Development and Delivery of Server-Side Water Resources Models Using Web-Based Interfaces

by

Caroline M. Neale, BSCE, MSCE, EIT

A Dissertation

In

Civil Engineering

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

DOCTOR OF PHILOSOPHY

May, 2018

ACKNOWLEDGEMENTS

Financial support for this research was provided by the National Center for Infrastructure Modeling and Management [1], the Texas Department of Transportation[2], the Texas Section of the American Society of Civil Engineers (Hawley Fellowship) [3], and the Department of Civil, Environmental, and Construction Engineering[4].

I would like to thank Dr. Theodore G. Cleveland, Dr. Kenneth A. Rainwater, and Dr. Annette Hernandez-Uddameri for all of their help and guidance both with my dissertation and in other aspects of my education. Y'all have been such amazing role models for me throughout my years at Texas Tech. I can't even begin to explain the impact all three of you have had on my life. I definitely couldn't have done it without you guys! All of you have been an inspiration to me. Thank you so much.

I further acknowledge the help of my many classmates over the last few years, especially Watney R. Thanks for being by my side in every single class for the past year and a half. Justine, thanks for being with me throughout this whole process. You are amazing.

---

[1] National Center for Sustainable Water Infrastructure Modeling Research, TTU Award Number: A16-0313-001

[2] Evaluating Use of Sub-Grade Drains with PFC for Stormwater Drainage, TTU Award Number: A15-0108-001, A15-0108-002, A15-0108-003

[3] Texas Section Scholarships and Fellowships for Graduate Students (2015 Hawley Fellowship)http://www.texasce.org/?page=Scholarships

[4] Teaching Assistant 2014-2015

TABLE OF CONTENTS

ABSTRACT

This research is a collection of three projects that utilize various coding concepts and framework required to implement web-based server-side processing of common water resources modeling scenarios. The three publications utilize combinations of HTML/CSS, Python, R, and PHP scripts with each chapter expanding upon the work of the previous. The first, SolidsInRivers develops a database-search engine to make estimates of Solids Flux and Solids Concentrations in river systems. NewANTS is a collection of analytical and semi- analytical solutions to various forms of contaminant transport in different geometries. Finally, EPANETOL describes methods to employ professionally successful hydrologic modeling tools into the web environment. It is run through a web-browser and the section also describes how to build a functioning web-server-accessed implementation of EPANETOL. These three papers form a logical sequence and represent the first steps toward the eventual goal of a fully browser-enabled EPANET interface thereby achieving true operating system and potentially architectural independence.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION & DISSERTATION STRUCTURE

This dissertation is a collection of three themed publications related to water resources models and the delivery of those models using a web-based (browser) interface. Each publication element explores and documents a different type of hydrologic modeling tool and the methodology behind the development of each web-based interface.

SolidsInRivers

Erosion and the flow of water in rivers and streams continually remove bed load material from the bed and banks of the channel and displace them further downstream. Estimations of this solids load transport is important in river engineering, fluvial geomorphology, ecohydrology, and hazard estimation. Many solids transport (bed load) equations relate metrics such as a characteristic dimension (grain diameter), hydraulic characteristics, and solids density to solids flux. These equations, however, exhibit inadequate estimation performance for a variety of reasons (Recking, 2010). The equations themselves are the result of

regression analysis on variables thought to explain solids flux behavior and are derived from data collected in flume studies and some river studies.

SolidsInRivers is a tool used to generate estimates of sediment transport in river systems. It uses HTML, Python, and R in a hybrid manner to presents a client with a web-based interface that the client populates with input values and then requests a result.

A database-search approach was designed that searches a literature-derived database with a set of predictor or input values expected to be available to an analyst. The database utilizes a distance-weighting database query with Z-score mapping of the variables and an L-2 distance in the multivariate Z-score space to generate an estimate of solids flux from a selected set of end-user supplied explanatory variables. The entire process is coded into HTML, Python, and R so that the estimate can be served from a web server, making the tool available to anyone using a modern browser. The search algorithm was tested by using every entry in the database that could be used to generate estimates using four existing models and the SolidsInRivers search model. After this examination, the search-engine was then adapted to run on a web-server, with a web-based interface and server-side search.

NewANTS

Groundwater contamination occurs when pollutants are released into an aquifer. While groundwater contamination does occur from natural sources, most often it is a result of human activity. Groundwater can become contaminated due to land surface activities, sub-surface releases/spills, and polluted recharge water. Some common contamination causes include septic systems, improper disposal of hazardous waste, releases/spills from stored chemicals (both above and below ground), leachate from landfills, sewers and other pipelines, pesticide and fertilizer use, contaminated water wells, and mining activities to name but a few. Contamination of groundwater can result in poor drinking water quality, loss of water supply, degraded surface water systems, high costs for remediation, and potential health problems. Groundwater typically moves slowly, therefore, contamination can remain undetected for long periods of time. Treatment of contaminated groundwater can be achieved by containing the contaminant to prevent migration, pumping the water and treating it before returning it to the aquifer, allowing the contaminant to attenuate naturally, or some combination of the three. Remediation technique selection is dependent on site specific factors such as local hydrogeological conditions, soil characteristics, contaminant

type, and cleanup goals. The pollutant creates a contaminated plume within the aquifer. The contamination spreads throughout the affected aquifer due to the movement of water and dispersive effects. The movement of a contamination plume is analyzed using hydrologic transport models. Some factors that influence contaminant transport include advection, dispersion, diffusion, adsorption, decay, and precipitation (US Congress, 2015).

NewANTS is used to evaluate analytical solutions to soluble contaminant transport scenarios in groundwater and was designed using a hybrid of HTML, Python, and R. A collection of solutions was built using R, then modified so they can be assessed using a modern browser. As an extension of SolidsInRivers, NewANTS documents the added HTML, Python, and web server configuration requirements to produce both tabular and graphical output and serve them from a web server. These detailed output files were designed to provide the user with more elaborate output results from the simulation.

NewANTS is a restoration of an old system that was designed to approximate analytical contaminant transport in groundwater. It is a collection of analytical and semi-analytical solutions to various forms of the advection-dispersion equation in different geometries.

NewANTS models transport scenarios with regional groundwater flow of soluble contaminants, at this time it does not incorporate pumping rates or insoluble constituents. It utilizes a hybrid of Python and R to take user-supplied inputs and export a contaminant profile and a table of the results.

The historical version of the **A**nalytical **N**umerical **T**ransport (modeling) **S**ystem (ANTS) was developed in 1998 (Chuang). NewANTS is designed to replicate and extend the original work. NewANTS utilizes graphical output directly through the HTML, CSS, and JavaScript standards which was a limitation of the original ANTS model.

There are 48 transport model scenarios available in NewANTS. Three different dimensional models are available: 1-Dimensional, 2-Dimensional, and 3-Dimensional. Four different source types are represented for each of the dimensional parameters, including impulse, injection, step-function, and finite length injection. For each source type, there are four different attenuation/transport types: Advection-Dispersion, Advection-Dispersion-Retardation, Advection-Dispersion-Decay, and Advection-Dispersion-Retardation-Decay.

The NewANTS interface will be available on the internet as a tool for students and/or researchers who are in need of a quick analytical solution to contaminant transport simulations. The benefit of having the tool be largely web-based is twofold. The first advantage is that a web page and corresponding server may be accessed remotely, allowing the user to use the system from anywhere, at any time. This feature of a web-based tool makes it much easier for the casual user to take advantage of this free analytical model. Secondly, computation engine updates can be done much more quickly and easily with a web-based approach. The current system relies only on one single server for the database; therefore, any updates would only have to be done on the server. The interface can be further developed and refined over time without requiring a complete redesign or requiring updates on each user's individual machine.

Once the contaminant transport analysis is complete, the results are returned to the user and displayed via the web interface. The solution is described in three different ways: a concentration profile plot, a table of the calculated concentration at equal intervals along the model distance, and a summary table of the final solution.

EPANET On-Line

The Environmental Protection Agency Network (EPANET) is a computer modeling program developed and maintained by the United States Environmental Protection Agency. EPANET performs extended period simulation of hydraulic and water quality behavior within pressurized pipe networks. A network, in this context, refers to a model made up of pipes, nodes or pipe junctions, pumps, valves, and storage tanks or reservoirs. EPANET models the flow of water in each pipe, the pressure at each node, the water level in each tank, and the concentration of chemical species throughout the network. EPANET is a research tool designed to model the movement and fate of drinking water constituents within water distribution systems. Some example applications within distribution systems analysis include sampling program design, hydraulic model calibration, Chlorine residual analysis, and consumer exposure assessment (EPANET2 User Manual, 2000). EPANET software provides an integrated environment for creating and editing input network data, running hydraulic and water quality simulations, and viewing the results of simulations. Results are reported in a variety of ways including color-coded network maps, data tables, time series graphs, and contour plots.

EPANET simulations are divided into two major categories, hydraulic modeling and water quality modeling. EPANET's hydraulic analysis engine include such features as simulating systems of any size, calculating friction head loss, computing minor head loss due to bends/fittings, designating constant or variable speed pumps, calculating pumping energy and cost, modeling using various valve types including shutoff, check, pressure regulating, and control valves, modeling of storage tanks of any shape, modeling pressure-dependent flow from emitters, and systems operating based on controls ranging from simple commands to complex rule-based controls. The water quality modeling capabilities of EPANET include modeling the movement of a non-reactive tracer material through the network over time, modeling the movement and fate of a reactive material as it grows or decays over time, modeling the age of water throughout a network, tracking the percent of flow from a designated noted reaching all other nodes over time, modeling reactions both in the bulk flow and at the pipe wall, modeling reactions in the bulk flow using n-th order kinetics, modeling reactions at the pipe wall using zero or first order kinetics, accounting for mass transfer limitations when modeling pipe wall reactions, allowing growth or decay reactions to proceed to a limiting concentration, employing global reaction rate coefficients that can be modified on a pipe-by-pipe

basis, allowing wall reaction rate coefficients to be correlated to pipe roughness, allowing for time-varying concentration or mass inputs at any network location, and modeling of storage tanks as being either complete mix, plug flow, or two-compartment reactors. (EPANET2 User Manual, 2000).

The first two chapters document the development and deployment of a single tool, SolidsInRivers, and a set of related tools, NewANTS, into a web environment. To extend these concepts further, EPANET On-Line describes a method to employ professionally successful hydrologic modeling software in a web environment using a browser-based interface.

EPANET On-Line, hereafter referred to as "EPANETOL", adds several additional components to the concepts explored during development of SolidsInRivers and NewANTS. The first is that the input data required is more extensive than the other two tools – the client needs to generate an input file, and the file itself needs to be uploaded. The second difference is remote storage of input, binary, and output files. Like SolidsInRivers and NewANTS, the program must be able to write files to the server, but with EPANETOL, these files need to be retained for a while for the client to retrieve them.

The main goal of the EPANETOL research focused on developing a server-side computational program in preparation for three anticipated advances. The first is EPANET evolving into a version that is available to users without requiring the download of EPANET software. Secondly, reducing the requirement for physical hardware (Graphical Processing Units) on the user's machine and increasing reliance on parallel processing on a remote server. And finally, deploying EPANET versions that are fully browser-enabled.

In anticipation of these future advances, this research aims to illustrate the maintenance advantage of a server-side copy of the software.

This project is comprised of two main components. Part 1 consisted of building a server-side implementation of the computation engine and the requisite inter-process communications necessary for a client to be able to upload an EPANET input file to the computation server, instruct the server to run the program, and retrieve the computational output for further processing.

Part 2 focused on creating a separate web-based tool that could build an input file on the client's machine without requiring the use of EPANET Graphical User Interface (GUI) software.

CHAPTER 2

SolidsInRivers

Introduction

Equations used to compute bed load transport in rivers and streams are based upon regression analysis of data collected for variables related to solids load calculations. However, these methods have been found to be difficult to use, too complex for practical use, and/or inadequate in terms of estimation precision. This study developed and deployed a database-search engine used to find approximate solutions of solids load transport based on model-specific input data. SolidsInRivers was created using R but is accessible via a web interface in an attempt to simplify and streamline sediment transport estimation. The search processing was all server-side, operating on a single, central database.

SolidsInRivers presents a client with a web-based interface that the client populates with input values and then requests a result. SolidsInRivers uses HTML, Python, and R in a hybrid manner to generate estimates of sediment transport in river systems. The interface generates a small input file on the server. This file is

ingested by the simple database program, which is written in R and run on the server using a system command generated by the web interface, and a result file and graphic are generated. These two files are then assembled into the output interface presented to the client.

The search algorithm was tested by using four different datasets, the Meyer-Peter and Mueller model, the Wong and Parker model, the Recking model, and the Schoklitsch model.

Several examples are presented to illustrate the use of the tool a potential application is demonstrated. The value of the search approach is simplicity of use because of the interface. Additionally, when new experimental studies become available, they can be employed immediately as the database is simply extended without any change to the underlying search engine. Because the search engine relies upon a database, that database can be made available for other researchers wishing to explore variable interactions and improve the various regression models in use.

Literature Review

Peterson (1975) presented a series of design charts that related certain hydraulic properties that are accessible to practicing

engineers, either by direct measurement or reasonable desktop approximation. These charts can be used to make estimates of mobile solids flux in a stream from predictor variables such as channel slope, clear-water discharge, and solids dimension. The charts in Peterson (1975) are visual representations of regression models of an underlying database.

An alternative to charts is to employ some form a transport model that estimates solids flux based on a set of parameters, such as the following set of models listed below. As with the Peterson charts, certain parameters in the equations are the result of a regression model applied to an underlying database. A common form of transport model is to express the transport rate as a function of the bed stress and some critical stress needed to produce motion as in Equation 1.

$$q_b^* = \alpha(\tau^* - \tau_{cr}^*)^\beta$$

$$\text{if } \tau^* > \tau_{cr}^*, \text{ otherwise } 0$$

Equation 1

where:

q_b^* = Dimensionless bed load transport rate

α, β = Constant coefficients related to chosen modelling method

τ^* = Dimensionless bed shear stress

τ_{cr}^* = Critical dimensionless shear stress

The dimensionless bed shear stress, τ^*, is computed using Equation 2.

$$\tau^* = \frac{\rho g R_h S}{(\rho_s - \rho) g d}$$

Equation 2

where:

ρ = Fluid density (kg/m^3)

g = Gravitational constant = 9.81 m/s^2

ρ_s = Sediment density (kg/m^3)

S = Channel bed slope (m/m)

R_h = Hydraulic radius (m)

d = Characteristic grain diameter (e.g. d_{50}, the median grain diameter) (mm)

The critical dimensionless shear stress, τ_{cr}^{*} , is computed using a correlation developed by Brownlie (1981) and later modified by Neil (1986) to form Equation 3.

$$\tau_{cr}^{*} = \lambda[0.22Re_{p*}^{-0.6} + 0.06\exp\left(-17.77\ Re_{p*}^{-0.6}\right)]$$

Equation 3

where:

λ = Constant coefficient

Re_{p*}= Particle Reynolds number

In Equation 3, lambda, λ, typically ranges from ½ to 1. These values can be interpreted as lower and upper bounds for natural

sediments. The Meyer-Peter and Muller (1948) equation for plane beds is given in the form of Equation 1 with:

α=8

β = 3/2

τ_{cr}^{*}=0.047

Wang and Parker (2006) modified form of Equation 1 using values of:

α=3.97

β = 3/2

τ_{cr}^{*}=0.0495

Equation 1 is made dimensional using Equation 4:

$$q_{bm} = \rho_s \sqrt{\left(\frac{\rho_s - \rho}{\rho}\right) g d^3}$$

Equation 4

where:

q_{bm} = Bed load transport rate (kg/s•m)

ρ = Fluid density (kg/m^3)

g = Gravitational constant = 9.81 m/s^2

ρ_s = Sediment density (kg/m^3)

d = Characteristic grain diameter (e.g. d_{50}, the median grain diameter) (mm)

Equation 5 is the (dimensional) excess unit discharge model of Schoklitsch (1949).

$$q_{bm} = 2500 S^{\frac{3}{2}} (q - q_c)$$

Equation 5

where:

q_{bm} = Bed load transport rate (kg/s•m)

q = Unit discharge rate (m^3/s•m)

q_c = Critical unit-wide discharge rate for non-uniform sediment (m^3/s•m)

Recking (2010) presents a model similar in form to Equation 4 but with an added factor

$$q_{bm} = \rho_s \sqrt{\left(\frac{\rho_s - \rho}{\rho}\right) g d_{84}^3 \Phi}$$

Equation 6

where:

q_{bm} = Bed load transport rate (kg/s•m)

ρ = Fluid density (kg/m^3)

g = Gravitational constant = 9.81 m/s^2

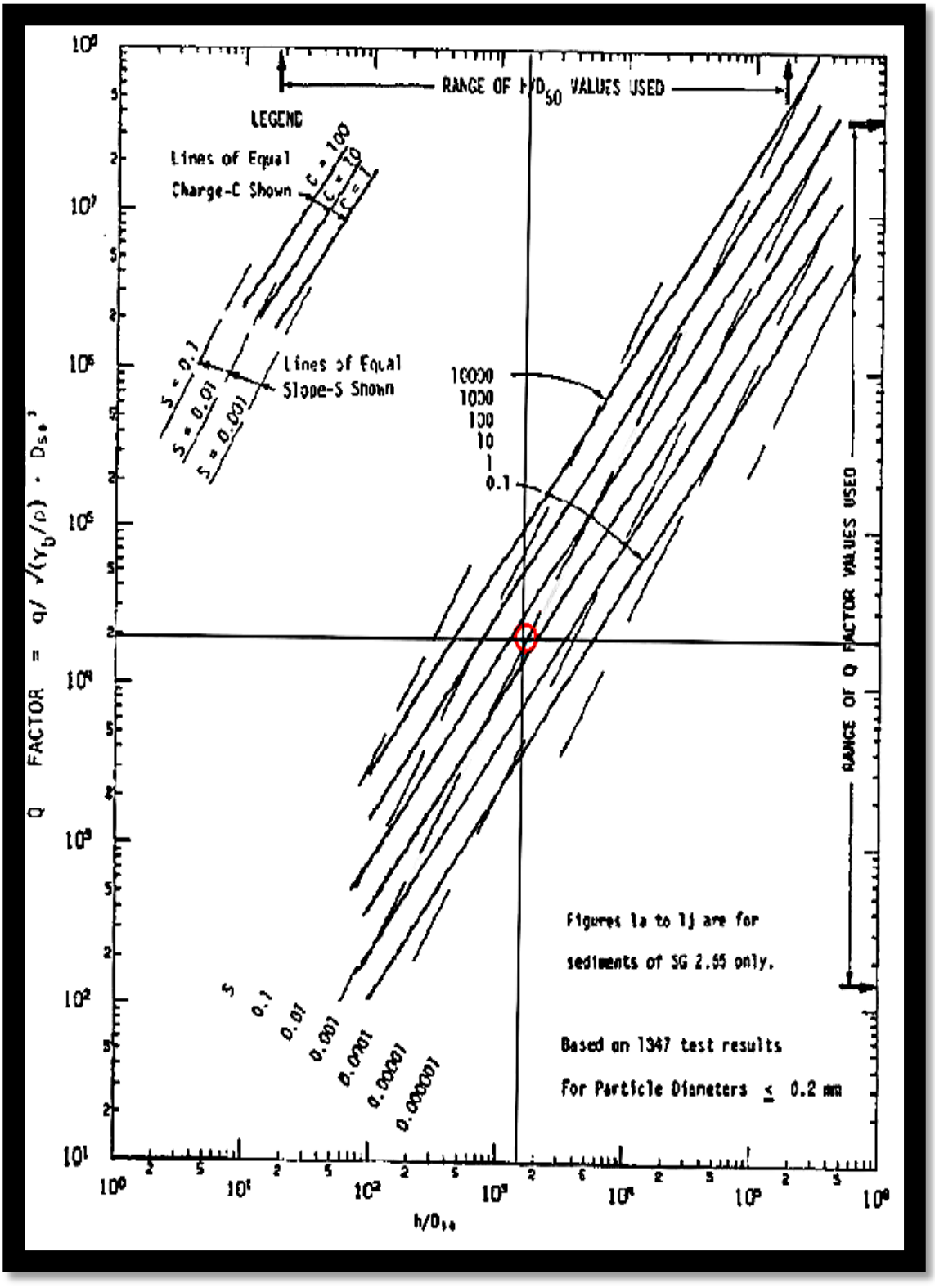
ρ_s = Sediment density (kg/m^3)

d_{84} = Characteristic grain diameter (84% smaller than) (millimeters)

φ = Einstein dimensionless parameter

In general, a minimal set of parameters needed to employ these equations are (in no particular order): Solids density, bed slope, d_{16}, d_{50}, d_{84}, hydraulic radius, mean section velocity, and discharge per unit width of stream.

Peterson's charts require a smaller set of parameters: discharge, channel width, hydraulic depth, d_{50}, solids density, and energy slope (bed slope is a suitable surrogate). To illustrate the use of the Peterson chart, consider 1 which is one of the dozen charts from the original document. The chart represents the relations between clear water discharge per unit width of channel, *q*, the depth of flow, *h*, a characteristic solids diameter, d_{50}, a solids concentration in parts per hundred thousand, C_{ppht}, and energy slope, *S*.



1.1. Peterson Chart for Mobile Solids Flux Estimate

Suppose a particular channel has a discharge of about 20 ft^3/s with a width and depth of about *8* and *0.90* feet, respectively. In SI units (the charts require SI entry), these predictor variables are: $Q=0.566\ m^3/s$, $W=2.439\ m$, and $h=0.274\ m$. Further suppose the channel material has a $d_{50} = 0.2\ mm$, that the solids specific gravity is *2.65* (quartz), and that energy slope is $S=0.001$ (0.1 percent). A design question is ``What is the anticipated solids flux in the system?"

To use the chart, the analyst performs the following computations:

1. Compute the discharge per unit width of the channel (the unit discharge is also used in the Schoklitsch (1949) equation.

$$q = \frac{0.566}{2.439} = 0.232\ \frac{m^3}{s \cdot m}$$

2. Compute the "size factor" from

$$\text{Size Factor} = \left(\frac{\rho_s}{\rho_w} - 1\right) g(1000 D_{50})^3$$

$$= \left(\frac{2.65}{1} - 1\right) 9.8(1000(0.2))^3 = 1.29 \times 10^{-10} \frac{m^6}{(s \cdot m)^2}$$

3. Compute the "Q-factor", which is the vertical axis of the chart from the ratio of discharge per unit width and the square root of the size factor,

$$\text{Q-Factor} = \frac{q}{\sqrt{\text{Size Factor}}} = \frac{0.232}{\sqrt{1.29 \times 10^{-10}}} = 20{,}400$$

This value is shown on 1 as the horizontal line that intersects the vertical axis at 20,000.

4. Compute the depth to diameter ratio

$$\frac{h}{1000 D_{50}} = \frac{0.274}{1000(0.2)} = 1371$$

This value is shown on 1.1 as the vertical line that intersects the horizontal axis at about 1,300.

5. The intersection of these two lines returns estimates of the energy slope and the solids load. The location of these estimates is indicated by the circle on 1.1 In this case, the slope is close to the observed slope of 0.001, and the solids load is somewhere between 10 and 100 ppht. For the sake of simplicity, a value of 50 is assumed.

6. Once the solids load is estimated the solids mass discharge is simply the product of the clear water discharge, clear water mass density, and the solids load, as

$$q_{bm} = \frac{C_s}{100{,}000} \rho_w q = \frac{50}{100{,}000} (1000)(0.232) = 0.11 \frac{kg}{s \cdot m}$$

This value when multiplied by the channel width estimates the solids mass flux in the system. In this case about 0.3 kg/s, or 0.6 lb/s.

As a comparison, consider the same information, but instead we choose the Schoklitsch (1949) equation, repeated as Equation 7.

$$q_{bm} = 2500 S^{\frac{3}{2}} (q - q_c)$$

Equation 7

The equation appears simple, except for the q_c value which is estimated from Equation 8 (Bathurst et. al., 1987)

$$q_c = 0.21 S^{-1.12} \sqrt{g d_{16}^3}$$

Equation 8

where d_{16} is dimensional. The first problem is that this value is not known so it must either be assumed or applied to the equation using the median grain diameter which may or may not be appropriate using this method. Using the d_{50} from above and the

channel width of *2.439 m* the estimated transport rate is *0.044 kg/s*, about an order of magnitude smaller than the Peterson result.

Performing the same analysis using the Recking (2010) model again requires estimation of inapplicable data parameters but results in an estimate of *0.21 kg/s*, which is comparable to the Peterson calculation. **Error! Reference source not found.** lists the results for the same input data (with some assumptions to populate missing information) using each of the various models. Most of the results are within a factor of two (except for Schoklitsch), hence on a typical log-log plot (like Peterson's) less than 1/3 log cycle variation.

Table 1: Solids Transport Estimate by Method

Method	Transport Estimate (kg/s)
Peterson (1975)	~ 0.3
Schoklitsch (1949)	0.044
Recking (2010)	0.21
Meyer-Peter & Mueller (1948)	0.337
Wong and Parker (2006)	0.168

Given that all the equations and the Peterson charts are derived from regressions on a database, an alternative approach considered

is to directly access the database and estimate values using a nearest-neighbor algorithm, which eliminates the need to interpolate (to use the charts) and potentially can be accessed using missing values. The remainder of SolidsInRivers discusses the development of a database search tool to perform the same estimation as the set of equations and the Peterson charts, and the deployment of that tool into a web-based tool that can be accessed from anywhere.

Search Method Algorithm

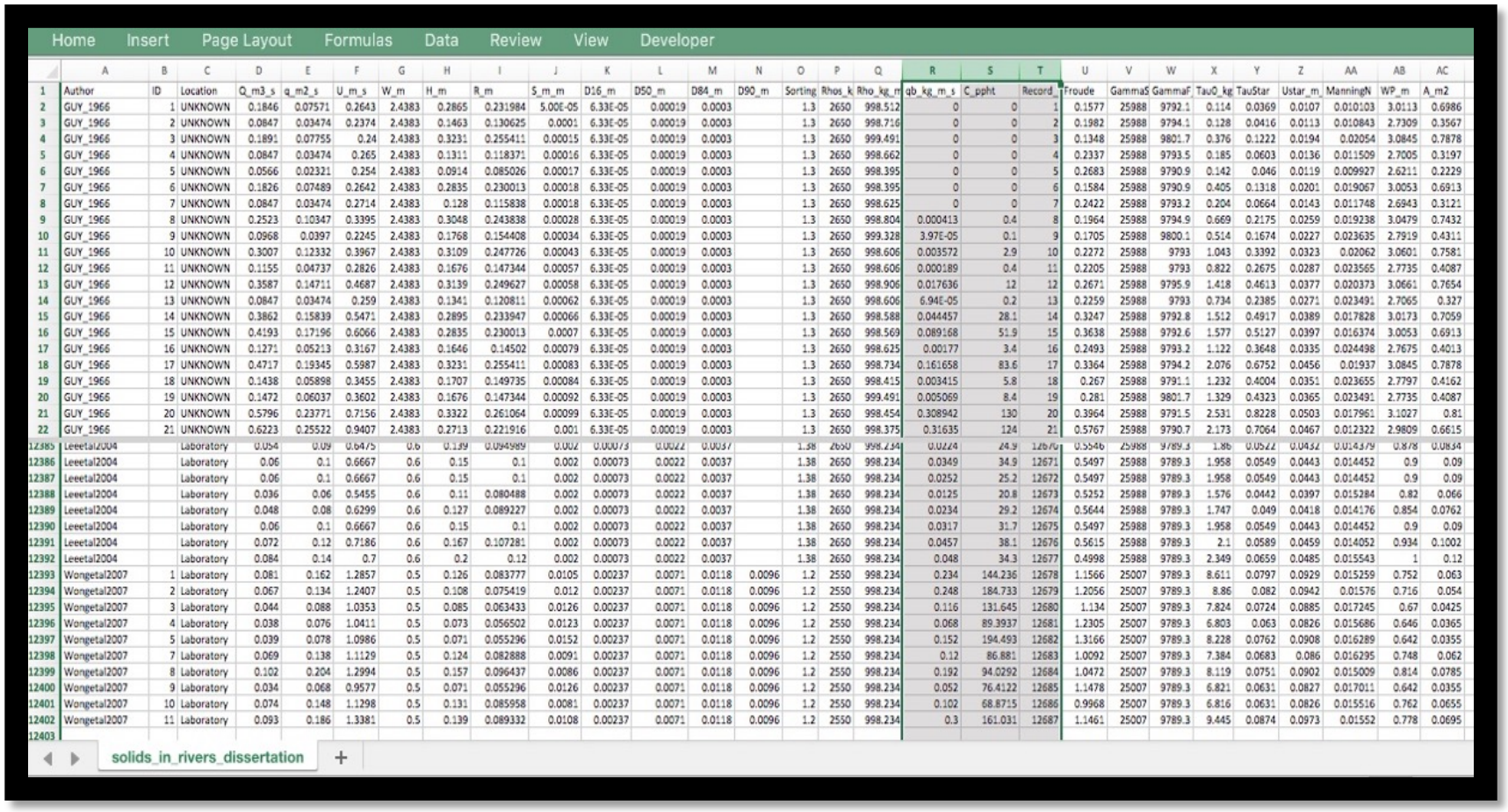
A database-search approach was designed that would search a literature-derived database with a set of predictor or input values expected to be available to an analyst. The initial values selected were channel slope of a stream crossing, a median particle diameter, and anticipated water discharge or mean velocity.

The 12,000+ record database used in this paper was developed by Cleveland et al. (2013). The database includes all of the original records in the Petersen and Howells (1973) database, unique records from Recking et al. (2008b), with additional unique records from King et al. (2004) and Recking (2010). Additional records include results from Wilcock et al. (2001), Lee at al. (2004), and Wong et al. (2007). The over 200 additional

experiments performed by Barnes (2013) are excluded in this paper to illustrate the utility of the database search tool.[5]

Figure 2 is a screen capture of the database to illustrate the underlying database. The database is stored as a single (flat) ASCII file – at its present size (12,000+ records) searching within Excel or R is feasible. Each record represents an observation and ancillary geometric and hydraulic information from a bedload study.

[5] These new records were added to the database currently deployed and in use on the web server.



Home Insert Page Layout Formulas Data Review View Developer

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	Author	ID	Location	Q_m3_s	q_m2_s	U_m_s	W_m	H_m	R_m	S_m_m	D16_m	D50_m	D84_m	D90_m	Sorting	Rhos_k	Rho_kg_m	qb_kg_m_s	C_ppht	Record_	Froude	GammaS	GammaF	Tau0_kg	TauStar	Ustar_m	ManningN	WP_m	A_m2
2	GUY_1966	1	UNKNOWN	0.1846	0.07571	0.2643	2.4383	0.2865	0.231984	5.00E-05	6.33E-05	0.00019	0.0003		1.3	2650	998.512	0	0	1	0.1577	25988	9792.1	0.114	0.0369	0.0107	0.010103	3.0113	0.6986
3	GUY_1966	2	UNKNOWN	0.0847	0.03474	0.2374	2.4383	0.1463	0.130625	0.0001	6.33E-05	0.00019	0.0003		1.3	2650	998.716	0	0	2	0.1982	25988	9794.1	0.128	0.0416	0.0113	0.010843	2.7309	0.3567
4	GUY_1966	3	UNKNOWN	0.1891	0.07755	0.24	2.4383	0.3231	0.255411	0.00015	6.33E-05	0.00019	0.0003		1.3	2650	999.491	0	0	3	0.1348	25988	9801.7	0.376	0.1222	0.0194	0.02054	3.0845	0.7878
5	GUY_1966	4	UNKNOWN	0.0847	0.03474	0.265	2.4383	0.1311	0.118371	0.00016	6.33E-05	0.00019	0.0003		1.3	2650	998.662	0	0	4	0.2337	25988	9793.5	0.185	0.0603	0.0136	0.011509	2.7005	0.3197
6	GUY_1966	5	UNKNOWN	0.0566	0.02321	0.254	2.4383	0.0914	0.085026	0.00017	6.33E-05	0.00019	0.0003		1.3	2650	998.395	0	0	5	0.2683	25988	9790.9	0.142	0.046	0.0119	0.009927	2.6211	0.2229
7	GUY_1966	6	UNKNOWN	0.1826	0.07489	0.2642	2.4383	0.2835	0.230013	0.00018	6.33E-05	0.00019	0.0003		1.3	2650	998.395	0	0	6	0.1584	25988	9790.9	0.405	0.1318	0.0201	0.019067	3.0053	0.6913
8	GUY_1966	7	UNKNOWN	0.0847	0.03474	0.2714	2.4383	0.128	0.115838	0.00018	6.33E-05	0.00019	0.0003		1.3	2650	998.625	0	0	7	0.2422	25988	9793.2	0.204	0.0664	0.0143	0.011748	2.6943	0.3121
9	GUY_1966	8	UNKNOWN	0.2523	0.10347	0.3395	2.4383	0.3048	0.243838	0.00028	6.33E-05	0.00019	0.0003		1.3	2650	998.804	0.000413	0.4	8	0.1964	25988	9794.9	0.669	0.2175	0.0259	0.019238	3.0479	0.7432
10	GUY_1966	9	UNKNOWN	0.0968	0.0397	0.2245	2.4383	0.1768	0.154408	0.00034	6.33E-05	0.00019	0.0003		1.3	2650	999.328	3.97E-05	0.1	9	0.1705	25988	9800.1	0.514	0.1674	0.0227	0.023635	2.7919	0.4311
11	GUY_1966	10	UNKNOWN	0.3007	0.12332	0.3967	2.4383	0.3109	0.247726	0.00043	6.33E-05	0.00019	0.0003		1.3	2650	998.606	0.003572	2.9	10	0.2272	25988	9793	1.043	0.3392	0.0323	0.02062	3.0601	0.7581
12	GUY_1966	11	UNKNOWN	0.1155	0.04737	0.2826	2.4383	0.1676	0.147344	0.00057	6.33E-05	0.00019	0.0003		1.3	2650	998.606	0.000189	0.4	11	0.2205	25988	9793	0.822	0.2675	0.0287	0.023565	2.7735	0.4087
13	GUY_1966	12	UNKNOWN	0.3587	0.14711	0.4687	2.4383	0.3139	0.249627	0.00058	6.33E-05	0.00019	0.0003		1.3	2650	998.906	0.017636	12	12	0.2671	25988	9795.9	1.418	0.4613	0.0377	0.020373	3.0661	0.7654
14	GUY_1966	13	UNKNOWN	0.0847	0.03474	0.259	2.4383	0.1341	0.120811	0.00062	6.33E-05	0.00019	0.0003		1.3	2650	998.606	6.94E-05	0.2	13	0.2259	25988	9793	0.734	0.2385	0.0271	0.023491	2.7065	0.327
15	GUY_1966	14	UNKNOWN	0.3862	0.15839	0.5471	2.4383	0.2895	0.233947	0.00066	6.33E-05	0.00019	0.0003		1.3	2650	998.588	0.044457	28.1	14	0.3247	25988	9792.8	1.512	0.4917	0.0389	0.017828	3.0173	0.7059
16	GUY_1966	15	UNKNOWN	0.4193	0.17196	0.6066	2.4383	0.2835	0.230013	0.0007	6.33E-05	0.00019	0.0003		1.3	2650	998.569	0.089168	51.9	15	0.3638	25988	9792.6	1.577	0.5127	0.0397	0.016374	3.0053	0.6913
17	GUY_1966	16	UNKNOWN	0.1271	0.05213	0.3167	2.4383	0.1646	0.14502	0.00079	6.33E-05	0.00019	0.0003		1.3	2650	998.625	0.00177	3.4	16	0.2493	25988	9793.2	1.122	0.3648	0.0335	0.024498	2.7675	0.4013
18	GUY_1966	17	UNKNOWN	0.4717	0.19345	0.5987	2.4383	0.3231	0.255411	0.00083	6.33E-05	0.00019	0.0003		1.3	2650	998.734	0.161658	83.6	17	0.3364	25988	9794.2	2.076	0.6752	0.0456	0.01937	3.0845	0.7878
19	GUY_1966	18	UNKNOWN	0.1438	0.05898	0.3455	2.4383	0.1707	0.149735	0.00084	6.33E-05	0.00019	0.0003		1.3	2650	998.415	0.003415	5.8	18	0.267	25988	9791.1	1.232	0.4004	0.0351	0.023655	2.7797	0.4162
20	GUY_1966	19	UNKNOWN	0.1472	0.06037	0.3602	2.4383	0.1676	0.147344	0.00092	6.33E-05	0.00019	0.0003		1.3	2650	999.491	0.005069	8.4	19	0.281	25988	9801.7	1.329	0.4323	0.0365	0.023491	2.7735	0.4087
21	GUY_1966	20	UNKNOWN	0.5796	0.23771	0.7156	2.4383	0.3322	0.261064	0.00099	6.33E-05	0.00019	0.0003		1.3	2650	998.454	0.308942	130	20	0.3964	25988	9791.5	2.531	0.8228	0.0503	0.017961	3.1027	0.81
22	GUY_1966	21	UNKNOWN	0.6223	0.25522	0.9407	2.4383	0.2713	0.221916	0.001	6.33E-05	0.00019	0.0003		1.3	2650	998.375	0.31635	124	21	0.5767	25988	9790.7	2.173	0.7064	0.0467	0.012322	2.9809	0.6615
12385	Leeetal2004		Laboratory	0.054	0.09	0.6475	0.6	0.139	0.094989	0.002	0.00073	0.0022	0.0037		1.38	2650	998.234	0.0224	24.9	12670	0.5546	25988	9789.3	1.86	0.0522	0.0432	0.014379	0.878	0.0834
12386	Leeetal2004		Laboratory	0.06	0.1	0.6667	0.6	0.15	0.1	0.002	0.00073	0.0022	0.0037		1.38	2650	998.234	0.0349	34.9	12671	0.5497	25988	9789.3	1.958	0.0549	0.0443	0.014452	0.9	0.09
12387	Leeetal2004		Laboratory	0.06	0.1	0.6667	0.6	0.15	0.1	0.002	0.00073	0.0022	0.0037		1.38	2650	998.234	0.0252	25.2	12672	0.5497	25988	9789.3	1.958	0.0549	0.0443	0.014452	0.9	0.09
12388	Leeetal2004		Laboratory	0.036	0.06	0.5455	0.6	0.11	0.080488	0.002	0.00073	0.0022	0.0037		1.38	2650	998.234	0.0125	20.8	12673	0.5252	25988	9789.3	1.576	0.0442	0.0397	0.015284	0.82	0.066
12389	Leeetal2004		Laboratory	0.048	0.08	0.6299	0.6	0.127	0.089227	0.002	0.00073	0.0022	0.0037		1.38	2650	998.234	0.0234	29.2	12674	0.5644	25988	9789.3	1.747	0.049	0.0418	0.014176	0.854	0.0762
12390	Leeetal2004		Laboratory	0.06	0.1	0.6667	0.6	0.15	0.1	0.002	0.00073	0.0022	0.0037		1.38	2650	998.234	0.0317	31.7	12675	0.5497	25988	9789.3	1.958	0.0549	0.0443	0.014452	0.9	0.09
12391	Leeetal2004		Laboratory	0.072	0.12	0.7186	0.6	0.167	0.107281	0.002	0.00073	0.0022	0.0037		1.38	2650	998.234	0.0457	38.1	12676	0.5615	25988	9789.3	2.1	0.0589	0.0459	0.014052	0.934	0.1002
12392	Leeetal2004		Laboratory	0.084	0.14	0.7	0.6	0.2	0.12	0.002	0.00073	0.0022	0.0037		1.38	2650	998.234	0.048	34.3	12677	0.4998	25988	9789.3	2.349	0.0659	0.0485	0.015543	1	0.12
12393	Wongetal2007	1	Laboratory	0.081	0.162	1.2857	0.5	0.126	0.083777	0.0105	0.00237	0.0071	0.0118	0.0096	1.2	2550	998.234	0.234	144.236	12678	1.1566	25007	9789.3	8.611	0.0797	0.0929	0.015259	0.752	0.063
12394	Wongetal2007	2	Laboratory	0.067	0.134	1.2407	0.5	0.108	0.075419	0.012	0.00237	0.0071	0.0118	0.0096	1.2	2550	998.234	0.248	184.733	12679	1.2056	25007	9789.3	8.86	0.082	0.0942	0.01576	0.716	0.054
12395	Wongetal2007	3	Laboratory	0.044	0.088	1.0353	0.5	0.085	0.063433	0.0126	0.00237	0.0071	0.0118	0.0096	1.2	2550	998.234	0.116	131.645	12680	1.134	25007	9789.3	7.824	0.0724	0.0885	0.017245	0.67	0.0425
12396	Wongetal2007	4	Laboratory	0.038	0.076	1.0411	0.5	0.073	0.056502	0.0123	0.00237	0.0071	0.0118	0.0096	1.2	2550	998.234	0.068	89.3937	12681	1.2305	25007	9789.3	6.803	0.063	0.0826	0.015686	0.646	0.0365
12397	Wongetal2007	5	Laboratory	0.039	0.078	1.0986	0.5	0.071	0.055296	0.0152	0.00237	0.0071	0.0118	0.0096	1.2	2550	998.234	0.152	194.493	12682	1.3166	25007	9789.3	8.228	0.0762	0.0908	0.016289	0.642	0.0355
12398	Wongetal2007	7	Laboratory	0.069	0.138	1.1129	0.5	0.124	0.082888	0.0091	0.00237	0.0071	0.0118	0.0096	1.2	2550	998.234	0.12	86.881	12683	1.0092	25007	9789.3	7.384	0.0683	0.086	0.016295	0.748	0.062
12399	Wongetal2007	8	Laboratory	0.102	0.204	1.2994	0.5	0.157	0.096437	0.0086	0.00237	0.0071	0.0118	0.0096	1.2	2550	998.234	0.192	94.0292	12684	1.0472	25007	9789.3	8.119	0.0751	0.0902	0.015009	0.814	0.0785
12400	Wongetal2007	9	Laboratory	0.034	0.068	0.9577	0.5	0.071	0.055296	0.0126	0.00237	0.0071	0.0118	0.0096	1.2	2550	998.234	0.052	76.4122	12685	1.1478	25007	9789.3	6.821	0.0631	0.0827	0.017011	0.642	0.0355
12401	Wongetal2007	10	Laboratory	0.074	0.148	1.1298	0.5	0.131	0.085958	0.0081	0.00237	0.0071	0.0118	0.0096	1.2	2550	998.234	0.102	68.8715	12686	0.9968	25007	9789.3	6.816	0.0631	0.0826	0.015516	0.762	0.0655
12402	Wongetal2007	11	Laboratory	0.093	0.186	1.3381	0.5	0.139	0.089332	0.0108	0.00237	0.0071	0.0118	0.0096	1.2	2550	998.234	0.3	161.031	12687	1.1461	25007	9789.3	9.445	0.0874	0.0973	0.01552	0.778	0.0695
12403																													

solids_in_rivers_dissertation +

Figure 2: SolidsInRivers Database in MS Excel

Some of the records date to the 1930s, and some are recent (circa 2010). Many fields (individual column elements) are missing from the records and cannot be estimated from the study documents. Such a situation is a limitation for the equations, but not necessarily for the search algorithm. The idea and justification behind the screening tool is to non-parametrically locate within the database a set of five nearest (most similar) values using an L2 norm (Euclidean distance) as the measure of “closest” or “most similar" in the high-dimensional space. The screening tool then returns the nearest observed value, and a distance-weighted interpolated value of the un-queried value of unit solids discharge and solids concentration. The associated explanatory entries of the five nearest records in the database are also returned.

Unlike the charts, the analyst enters the database in the native unit-space of the input variables and the algorithm searches for nearest matches. In other words, dimensionality, which is so important in Peterson (1975), is not required for the screening tool. Upon completion of the search, the five nearest matches are returned and used in a subsequent computation to estimate by either distance-weighted mean or arithmetic mean, the value of unit solids discharge associated with the search values. The tool greatly

speeds up the process of estimating charge for a stream based on the large database developed in Cleveland et al. (2013).

Concept of Distance in N-Dimensional Space

The concept of distance is vital to the search engine. In the screening tool the search input values, *S*, *Q*, *U*, and D_{50} are compared to their commensurate values in the database, and a distance is computed from the search values to values in the database. The nearest values in N-dimensional distance are selected (actually by a sort) and then used for the estimation of unit solids discharge for the search values. The search engine has several different kinds of distances that the engineer may select.

Minkowski Distance

The distance between the search values and a database record is computed using Equation 9 (Tan et al., 2008).

$$L_p = \left(\left|x_{1,data} - x_{1,search}\right|^p + \left|x_{2,data} - x_{2,search}\right|^p + \cdots + \left|x_{N,data} - x_{N,search}\right|^p\right)^{\frac{1}{p}}$$

Equation 9

In Equation 9, p represents a parameter that modifies the Minkowski distance based upon its magnitude. When p > 0 and integer produces a quantity known as the Minkowski distance. The Euclidean distance between the two vectors $\boldsymbol{x}_{data}$ and $\boldsymbol{x}_{search}$, which is the hypotenuse-type distance that engineers are readily familiar with, is the special case of Equation 9 when $p = 2$, and the distance itself is called the L_2-norm. In many situations, the Euclidean distance is insufficient for capturing the actual distances in a given high-dimensional space if traverse of that space along a hypotenuse is infeasible. For example, taxi drivers in Manhattan should measure distance not in terms of the length of the straight line to their destination, but in terms of the Manhattan (taxi distance) distance, which takes into account that streets are either orthogonal or parallel to each other. The taxi distance is also called the L1 norm and is the special case of Equation 9 when $p=1$. This distance measures the shortest path along Cartesian axes (like city streets).

When some elements are unknown (as may be the case in our searches) or the noise in the elements is substantial, the Euclidean distance is not the most appropriate measure of distance, hence the value of p is left as a variable (Erickson,2010).

Data Value Standardization

The variables in the database are not expressed in the same magnitude, range, and scale. For example, discharge values are several orders of magnitude larger in the database than median grain diameter, hence the two are not directly comparable when computing a distance for the search algorithm. In such a case, one way to facilitate direct interpretation for comparing composite indices of the original data having different magnitudes and unit systems is to use normalization. Normalization serves the purpose of bringing the indicators into the same unit scale or unit base and makes distance computations appropriate. Normalizing data is done using various standardization techniques to assign a value to each variable so that they may be directly compared without unintentional bias due to differences in unit scale.

