# Some explicit formulations of Colebrook–White friction factor considering accuracy vs. computational speed

O. Giustolisi, L. Berardi and T. M. Walski

## ABSTRACT

The Colebrook–White formulation of the friction factor is implicit and requires some iterations to be solved given a correct initial search value and a target accuracy. Some new explicit formulations to efficiently calculate the Colebrook–White friction factor are presented herein. The aim of this investigation is twofold: (i) to preserve the accuracy of estimates while (ii) reducing the computational burden (i.e. speed). On the one hand, the computational effectiveness is important when the intensive calculation of the friction factor (e.g. large-size water distribution networks (WDN) in optimization problems, flooding software, etc.) is required together with its derivative. On the other hand, the accuracy of the developing formula should be realistically chosen considering the remaining uncertainties surrounding the model where the friction factor is used. In the following, three strategies for friction factor mapping are proposed which were achieved by using the Evolutionary Polynomial Regression (EPR). The result is the encapsulation of some pieces of the friction factor implicit formulae within pseudo-polynomial structures.

**Key words** | Colebrook–White formula, computational speed, evolutionary polynomial regression, friction factor, pipe flow

**O. Giustolisi**
**L. Berardi** (corresponding author)
Department of Civil and Environmental Engineering,
Technical University of Bari,
Via E. Orabona 4,
I-70125 Bari,
Italy
E-mail: *l.berardi@poliba.it*

**T. M. Walski**
Haestad Methods Solution Center,
Bentley Systems, Incorporated,
27 Siemon Co Drive,
Suite 200 W,
Watertown,
CT 06795,
USA

## NOTATION, ABBREVIATIONS AND CONSTANT VALUES USED

| | |
|---|---|
| $D$: | pipe internal diamter [m]; |
| *EPR*: | Evolutionary Polynomial Regression; |
| $f$: | friction factor (dimensionless); |
| $f_\infty$: | fully rough friction factor (dimensionless); |
| $g$: | gravitational acceleration constant [9.806 m s$^{-2}$]; |
| $m$: | number of *EPR* pseudo-polynomial terms: |
| Ea: | exponent for scientific notation (i.e. $b\mathrm{E}a = b \times 10^a$) |
| Ke: | equivalent roughness (dimensionless); |
| Re: | Reynolds number (dimensionless); |
| Re$^*$: | friction/wall Reynolds number (dimensionless); |
| $V$: | flow velocity [ms$^{-1}$]; |
| $\mathbf{X}_i$: | $i$th *EPR* input variable; |
| $\mathbf{Y}$: | *EPR* output variable; |
| $\rho$ : | water density; |
| $\mu$ : | water viscosity; |
| $\nu$ : | water kinematic viscosity ($1.0079 \times 10^{-6}$ m$^2$ s$^{-1}$) |

## INTRODUCTION

The Darcy–Weisbach model for steady, uniformly distributed head losses reported in Equation (1) probably represents the most well-known formula where the friction factor $f$ is used to compute the slope hydraulic grade line $J$ (i.e. the head loss per

doi: 10.2166/hydro.2010.098

unit length of a pipe):

$$J = f(\mathrm{Ke}, \mathrm{Re})\frac{V^2}{2gD} \tag{1}$$

$$\mathrm{Re} = \frac{\rho VD}{\mu} = \frac{VD}{\nu}. \tag{2}$$

The friction factor $f$ depends on the equivalent roughness Ke and on the Reynolds number Re; $V$ is flow velocity; $D$ is pipe internal diameter; $g$ is the gravitational acceleration constant; $\rho$ is the water density, $\mu$ is the water viscosity and $\nu$ is the kinematic viscosity.

The Colebrook–White formulation of the friction factor $f$ (Colebrook & White 1937) is the most widely accepted one as reported in Equation (3):

$$\frac{1}{\sqrt{f}} = -2\log\left(\frac{\mathrm{Ke}}{3.71} + \frac{2.52}{\mathrm{Re}\sqrt{f}}\right) \tag{3}$$

where log is the base 10 logarithm.

Actually, using Equation (3) implicitly assumes that the Colebrook–White equation is a theoretically correct representation of empirical data, although it is often incorrect. In fact, it doesn't even represent Nikuradse's original data (Nikuradse 1932) all that well in some cases (Streeter 1971). Nonetheless, the Colebrook–White equation has been historically adopted since it provides sufficient accuracy for technical/engineering applications.

Equation (3) is assumed to describe the friction factor for a wide range of turbulent flow regimes (i.e. $\mathrm{Re} \geq 4000$): from transition to fully rough (turbulent) flow. In particular, the last term in brackets tends to be negligible with respect to the first one as the flow regime becomes fully rough. Thus, under fully rough flow Equation (3) goes back to the well-known formula of Prandtl–Karman, where the fully rough friction factor $f_\infty$ is explicit and depends solely on equivalent roughness Ke:

$$\frac{1}{\sqrt{f_\infty}} = -2\log\left(\frac{\mathrm{Ke}}{3.71}\right). \tag{4}$$

On this premise, the computation of the logarithm function (in both Equations (3) and (4)) in many computer languages is based on the computation of the natural logarithm. Therefore, the base 10 logarithm requires a base change. Thus, for computational reasons, Equation (3) is often written in the natural base considering the coefficient $1/\ln(10) = 0.4343$, where ln is the natural logarithm. Hence

$$\frac{1}{\sqrt{f}} = -0.8686\ \ln\left(\frac{\mathrm{Ke}}{3.71} + \frac{2.52}{\mathrm{Re}\sqrt{f}}\right). \tag{5}$$

Equations (3) and (5) are implicit and several attempts are reported in the technical literature in order to achieve an explicit formulation of $f$ to be used for computer applications where the iterative calculation could be cumbersome.

The explicit formulation of the Colebrook–White friction factor has been investigated by many authors (Idelchik 1994). Olujic (1981) found that the approximating expressions presented by Churchill (1977), Chen (1979) and Shacham (1980) provided the best accuracy, being within ± 1% of Colebrook–White's $f$. Other non-iterative expressions have been proposed by Barr (1981), Zigrang & Sylvester (1982), Serghides (1984) and Romeo *et al.* (2002). Recently Sonnad & Goudar (2007) proposed a mathematically exact formulations of the Colebrook–White $f$ whose maximum percent error is claimed to be less then $10^{-10}$, thus representing a point of reference in terms of accuracy.

However, it can be noted that all these equations were developed by looking at accuracy only, without accounting for actual computational requirements for massive numerical applications. In particular, all of them require the computation of some logarithms to achieve a sufficiently accurate solution, without caring how such an operation may affect real computer applications.

In fact, the computation of logarithm in many computer languages is based on series expansions that require several powers of the argument to be computed and added to each other (*Handbook of Mathematical Functions* 1964). The number of terms required varies according to the desired precision and might dramatically increase when the logarithm argument tends to 1. Equation (6) reports a classical Taylor series (first) along with a more efficient one (second) that converges quickly if $z$ is close to 1:

$$\ln(z) = \sum_{n=0}^{\infty} -\frac{(1-z)^n}{n}$$

$$= (z-1) - \left(\frac{z-1}{2}\right)^2 + \left(\frac{z-1}{3}\right)^3 - \left(\frac{z-1}{2}\right)^4 + \ldots \quad \text{with} \quad |1-z|<1$$

$$\ln(z) = 2\sum_{n=0}^{\infty} \frac{1}{2n+1}\left(\frac{z-1}{z+1}\right)^{2n+1} = 2\left(\frac{z-1}{z+1} + \frac{1}{3}\left(\frac{z-1}{z+1}\right)^3 + \frac{1}{5}\left(\frac{z-1}{z+1}\right)^5 + \ldots\right). \quad (6)$$

On the other hand, the convergence of the series used to compute the exponential function is quite fast, as well as the product and summations in commonly used computer languages. In fact, a methodology to improve the accuracy of the logarithm consists of computing a low-accuracy approximation $c \sim \ln(z)$ and then using the exponential function $e^c$ to correct the first guess by computing $\ln(z/e^c)$ through the second series of Equation (6) (i.e. $\ln(z) \sim c + \ln(z/e^c)$). In addition, when the argument is too large, then it is likely decomposed as $z = w \times 10^h$, so that $\ln(z) = \ln(w) + h\ln(10)$, thus requiring more than one logarithm to actually be computed. Therefore, the computation of logarithms is likely to represent the bottleneck in computing explicit formulae of $f$. Swamee & Rathie (2007) proposed a formulation of Colebrook–White's $f$ based on Lagrange's inversion expansion theorem, where $1/\sqrt{f}$ is approximated by a series made up of integer powers of Ke and Re: however, they neither reported any significant statistics on model performance nor a comparison with previous developments.

Davidson *et al.* (1999) used a symbolic regression approach to obtain a set of polynomial expressions of increasing complexity (i.e. computational cost) and accuracy. The main aim there was to eliminate the computation of the transcendental functions. Results were tested on Haaland's formula (1983) and proved that a 14-term polynomial equation may provide a mean squared error of about $2 \times 10^{-4}$. However, no speed/numerical analyses of formulae were reported. Tufail & Ormsbee (2006) showed the interpolation capabilities of a genetic-programming-based technique on the same problem; the use of genetic programming to reproduce the resistance coefficient in corrugated pipes was proposed first by Giustolisi (2004). In a recent work Yıldırım (2009) proposed a systematic comparative analysis of the most currently available explicit models for the Colebrook–White equation, for a wide range of Ke and Re values, and the detailed evaluation on the error properties of these models were presented. Özger & Yıldırım (2009) presented an investigation of the accuracy of an adaptive neuro-fuzzy inference system (ANFIS) for the friction coefficient determination.

Two alternative formulations were presented by Vatankhah & Kouchakzadeh (2008, 2009) based on previous work by Sonnad & Goudar (2006). Both formulations are aimed at allowing for a faster computation of $f$ when the accuracy required for some applications is less than that achievable using Sonnad & Goudar's (2007) formula.

