WDSLib User's Manual

Mengning Qiu, Bradley Alexander, Sylvan Elhay, Angus R. Simpson

October 7, 2018

1 General Information

1.1 Toolkit Overview

WDSLib is an extensible simulation toolkit for the steady-state analysis of a water distribution system using different solution methods. Currently, a number of solution methods have been implemented: namely the forest-core partitioning algorithm (FCPA) (Simpson et al. 2014), the global gradient algorithm (GGA) (Todini and Pilati 1988), and the reformulated co-tree flow method (RCTM) (Elhay et al. 2014). WDSLib also enables these methods to be combined in various solution method configurations. WDSLib has been created using a modularized object-oriented design and implemented in the C++ programming language and has been validated against a reference MATLAB implementation. WDSLib has been demonstrated to be faster and more numerically stable than EPANET.

1.2 Purpose of this toolkit

Hydraulic simulation is an essential process for the design, operation and management of water distribution systems in industry and research. Currently, the most widely used WDS simulation toolkit is EPANET ((Rossman 2000)). Despite its popularity, EPANET has been reported in the literature to have a number of design flaws (such as incorrectly computing the head-loss derivatives (Simpson and Elhay 2010), numerical failure in the presence of the zero flows (Elhay and Simpson 2011) and the extensive use of global variables (Guidolin et al. 2010). In addition, a number of new solution methods, such as FCPA and RCTM, have also been demonstrated in the literature to be superior, or at least comparable, in terms of speed and numerical reliability when compared with the GGA, that is implemented in EPANET. It is important to note that the relative performance of these

solution methods in terms of speed, rate of convergence, and accuracy depends upon the topological properties of the target network including: the size of the forest component, the number of network loops, and the density of these network loops. For a given network, it is difficult, a priori, to evaluate the impact of these topological factors. Therefore, it is important to have a framework that allows users to easily experiment with different configurations of these algorithms.

The purpose of WDSLib is to allow a user to easily mix and interchange solution components. Users can then easily benchmark the performance of different solution methods on a smaller number of evaluations for a given network and use that performance to inform the choice of algorithm to use either for a one-off simulation setting or for a multiple simulation setting (such as in genetic algorithm optimization). Moreover, the modular design of WDSLib allows the user to avoid re-computing invariant parts of the solution process in a multi-simulation setting, yielding significant time savings. Last but not the least, WDSLib will enhance the capabilities of the research community to demonstrate the efficacy of new methods without having to re-engineer the content of shared WDSLib functions and data representations.

2 Getting Started

2.1 Downloading the WDSLib

This section provides information on how to download WDSLib. WDSLib is distributed in both source and pre-built binary forms through the Github website (`https://github.com/a1184182/WDSLib`). From the GitHub repository, click the "Clone or download" button to download. The GitHub repository includes the WDSLib source codes, in the `src` folder, as well as pre-built binary packages for 32- and 64-bit Windows, 32- and 64-bit Linux, and MacOS platforms.

2.2 Third-Party Dependencies

The WDSLib source code bundles two additional packages, including:

- **SuiteSparse** (Davis et al. 2014) (available from `http://faculty.cse.tamu.edu/davis/suitesparse.html`) is a suite of sparse matrix algorithms with exceptional performance, from which the approximated minimum degree permutation (AMD) and the sparse Cholesky decomposition routines have been used; and

- **EPANET** (available from `https://www.epa.gov/water-research/epanet`) part of which is used in WDSLib to parse the network parameter from the EPANET input file.

2.3 Installing the WDSLib from Binaries

Just download the binary that is appropriate to your system. Instructions for executing these binaries are given in the following sections.

2.4 Installing the WDSLib from Source

WDSLib is built by using CMake, a cross-platform, open-source build system generator, to generate makefiles for different operating systems. In this section, an installation guide for four platforms, namely Windows, Cygwin, Linux, and MacOS is presented.

2.4.1 WINDOWS

A binary release of CMake and a C++ compiler, such as visual studio or MinGW, are required for installation on Windows (`https://cmake.org/download/`).

Step 1: After installing the CMake application, input the WDSLib source code directory into the red box and the build file directory as shown in Figure 1.

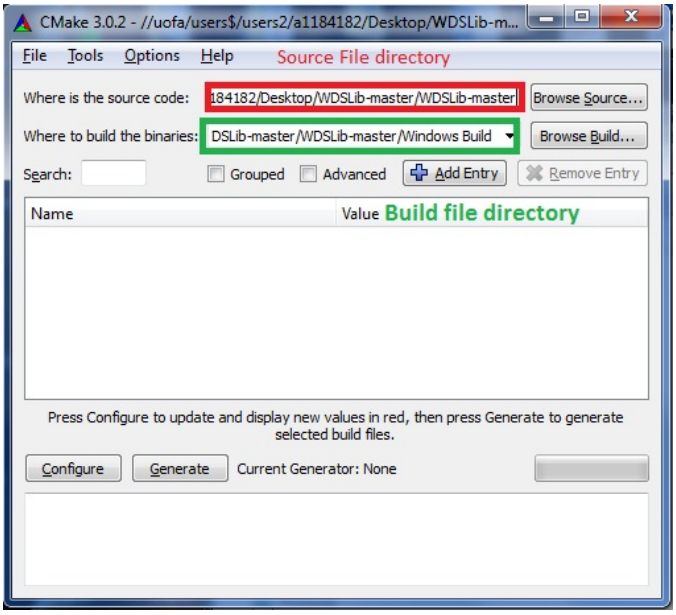


Figure 1: Windows installation Step 1: choose the source file directory and the build file directory

Step 2: After choosing the source file directory and the build file directory, press the configure button as shown in Figure 2.