Z-score Standardization

Z-score standardization is a commonly used normalization method that converts all indicators to a common scale with an average of zero and standard deviation of one. This transformation is the same as computing a standard-normal score for each data value. The average of zero avoids the introduction of aggregation

distortions stemming from differences in indicators' means. The scaling factor is the standard deviation of the indicator across, for instance, the velocities, slopes or unit solids discharges being ranked. Thus, an indicator with extreme values will have intrinsically a greater effect on the composite indicator. The raw score on each data entry is converted to a Z-score, then distances are calculated using the Z-scores for each variable rather than the raw value. Upon completion of the distance calculations and selection of the nearest neighbors, the results are transformed back into the raw values for subsequent presentation. Equation 10 shows the basic z-score formula used to normalize data sets (TIBCO).

$$z = \frac{(x - \mu)}{\sigma}$$

Equation 10

where:

x = Data point value

$\mu = \text{Mean}$

$\sigma = \text{Standard Deviation}$

Unit-Interval [0,1] Standardization

An alternate approach considered for the screening algorithm is an option to use a mapping of each variable in the database to a [0,1] scale and linearly weight within the scale. This standardization has the same goal as Z-score, which is to prevent one variable from overwhelming the distance computations because of its relative magnitude. The unit interval [0,1] standardization technique differs from the Z-score in that the variability is governed by the minimum and maximum value for each variable, and hence extrapolation is not feasible. Because extrapolation is likely necessary until new records are added to the database, this standardization method was removed in the R implementation.

Unstandardized

The unstandardized approach was initially considered an option, but using this approach, discharge and/or velocity completely dominate the search algorithm, almost to the exclusion of the other variables. The option was useful for method testing and database error detection but was removed in the implementation presented here.

Search Engine Testing

The search algorithm was tested by using every entry in the database that could be used to generate estimates using the Meyer-Peter and Mueller model, the Wong and Parker model, the Recking model, the Schoklitsch model, and the Search model. The testing procedure is summarized as follows:

1. Read a record, extract the solids density, bed slope, d_{16}, d_{50}, d_{84}, hydraulic radius, mean section velocity, and unit discharge from the record. If any item is missing from the database, skip the record, otherwise:
2. Estimate unit solids discharge using the five models, along with the recorded unit solids discharge in the database.
3. Plot the ordered pairs $(q_{bm\text{-}model}, q_{bm\text{-}observed})$.
4. Record the instances where the model estimated a zero unit solids discharge, when the recorded value is non-zero.
5. Record the instances where the model estimated a non-zero unit solids discharge when the recorded value is zero.
6. Interpret the plots and summarize the zero/non-zero results.

Of the original 12,000+ records, over 9,000 were complete enough for the testing. Figure 2.3 is a 4-panel plot showing the estimated

unit solids discharge on the vertical axis and the observed unit solids discharge on the horizontal axis for each panel. The vertical and horizontal lines in each chart at 10^{-10} represent the cases where either the model (vertical) or the observed (horizontal) contained "zero" values. Using 10^{-10} is an exaggerated depiction of the difference between zero and non-zero values for the database record versus the SolidsInRivers model output. Because the recorded and model calculated values have such low magnitude, the instances with "zero" values were set to 10^{-10} to not confuse the user when interpreting the plot.

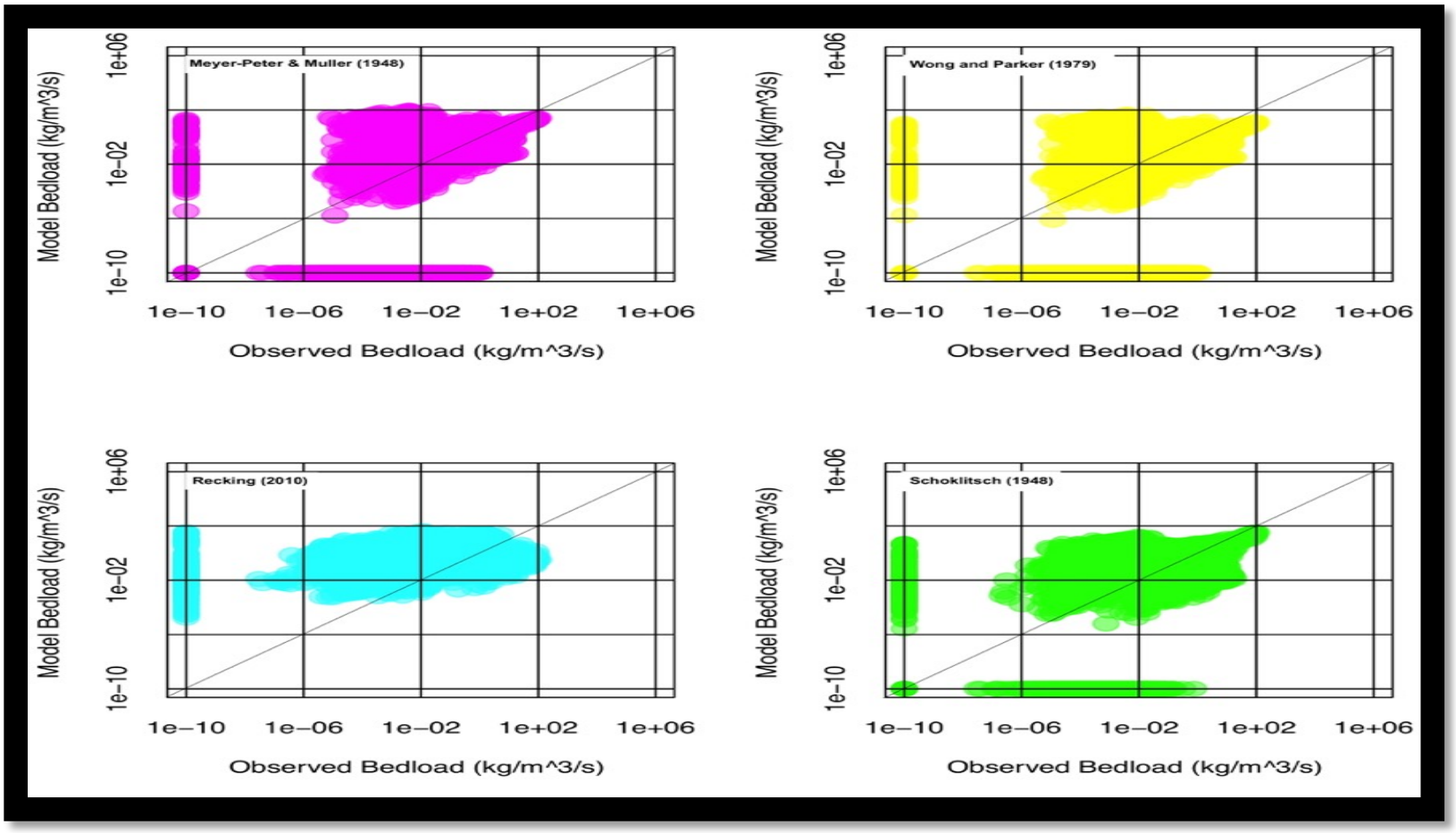


Figure 3: Estimated vs. Observed Solids Discharge by Method

Interestingly, the Recking (2010) model never estimated a zero unit solids discharge (when in fact there are several hundred such values in the database). The other three models did estimate zero values under certain conditions. The total number of records processed in the testing was 12,401 (the current database dimension), of those 12,401 records, 9,260 contained sufficient data for use in the testing. Table 1 is a summary of the zero/non-zero instances.

Table 1: Instances of Zero Values in Estimates or Observations

Model Name	Model=0, Observed=0	Model > 0, Observed=0	Model=0, Observed>0	Mismatch Rate at 0
MPM	102	203	3527	40.20%
WPK	102	203	3527	40.20%
REK	0	203	3527	40.20%
SHK	48	203	1640	19.90%
SRCH	19	203	0	2.20%

The summary of zero instances is meaningful in the context of the total records. If we consider the sum of the mismatched cases divided by the total number of useable records as a measure of efficiency, the database search is quite desirable. There is roughly a 2% chance that the search engine will return a zero value

"incorrectly" whereas the mismatch rate for the equation-based methods is at least 10-times greater.

Figure 4 is the more useful diagnostic of the utility of the search approach. The black cloud of markers is the result from the database search method (treated as if it were another equation) and the colored clouds represent the results using different models. The search for testing used the L_2 norm and estimated the unit solids discharge as the arithmetic mean of the five nearest neighbors. The value of five was selected as it is the simplex dimension of a 4-dimensional search space. It is not too surprising that the search appears to outperform the equation methods, as a database should be able to return its own values (if the averaging component is based on just the nearest neighbor, the resulting output degenerates to the equal value line. After this examination, the search-engine was then adapted to run on a web-server, with a web-based interface and server-side search.

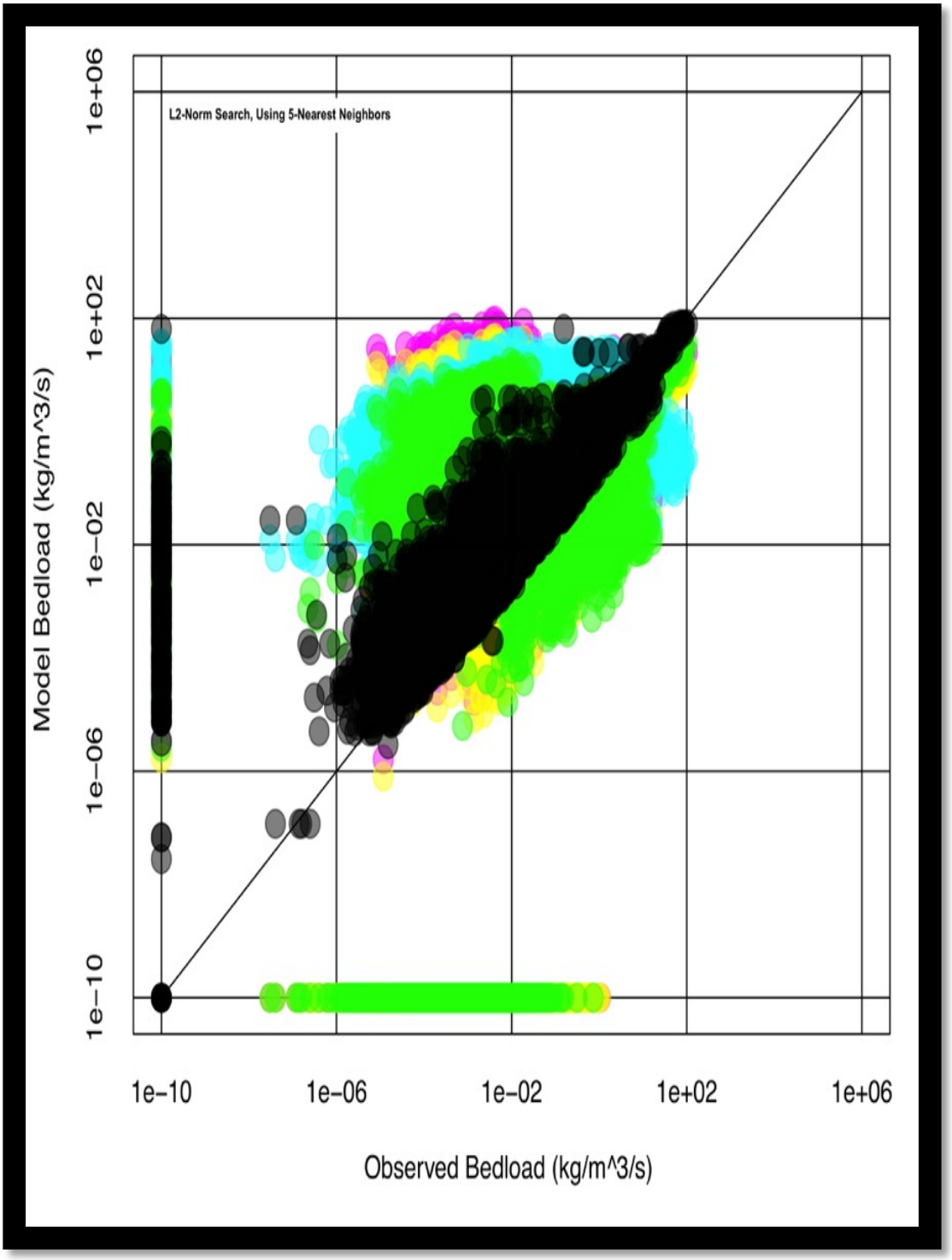


Figure 4: Estimated vs. Observed Solids Discharge Using SolidsInRivers Search Method

The Interface

Overview of Inter-Program Communication Concepts

This section will show figures of the SolidsInRivers interface and a flow chart of the inter-process communication.

The SolidsInRivers interface will be available on the internet as a tool for students and/or researchers who are in need of sediment transport estimates for river systems. The benefit of having the tool be largely web-based is twofold. The first advantage is that a web page and corresponding server may be accessed remotely, allowing the user to use the system from anywhere, at any time. This feature of a web-based tool makes it much easier for the casual user to take advantage of this free analytical model. Secondly, database updates can be done much more quickly and easily with a web-based approach. Updates will help the program, SolidsInRivers, evolve as new data becomes available which will increase the accuracy of the sediment transport estimations. The current system relies only on one single server for the database; therefore, any updates would only have to be done on the server. The interface can be further developed and refined over time

without requiring a complete redesign or requiring updates on each user's individual machine.

SolidsInRivers was developed using a hybrid method combining elements from HTML, Python, and R Scripts. The HTML script, hereafter known as "Entry", is used to introduce the model and collect data entry from the user. Next, the Python script, referred to as the "Launcher" harvests the input data and prepares it for the next step, the "Computation Engine". The Computation Engine is an R script that takes the inputs, performs the calculations, and outputs the raw data. Then another Python script, the "Responder", compiles the output and structures the results. Finally, the "Results" are presented to the end user using an HTML script. A flow chart depicting the interaction between these modules and each of their main responsibilities is shown below in Figure 5. Further detail on the interface development and in-depth discussion of inter-process methodology can be found in Chapter 2-NewANTS-Inter-Process Methodology.

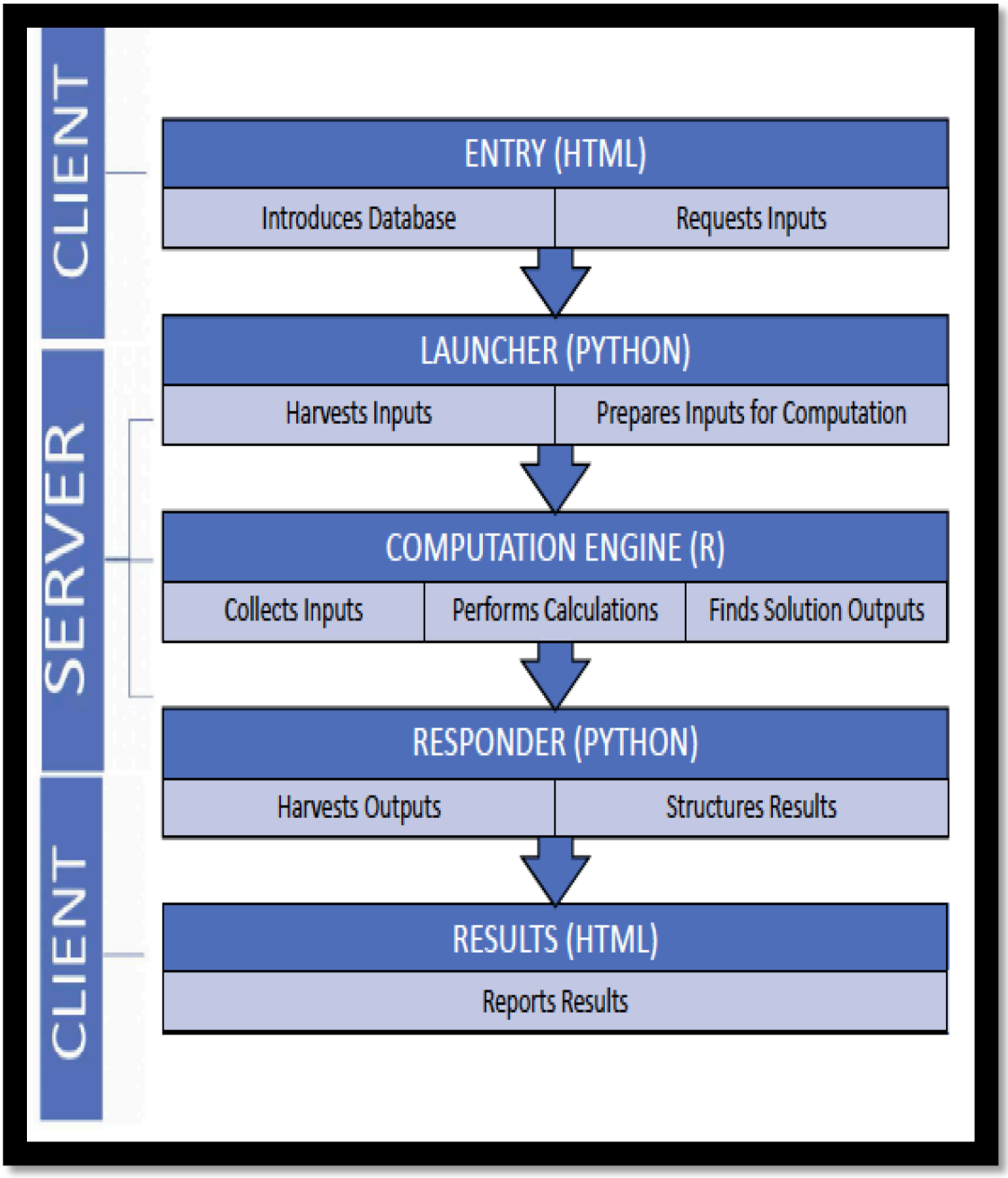


Figure 5: SolidsInRivers Inter-Process Communication Flow Chart

Interface Process/Screenshots

The Entry HTML script for SolidsInRivers (Shown in Appendix A) runs on a user's local computer. When the Entry script is opened in a web browser, a web page like Figure 6 is displayed. The Entry script requests the 6 inputs required by the distance weighting method. When the user clicks "Submit", the Entry script then calls the Launcher Python script that is located in the CGI-BIN directory of the web server. This directory is identified in the `http.conf` file when starting Apache (a similar configuration would apply to MS IIS).

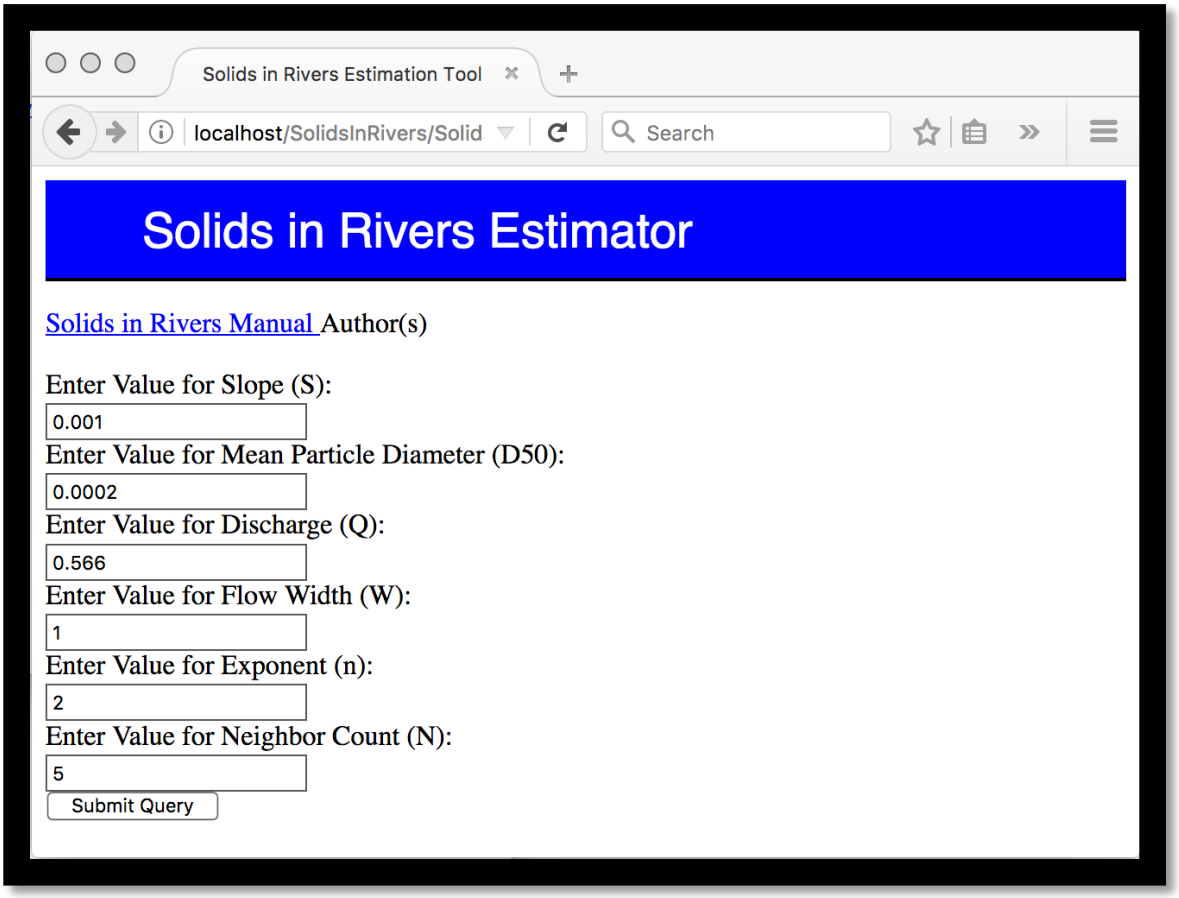


Figure 6: SolidsInRivers Interface Entry (HTML)

The Launcher script that actually parses the Entry form, builds an input file for the Computation Engine to process, and then runs the process can be found in Appendix B. The Launcher reads the form and writes the input data to a file located in a directory located in the web root. The directory is owned by `_www` (the web root) and permissions are set to allow a process running in CGI-BIN to write to the directory. Once the write is complete, the Launcher then calls the Computation Engine process (and executes the R script).

Once the Computation Engine process is complete, the output (which is sent to stdio) is captured and populates the `output` object. The Responder script, coded in Python, then builds the Results HTML script. The contents of the `output` object are then written to the Results output HTML file for display on the web server. Figure 7 is a screen capture of the web browser when the user submits the required inputs.

Solids in Rivers Estimate
Run Date : Sat Oct 7 20:04:17 2017
Return Code : 0

------ INPUT VALUES ------		
Channel Slope (S) =	0.001	Dimensionless
Mean Grain Diameter (D50) =	0.0002	meters
Discharge (Q) =	0.566	cubic meters per second
Mean Velocity (U) =	0.847	meters per second
Exponent (n) =	2.0	n = 2 is Euclidean Norm
Nearest Neighbor Count (N) =	5.0	Search Count
------ COMPUTED RESULT ------		
Estimated Solids Flux =	0.3784694	kg/m/sec
Estimated Solids Concentration =	250.4	parts per hundred thousand

Bar Plots of 5-Nearest Neighbor Values, Input Values, and Estimates (based on all search neighbors)

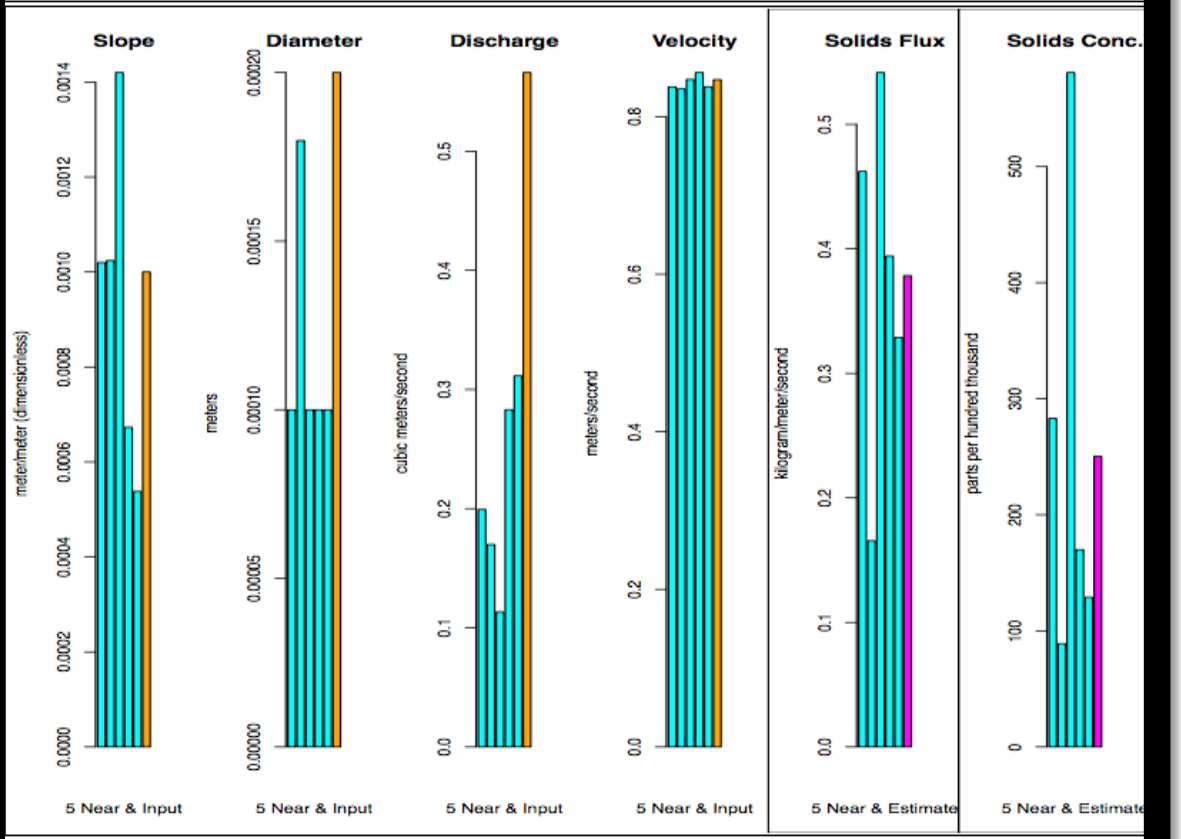


Figure 7: SolidsInRivers Interface Results (HTML)

Output Summary Table

SolidsInRivers outputs a summary table containing both the user-supplied inputs and the computed results of the database estimation tool. The units are specified on the right-hand side of the table in an attempt to avoid confusion.

Output Bar Plots

The database estimation tool also exports a series of bar plots of the five nearest neighbor values for the input and output variables. The input variables, Slope, Diameter, Discharge, and Velocity, are shown in the left four plots in Figure 7. The teal colors indicate the five nearest neighbor values, with the dark orange bars representing the Input values for each of the user supplied variables. The two bar plots on the right side of output interface show the result estimates of solids flux and solids concentration in the system. Again, the teal colors in Figure 7 represent the five nearest neighbor values computed by the algorithm. In the result plots, the estimates for the output values are shown on the bar plot in magenta. Both the output summary table and the output bar plots were shown previously in Figure 7.

Comparison of Results and Example Application

Several examples of use of the tool are presented here, as well as suggested practical uses. The first set of examples are queries based on the original Peterson (1975) work where three cases are presented. In practice the author found that the best way to use the tool is to supply a range of d_{50} values. In absence of other information, using a range built as $d_{50\text{-}low} = 0.1*d_{50}$, and $d_{50\text{-}high} = 10*d_{50}$ usually captured the observed response.

The results of using the search engine on these three cases are shown in Table 2. The search engine returned values that are order-of-magnitude correct when compared to the original observations, which is superior performance to the Peterson chart for the same application.

Table 2. Comparison of Peterson and Search Tool Estimates for d_{50} = 0.25 mm

Q (m3/s)	U (m/s)	Slope	"Observed" Result (ppht) (Blench, 1969)	"Chart" Result (ppht) (Peterson, 1975)	"Search Engine" Result (ppht)
0.14	0.3	0.000335	0 to 3	0.1	0.142 to 16.7
9.1	0.61	0.000165	0 to 3	1	0.032 to 0.096
580	1.21	0.000084	0 to 3	10	0.459**

** Search returned same value for the two size inputs

Additional examples are presented using the tool with selected experiments reported in Barnes (2013). These experiments are not yet part of the search database, so these comparisons represent a fair assessment of the utility of the search tool. Table 3 summarizes the results of using the size ranging input approach. Results where the estimation tool brackets the observation are marked with an asterisk (*).

Table 3. Comparison of Selected Experimental Results with Estimation Tool

Q (m3/s)	U (m/s)	Slope	d_{50} (m)	$q_{bm\text{-}obs}$ (kg/m/s)	$q_{bm\text{-}mod}$ (kg/m/s)
0.44	0.418	0.003	0.02	0.0037	3.8E-5 to 0.0085*
0.193	0.71	0.003	0.02	0.055	2.8E-5 to 0.073*
0.44	0.439	0.01	0.02	0.012	3.8E-5 to 0.007
0.029	1.16	0.01	0.02	0.163	0.0001 to 0.619*
0.432	0.442	0.01	0.02	0.006	2.4E-5 to 0.007*
0.0434	1.258	0.01	0.02	0.101	0.0003 to 3.26*

Example Application

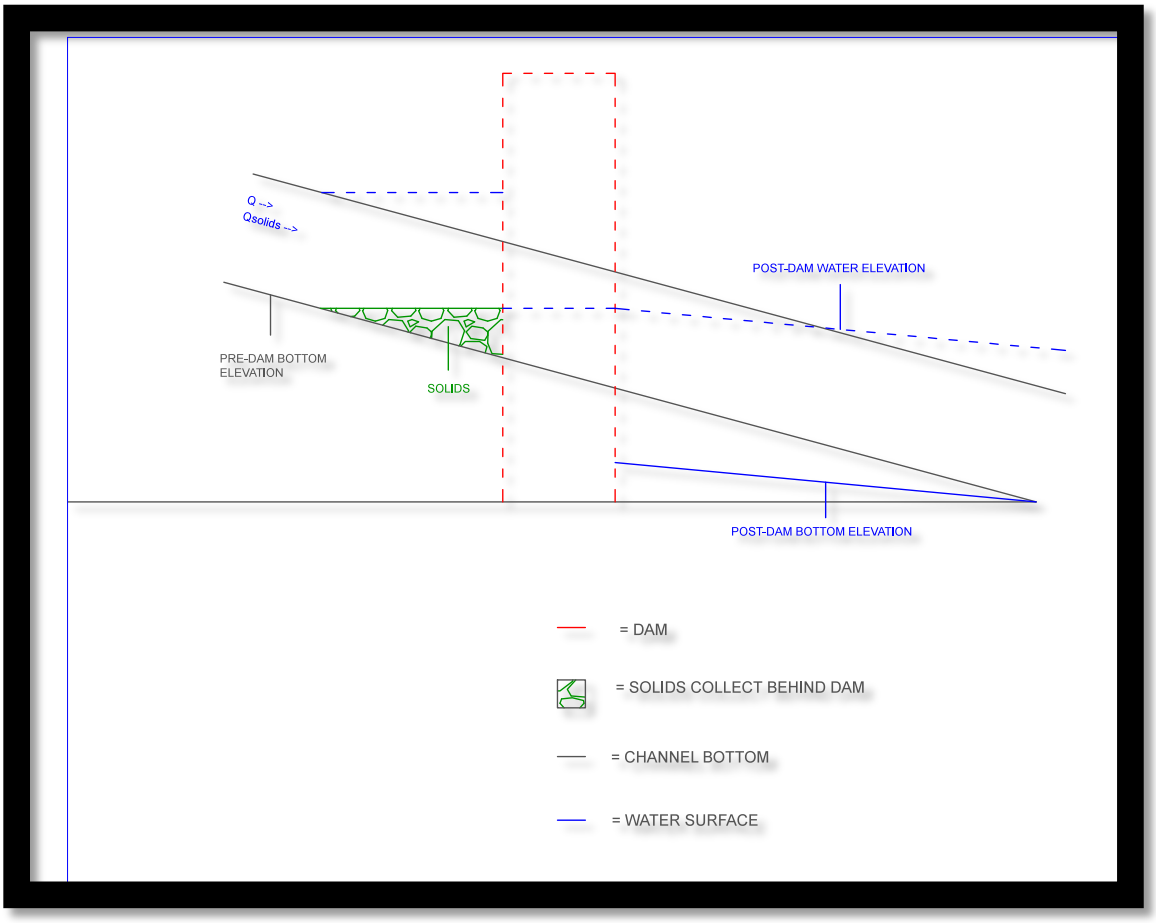


Figure 8: Example Application

A stream is to be blocked by a dam, and as a result the anticipated solids discharge downstream of the dam will be reduced to zero, but the average clear water discharge will be maintained. Assume the stream has slope of 0.0014, flow depth of 5 meters, discharge of 350 cubic meters per second, channel width of 36 meters, and a

median grain diameter of 0.02 meters (20 mm, about an inch). What channel changes are anticipated?

1. Calculate velocity using the equation Q = VA

$$Q = VA$$

$$V = \frac{Q}{A}$$

$$V = \frac{350}{5(36)}$$

$$V = 1.94\ m/s$$

2. Input data into SolidsInRivers. Output results give a calculated value of 0.102682 kg/m/s for the unit solids discharge

Solids in Rivers Estimate
Run Date : Thu Apr 5 04:05:51 2018
Return Code : 0

------ INPUT VALUES ------		
Channel Slope (S) =	0.0014	Dimensionless
Mean Grain Diameter (D50) =	0.02	meters
Discharge (Q) =	350.0	cubic meters per second
Mean Velocity (U) =	1.94	meters per second
Exponent (n) =	2.0	n = 2 is Euclidean Norm
Nearest Neighbor Count (N) =	5.0	Search Count
------ COMPUTED RESULT ------		
Estimated Solids Flux =	0.102682	kg/m/sec
Estimated Solids Concentration =	2.58426	parts per hundred thousand

3. Assume that once the dam is constructed, the unit solids discharge will decrease by several orders of magnitude→

Post-dam unit solids discharge estimate = 0.000102682 kg/m/s

4. Using SolidsInRivers, find values of slope and velocity that result in a unit solids discharge of 0.00102682 kg/m/s

Solids in Rivers Estimate
Run Date : Thu Apr 5 04:09:24 2018
Return Code : 0

------ INPUT VALUES ------		
Channel Slope (S) =	0.0005	Dimensionless
Mean Grain Diameter (D50) =	0.02	meters
Discharge (Q) =	350.0	cubic meters per second
Mean Velocity (U) =	0.51	meters per second
Exponent (n) =	2.0	n = 2 is Euclidean Norm
Nearest Neighbor Count (N) =	5.0	Search Count
------ COMPUTED RESULT ------		
Estimated Solids Flux =	0.0001748	kg/m/sec
Estimated Solids Concentration =	0.0075652	parts per hundred thousand

SolidsInRivers outputs a resultant unit solids discharge of 0.0001748 kg/m/s with a channel slope of 0.0005 and a velocity of 0.51 m/s.

Therefore, the channel downstream of the dam is anticipated to run at a slope of 0.0005 m/m and velocity of 0.51 m/s.

Future Extensions

Future extension of the database should be to enhance the algorithm to be able to weight the multiple responses in preparing the estimate. Another relatively immediate extension of the tool would be to include auxiliary functions to compute some of the terms that are done outside the current tool. In particular, the mean section velocity is related to the channel geometry as $Q=UWH$, where W is the width, and H is the hydraulic depth. All channels in the database are treated as equivalent rectangular channels. Such computations are relatively simple and could possibly be accomplished in JavaScript before submission of the form to the R Script.

A secondary extension would be the addition of capability to make estimates using the equation-based approach to provide the engineer with several estimates of unit solids discharge. An approximation for d_{16} and d_{84}, the missing variables needed for the equation-based models, is necessary. Further research should explore reasonable distribution shapes that would approximate the size distributions sufficiently to infer these two variables.

The database itself should be updated annually to include more recent unique studies. As the database grows in size, then the

granularity of estimates will become finer, approaching a continuum like the equation-based models.

Conclusions

SolidsInRivers presented the development and deployment a database-search engine running in R but accessible via a web interface. The search processing was all server-side--operating on a single, central database. The search algorithm results compared favorably to a set of equation-based models to accomplish the same task. Several examples were presented to illustrate the use of the tool, and a potential application was demonstrated. The drawback of search is that underlying physical model structure is unknown – the search cannot easily explore the meaning and interactions of the search variables. Nevertheless, it is a useful tool for estimation. The value of the search approach is simplicity of use (because of the interface) and when new experimental studies become available, they can be employed immediately – the database is simply extended without any change to the underlying search engine. Because the search engine relies upon a database, that database can be made available for other researchers wishing to explore the various variable interactions and improve the various regression models in use.

CHAPTER 3

NEWANTS

Introduction

NewANTS is a restoration of an old system (Chuang, 1998) that was designed to approximate analytical contaminant transport in groundwater. It is a collection of analytical and semi-analytical solutions to various forms of the advection-dispersion equation in different geometries.

NewANTS analytical solutions to contaminant transport scenarios were designed as an online tool. The online tool allows easy access to the system from the user's own machine and will aid in future development of the interface by relying on a single server.

NewANTS utilizes a hybrid of Python and R to take user supplied inputs and export a contaminant profile and a table of the results. The search engine was developed in R, the web-interface in HTML, and the inter-process communications using Python.

Once the contaminant transport analysis is complete, the results are returned to the user and displayed via the web interface. The solution is described in three different ways, a concentration profile plot, a table of the calculated concentrations at equal intervals along the entire model distance, and a condensed summary table of the final solution. These detailed output files were designed so the clients can plot (or otherwise interpret) their simulation results in greater detail and with improved accuracy.

This chapter documents the added HTML, Python, and web server configuration requirements to produce both tabular and graphical output and deliver them from a web server.

History

The historical version of the Analytical Numerical Transport modeling System (ANTS) was developed in Chuang (1998). NewANTS is designed to replicate and extend the original work. ANTS was created in Java and, initially, incorporating graphical elements into the system was a challenge. NewANTS utilizes graphical output directly through the HTML and JavaScript standards. These limitations in the original version of ANTS,

coupled with the loss of the original source codes for ANTS during a routine system backup, created a need for an updated system.

Literature Review

There are 48 transport model scenarios available in NewANTS. A flowchart depicting the individual modules that make up the NewANTS toolkit and a table of all modelling combinations available are included within this section.

Dimension and Source Type

Four different source types are represented for each of the dimensional systems (1D, 2D, 3D). These source types are impulse, injection, step-function, and finite length injection.

Instantaneous Source vs. Continuous Source

There are two general types of contamination conditions. The first is an instantaneous source. An instantaneous source is characterized as a one-time addition of a contaminant into an aquifer. The addition is assumed to happen so quickly that it is considered to be instantaneous. A spill from an oil tanker that runs into an aquifer through an open well is an example of a

contamination event that would be considered instantaneous. Two source types are instantaneous scenarios in NewANTS, Impulse and Step-Function. The other generalized contaminant condition is a continuous source. Continuous sources are constant strength contaminant additions to an aquifer over a period of time. A waste dump that leaches pollutants into an aquifer is an example of a continuous source. Two source types are continuous scenarios in NewANTS, Injection and Finite-Length Injection (Uddameri, 2016).

Governing Equation

Equation 11 shows the governing equation for contaminant transport analysis. It assumes unidirectional flow in the +x direction, and dispersion in the x, y, and z directions.

$$R\frac{\partial C}{\partial t} = D_x\frac{\partial^2 C}{\partial x^2} + D_y\frac{\partial^2 C}{\partial y^2} + D_z\frac{\partial^2 C}{\partial z^2} - v\frac{\partial C}{\partial t} - \lambda_D C$$

Equation 11

where:

$v = q/n$

q = Unit flow rate (m^2/d)

n = Porosity (dimensionless)

D_x = Dispersion coefficient in the x-direction (m^2/d)

D_y = Dispersion coefficient in the y-direction (m^2/d)

D_z = Dispersion coefficient in the z-direction (m^2/d)

R = Retardation Factor (dimensionless)

λ_D = First order decay rate (d^{-1})

The partial-order differential equation describes a generic three-dimensional geometry that includes the effects of Advection, Dispersion, Retardation, and Decay. The solutions for concentration levels are a function of position and time, C(x, y, z, t). The Equation 11 forms the basis for all calculations done in

NewANTS subject to various initial and boundary conditions. In addition, each modeling scenario uses a version of Equation 11 modified depending on dimensionality and the presence of reactions/Retardation. The different convolutions of Equation 11 for each available NewANTS modeling scenario will be discussed in further detail in the following section.

One-Dimensional Models

Impulse

Figure 9 below shows a side view depiction of a theoretical "physical system" and its related output concentration profile from an impulse source in one-dimension.

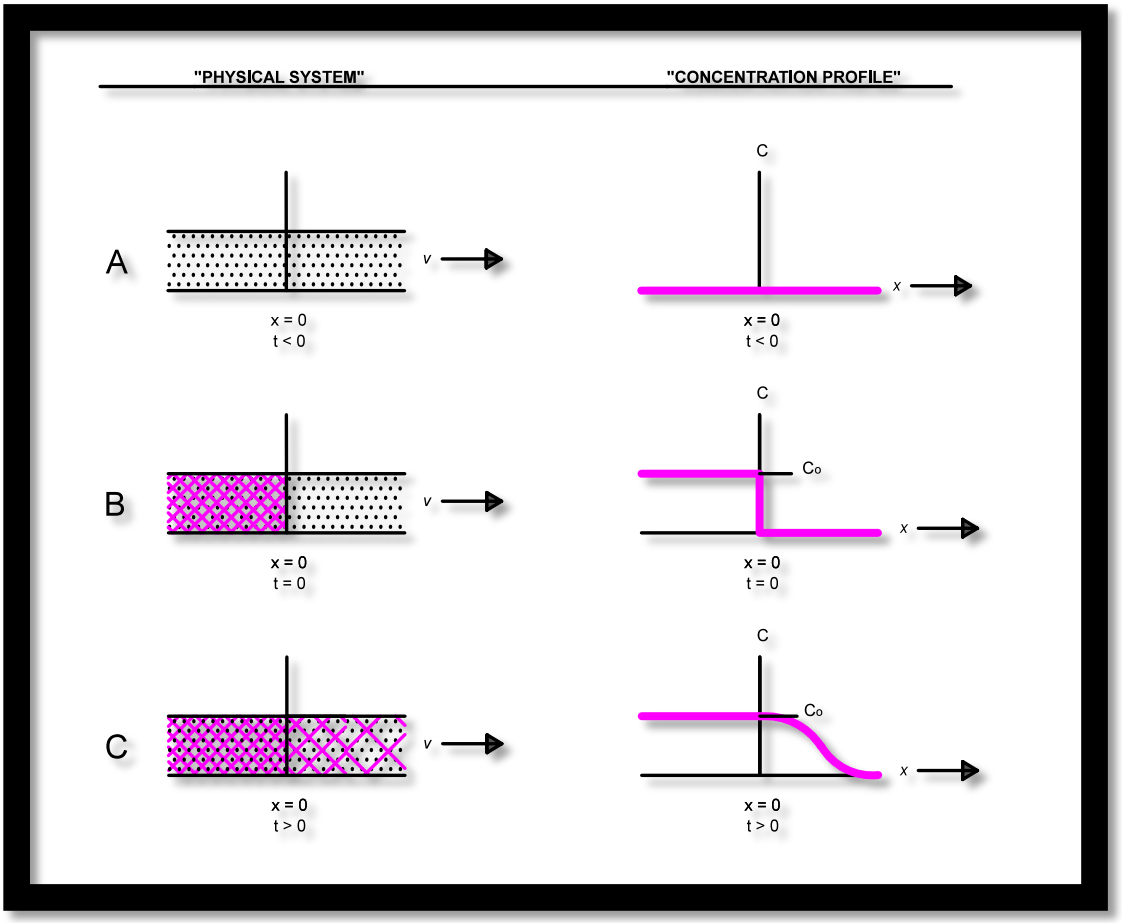


Figure 9: One-Dimensional, Impulse Source Conceptual Model

Figure 9-A shows the concentration profile along the x-axis at a time less than zero. The concentration is zero everywhere.

Figure 9–B shows the physical system and the concentration profile along the x-axis at time equal to zero. At $x < 0$, the concentration is suddenly raised to a value of C_o everywhere to the left of the origin ($x = 0$ is the origin). This condition represents a step function input and is a suitable approximation of an upstream source zone that has a constant concentration. The concentration

to the right of the origin is still zero because the model is at time = zero.

Figure 9–C shows the concentration profile along the x–axis at some time greater than zero. The contaminant source mass has moved to the right of the origin (along the direction of flow). The distance the contaminant has moved depends on is determined by the species velocity and dispersed along the translational front, proportional to the dispersivity in the system. The analytical solution to this transport model is the Ogata-Banks equation, Equation 12 (Ogata and Banks, 1961).

$$C(x,t) = \frac{C_o}{2}\left[erfc\left(\frac{x - vt}{2\sqrt{Dt}}\right) + exp\left(\frac{xv}{D}\right) erfc\left(\frac{x + vt}{2\sqrt{Dt}}\right)\right]$$

Equation 12

where:

C_0 = Source concentration (mg/L)

x = Distance (meters)

v = Mean seepage velocity (m/s)

t = Time (days)

D = Dispersion coefficient (m^2/d)

erfc = Complementary error function

This solution is also applicable to porous media flow when the velocity is equal to the seepage velocity divided by the porosity. The seepage velocity is the apparent groundwater velocity through the bulk of the porous medium.

Injection

Figure 10 below shows a side view depiction of a theoretical "physical system" and its related output concentration profile from an injection source in one-dimension.

