# Estimating Saturated Hydraulic Conductivity Using Genetic Programming

**Kamban Parasuraman***
**Amin Elshorbagy**
Centre for Advanced Numerical Simulation (CANSIM)
Dep. of Civil and Geological Engineering
Univ. of Saskatchewan
Saskatoon, SK, S7N 5A9
Canada

**Bing Cheng Si**
Dep. of Soil Science
Univ. of Saskatchewan
Saskatoon, SK, S7N 5A8
Canada

Saturated hydraulic conductivity ($K_s$) is one of the key parameters in modeling solute and water movement in the vadose zone. Field and laboratory measurement of $K_s$ is time consuming, and hence is not practical for characterizing the large spatial and temporal variability of $K_s$. As an alternative to direct measurements, pedotransfer functions (PTFs), which estimate $K_s$ from readily available soil data, are being widely adopted. This study explores the utility of a promising data-driven method, namely, genetic programming (GP), to develop PTFs for estimating $K_s$ from sand, silt, and clay contents and bulk density ($D_b$). A data set from the Unsaturated Soil Hydraulic Database (UNSODA) was considered in this study. The performance of the GP models were compared with the neural networks (NNs) model, as it is the most widely adopted method for developing PTFs. The uncertainty of the PTFs was evaluated by combining the GP and the NN models, using the nonparametric bootstrap method. Results from the study indicate that GP appears to be a promising tool for developing PTFs for estimating $K_s$. The better performance of the GP model may be attributed to the ability of GP to optimize both the model structure and its parameters in unison. For the PTFs developed using GP, the uncertainty due to model structure is shown to be more than the uncertainty due to model parameters. Moreover, the results indicate that it is difficult, if not impossible, to achieve better prediction and less uncertainty simultaneously.

Abbreviations: BR, Bayesian-regularization; GP, genetic programming; MARE, mean absolute relative error; MR, mean residual; NN, neural network; PTF, pedotransfer function; UNSODA, Unsaturated Soil Hydraulic Database.

During the past few decades, vadose zone modeling has received significant impetus due to the advancement in computing power and technology. Hence, the focus on vadose zone models has shifted from coarse lumped models to more realistic, spatially distributed models. These spatially distributed models have drastically increased the need for soil hydraulic data at a finer resolution. Direct (field and laboratory) measurement of soil hydraulic data is labor intensive, time consuming, and expensive, as these methods require restrictive initial and boundary conditions. For a detailed review of different laboratory and field measurements of soil hydraulic data, see Klute (1986). The problems (labor intensive, time consuming, and expensive) associated with direct measurement of soil hydraulic properties make it quite impractical to amass a data set at a resolution required for implementing such a spatially distributed model. Alternatively, these soil hydraulic properties can be estimated from more easily available soil data by the use of pedotransfer functions (PTFs) (Bouma, 1989).

Pedotransfer functions are predictive functions that can translate basic soil data like particle-size distributions, bulk density, and organic matter content into soil hydraulic properties. Because of this, interest in developing PTFs is increasing (Rawls and Brakensiek, 1983; Cosby et al., 1984; Saxton et al., 1986; Vereecken et al., 1990; van Genuchten et al., 1992; Leij et al., 2002). A detailed review of different PTFs was given by Wösten et al. (2001). Several methods have been adopted to develop PTFs. These methods range from simple look-up tables to more complex data-driven methods like regression analysis, neural networks (NNs), the group method of data handling, and regression trees. Gupta and Larson (1979) used linear regression to estimate the soil water characteristic. Rawls et al. (1991) and Minasny et al. (1999) used nonlinear regression to develop PTFs. The regression models are being gradually replaced by the NNs models in developing PTFs. Key examples of such studies include Pachepsky et al. (1996), Schaap and Bouten (1996), Minasny et al. (1999), and Tamarai et al. (1998). Another data-driven technique, the group method of data handling, has been used by Pachepsky et al. (1998), Nemes et al. (2005), and Ungaro et al. (2005) for developing PTFs. The technique of regression trees has been used by McKenzie and Jacquier (1997) for developing PTFs. As evident from a plethora of studies on NN-based PTFs, the NN appears to be the most widely adopted method for developing PTFs.

Recently, another promising, inductive, data-driven technique called *genetic programming* (GP) was proposed by Koza (1992). Genetic programming is a method for constructing populations of models using stochastic search methods, namely evolutionary algorithms. An important characteristic of GP is that both the variables and constants of the candidate models are optimized. Hence, compared with other regression techniques, it is not required to choose the model structure a priori. In water-related studies, GP has been applied to model flow over a flexible bed (Babovic and

Soil Sci. Soc. Am. J. 71:1676–1684
doi:10.2136/sssaj2006.0396
Received 17 Nov. 2006.
*Corresponding author (kap923@mail.usask.ca).

Abbott, 1997), rainfall–runoff processes (Whigham and Crapper, 2001; Savic et al., 1999), runoff forecasting (Khu et al., 2001), urban fractured-rock aquifer dynamics (Hong and Rosen, 2002), temperature downscaling (Coulibaly, 2004), the rainfall-recharge process (Hong et al., 2005), soil moisture (Makkeasorn et al., 2006), and evapotranspiration (Parasuraman et al., 2007).

Although GP and the most widely used method for developing PTFs, namely the NN, can be seen as alternative techniques for the same task, such as, e.g., classification and approximation problems, in contrast to NN, studies to determine the utility of GP in developing PTFs have not been attempted yet. Hence in this study, an attempt has been made to explore the efficacy of GP in developing PTFs for estimating saturated hydraulic conductivity ($K_s$). The specific objectives of this study include: (i) developing PTFs for estimating $K_s$ using GP; (ii) comparing the performance of the GP-based PTFs with the performance of the NN-based model, as it is the most widely used method for developing PTFs; and (iii) highlighting the potential as well as the shortcomings of the use of GP for geophysical applications.

## MATERIALS AND METHODS

### Data Set Used

The data set used in this study was derived from the Unsaturated Soil Hydraulic Database (UNSODA; Leij et al., 1996). The UNSODA was developed to provide a source of hydraulic data and other soil properties for practitioners and researchers, and derived from soil samples from Europe and North America. Compared with soils with finer texture, coarse-textured soils are in a majority in the UNSODA (Leij et al., 1996). The UNSODA has been widely used for developing PTFs (e.g., Schaap and Leij, 1998; Schaap et al., 2001; Ungaro et al., 2005). Sand, silt, and clay content (SSC), bulk density ($D_b$), and $K_s$ values of 314 samples were extracted from the UNSODA. Out of the 314 samples, 200 and 114 samples were selected for model calibration and validation, respectively. The $K_s$ values were log transformed to account for their lognormal distribution. One of the samples in the calibration data set had $K_s$ = 1 cm d$^{-1}$, which when log transformed resulted in $\log_{10}(K_s) = 0$. As detailed below, the mean absolute relative error (MARE) was used as one of the measures for evaluating the models' performance. Hence, the above-mentioned value was discarded from the calibration set, as it was not possible to calculate the relative error for that particular sample. Because the UNSODA was created by assembling different sources of data that came from different parts of the world, the data set as such is stratified randomly. Hence, no special consideration was given to specifically sample the data set between the training and the testing set. From the available 313 data instances, the first 199 instances were selected for training, and the remaining 114 data instances were used for testing. The descriptive statistics, along with the correlation matrix, of the data set used for calibration and validation are presented in Tables 1 and 2, respectively. The coefficient of variation (CVs) of different variables during training and testing are comparable (Tables 1 and 2).