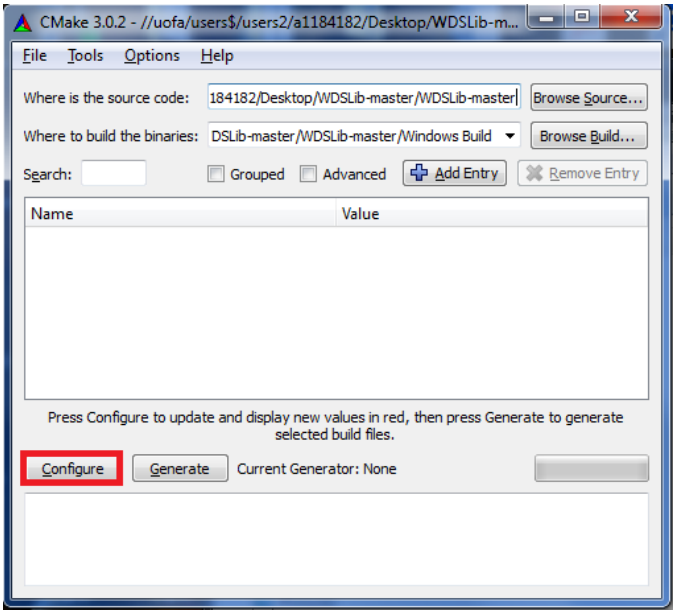


Figure 2: Windows installation Step 2: Configure the WDSLib using CMake

Step 3: After pressing the *configure* button, Cmake will ask the user to choose a project generator (compiler) as shown in Figure 3. Please note that the chosen project generator must be added to your path environment variables.

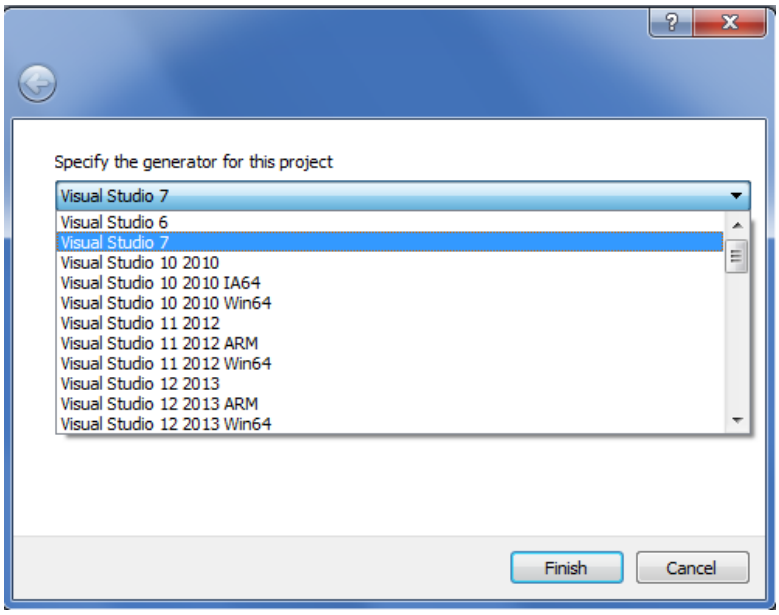


Figure 3: Windows installation step 3: choose a project generator (compiler)

Step 4: Generate build files using the selected project generator by pressing the *generate* button.

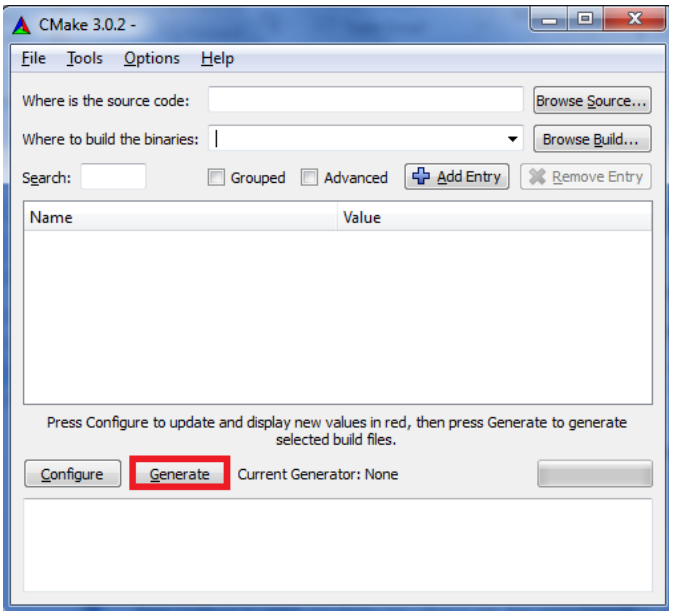


Figure 4: Windows installation step 4: generate selected build files

Step 5: Open the project solution created by the Cmake and compile the solution.

2.4.2 CYGWIN, LINUX, AND MACOS

Two packages, g++ and CMake, are required to build the WDSLib on Cygwin, both of which can be installed via the default Cygwin setup program.

