A symbolic data-driven technique based on evolutionary polynomial regression

Orazio Giustolisi and Dragan A. Savic

ABSTRACT

This paper describes a new hybrid regression method that combines the best features of conventional numerical regression techniques with the genetic programming symbolic regression technique. The key idea is to employ an evolutionary computing methodology to search for a model of the system/process being modelled and to employ parameter estimation to obtain constants using least squares. The new technique, termed Evolutionary Polynomial Regression (EPR) overcomes shortcomings in the GP process, such as computational performance; number of evolutionary parameters to tune and complexity of the symbolic models. Similarly, it alleviates issues arising from numerical regression, including difficulties in using physical insight and over-fitting problems. This paper demonstrates that EPR is good, both in interpolating data and in scientific knowledge discovery. As an illustration, EPR is used to identify polynomial formulæ with progressively increasing levels of noise, to interpolate the Colebrook-White formula for a pipe resistance coefficient and to discover a formula for a resistance coefficient from experimental data.

Key words | Chézy resistance coefficient, Colebrook–White formula, data-driven modelling, evolutionary computing, regression

Orazio Giustolisi (corresponding author)
Faculty of Engineering, Department of Civil and Environmental Engineering,
Technical University of Bari,
via Turismo 8, Q. re Paolo VI, 74100, Taranto,
Italy
Tel: +39 080 596 4214
E-mail: *o.giustolisi@poliba.it*

Dragan A. Savic
Centre for Water Systems,
Department of Engineering, School of Engineering, Computer Science and Mathematics,
University of Exeter,
North Park Road, Exeter EX4 4QF,
UK
Tel: +44 1392 263637
E-mail: *d.savic@ex.ac.uk*

INTRODUCTION

The process of building mathematical models of complex systems based on observed data is usually called system identification. Colour coding of mathematical modelling is often used to classify models according to the level of prior information required, i.e. white-box models, black-box models and grey-box models (Ljung 1999; Giustolisi 2004):

- A *white-box model is a system where all necessary information is available*, i.e. the model is based on first principles (e.g. physical laws), known variables and known parameters. Because the variables and parameters have physical meaning, they also explain the underlying relationships of the system.
- A *black-box model is a system for which there is no prior information available*. These are data-driven or regressive models, for which the functional form of relationships between variables and the numerical parameters in those functions are unknown and need to be estimated.
- *Grey-box models are conceptual models* whose mathematical structure can be derived through conceptualisation of physical phenomena or through simplification of differential equations describing the phenomena under consideration. These models usually need parameter estimation by means of input/output data analysis, though the range of parameter values is normally known.

In addition to being founded on first principles, white-box models have the advantage of describing the underlying relationships of the process being modelled. However, the construction of white-box models can be difficult because the underlying mechanisms may not always be wholly understood, or because experimental results obtained in the laboratory environment do not correspond well to the prototype environment. Owing to these problems, approaches based on data-driven techniques are garnering considerable interest.

doi: 10.2166/hydro.2006.020

Although there exist other general-purpose data-driven techniques, artificial neural networks (ANN) and genetic programming (GP) are probably the most well known. Based on our present understanding of the brain and its associated nervous systems, ANN use highly simplified models composed of many processing elements ('neurons') connected by links of variable weights (parameters) to form black-box representations of systems (Haykin 1999). These models have the ability to deal with a great deal of information and to learn complex model functions from examples, i.e. by 'training' using sets of input and output data. The greatest advantage of ANN over other modelling techniques is their capability to model complex, non-linear processes without having to assume the form of the relationship between input and output variables. Learning in ANN involves adjusting the parameters (weights) of interconnections in a highly parametrised system. However, ANN require that the structure of a neural network is identified *a priori* (e.g. model inputs, transfer functions, number of hidden layers, etc). Furthermore, parameter estimation and over-fitting problems represent the principal disadvantages of model construction by ANN, as reported in Giustolisi & Laucelli (2005). Another difficulty with the use of ANNs is that they do not allow knowledge derived from known physical laws to be incorporated into the learning process.

Genetic programming (GP) is another modelling approach that has recently increased in popularity. It is an evolutionary computing method that generates a 'transparent' and structured representation of the system being studied. The most frequently used GP method is so-called *symbolic regression*, which was proposed by Koza (1992). This technique creates mathematical expressions to fit a set of data points using the evolutionary process of genetic programming. Like all evolutionary computing techniques, symbolic regression manipulates populations of solutions (in this case mathematical expressions) using operations analogous to the evolutionary processes that operate in nature. The genetic programming procedure mimics natural selection as the 'fitness' of the solutions in the population improves through successive generations. The term 'fitness', in this instance, refers to a measure of how closely expressions fit the data points. The nature of GP allows global exploration of expressions and allows the user to resolve further information on the system behaviour, i.e. gives an insight into the relationship between input and output data. However, the genetic-programming method of performing symbolic regression has some limitations. Principally, these are that GP is not very powerful in finding constants and, more importantly, that it tends to produce functions that grow in length over time (Davidson *et al.* 1999, 2000). Some notable attempts to mitigate those disadvantages have been reported by Zhang & Muhlenbein (1995), Soule & Foster (1999) and De Jong & Pollack (2003).

From a modelling point of view, a physical system having an output value y dependent on a set of inputs $\mathbf{X}$ and parameters $\boldsymbol{\theta}$, can be mathematically formalized as

$$y = F(\mathbf{X}, \boldsymbol{\theta}) \tag{1}$$

where F is a function in the space dimensionally equal to the number of inputs. Data-driven techniques, i.e. ANNs and GP, aim at reconstructing F from input/output data. Therefore, GP generates formulæ/models for F, coded in tree structures of variable size, performing a global search of the expression for F as symbolic relationships among $\mathbf{X}$ while parameters usually do not play a central role. On the other hand, ANNs derive their modelling properties from their ability to map F, maintaining at a lower level the knowledge of the functional relationships among $\mathbf{X}$. Indeed, the ANN goal is to map F, rather than to find a feasible structure for F.

SYMBOLIC REGRESSION

Davidson *et al.* (1999, 2000) introduced a new regression method for creating polynomial models based on both numerical and symbolic regression. They used GP to find the form of *polynomial expressions* and least squares optimisation to find the values for the constants in the expressions. The incorporation of least squares optimisation within symbolic regression was made possible by a rule-based component that algebraically transforms expressions into equivalent forms that are suitable for least squares optimisation. The paper describes an improved regression methodology for creating polynomial models.

The method presented in this paper builds on this idea to combine numerical and symbolic regression, with the elimination of the cumbersome, and often slow, rule-based component. The method also borrows from the idea of stepwise regression (Draper & Smith 1998), using

instead an evolutionary process based on genetic algorithms (Goldberg 1989) rather than following a hill-climbing method of stepwise regression.

Rule-based symbolic regression

One of the major advantages of GP, or more precisely symbolic regression, over numerical regression methods is that the user does not have to specify the form of the regression model in advance. Genetic programming finds the form of expressions as well as parameter values. However, if the optimal form of the model is known, obtaining parameter values by numerical methods is more efficient and ensures optimal values. Davidson *et al.* (1999, 2000) introduced a hybrid that combines the two approaches: using GP to evolve the form of the expressions while simultaneously optimising parameter values through numerical methods. Their methodology limits the range of operators normally used in symbolic regression to a subset consisting of addition, multiplication and non-negative integer powers. The expressions that result from applying the limited set of operators are in the form of polynomials. A rule-based program consisting of 56 unique rules algebraically transforms all resulting expressions through the evolutionary process of GP to the form of the right hand side of Equation (2):

$$y = \sum_{j=1}^{m} a_j \cdot z_j + a_o \tag{2}$$

where:

- y is the least squares estimate of the target value;
- a_j is an adjustable parameter for the jth term;
- a_0 is an optional bias;
- m is the number of terms/parameters in the expression;
- z_j is a transformed variable which is a function of the independent predictor variables, inputs, $x_1 x_2 \ldots x_k$, evaluated at the jth data point;
- k is the number of independent predictor variables (inputs).

