pyHMT2D: Python-based two-dimensional hydraulic modeling tools

Xiaofeng Liu[*1]

1 Department of Civil and Environmental Engineering, Institute of Computational and Data Sciences, The Pennsylvania State University

DOI: DOIunavailable

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Reviewers:

- @Pending Reviewers

Submitted: N/A
Published: N/A

Summary

Efficient and accurate flood simulation is increasingly important to human society as climate keeps changing and floods of all magnitudes seem to occur more frequent. Two-dimensional (2D) hydraulic models, replacing one-dimensional (1D) models, have become the work horse for most academic researches and engineering designs in practice. Many agencies, such as U.S. Department of Transportation, Bureau of Reclamation (USBR), FEMA, and U.S. Army Corp of Engineers (USACE), have developed and promoted 2D hydraulic models to fulfill their missions. Example 2D models are SRH-2D by USBR (Lai, 2010) and HEC-RAS 2D by USACE (Brunner, 2016). Over the past several decades, these models have been greatly improved by their respective agencies independently. However, limitations still exist in these individual models. For example, it is desirable to have fair and meaningful model result inter-comparison. But due to the differences in numerical schemes and model approximations, such comparison is currently difficult, if not impossible. Another example of limitation is that most of these models evolved from legacy codes. They lack the flexibility to cope with the rapidly evolving landscape of computational science and utilize the power of modern computational ecosystems enabled by for example Python.

`pyHMT2D` is a Python package developed to partially remove these limitations. It stands for "2D Hydraulic Modeling Tools with Python." Objected-oriented programming is utilized to encapsulate the hydraulic models and simulation data (Fig. 1). The whole package is made of three submodules, namely `Calibration`, `Hydraulic_Models_Data`, and `Misc`. A user only needs to interface with `pyHMT2D` using the high-level APIs implemented in a handful of classes in the package. In addition to these core classes, `pyHMT2D` also provides a suite of miscellaneous tools to perform various tasks essential to hydraulic modeling, such as the creation and manipulation of geo-referenced terrain data. To compare model results and provide an alternative visualization means, VTK (Schroeder et al., 2006) is used as a common format. `pyHMT2D` can read results from all supported hydraulic models and translates them to VTK. Results in VTK format can be opened in ParaView (Ayachit, 2015) for visualization and further analysis. Another feature that makes `pyHMT2D` more attractive is the use of Python as a "glue" language to expose all supported hydraulic models to the large amount of powerful optimization packages. With them, `pyHMT2D` can facilitate automatic and more intelligent model calibrations. In summary, `pyHMT2D` can be used to control and (semi)automate 2D hydraulic modeling, pre-/post-process simulation results, convert simulation results into the unified format of VTK, and perform automatic calibration.