For Linux and MacOS, CMake source files can be downloaded from the CMake source distribution website (`https://cmake.org/download/`). After downloading the source files, the following commands are required to install CMake.

```
./bootstrap
make
make install
```

Step 1: Change the current directory to the WDSLib build directory using one the following commands that is applicable to the current operating system

```
cd WDSLib-master/Cygwin\ Build/
cd WDSLib-master/Linux\ Build/
cd WDSLib-master/Mac\ Build/
```

Step 2: Configure and generate makefiles using g++ as the compiler.

```
cmake ../WDSLib-master/
```

Step 3: compile and install WDSLib using the following command

```
make
```

2.5 Unit Tests

A suite of unit tests has been provided to check the validity of the results. These tests can be performed by using the following command:

```
make test
```

Five tests have been used for each of the four WDS solution methods, namely tests for each of: the continuity residuals, the energy residuals, the norm at the final iteration, the infinity norm of the difference between the users final flows and the results, and the infinity norm of the difference between the users final heads and the results.

Please note that the results from these tests are highly dependent on the stopping tolerance selected by the user as any digits less significant than the stopping tolerance will be unreliable.

2.6 Uninstalling WDSLib

WDSLib does not rely on a formal uninstaller to uninstall the package. To uninstall the package, users only need to delete the main WDSLib directory.

3 How to use WDSLib

WDSLib consists of a collection of functions which can be used either as a standalone application for fast one-off simulations or as a library of software components that can be integrated into a user's own WDS solution processes. This section presents two example applications. The first application is the setup for a basic one-off simulation of a WDS. The second application presents an example using WDSLib to implement a simple 1+1 Evolutionary Strategy (Beyer and Schwefel 2002) (1+1-ES or, more commonly, 1+1EA) for sizing pipes in a WDS.

Table 1: The default values for the simulation parameters

Parameter	Default value
maxIter	50
initV	1 ft/s
StopTol	1×10^{-6}
NormTyp	inf-norm
StopTest	$\frac{\|\|\boldsymbol{q}^{(m+1)}-\boldsymbol{q}^{(m)}\|\|}{\|\|\boldsymbol{q}^{(m+1)}\|\|}$
MinImproTol	1×10^{-4}

3.1 A Once-off Simulation setting

The setup for WDSLib as a standalone application is straightforward. The user provides a human-readable configuration text file that specifies input and output filenames; the name of the solver; the desired output variables; and simulation parameters. These values have sensible defaults, as shown in Table 1, so users can set up the solver by using a minimal configuration such as that shown in Fig. 5. By using this configuration file, WDSLib is configured to run a single hydraulic analysis of the network that is stored as "hanoi.inp", an EPANET-formatted input file, under the "Network/" sub-directory, using the reformulated co-tree flows method (solver 2) with the forest-core partitioning algorithm.

```
1  [InputFile]
2  <directory>      %the input file directory
3  Network/
4  <file>           %the input file name
5  hanoi.inp
6  <end>
7  %--------------------------------------------------%
8  [controlFlag]
9  <SolverFlag>     %1 for GGA 2 for RCTM
10 2
11 <FCPAFlag>         %0 for off 1 for on
12 1
13 <end>
14 %--------------------------------------------------%
```

Figure 5: A minimal configuration file to run the GGA in WDSLib

The full set of configuration parameters for once-off simulations is shown

```
[InputFile]
<directory>      %REQUIRED INFORMATION the input file directory
WDSnetwork/
<file>           %REQUIRED INFORMATION the input file name
n1.inp
<end>
%--------------------------------------------------%
[Parameter]
<maxIter>        %maximum number of iteration
50
<initV>          %initial velocity
0.3048
<StopTol>        %stopping tolerance used
1.000e-6
<NormTyp>      %1 for 1-norm, 2 for 2-norm, 3 for inf-norm
3.0
<StopTest>     %1 for q-norm, 2 for h-norm 3 for q&h-norm
1
<MinImproTol>
1.0000e-3
<end>
%--------------------------------------------------%
[controlFlag]
<SolverFlag>     %1 for GGA 2 for RCTM
2
<FCPAFlag>
0
<end>
%--------------------------------------------------%
```

Figure 6: A configuration file to run the RCTM in WDSLib

in Fig. 7. Apart from the options that are demonstated in the minimal configuration example, two more option categories, `parameter` and `dispFlag` are available. The `dispFlag` category will be discussed in the next section.

The settings available in the `parameter` category controls how the hydraulic computations are carried out. These settings include:

- **maxIter**: the maximum number of iterations that are allowed to be used for the Newton method otherwise the process terminates.
- **initV**: the initial guess of flow velocity for each appropriate pipe
- **StopTol**: the stopping tolerance (a value of 1.0×10^{-6} is suggested).
- **NormTyp**: the type of norm distance used: 1 for Manhattan norm (1-norm), 2 for Euclidean norm (2-norm), 3 for Maximum norm (inf-norm)

- **StopTest**: the stopping test used to test for convergence: 1 for $\frac{||\boldsymbol{q}^{(m+1)}-\boldsymbol{q}^{(m)}||}{||\boldsymbol{q}^{(m+1)}||}$, 2 for $\frac{||\boldsymbol{h}^{(m+1)}-\boldsymbol{h}^{(m)}||}{||\boldsymbol{h}^{(m+1)}||}$, and 3 for both $\frac{||\boldsymbol{q}^{(m+1)}-\boldsymbol{q}^{(m)}||}{||\boldsymbol{q}^{(m+1)}||}$ and $\frac{||\boldsymbol{h}^{(m+1)}-\boldsymbol{h}^{(m)}||}{||\boldsymbol{h}^{(m+1)}||}$.
- **MinImproTol**: A tolerance that is used to test for whether or not there is no improvement in the last five iterations of the Newton method. In WDSLib, the norms of the latest five iterations are used to fit to a curve by using singular value decomposition. Solution process will stop if the improvement in the last five iterations is less than *MinImproTol*.

