FACTA UNIVERSITATIS
Series: **Mechanical Engineering** Vol. 19, Nº 2, 2021, pp. 253 - 269
https://doi.org/10.22190/FUME210111044B

Review paper

EXCEL VBA-BASED USER DEFINED FUNCTIONS FOR HIGHLY PRECISE COLEBROOK'S PIPE FLOW FRICTION APPROXIMATIONS: A COMPARATIVE OVERVIEW

Dejan Brkić[1], Zoran Stajić[2]

[1]IT4Innovations, VSB – Technical University of Ostrava, Ostrava, Czech Republic
[2]Research and Development Centre "IRC Alfatec", Niš, Serbia

Abstract. *This review paper gives Excel functions for highly precise Colebrook's pipe flow friction approximations developed by users. All shown codes are implemented as User Defined Functions – UDFs written in Visual Basic for Applications – VBA, a common programming language for MS Excel spreadsheet solver. Accuracy of the friction factor computed using nine to date the most accurate explicit approximations is compared with the sufficiently accurate solution obtained through an iterative scheme which gives satisfying results after sufficient number of iterations. The codes are given for the presented approximations, for the used iterative scheme and for the Colebrook equation expressed through the Lambert W-function (including its cognate Wright ω-function). The developed code for the principal branch of the Lambert W-function has additional and more general application for solving different problems from variety branches of engineering and physics. The approach from this review paper automates computational processes and speeds up manual tasks.*

Key words: *Hydraulic resistance, Colebrook flow friction, Lambert W-function, Excel Macro Programming, Visual Basic for Applications (VBA), User Defined Functions (UDFs)*

1. INTRODUCTION

The Colebrook equation from 1939 [1], Eq. (1), is an informal standard widely accepted in engineering practice for calculation of turbulent Darcy's fluid flow friction factor. It is an empirical equation based on an experiment with air flow through a set of smooth to fully rough pipes performed by Colebrook and White in 1937 [2]. The Moody diagram [3] in its turbulent part represents a graphical interpretation of the Colebrook equation.

Received January 11, 2021 / Accepted April 08, 2021
Corresponding author: Dejan Brkić
IT4Innovations, VSB – Technical University of Ostrava, 17. listopadu 2172/15, Ostrava, Czechia
E-mails: dejan.brkic@vsb.cz, dejanbrkic0611@gmail.com

$$\frac{1}{\sqrt{f}} = -2 \cdot \log_{10}\left(\frac{2.51}{Re} \cdot \frac{1}{\sqrt{f}} + \frac{\varepsilon}{3.71}\right) \approx -0.8686 \cdot \ln\left(\frac{2.51}{Re} \cdot \frac{1}{\sqrt{f}} + \frac{\varepsilon}{3.71}\right) \quad (1)$$

In Eq. (1), dimensionless turbulent Darcy flow friction factor is given as f, the dimensionless Reynolds number as Re, the dimensionless relative roughness of inner pipe surface as ε, while Briggsian decimal logarithm (to base 10) is given as $\log_{10}$ and Napierian natural logarithm (to base e, where e≈2.718) as ln.

As shown in Eq. (1), the Colebrook equation is given in an implicitly entangled logarithmic form which cannot be solved in terms of elementary functions. It can be solved; 1) iteratively (such solution can be treated as accurate after sufficient number of iterations) [4,5], or 2) using one-step approximate formulas specially developed for such purpose (maximal error of such formulas can be estimated in advance). Therefore, for solving the Colebrook equation, various explicit approximations [6-11] can be used to avoid long computing times caused by iterative schemes during the numerical simulations of pipelines for transport of various fluids [12,13]. Also, the Colebrook equation can be analytically expressed through the Lambert W-function [14-18], but anyway the Lambert W-function itself is an implicit function which can only be solved either iteratively or approximately using specially developed one-step formulas [19,20].

Fast and accurate execution of codes during calculation of pipe flow friction is essential for calculation of pressure drop and flow rate in oil and gas industry, water distribution, in chemical engineering, etc. To facilitate use of the Colebrook equation in spreadsheet solver MS Excel [21,22], codes written in Visual Basic for Applications (VBA) based on the available highly precise explicit approximations [23-30] are given as User Defined Functions (UDFs) and compared in this review paper. Such approach automates computational processes and speeds up manual tasks.

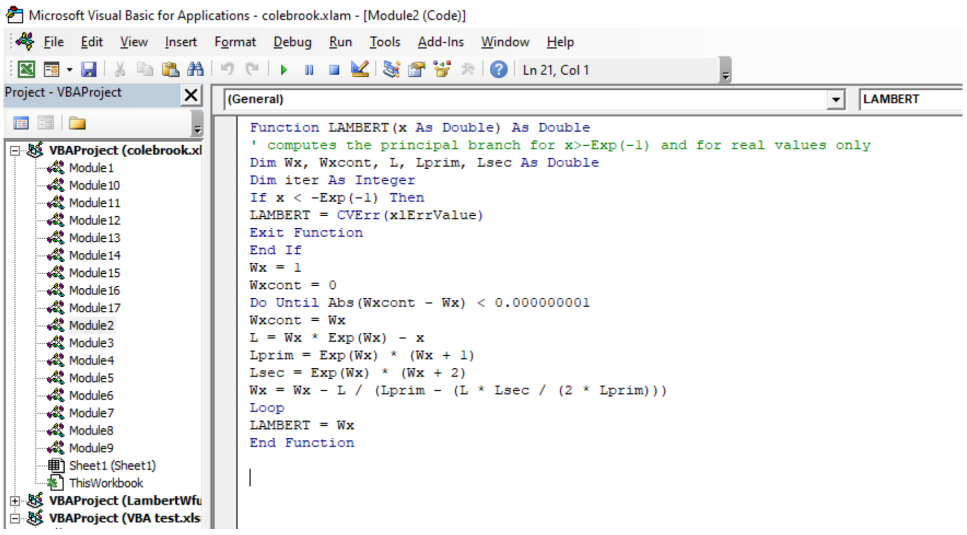
2. Visual Basic for Applications Excel User Defined Functions

The codes for solving the Colebrook equation used in this review paper are shown through macros for MS Excel written in Visual Basic for Applications (VBA). In essence, a macro is a term that refers to a set of programming instructions that automates tasks by creating custom calculations that can be used repeatedly throughout workbooks and which can be called by the host application as User Defined Function (UDF). The VBA is closely related to Visual Basic programming language, but on the contrary, VBA codes can only run within a host application, and not as a standalone program. The here presented codes are compiled to a proprietary intermediate language that can be executed by MS Excel, which is the host application in this case. An UDF should be placed in module following the appropriate syntax of the VBA programming language as shown in Fig. 1.

