gage argument which is passed through the function.

Below is an example of the format of this file required for the **get_coordinates** function to work properly:

```
Gage: SBP Lat: 616878.502335 Lon: 4293808.60825
Gage: YBY Lat: 617940.666062 Lon: 4281881.68027
Gage: LIS Lat: 623108.260000 Lon: 4256436.96000
```

Important points on this file are:

1. There should be three "fields" and they are "Gage:", "Lat:", and "Lon:"
2. Lat and Lon should be in meters, not decimal degrees, or any other coordinate system
3. There should be a blank line at the end

What the user needs to do here is to correlate the observed gage location to a RAS 1D Cross section. See *Enter two_dim_coords.txt Data* Section for an example of entering data correctly.

DSS_DATA_INPUT TEXT FILE

This is a temporary file. It does not need to contain anything, nor does it need to exist. The main code will write the file in the temp_folder location within the working directory.

The Jython scripts that communicate with HEC-DSSVue use this text file to relay Python variables to DSS and store into a DSS path.

DSS_DATA_OUTPUT TEXT FILE

This is a temporary file. It does not need to contain anything, nor does it need to exist. The main code will write the file in the temp_folder location within the working directory.

The Jython scripts that communicate with HEC-DSSVue use this text file to relay the data back to the main Python code. It is how the code accesses the values in the DSS file.

RAS_2D_CELLS TEXT FILE

The main code will output this file once it finds the closest (spatially) cell index to the gage point, the distance from the cells center to the gage, and the 2D Flow Area where the cell resides. The first line indicates which .p##.hdf file from which the indices were extracted, and the next lines correlate to each gage location.

If the code has already run for that hdf file, it will use the already stored locations, otherwise it will find the cell indices. This file needs to exist in the temp_folder location within the working directory.

MAIN CODE

The main code is in the py_HDF_DSS.py file found in the download package.

The user will need to change 4 lines for their purposes:

```
hdf_filename = 'C:\...\RAS_Project\ProjectName.p01.hdf'
obs_dss = r'C\...\ObservedDataFile.dss'
twoD_dss = r'C:\...\Example2D.dss'
plot_dir = r'C:\Users\lt\Documents\Python\Plots '
```

HDF_FILENAME

This needs to be the hdf file corresponding to the plan you want to evaluate, and it should be located within the RAS project folder (i.e. don't move this file to another location to run the script).

OBS_DSS

The path and DSS file where the observed data reside. It can be named anything (it doesn't need to be named ObservedDataFile.dss, in case you were worried).

TWOD_DSS

The path and DSS file where the code will store data derived from the 2D coordinates, or the gages. It can be a blank file, or a file which already has some entries.

PLOT_DIR

The plot_dir variable describes where the plot printed graphs (*.png) will be saved. The script allocates a special folder for each plan shortID and stores all of the graphs within the plot directory folder.

For example, if the gage is "HEY" and the plan is "TEST PLAN" then the plot for "HERE" will be located in:

plot_dir\TEST PLAN\HEY.png

TUTORIAL EXAMPLE

The tutorial example provided uses a project found in the example projects folder of HEC-RAS. If the projects haven't already been installed, you can go to:

http://www.hec.usace.army.mil/software/hec-ras/downloads/Example_Projects.exe to download them.

The first tutorial uses the Bald Eagle Creek example project found in Example Projects\2D Unsteady Flow Hydraulics\BaldEagleCrkMulti2D. Specifically it uses the 1d-2D Dambreak Refined Grid plan (p15).

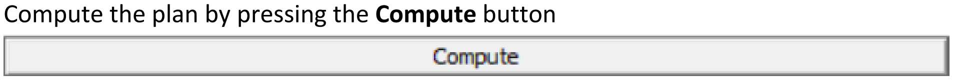
The tutorial document will walk you through how to enter the data into the txt files, but if you want to just see the code work, the correct files are located in the Tutorial/txtFiles folder. Just take those files and copy them over the files in the working directory.

OPEN RAS PROJECT

To start, you'll need to open and run the plan in RAS. Open the project by going to the folder listed above from within RAS.