3.2 A Multi-run Simulation setting

As a minimalist example of the application of WDSLib to a WDS network design problem, the following example uses a quite simple 1+1EA for optimization of sizing pipe diameters (obviously other more complex types of evolutionary algorithm could be substituted). This algorithm takes an existing network with randomly generated pipe diameters and optimizes the network to minimize cost, subject to given minimum allowable pressure head constraints. A 1+1EA is a very simple evolutionary strategy which starts with a randomly generated individual (in this case a WDS diameter configuration). This 1+1EA then progresses by applying a mutation to a random pipe diameter size, and then evaluating the new individual. If the new individual is better it replaces the old network. This process continues in a loop until a given number of evaluations is reached.

The C++ code for this example is included as a standalone driver (as shown in Appendix A). If the name of the file containing this code is: `simpEA.cc` then the simplest command to compile this code is:

```
g++ simpEA.cc -o simpEA -Llib -lWDSLib
```

To run this code the user would type:

```
./simpEA config.txt
```

where `config.txt` contains the same configuration text as for the previous example.

4 Display options

```
1  %-------------------------------------------------%
2  [dispFlag]
3  %0 no output
4  %1 print output to screen only
5  %2 print output to file only
6  %3 print output to both screen and file
7  <BasicFlag>
8  2
9  <NetInfoFlag>
10 2
11 <ConvergenceFlag>
12 2
13 <StatFlag>
14 2
15 <GFflag>
16 2
17 <ScalingFlag>
18 2
19 <NodalResultflag>
20 2
21 <LinkResultflag>
22 2
23 <QitersFlag>
24 2
25 <HitersFlag>
26 3
27 <timeFlag>
28 3
29 <end>            %end of dispFlag
30 %-------------------------------------------------%
```

Figure 7: Additional fields in the configuration file that control the report

The settings available in the `dispFlag` field control the information that is reported to a user. These settings include:

- **BasicFlag**: this flag controls the output of the basic information describing the simulation platform. An example output file produced by turning this flag on is shown in Figure 8.

```
==================
|Basic Information|
==================
Name=Mengnings-MBP,mengningqiu
Time: Thu Sep 20 20:52:59 2018
Machine epsilon: 2.22045e-16
```

Figure 8: An example output file that contains the basic information

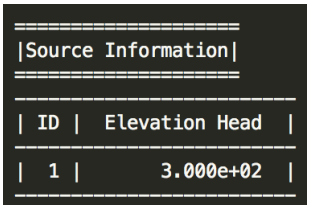
- **NetInfoFlag**: this flag controls the output of detailed information of the water distribution system that is been simulated. An example output file produced by turning this flag on can be found in Figures 9 to 12.

```
====================
|Network Information|
====================
------------------------------
Input filename: ../WDSnetwork/NYTP.inp
Number of Pipes:21
Number of Nodes:19
Number of Sources:1
Unit:US
Flow Unit:CFS
Pressure Unit:ft
Headloss Model:HW
```

Figure 9: An example output file that contains the basic network information

```
=================
|Node Information|
=================
----------------------------------------
| ID |  Elevation Head  |  Nodal Demand  |
----------------------------------------
|  2 |     0.000e+00    |    9.240e+01   |
|  3 |     0.000e+00    |    9.240e+01   |
|  4 |     0.000e+00    |    8.820e+01   |
|  5 |     0.000e+00    |    8.820e+01   |
|  6 |     0.000e+00    |    8.820e+01   |
|  7 |     0.000e+00    |    8.820e+01   |
|  8 |     0.000e+00    |    8.820e+01   |
|  9 |     0.000e+00    |    2.285e+02   |
| 10 |     0.000e+00    |    5.850e+01   |
| 11 |     0.000e+00    |    1.700e+02   |
| 12 |     0.000e+00    |    3.513e+02   |
| 13 |     0.000e+00    |    1.171e+02   |
| 14 |     0.000e+00    |    9.240e+01   |
| 15 |     0.000e+00    |    9.240e+01   |
| 16 |     0.000e+00    |    1.700e+02   |
| 17 |     0.000e+00    |    5.750e+01   |
| 18 |     0.000e+00    |    2.342e+02   |
| 19 |     0.000e+00    |    1.171e+02   |
| 20 |     0.000e+00    |    1.700e+02   |
----------------------------------------
```

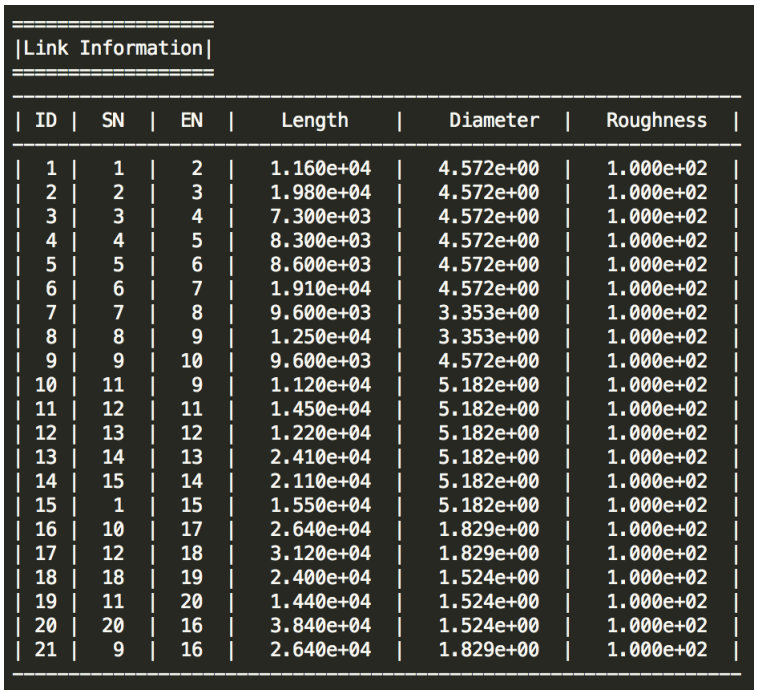
Figure 10: An example of node information