Probably the most widely used explicit approximation of the Colebrook–White formula was proposed by Swamee & Jain (1976). They basically approximated $1/\sqrt{f}$ in the logarithm argument of Equation (5) as a function of the Reynolds number:

$$\frac{1}{\sqrt{f}} \approx 2.2778\,\mathrm{Re}^{0.1} \Rightarrow f \approx 0.1927\,\mathrm{Re}^{-0.2} \quad (7)$$

in order to achieve the following explicit formulation of $f$:

$$\frac{1}{\sqrt{f}} = -2\log\left(\frac{\mathrm{Ke}}{3.71} + \frac{5.74}{\mathrm{Re}^{0.9}}\right) = -0.8686\,\ln\left(\frac{\mathrm{Ke}}{3.71} + \frac{5.74}{\mathrm{Re}^{0.9}}\right)$$

$$\Rightarrow f = \frac{1.3254}{\left[\ln\left(\frac{\mathrm{Ke}}{3.71} + \frac{5.74}{\mathrm{Re}^{0.9}}\right)\right]^2}. \quad (8)$$

Actually, if unlimited computing capacities were available, improving the accuracy of the $f$ prediction would be not a problem by itself. In fact, it can be easily found that using $f_\infty$ of Equation (4) as a starting guess of $f$ within the Colebrook–White formula, it takes about 14 iterations to achieve a maximum error of $1.4 \times 10^{-8}$ and a mean error of $2.2 \times 10^{-9}$. The same accuracy is obtained using the Swamee–Jain approximation of Equation (8) in 12 iterations. Unfortunately, every iteration requires computing one logarithm and realistic computer resources usually adopted for commercial software applications do not easily manage such an excessive burden.

This paper proposes developing an approximation of the Colebrook–White friction factor accounting for both accuracy and computational speed, so that resulting models can be profitable for future software applications.

## DEFINING THE RANGE OF VALIDITY OF EXPLICIT *f* FORMULAE

The first observation is that the approximation reported in Equation (7) is implicitly based on the assumption that the second term in brackets of the Colebrook–White formula (attributed to the fully rough flow) does not depend on equivalent roughness Ke, but rather is a function of the Reynolds number only. Actually, such an assumption is not true (this would make the implicit Colebrook–White formulation pointless).

In order to develop a more consistent (and accurate) explicit formulation, the Colebrook–White expression should be rewritten as a function of the so-called shear/wall Reynolds number $\mathrm{Re}^*$, which is defined as

$$\mathrm{Re}^* = \frac{\mathrm{KeRe}\sqrt{f}}{2\sqrt{2}.} \tag{9}$$

Recall that fully rough flow occurs when $\mathrm{Re}^* \geq 70$ (French 1985; Schlichting 1979).

Thus, Equation (5) becomes

$$\frac{1}{\sqrt{f}} = -0.8686\ \ln\left(\frac{\mathrm{Ke}}{3.71}\left(1+\frac{3.3054}{\mathrm{Re}^*}\right)\right) = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right)+\ln\left(1+\frac{3.3054}{\mathrm{Re}^*}\right)\right]. \tag{10}$$

Equation (10) shows the role of shear/wall Reynolds number $\mathrm{Re}^*$. For $\mathrm{Re}^* \geq 70$ the term $3.3054/\mathrm{Re}^*$ can be neglected, regardless of Ke (i.e. the dependency of $1/\sqrt{f}$ on Re can be neglected and the rough fully turbulent flow holds).

Thus the problem of approximating Equation (5) within an explicit formulation generally deals with encapsulating the right-hand side term of the following Equation (11) with a function dependent on Re and Ke, but not on *f*:

$$\mathit{Target} = \ln\left(1+\frac{3.3054}{\mathrm{Re}^*}\right). \tag{11}$$

For example, the Swamee Jain formula is equivalent to approximating the friction Reynolds number as follows:

$$\frac{\mathrm{KeRe}}{2\sqrt{2}\mathrm{Re}^*} = \frac{1}{\sqrt{f}} \approx 2.2778\mathrm{Re}^{0.1} \Rightarrow \mathrm{Re}^* \approx 0.1552\mathrm{KeRe}^{0.9}. \tag{12}$$

From this perspective Equation (10) can be used to define a meaningful range of the Reynolds number Re to be used in developing an explicit approximation of Equation (5). It should have a lower bound equal to 4000 (Moody 1944) and an upper bound obtained from the same $\mathrm{Re}^*$ ($\mathrm{Re}^* \geq 70$) for all Ke. Therefore, the maximum Reynolds number should depend on Ke. Such a value of Re can be obtained by assuming $\mathrm{Re}^* = x$ such that the Colebrook–White formula Equation (10) returns the corresponding friction factor $f_x$:

$$\frac{1}{\sqrt{f_x}} = -0.8686\ \ln\left(\frac{\mathrm{Ke}}{3.71}\left(1+\frac{3.3054}{x}\right)\right) \rightarrow \mathrm{Re}(\mathrm{Ke},\ x) = 2\frac{\sqrt{2}\,x}{\mathrm{Ke}\sqrt{f_x}}. \tag{13}$$

In the following, the upper bound of the range of Re (considered for developing approximations in the remainder of this work) is obtained by assuming $\mathrm{Re}^* = 100$, as reported below:

$$\mathrm{Re} = \left[4000;\ \mathrm{Re}_{\max} = \frac{2\sqrt{2}\cdot(x = \mathrm{Re}^* = 100)}{\mathrm{Ke}\sqrt{f_x}}\right] \quad \text{and} \quad \mathrm{Ke} \in [10^{-6}; 10^{-2}]. \tag{14}$$

The values of Re have been evenly distributed in the base 10 logarithm space.

It is worth noting that choosing the same range of Re independently on Ke might result in misleading conclusions about average model accuracy, especially for rough pipes (large Ke). In fact, for these pipes fully rough flow (where *f* is well approximated by $f_\infty$ in Equation (4)) occurs at lower Re values than smoother pipes, while the accuracy of developed models refers to a reduced portion of the Re range, where the error could be significant (this produces a biased estimation of average model performance). The Re range defined in Equation (14) is aimed at testing model performances on the same number of points, mainly corresponding to transition turbulent flow.

## BRIEF OVERVIEW ON EPR

From a system identification point of view (Ljung 1999), the Evolutionary Polynomial Regression (*EPR*) (Giustolisi & Savic 2006, 2009) is an evolutionary nonlinear stepwise regression that allows for a global exploration of the user-defined space of models and provides symbolic formulae for

them. The stepwise regression strategy of *EPR* originates from the Draper & Smith (1998) method which was aimed at selecting attributes for linear models, considering the objective of fitting a model to data. The space of 'linear solutions' is therefore explored by changing the model input according to a set of rules and by evaluating model agreement with data.

*EPR* generalizes the original stepwise regression method by considering nonlinear expressions, which are pseudo-polynomials. In *EPR* the space of models is defined by selecting a model class consisting of a base pseudo-polynomial structure, like Equation (15), the maximum number of terms $m$ and a set **EX** of possible exponents of the input variables $\mathbf{X}_i$:

$$\mathbf{Y} = a_0 + \sum_{j=1}^{m} a_j \cdot (\mathbf{X}_1)^{\mathbf{ES}(j,1)} \cdot \ldots \cdot (\mathbf{X}_k)^{\mathbf{ES}(j,k)}. \quad (15)$$

The construction of *EPR* model structures consists of selecting the exponents $\mathbf{ES}(j,i)$ of each $i$th variable in the $j$th term among the values defined in **EX** by using an evolutionary algorithm. Usually, the set of exponents **EX** also contains zero so that one or more variables (or even entire terms) can be deleted from the final model figure. Then, model parameters $a_j$ are estimated using either the least-squares or nonnegative least-squares methods. The search for models proceeds by simultaneously minimizing prediction error and model complexity (i.e. number of terms and number of inputs with non-null exponents).

The polynomial nature of the model ensures a two-way relationship between each model structure and its parameters and, consequently, the parameter estimation phase is cast as a linear inverse problem. Furthermore, the exploration of the solution space is performed using an evolutionary computing approach working on a combinatorial problem which often does not have a unique solution. This approach results in the exploration of the nonlinear models (the linear model is a special case) having a pseudo-polynomial structure and ensuring linearity with respect to parameter estimation.

From a regressive standpoint, *EPR* has some beneficial features that were missing in other data-driven techniques. There is only a small number of constants to be estimated (thus avoiding over-fitting to data, especially for small datasets); the linear parameter estimation ensures the uniqueness of the solution when the inverse problem is well-conditioned; the automatic model construction avoids preselecting the functional form and number of parameters in the model; a transparent form of regression characteristics makes model selection easier since the multi-objective feature allows selecting final models not just based on fitting statistics.

Such peculiarities distinguish *EPR* from the more popular regression techniques like, for example, input–output artificial neural networks (ANNs) that require selecting proper transfer functions for hidden neurons in order to develop linear and nonlinear models; *EPR* does not require the assumption of the model structure, while ANNs generally require prior selection of the input vector and the number of hidden neurons as well as their unique transfer function.

In addition, the *EPR* provides a Pareto set of the best models, trading off parsimony against model fit (to training data), which is extremely useful to help the user in selecting the right model based on the final modelling purpose. All these *EPR* features are aimed at overcoming some drawbacks encountered while using genetic programming, as described in more detail in Giustolisi & Savic (2006).

## THE PROPOSED APPROACH

In the following $\mathbf{X}_i$ are the product $v \times \mathrm{Re}$ (i.e. $v\mathrm{Re}$) and the equivalent roughness Ke. Note that ($v\mathrm{Re}$) has been used instead of Re in order to avoid numerical instability due to some negative powers of Re being comparable to the precision of the computing environment (MATLAB – The MathWorks®). In more detail, the order of magnitude of kinematic viscosity $v$ in $\mathrm{m^2\ s^{-1}}$ is about $10^{-6}$ (see notation); thus the minimum value of ($v\mathrm{Re}$) is about $4 \times 10^{-3}$. Actually, this does not impair the results since any power of the kinematic viscosity $v$ is a constant value as well as all terms in Ke only.