To prepare the UDFs for the explicit approximations of the Colebrook equation, the following steps in MS Excel need to be followed:

1) Keyboard shortcut "ALT + F11" should be pressed to open the Visual Basic Editor (a screen similar as in Fig. 1 should appear),
2) In the Visual Basic Editor, a Module for UDFs, should be opened using "Insert" button from the ribbon, and by choosing "Module" from the drop menu,
3) In the opened module, the UDF should be written using appropriate syntax of the VBA programming language,

4) Using "Debug" button from the ribbon, the current project should be compiled by choosing "Compile VBAProject" from the drop menu, and
5) Finally, the current UDF should be saved with extension "xlam", using "File" button and then "Save as" from drop menu (it will be saved by default in: 'C:\Users\[user name]\AppData\Roaming\Microsoft\AddIns\[name of the document].xlam').



```
Function LAMBERT(x As Double) As Double
' computes the principal branch for x>-Exp(-1) and for real values only
Dim Wx, Wxcont, L, Lprim, Lsec As Double
Dim iter As Integer
If x < -Exp(-1) Then
LAMBERT = CVErr(xlErrValue)
Exit Function
End If
Wx = 1
Wxcont = 0
Do Until Abs(Wxcont - Wx) < 0.000000001
Wxcont = Wx
L = Wx * Exp(Wx) - x
Lprim = Exp(Wx) * (Wx + 1)
Lsec = Exp(Wx) * (Wx + 2)
Wx = Wx - L / (Lprim - (L * Lsec / (2 * Lprim)))
Loop
LAMBERT = Wx
End Function
```

Fig. 1 Visual Basic Editor

The syntax of any UDF for MS Excel written in VBA programming language has few main parts, such as:

- Every function starts with "Function" and finishes with "End Function",
- Specific name of the function should be defined (designated by user and avoiding reserved names),
- After the designated name of the function, inputs should be specified in parentheses,
- Data type of inputs and output should be defined using "As" (the data type of other used parameters with "Dim" and "As"),
- In the body of the function, after the part with calculation but before "End Function", a return value should be assigned to the name of the function.
- Like any other Excel function, an UDF can be called from any Excel cell (if it is properly loaded).

The syntax of the MS Excel and of the VBA programming language are different. For example, in-built function which returns value for the Napierian natural logarithm, in the VBA programming language is "log" while in MS Excel is "ln" ("ln" is not reserved name in the VBA). However, until recently, reusable UDFs could be implemented only through scripts written using different syntax than for Excel formulas, using VBA or using JavaScript. Now, Excel users can use a new feature called "Lambda" which introduce the ability to create custom functions using Excel's formula language [31].

3. Solutions to the Colebrook Equation with Their Software Codes

Using iterative schemes [4,5,22], the Colebrook equation in its native implicit form can be solved with high accuracy after sufficient number of iterations. On the other hand, very accurate explicit approximations of the Colebrook equation introduce certain small error which can be predicted in advance [6-11] and which is analyzed in further text.

Alternatively, the Colebrook equation can be transformed analytically in an explicit form through the Lambert W-function [14-18]. This approach provides the same accuracy as obtained through an iterative solution, but with a constraint that an overflow error can occur in certain computational approaches for the high values of the argument of the Lamber W-function if the calculation is performed as usually in a computer with standard registers [32,33].

The Lambert W-function is itself an implicitly given function that needs to be further evaluated iteratively or using specially developed approximate formulas [34] (such solutions of the Lambert W-function have wide application in engineering and physics [19]).

After thorough examination of the approximations of the Colebrook equation from available literature [6-11], nine most accurate explicit approximations [23-30] were selected for analysis and for comparisons performed in this review paper. The examined approximations are ranked in Table 1 in terms of 1) accuracy, and 2) time taken for execution:

1) The relative error is calculated as $\left|(f-f_i)/f_i\right|\cdot 100\%$, where fi is the friction factor from an iterative scheme, while f is calculated using the observed approximation.
2) Approximations require computational resources in terms of the number of floating points for execution and therefore simpler approximations are faster in computer simulations [37-41] (speed of nine selected approximations are evaluated using methodology from [42,43]). Computational effort for the mathematical operations was determined by performing 100 million calculations for each mathematical operation using random input each repeated five times, with the average computational time recorded. The results are [44]: Addition-23.40sec, Subtraction-27.50sec, Multiplication-36.20sec, Division-31.70sec, Squared-51.10sec, Square root-53.70sec, Fractional exponential-77.60sec, Napierian natural logarithm-63.00sec, and Briggsian decimal logarithm to base 10-78.80sec.

Accuracy is checked using 2 Million quasi-random and as well 90 thousand and 740 uniformly distributed samples, as in [9,23,35,36], which covers the whole domain of the Reynolds number, Re and of the relative roughness of inner pipe surface, ε, which are commonly used in engineering practice; $2320<Re<10^8$ and $0<\varepsilon<0.05$.

Explicit approximations of the Colebrook equation should be not only accurate but also simple for computation (Fig. 2 shows error distribution for the four most accurate approximations).

The approximations of the Colebrook equation from Table 1 are shown in further text, while the related algorithms and codes are given for the four most accurate approximations.

The distribution of the maximal relative error is irregular for all available approximations of the Colebrook equation.

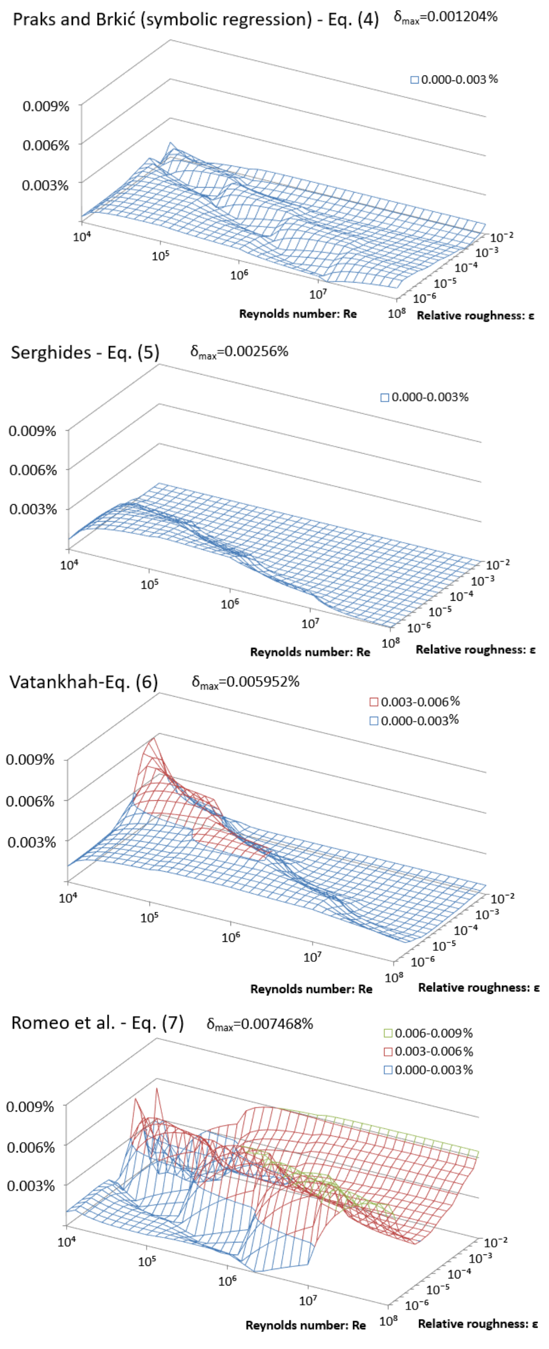


Fig. 2 Distribution of the maximal relative error of the extremely accurate explicit approximations (the error less than 0.01%)

Table 1 Results of accuracy and efficiency of the examined approximations with the relative error lower than 0.1%.