### Genetic Programming

Genetic programming, introduced by Koza (1992), is a new addition to a class of evolutionary algorithms such as evolutionary programming (Fogel et al., 1966), genetic algorithms (Holland, 1975), and evolution strategies (Schwefel, 1981). In GP, a population of solution candidates evolves through many generations toward an optimal solution, using the concepts of natural selection and genetics. Genetic symbolic regression (GSR) is a special application of GP in the area of symbolic regression, where the objective is to find a mathematical expression in symbolic form that provides an optimal fit between a finite sampling of values of the independent variable and its associated values of the dependent variable (Koza, 1992).

Genetic symbolic regression can be considered as an extension of numerical regression problems. In numerical regression problems, one predetermines the functional form (linear, quadratic, or polynomial), and the objective is to find the set of numerical coefficients that best fits the chosen model structure. Genetic symbolic regression does not require the functional form to be defined a priori, however, as GSR involves finding the optimal mathematical expression in symbolic form (both the discovery of the correct functional form and the appropriate numerical coefficients) that defines the predictand–predictor relationship. More information on GP can be found in Koza (1992) and Babovic and Keijzer (2000).

Figure 1 shows the flowchart of the GSR paradigm. For a given problem, the first step is to define the functional and terminal sets, along with the objective function and the genetic operators. The functional set and the terminal set are the main building blocks of the GSR, and hence their appropriate identification is central in developing a robust GSR model. The functional set consists of basic mathematical operators {+, −, ×, /, sin, exp,

**Table 1. Descriptive statistics and correlation matrix of the training data set.**

| Statistic† | Sand | Silt | Clay | Bulk density | $\log_{10}K_s$‡ |
|---|---|---|---|---|---|
| | % | % | % | g cm$^{-3}$ | |
| Minimum | 1.80 | 0.20 | 0.00 | 0.59 | −1.19 |
| Maximum | 99.10 | 81.40 | 63.00 | 1.97 | 4.44 |
| Median | 52.80 | 25.55 | 14.95 | 1.52 | 1.94 |
| Mean | 54.33 | 27.96 | 17.71 | 1.47 | 1.87 |
| SD | 30.53 | 20.73 | 15.46 | 0.25 | 1.08 |
| CV | 0.56 | 0.74 | 0.87 | 0.17 | 0.58 |
| | | | Correlations | | |
| Sand, % | 1.00 | | | | |
| Silt, % | −0.89 | 1.00 | | | |
| Clay, % | −0.79 | 0.41 | 1.00 | | |
| Bulk density, g cm$^{-3}$ | 0.47 | −0.38 | −0.42 | 1.00 | |
| $\log_{10}K_s$ | 0.52 | −0.39 | −0.50 | −0.11 | 1.00 |

† SD, standard deviation; CV, coefficient of variation.

‡ $K_s$, saturated hydraulic conductivity.

**Table 2. Descriptive statistics and correlation matrix of the testing data set.**

| Statistic† | Sand | Silt | Clay | Bulk density | $\log_{10}K_s$‡ |
|---|---|---|---|---|---|
| | % | % | % | g cm$^{-3}$ | |
| Minimum | 0.10 | 0.30 | 0.10 | 0.49 | −0.84 |
| Maximum | 99.60 | 80.70 | 54.40 | 1.76 | 3.58 |
| Median | 54.75 | 27.30 | 11.25 | 1.49 | 1.91 |
| Mean | 51.77 | 34.58 | 13.65 | 1.48 | 1.78 |
| SD | 33.21 | 26.83 | 11.74 | 0.16 | 0.89 |
| CV | 0.64 | 0.78 | 0.86 | 0.11 | 0.50 |
| | | | Correlations | | |
| Sand, % | 1.00 | | | | |
| Silt, % | −0.95 | 1.00 | | | |
| Clay, % | −0.67 | 0.39 | 1.00 | | |
| Bulk density, g cm$^{-3}$ | 0.34 | −0.24 | −0.41 | 1.00 | |
| $\log_{10}K_s$ | 0.39 | −0.40 | −0.20 | −0.07 | 1.00 |

† SD, standard deviation; CV, coefficient of variation.

‡ $K_s$, saturated hydraulic conductivity.

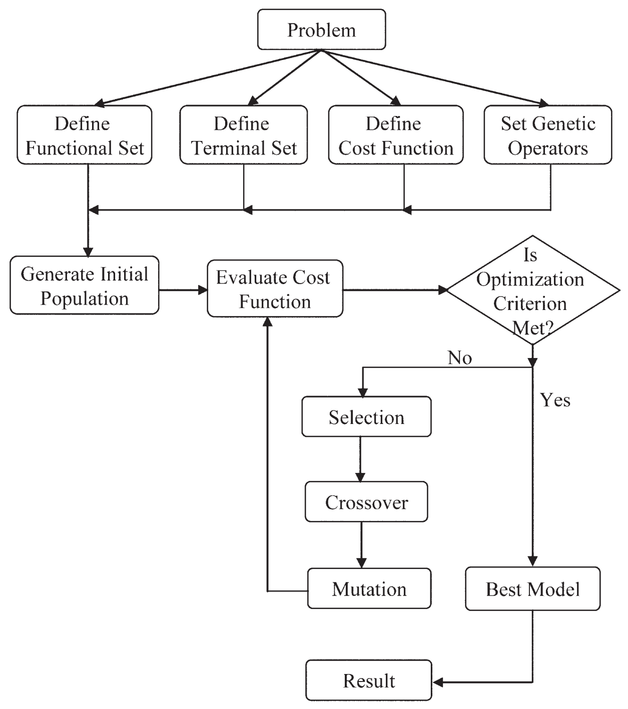

**Fig. 1. Flowchart of the genetic symbolic regression paradigm.**

…} that may be used to form the model. The choice of operators depends on the degree of complexity of the problem to be modeled. The terminal set consists of independent variables and constants. The constants can be either physical constants (e.g., Earth's gravitational acceleration, the specific gravity of fluid) or randomly generated constants. Different combinations of functional and terminal sets are used to construct a population of mathematical models. Each model (individual) in the population can be considered as a potential solution to the problem. The mathematical models are usually coded in a parse tree form. For example, Fig. 2 shows the parse tree notation of a mathematical model $f(x,y,z) = (x + y)(6/z)$. In Fig. 2, the connection points are called nodes, and it can be seen that the inner nodes of the parse tree are made up of functions, and the terminal nodes are made up of variables and constants. Hence in GP terminology, the variables and constants are referred to as *terminals*, and the functions are referred to as *non-terminals*. The depth of the sparse tree shown in Fig. 2 is three. Objective or fitness or cost function is used to evaluate the value or fitness of each individual in the population. Usually, the squared error statistic or its variant (MSE and RMSE) is used as the objective function. Genetic operators include selection, crossover, and mutation, and they are discussed in detail below.