Based on the above observations, the search for an explicit expression that approximates the original Colebrook–White $f$ will follow three separate strategies here:

**Case 1:** a similar methodology adopted by Swamee & Jain summarized in Equation (7), considering the actual

dependence of $\sqrt{f}$ on Re and Ke (i.e. $f$(Re, Ke)) rather than on Re only:

$$\frac{1}{\sqrt{f}} = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right) + \ln\left(1 + \frac{9.3492}{\mathrm{KeRe}y}\right)\right] \rightarrow y \cong EPR(\mathrm{Ke},\ v\mathrm{Re}). \tag{16}$$

In particular, the approximation of $y = \sqrt{f}$ (instead of $y = f$) permits avoiding the computation of the square root (i.e. power of 0.5) when it is used to compute $1/\sqrt{f}$ in Equation (16).

**Case 2:** mapping the argument of the second logarithm in Equation (10):

$$\frac{1}{\sqrt{f}} = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right) + \ln(y)\right] \rightarrow y \cong EPR(\mathrm{Ke},\ v\mathrm{Re}). \tag{17}$$

It is worth noting that such an approach requires strictly positive values of $y$ for any combinations of Re and Ke to allow computing the logarithm function.

**Case 3:** mapping the second logarithm in the last part of Equation (10) in order to avoid the logarithm computation:

$$\frac{1}{\sqrt{f}} = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right) + y\right]$$
$$\rightarrow y = \ln\left(1 + \frac{3.3054}{\mathrm{Re}^*}\right) \cong EPR(\mathrm{Ke}, v\mathrm{Re}) \tag{18}$$

In other words, this case consists of approximating the logarithm with a summation of many terms containing powers of Re and Ke. Actually, from a computational point of view, this would mean emulating the algorithms used for computing the natural logarithm, which are based on series expansion of the argument (see Equation (6)).

## RESULTS

In this section some formulations obtained by using the above-mentioned strategies are reported. For each of them the maximum (*maxE*) and mean (*meanE*) percent errors with respect to the Colebrook–White formula are reported. For the sake of clarity their formulations are reported below, where $f_{CW}$ stands for the Colebrook–White friction factor computed with a precision $10^{-20}$:

$$meanE = mean\left|1 - \frac{f}{f_{CW}}\right|; \quad maxE = max\left|1 - \frac{f}{f_{CW}}\right| \tag{19}$$

In addition, the number of natural logarithms ($n$ln) required beyond ln(Ke/3.71) (which must be computed in all formulations) is also reported to allow for a straight comparison between the computational efforts required.

As a first result based on Case 1, the Swamee–Jain formula has been obtained. It requires two logarithms: one for computing the power of Re plus the second from the last part of Equation (20):

$$y_{Swamee-Jain} = 0.12116\ \ (v\mathrm{Re})^{-0.1}\ = 0.12116\ v^{-0.1}\ e^{-0.1\ \ln(\mathrm{Re})}$$
$$\frac{1}{\sqrt{f}} = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right) + \ln\left(1 + \frac{9.3492}{0.12116\ \ v^{-0.1}\mathrm{KeRe}^{0.9}}\right)\right] \tag{20}$$

($meanE = 1.1095\%$ - $maxE = 2.9072\%$–$n\ln = 2$).

*EPR* also returned two expressions (Equations (21) and (22)) which are formally similar to the original Swamee–Jain formula, even if their performances are significantly improved. In both cases the number of logarithms required to assess $f$ is 2 (apart from ln(Ke/3.71)):

$$y = f(\mathrm{Re}) = 0.046576(v\mathrm{Re})^{-0.2} + 0.074291$$
$$= 0.046576e^{-0.2\ln(v\mathrm{Re})} + 0.074291$$
$$\frac{1}{\sqrt{f}} = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right) + \ln\left(1 + \frac{9.3492}{(0.046576v^{-0.2}\mathrm{Ke})\mathrm{Re}^{0.8} + 0.074291\mathrm{KeRe}}\right)\right] \tag{21}$$

($meanE = 1.0566\%$ - $maxE = 2.4969\%$ - $n\ln = 2$).

$$y = f(\mathrm{Ke}, \mathrm{Re}) = 0.037796v^{-0.2}\mathrm{Re}^{0.8} + 1.9242\mathrm{Ke}^{0.8} + 0.077555$$
$$= 0.037796e^{-0.2\ln(v\mathrm{Re})} + 1.9242\mathrm{Ke}^{0.8} + 0.077555$$
$$\frac{1}{\sqrt{f}} = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right) + \ln\left(1 + \frac{9.3492}{(0.037796v^{-0.2}\mathrm{Ke})\mathrm{Re}^{0.8} + (1.9242\mathrm{Ke}^{1.8} + 0.077555\mathrm{Ke})\mathrm{Re}}\right)\right] \tag{22}$$

($meanE = 0.4079\%$ - $maxE = 1.5847\%$ - $n\ln = 2$).

In addition, the following expression for $y$ in Equation (23) for Case 1 was obtained, which does not require any power of Re to be computed. It requires just one logarithm beyond ln(Ke/3.71) and will be further analyzed in the remainder of this paper:

$$y = f(\mathrm{Ke},\mathrm{Re}) = 0.14489$$
$$\frac{1}{\sqrt{f}} = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right) + \ln\left(1 + \frac{64.5262}{\mathrm{KeRe}}\right)\right] \quad (23)$$

($meanE = 2.8431\%$ - $maxE = 12.1838\%$ - $n\ln = 1$).

Among the models based on Case 1, two further expressions are obtained which allow for more accurate prediction of $f$ without increasing the number of logarithms to be computed. They are detailed in Equations (24) and (25):

$$y = f(\mathrm{Ke},\mathrm{Re}) = 0.0466(\nu\mathrm{Re})^{-0.2} + 2.2071(\nu\mathrm{Re})^{0.4} + 39.7799\mathrm{Ke}^{1.8} + 0.059335$$
$$\frac{1}{\sqrt{f}} = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right) + \ln\left(1 + \frac{9.3492}{\mathrm{Re}(a_1x_1 + a_2x_2 + a_0)}\right)\right]$$
$$\text{with} \quad \begin{cases} x_1 = (\nu\mathrm{Re})^{-0.2} = e^{-0.2\ln(\nu\mathrm{Re})} \\ x_2 = x_1^{-2} \end{cases}$$
$$\text{and} \quad \begin{cases} a_1 = 0.0466\mathrm{Ke} \\ a_2 = 2.2071\mathrm{Ke} \\ a_0 = 39.7799\mathrm{Ke}^{2.8} + 0.059335\mathrm{Ke} \end{cases} \quad (24)$$

($meanE = 0.0930\%$ - $maxE = 0.3070\%$ - $n\ln = 2$).

$$y = f(\mathrm{Ke},\mathrm{Re}) = 0.045975(\nu\mathrm{Re})^{-0.2} + 6.9771\mathrm{Ke}^{0.7}(\nu\mathrm{Re})^{0.4} + -66.0691\mathrm{Ke}^{1.2}(\nu\mathrm{Re})^{0.8} - 0.030081\mathrm{Ke}^{0.3} + 0.061242$$
$$\frac{1}{\sqrt{f}} = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right) + \ln\left(1 + \frac{9.3492}{\mathrm{Re}(a_1x_1 + a_2x_2 + a_3x_3 + a_0)}\right)\right]$$
$$\text{with} \quad \begin{cases} x_1 = (\nu\mathrm{Re})^{-0.2} = e^{-0.2\ln(\nu\mathrm{Re})} \\ x_2 = x_1^{-2} \\ x_3 = x_2^2 \end{cases}$$
$$\text{and} \quad \begin{cases} a_1 = 0.045975\mathrm{Ke} \\ a_2 = 6.9771\mathrm{Ke}^{1.7} \\ a_3 = -66.0691\mathrm{Ke}^{2.2} \\ a_0 = 0.061242\mathrm{Ke} - 0.030081\mathrm{Ke}^{1.3} \end{cases} \quad (25)$$

($meanE = 0.0513\%$ - $maxE = 0.1929\%$ - $n\ln = 2$).

These formulations allow for reducing the mean and maximum percent errors by about one order of magnitude lower than the Swamee–Jain formula, although they still require two logarithms to be computed.

Results obtained by using Case 2 strategy are omitted here since they did not actually overcome those obtained by adopting Case 1 in terms of accuracy, while the number of logarithms required (i.e. computational cost) was the same.

A significant reduction of the number of logarithms has been obtained using Case 3, where $y$ reproduces the entire second logarithm of Equation (10). The formulation in Equation (26) refers to this last case.