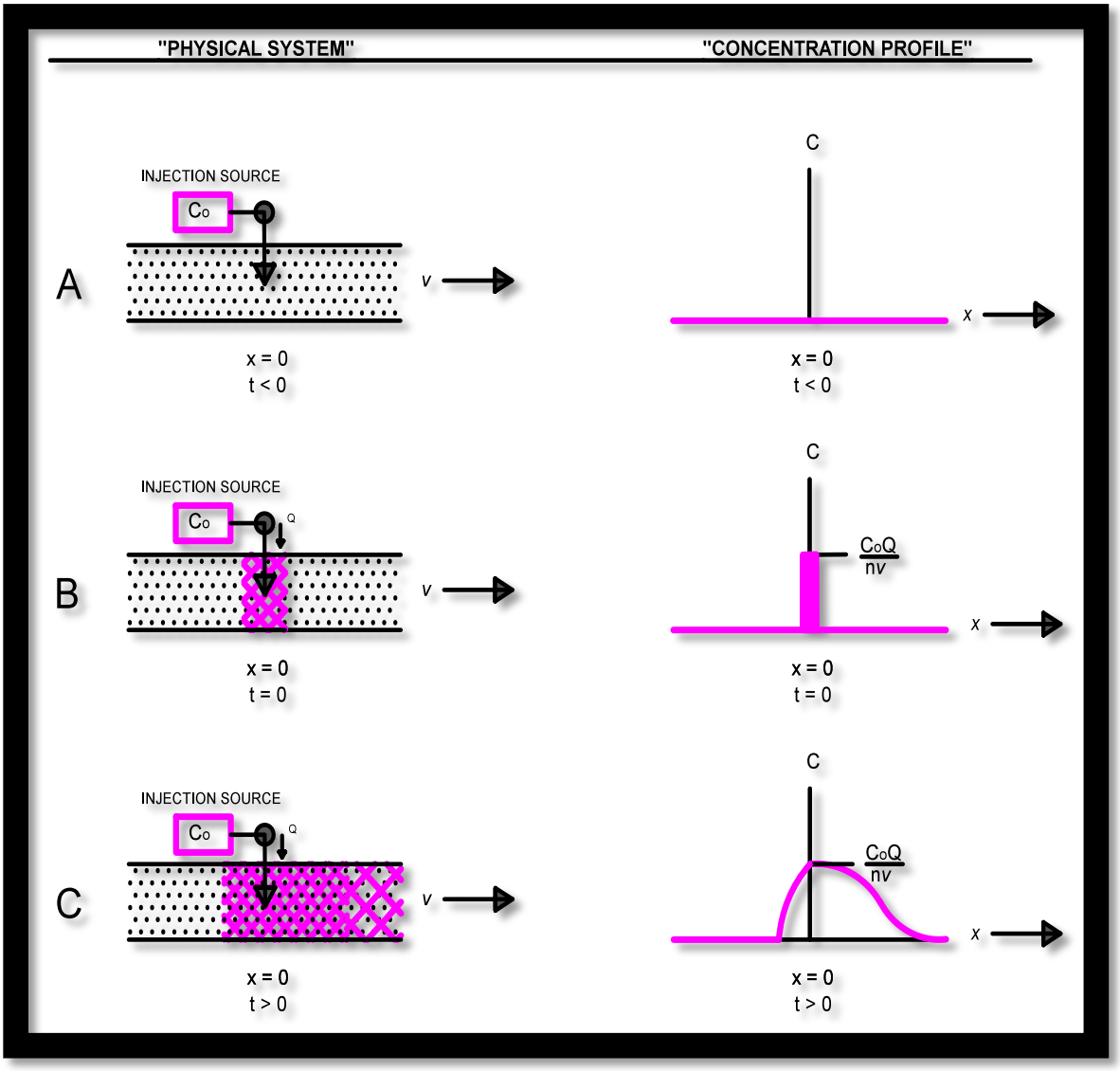


Figure 10: One-Dimensional, Injection Source Concptual Model

Figure 10-A shows the concentration profile along the x-axis at a time less than zero. The concentration is zero everywhere. The injection supply is depicted by the small reservoir and the injection path into the aquifer or stream. In this figure, the injection source is closed. Figure 10-B shows the physical system and the concentration profile along the x-axis at time equal to zero. At t = 0, the contaminant mass is added to the aquifer or stream at a rate of C_oQ located at the origin (x = 0). The mass flow rate is maintained at the injection site from t = 0 onward. Figure 10-C shows the concentration profile along the x-axis at some time greater than zero. The concentration front has moved to the right of the origin (along the direction of flow). The distance the contaminant has moved depends on is determined by the species velocity and dispersed along the translational front, proportional to the dispersivity in the system. In addition, some mass diffuses upstream from the injection site. The analytical solution to this transport model is the Sauty (1980) equation, Equation 13.

$$C(x,t) = \frac{C_o Q}{2nv} exp\left(\frac{xv}{2D}\right)\left[exp\left(\frac{-|x|v}{2D}\right) erfc\left(\frac{|x| - vt}{2\sqrt{Dt}}\right) - exp\left(\frac{|x|v}{2D}\right) erfc\left(\frac{|x| + vt}{2\sqrt{Dt}}\right)\right]$$

Equation 13

where:

C_0 = Source concentration (mg/L)

Q = Injection rate (m^3/s)

n = Porosity (dimensionless)

v = Mean seepage velocity (m/day)

x = Distance (meters)

D = Dispersion coefficient (m^2/day)

t = Time (days)

erfc = Complementary error function

This solution is also applicable to porous media flow where the velocity is equal to the seepage velocity divided by the porosity. The seepage velocity is the apparent groundwater velocity through the bulk of the porous medium.

Step-Function

Figure 11 below shows a depiction of a theoretical "physical system" and its related output concentration profile from a step-function source. The system has an initial concentration of zero at all locations. At t = 0, the concentration at the source suddenly increases to the initial concentration, C_0, and is maintained at that value from then on. At t > 0, the concentration front translates and spreads to occupy more and more space.

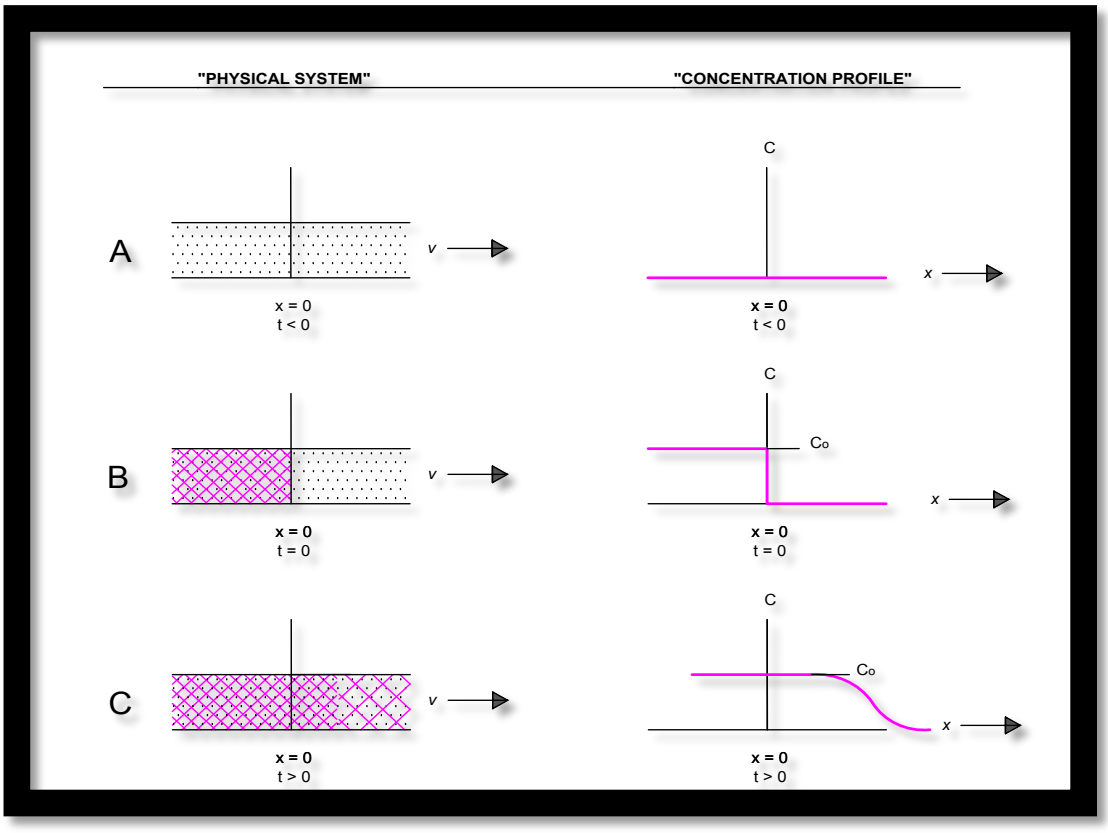


Figure 11: One-Dimensional, Step-Function Source Conceptual Model

The analytical solution to this transport model is Equation 14. (Yuan, 1995)

$$C(x,t) = \frac{C_o}{2}\left[erfc\left(\frac{x - vt}{2\sqrt{Dt}}\right) - exp\left(\frac{xv}{D}\right) erfc\left(\frac{x + vt}{2\sqrt{Dt}}\right)\right]$$

Equation 14

where:

C_0 = Source concentration (mg/L)

x = Distance (meters)

v = Mean seepage velocity (m/s)

t = Time (days)

D = Dispersion coefficient (m^2/d)

erfc = Complementary error function

Finite Length Injection

Figure 12 below depicts a finite-length injection source type. A corresponding source history is also shown. The system has an initial concentration of zero at all locations. At t = 0, the concentration at the source suddenly increases to the initial concentration, C_0, and mass is added at a rate of C_0Q at the origin. This rate is maintained at that value for a specified duration, tau, when the source then decreases back to zero. Tau is equal to Ct* where t* is the time when further addition of mass is terminated. At t > 0, a concentration front translates and spreads to occupy more and more space. Some mass diffuses upstream from the injection location during the injection period. When injection is terminated, the "pulse" translates and spreads downstream.

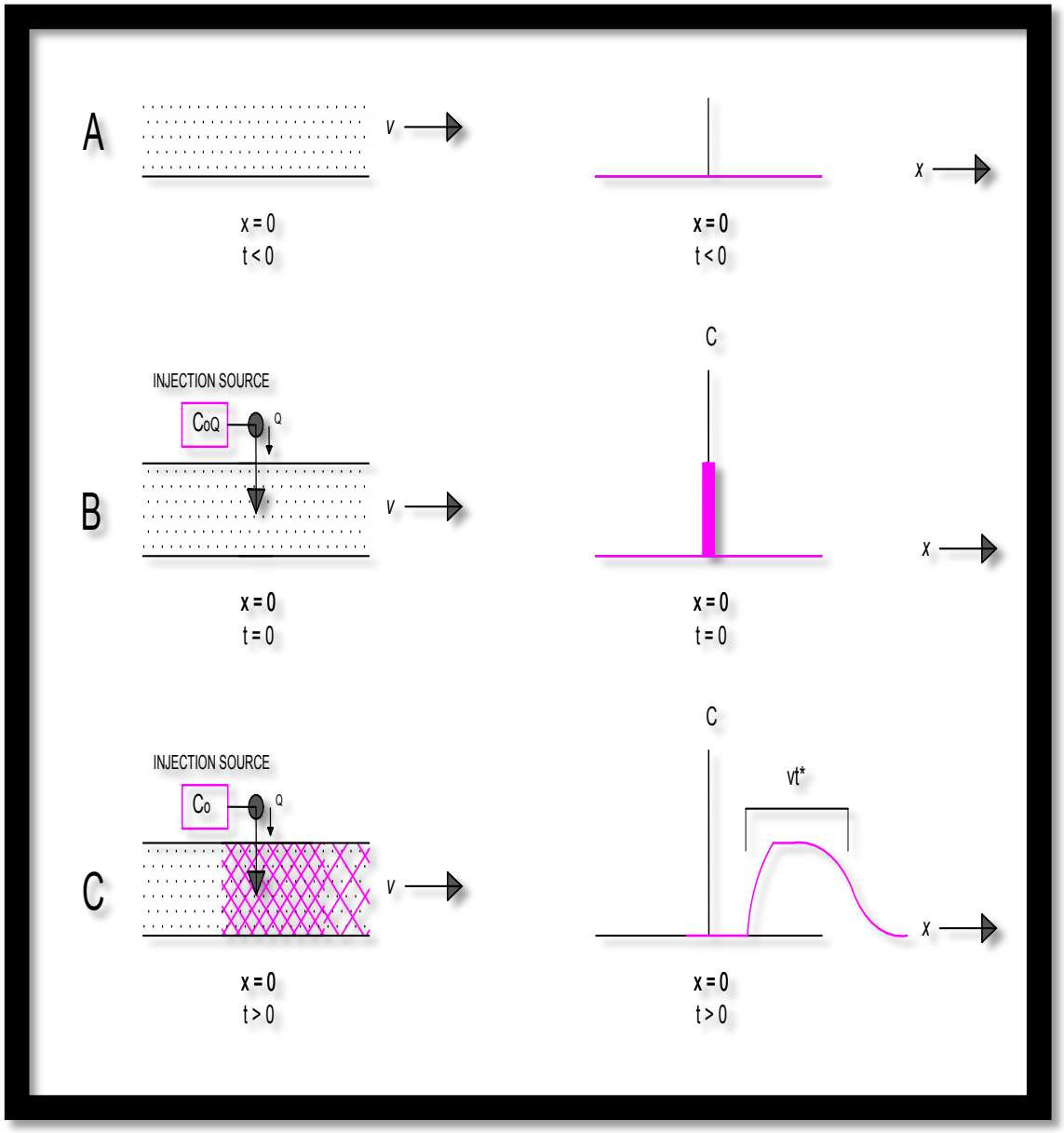


Figure 12: One-Dimensional, Finite Length Injection Source Conceptual Model

The function that satisfies these conditions is obtained by superposition of two injection solutions lagged in time. Both

Sauty and Ogata-Banks solutions can be used as the injection model, and the results will be nearly identical. The analytical solution to this transport model is Equation 15. (Yuan, 1995)

$$C(x,t) = \int_0^t \left[\left(\frac{C_0 v}{2\sqrt{D_x(t-\tau)}}\right) exp\left(\frac{(x - v(t-\tau))^2}{4D_x(t-\tau)}\right)\right] d\tau$$

Equation 15

where:

C_0 = Source concentration (mg/L)

x = Distance (meters)

v = Mean seepage velocity (m/s)

t = Time (days)

D_x = Dispersion coefficient (m^2/d)

τ = Source duration (days)

The closed form solution to the integral is given in Equation 16. Either the Ogata-Banks (1961) or the Sauty (1980) solutions can be used as the results will be nearly identical.

$$C(x,t) = \frac{C_o}{2}\left[erfc\left(\frac{x - vt}{2\sqrt{D_x t}}\right)\right] - \left\{exp\left[\frac{(xv)}{D_x}\right] erfc\left[\frac{x + vt}{2\sqrt{D_x t}}\right]\right\}$$

Equation 16

C_0 = Source concentration (mg/L)

x = Distance (meters)

v = Mean seepage velocity (m/s)

t = Time (days)

D_x = Dispersion coefficient in the x-direction m^2/d)

erfc = Complementary error function

Two-Dimensional Models

The 2-dimensional models assume that the aquifer length (in the longitudinal direction) is infinite and that a contaminant source is introduced at the origin (x,y) = (0,0). A conceptual model of a two-dimensional system is shown below in Figure 13. Figure 13-A depicts a two-dimensional continuous contamination scenario, which describes conditions from an Injection or Finite-Length Injection source type. Figure 13-B shows a two-dimensional instantaneous contamination scenario, which describes conditions from an Impulse or Step-Function source type (HWB, 2018).

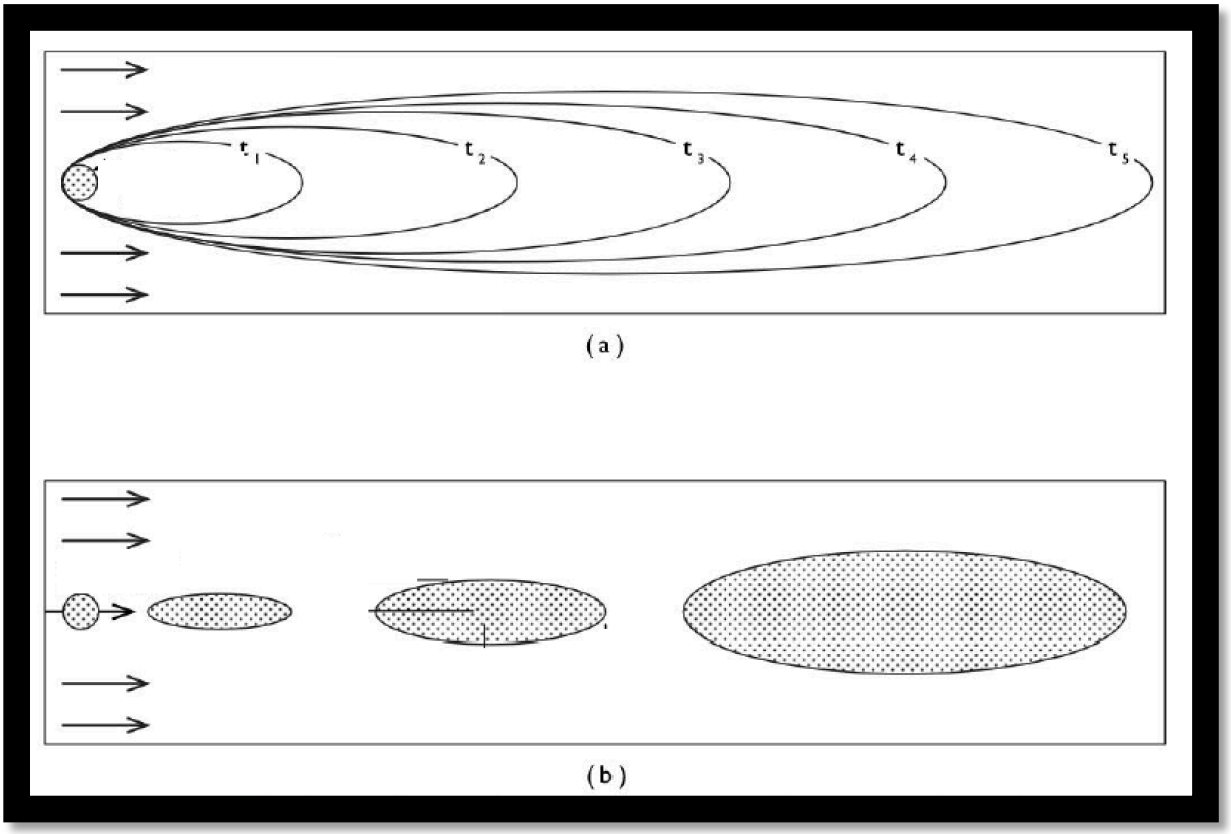


Figure 13: Two-Dimensional Source Type Conceptual Model

Impulse

An impulse consists of a one-time injection of a solute into the aquifer at time = 0. This source type is assumed to be instantaneous. Equation 17 is the analytical solution of the system (Yuan, 1995):

$$C(x,y,t) = \left(\frac{RC_0A}{4\pi t\sqrt{D_xD_y}}\right)exp\left(-\frac{(Rx - vt)^2}{4D_xt}\right) - \left(\frac{(Ry)^2}{4D_yt}\right)$$

Equation 17

where:

C_0 = Source concentration (mg/L)

R = Retardation factor (dimensionless)

D_i = Dispersion coefficient in the i-direction (m^2/d)

v = Mean seepage velocity (m/day)

x = Distance in the x-direction (m)

y = Distance in the y-direction (m)

t = Time (days)

Injection

A 2D transport model with a constant injection source located at the origin is assumed to have a line source with a fluid contribution that is negligible on the local hydraulics. The initial, boundary, and mass conservation conditions are shown below in Equation 18.

$$C(x, y, 0) = 0$$

$$C(\pm\infty, \pm\infty, t) = 0$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} C(x, y, t)dxdy = C_0 Qt$$

Equation 18

A solution is obtained by the time-convolution of an elementary line source and is:

$$C(x, y, t) = \left(\frac{C_0 Q}{4\pi nL\sqrt{D_x D_y}}\right) exp\left(\frac{xv}{2D_x}\right) W(a, b)$$

Equation 19

W (a , b) = Leaky aquifer well function

$$a = \frac{R^2}{4D_x t}$$

$$b = \frac{Rv}{2D_x}$$

$$R = x^2 + y^2 \left(\frac{D_x}{D_y}\right)$$

The leaky aquifer function can be evaluated numerically using the following recursive definition, Equation 20-Equation 22, or by efficient approximation. The solution utilized in NewANTS uses polynomial and finite-series approximations (Chuang, 1998).

$$W(a,b) = \sum_{n=0}^{\infty} \frac{E_{n+1}(a)}{n!} \left[\frac{b^2}{4a}\right]^n$$

Equation 20

$$E_{n+1}(a) = \frac{1}{n}[\exp(-a) - aE_n(a)]$$

Equation 21

$$E_1(a) = \int_{-\infty}^{a} \frac{\exp(x)}{x} dx$$

Equation 22

The solution is applicable for porous media flow. The velocity is specified as the mean seepage velocity which is equal to velocity divided by the porosity.

Step-Function

A two dimensional constant/step function source is a finite-length horizontal line source in an aquifer of infinite extent located at (x,y) = (0,0). The length of the source is defined as Y. The system has an initial concentration of zero at all locations. At t = 0, the concentration at the source suddenly increases to the initial concentration, C_0, and is maintained at that value from then on. The transport processes modeled are advection along the x-axis and dispersion in both the x- and y-direction.

Domenico and Robbins (1985, 1987) and Hunt (1985) published analytical solutions that were based on the product of three-orthogonal one-dimensional solutions. One solution, usually the

x-direction, accounts for the advection-dispersion component of the problem, and the other solutions account for spreading from dispersive effects transverse to the mean flow direction (assumed to be the x-direction).

The principle of superposition and convolution are used in the calculations to construct solutions for various source shapes. The analytical solution is obtained by the spatial integration of the 3D point-impulse model, shown below in Equation 23 (Domenico and Robbins 1985).

$$C(x,y,z,t) = \int_{-\frac{X}{2}}^{\frac{X}{2}}\int_{-\frac{Y}{2}}^{\frac{Y}{2}}\int_{-\frac{Z}{2}}^{\frac{Z}{2}}\left(\frac{M}{2n\sqrt{\pi D_x t}}\right)\left[exp\left(\frac{-(x-\ vt)^2}{4D_x t}\right)\left(\frac{1}{2\sqrt{\pi D_y t}}\right)\ exp\left(\frac{-y^2}{4D_y t}\right)\left(\frac{1}{2\sqrt{\pi D_x t}}\right)exp\left(\frac{-z^2}{4D_x t}\right)\right]dzdydx$$

Equation 23

The closed form solution to this integral is

$$C(x,y,z,t) = \ wC_1(x,t)C_2(y,t)C_3(z,t)$$

where

$$w = \frac{C_0}{8}$$

$$C_1(x,t) = \left(erf\left(\frac{x - vt + \frac{X}{2}}{2\sqrt{D_x t}}\right) - erf\left(\frac{x - vt - \frac{X}{2}}{2\sqrt{D_x t}}\right)\right)$$

$$C_2(y,t) = \left(erf\left(\frac{y + \frac{Y}{2}}{2\sqrt{D_y t}}\right) - erf\left(\frac{y - \frac{Y}{2}}{2\sqrt{D_y t}}\right)\right)$$

$$C_3(z,t) = \left(erf\left(\frac{z + \frac{Z}{2}}{2\sqrt{D_z t}}\right) - erf\left(\frac{z - \frac{Z}{2}}{2\sqrt{D_z t}}\right)\right)$$

Equation 24

Additional solutions are based on either the convolution of this solution and/or replacement of C_1 and w by appropriate values to approximate different source conditions.

Replace with Ogata Banks or Sauty's solution and adjust the weights

$$C_1(x,t) = \frac{C_0}{2}\left(erfc\left(\frac{x - vt}{2\sqrt{D_x t}}\right) - exp\left(\frac{xv}{D_x}\right) erfc\left(\frac{x + vt}{2\sqrt{D_x t}}\right)\right)$$

w = ¼ (Ogata and Banks, 1961)

Equation 25

$$C_1(x,t) = \frac{C_0 Q}{2nv} exp\left(\frac{xv}{2D_x}\right)\left(exp\left(\frac{-|x|v}{2D_x}\right) erfc\left(\frac{|x| - vt}{2\sqrt{D_x t}}\right) - exp\left(\frac{|x|v}{2D_x}\right) erfc\left(\frac{|x| + vt}{2\sqrt{D_x t}}\right)\right)$$

w = ¼ (Sauty, 1980)

Equation 26

where:

C_0 = Source concentration (mg/L)

x = Distance (meters)

v = Mean seepage velocity (m/s)

t = Time (days)

D_i = Dispersion coefficient in the i-direction (m^2/d)

erfc = Complementary error function

Q = Injection rate (m^3/s)

n = Porosity (dimensionless)

The next step is to remove the z-dimension for the 2-dimensional cases where the system is assumed to be well mixed in the vertical direction (z is usually upward). The solution can be approximated by assigning the z-direction terms a value of 1 and adjusting the weight in the model.

Finite Length Injection

A two-dimensional finite-length injection is a horizontal line source in an aquifer of infinite extent located at (x,y) = (0,0). The length of the source is defined as Y. At t = 0, the concentration at the source suddenly increases to the initial concentration, C_0, and is maintained at that value specified duration, tau, when the source then decreases back to zero. The transport processes modeled are advection along the x-axis and dispersion in both the x- and y-direction. The mathematical model is shown below in Equation 27 (Yuan, 1995).

$$C\ (x, y, t) = \int_{-\frac{Y}{2}}^{\frac{Y}{2}} \int_0^t \frac{C_0 v\ exp\left[-\frac{(x\ -\ v(t-\ \tau))^2}{4D_x(t-\ \tau)} - \frac{(y-\ y')^2}{4D_y(t-\ \tau)}\right]}{4\pi(t-\ \tau)\sqrt{D_x D_y}} d\tau dy'$$

Equation 27

The closed form solution to the integral is given in Equation 28. Either the Ogata-Banks (1961) or the Sauty (1980) solutions can be used as the results will be nearly identical.

$$C(x,y,z,t) =$$

$$\frac{C_o}{8}\left[erfc\left(\frac{x-vt}{2\sqrt{D_x t}}\right)\right]\left\{erf\left[\frac{\left(y+\frac{Y}{2}\right)}{\sqrt{2D_y\frac{x}{v}}}\right]-erf\left[\frac{\left(y-\frac{Y}{2}\right)}{\sqrt{2D_y\frac{x}{v}}}\right]\right\}\left\{erf\left[\frac{\left(z+\frac{Z}{2}\right)}{\sqrt{2D_z\frac{x}{v}}}\right]-erf\left[\frac{\left(z-\frac{Z}{2}\right)}{\sqrt{2D_z\frac{x}{v}}}\right]\right\}$$

Equation 28

where:

C_0 = Source concentration (mg/L)

x = Distance (meters)

v = Mean seepage velocity (m/s)

t = Time (days)

D_i = Dispersion coefficient in the i-direction (m^2/d)

erfc = Complementary error function

Q = Injection rate (m^3/s)

n = Porosity (dimensionless)

Y = Source dimension in the y-direction (m)

The solution above is given in three-dimensions, so the final step is to remove the z-dimension for the two-dimensional case by assigning the z-direction terms a value of 1. Further detail on this method is outside the scope of this research but can be found in Domenico and Robbins (1985).

Three-Dimensional Models

The 3-dimensional models assume that the river or aquifer length (in the longitudinal direction) is infinite with a contaminant source zone that is Y units wide and Z units tall entered at the origin (x, y, z) = (0,0,0). Conceptual models of three-dimensional systems are shown below in Figure 14-Figure 16. Figure 14 depicts a three-dimensional visualization of a contaminant and its movement through an aquifer (CLUIN Technologies).

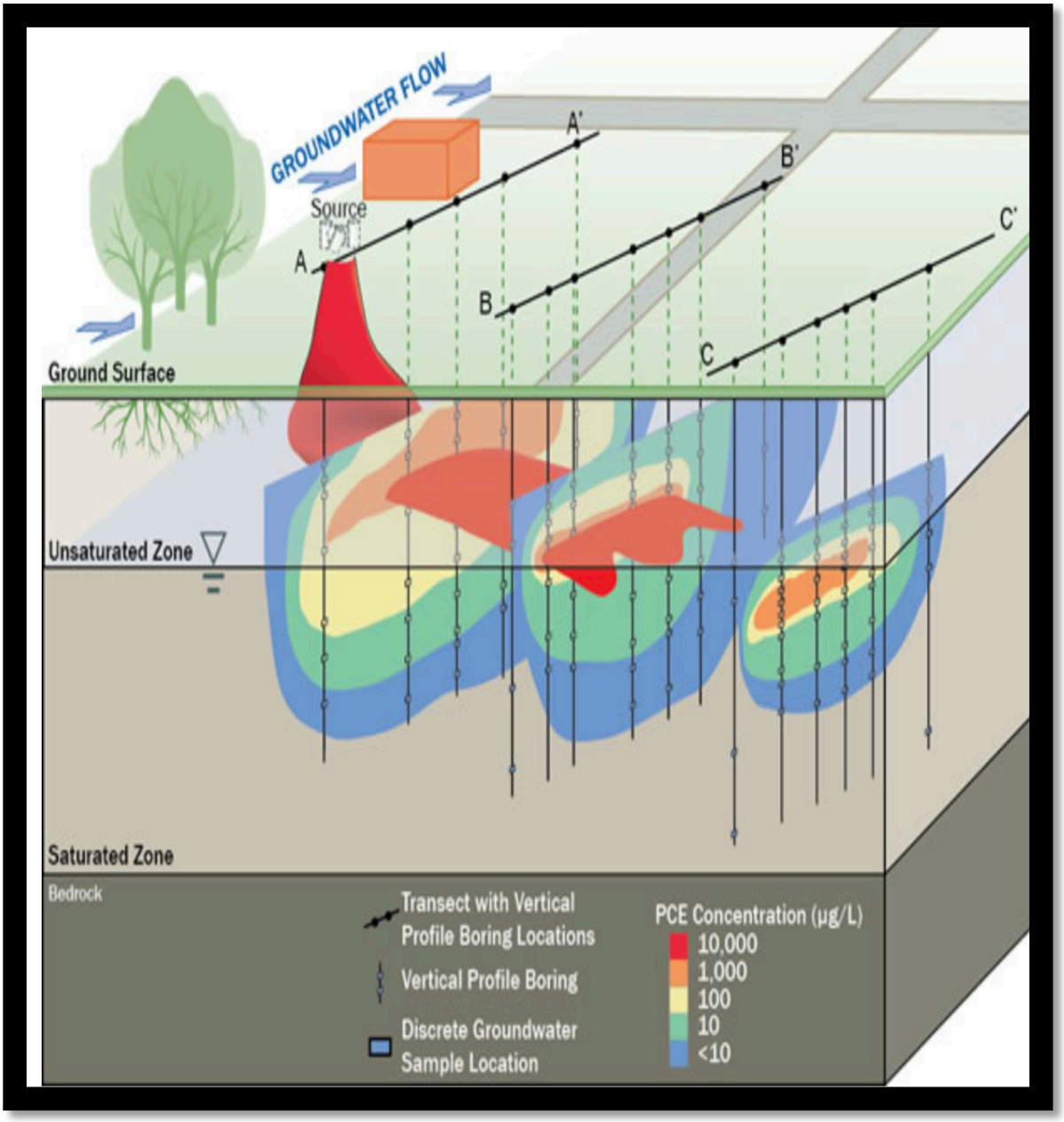


Figure 14: Three-Dimensional Source Type Visualization

Three-Dimensional contamination plume shapes vary between continuous and instantaneous source types, as shown in Figure 15. Instantaneous three-dimensional plumes are characterized by an

elliptical shape, Figure 15-A, and apply to Impulse and Step-Function modeling scenarios. Continuous three-dimensional plumes are asymmetric, broadening in the direction of flow, as shown in Figure 15-B. Continuous contamination events describe Injection and Finite-Length Injection source types (Pitts, 2017)

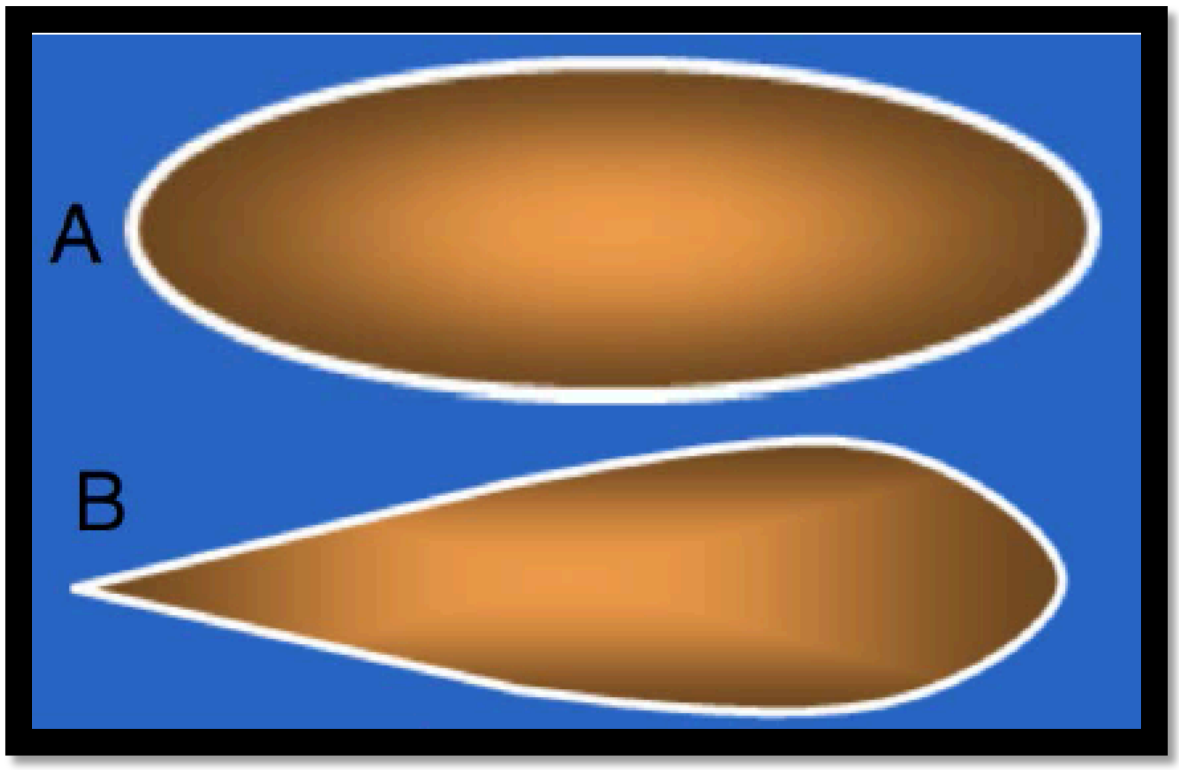


Figure 15: Three-Dimensional Source Type Conceptual Model

Figure 16 below is an additional conceptual model of a three-dimensional source type. The figure is included as further reference of the shape created by a contaminant plume as it moves downstream and expands in the x-. y-, and z-directions. The red

cone indicates a higher concentration than the blue cone due to proximity to the original contamination source and its travel path along the x-axis (Hähnlein et al. 2010).

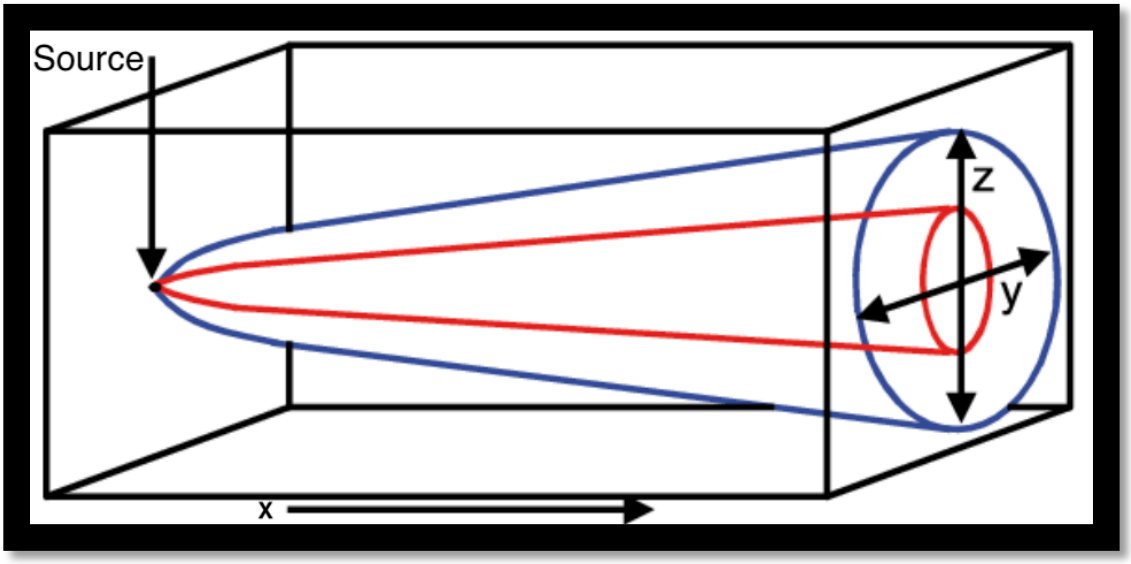


Figure 16: Three-Dimensional Source Type Additional Conceptual Model

Impulse

An impulse consists of a one-time injection of a solute into the aquifer at time = 0. This source type is assumed to be instantaneous.

The analytical solution is shown below in Equation 29 (Yuan, 1995).

$$C(x,y,z,t) = \frac{M\ exp\left[-\frac{(x-\ vt)^2}{4D_x t} - \frac{y^2}{4D_y t} - \frac{z^2}{4D_z t}\right]}{8n\sqrt{\pi^3 t^3 D_x D_y D_z}}$$

Equation 29

Injection

A 3D transport model with a constant injection source located at the origin is assumed to have an area source located at the origin (x, y, z) = (0, 0, 0). The analytical solution is shown below in Equation 30 (Yuan, 1995).

$$C(x,y,z,t) = \left(\frac{\bar{M} exp\left(\frac{xv}{2D_x}\right)}{8\pi nR\sqrt{D_y D_z}}\right)\left[exp\left(-\frac{Rv}{2D_x}\right) erfc\left(\frac{R\ -\ vt}{2\sqrt{D_x t}}\right) +\ exp\left(\frac{Rv}{2D_x}\right) erfc\left(\frac{R\ +\ vt}{2\sqrt{D_x t}}\right)\right]$$

Equation 30

where:

C_0 = Source concentration (mg/L)

x = Distance (meters)

v = Mean seepage velocity (m/s)

t = Time (days)

D_i = Dispersion coefficient in the i-direction (m^2/d)

erfc = Complementary error function

Q = Injection rate (m^3/s)

n = Porosity (dimensionless)

Step-Function

Domenico and Robbins (1985, 1987) and Hunt (1985) published analytical solutions that were based on the product of three-orthogonal one-dimensional solutions. One solution, usually the x-direction, accounts for the advection-dispersion component of the problem, and the other solutions account for spreading from dispersive effects transverse to the mean flow direction (assumed to be the x-direction). The principles of superposition and convolution are used to construct solutions for various source shapes.

The analytical solution is obtained by the spatial integration of the 3D point-impulse model, shown below in Equation 31 (Domenico and Robbins 1985).

$$C(x,y,z,t) = \int_{-\frac{X}{2}}^{\frac{X}{2}}\int_{-\frac{Y}{2}}^{\frac{Y}{2}}\int_{-\frac{Z}{2}}^{\frac{Z}{2}}\left(\frac{M}{2n\sqrt{\pi D_x t}}\right)\left[exp\left(\frac{-(x-vt)^2}{4D_x t}\right)\left(\frac{1}{2\sqrt{\pi D_y t}}\right) exp\left(\frac{-y^2}{4D_y t}\right)\left(\frac{1}{2\sqrt{\pi D_x t}}\right) exp\left(\frac{-z^2}{4D_x t}\right)\right] dzdyx$$

Equation 31

The closed form solution to this integral is

$$C(x,y,z,t) = wC_1(x,t)C_2(y,t)C_3(z,t)$$

where

$$w = \frac{C_0}{8}$$

and

$$C_1(x,t) = \left(erf\left(\frac{x-vt+\frac{X}{2}}{2\sqrt{D_x t}}\right) - erf\left(\frac{x-vt-\frac{X}{2}}{2\sqrt{D_x t}}\right)\right)$$

$$C_2(y,t) = \left(erf\left(\frac{y + \frac{Y}{2}}{2\sqrt{D_y t}}\right) - erf\left(\frac{y - \frac{Y}{2}}{2\sqrt{D_y t}}\right)\right)$$

$$C_3(z,t) = \left(erf\left(\frac{z + \frac{Z}{2}}{2\sqrt{D_z t}}\right) - erf\left(\frac{z - \frac{Z}{2}}{2\sqrt{D_z t}}\right)\right)$$

Equation 32

where:

C_0 = Source concentration (mg/L)

x = Distance (meters)

v = Mean seepage velocity (m/s)

t = Time (days)

D_i = Dispersion coefficient in the i-direction (m^2/d)

erfc = Complementary error function

Q = Injection rate (m^3/s)

n = Porosity (dimensionless)

X = Source dimension in the x-direction (m)

Y = Source dimension in the y-direction (m)

Z = Source dimension in the z-direction (m)

The equations for the closed form solution to this model scenario are the same as the solution for the two-dimensional step function case. However, with the three-dimensional model, the z-dimensional calculations are performed instead of being assigned a value of 1 as in the two-dimensional case.

Finite Length Injection

A three-dimensional finite-length injection has planar source in an aquifer of infinite extent located at (x,y,z) = (0,0,0). The source is defined as Y units wide and Z units high. At t = 0, the concentration at the source suddenly increases to the initial concentration, C_0, and is maintained at that value specified duration, tau, when the source then decreases back to zero. The transport processes modeled are advection along the x-axis and dispersion in the x-, y-, and z-directions.

The analytical solution is shown below in Equation 33 (Yuan, 1995).

$$C\ (x,y,z,t) = \int_{-\frac{Z}{2}}^{\frac{Z}{2}} \int_{-\frac{Y}{2}}^{\frac{Y}{2}} \int_{0}^{t} \frac{C_0 v\ exp\left[-\frac{\left(x - v(t-\tau)\right)^2}{4D_x(t-\tau)} - \frac{(y-y')^2}{4D_y(t-\tau)} - \frac{(z-z')^2}{4D_z(t-\tau)}\right]}{8\sqrt{\pi^3(t-\tau)^3 D_x D_y D_z}} d\tau dy' dz'$$

Equation 33

The closed form solution to the integral is given in Equation 34. (Domenico and Robbins 1985).

$C(x,y,z,t) =$

$$\frac{C_o}{8}\left[erfc\left(\frac{x-vt}{2\sqrt{D_x t}}\right)\right]\left\{erf\left[\frac{\left(y+\frac{Y}{2}\right)}{\sqrt{2D_y\frac{x}{v}}}\right] - erf\left[\frac{\left(y-\frac{Y}{2}\right)}{\sqrt{2D_y\frac{x}{v}}}\right]\right\}\left\{erf\left[\frac{\left(z+\frac{Z}{2}\right)}{\sqrt{2D_z\frac{x}{v}}}\right] - erf\left[\frac{\left(z-\frac{Z}{2}\right)}{\sqrt{2D_z\frac{x}{v}}}\right]\right\}$$

Equation 34

where:

C_0 = Source concentration (mg/L)

x = Distance (meters)

v = Mean seepage velocity (m/s)

t = Time (days)

D_i = Dispersion coefficient in the i-direction (m^2/d)

erfc = Complementary error function

Q = Injection rate (m^3/s)

n = Porosity (dimensionless)

X = Source dimension in the x-direction (m)

Y = Source dimension in the y-direction (m)

Z = Source dimension in the z-direction (m)

Attenuation Type

For each source type, Impulse, Injection, Step-function, and finite length injection, there are four different transport/attenuation combinations, Advection-Dispersion, Advection-Dispersion-Retardation, Advection-Dispersion-Decay, and Advection-Dispersion-Retardation-Decay. Figure 17 below shows a general layout of each attenuation type graphically. Part A shows Advection only (for Reference), Part B shows Advection-

Dispersion, Part C shows Advection-Dispersion-Retardation, and Part D shows Advection-Dispersion-Decay-Retardation (Schirmer and Butler, 2004).