[*]Corresponding author

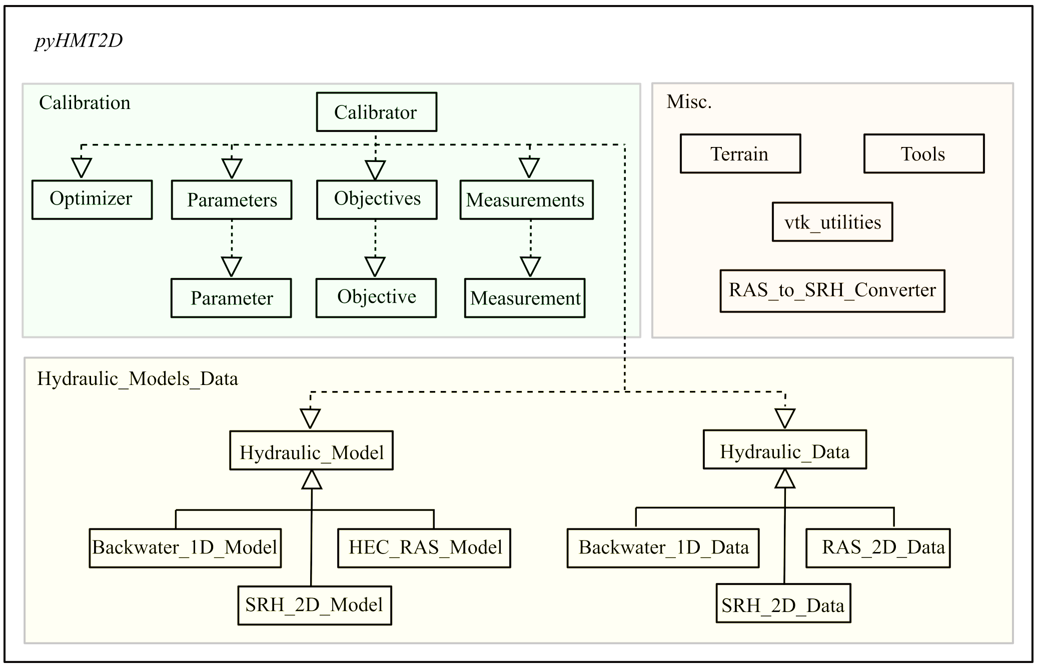

Figure 1: Package structure and scheme diagram for `pyHMT2D`.

Statement of need

Floods are one of the most common natural hazards which affect hundreds of millions of people worldwide and cost billions of dollars in damage. In the United States, about 40% of recorded 35,000 disasters since 1900 were major floods and related storms (Cigler, 2017). With climate change, floods are becoming more frequent and severe. To better understand the dynamics of floods and mitigate their impact, hydraulic models have been used extensively. In engineering practice, hydraulic models are also widely used to design and evaluate infrastructure, such as bridges and roads, against the damaging effect of flowing water. Flow in rivers and streams is naturally three-dimensional. However, it is computationally prohibitive to simulate three-dimensional flow field for large scale applications. Two-dimensional hydraulic modeling is the prevailing approach for both engineering practice and research. Many 2D models exist and the fundamental governing equations implemented are very similar. Indeed, the depth-averaged shallow water equations are the backbone of most 2D hydraulic models (Liu et al., 2008). However, different 2D hydraulic models have their own unique numerical schemes, mesh and result format, and even some variations to the governing equations. The diverse, isolated, and mostly incompatible ecosystems surrounding these models create some problems in practice.

Due to the importance of 2D hydraulic models to the society for flood prediction, prevention, mitigation, and engineering design of infrastructure, there is a great need to make them comparable. In addition, as for all other computational models, there is a perpetual need to make them more efficient, easy to use, and automatic to be calibrated. Therefore, the motivations of developing `pyHMT2D` are as follows:

- One major motivation is to efficiently and automatically run 2D hydraulic modeling simulations, for example, batch simulations to intelligently and efficiently calibrate models. Many 2D models have some automation in calibration. However, these models and their GUIs are closed source. Therefore, a modeler is limited to what he/she can do. Automated calibration using the Python code in `pyHMT2D` makes it possible to define any calibration error functions and utilize any optimization packages. It also makes it possible for these models to better ingest vast amount of data generated by millions of sensors deployed in the field in real time.

- pyHMT2D also serves as a bridge between 2D hydraulic models and the Python universe where many powerful libraries exist, for example statistics, optimization, machine learning, GIS, and parallel computing.
- pyHMT2D can make model inter-comparison and evaluation an easy task. Almost all 2D hydraulic models solve the shallow-water equations. However, every model does it differently. How these differences manifest in their results and how to quantify/interpret the differences are of great interest to practitioners.
- The read/write and transformation of 2D hydraulic model results can be used to feed other models which use the simulated flow field, for example external water quality models and fish migration models.
- Most 2D models have good user interface and they have the capability to produce good visualizations and analysis using their proprietary formats. However, with pyHMT2D and the power of the VTK library, 2D hydraulic modeling results can be visualized and analyzed with more flexibility and efficiency in VTK format.