Such a formulation requires computing one logarithm only (apart from ln(Ke/3.71)), which refers to computing the exponential $(\nu\mathrm{Re})^{0.1} = \mathrm{e}^{0.1\ln(\nu\mathrm{Re})}$, and its accuracy largely overcomes that achievable by using the Swamee–Jain one on the same data. Moreover, Equation (26) has been written to be computed using classical single-threading programming, which requires subsequent products to be executed; nonetheless, such products could be easily parallelized if multi-threading programming was adopted to use multi-core computers:

$$y = (0.76881\mathrm{Ke}^{-0.1} - 0.78929)(\nu\mathrm{Re})^{-0.1} - 32.351\mathrm{Ke}^{0.1}(\nu\mathrm{Re})^{0.1} + + \left(243.9395\mathrm{Ke}^{0.5} + 274.0562\mathrm{Ke}^{0.7}\right)(\nu\mathrm{Re})^{0.5} + - \left(1934.9751\mathrm{Ke}^{0.9} + 5100.2044\mathrm{Ke}^{1.1}\right)(\nu\mathrm{Re})^{0.9} + + 2305.9049\mathrm{Ke}^{1.1}(\nu\mathrm{Re})^{1.1} + 28930.5225\mathrm{Ke}^{1.5}(\nu\mathrm{Re})^{1.3} + + 10.0892 - 0.87698\mathrm{Ke}^{0.3}$$
$$\frac{1}{\sqrt{f}} = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right) + (a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 + a_5x_5 + a_6x_6 + a_0)\right]$$
$$z_{1d} = (\nu\mathrm{Re})^{0.1} = e^{0.1\ln(\nu\mathrm{Re})}$$
$$z_1 = (\nu\mathrm{Re})^{0.2} = z_{1d}^2$$
$$z_2 = (\nu\mathrm{Re})^{0.4} = z_1^2$$
$$\text{with} \quad \begin{cases} x_1 = (\nu\mathrm{Re})^{-0.1} = z_{1d}^{-1} \\ x_2 = (\nu\mathrm{Re})^{0.1} = z_{1d} \\ x_3 = (\nu\mathrm{Re})^{0.5} = z_2x_2 \\ x_4 = (\nu\mathrm{Re})^{0.9} = z_2x_3 \\ x_5 = (\nu\mathrm{Re})^{1.1} = z_1x_4 \\ x_6 = (\nu\mathrm{Re})^{1.3} = z_1x_5 \end{cases}$$
$$\text{and} \quad \begin{cases} a_1 = 0.76881\mathrm{Ke}^{-0.1} - 0.78929 \\ a_2 = -32.351\mathrm{Ke}^{0.1} \\ a_3 = 243.9395\mathrm{Ke}^{0.5} + 274.0562\mathrm{Ke}^{0.7} \\ a_4 = -\left(1934.9751\mathrm{Ke}^{0.9} + 5100.2044\mathrm{Ke}^{1.1}\right) \\ a_5 = 2305.9049\mathrm{Ke}^{1.1} \\ a_6 = 28930.5225\mathrm{Ke}^{1.5} \\ a_0 = 10.0892 - 0.87698\mathrm{Ke}^{0.3} \end{cases} \quad (26)$$

($meanE = 0.1589\%$ - $maxE = 0.5108\%$ - $n\ln = 1$).

The way Equation (26) is actually implemented depends on the computing environment.

If a more accurate prediction of $f$ is needed, the expression of $1/\sqrt{f}$ can be introduced in the second term of Equation (16) (in place of $1/y$) to achieve $maxE = 0.0112\%$ and $meanE = 0.0890\%$ with a number of logarithms $n\ln = 2$ to be computed. In the following this strategy for improving accuracy will be further exploited.

As a final remark, Equation (26) is based on the term $z_{1d}$ that represents the base logarithm upon which the expansion is based. The subscript *1d* emphasizes that such a base refer to 1 decimal digit (i.e. the exponents in *EPR* are set with step 0.1). However, a similar expansion could be based on a two-digit precision base $z_{2d} = (\nu\mathrm{Re})^{0.01}$; in this last case $z_{1d}$ could be obtained as $z_{2d}{}^{10}$. This way any accuracy on $f$ could be potentially achieved by increasing the number of expansion terms and/or the number of digits of the exponents, even without increasing the number of logarithms. Note that, from an implementation point of view, $z_{2d}{}^{10}$ does not require any additional exponential to be computed since it is obtained by multiplying other terms like $z_{2d}{}^{x}$ (with $x < 10$) that need to be computed first.

For example, the following Equation (27) allows for achieving $maxE = 0.1027\%$ and $meanE = 0.2689\%$; if this expression is included in Equation (16) (as mentioned in the model reported above) the accuracy can be further increased ($maxE = 0.0077\%$ - $meanE = 0.0395\%$ - $n\ln = 2$):

$$
\begin{aligned}
y = {} & 0.34432\mathrm{Ke}^{-0.13}(\nu\mathrm{Re})^{-0.12} - 0.53588(\nu\mathrm{Re})^{-0.1} \\
& - 33.4866\mathrm{Ke}^{0.1}(\nu\mathrm{Re})^{0.1} + 235.1682\mathrm{Ke}^{0.5}(\nu\mathrm{Re})^{0.5} + \\
& + 296.2652\mathrm{Ke}^{0.69}(\nu\mathrm{Re})^{0.51} + \\
& - \left(1485.7194\mathrm{Ke}^{0.9} + 5416.5735\mathrm{Ke}^{1.1}\right)(\nu\mathrm{Re})^{0.9} + \\
& + 3413.3539\mathrm{Ke}^{1.31}(\nu\mathrm{Re})^{1.32} + 30279.0025\mathrm{Ke}^{1.5}(\nu\mathrm{Re})^{1.29} + \\
& + 11.1791 - 1.0385\mathrm{Ke}^{0.19}
\end{aligned}
$$

$$
\frac{1}{\sqrt{f}} = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right) + (a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 + a_5x_5 + a_6x_6 + a_7x_7 + a_8x_8 + a_0)\right]
$$

$$
\begin{aligned}
z_{2d} &= (\nu\mathrm{Re})^{0.01} = e^{0.01\,\ln(\nu\mathrm{Re})} \\
z_{1d} &= (\nu\mathrm{Re})^{0.1} = e^{0.1\ln(\nu\mathrm{Re})} = z_{2d}^{10} \\
z_1 &= (\nu\mathrm{Re})^{0.4} = z_{1d}^{4}
\end{aligned}
$$

$$
\text{with} \quad \begin{cases}
x_1 = (\nu\mathrm{Re})^{-0.12} = z_{1d}^{-1}z_{2d}^{-2} \\
x_2 = (\nu\mathrm{Re})^{-0.1} = z_{1d}^{-1} \\
x_3 = (\nu\mathrm{Re})^{0.1} = z_{1d} \\
x_4 = (\nu\mathrm{Re})^{0.5} = z_1x_3 \\
x_5 = (\nu\mathrm{Re})^{0.51} = z_{2d}x_4 \\
x_6 = (\nu\mathrm{Re})^{0.9} = z_2x_4 \\
x_7 = (\nu\mathrm{Re})^{1.32} = z_{2d}^{2}z_2x_6 \\
x_8 = (\nu\mathrm{Re})^{1.29} = z_{2d}^{-1}z_2x_6
\end{cases}
$$

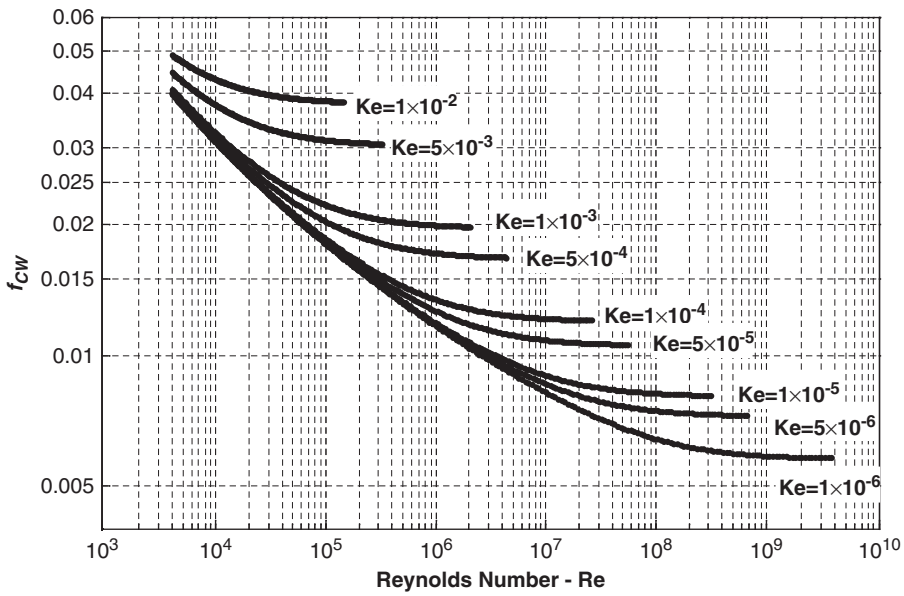


**Figure 1** | Data used for model development and tests.

and
$$\begin{cases} a_1 = 0.34432\mathrm{Ke}^{-0.13} \\ a_2 = -0.53588 \\ a_3 = -33.4866\mathrm{Ke}^{0.1} \\ a_4 = 235.1682\mathrm{Ke}^{0.5} \\ a_5 = 296.2652\mathrm{Ke}^{0.69} \\ a_6 = -\left(1485.7194\mathrm{Ke}^{0.9} + 5416.5735\mathrm{Ke}^{1.1}\right) \\ a_7 = 3413.3539\mathrm{Ke}^{1.31} \\ a_8 = 30279.0025\mathrm{Ke}^{1.5} \\ a_0 = 11.1791 - 1.0385\mathrm{Ke}^{0.19} \end{cases} \quad (27)$$

($meanE = 0.2689\%$ - $maxE = 0.1027\%$ - $n\ln = 1$).

It is worth noting that even better performance could be obtained by adding Re terms in $y$ at the cost of increased complexity. On the one hand, achieving a maximum error even less than 1% on an empirical formula (like the Colebrook–White one) could be questionable due to the irreducible errors of the experimental apparatus where it was calibrated. On the other hand, the propagation of errors for nonlinear systems should be accounted for when the intensive computation of the friction factor is required, especially when low Reynolds numbers occur.

The choice among the formulations reported above should be driven by both available computing resources and the purpose of calculating $f$. Thus, using classical single-thread programming, the classical Swamee–Jain formula can be significantly improved at the cost of summing a constant term or just changing the exponent of Re. Otherwise, if multithreading programming resorting to multiple-core technology is available, then computing multiple terms of the polynomial expansion could be faster than computing one more logarithm.

## BENCHMARK COMPARISONS

As stated above, the propagation of errors for nonlinear systems should be accounted for when the intensive computation of the friction factor is required. Thus the selection of the explicit friction factor formula should be based on the best trade-off between accuracy (depending on simulation type and problem size) and computational speed (depending on problem size and computing environment).

Therefore, this section compares the performances of some approximations of the Colebrook–White friction factor reported above in terms of both accuracy and computational speed. As first, data used both for developing *EPR* models and comparing them with each other have been plotted in Figure 1. It basically reproduces Moody's diagram except that, for each equivalent roughness considered, it consists of 200 points between $\mathrm{Re} = 4000$ and $\mathrm{Re}^* = 100$. The values of Colebrook–White $f_{CW}$ have been obtained by iteratively solving the implicit equation starting from the Swamee–Jain approximation until a precision of about $10^{-20}$ was achieved. The entire database used consisted of 1800 points.

