Soft computing-based nonlinear fusion algorithms for describing non-Darcy flow in porous media

Algorithmes non-linéaires de fusion en soft computing pour décrire les écoulements de type non-Darcy dans les milieux poreux

A.-R. NAZEMI, Graduate Student, Department of Civil Engineering, Ferdowsi University of Mashhad, PO Box 91775-1111, Mashhad, Iran. E-mail: ali_r_nazemi@yahoo.com (Present address: Department of Civil Engineering, The University of Birmingham, Birmingham B15 2TT, UK)

S.M. HOSSEINI, Assistant Professor, Department of Civil Engineering, Ferdowsi University of Mashhad, PO Box 91775-1111, Mashhad, Iran. Tel.: +98 511 8629533; fax: +98 511 8629541; e-mails: shossein@ferdowsi.um.ac.ir; shossein2@yahoo.com; (author for correspondence)

M.-R. AKBARZADEH-T, Assistant Professor, Department of Electrical Engineering, Ferdowsi University of Mashhad, PO Box 91775-1111, Mashhad, Iran. E-mail: akbarzadeh@kiaee.org

ABSTRACT
By increasing the velocity of flow in coarse grain materials, local turbulences are often imposed to the flow. As a result, the flow regime through rockfill structures deviates significantly from linear Darcy law; and nonlinear or non-Darcy flow equations will be applicable. Even though the structures of these nonlinear equations have some physical justifications, empirical studies are still necessary to estimate the parameters of these equations amid a great deal of uncertainty inherent to this estimation process. Consequently, none of the current empirical equations alone seem to be able to model the flow process exactly. In recent years, soft computing, in contrast to classical modeling techniques, has been advocated as a hybrid approach to intelligent paradigms such as neural networks, fuzzy logic, and neuro-fuzzy systems that aim to handle the uncertainties and vagueness in such systems. In this paper, we investigate several soft computing paradigms to combine three of the most commonly validated and utilized empirical solutions in the current literature. In this way, the estimates from the three empirical equations drive a soft computing-based fusion algorithm. The results show that soft computing-based approaches provide a powerful paradigm with a strong ability to model reality. Specifically, this paper concludes that cascade correlation neural networks provide the best fusion algorithm with the highest accuracy among the considered conventional alternatives as well as several other soft computing paradigms.

RÉSUMÉ
Dans les matériaux grossiers, des turbulences locales sont souvent imposées à l’écoulement lorsque la vitesse augmente. De ce fait, le régime d’écoulement dans les enrochements s’écarte de manière significative de la loi linéaire de Darcy ; et les équations non-linéaires, ou non-Darcy, de l’écoulement sont applicables. Mais, bien que les structures de ces équations non-linéaires aient certaines justifications physiques, des études empiriques sont encore nécessaires pour estimer les paramètres de ces équations avec un grand nombre d’incertitudes inhérentes au processus d’évaluation. En conséquence, aucunes des équations empiriques courantes ne semblent pouvoir modéliser à elles seules le processus d’écoulement avec exactitude. Ces dernières années, le soft computing, contrairement aux techniques classiques de modélisation, a été préconisé comme approche hybride des paradigmes intelligents tels que les réseaux neuronaux, la logique floue, et les systèmes neuro-flous qui visent à manipuler les incertitudes et les imprécisions dans de tels systèmes. Dans cet article, nous étudions plusieurs paradigmes en soft computing pour combiner trois des solutions empiriques les plus validées et utilisées dans la littérature. De cette façon, les évaluations des trois équations empiriques pilotent un algorithme de fusion en soft computing. Les résultats prouvent que les approches basées sur le soft computing fournissent un paradigme puissant avec une forte capacité à modéliser la réalité. En particulier, cet article conclut que les réseaux neuronaux corrélés en cascade fournissent le meilleur algorithme de fusion le plus exact parmi les alternatives conventionnelles considérées et parmi plusieurs autres paradigmes de soft computing.

Keywords: Porous media, non-Darcy flow, fusion algorithm, soft computing.

1 Introduction

In recent years, there has been notable increase in application of rockfill and coarse grain materials for hydraulic structures such as rockfill dams, breakwaters, and gabions (Hansen et al., 1995; Hosseini, 2000). The availability of construction materials, increased knowledge about the behavior of rockfills, and their short construction duration are the main reasons for this development. The main feature of rockfill structures is that, because of increased local turbulences, flow through them is not
accurately described by Darcy linear flow; instead non-Darcy nonlinear flow is the dominant phenomenon (McCorquodale et al., 1978; Stephenson, 1979; Martins, 1990; Joy, 1991a, b; Hansen et al., 1995; Hosseini, 2000).

Furthermore, the hydraulic behavior of rockfill depends on many parameters such as size and size distribution, porosity, orientation, shape and roughness of the grains (Hansen et al., 1995). The overall hydraulic effect of these variables is hard to quantify and therefore uncertainty will be an inherent element in estimation of hydraulic parameters. A commonly used method to obtain the hydraulic parameters is the use of empirical relations, based on the physical features of the media. Although the research in this area has been extensive, there is no general agreement on one specific equation. Consequently, as reported by previous researchers, the empirical equations are biased and therefore produce underestimated and/or overestimated results (Joy, 1991b; Hansen et al., 1995; Hosseini, 2000). Additionally, empirical studies do not encompass all situations and therefore do not always reflect reality.

Considering the ever increasing computational abilities and introduction of new methods of deduction, combination and modeling, it is now reasonable to use different equations simultaneously and then combine them together (Shamseldin et al., 1997; Xiong et al., 2001). Averaging is the most obvious and simplest method for combination. An improved but more complex method is weighted averaging or linear regression. However, these methods may produce inadequate results when interrelations among simultaneous equations are nonlinear. Hence, nonlinear approaches may be deemed superior. But, traditional nonlinear regression often leads to time consuming and complex mathematical programming. These methods are also very sensitive, and a small change in parameters may change the regression line violently (Jang et al., 1997).

As an alternative to the above conventional approaches, soft computing and hybrid intelligent paradigms are proposed to handle uncertainty and ambiguity in natural systems through the past decade. At the core of soft computing frameworks for intelligence are neural computation and fuzzy logic (Jang et al., 1997; Gorzalczany, 2002). On one hand neural networks are low-level (numeric) distributed processing units that learn and generalize nonlinear relations. On the other hand, fuzzy logic provides a high-level (linguistic) framework to model conceptual knowledge. Neuro-fuzzy systems combine these two paradigms taking advantage of their strength for a more powerful approach. There is an extensive literature on soft computing, and new applications are found everyday in solving real engineering problems.

