Analysis of Water Distribution Networks Using MATLAB and Excel Spreadsheet: Q-Based Methods

MAJID NIAZKAR, SEIED HOSEIN AFZALI

Department of Civil and Environmental Engineering, School of Engineering, Shiraz University, Shiraz, Fars, Iran

Received 3 October 2016; accepted 24 December 2016

ABSTRACT: With the emergence of computers, literatures of water distribution network (WDN), and application of computer-aided programs for modeling WDNs have been significantly improved. Although MATLAB and Excel spreadsheet provide suitable facilities for both academic and practical purposes, the comprehensive application of these programs in WDN analysis has not been addressed. In this paper, the step-by-step implementation of three Q-based methods is presented for solving WDNs using these programs. In order to focus more on the educational aspects of computer application, a simple pipe network is analyzed using these Q-based methods. However, as basics of the implementation are sufficiently covered, the provided codes can be improved to analyze more complicated pipe networks. © 2017 Wiley Periodicals, Inc. Comput Appl Eng Educ; View this article online at wileyonlinelibrary.com/journal/cae; DOI 10.1002/cae.21796

Keywords: water distribution network; computer application in water network analysis; Q-based method; MATLAB; Excel spreadsheet

INTRODUCTION

Hydraulic modeling of water distribution networks (WDNs) is one of the most practical facets of water, civil, and also mechanical engineering curriculum. Appraisal of computer programs for this modeling is so essential in educational programs that some international academic references written in this field from time to time introduced relevant written codes, mostly in FORTRAN language [1–2]. By emerging computer improvements, not only computer-aided programs have been recommended for modeling WDNs, but also more efficient and computer-based approaches were proposed. Finally, computer application to WDN modeling brings more efficiency and improvements in this area.

Several robust and commercial programs, such as EPANET [3] and WaterGem [4], have been utilized for WDN modeling. However, these kinds of softwares do not provide the chance for their applicants and specially engineering students to become familiar with the fundamental background of this modeling. In this regard, several studies were conducted to utilize sufficient either codes or programs for educational purposes [5–8]. The main contribution of these efforts was to present how to model a pipe network using different programs for educational purposes. These contributions not only emphasize on the assessment of computer in WDN modeling but also provide valuable opportunities for engineering pupils to achieve a better insight about the relevant fundamental subjects.

In essence, analysis of WDN is considered as an inevitable part of its modeling since whatever is planned for a typical WDN cannot be done without analyzing it. The aim of analyzing a pipe network is exclusively to determine either flow rates in pipes (Q) or hydraulic heads in nodal points (h). Since these state variables may vary spatially or temporally in a WDN, the equations governing flow field in pipe network depends upon the condition in which the flow varies with time. In this regard, analysis of WDN under the steady-state condition is the most common one in modeling of a typical water network.

The state of art of computer application to WDN analysis shows that appraisal of computer programs for solving WDN has been an active area in the literature [5–12]. In this regard, application of MATLAB for this purpose not only is not adequately presented but also is confined to implementation of specific approach [9]. Additionally, Excel Spreadsheet has been considered as powerful platform for implementing numerical

Correspondence to S. H. Afzali (afzali@shirazu.ac.ir).

methods for various applications for educational purposes [13]. Regarding solving pipe network, Huddleston et al. [5] and Brkic [11] utilized Excel to implement linear theory and Hardy Cross method, respectively. More recently, Niazkar and Afzali implemented h-based methods in MATLAB and Excel spreadsheet [12]. Finally, the literature review indicates that implementing Q-based methods for solving WDN is not adequately addressed using Excel and MATLAB programs while their application may bring about a better chance to teach the basics of modeling WDNs in engineering curricula.

In this paper, MATLAB and Excel are used to solve a sample pipe network under a time-independent condition by focusing exclusively on step-by-step implementation. In the presented codes, the input data is first inserted in Excel Spreadsheet. Afterwards, the implementation of Q-based methods including Hardy Cross, Linear Theory, and Q-based Newton-Raphson are introduced step by step with the aid of coding in MATLAB program. By using each one of the Q-based methods, the simple example is analyzed and the output results are presented in Excel Spreadsheet. In reference to the present state of the art of the subject, the presented programs may bring a novel computer application in WDN modeling for both educators and engineers.

ANALYSIS OF WATER DISTRIBUTION NETWORK

Analysis of a typical WDN is conducted to find the flow field throughout the network. Under the steady-state condition, this flow field varies exclusively with respect to space whereas temporal variation is considered in extended-period simulation and transient conditions. For a specified network layout, the state variables depend upon nodal demands under the demand-driven condition. In this condition, energy and continuity equations govern the state variables of the problem. Generally, the governing equations are mainly casted in two different ways: (1) Q-based (Hardy Cross method, Linear Theory, and Newton-Raphson method) and (2) h-based (Newton-Rapshon method, finite element method, and Gradient algorithm). As it was mentioned in parentheses, different methods can be utilized for solving WDN for different casting. In this paper, the step-by-step implementation of the Q-based methods in MATLAB is presented. In order to more focus on the educational aspect of these implementations, a simple network including seven pipes, six nodal points and a reservoir is selected from the literature [12]. The layout of this network is shown in Figure 1. The hydraulic head of the reservoir

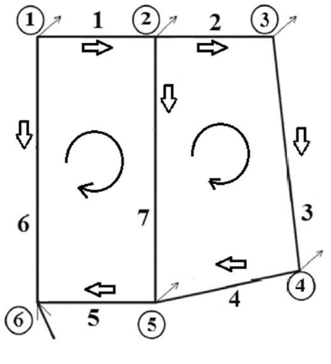


Figure 1 Sample pipe network.

is equal to 200 m and it is applied to the sixth node. The pipe characteristics and nodal demands of this network are presented in [12].

METHODS

In this section, the governing equations are casted for the three Q-based methods considering the sample network. Since the governing equations of WDN under the steady-state condition comprise a nonlinear algebraic system of equations, an iteration-based scheme using initial guesses for state variables should be used in each method to solve WDN. The description and formulations of these methods are presented in the following:

(1) Hardy Cross method: This method was basically proposed to solve WDNs using handy calculations. In this non-matrix method, continuity equations are put aside from the list of governing equations since initial guesses for pipe flow rates are selected in a way that they satisfy these equations. Furthermore, Newton-Rapshon scheme is applied to energy equations to compute loop correctors. Consequently, loop correctors, which are particularly used to improve pipe flow rates, are determined for all network loops in each iteration. In case a pipe participates in two adjacent loops, two related loop correctors are utilized to improve its flow rate. The loop corrector is defined in Equation (1):

$$\Delta Q_j = -\frac{\sum_{i=1}^{m} K_i sgn(Q_i) Q_i^n}{n \sum_{i=1}^{m} K_i Q_i^{n-1}} \tag{1}$$

where ΔQ_j denotes the loop corrector in the jth loop, K_i is the pipe coefficient of the ith pipe, $sgn(Q_i)$ is the sign function calculated for Q_i, m is the number of pipes in the jth loop, and n is a power.

The pipe coefficient and power introduced in Equation (1) depend upon the resistance equation which are equal to $\frac{8 f_i L_i}{\pi^2 g D_i^5}$ and 2 for Darcy-Weisbach (D-W) equation, respectively. In the former relation for pipe coefficient, f is D-W friction factor, L is pipe length, π is pi number, g is gravity acceleration, and D is pipe diameter. It should be noted that the pipe coefficient based on D-W equation consists of two parts: (1) a constant part which is a function of pipe diameter and length, and (2) a variable part (f). Unlike the variable part of the pipe coefficient, the constant part does not vary in iterations and is constant throughout WDN analysis.