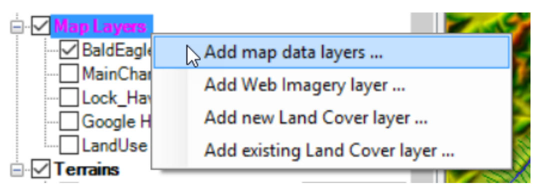
RUN RAS PROJECT

Select the Unsteady Flow Analysis Editor  and choose **File > Open** 1d-2D Dambreak Refined Grid

- Go to **Options > Stage and Flow Output Locations…** and select Bald Eagle Cr. Lock Haven 103189 and add to the list using the arrow 
  This creates an output hydrograph at the location at one of our gages
  - Press **Ok**
- Compute the plan by pressing the **Compute** button
  

DISPLAY GAGE SHAPEFILE

(Optional) While the simulation is running you can see the location of the gages by adding the gage shapefile to RAS Mapper



- From the main RAS window, select RAS Mapper 
  - Right click on **Map Layers**
    - Select **Add map data layers…**
    - Navigate to the **BaldEagleGages.shp** file found in the download package under the Tutorial folder
    - (extra bonus) To add labels:
      - Double-click on BaldEagleGages to open the Layer Properties
      - Select Label Features with Attribute Column(s)
      - Click Edit
        - Choose StationID in the dropdown under Attribute Text

Now you can see where the gages are in the geometry

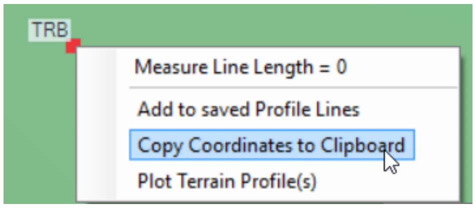
ENTER TWO_DIM_COORDS.TXT DATA

You can see in RAS Mapper that 4 gages are in the 2D Flow Area and 1 is on a 1D Cross Section. In order to get the data from the 2D Flow Area cell points, we need to enter gage coordinates.

From the working directory (where the download files are found, and where this file is found!) open the **two_dim_coords.txt** file

For a description of this file, go to *the* *two_dim_coords* Text File section.

- The coordinates are Latitude and Longitude in *meters*. This is how RAS finds points.
- There are 2 ways to get the XY coordinates from the shapefile points:
  - Populate the XY coordinates of the shapefile in ArcGIS using the Add XY Coordinates tool
  - Using RAS Mapper:
    - Zoom in pretty tight on a gage
    - Select the Measuring tool
    - Double-click with (left-click) on the gage point
      - Select Copy Coordinates to Clipboard
      - Paste them somewhere and you can choose either row (they should be about the same)



- Enter a Latitude and Longitude for the 4 gages found in the 2D Flow Area using the format:

```
Gage: SBP Lat: 616878.502335 Lon: 4293808.60825

```

* Make sure to leave an extra blank line at the end!

- For this tutorial the Coordinates are listed here:

Gage	Latitude	Longitude
BDL	2012154.86014	326091.04372
FAR	2082421.16365	365397.77488
NES	2071093.97205	359607.937664
TRB	2044957.56598	347579.767861

ENTER OBS_PATHS.TXT DATA

For more information on the text file go to the *obs_path Text File* section.

Now tell the code where to look for the observed data. Each gage needs an observed data path, and currently, all the observed data need to be in the same DSS file.

The Observed DSS data file can be found in the working directory in the Tutorial folder, it is called **BaldCreekObservedData.dss**

You can use the DSS file to fill out the **obs_paths.txt** file found in the working directory.

- Enter the pathnames in the following format:

```
Gage: SUT Path: /SUTTER SLOUGH AT COURTLAND/SUT/STAGE/31DEC2005/IR-DAY/USGS/

```

* Again, make sure to leave an extra blank line at the end!

- For the tutorial, the paths are listed here:

Gage	Pathname
BDL	/BDL/BELOW THE DAM/STAGE/01JAN1999/1HOUR/EXAMPLE DATA/
FAR	/FAR/FARTHEST POINT/STAGE/01JAN1999/1HOUR/EXAMPLE DATA/
NES	/NES/NEAR THE STREAM/STAGE/01JAN1999/1HOUR/EXAMPLE DATA/
TRB	/TRB/TRIBUTARY/STAGE/01JAN1999/1HOUR/EXAMPLE DATA/
XSG	/XSG/IN-RESERVOIR/STAGE/01JAN1999/1HOUR/EXAMPLE DATA/