In this study, several soft computing paradigms—specifically neural networks, fuzzy systems and a well-known neuro-fuzzy system, Adaptive Neuro Fuzzy Inference System (ANFIS)—are examined. Inputs to the above systems are the predictions of the three most accurate empirical equations in the literature to date, and by a proposed fusion algorithm, we expect to better represent the reality of non-Darcy flow in porous media. One may question the wisdom in above fusion approach, i.e. why not use the soft computing paradigm as a nonlinear framework to directly estimate the hydraulic gradient from grain parameters. However, one should also consider that it is not wise to forgo of previous approaches of significant merit. Fusion of these approaches, by some optimality criteria, can only expect to improve performance further. Additionally, even though soft computing techniques have admirable interpolation capability, equations that are based on physical laws of nature remain more reliable in terms of extrapolation. In the next section, previous studies in the area of non-Darcy flow equations are summarized and the three most accurate equations (as proposed by McCorquodale et al., Stephenson, and Adel) are selected based on the recommendations of Hosseini (2000). In Section 3, the data set of our case study will be introduced and the performance of the selected equations will be evaluated. In Section 4, the motivations and applications of soft computing methods are discussed and the fusion framework for this problem is presented. Then in Section 5, a neural network-based as a fusion framework is introduced and tested using different architectures and networks. All training and testing is done with laboratory data. In Sections 6 and 7, these steps are applied to the Mamdani fuzzy system and adaptive neuro-fuzzy inference system (ANFIS) model, respectively. In Section 8, the results are integrated and analyzed. Finally, this study’s conclusions are presented in Section 9.

2 Non-Darcy flow studies

In general, two distinct approaches for describing nonlinear flow conditions can be found in the literature. The first approach was proposed by Forchheimer who introduced the following one-dimensional quadratic equation as the constitutive relationship for nonlinear flow (Joy, 1991a; Hansen et al., 1995; Hosseini, 2000):

\[ i = aV + bV^2 \]  
\[ (1) \]

where \( i \) is hydraulic gradient, \( V \) is bulk velocity, and \( a \) and \( b \) are constants. Another commonly used approach is the Missbach equation, which assumes the following exponential relation between hydraulic gradient and bulk velocity (Martins, 1990; Hansen et al., 1995; Hosseini, 2000).

\[ i = lV^\lambda \]  
\[ (2) \]

where \( l \) and \( \lambda \) are constants which depend on media and fluid properties. \( \lambda \) is a variable between 1 and 2, which changes from case to case.

Although the above equations have some theoretical justifications, constants \( a \) and \( b \) in the Forchheimer equation, \( l \) and \( \lambda \) in the Missbach equation) relate to media characteristics and are generally estimated by empirical equations resulting from experimental studies. Many proposed empirical equations can be found in the literature (McCorquodale et al., 1978, Stephenson, 1979; Martins, 1990). Hansen et al. (1995) present a comprehensive review of different nonlinear empirical equations, which are widely used for estimating hydraulic properties in non-Darcy flow conditions. Also, there are several other studies that evaluate the above nonlinear flow equations (Joy, 1991a; Hansen et al., 1995; Hosseini, 2000). In general, it can be concluded from these
studies that Forchheimer-based equations are superior and more accurate than Missbach type equations. Also, statistical analysis by Hosseini (2000) revealed that the most accurate models are as follows:

- **McCorquodale et al. equation** (McCorquodale et al., 1978)

  For low Reynolds numbers, i.e. \( R_p = Vm/\nu n \leq 500 \) or \( R_w = Vm/\nu n \leq 125 \):
  \[
  i = \frac{4.6V}{g m n^4} + \frac{0.79}{g n^{1/2} m} V^2
  \] (3a)

  For high Reynolds number, i.e. \( R_p > 500 \) or \( R_w > 125 \):
  \[
  i = \frac{70V}{g m n^4} + \frac{0.27(1 + (f_ε/f_o))}{g n^{1/2} m} V^2
  \] (3b)

- **Stephenson equation** (Stephenson, 1979)
  \[
  i = \frac{800V}{g n d^2} + \frac{k}{n^2 g d} V^2
  \] (4)

- **Adel equation** (Hosseini, 2000)
  \[
  i = \frac{160V(1-n)^2}{g n^3 d_{15}^2} + \frac{2.2}{g n^2 d_{15}} V^2
  \] (5)

In Eqs (3) – (5), \( i \) is hydraulic gradient, \( V \) is bulk velocity, \( n \) is porosity, \( \nu \) is kinematic viscosity, \( g \) is gravitational acceleration, \( d \) is harmonic mean of particle sizes, \( R_p \) is pore Reynolds number, \( R_w \) is Ward Reynolds number, \( k \) is the friction factor in the turbulent region of flow (=1 for smooth polished marble, =2 for semi-rounded stones and =4 for angular rocks) and \( d_{15} \) is the grain size which 15% of the particles, by weight, are smaller than. Also, \( f_ε \) is the Darcy friction factor for rock and permeameter, with the wall effect removed from the data; and \( f_o \) is the Darcy–Weisbach friction factor for a hydraulically smooth surface with the same Reynolds number (obtained from a Moody diagram for pipe flow). Hansen et al. (1995) state that ratio \( f_ε/f_o \approx 1.5 \) for crushed rock. According to an example given by McCorquodale et al. (1978), river gravel and crushed rock ratios can be set at 1.15 and 1.75, respectively. And \( m' \) is effective pore hydraulic radius that can be calculated by:

\[
m' = \frac{\text{Total effective volume of voids}}{\text{Total effective surface area}} = \frac{n}{6 f_0 \int_0^1 (1 - n) (\alpha d f/D(f)) + c_w/R_x}
\] (6)

where \( f \) is accumulated frequency of the granular material, \( D(f) \) is grain size finer than \( f \) by weight, \( R_x \) is hydraulic radius of parameter, \( \alpha \) is the ratio of the surface area of the particle to the surface area of the sphere with the same volume and \( c_w \) is the empirical coefficient to account for wall effect (=−0.5). In contrast, \( m \) is the pore hydraulic radius, the same as Eq. (6) with the wall effect removed. The conclusions of Hosseini (2000) in evaluation of several empirical relations led to the selection of above Eqs (3)–(5) and are used for the proposed fusion algorithms in this paper.

### 3 Data set used for this study

The data set used for this study was produced by Joy (1991a, b). He constructed a simple experimental device to collect a consistent set of \( i \) versus \( V \) data for 23 different materials. The device consisted of a vertical cylinder, 750 mm long and 152 mm in diameter, containing the media. Head losses were determined using five piezometer taps on the cylinder and discharge was determined using either an orifice for large flows or by timed weighting for smaller flows. Twenty-three samples of different coarse media were tested. Mean \( (d_{50}) \) material sizes ranged from 3 to 31 mm while hydrodynamic conditions were all outside the laminar range with Reynolds numbers in the range of 50–600. The media was homogenous; the flow was steady and hydraulic gradient was from 0.012 to 1.5 while velocity ranges from 0.159 to 0.014 m/s. Totally 483 bulk velocity-hydraulic gradient data were observed for all 23 coarse materials. Physical properties related to three selected empirical equations are extracted from Joy (1991a, b) by following the recommendations made by the developer of the equations to calculate or estimate their physical properties. Therefore, this data set can be used to compare predicted hydraulic gradients with the observed values in this study.

The physical properties associated with each empirical equation are applied to the equations to find the hydraulic gradients for all velocity values corresponding to permeameter tests. Figure 1 shows scatter plots of \( i \)(simulated) versus \( i \)(observed) for the McCorquodale et al., Stephenson, and Adel equations, respectively. It can be seen that these plots scatter widely especially

![Figure 1](image-url)
when the velocity increases. Hence, there is a great deal of uncertainty in the equations abilities to represent actual hydraulic gradient.