In Hardy Cross method, all pipe flow rates (Q_i) are positive and arbitrary directions for flow in each pipe and each loop are considered. These directions are used not only to evaluate the sign function but also to improve the flow rates after the loop corrector is computed in each iteration. In this regard, the sign function for a typical pipe is positive only if the direction of loop corrector is the same as the direction of flow in that pipe. The denominator of Equation (1) is always positive while the loop corrector may be obtained either positive or negative based on the sign function. Regardless of the sign obtained for loop corrector, it will be added to the pipe flow rate if the directions of the loop corrector and flow in the pipe are the same. Otherwise, it will be subtracted from the pipe flow rate. Considering this notation on how to revise pipe flow rates in each loop after the loop corrector is determined in each iteration is quite important in implementation of this method in computer-aided programs.

(2) Linear Theory: In this method, the governing equations are casted in a way that the flow rates in pipes are the unknowns of the system of equations. Consequently, the number of unknowns is equal to the number of pipes in the network. The continuity and energy equations are linear and nonlinear in respect to Q, respectively, while the former is written for nodal points and the latter is written for loops. The system of equations used in Linear Theory is illustrated in Equation (2) in the matrix format for the sample network:

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 1 \\ K_1Q_1^{n-1} & 0 & 0 & 0 & K_5Q_5^{n-1} & -K_6Q_6^{n-1} & K_7Q_7^{n-1} \\ 0 & K_2Q_2^{n-1} & K_3Q_3^{n-1} & K_4Q_4^{n-1} & 0 & 0 & -K_7Q_7^{n-1} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ Q_5 \\ Q_6 \\ Q_7 \end{bmatrix} = \begin{bmatrix} 0.015 \\ 0.010 \\ 0.035 \\ 0.040 \\ 0.030 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

In contrast to the procedure in Hardy Cross method, the pipe flow rates are directly computed in this method. Moreover, the initial guesses for flow rates can be chosen arbitrarily and, unlike the ones in the Hardy Cross method, they do not have to satisfy the continuity equations.

(3) Q-based Newton-Raphson method: This method utilizes the same equations as Linear Theory whereas the Newton-Rapshon scheme is applied to these equations. The values of pipe flow rates are improved in each iteration of Q-based Newton-Raphson method after solving the system of equations. This system is shown in Equation (3) for the first iteration for the sample network.

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 1 \\ (n-1)K_1Q_1^{n-1} & 0 & 0 & 0 & (n-1)K_5Q_5^{n-1} & -(n-1)K_6Q_6^{n-1} & (n-1)K_7Q_7^{n-1} \\ 0 & (n-1)K_2Q_2^{n-1} & (n-1)K_3Q_3^{n-1} & (n-1)K_4Q_4^{n-1} & 0 & 0 & -(n-1)K_7Q_7^{n-1} \end{bmatrix} \begin{bmatrix} \Delta Q_1 \\ \Delta Q_2 \\ \Delta Q_3 \\ \Delta Q_4 \\ \Delta Q_5 \\ \Delta Q_6 \\ \Delta Q_7 \end{bmatrix} = \begin{bmatrix} Q_1 - Q_6 - 0.030 \\ Q_1 - Q_2 - Q_7 - 0.040 \\ Q_2 - Q_3 - 0.035 \\ Q_3 + Q_4 - 0.010 \\ Q_4 - Q_5 + Q_7 - 0.015 \\ K_1Q_1^n + K_5Q_5^n - K_6Q_6^n + K_7Q_7^n \\ K_3Q_3^n + K_2Q_2^n + K_4Q_4^n - K_7Q_7^n \end{bmatrix} \tag{3}$$

Unlike the process in the Hardy Cross method, only flow directions in pipes are considered in casting equations in Linear Theory and Q-based Newton-Raphson methods. The two latter Q-based methods build matrix equations and their left hand side matrices, which should be calculated in each iteration, are not symmetric. The right hand side matrix in Linear Theory is constant throughout the WDN analysis whereas it is a function of pipe flow rates in the Q-based Newton-Raphson. In Q-based methods, flow rates and D-W friction factors in pipes are computed while the hydraulic heads at nodal points can be computed at the end of WDN analysis using the considered resistance equation.

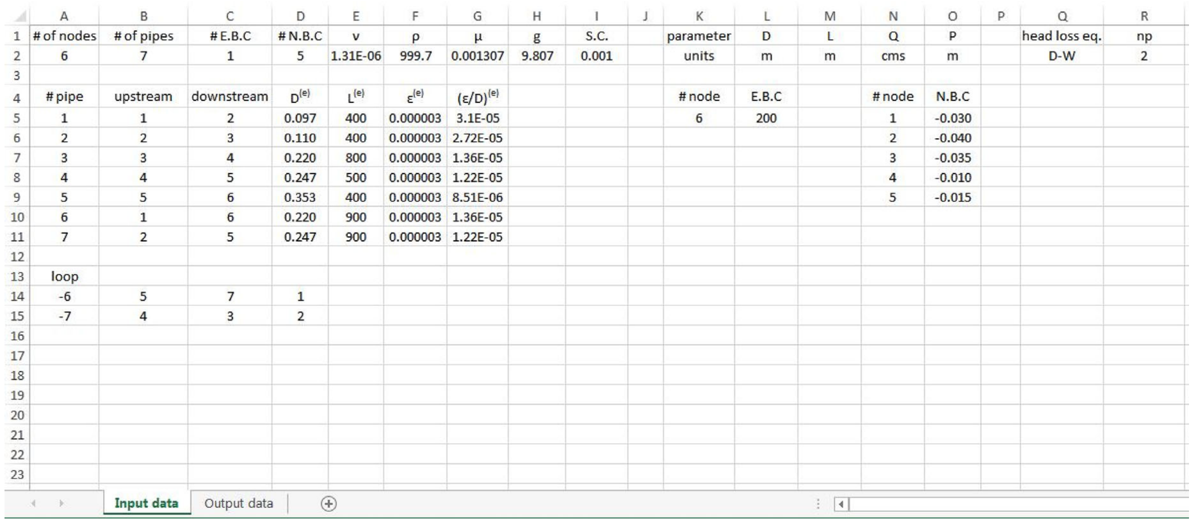
IMPLEMENTATION OF Q-BASED METHODS FOR SOLVING WDNs

In this section, the step-by-step procedures of the introduced methods are separately presented with the corresponding MATLAB codes. In these codes, the hydraulic solver is coded in MATLAB while Excel spreadsheet is the interface used to transform input and output data with the main code written in MATLAB. In this regard, the input data is first inserted into specific cells of Excel which will be later called from MATLAB codes. As each MATLAB code performs WDN analysis, the results will be automatically presented in Excel spreadsheet for ease of further usage.

In the provided MATLAB codes for Q-based methods, first the nodes with demands are enumerated and then the nodes with specified heads. Consequently, the designated numbers for reservoirs are the larger ones. Furthermore, the convention for direction for loop is clockwise while flow direction in pipes is first assumed to be from the node with lower number to the one with larger number. The latter may be revised in the process of analysis. Finally, it should be noted that even though the presented codes are written for solving the sample network for educational purposes, they can be simply modified to analyze larger or more complex networks.

Input and Output Data Presented in Excel Spreadsheet

The common required input data for analyzing a typical WDN like the sample includes: (1) numbers of nodes and pipes, (2) number of nodes with specified heads (essential boundary condition or EBC), (3) number of nodes with specified demand (natural boundary condition or NBC), (4) kinematic viscosity of water, (5) water density, (6) gravitational acceleration, (7) stopping criterion (S.C), (8) continuity table showing the relation between pipe and



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	# of nodes	# of pipes	# E.B.C	# N.B.C	ν	ρ	μ	g	S.C.		parameter	D	L	Q	P		head loss eq.	np
2	6	7	1	5	1.31E-06	999.7	0.001307	9.807	0.001		units	m	m	cms	m		D-W	2
3																		
4	# pipe	upstream	downstream	$D^{(e)}$	$L^{(e)}$	$\varepsilon^{(e)}$	$(\varepsilon/D)^{(e)}$				# node	E.B.C		# node	N.B.C			
5	1	1	2	0.097	400	0.000003	3.1E-05				6	200		1	-0.030			
6	2	2	3	0.110	400	0.000003	2.72E-05							2	-0.040			
7	3	3	4	0.220	800	0.000003	1.36E-05							3	-0.035			
8	4	4	5	0.247	500	0.000003	1.22E-05							4	-0.010			
9	5	5	6	0.353	400	0.000003	8.51E-06							5	-0.015			
10	6	1	6	0.220	900	0.000003	1.36E-05											
11	7	2	5	0.247	900	0.000003	1.22E-05											
12																		
13	loop																	
14	-6	5	7	1														
15	-7	4	3	2														
16																		
17																		
18																		
19																		
20																		
21																		
22																		
23																		

Figure 2 Inserting input data in Excel spreadsheet.