All numerical tests have been performed using a Pentium T4200 processor, deselecting the two cores' parallelization capability in the MATLAB environment.

Based on these data, the following formulations have been compared: Equations (20) (Swamee–Jain), (23), (25), (26) and (27). Each of them has been calculated 100,000 times on 900 data points (thus simulating a grid of 90 million runs), and relevant prediction error and time taken for calculation (i.e. computational speed) are reported in Figure 2 and Table 1. Computing times are obtained by using the built-in "cputime" function of MATLAB.

It is worth noting that the code has been optimized for the MATLAB language by resorting to matrix representation in order to speed up the computation of each 900-point dataset which has been repeated 100,000 times.

It is evident that Equations (26) and (27) perform better than the Swamee–Jain one both in terms of mean and maximum percent error and CPU time required. This proves that a significant reduction of CPU time needed is due to the elimination of one logarithm. Equation (25) allows further improvement in accuracy but at the cost of about 3s longer than Equation (27), because of the calculation of one more logarithm per run.

It is also interesting that Equation (23) is the fastest to be computed, although it is not very accurate. However, it could be a viable solution for those applications where extremely accurate $f$ values are not necessary but it is important to save as much time and memory as possible. In fact, Equation (23) does not require computing preliminary variables and calculation of $f$ is immediate. Moreover, it could be a good alternative formulation for large Ke and Re combinations.

Table 1 also reports the accuracy and CPU time required to compute the two exact formulations of Sonnad & Goudar (2007) (i.e. SG1 and SG2) on data used in this study. They are clearly quite a bit more accurate, although they take much

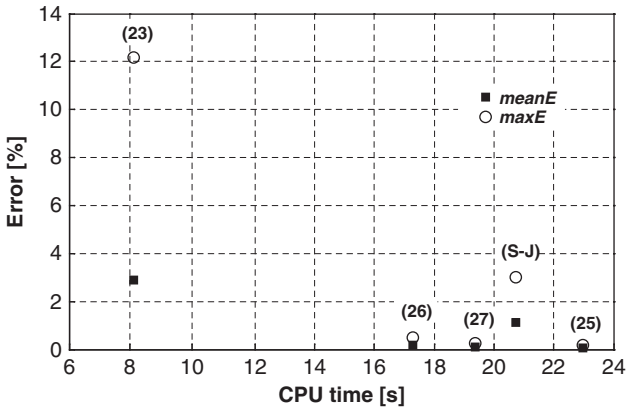


**Figure 2** | Comparison between different explicit formulations of *f* on 90 millions runs.

more time than the others. In addition, the last three rows refer to the performance of formulae reported in Sonnad & Goudar (2006) (i.e. SG3) and in Vatankhah & Kouchakzadeh (2008, 2009) (i.e. VK1 and VK2) on the same data. These last three formulae are about 15% faster than SG1, although their accuracy is significantly reduced.

A further analysis has been carried out to verify how such different explicit expressions of *f* affect the convergence of the Colebrook–White equation. In more detail, the *f* values obtained in the first analysis (i.e. Figure 2) have been introduced on the right-hand side of Equation (5) to obtain a new value of *f* (i.e. iteration 1); such an operation has been repeated up to 15 iterations. Accuracy and CPU time required for a progressively increasing number of iterations are reported in Figure 3 and Table 2. In other words, each point of Figure 3 encompasses the calculation of the relevant explicit formula, plus a number of successive iterations. Actually, the CPU time is expressed as a multiple of the time taken to compute the explicit Equation (23) (i.e. $t023 = 8.13$ s) which is the fastest of those analyzed.

For the sake of clarity data referred to *meanE* and *maxE* are drawn on two separate subplots of Figure 3, and percent error are on a log scale. The top-left points of each set refer to the explicit calculation (i.e. that of Figure 2 and Table 1); while the following 15 points correspond to 1, 2, 3, …, 15 iterations. Also in this case, the iterations have been executed for 90 million runs.

Thus, if the maximum percent error required is, for example, less than $10^{-4}$, then the explicit Swamee–Jain formula should be computed and six iterations of Equation (5) are needed (i.e. black diamonds in Figure 3), taking more than 63s of CPU time. On the other hand, Equation (25) (which was the slower explicit expression in Figure 2) achieves the same accuracy in five iterations after its direct application (i.e. triangles), taking about 59s. It is worth noting that, although Equation (23) reaches the same accuracy in seven iterations (i.e. black dots) after its direct calculation, it is quite a bit faster than Swamee–Jain (i.e. it takes about 58s).

Such an analysis proves that the proposed expressions represent more effective starting points for the Colebrook–White implicit formula than Swamee–Jain. Moreover, Figure 3

**Table 1** | Comparison of accuracy vs. computational speed of different explicit formula of *f*

| | *meanE [%]* | *maxE [%]* | *CPU time [s]* |
|---|---|---|---|
| Equation (20) (Swamee–Jain (1976))* | 1.11 | 2.99 | 20.73 |
| Equation (23) | 2.87 | 1.22E+01 | 8.13 |
| Equation (25) | 5.20E-02 | 1.94E-01 | 22.96 |
| Equation (26) | 1.59E-01 | 5.10E-01 | 17.28 |
| Equation (27) | 1.03E-01 | 2.59E-01 | 19.37 |
| Sonnad–Goudar (2007) - SG1* | 4.58E-05 | 3.64E-04 | 46.99 |
| Sonnad–Goudar (2007) - SG2* | 2.29E-09 | 1.43E-08 | 51.03 |
| Sonnad–Goudar (2006) - SG3* | 3.03E-01 | 8.76E-01 | 39.37 |
| Vatankhah–Kouchakzadeh (2008) - VK1* | 4.53E-02 | 1.58E-01 | 39.64 |
| Vatankhah–Kouchakzadeh (2009) - VK2* | 4.34E-02 | 1.50E-01 | 40.43 |

* Values computed using data plotted in Figure 1.

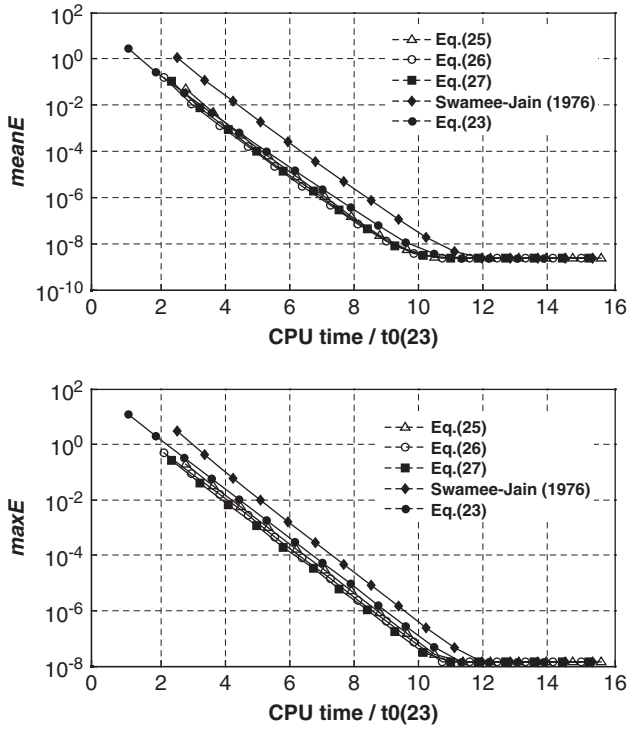


**Figure 3** | CPU time vs. accuracy achievable after 1–15 iterations of Colebrook–White formula using different explicit expressions of $f$ as initial point.

shows that their speed vs. accuracy performance are comparable, being quite close to each other.

Among the above-mentioned formulations Equation (26) represents a good compromise between accuracy and computational speed, thus being the more versatile with respect to different types of applications. Nonetheless, it requires a significantly higher storage of intermediate variables than Equation (23), which can be used when a wider interval of accuracy is allowed.

Overall, the choice among these two formulations depends on the required level of accuracy as reported in Figure 4, which partially reproduces Figure 3 with respect to Equations (23) and (26) only. For example, if a maximum percent error required is of about $10^{-5}$ (points in the dashed box), then Equation (23) is preferable since it takes less CPU time (i.e. about 65s) than its Equation (26) counterpart (i.e. about 67s). In contrast, if the maximum error allowed is $10^{-4}$ (points in the solid ellipse), then Equation (26) is faster than (23) since it requires less iterations. This is due to the different accuracy achieved at each iteration.

It is worth noting that all series of points in Figures 3 and 4 flatten out at mean and maximum percent error on $f$ of about $10^{-9}$ and $10^{-8}$, respectively. Such values represent the maximum precision achievable on $f$ from convergence of the Colebrook–White equation.

