Dual Solution Engines for a Water System Model Require a Flexible API

Timothy Hirrel, P.E.
August, 2023

Abstract

A water system hydraulic model can be viewed from many different perspectives. Often, model users might not concern themselves with how a model produces results, but rather with its ease of use and the legitimacy of the results. Ultimately, though, the legitimacy of the results is dependent on how the model works. The purpose of this paper is to discuss how a model works and how a new feature will allow model users to be more confident in the legitimacy of the results.

A New Approach

“A man with a watch knows what time it is, a man with two watches is never sure.” This well known comment on the trade off between certainty and correctness can be appropriately applied to water system hydraulic modeling.

A model provides an important perspective on a water system by simulating hydraulic performance with math. These mathematical simulations provide information about pressures and flows for an entire system and for situations that have not yet occurred or that occur with low frequency and high consequence. Along with the benefits of providing such information, a model comes with an overriding concern: how well will the model match reality. The process of satisfying that concern has commonly been referred to as model calibration[i].

The calibration process is typically focused on the accuracy of the input data, for good reason. But calibration should also be concerned about the appropriateness of the model math and logic. Since the heart of model, the part that solves the hydraulic equations, is typically contained in what is called the solution engine, it might be advantageous to have two independently developed solution engines readily available.

Many model users are familiar with the Epanet[ii] solution engine, either from using Epanet itself, or from using one of the many commercial models based on the Epanet solution engine. Another solution engine has been developed independently and shares no code with Epanet, taking very different approaches to both the programming and the solution techniques. Now, both the Epanet solution engine and this second solution engine, the TdhNet[iii] solution engine, can be used from within one application, using the same input data and user interface. This will allow results from the two solution engines to be easily compared and to identify situations where one solution engine may be preferable over the other. The application, TdhNet, runs under both MS Windows and Linux, which allows the use of a wide range of computer capabilities.

Dual Solution Engines for a Water System Model Require a Flexible API

How Does it Work?

The key to using different solution engines within one application is the Application Programming Interface, or API, for each engine. As with the overall programming, the approach taken by each solution engine API is quite different from the other.

Using lego blocks as an analogy to coding, an individual code function is like an individual lego piece. A code class is like a collection of lego pieces put together to form a specific feature. An application can be formed from individual functions, but the use of classes allows the application to be formed by building and arranging entire features.

For Epanet, the API consists of a number of independent procedures, each performing specific functions, such as providing input data. For TdhNet, the API consists of a number of classes defining a collection of related functions that are mostly abstract, i.e. they don't actually do anything except define what is supposed to be done. Thus, an abstract class can be implemented by any number of implementation classes, which may or may not build from other implementation classes. The programming that uses the abstract classes need not be changed for different implementations.

For example, an abstract class that allows for control of the solution process might define functions to:

- prepare for a solution sequence
- implement any data changes before a solution
- perform a single solution
- perform post solution processing

Post solution processing might include the programming necessary to implement an Extended Period Simulation (EPS) or a Water Quality Analysis. An abstract class to implement an EPS might define functions to:

- Determine the time step to the next solution based on tank level limits.
- Determine a new level for each tank based on the flow and time step.
- Perform energy related calculations based on pump performance and the time step.

These abstract classes can be mixed and matched such that a particular EPS implementation can be used with different solution engines. In general, the more granularity and independence within the classes, the better. This means that a class should provide specific functionality and not interfere with any other functionality. For example, the TdhNet API includes a class that handles pressure dependent demands. It gets the data it needs from the classes that provide access to solution engine results and informs the input data classes of any modified demands. Therefore, the functionality of this class can be applied with either solution engine and be changed without affecting any other class.

If all this sounds abstract with little relevance to the end user, consider the practical implication at the heart of this article: two very different solution engines developed for two different models becoming accessible from one program. That portends the type of options that can be provided by implementing different classes for energy calculations or water quality analysis, as examples. With well developed

Dual Solution Engines for a Water System Model Require a Flexible API

API's, new implementations for specific functionality can be developed without having to modify other functionality. Indeed, they can be applied to entirely different solution engines without modification to the solution engines.

To use the Epanet solution engine within the TdhNet application, implementations of the abstract classes within the TdhNet API were developed (within a new library) to call the Epanet solution engine functions. The Epanet solution engine was not modified and the TdhNet application code does not know or care which solution engine is being used, other than allowing user selection of the solution engine. All the functionality within TdhNet, including Change Situations, Fire Flow Analysis and Pressure Dependent Demands, is available when using the Epanet solution engine.