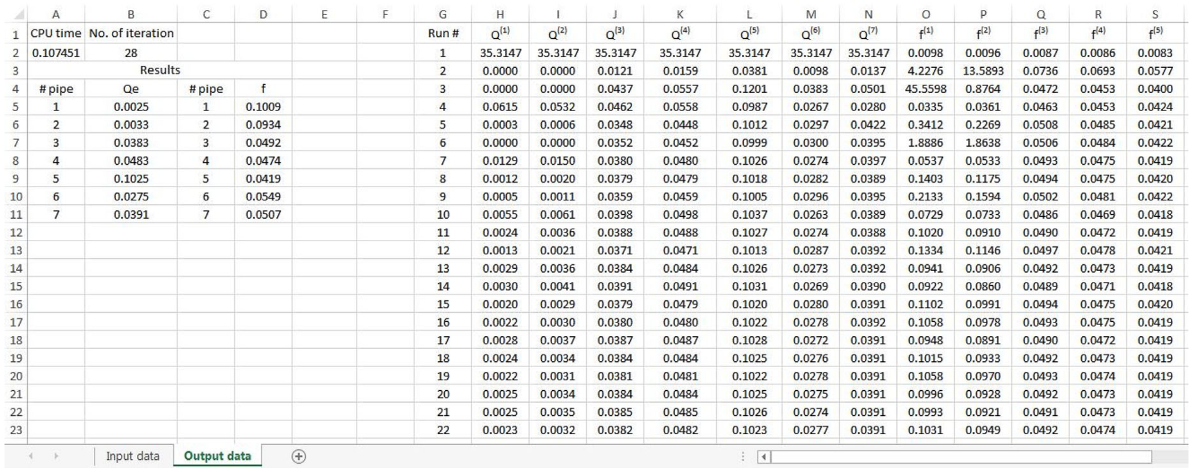
node numberings in the network under consideration, (9) pipe lengths, (10) pipe diameters, (11) either relative roughness for D-W equation or Hazen-Williams coefficient for each pipe, (12) nodal demands, (13) specified hydraulic heads, and (14) the resistance equation specified (D-W equation is used in this study). These input data should be inserted in the specific cells of a worksheet in an Excel spreadsheet file, which are named "Input data" and "example.xlsx", respectively, as shown in Figure 2. These titles and the locations of cells, in which data are inserted, are particularly arbitrary and they are just used in MATLAB program for calling input data form Excel spreadsheet.

The obtained results from analyzing WDN are transformed from MATLAB to Excel spreadsheet as the method satisfies the specified S.C. The output results are (1) CPU time, (2) number of iterations, (3) pipe flow rates or hydraulic heads at nodal points, and (4) D-W friction factor in each pipe. It should be noted that the two last output data is reported for both final iteration and all iterations separately in the output worksheet of Excel so called, "output data". The results in Excel spreadsheet are shown in Figure 3 for better illustration.

MATLAB Codes

The step-by-step process of solving WDN accompanied with MATLAB codes are presented below for the introduced Q-based methods:

(1) Hardy Cross method: The process of this method is illustrated as the following:
   (1.1) First step: transforming input data from Excel into MATLAB: In this step, the inserted input data first should be transformed to MATLAB. In addition to the variables introduced in the input data section, one additional input data, which can be obtained using network layout, is required to solve the sample network using this method. This input data shows



	A	B	C	D	E	F
1	CPU time	No. of iteration				
2	0.107451	28				
3		Results				
4	# pipe	Qe	# pipe	f		
5	1	0.0025	1	0.1009		
6	2	0.0033	2	0.0934		
7	3	0.0383	3	0.0492		
8	4	0.0483	4	0.0474		
9	5	0.1025	5	0.0419		
10	6	0.0275	6	0.0549		
11	7	0.0391	7	0.0507		

	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Run #	$Q^{(1)}$	$Q^{(2)}$	$Q^{(3)}$	$Q^{(4)}$	$Q^{(5)}$	$Q^{(6)}$	$Q^{(7)}$	$f^{(1)}$	$f^{(2)}$	$f^{(3)}$	$f^{(4)}$	$f^{(5)}$
2	1	35.3147	35.3147	35.3147	35.3147	35.3147	35.3147	35.3147	0.0098	0.0096	0.0087	0.0086	0.0083
3	2	0.0000	0.0000	0.0121	0.0159	0.0381	0.0098	0.0137	4.2276	13.5893	0.0736	0.0693	0.0577
4	3	0.0000	0.0000	0.0437	0.0557	0.1201	0.0383	0.0501	45.5598	0.8764	0.0472	0.0453	0.0400
5	4	0.0615	0.0532	0.0462	0.0558	0.0987	0.0267	0.0280	0.0335	0.0361	0.0463	0.0453	0.0424
6	5	0.0003	0.0006	0.0348	0.0448	0.1012	0.0297	0.0422	0.3412	0.2269	0.0508	0.0485	0.0421
7	6	0.0000	0.0000	0.0352	0.0452	0.0999	0.0300	0.0395	1.8886	1.8638	0.0506	0.0484	0.0422
8	7	0.0129	0.0150	0.0380	0.0480	0.1026	0.0274	0.0397	0.0537	0.0533	0.0493	0.0475	0.0419
9	8	0.0012	0.0020	0.0379	0.0479	0.1018	0.0282	0.0389	0.1403	0.1175	0.0494	0.0475	0.0420
10	9	0.0005	0.0011	0.0359	0.0459	0.1005	0.0296	0.0395	0.2133	0.1594	0.0502	0.0481	0.0422
11	10	0.0055	0.0061	0.0398	0.0498	0.1037	0.0263	0.0389	0.0729	0.0733	0.0486	0.0469	0.0418
12	11	0.0024	0.0036	0.0388	0.0488	0.1027	0.0274	0.0388	0.1020	0.0910	0.0490	0.0472	0.0419
13	12	0.0013	0.0021	0.0371	0.0471	0.1013	0.0287	0.0392	0.1334	0.1146	0.0497	0.0478	0.0421
14	13	0.0029	0.0036	0.0384	0.0484	0.1026	0.0273	0.0392	0.0941	0.0906	0.0492	0.0473	0.0419
15	14	0.0030	0.0041	0.0391	0.0491	0.1031	0.0269	0.0390	0.0922	0.0860	0.0489	0.0471	0.0418
16	15	0.0020	0.0029	0.0379	0.0479	0.1020	0.0280	0.0391	0.1102	0.0991	0.0494	0.0475	0.0420
17	16	0.0022	0.0030	0.0380	0.0480	0.1022	0.0278	0.0392	0.1058	0.0978	0.0493	0.0475	0.0419
18	17	0.0028	0.0037	0.0387	0.0487	0.1028	0.0272	0.0391	0.0948	0.0891	0.0490	0.0472	0.0419
19	18	0.0024	0.0034	0.0384	0.0484	0.1025	0.0276	0.0391	0.1015	0.0933	0.0492	0.0473	0.0419
20	19	0.0022	0.0031	0.0381	0.0481	0.1022	0.0278	0.0391	0.1058	0.0970	0.0493	0.0474	0.0419
21	20	0.0025	0.0034	0.0384	0.0484	0.1025	0.0275	0.0391	0.0996	0.0928	0.0492	0.0473	0.0419
22	21	0.0025	0.0035	0.0385	0.0485	0.1026	0.0274	0.0391	0.0993	0.0921	0.0491	0.0473	0.0419
23	22	0.0023	0.0032	0.0382	0.0482	0.1023	0.0277	0.0391	0.1031	0.0949	0.0492	0.0474	0.0419