4 Fusion algorithms: motivations and applications

As shown in Fig. 1, none of the above three models provide exact results. These equations may have subjective features that should be determined from engineering judgment; and the process may have different sources of uncertainty that are not fully reflected in the empirical equations. It should be considered that all empirical equations have been developed based on laboratory condition so the real process in nature may deviate significantly from these equations. As noted in literature, every equation has its “plateau of maximum efficiency,” which falls substantially short of perfection. And even when they arrive at correct numbers (i.e. achieved a good fit), it “can be right for all the wrong reasons” (Xiong et al., 2001).

Assuming that each single empirical equation is best only for a certain range of conditions, it can be reasonably expected that the hydraulic gradient estimates that are obtained by fusion of the results from a number of different empirical equations through an appropriate weighting procedure to be more comprehensive and accurate in representing the relationship between hydraulic gradient and bulk velocity than any single equation or relation. In this model, instead of relying on one individual nonlinear empirical equation, or even switching from one to another, the hydraulic gradients are generated simultaneously from a number of different nonlinear equations and then the results are combined in order to improve performance. The essence of the concept is that each equation output captures certain important aspects of the information available about the process being modeled, thereby providing a source of information, which may be different from that of other equations (Shamseldin et al., 1997; Xiong et al., 2001). The addition of the above fusion algorithm, however, presents additional parameters and therefore added complexity. This aspect of modeling process is later addressed through model selection criteria like Akaike information criterion (AIC) and Bayesian information criterion (BIC) in Section 8. These two measures of information penalize the model for having more parameters and therefore tend to result in more parsimonious models.

Mathematically, if we have $p$ different empirical equations for estimating the hydraulic gradient for a certain velocity, this fusion function is generally expressed as (Xiong et al., 2001):

$$ i_c = F(\hat{i}_1, \hat{i}_2, \ldots, \hat{i}_p) $$

where $\hat{i}_1, \hat{i}_2, \ldots, \hat{i}_p$ are the hydraulic gradient of $p$ different equations, respectively, for a specified bulk velocity; $i_c$ is their combined result and $F(\cdot)$ is the fusion function. A block diagram of the process of combining different empirical equations is shown in Fig. 2 (Xiong et al., 2001).

Xiong et al. (2001) provided a review about the background of the concept of combining forecasts obtained from different models or methods. Shamseldin et al. (1997) discussed several methods for combining the outputs of different rainfall-runoff models. In continuing their work, in 2001, they introduced a method for combining the results of different rainfall-runoff models, based on the concept of first order Takagi–Sugeno–Kang (TSK) fuzzy systems (Xiong et al., 2001). Their study motivated the basic idea for combining different empirical equations in order to better estimate nonlinear flow in porous media.

The most obvious method for combining the results of different empirical equations is simple averaging. Another intuitive method, but more complex, is finding a linear relationship among the variables taking the results of empirical equations as independent variable and observed hydraulic gradients as dependent one, as follows:

$$ i_c = a\hat{i}_{Adel} + b\hat{i}_{Stephenson} + c\hat{i}_{McCorquodale} $$

Figure 3 shows the results of the simple averaging method and the multiple linear regression equation. Numerical measures to compare averaging method and other more advanced combined methods are discussed and compared in results section (Section 8). However, observing Figs 1 and 3, and one may conclude that the traditional combination methods are not successful since the funnel-shaped trend is still dominant. So, the failure of these conventional methods in capturing the nonlinearity, which

![Figure 2](image-url)  
Figure 2  Schematic procedure for combining different empirical equations of non-Darcy flow.
are inherent in the system, is understood. Therefore, the application of a decision fusion algorithm, based on soft computing that can handle this nonlinearity, is plausible.

Soft computing is another methodology for combining the results of different models. Soft computing generally consists of fuzzy logic, neural networks and evolutionary algorithms for dealing with uncertainty and ambiguity in a phenomenon or its measurements (Jang et al., 1997; Gorzalczany, 2002). Fuzzy systems and neural networks have been shown to be universal approximators (Jang et al., 1997; Xiong et al., 2001). Applications of soft computing are numerous and cover diverse fields. Applications can be found in different engineering fields such as control, decision-making, and robotics. Recently, applications of soft computing are reported in the field of water resources engineering (Bardossy, 1990, 1996; Hsu et al., 1995; Nazemi et al., 2002). In the following sections we will evaluate neural networks, fuzzy systems and a well-known neuro-fuzzy system ANFIS as combination methods for dealing with this problem.

5 Neural networks as a fusion framework

Inspired by biological nervous systems, many researchers, especially brain modelers, have explored the application of artificial neural networks (ANN) to information processing (Jang et al., 1997; Demuth and Beale, 2000). Adaptation is one of the main characteristics of artificial neurons, which means that the outputs of these nodes depend on modifiable parameters pertaining to these nodes. Because of universal approximation capability, and their ability to represent highly nonlinear relationships, ANN shows greater performance in comparison with conventional approaches based on regression analysis (Hsu et al., 1995; Jang et al., 1997; Furundzic, 1998). Neuro-based models require three general steps. In the first step, an appropriate architecture must be selected. This is often done using modeler expertise. After selecting the ANN architecture, the network must be trained. This step is the most important modeling step since weights and biases are calibrated during training. Finally, the trained network must be validated with an independent data set. In this step the generalization capability of trained network is examined. If the number of data is sufficient, validation can be divided in two distinct steps: checking and testing. If the network is validated then it can be used for modeling; otherwise, we must return to the first step and select a different architecture. Among various neural structures, the feedforward backpropagation network, the cascade correlation network, and radial basis function networks are the most common architectures and are considered in this study.

5.1 Feedforward backpropagation networks

Feedforward backpropagation networks or multi-layer perceptron (MLP) with the backpropagation learning method are the most popular neural architecture in engineering (Jang et al., 1997; Demuth and Beale, 2000). It consists of several layers (at least three) and each layer has several neurons. Each layer has a weight matrix, a bias vector and an output vector. For each neuron its effective incoming signal (\(S_j\)) to node \(j\) is the weighted sum of all the incoming signals (Hsu et al., 1995):

\[
S_j = \sum_{i=0}^{m_0} w_{ji} x_i
\]  

where \(x_0\) and \(w_{ji}\) are called the bias (\(x_0 = 1.0\)) and the bias weights, respectively. The effective incoming signal, \(S_j\), is passed through a nonlinear activation function to produce the outgoing signal (\(y_j\)) of the node. The most commonly used activation function is the differentiable sigmoid function. The functionality of sigmoid function is described as below (Hsu et al., 1995; Jang et al., 1997; Gorzalczany, 2002):

\[
y_j = f(S_j) = \frac{1}{1 + \exp(-S_j)}
\]  

