18

A Matlab Genetic Programming Approach to Topographic Mesh Surface Generation

Katya Rodríguez V.[1] and Rosalva Mendoza R.[2]
[1]*National Autonomous University of México, Research in Applied Mathematics and Systems Institute;*
[2]*National Autonomous University of México, Engineering Institute;*
México

1. Introduction

The problem of surface approximation by means of soft mathematical functions is a relevant topic in Hydrology. The generation of these functions allows solving implicitly some of the most important calculation in order to predict the behavior of the hydrological basin. Thus, this work proposes the use of an Evolutionary Algorithm (EA) (Bäck, 1996) to generate 3-D mesh surface from a set of topographic data. In literature, there are only few existing works about the use of Evolutionary Algorithms (EAs) applied to the reconstruction of topographic surfaces, most of them are based on Genetic Algorithms (GAs) (Holland, 1975; Goldberg, 1989) as an approximation polynomial parameter estimator. Thus, this paper introduces a Genetic Programming (GP) approach whose aim is to obtain a mathematical function that allows a compact representation of the surface of the topographic information. This surface generation problem is then formulated as symbolic regression. The use of EAs, specifically GP (Koza, 1990; Banzhaf et al., 1998), constitute a promise alternative for the traditional interpolation techniques that employ approximation polynomials, due to GP integrates in a natural way the common non-linearities present in complex interpolation problems. This proposal is then applied to a set of topographic data corresponding to the Mezcalapa River zone, which is the local name of the Grijalva River located at the southeast of the Mexican Republic and it is one of the most important rivers due to its flow and generation of electric energy.

The GP algorithm is programmed in MATLAB® and the results produced by means of this GP approach give indication of a significant improvement in terms of the quality of the approximation in relation to the results obtained by means of approximation polynomials method applied to this region. In the following section a brief review of some works on mathematical modeling applied to Civil and Hydraulic Engineering are detailed. After that, description of genetic programming algorithm and its implementation in MATLAB are presented. The application of this evolutionary method to evolve mathematical models in order to construct topographic surface is presented. Finally results and conclusions are drawn.

2. Previous works

The literature related to the application of EAs to the problem of topographic surface generation is sparse. Some related papers, in terms of mathematical modeling, are the one by Fujiwara and Sawai (1999), where the use of EAs is proposed to optimize 3-D facial images. The problem is formulated as the selection of n points from a total set of N points that constitutes the original image; but, by selecting only n points (n<N), the image can be reconstructed with a good approximation to the original one.

Huang and Ho (2003) proposed a genetic algorithm again to select the n points where n<N and N is the number of points of the original image in order to approximate a surface. In this work, a crossover operator named OAX (Orthogonal Arrays Crossover) was introduced. Kodoma et al (2005) proposed the use of hybrid algorithms by combining matrix-based representation genetic algorithm and a simulated annealing algorithm to reduce the computing time; however, performance presented by this hybrid algorithm and the original proposal based only on genetic algorithms are similar.

The work by Gálvez et al (2007) concerns to the problem of curve and surface fitting. They focus on the case of 3D point clouds fitted with Bézier curves and surfaces. Two Artificial Intelligence (AI) techniques are considered in this paper: the use of GAs and the functional networks scheme. Wagner et al (2007) present the ability of a state-of-the-art multi-objective EA to be successfully integrated in surface reconstruction software.

Goinski (2008) proposes a novel technique for surface reconstruction from a points cloud in 3D. The aim is to combine EAs with a recursive subdivision scheme. Paszkowicz (2009) reports recent use of GAs in various domains related to materials science, solid state physics and chemistry, crystallography, biology, and engineering. Shape and topology optimization is one of the applications reported in the field of engineering.

Periaux et al (2009) compare the performances of two different optimization techniques for solving inverse problems; the first one deals with the Hierarchical Asynchronous Parallel Evolutionary Algorithms software (HAPEA) and the second is implemented with a game strategy named Nash-EA.

In the context of GP, this has been used to solve symbolic regression problems. In Koza (1990), GP was used to generate a program to represent the colors of an image as a two dimensions array. Keller et al. (1999) proposed the use of GP to reconstruct surfaces of prototype pieces of industrial equipment. In this case, the objective is related to the works by Kodoma et al. (2005), Fujiwara and Sawai (1999) and Huang and Ho (2003).

The generation of mathematical function by means of symbolic regression has been widely studied in the GP field, as shown by the following papers by Keijzer (2003), Streeter and Becker (2001), Iba and Nikolaev (2001), Parasuraman et al (2007), Miller and Harding (2008), Baumes et al (2009), Barmpalexis et al (2011), among others. It seems that the representation of surface plays an important role in a variety of disciplines including aided-design, computer vision, graphic computation and geographical signal and image processing. Thus, evolutionary algorithms, in particular genetic programming, have been promising areas to these applications.

In the field of hydraulic engineering, the problem of approximating a soft mathematical function to a set of topographic data was considered (Mendoza et al., 1996). It required of getting an explicit mathematical expression and then the derivative of it in order to construct a model of coordinates curves adjusted to free surface. The problem was initially solved by representing the topographic elevation as a function of the dependent variables x-

y, which was approximated to a third order Taylor series (Arfken, 1980). Coefficients were adjusted by a Least Square Algorithm. Obtained results were, in general, acceptable. However, there were a significant number of points where the proposed method did not provide appropriate estimation (Mendoza et al., 1996). These points corresponded to regions of peaks and valleys surrounded by very different topographic points.
In order to improve the quality of the approximation, Mendoza (2002) proposed the use of algorithms belonging to the EAs field. As it is known, EAs are optimization techniques based on the concepts of natural selection and evolution. This work is then focused on the use of one of these evolutionary techniques, Genetic Programming (Banzhaf, et al., 1998).
In the present work, representing a topographic surface by means of a mathematical function is proposed and the problem is formulated as a symbolic regression using traditional genetic programming. A GP Toolbox for MATLAB is then developed and detailed in next sections.