|Source Information|

ID	Elevation Head
1	3.000e+02

Figure 11: An example of source information

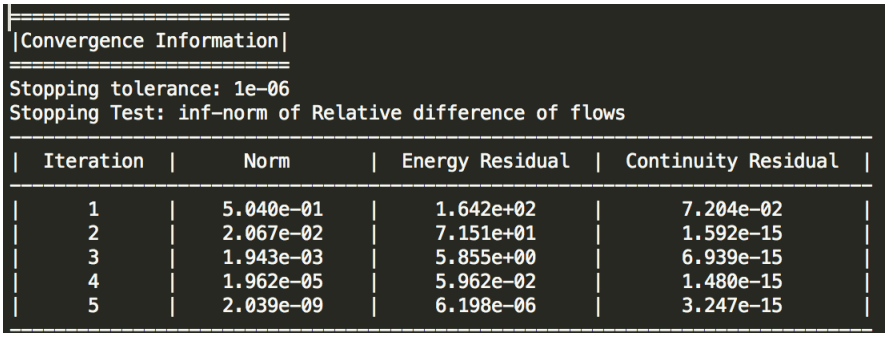


|Link Information|

ID	SN	EN	Length	Diameter	Roughness
1	1	2	1.160e+04	4.572e+00	1.000e+02
2	2	3	1.980e+04	4.572e+00	1.000e+02
3	3	4	7.300e+03	4.572e+00	1.000e+02
4	4	5	8.300e+03	4.572e+00	1.000e+02
5	5	6	8.600e+03	4.572e+00	1.000e+02
6	6	7	1.910e+04	4.572e+00	1.000e+02
7	7	8	9.600e+03	3.353e+00	1.000e+02
8	8	9	1.250e+04	3.353e+00	1.000e+02
9	9	10	9.600e+03	4.572e+00	1.000e+02
10	11	9	1.120e+04	5.182e+00	1.000e+02
11	12	11	1.450e+04	5.182e+00	1.000e+02
12	13	12	1.220e+04	5.182e+00	1.000e+02
13	14	13	2.410e+04	5.182e+00	1.000e+02
14	15	14	2.110e+04	5.182e+00	1.000e+02
15	1	15	1.550e+04	5.182e+00	1.000e+02
16	10	17	2.640e+04	1.829e+00	1.000e+02
17	12	18	3.120e+04	1.829e+00	1.000e+02
18	18	19	2.400e+04	1.524e+00	1.000e+02
19	11	20	1.440e+04	1.524e+00	1.000e+02
20	20	16	3.840e+04	1.524e+00	1.000e+02
21	9	16	2.640e+04	1.829e+00	1.000e+02

Figure 12: An example of pipe information

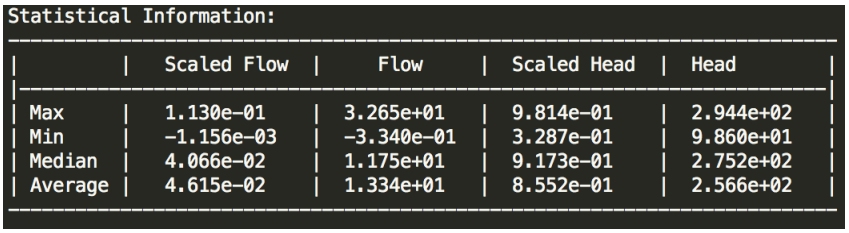
- **ConvergenceFlag**: this flag controls the output of convergence information of the solution methods that are been used.



```
|======================
|Convergence Information|
======================
Stopping tolerance: 1e-06
Stopping Test: inf-norm of Relative difference of flows
-------------------------------------------------------------------------------
|  Iteration  |      Norm       |  Energy Residual  |  Continuity Residual  |
-------------------------------------------------------------------------------
|      1      |    5.040e-01    |     1.642e+02     |       7.204e-02       |
|      2      |    2.067e-02    |     7.151e+01     |       1.592e-15       |
|      3      |    1.943e-03    |     5.855e+00     |       6.939e-15       |
|      4      |    1.962e-05    |     5.962e-02     |       1.480e-15       |
|      5      |    2.039e-09    |     6.198e-06     |       3.247e-15       |
-------------------------------------------------------------------------------
```