Multilayer networks are quite powerful; a simple three-layer network, where the hidden layer is sigmoid and the output layer is linear, can be trained to approximate any functional relationship (with finite number of discontinuities) with any desirable degree of accuracy (Hsu et al., 1995; Jang et al., 1997). Back-propagation is the most commonly used method for training. The backpropagation training method is based on gradient-based optimization. In brief, the modified version of backpropagation can be expressed by the following equation (6):

\[
u_{ij}^{k}(t + 1) = u_{ij}^{k}(t) + 2\eta_\varepsilon^k(t) f'(S_i^k(t)) x_j^k(t) + \alpha(u_{ij}^{k}(t) - u_{ij}^{k}(t - 1))
\]
where $w_{ij}^k$ denotes the weight of the $i$th neuron in the $k$th layer, referring to the $j$th input, $e_i^j$ is the output of the linear part of the $i$th neuron in the $k$th layer, $f(\cdot)$ is the activation function, $t = 1, 2, \ldots$ is a time instant (indicating a learning step), $\eta$ is learning rate (a small positive value), and $\alpha$ is momentum term. There are several types of optimization schemes that can be used for updating weights and biases in backpropagation method (Demuth and Beale, 2000). Trained backpropagation networks tend to give reasonable answers when presented with inputs that they have never seen (Jang et al., 1997; Demuth and Beale, 2000; Gorzalczany, 2002). However because the backpropagation training is based on gradient programming, it may lead to a local rather than a global error minimum (Hsu et al., 1995; Jang et al., 1997). If this local error minimum is not satisfactory, a network with more neurons may be required.

Combining the results of Adel, Stephenson and McCorquodale et al. equations with a neural network requires data to be randomly divided into three distinct data sets: training, checking, and testing data sets. The training data set is used first to train the fusion algorithms, the checking data set is then used to check the generalization performance of training. Finally, the testing data set is used to evaluate prediction performance of the fusion algorithm on unseen data set. Training data set contains 350 data pairs, the checking data set contains 50 data pairs, and testing data set consists of 83 data pairs. Each data pair contains results of the Adel, Stephenson and McCorquodale et al. equations as inputs and observed hydraulic gradient as output. Then data quantities are divided by their maximum value in the training data set to create a normalized data set. The MATLAB® neural network toolbox was used for simulation. Figure 4 shows the performance of multilayer network with three layers in which input layer has three neurons, hidden layer has four neurons and output layer has one neuron. The network is trained in 1000 epochs (number of training cycles). The sum of squared error function is the objective function used for optimization. Tangent sigmoid functions were selected as activation functions for the neurons. The network was trained with Levenberg–Marquardt method. The results demonstrate a significantly higher correlation between observed and simulated values, as a result of the fusion process, when compared with each of the three methods individually, as in Fig. 1. Although neural network paradigm is a black-box approach, some parameters should be identified by the modeler such as the optimization algorithm, number of hidden neurons, and training objective function. These parameters are chosen by running different scenarios and evaluating them by several statistical measures as will be introduced in Section 8.

5.2 Cascade forward backpropagation networks

Cascade feedforward is a layered network in which each layer only receives input from previous layers (Demuth and Beale, 2000). The first layer has weights coming from the inputs, and the last layer is the network output. This type of the network can be viewed as a generalization of the feedforward backpropagation network. This network combines the idea of incremental architecture and learning in its training procedure (Karunanithi et al., 1994). In brief, training starts with a minimal network consisting of input and output layers as in a linear model. If the training algorithm can no longer reduce the residual error, then it stops this phase of training, and enters the next phase for training with a potential hidden unit. The potential hidden unit has associated connections (or weights) from the input layer and all preexisting hidden units but not toward the output layer. Weights associated with the potential hidden units are optimized by a gradient ascent method so as to maximize the correlation between its outputs and the residual error of the network. When a potential hidden unit is trained, weights associated with the output layer are frozen. Once a potential hidden unit is added to the network, it becomes a new hidden unit and its incoming weights are frozen for the rest of the training period. After installing a hidden unit the weights are updated for all connections, which directly feed the output layer. This dynamic expansion of the network continues until the problem is successfully learned. Thus the cascade correlation algorithm automatically constructs suitable network architecture for a given problem (Karunanithi et al., 1994). In the application of this paper, each data pair contains results of the Adel, Stephenson and McCorquodale et al. equations for specified velocity as inputs and observed hydraulic gradient as output, as previously explained. The MATLAB® neural network toolbox was used for this simulation. Figure 5 shows the output of cascade forward backpropagation network with three layers in which input layer has three neurons, hidden layer has four neurons and output layer has one neuron. The network is trained for 1000 epochs. Other parameters are set as the simulation previously reported in Section 5.1.
5.3 Radial basis function networks

Radial basis function neural networks can be implemented within the standard architecture of the feedforward backpropagation networks with an input layer, one hidden layer, and an output layer. The radial basis function can be written as a product of univariate functions, in the following form (Gorzalczany, 2002)

\[
G_j(X) = \prod_{i=1}^{n} \exp \left[ - \left( \frac{x_i - \mu_{ji}}{\sigma_j} \right)^2 \right]
\]  

where \( X = [x_1, \ldots, x_n]^T \) is the input vector, \( \mu_j = [\mu_{j1}, \ldots, \mu_{jn}]^T \) is the center vector and \( \sigma_j \) is the spread constant of \( j \)th unit. The final output of the network is the weighted sum of the output values of all units:

\[
y = \sum_{j=1}^{H} c_j G_j(X)
\]

where \( c_j \) is the connection weight between the unit \( j \) and the output unit and \( H \) is total number of units in hidden layer (receptive fields) (Jang et al., 1997). The purpose of the radial basis function network is to cover the input space with overlapping clusters (receptive fields). For an input vector \( X \) lying somewhere in the input space, the clusters with centers close to it will be appreciably activated.

Radial basis networks may require more neurons than the standard feedforward backpropagation networks, but often they can be designed in a fraction of time it takes to train standard backpropagation networks. Although the universality of the radial basis function has been proved (Jang et al., 1997; Demuth and Beale, 2000; Gorzalczany, 2002), however they perform best when adequate data is available (Demuth and Beale, 2000). The MATLAB® neural network toolbox was used for this simulation. Figure 6 shows performance of the radial basis function. In the network, the spread constant is selected as one. Although the result shows a higher correlation in comparison with individual equations and also traditional fusion methods, the radial basis network is very sensitive and a little change in spread function may lead to completely incoherent results.

6 Fuzzy systems as a fusion framework

Complex systems have imprecise, vaguely defined, or uncertain elements (Bardossy and Duckstein, 1995; Jang and Gulley, 1995; Jang et al., 1997; Wang, 1997). Fuzzy systems provide a robust method for dealing with these uncertainties linguistically. The core of fuzzy systems is fuzzy rules that provide appropriate means to translate natural statements into a computationally usable form (Bardossy and Duckstein, 1995). In contrast to black box modeling, fuzzy rules are very transparent as they are explicitly stated. These properties have helped fuzzy systems and fuzzy modeling to find many interesting and challenging applications in engineering. The principles and formulations of different fuzzy systems can be found in many textbooks (Bardossy and Duckstein, 1995; Jang et al., 1997; Wang, 1997).
There are two distinct approaches in fuzzy systems: Mamdani (logical) and Sugeno (functional) approaches. The difference between these approaches lies in the expression of rules’ consequents; in the Mamdani approach a rule’s consequent is also a fuzzy membership function but in Sugeno-based systems it is exchanged with a conventional mathematical function such as a polynomial (Jang et al., 1997; Nazemi et al., 2002). Fuzzy systems can be easily used as a fusion framework (Xiong et al., 2001). In this configuration, the output of each empirical equation is an input variable to the fuzzy fusion system. A fusion Mamdani fuzzy rule base consists of a set of rules in the following form:

If \( i \) is \( A_k \) and \( i \) is \( B_k \) and \( i \) is \( C_k \)
then \( i \) is \( D_l \) \hspace{1cm} (14)

where \( A_k \), \( B_k \), and \( C_k \) are the \( k \)th membership functions for \( i \)th input variable, \( i \) is \( A_k \) is the membership function of \( i \)th input variable, i.e., each empirical equation, in \( l \)th rule, \( \mu_{A_l} \) is the membership function of real input and \( \mu_{B_l} \) is the membership function of consequent in \( l \)th rule. \( Smorm \) and \( Tnorm \) are logical union and intersection operators, respectively.