3. Genetic programming

Nature has provided the inspiration for the design of computational algorithms in a variety of ways. These computational processes have taken two main natural systems as their basis that is the *brain* and the *genetic evolution theory*. EAs are one of these computational models and are proposed in this work for modelling topographic surface.
EAs, also known as Evolutionary Computation (EC), use computational models of evolutionary processes in the design and implementation of computer-based problem solving. A general definition and classification of these evolutionary techniques is given in Bäck (1996). He defines an EA as a search and optimisation algorithm, inspired by the process of natural evolution, which maintains a population of structures that evolve according to rules of selection and other operators such as recombination and mutation. Here, the structure of all evolution-based algorithms is shown in Figure 1.
The adaptive search algorithm called Genetic Programming (GP) was designed by Koza (1990). GP is an evolution-based search model that is a subclass of the popular GAs [Holland, 1975; Goldberg, 1989]. Koza introduced a more complex representation based on *computer programs*. Although finding algorithms or programs is more difficult than finding a single solution, it is more useful since generalised solutions work for an entire class of tasks.

```
PROGRAM Evolution-Based Algorithm
        t = 0
        Create Initial Population P(t)
        Evaluate Initial Population P(t)
        While (not termination_criterion) do
                t = t + 1
                Select Individuals for Reproduction P(t) from P(t-1)
                Alter P(t)
                Evaluate New Population P(t)
        end
```

Fig. 1. Evolution-based algorithm

To illustrate the hierarchical encoding used for GP, Figure 2 gives a simple example where the operations +, -, *, %, sin, cos, exp, sqrtp belong to the function set and the variables X and Y and the set of constants pi, 1, 2, 3, 4, and 5, constitute the terminal set. It is important to mention that division has been assigned the symbol "%"; this means protected division in order to avoid infinity results producing by operations like dividing by zero. It is also de case for square root operation where *sqrtp* function takes the absolute value of its argument. Note in Figure 2 that the parse tree is also equivalent to the prefix expression, as well as to the mathematical function and the MATLAB function.

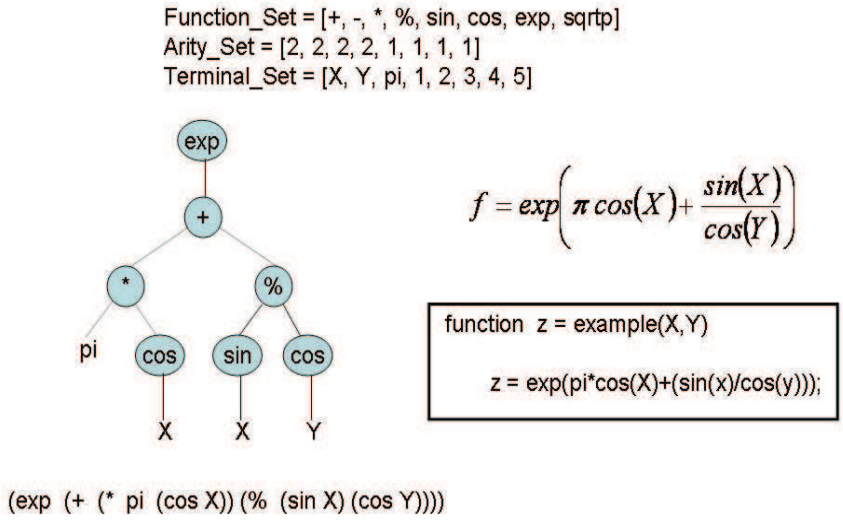


Fig. 2. A tree-based individual encoding and its equivalent representation in prefix notation, MATLAB program and mathematical function

3.1 Genetic programming operators

As for the conventional GA, reproduction and crossover are considered the main genetic operators, mutation being a secondary operator.

3.1.1 Reproduction

Reproduction in GP works in a similar way to that in a GA, being one of the foundations of the survival of the fittest. It is an asexual operator that selects an individual structure according to some selection method based on the fitness measures. The selected individual is then copied without any alteration to the new population.

3.1.2 Crossover

One of the main differences between GP and the traditional implementation of GA is the fact that GP crossover does not preserve any kind of context in the chromosome. This is due to the fact that the standard crossover defined by Koza (1990) exchanges subtrees which are chosen at random in both parents. Koza has pointed out that random subtree crossover maintains diversity in the population because crossing two identical structures, generally,

will create different offspring. This is because the crossover points are, in general, different in the two parents.
Crossover works by first selecting a pair of structures from the current population. Then, a node rooted from each parent is randomly selected. These nodes become the roots for the sub-structures lying below the crossover point. In the next step, the sub-structures are exchanged between the parents producing two new structures which are usually of different sizes to their parents. Figure 4 illustrates the crossover operation over a function set and terminal set defined as for Figure 2.
Note that for GP-crossover, the crossover point can be either a terminal or an internal point. If the crossover points in both parents are internal nodes, this means that function nodes are chosen as roots for the substructures to be exchanged. A second case of crossover occurs when a terminal node and an internal node, as the root of the substructure, are chosen in the first and second parents, respectively. When an internal node is selected, the number of arguments taken by the associated function must be considered in order to exchange a valid substructure.
A third case of crossover occurs when the crossover node is a terminal in both parents. In this case, the size and shape of the parents do not modify but the arguments of the two functions are swapped.