Figure 13: An example of numerical convergence information

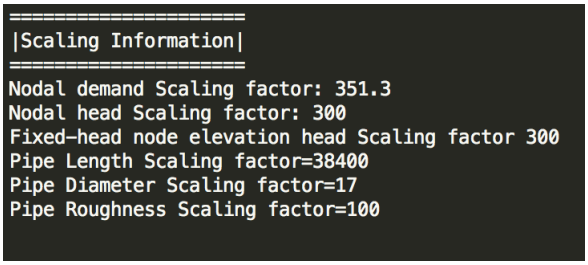
- **StatFlag**: this flag controls the output of summary statistical information from the results of the simulation. An example output file produced by turning this flag on can be found in Figure 14.



```
Statistical Information:
-------------------------------------------------------------------------------
|         |   Scaled Flow  |     Flow     |  Scaled Head  |  Head         |
|-----------------------------------------------------------------------------|
| Max     |   1.130e-01    |  3.265e+01   |  9.814e-01    |  2.944e+02    |
| Min     |   -1.156e-03   |  -3.340e-01  |  3.287e-01    |  9.860e+01    |
| Median  |   4.066e-02    |  1.175e+01   |  9.173e-01    |  2.752e+02    |
| Average |   4.615e-02    |  1.334e+01   |  8.552e-01    |  2.566e+02    |
-------------------------------------------------------------------------------
```

Figure 14: An example of statistical information of the final solution

- **ScalingFlag**: this flag controls the output of the scaling factors that are used to scale the input parameters. An example output file produced by turning this flag on can be found in Figure 15.



```
====================
|Scaling Information|
====================
Nodal demand Scaling factor: 351.3
Nodal head Scaling factor: 300
Fixed-head node elevation head Scaling factor 300
Pipe Length Scaling factor=38400
Pipe Diameter Scaling factor=17
Pipe Roughness Scaling factor=100
```

Figure 15: An example of scaling factors

- **NodalResultFlag**: this flag controls the output of the head at each node. An example output file produced by turning this flag on can be found in Figure 16.

Final Nodal Head

ID	Scaled Nodal Head	Nodal Head
2	9.814e-01	2.944e+02
3	9.557e-01	2.867e+02
4	9.483e-01	2.845e+02
5	9.417e-01	2.825e+02
6	9.366e-01	2.810e+02
7	9.288e-01	2.786e+02
8	9.173e-01	2.752e+02
9	9.090e-01	2.727e+02
10	9.088e-01	2.727e+02
11	9.094e-01	2.728e+02
12	9.140e-01	2.742e+02
13	9.243e-01	2.773e+02
14	9.502e-01	2.851e+02
15	9.770e-01	2.931e+02
16	7.048e-01	2.114e+02
17	8.846e-01	2.654e+02
18	5.284e-01	1.585e+02
19	3.287e-01	9.860e+01
20	7.003e-01	2.101e+02

Figure 16: An example of the final node heads

- **LinkResultFlag**: this flag controls the output of pipe results. An example output file produced by turning this flag on can be found in Figure 17.

Link Result:

ID -	SN -	EN -	Scaled Flow Rate -	Flow Rate (CMS)	Velocity (m/s)	G -	F -
1	1	2	8.469e−02	2.448e+01	1.491e+00	2.275e−01	4.213e−01
2	2	3	7.564e−02	2.186e+01	1.331e+00	3.526e−01	6.530e−01
3	3	4	6.658e−02	1.924e+01	1.172e+00	1.166e−01	2.160e−01
4	4	5	5.794e−02	1.675e+01	1.020e+00	1.178e−01	2.182e−01
5	5	6	4.930e−02	1.425e+01	8.678e−01	1.064e−01	1.970e−01
6	6	7	4.066e−02	1.175e+01	7.157e−01	2.005e−01	3.712e−01
7	7	8	3.202e−02	9.253e+00	1.048e+00	3.723e−01	6.895e−01
8	8	9	2.337e−02	6.755e+00	7.651e−01	3.708e−01	6.867e−01
9	9	10	5.732e−03	1.657e+00	1.009e−01	1.899e−02	3.516e−02
10	11	9	1.683e−02	4.863e+00	2.306e−01	3.014e−02	5.581e−02
11	12	11	4.899e−02	1.416e+01	6.714e−01	9.696e−02	1.796e−01
12	13	12	8.341e−02	2.410e+01	1.143e+00	1.284e−01	2.377e−01
13	14	13	9.488e−02	2.742e+01	1.300e+00	2.830e−01	5.241e−01
14	15	14	1.039e−01	3.004e+01	1.424e+00	2.678e−01	4.959e−01
15	1	15	1.130e−01	3.265e+01	1.549e+00	2.112e−01	3.911e−01
16	10	17	5.634e−03	1.628e+00	6.199e−01	4.462e+00	8.264e+00
17	12	18	2.295e−02	6.632e+00	2.525e+00	1.745e+01	3.231e+01
18	18	19	1.147e−02	3.316e+00	1.818e+00	1.807e+01	3.346e+01
19	11	20	1.550e−02	4.480e+00	2.456e+00	1.401e+01	2.594e+01
20	20	16	−1.156e−03	−3.340e−01	−1.831e−01	4.092e+00	7.578e+00
21	9	16	1.781e−02	5.148e+00	1.960e+00	1.190e+01	2.203e+01

Figure 17: An example output file that contains the link results

- **QitersFlag**: this flag controls the output of the flow rates ($\boldsymbol{q}$) at every iteration. An example output file produced by turning this flag on can be found in Figure 18.

```
================
|Iterative Flows|
================
```