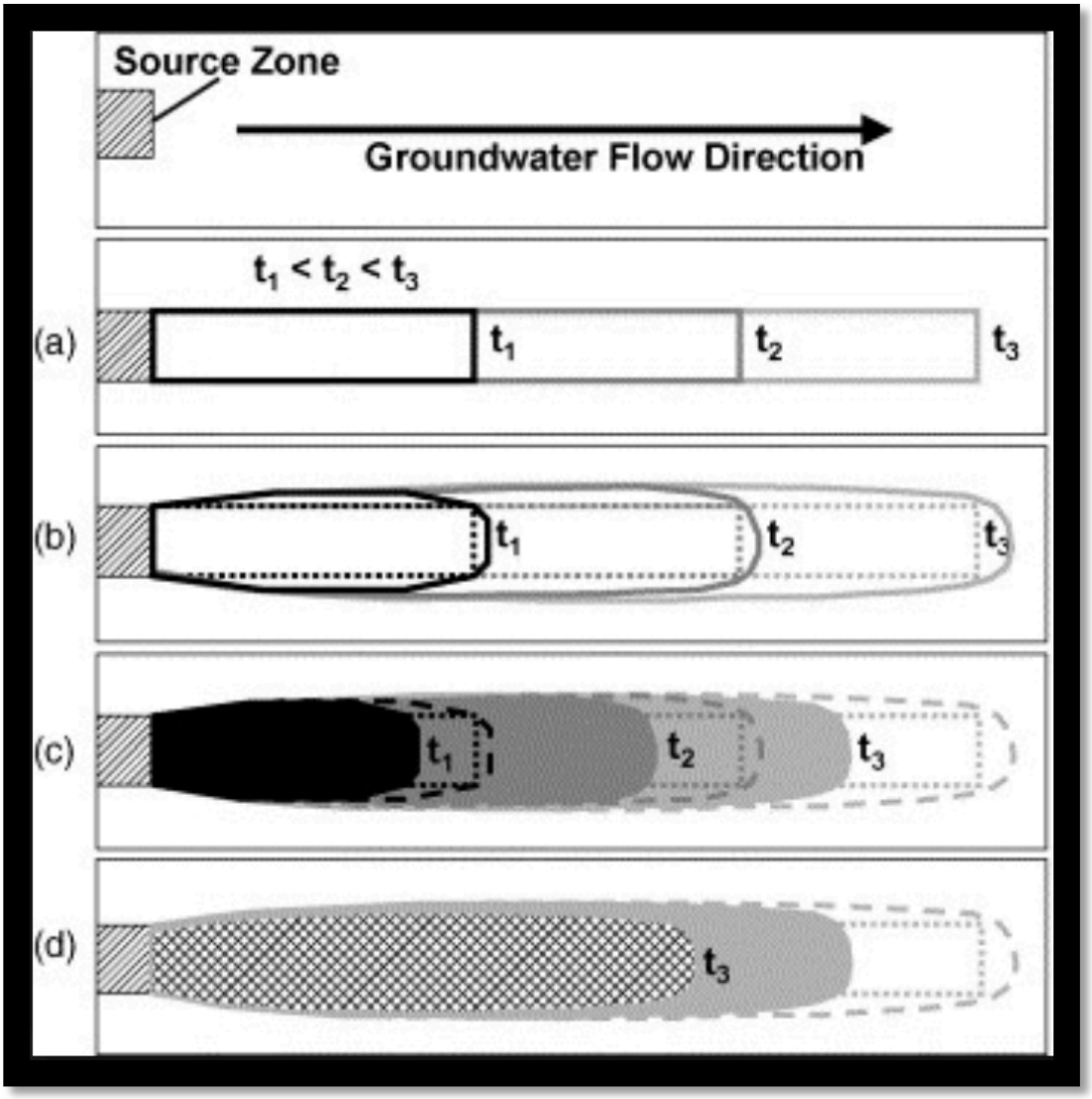


Figure 17: Attenuation Type Conceptual Models

Conservative Tracers

Two transport modeling scenarios are utilized for conservative tracers in NewANTS; Advection-Dispersion and Advection-Dispersion-Retardation. Flow is assumed to be unidirectional, steady, and uniform. Source input rates are negligible compared to regional flow rates so that the input does not affect the hydraulics. Density and viscosity differences between source input and native liquid are negligible. Solutions are functions, C(x, t), subject to various initial and boundary conditions.

Advection-Dispersion

Transport models that are computed using Advection-Dispersion only represent steady-state conditions at long times with the boundary conditions assumed to be constant for all times, t. These models do not go through any reaction and only represent the transport due to advection and dispersion.

Advection-Dispersion-Retardation

Transport models that combine advection, dispersion, and retardation. The contaminant is not reactive with the solid phase of the porous media and the computation includes retardation of the contaminant. Calculations are based upon the local equilibrium assumption (LEA).

Non-Conservative Tracers

Two transport modeling scenarios are utilized for non-conservative tracers in NewANTS; Advection-Dispersion-Decay and Advection-Dispersion-Decay-Retardation. Source input rates are negligible compared to regional flow rates so that the input does not affect the hydraulics. Density and viscosity differences between source input and native liquid are negligible. Furthermore, for non-conservative tracers, the adsorption process is assumed to be governed by instantaneous linear-equilibrium isotherm, and any decay process is assumed to be governed by first order decay. Solutions are functions, C(x, t), subject to various initial and boundary conditions.

Advection-Dispersion-Decay

Transport models that combine advection, dispersion, and decay. Degradation of the contaminant occurs.

Advection-Dispersion-Decay-Retardation

Transport models that combine advection, dispersion, retardation, and decay. The contaminant is reactive with the solid phase of the porous media and the computation includes both retardation of the contaminant and degradation. Calculations are based upon the local equilibrium assumption (LEA).

Model Classes

Table 4 shows the full range of model classes that can be described using NewANTS. Future suggestions include expanding the transport model types that are available for calculations within the tool. An explanation of the naming convention will be in the following section. Figure 18 below shows an example hierarchial tree for model selection. In the example shown, in the user has selected a 1-Dimensional model with an Impulse source type modelling Advection-Dispersion-Retardation in the system.

Table 4: NewANTS Available Model Scenarios

NewANTS																							
3-D												2-D											
FiniteLength				StepFunction				Injection				Impulse				FiniteLength				StepFunction			
A-D-1-R	A-D-1	A-D-R	A-D	A-D-1-R	A-D-1	A-D-R	A-D	A-D-1-R	A-D-1	A-D-R	A-D	A-D-1-R	A-D-1	A-D-R	A-D	A-D-1-R	A-D-1	A-D-R	A-D	A-D-1-R	A-D-1	A-D-R	A-D

Injection				Impulse				1-D											
A-D-1-R	A-D-1	A-D-R	A-D	A-D-1-R	A-D-1	A-D-R	A-D	FiniteLength				StepFunction				Injection			
								A-D-1-R	A-D-1	A-D-R	A-D	A-D-1-R	A-D-1	A-D-R	A-D	A-D-1-R	A-D-1	A-D-R	A-D

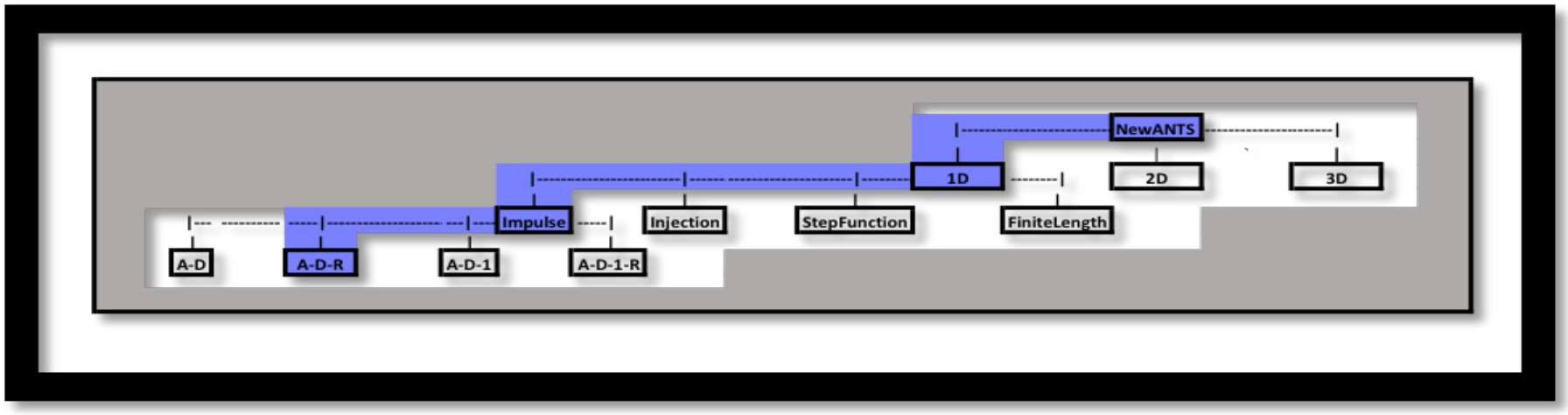


Figure 18: Example Hierarchical Tree

Model Classification Naming Convention

This section explains the naming convention used to identify each type of coding script used in NewANTS and their associated functions. These scripts each represent a part of the NewANTS process and their individual roles are first described conceptually and then elaborated upon in the following section.

HTML/CSS/JAVASCRIPT Concepts

The NewANTS interface will be available on the internet as a tool for students and/or researchers who are in need of a quick analytical solution to contaminant transport simulations. The benefit of having the tool be largely web-based is twofold. The first advantage is that a web page and corresponding server may be accessed remotely, allowing the user to use the system from anywhere, at any time. This feature of a web-based tool makes it much easier for the casual user to take advantage of this free analytical model. Secondly, computation engine updates can be done much more quickly and easily with a web-based approach. The current system relies only on one single server for the database; therefore, any updates would only have to be done on the

server. The interface can be further developed and refined over time without requiring a complete redesign or requiring updates on each user's individual machine.

NewANTS was developed using a hybrid method combining elements from HTML, JavaScript, Python, and R Scripts. The HTML script, hereafter known as "Entry", is used to introduce the model and collect data entry from the user. Next, the Python script, referred to as the "Launcher" harvests the input data and prepares it for the next step, the "Computation Engine". The Computation Engine is an R script that takes the inputs, performs the calculations, and outputs the raw data. Then another Python script, the "Responder", compiles the output, structures the results, and creates the graphical and tabular output. Finally, the "Results" are presented to the end user using an HTML script. A flow chart depicting the interaction between these tools is shown below in Figure 19. More detail on the purpose and concepts of each script type can be found in the following section.

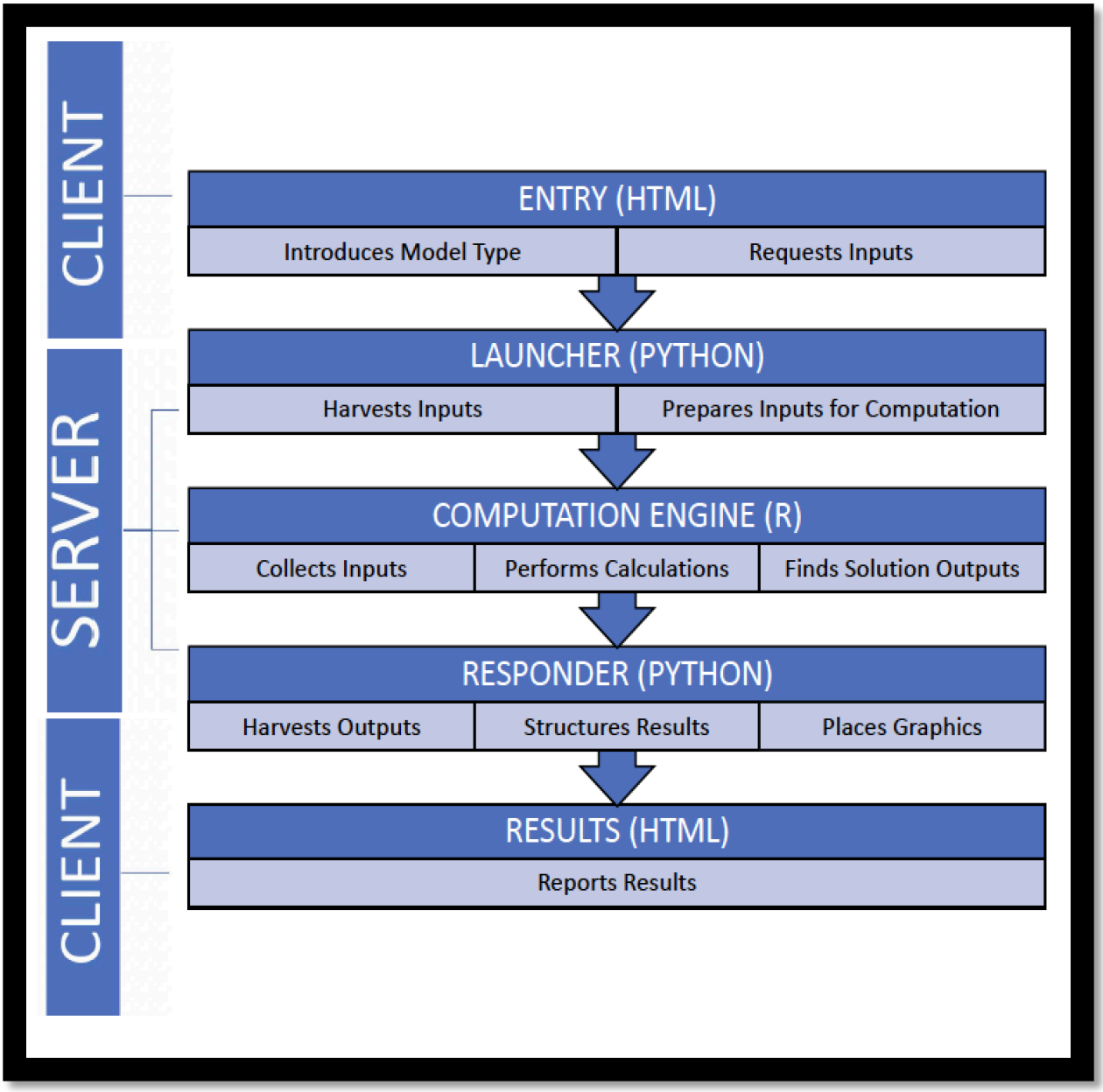


Figure 19: NewANTS Inter-Process Communication Flow Chart

HyperText Markup Language

HyperText Markup Language (HTML) is the most commonly used platform for designing web pages. The NewANTS system uses web-pages designed in HTML as a way to communicate with the user. The HTML requires the user to input specific information, and then also reports the output to the user after the computation engine finds the results. The HTML scripts' major function is taking plain text and formatting it in a way that is easy for the user to interpret and understand. These commands tell the script to present the text in a certain format and also communicate the results back to the user once a solution has been found. Two major benefits of HTML scripts follow.

1. An HTML script typically can run regardless of what platform the user is on. In other words, the interface created using HTML will look and perform similarly on either a Mac, PC, or Linux. This reduces the need to have different working interfaces that are platform specific making the tool easier to understand and use.
2. HTML scripts are stored as plain text (ASCII) files and therefore, the developers of NewANTS can access, edit, and

modify the tool without the need for anything other than a simple text editor.

CSS

A Cascading Style Sheet (CSS) is a language that is used to describe the presentation of a web page that is written in HTML. Its primary function is to set the style of a webpage by specifying layout, colors, fonts, margins, spacing, and other stylistic elements. It is an important aspect of web design because it allows for the separation of the content of a page and its presentation to the user. CSS allows for quick and easy editing of website layout without the need to edit the main HTML script. CSS also can be used to present different HTML pages using identical formatting.

CGI

CGI, or Common Gateway Interface, allows HTML scripts to call in programs written in other coding languages. The CGI is connected to an external operating system and allows the interface to call in external data and computational engines and get back results on the HTML page.

CGI programs are called in when the user selects an option on the HTML page to begin the analysis. Once the web browser gets permission from the web server to run the CGI program (and the user's credentials have been verified), the CGI program is then found and executed. The output produced by the CGI program is then sent back to the HTML interface to display the results to the user. Besides the obvious benefit of acting as a connection between HTML (users) and an external server (database), CGI programs also can be used to manipulate a server's database from any user's computer. The end user has the advantage of being able to manipulate the database in any way they want when using NewANTS.

JavaScript

JavaScript is a programming language that allows World Wide Web pages to contain code that is automatically executed on the user's web browser. It is a high-level, general purpose, dynamic, object-based programming language specifically designed to have as few implementation dependencies as possible.

The major advantage of JavaScript over other similar programming tools is that JavaScript is platform independent. In other words, it allows a single application to run on multiple platforms. Using JavaScript, programmers can develop a script that will run anywhere on the internet regardless of the user's computer platform. In addition, unlike CGI, JavaScript runs on the user's machine.

Inter-Process Methodology

This section will describe each of the four major parts of the underlying server codes for NewANTS. Relevant code fragments from the scripts are included as figures for reference, and the full scripts are included as Appendices D-F.

Entry

The first step in the NewANTS system is the HTML Input Interface, referred to as the Entry. The user connects to the NewANTS system via a web-browser. The Entry script prompts the user for the required inputs for the selected model scenario. Figure 20 below is a screenshot of the relevant portions of the

Entry script that specify what inputs are needed for the computation. For example, line 19 in Figure 20 specifies that the Entry web page will request that the user "Enter Value for Distance (x)".

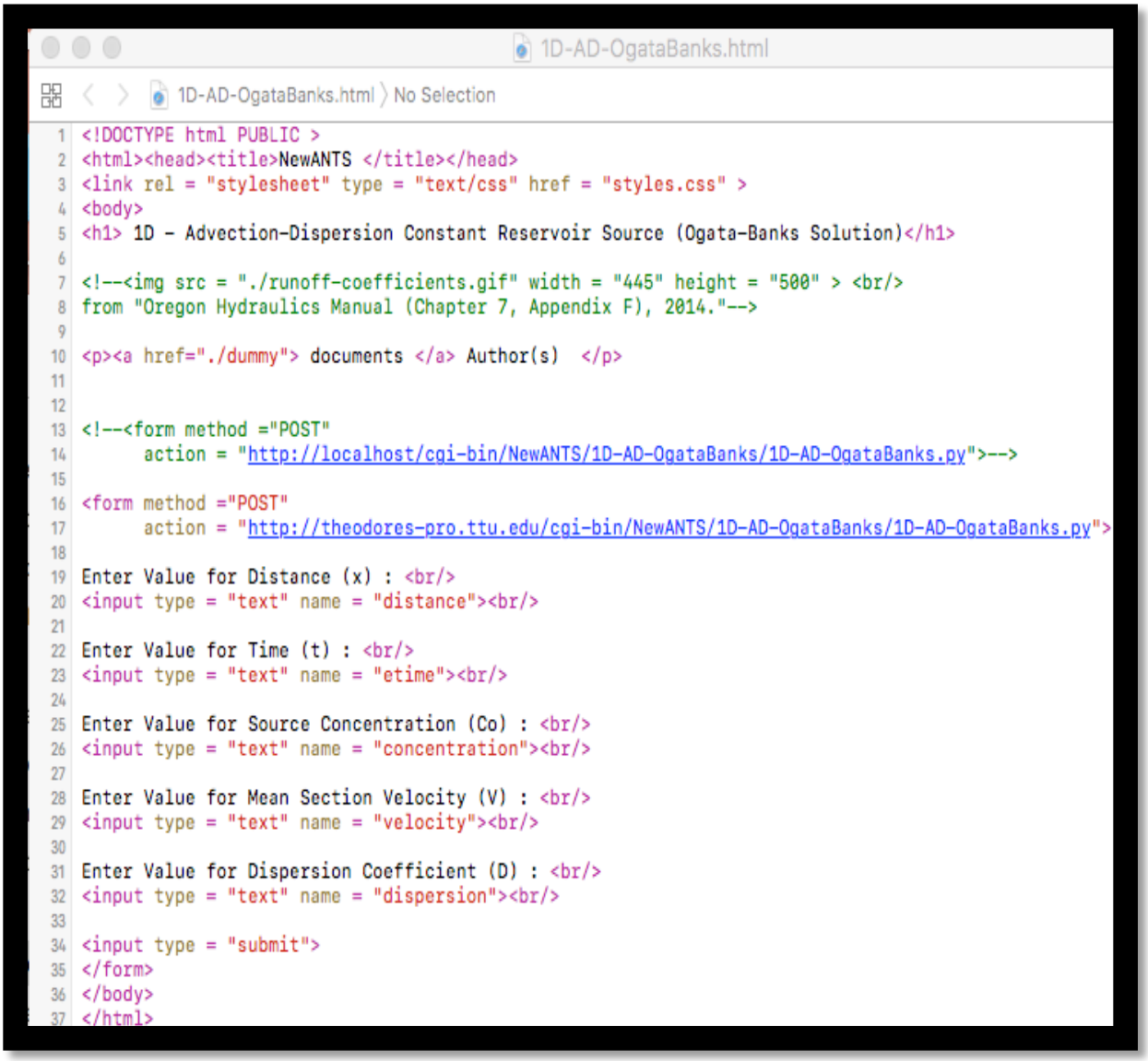
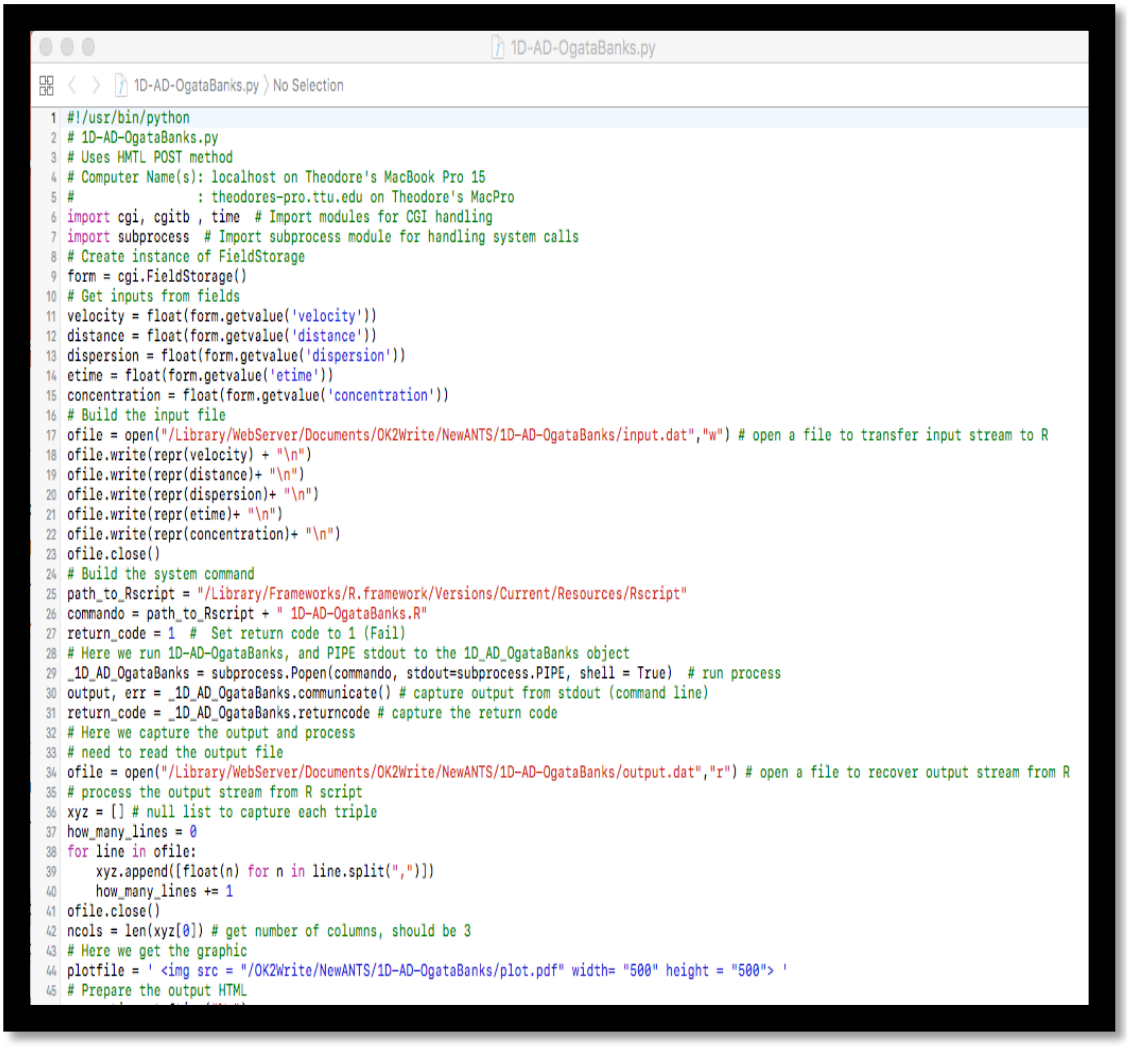


Figure 20: NewANTS Interface Entry (HTML Script)

Launcher

Next, a Python script, the Launcher, is called to read the data from the Entry and communicate that information to the computation engine. The Launcher script also contains code to execute the Computation Engine script (the next step in the process) using the data from the Entry. This group of commands can be found starting on line 25 in Figure 21 below, which also shows the additional relevant portions of the Launcher script. This step makes up the first half of the Python script's responsibilities within NewANTS.



```
#!/usr/bin/python
# 1D-AD-OgataBanks.py
# Uses HMTL POST method
# Computer Name(s): localhost on Theodore's MacBook Pro 15
#                 : theodores-pro.ttu.edu on Theodore's MacPro
import cgi, cgitb , time  # Import modules for CGI handling
import subprocess  # Import subprocess module for handling system calls
# Create instance of FieldStorage
form = cgi.FieldStorage()
# Get inputs from fields
velocity = float(form.getvalue('velocity'))
distance = float(form.getvalue('distance'))
dispersion = float(form.getvalue('dispersion'))
etime = float(form.getvalue('etime'))
concentration = float(form.getvalue('concentration'))
# Build the input file
ofile = open("/Library/WebServer/Documents/OK2Write/NewANTS/1D-AD-OgataBanks/input.dat","w") # open a file to transfer input stream to R
ofile.write(repr(velocity) + "\n")
ofile.write(repr(distance)+ "\n")
ofile.write(repr(dispersion)+ "\n")
ofile.write(repr(etime)+ "\n")
ofile.write(repr(concentration)+ "\n")
ofile.close()
# Build the system command
path_to_Rscript = "/Library/Frameworks/R.framework/Versions/Current/Resources/Rscript"
commando = path_to_Rscript + " 1D-AD-OgataBanks.R"
return_code = 1  #  Set return code to 1 (Fail)
# Here we run 1D-AD-OgataBanks, and PIPE stdout to the 1D_AD_OgataBanks object
_1D_AD_OgataBanks = subprocess.Popen(commando, stdout=subprocess.PIPE, shell = True)  # run process
output, err = _1D_AD_OgataBanks.communicate() # capture output from stdout (command line)
return_code = _1D_AD_OgataBanks.returncode # capture the return code
# Here we capture the output and process
# need to read the output file
ofile = open("/Library/WebServer/Documents/OK2Write/NewANTS/1D-AD-OgataBanks/output.dat","r") # open a file to recover output stream from R
# process the output stream from R script
xyz = [] # null list to capture each triple
how_many_lines = 0
for line in ofile:
    xyz.append([float(n) for n in line.split(",")])
    how_many_lines += 1
ofile.close()
ncols = len(xyz[0]) # get number of columns, should be 3
# Here we get the graphic
plotfile = ' <img src = "/OK2Write/NewANTS/1D-AD-OgataBanks/plot.pdf" width= "500" height = "500"> '
# Prepare the output HTML
```

Figure 21: NewANTS Interface Launcher (Python Script)

Computation Engine

The Computation Engine is coded using R. The Computation Engine script receives model inputs from the Launcher and begins

the process of running the contaminant model simulation. In addition, the Computation Engine code generates the graphical elements of the results and saves the output to a file. Figure 22 shows the Computation Engine script in NewANTS, with line 36 beginning the group of commands that runs the Computation Engine.

```r
# analytical model advection-dispersion for CGI-BIN
rm(list=ls())
####### Disable for CGI-BIN ########
#this.dir <- dirname(parent.frame(2)$ofile)
#setwd(this.dir)
#########################################
# prototype special functions in generic R #
#########################################
# error function (real valued input/output)
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
# complimentary error function (real values input/output)
erfc <- function(x) 2 * pnorm(x * sqrt(2), lower = FALSE)
#############################################################################
# ogata-banks solution for 1D constant reservoir, advection-dispersion case #
#############################################################################
c1dad <- function(distance,time,dispersion,velocity,concentration){
# bear, dynamics of fluids in porous media, pg 630 eqn 10.6.22
# verified theodore g. cleveland 2017-0823
  arg1 <- (distance-velocity*time)/(2*sqrt(dispersion*time));
  arg2 <- (distance*velocity)/dispersion;
  arg3 <- (distance+velocity*time)/(2*sqrt(dispersion*time));
  if (arg3 > 27) {
      c1dad <- 0.5*concentration*(erfc(arg1)                    );
  }
  else {
      c1dad <- 0.5*concentration*(erfc(arg1)+exp(arg2)*erfc(arg3));
  }
return(c1dad) ;
}
# read from an input file
####### Disable for CGI-BIN #########
#zz <- file("./input.dat","r")
#####################################

####### Enable for CGI-BIN ##########
zz <- file("/Library/WebServer/Documents/OK2Write/NewANTS/1D-AD-OgataBanks/input.dat","r") # Open a connection to zz
#####################################
velocity <- as.numeric(readLines(zz, n=1, ok=TRUE, warn=TRUE,encoding="unknown",skipNul = FALSE))
distance <- as.numeric(readLines(zz, n=1, ok=TRUE, warn=TRUE,encoding="unknown",skipNul = FALSE))
dispersion <- as.numeric(readLines(zz, n=1, ok=TRUE, warn=TRUE,encoding="unknown",skipNul = FALSE))
time <- as.numeric(readLines(zz, n=1, ok=TRUE, warn=TRUE,encoding="unknown",skipNul = FALSE))
concentration <- as.numeric(readLines(zz, n=1, ok=TRUE, warn=TRUE,encoding="unknown",skipNul = FALSE))
close(zz)
######## To Do Error Traps ##########
# Input verify in HTML/Javascript   #
#####################################
# function
output <- c1dad(distance,time,dispersion,velocity,concentration)
cat(output)
####### profile output ##########
x <- numeric(0)
conc <- numeric(0)
xlow <- distance/10
xhigh <- distance*10
step <- ((xhigh-xlow)/100)
### profile vector output
outfile <- file("/Library/WebServer/Documents/OK2Write/NewANTS/1D-AD-OgataBanks/output.dat","w")
for (i in 1:199) {
   x[i] <- 0+i;
   conc[i] <- c1dad(x[i],time,dispersion,velocity,concentration);
   write(c(x[i],time,conc[i]), outfile ,sep = ",",ncolumns = 3)  #write to stdio
}
close(outfile)
##build chart
pdf("/Library/WebServer/Documents/OK2Write/NewANTS/1D-AD-OgataBanks/plot.pdf")
title_string <- paste("Concentration Profile at Time = ",time)
plot(x,conc,type="l",main=title_string,xlab="Distance",ylab="Concentration")

```

Figure 22: NewANTS Interface Computation Engine (R Script)

Responder/Results

Once the graphics element is saved by the Computation Engine, the second half of the Python script, the Responder, begins rendering the information for the Results module (the next and final step). The Responder receives the output from the Computation Engine and transfers it to the Results script, coded in HTML, in the form of both numerical and graphical elements. The Results script then transmits the information to the user in the form of a web page. Figure 23 is a screen shot of the Responder script, and the commands to transmit and format the output information begin on line 46.

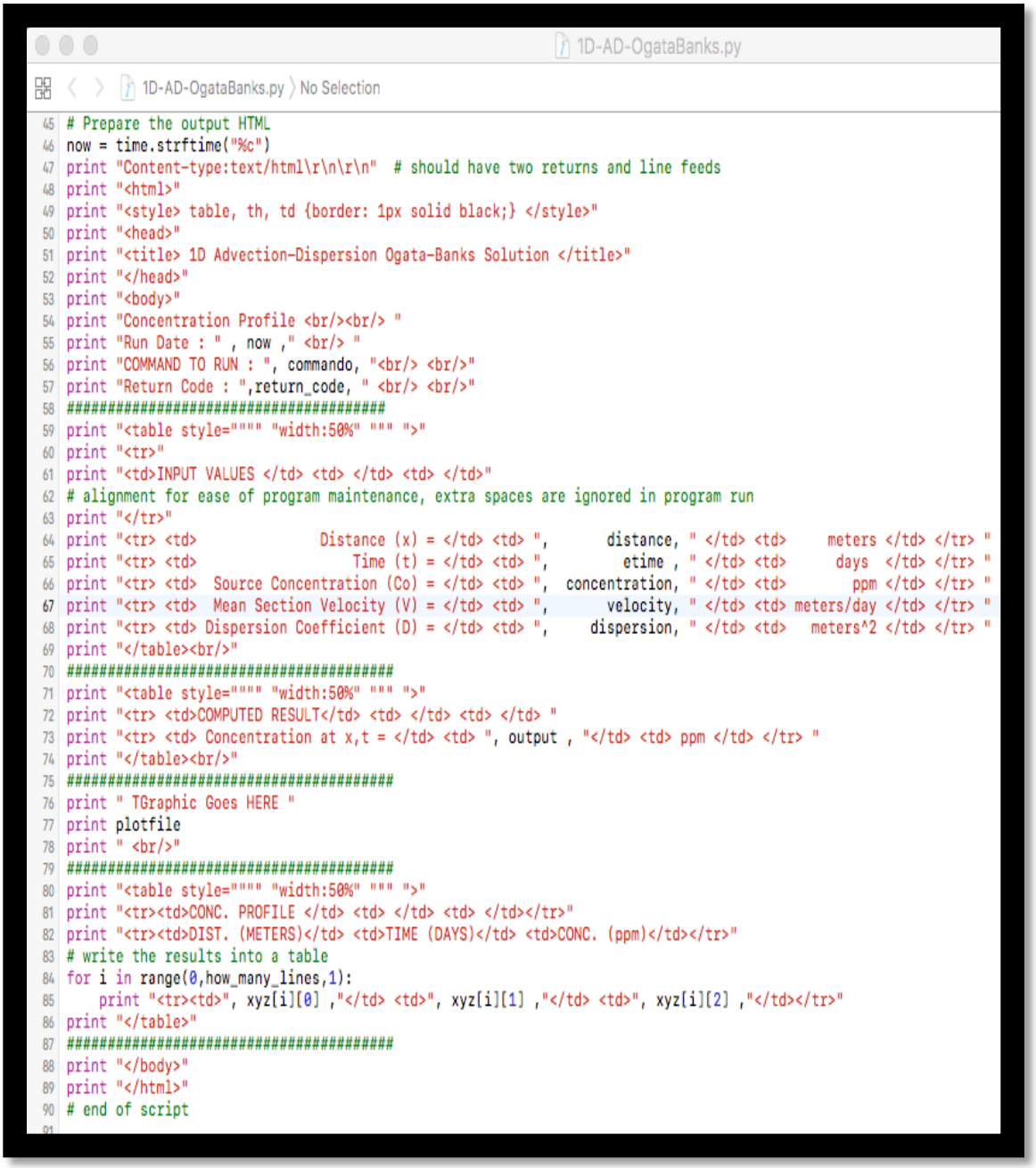


Figure 23: NewANTS Interface Responder (Python Script)

The Interface

The following section describes the NewANTS interface and its layout, required inputs, and sample outputs. The examples and figures in this section are of a One-Dimensional, Impulse contamination source type, modeled using Advection-Dispersion as the transport model. The HTML, JavaScript, CSS, CGI, Python, and R scripts that are used in this section can be found in Appendices D-F. This section defines the required input for NewANTS and shows screen captures of the interface and its output.

General Interface Layout

Home Page

The NewANTS homepage is shown below in Figure 24. It gives an overview of the history of NewANTS, briefly explains what computational processes are used within the interface, and describes the layout of the site.

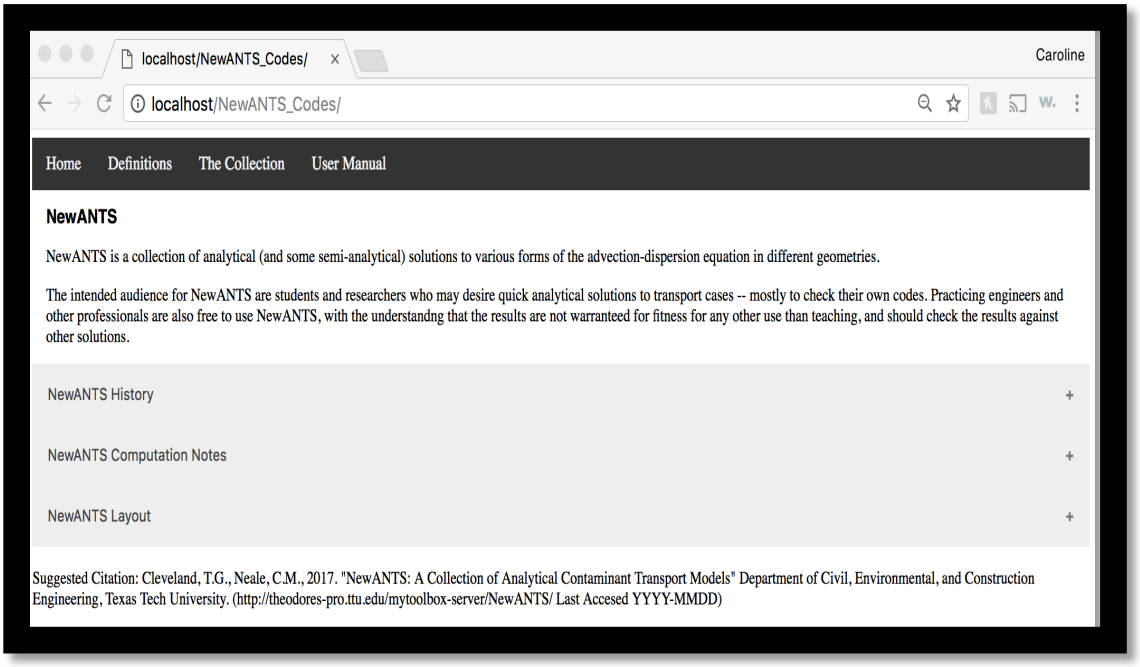


Figure 24: NewANTS Interface Homepage (HTML)

Definitions

The next part of the NewANTS interface is the Definitions tab. This section defines hydraulic and transport parameters necessary simulation inputs and defines what a concentration profile depicts so that the user is aware of what information the model simulations will export.

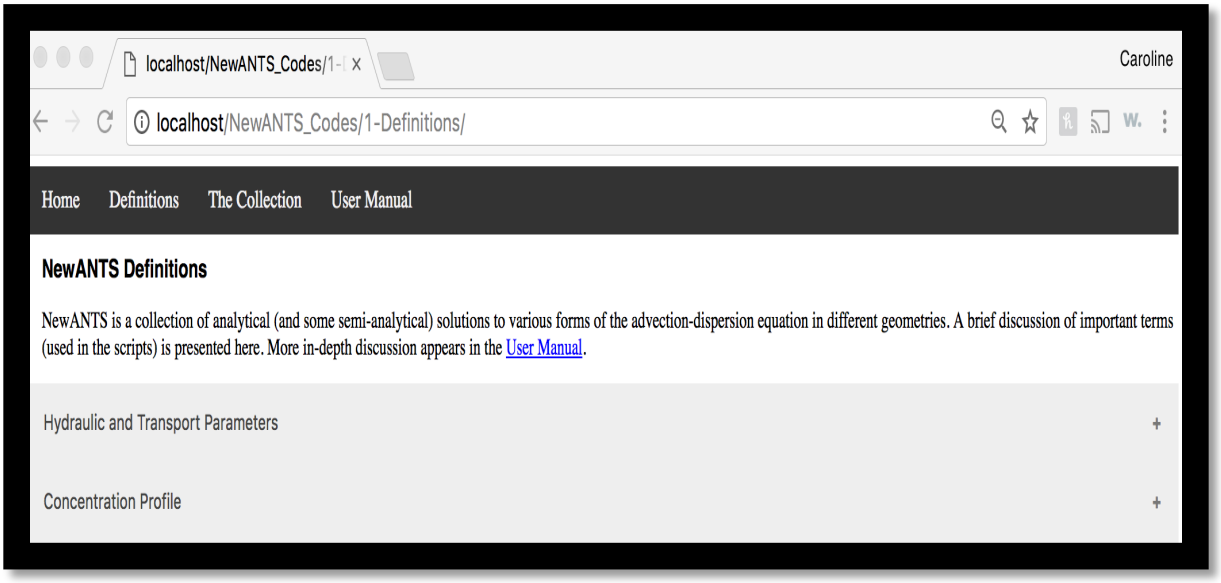


Figure 25: NewANTS Interface Definitions Page (HTML)

The Collection

The compilation of transport models NewANTS can simulate is found under the Collection tab. The user selects either one-, two-, or three- dimensional models from the menu which will redirect them to a page showing the four Source Types. Once the user selects a source type, the final step is to select a Transport method which will open up the web interface for the specified modeling parameters. Figure 26 and Figure 27 show screen captures of the

interface menus that prompt the user to select their modeling conditions.

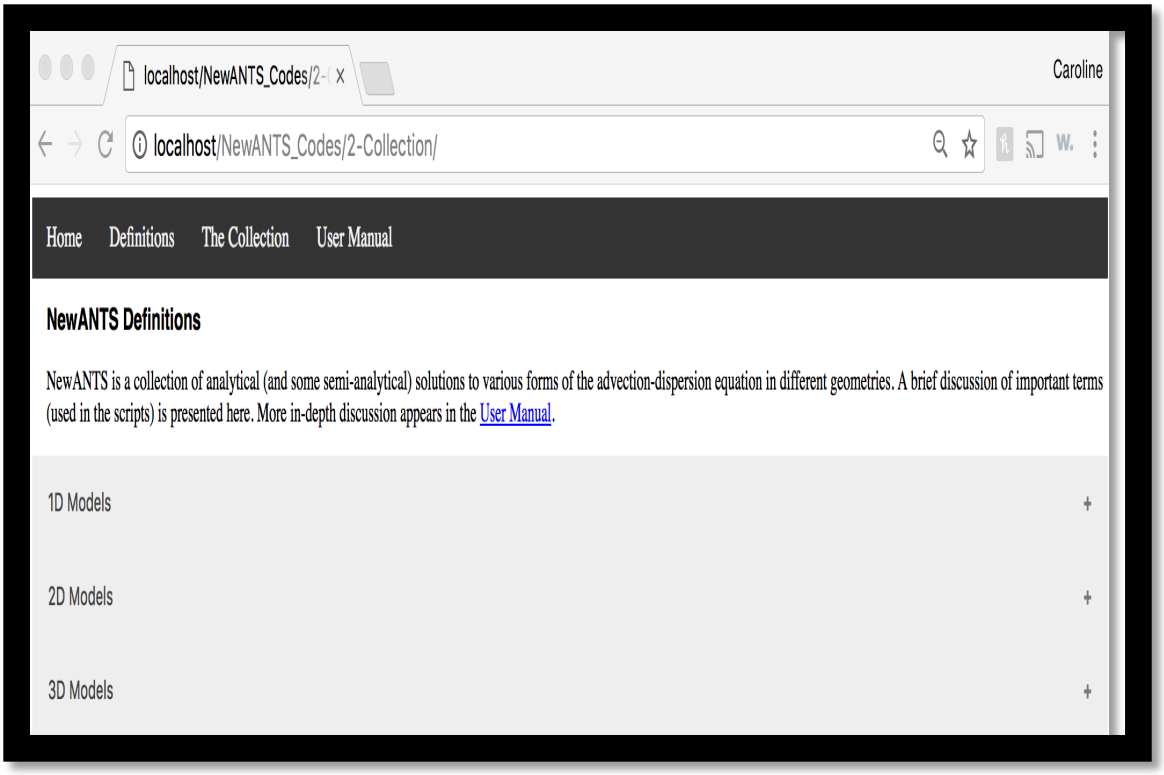


Figure 26: NewANTS Interface Collections Page (HTML)

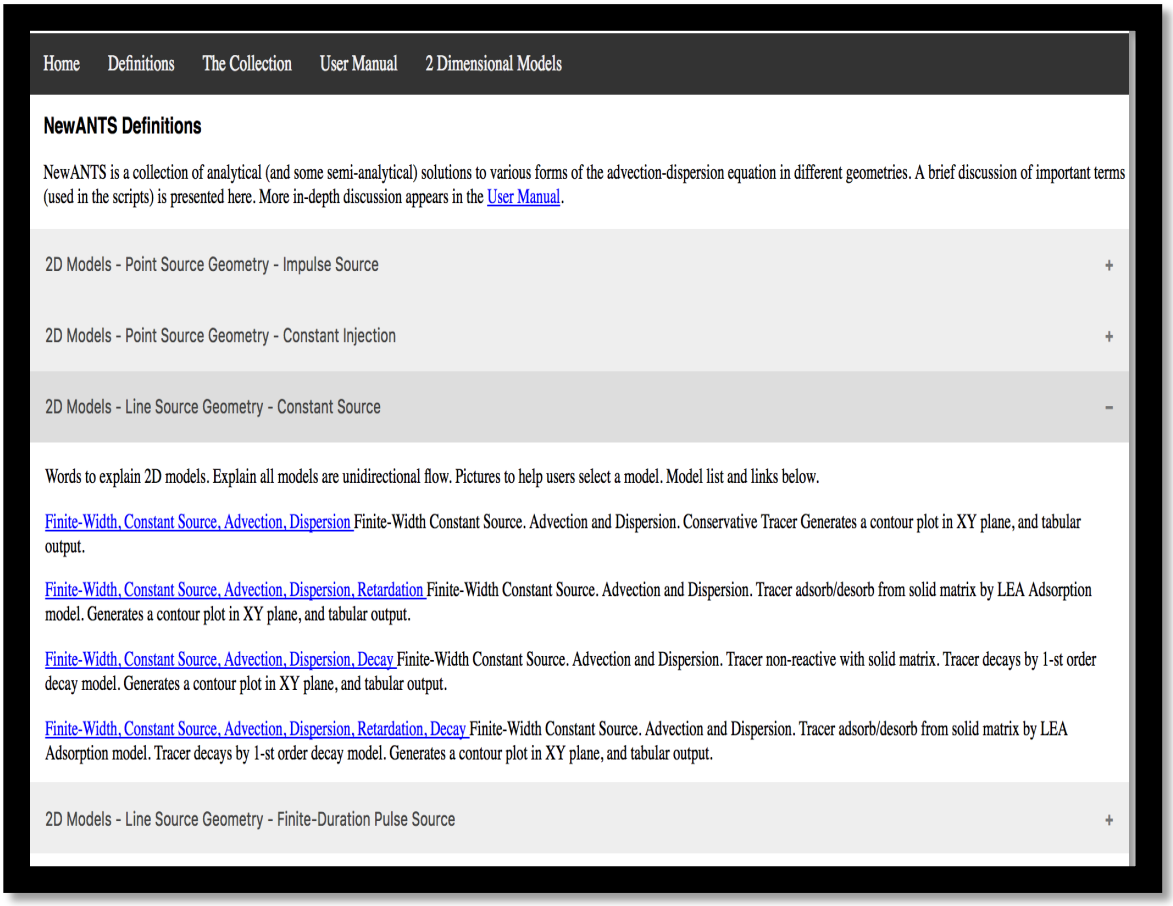


Figure 27: NewANTS Interface Expanded Collections Page (HTML)

Hydraulic and Transport Input Parameters

NewANTs employs different transport model types that require inputs that correspond to specific hydraulic and transport parameters for the simulation in question. A discussion of the

important terms used in the scripts is presented below. The various parameters required are listed in this section.