3.1.3 Mutation

Mutation is considered a secondary operator. It operates by randomly selecting a node, which can be either a terminal or internal point, and replacing the associated sub-structure with a randomly generated subtree up to a maximum size. A Maximum Mutation Size (MMS) parameter is introduced which is different from the maximum tree size parameter, MS.
In a conventional GA, the mutation operator introduces a certain degree of diversity into the population which is being beneficial. In contrast, the GP-crossover operation is the mechanism for diversification in the GP population. This fact is the justification given by Koza (1990) for using a 0% mutation probability. Hence, convergence of the population is unlikely in genetic programming.
Nevertheless, Angeline (1996) has described a set of mutation operators named: grow, shrink, cycle, switch and numerical terminal mutation. These mutation schemes are defined as follows:
Grow exchanges a randomly selected terminal point with a randomly generated subtree.
Shrink substitutes a selected subtree with a single terminal.
Cycle replaces a selected internal (a function) node by another function.
Switch selects two subtrees from the same parent and then switches their positions.
Numerical Terminal selects a single real-valued numerical (not a variable) terminal and adds to it Gaussian noise with a particular variance.

4. A GA toolbox for MATLAB

MATLAB is a high-level language and possesses a variety of already implemented functions, where problems can be easily coded in *m*-files. These facts make the programming of a GA in MATLAB an easy process.
The Genetic Algorithm Toolbox uses MATLAB matrix functions to build a set of routines for implementing a wide range of genetic algorithm methods (Chipperfield et al., 1994).

MATLAB essentially supports only one data type, a rectangular matrix of real or complex numeric elements. Thus, four data structures are defined for the implementation of the GA Toolbox developed by Chipperfield et al. (1994):

1. Chromosomes: It is a matrix of size *Nind*Lind*, where *Nind* is the population size and *Lind* is the length of the strings (rows of chromosomes) representing individuals.
2. Phenotypes: This data structure corresponds to the decision variables matrix and is obtained by applying a mapping process (decoding) from the chromosome representation into the decision variable space. Thus, this structure is a matrix of size *Nind*Nvar*, where *Nvar* is the number of variables that are encoding into chromosomes and each row corresponds to an individual's phenotype.
3. Objective Function: It is used to evaluate the performance of each individual (first chromosomes and after decoding phenotypes) of the population in the problem domain. This can be scalar (for mono-objective GA), or a matrix in the case of a multiobjective GA. Then, this data structure is a matrix of size *Nind*Nobj*, where *Nobj* is the number of objective (*Nobj*=1 for single objective problems).
4. Fitness Values: These are derived from the objective function by means of a fitness assignment function (scaling or ranking). Fitness values are defined in *Nind*1* matrix and are non-negative scalars.

5. GP structures in MATLAB

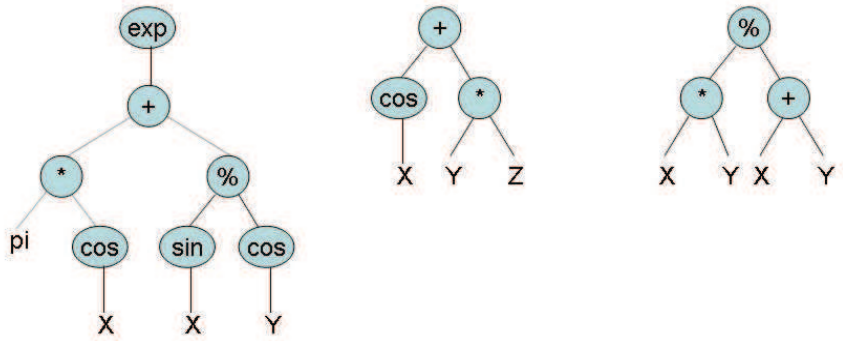
From Figure 2, it is seen that the parse-tree has an equivalent prefix notation (a LISP structure); thus, this codification is adopted in order to implement genetic programming in MATLAB. Then, a population is defined by a *Nind*Maxnodes* matrix whose content is initially zeros. By means of this encoding, the initial population matrix is:

$$pop = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Then, random parse-trees are generated taking random values from the primitive sets. It is important to mention that the root node is always defined as a function node and the *arity* (number of input arguments of each function) is taking into account in order to generate syntactically valid structures. An example is presented as follows:

1. The root node has been randomly chosen from the function set. For this example, this function is *"exp"*.
2. This function takes one argument, thus another node is randomly selected from the function or terminal sets. Here, a function node was chosen ("+"); the *"exp"* function has its argument but the "+" function takes two arguments. Arguments for the "+" must be randomly selected.
3. This process continues until terminals are selected and the expression cannot increase its size and $(Nodes_{remain} < (Maxnodes - Nodes_{curr}))$, where $Nodes_{remain}$ means the nodes needed in the structure in order to produce a syntactically valid expression and $Nodes_{curr}$ is the number of nodes selected at the moment to conform an expression that is still incomplete. In the case where $(Nodes_{remain} = (Maxnodes - Nodes_{curr}))$, only terminal nodes are selected for a syntactically valid expression and the process concludes.