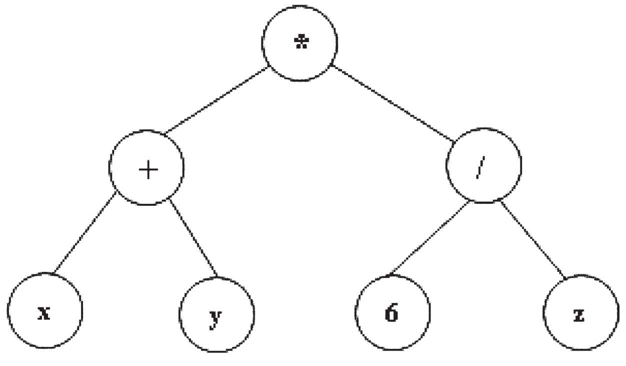

**Fig. 2. Sparse tree notation.**

Once the functional and terminal sets are defined, the next step is to generate the initial population for a given population size. The initial population can be generated in a multitude of ways, including the full method, the grow method, and the ramped half-and-half method. In the full method, the new trees are generated by assigning non-terminal nodes until a predescribed initial maximum tree depth is reached, and the last depth level is limited to the terminal node. The full method usually results in perfectly balanced trees with branches of the same length. In the grow method, each new node is randomly chosen between the terminals and the non-terminals, with the terminals making up the node at the initial maximum tree depth. As a consequence, the grow method usually results in highly unbalanced trees. The ramped half-and-half method is a combination of the full and the grow methods. For each depth level considered, half of the individuals are initialized using the full method and the other half using the grow method. The ramped half-and-half method produces highly diverse trees, both in terms of size and shape (Koza, 1992), and thereby provides a good coverage of the search space. More information on the different methods of generating the initial population can be found in Koza (1992). Once initialized, the fitness of each individual (mathematical model) in the population is evaluated based on the selected objective function. The higher the fitness of an individual, the greater is the chance of the individual being carried over to the next generation. At each generation, new sets of models are evolved by applying the genetic operators: selection, crossover, and mutation (Koza, 1992; Babovic and Keijzer, 2000). These new models are termed *offspring*, and they form the basis for the next generation.

After the fitness of the individual models in the population is evaluated, the next step is to carry out selection. The objective of the selection process is to create a temporary population called the *mating pool*, which can be acted on by the genetic operators crossover and mutation. Selection can be performed by several methods such as truncation selection, tournament selection, and roulette wheel selection (Koza, 1992). As roulette wheel selection is the most widely used method, including Koza (1992), it has been adopted in this study. A roulette wheel is constructed by proportioning the space in the roulette wheel based on the fitness of each model in the population. The selection process ensures that the models with higher fitness have more chance of being carried over to the next generation.

Crossover is performed by choosing two parent models from the mating pool and swapping corresponding subtree structures across a randomly chosen point to produce two different offspring with different characteristics. The number of models undergoing crossover depends on the chosen probability of crossover, $P_c$. Mutation involves random alteration of the parse tree at the branch or node level. This alteration is done based on the probability of mutation, $P_m$. For an overview of different types of computational mutations, see Babovic and Keijzer (2000). While the role of the crossover operator is to generate new models that did not exist in the old population, the mutation operator guards against premature convergence by constantly introducing new offspring into the population. Figure 3 demonstrates the crossover and mutation operators. The crossover point between Parent 1 and Parent 2 is shown by the dashed line, and the corresponding subtree structures are swapped, resulting in Offspring 1 and Offspring 2. In Offspring 1, the terminal node has undergone mutation (2 replaced by $z$). The genetic operators, crossover and mutation, are shown to produce new models (offspring), which are structurally different from their parent models (Fig. 3). These operators ensure that the model space is sampled thoroughly to arrive at the optimal model. After the initial population has been acted on by the genetic operators, the resultant individuals form the new population for the next generation. This iterative process is performed for a predetermined number of iterations or until a specified value of cost function is reached.

In this study, the GP system used is an adaptation of GPLAB (Silva, 2007), a GP toolbox for MATLAB. Since the values of the GP system parameters (e.g., crossover rate, mutation rate, population size) are problem dependent, the usual practice is to determine them using a trial-and-error process with the objective of minimizing the cost function during the training process. This study adopted a similar approach in arriving at the GP parameters, and the resulting parameter values are shown in Table 3. One of the main issues that needs to be addressed in developing a GP system is that of "bloating." Bloating refers to the exponential growth of redundant and functionally useless trees. This is caused by the genetic operators (crossover and mutation) in their quest to arrive at better solutions. Several bloat control techniques have been proposed, and a review of these methods can be found in Soule and Foster (1999), Poli (2003), and Silva and Costa (2004). This study adopted the heavy dynamic limit method proposed by Silva and Costa (2004), which is based on attaching a dynamic limit on the depth of the trees allowed in the population, initially set with a low value, and only raised and lowered when needed to accommodate an individual with better performance that would otherwise break the limit. More information on the heavy dynamic limit method can be found in Silva and Costa (2004).

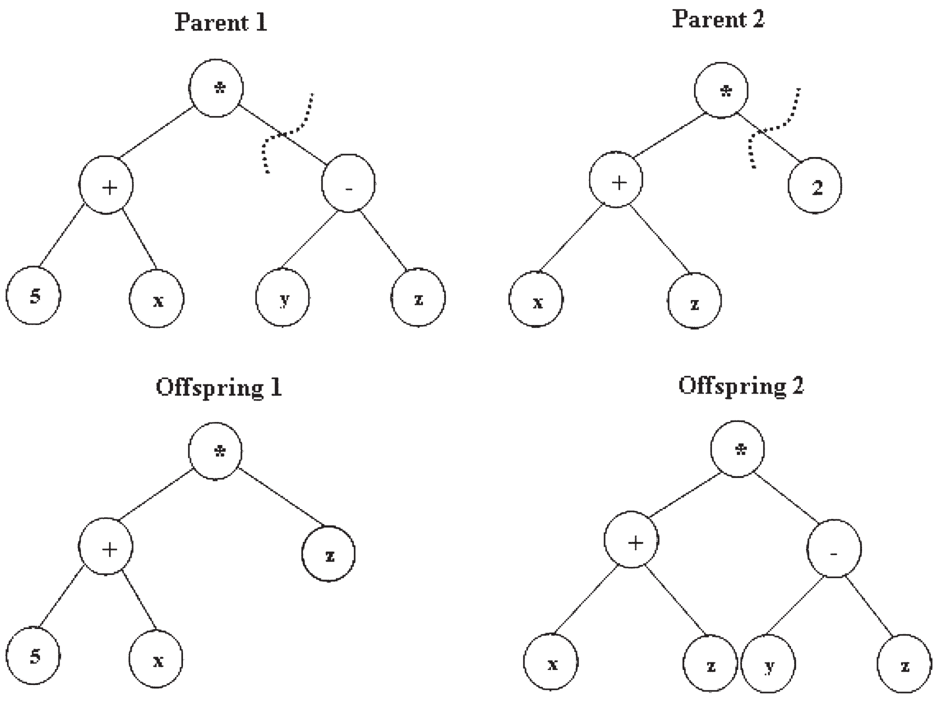

**Fig. 3. Crossover coupled with mutation. The dashed line indicates the crossover point and the shaded region represents the mutated node.**