Source Location: The location of the contaminant source coordinates. The default source location (if not specifically requested in the input interface) is located at (x,y,z,t) = (0,0,0,0)

Distance: The location of interest from the contaminant source. The default distance from source (if not specifically requested in the input interface) is located along the x-axis in the units specified on the input parameter webpage. In 2-Dimensional and 3-Dimensional cases, the specific axis will be indicated.

Time: The elapsed time from t = 0 in units specified on the input parameter webpage.

Velocity: The mean seepage velocity of the flow field. For modeling a groundwater system, the value of this parameter is equal to the ratio of specific discharge and porosity. The velocity should be determined by the user prior to interrogating the models.

Dispersion Coefficient: The dispersion coefficient along the flow axis. For 2-Dimensional and 3-Dimensional cases, the specific axis (x, y, and/or z), will be indicated.

Dispersivity: The dispersivity along the flow axis. For 2-Dimensional and 3-Dimensional cases, the specific axis (x, y, and/or z), will be indicated.

Retardation Coefficient: The species retardation coefficient based on the linear equilibrium assumption (LEA) for approximating species interaction with the solids matrix.

Source Decay Coefficient: The decay rate of mass at the location of the contaminant source.

1-st Order Species Reaction (Decay) Coefficient: The First-Order reaction rate of mass of the constituent in the flow field (after it has left the source).

Injection Rate: The mass (volume) flow rate of the injection source liquid.

Interface Example

Once the user selects modeling parameters, the interface will open up a webpage where the user can input the required information for

the simulation to run. A screen capture of the interface is shown below in Figure 28.

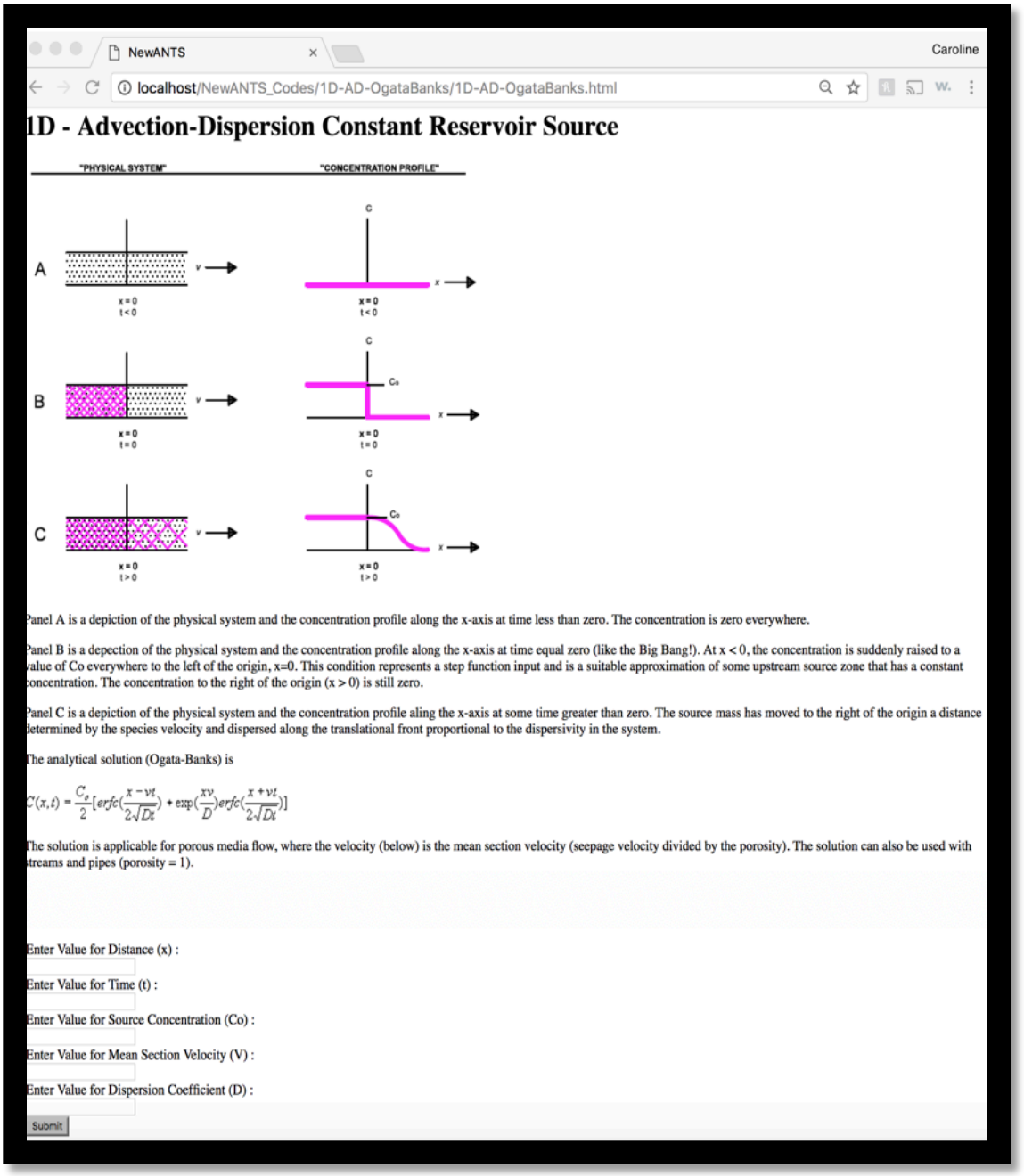


Figure 28: NewANTS Interface Example Model Scenario Page (HTML)

On the page corresponding to the users modeling type selection, there is a brief explanation of the theory behind the model (which is also a QA/QC measure to ensure the user has selected the correct transport model) followed by the data entry area where the user supplies the required information for the model. Once the users press "submit", they are redirected a page that displays the calculated information using the procedure explained previously. The Entry, Launcher, Computation Engine, Responder, and Results scripts for this example can be found in Appendices D, E, and F.

Output

Once the contaminant transport analysis is complete, the results are returned to the user and displayed via the web interface. The solution is described in three different ways: a concentration profile plot, a table of the calculated concentration at equal intervals along the model distance, and a summary table of the final solution.

The decision to supply the user with both a solution and a concentration profile/corresponding data table was made in an

attempt to provide the most detailed and informative transport solution possible. Instead of simply computing and returning a solution, NewANTS attempts to go one step further by supplying the users with the data necessary for customization of their transport model solutions. The table can also be used to find an exact solution for the concentration at any distance along the length to minimize potential errors that could occur from the use of a chart to estimate intermediate concentration values.

For example, if only the concentration plot was supplied, as shown in Figure 30, and the users wanted to estimate what the concentration would be at XX m, they would have to estimate the concentration using the curve. Graphical estimation could lead to inaccurate approximations of contaminant concentration which, depending on the contaminant, could have major implications and consequences.

Example Application

A landfill is leaking leachate with chloride concentration 725 mg/L that enters an aquifer with the following material properties.

K = 3 E-05 m/sec Gradient = 0.002 Porosity (n) = 0.23 Diffusivity = 1e-9 m^2/sec

Estimate the chloride concentration after 1 year at a distance of 15 meters from the release point (Fetter, 2014).

1. Calculate the Average Linear Velocity:

$$U = \frac{K}{n} * Gradient$$

$$U = \left(\frac{3E-05m}{s}\right)\left(\frac{86400s}{day}\right)\left(\frac{1}{0.23}\right)(0.002)$$

$$U = .02254\, m/day$$

2. Calculate Dispersion Coefficient:

$$a = 0.0175L^{1.46}$$

$$a = 0.0175(15)^{1.46}$$

$$a = 0.91m$$

$$D = aU$$

$$D = (0.91m)(\frac{0.02254m}{day})$$

$$D = .0205106\, m^2/day$$

The data calculated is input to NewANTS and the output solution screenshots are below.

Output Plot

NewANTS creates an output plot called a concentration profile to model the contaminant transport. A concentration profile is a plot of concentration versus position at some user defined point in time. The plots show C(x,t) where t (elapsed time) is fixed and x (distance from source location) varies.

Figure 29 below shows the x – t plane, with the vertical axis representing C (x,t) or concentration. At the source, the concentration will represent some fixed, user-supplied value. The axis labeled "t" is the time axis. The axis labeled "x" is the space

(distance) axis. The fixed value of t* is indicated along the time axis. This is the location of the concentration profile plot that represents contaminant levels at increasing distances. This plot is shown below in blue.

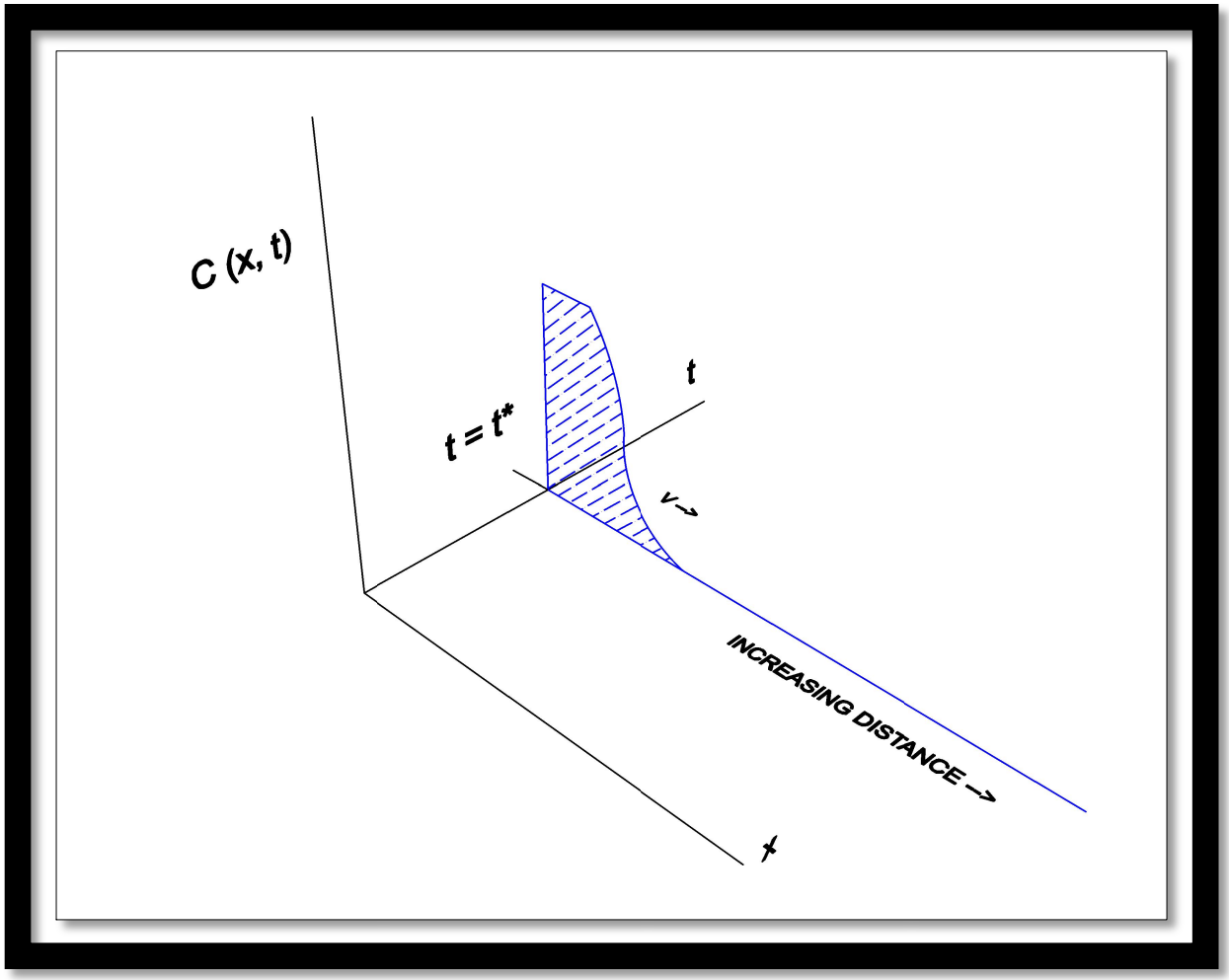


Figure 29: Concentration Profile Conceptual Drawing

A screen capture showing an example concentration profile is shown below in Figure 30. The concentration profile shows the change in concentration vs. distance.

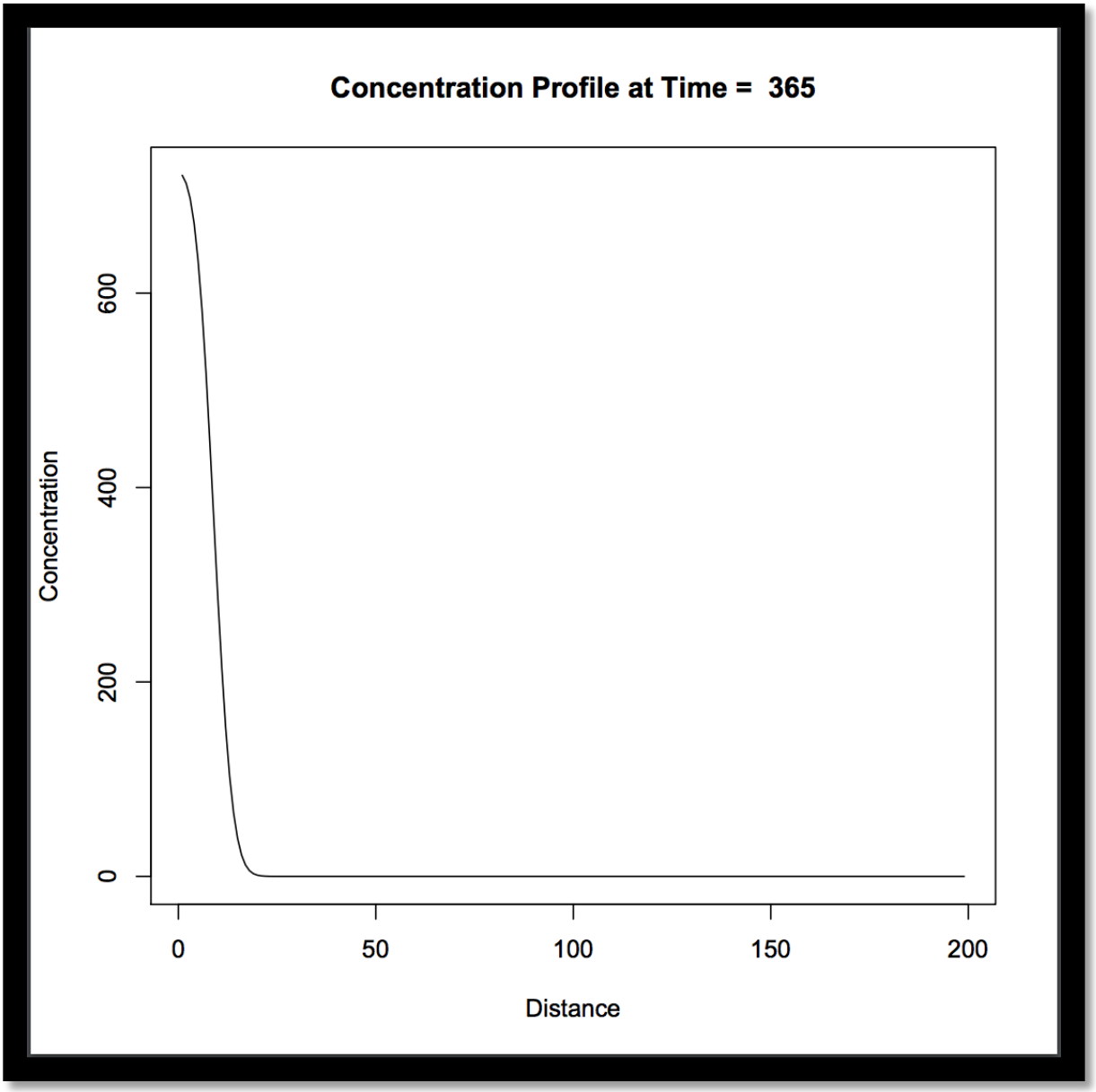


Figure 30: NewANTS Output Concentration Plot

Output Table

In addition to the concentration profile, NewANTS is designed to export a table of computed values of concentration at regular intervals along the distance supplied by the user. The output table also includes a column making note of the fixed point in time that the user has chosen for the computation. This feature of NewANTS gives the user the freedom to take the calculated data and use it in the graphics program of their choice, allowing the user infinite freedom of choice on how to display their concentration profile.

Table 5 is a screen capture of the output table with data for distance (m), time(days), and concentration (ppm).

Table 5: NewANTS Example Output Table

CONC. PROFILE		
DIST. (METERS)	TIME (DAYS)	CONC. (ppm)
1	365	721.1977
2	365	712.8375
3	365	697.3212
4	365	671.844
5	365	634.0397
6	365	582.765
7	365	518.7645
8	365	444.9304
9	365	365.9732
10	365	287.544
11	365	215.074
12	365	152.713
13	365	102.6968
14	365	65.283
15	365	39.16729
16	365	22.14966
17	365	11.79422
18	365	5.908104
19	365	2.78219
20	365	1.230892
21	365	0.5113579
22	365	0.1993947
23	365	0.07295005
24	365	0.02503366
25	365	0.008055491
26	365	0.002430116
27	365	0.000687133
28	365	0.000182077
29	365	4.52E-05
30	365	1.05E-05
31	365	2.29E-06
32	365	4.68E-07
33	365	8.94E-08
34	365	1.60E-08
35	365	2.68E-09
36	365	4.21E-10
37	365	6.18E-11
38	365	8.51E-12
39	365	1.10E-12
40	365	1.32E-13

CONC. PROFILE		
DIST. (METERS)	TIME (DAYS)	CONC. (ppm)
41	365	1.49E-14
42	365	1.57E-15
43	365	1.56E-16
44	365	1.44E-17
45	365	1.25E-18
46	365	1.01E-19
47	365	7.68E-21
48	365	5.45E-22
49	365	3.62E-23
50	365	2.25E-24
51	365	1.31E-25
52	365	7.14E-27
53	365	3.64E-28
54	365	1.74E-29
55	365	7.74E-31
56	365	3.23E-32
57	365	1.26E-33
58	365	4.62E-35
59	365	1.58E-36
60	365	5.06E-38
61	365	1.51E-39
62	365	4.25E-41
63	365	1.11E-42
64	365	2.73E-44
65	365	6.27E-46
66	365	1.35E-47
67	365	2.70E-49
68	365	5.09E-51
69	365	8.95E-53
70	365	1.47E-54
71	365	2.27E-56
72	365	3.26E-58
73	365	4.40E-60
74	365	5.55E-62
75	365	6.55E-64
76	365	7.22E-66
77	365	7.46E-68
78	365	7.21E-70
79	365	6.51E-72
80	365	5.51E-74

Output Solution

Finally, NewANTS exports a summary table that depicts both the user-supplied values for the required input parameters and the final

solution of concentration at given distance and time. Figure 31 shows an example output summary table.

Concentration Profile

Run Date : Fri Mar 2 15:01:29 2018
COMMAND TO RUN : /Library/Frameworks/R.framework/Versions/Current/Resources/Rscript 1D-AD-OgataBanks.R

Return Code : 0

INPUT VALUES		
Distance (x) =	15.0	meters
Time (t) =	365.0	days
Source Concentration (Co) =	725.0	ppm
Mean Section Velocity (V) =	0.02254	meters/day
Dispersion Coefficient (D) =	0.0205105	meters^2

COMPUTED RESULT		
Concentration at x,t =	39.16729	ppm

Figure 31: NewANTS Example Output Summary

Results Comparison

The results from NewANTS find that there will be a concentration of 39.16 ppm 15 meters from the source at time = 365 days. The

results from the solutions given in the textbook find that there will be a concentration of 30 ppm 15 meters from the source at time = 365 days. While the solutions are not exactly identical, the NewANTS tool is shown to be capable of estimating the concentration with some degree of accuracy.

Conclusion

This chapter detailed the development and deployment of the NewANTS contaminant transport modeling system. The interface is user friendly and can be accessed from any user machine that has web-browser capabilities -- successfully achieving the goal of an accessible contaminant mocf6deling system. NewANTS expanded on the concepts of SolidsInRivers and was able to include an additional component of output graphics intended to increase the level of detail included in the output/results of a modeling scenario. Additional graphical output is also a successful extension of the original ANTS system that had the limitation of being unable to incorporate graphical elements into its output results. NewANTS can accurately represent estimates of contaminant transport in a system requiring only the user-supplied

inputs. In addition, future updates and additional modeling scenarios can be introduced to the system by simply updating the database on the server, eliminating the need for changes and/or updates to the infrastructure on the user's machine. User error when selecting model type and properties represents one major drawback of NewANTS. If the user attempts to model a scenario using an incorrect source type or attenuation type, that choice would render the results of the simulation meaningless. However, beyond requiring a working knowledge of contaminant transport concepts, an in-depth knowledge of the behind the scenes inter-process communication between R, HTML, and Python is not necessary. Another limitation of the NewANTS system is with the web-server permissions. NewANTS is designed to write to a file from a remote request of unknown (unauthenticated) origin. The method employed in this work was to give permissions to "www" (the web root) to write to a single, specified location. This method generates files that the Computation Engine needs to function and that the Launcher and Responder scripts can recognize and understand. However, allowing unauthenticated file uploads to a server can present a security risk for two reasons. The first is that if the upload file is executable, there would be no way to track the malicious upload without authentication. In addition, an open file storage space can attract users that take advantage of the server as

a storage device, which could lead to misuse of the server resources.

Future Extensions

Some areas that could be further developed to address drawbacks and limitations of the system include the following

- Requiring web-server permissions to authenticate users in order to increase security
    - This concept was investigated as a feature of EPANETOL, which will be discussed in the following chapter.
- The addition of more model scenarios to encompass other source types and attenuation types
- Server-side storage of models so that a user could access and re-use them without the need for repeat input of required user-supplied information
- Implementation of an internet-based model selection assistant to decrease the potential for user error in identifying the simulation that corresponds to their particular contaminant transport problem.

CHAPTER 4

EPANET-On-Line

Introduction

The previous chapters documented the development and deployment of a single tool, SolidsInRivers, and a set of related tools, NewANTS, into a web environment. This chapter, EPANET-On-Line, describes methods to employ professionally successful tools into the web environment using the EPANET pipe network modeling tool (US EPA, 2018). EPANET-On-Line will hereafter be referred to as EPANETOL. To extend these concepts further, EPANETOL documents the requisite HTML and Python codes to run EPANETOL through a modern browser and details the added server configuration to allow the end-user to upload an EPANET format input file to the web server, perform a simulation, and recover the results. This chapter is organized in two parts. The first explains the procedure for building a functioning web-server-

accessed, server-side implementation of EPANET, known as EPANETOL. The second details the advantages of server-side copies of the software, specifically as it applies to future advancements and expansions of the capabilities of EPANET

Purpose

EPANET is a well-known tool used to model water distribution systems. US EPA web analytics show roughly 60,000 downloads per year for the tool (US EPA, 2018). In addition, EPANET's influence and applicability extends beyond simple water resources simulations. Many commercial products have incorporated EPANET's computation engine (or have engines tested against EPANET). Some examples include Innovyze InfoWater, which uses an enhanced computation engine evolved from EPANET, and DHI Mike URBAN (Studioars, 2018), a water distribution module that also uses an enhanced computation engine evolved from EPANET.

The software itself is evolving in an Open Source Project under the direction of a group called Open Water Analytics (OWA) (GitHub, 2018). To date, they have produced an EPANET2.2 release that is compatible with both the original ASCII input file format and the

DLL access structure pioneered in the EPANET toolbox. OWA is also working on EPANET3, which is the proposed next version of EPANET. EPANET3 will still operate using the original ASCII input file format, but the DLL access structure will no longer be supported. EPA has produced an updated GUI to support future versions on EPANET and SWMM.

Typical Use

Typical EPANET users begin by downloading and installing a copy of the program from the EPA website (US EPA, 2018). Because the newer EPANET builds are available online, a user familiar with build-from-source could choose to download, compile, and run any of the more recent versions from OWA (GitHub, 2018) or from other development teams by visiting their websites.

Once the software has been successfully installed, the users construct an input file using the Graphical User Interface (GUI). If the GUI version is an older model, hereafter referred to as the "Legacy GUI" or "L-GUI", the users can run their simulation directly from the GUI. The new GUI (pending formal release) is

hereafter referred to as the "Python GUI" or "P-GUI" (Respec, 2018). As part of this research, a Python GUI beta version (GitHub, 2018) was tested and will be discussed in further detail within this section (Cleveland and Neale, 2017).

The goal of this project is to construct a web-based tool to build, run, and interpret EPANET cases entirely on remote servers. Commercial enterprises have explored ways to allow a user to run the program using a remote machine, and this concept is further investigated in this research. The information presented in this paper is different from commercial enterprises that allow web-based modeling in that the work here is an explicit first step to provide the ability to build input files without a local copy of the EPANET software. With this new variation the user must first upload an input file and then run the program on the remote server. This paper explores that activity – as well as suggestions of how to construct a functioning web-based GUI for input file construction.

This project had two main components. Part 1 consisted of building a server-side implementation of the computation engine and the requisite inter-process communications necessary for a client to be able to upload an EPANETOL input file to the

computation server, instruct the server to run the program, and retrieve the computational output for further processing.

Part 2 focused on creating a separate web-based tool that could build an input file on the client's machine without requiring the use of EPANET GUI software.

The next section examines the methodology behind an EPANET modeling scenario using the downloaded version to perform the computations on the user's machine. The following example describes the method that a client typically uses EPANET software.

Typical Interaction Using Client-Side Only

This section illustrates the use of EPANET as a client-side activity where the software is downloaded and executed using the client machine. Figure 32 below is a simple network example. This example contains an adequate number of EPANET features to encompass a general use modelling scenario and will be used to compare client-side and server-side computations (US EPA, 2018). In the figure, all the nodes are at an elevation of 100 feet and the

supply reservoir is at an elevation of 0 feet. Therefore, the pump must supply the necessary added head for the system to operate.

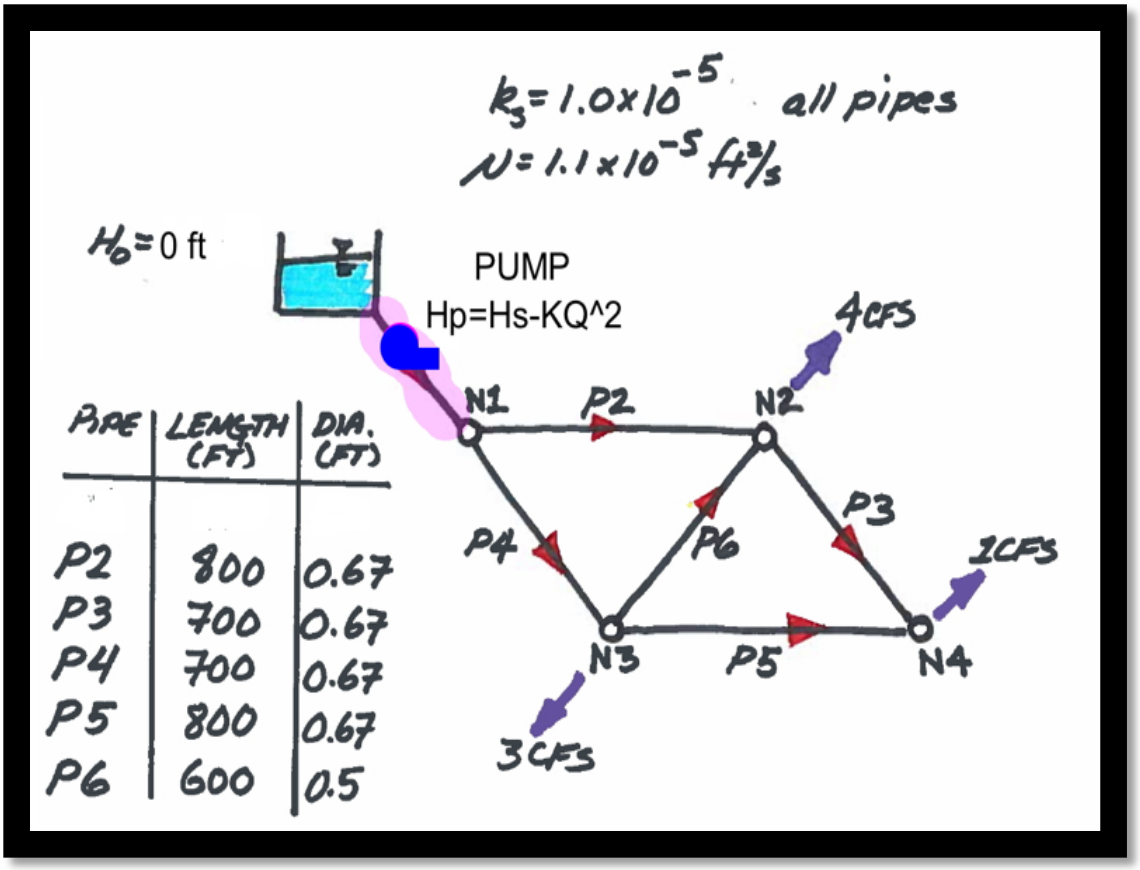


Figure 32: Example Network

Figure 33 is a screen capture of the EPANET model used within the Legacy GUI. The pump curve is shown in the lower right of the figure.

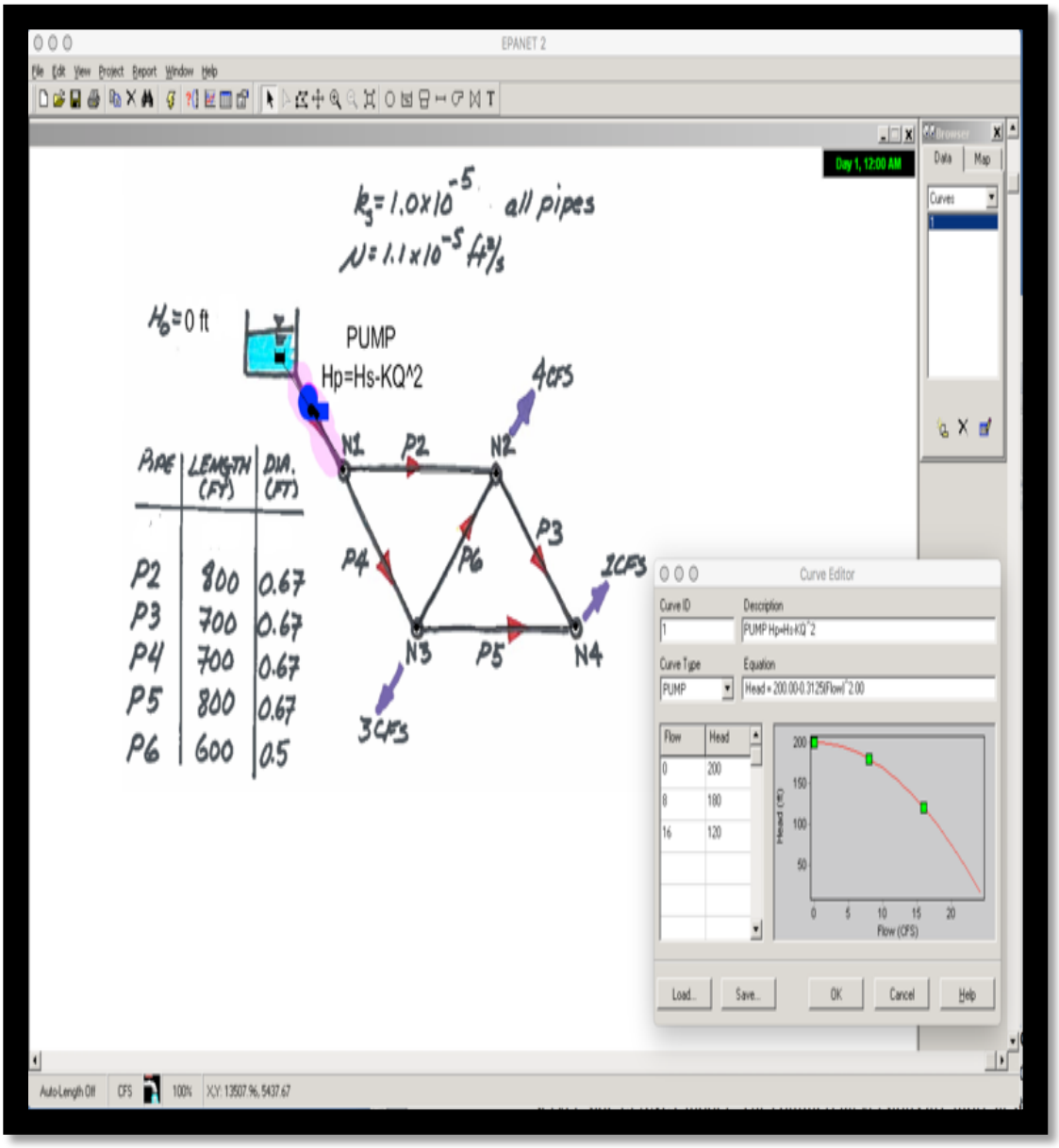


Figure 33: EPANET Example Network (L-GUI)

The EPANET ASCII input file is not directly built by the user. Instead, the user creates a graphical representation of the model

network and the GUI builds the input file for use in EPANET computation. The ASCII input file for the example is shown below in Figure 34. These input files can be saved by the user for future model implementations, sent by email, or loaded into another user's implementation.

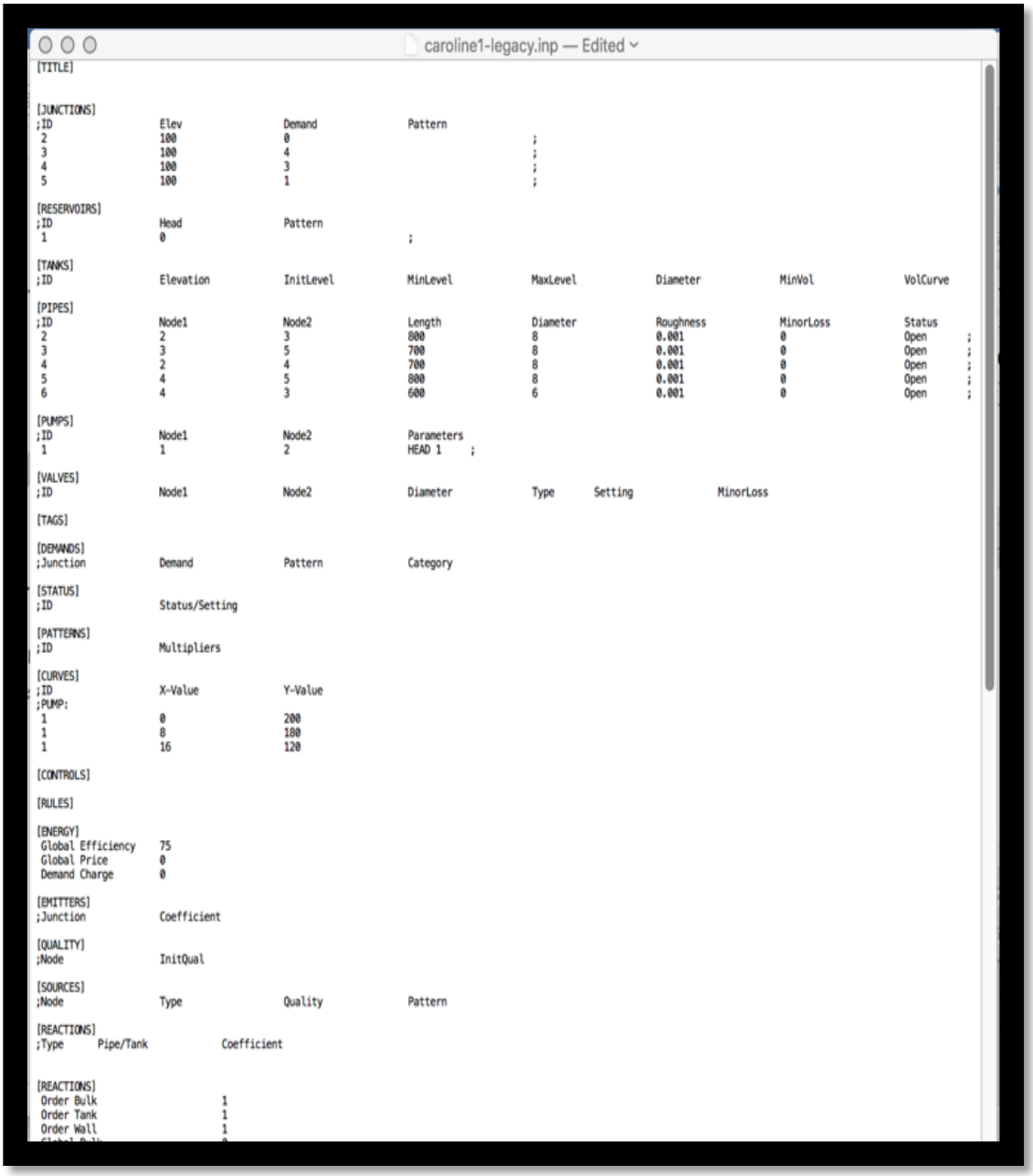


Figure 34: EPANET Input File (ASCII)

Comparison of L-GUI and P-GUI

The input file created in the L-GUI was loaded into the P-GUI implementation to demonstrate that indeed, the two networks are

topologically identical. In the P-GUI, the backdrop (hand drawn image) is intentionally suppressed because P-GUI is intended to be used with geo-referenced components, and the coordinate reference system (CRS) is unknown. The EPANET model used within the P-GUI is shown in Figure 35 (US EPA, 2018).

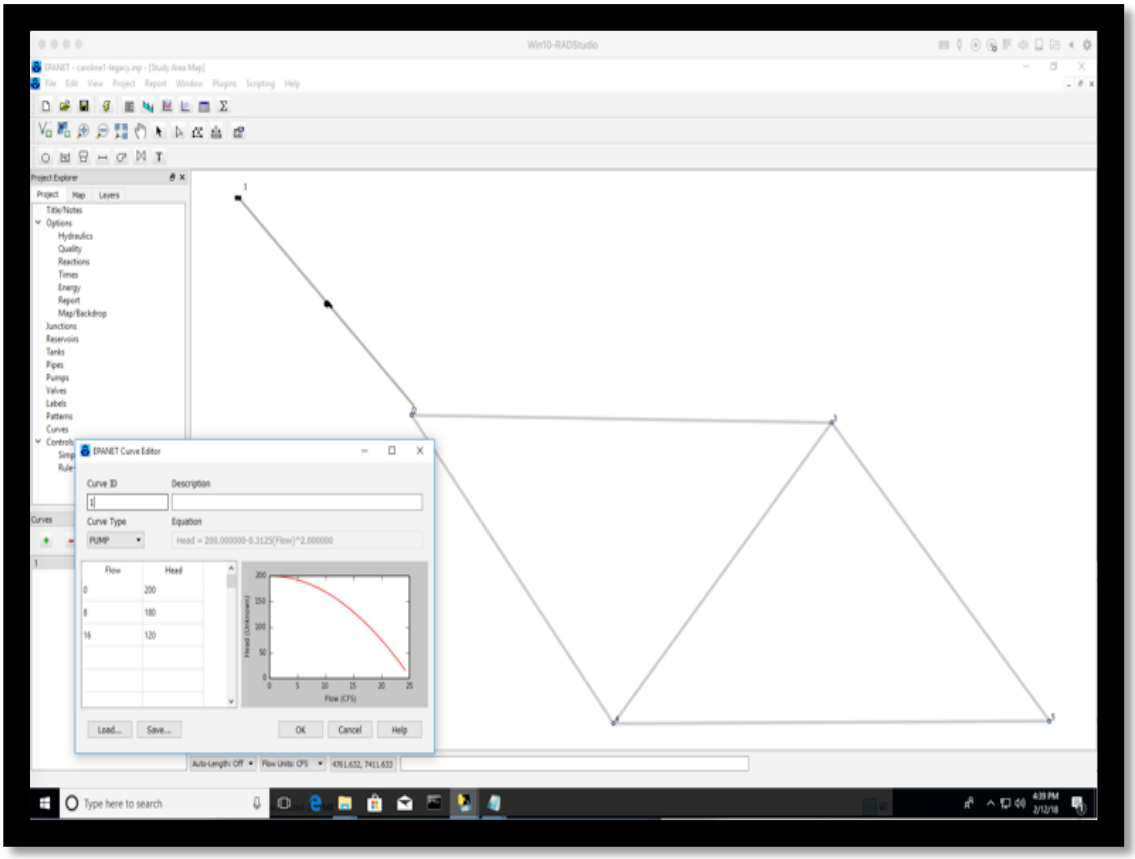


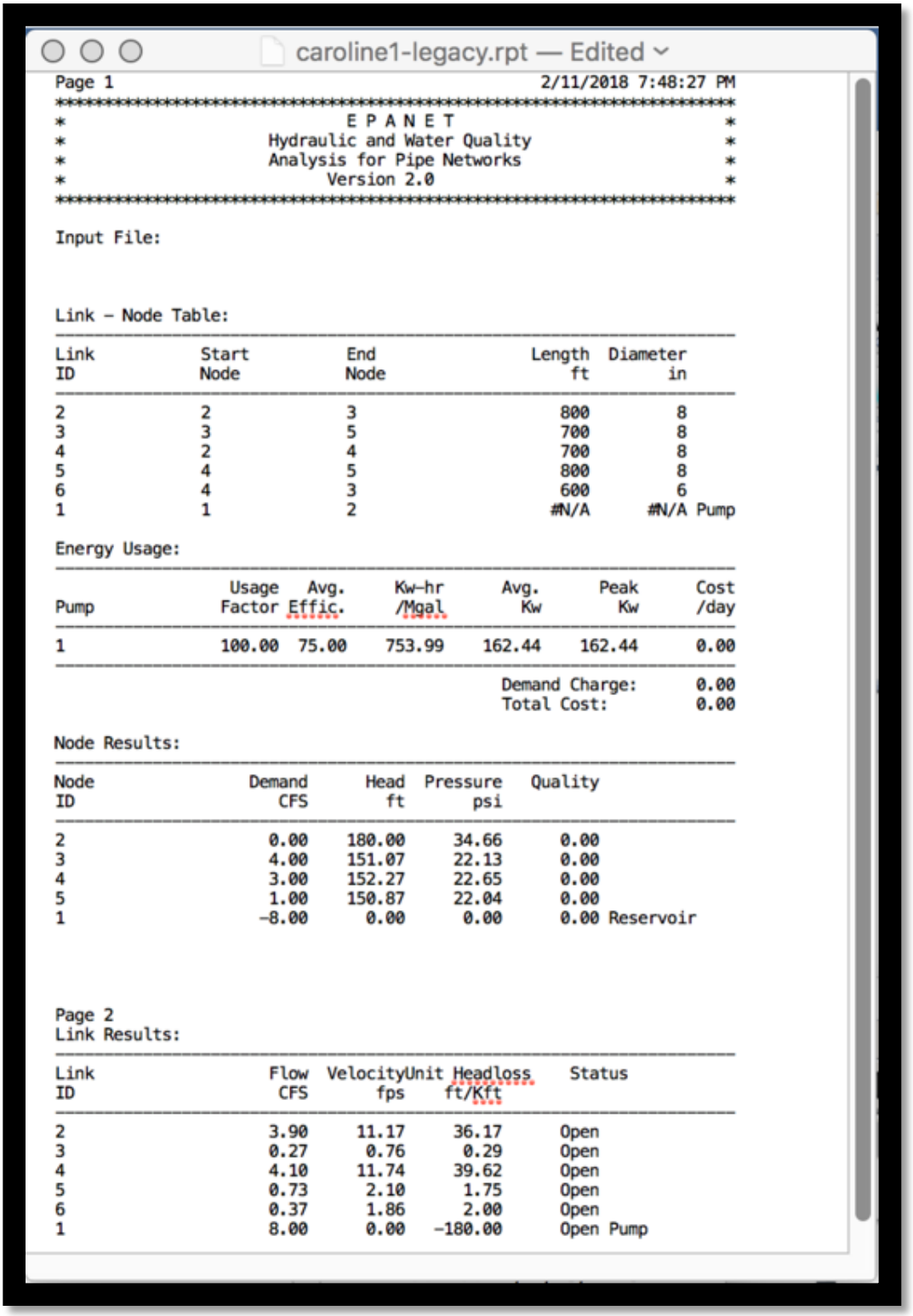
Figure 35: EPANET Example Network Rendering (P-GUI)

Simulation Output

When the simulation is performed, the results are saved to a binary file that is accessed by the GUI or external toolkit commands.

EPANET also produces an output file, shown below in Figure 36[6] (US EPA, 2018). The remainder of this chapter will focus on these files to illustrate the evolution to a server-side approach and a comparison of the two methods.

[6] The output file here is from L-GUI, however P-GUI is identical except for the pump energy reporting, which in L-GUI is known to be incorrect. The P-GUI uses a corrected computation engine; the actual release is EPANET 2.0.12.



caroline1-legacy.rpt — Edited

```
Page 1                                                    2/11/2018 7:48:27 PM
*************************************************************
*                          E P A N E T                      *
*                  Hydraulic and Water Quality              *
*                  Analysis for Pipe Networks               *
*                        Version 2.0                        *
*************************************************************

Input File:


Link - Node Table:
-------------------------------------------------------------
Link            Start           End                 Length  Diameter
ID              Node            Node                    ft        in
-------------------------------------------------------------
2               2               3                      800         8
3               3               5                      700         8
4               2               4                      700         8
5               4               5                      800         8
6               4               3                      600         6
1               1               2                     #N/A      #N/A Pump

Energy Usage:
-------------------------------------------------------------
                  Usage   Avg.     Kw-hr     Avg.      Peak      Cost
Pump             Factor  Effic.    /Mgal       Kw        Kw      /day
-------------------------------------------------------------
1                100.00   75.00   753.99   162.44    162.44      0.00
-------------------------------------------------------------
                                          Demand Charge:          0.00
                                          Total Cost:             0.00

Node Results:
-------------------------------------------------------------
Node                Demand       Head   Pressure    Quality
ID                     CFS         ft        psi
-------------------------------------------------------------
2                      0.00     180.00      34.66       0.00
3                      4.00     151.07      22.13       0.00
4                      3.00     152.27      22.65       0.00
5                      1.00     150.87      22.04       0.00
1                     -8.00       0.00       0.00       0.00 Reservoir


Page 2
Link Results:
-------------------------------------------------------------
Link                Flow  VelocityUnit Headloss     Status
ID                   CFS         fps    ft/Kft
-------------------------------------------------------------
2                    3.90       11.17     36.17       Open
3                    0.27        0.76      0.29       Open
4                    4.10       11.74     39.62       Open
5                    0.73        2.10      1.75       Open
6                    0.37        1.86      2.00       Open
1                    8.00        0.00   -180.00       Open Pump
```

Figure 36: EPANET Output File (ASCII)

Typical Interaction Using Server-Side

This section repeated the example from the previous section using the same input file, but instead the computations were performed remotely.