In Figure 3, it is then showed some parse-trees and their equivalent prefix notation into a population matrix.



$$pop = \begin{bmatrix} \exp & + & * & pi & \cos & X & \% & \sin & X & \cos & Y & 0 \\ + & \cos & X & * & Y & Z & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ \% & * & X & Y & + & X & Y & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 3. Initial GP population in MATLAB

But, in order to facilitate the use of MATLAB to manipulate matrix values of the same type, an identifier is considered for each primitive (function or terminal) as shown in Table 1.

Integer Indentifier	Primitive Value
1	+
2	-
3	*
4	%
5	exp
6	cos
7	sin
1000	random constant
1001	X
1002	Y
1003	Z
1004	pi

Table 1. ID for GP Primitive Sets

Then, matrix presented in Figure 3 is transformed to the following matrix:

$$pop = \begin{bmatrix} 5 & 1 & 3 & 1004 & 6 & 1001 & 4 & 7 & 1001 & 6 & 1002 & 0 \\ 1 & 6 & 1001 & 3 & 1002 & 1003 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 4 & 3 & 1001 & 1002 & 1 & 1001 & 1002 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Genetic operators are applied on these prefix representations by selecting a random crossover node in the first parent in the interval [1, $MaxNodesPop_i$] and a random node in the second parent between [1, $MaxNodesPop_{i+1}$], where *MaxNodesPop* is a *Nind* column vector containing information related to the number of nodes of each individual into the population. This vector is updated each time crossover or mutation is performed. After that, associate expression to these selected nodes are taken and exchanged creating two new individuals.

In the case of mutation, again a node is randomly selected in the range [1, $MaxNodesPop_i$] and the syntactically valid associate sub-expression is eliminated and a new sub-expression is inserted. This is created from the primitive sets and using the routine of creating initial population.

If a new individual generated by means of crossover or mutation exceeds the allowed maximum size (maximum number of nodes), a new randomly selected node is taken in the range defined by the position of the previously selected node and $MaxNodePop_i$. This fact avoids that individuals grow rapidly causing bloat[1].

An example of crossover on the MATLAB GP representation is exemplified in Figure 4. Previous **pop** matrix is considered in this example. It is important to mention that the individual selection mechanism can be any method (roulette wheel, tournament, stochastic universal selection) and it is borrowed from the GA Toolbox, as well as the fitness assignment mechanism.

5.1 Function evaluation

In order to evaluate each individual into the population, a bottom-up parser must be constructed as a MATLAB function. Based on primitive set defined in Table 1 and the last individual of **pop** matrix from Figure 3, this program is evaluate as illustrated in Figure 5 considering that the variables X and Y take the following values $[-3, -2, -1, 0, 1, 2, 3]^T$ and $[0, 1, 2, 3, 4, 5, 6]^T$, respectively. The output of the evaluated individual (information at the root node) is a vector of size *Nx1*, where *N* is the number of data points, in this simple example *N* is equal 7. Thus, the objective function is defined as the minimization of the estimated mean quadratic error produced between the output of each program (individual) and the real values of the topographic elevation. This is expressed in the following equation:

$$f_i = \frac{1}{N} \sum_{j=1}^{N} \left| z_j - z'_{ij} \right|^2$$

where f_i is the objective value of the *i-th* individual, *z* is the vector of measured topographic elevations, *z'* is the vector of estimated topographic elevations for *N* recorded coordinates. The objective value is scalar and the fitness assignment mechanism described in Chipperfield et al. (1994) can be straightforward applied. Observe that for the GP Toolbox

[1] Bloat is the rapid growth of programs produced by genetic programming.

only three data structures must be defined: **pop**, a *Nind*MaxNodes* matrix; **objective value**, a *Nind* column vector; and a **fitness value**, a *Nind* column vector.

$$pop = \begin{bmatrix} 5 & 1 & 3 & 1004 & 6 & 1001 & 4 & 7 & 1001 & 6 & 1002 & 0 \\ 1 & 6 & 1001 & 3 & 1002 & 1003 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 4 & 3 & 1001 & 1002 & 1 & 1001 & 1002 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$pop = \begin{bmatrix} 5 & 1 & 1002 & 4 & 7 & 1001 & 6 & 1002 & 0 & 0 & 0 & 0 \\ 1 & 6 & 1001 & 3 & 1002 & 1003 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 4 & 3 & 1001 & 3 & 1004 & 6 & 1001 & 1 & 1002 & 1002 & 0 & 0 \end{bmatrix}$$