In addition to transforming all expressions to this form, the rule-based program eliminates all non-functional code produced by evolutionary operations, such as terms formed by the product of zero and other coefficients. The rule base simplifies expressions by evaluating terms and coefficients that consist entirely of constants and replaces them with a single constant where possible. Once the rule base has transformed the expressions to the form in Equation (2) the program computes the optimal value for constants in the expression (adjustable parameters a_j) by the method of least squares. Davidson *et al.* (1999) provide a description of the rule based program and methods for optimising adjustable constants. However, they also describe the problem of 'combinatorial explosion', i.e. when a small polynomial expression requires an extremely lengthy transformation process. In an example, they show that a program applied rules over 3000 times to a simple expression.

Rule-based symbolic regression by Davidson *et al.* (1999) keeps the traditional genetic programming representation, i.e. uses a tree structure to represent expressions/computer programs. Unlike genetic algorithms which manipulate fixed-length 'chromosomes', the length or depth of the GP trees can vary as programs evolve. The structure of the tree is reflective of the hierarchical structure of the expressions/computer programs they represent.

EVOLUTIONARY POLYNOMIAL REGRESSION

General framework

As well as Rule-Based Symbolic Regression (R-BSR), Evolutionary Polynomial Regression (EPR) is a two-stage technique for constructing symbolic models: (i) structure identification, and (ii) parameter estimation. The main difference between the two approaches is in exploring the model structure space. While R-BSR uses traditional parse tree GP and rules to simplify expressions, EPR employs a simple Genetic Algorithm (GA) to search in the model structure space. The prior discussion on modelling techniques highlighted the fact that the underlying function F of the physical phenomena can be reconstructed assuming general mathematical mapping structures and estimating constants, i.e. through numerical regression by ANNs as in Giustolisi (2000), or by searching for symbolic structures more than estimating parameters. EPR searches for symbolic structures in the first stage by GA and estimates constant values by solving a Least Squares (LS) linear

problem in the second stage, thus assuming a biunique relationship between a structure and its parameters.

Over-fitting problems due to the flexibility of EPR may occur as in any other modelling strategy. Despite this, it is to be emphasised that parameter estimation is not the foremost problem causing over-fitting as in ANNs (Giustolisi & Laucelli 2005), because of the low number of parameters as will be shown. Accordingly, EPR will perform parameter estimation minimising the sum of squared errors (SSE), without employing a technique to avoid over-fitting, attending only to the numerical problems. On the other hand, an EPR model can fit training data very well because enough flexibility exists for describing a particular noise realisation by means of selecting an *ad hoc* structure. Consequently, this paper will discuss some techniques to avoid over-fitting for EPR: penalising the number of inputs involved in structures (model complexity); controlling the constant values whose term may describe noise when the related constant is close to zero; and controlling the variance of EPR terms with respect to noise variance in data (estimated by model residuals).

Model structure search

For the development of the new methodology, it is useful to transform Equation (2) into the following vector form:

$$\mathbf{Y}_{N\times 1}(\boldsymbol{\theta},\mathbf{Z}) = \left[\begin{array}{cccc}\mathbf{I}_{N\times 1} & \mathbf{Z}^j_{N\times m}\end{array}\right]\times\left[\begin{array}{cccc}a_0 & a_1 & \ldots & a_m\end{array}\right]^T = \mathbf{Z}_{N\times d}\times\boldsymbol{\theta}^T_{d\times 1} \quad (3)$$

where

- $\mathbf{Y}_{N\times 1}(\boldsymbol{\theta},\mathbf{Z})$ is the least squares estimate vector of N target values;
- $\boldsymbol{\theta}_{1\times d}$ is the vector of $d = m + 1$ parameters a_j, $j = 1{:}m$, and a_0;
- $\mathbf{Z}_{N\times d}$ is a matrix formed by $\mathbf{I}$, unitary column vector for bias a_0, and m vectors of variables $\mathbf{Z}^j$ that for a fixed j are a product of the independent predictor vectors of variables/inputs, $X = \langle X_1 X_2 \ldots X_k \rangle$.

The key idea of the EPR is to start from Equation (3) and search first for the best form of the function, i.e. a combination of vectors of independent variables (inputs), $\mathbf{X}_{S=1:k}$, and then to perform least squares regression to find the adjustable parameters $\boldsymbol{\theta}$ for each combination of inputs. To avoid the pitfalls of hill-climbing search methodologies, a global search algorithm is implemented for both the best set of input combinations and related exponents simultaneously, according to the user-defined cost function.

The matrix of inputs $\mathbf{X}$ is given as

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \ldots & x_{1k} \\ x_{21} & x_{22} & x_{23} & \ldots & x_{2k} \\ x_{31} & x_{32} & x_{33} & \ldots & x_{3k} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{N1} & x_{N2} & x_{N3} & \ldots & x_{Nk} \end{bmatrix} = \left[\begin{array}{ccccc}\mathbf{X}_1 & \mathbf{X}_2 & \mathbf{X}_3 & \ldots & \mathbf{X}_k\end{array}\right] \quad (4)$$

where the kth column of $\mathbf{X}$ represents the candidate variables for the jth term of Equation (3). Therefore, the jth term of Equation (3) could be written as

$$\mathbf{Z}^j_{N\times 1} = \left[(\mathbf{X}_1)^{\mathbf{ES}(j,1)} \cdot (\mathbf{X}_2)^{\mathbf{ES}(j,2)} \cdot (\mathbf{X}_3)^{\mathbf{ES}(j,3)} \cdot \ldots \cdot (\mathbf{X}_k)^{\mathbf{ES}(j,k)} \right] \quad \forall j = 1 \ldots m \quad (5)$$

where, $\mathbf{Z}^j$ is the jth column vector whose elements are products of candidate-independent inputs and $\mathbf{ES}$ is a matrix of exponents. Therefore, the problem is to find the matrix $\mathbf{ES}_{k\times m}$ of exponents whose elements can assume values within user-defined bounds.

For example, if a vector of candidate exponents for columns (inputs) in $\mathbf{X}$ is chosen to be $\mathbf{EX} = [-1, 0, 1]$ and $m = 4$ (the number of terms, bias excluded) and $k = 3$ (the number of candidate-independent variables/inputs), the polynomial regression problem is to find a matrix of exponents $\mathbf{ES}_{4\times 3}$. An example of such a matrix is given here:

$$\mathbf{ES}_{m\times k=4\times 3} = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 1 & -1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \quad (6)$$

When this matrix is substituted into Equation (5) the following set of expressions is obtained:

$$\begin{aligned}\mathbf{Z}_1 &= (\mathbf{X}_1)^{-1}\cdot(\mathbf{X}_2)^0\cdot(\mathbf{X}_3)^1 = \mathbf{X}_1^{-1}\cdot\mathbf{X}_3\\ \mathbf{Z}_2 &= (\mathbf{X}_1)^0\cdot(\mathbf{X}_2)^1\cdot(\mathbf{X}_3)^{-1} = \mathbf{X}_2\cdot\mathbf{X}_3^{-1}\\ \mathbf{Z}_3 &= (\mathbf{X}_1)^1\cdot(\mathbf{X}_2)^0\cdot(\mathbf{X}_3)^0 = \mathbf{X}_1\\ \mathbf{Z}_4 &= (\mathbf{X}_1)^1\cdot(\mathbf{X}_2)^1\cdot(\mathbf{X}_3)^0 = \mathbf{X}_1\cdot\mathbf{X}_2\end{aligned} \tag{7}$$

Therefore, based on the matrix given in Equation (6), the expression of Equation (3) is given as

$$\begin{aligned}\mathbf{Y} &= a_0 + a_1\cdot\mathbf{Z}_1 + a_2\cdot\mathbf{Z}_2 + a_3\cdot\mathbf{Z}_3 + a_4\cdot\mathbf{Z}_4\\ &= a_0 + a_1\cdot\mathbf{X}_3/\mathbf{X}_1 + a_2\cdot\mathbf{X}_2/\mathbf{X}_3 + a_3\cdot\mathbf{X}_1 + a_4\cdot\mathbf{X}_1\mathbf{X}_2\end{aligned} \tag{8}$$

The adjustable parameters a_j could now be computed by means of the linear Least Squares (LS) method using the minimisation of the sum of squared errors (SSE) as the cost function. Note that each row of **ES** determines the exponents of the candidate variables of the jth term in Equations (2) and (3). Each of the exponents in **ES** corresponds to a value from the user-defined vector **EX**. This allows the transformation of the symbolic regression problem into one of finding the best **ES**, i.e. the best structure of the EPR equation, e.g. in Equation (8).