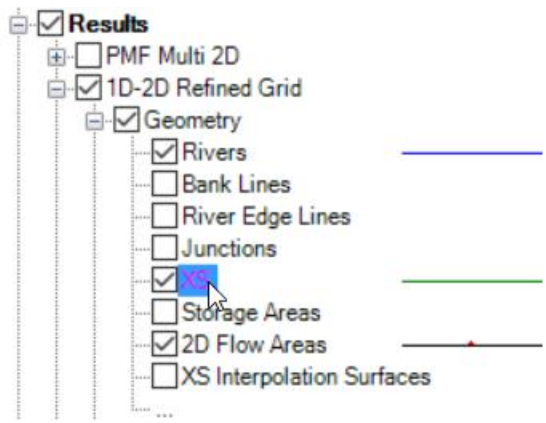
ENTER ONE_DIM_COMP_PATHS.TXT DATA

This is the last text file you need to fill out. This tells the code where to look for the computed data in a 1D reach.

For more information go to the *one_dim_comp_paths Text File* section.

You can find which cross section you should select by following these steps:

- In RAS Mapper, expand the Results tree for the 1D-2D Refined Grid Plan
  - Expand Geometry
    - Select XS
    - Left Click on XS
  - Zoom in on the XSG gage (on the XS)
  - Select the pointer tool 
    - Right click the Cross Section which is on the gage
    - A pop-up will appear and has the name of the XS node



Then, you'll need to find the DSS pathname, which you can do by selecting the DSS button  in the main RAS window, then navigating to the correct path.

- Use the following format to enter the pathname:

```
Gage: SSS Path: /STEAMBOAT SLOUGH SAC TO SUTTER/16027.84/STAGE/%s/%s/%s/

```

* Again, make sure to leave an extra blank line at the end!

The path for this tutorial is: /BALD EAGLE CR. LOCK HAVEN/103189/STAGE/%s/%s/%s/

The %s for the last three allow the code to enter the appropriate path parts which correspond to the simulation options (time window, hydrograph output interval, and plan short ID)

MODIFY PY_HDF_DSS.PY PREAMBLE

The description of the different parts of the preamble are found in the *Main Code* section.
Open the **pyHDF_DSS.py** python script in the working directory.
Now you'll need to change 5 lines in the preamble under "# User Input"

HDF_FILENAME

This will correspond to the plan file where the data reside. In this case, it's the **1D-2D Refined Grid** plan which has extension .p15

If your example projects folder is located in the Documents folder then this line will look like this:

```
hdf_filename = r'C:/Users/lt/Documents /Example Projects/2D Unsteady Flow Hydraulics/BaldEagleCrkMulti2D/BaldEagleDamBrk.p15.hdf'
```

Be sure to use an r before the quote, this avoids escape characters in the filename.

OBS_DSS

This is the observed data DSS file which will be used to make the observed v. computed plots.

For this tutorial, it will be located in the working directory (where the download package was installed) in the Tutorial folder. If your working directory is C:/Documents/HaDtoPy then the obs_dss line will look like this:

```
obs_dss = r'C:/Users/lt/Documents/HaDtoPy/Tutorial/BaldCreekObservedData.dss'
```

TWOD_DSS

The twoD_dss.dss file is where the code will store the 2D results as a DSS element.

For this tutorial, it will be located in the working directory in the Tutorial folder, as well. Similar to above, it will look like:

```
twoD_dss = r'C:/Users/lt/Documents/HaDtoPy/Tutorial/BaldCreek2D.dss'
```

PLOT_DIR

The plot directory, or plot_dir, is the folder where you would like the resulting figures to go. Each time the code is run, it will create a folder with the RAS plan short ID as the folder name, then output printed graphs (*.png) with the gage name as the image name within that folder.