Previously, tools have been developed to control hydraulic models and process their simulation results. Most of these existing tools use scripting languages, such as VBA, Matlab, and Python. For example, Moya Quiroga et al. (2013) showed the evaluation of HEC-RAS modeling uncertainties with the combination of Delphi, VB.NET, and Windows command line batch scripts. In Goodell (2014) and its companion code, VBA was used to interact with HEC-RAS and a variety of demonstration examples were provided. Leon & Goodell (2016) developed several MATLAB scripts for controlling HEC-RAS, which include input file reading and writing, output file reading, plotting, and parallel computation. Similar implementation using Python has been reported in Dysarz (2018). If one searches on GitHub with the keyword "HEC-RAS," there are many repositories and code snippets which implement certain functionality mentioned above. All these codes are for one particular model only. On the other hand, pyHMT2D is designed to be more general and extensible. The class hierarchy shown in Fig. 1 can easily add other hydraulic models into pyHMT2D. Model inter-comparison functionality is also missing in most existing tools.

Functionality

pyHMT2D requires the user to have some basic knowledge about Python programming, which is easy to acquire because of Python's intuitiveness. A detailed user manual for pyHMT2D can be found in its repository on GitHub. API documentation for pyHMT2D is also provided on GitHub Pages. And of course, the user should have knowledge on how to operate the particular 2D hydraulic models that he/she wants to use pyHMT2D for. For most of the functionality, the user needs to first create a simulation case using a 2D hydraulic model, run the case, and obtain some results. The creation of a simulation case from scratch with pyHMT2D is currently not supported.

Basic Functionality

In general, the basic functionality of pyHMT2D is to read simulation results, convert to different format, and control simulation runs. Specifically, for SRH-2D modeling:

- read SRH-2D results (mainly into Python's Numpy arrays) through the `SRH_2D_Data` class.

- convert SRH-2D results to VTK format, one of the most popular formats for scientific data. The conversion is through the `SRH_2D_Data` class and the VTK library.

  - point and cell center data (water, depth, water surface elevation, velocity, and all other solution variables from SRH-2D)
  - interpolate between point and cell data

- sample and probe converted simulation results in VTK format with the VTK library.
- control and modify SRH-2D simulations through the `SRH_2D_Model` class.

For HEC-RAS 2D modeling, the basic functionality includes:

- read HEC-RAS 2D results (mainly into Python's Numpy arrays) through the `RAS_2D_Data` class.
- convert HEC-RAS 2D results to VTK format through the `RAS_2D_Data` class and the VTK library. In addition to the similar functions as SRH-2D, the `RAS_2D_Data` class can also output the following:
  - face data (e.g., subterrain data), which is one of the features of HEC-RAS.
- convert HEC-RAS 2D mesh, boundary conditions, and Manning's roughness n data into SRH-2D format such that SRH-2D and HEC-RAS 2D can run a case with exactly the same mesh for fair comparison. Additionally, with `pyHMT2D`, HEC-RAS 2D can be used as a mesh generator for SRH-2D.
- sample and probe simulation results in the converted VTK format with the VTK library.
- control and modify HEC-RAS 2D simulations through the `HEC_RAS_Model` class. The control is through the Python library pywin32 (Hammond & Robinson, 2000). `pyHMT2D` interfaces with HEC-RAS through the Component Object Model (COM), which is the foundation for Microsoft's OLE and ActiveX technologies.

As an example of the basic functionality, Fig. 2 shows the comparison of simulated water depth from SRH-2D and HEC-RAS in ParaView using the `pyHMT2D` package. This example is in the "examples/compare_SRH_2D_RAS_2D" directory of the repository on GitHub. The case was first created and simulated in HEC-RAS. Then the HEC-RAS result was read in with the `RAS_2D_Data` class, converted to VTK format and to SRH-2D mesh format. SRH-2D was invoked to run the converted case, whose result was then converted to VTK too. Both VTK results from HEC-RAS and SRH-2D were loaded into ParaView for comparison.