The fuzzy inference mechanism consists of four steps: (1) fuzzification in which the numerical inputs are fuzzified, (2) intersection in which the degree of fulfillment of each rule is computed, (3) implication in which the output of each rule is calculated in term of fuzzy membership function and (4) aggregation in which the results of different rules are combined together and the overall output membership function is calculated with Eq. (15). The overall fuzzy output is then defuzzified and converted back to real numbers. There are many alternative methods for each of the above sections of a fuzzy system mostly chosen based on their mathematical properties and cohesion with expert knowledge. For example, from among many defuzzification operators in the literature, center average defuzzifier is found to be intuitive, continuous as well as computationally speedy (Wang, 1997):

\[
y^* = \frac{\sum_{l=1}^{M} y^*_l w_l}{\sum_{l=1}^{M} w_l}
\]

where \( y^*_l \) is the center of \( l \)th fuzzy rule and \( w_l \) is its height.

Although the structures of fuzzy systems are simple and intuitive, it must be noted that the number of rules increases exponentially as the number of input variables increases. Hence, user must carefully choose the number of input variables and

![Figure 7](image-url) (Left) antecedent and (Right) consequence membership functions obtained by weighted counting algorithm.

![Figure 8](image-url) Combined hydraulic gradient obtained by Mamdani fuzzy system versus observed hydraulic gradient: (a) training phase, (b) checking phase, (c) testing phase (based on normalized data).
their membership functions. In this application, four triangular membership functions are assumed for each input. For rule generation, the weighted counting algorithm (Bardossy and Duckstein, 1995; Shereuhara and Duckstein, 1996) is applied. Figure 7 shows antecedent and consequent membership functions obtained by the weighted counting algorithm. If we choose the Mamdani fuzzy system as the type of fuzzy inference model, algebraic product as AND (Tnorm) operator, algebraic sum as OR (Tnorm) operator, Min as implication operator, Max as aggregation operator and center of average as the defuzzification method, then the results for training, checking and testing phases are as in Fig. 8. Different simulations reveal that several other combinations of Snorms, Tnorms and implication operators provided inferior performance. MATLAB® fuzzy logic toolbox was used for simulation (Jang and Gulley, 1995).

7 Neuro-fuzzy systems as a fusion framework

A hybrid fuzzy logic and neural systems paradigm may provide a unique framework where the strength of each paradigm compensates for the weakness of the other. Fuzzy logic is known for its ability to model human knowledge qualitatively, whereas neural networks can be considered as physical and quantitative model of human brain. Combining these two differing views of the human mind in a unifying framework allows for a stronger approach to modeling (Jang et al., 1997). In neuro-fuzzy systems, a fuzzy system is interpreted in terms of a neural network such that each step in the process is equivalent to at least one layer of the network. Consequently, most of these structures have at least four layers corresponding to fuzzification, intersection, implication and defuzzification, respectively (Gorzalczany, 2002). Different neuro-fuzzy structures can be found in the literature (Jang et al., 1997; Gorzalczany, 2002). One of the most popular and powerful architectures is the ANFIS (Jang et al., 1997) in which a TSK neuro-fuzzy system is represented in a special five-layer feedforward network architecture. It has been proven that the ANFIS structure has universal approximation capability (Jang et al., 1997). For this fusion process, rules are implemented in the following form:

If \( i_{Adel} \) is \( A_{1,r} \) and \( i_{Stephenson} \) is \( A_{2,r} \) and \( i_{McCorquodale} \) is \( A_{3,r} \),

then \( i_{\text{fusion}} = \alpha_0 + \sum_{k=1}^{3} \alpha_k i_{\text{sim},k} \) \hspace{1cm} (17)

where, \( r = 1, 2, \ldots, R \) is the rule index, \( k \) is the index of each empirical equation (Adel, Stephenson and McCorquodale et al., respectively), \( A_{k,r} \) is the membership function of \( k \)th empirical equation representing linguistic descriptions of inputs in \( r \)th rule, \( i_{\text{sim},k} \) is the numerical value of \( k \)th empirical equation and \( \alpha_0 \) and \( \alpha_k \) (\( k = 1, 2, 3 \)) are equation constants. The rule-base must be known in advance and ANFIS adjusts the membership functions of the antecedents and the consequent parameters. In brief, each layers’ functionalities are in the following form:

Layer 1 (input layer): Each unit of this layer stores three parameters \( a_{k,r}, b_{k,r}, c_{k,r} \) to define a bell-shaped membership function for \( k \)th input variable in \( r \)th rule:

\[ \mu_{A_{k,r}}(i_{\text{sim},k}) = \frac{1}{1 + [(i_{\text{sim},k} - c_{k,r})/a_{k,r}]} \] \hspace{1cm} (18)

Layer 2 (intersection layer): Each fuzzy rule in form of Eq. (17) is represented by one unit in this layer. The goal of this layer is to calculate the degree of fulfillment \( w_r \) (\( r = 1, 2, \ldots, R \)) of particular fuzzy rule implemented in the system. AND operators are represented with product. Hence:

\[ w_r = \prod_{i=1}^{n} \mu_{A_{k,r}}(i_{\text{sim},k}) \] \hspace{1cm} (19)

Layer 3 (normalization layer): The normalized degree of fulfillment of each rule is computed in this layer:

\[ \bar{w}_r = \frac{w_r}{\sum_{r=1}^{R} w_j} \] \hspace{1cm} (20)

Layer 4 (defuzzification layer): The units in this layer are connected to all inputs and to exactly one layer in previous layer. Each unit in layer 4, computes the output response \( o_r \) of the \( r \)th rule by:

\[ o_r = \bar{w}_r i_{\text{fusion},r} = \bar{w}_r \left( \alpha_{0,r} + \sum_{k=1}^{3} \alpha_k i_{\text{sim},k} \right) \] \hspace{1cm} (21)

Layer 5 (aggregation layer): One unit in this layer compute the final output of the whole system by summing all the outputs from previous layer:

\[ i_{\text{fusion}} = \sum_{r=1}^{R} o_r \] \hspace{1cm} (22)

A hybrid of least square estimator (LSE) and backpropagation is used for the learning of ANFIS. Backpropagation is used to learn the antecedent parameters and LSE is used to determine the coefficients of rules consequents (Jang and Gulley, 1995; Jang et al., 1997). MATLAB® fuzzy logic toolbox was used for simulation. In ANFIS system, the type of membership function must be identified. Also two different methods for rule generation, i.e., sub-clustering and grid partitioning (Jang and Gulley, 1995; Jang et al., 1997) can be used. It was seen that ANFIS system with three Inputs, one Output, four Gaussian membership functions for each variable and four linear consequence functions obtained by sub-clustering with 100 epochs hybrid training would be the best neuro-fuzzy model. Figure 9 shows the structure of ANFIS system with 3/10, four membership functions for each input variable obtained by grid partitioning and sub-clustering, respectively. Membership functions tuning procedure and performance of rule bases obtained by grid partitioning and sub-clustering are shown in the following figures.