Figure 3 Analysis results in Excel spreadsheet.

```
%% Input data & Define variables
n = xlsread('example.xlsx','Input data','A2'); % n = number of nodes
m = xlsread('example.xlsx','Input data','B2'); % m = number of pipes
n1 = xlsread('example.xlsx','Input data','C2'); % n1 = number of essential boundary conditions
n2 = n - n1; % n2 = number of natural boundary conditions
E = xlsread('example.xlsx','Input data','K5:L5'); % E = essential boundary conditions matrix
N = xlsread('example.xlsx','Input data','N5:O9'); % N = natural boundary conditions matrix
con = xlsread('example.xlsx','Input data','B5:C11'); % con = connectivity matrix
rr = xlsread('example.xlsx','Input data','G5:G11'); % rr = relative roughness
np = xlsread('example.xlsx','Input data','R2'); % np = n (= power of flow rate in head loss equations)
% np = 2(for Darcy-Weisbach and Manning) & 1.852(for Hazen-Williams)
g= xlsread('example.xlsx','Input data','H2'); %gravity acceleration (L^2/T)
nu= xlsread('example.xlsx','Input data','E2'); %Dynamic viscosity
mu= xlsread('example.xlsx','Input data','G2'); %Kinematic viscosity
stopc= xlsread('example.xlsx','Input data','I2'); %accuracy of stopping criterion
D= xlsread('example.xlsx','Input data','D5:D11'); % D = pipe diameters
L= xlsread('example.xlsx','Input data','E5:E11'); % L = pipe lengths
lo = xlsread('example.xlsx','Input data','A14:D15'); %introducing the closed loops
```

Figure 4 MATLAB code for the first step in Hardy Cross method.

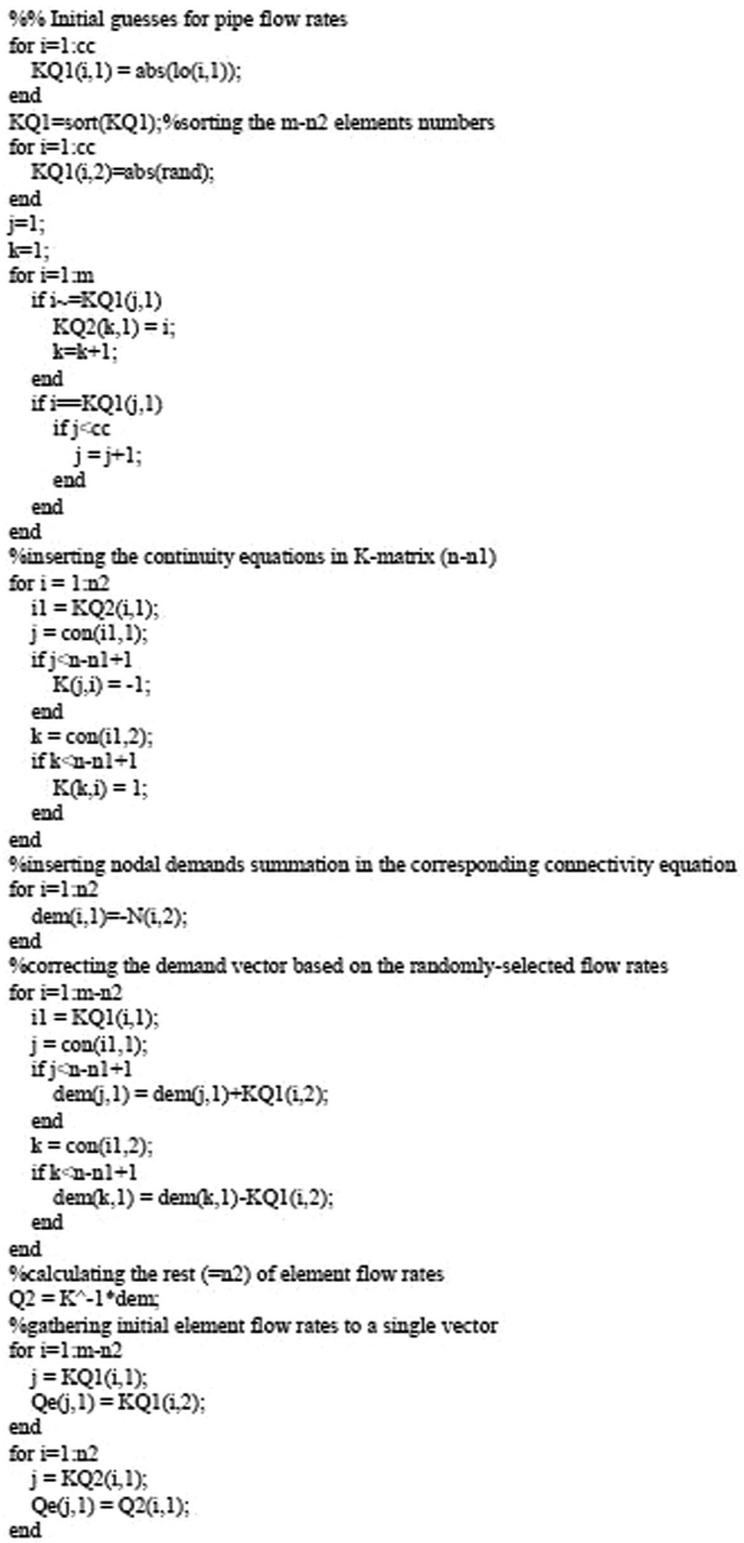
the arrangement of nodal points in different loops of network. Since a clockwise direction is assumed for the water circulation in each loop, the agreement of flow in pipe with the loop direction is also should be specified by designating a sign for nodal number. If the loop correction and flow in pipe have the same direction, positive sign will be used before nodal number. This agreement is exclusively utilized to evaluate the sign function used in Equation (1) in the MATLAB code. The MATLAB code for this step is shown in Figure 4. In this figure, all the input data, which their acronyms in MATLAB are introduced, are called from Excel spreadsheet. As shown in Figure 4, the input data can simply be accessed with MATLAB using a built-in function so called, "xlsread".

(1.2) Second step: defining variables in MATLAB: The variables which are used in this program are all introduced as zero-value matrices or vectors in this step. These variables will be used and evaluated in the process of analyzing WDN later. The maximum number of iteration is also defined in this step. The time elapsed for the method is started from this step. The utilized code for this step is illustrated in Figure 5.

(1.3) Third step: computing random initial guess for Hardy Cross method: As it was previously mentioned, Hardy Cross method uses a set of initial guesses which satisfies the continuity equations. In the provided code, a simple procedure for calculating random values for pipe flow rate as a proper initial guess is presented as depicted in Figure 6. The mechanism of this process is to generate random values for several pipes in each loop and determine the rest using continuity equations. The detail of this procedure is shown in Figure 6. It should be noted that one of the shortcomings of Hardy Cross method

```
%% Program Beginning for computing CPU time
tic
%% Pre-analysis compute
%program constants built-in
counter = 200; % counter = number of turbulent analysis iteration
b = 0;
conv=zeros(counter-1,1); % conv = convergence vector
%Making zero required matrices
K2 = zeros(n,1); % K2 = fixed terms of stiffness matrix
Qe = zeros(m,1); % Qe = pipe flow rate vector
Re = zeros(m,1); % Re = Reynolds number
f = zeros(m,1); % f = f coefficient in Darcy-Weisbach head loss equation
V = zeros(counter,m); % V = matrix of element flow rates of all iterations of analysis
Z = zeros(counter,m); % Z = matrix of f coefficient- values of all iterations of analysis
HL = zeros(m,1);%HL = hydraulic loss
power = zeros(m,1); % power = a vector of m-columns equal to (np-1)
N1 = N;
dem=zeros(n2,1);%dem= nodal demand vector
KQ2=zeros(n2,1);%KQ2= matrix of element numbers of the rest of (=n2) elements
K=zeros(n2,n2);%K = matrix of continuity equation for m-n1 elements which the corresponding
%flow rate will be computed via connectivity equations and the randomly-selected flow rates
KQ1=zeros(m-n2,2);%KQ1= matrix of flow rates of m-n2 elements which are randomly selected
% at first loop data
[cc dd] = size (lo); %cc = number of closed loops
dQ = zeros(cc,1);%dQ = the loop flow correction
cce = zeros(n2,1);%cce = a vector for checking continuity equations at nodal points
```