Approximation	Maximum relative error (%)	Execution time (sec)	Ratio of maximum relative error	Ratio of execution time
Praks and Brkić -sr[1] [23] – Eq. (4)	0.001204	450.7	1	1.02
Serghides [29] – Eq. (5)	0.002560	906.4	2.13	2.06
Vatankhah [25] – Eq. (6)	0.005952	760	4.94	1.73
Romeo et al. [28] – Eq. (7)	0.007468	817.2	6.20	1.86
Buzzelli [27] – Eq. (8)	0.019944	667.8	16.56	1.52
Praks and Brkić -se[2] [23] – Eq. (9)	0.058517	573.9	48.60	1.30
Offor and Alabi [26] – Eq. (10)	0.062704	477.1	52.08	1.08
Shacham [30] – Eq. (11)	0.083068	567	68.99	1.29
Lamri [24] – Eq. (12)	0.097438	440.2	80.93	1

[1]sr - symbolic regression of the Wright ω-function, [2]se - series expansion of the Wright ω-function; Ratios are given in respect to the most accurate and fastest approximation.

3.1. Iterative Solutions

Iterative schemes are most suitable for the implicitly given types of equations, such as for the Colebrook equation [4,5]. The Lambert W-function which is used to express the Colebrook equation in explicit form, is also implicitly given and therefore is also suitable for evaluation using iterative methods [34].

3.1.1. Simple fixed-point iterative scheme

The Colebrook equation for flow friction is suitable for calculation in its native form through an iterative method. A chosen simple starting point $x_0=5$ for the fixed-point iterative scheme as from the algorithm in Fig. 3 is in the range of applicability of the Colebrook equation and can assure fast convergence, while the final solution can be reached after 2 to 11 iterations using $x_i = -0.8686 \cdot \ln\left(\frac{2.51 \cdot x_{i-1}}{Re} + \frac{\varepsilon}{3.71}\right)$, where $x = \frac{1}{\sqrt{f}}$.

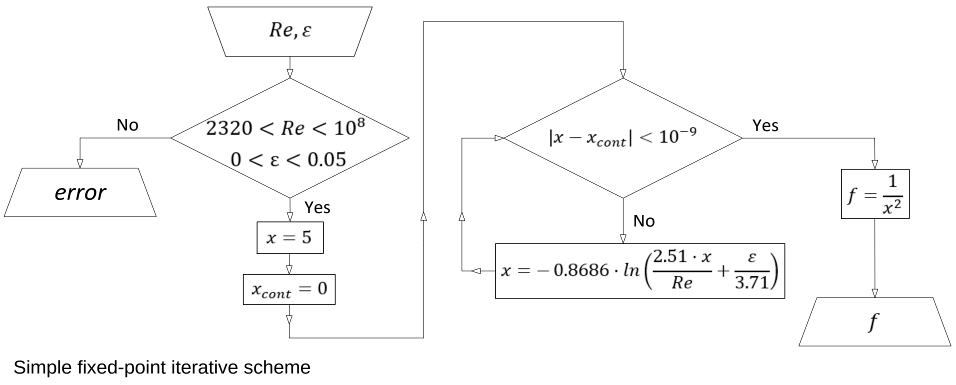

Re, ε
No
$2320 < Re < 10^8$
$0 < \varepsilon < 0.05$
error
Yes
$x = 5$
$x_{cont} = 0$
$|x - x_{cont}| < 10^{-9}$
Yes
No
$x = -0.8686 \cdot ln\left(\frac{2.51 \cdot x}{Re} + \frac{\varepsilon}{3.71}\right)$
$f = \frac{1}{x^2}$
f
Simple fixed-point iterative scheme

Fig. 3 Algorithm for simple fixed-point iterative scheme for solving the Colebrook equation

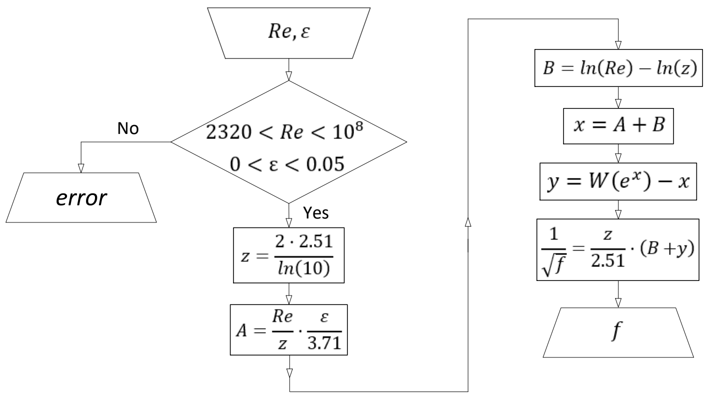
The VBA code for solving the Colebrook equation through simple fixed-point iterative scheme is given as follows:

```
Function COLEBROOKITE(REYNOLDS As Double, EPSILON As Double) As Double
Dim x, xcont As Double
If REYNOLDS < 2320 Or REYNOLDS > 100000000# Or EPSILON < 0 Or EPSILON > 0.05 Then
COLEBROOKITE = CVErr(xlErrValue)
Exit Function
End If
xcont = 0
x = 5
Do Until Abs(x − xcont) < 0.000000001
xcont = x
x = −2 / Log(10) ∗ Log(2.51 ∗ x / REYNOLDS + EPSILON / 3.71)
Loop
x = x ^ −2
COLEBROOKITE = x
End Function
```

3.1.2. Lambert W-function

To date, the only way to transform analytically the Colebrook equation from its native implicit form into an explicit form is through the Lambert W-function [18]. A version from [23,36] is given in Eq. (2), with the related algorithm in Fig. 4 and the code as follows (to use this code, additional UDF "LAMBERT" should be defined as explained in further text).

$$\left.\begin{aligned} \frac{1}{\sqrt{f}} &= \frac{z}{2.51}\cdot(B+y) \\ z &= \frac{2\cdot 2.51}{\ln(10)} \\ A &= \frac{Re}{z}\cdot\frac{\varepsilon}{3.71} \\ B &= \ln(Re)-\ln(z) \\ x &= A+B \\ y &= W(e^{x})-x \end{aligned}\right\} \quad (2)$$



Re, ε
$2320 < Re < 10^8$
$0 < \varepsilon < 0.05$
No
Yes
error
$z = \frac{2\cdot 2.51}{ln(10)}$
$A = \frac{Re}{z}\cdot\frac{\varepsilon}{3.71}$
$B = ln(Re) - ln(z)$
$x = A + B$
$y = W(e^x) - x$
$\frac{1}{\sqrt{f}} = \frac{z}{2.51}\cdot(B+y)$
f

