Design of Hybrid Differential Evolution and Group Method of Data Handling for Inductive Modeling

Godfrey C. Onwubolu

School of Engineering & Physics, University of the South Pacific, Private Bag, Fiji

Onwubolu_g@usp.ac.fj

Abstract. *The group method of data handling (GMDH) and differential evolution (DE) population-based algorithm are two well-known nonlinear methods of mathematical modeling. In this paper, both methods are explained and a new design methodology which is a hybrid of GMDH and DE is proposed. The proposed method constructs a GMDH network model of a population of promising DE solutions. The new hybrid implementation is then applied to modeling and prediction of practical datasets and its results are compared with the results obtained by GMDH-related algorithms. Results presented show that the proposed algorithm appears to perform reasonably well and hence can be applied to real-life prediction and modeling problems.*

Keywords:

Inductive modeling, GMDH, DE, complex systems

1 Introduction

The GMDH is a heuristic self-organizing modeling method which Ivakhnenko [1—4] introduced as a rival to the method of stochastic approximations. The method is particularly useful in solving the problem of modeling multi-input to single-output data. The GMDH-type modeling algorithm is self-organizing because the number of neurons, the number of layers and the actual behavior of each created neuron are adjusting during the process of self-organization [7]. For readers who wish to have an understanding of fundamental concepts of GMDH, the following books are amongst the best in the field [5—7]. Although GMDH provides for a systematic procedure of system modeling and prediction, it has also a number of shortcomings. Among the most problematic can be stated: (i) a tendency to generate quite complex polynomial (since the complexity of the network increases with each training and selection cycle through addition of new layers) for relatively simple systems (data input); (ii) an inclination to producing overly complex network (model) when dealing with highly nonlinear systems owing to its limited generic structure (quadratic two-variable polynomial). Since the introduction of GMDH, there have been variants devised from different perspectives to realize more competitive networks and to alleviate the problems inherent with the standard GMDH algorithm [8-12]. In this paper, we introduce a hybrid modeling paradigm based on DE and GMDH.

2 The Group Method of Data Handling

The basics steps involved in the original Group Method of Data Handling (GMDH) modeling are:
Preamble: collect regression-type data of *n*-observations and divide the data into training and testing sets: $x_{ij};\ y_i \qquad i=1,2,...,n;\ j=1,2,...,m$

Step 1: Construct ${}^{m}C_2$ new variables $Z_1, Z_2, Z_3,..., Z_{\binom{m}{2}}$, in the *training dataset* for all independent variables (columns of X), two at a time $\left(x_{i,k-1}, x_{i,k};\ i\in[1,m]; k\in\left[2,\binom{m}{2}\right]\right)$ and construct the regression polynomial:

$$Z_1 = A + Bx_1 + Cx_2 + Dx_1^2 + Ex_2^2 + Fx_1x_2 \ \text{ at points } (x_{11}, x_{12}) \tag{1}$$

$$Z_k = A + Bx_{k-1} + Cx_k + Dx_{k-1}^2 + Ex_k^2 + Fx_{k-1}x_k \ \text{ at points } (x_{i,k-1}, x_{i,k}) \tag{2}$$

Step 2: For each of these regression surfaces, evaluate the polynomial at all *n* data points (i.e. using *A*, *B*, *C*, *D*, *E*, and *F* obtained from $x_{i,k-1}, x_{i,k};\ y_i$ for training). The coefficients of the polynomial are found by least square fitting as given in [13], or singular value decomposition (SVD) for singular-value problems as given in [14] using the data in the training set.

Step 3: Eliminate the least effective variables: replace the columns of X (old variables) by those columns of Z (new variables) that best estimate the dependent variable *y* in the testing dataset such that

$$d_k^2 = \sum_{i=n_t+1}^{n}(y_i - z_{i,k})^2, \qquad k\in\left[1,2,...,\binom{m}{2}\right] \tag{3}$$

Order Z according to the least square error $d_k \mid \|d_j\| < R$ where *R* is some prescribed number chosen a priori. Replace columns of X with the best Z's $(Z_{<R})$; in other words $X_{<R} \leftarrow Z_{<R}$

Step 4: Test for convergence. Let $DMIN = d_l$ where *l* = number of iterations. If $DMIN_l = DMIN_{l-1}$ go to Step 1, else stop the process.