Figure 5 MATLAB code for the second step in Hardy Cross method.



```
%% Initial guesses for pipe flow rates
for i=1:cc
   KQ1(i,1) = abs(lo(i,1));
end
KQ1=sort(KQ1);%sorting the m-n2 elements numbers
for i=1:cc
   KQ1(i,2)=abs(rand);
end
j=1;
k=1;
for i=1:m
   if i~=KQ1(j,1)
      KQ2(k,1) = i;
      k=k+1;
   end
   if i==KQ1(j,1)
      if j<cc
         j = j+1;
      end
   end
end
%inserting the continuity equations in K-matrix (n-n1)
for i = 1:n2
   i1 = KQ2(i,1);
   j = con(i1,1);
   if j<n-n1+1
      K(j,i) = -1;
   end
   k = con(i1,2);
   if k<n-n1+1
      K(k,i) = 1;
   end
end
%inserting nodal demands summation in the corresponding connectivity equation
for i=1:n2
   dem(i,1)=-N(i,2);
end
%correcting the demand vector based on the randomly-selected flow rates
for i=1:m-n2
   i1 = KQ1(i,1);
   j = con(i1,1);
   if j<n-n1+1
      dem(j,1) = dem(j,1)+KQ1(i,2);
   end
   k = con(i1,2);
   if k<n-n1+1
      dem(k,1) = dem(k,1)-KQ1(i,2);
   end
end
%calculating the rest (=n2) of element flow rates
Q2 = K^-1*dem;
%gathering initial element flow rates to a single vector
for i=1:m-n2
   j = KQ1(i,1);
   Qe(j,1) = KQ1(i,2);
end
for i=1:n2
   j = KQ2(i,1);
   Qe(j,1) = Q2(i,1);
end
```

Figure 6 MATLAB code for the third step in Hardy Cross method.

is determination of proper initial guess since the selection of required initial guess not only is not systematic in this method but also is commonly done manually.

(1.4) Fourth step: computing D-W friction factor for the initial guess: In this step, the presumed values for pipe flow rates are utilized to compute Reynolds number and D-W friction factor in each pipe. The

```
%calculating the friction factor
for i=1:m
   Re(i,1) = (abs(Qe(i,1))*D(i,1))/(0.25*pi()*D(i,1)^2*nu); %Re=VD/nu where V=Q/A (ft/sec)
   %computing f Darcy-Weisbach friction factor (f) depending on Reynolds number value
   if Re(i,1) < 2000 %laminar condition (Re<2000)
      f(i,1) = 64/Re(i,1); %Hagen-Poiseuille
   end
   if 2000 < Re(i,1) && Re(i,1) < 4000 %transient condition (2000<Re<4000)
      R = Re(i,1)/2000;
      Y2 = (rr(i,1)/3.7)+(5.74/Re(i,1)^0.9);
      Y3 = -0.86859*log((rr(i,1)/3.7)+(5.74/4000^0.9));
      FA = (Y3)^-2;
      FB = FA*(2-(0.00514215/(Y2*Y3)));
      X1 = 7*FA-FB;
      X2 = 0.128-17*FA+2.5*FB;
      X3 = -0.128+13*FA-2*FB;
      X4 = R*(0.032-3*FA+0.5*FB);
      f(i,1) = X1+R*(X2+R*(X3+X4));
   end
   if Re(i,1) > 4000 %turbulent condition (Re<4000)
      f(i,1) = (-2*log10((rr(i,1)/3.7)+(5.74/Re(i,1)^0.9)))^-2; % Swamee-Jain(1976)
   end
end
%% Insert iteration results into S, V and f matrices
for i = 1:m
   V(1,i) = Qe(i,1);
end
for i = 1:m
   Z(1,i) = f(i,1);
end
```

Figure 7 MATLAB code for the fourth step in Hardy Cross method.

```
%% Analysis using Hardy Cross method
%producing fixed terms of stiffness matrix in turbulent condition
for i=1:m
   K2(i,1) = (8*L(i,1))/(g*pi()^2*D(i,1)^5);
end
%providing positive flow rates
for i=1:m
   if Qe(i,1)<0
      Qe(i,1) = - Qe(i,1);
      r = con(i,1);
      con(i,1) = con(i,2);
      con(i,2) = r;
      for i1=1:cc
         for i2=1:dd
            k = abs(lo(i1,i2));
            if k==i
               lo(i1,i2) = - lo(i1,i2);
            end
         end
      end
   end
end
%% Checking the continuity equation at each nodal point
cce = zeros(n2,1);
for i=1:n2
   cce(i,1) = N(i,2);
end
for i=1:m
   r = con(i,1);
   if r<n2+1
      cce(r,1) = cce(r,1)-Qe(i,1);
   end
   w = con(i,2);
   if w<n2+1
      cce(w,1) = cce(w,1)+Qe(i,1);
   end
 end
```

Figure 8 MATLAB code for the fifth step in Hardy Cross method.

equation for calculating the D-W friction factor is based on the value of Reynolds number. In turbulent condition, Swamee-Jain equation, which is used in EPANET for this purpose [3], is utilized in the provided MATLAB code (Fig. 7). At the end of this step, the flow rate and D-W friction factor in each pipe are stored in the corresponding matrices designated for saving the corresponding values throughout the analysis process. As it was previously mentioned, these matrices will be transformed from MATLAB into Excel as output data as the method converges.

(1.5) Fifth step: essential computations before starting the iteration: The pipe coefficient for each pipe consists of two parts: one fixed part and one variable part, which should be computed in each iteration as pipe flow rate changes. Since the fixed terms in pipe coefficient remain constant in the whole process, they are computed once before starting the analysis procedure to enhance the performance of the code. Furthermore, since initial guesses are generated randomly, they may be negative. In this step, the corresponding values are once again checked and will be changed if they are negative. Regarding the probable change(s), the continuity equations are once again utilized to make sure whether the positive initial guesses satisfy these equations. In order to better demonstrate the procedure, the MATLAB code of this step is depicted in Figure 8.

(1.6) Sixth step: computing loop corrections in each loop: The assumed values for pipe flow rates should be revised using loop correction (ΔQ) in each iteration. The MATLAB code for computing loop corrections in each loop using Equation (1) is depicted in Figure 9. As shown, the numerator and denominator of Equation (1) are determined for

```
%% Hardy Cross Method
for a = 2:counter
%calculating the flow rate correction for each loop
for i=1:cc
   numerator = 0;
   denominator = 0;
   for j=1:dd
      k = abs(lo(i,j));
      if k>0
      ee = sign(lo(i,j))*K2(k,1)*f(k,1)*Qe(k,1)^np;
      ff = K2(k,1)*f(k,1)*Qe(k,1)^(np-1);
      numerator = numerator + ee;
      denominator = denominator + abs(ff);
      end
   end
   dQ(i,1) = - (numerator/(np*denominator));
end
dQe = zeros(m,1);%dQe = the individual loop flow correction for each pipe
%correcting the flow rate of each pipe based on the obtained loop corrections
for i=1:cc
   for j=1:dd
      k = abs(lo(i,j));
      if k>0
         Qe(k,1) = Qe(k,1) + sign(lo(i,j))*dQ(i,1);
      end
   end
end
%flow rate and loops data corrections
for i=1:m
   if Qe(i,1)<0
      Qe(i,1) = - Qe(i,1);
   r = con(i,1);
   con(i,1) = con(i,2);
   con(i,2) = r;
      for i1=1:cc
         for i2=1:dd
            if abs(lo(i1,i2)) == i
               lo(i1,i2) = - lo(i1,i2);
            end
         end
      end
   end
end
%% Checking the continuity equation at each nodal point
for i=1:n1
   cce(i,1) = N(i,2);
end
for i=1:m
   r = con(i,1);
   if r<n2+1
      cce(r,1) = cce(r,1)-Qe(i,1);
   end
   w = con(i,2);
   if w<n2+1
      cce(w,1) = cce(w,1)+Qe(i,1);
   end
end
```

