Journal of
Applied Research in Water and Wastewater

Journal homepage: www.arww.razi.ac.ir

Original paper

Series and parallel pipelines system solutions using Regula Falsi method scripted in Mathematica

Majid Rahimpour[1,*], Mohamad Reza Madadi[2]

[1]Department of Water Engineering, Faculty of Agriculture, Shahid Bahonar University of Kerman, Kerman, Iran.
[2]Department of Water Engineering, Faculty of Agriculture, Jiroft University, Jiroft, Iran.

ARTICLE INFO

Article history:
Received 19 January 2019
Received in revised form 27 February 2019
Accepted 20 March 2019

Keywords:
Hydraulic design
Pipeline systems
Mathematica
Frictional head losses

ABSTRACT

Friction factor is an important hydraulic parameter for design of pipeline systems. There are several formulations for calculating the friction factor, among which Colebrook–White equation is the most accurate and repute formula. Owing to the implicit nature of friction factor in Colebrook–White equation, iterative methods are required to calculate this factor. In this study, Regula Falsi iterative numerical scheme was used to solve the implicit nonlinear equation of friction factor in the Mathematica programming tool. Case examples including different series and parallel pipeline systems were presented and solved. The results indicated high capability of Regula Falsi method in solving both the parallel and series systems. It was found that the solution by Mathematica differ significantly from conventional methods and can be desirably used for solving different hydraulic problems. The use of Mathematica with its huge features permits the researchers to be more professional in formulations of engineering problems and interpretations of results.

1. Introduction

Friction factor is one of the most important hydraulic parameters in design of close conduits and pipeline networks. Accurate determination of this parameter can directly influence the pipe and pump sizing and reduce the total costs of such projects. Furthermore, the hydraulic balance of pipeline network is established by correct estimation of friction factor. Therefore, the determination of friction factor has always been of great interest to engineers and scientists. In this way, several theoretical and empirical formulas were presented for determination of friction factor including those documented in literature (Churchill. 1977; Haaland. 1983; Slatter. 1995; Avci and Karagoz. 2009; Papaevangelou et al. 2010; Danish et al. 2011; Swamee and Aggarwal. 2011; Morrison. 2013 and Taler. 2016). During the last decade, worthy attempts were applied for accurate calculating the friction factor. Sonnad and Goudar (2008) presented a simple formula for computing the friction factor at rough pipes with turbulent flow. The accuracy of this formula was somewhat acceptable but it was limited to the turbulent flow regime. Diniz and Souza (2009) presented a number of explicit relations to calculate friction factor for different flow regimes. These relations did not have high precision. Li et al. (2011) presented an explicit equation for accurate calculation of friction factor at smooth pipes. The equation was not applicable for all pipe flow regimes. Cojbasic and Brkic (2013) developed two explicit approximations based on genetic algorithms optimization for calculation of the Colebrook friction factor. The models were very accurate so that the relative error was negligible for both models. The main disadvantage of these models was their high complexity. Li and Huai (2016) derived an explicit equation for calculating friction factor in the fully turbulent flows and stated that the proposed model exhibits high computational accuracy for the pipe experiment data.

This relative precise model could not be generalized to all flow regimes. Offor and Alabi (2016) developed a regression model in explicit form consisting two non-linear functions. They claimed that their model was more accurate and requires far less computational time for predicting pipe friction factor. Albeit, this model was only usable for fully developed turbulent flow regime.

As seen, several models were recently proposed to substitute implicit Colebrook White equation. But almost all these models were developed by considering the accuracy only, without accounting the computational burden. Indeed, all of proposed models require the computation of many complicated non-linear equations to achieve a sufficiently accurate solution without attention to this point that most realistic computer resources cannot easily manage such an excessive burden. That's while, solution of practical engineering problems (design of pipeline systems) does not require such precision. So it can be claimed that these models are obviously too complex to be of practical use.

Nevertheless, it should be admitted that one of the best approximations to friction factor is given by the well-known Colebrook–White equation (Coban 2012; Turgut et al. 2014). This equation that represents friction factor as a dependent of Reynolds number and relative roughness, is an implicit-form equation that need an iterative procedure to be determined.

The main purpose of this study is to present hydraulic engineers with a practical application of Mathematica in hydraulic engineering, for accurate determination of friction losses in both series and parallel pipeline systems. To this purpose, the simple and very accurate Regula Falsi iterative numerical scheme within the Mathematica programming tool (Bahder 1994) was used to solve the implicit nonlinear equation of friction factor. Regardless of the complexity of pipeline system, all related problems can be solved over a desired length of a pipe using the energy-equation, as described in following section. The main contribution of this work is preserving the accuracy of estimates while reducing the computational burden. This study uses simple and efficient method of Regula Falsi to solve a set of parallel and series pipelines systems problems. Furthermore, the Mathematica procedures discussed here indicate several features of Mathematica and illustrate how these features may be used to solve all different pipe flow problems. The next purpose of this work was to present a new paradigm

*Corresponding author Email: Rahimpour@uk.ac.ir

for engineering computations and education by arithmetic systems such as Mathematica.

2. Materials and methods

2.1. Governing equations

The continuity principle for an incompressible fluid which steadily flows in a pipe is presented as equation1 (Larock et al. 2000).

$$Q = \int_A v dA = AV \tag{1}$$

where, Q is flow discharge, V is mean velocity and A is pipe cross-sectional area. The next basic principle is the energy equation. Considering Fig. 1, the energy equation can be written for steady one-dimensional flow as equation 2.

$$\frac{P_a}{\gamma} + \frac{V_a^2}{2g} + Z_a = \frac{P_b}{\gamma} + \frac{V_b^2}{2g} + Z_b + \sum h_{L_{a-b}} - h_m \tag{2}$$

where, $V^2/2g$, P/ γ and Z are velocity, pressure and elevation head, respectively. $\sum h_L$ is the head losses between a and b, and h_m is the energy head by hydraulic machinery.