Two GP models, GP(1) and GP(2), were developed to estimate $K_s$. While the functional set of the GP(1) and GP(2) models was {+, −} and {+, −, /, ×}, respectively, the terminal set of both the models remained the same. Along with randomly generated constants, {sand, silt, clay, $D_b$} constituted the terminal set of both the GP(1) and GP(2) models. This exercise was performed to evaluate the performance of the GP models with varying levels of mathematical operators (complexity) that can be used to define the predictand–predictor relationship. Before developing PTFs for estimating $K_s$ using GP, both the independent (sand, silt, clay, $D_b$) and the dependent [log10($K_s$)] variables were normalized by dividing each variable by its corresponding maximum value. This was done to overcome the problem of dimensional inconsistency and to achieve better generalization. These standardized values are simply referred to as sand, silt, clay, $D_b$, and $K_s$.

## Performance Evaluation

The performance of the GP-based models was compared with the NN model, as it is the most widely used method for developing PTFs. A detailed description of NN is beyond the scope of this study. For a detailed understanding of NNs, see Haykin (1999). The NN model adopted in this study used a Bayesian-regularization (BR) algorithm for training the networks. The BR algorithm has the advantage of producing networks with a good generalization property, as the cost function (Eq. [1]) involves minimizing both the mean sum of squares of the network errors and the mean of the sum of squares of the network weights and bias. In Eq. [1], $y_i$ and $y_i'$ are the measured and computed log10($K_s$) values, α is the regularization parameter, $w_j$ are the connection weights and bias values, and $n$ and $N$ are the number of training instances and the number of network parameters, respectively. More information on BR algorithms can be found in Demuth and Beale (2001). By the trial-and-error method, the optimal number of hidden neurons was found to be six. The hidden layer neurons use a tan-sigmoidal activation function and the output layer neurons use a linear activation function. This NN model will be referred as NN(BR). The performance of the different models was evaluated based on (i) RMSE, (ii) MARE, and (iii) mean residual (MR). The RMSE, MARE, and MR statistics are calculated using Eq. [2], [3], and [4], respectively. In Eq. [2], [3], and [4], similar to Eq. [1], $y_i$ and $y_i'$ represent the measured and computed $\log_{10}(K_s)$ values and $n$ represents the number of data instances.

$$\text{MSE REG} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_i'\right)^2 + (1-\alpha)\left(\frac{1}{N}\sum_{j=1}^{N} w_j^2\right) \qquad [1]$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_i'\right)^2} \qquad [2]$$

**Table 3. Genetic programming parameters.**

| Parameter | Value |
|---|---|
| Population size | 20 |
| Initialization method | Ramped half-and-half |
| Sampling method | Roulette |
| Maximum initial tree depth | 8 |
| Probability of crossover | 0.6 |
| Probability of mutation | 0.3 |
| Cost function | RMSE |
| Number of generations | 1000 |

**Table 4. Performance statistics† of the neural network model using the Bayesian-regularization algorithm [NN(BR)], genetic programming model 1 [GP(1)], and genetic programming model 2 [GP(2)] in estimating saturated hydraulic conductivity.**

| Model | Training | | | | Testing | | | |
|---|---|---|---|---|---|---|---|---|
| | Uncertainty | RMSE | MARE | MR | Uncertainty | RMSE | MARE | MR |
| NN(BR) | 0.26 | 0.61 | 0.55 | -0.01 | 0.27 | 1.04 | 2.23 | -0.09 |
| GP(1) | 0.27 | 0.83 | 0.76 | 0.02 | 0.26 | 0.90 | 2.24 | -0.22 |
| GP(2) | 0.27 | 0.70 | 0.68 | 0.02 | 0.30 | 0.89 | 1.98 | -0.13 |

† MARE, mean absolute relative error; MR, mean residual; RMSE, root mean squared error.

$$\mathrm{MARE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - y_i{'}}{y_i}\right| \qquad [3]$$

$$\mathrm{MR} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_i{'}\right) \qquad [4]$$

## Pedotransfer Function Uncertainty

Wösten (1990) and McBratney et al. (2002) underscored the need for uncertainty estimates as part of the PTF performance evaluation. The uncertainty estimate, unlike the RMSE, MARE, and MR statistics, indicates the reliability of $K_s$ estimated by the model. Calculation of RMSE, MARE, and MR is possible only when measured $K_s$ values are available. In such instances, an uncertainty estimate can still provide the measure of reliability of $K_s$ estimated by the model. Usually, PTF uncertainty is calculated using the nonparametric bootstrap method (Efron and Tibshirani, 1993). The bootstrap method presumes that the training data set is a good representation of the original population, and that this data set is only one particular realization of that population. Hence, training the model on a different realization of the population would result in a slightly different prediction of $K_s$. To account for such uncertainty in prediction, $B$ independent data sets ($T_B$), of size $N$, can be generated by repeated random resampling with replacement of the training data set ($T$), of size $N$. Hence each bootstrap data set $T_B$ may have many instances of $T$ repeated several times, while other instances may be left out. Since $T_B$ contains a different realization of $T$, models trained on each of the $T_B$ may be slightly different. It should be noted that the different realizations produced by the bootstrap method are different not in terms of relative values, but only in the order and occurrence of the data instances. For this study, bootstrap size, $B$, was assumed to be 50.

For a rational comparison, initially the resampled data sets ($T_B$) were generated and predetermined so that the NN and GP models could be trained on the same resampled data sets. The model accuracy and its related uncertainty was calculated in the following manner: (i) since a bootstrap size of 50 was used in this study, for each input vector in the data set, the NN and GP PTFs resulted in 50 different model predictions, based on models trained on the 50 resampled data sets (i.e., for each input vector, the PTF resulted in 50 different model predictions); (ii) for that particular input vector, the estimated PTF value and its related uncertainty was determined by calculating the mean and standard deviations of the 50 different model predictions. The mean represents the model-estimated value, and the standard deviation represents the uncertainty associated with that particular model estimate; (iii) similarly, the model estimates and their related uncertainty were evaluated for all the input vectors in the training and testing data sets; (iv) for a particular PTF, the performance in terms of RMSE, MARE, and MR statistics was then calculated by comparing the model estimates with their measured counterparts across the entire training and testing ranges; and (v) the overall uncertainty associated with that particular PTF was calculated by averaging the standard deviations of the ensemble model predictions across the entire training and testing ranges.