You can put this folder wherever you'd like. This is an example of where this could go:

```
plot_dir = r'C:/Users/lt/Documents/HaDtoPy/Figures'
```

For this tutorial, if you were to use the above path (with your username instead of 'lt') the plot for XSG will be in this path: C:/Users/lt/Documents/HaDtoPy/Figures/1D-2D Refined Grid/XSG.png

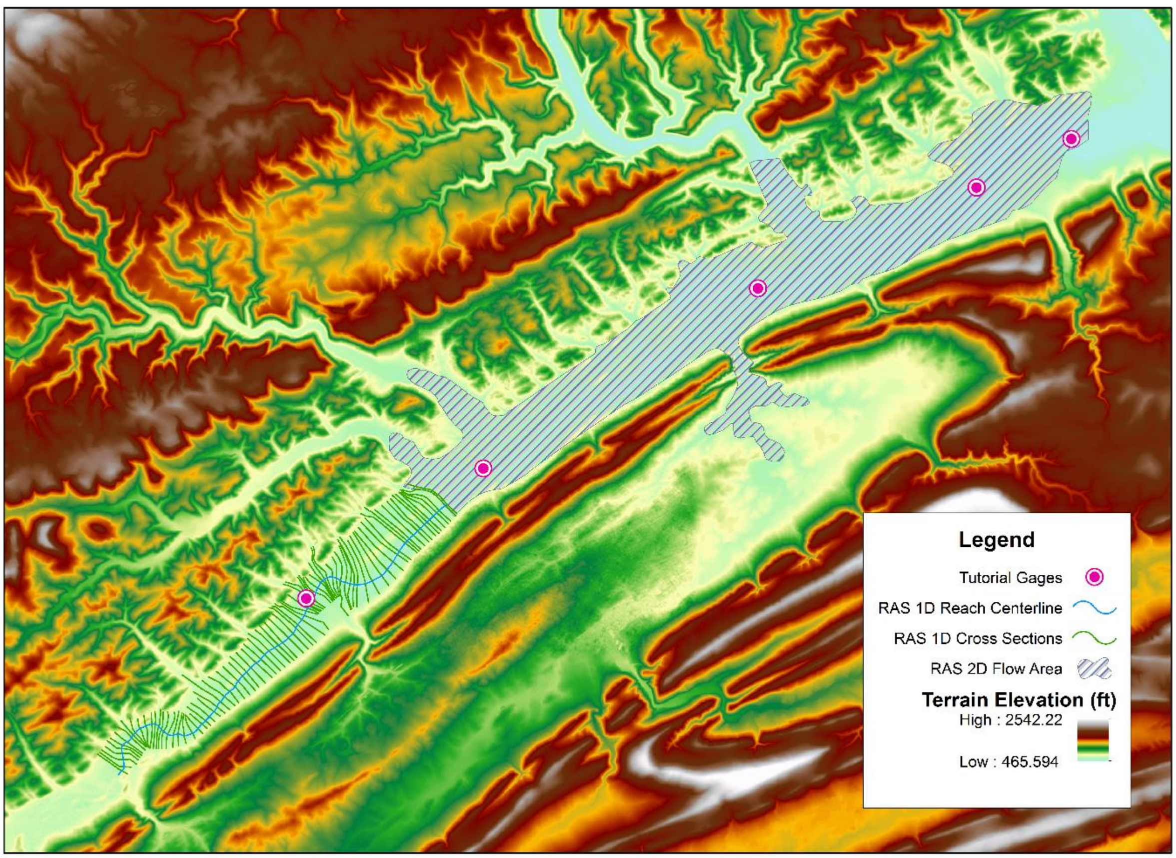
RUN THE CODE!

Once you run the pyHDF_DSS.py script, it will produce 5 plots in whichever directory you chose. After the script has run, you can view the plots by navigating to the plot directory in your file explorer.

VIEWING THE PLOTS INTERACTIVELY WITH GIS

(Optional) This is a quick walk-through of how one would go about making the plots appear spatially in an interactive fashion.

The screenshots in this section have shapefiles that are not included in the HaD to Py download, but are easily exported from RAS Mapper (i.e. the 2D Area and the 1D Reach and Cross Sections). This section is shown in ArcMap but the same task can be accomplished in QGIS, a free and open-source GIS platform.

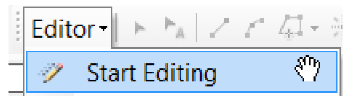


Above, you can see the layout of the gages, and the RAS geometry.

If you open the attribute table of the Tutorial Gages (the BaldEagleGages shapefile found in the Tutorial folder within the working directory), you can see the Images field which has the html code for the gage plot.