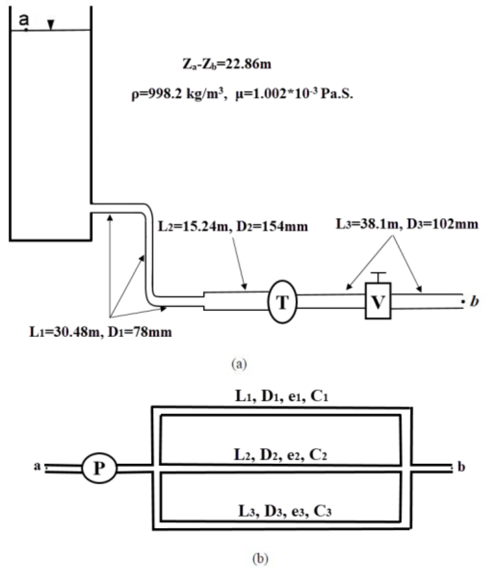


Fig. 1. Pipeline system schematic; (a) Series pipeline system for example 1; (b) Parallel pipeline system.

The Darcy–Weisbach equation and the Colebrook–White formula are the most accurate relations for calculating the major head losses (Larock et al. 2000). The Darcy–Weisbach equation can be demonstrated as equation 3.

$$h_f = f\,\frac{L}{D}\frac{V^2}{2g} \tag{3}$$

or

$$h_f = f\,\frac{L}{D}\frac{8Q^2}{g\pi^2 D^4} \tag{4}$$

in which, f, L, D, V and g denote the: Darcy–Weisbach friction factor, pipe length, pipe diameter, mean velocity, and gravity acceleration, respectively. For turbulent flow condition, friction factor can be calculated by the implicit Colebrook–White semi-theoretical formula (Larock et al. 2000).

$$\frac{1}{\sqrt{f}} = -2log\left(\frac{2.51}{R_e\sqrt{f}} + \frac{e}{3.7D}\right) \tag{5}$$

where, e is the equivalent roughness, $R_e = 4\rho Q/(\pi\mu D)$ is the Reynolds number, ρ is the fluid density, and μ is the dynamic viscosity. If $R_e \to \infty$, equation 5 becomes:

$$\frac{1}{\sqrt{f_T}} = -2log\left(\frac{e}{3.7D}\right) \tag{6}$$

where, f_T is fully rough friction factor. For laminar flow, $f = 64/R_e$ (Streeter et al. 1998). The minor losses are usually obtained as a multiple of an entrance loss, or an equivalent lengths of pipe (Hodge and Taylor 1999). References (Shames 1982; Hodge and Taylor 1999; Larock et al. 2000) are valid sources for determining the minor loss coefficients (K). The friction factor and K are used in the following sections as series and parallel pipeline systems for which examples are presented and discussed. Mathematica solutions for both pipe systems are considered in detail in the following sections.

2.2. The Regula Falsi iterative scheme

The Regula Falsi method is an iterative algorithm, a subset of the bracketing methods, for finding roots of equations. Regula Falsi always converges and has versions that do well at avoiding slowdowns, so it is a good choice when speed is needed. Here, a brief introduction to this algorithm is presented (Conte, 1980). A function f(x) on floating number x and two numbers 'a' and 'b' exists so that f(a)*f(b) < 0 and f(x) is continuous in [a, b]. Suppose that in the k-th iteration the bracketing interval is (a_k, b_k). As illustrated, a line through the points (ak, f (ak)) and (bk, f (bk)) is constructed. The root is being approximated by replacing the actual function by a line segment on the bracketing interval.

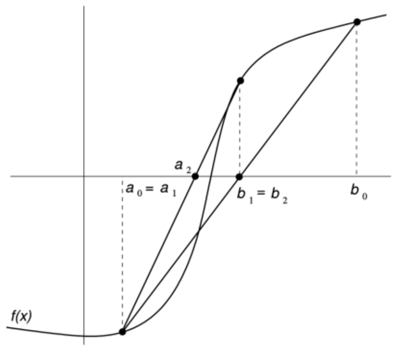


Fig. 2. Representation of Regula Falsi method.

This line is a secant or chord of the graph of the function *f*. In point-slope form, its equation is given by:

$$y - f(b_k) = \frac{f(b_k) - f(a_k)}{b_k - a_k}(x - b_k) \tag{7}$$

Now choose c_k to be the x-intercept of this line, that is, the value of *x* for which *y=0*, and substitute these values to obtain:

$$f(b_k) + \frac{f(b_k) - f(a_k)}{b_k - a_k}(c_k - b_k) = 0 \tag{8}$$

Solving this equation for c_k gives:

$$c_k = b_k - f(b_k) + \frac{b_k - a_k}{f(b_k) - f(a_k)} = \frac{a_k f(b_k) - b_k f(a_k)}{f(b_k) - f(a_k)} \tag{9}$$

As a solution is approached, a_k and b_k will be very close together, and nearly always of the same sign. Such a subtraction can lose significant digits. Because $f(a_k)$ and $f(b_k)$ are always of opposite sign, the subtraction in the numerator of the improved formula is effectively an addition as is the subtraction in the denominator too. At iteration number k, the number c_k is calculated as above and then, if $f(a_k)$ and $f(c_k)$ have the same sign, set $a_{k+1}=c_k$ and $b_{k+1}=b_k$, otherwise set $a_{k+1}=a_k$ and $b_{k+1}=c_k$. This process is repeated until the root is approximated sufficiently well. Though Regula Falsi always converges, there may be some situations that can slow its convergence like all of the numerical equation-solving methods. To solve this problem, a number of improvements to Regula Falsi have been proposed like the Illinois algorithm and the Anderson−Björk algorithm, among others (Ford 1995; Galdino. 2011).

2.3. Series pipeline system

Series layout for pipeline systems means that each pipe elements (pipe segments, valves, pumps, turbines, etc.) arranged in series. In this layout, the flow rate is identical in all elements. Fig. 1a illustrates such a system. The energy equation (equation 2) for this system becomes:

$$\frac{P_a}{\gamma}+\frac{V_a^2}{2g}+Z_a=\frac{P_b}{\gamma}+\frac{V_b^2}{2g}+Z_b+\sum_{i=1}^{M}\frac{8}{g\pi^2}\frac{Q^2}{D_i^4}\left(f_i\frac{L_i}{D_i}+K_i+C_if_{T_i}\right)-h_m \qquad (10)$$

where, P_a, P_b and V_a, V_b are the pressure and velocity at section a and b, respectively, Z_a and Z_b are the elevation at point a and b, respectively, M is the number of each pipe, D_i is the diameter, L_i is the length, f_i is the Darcy–Weisbach friction factor, K_i is the minor loss coefficient, C_if_{Ti} is the minor loss coefficient for pipe i.