Lambert W-based solution of the Colebrook equation

Fig. 4 Algorithm for solving the Colebrook equation expressed through the Lambert W-function

```
Function COLEBROOKLAMBERT(REYNOLDS As Double, EPSILON As Double) As Double
Dim z, A, B, x, y, f As Double
If REYNOLDS < 2320 Or REYNOLDS > 100000000# Or EPSILON < 0 Or EPSILON > 0.05 Then
COLEBROOKLAMBERT = CVErr(xlErrValue)
Exit Function
End If
z = 2 * 2.51 / Log(10)
A = REYNOLDS * EPSILON / (3.71 * z)
B = Log(REYNOLDS) − Log(z)
x = A + B
y = LAMBERT(Exp(x)) − x
f = ((z / 2.51) * (B + y)) ^ − 2
COLEBROOKLAMBERT = f
End Function
```

In some of the cases such as in [18,32], the argument of the Lambert W-function is fast-growing and for the values of $x>e^{709.7827}$, e≈2.718, an overflow error can occur [33] while the Colebrook equation expressed in that way cannot be solved always in a computer due to its limited capacity of registers (see Fig. 5). However, a version from [23,36] as given in Eq. (2) uses the Lambert W-function with a shifted argument which allows computation avoiding the explained overflow error.

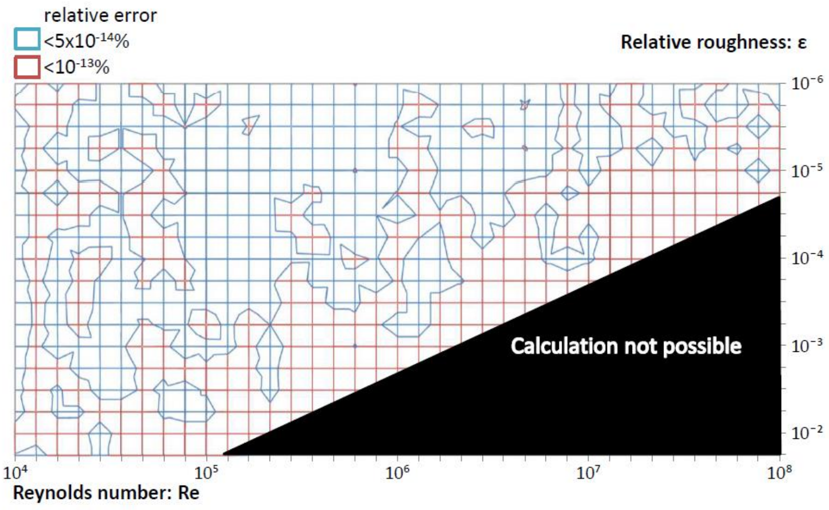


Fig. 5 Constraints for using the Lambert W-function for solving the Colebrook equation (based on [18,32])

The Halley iterative scheme, Eq. (3), is used here for evaluation of the principal branch of the Lambert W-function. The Lambert W-function function in suitable form is given as Li-1, with its first and second derivative given as L'_{i-1} and L''_{i-1}. This solution is valid for real values for x>-1/e, where e≈2.718.

To start the Halley iterative scheme for solving the principal branch of the Lambert W-function, a simple and sufficient starting point x0=1 can be chosen, which makes the algorithm from Fig. 6 fast for execution. The principal branch of the Lambert W-function is used often in engineering and physics [19] meaning that the algorithm from Fig. 6 can have much wider application aside for the Colebrook equation.

$$\left.\begin{aligned} W_i(x) &= W_{i-1}(x) - \frac{L_{i-1}}{L'_{i-1} - \frac{L_{i-1} \cdot L"_{i-1}}{2 \cdot L'_{i-1}}} \\ L_{i-1} &= W_{i-1}(x) \cdot e^{W_{i-1}(x)} - x = 0 \\ L'_{i-1} &= e^{W_{i-1}(x)}(W_{i-1}(x) + 1) \\ L"_{i-1} &= e^{W_{i-1}(x)}(W_{i-1}(x) + 2) \end{aligned}\right\} \quad (3)$$

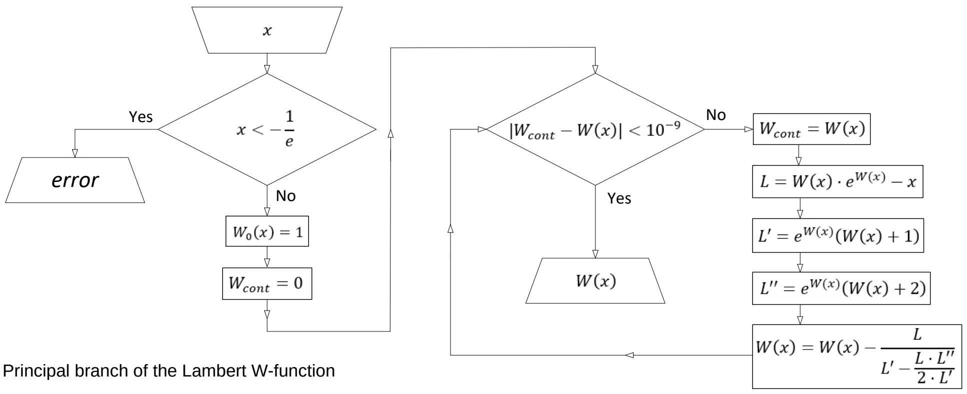

x
Yes
$x < -\frac{1}{e}$
error
No
$W_0(x) = 1$
$W_{cont} = 0$
$|W_{cont} - W(x)| < 10^{-9}$
No
Yes
$W(x)$
$W_{cont} = W(x)$
$L = W(x) \cdot e^{W(x)} - x$
$L' = e^{W(x)}(W(x) + 1)$
$L'' = e^{W(x)}(W(x) + 2)$
$W(x) = W(x) - \frac{L}{L' - \frac{L \cdot L''}{2 \cdot L'}}$
Principal branch of the Lambert W-function

Fig. 6 Algorithm for the principal branch of the Lambert W-function

The code for solving the principal branch of the Lambert W-function is given as follows:

```
Function LAMBERT(x As Double) As Double
' computes the principal branch for x > −Exp(−1) and for real values only
Dim Wx, Wxcont, L, Lprim, Lsec As Double
Dim iter As Integer
If x < −Exp(−1) Then
LAMBERT = CVErr(xlErrValue)
Exit Function
End If
Wx = 1
Do Until Abs(Wxcont − Wx) < 0.000000001
Wxcont = Wx
L = Wx * Exp(Wx) – x
Lprim = Exp(Wx) * (Wx + 1)
Lsec = Exp(Wx) * (Wx + 2)
Wx = Wx − L / (Lprim − (L * Lsec / (2 * Lprim)))
Loop
LAMBERT = Wx
End Function
```

3.2. Explicit Approximations of the Colebrook Equation

Explicit approximations of the Colebrook equation which introduce a maximal relative error less than 0.1% are given in Table 1. Four of them, Praks and Brkić based on symbolic regression and on the Wright ω-function [23], Serghides [29], Vatankhah [25] and Romeo et al. [28], introduce a relative error of less than 0.01% and can be classified as extremely accurate, while those five, Buzzelli [27], Praks and Brkić based on series expansion of the