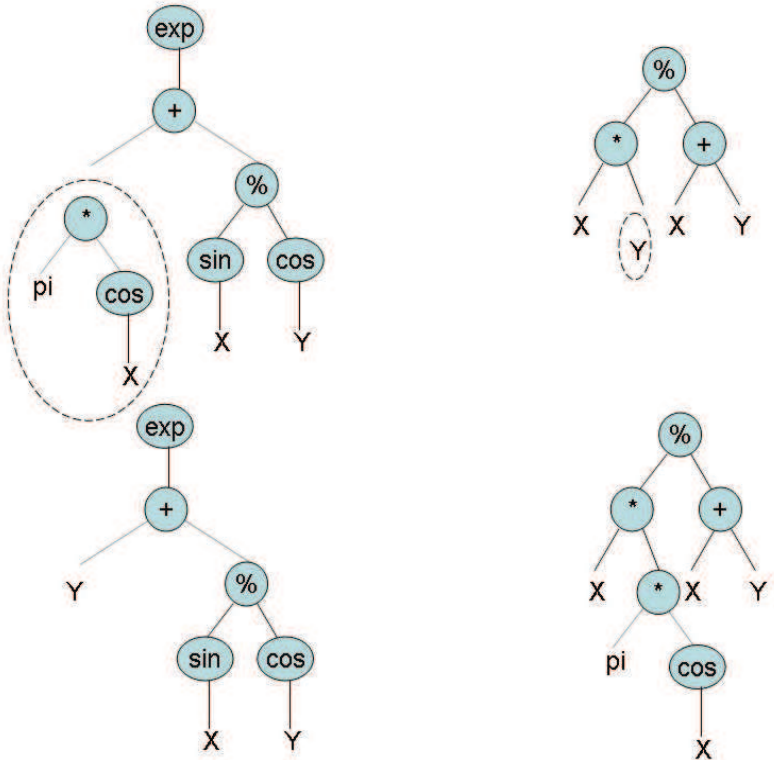


Fig. 4. GP toolbox crossover mechanism

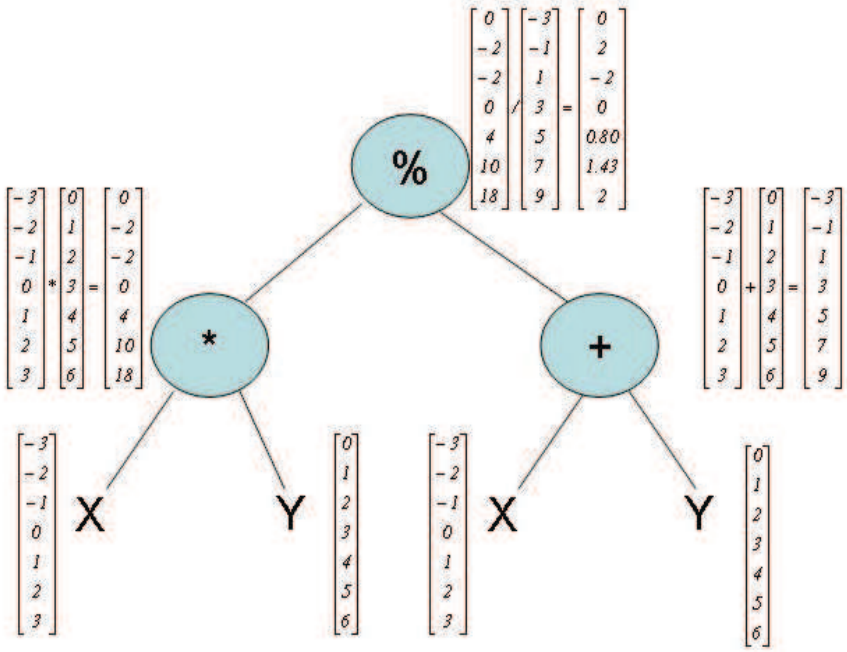


Fig. 5. A bottom-up parser to evaluate GP individuals

6. Estimation of topographic surface by means of GP toolbox in MATLAB

In this section, the MATLAB GP toolbox is applied to model and estimate the topographic elevation of the region shown in Figure 6. The number of available topographic data was 1600 points corresponding to the Mezcalapa river zone located at the southeast of the Mexican Republic. In order to apply the evolutionary methods, the following considerations were taken into account:

a. The function set was composed of the four basic arithmetic operators, trigonometric functions (sine and cosine) and the square root *sqrt* function. Thus, the arity set was defined as {2, 2, 2, 2, 1, 1, 1}, the arithmetic functions take two input arguments and the remaining functions take one input argument.
b. The terminal set consisted of the independent variables (coordinates) x and y, and the ephemeral random constants in the range [-1, 1].
c. The termination criterion was set as the maximum number of generations.
d. In order to evaluate the performance of each individual into the population, estimate mean squared error between the topographic elevation obtained by the individual and the known elevation z was used.
e. The selection mechanism used in these experiments was tournament selection with tournament of size 3.
f. The population was composed of 100 individuals of a maximum size of 256 nodes.
g. Probabilities of crossover and mutation were set to 0.95 and 0.05, respectively.
h. Finally, ten independent runs were carried out for each sub-region.

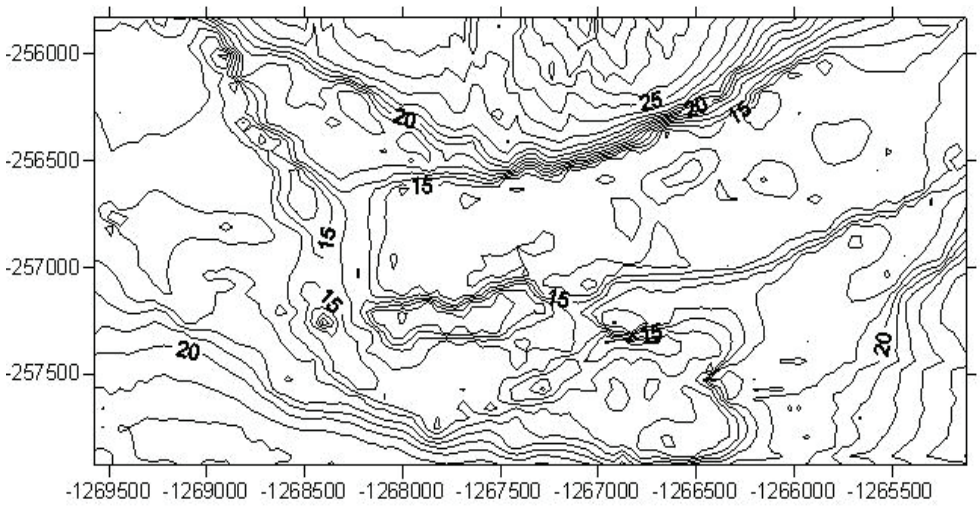


Fig. 6. Mezcalapa river zone, southeast of Mexican Republic

7. Results analysis

The strategy followed in order to reproduce the topography of The Mezcalapa fork River was to divide into ten regions the total area; each one of 160 triples of points with coordinates (x_i, y_i, z_i). In order to avoid numerical noise a constant value in x and y coordinates was added; then, the wireframe map of the total area is shown in Figure 7. Figure 8 shows a wireframe map of the same area reconstructed with the estimated values of the topographic level; comparing the results, they show that the map based on the estimated values is softer, reproduces well the peaks and the valleys but it does not reach the values they present.