Three different types of problem exist in series pipeline systems: in category I, hp is the unknown; in category II, Q is unknown; and in category III pipe diameter should be obtained. The way of solution for category I problems is direct, while for Category II and III is iterative. The same Mathematica scheme can be applied for solving all categories of series pipeline problems. Example (1) indicates several capabilities of proposed Mathematica procedure in solving of the series pipeline problem.

Example 1: Problem description: Water with 20 ^{0}C temperature, flows from a reservoir through a series pipeline system as demonstrated in Fig. 1a. Pipe material is wrought iron. For this system:
(1) Determine Q if the turbine doesn't exist in the system;
(2) Determine the extracted power by turbine if Q=0.00453 m^3/sec;
(3) Determine Q if 2 hp ia extracted by the turbine;
(4) Determine the relationship between Q and power extracted for this system.

2.4. Parallel pipeline system

In parallel pipeline systems, the role of energy loss and flow discharge are reversed from their roles in series systems. In series pipes, the flow rate is identical in all pipes while the energy losses are additive. But for a parallel pipes system, the energy losses between two given junctions is identical for each pipe while the total flow rate equals to sum of the individual discharges. Parallel pipeline systems, as demonstrated in Fig. 1b, have long been solved by iterative methods as explained here. The mass conservation principle for the parallel pipeline system at Fig. 1b can be stated as:

$$Q_1+Q_2+Q_3=Q_T \qquad (11)$$

If the pump in Fig. 1b increases the flow head to make P_a=P_b, then the energy equations can be written as follow (Hodge and Taylor. 2002).

$$\begin{aligned}
Z_b-Z_a+\frac{8}{g\pi^2}\frac{Q_1^2}{D_1^4}\left(f_1\frac{L_1}{D_1}+C_1f_{T_1}+K_1\right)&=h_m\\
Z_b-Z_a+\frac{8}{g\pi^2}\frac{Q_2^2}{D_2^4}\left(f_2\frac{L_2}{D_2}+C_2f_{T_2}+K_2\right)&=h_m\\
Z_b-Z_a+\frac{8}{g\pi^2}\frac{Q_3^2}{D_3^4}\left(f_3\frac{L_3}{D_3}+C_3f_{T_3}+K_3\right)&=h_m
\end{aligned} \qquad (12)$$

Equations 11 and 12 show the system of equations which will be utilized in the Mathematica program to solve problems of parallel pipeline systems. The needed information for the program will be prepared similar to that expressed for series pipeline problems. The two prevalent parallel pipeline problem types are known h_m, find Q_T, Q_1, Q_2, and Q_3, or known Q_T, find h_m, Q_1, Q_2, and Q_3. Although the first type of problems can be placed in Category I, the solution technique presented herein makes use of the Mathematica program for all types of parallel pipeline systems. Example (2) is presented for a parallel pipeline system.

Example 2 Problem description

Fig. 1b displays a parallel pipeline system with three pipes. For this system, Z_a=Z_b, $\rho=701kg/m^3$ and $\mu=0.00051N.s/m^2$. Table 1 presents details of each pipe (Hodge and Taylor. 2002). For this system:
(1) Determine the increase in head and power imparted to the fluid for Q=0.036 m^3/s;
(2) Determine QT if the increase of pump head was 51 m;
(3) Determine the power delivered to the fluid if a 10-kW booster pump was existed in line 3 and the QT were maintained at 0.036 m^3/s.

Table 1. Pipe Characteristics for Example 2.

Pipe	D, cm	L , m	e, mm	K	C
1	5	60	0.1	0	60
2	5	60	0.1	0	60
3	4	55	1.0	1.5	60

3. Results and discussion
3.1. Example 1 solution

The first step is to apply and reduce the energy equation for the system.

$$\begin{aligned}
\frac{P_a}{\gamma}+\frac{V_a^2}{2g}+Z_a=\frac{P_b}{\gamma}+\frac{V_b^2}{2g}+Z_b+\frac{8}{g\pi^2}\frac{Q^2}{D_1^4}\left(f_1\frac{L_1}{D_1}+K_{ent}+2K_{elbow}\right)\\
+\frac{8}{g\pi^2}\frac{Q^2}{D_2^4}\left(f_2\frac{L_2}{D_2}+K_{\exp}\right)+\frac{8}{g\pi^2}\frac{Q^2}{D_3^4}\left(f_3\frac{L_3}{D_3}+K_{gv}\right)-h_m
\end{aligned} \qquad (13)$$

where, h_m is positive for a pump and negative for a turbine. For this system

$$P_a=P_b\ ,V_a=0\ \text{and}\ \frac{V_b^2}{2g}=\frac{8Q^2}{g\pi^2D_3^4} \qquad (14)$$

From Hodge and Taylor (2002) and Shames (1982), values of the minor loss coefficients for the system of Example 1 are $K_{ent}=0.5$, $K_{elbow}=30f_T$, $K_{\exp}=9$ and $K_{gv}=55f_T$. So, the energy equation reduces to

$$\begin{aligned}
Z_a-Z_b+h_m=\frac{8}{g\pi^2}\frac{Q^2}{D_1^4}\left(f_1\frac{L_1}{D_1}+0.5+60f_{T_1}\right)\\
+\frac{8}{g\pi^2}\frac{Q^2}{D_2^4}\left(f_2\frac{L_2}{D_2}+9\right)+\frac{8}{g\pi^2}\frac{Q^2}{D_3^4}\left(f_3\frac{L_3}{D_3}+55f_{T_3}+1\right)
\end{aligned} \qquad (15)$$

Equation15 will be used as the basis for solution for all parts of this problem. Of particular interests are the similarities of the Mathematica solution for each part and the general congruence of the solution of each part with the problem statement.

Part (1): The flow rate if the turbine does not exist in the system. This is a problem from Category II, where the flow rate should be calculated for h_m=0 (no turbine or pump). The Mathematica program for the solution is scripted as Fig. 2. For this part of the problem, the unknown is the flow rate, Q, but f_1, f_2 and f_3 are function of Q, thus for calculation of friction factor we have an implicit equation, and requires either by an iterative numerical scheme or by utilizing the Moody diagram. However, these procedures are not simple.

In the Mathematica program, the first part (a) determines the values of the variables, constants, physical properties and functional definitions for the fully rough friction factor. The second part (b) is the Regula Falsi iterative numerical scheme for solving the implicit nonlinear equation. The third part (c) is the main program. For a problem of Category, I or III, only the required solution variable (and an initial estimate) should be changed. For Part 1 of Example 1, the Q for system with no turbine (or pump) is 0.029m3/sec.