Adopting the ensemble technique in PTF development not only assists in evaluating the uncertainty of the developed PTFs, but also helps in addressing one of the pertinent issues in any machine learning (e.g., artificial neural networks, GP) algorithm, namely generalization. Iba (1999) applied the ensemble method of bagging and boosting within the framework of GP and obtained encouraging results. Keijzer and Babovic (2000) and Folino et al. (2006) demonstrated that ensemble methods like bagging and boosting can reduce the generalization error in GP. Hence, the models developed in this study by combining a self-organizing algorithm with statistical resampling techniques were expected to reduce, if not fully overcome, the generalization error. It should be noted that in the case of the GP models [GP(1) and GP(2)], for each $T_B$, both the model structure and the model parameters were evolved simultaneously by the self-organizing nature of the GP algorithm. Nevertheless, in the NN(BR) model, for each $T_B$, the model structure was assumed to be deterministic, with the model parameters alone optimized based on $T_B$. Although, the framework of the GP and NN(BR) models are not functionally identical, the comparison of the these models was effected to illustrate the value of the self-organizing ability of the GP-based PTFs, proposed in this study, against the conventional method by which NN-based PTFs were developed.

## RESULTS AND DISCUSSION

The performance statistics of different models in estimating $K_s$, during both training and testing, are presented in Table 4. During training, NN(BR) resulted in a RMSE of 0.61, MARE of 0.55, MR of −0.01, with an uncertainty of 0.26 (Table 4). During testing, however, the corresponding values were 1.04, 2.23, −0.09, and 0.27 (Table 4). It can be observed that, compared with training, the testing RMSE and MARE estimates increased significantly in the case of the NN(BR) model compared with the GP models (Table 4). This demonstrates that although the NN(BR) model was robust in learning the input–output patterns in the training data set, it was not able to generalize the relationship.

During training, the GP(2) (more complex) model performed better than the GP(1) (less complex) model in terms of RMSE and MARE, and equally in terms of MR and the uncertainty estimate (Table 4). During testing, the GP(2) model performed better than the GP(1) model in terms of RMSE, MARE, and MR. Compared with the GP(1) model, however, the GP(2) model resulted in a higher value of the uncertainty estimate (Table 4). This indicates that increasing the number of mathematical operators, which can be used to define the predictand–predictor relationship, would lead to a better fit (less error) of the function, but at an increasing level of uncertainty. Based on this result, it can be concluded that it is difficult, if not impossible, to achieve both higher prediction accuracy and less uncertainty in tandem.

In general during training, the NN(BR) model performed better than both the GP(1) and GP(2) models in terms of RMSE, MARE, MR and the uncertainty estimates (Table 4). While both the GP models were slightly overpredicting (positive MR) $K_s$, the NN(BR) model was slightly underpredicting (negative MR) $K_s$ (Table 4). Because of the very small MR values, both the NN(BR) and the GP models can be considered unbiased. During testing, however, all the models resulted in a negative MR, which indicates that the models were underpredicting the $K_s$ values. Nevertheless, the least RMSE and MARE values of 0.89 and 1.98 were achieved by the GP(2) model. Overall during testing, the GP(1) model resulted in the smallest uncertainty estimate of 0.26, and the NN(BR) model resulted in the smallest MR estimate of −0.09. One of the main differences between the NN(BR) model and the GP models adopted in this study is that, in the case of the NN(BR) model, we predefined the initial structure of the network (network inputs, number of hidden layers and hidden neurons). Hence, the structure of the neural networks remained static for each of the resampled data sets, with the values of connection weights of the network optimized every time to fit the corresponding resampled data set. On the other hand, the GP models are self-organizing in nature, i.e., the model structure and its parameters are evolved simultaneously during the estimation process, learning from the features of the data set. Therefore, each of the resampled data sets would result in a GP model with different structure and parameters, which would near optimally fit the corresponding resampled data set. Figure 4 shows the correlation plots between the measured and estimated $K_s$ values by different models during training and testing. Although the regressions (Fig. 4) are not substantial and account for only a moderate proportion of the variance, the results are encouraging considering the presence of large scatter between the independent and the dependent variables, typical in the application of PTFs (Pachepsky and Rawls, 1999).

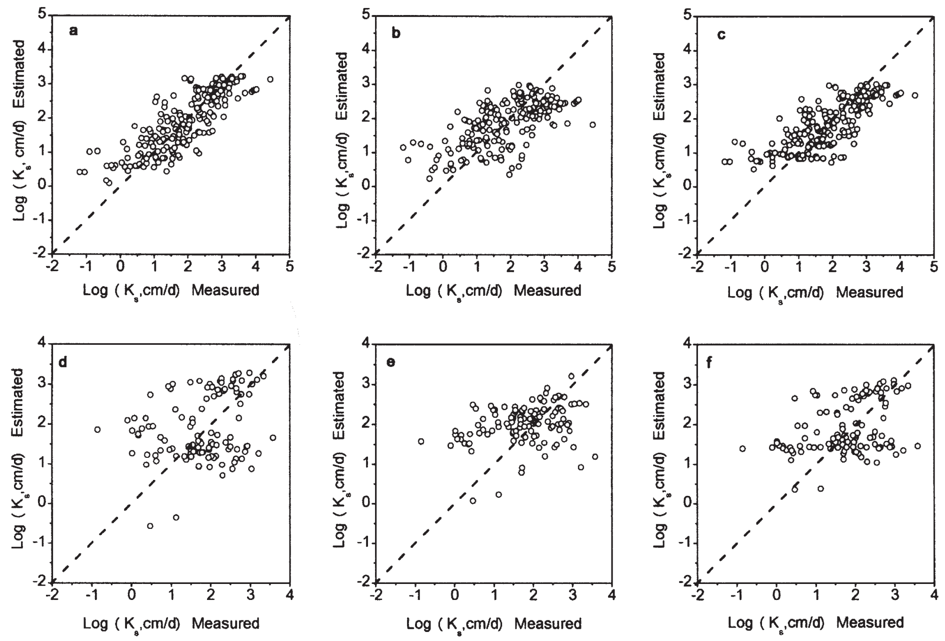

**Fig. 4. Comparison of measured and estimated saturated hydraulic conductivity ($K_S$) during training using (a) neural network model using the Bayesian-regularization algorithm [NN(BR)], (b) genetic programming model 1 [GP(1)], and (c) genetic programming model 2 [GP(2)] and testing using (d) NN(BR), (e) GP(1), and (f) GP(2).**

One of the important issues in the development of neural network models is the determination of the optimal configuration of the model. The optimal number of hidden neurons and the most significant inputs are usually determined by the trial-and-error method. This innate problem in NN modeling can be overcome, however, by the ability of GP to evolve its own model structure with relevant inputs. Table 5 shows the most relevant inputs identified by the GP models for the 50 realizations of the training data set. The values in Table 5 represent the percentage of occurrence of each of the variables of the terminal set in the 50 optimum models. As in the case of the GP(1) model, when only additive operators like + and − are included as part of the functional set, the corresponding percentage of occurrence of sand, silt, clay, and $D_b$ among the optimal models identified for each bootstrapped data set are 20.6, 20.6, 26.5, and 32.4%, respectively. When multiplicative operators × and / are added as part of the functional set, however, as in the case of the GP(2) model, the percentage of occurrence of sand, silt, and $D_b$ increased to 21.7, 21.4, and 38.5, respectively. Meanwhile, the percentage of occurrence of clay content decreased to 18.4%. It can be observed that both GP(1) and GP(2) identified $D_b$ as the most significant input variable (Table 5), followed by clay, silt, and sand contents in the case of the GP(1) model, and by sand, silt, and clay contents in the case of the GP(2) model. These results indicate that the most relevant input variables (model structure) for estimating $K_s$ using GP are not unique, but rather depend on the kind of mathematical operators (model complexity) that are considered part of the functional set of the GP paradigm to find the predictand–predictor relationship.