## ACCELERATING ITERATION OF COLEBROOK–WHITE FORMULA

Figures 3 and 4 clearly show that the improvement of accuracy obtained by iterating the Colebrook–White formula of Equation (16) is roughly linear in the semi-logarithmic diagram. In particular, regardless of the initial explicit formula, accuracy increases (i.e. percent error decreases) by about one order of magnitude every two iterations. However, it can be argued that every iteration a logarithm is computed, as reported on the right-hand side of Equation (28):

$$y = -0.8686\left[\ln\left(\frac{\mathrm{Ke}}{3.71}\right) + \ln\left(1 + \frac{9.3492}{\mathrm{ReKe}}y\right)\right]$$
$$\Rightarrow y - c = -0.8686\ln\left(1 + \frac{9.3492}{\mathrm{ReKe}}y\right)$$
$$\text{with} \quad c = -0.8686\ \ln\left(\frac{\mathrm{Ke}}{3.71}\right). \tag{28}$$

Actually, the difference between two successive iterations can be expressed in terms of the logarithm of a quantity ($z$) which is close to 1, as shown in the following equation:

$$y(i+1) + c = -0.8686\ln\left(1 + \frac{9.5054}{\mathrm{ReKe}}y(i)\right)$$
$$y(i+2) + c = -0.8686\ln\left(1 + \frac{9.5054}{\mathrm{ReKe}}y(i+1)\right)$$
$$y(i+2) - y(i+1) = -0.8686$$
$$\left[\ln\left(1 + \frac{9.5054}{\mathrm{ReKe}}y(i+1)\right) - \ln\left(1 + \frac{9.5054}{\mathrm{ReKe}}y(i)\right)\right]$$
$$\Rightarrow y(i+2) = -0.8686[\ln(z)] + y(i+1)$$
$$\text{with} \qquad z = \frac{\mathrm{ReKe} + 9.5054y(i+1)}{\mathrm{ReKe} + 9.5054y(i)}. \tag{29}$$

Therefore, the natural logarithm of $z$ can be approximated with one of the series, like the second of Equation (6). Actually, from numerical experiments it has been noted that the advantage of using the series expansion with respect to the original logarithm calculation (in terms of CPU

**Table 2** | Comparison of accuracy vs. computational speed using different approximations of $f$ as an initial value for the Colebrook–White equation

| | Eq. (25) | Eq. (26) | Eq. (27) | Eq. (20) (S-J) | Eq. (23) |
|---|---|---|---|---|---|
| *Iteration* | | | *meanE [%]* | | |
| 1 | 4.64E−03 | 1.13E−02 | 7.76E−03 | 1.17E−01 | 2.83E−01 |
| 2 | 5.04E−04 | 1.25E−03 | 8.39E−04 | 1.43E−02 | 3.48E−02 |
| 3 | 6.07E−05 | 1.60E−04 | 1.04E−04 | 1.85E−03 | 4.70E−03 |
| 4 | 7.85E−06 | 2.21E−05 | 1.39E−05 | 2.51E−04 | 6.76E−04 |
| 5 | 1.07E−06 | 3.18E−06 | 1.96E−06 | 3.52E−05 | 1.01E−04 |
| 6 | 1.55E−07 | 4.74E−07 | 2.86E−07 | 5.06E−06 | 1.54E−05 |
| 7 | 2.35E−08 | 7.29E−08 | 4.37E−08 | 7.44E−07 | 2.41E−06 |
| 8 | 5.31E−09 | 1.28E−08 | 8.19E−09 | 1.12E−07 | 3.80E−07 |
| 9 | 2.64E−09 | 3.69E−09 | 2.99E−09 | 1.84E−08 | 6.23E−08 |
| 10 | 2.30E−09 | 2.45E−09 | 2.37E−09 | 4.50E−09 | 1.13E−08 |
| 11 | 2.30E−09 | 2.30E−09 | 2.29E−09 | 2.44E−09 | 3.72E−09 |
| 12 | 2.30E−09 | 2.30E−09 | 2.30E−09 | 2.33E−09 | 2.37E−09 |
| 13 | 2.30E−09 | 2.30E−09 | 2.30E−09 | 2.29E−09 | 2.32E−09 |
| 14 | 2.30E−09 | 2.30E−09 | 2.30E−09 | 2.30E−09 | 2.29E−09 |
| 15 | 2.30E−09 | 2.30E−09 | 2.30E−09 | 2.30E−09 | 2.30E−09 |
| *Iteration* | | | *maxE [%]* | | |
| 1 | 3.26E−02 | 8.89E−02 | 3.97E−02 | 4.15E−01 | 1.97E+00 |
| 2 | 5.61E−03 | 1.54E−02 | 6.67E−03 | 6.19E−02 | 3.46E−01 |
| 3 | 9.68E−04 | 2.68E−03 | 1.15E−03 | 9.83E−03 | 5.99E−02 |
| 4 | 1.67E−04 | 4.65E−04 | 2.00E−04 | 1.62E−03 | 1.04E−02 |
| 5 | 2.88E−05 | 8.07E−05 | 3.45E−05 | 2.77E−04 | 1.81E−03 |
| 6 | 4.99E−06 | 1.40E−05 | 5.97E−06 | 4.80E−05 | 3.13E−04 |
| 7 | 8.64E−07 | 2.43E−06 | 1.03E−06 | 8.34E−06 | 5.44E−05 |
| 8 | 1.48E−07 | 4.23E−07 | 1.78E−07 | 1.45E−06 | 9.44E−06 |
| 9 | 2.70E−08 | 7.21E−08 | 3.21E−08 | 2.50E−07 | 1.64E−06 |
| 10 | 1.43E−08 | 1.43E−08 | 1.43E−08 | 4.48E−08 | 2.84E−07 |
| 11 | 1.43E−08 | 1.43E−08 | 1.43E−08 | 1.43E−08 | 5.06E−08 |
| 12 | 1.43E−08 | 1.43E−08 | 1.43E−08 | 1.43E−08 | 1.43E−08 |
| 13 | 1.43E−08 | 1.43E−08 | 1.43E−08 | 1.43E−08 | 1.43E−08 |
| 14 | 1.43E−08 | 1.43E−08 | 1.43E−08 | 1.43E−08 | 1.43E−08 |
| 15 | 1.43E−08 | 1.43E−08 | 1.43E−08 | 1.43E−08 | 1.43E−08 |
| *Iteration* | | | *CPU time / t023* | | |
| 1 | 3.69 | 3.02 | 3.26 | 3.44 | 1.88 |
| 2 | 4.57 | 3.90 | 4.17 | 4.31 | 2.75 |
| 3 | 5.44 | 4.80 | 5.06 | 5.20 | 3.64 |
| 4 | 6.32 | 5.65 | 5.92 | 6.07 | 4.51 |
| 5 | 7.21 | 6.52 | 6.86 | 6.93 | 5.39 |
| 6 | 8.07 | 7.42 | 7.67 | 7.81 | 6.26 |
| 7 | 8.96 | 8.29 | 8.58 | 8.69 | 7.14 |
| 8 | 9.83 | 9.17 | 9.44 | 9.56 | 8.03 |
| 9 | 10.70 | 10.04 | 10.33 | 10.44 | 8.89 |
| 10 | 11.59 | 10.95 | 11.19 | 11.32 | 9.77 |
| 11 | 12.45 | 11.81 | 12.07 | 12.19 | 10.66 |
| 12 | 13.35 | 12.68 | 12.95 | 13.08 | 11.52 |
| 13 | 14.21 | 13.58 | 13.85 | 13.95 | 12.40 |
| 14 | 15.07 | 14.43 | 14.74 | 14.83 | 13.29 |
| 15 | 15.96 | 15.33 | 15.57 | 15.70 | 14.14 |

time) progressively reduces as the number of terms in the series increases. Moreover, the series tends to converge more quickly as $z$ tends to 1. Thus it seems reasonable to apply the series expansion once a sufficient accuracy has been reached by using the classical Colebrook–White iterations.

Figure 5 and Table 3 show that using the series expansion allows saving about 20s in reaching the maximum accuracy allowed in the adopted computing environment. The iteration after which the series expansion of the logarithm is used corresponds to the deviation from the original (roughly linear) trend starting from the top-left points of each point set.

Such a figure also reports the performance of the two explicit formulae of Sonnad & Goudar (2007) (i.e. SG1 and SG2), which are taken as reference points in terms of accuracy, as well as formulae reported in Sonnad & Goudar (2006) (SG3) and in Vatankhah & Kouchakzadeh (2008, 2009) (VK1 and VK2). This plot further proves that the Sonnad–Goudar (2007) second expression (SG2) is an exact formulation since its accuracy is the same as that "asymptotically" (i.e. 15 iterations) achieved by previous formulae.

As a first remark, the fastest (but less accurate) explicit formula proposed by Sonnad & Goudar (2007) (SG1) is largely overcome in terms of both accuracy and computational speed by all *EPR*-based approximations drawn in the

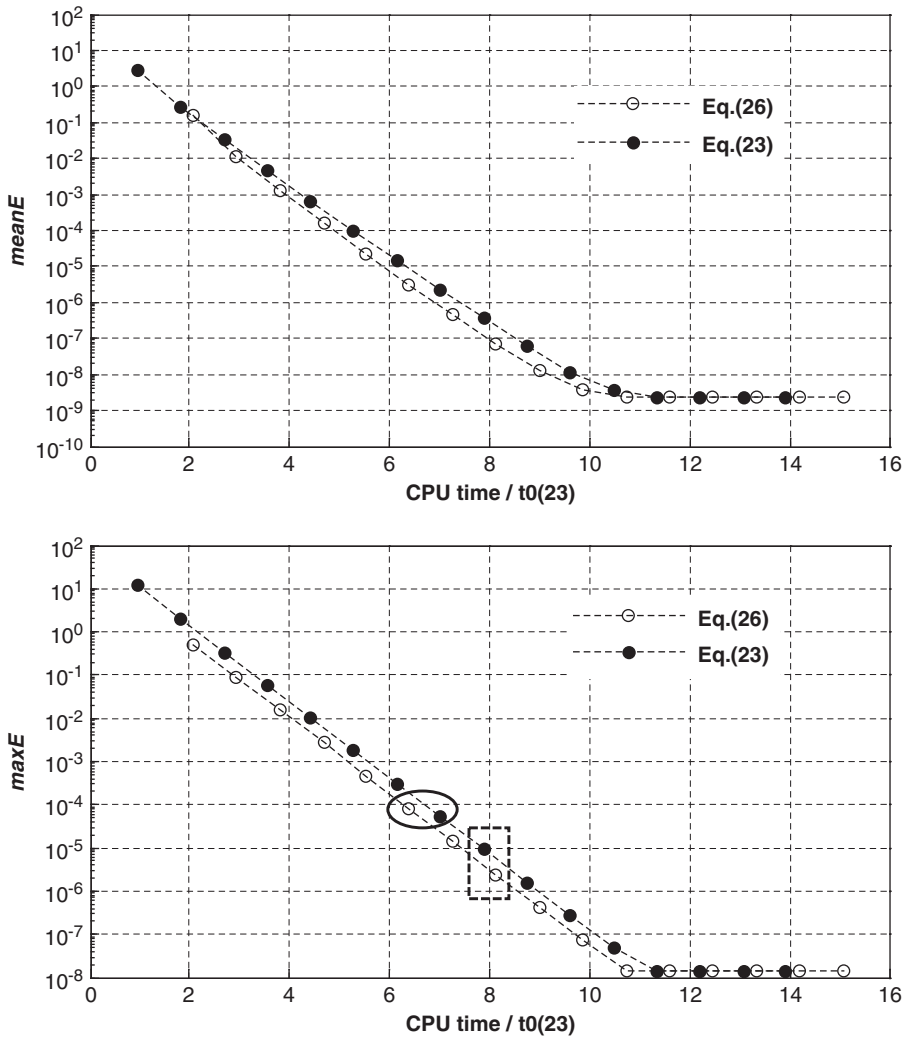


**Figure 4** | CPU time vs. accuracy achievable after 1–15 iterations of Colebrook–White formula using Equations (23) and (26) as initial points.