Part (2): The extracted power by turbine if Q= 0.00453 m^3/sec. This is a problem within Category I and can be directly calculated. Fig. 3 presents that portion of the Mathematica program. The omitted part of the program in Fig. 3 is identical to the part (a) and (b) of the program in Fig. 2. The run of program provides the turbine decrease in head, 22.21 m, from which the power extracted is computed to be 1.321 hp. The negative signs on the program run indicate a turbine and power extracted.

Part (3): The flow rate if 2 hp were extracted by the turbine. Fig. 4 represents the relevant part of the Mathematica program for this solution. As illustrated in Fig. (3), the omitted part of the program is similar to the parts (a) and (b) of Fig. (2). The first solution, with an initial flow rate estimate of 0.00453 m^3/sec, yields a flow rate of 0.0246 m^3/sec with a turbine head reduction of 6.19 m. As confirmation, the power is calculated to be the required 2 hp. However, a second solution of flow rate is 0.00714 m^3/sec and a turbine head reduction of 21.33 m. For the second solution, the extracted power is again 2 hp. Hence, the solution is double valued in Q for a given power extracted. This behavior will be further investigated in Part 4.

```
d={0.078, 0.154, 0.102};
L={30.48, 15.24, 38.1};
e={0.000045, 0.000045, 0.000045}
Pa=0; Pb=0; Za=22.86; Zb=0;
k={0.5, 9, 1};
C={60, 0, 55};
ρ=998.2; μ=0.00102;
Qguess=0.00453;
Epsilon=10^-7;
gravity=9.81;
hm=0;
M= Length[d];
ft=Table[{(-1/(2 * Log[10, e[[i]]/(3.7 * d[[i]])]))^2 , {I,M}];
```

(a)

```
RegulaFalsi[a0_, b0_, m_]:=
Module[{},
a=N[a0];
b=N[b0];
cc = (ag[b] - b[ga])/(g[b] - g[a]) ;
Kk=0
Output={{kk, a, cc, b, g[cc]}};
While[kk<m,
If[Sign[g[b]]==Sign[g[cc]]'
b=cc, a=cc;];
cc = (ag[b] - b[ga])/(g[b] - g[a]);
Kk=kk+1;
Output=Append[output,
{kk, a, cc, b, g[cc]}];];]
```

(b)

```
Print["Guess Q1=", Qguess]; itt=0; conditionstop=1;
While[conditionstop>epsilon,{itt=itt+1; re=Table[(4 * ρ * Qguess)/(π * μ * d[[i]]) ,
{I,M}];
ff={};Do[If[re[[i]]>2300,{g[f_]=
1/√f + 2 * Log[10, 2.51/(re[[i]] * √f) + e[[i]]/(3.7 * d[[i]])];
RegulaFalsi[0.008, 0.2, 100];ff=Append[ff,cc]}, {cc=64/re[[i]] ;
ff=Append[ff,cc]}], {i,M}];
equationsolve=Solve[(Pb - Pa)/(ρ * gravity) + Zb - Za + Sum[(8 * Q^2)/(π^2 * gravity * d[[i]]^4)
((ff[[i]] * L[[i]])/d[[i]] + k[[i]] + c[[i]] * ft[[i]]), {i,1,M}] == hm, Q] ;
discharge=Q/.N[equationsolve];Qnew=Abs[discharge[[2]]];
Conditionstop=Abs[Qnew-Qguess];Qguess=Qnew;
Print["Q"itt+1,"z",Qguess, "{f1,f2,f3}=",ff,"   {Re1,Re2,Re3}=",
re]}];
```

(c)

Fig. 2. The Mathematica program for Part 1 of Example 1; (a) constants and physical properties; (b) the Regula Falsi iterative numerical scheme; (c) the main program.

```
Q=0.00453;
re=Table[(4 * ρ * Q)/(π * μ * d[[i]]) , {i,M}];ff={};
Do[If[re[[i]]>2300,{g[f_]=1/√f + 2 * Log[10, 2.51/(re[[i]] * √f) + e[[i]]/(3.7 * d[[i]])];
RegulaFalsi[0.008, 0.2, 100];ff=Append[ff,cc]}, {cc=64/re[[i]] ;
ff=Append[ff,cc]}], {i,M}]; Solve[
(Pb - Pa)/(ρ * gravity) + Zb - Za + Sum[(8 * Q^2)/(π^2 * gravity * d[[i]]^4)
((ff[[i]] * L[[i]])/d[[i]] + k[[i]] + c[[i]] * ft[[i]]), {i,1,M}] == ht, ht] ;
headturbine=ht/.N[%];
powerturbine=ρ*gravity*Q*headturbine
```

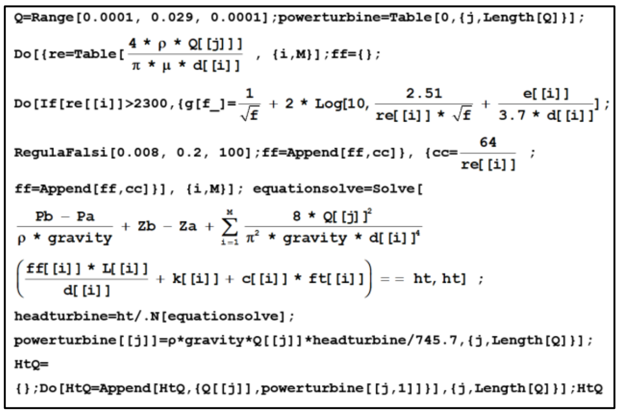
Fig. 3. A portion of the Mathematica program for Part 2 of Example 1.

```
Powerturbine=2;
Print["Guess Q1=", Qguess]; itt=0; conditionstop=1;
While[conditionstop>epsilon,{itt=itt+1; re=Table[(4 * ρ * Qguess)/(π * μ * d[[i]]) ,
{i,M}]; ff={};Do[If[re[[i]]>2300,{g[f_]=
1/√f + 2 * Log[10, 2.51/(re[[i]] * √f) + e[[i]]/(3.7 * d[[i]])];
RegulaFalsi[0.008, 0.2, 100];ff=Append[ff,cc]}, {cc=64/re[[i]] ;
ff=Append[ff,cc]}], {i,M}];
equationsolve=Solve[(Pb - Pa)/(ρ * gravity) + Zb - Za + Sum[(8 * Q^2)/(π^2 * gravity * d[[i]]^4)
((ff[[i]] * L[[i]])/d[[i]] + k[[i]] + c[[i]] * ft[[i]]), {i,1,M}] == (-powerturbine * 745.7)/(ρ * gravity * Q) , Q]
 ;
discharge=Q/.N[equationsolve];Qnew=Abs[discharge[[3]]];
Conditionstop=Abs[Qnew-Qguess];Qguess=Qnew;
Print["Q"itt+1,"=",Qguess,re]}]; ht=(-powerturbine * 745.7)/(ρ * gravity * Qguess)
```

Fig. 4. A section of the Mathematica program for Part 3 of Example 1.