## DISCUSSION

Although NNs have been successfully adopted in developing PTFs for estimating different hydraulic characteristics of soils, their interpretation is often difficult. In NN-based PTFs, the knowledge of the predictand–predictor relationships is represented in the form of a weight matrix, which is difficult to comprehend. For the same problem, however, a GP-based PTF gives an explicit equation that can be elucidated with relative ease, depending on the complexity of the evolved equations. As described above, in this study, the

**Table 5. Percentage of different input variable selection in the genetic programming (GP) models.**

| Model | Sand | Silt | Clay | Bulk density |
|---|---|---|---|---|
| | % | | | |
| GP(1) | 20.6 | 20.6 | 26.5 | 32.4 |
| GP(2) | 21.7 | 21.4 | 18.4 | 38.5 |

GP models were combined with the parametric bootstrap method, which generated 50 realizations of the training data set. Hence, for the 50 realizations of the training data set, GP based on self-organizing learning would have arrived at 50 different PTFs. It is not feasible to present all 50 different PTFs for each of the GP models. Nevertheless, to enunciate the transparency of the GP models, both GP(1) and GP(2) were applied to the actual training data to arrive at corresponding representative PTFs. The GP parameters used in the previous simulation runs (Table 3) were retained for this analysis. Equations [5] and [6] show the representative PTFs obtained by the GP(1) and GP(2) models, respectively, based on the original training data set. Also, it should be noted that the relationships shown in Eq. [5] and [6] are based on normalized values of sand, silt, clay, $D_b$, and $\log_{10}(K_s)$. From the equations, it can be observed that the structure of PTFs found by the GP(1) and GP(2) models were different as they are influenced by the kind of mathematical operators used as part of the functional set. Applying Eq. [5] to the actual training data set (without resampling), resulted in an RMSE of 1.39, MARE of 0.79, and MR of –0.07. The corresponding values for the testing data set were 0.94, 2.4, and –0.31, respectively. Similarly, Eq. [6] applied to the actual training data set resulted in an RMSE of 1.34, MARE of 0.84, and MR of 0.09. The corresponding values during testing were 0.84, 1.86, and 0.02, respectively.

$$K_s = 1.36 - D_b - \text{clay} \quad [5]$$

$$K_s = (0.59 \text{ sand}) + (1 - D_b)\text{silt} \quad [6]$$

Uncertainty in the PTF can result from data, model parameters, and model structure. Data uncertainty stems from natural uncertainty and variability. For calibrated models, the model parameter uncertainty also incorporates the effects of data uncertainties because of the curve-fitting property of the calibration process. Model structure uncertainty arises from the inability to truly represent the predictand–predictor relationship. These aspects of uncertainty have led to the concept of equifinality (Beven and Freer, 2001), which argues that there are many different model structures and many different parameter sets within a chosen model structure that may be behavioral or acceptable in reproducing the observed behavior of a complex environmental system. In most of the PTF literature, NN-based PTFs usually account for the model parameter uncertainty by including the parametric bootstrap method. In this case, the configuration of the NN models remains the same for each of the bootstrapped data sets, with the connection weights and bias alone varying depending on the samples present in each of the resampled data sets. In this study, however, both the model parameter uncertainty and model structure uncertainty are accounted for by combining GP and the parametric bootstrap method. In this case, for each of the bootstrapped data sets, both the model structure and the model parameters are optimized in unison by GP. Hence, the value of uncertainty of the NN(BR) model (Table 4) indicates the model parameter uncertainty only, while the GP models' uncertainty (Table 4) indicates both the model parameter and model structure uncertainty.

For the GP models, the relative contribution of the model parameter uncertainty and the model structure uncertainty to the total uncertainty reported in Table 4 was also examined in this study. To accomplish this, instead of simultaneously evolving both the model structure and the model parameters using the self-organizing nature of GP, by predefining the model structure and optimizing its corresponding model parameters alone, the uncertainty due to model parameters could be determined. By comparing this uncertainty with the total uncertainty (model parameter and model structure) reported in Table 4, it was possible to determine the relative contributions of the model parameter uncertainty and the model structure uncertainty to the total uncertainty. The model structures based on Eq. [5] and [6], which represent the PTFs obtained by the GP(1) and the GP(2) models, respectively, for the original training data set, were chosen to be the representative model structures for all the 50 resampled data sets. Keeping the model structure static, their respective model parameters alone were optimized for each of the 50 resampled data sets. Equation [5] and [6] have three and four parameters, respectively, that need to be optimized. The uncertainty estimate and the performance statistics were calculated as outlined above. When Eq. [5] [based on GP(1)] was used as the representative model structure, during training it resulted in an uncertainty of 0.11, RMSE of 0.85, MARE of 0.78, and MR of 0. The corresponding values during testing were 0.09, 0.93, 2.31, and –0.26, respectively. Comparing these statistics with those of the GP(1) model statistics (Table 4), it can be concluded in general that, keeping the model structure static and optimizing the model parameters alone during both training and testing resulted in less uncertainty but at the expense of higher error statistics. The uncertainty estimates of 0.11 during training and 0.09 during testing are relatively small when compared with the uncertainty estimates of the GP(1) model (Table 4), which resulted in uncertainty estimates of 0.27 and 0.26 during training and testing, respectively. When Eq. [6] [based on GP(2)] was used as the representative model structure, during training it resulted in an uncertainty of 0.09, RMSE of 0.77, MARE of 0.76, and MR of –0.01. The corresponding values during testing were 0.10, 0.87, 1.89, and –0.01, respectively. Comparing these statistics with that of the GP(2) model statistics (Table 4), keeping the model structure static and optimizing the model parameters alone during training resulted in less uncertainty but with higher error statistics. Nevertheless, during testing, compared with the GP(2) model, the performance of the model with static model structure and optimized model parameters resulted in a considerably lower uncertainty estimate and also relatively better error statistics. Hence, in general, it can be stated that, compared with the models with optimized model structure and model parameters [GP(1) and GP(2)], the models with static model structure and optimized model parameters resulted in markedly lower uncertainty estimates. As argued above, since the former models account for both the model parameter and model structure uncertainty, and the latter models account for the model parameter uncertainty alone, it can be concluded that, compared with the model parameter uncertainty, the contribution of model structure uncertainty to the actual uncertainty is more significant. Based on the above analysis, it can be stated that for ensemble modeling of $K_s$ using GP, for each of the resampled data sets, the choice between (i) keeping the model structure static and optimizing the model parameters alone and (ii) self-organizing both the model structure and model parameters should be made considering the kind of uncertainty (model parameter or both model parameter and model structure) that needs to be accounted for in the ensemble modeling of $K_s$.