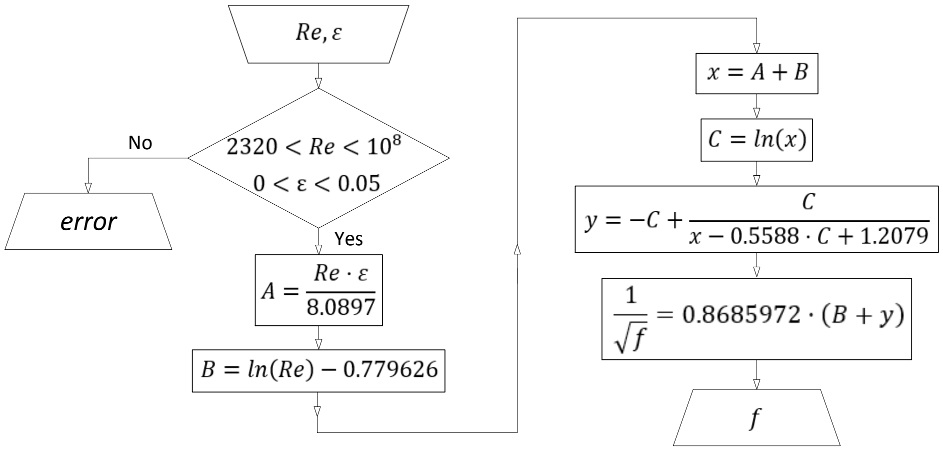
Wright ω-function [23], Offor and Alabi [26], Shacham [30] and Lamri [24], with the maximal relative error between 0.01% and 0.1% can be classified as very accurate approximations.

Algorithms and VBA codes are given here only for extremely accurate approximations while coding of the further approximations is not shown [44].

3.2.1.Praks and Brkić approximation based on the Wright ω-function and symbolic regression

Praks and Brkić approximation [23], given in Eq. (4) with the algorithm in Fig. 7, is based on the Wright ω-function and on symbolic regression. The Wright ω-function is a cognate of the Lambert W-function where W(ex)-x=ω(x)-x, which is used to eliminate fast-growing term ex from calculation. The shown approximation y of ω(x)-x is very accurate within the domain valid for the Colebrook equation, i.e., between 7.51<x<619 (symbolic regression is used to approximate ω(x)-x in this domain).

$$\left.\begin{array}{c} \frac{1}{\sqrt{f}} \approx 0.8685972 \cdot (B+y) \\ A \approx \frac{Re \cdot \varepsilon}{8.0897} \\ B \approx \ln(Re) - 0.779626 \\ x \approx A + B \\ C \approx \ln(x) \\ y \approx \frac{C}{x - 0.5588 \cdot C + 1.2079} - C \end{array}\right\} \tag{4}$$



Re, ε
$2320 < Re < 10^8$
$0 < \varepsilon < 0.05$
No
Yes
error
$A = \frac{Re \cdot \varepsilon}{8.0897}$
$B = ln(Re) - 0.779626$
$x = A + B$
$C = ln(x)$
$y = -C + \frac{C}{x - 0.5588 \cdot C + 1.2079}$
$\frac{1}{\sqrt{f}} = 0.8685972 \cdot (B + y)$
f

Praks and Brkić (symbolic regression) - Eq. (4)

Fig. 7 Algorithm for the Praks-Brkić (symbolic regression) approximation

The VBA code based on algorithm from Fig. 7 is given as:

```
Function PRAKSBRKIC(REYNOLDS As Double, EPSILON As Double) As Double
Dim A, B, x, c, y, f As Double
If REYNOLDS < 2320 Or REYNOLDS > 100000000# Or EPSILON < 0 Or EPSILON > 0.05 Then
PRAKSBRKIC = CVErr(xlErrValue)
Exit Function
End If
A = REYNOLDS * EPSILON / 8.0897
B = Log(REYNOLDS) − 0.779626
x = A + B
C = Log(x)
y = C / (x − 0.5588 * C + 1.2079) − C
f = (0.8685972 * (B + y)) ^ − 2
PRAKSBRKIC = f
End Function
```

3.2.2. Serghides approximation

The Serghides approximation [29] is based on Steffensen iterative scheme [4], and the shown version, Eq. (5) is improved by genetic algorithms [35,45,46]. It is given in Eq. (5), with related algorithm in Fig. 8.

$$\left.\begin{aligned}\frac{1}{\sqrt{f}} &\approx A-\frac{(B-A)^2}{C-2\cdot B+A}\\ A &\approx -0.8686\cdot\ln\left(\frac{\varepsilon}{3.71}+\frac{12.585}{Re}\right)\\ B &\approx -0.8686\cdot\ln\left(\frac{\varepsilon}{3.71}+\frac{2.51\cdot A}{Re}\right)\\ C &\approx -0.8686\cdot\ln\left(\frac{\varepsilon}{3.71}+\frac{2.51\cdot B}{Re}\right)\end{aligned}\right\} \tag{5}$$

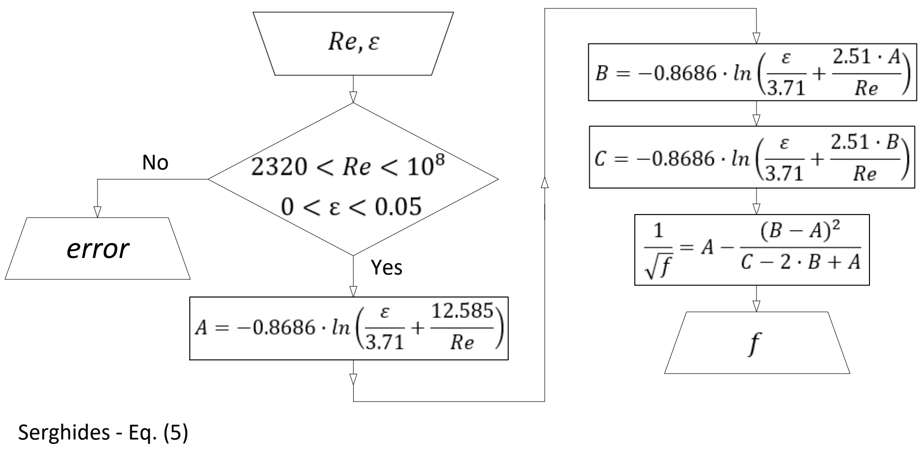

Re, ε
2320 < Re < 10^8
0 < ε < 0.05
No
error
Yes
A = −0.8686 · ln(ε/3.71 + 12.585/Re)
B = −0.8686 · ln(ε/3.71 + 2.51 · A/Re)
C = −0.8686 · ln(ε/3.71 + 2.51 · B/Re)
1/√f = A − (B − A)²/(C − 2 · B + A)
f
Serghides - Eq. (5)