Part (4): The relationship between Q and extracted power

Considering the system operation, Parts 1, 2, and 3 clearly show a clear relationship between Q and the power extracted. Without power extraction, the flow rate will be maximum, as presented in Part 1. Indeed, Part 1 is characterized by the maximum flow rate and a turbine head reduction equal to zero. If the turbine extracts all available flow head, both the Q and the extracted power would be zero. In Parts 2 and 3, different Qs and turbine head reduction lead to different power extractions. Indeed, in Part 3, two different Qs were found to generate 2 hp from the turbine. For the system, a more negative number of turbine head reduction leads to a smaller Q, but the power contains the product of Q and head reduction. Therefore, a relative maximum or minimum is demonstrated. The Mathematica computations for extracted power are presented in Fig. 5 as a function of flow rate. As shown in previous Figs., the omitted part of the program is as the part (a) and (b) of Fig. 2. The power is then computed for each Q, and the results are illustrated graphically in Fig. 6. The maximum power value (about 3.29 hp) which extracted from the system occurs at Q= 0.0166 m^3/sec. Fig. (6) demonstrates the double-valued, in-power extraction nature of the system operation. The series pipeline examples demonstrate Mathematica solutions for the prevalent types of series pipeline problems. Somewhat parallel piping systems are more complex than series pipeline systems, as are examined in the next section.



```
Q=Range[0.0001, 0.029, 0.0001];powerturbine=Table[0,{j,Length[Q]}];
Do[{re=Table[(4 * ρ * Q[[j]])/(π * μ * d[[i]]) , {i,M}];ff={};
Do[If[re[[i]]>2300,{g[f_]=1/√f + 2 * Log[10, 2.51/(re[[i]] * √f) + e[[i]]/(3.7 * d[[i]])];
RegulaFalsi[0.008, 0.2, 100];ff=Append[ff,cc]}, {cc=64/re[[i]] ;
ff=Append[ff,cc]}], {i,M}]; equationsolve=Solve[
(Pb − Pa)/(ρ * gravity) + Zb − Za + Sum[i=1..M] (8 * Q[[j]]^2)/(π^2 * gravity * d[[i]]^4)
((ff[[i]] * L[[i]])/d[[i]] + k[[i]] + c[[i]] * ft[[i]]) == ht, ht] ;
headturbine=ht/.N[equationsolve];
powerturbine[[j]]=ρ*gravity*Q[[j]]*headturbine/745.7,{j,Length[Q]}];
HtQ=
{};Do[HtQ=Append[HtQ,{Q[[j]],powerturbine[[j,1]]}],{j,Length[Q]}];HtQ
```

Fig. 5. A portion of the Mathematica program for Part 4 of Example 1.

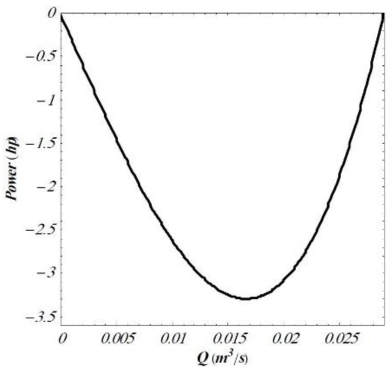
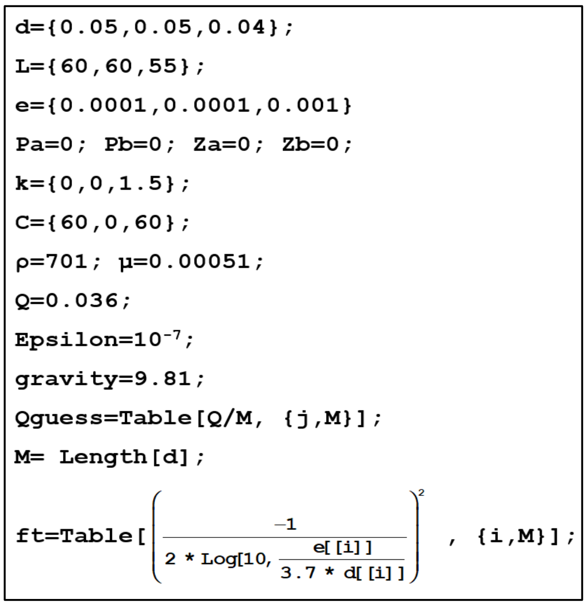


Fig. 6. The relationship between Q and power extracted for turbine

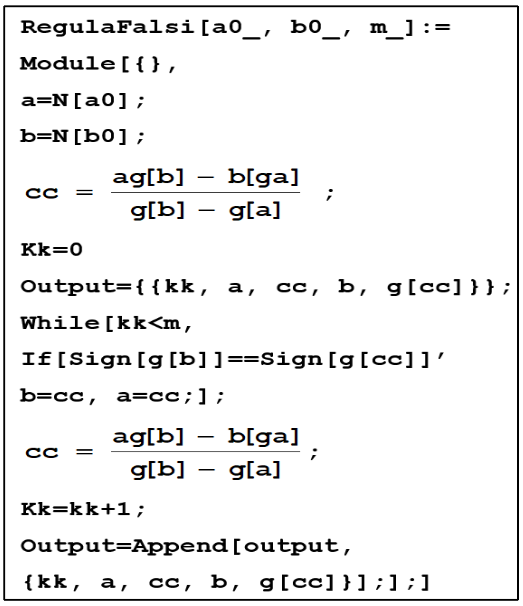
3.2. Example 2 solution

Part 1. The head increase and power imparted to the fluid for $Q_t = 0.036$ m^3/sec. Like example 1, the Similar Mathematica formulation will be used for all three parts of example 2; only the number of equations and variables will be changed. Fig. 7 contains the complete Mathematica program for the solution of Part 1.