The global search for the best form of Equation (8) is performed by means of a standard GA (Holland 1975; Goldberg 1989). The GA is an algorithmic model of Darwinian evolution that begins with the creation of a set of solutions referred to as a population of individuals. The parameters being optimised are coded using 'chromosomes', i.e. a set of character strings that are analogous to the chromosomes found in DNA. Standard GAs use a binary alphabet (characters may be 0's or 1's) to form chromosomes. Instead, integer GA coding is used here to determine the location of the candidate exponents of **EX** in the matrix **ES**. For example the positions in $\mathbf{EX} = [-1,\ 0,\ 1]$ correspond to the following string for the matrix of Equation (6) and the expression of Equation (8):

$$[1\,2\,3,\ 2\,3\,1,\ 3\,2\,2,\ 3\,2\,2] \tag{9}$$

Additionally, it is clear that the presence of at least one zero in **EX** ensures the ability to exclude some of the inputs and/or input combinations from the regression equation.

The following GA parameters were also used in the current EPR implementation:

1. Multiple-point crossover (Spears & De Jong 1991);
2. Single point mutation;
3. Ranking selection based on the normalised geometric distribution;
4. Termination criterion as a function of the length of the chromosome, the number of polynomial terms j and the number of inputs k in the matrix **X**.

Least squares solution by singular value decomposition

Computing a_j in Equation (8) is an inverse problem that corresponds to solving an over-determined linear system as a LS problem. This problem is traditionally solved by Gaussian elimination. However, an evolutionary search procedure may generate candidate solutions (e.g. a combination of exponents of **X**) that correspond to an ill-conditioned inverse problem. This often means that the rectangular matrix $\mathbf{Z}_{N\times d}$:

$$\mathbf{Z} = \left[\ \mathbf{I}_{N\times 1}\quad \mathbf{Z}^1_{N\times 1}\quad \mathbf{Z}^2_{N\times 1}\quad \mathbf{Z}^3_{N\times 1}\quad \cdots\quad \mathbf{Z}^m_{N\times 1}\ \right]_{N\times(m+1)=N\times d} \tag{10}$$

may not be of full rank (if a solution contains a column of zeros) or the columns $\mathbf{Z}^j$ are linearly dependent. This could pose serious problems to Gaussian elimination and a more robust solver is therefore needed. Parameter estimation of a_j (or $\boldsymbol{\theta}$) in EPR is performed by means of the Singular Value Decomposition (SVD) of the matrix **Z**. This approach makes the process of finding the solution to the LS problem more robust, although in general the SVD is slower than Gaussian elimination (Golub & Van Loan 1993). Finally, the Moore–Penrose pseudo-inverse (Golub & Van Loan 1993) or the Tikhonov (1963) regularisation method can be used to mitigate high condition number of **Z**. In this paper the LS solution is given by,

$$\theta = Pinv(\mathbf{Z})\times(\mathbf{Y}) \tag{11}$$

where *Pinv* is the pseudo-inverse matrix.

Model selection

In regression-based modelling the 'fitness' usually refers to a measure of how closely the regression expression fits the data points. However, it is widely accepted that the best modelling approach is also the simplest that fits the purpose of the

application. This principle, often called *Occam's razor*, is attributed to the medieval philosopher William of Occam (or Ockham, 1300–1349). The so-called *principle of parsimony* states that for a set of otherwise equivalent models of a given phenomenon one should choose the simplest one to explain a set of data. There is also a need to include a measure of trade-off between the model complexity (i.e. addition of new parameters) and the quality of fit in the fitness in regression-based models.

For a given set of observations or data, a regression-based technique needs to search among a large if not an infinite number of possible models to explain those data. By varying the exponents for the columns of matrix $\mathbf{X}$ and by searching for the best-fit set of parameters $\boldsymbol{\theta}$, the EPR methodology searches among all those models. It does, however, require an objective function that will ensure the best fit, without the introduction of unnecessary complexity. Unnecessary complexity is here defined as the addition of new terms or combinations of inputs that fit some noise in the raw data rather than the underlying phenomenon. Therefore, the key objective here is to find a systematic means to avoid the problem of over-fitting. There are three possible approaches to this problem: (1) to penalise the complexity of the expression by minimising the number of terms; (2) to control the variance of a_j constants (the variance of estimates) with respect to the their values; and (3) to control the variance of $a_j \cdot \mathbf{Z}_j$ terms with respect to the variance of residuals.

Complexity penalisation

In order to choose a model of optimal complexity corresponding to the smallest prediction (generalisation) error for future data, one needs to be able to compare two models with different levels of complexity and model fit. The sum of squared errors (SSE) is normally used to guide the search toward the best-fit model:

$$SSE = \frac{\sum_{i=1}^{N} (y_i(\boldsymbol{\theta}, \mathbf{Z}) - y_i)^2}{N} \tag{12}$$

where y_i are the target values in the training data set and $y_i(\boldsymbol{\theta},\mathbf{Z})$ are the model predictions computed by using the polynomial expression obtained by EPR. In order to allow the trade-off between the quality of fit (SSE) and the model complexity (number of input combinations), the following penalisation of complexity (PCS) fitness function is proposed:

$$PCS = \frac{SSE}{(Nd - px + 1)^{\alpha}} \tag{13}$$

where $Nd = k{\cdot}m$ is the maximum number of inputs that can be considered, px is the actual number of inputs selected by the GA and α is an adjustable exponent, which controls the degree of pressure to control complexity. This form of the fitness function will be better understood if the derivative of the fitness function with respect to px is derived:

$$\frac{\partial}{\partial px}\left(\frac{\mathrm{SSE}}{(Nd - px + 1)^{\alpha}}\right) = \frac{\partial \mathrm{SSE}}{\partial px} + \frac{\alpha \cdot \mathrm{SSE}}{Nd - px + 1} \tag{14}$$

The fitness decreases with respect to px if the derivative in Equation (14) is negative (see Figure 1). Therefore the following inequality should hold:

$$\frac{\alpha \cdot \mathrm{SSE}}{Nd - px + 1} \leq -\frac{\partial \mathrm{SSE}}{\partial px} = -VAR_{px}(\mathrm{SSE}) \tag{15}$$

In other words, the addition of another combination of inputs $\mathbf{X}$, needs not only to be justified on the basis of decreasing SSE, but also needs to take into account the terms (Nd-px) and a·SSE. The concept is shown in Figure 1. The bold line is the derivative of SSE with respect to px(P), while the curve is the natural SSE variation due to the increase in the number of input parameters. Equation (15) requires a value of the SSE derivative at P greater than or equal to the term on the left side of the inequality. Equation

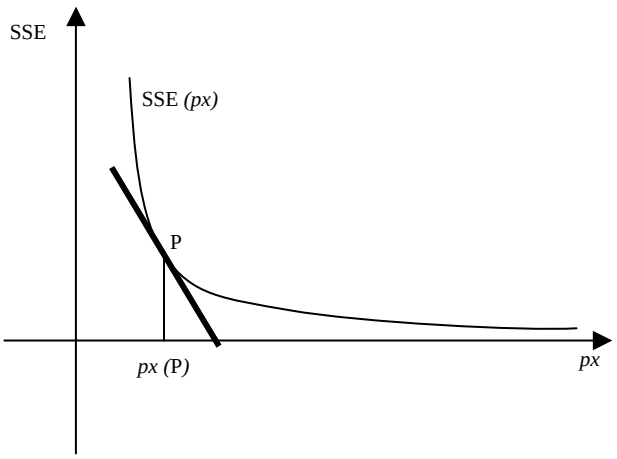

Figure 1 | SSE variation vs. *px*.

(15) illustrates that, when the actual number of inputs *px* approaches the maximum *Nd*, the left term of the inequality increases and, consequently, a high absolute variation of SSE is required (*VAR*(SSE) is always negative). This fact results in penalisation of complex structures by way of controlling the total number of inputs in the formula.