8 Comparison and discussion

A total of 103 different soft computing-based fusion algorithms were simulated in this research. For each of these soft computing-based fusion algorithms, several modeling choices were trained based on training data and then compared with each other using
checking data. As an illustration, for feedforward neural networks, several modeling choices can be created based on different number of neurons, different number of layers, and different training procedures. These choices were then compared with each other and the best modeling choices were selected for comparison with best choices of other fusion algorithms, i.e., cascade forward, radial basis, Mamdani, and ANFIS frameworks. In other words, three steps were followed for finding the best fusion algorithm. First, selecting different choices for each algorithm with training data; then comparing the choices with each other and finding the best choice as the representative of each algorithm based on checking data; and finally comparing the best choices of all different methodologies for finding the best fusion algorithm based on testing data. Table 1 shows the description of
Soft computing-based nonlinear fusion algorithms

Figure 11 Combined hydraulic gradient obtained by ANFIS system with 3I/1O that each input has four membership functions and four rules obtained by sub-clustering versus observed hydraulic gradient: (a) training phase, (b) checking phase, (c) testing phase (based on normalized data).

Figure 12 (Left) Initial membership functions, (Right) membership function after 1000 epochs, (a) first variable (Adel equation), (b) second variable (Stephenson equation), (c) third variable (McCorquodale et al. equation). This ANFIS system is obtained by grid partitioning.

12 preferred models, among them are two well-known conventional fusion models, i.e., simple averaging and multiple linear regression, as well as the three empirical non-Darcy equations. In Table 2, four performance indices are selected for evaluation and comparison of these models. These are sum of square of errors (sum of square of differences between simulated and observed values), mean of errors, variance of errors and correlation between simulated and observed values. Sum of square of errors and variance of error provide a measure of accuracy of a model while mean of errors is a measure of a bias of a model. These parameters are compared in training, checking and testing phases, each demonstrating different characteristics of each fusion model. In our methodology, training phase indices show the learning capability of different models, checking indices estimate generalization capability and testing indices report prediction ability.
Table 1: Definition and description of best systems in last sections, traditional combination methods and empirical equations

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Properties and definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FF1</td>
<td>Feedforward backpropagation with (3 3 1) architecture, tangent sigmoid activation function, mean squares of error as objective function of training, 1000 epoch training with Levenberg–Marquardt backpropagation</td>
</tr>
<tr>
<td>2</td>
<td>FF2</td>
<td>Feedforward backpropagation with (3 2 1 1) architecture, tangent sigmoid activation function, mean squares of error as objective function of training, 1000 epoch training with Levenberg–Marquardt backpropagation</td>
</tr>
<tr>
<td>3</td>
<td>FF3</td>
<td>Feedforward backpropagation with (3 4 3 1) architecture, tangent sigmoid activation function, mean squares of error as objective function of training, 1000 epoch training with Levenberg–Marquardt backpropagation</td>
</tr>
<tr>
<td>4</td>
<td>CC1</td>
<td>Cascade forward backpropagation with (3 5 1) architecture, tangent sigmoid activation function, mean squares of error as objective function of training, 1000 epoch training with Levenberg–Marquardt backpropagation</td>
</tr>
<tr>
<td>5</td>
<td>CC2</td>
<td>Cascade forward backpropagation with (3 4 3 1) architecture, tangent sigmoid activation function, mean squares of error as objective function of training, 1000 epoch training with Levenberg–Marquardt backpropagation</td>
</tr>
<tr>
<td>6</td>
<td>CC3</td>
<td>Cascade forward backpropagation with (3 5 5 1) architecture, tangent sigmoid activation function, mean squares of error as objective function of training, 1000 epoch training with Levenberg–Marquardt backpropagation</td>
</tr>
<tr>
<td>7</td>
<td>CC4</td>
<td>Cascade forward backpropagation with (3 5 5 1) architecture, logistic sigmoid activation function, mean squares of error as objective function of training, 1000 epoch training with Levenberg–Marquardt backpropagation</td>
</tr>
<tr>
<td>8</td>
<td>RBF1</td>
<td>Radial basis function network with spread constant equals to 1</td>
</tr>
<tr>
<td>9</td>
<td>MAM4</td>
<td>Mamdani fuzzy system with 3I/1O, four Gaussian membership function for each variable, Max–Min operators and 16 rules obtained by weighted counting algorithm</td>
</tr>
<tr>
<td>10</td>
<td>MAM9</td>
<td>Mamdani fuzzy system with 3I/1O, four triangular membership function for each variable, product–sum operators and 16 rules obtained by weighted counting algorithm</td>
</tr>
<tr>
<td>11</td>
<td>ANFIS1</td>
<td>ANFIS system with 3I/1O, four Gaussian membership function for each variable and four linear consequence functions obtained by sub-clustering with 100 epoch hybrid training</td>
</tr>
<tr>
<td>12</td>
<td>ANFIS6</td>
<td>ANFIS system with 3I/1O, four Gaussian membership function for each variable and 64 linear consequence functions obtained by partitioning with 100 epoch hybrid training</td>
</tr>
<tr>
<td>13</td>
<td>SAM</td>
<td>Simple average method for combination</td>
</tr>
<tr>
<td>14</td>
<td>MLR</td>
<td>Multiple linear regression method by least squares estimation for combination</td>
</tr>
<tr>
<td>15</td>
<td>Adel</td>
<td>Adel empirical equation</td>
</tr>
<tr>
<td>16</td>
<td>Stephenson</td>
<td>Stephenson empirical equation</td>
</tr>
<tr>
<td>17</td>
<td>McCorquodale et al.</td>
<td>McCorquodale et al. empirical equation</td>
</tr>
</tbody>
</table>

Table 2: Comparison between different combination procedures and empirical equations, sum of square of errors (SSE), mean of errors (ME), variance of errors (VE), correlation coefficient (CC) (based on normalized data)