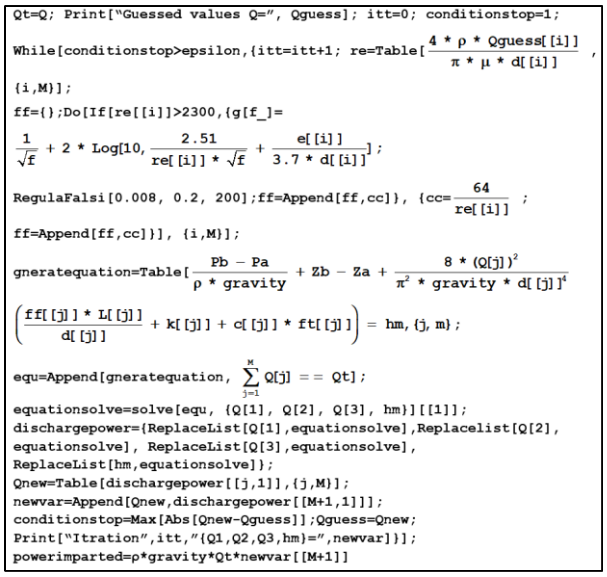
```
d={0.05,0.05,0.04};
L={60,60,55};
e={0.0001,0.0001,0.001}
Pa=0; Pb=0; Za=0; Zb=0;
k={0,0,1.5};
C={60,0,60};
ρ=701; μ=0.00051;
Q=0.036;
Epsilon=10^-7;
gravity=9.81;
Qguess=Table[Q/M, {j,M}];
M= Length[d];
ft=Table[(−1/(2 * Log[10, e[[i]]/(3.7 * d[[i]])]))^2 , {i,M}];
```

(a)



```
RegulaFalsi[a0_, b0_, m_]:=
Module[{},
a=N[a0];
b=N[b0];
cc = (ag[b] − b[ga])/(g[b] − g[a]) ;
Kk=0
Output={{kk, a, cc, b, g[cc]}};
While[kk<m,
If[Sign[g[b]]==Sign[g[cc]]’
b=cc, a=cc;];
cc = (ag[b] − b[ga])/(g[b] − g[a]);
Kk=kk+1;
Output=Append[output,
{kk, a, cc, b, g[cc]}];];]
```

(b)



```
Qt=Q; Print["Guessed values Q=”, Qguess]; itt=0; conditionstop=1;
While[conditionstop>epsilon,{itt=itt+1; re=Table[(4 * ρ * Qguess[[i]])/(π * μ * d[[i]]) ,
{i,M}];
ff={};Do[If[re[[i]]>2300,{g[f_]=
1/√f + 2 * Log[10, 2.51/(re[[i]] * √f) + e[[i]]/(3.7 * d[[i]])];
RegulaFalsi[0.008, 0.2, 200];ff=Append[ff,cc]}, {cc=64/re[[i]] ;
ff=Append[ff,cc]}], {i,M}];
gneratequation=Table[(Pb − Pa)/(ρ * gravity) + Zb − Za + (8 * (Q[j])^2)/(π^2 * gravity * d[[j]]^4)
((ff[[j]] * L[[j]])/d[[j]] + k[[j]] + c[[j]] * ft[[j]]) = hm, {j, m};
equ=Append[gneratequation, Sum[j=1..M] Q[j] == Qt];
equationsolve=solve[equ, {Q[1], Q[2], Q[3], hm}][[1]];
dischargepower={ReplaceList[Q[1],equationsolve],Replacelist[Q[2],
equationsolve], ReplaceList[Q[3],equationsolve],
ReplaceList[hm,equationsolve]};
Qnew=Table[dischargepower[[j,1]],{j,M}];
newvar=Append[Qnew,dischargepower[[M+1,1]]];
conditionstop=Max[Abs[Qnew-Qguess]];Qguess=Qnew;
Print["Itration”,itt,”{Q1,Q2,Q3,hm}=”,newvar]}];
powerimparted=ρ*gravity*Qt*newvar[[M+1]]
```

(c)

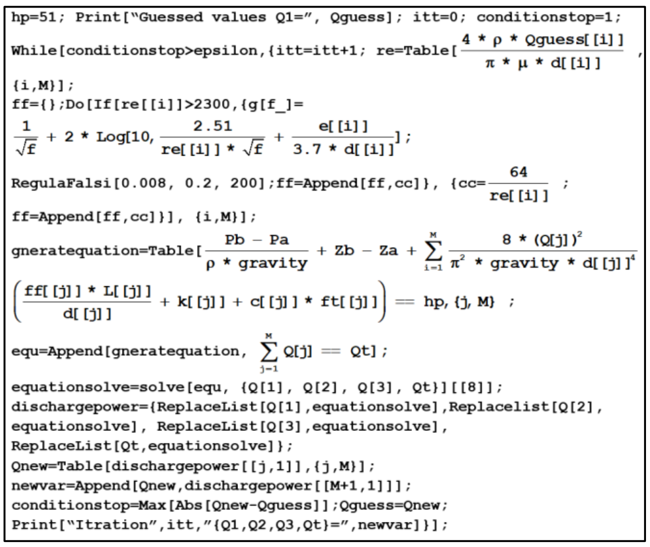
Fig. 7. The Mathematica program for Part 1 of Example 2. (a) Constants and physical properties, (b) The Regula Falsi iterative numerical scheme, (c) The main program.

This program is the kernel for the solutions to all parts of this problem. The first part of program (part a) defines the values of the variables, constants, physical properties, and functional definitions for the fully rough friction factor. The second part (b) is the Regula Falsi iterative numerical scheme for solve the implicit nonlinear equation. The third part (c) is the main program.

Each variable should be specified either with values or given estimations. The unknowns for this part of problem include Q in each legs and the required head increment. For other types of parallel pipeline systems, just the required solution variables (and the initial estimates) should be changed. Similar to series pipeline problems, the algorithm of solution is less important compared to the problem formulation and results interpretation. For Part 1 of Example 2, Q is 0.0149, 0.0152, and 0.0059 m3/sec, respectively, for pipes 1, 2 and 3. The head increment of pump is 87.5 m, and the power delivered to the fluid is 21.7 kW.