AccuWater is an engineering services company that provides water distribution systems design, analysis, maintenance and operations guidance. They provide water distribution system modeling services to clients using EPANET. As part of this service, a client creates an account with their company and then, using a web-interface, provides a model to Accuwater for build and simulation. This setup provides clients with EPANET assistance using AccuWater's servers. The client's system is maintained by Accuwater, and the client can access the model remotely as necessary for their needs (AccuModel, 2018).

The commercial product requires a contract/subscription for meaningful interaction, but as part of AccuWater's marketing strategy, a free version that allows a remote user to upload and run a **single** model is provided. The commercial version is identical,

except the client can maintain multiple models in their account.[7] The free version is used to illustrate the modeling process using a remote server.

User Supplied Model Input

The first step is to obtain a user account to gain access to the AccuWater system. The user then uploads an input (.INP) file to the remote server. The file upload process first prompts the user to identify the coordinate reference system to be used. The Accuwater interface is shown below in Figure 37 (AccuModel, 2018).

[7] Multiple models mean multiple input files –each could represent a single system of interest to the client, but with different scenarios being modeled.

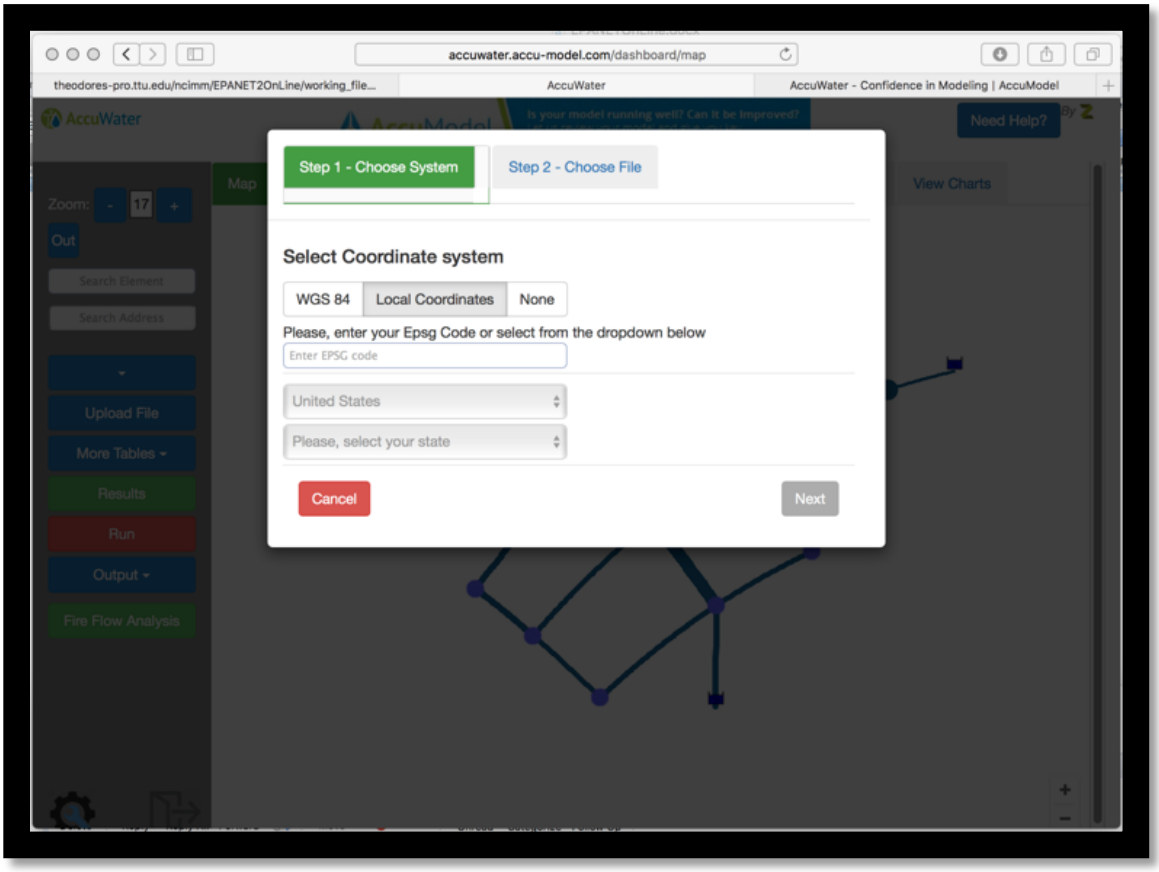


Figure 37: AccuWater Coordinate Reference System Identification

Next, the client selects a .INP file from his/her own machine to upload to the remote server. This step is shown in Figure 38 below.

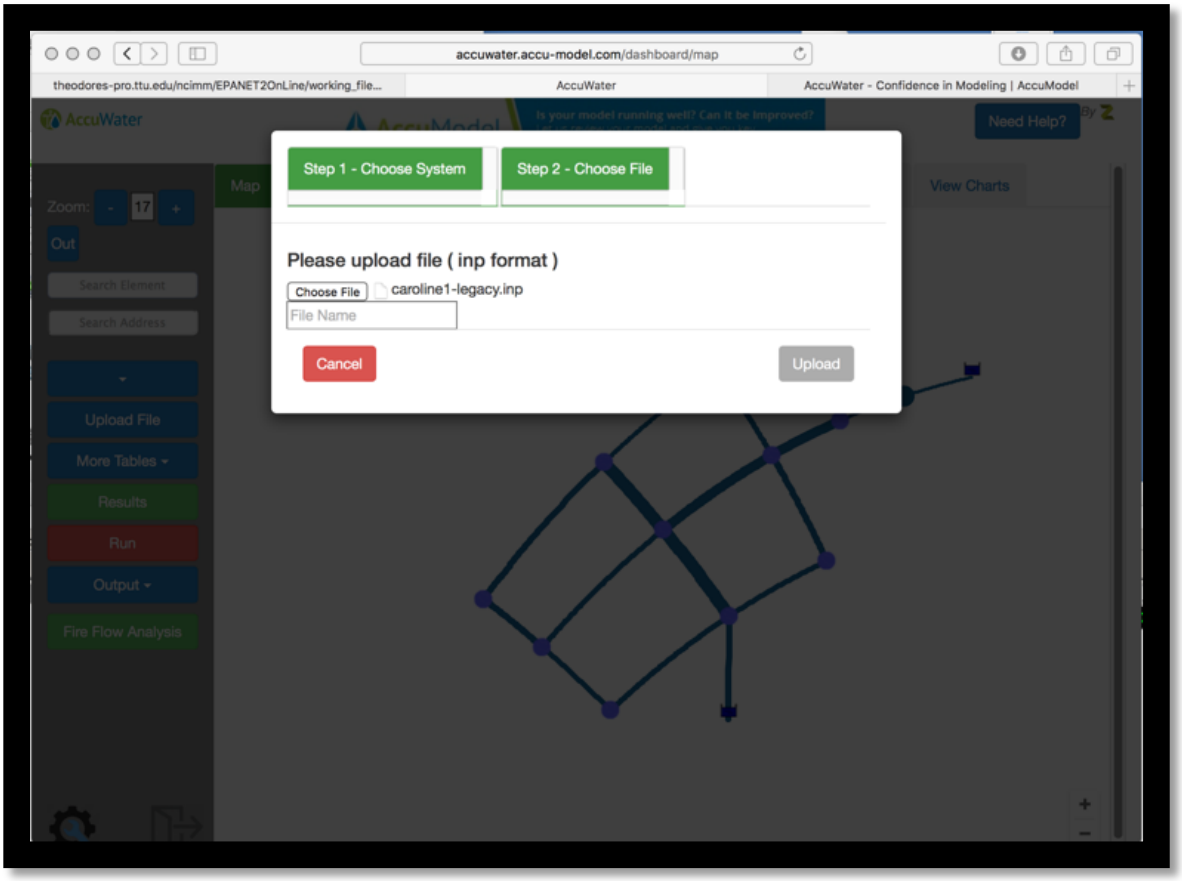


Figure 38: AccuWater .INP File Upload

Once the file is uploaded, the remote machine renders a map of the system, shown in Figure 39. The file is parsed and stored in a database and visual assessment of the uploaded network confirms that the topology is correct.

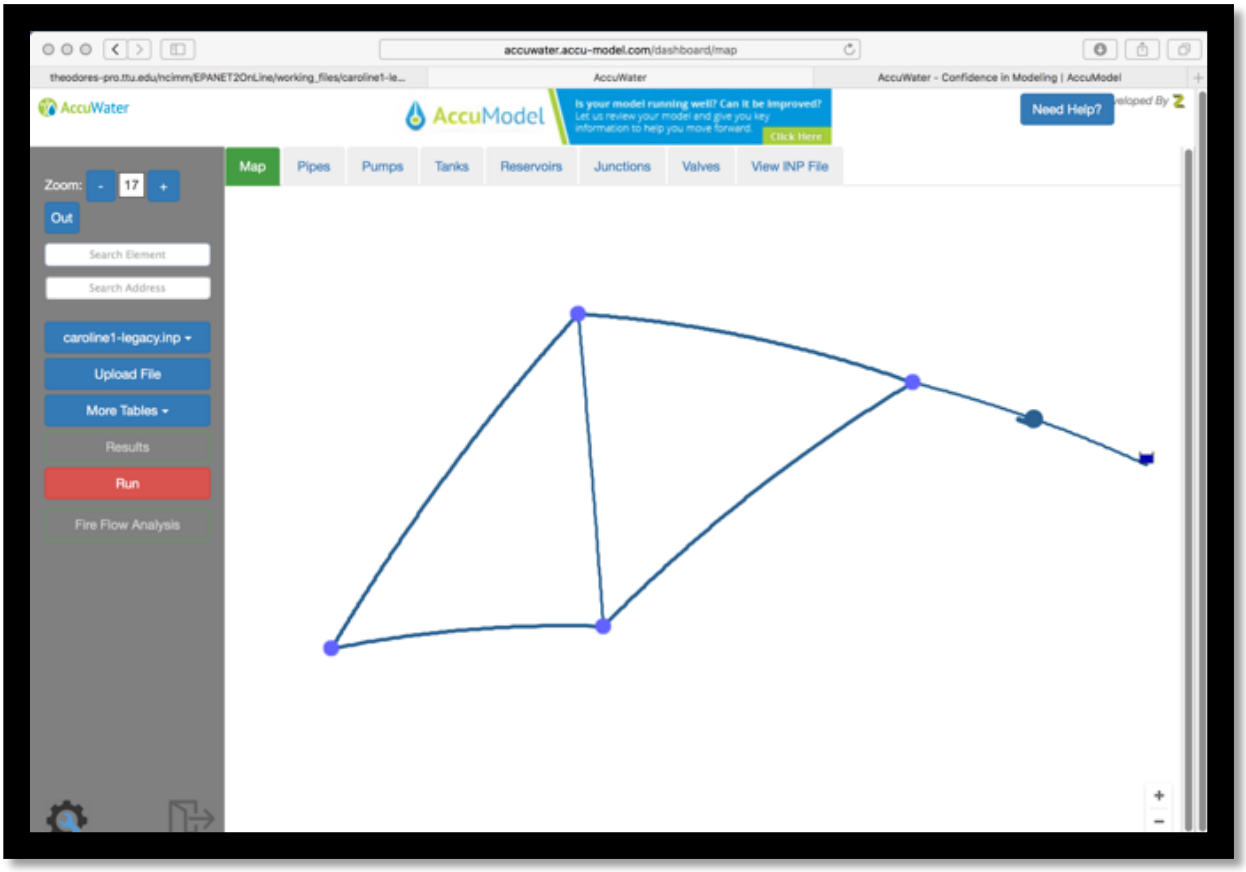


Figure 39: AccuWater Example Network Rendering

Once the upload is complete, the remote user runs the simulation and receives notification that the output file has been rendered, shown in Figure 40.

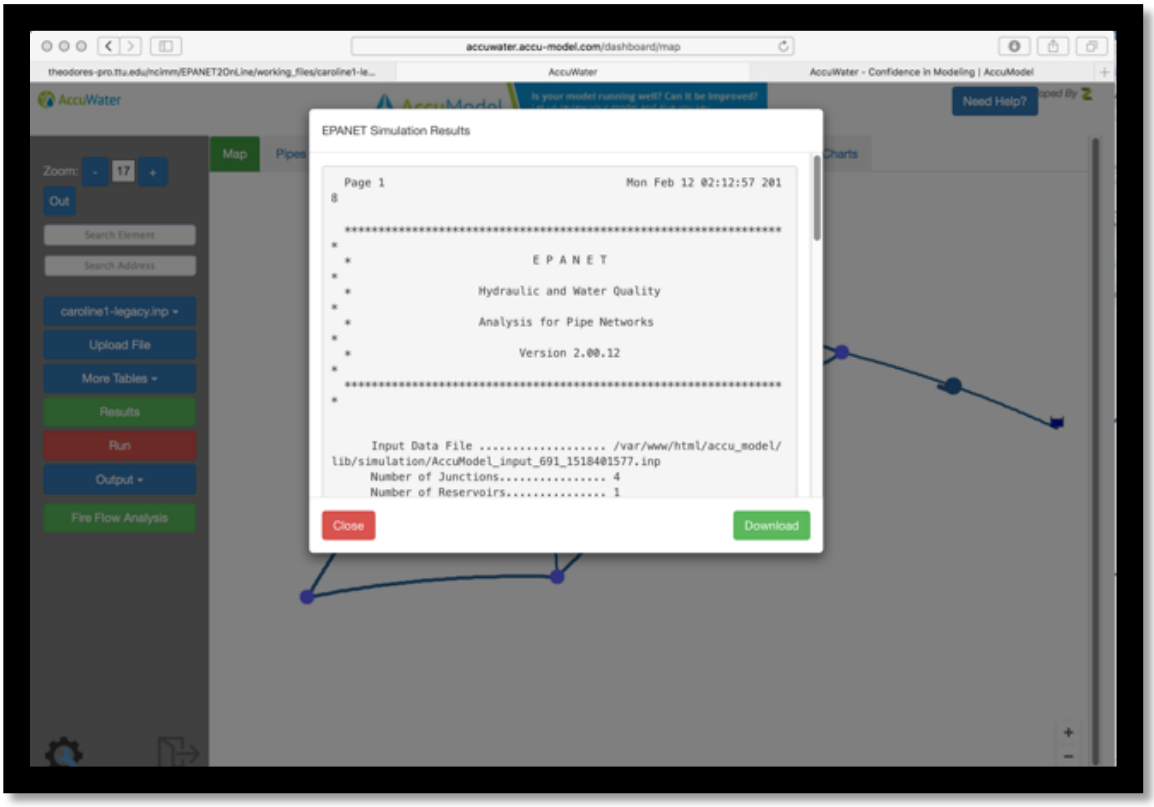
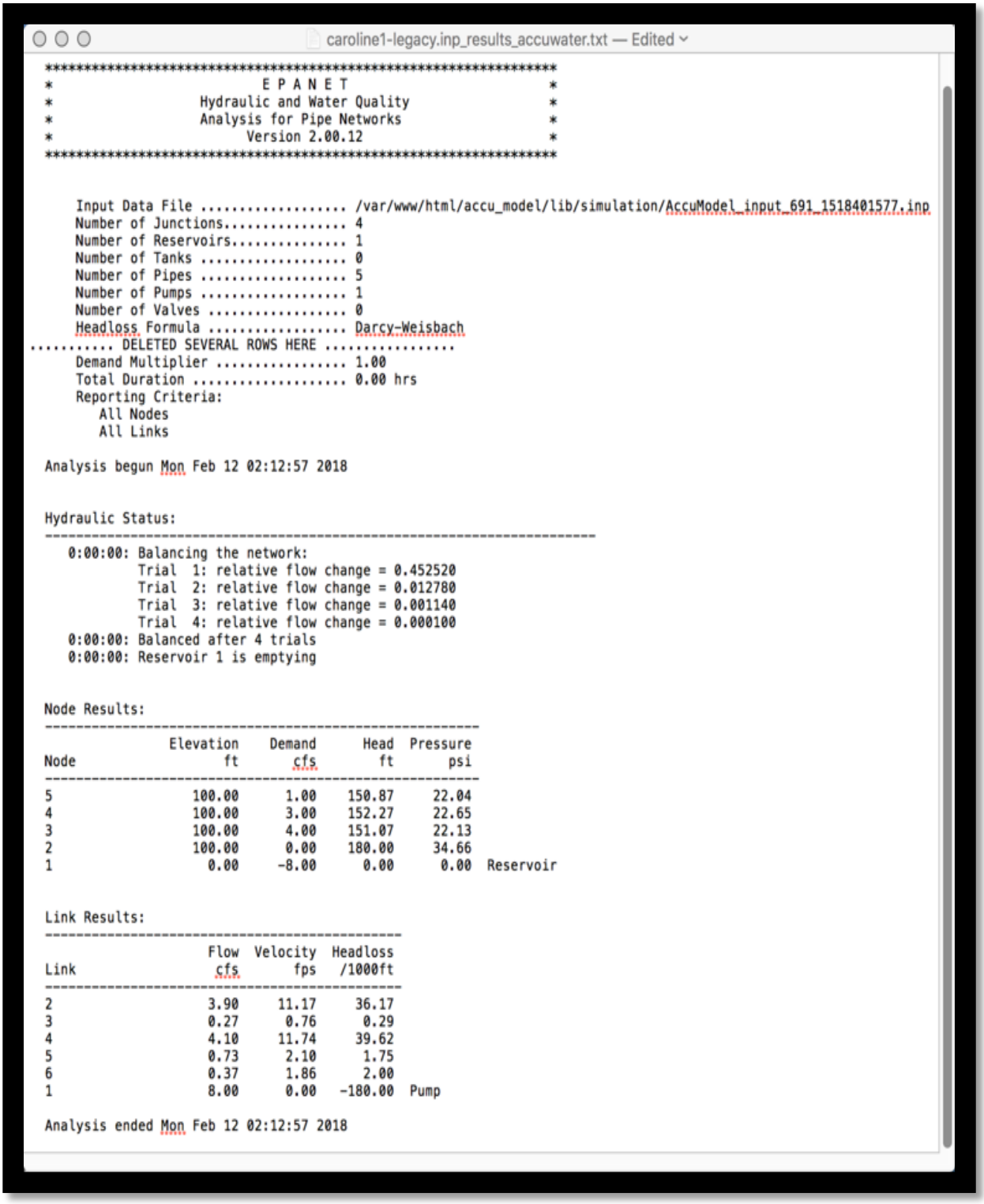


Figure 40: AccuWater Successful Simulation Notification

Discussion of Results

Figure 41 is the output file after download. The model results found using the two different methods were identical with two minor exceptions. First, the AccuWater output file header indicated the use of EPANET2.00.12, whereas the first method used EPANET2.0. Second, the output file from AccuWater did not report the energy results as the downloaded EPANET model output did. Therefore, it was concluded that the same input file, whether run on a client's machine or on a remote server, produced the same results – a desirable and anticipated outcome (AccuModel, 2018).



caroline1-legacy.inp_results_accuwater.txt — Edited

```
******************************************************************
*                          E P A N E T                           *
*                  Hydraulic and Water Quality                   *
*                  Analysis for Pipe Networks                    *
*                        Version 2.00.12                         *
******************************************************************


    Input Data File ..... /var/www/html/accu_model/lib/simulation/AccuModel_input_691_1518401577.inp
    Number of Junctions..... 4
    Number of Reservoirs..... 1
    Number of Tanks ..... 0
    Number of Pipes ..... 5
    Number of Pumps ..... 1
    Number of Valves ..... 0
    Headloss Formula ..... Darcy-Weisbach
..... DELETED SEVERAL ROWS HERE .....
    Demand Multiplier ..... 1.00
    Total Duration ..... 0.00 hrs
    Reporting Criteria:
       All Nodes
       All Links

Analysis begun Mon Feb 12 02:12:57 2018


Hydraulic Status:
--------------------------------------------------------------------------
   0:00:00: Balancing the network:
            Trial  1: relative flow change = 0.452520
            Trial  2: relative flow change = 0.012780
            Trial  3: relative flow change = 0.001140
            Trial  4: relative flow change = 0.000100
   0:00:00: Balanced after 4 trials
   0:00:00: Reservoir 1 is emptying


Node Results:
-----------------------------------------------------
                Elevation    Demand      Head  Pressure
Node                   ft       cfs        ft       psi
-----------------------------------------------------
5                  100.00      1.00    150.87     22.04
4                  100.00      3.00    152.27     22.65
3                  100.00      4.00    151.07     22.13
2                  100.00      0.00    180.00     34.66
1                    0.00     -8.00      0.00      0.00  Reservoir


Link Results:
-----------------------------------------------
                    Flow  Velocity  Headloss
Link                 cfs       fps   /1000ft
-----------------------------------------------
2                   3.90     11.17     36.17
3                   0.27      0.76      0.29
4                   4.10     11.74     39.62
5                   0.73      2.10      1.75
6                   0.37      1.86      2.00
1                   8.00      0.00   -180.00  Pump

Analysis ended Mon Feb 12 02:12:57 2018
```

Figure 41: AccuWater Output File (ASCII)

Comparison Conclusion

The results are essentially the same as using the client-side implementation.[8] The AccuWater interface ingests the uploaded input file and parses it into a server-side database so the user can change values of existing components (AccuModel, 2018).

Related Commercial Products Available

Aquaveo

Another commercial product with equivalent capability is offered by Aquaveo (Aquaveo, 2018). The software product is called "City Water". Similar to AccuWater, it ingests an input file and coordinate system, parses the file into a database, and passes geographic information to a map server.

[8] The Node and Link results are identical using either the accuwater.com tool or the downloaded EPANET program.

EPANET Java

Another related product is offered by Baseform (Baseform, 2018). The product is called EPANET Java and is an EPANET simulation engine able to perform full-range hydraulic and water quality network simulations. Using Baseform's 2D/3D network and results visualization, EPANET Java offers a simulation engine that is a Java re-write of the EPANET standard with EPANET's full network modeling functionality. It can perform static or extended-period simulation on .INP standard model files (Baseform, 2018).

EPANET.DE

Another product that runs client-side operations is a JavaScript implementation of EPANET called *EPANET.DE* (Epanet.de, 2018). An examination of the distribution codes suggests that *EPANET.DE* is a refactoring into JavaScript using various compilers that generate a functional JavaScript copy of the EPANET computation engine. This tool runs entirely on the client side, but the code is interpreted "Just-In-Time" by the client's web browser.

Like the other products, EPANET.DE ingests an .INP file. However, unlike the other products, the input file can be edited in the interface. A user knowledgeable in how an input file must be structured could actually add components, which is a major step towards a fully web-based implementation of EPANET.

EPANET.DE Example

EPANET.DE produces yet another identical output report when supplied with the same input file from the AccuWater/EPANET example, further validating the conclusion stated previously. The EPANET.DE interface is shown below in Figure 42 (Epanet.de, 2018).

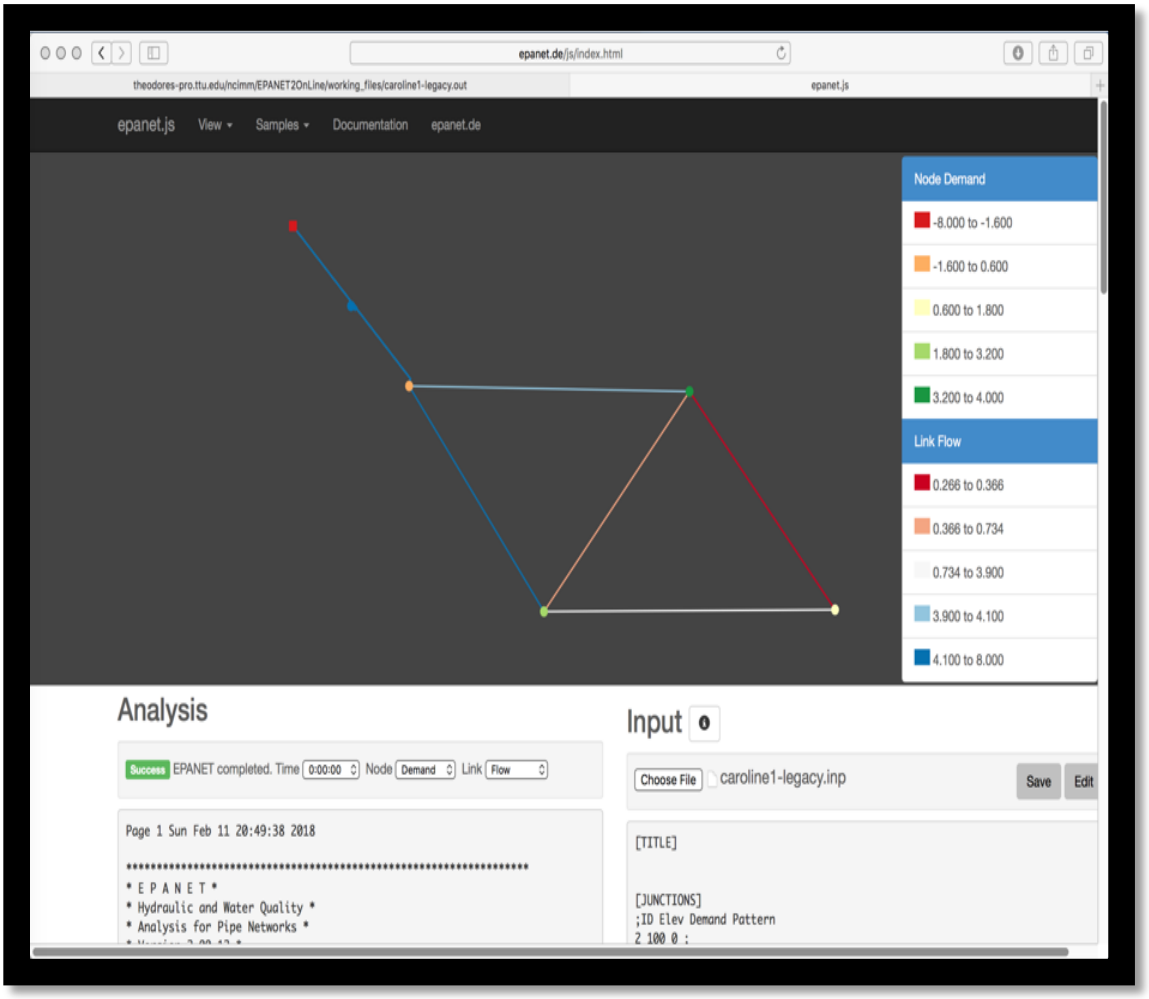


Figure 42: EPANET.DE Example Network Rendering (JavaScript)

EPANET-On-Line (NCIMM/TTU) Research Application

The NCIMM/TTU EPANET On-Line application, EPANETOL, was developed to reverse-engineer the server-side computational capabilities demonstrated by the commercial products in anticipation of three near-future advances. The first was to present a seamless way to build EPANET3 implementations and immediately make them available to users who many not have the desire to download and compile from source. As of now, EPANET3 does not have an interface, although it is anticipated that the P-GUI is readily adaptable to the software. The second is in expectation of formal parallelization of EPANET to provide a mechanism for clients to use parallel processing without having to invest in physical hardware Graphical Processing Units (GPU) for occasional simulation needs. By moving the parallel processing to a remote machine, the maintenance of the GPU-capable program becomes tied to the resident hardware. The third is in anticipation of a fully browser-enabled interface. The GUI itself runs in a browser – thereby achieving true operating system and potentially architecture independence.

This section documents how one can build a functioning, web-server-accessed, server-side implementation of EPANET. The

results will show both an EPANET2 and an EPANET3 example. The EPANET3 implementation is to illustrate the maintenance advantage of a server-side copy of the software.

Server-Side Requirements

The requirements for EPANETOL server-side processing are an extension of the communications and methodology developed in SolidsInRivers and NewANTS. The EPANETOL implementation requires an additional step in that the computation engine itself is pre-compiled and the required input information is to be communicated by an uploaded file. In addition, unlike the other two interfaces (SolidsInRivers and NewANTS), direct modification of the source code is undesirable.

SolidsInRivers presents a client with a web-based interface that the client populates with input values and then requests a result. The interface generates a small input file on the server. This file is ingested by the simple database program (written in R, run on the server using a system command generated by the web interface) and a result file and graphic are generated. These two files are then assembled into the output interface presented to the client.

All the effort is server side, and the client need not be authenticated.

The NewANTS toolkit extends the SolidsInRivers concept by adding more elaborate output, specifically the tabular results associated with the simulation. These detailed output files were designed so the clients can plot (or otherwise interpret) their simulation results in greater detail and with improved accuracy; however, the results are really just longer web pages. The client still needs to copy and paste the results for their needs.

EPANETOL adds several additional components. The first is that the input data required are more extensive than the other two tools – the client needs to generate an input file. Like the commercial examples, this activity is still client-side, not yet supported directly in the web-browser. Next, the file itself needs to be uploaded. To have some control over the server use, the application herein demands authentication. The second difference is remote storage of input, binary, and output files. Like SolidsInRivers and NewANTS, the program must be able to write files to the server, but with EPANETOL, these files must be retained for a while for the client to retrieve them. Retrieval is an issue because the program itself may take minutes to run. The web connection is

designed to time out after a specified length of time, but the program continues to run, and the client can check in upon completion to retrieve the file.

PHP Concepts

PHP is a server-side scripting language designed for web development. PHP originally stood for Personal Home Page but has since been modified to mean Hypertext Preprocessor. PHP codes are typically embedded in HTML codes and are usually processed as a CGI executable. PHP is a general-purpose scripting language that is especially suited to server-side web development, where it runs on a web server. Client-side graphical user interface applications use PHP and provide dynamic content from a web server to a client. The web server combines the results of the interpreted and executed PHP code with the generated web page.

Demonstration Using the Example

The example described in preceding sections of this document will be used to demonstrate the implementation, followed by more detailed information in this section. Shown below, Figure 43, is a screen shot of the EPANETOL homepage. The page has information about the mission, archives of previous versions, and current updates on EPANETOL research. Under the NCIMM EPANET Research tab is a link that will redirect the user to EPANET On-Line.

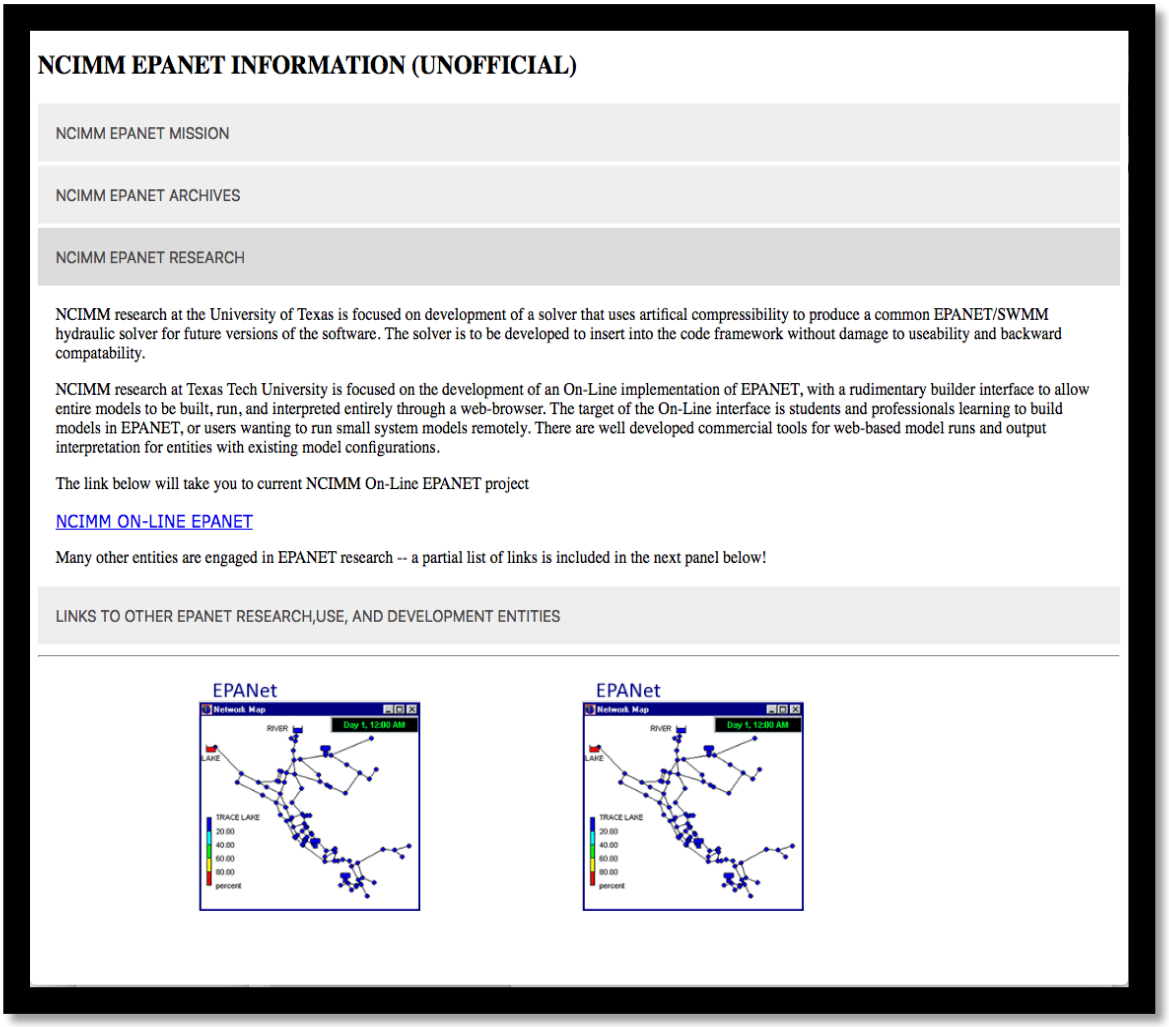


Figure 43: NCIMM/TTU EPANET Interface Homepage (HTML)

Figure 44 is a screenshot of the EPANETOL entry page. The right-hand column shows links that will allow the user to view

online input/output files, upload a new EPANETOL input file, run an EPANETOL model, and an experimental input file generator.

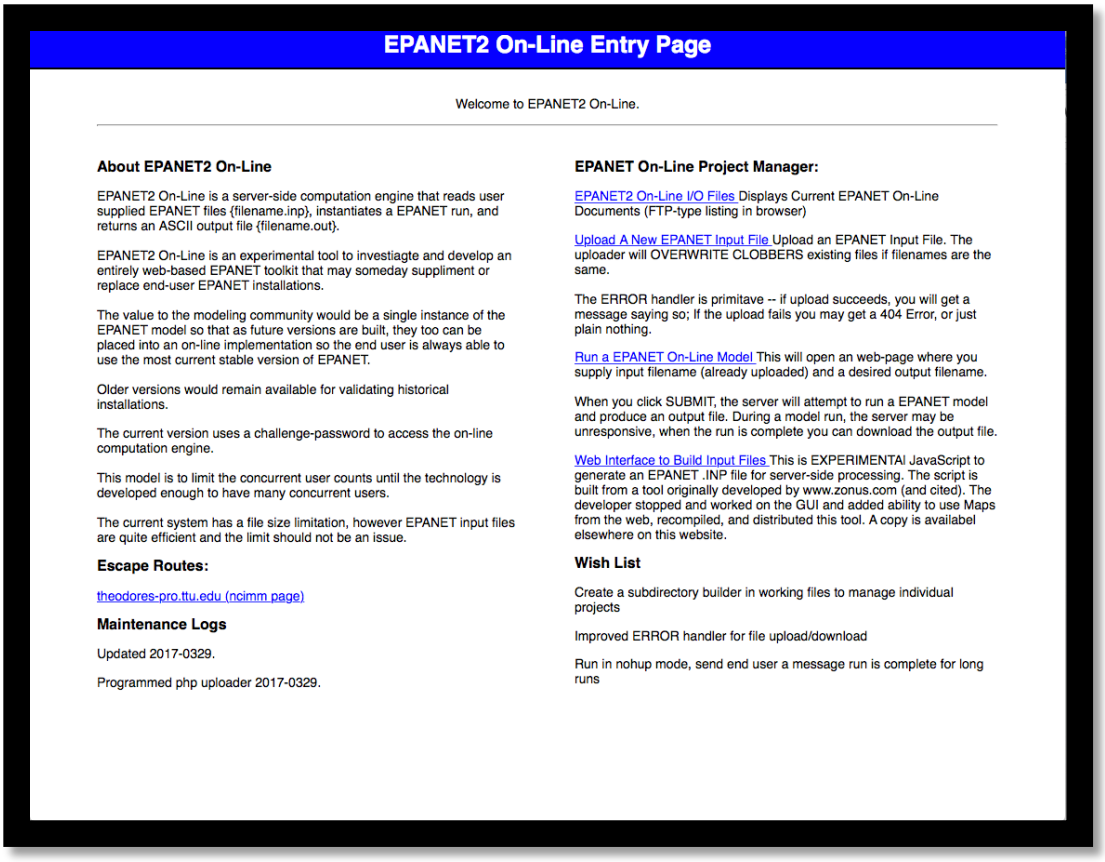
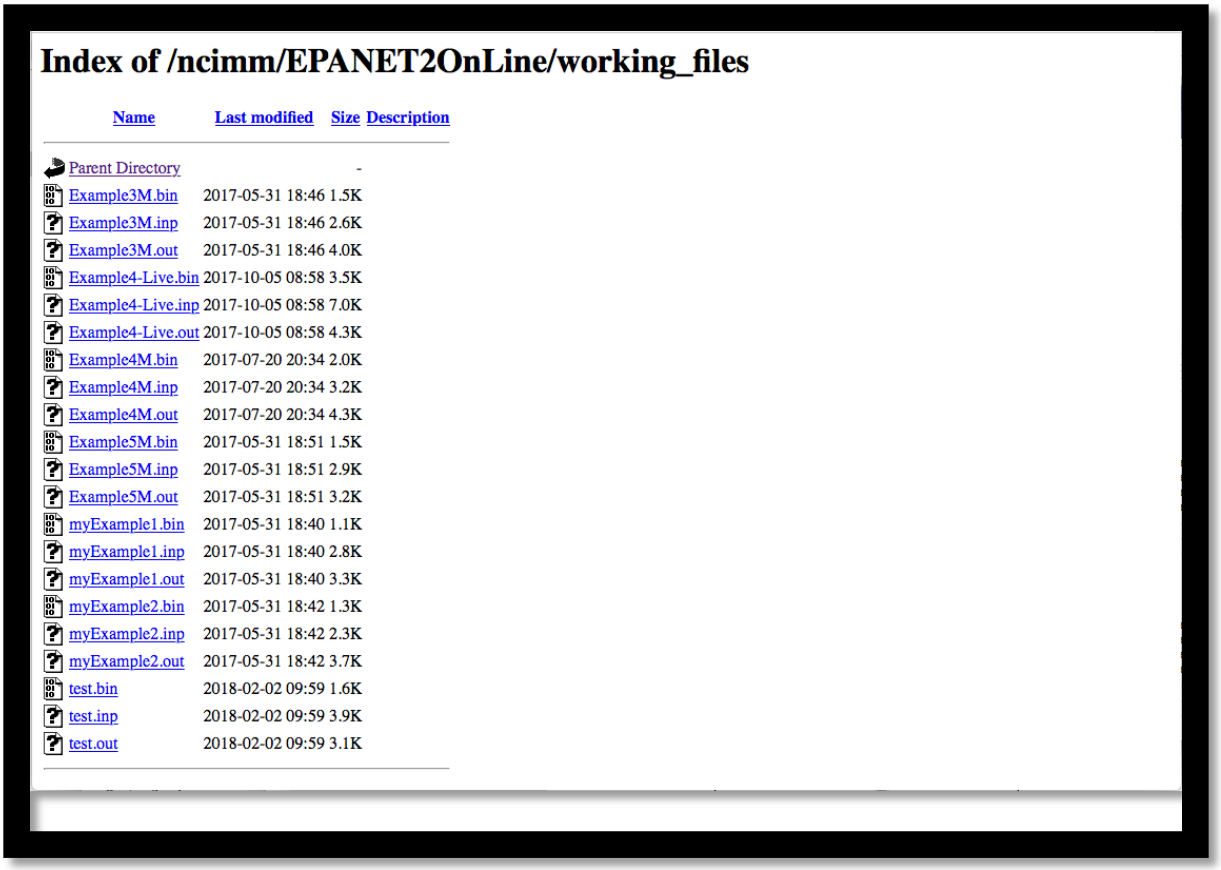


Figure 44: EPANETOL Interface Directory Page (HTML)

Figure 45 below is a screen shot of the directory where the files are located on the remote server. This contains all the existing files used in EPANETOL. Note that the directory below contains

several files, however, this example does not yet appear in the file directory with the name caroline1-legacy.inp.



Index of /ncimm/EPANET2OnLine/working_files

Name	Last modified	Size	Description
Parent Directory		-	
Example3M.bin	2017-05-31 18:46	1.5K	
Example3M.inp	2017-05-31 18:46	2.6K	
Example3M.out	2017-05-31 18:46	4.0K	
Example4-Live.bin	2017-10-05 08:58	3.5K	
Example4-Live.inp	2017-10-05 08:58	7.0K	
Example4-Live.out	2017-10-05 08:58	4.3K	
Example4M.bin	2017-07-20 20:34	2.0K	
Example4M.inp	2017-07-20 20:34	3.2K	
Example4M.out	2017-07-20 20:34	4.3K	
Example5M.bin	2017-05-31 18:51	1.5K	
Example5M.inp	2017-05-31 18:51	2.9K	
Example5M.out	2017-05-31 18:51	3.2K	
myExample1.bin	2017-05-31 18:40	1.1K	
myExample1.inp	2017-05-31 18:40	2.8K	
myExample1.out	2017-05-31 18:40	3.3K	
myExample2.bin	2017-05-31 18:42	1.3K	
myExample2.inp	2017-05-31 18:42	2.3K	
myExample2.out	2017-05-31 18:42	3.7K	
test.bin	2018-02-02 09:59	1.6K	
test.inp	2018-02-02 09:59	3.9K	
test.out	2018-02-02 09:59	3.1K	

Figure 45: EPANETOL Interface Expanded Directory Page (HTML)

The EPANET On-Line authentication form is shown below in Figure 46. This form is where users enter their individual login

and password (the credentials shown in the figure below are simply placeholders). Upon successful login, the user is taken to the upload page. Failed authentication will return an error message.

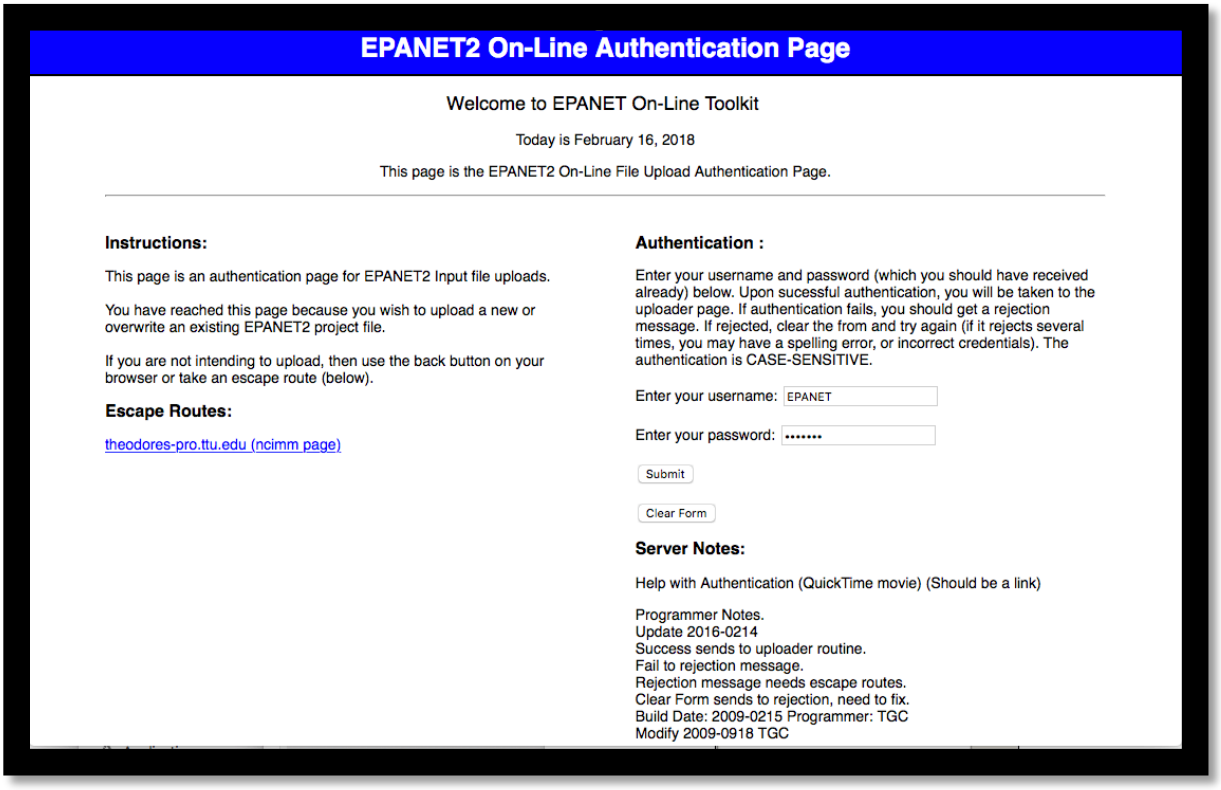


Figure 46: EPANETOL Interface Authentication Page (HTML)

Figure 47 is the EPANET On-Line uploader page. A user can select and upload a file from this page for modeling in EPANETOL.