The real topography is more rugged and steep; in general, the estimated values of the topographic height fall short in the values of the peaks and valleys, leading to smooth the values of these. Perhaps the most evidence of this softening is the upper right of the region, the real topography exhibits a series of peaks, which show vaguely in the topography generated with the estimated values. In general the border values are well reproduced, but the extreme internal values (peaks or valleys) are the ones with the information surrounding do not have a good estimate.

In general, when the estimate is good the calculated value is almost the same as the measured, however if the information does not help the estimation errors are large (over one meter) that future studies will try to improve.

The analysis of the results shows good estimations of the z values but there are some particular areas where it is necessary to refine the set of functions and terminals for better estimations of the value of the topographic level. The average error in the ten areas was 0.70 masl; the maximorum maximum value is in area 1 and is 4.4 masl; minimorum minimum value is in area 7 and it has a value of 0.0096 masl. Table 2 shows for each region, the average, the maximum and the minimum errors.

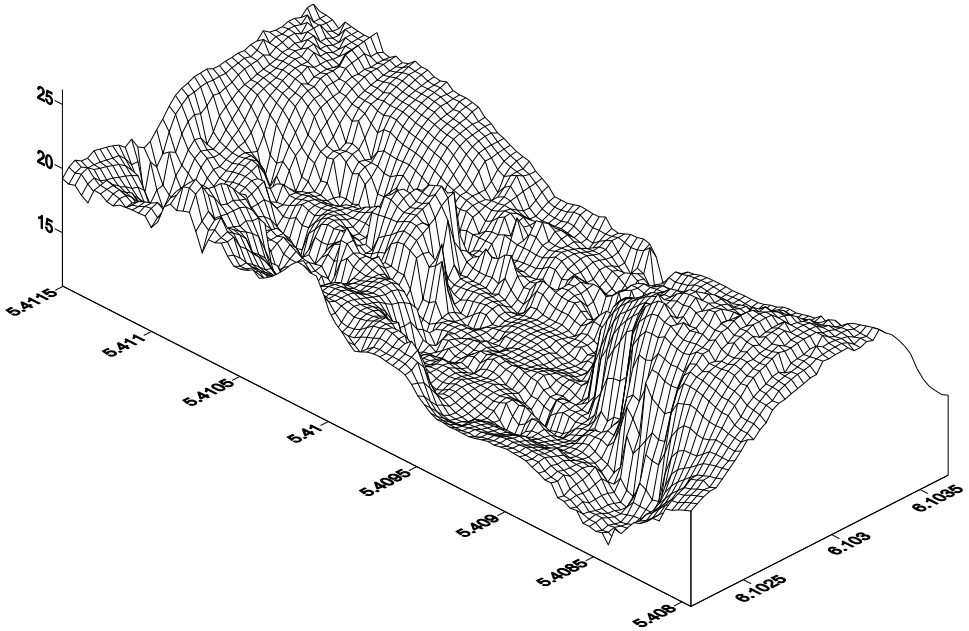


Fig. 7. Total region, real values of the topographic level

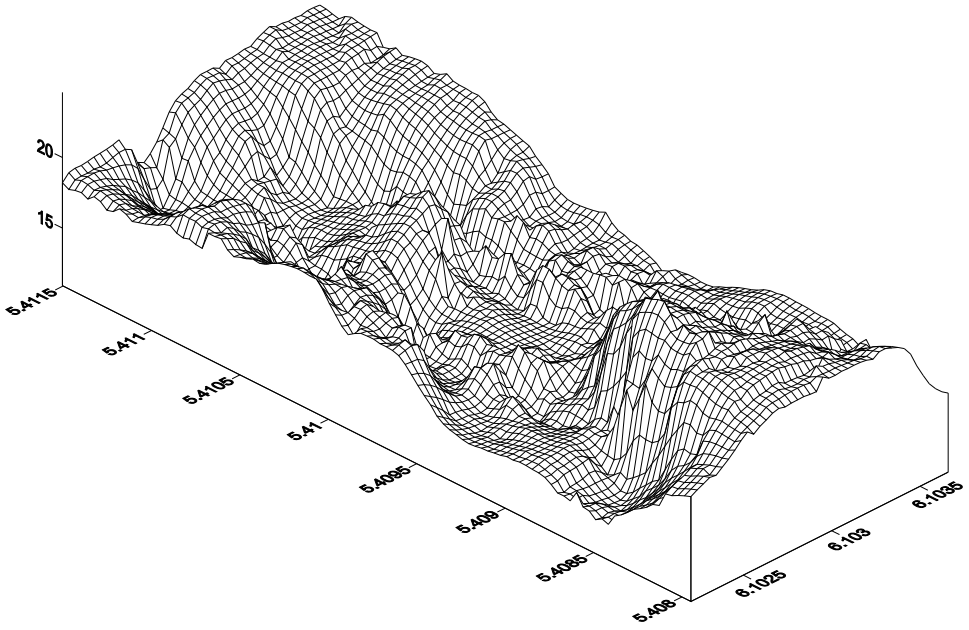


Fig. 8. Wireframe map, estimated values of the topographic level

Region	average error (masl)	max error (masl)	min error (masl)
1	0.64	4.39	0.0023
2	0.72	2.34	0.0014
3	0.68	2.92	0.012
4	0.74	3.26	0.033
5	0.83	2.87	0.0028
6	0.69	3.85	0.0042
7	0.71	3.68	0.00096
8	0.6	3.05	0.0013
9	0.69	2.83	0.0015
10	0.69	3.14	0.003