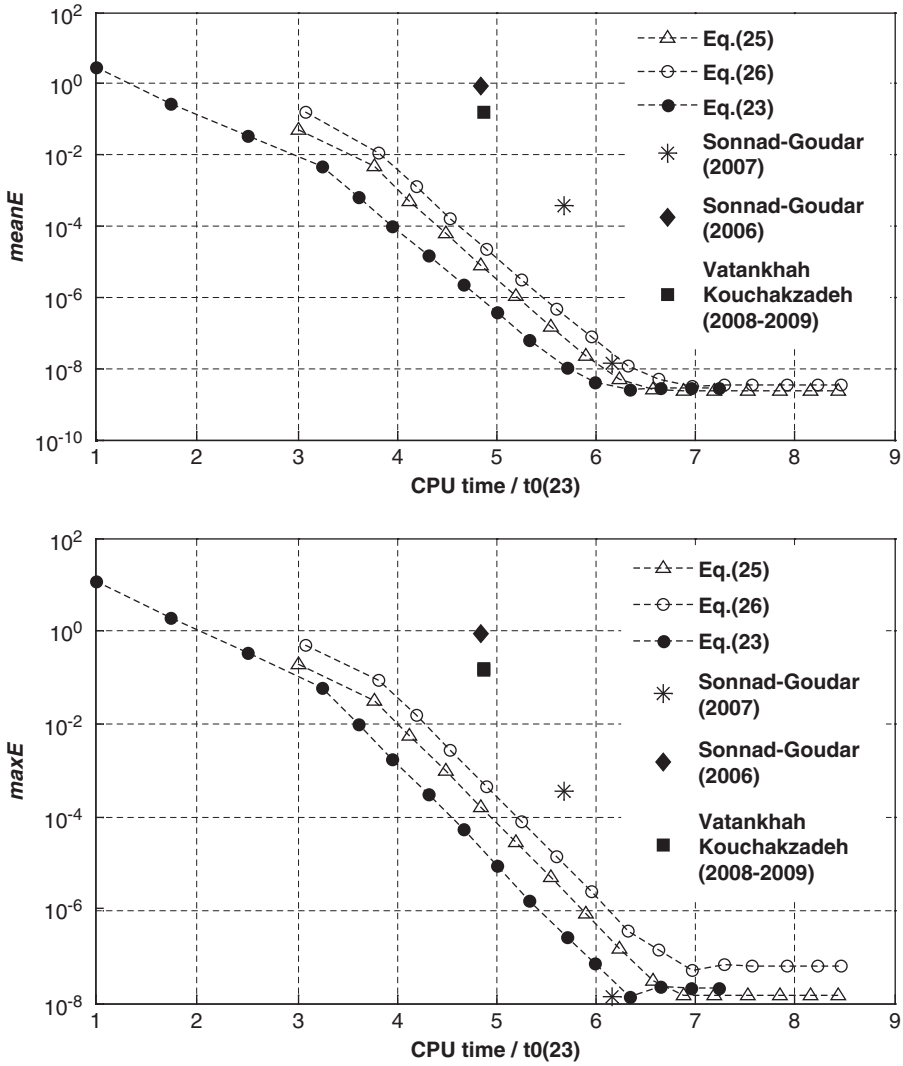


**Figure 5** | CPU time vs. accuracy achievable using *accelerated* iterations of Colebrook–White formula using Equations (23), (25) and (26) as initial points.

figure. A similar behavior is shown by formulae SG3, VK1 and VK2 whose computational speed increment has been obtained at the cost of a decreased accuracy.

Moreover, iterations based on Equations (23), (25) and (26) are further proved to represent an effective alternative to the most accurate Sonnad & Goudar explicit formula for a wide range of computational speeds and accuracies. This result is of foremost importance for those applications where the tuning accuracy of $f$ could result in a significant saving of computational time.

The fluctuations reported after the minimum percent errors are imputed to the fact that further accuracy improvement is comparable with the neglected terms of the series. However, such a phenomenon happens beyond the second point of the Sonnad–Goudar (2007) formulations (SG2), which is a reference point for our analyses.

## IMPLICATIONS

However, as reported above, the selection of the right formulation for $f$ should be based on the particular application. For example, the accuracy actually required for some software applications might be quite a bit less than that achievable with SG2 or the user might require setting a minimum level of accuracy in order to speed up each run. In these cases

**Table 3** | Comparison of accuracy vs. computational speed using Equations (23), (25) and (26) as approximations of *f* as an initial value for *accelerated* Colebrook–White iterative equation

| | Eq. (23) | Eq. (25) | Eq. (26) |
|---|---|---|---|
| *Iteration* | | *meanE [%]* | |
| 1 | 2.83E−01 | 4.64E−03 | 1.13E−02 |
| 2 | 3.48E−02 | 5.04E−04 | 1.25E−03 |
| 3 | 4.70E−03 | 6.07E−05 | 1.60E−04 |
| 4 | 6.76E−04 | 7.85E−06 | 2.21E−05 |
| 5 | 1.01E−04 | 1.07E−06 | 3.19E−06 |
| 6 | 1.54E−05 | 1.55E−07 | 4.72E−07 |
| 7 | 2.41E−06 | 2.36E−08 | 7.45E−08 |
| 8 | 3.79E−07 | 5.20E−09 | 1.16E−08 |
| 9 | 6.29E−08 | 2.64E−09 | 5.05E−09 |
| 10 | 1.08E−08 | 2.25E−09 | 3.21E−09 |
| 11 | 4.23E−09 | 2.28E−09 | 3.49E−09 |
| 12 | 2.54E−09 | 2.27E−09 | 3.44E−09 |
| 13 | 2.81E−09 | 2.28E−09 | 3.45E−09 |
| 14 | 2.77E−09 | 2.28E−09 | 3.45E−09 |
| 15 | 2.77E−09 | 2.28E−09 | 3.45E−09 |
| *Iteration* | | *maxE [%]* | |
| 1 | 1.97 | 3.26E−02 | 8.89E−02 |
| 2 | 3.46E−01 | 5.61E−03 | 1.54E−02 |
| 3 | 5.99E−02 | 9.68E−04 | 2.68E−03 |
| 4 | 1.04E−02 | 1.67E−04 | 4.65E−04 |
| 5 | 1.81E−03 | 2.88E−05 | 8.08E−05 |
| 6 | 3.13E−04 | 4.98E−06 | 1.39E−05 |
| 7 | 5.44E−05 | 8.67E−07 | 2.50E−06 |
| 8 | 9.42E−06 | 1.45E−07 | 3.57E−07 |
| 9 | 1.66E−06 | 3.01E−08 | 1.38E−07 |
| 10 | 2.63E−07 | 1.47E−08 | 5.25E−08 |
| 11 | 7.09E−08 | 1.47E−08 | 6.74E−08 |
| 12 | 1.43E−08 | 1.47E−08 | 6.48E−08 |
| 13 | 2.30E−08 | 1.47E−08 | 6.53E−08 |
| 14 | 2.12E−08 | 1.47E−08 | 6.52E−08 |
| 15 | 2.16E−08 | 1.47E−08 | 6.52E−08 |
| *Iteration* | | *CPU time /t023* | |
| 1 | 1.75 | 3.77 | 3.82 |
| 2 | 2.50 | 4.12 | 4.20 |
| 3 | 3.24 | 4.49 | 4.53 |
| 4 | 3.61 | 4.83 | 4.90 |
| 5 | 3.94 | 5.19 | 5.25 |
| 6 | 4.31 | 5.54 | 5.60 |
| 7 | 4.66 | 5.90 | 5.96 |
| 8 | 5.01 | 6.24 | 6.32 |
| 9 | 5.33 | 6.57 | 6.64 |
| 10 | 5.71 | 6.88 | 6.97 |
| 11 | 5.99 | 7.19 | 7.28 |
| 12 | 6.34 | 7.53 | 7.57 |
| 13 | 6.64 | 7.84 | 7.92 |
| 14 | 6.96 | 8.16 | 8.23 |
| 15 | 7.24 | 8.44 | 8.47 |

formula SG2 is not as versatile as the others reported above. In addition SG2 is difficult to be used for formulating integrals of $f$.

Therefore, Figure 5 encompasses different formulations which could be suited for different applications of $f$ as summarized below:

- Equation (23) is clearly the simplest one; it is easy to implement and does not require any additional storage capacity since any intermediate variable is computed; it allows tuning accuracy vs. speed performance since it spans a wide range of mean and max errors; however, it cannot be easily used when integrals of $f$ need to be formulated.
- Equation (26) is more articulated than Equation (23), requires a larger storage capacity for computing intermediate variables and might result in significantly slower computation for large size problems (where access to the hard disk might be necessary to store intermediate variables); it allows tuning accuracy vs. speed performance, even in a complementary way with Equation (26) as discussed above, for low accuracy intervals; it is suited for computing integrals of $f$.
- Equation (25) presents a similar drawback about the formulation of integral of $f$ with respect to Re, although its explicit computation is more accurate than Equation (23) and it ranges over a wide range of computational speeds.
- The SG2 explicit formula is probably the best suited for those applications where an exact reproduction of the Colebrook–White formula is required. However, its integral cannot be easily formulated for successive applications and its computational speed is severely limited by the calculation of four logarithms. Moreover it requires a comparable storage of intermediate variables as Equation (26) or (27) and could suffer from the same slowing-down effects for large size problems. Finally, it does not allow for tuning accuracy vs. speed performance.