Fig. 8 Algorithm for the Serghides approximation

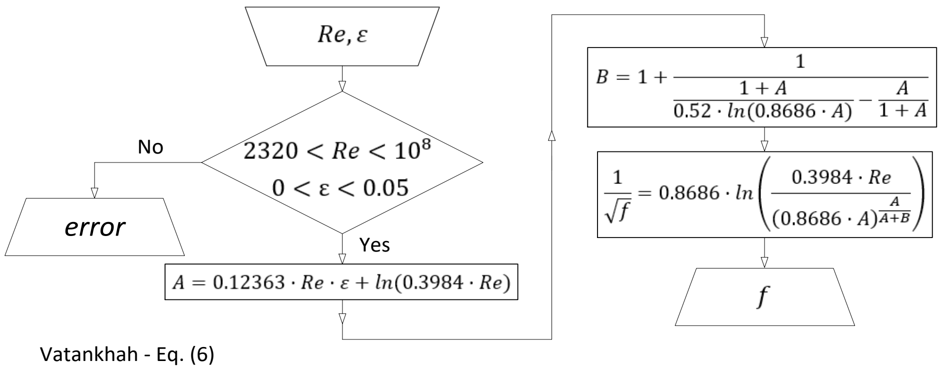
The VBA code based on algorithm from Fig. 8 is given as:

```
Function SERGHIDES(REYNOLDS As Double, EPSILON As Double) As Double
Dim A, B, C, f, ln As Double
If REYNOLDS < 2320 Or REYNOLDS > 100000000# Or EPSILON < 0 Or EPSILON > 0.05 Then
SERGHIDES = CVErr(xlErrValue)
Exit Function
End If
ln = 2 / Log(10)
A = −ln * Log(EPSILON / 3.71 + 12.585 / REYNOLDS)
B = −ln * Log(EPSILON / 3.71 + 2.51 * A / REYNOLDS)
C = −ln * Log(EPSILON / 3.71 + 2.51 * B / REYNOLDS)
f = A − (B − A) ^ 2 / (C − 2 * B + A)
f = f ^ −2
SERGHIDES = f
End Function
```

3.2.3. Vatankhah approximation

The Vatankhah approximation [25] is given in Eq. (6) with the related algorithm in Fig. 9 (few versions of this approximation are available in [25], where for one of those approximations, Brkić and Praks [36] estimate its maximal relative error of no more than 0.0028%). This approximation is related to [47,48].

$$\left.\begin{array}{c} \frac{1}{\sqrt{f}} \approx 0.8686 \cdot \ln\left(\frac{0.3984 \cdot Re}{(0.8686 \cdot A)^{\frac{A}{A+B}}}\right) \\ A \approx 0.12363 \cdot Re \cdot \varepsilon + \ln(0.3984 \cdot Re) \\ B \approx 1 + \frac{1}{\frac{1+A}{0.52 \cdot \ln(0.8686 \cdot A)} - \frac{A}{1+A}} \end{array}\right\} \quad (6)$$



Re, ε
$2320 < Re < 10^8$
$0 < \varepsilon < 0.05$
No
error
Yes
$A = 0.12363 \cdot Re \cdot \varepsilon + ln(0.3984 \cdot Re)$
$B = 1 + \frac{1}{\frac{1+A}{0.52 \cdot ln(0.8686 \cdot A)} - \frac{A}{1+A}}$
$\frac{1}{\sqrt{f}} = 0.8686 \cdot ln\left(\frac{0.3984 \cdot Re}{(0.8686 \cdot A)^{\frac{A}{A+B}}}\right)$
f
Vatankhah - Eq. (6)

Fig. 9 Algorithm for the Vatankhah approximation

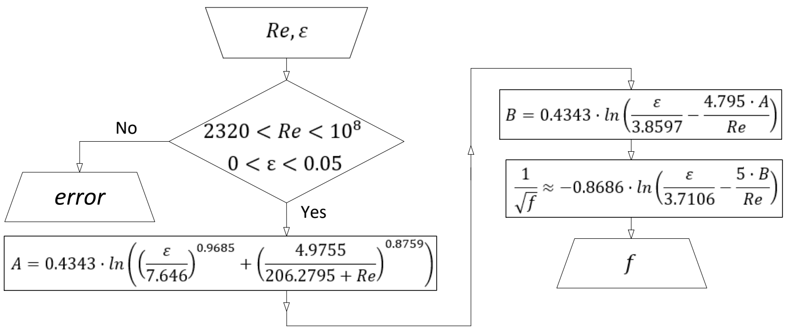
The VBA code based on algorithm from Fig. 9 is given as:

```
Function VATANKHAH(REYNOLDS As Double, EPSILON As Double) As Double
Dim A, B, f, ln As Double
If REYNOLDS < 2320 Or REYNOLDS > 100000000# Or EPSILON < 0 Or EPSILON > 0.05 Then
VATANKHAH = CVErr(xlErrValue)
Exit Function
End If
ln = 2 / Log(10)
A = 0.12363 * REYNOLDS * EPSILON + Log(0.3984 * REYNOLDS)
B = ((1 + A) / (0.52 * Log(ln * A))) − (A / (1 + A))
B = 1 + (1 / B)
f = ln * Log(0.3984 * REYNOLDS / ((ln * A) ^ (A / (A + B))))
f = f ^ −2
VATANKHAH = f
End Function
```

3.2.4. Romeo et al. approximation

The Romeo et al. approximation [28] is given in Eq. (7) with the related algorithm in Fig. 10. Eq. (7) is improved by genetic algorithms [35,45,46].

$$\left.\begin{aligned}\frac{1}{\sqrt{f}} &\approx -0.8686\cdot\ln\left(\frac{\varepsilon}{3.7106}-\frac{5\cdot B}{Re}\right)\\ A &\approx 0.4343\cdot\ln\left(\left(\frac{\varepsilon}{7.646}\right)^{0.9685}+\left(\frac{4.9755}{206.2795+Re}\right)^{0.8759}\right)\\ B &\approx 0.4343\cdot\ln\left(\frac{\varepsilon}{3.8597}-\frac{4.795\cdot A}{Re}\right)\end{aligned}\right\} \tag{7}$$


Re, ε
No
$2320 < Re < 10^8$
$0 < \varepsilon < 0.05$
Yes
error
$A = 0.4343 \cdot ln\left(\left(\frac{\varepsilon}{7.646}\right)^{0.9685} + \left(\frac{4.9755}{206.2795 + Re}\right)^{0.8759}\right)$
$B = 0.4343 \cdot ln\left(\frac{\varepsilon}{3.8597} - \frac{4.795 \cdot A}{Re}\right)$
$\frac{1}{\sqrt{f}} \approx -0.8686 \cdot ln\left(\frac{\varepsilon}{3.7106} - \frac{5 \cdot B}{Re}\right)$
f

Romeo et al. - Eq. (7)

Fig. 10 Algorithm for the Romeo et al. approximation

The VBA code based on algorithm from Fig. 10 is given as:

```
Function ROMEO(REYNOLDS As Double, EPSILON As Double) As Double
Dim A, B, f, ln As Double
If REYNOLDS < 2320 Or REYNOLDS > 100000000# Or EPSILON < 0 Or EPSILON > 0.05 Then
ROMEO = CVErr(xlErrValue)
Exit Function
End If
ln = 1 / Log(10)
A = ln * Log((EPSILON / 7.646) ^ 0.9685 + (4.9755 / (206.2975 + REYNOLDS)) ^ 0.8759)
B = ln * Log((EPSILON / 3.8597) − (4.795 * A / REYNOLDS))
f = −2 * ln * Log((EPSILON / 3.7106) − 5 * B / REYNOLDS)
f = f ^ −2
ROMEO = f
End Function
```