Table 2. Results for each region

Figure 9 shows the difference in absolute value between real z value and estimated z value. It can see that the vast majority of points are in the area where the error is less than 0.5 masl. However it is also showed that there are points where the estimation error exceeds the value of a meter, the latter leads to recommend a further study to refine the areas which have picks or valleys and normally these values are surrounded by information that does not provide much help for their estimation.

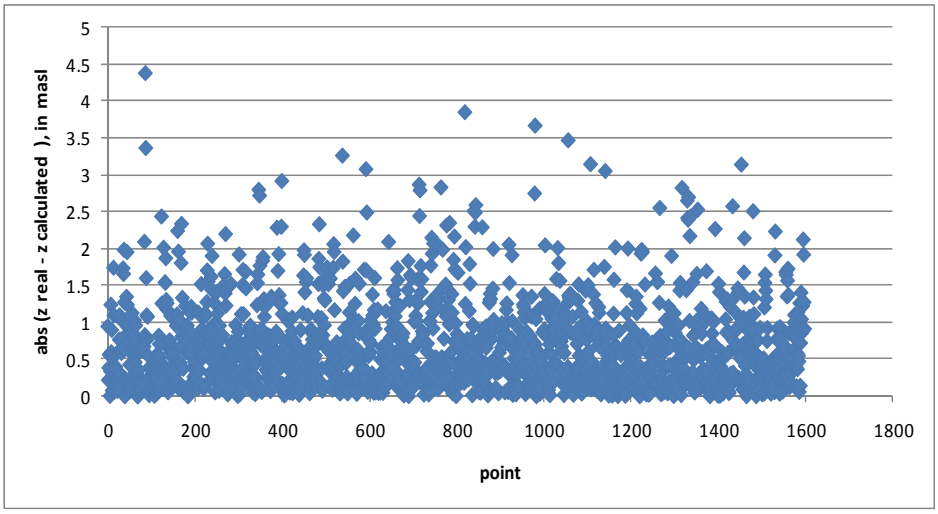


Fig. 9. Differences between real and estimated z values

8. Conclusions

In this work, a GP MATLAB Toolbox has been introduced exploiting the facilities that this interpreter offers. Individual trees are mapped into matrix where each row corresponds to

an individual in prefix notation. This type of representation allows to exploit the MATLAB data type, rectangular matrix. Then, the GP Toolbox was applied to model topographic surfaces; the study region was, in this case the Mezcalapa fork river. Modeling problem was formulated as a symbolic regression and obtained results showed considerably good the reconstruction of the topographic surface. However, it is necessary to continue the study to refine the model's estimations in the areas in which the values of the peaks and valleys are not reached. In general, it reproduces well the topography but it can be improved by considering different function sets, genetic operators or more complex individuals in order to reduce the estimation errors since the standard GP was the one implemented in this work.

9. Aknowledgment

The authors wish to thank Dr. Abel A. Jiménez C, Researcher at the Engineering Institute, for providing the data set of Mezcalapa river for this study.

10. References

Angeline, P.J. (1996) An Investigation into the Sensitivity of Genetic Programming to the Frequency of Leaf Selection During Subtree Crossover. In *Proc. of the First Annual Conference on Genetic Programming*. MIT Press, pp. 21-29.

Arfken, G. (1980) *Métodos matemáticos para físicos*. Diana.

Bäck, T. (1996) Evolutionary Algorithms in Theory and Practice. Oxford University Press.

Banzhaf, W., P. Nordin, R.E. Keller and F.D. Francone (1998) Genetic Programming: An Introduction. Morgan Kaufmann Publishers.

Barmpalexis, K., K. Kachrimanis, A. Tsakonas and E. Georgarakis (2011) Symbolic Regression via Genetic Programming in the Optimization of a Controlled Release Pharmaceutical Formulation. *Chemometrics and Intelligent Laboratory Systems*. 107:1, pp. 75-82.

Baumes, L. A., A. Blansché, P. Serna, A. Tchougang, N. Lachinche, P. Collet and A. Corma (2009) Using Genetic Programming for an Advanced Performance Assessment of Industrially Relevant Heterogeneous Catalysts. *Journal of Materials and Manufacturing Processes*. 24:3, pp. 282-292.

Chipperfield, A., P.J. Fleming, H. Pohlheim and C.M. Fonseca. (1994) *Genetic Algorithm Toolbox User's Guide*. Research Report 512. Dept. Automatic Control and Systems Engineering, University of Sheffield, U.K.

Fujiwara, Y. and H. Sawai (1999) Evolutionary Computation Applied to Mesh Optimization of a 3-D Facial Image. *IEEE Transactions on Evolutionary Computation*. 32, pp. 113-123.