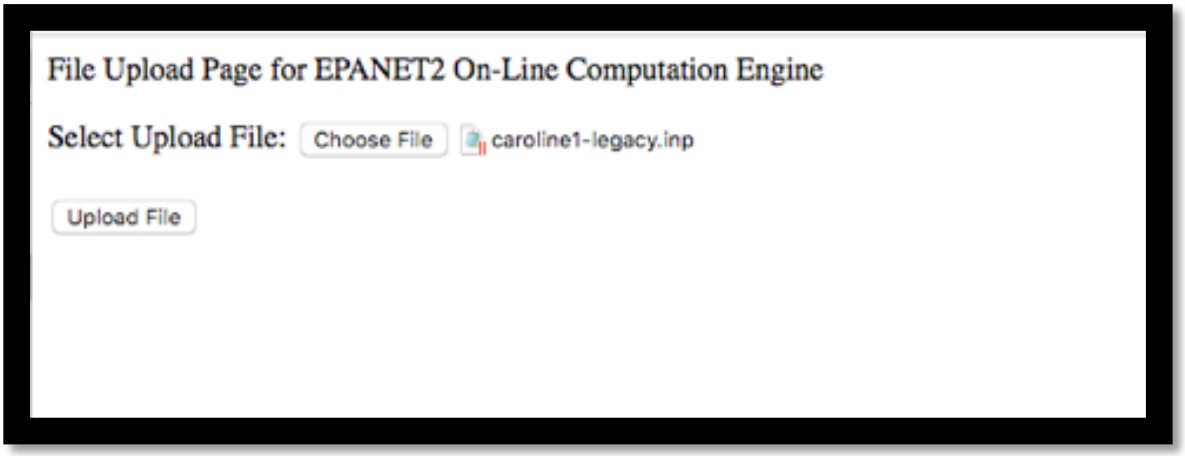


Figure 47: EPANETOL Interface Input File Upload Page (HTML)

When a user uploads a file successfully, the screen shown in Figure 48 is returned.

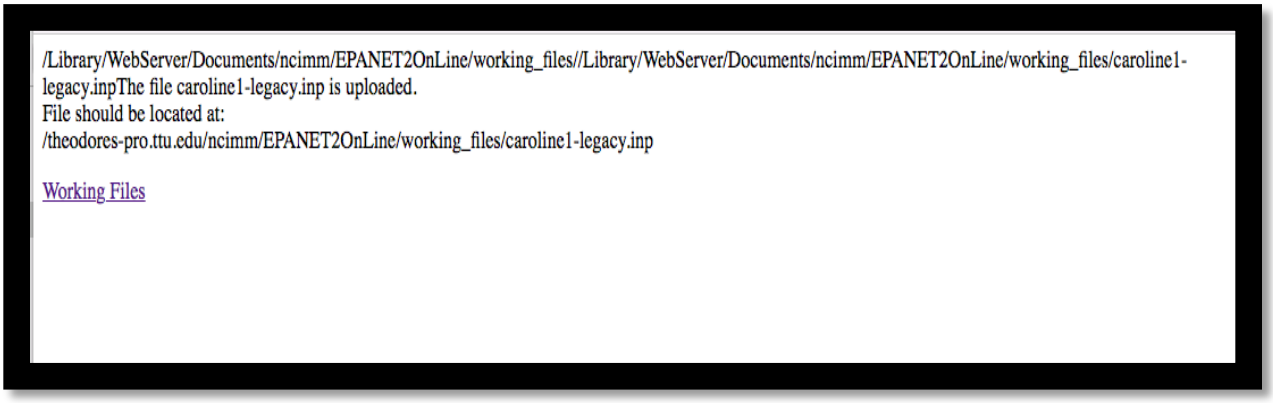
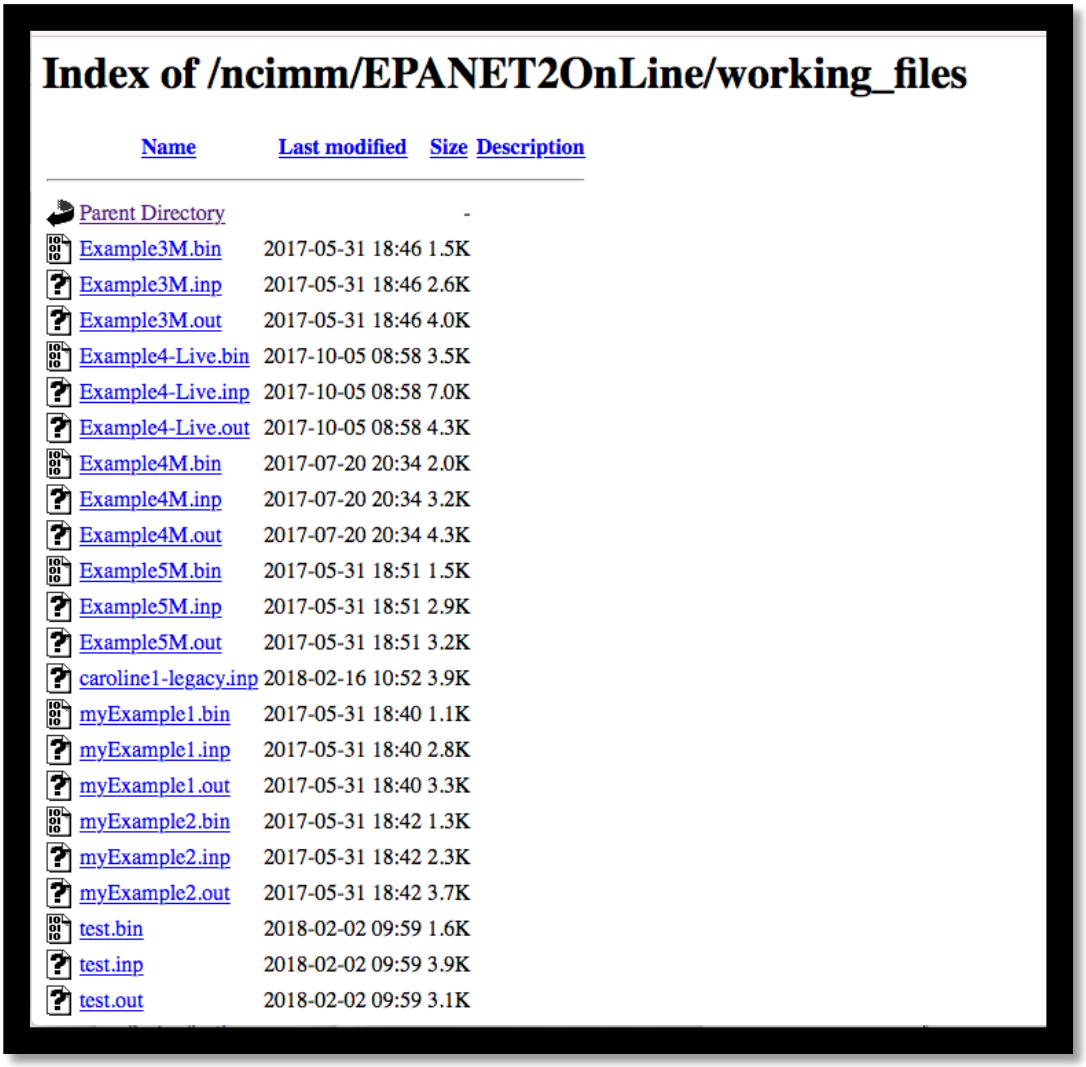


Figure 48: EPANETOL Interface Successful Input File Upload Page (HTML)

Figure 49 is a screen shot of the directory where EPANETOL files are stored. Note that the example problem appears in the file directory (caroline1-legacy.inp). Here, a client can select and download files from the server.



Index of /ncimm/EPANET2OnLine/working_files

Name	Last modified	Size	Description
Parent Directory		-	
Example3M.bin	2017-05-31 18:46	1.5K	
Example3M.inp	2017-05-31 18:46	2.6K	
Example3M.out	2017-05-31 18:46	4.0K	
Example4-Live.bin	2017-10-05 08:58	3.5K	
Example4-Live.inp	2017-10-05 08:58	7.0K	
Example4-Live.out	2017-10-05 08:58	4.3K	
Example4M.bin	2017-07-20 20:34	2.0K	
Example4M.inp	2017-07-20 20:34	3.2K	
Example4M.out	2017-07-20 20:34	4.3K	
Example5M.bin	2017-05-31 18:51	1.5K	
Example5M.inp	2017-05-31 18:51	2.9K	
Example5M.out	2017-05-31 18:51	3.2K	
caroline1-legacy.inp	2018-02-16 10:52	3.9K	
myExample1.bin	2017-05-31 18:40	1.1K	
myExample1.inp	2017-05-31 18:40	2.8K	
myExample1.out	2017-05-31 18:40	3.3K	
myExample2.bin	2017-05-31 18:42	1.3K	
myExample2.inp	2017-05-31 18:42	2.3K	
myExample2.out	2017-05-31 18:42	3.7K	
test.bin	2018-02-02 09:59	1.6K	
test.inp	2018-02-02 09:59	3.9K	
test.out	2018-02-02 09:59	3.1K	

Figure 49: EPANETOL Interface Expanded Directory Page-With Example (HTML)

Figure 50 is a screenshot of EPANETOL launcher page. The launcher begins the EPANETOL simulation process.

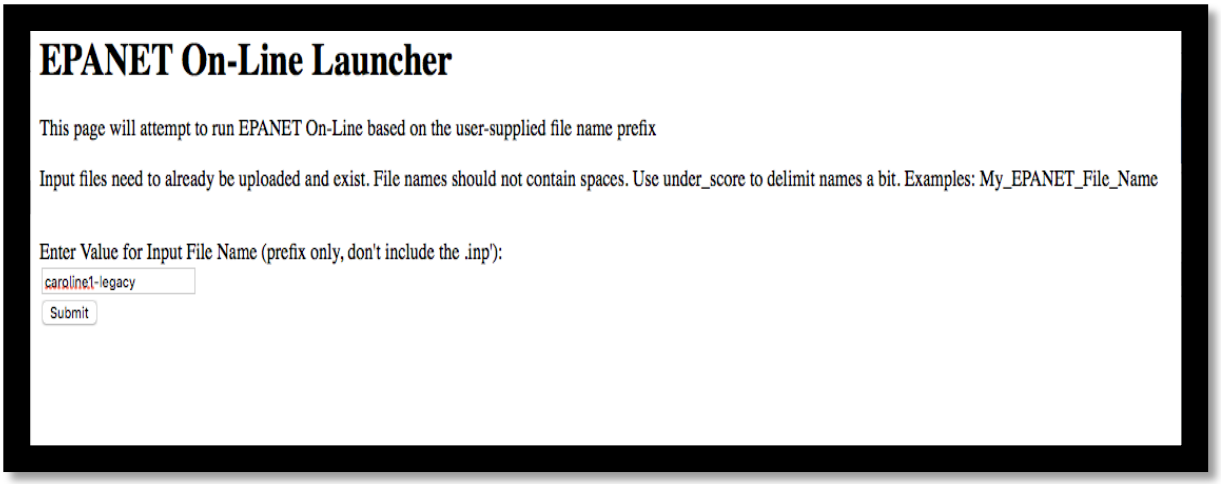
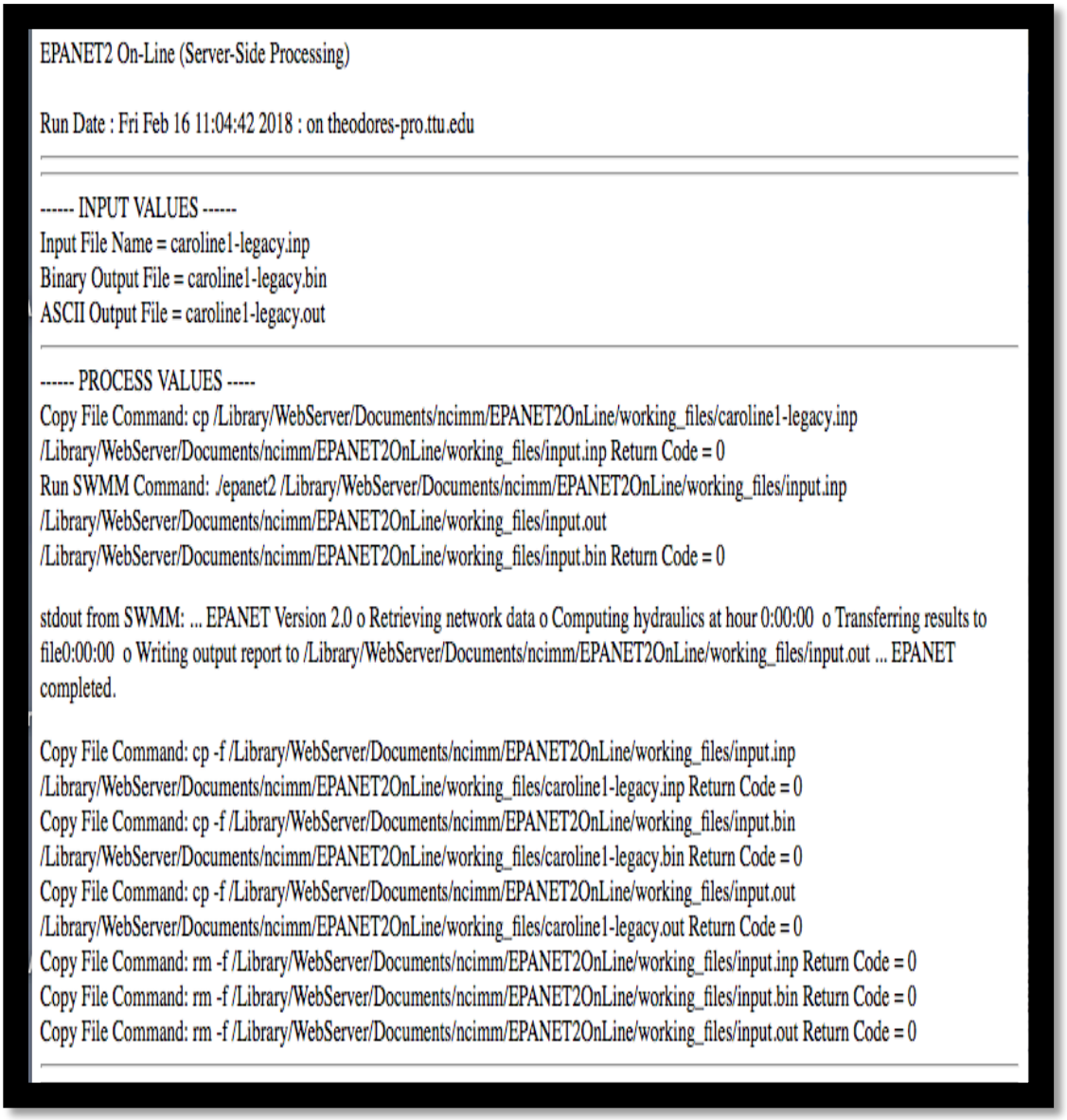


Figure 50: EPANETOL Interface Simulation Launch Page (HTML)

Shown below in Figure 51 is the EPANETOL web report on the model simulation. The error code == 0, shown in the last three lines of Figure 51, indicates that the various processes returned normally, which is a desirable result.



EPANET2 On-Line (Server-Side Processing)

Run Date : Fri Feb 16 11:04:42 2018 : on theodores-pro.ttu.edu

---

------ INPUT VALUES ------
Input File Name = caroline1-legacy.inp
Binary Output File = caroline1-legacy.bin
ASCII Output File = caroline1-legacy.out

------ PROCESS VALUES -----
Copy File Command: cp /Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/caroline1-legacy.inp
/Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/input.inp Return Code = 0
Run SWMM Command: /epanet2 /Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/input.inp
/Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/input.out
/Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/input.bin Return Code = 0

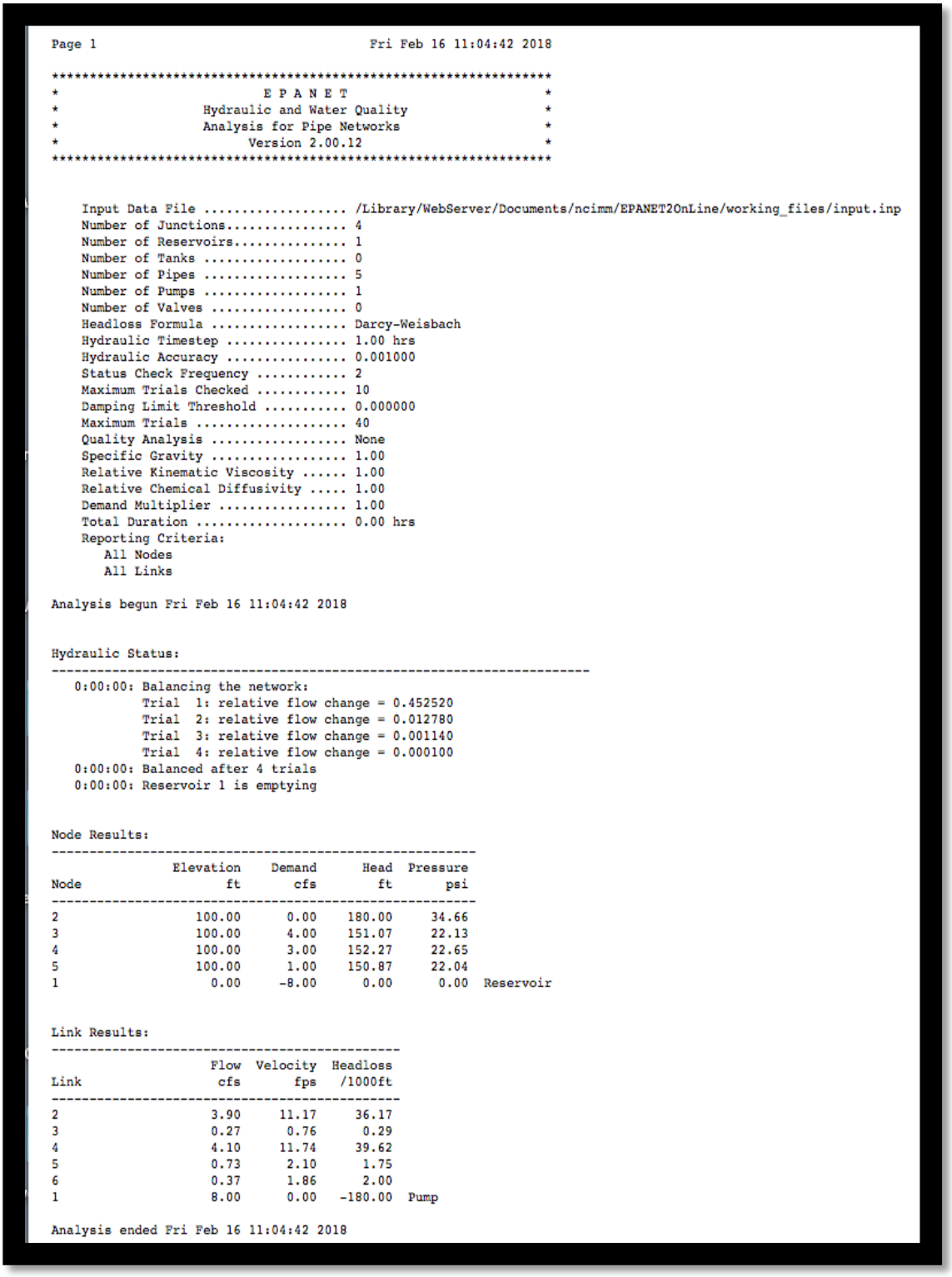
stdout from SWMM: ... EPANET Version 2.0 o Retrieving network data o Computing hydraulics at hour 0:00:00 o Transferring results to file0:00:00 o Writing output report to /Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/input.out ... EPANET completed.

Copy File Command: cp -f /Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/input.inp
/Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/caroline1-legacy.inp Return Code = 0
Copy File Command: cp -f /Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/input.bin
/Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/caroline1-legacy.bin Return Code = 0
Copy File Command: cp -f /Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/input.out
/Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/caroline1-legacy.out Return Code = 0
Copy File Command: rm -f /Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/input.inp Return Code = 0
Copy File Command: rm -f /Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/input.bin Return Code = 0
Copy File Command: rm -f /Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/input.out Return Code = 0

Figure 51: EPANETOL Interface Successful Simulation Notification

The output file from EPANETOL is below, Figure 52. As with the commercial implementations, the computational results are the same.[9]

[9] The common computational output is anticipated – all the machines are running the same underlying code. AccuWater is running on a Windows or Linux server, AquaVeo on a Windows or Linux server; EPANET On-Line is running on a macOS server; EPANET.DE was run on a macOS laptop; and the original example on a Windows 10 desktop.



```
Page 1                                    Fri Feb 16 11:04:42 2018

*****************************************************************
*                         E P A N E T                           *
*                  Hydraulic and Water Quality                  *
*                  Analysis for Pipe Networks                   *
*                        Version 2.00.12                        *
*****************************************************************


    Input Data File ..... /Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/input.inp
    Number of Junctions..... 4
    Number of Reservoirs..... 1
    Number of Tanks ..... 0
    Number of Pipes ..... 5
    Number of Pumps ..... 1
    Number of Valves ..... 0
    Headloss Formula ..... Darcy-Weisbach
    Hydraulic Timestep ..... 1.00 hrs
    Hydraulic Accuracy ..... 0.001000
    Status Check Frequency ..... 2
    Maximum Trials Checked ..... 10
    Damping Limit Threshold ..... 0.000000
    Maximum Trials ..... 40
    Quality Analysis ..... None
    Specific Gravity ..... 1.00
    Relative Kinematic Viscosity ..... 1.00
    Relative Chemical Diffusivity ..... 1.00
    Demand Multiplier ..... 1.00
    Total Duration ..... 0.00 hrs
    Reporting Criteria:
       All Nodes
       All Links

Analysis begun Fri Feb 16 11:04:42 2018


Hydraulic Status:
---------------------------------------------------------------------------
   0:00:00: Balancing the network:
            Trial  1: relative flow change = 0.452520
            Trial  2: relative flow change = 0.012780
            Trial  3: relative flow change = 0.001140
            Trial  4: relative flow change = 0.000100
   0:00:00: Balanced after 4 trials
   0:00:00: Reservoir 1 is emptying


Node Results:
-----------------------------------------------------
                Elevation    Demand      Head  Pressure
Node                   ft       cfs        ft       psi
-----------------------------------------------------
2                  100.00      0.00    180.00     34.66
3                  100.00      4.00    151.07     22.13
4                  100.00      3.00    152.27     22.65
5                  100.00      1.00    150.87     22.04
1                    0.00     -8.00      0.00      0.00  Reservoir


Link Results:
-----------------------------------------------
                    Flow  Velocity  Headloss
Link                 cfs       fps   /1000ft
-----------------------------------------------
2                   3.90     11.17     36.17
3                   0.27      0.76      0.29
4                   4.10     11.74     39.62
5                   0.73      2.10      1.75
6                   0.37      1.86      2.00
1                   8.00      0.00   -180.00  Pump

Analysis ended Fri Feb 16 11:04:42 2018
```

Figure 52: EPANETOL Interface Output File

Implementation Details

Implementation details describes EPANETOL's underlying server codes and explains how the processes interact. EPANET On-Line uses a hybrid method combining elements from HTML, JavaScript, PHP, and Python Scripts. Because EPANETOL incorporated additional elements beyond those in SolidsInRivers and NewANTS, the naming convention for the modules and their coding language is slightly different. The HTML script, hereafter known as "Entry", is used to introduce the program and help the user navigate to the correct location for their needs. Next, the "Authenticate" script (coded in PHP) requires the user to login with a username and password. Upon successful login, the client passes to the "Uploader" which is also a PHP script. The Uploader receives the filename and file contents and then saves the file to the server. The next step in the process is the "Launcher" script, coded using Python, which begins processing and executing the simulation. Finally, the "Responder" Python script builds the output for transmission to the "Results" HTML page. A flow chart depicting the interaction between these tools is shown below in Figure 53. This section contains relevant portions of the server codes for demonstration purposes. The full scripts can be found in Appendices G-L.

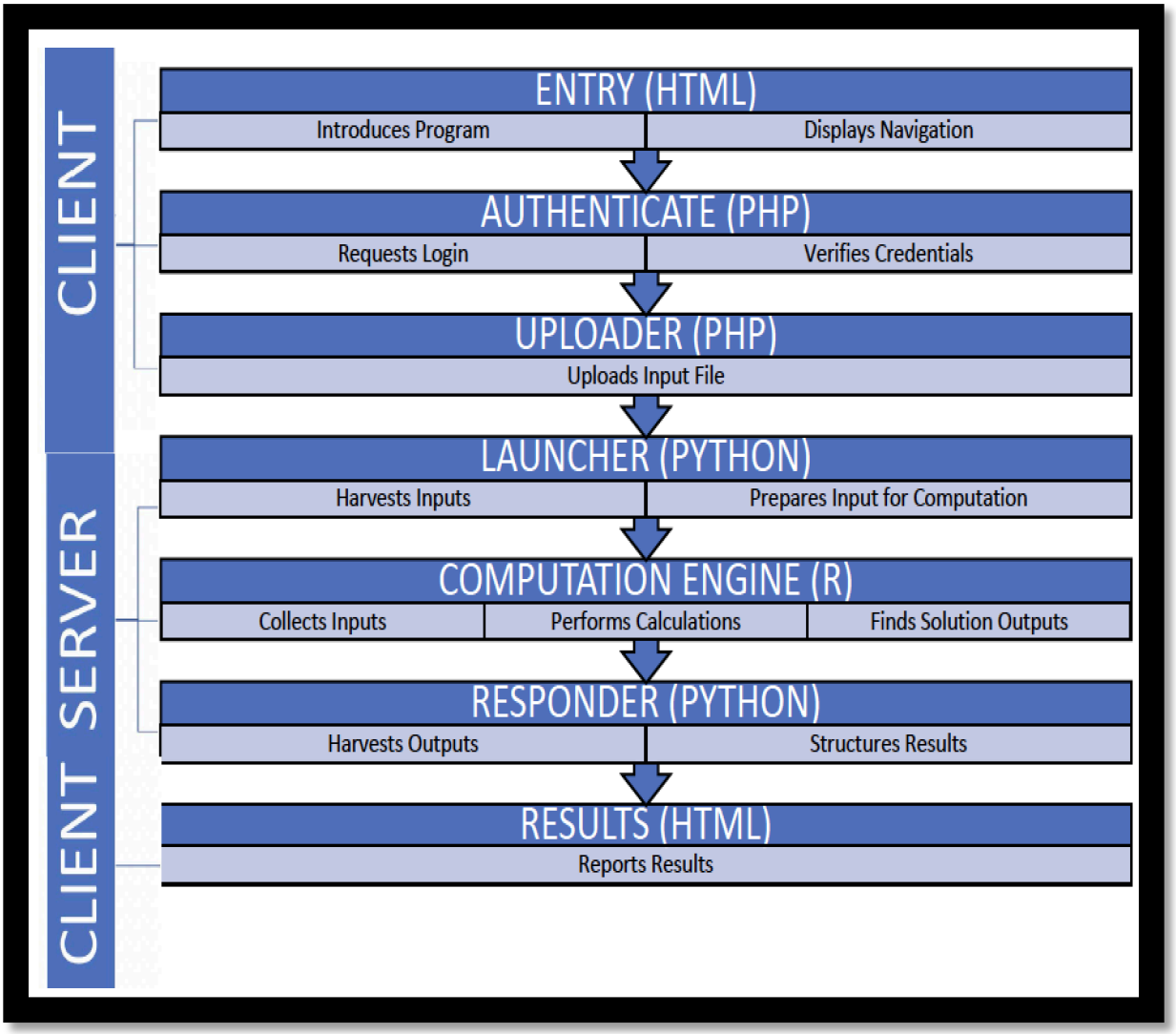


Figure 53: EPANETOL Inter-Process Communication Flow Chart

The client accesses the program via a web browser. The Entry module is an ordinary HTML script. Figure 54 is a screen capture of the relevant portions of the Entry script. The figure shows the lines of code that link to the Authenticate and Uploader scripts for

uploading the input file (line 134) and the Launcher script for running EPANETOL (line 142). The commercial examples presented previously use similar constructs, with some additional stylistic modification to improve the appearance of the interface.

```
126 <h2>EPANET On-Line Project Manager:</h2>
127
128     <p>
129     <a href="./working_files/">EPANET2 On-Line I/O Files </a>
130     Displays Current EPANET On-Line Documents (FTP-type listing in browser)
131     </p>
132
133     <p>
134     <a href="./login2.php">Upload A New EPANET Input File </a>
135     Upload an EPANET Input File.
136     The uploader will OVERWRITE CLOBBERS existing files if filenames are the same. <br><br>
137     The ERROR handler is primitave -- if upload succeeds, you will get a message saying so;
138     If the upload fails you may get a 404 Error, or just plain nothing.
139     </p>
140
141     <p>
142     <a href="./epanetOnLine.html">Run a EPANET On-Line Model </a>
143     This will open an web-page where you supply input filename (already uploaded) and a desired output
            filename.<br><br>
144     When you click SUBMIT, the server will attempt to run a EPANET model and produce an output file.
            During a model run, the server may be unresponsive, when the run is complete you can download the
            output file.
145     </p>
146
147     <p>
148     <a href="./web2net/web2net.html"> Web Interface to Build Input Files </a>
149     This is EXPERIMENTAl JavaScript to generate an EPANET .INP file for server-side processing.
150     The script is built from a tool originally developed by www.zonus.com (and cited).
151     The developer stopped and worked on the GUI and added ability to use Maps from the web, recompiled,
            and distributed this tool.  A copy is availabel elsewhere on this website.
152     </p>
```

Figure 54: EPANETOL Interface Entry (HTML Script)

Figure 55 is a screen capture of the relevant portion of the Authenticate script that is coded in PHP and used to allow a client to pass to the Uploader form. The Authenticate module requires an input user name and password.

```
This page is an authentication page for EPANET2 Input file uploads.  <br><br>
You have reached this page because you wish to upload a new or overwrite an existing EPANET2 project file.
<br><br>
If you are not intending to upload, then use the back button on your browser or take an escape route (below).
</p>

<h2>Escape Routes:</h2>
<p><a href="http://theodores-pro.ttu.edu/ncimm/">theodores-pro.ttu.edu (ncimm page)</a></p>

</div>
<!END LEFT COLUMN>

<!BEGIN RIGHT COLUMN>
<div class="content-column-right">
<h2>Authentication :</h2>

Enter your username and password (which you should have received already) below.
Upon sucessful authentication, you will be taken to the uploader page.
If authentication fails, you should get a rejection message.
If rejected, clear the from and try again (if it rejects several times, you may have a spelling error, or incor
    credentials).  The authentication is CASE-SENSITIVE.

<form method="post" action="myepanet2upload.php">
   <p>
   Enter your username:
   <input type="text" name="user">
   </p>

   <p>
   Enter your password:
   <input type="password" name="pass">
   </p>

   <p>
   <input type="submit" name="Submit" value="Submit">
   </p>

   <p>
   <input type="submit" name="Submit2" value="Clear Form">
   </p>

</form>
```

Figure 55: EPANETOL Interface Authenticate (PHP Script)

Upon choosing "submit", the form passes the character streams (username and password) to the formal authentication process encoded into the script on Figure 56 beginning on line 9.

The formal authentication process involves several important steps – the first is actual authentication. To maintain the connection, the form sets a session variable so the authentication is only needed once per session. The form obtains the username and password using a POST request rather than a GET request. The POST request is preferred to avoid unintentional system commands from being sent as part of the URL (a buffer overloading technique to compromise the remote server).

myepanet2upload.php

myepanet2upload.php 〉 No Selection

```php
<?php
session_start();
$_SESSION['username'] = $_POST['user'];
$_SESSION['userpass'] = $_POST['pass'];
$_SESSION['authuser'] = 0;

// check username and password
// Use "Include" to build multi-user database
if( ($_SESSION['username'] == 'EPANET' ) and ($_SESSION['userpass'] == 'On-Line') )
     {$_SESSION['authuser'] = 1;}
else
     {echo "Sorry, you do not have permission to view this page <br>
Use browser BACK arrow to try again (or escape)
 "; exit();}
?>

<!HTML section below>
<html>
<head>
<title>Upload files to shared file space</title>
</head>
<! File upload page -- need to add session variables for authentication>
<body>
<p>File Upload Page for EPANET2 On-Line Computation Engine </p>

<! TGC 2009-02-15 == Script adapted from http://www.tizag.com/php/phpT/fileupload.php >
<! TGC 2009-02-15 == HTML renders properly.  Local (client) browse enabled.>



<!============================>
<! set form type, identify action, identify data transfer method >
<form enctype="multipart/form-data" action="uploader.php" method="POST">
<!============================>

<!============================>
<! Set max file size to 100KB, change when working, php will error trap to prevent user defeat>
<input type="hidden" name="MAX_FILE_SIZE" value="10000000"/>
<!============================>

<!============================>
<!select file, should allow client to browse local disks to locate PATH+FILENAME>
Select Upload File: <input name="uploadedfile" type="file" />
<br />
<br />
<!============================>

<!============================>
<! Send form to server >
<input type="submit" value="Upload File" />
</form>
<!============================>


</body>
</html>
```

Figure 56. EPANETOL Interface Authenticate Continued (PHP Script)

Assuming the values sent are valid, the session transfers control to another PHP script, the Uploader, which accepts the file name from the client and accompanying contents. The portion of the Uploader script that receives the filename, file contents, and creates and saves the file on the server is shown in Figure 57, line 10.

```
10 $target_path ="/Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/";
11 print_r($target_path);
12 //============== Add original file name to target path ==========
13 //============== Result is : "uploads/filename.extension ========
14 $target_path = $target_path . basename( $_FILES['uploadedfile']['name']);
15 print_r($target_path);
16 //=========== Attempt to move the file to target location =======
17 // "move_uploaded_file" is PHP function to move file from temp locale to destination
18 //                      the function also returns logical values: TRUE = success
19 if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'],$target_path))
20 // TRUE
21 { echo "The file " . basename( $_FILES['uploadedfile']['name']) . " is uploaded." ;}
22 // FALSE
23 else
24 { echo "There was an error uploading the file.";}
```

Figure 57: EPANETOL Interface Uploader (PHP Script)

Upon successful upload, the Uploader script reports success and provides the client the ability to check that the file exists remotely.

The portion of the Uploader that performs these activities is shown in Figure 58, line 34. A failed upload will return a URL 404 error message. Upon successful upload, the client can either view the file directory or return to the switchboard page to select other actions.

```
31 //========== DEBUG SECTION ===========
32 // Return variable contents
33 echo "<br>";
34 echo "File should be located at: <br>";
35 print("/theodores-pro.ttu.edu/ncimm/EPANET2OnLine/working_files/" .
       basename($_FILES['uploadedfile']['name']));
36 //echo "<br>";
```

Figure 58: EPANETOL Interface Uploader Continued (PHP Script)

The next action would be to run the Launcher script. To run the simulation, the client uses the Launcher interface page that was depicted in Figure 50.

The Launcher script quite simply posts a single character stream, the file name prefix, to a Python script that resides in the CGI-BIN directory of the web server. When the Launcher script begins,

control is passed from the web application to the actual server and the processing begins. The relevant portion of the Launcher script is shown below in Figure 59. Line 16 is the command that requests the input file name from the user so that the program can begin the simulation.

```
10 Examples:  My_EPANET_File_Name <br><br>
11 </p>
12
13 <form method ="POST"
14       action = "http://theodores-pro.ttu.edu/cgi-bin/EPANET2OnLine/
             EPANET2OnLine.py">
15
16 Enter Value for Input File Name (prefix only, don't include the .inp'):
       <br/>
17 <input type = "text" name = "input_file_name"><br/>
18
19
20 <input type = "submit">
21 </form>
22 </body>
23 </html>
```

Figure 59: EPANETOL Interface Launcher (Python Script)

The Launcher script performs several tasks as it runs. Its first task is to obtain the input file name from the web interface. Figure 60 is a screen capture of the first part of the Launcher script that imports the modules to interface with the web server and generates/executes system commands beginning with line 9.

The Launcher obtains the file name from the CGI field storage variable. It assumes the file exists in the absolute path where the Uploader places files. Next, the return code is set to a value of 1 (the fail value). The Launcher then generates a system command to copy the input file to a file named "input.inp", line 26. Next, the Launcher sends the command to the operating system and recovers the return code. Regardless of the value of the code the Launcher continues, but the value is reported to the user to identify if different parts of the process failed. For instance, if the code returns fail, then the user will know that there was an issue with the file upload or that an incorrect file name was input when they started the program launch.

```
#!/usr/bin/python
# EPANET2OnLine.py
# On theodores-pro.ttu.edu
# Launches an instance of EPANET in CGI-BIN, waits for completion, and exits
# Use HMTL POST method
# Use python language

# Import modules for CGI handling
import cgi, cgitb , time
# Import subprocess module for handling system calls
import subprocess

# Create instance of FieldStorage
form = cgi.FieldStorage()
# Get inputs from fields
input_file_name = str(form.getvalue('input_file_name'))
# Dammit -- need the absolute path to the files!
absolute_path_1 = "/Library/WebServer/Documents/ncimm/EPANET2OnLine/working_files/"
# Generate a system command to change the filename
#  This is a security precaution to prevent accidental malicious code run from user-supplied filename
#  Set return code to 1 (Fail)
return_code_1 = 1
#
# Build the system command as a string -- use absolute path to a writeable location
#
commando1 = "cp " + absolute_path_1 + input_file_name +".inp " + absolute_path_1+ "input.inp "
#
# Issue the system command -- in this case only want a return code -- stdout irrelevant
# Here we copy the input file to input.inp (a default, always used input name)
#
return_code_1 = subprocess.call(commando1, shell = True)
################################################################
```

Figure 60: EPANETOL Interface Launcher Continued (Python Script)

The next part of the process is the Computation Engine Python script which actually runs EPANETOL. Again, a return code is set to fail, then the command literal is constructed. The command is

launched using the subprocess function in the Computation Engine. Subprocess essentially sends the command to the operating system with correct permissions to run an executable file. The output is captured (and later reported to the client) as is the return code. As before, if it remains in the fail state, the user will know that the file existed but did not run. More importantly, the capture and reporting of the standard output and standard error from the operating system will contain any warnings or errors that EPANETOL sends to the operating system. Figure 61 is a screen capture of the Computation Engine script that executes EPANETOL, and the subprocess command is located on line 46.

```
33 ##################################################################
34 #  Set return code to 1 (Fail)
35 return_code_2 = 1
36 #
37 # Build the system command to run an instance of SWMM command line
38 #   modify because epanet needs to run in a writeable location
39 #
40 commando2 = "./epanet2 " + absolute_path_1 + "input.inp " + absolute_path_1 + "input.out " + absolute_path_1 + "input.bin
41 #
42 # Issue the system command -- in this case only want to capture stdout
43 # Here we run SWMM, and PIPE stdout to the swmmprocess object
44 # We also get the return code from the swmmprocess object
45 #
46 swmmprocess = subprocess.Popen(commando2, stdout=subprocess.PIPE, shell = True)  # run SWMM
47 output, err = swmmprocess.communicate() # capture output from stdout (command line)
48 return_code_2 = swmmprocess.returncode # capture the return code
49 ##################################################################
```

Figure 61: EPANETOL Interface Computation Engine (Python Script)

This section of the Computation Engine bears some further study. First, if the process runs for too long, the web connection will drop. In a browser-only environment this would be a problem, however, the program is being run independently of the browser. The EPANETOL process will run until completion and the Computation Engine script has already built the system call to direct the output to the correct location. Thus, long program runs can be accommodated even when the connection is dropped.

The most important component of this portion if the Computation Engine is the line:

```
commando2 = "./epanet2 " + absolute_path_1 +
"input.inp " + absolute_path_1 + "input.out
" + absolute_path_1 + "input.bin"
```

As different versions of EPANETOL are produced, the server administrator only needs to compile the newer version, test that it works, place an executable copy in the CGI-BIN directory on the web server, and change the name of the executable in the line to the new name – for example, "`./epanet3 "`. The remainder of the web interface and supporting files ***do not need to be changed***. So, as new models are developed, the upgrades only need to be on

the server and do not need to propagate to the 60,000+ copies that have been downloaded.

Output Scripts

The next module is the Responder script. The Responder is a Python script that builds output for the HTML page (the Results script), assuming the program run is short enough for the connection to remain valid (on the demonstration server, the connection length is set to 5 minutes). Figure 62 is the code fragment that renames the files from the generic "input.inp" back to the user supplied values (line 56) again using the return codes to detect errors.

```
#################################################################
#
# Now move the files back to the user supplied names
#
binary_file_name = input_file_name + ".bin"
ascii_file_name = input_file_name + ".out"
commando3 = "cp -f " + absolute_path_1 + "input.inp "  + absolute_path_1 + input_file_name +".inp "
commando4 = "cp -f " + absolute_path_1 + "input.bin "  + absolute_path_1 + binary_file_name
commando5 = "cp -f " + absolute_path_1 + "input.out "  + absolute_path_1 + ascii_file_name
#
# Set return codes to 1 (Fail)
#
return_code_3 = 1
return_code_4 = 1
return_code_5 = 1
#
# Issue the system commands
#
return_code_3 = subprocess.call(commando3, shell = True)
return_code_4 = subprocess.call(commando4, shell = True)
return_code_5 = subprocess.call(commando5, shell = True)
#
# Now clean up the directory
#
commando6 = "rm -f " + absolute_path_1 + "input.inp "
commando7 = "rm -f " + absolute_path_1 + "input.bin "
commando8 = "rm -f " + absolute_path_1 + "input.out "
#
# Set return codes to 1 (Fail)
#
return_code_6 = 1
return_code_7 = 1
return_code_8 = 1
#
# Issue the system commands
#
return_code_6 = subprocess.call(commando6, shell = True)
return_code_7 = subprocess.call(commando7, shell = True)
return_code_8 = subprocess.call(commando8, shell = True)

#################################################################
```

Figure 62: EPANETOL Interface Responder (Python Script)

Figure 63 is a screenshot of the Responder Python script fragment that constructs the report page for the Results HTML script,

beginning on line 98. The output from this script was displayed previously in the example as Figure 52. This is the final step of the EPANETOL process.

```
91  ############################################################
92
93  # Prepare the output HTML
94  now = time.strftime("%c")
95
96  print "Content-type:text/html\r\n\r\n"
97  # should have two returns and line feeds
98  print "<html>"
99  print "<head>"
100 print "<title> EPANET2 On-Line (Server Side Procesing) </title>"
101 print "</head>"
102 print "<body>"
103 print " EPANET2 On-Line (Server-Side Processing) <br/><br/> "
104 print "Run Date : " , now ," : on theodores-pro.ttu.edu <br/> "
105 print "<hr/><hr/>"
106 print "------ INPUT VALUES ------ <br/> "
107 print "        Input File Name = ", input_file_name + ".inp", " <br/>"
108 print "    Binary Output File  = ", binary_file_name,"  <br/>"
109 print "      ASCII Output File = ", ascii_file_name, "  <br/>"
110 print "<hr/>"
111 print "------ PROCESS VALUES ----- <br/> "
112 print "   Copy File Command: ", commando1 ," Return Code = ", return_code_1, "<br/> "
113 print "    Run SWMM Command: ", commando2 ," Return Code = ", return_code_2, "<br/> "
114 print "<p>"
115 print " stdout from SWMM: ", output, "<br/>"
116 print "</p>"
117 print "   Copy File Command: ", commando3 ," Return Code = ", return_code_3, "<br/> "
118 print "   Copy File Command: ", commando4 ," Return Code = ", return_code_4, "<br/> "
119 print "   Copy File Command: ", commando5 ," Return Code = ", return_code_5, "<br/> "
120 print "   Copy File Command: ", commando6 ," Return Code = ", return_code_6, "<br/> "
121 print "   Copy File Command: ", commando7 ," Return Code = ", return_code_7, "<br/> "
122 print "   Copy File Command: ", commando8 ," Return Code = ", return_code_8, "<br/> "
123 print "<hr/><hr/>"
124 print " Links to Files "
125 followLink = "http://theodores-pro.ttu.edu/ncimm/EPANET2OnLine/working_files/" + ascii_file_name
126 print followLink
127 print "<p><a href=",followLink,"> EPANET2 Output File(s) </a> Link to EPANET output files </p>"
128 print "</body>"
129 print "</html>"
130
131 # end of script
```

Figure 63: EPANETOL Interface Responder Continued (Python Script)

Conclusions

Documented herein are the requisite HTML, R, and Python codes needed to run EPANET2 through a modern browser. As an extension of SolidsInRivers and NewANTS, EPANETOL documented the added server configuration to allow the end-user to upload an EPANET input file (US EPA, 2018) to the web server, perform a simulation, and recover the results.

EPANETOL has successfully added verification of user credentials as an extension of the concepts discussed in the first two chapters. Authentication is necessary to ensure the security of the server when users are able to (and required to) upload files as part of the procedure for using the system. In addition, EPANETOL utilizes remote storage of files, a successful extension of the concepts developed in SolidsInRivers and NewANTS. EPANETOL requires files to be stored on the remote server for client retrieval or future access. Direct modification of the source code is not desired in the case of EPANETOL, as opposed to SolidsInRivers and NewANTS, and this was avoided by specific utilization of a server-side-based modeling approach. In the event of updates or modification to EPANETOL, the server administrator would only need to place an executable copy of the new version on the server

and change a single instance of the file name in the Computation Engine script that is responsible for executing EPANETOL. As mentioned in the beginning of this chapter. There are over 60,000 downloads per year for the current version of EPANET. There is the potential for EPANETOL to only require one single model upgrade on the server as opposed to having to propagate to the 60,000 instances of EPANET on each individual computer—a 99.9983% decrease from the existing model.

EPANETOL was directly compared to existing commercial enterprises that offer some version of server-side EPANET modeling. The results from an example network modeling problem simulated in AccuWater, EPANET.DE, EPANET 2, and EPANETOL all have identical output reports (node and link results), confirming the accuracy of EPANETOL model simulation.

EPANETOL shares some limitations of currently available commercial enterprises. One notable drawback of these existing EPANET server-side models is that the interface does not appear to be able to add components to an existing model. To edit a model, the user would have to download the input file, make changes, then upload the file back to the server.