3.2.5. Buzzelli approximation

The Buzzelli approximation [27] is given in Eq. (8).

$$\left.\begin{array}{c} \frac{1}{\sqrt{f}} \approx A - \left(\frac{A+0.8686\cdot\ln\left(\frac{B}{Re}\right)}{1+\frac{2.1018}{B}}\right) \\ A \approx \frac{0.7314\cdot\ln(Re)-1.3163}{1.0025+1.2435\cdot\sqrt{\varepsilon}} \\ B \approx \frac{\varepsilon}{3.71}\cdot Re + 2.51\cdot A \end{array}\right\} \tag{8}$$

3.2.6.Praks and Brkić approximation based on the Wright ω-function and series expansion

The Praks and Brkić approximation [23] are based on the Wright ω-function and on its series expansion. It is given in Eq. (9).

$$\left.\begin{array}{c} \frac{1}{\sqrt{f}} \approx 0.8686\cdot(B+y) \\ A \approx \frac{Re\cdot\varepsilon}{8.0897} \\ B \approx \ln(Re) - 0.779626 \\ x \approx A + B \\ C \approx \ln(x) \\ y \approx C\cdot\left(\frac{1}{x-1}+\frac{C-2}{2\cdot x^2}\right) - 0.0014 \end{array}\right\} \tag{9}$$

3.2.7. Offor and Alabi approximation

The Offor and Alabi approximation [26] is given in Eq. (10).

$$\left.\begin{array}{c} \frac{1}{\sqrt{f}} \approx -0.8686\cdot\ln\left(\frac{\varepsilon}{3.71}-\frac{1.975\cdot A}{Re}\right) \\ A \approx \ln\left(\left(\frac{\varepsilon}{3.93}\right)^{1.092}+\frac{7.627}{Re+395.9}\right) \end{array}\right\} \tag{10}$$

3.2.8. Shacham approximation

The Shacham approximation [30] is given in Eq. (11) and is known also as Zigrang and Sylvester approximation [49].

$$\left.\begin{array}{c} \frac{1}{\sqrt{f}} \approx -08691\cdot\ln\left(\frac{\varepsilon}{3.7027}+\frac{5.0605\cdot B}{Re}\right) \\ A \approx 0.4343\cdot\ln\left(\frac{\varepsilon}{3.7027}+\frac{12.543}{Re}\right) \\ B \approx 0.4343\cdot\ln\left(\frac{\varepsilon}{3.7027}+\frac{5.0605\cdot A}{Re}\right) \end{array}\right\} \tag{11}$$

3.2.9. Lamri approximation

The Lamri approximation [24] is given in Eq. (12).

$$\left.\begin{aligned}\frac{1}{\sqrt{f}} \approx A + 0.8686 \cdot \left(\frac{0.8686}{B} - 1\right) \cdot \ln(B)\\ A \approx 0.8686 \cdot \ln\left(\frac{Re}{2.51}\right)\\ B \approx A + \frac{Re \cdot \varepsilon}{9.3125}\end{aligned}\right\} \quad (12)$$

4. Conclusions

Nine explicit approximations of the Colebrook equation are examined in this review paper. They are divided in two groups: 1) Extremely accurate approximations with the relative error of no more than 0.01% and 2) Very accurate approximations with the relative error between 0.01% and 0.1%. The most complex approximation is executed using the here presented VBA-Excel code only 2.06 times slower compared with the code for the simplest approximation of nine presented in this review paper. Therefore, using balance between the smallest relative error and the speed of execution in computers as a criterion for choosing the appropriate approximation for use in large computing simulations, the Praks and Brkić approximation [23] based on the Wright ω-function and on symbolic regression, given in this review paper in Eq. (4), is the most suitable and can be recommended for use. Almost equally suitable are approximations by Serghides [29], Vatankhah [25], and Romeo et al. [28].

UDFs written in the VBA, a common programming language for MS Excel spreadsheet solver prepared for the presented approximations to the Colebrook equation are suitable for use of those engineers who use spreadsheet solvers in their everyday work. Also, the shown UDF for the principal branch of the Lambert W-function [50] can find much wider use in engineering than those for solving of the Colebrook equation.

Acknowledgement: *The authors acknowledge a support from the Technology Agency of the Czech Republic through the project CEET – "Center of Energy and Environmental Technologies" TK03020027 and the Ministry of Education, Science, and Technological Development of the Republic of Serbia.*

References

1. Colebrook, C.F., 1939, *Turbulent flow in pipes, with particular reference to the transition region between the smooth and rough pipe laws*, Journal of the Institution of Civil Engineers, 11(4), pp. 133-156.
2. Colebrook, C.F., White, C.M., 1937, *Experiments with fluid friction in roughened pipes*, Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences, 161(906), pp. 367-381.
3. Moody, L., 1944, *Friction factors for pipe flow*, Transactions of the A.S.M.E., 66(8), pp. 671–684.
4. Praks, P., Brkić, D., 2018, *Choosing the optimal multi-point iterative method for the Colebrook flow friction equation*, Processes, 6(8), 130.
5. Praks, P., Brkić, D., 2018, *Advanced iterative procedures for solving the implicit Colebrook equation for fluid flow friction*, Advances in Civil Engineering, 2018, 5451034.
6. Pimenta, B.D., Robaina, A.D., Peiter, M.X., Mezzomo, W., Kirchner, J.H., Ben, L.H., 2018, *Performance of explicit approximations of the coefficient of head loss for pressurized conduits*, Revista Brasileira de Engenharia Agrícola e Ambiental, 22(5), pp. 301-307.
7. Winning, H.K., Coole, T., 2013, *Explicit friction factor accuracy and computational efficiency for turbulent flow in pipes*, Flow, Turbulence and Combustion, 90(1), pp. 1-27.
8. Brkić, D., 2012, *Determining friction factors in turbulent pipe flow*, Chemical Engineering (New York), 119, pp. 34-39.
9. Brkić, D., 2011, *Review of explicit approximations to the Colebrook relation for flow friction*, Journal of Petroleum Science and Engineering, 77(1), pp. 34-48.