FID	Shape *	Id	POINT_X	POINT_Y	StationID	Image
0	Point	0	1990994.36716	310580.861243	XSG	
1	Point	0	2012154.86014	326091.04372	BDL	
2	Point	0	2044957.56598	347579.767861	TRB	
3	Point	0	2071093.97205	359607.937664	NES	
4	Point	0	2082421.16365	365397.77488	FAR	

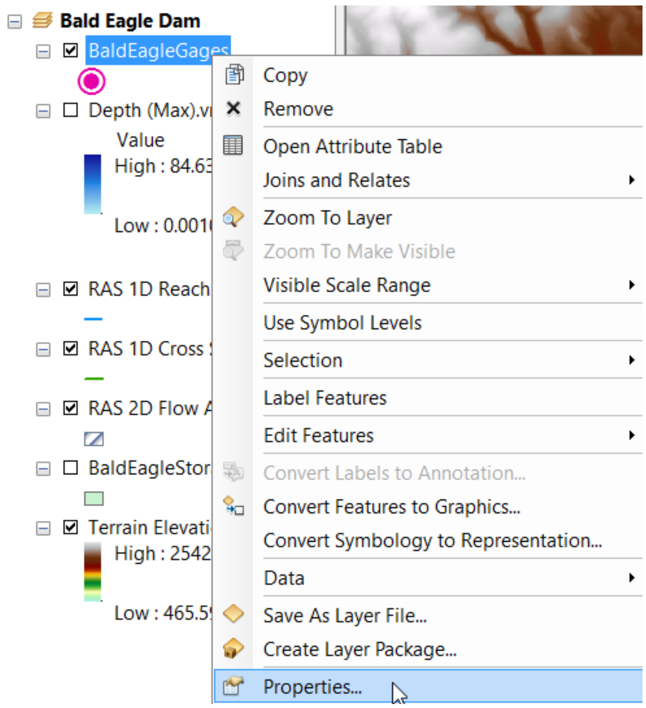
In order to change these field attributes, begin editing by Selecting **Editor> Start Editing**. This allows you to edit each attributes Image field individually.



For each image, you'll want to modify the highlighted text in the code below to match the file location of your plot:

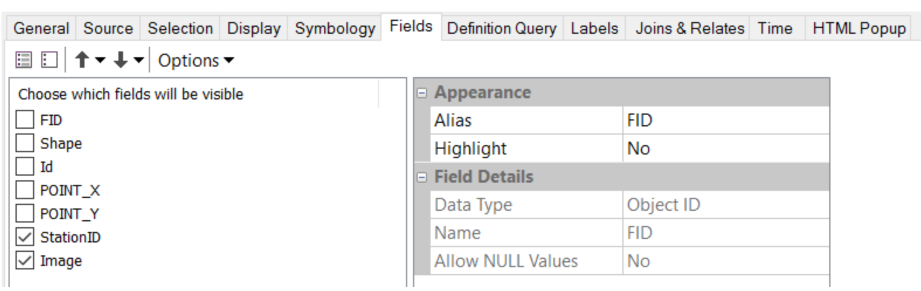
Once all of the fields are changed, just select **Stop Editing** and **Save Edits**.

Now to get the popups in ArcMap!

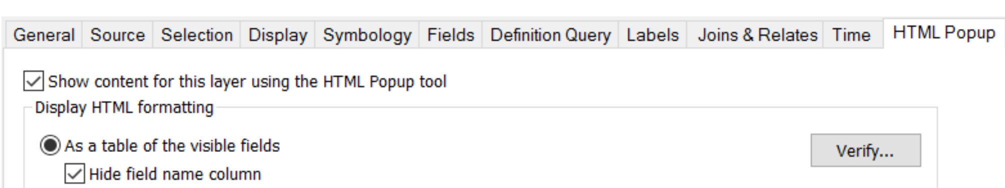


Right-click on the BaldEagleGages layer (see left figure) in the Table of Contents (TOC) and select **Properties...**

Navigate to the **Fields** tab, and uncheck everything but StationID and Image (this makes the pop-up less cluttered).



Navigate to the **HTML Popup** tab, and check the "Show content for this layer using the HTML Popup tool" box, and activate the "As a table of the visible fields" radio button, and check the "Hide field name column" box.



Press Ok and return to the regular GIS screen. Select the HTML Popup tool:

Now, when you click on the points in the BaldEagleGages shapefile, a popup will appear which has the stationID and the plot.

Below is an example of what a couple of open popups would look like.