## CONCLUSIONS

In this study, the utility of GP as a model induction engine for deriving PTFs for estimating $K_s$ as a function of sand, silt, and clay contents and bulk density was explored. Out of the 314 samples derived from the UNSODA, 199 samples were used for calibration and the remaining 114 samples are used for validating the developed models. Two different GP models, GP(1) and GP(2), with different combinations of mathematical operators in the functional set were developed. The GP(1) model uses only additive (+, −) operators as part of the functional set, and the GP(2) model uses both additive and multiplicative (×, /) operators as part of the functional set. This exercise was performed to evaluate the performance of the GP models with varying levels of mathematical operators (complexity) that can be used to define the predictand–predictor relationship. The performance of the GP(1) and GP(2) models were compared with a NN model using a Bayesian-regularization algorithm for training the networks. This algorithm has a better generalization property as it minimizes both the mean sum of squares of the network errors and the mean of the sum of squares of the network weights and bias.

The performance of the models was evaluated in terms of RMSE, MARE, MR, and model uncertainty. The uncertainty of the models was evaluated by combining the models with the nonparametric bootstrap method. Fifty different bootstrapped data sets were created by statistical resampling of the training data set, and the NN and GP models were applied to these bootstrapped data sets, from which the average error estimates and uncertainty of the model predictions were evaluated. Results from the study indicate that the GP(2) model resulted in the least MRE and MR estimates, signifying less bias attached to the model. The relatively better performance of the GP models, compared with the NN model, may be attributed to the self-organizing nature of the model, in which both the model structure and parameters are evolved simultaneously during the estimation process, learning the features of the data set. Increasing the number of mathematical operators in the functional set of the GP models has been found to lead to a better fit of the function, but at an increasing level of uncertainty. This indicates that, if it is not unfeasible, it is at least difficult to achieve both higher prediction accuracy and less uncertainty in tandem. The results presented in this study, in general, indicate that GP is a promising tool for developing PTFs for estimating $K_s$.

Analyzing the optimal equations identified by the GP models for each of the bootstrapped data sets, $D_b$ is the most significant input in characterizing the $K_s$ for both the GP(1) and GP(2) models. In the case of the GP(1) model, in the order of importance, $D_b$ is followed by clay, sand, and silt contents. For the GP(2) model, however, in the order of importance, $D_b$ is followed by sand, silt, and clay contents. These results indicate that the most relevant input variables for estimating $K_s$ are not unique, rather depend on the mathematical operators that are used as part of the functional set of the GP paradigm. Also, it has been shown that the uncertainty reported by the NN(BR) model is only the model parameter uncertainty, whereas the uncertainty reported by the GP models include both the model parameter and model structure uncertainty. Examining the relative contribution of model structure uncertainty and model parameter uncertainty to the total uncertainty estimated by the GP models, it has been shown that, compared with the model parameter uncertainty, the uncertainty due to the model structure dominates the total uncertainty of the GP models. The study reported here is a first step to evaluate the utility of GP in developing PTFs. The results of the study need to be further explored by extending the GP models to different data sets with different functional and terminal sets. In this regard, analyzing the performance of grammar-guided GP in developing PTFs would be of particular interest.

## ACKNOWLEDGMENTS

Funding for this project was provided by the Natural Science and Engineering Research Council of Canada (NSERC) to A. Elshorbagy, and B.C. Si. Financial support from the Univ. of Saskatchewan through the Departmental Scholarship Program is also gratefully acknowledged.

## REFERENCES

Babovic, V., and M.B. Abbott. 1997. Evolution of equation from hydraulic data. Part I: Theory. J. Hydraul. Res. 35:1–14.

Babovic, V., and M. Keijzer. 2000. Genetic programming as model induction engine. J. Hydroinform. 2:35–60.

Beven, K., and J. Freer. 2001. Equifinality, data assimilation, and uncertainty estimation in mechanistic modelling of complex environmental systems using the GLUE methodology. J. Hydrol. 249:11–29.

Bouma, J. 1989. Using soil survey data for quantitative land evaluation. Adv. Soil Sci. 9:177–213.

Cosby, B.J., G.M. Hornberger, R.B. Clapp, and T.R. Ginn. 1984. A statistical exploration of the relationships of soil moisture characteristics to the physical properties of soils. Water Resour. Res. 20:682–690.

Coulibaly, P. 2004. Downscaling daily extreme temperatures with genetic programming. Geophys. Res. Lett. 31:L16203, doi:10.1029/2004GL020075.

Demuth, H., and M. Beale. 2001. Neural network toolbox learning. For use with MATLAB. The Math Works, Natick, MA.

Efron, B., and T.J. Tibshirani. 1993. An introduction to bootstrap. Chapman and Hall, New York.

Fogel, L.J., A.J. Owens, and M.J. Walsh. 1966. Artificial intelligence through simulated evolution. John Wiley & Sons, New York.

Folino, G., C. Pizzuti, and G. Spezzano. 2006. GP ensembles for large-scale data classification. IEEE Trans. Evolut. Comput. 10:604–616.

Gupta, S.C., and W.E. Larson. 1979. Estimating soil water characteristic from particle size distribution, organic matter percent, and bulk density. Water Resour. Res. 15:1633–1635.

Haykin, S. 1999. Neural networks: A comprehensive foundation. 2nd ed. MacMillan, New York.

Holland, J.H. 1975. Adaptation in natural and artificial systems. MIT Press, Cambridge, MA.

Hong, Y.S., and M.R. Rosen. 2002. Identification of an urban fractured-rock aquifer dynamics using an evolutionary self-organizing modelling. J. Hydrol. 259:89–104.

Hong, Y.S., P.A. White, and D.M. Scott. 2005. Automatic rainfall recharge model induction by evolutionary computational intelligence. Water Resour. Res. 41:W08422, doi:10.1029/2004WR003577.

Iba, H. 1999. Bagging, boosting, and bloating in genetic programming. p. 1069–1076. *In* W. Banzhaf et al. (ed.) Proc. Genetic and Evolutionary Computation Conf., Orlando, FL. 13–17 July 1999. Vol. 2. Morgan Kaufmann, San Francisco.

Keijzer, M., and V. Babovic. 2000. Genetic programming, ensemble methods and the bias/variance tradeoff: Introductory investigations. p. 76–90. *In* R. Poli et al. (ed.) Proc. EuroGP 2000, Edinburgh, Scotland. Vol. 1802. 15–16 Apr. 2000. Springer-Verlag, Berlin.