Variance of $\mathbf{a_j}$

EPR may control the polynomial term contribution to variance of $\mathbf{Y}$ expressed through their parameters during GA search. It may be argued that low constant value with respect to variance of estimates corresponds to terms that begin to describe noise in preference to the underlying function of phenomena. Therefore, the distribution of estimated constant values is used to eliminate those parameters whose value is not sufficiently larger than zero. Hence, we can write

$$\mathbf{P_N} = \lambda_o \cdot (Pinv(\mathbf{Z}) \times Pinv(\mathbf{Z}^T)) \tag{16}$$

where *Pinv* is the Moore–Penrose pseudo-inverse (Golub & Van Loan 1993) matrix (here used to be consistent with the SVD solution) of $\mathbf{Z}$, λ_0 is the noise variance estimated by Equation(12) and $\mathbf{P_N}$ is its covariance matrix. It is assumed that the parameter variation follows the Gaussian probability density function $N(a_{j0}, \mathbf{P_N})$. Hence, the following expression is used:

$$|a_{j0}| - \gamma \cdot \sqrt{P_{jj}} \cong |a_j| - \gamma \cdot \sqrt{P_{jj}} \leq 0 \Rightarrow a_j = 0 \tag{17}$$

where $\sqrt{P_{jj}}$ is the standard deviation of the estimated constant a_j from the diagonal elements of the covariance matrix and γ is the standard score (from the Standard Normal Table). Equation (17) states that if, for example, the modulus of the estimated constant value is lower than 2.578 $\sqrt{P_{jj}}$, which corresponds to a confidence level of 99%, the constant value is assumed to be equal to zero.

Variance of $\mathbf{a_j \cdot Z_j}$

EPR may control the polynomial term contribution to the variance of $\mathbf{Y}$ explained through evaluating $a_j \cdot \mathbf{Z_j}$ with respect to variance of the noise in the raw data during GA search. Indeed, a level of noise may exist under which the variance of the terms $a_j \cdot \mathbf{Z_j}$ will describe noise, causing over-fitting related problems. This level of noise is not known *a priori* and, therefore, the residual vector $\mathbf{E}$ could be used to estimate noise. In this manner, we can compare the standard deviation of $\mathbf{E}$ with the standard deviation of terms $a_j \cdot \mathbf{Z_j}$, obtaining

$$|\mathrm{StD}(a_j \cdot \mathbf{Z}_j)| < \beta \cdot |\mathrm{StD}(\mathbf{E})| \Rightarrow a_j = 0 \tag{18}$$

where β is a user-selected tuning parameter. In point of fact, it is not easy to choose β, but it is possible to consider $\beta = 1$ as giving a pressure to EPR for formulae having variance of each term greater than the variance of the residuals.

EXTENSION OF EVOLUTIONARY POLYNOMIAL REGRESSION

EPR allows pseudo-polynomial expressions as in Equations (2) and (3), allowing structures such as

$$\begin{aligned}
&\mathbf{Y} = a_0 + \sum_{j=1}^{m} a_j \cdot (\mathbf{X}_1)^{\mathbf{ES}(j,1)} \cdot \ldots \cdot (\mathbf{X}_k)^{\mathbf{ES}(j,k)} \cdot \\
&\qquad f\left((\mathbf{X}_1)^{\mathbf{ES}(j,k+1)}\right) \cdot \ldots \cdot f\left((\mathbf{X}_k)^{\mathbf{ES}(j,2k)}\right) && \text{case 0} \\
&\mathbf{Y} = a_0 + \sum_{j=1}^{m} a_j \cdot f\left((\mathbf{X}_1)^{\mathbf{ES}(j,1)} \cdot \ldots \cdot (\mathbf{X}_k)^{\mathbf{ES}(j,k)}\right) && \text{case 1} \\
&\mathbf{Y} = a_0 + \sum_{j=1}^{m} a_j \cdot (\mathbf{X}_1)^{\mathbf{ES}(j,1)} \cdot \ldots \cdot (\mathbf{X}_k)^{\mathbf{ES}(j,k)} \cdot \\
&\qquad f\left((\mathbf{X}_1)^{\mathbf{ES}(j,k+1)} \cdot \ldots \cdot (\mathbf{X}_k)^{\mathbf{ES}(j,2k)}\right) && \text{case2} \\
&\mathbf{Y} = g\left(a_0 + \sum_{j=1}^{m} a_j \cdot (\mathbf{X}_1)^{\mathbf{ES}(j,1)} \cdot \ldots \cdot (\mathbf{X}_k)^{\mathbf{ES}(j,k)}\right) && \text{case3}
\end{aligned} \tag{19}$$

Thus, EPR's model space may be extended by the structures in Equations (19), which remain based on polynomial regression as in Equation (3). User-specified functions *f* reported in Equations (19) may be natural logarithmic, exponential, tangent hyperbolic, etc. Note that the last structure in Equations (19) requires the assumption of an invertible function *g*, because of the subsequent stage of parameter estimation. The term 'pseudo-polynomial expressions' is used here because the parameters of any of the expressions in (19) can be computed as for a linear problem and/or for true polynomial expressions. Moreover, Equations (19) are transformed into the form of Equation (3) during evolutionary search. Finally, the inclusion of exponential and logarithmic functions in the general

expression of Equations (19) allows EPR to explore a large space of formulae where the analyst's understanding of the physical process warrants their inclusion. However, if such functions are not naturally describing the phenomenon being modelled EPR search would find exponent values for such inputs to be equal to zero.

TESTING THE ABILITY OF EPR TO RETAIN MATHEMATICAL STRUCTURE OF PHENOMENA

EPR was tested by generating some artificial outputs $\mathbf{Y}i$ as

$$\mathbf{Y} = a_0 + a_1 \cdot \mathbf{Z}_1 + a_2 \cdot \mathbf{Z}_2 + a_3 \cdot \mathbf{Z}_3 = 10 + 1 \cdot \mathbf{X}_1/\mathbf{X}_2 + 1 \cdot \mathbf{X}_2/\mathbf{X}_3 + 1 \cdot \mathbf{X}_3/\mathbf{X}_1 \quad (20)$$

$$\mathbf{Y}_i = 10 + 1 \cdot \mathbf{X}_1/\mathbf{X}_2 + 1 \cdot \mathbf{X}_2/\mathbf{X}_3 + 1 \cdot \mathbf{X}_3/\mathbf{X}_1 + N(0, \sigma i) \quad (21)$$

where $\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$, are three random variables (uniformly distributed random numbers in the range [0,1]), which play the role of inputs; and $N(0,\sigma_i)$ are four sequences of normally distributed random numbers having zero mean and standard deviation σ_i equal to 5%, 10%, 20%, 30% and 35% of the standard deviation of $\mathbf{Y}$. Table 1 reports some statistical values for $\mathbf{Y}$ and $\mathbf{Z}i$. Therefore, from Equations (20) and (21) it is possible to show $\boldsymbol{\epsilon}_i = N(0, \sigma_i) = \mathbf{Y}_i - \mathbf{Y}$ for $i = 1, 2, 3, 4, 5$. Figure 2 shows the output variable values $\mathbf{Y}$ (without noise) and error term $\boldsymbol{\epsilon}_i$, while Table 2 gives some error statistics.

The EPR approach was used to find the formula in Equation (21) starting from inputs ($\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$) and output (six $\mathbf{Y}_i$) data. The aim was to test the ability of EPR to get both the structure and constant values of Equation (21) with a progressively increasing level of noise $N(0,\sigma_i)$ introduced. The vector **EX** of candidate exponents was fixed to [0, −2, −1, 1, 2] and the PCS objective function (Equation (13)) was used. With respect to constants a_j, EPR will find one model for each number of constants given, thus the user has to select the number of constants and whether to include the bias term. For example, if the user selects 3, 4 and 5 constants, the EPR algorithm will sequentially search for three models not exceeding the prescribed maximum number of terms. However, shorter models are likely to be found, as shown in Table 4. The table gives the results of the tests performed with the following requirements: 3 models with 3, 4 and 5 constants (plus the bias term) for output $\mathbf{Y}_{i=0:4}$; 5 models with 3, 4, 5, 6, and 7 constants (plus the bias term) for output $\mathbf{Y}_{i=4}$; and 11 models with 3–13 constants (plus the bias term) for output $\mathbf{Y}_{i=5}$. Table 3 reports parameters for EPR evidencing the fact that EPR was forced to search in a larger model space (candidate structures through exponents and number of terms) than that of Equation (20).