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Training phase (number of data: 350)</th>
<th>Checking phase (number of data: 50)</th>
<th>Testing phase (number of data: 83)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SSE</td>
<td>ME</td>
<td>VE</td>
</tr>
<tr>
<td>1</td>
<td>FF1</td>
<td>0.5845</td>
<td>0.0000</td>
<td>0.0017</td>
</tr>
<tr>
<td>2</td>
<td>FF2</td>
<td>0.7197</td>
<td>−0.0000</td>
<td>0.0021</td>
</tr>
<tr>
<td>3</td>
<td>FF3</td>
<td>0.2556</td>
<td>0.0001</td>
<td>0.0007</td>
</tr>
<tr>
<td>4</td>
<td>CC1</td>
<td>0.1194</td>
<td>0.0003</td>
<td>0.0004</td>
</tr>
<tr>
<td>5</td>
<td>CC2</td>
<td>0.1044</td>
<td>0.0003</td>
<td>0.0005</td>
</tr>
<tr>
<td>6</td>
<td>CC3</td>
<td>0.041</td>
<td>0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>7</td>
<td>CC4</td>
<td>0.032</td>
<td>0.0000</td>
<td>0.0001</td>
</tr>
<tr>
<td>8</td>
<td>RBF1</td>
<td>0.2865</td>
<td>−0.0001</td>
<td>0.0008</td>
</tr>
<tr>
<td>9</td>
<td>MAM4</td>
<td>9.7121</td>
<td>0.0500</td>
<td>0.0200</td>
</tr>
<tr>
<td>10</td>
<td>MAM9</td>
<td>5.5794</td>
<td>0.0384</td>
<td>0.0145</td>
</tr>
<tr>
<td>11</td>
<td>ANFIS1</td>
<td>1.1108</td>
<td>0.0000</td>
<td>0.0032</td>
</tr>
<tr>
<td>12</td>
<td>ANFIS6</td>
<td>0.360</td>
<td>0.0000</td>
<td>0.0010</td>
</tr>
<tr>
<td>13</td>
<td>SAM</td>
<td>9.1897</td>
<td>0.1243</td>
<td>0.0100</td>
</tr>
<tr>
<td>14</td>
<td>MLR</td>
<td>3.1980</td>
<td>0.0618</td>
<td>0.0050</td>
</tr>
<tr>
<td>15</td>
<td>Adel</td>
<td>5.1916</td>
<td>0.1061</td>
<td>0.0086</td>
</tr>
<tr>
<td>16</td>
<td>Stephenson</td>
<td>6.9427</td>
<td>0.1259</td>
<td>0.0135</td>
</tr>
<tr>
<td>17</td>
<td>McCorquodale et al.</td>
<td>10.232</td>
<td>0.1407</td>
<td>0.0164</td>
</tr>
</tbody>
</table>
Akaike information criterion (AIC), and Bayesian information criterion (BIC) are reported for all the models introduced in this study. For this reason, in Table 3, the number of parameters, variance of errors (VE), and AIC and BIC statistics penalize the model for having more parameters and therefore tend to advocate more parsimonious models. According to both AIC and BIC, a model with a lower value of AIC and/or BIC measure is considered to be more fit. Based on Table 3, it can be seen that all neural network-based fusion frameworks as well as the ANFIS model obtained by subtractive clustering reach lower values of AIC in comparison with those of SAM and MLR. But, if BIC is considered, it can be found that the values for RBF and ANFIS1 are higher than those of SAM and MLR. The reason for this fact is the large number of parameters for RBF and ANFIS in comparison with SAM and MLR.

In general, all applied soft computing-based solutions have large number of internal parameters in comparison with conventional methods. We believe that large number of parameters with current computational strength is not an important issue, if we can estimate them properly in reasonable time and if the model reaches significantly higher performance while remaining robust. Based on this hypothesis, although both feedforward and cascade forward ANN frameworks have more parameters than SAM and MLR, they are superior than conventional methods, because of their lower error measures indices reported in Table 2 and also better model selection measures resulted from Table 3.

Among the soft computing methodologies, simulation results indicate that cascade forward network with five hidden neuron is the best fusion architecture for describing the nonlinear flow in porous media when compared with pure Mamdani fuzzy system derived by weighted counting algorithm and ANFIS. For example, comparing CC1, FF1, MAM9, and ANFIS1 (best examples in each category in Table 2) it can be seen that CC1 offers a significantly lower sum squared error (SSE = 0.04), mean error (ME = –0.0008), and error variance (VE = 0.0005), while providing a desirable correlation coefficient (CC = 0.99) during testing phase and also the best AIC and BIC measures in Table 3.

As commonly recognized, there is a significant difference between the nature of the results provided by artificial neural networks and pure Mamdani fuzzy systems. Because the origin of our case study was a numerical one (the value of three best empirical-based equations as independent variable and observed hydraulic gradient as dependent variable) and lacked human intuition, the superior performance of neural network is reasonable. On the one hand it is shown that neural networks have high efficiency in low-level (numeric) information environment (Jang, 1996; Gorzalczany, 2002), and on the other hand, the learning ability of this paradigm could make their fusion process better than pure Mamdani fuzzy systems. It is interesting to observe that the performance of the ANFIS neuro-fuzzy system, that combines

Figure 13 Combined hydraulic gradient obtained by ANFIS system with 3I/1O that each input has four membership functions and four rules obtained by partitioning versus observed hydraulic gradient: (a) training phase, (b) checking phase, (c) testing phase (based on normalized data).

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Number of parameters</th>
<th>Variance of error</th>
<th>AIC</th>
<th>BIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FF1</td>
<td>16</td>
<td>0.0017</td>
<td>–6.2857</td>
<td>–6.10934</td>
</tr>
<tr>
<td>2</td>
<td>FF2</td>
<td>13</td>
<td>0.0021</td>
<td>–6.09153</td>
<td>–5.94824</td>
</tr>
<tr>
<td>3</td>
<td>FF3</td>
<td>35</td>
<td>0.0007</td>
<td>–7.06443</td>
<td>–6.67864</td>
</tr>
<tr>
<td>4</td>
<td>CC1</td>
<td>29</td>
<td>0.0004</td>
<td>–7.65833</td>
<td>–7.33867</td>
</tr>
<tr>
<td>5</td>
<td>CC2</td>
<td>51</td>
<td>0.0003</td>
<td>–7.8203</td>
<td>–7.25814</td>
</tr>
<tr>
<td>6</td>
<td>CC3</td>
<td>79</td>
<td>0.0001</td>
<td>–8.75891</td>
<td>–7.88812</td>
</tr>
<tr>
<td>7</td>
<td>CC4</td>
<td>79</td>
<td>0.0001</td>
<td>–8.75891</td>
<td>–7.88812</td>
</tr>
<tr>
<td>8</td>
<td>RBF1</td>
<td>210</td>
<td>0.0008</td>
<td>–5.9309</td>
<td>–3.61614</td>
</tr>
<tr>
<td>9</td>
<td>MAM4</td>
<td>48</td>
<td>0.02</td>
<td>–3.63774</td>
<td>–3.10865</td>
</tr>
<tr>
<td>10</td>
<td>MAM9</td>
<td>48</td>
<td>0.0145</td>
<td>–3.95932</td>
<td>–3.43023</td>
</tr>
<tr>
<td>11</td>
<td>ANFIS1</td>
<td>52</td>
<td>0.0032</td>
<td>–5.44746</td>
<td>–4.87428</td>
</tr>
<tr>
<td>12</td>
<td>ANFIS6</td>
<td>292</td>
<td>0.001</td>
<td>–5.23918</td>
<td>–2.02057</td>
</tr>
<tr>
<td>13</td>
<td>SAM</td>
<td>3</td>
<td>1.000</td>
<td>–4.59946</td>
<td>–4.58843</td>
</tr>
<tr>
<td>14</td>
<td>MLR</td>
<td>3</td>
<td>0.005</td>
<td>–5.28117</td>
<td>–5.24811</td>
</tr>
</tbody>
</table>