## DERIVATIVES

Some applications of friction factor might require the computation of the derivative of $f$ with respect to Re (e.g. this is the case for some WDN hydraulic simulators). For the sake of completeness the following equation reports the expression of $df/d\mathrm{Re}$ obtained from Equations (9) and (10); it is explicit and its value can be easily computed based on $f$, whatever the formulation adopted:

$$\frac{df}{d\mathrm{Re}} = -\frac{g\left(\mathrm{Ke}, \mathrm{Re}, \sqrt{f}\right)}{1 + g\left(\mathrm{Ke}, \mathrm{Re}, \sqrt{f}\right)} \frac{2f}{\mathrm{Re}} \qquad \text{with}$$

$$g\left(\mathrm{Ke}, \mathrm{Re}, \sqrt{f}\right) = \frac{8.1207\sqrt{f}}{\mathrm{KeRe}\sqrt{f} + 9.3492}. \tag{30}$$

The order of magnitude of the maximum $df/d\mathrm{Re}$ value with respect to Ke and Re ranges reported in Equation (14) is about $10^{-6}$; thus it could be negligible from a numerical point of view although its computation is immediate.

## CONCLUSIONS

This work proposes an investigation on possible approaches to reproduce the Colebrook–White pipe friction factor $f$ with explicit expressions. While doing so, the formula proposed by Swamee & Jain (1976) is taken as a reference point since it is probably the most widely adopted (especially for computer routines/software).

As first, the CW formula has been written by using the friction Reynolds number ($\mathrm{Re}^*$) in order to obtain significant upper bounds of the Reynolds number (Re) range to be used for developing models.

The approach proposed is drawn from that adopted by Swamee–Jain and consists of modelling the friction factor in the second term of the logarithm argument in the Colebrook–White formula, by means of a function of Re and Ke only. Such models can be viewed to return a feasible first guess of $f$ to be used within the original Colebrook–White implicit formula. The *EPR* technique has been used to develop several alternative models that reproduce different representations of the target.

The models proposed herein represent different trade-offs between mathematical (and computational) complexity and accuracy. Obviously, an increased accuracy can be obtained at the price of augmented computational burden. However, the pseudo-polynomial structure of the *EPR* expression is well suited for parallelizing the computation of the model

output by taking advantage of multithreading technology. Thus, actual model complexity should be evaluated with respect to those (elementary) operations that require single-threaded programs and could be cumbersome in computer routines involving thousands of estimation of $f$.

This paper shows that some of the explicit expressions of $f$ obtained by reproducing the logarithm with an *EPR* model (i.e. Equations (26) and (27)) are significantly more accurate and fast than the Swamee–Jain expression. This is mainly due to the reduction of the number of logarithms involved. Accuracy could be further improved by using a slightly more articulate expression (Equation (25)) than Swamee–Jain, which requires computing the same number of logarithms but is quite a bit slower. Moreover, the analysis provides a very simple expression (Equation (23)), which is less accurate but significantly faster than the others, and thus it could be quite useful for some practical applications.

The second analysis demonstrates that formulae reported here represent even effective starting points for the iterative calculation of Colebrook–White $f$ in terms of time required to achieve any given accuracy. In particular the Swamee–Jain formula is proved to return less accurate results in a given CPU time or, conversely, to require the longest calculation to achieve a given accuracy.

A methodology to accelerate the Colebrook–White recursive formula has been presented which allows us to achieve progressively increasing accuracy at the cost of longer computational time, up to the maximum achievable precision represented by the Sonnad–Goudar (2007) exact formulation (SG2). In particular, the sets of possible trade-offs between accuracy and computational time based on the proposed formulae overcome both the fastest Sonnad–Goudar (2007) formulation (SG1) as well as other literature formulae (SG3, VK1 and VK2).

The formulations obtained by using the *EPR* are proved to represent useful alternatives with respect to some criteria dealing with the practical use of $f$ computation including possible tuning of computational speed vs. accuracy and ease of mathematical formulation of the integral of $f$.

These analyses emphasizes that accuracy is not the only criterion in selecting an explicit formula, especially for computer application, but it should be coupled with the computational speed required. Moreover, when high accuracy is necessary in order to combat propagation of errors on $f$ (e.g. in nonlinear systems), selecting a proper formulation helps converging to the desired accuracy in terms of computational time required rather than the mere number of iterations.

## REFERENCES


Barr, D. I. H. 1981 Solutions of the Colebrook-White function for resistance to uniform turbulent flow. *Proc. Inst. Civil Engng.* **71**, 529–535.

Chen, N. H. 1979 An explicit equation for friction factor in pipe. *Ind. Engng. Chem. Fundam.* **18**(3), 296–297.

Churchill, S. W. 1977 Friction factor equation spans all fluid-flow regimes. *Chem. Engng. Prog.* **84**(24), 91–92.

Colebrook, C. F. & White, C. M. 1937 Experiments with fluid friction in roughened pipes. *Proc. Royal Soc. Ser. A Math. Phys. Sci.* **161**(904), 367–381.

Davidson, J. W., Savic, D. & Walters, G. A. 1999 Method for the identification of explicit polynomial formulae for the friction in turbulent pipe flow. *J. Hydroinf.* **1**(2), 115–126.

Draper, N. R. & Smith, H. 1998 *Applied Regression Analysis*. John Wiley & Sons, New York

French, H. R. 1985 *Open Channel Hydraulics*. McGraw-Hill, New York

Giustolisi, O. 2004 Using Genetic Programming to determine Chèzy resistance coefficient in corrugated channels. *J. Hydroinf.* **6**(3), 157–173.

Giustolisi, O. & Savic, D. A. 2006 A symbolic data-driven technique based on Evolutionary Polynomial Regression. *J. Hydroinf.* **8**(3), 207–222.

Giustolisi, O. & Savic, D. A. 2009 Advances in data-driven analyses and modelling using EPR-MOGA. *J. Hydroinf.* **11**(3), 225–236.

Haaland, S. 1983 Simple and explicit formulas for the friction factor in turbulent pipe flow. *J. Fluid Engng.* **105**, 89–90.

*Handbook of Mathematical Functions* 1964 National Bureau of Standards (Applied Mathematics Series no.55), Washington, DC, p 68.

Idelchik, I. E. 1994 *Handbook of Hydraulic Resistances*. 3rd edn. Begell House, Redding, CT.

Ljung, L. 1999 *System Identifcation: Theory for the User* 2nd edn. Prentice-Hall, Englewood Cliffs, NJ.

Moody, L. F. 1944 Friction factors for pipe flow. *Trans. Am. Soc. Mech. Engng.* **66**, 671–684.

Nikuradse, J. 1932 *Gesetzmassigkeiten der turbulennten Stomung in glatten Rohren*. Forschungsheft Vol. 356, VDI-Verlag GMBH, Berlin, pp. 2 –35.

Olujic, Z. 1981 Compute friction factors fast for flow in pipes. *Chem. Engng.* Dec., 91–93.

Özger, M. & Yîldîrîm, G. 2009 Determining turbulent flow friction coefficient using adaptive neuro-fuzzy computing technique. *Adv. Engng. Software* **40**(4), 281–287.

Romeo, E., Royo, C. & Monzon, A. 2002 Improved explicit equations for estimation of the friction factor in rough and smooth pipes. *Chem. Engng.* **86**, 369–374.

Schlichting, H. 1979 *Boundary Layer Theory* 7th edn. McGraw-Hill, New York

Serghides, T. K. 1984 Estimate friction factor accurately. *Chem. Engng.* **91**, 63–64.

Shacham, M. 1980 Comment on "Explicit equation for friction factor in pipe". *Ind. Engng. Chem. Fundam*. **19**(5), 228–229.

Sonnad, J. R. & Goudar, C. T. 2006 Turbulent flow friction factor calculation using a mathematically exact alternative to the Colebrook–White equation. *J. Hydraul. Engng.* **132**(8), 863–867.

Sonnad, J. R. & Goudar C. T. 2007 Explicit reformulation of the Colebrook–White equation for turbulent flow friction factor calculation. *Ind. Engng. Chem. Res*. **46**, 2593–2600.

Streeter, V. L. 1971 *Fluid Mechanics* 5th edn. McGraw-Hill, New York

Swamee, P. K. & Jain, A. K. 1976 Explicit equations for pipe flow problems. *J. Hydraul. Engng.* **102**(5), 657–664.

Swamee, P. K. & Rathie, P. N. 2007 Exact equations for pipe-flow problems. *J. Hydrol. Res.* **45**(1), 131–134.

Tufail, M. & Ormsbee, L. E. 2006 A fixed functional set genetic algorithm (FFSGA) approach for functional approximation. *J. Hydroinf*. **8**(3) 193–206.

Vatankhah, A. R. & Kouchakzadeh, S. 2008 Discussion of "Turbulent flow friction factor calculation using a mathematically exact alternative to the Colebrook–White equation" by Jagadeesh, R., Sonnad, J.R., Chetan, T. & Goudar, C.T. *J. Hydraul. Engng.* **134**(8), 1187–1187.

Vatankhah, A. R. & Kouchakzadeh, S. 2009 Discussion of "Exact equations for pipe-flow problems" by P.K. Swamee. *J. Hydraul. Res.* **47**(4), 537–538.

Yıldırım, G. 2009 Computer-based analysis of explicit approximations to the implicit Colebrook–White equation in turbulent flow friction factor calculation. *Adv. Engng. Software* **40**(11), 1183–1190.

Zigrang, D. J. & Sylvester, N. D. 1982 Explicit approximation to the solution of the Colebrook's friction factor equation. *AIChE J*. **28**, 514–515.

First received 16 November 2009; accepted in revised form 10 February 2010. Available online 26 October 2010