The general conclusion that can be drawn from the results in Table 4 is that the agreement between the formulæ found by EPR and that of Equation (20) is excellent with respect to both the structure and parameter values. For example, for all 3 values for constants a_j the expression of Equation (20) is accurately reproduced in the first row of Table 4. Rows 2 and 3 give the resulting expressions for the variance of $\boldsymbol{\epsilon}$ equal to 2.6% and 15.9% of $a_2\mathbf{Z}_2$ variance, respectively. Despite the levels of noise introduced and the fact that the term $a_2\mathbf{Z}_2$ is very sensitive to it (see Table 2),

Table 1 | Some statistics of output **Y** and terms $\mathbf{Z}_i$

	Y	$\mathbf{Z}_1 = \mathbf{X}_1/\mathbf{X}_2$	$\mathbf{Z}_2 = \mathbf{X}_2/\mathbf{X}_3$	$\mathbf{Z}_3 = \mathbf{X}_3/\mathbf{X}_1$
Maximum	16.205	30.229	11.623	61.177
Minimum	13.030	0.026	0.024	0.080
Variance	50.972	20.299	4.132	39.382
Mean	16.205	2.379	1.612	2.214
	var(**Y**)	var($\mathbf{Z}_1$)/var(**Y**)	var($\mathbf{Z}_2$)/ var(**Y**)	var($\mathbf{Z}_3$)/ var(**Y**)
	50.972	0.398	0.081	0.773

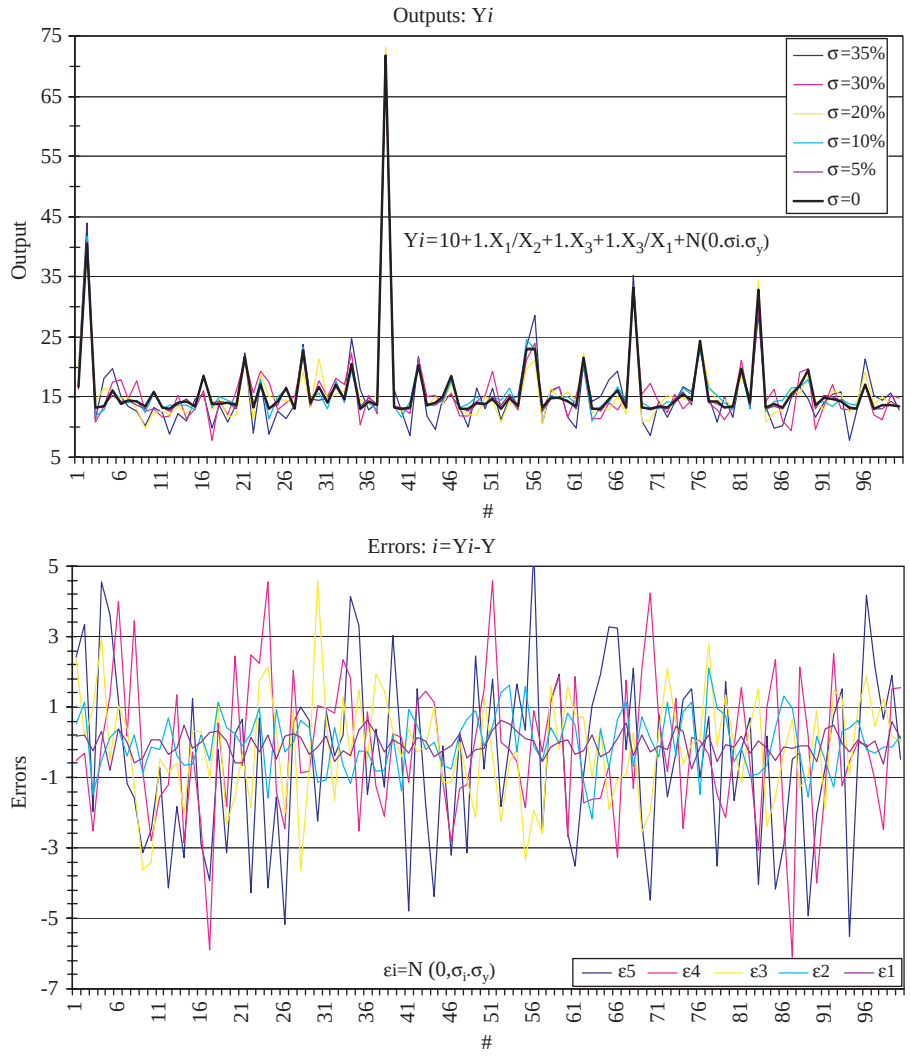

Figure 2 | Outputs of Equation (20) and errors of Equation (21).

Table 2 | Some statistics of errors

	ϵ_1	ϵ_2	ϵ_3	ϵ_4	ϵ_5
Maximum	0.629	2.106	4.573	4.570	5.491
Minimum	−0.799	−2.163	−3.651	−6.089	−5.521
Variance	0.108	0.659	2.296	4.067	6.037
Mean	−0.053	0.005	−0.148	−0.170	-0.437
$var(\epsilon_i)/var(a_1\mathbf{Z}_1)$	0.005	0.032	0.113	0.200	0.297
$var(\epsilon_i)/var(a_2\mathbf{Z}_2)$	0.026	0.159	0.556	0.984	1.461
$var(\epsilon_i)/var(a_3\mathbf{Z}_3)$	0.003	0.017	0.058	0.103	0.153

the EPR algorithm identified expressions very similar to that of Equation (20). Even in the cases of the ratio $var(\boldsymbol{\epsilon}_3)/var(a_2\mathbf{Z}_2)$ equal to 55.6% ($\mathbf{Y}_{i=3}$) and of the ratio $var(\boldsymbol{\epsilon}_4)/var(a_2\mathbf{Z}_2)$ equal to 98.4% ($\mathbf{Y}_{i=4}$), EPR identified the correct form of the equation and that of the term $a_2\mathbf{Z}_2$ (for $m = 6$ and m = 7 terms, respectively). In the final case, for the ratio $var(\boldsymbol{\epsilon}_5)/var(a_2\mathbf{Z}_2)$ equal to 146.1% ($\mathbf{Y}_{i=5}$), the term $a_2\mathbf{Z}_2$ was correctly identified for $m = 13$ (the last row).

Furthermore, referring to Table 4, it can be seen that for increasing variance of $\boldsymbol{\epsilon}_i$ EPR requires more than three terms to find Equation (20). This behaviour is expressed in the choice of parameters for the PCS criterion ($\alpha = 1$) that influences model selection in EPR. Indeed, for example in

Table 3 | Parameters of EPR

	Values
Search exponents	[−2 −1 0 1 2]
EPR type	Case 0 of Equations (19) without *function*
Number of a_j	See Table 4
Bias	Yes
Scale parameters	No scaling of Y and X
O.F. for model selection	Complexity penalisation $\alpha = 1$
GA population size	40
Crossover probability rate	0.4
Mutation probability rate	0.1

the case of $\boldsymbol{\epsilon}_3$, assuming $m = 3$ the formula in Table 4 is the best fitting with respect to PCS in Equation (13) because the true formula has $Nd\text{-}px + 1 = 9 - 6 + 1 = 4$, while the EPR formula has $Nd\text{-}px + 1 = 9 - 5 + 1 = 5$, thus being better for the PCS criterion. However, Nd becomes greater, increasing the number of terms m, for example $Nd = 12$ for $m = 4$. Thus the true formula is less penalised by the PCS criterion because $Nd\text{-}px + 1 = 12 - 6 + 1 = 7$ increases. This self-tuning effect of the PCS criterion results in the location of the correct formula when EPR is exploring in a larger model space.