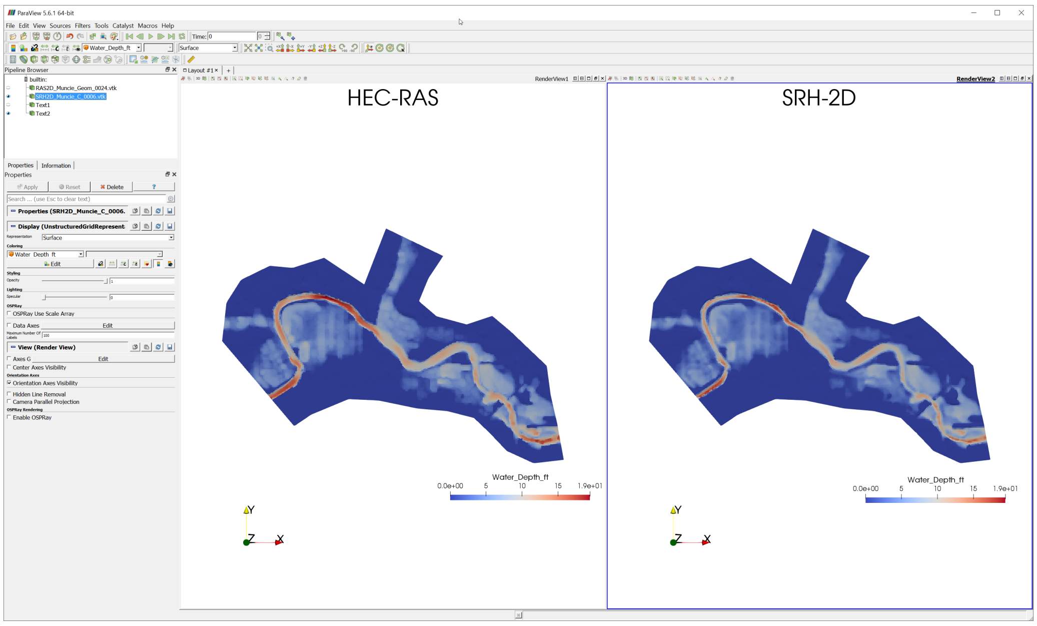

Figure 2: Comparison of simulated water depth from SRH-2D and HEC-RAS in ParaView using the `pyHMT2D` package. The case was first created in HEC-RAS and converted to SRH-2D. Thus, the comparison is based on exactly the same mesh.

Advanced Functionality

Advanced functionality of `pyHMT2D` is built upon its basic functionality. With the control capability above, it is possible to perform the following tasks among many others:

- automatic calibration of models with available optimization and calibration Python packages. Currently, `scipy`'s `optimize` module is supported, which includes many local and global optimization methods. The automatic calibration is mainly through the `Calibration` submodule, which includes five major classes, namely `Calibrator`, `Optimizer`, `Parameters`, `Objectives`, and `Measurements`. The last three classes are in fact containers which have a list of objects of `Parameter`, `Objective`, and `Measurement` classes, respectively (Fig. 1). The configuration of a calibration task is specified in a JSON file, which contains information on which model to run, simulation case, calibration parameters, objectives (cost or error function), which optimizer to use and its configuration.
- Monte-Carlo simulations with scripting and Python's statistic libraries. This is useful for uncertainty analysis.