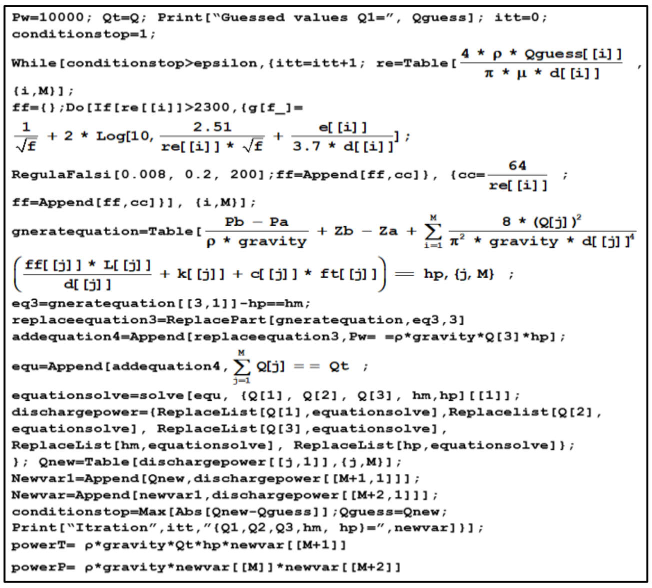
Part 2. The Q_t if the increase in head were 51 m.

Fig. 8 displays a section of the Mathematica program showing the main program for the solution. Only the main program is presented in the Figure as the first and second parts of the program (part a, b) for part 2 are identical to that of part 1. For Part 2, hp=51m is given and Q_t, Q_1, Q_2, Q_3 and power of pump are the unknowns. For this part of Example 2, Q_t is 0.0274 m3/sec, and the Q_1, Q_2 and Q_3 are 0.0113, 0.0116, and 0.0045 m3/sec, respectively. The delivered power to the fluid is 9.62 kW.



```
hp=51; Print["Guessed values Q1=", Qguess]; itt=0; conditionstop=1;
While[conditionstop>epsilon,{itt=itt+1; re=Table[(4 * ρ * Qguess[[i]])/(π * μ * d[[i]]) ,
{i,M}];
ff={};Do[If[re[[i]]>2300,{g[f_]=
1/√f + 2 * Log[10, 2.51/(re[[i]] * √f) + e[[i]]/(3.7 * d[[i]])];
RegulaFalsi[0.008, 0.2, 200];ff=Append[ff,cc]}, {cc=64/re[[i]] ;
ff=Append[ff,cc]}], {i,M}];
gneratequation=Table[(Pb − Pa)/(ρ * gravity) + Zb − Za + Σ_{i=1}^{M} (8 * (Q[j])^2)/(π^2 * gravity * d[[j]]^4)
((ff[[j]] * L[[j]])/d[[j]] + k[[j]] + c[[j]] * ft[[j]]) == hp, {j, M} ;
equ=Append[gneratequation, Σ_{j=1}^{M} Q[j] == Qt];
equationsolve=solve[equ, {Q[1], Q[2], Q[3], Qt}][[8]];
dischargepower={ReplaceList[Q[1],equationsolve],Replacelist[Q[2],
equationsolve], ReplaceList[Q[3],equationsolve],
ReplaceList[Qt,equationsolve]};
Qnew=Table[dischargepower[[j,1]],{j,M}];
newvar=Append[Qnew,dischargepower[[M+1,1]]];
conditionstop=Max[Abs[Qnew-Qguess]];Qguess=Qnew;
Print["Itration",itt,"{Q1,Q2,Q3,Qt}=",newvar]}];
```

Fig. 8. A portion of the Mathematica program for Part 2 of Example 2.



```
Pw=10000; Qt=Q; Print["Guessed values Q1=", Qguess]; itt=0;
conditionstop=1;
While[conditionstop>epsilon,{itt=itt+1; re=Table[(4 * ρ * Qguess[[i]])/(π * μ * d[[i]]) ,
{i,M}];
ff={};Do[If[re[[i]]>2300,{g[f_]=
1/√f + 2 * Log[10, 2.51/(re[[i]] * √f) + e[[i]]/(3.7 * d[[i]])];
RegulaFalsi[0.008, 0.2, 200];ff=Append[ff,cc]}, {cc=64/re[[i]] ;
ff=Append[ff,cc]}], {i,M}];
gneratequation=Table[(Pb − Pa)/(ρ * gravity) + Zb − Za + Σ_{i=1}^{M} (8 * (Q[j])^2)/(π^2 * gravity * d[[j]]^4)
((ff[[j]] * L[[j]])/d[[j]] + k[[j]] + c[[j]] * ft[[j]]) == hp, {j, M} ;
eq3=gneratequation[[3,1]]-hp==hm;
replaceequation3=ReplacePart[gneratequation,eq3,3]
addequation4=Append[replaceequation3,Pw= =ρ*gravity*Q[3]*hp];
equ=Append[addequation4, Σ_{j=1}^{M} Q[j] = = Qt ;
equationsolve=solve[equ, {Q[1], Q[2], Q[3], hm,hp][[1]];
dischargepower={ReplaceList[Q[1],equationsolve],Replacelist[Q[2],
equationsolve], ReplaceList[Q[3],equationsolve],
ReplaceList[hm,equationsolve], ReplaceList[hp,equationsolve]};
}; Qnew=Table[dischargepower[[j,1]],{j,M}];
Newvar1=Append[Qnew,dischargepower[[M+1,1]]];
Newvar=Append[newvar1,dischargepower[[M+2,1]]];
conditionstop=Max[Abs[Qnew-Qguess]];Qguess=Qnew;
Print["Itration",itt,"{Q1,Q2,Q3,hm, hp}=",newvar]}];
powerT= ρ*gravity*Qt*hp*newvar[[M+1]]
powerP= ρ*gravity*newvar[[M]]*newvar[[M+2]]
```

Fig. 9. A portion of the Mathematica program for Part 3 of Example 2.

Part 3. The delivered power to the fluid if a 10kW booster pump existed in line 3 and Qt was maintained at 0.036 m^3/sec. This is like Part 1, except that a 10kW booster pump exists in line 3. The Mathematica program for this part of the problem is demonstrated in Fig. 9. One equation should be added to solve block, $10kW = \gamma Q_3 h_p$, that represents the delivered power to the fluid in line 3 and by the addition to the leg 3 energy equation of -hp, the developed head by the pump located at leg 3. For Part 3 of Example 2, the Q_1, Q_2 and Q_3 are 0.0131, 0.0135, and 0.0094 m^3/sec, respectively. The head of main pump is 68.45 m, and the delivered power to the fluid is 16.9kW. The power delivered by booster pump to the fluid in leg 3 is confirmed to be 10 kW or 154.38 m in head. This is a relatively hard problem to solve ‘‘by hand’’ but the Mathematica solution is simple, straight forward, and congruent with the problem formulation.