	Flow Rates				
Pipe ID	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
1	2.608e+01	2.451e+01	2.448e+01	2.448e+01	2.448e+01
2	2.346e+01	2.190e+01	2.186e+01	2.186e+01	2.186e+01
3	2.084e+01	1.928e+01	1.924e+01	1.924e+01	1.924e+01
4	1.835e+01	1.678e+01	1.675e+01	1.675e+01	1.675e+01
5	1.585e+01	1.429e+01	1.425e+01	1.425e+01	1.425e+01
6	1.335e+01	1.179e+01	1.175e+01	1.175e+01	1.175e+01
7	1.085e+01	9.290e+00	9.253e+00	9.253e+00	9.253e+00
8	8.357e+00	6.793e+00	6.755e+00	6.755e+00	6.755e+00
9	1.657e+00	1.657e+00	1.657e+00	1.657e+00	1.657e+00
10	2.669e+00	4.797e+00	4.863e+00	4.863e+00	4.863e+00
11	1.256e+01	1.412e+01	1.416e+01	1.416e+01	1.416e+01
12	2.250e+01	2.407e+01	2.410e+01	2.410e+01	2.410e+01
13	2.582e+01	2.738e+01	2.742e+01	2.742e+01	2.742e+01
14	2.844e+01	3.000e+01	3.004e+01	3.004e+01	3.004e+01
15	1.628e+00	1.628e+00	1.628e+00	1.628e+00	1.628e+00
16	6.632e+00	6.632e+00	6.632e+00	6.632e+00	6.632e+00
17	3.316e+00	3.316e+00	3.316e+00	3.316e+00	3.316e+00
18	5.073e+00	4.509e+00	4.480e+00	4.480e+00	4.480e+00
19	2.589e-01	-3.052e-01	-3.342e-01	-3.340e-01	-3.340e-01
20	3.105e+01	3.262e+01	3.265e+01	3.265e+01	3.265e+01
21	4.555e+00	5.119e+00	5.148e+00	5.148e+00	5.148e+00

Figure 18: An example output file that contains the flow iterates

- **HitersFlag**: this flag controls the output of heads ($\boldsymbol{h}$) at every iteration (automatically turned off for RCTM). An example output file produced by turning this flag on can be found in Figure 19.

|Iterative Heads|

	Node Heads					
Node ID	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
2	2.950e+02	2.944e+02	2.944e+02	2.944e+02	2.944e+02	2.944e+02
3	2.879e+02	2.867e+02	2.867e+02	2.867e+02	2.867e+02	2.867e+02
4	2.857e+02	2.845e+02	2.845e+02	2.845e+02	2.845e+02	2.845e+02
5	2.838e+02	2.825e+02	2.825e+02	2.825e+02	2.825e+02	2.825e+02
6	2.824e+02	2.810e+02	2.810e+02	2.810e+02	2.810e+02	2.810e+02
7	2.805e+02	2.786e+02	2.786e+02	2.786e+02	2.786e+02	2.786e+02
8	2.773e+02	2.752e+02	2.752e+02	2.752e+02	2.752e+02	2.752e+02
9	2.753e+02	2.727e+02	2.727e+02	2.727e+02	2.727e+02	2.727e+02
10	2.767e+02	2.727e+02	2.727e+02	2.727e+02	2.727e+02	2.727e+02
11	2.742e+02	2.728e+02	2.728e+02	2.728e+02	2.728e+02	2.728e+02
12	2.754e+02	2.742e+02	2.742e+02	2.742e+02	2.742e+02	2.742e+02
13	2.786e+02	2.773e+02	2.773e+02	2.773e+02	2.773e+02	2.773e+02
14	2.861e+02	2.851e+02	2.851e+02	2.851e+02	2.851e+02	2.851e+02
15	2.937e+02	2.931e+02	2.931e+02	2.931e+02	2.931e+02	2.931e+02
16	2.355e+02	2.136e+02	2.114e+02	2.114e+02	2.114e+02	2.114e+02
17	2.715e+02	2.654e+02	2.654e+02	2.654e+02	2.654e+02	2.654e+02
18	1.958e+02	1.585e+02	1.585e+02	1.585e+02	1.585e+02	1.585e+02
19	1.460e+02	9.860e+01	9.860e+01	9.860e+01	9.860e+01	9.860e+01
20	2.214e+02	2.096e+02	2.101e+02	2.101e+02	2.101e+02	2.101e+02

Figure 19: An example output file that contains the head iterates

- **timeFlag**: this flag controls the output of the time consumed by every function. An example output file produced by turning this flag on can be found in Figure 20. The units are nanoseconds.

```
timespend[AMD] = 73
timespend[FCPA] = 22
timespend[GetGF-1] = 5
timespend[Pre-other] = 7
timespend[getGF-2] = 8
timespend[init] = 8
timespend[iter] = 452
timespend[linear solver] = 258
timespend[normTest] = 34
timespend[reverseFCPA] = 10
timespend[second phase] = 11
timespend[undoPermutation] = 3
```

Figure 20: An example output file that contains the time spent in the functions of interest

References

Beyer, H. G. and H. P. Schwefel (2002). "Evolution strategies–A comprehensive introduction". *Natural Computing* 1.1, pp. 3–52. DOI: 10.1023/A:1015059928466.

Davis, Tim, WW Hager, and IS Duff (2014). "SuiteSparse". URL: http://faculty.cse.tamu.edu/davis/suitesparse.html.

Elhay, S. and A. R. Simpson (2011). "Dealing with zero flows in solving the nonlinear equations for water distribution systems". *Journal of Hydraulic Engineering* 137.10, pp. 1216–1224. DOI: 10.1061/(ASCE)WR.1943-5452.0000561.

Elhay, S., A. R. Simpson, J. Deuerlein, B. Alexander, and W. H. Schilders (2014). "Reformulated co-tree flows method competitive with the global gradient algorithm for solving water distribution system equations". *Journal of Water Resources Planning and Management* 140.12, pp. 04014040-1–04014040-10. DOI: 10.1061/(ASCE)WR.1943-5452.0000431.