To demonstrate the automatic calibration functionality, Fig. 3 shows a simple, yet complete, example which is the water surface profile in a 10 km long river. This example is located in the "examples/calibration" directory of the repository on GitHub. The channel is divided into two segments, each of which has their own calibration parameter (the Manning's roughness n in this case). Fig. 3(a) shows the layout of the case (river flows from left to right) and the simulated profile using the simple `Backwater-1D` class. This result is used as a manufactured solution. Four sampling points are shown in Fig. 1(a), which are used to sample results as measurement data. Then the `Calibrator` is constructed and the calibration uses `scipy.optimize`'s "Nelder-Mead" method. Fig. 3 (b) shows the trajectory of the calibration in the parameter space which started from $(n_1, n_2) =$ (0.02, 0.02). Fig. 3(c) shows the calibration error as a function of iteration number. At the end, the calibration succeed with 26 iterations and the two calibration parameters n_1 and n_2 are very close to the true values of (0.04, 0.055).

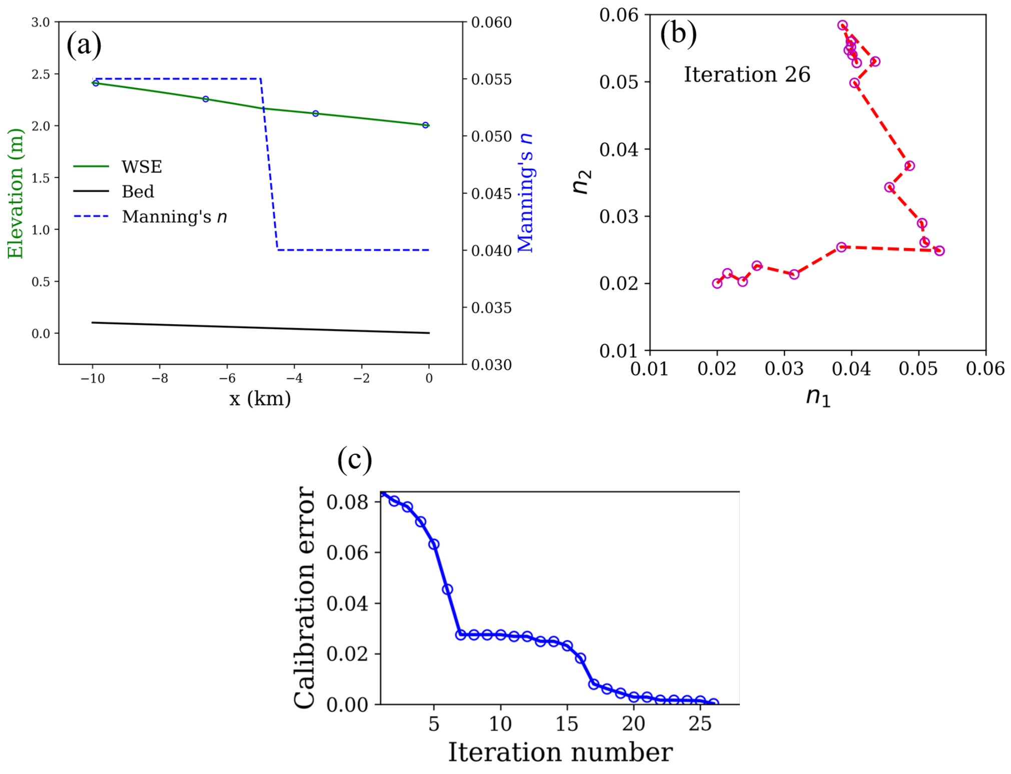

Figure 3: Automatic calibration using pyHMT2D.

In addition to the basic and advanced functionalities, other useful features are as follows:

- calculate the difference between simulation results (regardless they are on the same mesh or not). This can be done through the vtkHandler class in the Misc submodule because all model results can be converted into the unified VTK format.
- create and manipulate georeferenced terrain data for 2D modeling. This can be done through the Terrain class in the Misc submodule.
- conversion of SRH-2D mesh and result to 3D through vertical extrusion (one layer or multiple layers) and VTK interpolation. It can be done through the SRH_2D_Data class. This feature is useful if one wants to use 2D simulation result in 3D applications, e.g., fish passage design or use 2D result as initial condition for 3D simulations. Currently, conversion to OpenFOAM (OpenFOAM Foundation, 2021) is supported through Gmsh's MSH file format (Geuzaine & Remacle, 2009).