The case for the ratio $\text{var}(\boldsymbol{\epsilon}_5)/\text{var}(a_2\mathbf{Z}_2)$, formula for $m = 13$, is different because an *error term* appears together with terms of the true formula in Equation (20). The effect of this *error term* is to describe the specific realisation of noise, but doing so, it corrupts the estimation of constant values of the other terms with respect to true formula in Equation (20). It is a difficult task to contend with *error terms* and specific strategies or physical insight (i.e. dimensional information) are required to undertake this.

Table 5 reports the number of generations and processor time for finding the best formula (results relate to a PC equipped with an Intel Pentium 4, 2600 MHz processor and the Windows XP operating system). Both the number of generations and computing time confirm that EPR is very fast in exploring the space of formulae. Clearly, this fact is directly related to EPR's evolutionary approach which transforms the general evolutionary search of GP into a very simplified evolutionary search for exponents, therefore requiring a simpler GA engine.

Table 4 | Selected EPR results

$Y_i - Y$	Number of a_j	EPR formulae
$\boldsymbol{\epsilon} = \mathbf{0}$	3, 4, 5	$\mathbf{Y}_{\text{EPR}} = 10 + 1 \cdot \mathbf{X}_1/\mathbf{X}_2 + 1 \cdot \mathbf{X}_2/\mathbf{X}_3 + 1 \cdot \mathbf{X}_3/\mathbf{X}_1$
$\boldsymbol{\epsilon}_1$	3, 4, 5	$\mathbf{Y}_{\text{EPR}} = 9.9884 + 1.0008 \cdot \mathbf{X}_1/\mathbf{X}_2 + 0.97882 \cdot \mathbf{X}_2/\mathbf{X}_3 + 0.99574 \cdot \mathbf{X}_3/\mathbf{X}_1$
$\boldsymbol{\epsilon}_2$	3, 4, 5	$\mathbf{Y}_{\text{EPR}} = 9.9983 + 0.99823 \cdot \mathbf{X}_1/\mathbf{X}_2 + 1.0317 \cdot \mathbf{X}_2/\mathbf{X}_3 + 0.98168 \cdot \mathbf{X}_3/\mathbf{X}_1$
$\boldsymbol{\epsilon}_3$	3	$\mathbf{Y}_{\text{EPR}} = 10.2326 + 0.78962 \cdot \mathbf{X}_1/\mathbf{X}_2 + 0.49317/\mathbf{X}_3 + 1.0136 \cdot \mathbf{X}_3/\mathbf{X}_1$
	4, 5	$\mathbf{Y}_{\text{EPR}} = 9.8774 + 1.0047 \cdot \mathbf{X}_1/\mathbf{X}_2 + 0.95116 \cdot \mathbf{X}_2/\mathbf{X}_3 + 1.019 \cdot \mathbf{X}_3/\mathbf{X}_1$
$\boldsymbol{\epsilon}_4$	3	$\mathbf{Y}_{\text{EPR}} = 13.7599 + 0.01685 \cdot \mathbf{X}_1/\mathbf{X}_2 + 0.01344/(\mathbf{X}_1)^2 + 0.018073/(\mathbf{X}_3)^2$
	6, 7	$\mathbf{Y}_{\text{EPR}} = 10.193 + 0.92388 \cdot \mathbf{X}_1/\mathbf{X}_2 + 0.92137 \cdot \mathbf{X}_2/\mathbf{X}_3 + 0.97504 \cdot \mathbf{X}_3/\mathbf{X}_1$
$\boldsymbol{\epsilon}_5$	3, 4	$\mathbf{Y}_{\text{EPR}} = 11.116 + 0.9688 \cdot \mathbf{X}_1/\mathbf{X}_2 + 0.0074/(\mathbf{X}_1)^2 + 0.4305/\mathbf{X}_1$
	13	$\mathbf{Y}_{\text{EPR}} = 9.33 + 1.054 \cdot \mathbf{X}_1/\mathbf{X}_2 + 0.735 \cdot \mathbf{X}_2/\mathbf{X}_3 + 0.685 \cdot \mathbf{X}_3/\mathbf{X}_1 + 0.297/\mathbf{X}_1$

Table 5 | EPR computational performance on a PC with an Intel Pentium 4 2600 MHz processor

$Y_i - Y$	Number of a_j	Generation number	Time in seconds
$\varepsilon = 0$	3	255	11.38
	4	34	2.64
	5	38	3.81
ε_1	3	25	2.40
	4	111	8.40
	5	32	3.24
ε_2	3	36	3.25
	4	66	5.93
	5	37	6.15
ε_3	3	66	3.67
	4	57	5.24
	5	86	7.59
ε_4	3	28	1.24
	6	39	4.58
	7	52	5.94
ε_5	3	201	9.42
	4	26	2.20
	13	196	35.96

SOME APPLICATIONS

Interpolation of Colebrook–White formula

The objective of the example application was to find an explicit polynomial function for the friction factor f for Reynolds number $\mathbf{R}$ ranging from 100,000 to 1000,000 and relative roughness $\mathbf{K}$ from 0.001 to 0.01 as in Davidson *et al.* (1999). The data set consists of a two-dimensional grid of 100 data points, created from ten Reynolds values selected in equal increments of 100,000 on the interval of 100 000 to 1000,000, and 10 relative roughness values selected in equal increments of 0.001 on the interval of 0.001 to 0.01. The target friction values for the 100 points are values obtained using the Colebrook–White formula (Colebrook & White 1937):

$$\frac{1}{\sqrt{f}} = -2\log_{10}\left(\frac{2.51}{\mathbf{R}\sqrt{f}} + \frac{\mathbf{K}}{3.71}\right) \tag{22}$$

Haaland (1983) provides Equation (23) as an explicit approximation to the Colebrook–White formula above:

$$\frac{1}{\sqrt{f}} = -1.8\log_{10}\left[\frac{6.9}{\mathbf{R}} + \left(\frac{\mathbf{K}}{3.7}\right)^{1.11}\right]. \tag{23}$$

In comparison to Equation (23) the fourteen-term polynomial (Davidson *et al.* 1999) is 33% more accurate over the selected region. Davidson *et al.* (1999) used a scaling of f $\mathbf{K}$ and $\mathbf{R}$. The largest absolute value of error of Davidson's fourteen-term polynomial over 100 selected data points was 1.94×10^{-4} while the absolute value of errors for Equation (23) was 2.08×10^{-4}. The sum of absolute values of errors for the 100 data points was 2.967×10^{-3} for the fourteen-term polynomial and 4.55×10^{-3} for Equation (23).

EPR searched for 12 potential formulae, using the parameters as shown in Table 6. Table 7 shows SSE values for increasing m, showing that the best result is obtained with an eleven-terms polynomial plus bias:

$$\begin{aligned} f = {} & 6.1661 \cdot 10^{-7}\mathbf{K}^{-1} - 9.5754 \cdot 10^{-5}\mathbf{K}^{-0.5} + 0.2522\mathbf{K}^{0.5} \\ & -0.14174\mathbf{K} + 0.79627\mathbf{K}^{1.5} - 5.6256 \cdot 10^{2}\mathbf{R}^{-1.5} \\ & -9.991 \cdot 10^{-3}\mathbf{R}^{-0.5} - 0.92716\mathbf{K}^{-1.5}\mathbf{R}^{-1.5} \\ & +0.2055\mathbf{K}^{-1}\mathbf{R}^{-1} + 4.6695\mathbf{K}^{-0.5}\mathbf{R}^{-1} \\ & +3.980 \cdot 10^{3}\mathbf{K}^{2}\mathbf{R}^{-1} + 0.014171 \end{aligned} \tag{24}$$

Figure 3(a) shows that the largest absolute value of errors for Equation (24) was 1.12×10^{-6}, while the sum of absolute values of errors for the 100 data points was 1.23×10^{-5}.

The EPR result was much better than that of R-BGP both from an interpolation and from a computing performance point of view, as can be seen from Table 8. Moreover, EPR was easier to use from a numerical standpoint because it was not necessary to scale the input–output variables as in Davidson *et al.* (1999).