Figure 9 MATLAB code for the sixth step in Hardy Cross method.

each loop and subsequently the loop corrections are used to improve the pipe flow rates. At the end of this step, wherever a negative pipe flow rate is obtained, the negative sign will be changed into positive and the loop data, which indicates flow directions in pipes and loops, is corrected accordingly. Finally, it should be noted that the iteration calculation is started from this step.

```
%% Stopping criterion
if b == 0
sum = 0;
sum1 = 0;
for i=1:m
   sum = sum + abs(V(a,i) - V(a-1,i));
   sum1 = sum1 + V(a,i);
end
conv(a-1,1)=sum/sum1;
if sum/sum1 < stopc
cpu = toc;
toc %Program end for computing CPU time
aa = a;
display(a)
b = 1;
break
end
end
end
```

Figure 10 MATLAB code for the eighth step in Hardy Cross method.

```
%% Output Data
% Iteration results
xlswrite('example.xlsx',V,'Output data','H2:N37');
xlswrite('example.xlsx',Z,'Output data','O2:U37');
% Final Results
xlswrite('example.xlsx',V(aa,:)','Output data','B5:B11');
xlswrite('example.xlsx',Z(aa,:)','Output data','D5:D11');
xlswrite('example.xlsx',cpu,'Output data','A2');
xlswrite('example.xlsx',aa,'Output data','B2');
xlswrite('example.xlsx',aa,'turbulent','F5');
```

Figure 11 MATLAB code for the ninth step in Hardy Cross method.

```
%% Program Beginning for computing CPU time
tic
%% Pre-analysis compute
%program constants built-in
counter = 200;
b = 0;
conv=zeros(counter-1,1);
%Making zero required matrices
K = zeros(n);
K1 = zeros(n,1);
K2 = zeros(n,1);
Q = zeros(n,1);
Qe = zeros(m,1);
Re = zeros(m,1);
f = zeros(m,1);
V = zeros(counter,m);
Z = zeros(counter,m);
HL = zeros(m,1);
power = zeros(m,1);
%making power matrix
for i = 1:m
   power(i,1) = np-1;
end
Rhs = zeros(m,1);
```

Figure 12 MATLAB code for the second step in Linear Theory.

```
%producing fixed terms of stiffness matrix in turbulent condition
for i=1:m
   K2(i,1) = (8*L(i,1))/(g*pi()^2*D(i,1)^5);
end
for i=1:n-n1
   Rhs(i,1) = - N(i,2);
end
for i=n-n1+1:n-n1+cc
   Rhs(i,1) = 0;
end
```

Figure 13 MATLAB code for the fifth step in Linear Theory.

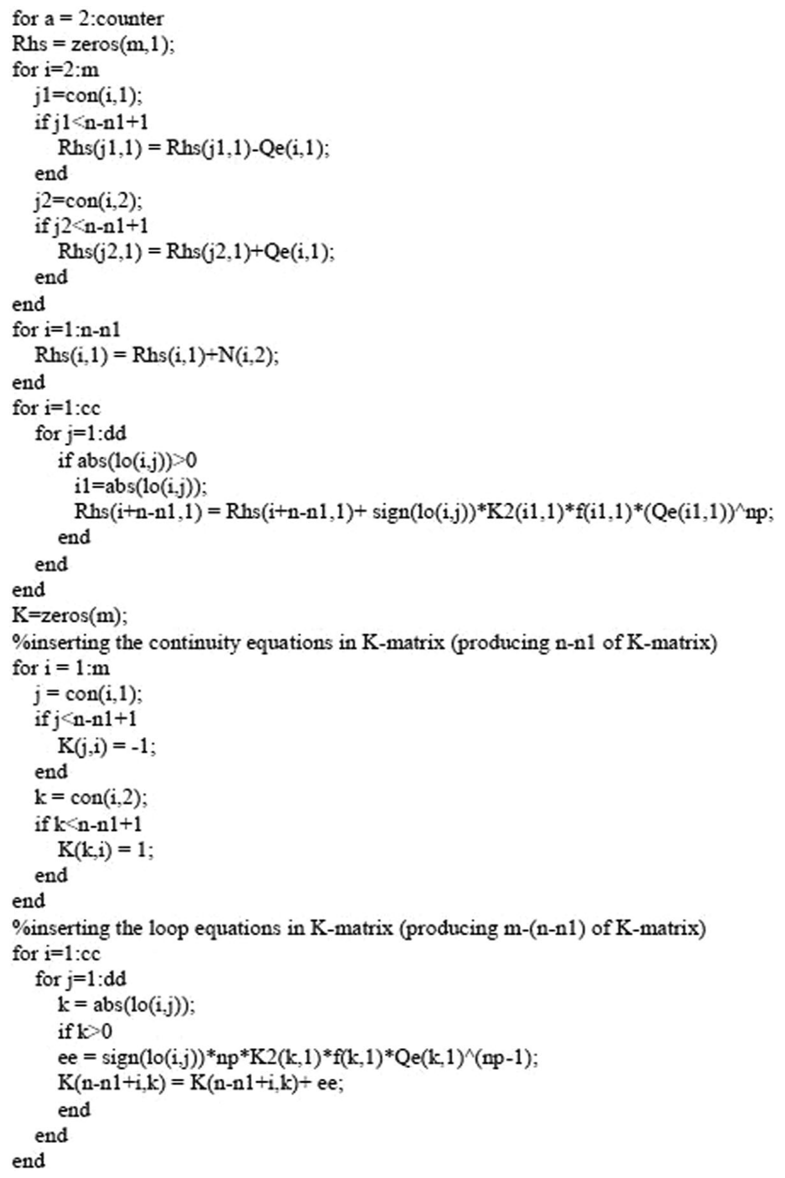
```
for a = 2:counter
K=zeros(m);
%inserting the continuity equations in K-matrix (producing n-n1 of K-matrix)
for i = 1:m
   j = con(i,1);
   if j<n-n1+1
      K(j,i) = -1;
   end
   k = con(i,2);
   if k<n-n1+1
      K(k,i) = 1;
   end
end
%inserting the loop equations in K-matrix (producing m-(n-n1) of K-matrix)
for i=1:cc
   for j=1:dd
      k = abs(lo(i,j));
      if k>0
      ee = sign(lo(i,j))*K2(k,1)*f(k,1)*Qe(k,1)^(np-1);
      K(n-n1+i,k) = K(n-n1+i,k)+ ee;
      end
   end
end
```

Figure 14 MATLAB code for the sixth step in Linear Theory.

```
%% Solve system equations (Determining the unknown vector of flow rates)
Q = K ^ -1 * Rhs;
for i = 1:m
   if Q(i,1)<0
      Q(i,1) = -Q(i,1);
   end
end
%% Computing the hydraulic head loss for each elements
for i=1:m
   HL(i,1) = K2(i,1)*f(i,1)*Q(i,1)^np;
end
%% Explicit Colebrook-White approach
for i=1:m
  B = (2*g*D(i,1)*HL(i,1)*(D(i,1)/nu)^2/L(i,1))^0.5;
  f(i,1) = (-2*log10((rr(i,1)/3.7)+(2.51/B)))^-2;
  Re(i,1) = B/f(i,1)^0.5;
% Do additional computations if required
  Qe(i,1) = pi*nu*D(i,1)*Re(i,1)/4;
end
%% Insert iteration results into S, V and f matrices
for i = 1:m
   V(a,i) = Qe(i,1);
   Qe(i,1) = 0.5*(V(a,i)+V(a-1,i));
end
for i = 1:m
   Z(a,i) = f(i,1);
end
```

Figure 15 MATLAB code for the seventh step in Linear Theory.