Khu, S.T., S.Y. Liong, V. Babovic, H. Madsen, and N. Muttil. 2001. Genetic programming and its application in real-time runoff forecasting. J. Am. Water Resour. Assoc. 8:201–220.

Klute, A. 1986. Methods of soil analysis. Part 1: Physical and mineralogical methods. SSSA Book Ser. 5. SSSA, Madison, WI.

Koza, J.R. 1992.Genetic programming: On the programming of computers by means of natural selection. The MIT Press, Cambridge, MA.

Leij, F.J., W.J. Alves, M.Th. van Genuchten, and J.R. Williams. 1996.

Unsaturated soil hydraulic database, UNSODA 1.0 user's manual. Rep. EPA/600/R-96/095. U.S. Environ. Protection Agency, Ada, OK.

Leij, F.J., M.G. Schaap, and L.M. Arya. 2002. Water retention and storage: Indirect methods. p. 1009–1045. *In* J.H. Dane and G.C. Topp (ed.) Methods of soil analysis. Part 4: Physical methods. SSSA Book Ser. 5. SSSA, Madison, WI.

Makkeasorn, A., N.B. Chang, M. Beaman, C. Wyatt, and C. Slater. 2006. Soil moisture estimation in a semiarid watershed using RADARSAT-1 satellite imagery and genetic programming. Water Resour. Res. 42: W09401, doi:10.1029/2005WR004033.

McBratney, A.B., B. Minasny, S.R. Cattle, and R.W. Vervoort. 2002. From pedotransfer functions to soil inference systems. Geoderma 109:41–73.

McKenzie, N.J., and D.W. Jacquier. 1997. Improving the field estimation of saturated hydraulic conductivity in soil survey. Aust. J. Soil Res. 35:803–825.

Minasny, B., A.B. McBratney, and K.L. Bristow. 1999. Comparison of different approaches to the development of pedotransfer functions for water retention curves. Geoderma 93:225–253.

Nemes, A., W.J. Rawls, and Ya. Pachepsky. 2005. Influence of organic matter on the estimation of saturated hydraulic conductivity. Soil Sci. Soc. Am. J. 69:1330–1337.

Pachepsky, Ya., and W.J. Rawls. 1999. Accuracy and reliability of pedotransfer functions as affected by grouping soils. Soil Sci. Soc. Am. J. 63:1748–1757.

Pachepsky, Ya., W.J. Rawls, D. Gimenez, and J.P.C. Watt. 1998. Use of soil penetration resistance and group method of data handling to improve soil water retention estimates. Soil Tillage Res. 49:117–126.

Pachepsky, Ya., D.J. Timlin, and G. Varallyay. 1996. Artificial neural networks to estimate soil water retention from easily measurable data. Soil Sci. Soc. Am. J. 60:727–773.

Parasuraman, K., A. Elshorbagy, and S.K. Carey. 2007. Modeling the dynamics of the evapotranspiration process using genetic programming. Hydrol. Sci. J. 52:563–578.

Poli, R. 2003. A simple but theoretically-motivated method to control bloat in genetic programming. p. 204–217. *In* C. Ryan et al. (ed.) Proc. EuroGP 2003, Essex, UK. 14–16 Apr. 2003. Springer, Berlin.

Rawls, W.J., and D.L. Brakensiek. 1983. A procedure to predict Green and Ampt infiltration parameters. p. 102–112. *In* Advances in infiltration: Proc. Natl. Conf. on Advances in Infiltration, Chicago. 12–13 Dec. 1983. ASAE Publ. 83-11. Am. Soc. Agric. Eng., St. Joseph, MI.

Rawls, W.J., T.J. Gish, and D.L. Brakensiek. 1991. Estimating soil water retention from soil physical properties and characteristics. Adv. Soil Sci. 9:213–234.

Savic, D.A., G.A. Walters, and J.W. Davidson. 1999. Genetic programming approach to rainfall–runoff modelling. Water Resour. Manage. 13:219–231.

Saxton, K.E., W.J. Rawls, J.S. Romberger, and R.I. Papendick. 1986. Estimating generalized soil-water characteristics from texture. Soil Sci. Soc. Am. J. 50:1031–1036.

Schaap, M.G., and W. Bouten. 1996. Modeling water retention curves of sandy soils using neural networks. Water Resour. Res. 32:3033–3040.

Schaap, M.G., and F.J. Leij. 1998. Database-related accuracy and uncertainty of pedotransfer functions. Soil Sci. 163:765–779.

Schaap, M.G., F.J. Leij, and M.Th. van Genuchten. 2001. ROSETTA: A computer program for estimating soil hydraulic parameters with hierarchical pedotransfer functions. Soil Sci. 163:765–779.

Schwefel, H.P. 1981. Numerical optimization of computer models. John Wiley & Sons, New York.

Silva, S. 2007. GPLAB: A genetic programming toolbox for MATLAB. Available at http://gplab.sourceforge.net (verified 25 July 2007). S. Silva, Coimbra, Portugal.

Silva, S., and E. Costa. 2004. Dynamic limits for bloat control. p. 666–677. *In* K. Deb et al. (ed.) Proc. Genetic and Evolutionary Computation—GECCO 2004: Genetic and Evolutionary Computation Conf., Seattle, WA. 26–30 June 2004. Springer-Verlag, Berlin.

Soule, T., and J.A. Foster. 1999. Effects of code growth and parsimony pressure on populations in genetic programming. Evolut. Comput. 6:293–309.

Tamarai, S., J.H.M. Wösten, and J.C. Ruiz-Suarez. 1998. Testing an artificial neural network for predicting soil hydraulic conductivity. Soil Sci. Soc. Am. J. 60:1732–1741.

Ungaro, F., C. Calzolari, and E. Busoni. 2005. Development of pedotransfer functions using a group method of data handling for the soil of the Pianura Padano-Veneta region of North Italy: Water retention properties. Geoderma 124:293–317.

van Genuchten, M.Th., F.J. Leij, and L.J. Lund. 1992. On estimating the hydraulic properties of unsaturated soils. p. 1–14. *In* M.Th. van Genuchten et al. (ed.) Indirect methods for estimating the hydraulic properties of unsaturated soils. Proc. Int. Worksh., Riverside, CA. 11–13 Oct. 1989. Univ. of California, Riverside.

Vereecken, H., J. Maes, and J. Feyen. 1990. Estimating unsaturated hydraulic conductivity from easily measured soil properties. Soil Sci. 149:1–12.

Whigham, P.A., and P.F. Crapper. 2001. Modelling rainfall–runoff using genetic programming. Math. Comput. Modell. 33:707–721.

Wösten, J.H.M. 1990. Use of soil survey data to improve simulation of water movement in soils. Ph.D. diss. Univ. of Wageningen, Wageningen, the Netherlands.

Wösten, J.H.M., Ya. Pachepsky, and W.J. Rawls. 2001. Pedotransfer functions: Bridging the gap between available basic soil data and missing soil hydraulic characteristics. J. Hydrol. 251:123–150.