A search for a formula like Equation (23) by means of EPR obtained a solution in 8 generations of GA (about 0.5 s):

Table 6 | Parameters of EPR

	Davidson test equation (24)	Haaland test equation (25)
Search exponents	[−2 −1.5 −1 −.5 0 .5 1 1.5 2]	[−2 −1 0 1 2]
EPR type	Case 0 of equations (19) without *function*	Case 3 of equations (19) with $\mathbf{Y} = 1/\left(-2\sqrt{f}\right)$
Number of a_j	[1:12]	3
Bias	Yes	No
Scale parameters	No scaling of f and **K**,**R**	No scaling of f and **K**,**R**
O.F. for model selection	Control of a_j	Control of a_j
GA population size	40	40
Crossover probability rate	0.4	0.4
Mutation probability rate	0.1	0.1

$$\frac{1}{\sqrt{f}} = -2\log_{10}\left[\frac{16.84998}{\mathbf{R}} + \frac{\mathbf{K}}{3.70939} - 447.74053\frac{\mathbf{K}}{\mathbf{R}}\right] \quad (25)$$

Table 7 | EPR results in search for 12 potential formulae

Search terms (a_j + bias)	Actual terms (a_j + bias)	SSE
2	2	1.08×10^{-7}
3	3	9.19×10^{-9}
4	4	5.27×10^{-10}
5	5	9.26×10^{-11}
6	5	2.93×10^{-11}
7	5	9.05×10^{-12}
8	8	1.04×10^{-12}
9	6	2.58×10^{-12}
10	10	1.21×10^{-13}
11	11	6.49×10^{-14}
12	11	8.37×10^{-13}
13	12	3.90×10^{-14}

whose largest absolute value of errors was 7.216×10^{-5}, while the sum of absolute values of errors for the 100 data points was 8.452×10^{-4}. Also in this case, EPR is able to interpolate the Colebrook–White formula by means of three parameters whilst preserving von Kàrmàn's universal constant related to coefficient 2 of $\log_{10}$. The computational performance and the interpolation accuracy were, again, very good (Cunge 2003).

Knowledge discovery based on data: friction factor of corrugated pipes

GP was used to determine the Chèzy resistance coefficient for full circular corrugated channels (Giustolisi 2004). Three corrugated plastic pipes were experimentally studied in order to generate test data. Experiments were undertaken to measure hydraulic parameters of the open-channel flow for some slopes, from 3.49–17.37% (2–10°), in order to discover the dependence of channel resistance coefficient – or friction factor – when wake-interference flow occurs (Giustolisi *et al.* 2003). Giustolisi (2004) presented a very parsimonious formula obtained by GP:

$$Cad = \frac{C}{\sqrt{g}} = \sqrt{\frac{8}{f}} = \frac{1}{0.534}\ln\left(\frac{2R}{hs}\right) + \left(1.0973\ln\frac{hs^{\frac{ds}{hs}}}{RS} + 20.6863\right) \quad (26)$$

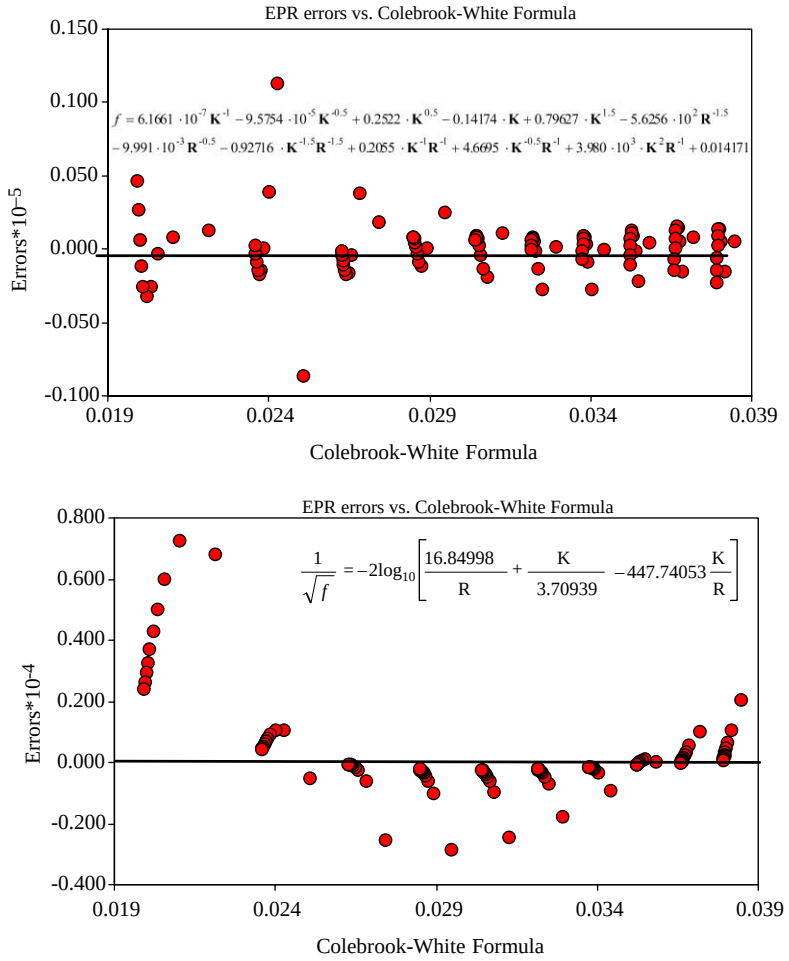

Figure 3 | (a) Errors of Colebrook–White formula interpolated by EPR, Equation (24). (b) Errors of Colebrook–White formula interpolated by EPR, Equation (25).

where *Cad* is the dimensionless resistance coefficient of Chèzy, R is the hydraulic radius of flow, S is the pipe slope, d_S is the longitudinal spacing of the wall-roughness elements and h_S is the height of the wall-roughness elements. The formula in Equation (26) fits very well the whole set of data, see Table 10 and Figure 4. This formula was obtained by 'physical post-refinement' (Giustolisi 2004) of the symbolic result of the GP strategy.

Equation (26) appears to explain better the role of roughness in the Chèzy resistance coefficient for corrugated channels with respect to its traditional expression for rough channels (Morris 1959; Giustolisi 2004). Finally, Giustolisi's work stressed the fact that the GP hypothesis can be easily manipulated by means of 'human' physical insight (Keijzer & Babovic 2002). Therefore, GP should be considered more than a simple data-driven technique, especially when it is used to perform scientific knowledge discovery (Giustolisi 2004).

A formula was sought by means of EPR with natural logarithm as shown in Table 9. In this instance, the number of adjusting parameters was fixed to 4 (including bias). The result was the expression

$$Cad = \frac{1}{0.6489}\ln\left(\frac{2R}{ds}\right) + \left\{0.39728\left(\frac{ds}{hs}\right)^{2}\ln\left(\frac{hs}{ds}\right) + \frac{4.0508 \cdot 10^{-3}S}{ds^{2}}\ln\left(\frac{S \cdot ds}{R}\right) + 18.146\right\} \quad (27)$$

Table 10 and Figure 4 show that the formula of Equation (27) fits the data from the training set extremely well, while

Table 8 | EPR velocity performances on PC with an Intel Pentium 4 2600 MHz processor

Number of a_j	Generation number	Time in seconds
1	5	0.2628
2	16	0.8832
3	19	1.3056
4	16	1.3776
5	17	1.4364
6	186	17.688
7	496	52.092
8	465	51.864
9	703	84.936
10	256	32.88
11	821	104.81
12	829	111.59

the statistical performance on the test set of 'unseen data' demonstrates that over-fitting did not occur. Indeed, the expression in Equation (27) has few parameters and an understandable structure from a physical point of view.