4. Conclusions

Computation of friction factor in Darcy-Weisbach equation is one of the main steps in solving pipe flow problem. Since it depends on relative roughness and Reynolds number, an iteration process is needed for estimation of friction factor. In this study, a theoretical solution of implicit equation in pipeline systems was presented using Mathematica software. The results showed that using Mathematica, the problems can be more involved, open-ended and integrated. The Mathematica procedures discussed here indicate several features of Mathematica and illustrate how these features may be used to solve all different pipe flow problems. The Mathematica procedures are more compliant with the problem formulation compared to the traditional procedures. The concrete result is that the use of Mathematica permits researchers to focus on engineering rather than computational aspects of problem solutions. It was found that arithmetic systems such as Mathematica, present a new paradigm for engineering computations and education. Using this new paradigm, not only the existing techniques will not be replaced but also another option for computations is offered, that is, engineering tasks become the focus instead of programming tasks. The examples presented in this paper indicate the potency of Mathematica in a variety of pipeline systems calculations of pedagogical interest. One of the main limitation of this study was that the real data on series and parallel pipe line systems are scarce. That’s why the researchers only use benchmark problems for assessing their proposed models.

Nomenclature

A	Cross-sectional area of flow
C	Equivalent pipe lengths for minor energy losses
D	Diameter of pipe
E	Equivalent roughness
F	Darcy–weisbach friction factor
f_T	Fully rough friction factor
G	Gravity acceleration
h_f	Major head loss
hL	Total energy loss per unit weight
Hm	Mechanical energy per unit weight
Hp	Pump head
Ht	Turbine head l
K	The number indicating minor loss Coefficient
L	Length of pipe
M	Number of pipes
P	Pressure
Q	Discharge
Re	Reynolds number
V	Velocity distribution
V	Average velocity of flow
Z	Elevation head
γ	Specific weight
ρ	Density
μ	Dynamic viscosity

Acknowledgment

The authors would like to thank the editor(s) and anonymous reviewers for their careful reading of the manuscript and constructive comments.

References

Avci A., and Karagoz I., A novel explicit equation for friction factor in smooth and rough pipes, Journal of Fluids Engineering 131 (2009) 1-4.

Bahder T.B., Mathematica for scientists and engineers. Addison-Wesley Longman Publishing Co., Inc. (1994)

Churchill S.W., Friction-factor equation spans all fluid-flow regimes, Chemical Engineering 84 (1977) 91-92.

Çoban M.T., Error analysis of non-iterative friction factor formulas relative to Colebrook-White equation for the calculation of pressure drop in pipes, Journal of Naval Sciences and Engineering 8 (2012) 1-13.

Cojbasic Z., and Brkic D., Very accurate explicit approximations for calculation of the Colebrook friction factor, International Journal of Mechanical Sciences 67 (2013) 10-13.

Danish M., Kumar S., Kumar S., Approximate explicit analytical expressions of friction factor for flow of Bingham fluids in smooth pipes using Adomian decomposition method, Communications in Nonlinear Science and Numerical Simulation 16 (2011) 239-251.

Diniz V. E.M.G., Souza P.A., Four explicit formulae for friction factor calculation in pipe flow, WIT Transactions on Ecology and the Environment 125 (2009) 369-380.

Ford J.A., Improved algorithms of illinois-type for the numerical solution of nonlinear equations, University of Essex, Department of Computer Science (1995).

Galdino S., A family of regula falsi root-finding methods, In Proceedings of the 2011 World Congress on Engineering and Technology (2011).

Haaland S.E., Simple and explicit formulas for the friction factor in turbulent pipe flow, Journal of Fluids Engineering 105 (1983) 89-90.

Hodge B.K. and Taylor R.P. Piping-system solutions using Mathcad, Computer Applications in Engineering Education 10 (2002) 59-78.

Hodge B.K., and Taylor R.P., Analysis and design of energy systems, 3rd ed., Prentice Hall, Upper Saddle River, NJ (1999).

Jagadeesh R., Sonnad J.R., Goudar C.T., Closure to turbulent flow friction factor calculation using a mathematically exact alternative to the colebrook–white equation, Journal of Hydraulic Engineering 134 (2008) 1188-1188.

Larock B.E., Jeppson R.W., Watters G.Z., Hydraulics of pipeline systems, CRC Press (2000).

Li P., Seem J.E., Li Y., A new explicit equation for accurate friction factor calculation of smooth pipes, International Journal of Refrigeration 34 (2011) 1535-1541.

Li S., and Huai W., United formula for the friction factor in the turbulent region of pipe flow, PloS One 11 (2016) 1-10.

Morrison F.A., Data correlation for friction factor in smooth pipes, department of Chemical Engineering, Michigan Technological University, Houghton, MI, (2013) 1-2.

Offor U.H., and Alabi S.B., An accurate and computationally efficient explicit friction factor model, Advances in Chemical Engineering and Science 6 (2016) 237.

Papaevangelou G., Evangelides C., Tzimopoulos C., A new explicit relation for friction coefficient f in the Darcy-Weisbach equation, In Proceedings of the Tenth Conference on Protection and Restoration of the Environment 166 (2010) 6-09.

Rennels D.C., Hudson H.M., Pipe flow: A practical and comprehensive guide, John Wiley and Sons (2012).

Shames I. H., Mechanics of fluids, McGraw Hill, Inc. (1982).

Slatter P.T., The turbulent flow of non-Newtonian slurries in pipes. In Proceedings of 8th International Conf. on Transport and Sedimentation of Solid Particles, Prague, Paper A (1995).

Streeter V.L., Wylie E.B., Bedford K.W., Fluid mechanics, McGraw-Hill. New York (1998).

Swamee P.K. and Aggarwal N. Explicit equations for laminar flow of Bingham plastic fluids, Journal of Petroleum Science and Engineering 76 (2011) 178-184.

Taler D., Determining velocity and friction factor for turbulent flow in smooth tubes, International Journal of Thermal Sciences 105 (2016) 109-122.

Turgut O.E., Asker M. and Coban M.T., A review of non-iterative friction factor correlations for the calculation of pressure drop in pipes, Bitlis Eren University Journal of Science and Technology 4 (2014) 1-8.