Guidolin, M., P. Burovskiy, Z. Kapelan, and D. A. Savic (2010). "CWSNet: An object-oriented toolkit for water distribution system simulations". *Proceedings of the 12th Annual Water Distribution Systems Analysis Conference, WDSA*, pp. 12–15.

Qiu, M., B. Alexander, A. R. Simpson, and S. Elhay (2018a). "A Software Tool for Assessing the Performance of and Implementing Water Distribution System Solution Methods". Manuscript submitted for publication to Journal of Environmental Modelling and Software.

Qiu, M., S. Elhay, A. R. Simpson, and B. Alexander (2018b). "A Benchmarking Study of Water Distribution System Solution Methods". Manuscript accepted for publication to Journal of Water Resources Planning and Management.

Rossman, L. A. (2000). "Epanet 2 users manual, us environmental protection agency". *Water Supply and Water Resources Division, National Risk Management Research Laboratory, Cincinnati, OH* 45268.

Simpson, A. R. and S. Elhay (2010). "Jacobian matrix for solving water distribution system equations with the Darcy-Weisbach head-loss model". *Journal of hydraulic engineering* 137.6, pp. 696–700. DOI: 10.1061/(ASCE)HY.1943-7900.0000341.

Simpson, A. R., S. Elhay, and B. Alexander (2014). "Forest-Core Partitioning Algorithm for Speeding Up Analysis of Water Distribution Systems". *Journal of Water Resources Planning and Management* 140.4, pp. 435–443. DOI: 10.1061/(ASCE)WR.1943-5452.0000336.

Todini, E. and S. Pilati (1988). "A gradient algorithm for the analysis of pipe networks". *Computer applications in water supply: vol. 1—systems analysis and simulation*. Research Studies Press Ltd., pp. 1–20.

Appendix A Standalone driver for the 1+1EA example

```
#include "Simulation.h"
#include "result.h"
using namespace std;
#define GTN 100000                              //the total number of generations
/*available pipe diameters (inches)*/
vector<int> ADiameter={36,48,60,72,84,96,108,120,132,144,156,168,180,192,204};
/*dollar per unit length*/
vector<int>unitCost={93.6,133.7,176.3,221,267.6,315.8,365.4,416.5,468.7,
              522.1,576.6,632.1,688.5,745.1,804.1};
vector<int>  generateinitialDiameters(int);     // see fig. 6
vector<int>  mutate(vector<int>);               // see fig. 7
double evaluate(Net*, Result*);                 // see fig. 8
int  main(int argc, char *argv[]){
  srand (1);
  char *config=argv[1];
  Result *result=new Result();
  Simulation *simulation1=new Simulation();     //initialize the simulation class
  Net *net=simulation1->L1a(result,config);     //perform the L1a functions
  vector<int> p1=generateinitialDiameter(net->getNp());//initial guesses of diameter
  vector<int> p2;                               //work space for current best
  double cost,currentbest;
  for(int i=0;i<GTN;i++){
    simulation1->setD(&p1,&ADiameter,net->getPIPE()->Diascale());
    simulation1->solve(result);                 //perform the L2 and L3 functions
    simulation1->L1b(result);                   //reverse the permutation
    cost=evaluate(result,net,&p1);              //evaluate the network cost
    if(cost<currentbest||i==0){                 //selection operator
      currentbest=cost;
      p2=p1;
      cout<<i<<"\t"<<currentbest<<"\t"<<penaltyCost<<endl;
    }
    p1=mutate(&p2);                             //mutation operator
  }
  simulation1->dispResult(result);
  delete simulation1;
  delete result;
  return 0;
 }
```

Figure 21: Example code for 1+1EA for Pipe Sizing

```
vector<int> generateinitialDiameters(int np)
{
  vector<int> Diameter = vector<int>(np);
  for (int i=0;i<np;i++)
    Diameter[i]=rand()%ADiameter.size();  //set the index for pipe diameters
  return Diameter;
}
```

Figure 22: Implementation code for pipe size initialization

```
vector<int>  mutate(vector<int>* string){
  vector<int> string1;
  string1=*string;
  int aa=rand()%(string->size()); //choose which pipe to mutate
  int a=rand()%ADiameter.size();  //choose a pipe diameter after the mutation
  (string1)[aa]=a;                //set the pipe index
  return string1;
}
```

Figure 23: Implementation code for the mutation operator

```
double evaluate (Result* result,Net* net,vector<int> *p1) {
  PIPE* pipe =net->getPIPE();
  double np=net->getNp();
  double nj=net->getNj();
  double P=1e7;
  double cost1=0;
  penaltyCost=0;
  vector<double> hsol=result->getHsol();      //get the vector of nodal heads
  vector<double>* L=pipe->getL();             //get the vector of pipe lengths
  vector<double>* zu=net->getNODE()->getZU(); //get the vector of nodal elevations
  double Lscale=pipe->Lenscale();             //get the scaling factor for length
  for (int i = 0; i<np; i++)
    cost1+=(*L)[i]*Lscale*unitCost[(*p1)[i]]; //calculate the material cost
  for (int i = 0; i<nj; i++)
    if ((hsol)[i]-(*zu)[i]<minP)
      penaltyCost+=(minP-(hsol)[i]+(*zu)[i])*scaleP;//calculate penalty cost
  return cost1+penaltyCost;
}
```

Figure 24: Implementation code for the evaluation function