Table 9 | Parameters of EPR

	Values
Search exponents	[−2 −1 0 1 2]
EPR type	Case 2 of Equations (19), function = natural logarithm
Number of a_j	3
Bias	Yes
Scale parameters	No scaling of *Cad* and *hs, ds, R, S*
O.F. for model selection	SSE i.e. complexity penalisation with $\alpha = 0$
GA population size	40
Crossover probability rate	0.4
Mutation probability rate	0.1

It should be emphasised that EPR performed far better than the classical GP algorithm. Moreover, both Equations (26) and (27) allow some physical interpretations of the roughness effect in corrugated pipes: for example, regarding the value of von Kàrmàn's universal turbulence constant (which is higher than 0.4, i.e. the value normally found in the literature), roughness index (*hs*/d*s*) and relative

Table 10 | EPR results vs. GP results

	Training set		Test set	
	AVG	CoD	AVG	CoD
Eq. (26) – GP	2.17%	0.9224		
Eq. (27) – EPR	1.24%	0.9709	1.29%	0.9728

$$\mathrm{AVG} = 100^{\frac{\sum_{\#\,of\,data}\sqrt{(1 - Cad_{computed}/Cad_{experimental})^2}}{\#\,of\,data}}$$

$$\mathrm{CoD} = 1 - \frac{\#\,of\,data - 1}{\#\,of\,data} \frac{\sum_{\#\,of\,data}(Cad_{computed} - Cad_{experimental})^2}{\sum_{\#\,of\,data}(Cad_{experimental} - mean(Cad_{experimental}))^2}$$

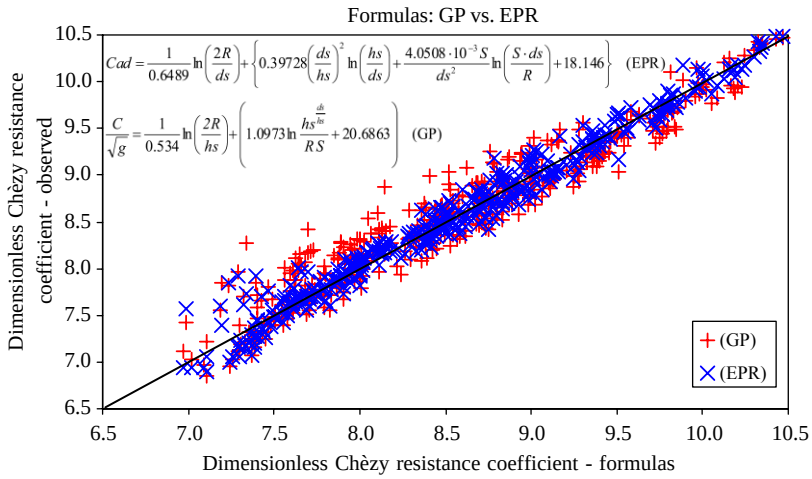

Figure 4 | EPR vs. GP formulae (the whole set of experimental data is graphed).

roughness (R/ds). See Morris (1959), Giustolisi *et al.* (2003) and Giustolisi (2004) for further details about the hydraulics of the problem. This case also showed that the performance of the EPR search was very fast, taking about 384 generations in a time of about 25 s on PC with an Intel Pentium 4 2600 MHz processor.

CONCLUSIONS

In this paper a new methodology for symbolic modelling of environmental phenomena is presented. The methodology, named Evolutionary Polynomial Regression (EPR), is based on both numerical and symbolic regression. EPR uses a *genetic algorithm* to find the form of polynomial expressions and *least squares optimisation* to find the values for the constants in the expressions. The incorporation of least squares optimisation within symbolic regression enables fast and effective model building.

In order to test the ability of EPR to discover the mathematical structure of a phenomenon, an *ad hoc* artificial formula was used. The experiments conducted show that EPR is able to identify both the symbolic structure and constants from data which were generated by this formula corrupted with progressively increasing Gaussian noise. The efficiency of the algorithm is also high, which is attributed to EPR's evolutionary approach that transforms the general evolutionary (and often slow) search of GP into a very simplified evolutionary search for exponents. This is then performed by a simpler GA which is computationally more efficient.

Furthermore, the performance of EPR has been compared with Rule-Based Genetic Programming (Davidson *et al.* 1999) at interpolating the Colebrook–White formula and with Dimensionally Aware Genetic Programming (Babovic & Keijzer 2000) for performing scientific knowledge discovery from data for the resistance coefficient in corrugated pipes. In both cases EPR identified more accurate formulae than those found by the Rule-Based or Dimensionally-Aware GP algorithms. Moreover, the formulae obtained allow some physical interpretations of the phenomena studied: for example, regarding the value of von Kàrmàn's universal constant of turbulence.

The methodology described in this paper is implemented in a software program developed within the MATLAB environment. The program fills a need in the hydroinformatics research community for a freely available tool for developing and testing data-based models. The software is freely available for research and evaluation purposes from the authors or via its web site: http://www.poliba.it/Taranto/software/hydroinformatics/index.htm.

ACKNOWLEDGEMENTS

This work was supported by the UK Engineering and Physical Sciences Research Council, grant AF/000964. The authors are also grateful for the insights of Dr James

Davidson, whose work inspired the development presented in the paper, and to Mark Morley who commented on the first draft of the paper.

REFERENCES

Babovic, V. & Keijzer, M. 2000 Genetic programming as a model induction engine. *Journal of Hydroinformatics* **2** (1), 35–60.

Colebrook, C. F. & White, C. M. 1937 Experiments with fluid friction in roughened pipes. *Proc. R. Soc. London, Ser. A* **161**, 367–381.

Cunge, J. A. 2003 Of data and models. *Journal of Hydroinformatics* **5** (2), 75–98.

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

Davidson, J. W., Savic, D. A. & Walters, G. A. 2000 Approximators for the Colebrook-White formula obtained through a hybrid regression method. In *Computational Methods in Water Resources Vol 2 Computational Methods, Surface Water Systems and Hydrology* (ed. L. R. Bentley, J. F. Sykes, C. A. Brebbia, W. G. Gray & G. F. Pinder), pp. 983–989. Balkema, Rotterdam.

De Jong, E. D. & Pollack, J. B. 2003 Multi-objective methods for tree size control. *Genetic Programming and Evolvable Machines* **4** (3), 211–233.

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

Giustolisi, O. 2000 Input-output dynamic neural networks simulating inflow-outflow phenomena in a urban hydrological basin. *Journal of Hydroinformatics* **2** (5), 269–279.

Giustolisi, O. 2004 Using genetic programming to determine Chèzy resistance coefficient in corrugated channels. *Journal of Hydroinformatics* **3** (6), 157–173.

Giustolisi, O., Doglioni, A. & Laucelli, D. 2003 Experimental evaluation of resistance coefficient formula for corrugated channels. In *Proceedings of the XXX IAHR International Conference in Thessaloniki* (eds J. Ganoulis & P. Prinos), pp. 503–513. IAHR, Thessaloniki, Greece. Theme B.

Giustolisi, O. & Laucelli, D. 2005 Increasing generalisation of input-output artificial neural networks in rainfall-runoff modelling. *Hydrological Sciences Journal* **3** (50), 439–457.

Goldberg, D. E. 1989 *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, New York.

Golub, G. H. & Van Loan, C. F. 1993 *Matrix Computations*. The Johns Hopkins University Press, London.

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

Haykin, S. 1999 *Networks: A Comprehensive Foundation*, 2nd edn. Prentice-Hall, Englewood Cliffs, NJ.

Holland, J. 1975 *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Keijzer, M. & Babovic, V. 2002 Declarative and preferential bias in GP-based scientific discovery. *Genetic Programming and Evolvable Machines* **3** (1), 41–79.

Koza, J. R. 1992 *Genetic Programming: On the Programming of Computers by Natural Selection*. MIT Press, Cambridge, MA.

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

Morris, H. M., Jr 1959 Design methods for flow in rough conduits. *J. Hydrol. Division, Proceedings ASCE* **85** (HY 7), 43–62.

Soule, T. & Foster, J. A. 1999 Effect of code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation* **4** (6), 293–309.

Spears, W. M. & De Jong, K. A. 1991 An analysis of multi-point crossover. In *Foundations of Genetic Algorithms* (ed. J. E. Rawlins), pp. 301–315. Morgan Kaufmann, San Mateo, CA, USA.

Tikhonov, A. N. 1963 Solution of incorrectly formulated problems and the regularization method. *Doklady Akademii Nauk SSSR* **151**, 501–504.

Zhang, B. & Muhlenbein, H. 1995 Balancing accuracy and parsimony in genetic programming. *Evolutionary Computation* **3** (1), 17–38.