A collateral consequence of this approach is that the TdhNet API now provides an alternative method of using the Epanet solution engine for third party applications. This alternative API has the following attributes:

- Finer control over the solution process.
- Can be extended through object-oriented classes.
- Data management tools including database access for both input data and solution results.

A parallel effort was undertaken to make the TdhNet solution engine accessible through the Epanet API. An abstract class encapsulates the Epanet API functions. One implementation of that class calls the Epanet functions and another implementation calls the TdhNet functions. Switching between the solution engines is simply a matter of specifying which implementation class is to be used. The TdhNet implementation may be useful for applications currently using the Epanet API and wishing to explore the use of a different solution engine.

Differences Between the Solution Engines

There are some important difference between the two solution engines, including:

- TdhNet identifies the paths of known energy loss and the relationships between those paths prior to each solution. Identifying and using the relationships between paths is what differentiates the TdhNet solution method from the Hardy Cross method[iv]. The processing required for identifying paths is offset by the efficiency of solving the equations resulting from the known path relationships. A secondary benefit is that closed pipes are not included in the path analysis and therefore have no effect on the solution.

- In Epanet, a matrix describing the connectivity between nodes is established at the beginning of a solution sequence and does not change during the sequence[v]. Therefore, closed pipes are not actually closed, they are assigned a large energy gradient which can be changed if the pipe status changes. The disadvantage is that a supposedly closed pipe that should isolate a demand allows flow to satisfy the demand, with very high energy losses.

- The TdhNet solution engine does not establish data structures for input data, it access input data through abstract classes. The Epanet solution engine establishes data structures for input data

into which the input data must be copied. Therefore, the Epanet solution engine will use more memory for a given network.

- TdhNet uses linked lists while Epanet uses mostly arrays. The trade off is that linked lists are more scalable while arrays provide more efficient random access. Unlike an editor, a solution engine doesn't require much random access and TdhNet uses indexed linked lists to greatly improve random access in large lists.

- In TdhNet, pumps and valves are not separate link types but attributes of pipes. The code for defining the relationship between energy change vs. flow for pipes is external to and called by the solution engine. Therefore, the relationship can be modified, by creating new pump or valve functionality, for example, without modifying, and possibly corrupting, the solution engine itself.
- In Epanet, junction nodes and fixed grade nodes are maintained in a single list, so a fixed grade node may not have the same identifier as a junction node. In TdhNet, fixed grade nodes and junction nodes are maintained in separate lists, so a fixed grade node may have the same identifier as a junction node. When using the Epanet solution engine, the TdhNet application ensures fixed grade node identifiers are unique among all nodes by adding a suffix. This process is transparent to the user.

A comprehensive study of the differences in results from the two solution engines is beyond the scope of this paper. Results will definitely differ when there is a demand isolated by a closed pipe, as noted above. Results might also differ significantly when there is complex interaction between valves and/or rules. It also should be noted that results from using the Epanet engine within TdhNet will not necessarily be the same as results from Epanet itself, because:

- The TdhNet EPS functions are used with the Epanet solution engine and there are differences from the Epanet EPS functions:
  - When a tank reaches full, empty or some rule limit before a scheduled solution, TdhNet will perform a solution at that time.
  - TdhNet uses a trapezoidal function to calculate tank volume between specified volume points.
- The TdhNet Rules functions are used with the Epanet solution engine.
- The TdhNet Pressure Adjusted Demand functions are used with the Epanet solution engine.
- The TdhNet Automatic Pump Speed Controls are used with the Epanet solution engine.

Conclusion

The ability to use two different solution engines within a single water system model will allow the user to compare results and identify situations where one solution engine may be preferable over the other. The ability to use two different solution engines is facilitated by Application Programming Interfaces, or API's. A well developed API will allow specific functionality within a model to be modified, replaced or used within other applications without having to modify and potentially disrupt other functionality. The result will be a better experience for both model users and model developers.

Dual Solution Engines for a Water System Model Require a Flexible API

About the Author

Tim Hirrel is a professional engineer with significant experience in the use and development of water system hydraulic models. He has been working on the development of TdhNet and related software for over 30 years. His previously published papers include “Systems Curves and Pump Selection” and the award wining “How Not to Calibrate a Hydraulic Network Model”.

i "How Not to Calibrate a Hydraulic Network Model", T. Hirrel, Aug. 2008 Journal AWWA.

ii The Epanet application and API may be downloaded from the website www.epa.gov/water-research/epanet and used under the licensing term provided within the website.

iii The TdhNet application and API may be downloaded from the website www.tdhnet.com and used under the licensing term provided within the website.

iv "Lossless Network Skeletonization", T. Hirrel, Sept. 2010, Journal AWWA,

v "Analysis Algorithms", User's Manual, Epanet version 2.2