Another issue that should be addressed is the residual statistics that account for the model complexity. For this reason, in Table 3, the number of parameters, variance of errors (VE), Akaike information criterion (AIC), and Bayesian information criterion (BIC) are reported for all the models introduced in Table 1. AIC and BIC are defined as follows:

\[
\text{AIC}(p, n) = \ln(VE) + \frac{2p}{n}
\]

\[
\text{BIC}(p, n) = \ln(VE) + \frac{p \ln(n)}{n}
\]

where \( p \) is the number of model parameters, \( n \) is the sample size and VE is the maximum likelihood estimator of the error variance. While the previously mentioned statistics are expected to progressively improve as more parameters are added to the model, the AIC and BIC statistics penalize the model for having more parameters and therefore tend to advocate more parsimonious models. According to both AIC and BIC, a model with a lower value of AIC and/or BIC measure is considered to be more fit. Based on Table 3, it can be seen that all neural network-based fusion frameworks as well as the ANFIS model obtained by subtractive clustering reach lower values of AIC in comparison with those of SAM and MLR. But, if BIC is considered, it can be found that the values for RBF and ANFIS1 are higher than those of SAM and MLR. The reason for this fact is the large number of parameters for RBF and ANFIS in comparison with SAM and MLR.

In general, all applied soft computing-based solutions have large number of internal parameters in comparison with conventional methods. We believe that large number of parameters with current computational strength is not an important issue, if we can estimate them properly in reasonable time and if the model reaches significantly higher performance while remaining robust. Based on this hypothesis, although both feedforward and cascade forward ANN frameworks have more parameters than SAM and MLR, they are superior than conventional methods, because of their lower error measures indices reported in Table 2 and also better model selection measures resulted from Table 3.

Among the soft computing methodologies, simulation results indicate that cascade forward network with five hidden neuron is the best fusion architecture for describing the nonlinear flow in porous media when compared with pure Mamdani fuzzy system derived by weighted counting algorithm and ANFIS. For example, comparing CC1, FF1, MAM9, and ANFIS1 (best examples in each category in Table 2) it can be seen that CC1 offers a significantly lower sum squared error (SSE = 0.04), mean error (ME = –0.0008), and error variance (VE = 0.0005), while providing a desirable correlation coefficient (CC = 0.99) during testing phase and also the best AIC and BIC measures in Table 3.

As commonly recognized, there is a significant difference between the nature of the results provided by artificial neural networks and pure Mamdani fuzzy systems. Because the origin of our case study was a numerical one (the value of three best empirical-based equations as independent variable and observed hydraulic gradient as dependent variable) and lacked human intuition, the superior performance of neural network is reasonable. On the one hand it is shown that neural networks have high efficiency in low-level (numeric) information environment (Jang et al., 1997; Gorzalczany, 2002), and on the other hand, the learning ability of this paradigm could make their fusion process better than pure Mamdani fuzzy systems. It is interesting to observe that the performance of the ANFIS neuro-fuzzy system, that combines
learning ability of neural networks with the high-level (linguis-
tic) information processing of fuzzy system, lies between these
two extremes and produces an intermediary fusion performance;
however, the number of internal parameter for ANFIS paradigm
is too many for our sample data set and the application of the
algorithm does not seem to be justified in this context.

9 Conclusion

The most commonly used method for estimating hydraulic
parameters for flow through coarse porous media is empirical-
based equations. These equations do not fully reflect the flow
behavior in coarse porous media as shown by the work of pio-
ners in this field such as Joy and Hanson as well as in this
research. In this paper, we investigate a methodology for fusion of
tree benchmark empirical equations, i.e., McCorquodale et al.,
Stephenson and Adel, based on soft computing paradigms in
order to find a better estimation. The results indicate significant
superiority of the feedforward and cascade forward neural net-
works fusion algorithms as compared with each of the individual
equations or their investigated linear conventional combination
methods. In particular, cascade correlation feedforward neural
paradigm demonstrates the best estimation accuracy with low-
est computational complexity and highest performance indices.
Future directions of this research include application of other
soft computing-based methodologies for fusion and/or creating
a direct mapping from actual physical properties and sensory
measurement of the system to a hydraulic gradient.

Acknowledgment

Authors would like to thank Prof. D. M. Joy from University of
Guelph, Canada, for providing the data set of this study.

References

Modeling with Applications to Geophysical, Biological and
Sets Systems 37, 65–75.
Description of Elements of the Hydrological Cycle”. Ecol.
Modeling 85, 59–65.
4. DEMUTH, H. and BEALE, M. (2000). Neural Network Tool-
box for Use with MATLAB®, MathWorks Inc., Natick, MA.
ral Networks for Time Series Analysis: Rainfall-runoff
Modeling”. Signal Processing 64, 383–396.
6. GORZALCZANY, M.B. (2002). Computational Intelli-
gence Systems and Applications. Physica-Verlag Company,
Heidelberg, Germany.
“Selection and Application of a One-dimensional Non-
Darcy Flow Equation for Two-dimensional Flow through
ic Equations that Estimate Hydraulic Parameters for Flow
through Rockfill”. In: WANG & HU (eds), Stochastic
Hydraulics 2000. Balkema, Rotterdam, The Netherlands,
p. 547–552.
“Artificial Neural Networks Modeling of the Rainfall-runoff
for Use with MATLAB®, MathWorks Inc., Natick, MA.
fuzzy and Soft Computing: A Computational Approach to
Learning and Machine Intelligence. Prentice Hall, Englewood Cliffs, NJ.
Media”. Proceedings of Annual Conference of the CSCE,
mination of Parameters for a Coarse Granular Media”.
13th Canadian Congress on Applied Mechanics, Winnipeg,
Canada.
14. KARUNANITHI, N., GRENNEY, W.J., WHITLEY, D. and
Prediction”. J. Comput. Civil Engng. ASCE 8(2),
201–220.
Rockfill Structures”. J. Water Power Dam Construc-
tion March, 41–45.
Res. IAHR 16(2), 123–137.
17. NAZEMI, A.-R., AKBARZADEH M.-R., and HOSSEINI, S.M.
Resources Engineering”. In: KELLER, J. and NASRAOUI, K.
(eds), Proceedings of NAFIPS-FLINT 2002. IEEE,
18. SHAMSOLDIN, A.Y., O’ CONNOR, K.M. and LIANG, G.C.
(1997). “Methods for Combining the Outputs of Different
Based Modeling of Reservoir Operation”. J. Water Resour.
Planning Manage. ASCE 122(4), 262–269.
Developments in Geotechnical Engineering 27. Elsevier
Prentice Hall Inc., Englewood Cliffs, NJ.
“A Nonlinear Combination of the Forecasts of Rainfall-
Runoff Models by the First Order Takagi-Sugeno Fuzzy