While remote users can view and run a model and change values of existing components of a network. At present, network editing is not available, and the programs are essentially intended for use with network model files prepared elsewhere. This limitation applies not only to EPANETOL, but also to Aquaveo, AccuWater, and EPANET.DE.

Future Work

The implementation presented demonstrates an on-line application that performs like commercial offerings and establishes the necessary code structure to build and implement on a user-controlled web server.

While the program performs as intended, some areas that could be expanded upon include the following:

- Modify the script to intentionally disconnect during long simulation runs and message the user when the program run is complete.
- Develop and deploy an interface that runs in a web-browser environment on the user's machine so the client can construct the input files to upload to the remote server.

    - EPANET.DE would be a starting point because the authors of that script have already worked out how to run and edit on the client machine – what remains is the construction of a fully functional GUI.
- Modify the web structure to generate user-lever subdirectories so that input file management is less complex.
- Modify the interface so a drag-and-drop structure can be used for selecting files to run (or generate a pull-down menu of input files).

CHAPTER 5

CONCLUSION

Introduction

The goal of this research was to communicate the development and methodology behind the deployment of hydrologic modeling tools into a web-based environment. The following section gives an overview of the purpose and use of the three simulation tools described previously. A summary of the innovations found during development of SolidsInRivers, NewANTS, and EPANETOL and their fundamental intellectual contribution to the field of study is presented, followed by recommendations and areas for future improvements.

Modeling Programs

The fundamental motivation for making these hydrologic modeling tools available via a web interface is to increase the availability and accessibility of modeling software. A secondary objective was to investigate requirements for implementing a server-side

processing system. The SolidsInRivers database search tool is designed to provide sediment transport estimates based upon previously gathered data from both laboratory experiments and field data. While NewANTS is used to provide the user with calculated approximations of contamination levels in groundwater, it also seeks to inform the user about the different modeling types that can be simulated using the program through a user-friendly interface. It is intended to be a source of information that attempts to explain complicated transport modeling geometries in a simple, straightforward way. Both SolidsInRivers and NewANTS are designed to be used in an academic setting as a way to verify results calculated through other means. EPANETOL is intended for use by those who need to perform water distribution system analysis but do not wish to invest in the expensive and time-consuming process of obtaining commercially available software. EPANETOL is designed to be applicable to the consumer as opposed to researchers/students like the first two tools. EPANETOL represents the first steps toward creating a fully web browser enabled version of EPANETOL. While that goal is still in the distant future, the process of developing EPANETOL was useful in that it taught us how to require authentication and store files on a remote server for later use.

Some common processes appeared throughout the course of this research. This led to the understanding that server-side processing (in this context) will typically follow the same basic pattern. There is a required process that first presents to the user what needs to be supplied as input, a "translation" from the input data into a usable form for the computation engine to calculate, and translation back to the interface and reporting results to the user. These processes themselves communicate with each other as part of the function of the modeling tools.

After the realization that server-side processing is based upon a generalized procedure type, it became even more important to document the modules' interaction in detail. The knowledge gained from each tool was used to improve on the concepts of the next tool. These three formed a natural progression from a simple database query to a program that mimics actual simulation technology that is used by practicing engineers.

Objective Achievements

Four important objectives were met during the design of the modeling programs. The first success is merely that each tool functions properly. They all can be accessed via webpage,

communicate with the other module processes, compute results using input data, and effectively return the results to the user. Even had the actual calculations been inaccurate, this would still be considered an achievement simply because the fundamental theory behind server-side processing was successfully implemented. The next major triumph is that the results calculated using all three online modeling tools replicate results found using broadly accepted and widely used traditional methods. Repeated outputs that are near approximations of true solutions validate the accuracy and reliability of developing research. Another important success is the documentation of the process and methodology behind the development of these server-side water resources models. Documented herein is the minimal HTML, Python, JavaScript, and R configuration required to accomplish the Entry, Launcher, Computation Engine, Responder, and Results processes. The appendices included are an example of the documentation of the research findings. Anyone wishing to replicate or expand upon the concepts developed in this research could use the exact code files in the appendices to run their own models. If the next researcher did not wish to exactly duplicate the findings developed here, he/she could follow the procedure outlined by the figures depicting the relevant code fragments that perform the most important commands in the modelling process. The reference to

exact lines in the code figures that correspond to the commands specified by the text is an attempt to detail the methodology so that a novice-intermediate user could still follow the structure of the research and the modeling process. Additional documentation not included as part of this dissertation details both the full extent of the codes that make up the underlying computational models and the pitfalls and dead ends encountered when trying to determine an efficient model development procedure.

The final major accomplishment of the research is that even though, on occasion, it was difficult find successful methods to develop and deploy these tools running in a browser, the project was still completed and the tools function appropriately. The task of setting permissions and authenticating the credentials of an EPANETOL user is one such instance of a difficulty that arose during the course of this project. My team was of invaluable assistance during this time period. Another problem encountered was in trying to decide how to adequately convey to the NewANTS user how a transport model type corresponds to a physical system. Analytical contaminant transport is a complicated subject and describing transport model conditions in such a condensed manner was problematic at times. The knowledge gathering process necessary to overcome these two

issues was difficult and time consuming. Analytic solutions to contaminant transport models are not uncommon, however, the majority of model solutions were either not applicable to the NewANTS geometries or were missing important information regarding source type, source shape, and model constraints.

Contributions

Despite of the problems encountered, the overall body of work adds to the combined field of computer programming and civil engineering through several contributions. SolidsInRivers made use of previous laboratory and field research into data sets for bed load transportation estimates. The SolidsInRivers database of over 12,000 records represents the combined findings of roughly 191 authors over a period of 94 years. The SolidsInRivers database search engine adds to this incredible compilation of nearly a century's worth of engineering research by creating a tool that represents a fundamental change in the way that solids load transport is calculated. It represents a migration away from reliance on empirical data and instead focuses on the concept of predictive algorithms. In other words, it is based on the idea that if there is a large number of records for a certain property, the probability of a new SolidsInRivers simulation being similar to

existing data sets is high and using that assumption to generate new estimates. This is a reflection of the predictive algorithms used by large internet corporations, such as Amazon, to predict what a client will buy in the future based upon their previous purchases.

NewANTS is a restoration of a program called ANTS that was originally completed in 1998. Sometime in the intervening years, the ANTS system was irreparably damaged. As a result, the only documentation of the original system is one that focuses mostly on quantifying a user's ability to select transport conditions that correspond to a given model scenario. Because the original version did in fact exist at one point in time, but is now lost, there was a need for a dedicated effort towards restoring and improving upon the original concept of an online analytical contaminant transport simulation. The absence of source codes or virtually any other type of supporting documentation related to the development of the ANTS models made it apparent that while the actual modeling program NewANTS is useful and could potentially help many engineers perform transport approximations, the more important part of the research would be the documentation describing the concepts behind the programs construction and implementation.

NewANTS attempts to take an extremely complex subject and present it to a client in an easy to understand, user-friendly manner. NewANTS can approximate solutions to 48 different contaminant transport geometries, however, there are a daunting number of combinations of dimensionality, source type, and attenuation type that can occur during a contamination event. The expectations of NewANTS is that it will be made up of an ever-increasing number of modeling scenarios as it is improbable to assume that it will sufficiently describe every single modeling scenarios possible any time in the near future.

Instead of trying to reinvent the wheel, EPANETOL makes use of the existing EPANET computation engine and improves upon it by making the modeling program accessible via a web browser. Since EPANETOL is based on a single server, it paves the way for easy transitions to anticipated future advances such as parallel processing capabilities and the future implementation of a fully browser enabled version of EPANET. The concept of housing the Graphical Processing Unit on a remote server intriguing because it would completely eliminate the responsibility of the client to install and maintain EPANET related software on their own machine. The advantages of this concept also extend to EPANET training courses. A user could be easily walked through training

because the most time-consuming element of typical training seminars stems from instructors having to configure software that is compatible with and working properly on the trainee's personal computer. Use of a remote server to allow modeling via a web browser would increase the availability of hydrologic modeling tools. Smaller communities and infrequent users could access the programs without having to purchase and maintain the software on their own. Individuals could work remotely if necessary as long as they have access to a web browser. The current EPANETOL software is even fully accessible and functional when used on a mobile device. This caters to the culture of immediate and on-the-go access to most things at any given time via a cell phone.

Another innovative quality of the hydrologic modeling tools presented is the way that, even the preliminary research planning stages, technological advancements are expected. Web browsers will most certainly be updated in the near future, so it is imperative that the modeling tools be flexible enough to handle new developments as they arise. Since each of the simulation programs is made up of (at minimum) five separate process modules, upgrades will likely only be required on a single process at any given time. It is also assumed that graphics rendering technology will be updated regularly, so the tools are designed to

accommodate new formats and/or coding structures as they become available. Finally, due to their web-based nature, these modeling tools have overcome the ever-present Windows vs. Linux vs. Macintosh issues that plague users when software is incompatible with the different operating systems.

Recommendations/Future Improvements

Recommendations for future developments include the following

- Additional features to improve the functionality of each tool such as: An increased number of database records for SolidsInRivers, additional contaminant transport modeling geometries for NewANTS, and improved aspects of EPANETOL, for example the ability to create and edit networks within the remote modeling system.
- A tutorial designed to help users select accurate transport models in NewANTS
- Improved stylistic elements for SolidsInRivers
- Deployment of an EPANETOL interface using GPUs that are housed on a remote server as opposed to the client's machine. Relocating the GPU to a server-side processor would allow the client to create input files in a web-based

environment and then upload them to the remote server using nothing more than an internet browser.

Conclusion

The deployment of these three hydrologic modeling tools in a web-based environment represented the first steps toward the accumulation of knowledge necessary to achieve the eventual goal of a fully web browser enabled version of EPANET. With each modeling program, the research team discovered more about what will be required in order to reach this lofty goal. From the basic SolidsInRivers database search engine to the increased graphical output in NewANTS and finally to the password protected EPANETOL utilizing server-side storage of files, each modeling tool brought the team one step closer to an EPANET version that is accessible by anyone, anywhere, at any time. These three chapters have each been submitted to peer-reviewed journals in an attempt to increase visibility and interest in this and other related projects that lie at the intersection of Engineering and Technology. The hope is that, with publication and additional circulation, the works described herein will be both useful in practical application and developed further so as to increase the tools' utility and applicability to the engineering community.

REFERENCES: SOLIDS IN RIVERS

Barnes, W. J. 2013. Experimental Study of Sediment Transport and Culvert Capacity. Ph. D. Dissertation, Texas Tech University.

Meyer-Peter, E., and Müller, R., (1948) Formulas for bed load transport, proceedings. The Second Meeting of the International Association for Hydraulic Structures Research, Stockholm, pp. 39–64.

Neill, C.R., and M. Yalin, (1969) Quantitative definition of beginning of bed movement.

Peterson, A.W. and Howells, R.F. (1973) A compendium of solids transport data for mobile boundary channels. Technical report, Inland Waters Directorate, Envrionment Canada.

Cleveland, T.G., Strom, K.B., Barnes, W. J., and Dixon, J. V. 2013. Hydraulic Performance of Staggered-Barrel Culverts for Stream Crossings. Texas Department of Transportation, Research Report 0-6549-1.

Dixon, J. 2011. A Relation Between Select Hydraulic Properties and Sediment Transport Volume Through Experimental Culvert Configurations and Techniques for Measuring Sediment Transport Volumes. Master's Thesis, Texas Tech University.

Eriksson, A. and A. van den Hengel (2010). Ecient computation of robust low-rank matrix approximations in the presence of missing data using the L1 norm. http: //acvtech. files. wordpress. com/ 2010/ 06/ robustl1_ eriksson. pdf .

King, J., W. Emmett, P. Whiting, R. Kenworthy, and J. Barry (2004). Sediment transportdata and related information for selected coarse-bed streams and rivers in idaho. GeneralTechnical Report RMRS-GTR-131, USDA Forest Service, Rocky Mountain ResearchStation, Fort Collins, CO.

Lee, K. T., Y.-L. Liu, and K.-H. Cheng (2004). Experimental investigation of bedloadtransport processes under unsteady ow conditions. Hydrological Processes 18, 2439{2454.

Peterson, A. (1975). Universal ow diagrams for mobile boundary channels. CanadianJournal of Civil Engineering 2, 549{557.

Recking, A. (2010). A comparison between ume and eld bed load transport data andconsequences for surface-based bed load transport prediction. Water Resources Re-search 46 (3), W03518.

Recking, A., P. Frey, A. Paquier, P. Belleudy, and J. Y. Champagne (2008a). Bed-load transport flume experiments on steep slopes. Journal of Hydraulic Engineering 134 (9),1302{1310.

Recking, A., P. Frey, A. Paquier, P. Belleudy, and J. Y. Champagne (2008b). Feedback between bed load transport and ow resistance in gravel and cobble bed rivers. Water Resources Research 44 (5), W05412.

Schoklitsch, A. (1949). Berechnung der geschiebefracht. Wasser und Energiewirtschaft 1.

Wong, M. and G. Parker (2006). Reanalysis and correction of bed-load relation of meyerpeterand m□uller using their own database. Journal of Hydraulic Engineering 132 (11),1159{1168.

Docs.tibco.com. (2018). Normalization by Z-score. [online] Available at: https://docs.tibco.com/pub/spotfire/6.5.1/doc/html/norm/norm_z_score.htm [Accessed 6 Apr. 2018].

Wong, M., G. Parker, P. DeVries, T. M. Brown, and S. J. Burges (2007). Experiments ondispersion of tracer stones under lower-regime plane-bed equilibrium bed load transport.Water Resources Research 43, W03440.

Bathurst, J. C., W. H. Graf, and H. H. Cao (1987). Bed load discharge equations forsteep mountain rivers. In C. R. Thorne, J. C. Bathurst, and R. D. Hey (Eds.), Sediment

Brownlie, W. R. (1981). Prediction of ow depth and sediment discharge in open channels.

Wilcock, P. R., S. T. Kenworthy, and J. C. Crowe (2001), Experimental study of the transport of mixed sand and gravel, Water Resour. Res., 37, 3349-3358, doi:10.1029/2001WR000683.

REFERENCES: NEWANTS

Chuang, L. (1998). *A Guidance System for Choosing Analytical Contaminant Transport Models*.

Clu-in.org. (2018). *CLU-IN | Technologies > What is HRSC?*. [online] Available at: https://clu-in.org/characterization/technologies/hrsc/hrscintro.cfm [Accessed 6 Apr. 2018].

Domenico, P. A. and V. V. Palciauskas (1982). "Alternative boundaries in solid waste management." Ground Water, Vol. 20, 303-311.

Domenico, P. and Robbins, G. (1985). A New Method of Contaminant Plume Analysis. *Ground Water*, 23(4), pp.476-485.

"EPANET 2 USER MANUAL." Innovyze - Innovating for Sustainable Infrastructure, www.innovyze.com/products/epanet/download/P1007WWU.

Epa.gov. (2018). [online] Available at: https://www.epa.gov/sites/production/files/2015-08/documents/mgwc-gwc1.pdf [Accessed 6 Apr. 2018].

Fetter, C. (2014). *Applied hydrogeology*.

Hähnlein, S., Molina-Giraldo, N., Blum, P., Bayer, P. and Grathwohl, P. (2010). Ausbreitung von Kältefahnen im Grundwasser bei Erdwärmesonden. *Grundwasser*, 15(2), pp.123-133.

Hydrogeologistswithoutborders.org. (2018). *Chapter 9: Groundwater Contamination | HWB*. [online] Available at: http://hydrogeologistswithoutborders.org/wordpress/1979-english/chapter-9/ [Accessed 6 Apr. 2018].

Ogata and Banks (1961). *A Solution of the Differential Equation of Longitudinal Dispersion in Porous Media.*.

Sauty, J. (1980). An analysis of hydrodispersive transfer in aquifers. *Water Resources Research*, 16(1), pp.145-158.

SCHIRMER, M. and BUTLER, B. (2004). Transport behaviour and natural attenuation of organic contaminants at spill sites. *Toxicology*, 205(3), pp.173-179.

Slideplayer.com. (2018). *Distance Measures Tan et al. From Chapter ppt video online download*. [online] Available at: http://slideplayer.com/slide/5139522/ [Accessed 6 Apr. 2018].

Slideplayer.com. (2018). *Unit 07 : Advanced Hydrogeology - ppt video online download*. [online] Available at: http://slideplayer.com/slide/4739996/ [Accessed 6 Apr. 2018].

Uddameri, V. (n.d.). CE 5364: Groundwater Transport Phenomena.

Yuan, D. (1995). *Accurate Approximations for One-, Two-, and Three-Dimensional Groundwater Mass Transport from an Exponentially Decaying Contaminant Source*. University of Houston.

REFERENCES: EPANETOL

AccuModel. (2018). *AccuWater - Confidence in Modeling | AccuModel*. [online] Available at: http://www.accu-model.com/ [Accessed 28 Feb. 2018].

Aquaveo.com. (2018). *CityWater - EPANET Web Solution | Aquaveo.com*. [online] Available at: https://www.aquaveo.com/software/citywater-introduction [Accessed 28 Feb. 2018].

Baseform. (2018). *Baseform*. [online] Available at: http://baseform.com/np4/185.html [Accessed 28 Feb. 2018].

Cleveland, T.G., and Neale, C.M. (2017). Explorations of the New EPANET Interface (EPANET-UI). Department of Civil and Environmental Engineering, Texas Tech University

Epanet.de. (2018). *epanet.js*. [online] Available at: http://epanet.de/js/index.html [Accessed 28 Feb. 2018].

GitHub. (2018). *OpenWaterAnalytics/EPANET*. [online] Available at: https://github.com/OpenWaterAnalytics/EPANET [Accessed 28 Feb. 2018].

GitHub. (2018). *USEPA/SWMM-EPANET_User_Interface*. [online] Available at: https://github.com/USEPA/SWMM-EPANET_User_Interface [Accessed 28 Feb. 2018].

Respec Inc. 2018. SWMM/EPANET "User Interface Reengineering." (In response to RFQ-DC-15-00153) http://www.respec.com/project/swmmepanet-user-interface-reengineering/

Studioars. (2018). *Urbano Hydra 9.0*. [online] Available at: http://www.studioars.com/en/urbano_9_hydra/96/2 [Accessed 28 Feb. 2018].

US EPA. (2018). *EPANET | US EPA*. [online] Available at: https://www.epa.gov/water-research/epanet [Accessed 28 Feb. 2018].

Zonums. (2018). *ZMaps: Net2Epa (Web to Epanet)*. [online] Available at: http://www.zonums.com/gmaps/net2epa.html [Accessed 28 Feb. 2018].

APPENDIX A

SOLIDS IN RIVERS ENTRY (HTML)

```
<!--
 #####SOLIDS IN RIVERS
 #####BY CAROLINE M. NEALE
 #####MODIFIED FROM T.G.CLEVELAND
 #####AUG2017
-->
<!DOCTYPE html PUBLIC >
<html><head><title>Solids in Rivers
Estimation Tool</title></head>
<link rel = "stylesheet" type = "text/css"
href = "styles.css" >
<body>
<h1> Solids in Rivers Estimator </h1>

<p><a href="./dummy"> Solids in Rivers
Manual </a> Author(s)  </p>

<!--<form method ="POST"
      action = "http://localhost/cgi-
bin/SolidsInRivers/SolidsInRivers.py">-->

<form method ="POST"
      action = "http://theodores-
pro.ttu.edu/cgi-
bin/SolidsInRivers/SolidsInRivers.py">

Enter Value for Slope (S): <br/>
```

```
<input type = "text" name = "mySlope"><br/>

Enter Value for Mean Particle Diameter
(D50): <br/>
<input type = "text" name = "myD50"><br/>

Enter Value for Discharge (Q): <br/>
<input type = "text" name = "myQ"><br/>

Enter Value for Mean Section Velocity (U):
<br/>
<input type = "text" name = "myW"><br/>

Enter Value for Exponent (n): <br/>
<input type = "text" name = "nxp"><br/>

Enter Value for Neighbor Count (N): <br/>
<input type = "text" name = "near"><br/>

<input type = "submit">
</form>
</body>
</html>
```

APPENDIX B

SOLIDS IN RIVERS LAUNCHER (PYTHON)

```
 #####SOLIDS IN RIVERS
 #####BY CAROLINE M. NEALE
 #####MODIFIED FROM T.G.CLEVELAND
 #####AUG2017

#!/usr/bin/python
# SolidsInRivers.py
# Uses HMTL POST method
# Computer Name(s): localhost on Caroline's
MacBook Pro 15
#                 : theodores-pro.ttu.edu on
Theodore's MacPro
import cgi, cgitb , time  # Import modules
for CGI handling
import subprocess  # Import subprocess
module for handling system calls
# Create instance of FieldStorage
form = cgi.FieldStorage()
# Get inputs from fields
mySlope = float(form.getvalue('mySlope'))
myD50 = float(form.getvalue('myD50'))
myQ = float(form.getvalue('myQ'))
myW = float(form.getvalue('myW'))
nxp = float(form.getvalue('nxp'))
near = float(form.getvalue('near'))
# Build the input file
```

```
ofile =
open("/Library/WebServer/Documents/OK2Write/
SolidsInRivers/input.dat","w") # open a file
to transfer input stream to R
ofile.write(repr(mySlope) + "\n")
ofile.write(repr(myD50)+ "\n")
ofile.write(repr(myQ)+ "\n")
ofile.write(repr(myW)+ "\n")
ofile.write(repr(nxp)+ "\n")
ofile.write(repr(near)+ "\n")
ofile.close()
# Build the system command
path_to_Rscript =
"/Library/Frameworks/R.framework/Versions/Cu
rrent/Resources/Rscript"
commando = path_to_Rscript + "
SolidsInRiversCGI-BINcopy.R"
return_code = 1  #  Set return code to 1
(Fail)
# Here we run SolidsInRiversCGI-BIN.R, and
PIPE stdout to the solidsInrivers object
solidsInrivers = subprocess.Popen(commando,
stdout=subprocess.PIPE, shell = True)  # run
process
output, err = solidsInrivers.communicate() #
capture output from stdout (command line)
return_code = solidsInrivers.returncode #
capture the return code
plotfile = ' <img src =
"/OK2Write/SolidsInRivers/plot.pdf" width=
"800" height = "500"> '
# Prepare the output HTML
now = time.strftime("%c")
```

```
print "Content-type:text/html\r\n"  # should
have two returns and line feeds
print "<html>"
print "<style> table, th, td {border: 1px
solid black;} </style>"
print "<head>"
print "<title> Solids in Rivers Estimate
</title>"
print "</head>"
print "<body>"
print "<p style =""""" "font-family:arial"
""" ">"
print "Solids in Rivers Estimate <br/> "
print "Run Date : " , now ," <br/> "
print "<table style=""""" "width:99%" """ ">"
print "<tr>"
print "<td>------ INPUT VALUES ------</td>
<td> </td> <td> </td>"
print "</tr>"
print "<tr> <td> Channel Slope (S) = </td>
<td> ", mySlope," </td> <td> Dimensionless
</td> </tr> "
print "<tr> <td> Mean Grain Diameter (D50) =
</td> <td> ", myD50 , " </td> <td> meters
</td> </tr> "
print "<tr> <td> Discharge (Q) = </td> <td>
", myQ , "</td> <td> cubic meters per second
</td> </tr> "
print "<tr> <td> Mean Velocity (U) = </td>
<td> ", myW, "</td> <td> meters per second
</td> </tr> "
print "<tr> <td> Exponent (n) = </td> <td>
", nxp, "</td> <td> n = 2 is Euclidean Norm
</td> </tr> "
```

```
print "<tr> <td> Nearest Neighbor Count (N)
= </td> <td> ", near, "</td> <td> Search
Count </td> </tr> "
#print "COMMAND TO RUN : ", commando,
"<br/>"
print "<tr> <td>------ COMPUTED RESULT -----
- </td> <td> </td> <td> </td> "
print "Return Code : ",return_code, " <br/>"
# split the output from R script
op1,op2 = output.split(" ")
print "<tr> <td> Estimated Solids Flux =
</td> <td> ", op1, "</td> <td> kg/m/sec
</td> </tr> "
print "<tr> <td> Estimated Solids
Concentration = </td> <td> ", op2, "</td>
<td> parts per hundred thousand </td> </tr>
"
print "</table><br/>"
print "Bar Plots of 5-Nearest Neighbor
Values, Input Values, and Estimates (based
on all search neighbors)"
print "<tr>"
print "<table style=""""" "width:99%" """ ">"
print "<td>"
print plotfile
print "</td>"
print "</tr>"
print "</p>"
print "</body>"
print "</html>"
# end of script
```

APPENDIX C

SOLIDS IN RIVERS COMPUTATION ENGINE (R)

```
 #####SOLIDS IN RIVERS
 #####BY CAROLINE M. NEALE
 #####MODIFIED FROM T.G.CLEVELAND
 #####AUG2017

# SolidsInRiverCGI-BIN
# CGI-BIN implementation for Solids in Rivers Paper
# uses distance-based search for estimate solids flux in
river
rm(list=ls()) # deallocate analysis environment
# prototype functions
# NONE
# read the database from the ASCII file
solids_in_rivers.db<-
read.csv("solids_in_rivers.csv",header=TRUE, sep=",")
DB<-solids_in_rivers.db # use a short name for building the
reduced database
# uses order function to search database for nearest
observations
# predictors: S_m_m, D50_m, Q_m3_m, U_m_s
# predicted : C_ppht, qb_kg_m_s
zz <-
file("/Library/WebServer/Documents/OK2Write/SolidsInRivers/
input.dat", "r") # Open a connection named zz to file named
input.dat
#zz <- file("input.txt", "r") # Open a connection named zz
to file named input.dat
# read the simulation conditons
mySlope <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn =
TRUE,encoding = "unknown", skipNul = FALSE))
myD50 <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn =
TRUE,encoding = "unknown", skipNul = FALSE))
myQ <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn =
TRUE,encoding = "unknown", skipNul = FALSE))
```

```
myU <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn =
TRUE,encoding = "unknown", skipNul = FALSE))
nxp <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn =
TRUE,encoding = "unknown", skipNul = FALSE))
near <-as.numeric(readLines(zz, n = 1, ok = TRUE, warn =
TRUE,encoding = "unknown", skipNul = FALSE))
close(zz)
######### COMPUTE Z-SCORE VALUES OF INPUTS
mSlope <- mean(DB$S_m_m,na.rm=TRUE)
sdSlope <- sd(DB$S_m_m,na.rm=TRUE)
mySlopeZ <- (mySlope-mSlope)/sdSlope
mD50 <- mean(DB$D50_m,na.rm=TRUE)
sdD50 <- sd(DB$D50_m,na.rm=TRUE)
myD50Z <- (myD50-mD50)/sdD50
########################################
mQ <- mean(DB$Q_m3_s,na.rm=TRUE)
sdQ <- sd(DB$Q_m3_s,na.rm=TRUE)
myQZ <- (myQ-mQ)/sdQ
mU <- mean(DB$U_m_s,na.rm=TRUE)
sdU <- sd(DB$U_m_s,na.rm=TRUE)
##mW <- mean(DB$W_m,na.rm=TRUE)
##sdW <- sd(DB$W_m,na.rm=TRUE)
myUZ <- (myU-mU)/sdU
######### BUILD Z-SCORE VECTORS OF PREDICTORS
zSlope <- scale(DB$S_m_m,center=TRUE,scale=TRUE)
zD50 <- scale(DB$D50_m,center=TRUE,scale=TRUE)
zQ <- scale(DB$Q_m3_s,center=TRUE,scale=TRUE)
zU <- scale(DB$U_m,center=TRUE,scale=TRUE)
########## BUILD THE DISTANCE VECTOR
zdist <- (zSlope-mySlopeZ)^nxp +
         (zD50 - myD50Z  )^nxp +
         (zQ   - myQZ    )^nxp +
         (zU   - myUZ    )^nxp
zdist <- zdist^(1/nxp)
zorder <- order(zdist,DB$qb_kg_m_s)
response <- DB$qb_kg_m_s[zorder]
otherResponse <- DB$C_ppht[zorder]
estimate <- mean(response[1:near])
estimate2 <- mean(otherResponse[1:near])
cat(estimate,estimate2)
####################################
# Build a graphic
# Simple Bar Plot
nsSlope <- DB$S_m_m[zorder]
nsD50   <- DB$D50_m[zorder]
```

```
nsQ     <- DB$Q_m3_s[zorder]
nsU     <- DB$U_m_s[zorder]
groupID <-
c(1,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,4,5,5,5,5,5)
Values <-
matrix(c(nsSlope[1:5],nsD50[1:5],nsQ[1:5],nsU[1:5],response
[1:5],otherResponse[1:5]),nrow=5,ncol=6,byrow=FALSE)
groupValue <-
c(nsSlope[1:5],nsD50[1:5],nsQ[1:5],nsU[1:5],response[1:5])
counts <- table(groupID,groupValue)
namesC <- c("1","2","3","4","5")
pdf("/Library/WebServer/Documents/OK2Write/SolidsInRivers/p
lot.pdf")
par(mfrow=c(1,6))
inputValue <- mySlope
barplot(c(Values[,1],inputValue),beside=TRUE,xlab="5 Near &
Input",main="Slope",ylab="meter/meter
(dimensionless)",col=c("cyan","cyan","cyan","cyan","cyan","
orange"))
inputValue <- myD50
barplot(c(Values[,2],inputValue),beside=TRUE,xlab="5 Near &
Input",main="Diameter",ylab="meters",col=c("cyan","cyan","c
yan","cyan","cyan","orange"))
inputValue <- myQ
barplot(c(Values[,3],inputValue),beside=TRUE,xlab="5 Near &
Input",main="Discharge",ylab="cubic
meters/second",col=c("cyan","cyan","cyan","cyan","cyan","or
ange"))
inputValue <- myU
barplot(c(Values[,4],inputValue),beside=TRUE,xlab="5 Near &
Input",main="Velocity",ylab="meters/second",col=c("cyan","c
yan","cyan","cyan","cyan","orange"))
#actualEstimate <- mean(Values[,5])
actualEstimate <- estimate
barplot(c(Values[,5],actualEstimate),beside=TRUE,xlab="5
Near & Estimate",main="Solids
Flux",col=c("cyan","cyan","cyan","cyan","cyan","magenta"),y
lab="kilogram/meter/second")
box("figure")
#actualEstimate <- mean(Values[,6])
actualEstimate <- estimate2
barplot(c(Values[,6],actualEstimate),beside=TRUE,xlab="5
Near & Estimate",main="Solids
Conc.",col=c("cyan","cyan","cyan","cyan","cyan","magenta"),
ylab="parts per hundred thousand")
```

```
box("figure")
#dev.off()# Disabled to prevent error code upon return to
webserver
```

APPENDIX D

NEWANTS ENTRY (HTML)

```
<!--
 #####1-DIMENSIONAL
 #####IMPULSE SOURCE
 #####ADVECTION-DISPERSION MODEL
 #####BY CAROLINE M. NEALE
 #####AUG25.2017
-->

<!DOCTYPE html PUBLIC >
<html><head><title>NewANTS </title></head>
<link rel = "stylesheet" type = "text/css" href = "styles.css" >
<body>
<h1> 1D - Advection-Dispersion Constant Reservoir Source </h1>

<img src = "./OgataBanks.gif" width = "500" height = "500" > <br/>


<p>Panel A is a depiction of the physical system and the concentration
profile along the x-axis at time less than zero. The concentration is
zero everywhere.</p>
<p>Panel B is a depection of the physical system and the concentration
profile along the x-axis at time equal zero (like the Big Bang!). At x
< 0, the concentration is suddenly raised to a value of Co everywhere
to the left of the origin, x=0.  This condition represents a step
function input and is a suitable approximation of some upstream source
zone that has a constant concentration.  The concentration to the
right of the origin (x > 0) is still zero.</p>
<p>Panel C is a depiction of the physical system and the concentration
profile aling the x-axis at some time greater than zero.  The source
mass has moved to the right of the origin a distance determined by the
species velocity and dispersed along the translational front
proportional to the dispersivity in the system. </p>
<p> The analytical solution (Ogata-Banks) is </p>
<img src = "./OgataBanksEquation.gif" width = "330" height = "50" >
</p>
<p>The solution is applicable for porous media flow, where the
velocity (below) is the mean section velocity (seepage velocity
divided by the porosity).  The solution can also be used with streams
and pipes (porosity = 1).</p>

<p><a href="#"> Detailed Explaination </a> (Under Construction) </p>


<!--<form method ="POST"
      action = "http://localhost/cgi-bin/NewANTS/1D-AD-OgataBanks/1D-
AD-OgataBanks.py">-->

<form method ="POST"
      action = "http://theodores-pro.ttu.edu/cgi-bin/NewANTS/1D-AD-
OgataBanks/1D-AD-OgataBanks.py">
```

```
Enter Value for Distance (x) : <br/>
<input type = "text" name = "distance"><br/>

Enter Value for Time (t) : <br/>
<input type = "text" name = "etime"><br/>

Enter Value for Source Concentration (Co) : <br/>
<input type = "text" name = "concentration"><br/>

Enter Value for Mean Section Velocity (V) : <br/>
<input type = "text" name = "velocity"><br/>

Enter Value for Dispersion Coefficient (D) : <br/>
<input type = "text" name = "dispersion"><br/>

<input type = "submit">
</form>
</body>
</html>
```

APPENDIX E

NEWANTS LAUNCHER (PYTHON)

```
#####1-DIMENSIONAL
#####IMPULSE SOURCE
#####ADVECTION-DISPERSION MODEL
#####BY CAROLINE M. NEALE
#####MODIFIED FROM T.G.CLEVELAND
#####AUG25.2017



#!/usr/bin/python
# 1D-AD-OgataBanks.py
# Uses HMTL POST method
# Computer Name(s): localhost on Caroline's
MacBook Pro 15
#                 : theodores-pro.ttu.edu on
Theodore's MacPro
import cgi, cgitb , time  # Import modules
for CGI handling
import subprocess  # Import subprocess
module for handling system calls
# Create instance of FieldStorage
form = cgi.FieldStorage()
# Get inputs from fields
velocity = float(form.getvalue('velocity'))
distance = float(form.getvalue('distance'))
dispersion =
float(form.getvalue('dispersion'))
```

```
etime = float(form.getvalue('etime'))
concentration =
float(form.getvalue('concentration'))
# Build the input file
ofile =
open("/Library/WebServer/Documents/OK2Write/
NewANTS/1D-AD-OgataBanks/input.dat","w") #
open a file to transfer input stream to R
ofile.write(repr(velocity) + "\n")
ofile.write(repr(distance)+ "\n")
ofile.write(repr(dispersion)+ "\n")
ofile.write(repr(etime)+ "\n")
ofile.write(repr(concentration)+ "\n")
ofile.close()
# Build the system command
path_to_Rscript =
"/Library/Frameworks/R.framework/Versions/Cu
rrent/Resources/Rscript"
commando = path_to_Rscript + " 1D-AD-
OgataBanks.R"
return_code = 1  #  Set return code to 1
(Fail)
# Here we run 1D-AD-OgataBanks, and PIPE
stdout to the 1D_AD_OgataBanks object
_1D_AD_OgataBanks =
subprocess.Popen(commando,
stdout=subprocess.PIPE, shell = True)  # run
process
output, err =
_1D_AD_OgataBanks.communicate() # capture
output from stdout (command line)
return_code = _1D_AD_OgataBanks.returncode #
capture the return code
# Here we capture the output and process
```

```
# need to read the output file
ofile =
open("/Library/WebServer/Documents/OK2Write/
NewANTS/1D-AD-OgataBanks/output.dat","r") #
open a file to recover output stream from R
# process the output stream from R script
xyz = [] # null list to capture each triple
how_many_lines = 0
for line in ofile:
    xyz.append([float(n) for n in
line.split(",")])
    how_many_lines += 1
ofile.close()
ncols = len(xyz[0]) # get number of columns,
should be 3
# Here we get the graphic
plotfile = ' <img src =
"/OK2Write/NewANTS/1D-AD-
OgataBanks/plot.pdf" width= "500" height =
"500"> '
# Prepare the output HTML
now = time.strftime("%c")
print "Content-type:text/html\r\n\r\n"  #
should have two returns and line feeds
print "<html>"
print "<style> table, th, td {border: 1px
solid black;} </style>"
print "<head>"
print "<title> 1D Advection-Dispersion
Ogata-Banks Solution </title>"
print "</head>"
print "<body>"
print "Concentration Profile <br/><br/> "
print "Run Date : " , now ," <br/> "
```

```
print "COMMAND TO RUN : ", commando, "<br/>
<br/>"
print "Return Code : ",return_code, " <br/>
<br/>"
########################################
print "<table style="""" "width:50%" """ ">"
print "<tr>"
print "<td>INPUT VALUES </td> <td> </td>
<td> </td>"
# alignment for ease of program maintenance,
extra spaces are ignored in program run
print "</tr>"
print "<tr> <td>                Distance (x)
= </td> <td> ",       distance, " </td> <td>
meters </td> </tr> "
print "<tr> <td>                    Time (t)
= </td> <td> ",         etime , " </td> <td>
days  </td> </tr> "
print "<tr> <td>  Source Concentration (Co)
= </td> <td> ",  concentration, " </td> <td>
ppm </td> </tr> "
print "<tr> <td>  Mean Section Velocity (V)
= </td> <td> ",       velocity, " </td> <td>
meters/day </td> </tr> "
print "<tr> <td> Dispersion Coefficient (D)
= </td> <td> ",     dispersion, " </td> <td>
meters^2 </td> </tr> "
print "</table><br/>"
########################################
print "<table style="""" "width:50%" """ ">"
print "<tr> <td>COMPUTED RESULT</td> <td>
</td> <td> </td> "
print "<tr> <td> Concentration at x,t =
</td> <td> ", output , "</td> <td> ppm </td>
```

```
</tr> "
print "</table><br/>"
########################################
print " TGraphic Goes HERE "
print plotfile
print " <br/>"
########################################
print "<table style="""" "width:50%" """ ">"
print "<tr><td>CONC. PROFILE </td> <td>
</td> <td> </td></tr>"
print "<tr><td>DIST. (METERS)</td> <td>TIME
(DAYS)</td> <td>CONC. (ppm)</td></tr>"
# write the results into a table
for i in range(0,how_many_lines,1):
    print "<tr><td>", xyz[i][0] ,"</td>
<td>", xyz[i][1] ,"</td> <td>", xyz[i][2]
,"</td></tr>"
print "</table>"
########################################
print "</body>"
print "</html>"
# end of script
```

APPENDIX F

NEWANTS COMPUTATION ENGINE (R)

```
#####1-DIMENSIONAL
#####IMPULSE SOURCE
#####ADVECTION-DISPERSION MODEL
#####BY CAROLINE M. NEALE
#####AUG25.2017



# analytical model advection-dispersion for CGI-BIN
rm(list=ls())
####### Disable for CGI-BIN ########
#this.dir <- dirname(parent.frame(2)$ofile)
#setwd(this.dir)
##########################################
# prototype special functions in generic R #
##########################################
# error function (real valued input/output)
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
# complimentary error function (real values input/output)
erfc <- function(x) 2 * pnorm(x * sqrt(2), lower = FALSE)
#############################################################################
# ogata-banks solution for 1D constant reservoir, advection-dispersion case #
#############################################################################
c1dad <- function(distance,time,dispersion,velocity,concentration){
# bear, dynamics of fluids in porous media, pg 630 eqn 10.6.22
# verified theodore g. cleveland 2017-0823
  arg1 <- (distance-velocity*time)/(2*sqrt(dispersion*time));
  arg2 <- (distance*velocity)/dispersion;
  arg3 <- (distance+velocity*time)/(2*sqrt(dispersion*time));
  if (arg3 > 27) {
      c1dad <- 0.5*concentration*(erfc(arg1)                    );
  }
  else {
      c1dad <- 0.5*concentration*(erfc(arg1)+exp(arg2)*erfc(arg3));
  }
return(c1dad) ;
}
# read from an input file
####### Disable for CGI-BIN #########
#zz <- file("./input.dat","r")
#####################################
```

```
####### Enable for CGI-BIN ##########
zz <- file("/Library/WebServer/Documents/OK2Write/NewANTS/1D-AD-OgataBanks/input.dat","r") # Open a connection to zz
#####################################
velocity <- as.numeric(readLines(zz, n=1, ok=TRUE, warn=TRUE,encoding="unknown",skipNul = FALSE))
distance <- as.numeric(readLines(zz, n=1, ok=TRUE, warn=TRUE,encoding="unknown",skipNul = FALSE))
dispersion <- as.numeric(readLines(zz, n=1, ok=TRUE, warn=TRUE,encoding="unknown",skipNul = FALSE))
time <- as.numeric(readLines(zz, n=1, ok=TRUE, warn=TRUE,encoding="unknown",skipNul = FALSE))
concentration <- as.numeric(readLines(zz, n=1, ok=TRUE, warn=TRUE,encoding="unknown",skipNul = FALSE))
close(zz)
######## To Do Error Traps ##########
# Input verify in HTML/Javascript   #
#####################################
# function
output <- c1dad(distance,time,dispersion,velocity,concentration)
cat(output)
####### profile output ##########
x <- numeric(0)
conc <- numeric(0)
xlow <- distance/10
xhigh <- distance*10
step <- ((xhigh-xlow)/100)
### profile vector output
outfile <- file("/Library/WebServer/Documents/OK2Write/NewANTS/1D-AD-OgataBanks/output.dat","w")
for (i in 1:199) {
   x[i] <- 0+i;
   conc[i] <- c1dad(x[i],time,dispersion,velocity,concentration);
   write(c(x[i],time,conc[i]), outfile ,sep = ",",ncolumns = 3)  #write to stdio
}
close(outfile)
##build chart
pdf("/Library/WebServer/Documents/OK2Write/NewANTS/1D-AD-OgataBanks/plot.pdf")
title_string <- paste("Concentration Profile at Time = ",time)
plot(x,conc,type="l",main=title_string,xlab="Distance",ylab="Concentration")


#print(cbind(x,time,conc))
#new <-print(output)
#cat(new)
 ####### history output ##########
# read from input file -- OK
# write to stdio -- OK
# write a graph to pdf
```

APPENDIX G

EPANETOL ENTRY (HTML)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
    <head>
     <title>EPANET2 On-Line Entry Page</title>
     <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
     <style type="text/css">
          body {
               background-color: #fff;
               color: #000;
               font-size: 0.9em;
               font-family: sans-serif,helvetica;
               margin: 0;
               padding: 0;
          }
          :link {
               color: #0000FF;
          }
          :visited {
               color: #0000FF;
          }
          a:hover {
               color: #3399FF;
          }
          h1 {
               text-align: center;
               margin: 0;
               padding: 0.6em 2em 0.4em;
               background-color: #0000FF;
               color: #ffffff;
               font-weight: normal;
               font-size: 1.75em;
               border-bottom: 2px solid #000;
          }
          h1 strong {
               font-weight: bold;
          }
          h2 {
               font-size: 1.1em;
               font-weight: bold;
          }
          .content {
               padding: 1em 5em;
          }
          .content-columns {
```

```
<! BEGIN LEFT COLUMN>
<div class="content-column-left">

<h2>About EPANET2 On-Line</h2>
<p>
EPANET2 On-Line is a server-side computation engine that reads
user supplied EPANET files {filename.inp},
instantiates a EPANET run, and returns an ASCII output file
{filename.out}.<br><br>

EPANET2 On-Line is an experimental tool to investiagte and
develop an entirely web-based EPANET toolkit that
may someday suppliment or replace end-user EPANET installations.
<br><br>
The value to the modeling community would be a single instance of
the EPANET model so that as future
versions are built, they too can be placed into an on-line
implementation so the end user is always
able to use the most current stable version of EPANET.  <br><br>
Older versions would remain available for validating historical
installations.
</p>
<p>
The current version uses a challenge-password to access the on-
line computation engine. <br><br>
This model is to limit the concurrent user counts until the
technology is developed enough to
have many concurrent users.  <br><br>The current system has a
file size limitation,
however EPANET input files are quite efficient and the limit
should not be an issue. </p>

<h2>Escape Routes:</h2>
   <p><a href="http://theodores-pro.ttu.edu/ncimm/">theodores-
pro.ttu.edu (ncimm page) </a></p>

   <h2>Maintenance Logs</h2>
   <p> Updated 2017-0329. </p>
   <p> Programmed php uploader 2017-0329.</p>

</div>
<!END LEFT COLUMN>

<!BEGIN RIGHT COLUMN>
<div class="content-column-right">

<h2>EPANET On-Line Project Manager:</h2>
```

```
<p>
   <a href="./working_files/">EPANET2 On-Line I/O Files </a>
   Displays Current EPANET On-Line Documents (FTP-type listing in
browser)
   </p>

   <p>
   <a href="./login2.php">Upload A New EPANET Input File </a>
   Upload an EPANET Input File.
   The uploader will OVERWRITE CLOBBERS existing files if
filenames are the same. <br><br>
   The ERROR handler is primitave -- if upload succeeds, you will
get a message saying so;
   If the upload fails you may get a 404 Error, or just plain
nothing.
   </p>

   <p>
   <a href="./epanetOnLine.html">Run a EPANET On-Line Model </a>
   This will open an web-page where you supply input filename
(already uploaded) and a desired output filename.<br><br>
   When you click SUBMIT, the server will attempt to run a EPANET
model and produce an output file.  During a model run, the server
may be unresponsive, when the run is complete you can download
the output file.
   </p>

   <p>
   <a href="./web2net/web2net.html"> Web Interface to Build Input
Files </a>
   This is EXPERIMENTAl JavaScript to generate an EPANET .INP
file for server-side processing.
   The script is built from a tool originally developed by
www.zonus.com (and cited).
   The developer stopped and worked on the GUI and added ability
to use Maps from the web, recompiled, and distributed this tool.
A copy is availabel elsewhere on this website.
   </p>


<h2>Wish List</h2>
<p> Create a subdirectory builder in working files to manage
individual projects </p>
<p> Improved ERROR handler for file upload/download </p>
<p> Run in nohup mode, send end user a message run is complete
for long runs </p>
</div>
<!END RIGHT COLUMN>
  </div>
```

```
 </div>
</div>

</div>
</body>
</html>
<!      <img src = "./EPANET2On-Line.gif"
width="300" height="300"><br>
```