10. Zigrang, D.J., Sylvester, N.D., 1985, *A review of explicit friction factor equations*, Journal of Energy Resources Technology, 107(2), pp. 280-283.
11. Gregory, G.A., Fogarasi, M., 1985, *Alternate to standard friction factor equation*, Oil & Gas Journal, 83(13), pp. 120-127.
12. Zeyu, Z., Junrui, C., Zhanbin, L., Zengguang, X., Peng, L., 2020, *Approximations of the Darcy–Weisbach friction factor in a vertical pipe with full flow regime*, Water Supply, 20(4), pp. 1321-1333.
13. Niazkar, M., Eryılmaz Türkkan, G., 2021, *Application of third-order schemes to improve the convergence of the Hardy Cross method in pipe network analysis*, Advances in Mathematical Physics, 2021, 6692067.
14. Praks, P., Brkić, D., 2020, *Suitability for coding of the Colebrook's flow friction relation expressed through the Wright ω-function*, Reports in Mechanical Engineering, 1(1), pp. 174-179.
15. Brkić, D., 2012, *Lambert W function in hydraulic problems*, Mathematica Balkanica (New Series), 26(3-4), pp. 285-292.
16. Viccione, G., Tibullo, V., 2012, *An effective approach for designing circular pipes with the Colebrook-White formula*, American Institute of Physics Conference Proceedings, 1479(1), pp. 205-208.
17. Brkić, D., 2011, *W solutions of the CW equation for flow friction*, Applied Mathematics Letters, 24(8), pp. 1379-1383.
18. Keady, G., 1998, *Colebrook-White formula for pipe flows*, Journal of Hydraulic Engineering, 124(1), pp. 96-97.
19. Hayes, B., 2005, *Why W?* American Scientist, 93(2), pp. 104-108.
20. Corless, R.M., Gonnet, G.H., Hare, D.E., Jeffrey, D.J., Knuth, D.E., 1996, *On the LambertW function*. Advances in Computational mathematics, 5(1), pp. 329-359.
21. Alfaro-Guerra, M., Guerra-Rojas, R., Olivares-Gallardo, A., 2020. *Evaluación de la profundidad de recursión de la solución analítica de la ecuación de Colebrook-White en la exactitud de la predicción del factor de fricción*, Ingeniería, Investigación y Tecnología, 21(4), Epub 20-Nov-2020.
22. Brkić, D., 2017, *Solution of the implicit Colebrook equation for flow friction using Excel*, Spreadsheets in Education, 10(2). 4663.
23. Praks, P., Brkić, D., 2020, *Review of new flow friction equations: Constructing Colebrook explicit correlations accurately*, Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería, 36(3), 41.
24. Lamri, A.A., 2020, *Discussion of "Approximate analytical solutions for the Colebrook equation"*, Journal of Hydraulic Engineering, 146(2), 07019012.
25. Vatankhah, A.R., 2018, *Approximate analytical solutions for the Colebrook equation*, Journal of Hydraulic Engineering, 144(5), 06018007.
26. Offor, U.H., Alabi, S.B., 2016, *An accurate and computationally efficient explicit friction factor model*, Advances in Chemical Engineering and Science, 6(3), pp. 237-245.
27. Buzzelli, D., 2008, *Calculating friction in one step*, Machine Design 80(12), pp. 54–55.
28. Romeo, E., Royo, C., Monzón, A., 2002, *Improved explicit equations for estimation of the friction factor in rough and smooth pipes*, Chemical Engineering Journal, 86(3), pp. 369-374.
29. Serghides, T.K., 1984, *Estimate friction factor accurately*, Chemical Engineering (New York), 91(5), pp. 63–64.
30. Schorle, B.J., Churchill, S.W., Shacham, M., 1980, *Comments on: "An explicit equation for friction factor in pipe"*, Industrial & Engineering Chemistry Fundamentals, 19(2), pp. 228–230.
31. Gross, C.J., Campbell, J., Becker, A.J., Russo, C.V. Microsoft Technology Licensing LLC, 2020, *Automatically creating lambda functions in spreadsheet applications*, U.S. Patent Appl. 16/024,580.
32. Sonnad, J.R., Goudar, C.T., 2004, *Constraints for using Lambert W function-based explicit Colebrook–White equation*, Journal of Hydraulic Engineering, 130(9), pp. 929-931.
33. Brkić, D., 2012, *Comparison of the Lambert W-function based solutions to the Colebrook equation*, Engineering Computations, 29(6), pp. 617–630.
34. Barry, D.A., Parlange, J.Y., Li, L., Prommer, H., Cunningham, C.J., Stagnitti, F., 2000, *Analytical approximations for real values of the Lambert W-function*, Mathematics and Computers in Simulation, 53(1-2), pp. 95-103.
35. Brkić, D., Ćojbašić, Ž., 2017, *Evolutionary optimization of Colebrook's turbulent flow friction approximations*, Fluids, 2(2), 15.
36. Brkić, D., Praks, P., 2019, *Accurate and efficient explicit approximations of the Colebrook flow friction equation based on the Wright ω-function*. Mathematics, 7(1), 34.
37. Olivares, A., Guerra, R., Alfaro, M., Notte-Cuello, E., Puentes, L., 2019, *Evaluación experimental de correlaciones para el cálculo del factor de fricción para flujo turbulento en tuberías cilíndricas*, Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería, 35(1), 15.

38. Mileikovskyi V., Tkachenko T., 2020, *Precise explicit approximations of the Colebrook-White equation for engineering systems*, Proceedings of EcoComfort, Lecture Notes in Civil Engineering
39. Muzzo L.E., Pinho D., Lima L.E.M., Ribeiro L.F. 2019, *Accuracy/speed analysis of pipe friction factor correlations*, Proceedings of INCREaSE 2019
40. Biberg, D., 2017, *Fast and accurate approximations for the Colebrook equation*, Journal of Fluids Engineering, 139(3), 031401.
41. Clamond, D., 2009. *Efficient resolution of the Colebrook equation*, Industrial & Engineering Chemistry Research, 48(7), pp. 3665-3671.
42. Winning, H.K., Coole, T., 2015, *Improved method of determining friction factor in pipes*, International Journal of Numerical Methods for Heat & Fluid Flow, 25(4), pp. 941–949.
43. Pérez-Pupo, J.R., Navarro-Ojeda, M.N., Pérez-Guerrero, J.N., Batista-Zaldívar, M.A., 2020, *On the explicit expressions for the determination of the friction factor in turbulent regime*, Revista Mexicana de Ingeniería Química, 19(1), pp. 313-334.
44. Brkić, D., Praks, P., 2019, *What can students learn while solving Colebrook's flow friction equation?* Fluids, 4(3), 114.
45. Ćojbašić, Ž., Brkić, D., 2013, *Very accurate explicit approximations for calculation of the Colebrook friction factor*, International Journal of Mechanical Sciences, 67, pp. 10-13.
46. Petrović, G., Mihajlović, J., Ćojbašić, Ž., Madić, M., Marinković, D., 2019, *Comparison of three fuzzy MCDM methods for solving the supplier selection problem*, Facta Universitatis-Series Mechanical Engineering, 17(3), pp. 455-469.
47. Sonnad, J.R., Goudar, C.T., 2006, *Turbulent flow friction factor calculation using a mathematically exact alternative to the Colebrook–White equation*, Journal of Hydraulic Engineering, 132(8), pp. 863-867.
48. Mikata, Y., Walczak, W.S., 2016, *Exact analytical solutions of the Colebrook-White equation*, Journal of Hydraulic Engineering, 142(2), 04015050.
49. Zigrang, D.J., Sylvester, N.D., 1982, *Explicit approximations to the solution of Colebrook's friction factor equation*, AIChE Journal, 28(3), pp. 514-515.
50. Kesisoglou, I., Singh, G., Nikolaou, M., 2021, *The Lambert function should be in the engineering mathematical toolbox*, Computers & Chemical Engineering, 148, 107259.