(1.7) Seventh step: computing D-W friction factor: Based on the revised flow rates, the D-W friction factor in each pipe is calculated using the same process as the one used in the fourth step. Furthermore, the new values of pipe flow rates and D-W friction factors are stored in the corresponding output matrices as



```
for a = 2:counter
Rhs = zeros(m,1);
for i=2:m
   j1=con(i,1);
   if j1<n-n1+1
      Rhs(j1,1) = Rhs(j1,1)-Qe(i,1);
   end
   j2=con(i,2);
   if j2<n-n1+1
      Rhs(j2,1) = Rhs(j2,1)+Qe(i,1);
   end
end
for i=1:n-n1
   Rhs(i,1) = Rhs(i,1)+N(i,2);
end
for i=1:cc
   for j=1:dd
      if abs(lo(i,j))>0
        i1=abs(lo(i,j));
        Rhs(i+n-n1,1) = Rhs(i+n-n1,1)+ sign(lo(i,j))*K2(i1,1)*f(i1,1)*(Qe(i1,1))^np;
      end
   end
end
K=zeros(m);
%inserting the continuity equations in K-matrix (producing n-n1 of K-matrix)
for i = 1:m
   j = con(i,1);
   if j<n-n1+1
      K(j,i) = -1;
   end
   k = con(i,2);
   if k<n-n1+1
      K(k,i) = 1;
   end
end
%inserting the loop equations in K-matrix (producing m-(n-n1) of K-matrix)
for i=1:cc
   for j=1:dd
      k = abs(lo(i,j));
      if k>0
      ee = sign(lo(i,j))*np*K2(k,1)*f(k,1)*Qe(k,1)^(np-1);
      K(n-n1+i,k) = K(n-n1+i,k)+ ee;
      end
   end
end
```

Figure 16 MATLAB code for the sixth step in Q-based Newton-Raphson method.

the achieved solution in this iteration. At the end of this step, the continuity equations is once again checked to ensure that the obtained pipe flow rates satisfy these equations.

(1.8) Eighth step: checking the stopping criterion: In this step, the values of pipe flow rates of two successive iterations are compared and the absolute relative error is computed. In this regard, the analysis will be terminated if S.C. is less than the specified value, that is 0.001 in the sample network. Otherwise, the 6th to 8th steps should be repeated until the code converges. The MATLAB code for this step is shown in Figure 10.

(1.9) Ninth step: reporting the obtained results: As the Hardy Cross method converges, the obtained results are transformed from MATLAB into Excel spreadsheet for ease of further use. The MATLAB code for this step is illustrated in Figure 11.

(2) Linear Theory: The step-by-step process of computer application for implementation of this Q-based method is presented below:

(2.1) First step: transforming input data from Excel into MATLAB: Since the same input data as the ones in Hardy Cross method is used in Linear Theory, the MATLAB code illustrated for Hardy Cross method can be utilized for this step, too.

(2.2) Second step: defining variables in MATLAB: First of all, the CPU time for this code is started from this step. Second, the maximum number of iterations is set. Third, the variables used in this program are all introduced as zero-value matrices or vectors in this step. The MATLAB code for this step is presented in

```
%% Solve system equations (Determining the unknown vector of flow rates)
dQ = K ^ -1 * Rhs;
Qe = Qe - dQ;
for i=1:m
   r = con(i,1);
   w = con(i,2);
   if Qe(i,1)<0
     r = con(i,1);
     con(i,1) = con(i,2);
     con(i,2) = r;
     Qe(i,1) = - Qe(i,1);
      for i1=1:cc
         for i2=1:dd
            if abs(lo(i1,i2)) == i
               lo(i1,i2) = - lo(i1,i2);
            end
         end
      end
   end
end
```

Figure 17 MATLAB code for the first part of the seventh step in Q-based Newton-Raphson method.

Figure 12. As this figure shows, the utilized variables defined in this method are different from the ones in Figure 5 for Hardy Cross method.

(2.3) Third step: assuming initial guess for Linear Theory: In Linear Theory, the initial guess for flow rate in all pipes is assumed to be one cubic meters per second [14]. By considering this initial guess, the energy equations become linear and the system of equations for this method will become a linear algebraic one.

(2.4) Fourth step: computing D-W friction factor for the assumed initial guess: The fourth step of this method is exactly like the fourth step in Hardy Cross method except that the initial guesses assumed in both methods are different. Therefore, the MATLAB code provided for the fourth step in Hardy Cross method can be utilized for this step, too.

(2.5) Fifth step: essential computations before starting the iteration: In this step, the fixed part of pipe coefficient for each pipe is first computed to avoid the repentance of its calculation in each iteration. Moreover, as the right hand side of Equation (2) is also constant throughout WDN analysis, it is computed before starting the iterations (Fig. 13).

(2.6) Sixth step: computing the left hand side matrix: The left hand side matrix of Equation (2), which multiplies into the unknown vector of pipe flow rates, is computed in this step. This matrix consists of the coefficients of continuity and energy equations which calculation of each part is conducted in two separate loops in the MATLAB code as shown in Figure 14.

(2.7) Seventh step: solving the system of equations casted in Linear Theory: As the Q-based system used in Linear Theory is provided, the unknown vector of pipe flow rates is determined simply by multiplying the inverse of the left hand side matrix into the right hand side vector. The computed flow rate values are checked and in case they are negative, their sign will be changed. Afterwards, the hydraulic head loss is computed using the D-W resistance equation based on the new computed pipe flow rates. The computed hydraulic heads are used to calculate the D-W friction factor in each pipe. The Reynolds number in each pipe is computed using the corresponding D-W friction factor and the pipe flow rate is recomputed using the obtained Reynolds number. In order to improve the efficiency of Linear Theory, flow rate in each pipe is replaced with arithmetic mean of flow rate in two last successive iterations. These flow rates and D-W friction factors are stored in matrices specified for output results. The MATLAB code of this step is shown in Figure 15.

(2.8) Eighth step: checking the stopping criterion: The same procedure as the one for Hardy Cross method should be conducted here.

(2.9) Ninth step: reporting the obtained results: As the MATLAB code of Linear Theory satisfies the stopping criterion, the obtained results are transformed from MATLAB into Excel spreadsheet.

(3) Q-based Newton-Raphson method: The procedure of WDN analysis using this method is illustrated as following:

(3.1) to (3.4): The first to the fourth steps in this method are the same as the first fourth ones in Linear Theory. In these steps, input data is transformed from Excel into MATLAB and required variables are defined. A velocity equal to one foot per second is assumed in all pipes of the network [3]. Afterwards, D-W friction factor in each pipe is

Table 1 Final Results of Analyzing the Sample Network

Hardy Cross			Linear theory			Q-based Newton-Raphson		
# pipe	*Q*	*f*	# pipe	*Q*	*f*	# pipe	*Q*	*f*
1	0.0026	0.0243	1	0.0027	0.0241	1	0.0028	0.0238
2	0.0045	0.0220	2	0.0036	0.0232	2	0.0037	0.0229
3	0.0305	0.0169	3	0.0386	0.0162	3	0.0387	0.0161
4	0.0405	0.0163	4	0.0486	0.0159	4	0.0487	0.0158
5	0.0974	0.0148	5	0.1027	0.0147	5	0.1028	0.0146
6	0.0326	0.0167	6	0.0273	0.0174	6	0.0272	0.0173
7	0.0419	0.0162	7	0.0391	0.0165	7	0.0391	0.0164
# of iteration		12	# of iteration		18	# of iteration		5
CPU time		0.2684	CPU time		0.1214	CPU time		0.1919

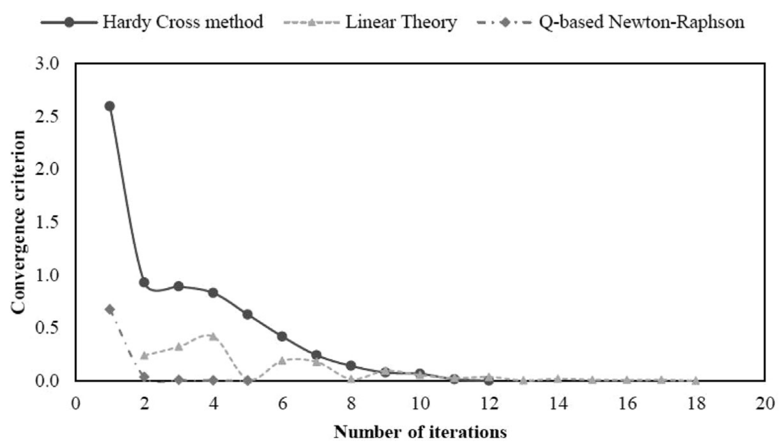


Figure 18 Variation of convergence criterion in WDN analysis for different Q-based methods.

computed for the assumed initial guess. The MATLAB codes for these steps are similar to the ones provided for the first four steps in Linear Theory.