Gálvez, A., A. Iglesias, A. Cobo, J. Puig-Pey and J. Espinola (2007) Bézier Curve and Surface fitting of 3D Point Clouds Through Genetic Algorithms, Functional Networks and Least-Square Approximation. *ICCSA Proceedings of The 2007 International Conference on Computational Science and Its Applications. Lecture Notes in Computer Science*. 4706/2007. pp. 680-693.

Goinski, A. (2008) Evolutionary Surface Reconstructions. *Conference on Human System Interactions*. pp. 464-469.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

Huang, H. L. and S. Y. Ho (2003) Mesh Optimization for Surface Approximation Using an Efficient Coarse-to-Fine Evolutionary Algorithm. *Pattern Recognition*. Elsevier Science. 36(5), pp. 1065-1081.

Iba, H. and N. Nikolaev (2000) Genetic Programming Polynomial Models of Financial Data Series. *Proc. Congress on Evolutionary Computation CEC 2000*. IEEE Press, pp. 1459-1466.

Keijzer, M. (2003) Improving Symbolic Regression with Interval Arithmetic and Linear Scaling. *6th European Conference on Genetic Programming EuroGP 200*. LNCS 2610. (Ryan *et al*., Eds.). Springer-Verlag, pp. 70-82.

Keller, R.E., W. Banzhaf, J. Mehnen and K. Weinert (1999) CAD Surface Reconstruction from Digitized 3D Point Data with a Genetic Programming/Evolution Strategy Hybrid. *Advances in Genetic Programming 3*, (Spector *et al*., Eds.), chapter 3. MIT Press., pp. 41-66.

Kodama, T., X. Li, K. Nakahira and D. Ito (2005) Evolutionary Computation Applied to the Reconstruction of 3-D Surface Topography in the SEM. *Journal of Electron Microscopy*, 54, Oxford University Press, pp. 429-435.

Koza, J.R. (1990) *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems*. Stanford University, Computer Science Dept. Technical Report STAN-CS-90-1314.

Mendoza, R., P. Alarcón and M. Berezowsky (1996) Cálculo del Campo de Velocidades en Cuerpos de Agua con Modelo Matemático Bidimensional en Coordenadas Curvilíneas Adaptables. *Informe Final Proyecto CONACYT 0641P-A9506*, Vol. 2, Instituto de Ingeniería, UNAM, México, D.F.

Mendoza, R. (2002) *Aplicación de la computación evolutiva en la estimación de cotas topográficas*. Tesis de maestría: Programa de Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, México, D.F.

Miller, J. F. and S. L. Harding (2008) Cartesian Genetic Programming. *Proc. of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. ACM, New York, NY, USA, p. 2701.

Parasuraman, K., A. Elshorbagy, S. K. Carey (2007) Modelling The Dynamic of The Evaporation Process using Genetic Programming. *Hydrological Sciences Journal*. 52:3. pp. 563-578.

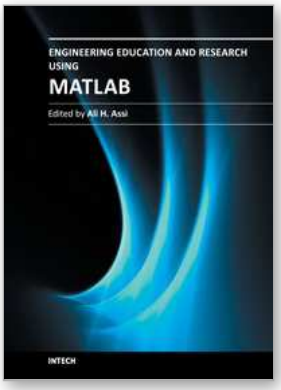
Paszkowicz, w. (2009) Genetic Algorithms, a Nature-Inspired Tool: survey of Applications in Material Science and Related Fields. *Materials and Manufacturing Processes*. 24:2. pp. 174-197.

Periaux, J., D. S. Lee, L. F. González and K. Srinivas (2009) Fast Reconstruction of Aerodynamic Shapes Using Evolutionary Algorithms And Virtual Nash Strategies in a CFD Design Environment. *Journal of Computational and Applied Mathematics*. 232-1, pp. 61-71.

Streeter, M. and L.E. Becker (2001) Automated Discovery of Numerical Approximation Formulae Via Genetic Programming. *Proc. Genetic and Evolutionary Computation Conference GECCO 2001* (Spector *et al*., editors). Morgan Kaufmann, pp. 147-154.

The Mathworks (1999) *Matlab Reference Guide*. The MathWorks Inc.

Wagner, T., T. Michelitsch and A. Sacharow (2007) On The Design of Optimisers for Surface Reconstruction. *Proc. of The 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2009)*. London, U.K., ACM, New York, pp. 2195-2202.



Engineering Education and Research Using MATLAB

Edited by Dr. Ali Assi

ISBN 978-953-307-656-0
Hard cover, 480 pages
Publisher InTech
Published online 10, October, 2011
Published in print edition October, 2011

MATLAB is a software package used primarily in the field of engineering for signal processing, numerical data analysis, modeling, programming, simulation, and computer graphic visualization. In the last few years, it has become widely accepted as an efficient tool, and, therefore, its use has significantly increased in scientific communities and academic institutions. This book consists of 20 chapters presenting research works using MATLAB tools. Chapters include techniques for programming and developing Graphical User Interfaces (GUIs), dynamic systems, electric machines, signal and image processing, power electronics, mixed signal circuits, genetic programming, digital watermarking, control systems, time-series regression modeling, and artificial neural networks.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Katya Rodríguez V. and Rosalva Mendoza R. (2011). A Matlab Genetic Programming Approach to Topographic Mesh Surface Generation, Engineering Education and Research Using MATLAB, Dr. Ali Assi (Ed.), ISBN: 978-953-307-656-0, InTech, Available from: http://www.intechopen.com/books/engineering-education-and-research-using-matlab/a-matlab-genetic-programming-approach-to-topographic-mesh-surface-generation

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821