Dependencies

pyHMT2D's core functionality relies the following Python packages: h5py (Collette, 2013), VTK (Schroeder et al., 2006), GDAL (GDAL/OGR contributors, 2020), NumPy (Harris et al., 2020), and Matplotlib (Hunter, 2007). Optionally, pywin32 (Hammond & Robinson, 2000) is needed to control the simulation of HEC-RAS and affine (Gillies, 2014) is needed to read HEC-RAS results.

Acknowledgements

I thank Dr. Yong Lai at U.S. Bureaul of Reclamation and Scott Hogan at U.S. Federal Highway Administration for their valuable knowledge and suggestions regarding SRH-2D. I also thank

Gary Brunner from U.S. Army Corps of Engineers for his time answering my questions regarding HEC-RAS.

pyHMT2D was developed as a tool box for several past and ongoing research projects in the author's research group. Though pyHMT2D did not directly receive financial support from these projects nor it was in any of these projects' workscope, pyHMT2D did benefit from the knowledge on various aspects of 2D hydraulic models accumulated through these projects.

References

Ayachit, U. (2015). *The ParaView guide: A parallel visualization application*. Kitware. ISBN: 978-1-930934-30-6

Brunner, G. W. (2016). *HEC-RAS river analysis system user's manual, version 5.0* [Manual]. Hydrologic Engineering Center, Institute for Water Resources, U.S. Army Corps of Engineers.

Cigler, B. A. (2017). U.s. Floods: The necessity of mitigation. *State and Local Government Review*, *49*(2), 127–139. https://doi.org/10.1177/0160323X17731890

Collette, A. (2013). *Python and HDF5*. O'Reilly. ISBN: 978-1-449367-83-1

Dysarz, T. (2018). Application of python scripting techniques for control and automation of HEC-RAS simulations. *Water*, *10*(10), 1382. https://doi.org/10.3390/w10101382

GDAL/OGR contributors. (2020). *GDAL/OGR geospatial data abstraction software library* [Manual]. Open Source Geospatial Foundation.

Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, *79*(11), 1309–1331. https://doi.org/10.1002/nme.2579

Gillies, S. C. (2014). Affine. In *GitHub repository*. https://github.com/sgillies/affine; GitHub.

Goodell, C. R. (2014). *Breaking the HEC-RAS code: A user's guide to automating HEC-RAS* (1st ed.). h2ls.

Hammond, M., & Robinson, A. (2000). *Python programming on Win32*. O'Reilly. ISBN: 1-56592-621-8

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Lai, Y. G. (2010). Two-dimensional depth-averaged flow modeling with an unstructured hybrid mesh. *Journal of Hydraulic Engineering*, *136*(1), 12–23. https://doi.org/10.1061/(ASCE)HY.1943-7900.0000134

Leon, A. S., & Goodell, C. R. (2016). Controlling HEC-RAS using MATLAB. *Environmental Modelling & Software*, *84*, 339–348. https://doi.org/https://doi.org/10.1016/j.envsoft.2016.06.026

Liu, X., Landry, B. J., & Garcia, M. H. (2008). Coupled two-dimensional model for scour based on shallow water equations with unstructured mesh. *Coastal Engineering*, *50*, 800–810.

Moya Quiroga, V., Popescu, I., Solomatine, D. P., & Bociort, L. (2013). Cloud and cluster computing in uncertainty analysis of integrated flood models. *Journal of Hydroinformatics*, *15*(1), 55–70.

OpenFOAM Foundation. (2021). *OpenFOAM v8 User Guide*. https://cfd.direct/openfoam/user-guide

Schroeder, W. J., Lorensen, B., & Martin, K. (2006). *The visualization toolkit (4th ed.)*. Kitware. ISBN: 978-1-930934-19-1