(3.5) Fifth step: essential computations before starting the iterations: Before commencing the iteration procedure, the fixed part of pipe coefficient for each pipe is calculated which will be later used to compute the right hand side vector of Equation (3) in each iteration.

(3.6) Sixth step: computing the left and right hand side matrices: Unlike the right hand side vector in Linear Theory, the one in this method depends upon pipe flow rates and subsequently is not constant in the whole process of analysis. Therefore, this vector was not computed in the fifth step (previous step in this method) where constant parameters were calculated before starting iterations. On the other hand, the left hand side matrix is also a function of pipe flow rates. Hence, these two matrices are computed in each iteration in this step as depicted in Figure 16.

(3.7) Seventh step: solving the system of equations casted in Q-based Newton-Raphson method: In each iteration, the change in flow rate in each pipe is directly obtained by solving the system of equations. These values are utilized to determine pipe flow rates. The sign of each computed flow rate will be changed unless it is positive. It should be noted that changing the sign of negative flow rate(s) means that the assumed direction for flow rate in the pipe is corrected. Therefore, wherever the sign of pipe flow rate is changed, the loop data is updated to take into account this change (Fig. 17). This update should be conducted to the loop data since this data is used in each iteration for calculating left and right hand side matrices. Then, the Reynolds number and D-W friction factor in each pipe are computed using the same process as the one illustrated in the fourth step of Hardy Cross method. The arithmetic mean of flow rate in two successive iterations is used instead of computed flow rates to enhance the efficiency of this method. The achieved flow rates and D-W friction factors are transformed in specific matrices used for output results.

(3.8) to (3.9): The eighth and ninth steps are the same as the ones in Linear Theory and finally the obtained results are transformed from MATLAB into Excel spreadsheet.

RESULTS

The sample network is analyzed using the three Q-based codes and the achieved results, which are transformed from MATLAB to Excel spreadsheet, are shown in Table 1. Since the considered initial guesses are different for these methods, their number of iteration and CPU time cannot be technically compared with one another. However, the stopping criterion, that is $\frac{\sum_{i=1}^{m}|Q_i^{k-1}-Q_i^k|}{\sum_{i=1}^{m}Q_i^k}$, is computed for each method and its variation is depicted in Figure 18 to illustrate how different method converges to the final solution. This convergence criterion is dimensionless where Q_i^k denotes the flow rate in the i$^{\text{th}}$ pipe obtained in the k$^{\text{th}}$ iteration. Since the stopping criterion for the first iteration of Linear Theory was order of magnitudes larger than other data in Figure 18, it was removed for better illustration. Although the presented MATLAB codes solve a relatively simple WDN in light of more focusing on implementation of Q-based methods, the codes can be modified to solve other and much more complex networks as well. Finally, since different Q-based methods are successfully coded for the sample network, similar appraisal of MATLAB and Excel spreadsheet for both academic and practical purposes is recommended.

CONCLUSION

In spite of MATLAB and Excel spreadsheet facilities, appraisal of these programs has not been adequately addressed in solving WDNs in the literature. In this paper, the implementation of three Q-based methods including Hardy Cross method, Linear Theory, and Q-based Newton-Raphson is presented. In these codes, first the input data is inserted into specific cells of Excel spreadsheet. Afterwards, the MATLAB codes, which are written for each of Q-based methods, read the input data from Excel and conduct WDN analysis. The procedure of solving WDNs is revisited in a step-by-step manner accompanied with MATLAB codes. As these codes solve the WDN under consideration, the results are transformed from MATLAB to Excel for convenience of future usage. Finally, not only these softwares are successfully utilized to solve a sample pipe network, but also the basics and codes are presented in way that the applicants can simply improve them to solve other networks for both educational and practical purposes.

REFERENCES

[1] R. W. Jeppson, Analysis of flow in pipe networks. Sixth printing. Ann Arbor Science, Ann Arbor, Michigan, 1983.

[2] B. E. Larock, R. W. Jeppson and G. Z. Watters, Hydraulics of pipeline systems. CRC Press, New York, 2000.

[3] L. A. Rossman, EPANET 2.0 User's manual, water supply, and water resource division. US Environmental Protection Agency, Cincinnati, OH, 2000.

[4] Bentley Systems, Incorporated 2006 WaterGEMS v8 User manual. 27 Siemon Co Dr, Suite200W, Watertown, CT06795, USA, 2006.

[5] D. H. Huddleston, V. J. Alarcon and W. Chen, Water distribution network analysis using excel, J Hydraul Eng 130 (2004), 1033–1035.

[6] A. M. G. Lopes, Implementation of the Hardy-Cross method for the solution of piping networks, Comput Appl Eng Educ 12 (2004), 117–125.

[7] E. M. Wahba, An improved computational algorithm for teaching hydraulics of branching pipes in engineering curricula, Comput Appl Eng Educ 23 (2015), 537–541.

[8] H. Zhang, J. L. Xu and X. Cheng, Interactive educational software for teaching water distribution networks design, Comput Appl Eng Educ 24 (2016), 615–628.

[9] G. Recktenwald, Pipe flow analysis with MATLAB. Mechanical and materials engineering department, Portland State University, 2007.

[10] M. Behbahani-Nejad and A. Bagheri, The accuracy and efficiency of a MATLAB-Simulink library for transient flow simulation of gas pipelines and networks, J Petrol Sci Eng 70 (2010), 256–265.

[11] D. Brkic, Spreadsheet-based pipe networks analysis for teaching and learning purpose, eJSiE 9 (2016), 4.

[12] M. Niazkar and S. H. Afzali, Analysis of water distribution networks using MATLAB and Excel spreadsheet: h-based methods, Comput Appl Eng Educ 25 (2017), 129–141.

[13] M. Niazkar and S. H. Afzali, Streamline performance of Excel in stepwise implementation of numerical solutions, Comput Appl Eng Educ 24 (2016), 555–566.

[14] D. J. Wood and C. O. Charles, Hydraulic network analysis using linear theory, J Hydraul Eng 98 (1972), 1157–1170.

BIOGRAPHIES

Majid Niazkar received his BS degree in Civil Engineering (2012) and MS degree in Hydraulic Structures (2014) both from the Department of Civil and Environmental Engineering, Shiraz University, Shiraz, Iran. He is currently a PhD candidate of the Department of Civil and Environmental Engineering, Shiraz University, Shiraz, Iran, in major of Water Resources Management. His research interests include hydraulic engineering, parameter estimation, water distribution systems, optimization problems, river engineering. He has published several papers in international journals and conference proceedings.

Seied Hosein Afzali received the BS degree from the Department of Civil and Environmental Engineering, Shiraz University, Shiraz, Iran, in 1986, and the MSc degree from the Department of Civil Engineering, Amirkabir University of Technology, Tehran, Iran, in 1988. His PhD is in Hydraulic Structures (2008) from the Department of Civil and Environmental Engineering, Shiraz University, Shiraz, Iran. Dr. Afzali is currently an associate professor of Civil Engineering at Shiraz University, Shiraz, Iran. His research interests include groundwater, hydraulic structures, river engineering, optimization problems, and fluid mechanics. He is the author of more than 55 papers in national and international journals and conference proceedings. He is also the first author of a textbook about open channel hydraulics translated in Persian.