3 Differential Evolution Algorithm

The differential evolution (exploration) [DE] algorithm introduced by Storn and Price [15] is a novel parallel direct search method, which utilizes *NP* parameter vectors as a population for each generation *G*. DE can be categorized into a class of *floating-point encoded, evolutionary optimization algorithm*s. Detailed descriptions of DE are provided [15-20]. DE algorithm was originally designed to work with continuous variables [20]. To solve discrete or combinatorial problems in general, Onwubolu [21], introduced the forward/backward transformation techniques, which facilitate solving any discrete or combinatorial problem. Successful applications of DE to a number of combinatorial problems are found in [22-24]. The steps involved in DE are as follows [20]:

Step 1 Initialization: DE works with a population of solutions, not with a single solution. Population *P* of generation *G* contains *NP* solution vectors (individuals of the population) and each vector represents potential solution for the optimization problem: $P^{(G)} = X_i^{(G)} = x_{j,i}^{(G)} \quad i=1,...,NP;\ j=1,...,D;\ G=1,...,G_{\max}$ (4)

In order to establish a starting point for optimum seeking, the population must be initialized:

$$P^{(0)} = x_{j,i}^{(0)} = x_j^{(L)} + rand_j[0,1]\bullet\left(x_j^{(U)} - x_j^{(L)}\right) \ \forall i\in[1,NP];\ \forall j\in[1,D] \tag{5}$$

where $rand_j[0,1]$ represents a uniformly distributed random value ranging from *0* to *1*.

Step 2 Mutation: In DE self-referential population recombination scheme, a temporary or trial population of *candidate vectors* for the subsequent generation, $V^{(G)} = v_{j,i}^{(G)}$, is generated as follows:

$$v_{j,i}^{(G)} = x_{j,r3}^{(G)} + F\bullet\left(x_{j,r1}^{(G)} - x_{j,r2}^{(G)}\right) \tag{6}$$

where $i\in[1,NP];\ j\in[1,D]$, $r1, r2, r3\in[1,NP]$, randomly selected, except $r1\neq r2\neq r3\neq i$, $k=(\text{int}(rand_i[0,1]\bullet D)+1)$, and $CR\in[0,1],\ F\in(0,1]$ are DE's control parameters (see [16, 17] for practical advice for selecting them).

Step 3 Crossover: DE also employs uniform crossover by building trial vectors out of parameter values that have been copied from two different vectors. In particular, DE crosses each vector with a mutant vector:

$$u_{j,i}^{(G)} = \begin{cases} v_{j,i,g} \ \ if \ \ rand_j[0,1] < CR \vee j = j_{rand} \\ \\ x_{i,j}^{(G)} \ \ if \ \ otherwise \end{cases} \quad (7)$$

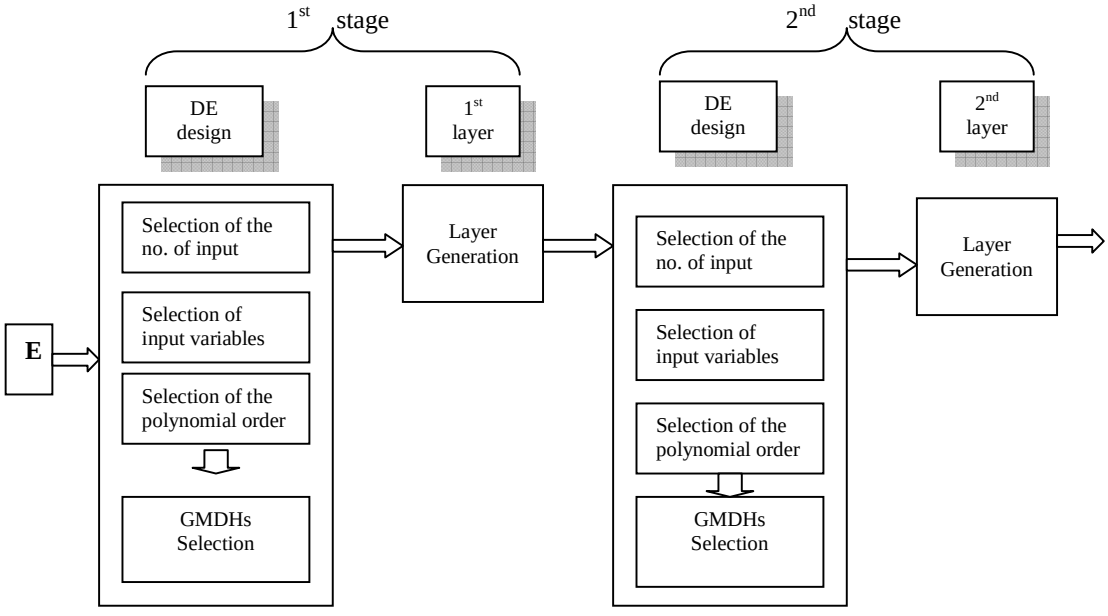
Step 4 Selection: In DE, if the trial vector, $u_{i,g}$, has an equal or lower objective function value than that of its target vector, $x_{i,g}$, it replaces the target vector in the next generation; otherwise, the target vector remains in place in the population for the least one more generation; these conditions are written as follows:

$$x_i^{(G+1)} = \begin{cases} u_i^{(G)} \ \ if \ \ \Im\left(u_i^{(G)}\right) \le \Im\left(x_i^{(G)}\right) \\ \\ x_i^{(G)} \ \ if \ \ otherwise \end{cases} \quad (8)$$

Step 5 Stopping criteria: After initialization, the process of mutation, recombination and selection is repeated until the optimum is located, or the number of generations reaches a preset maximum $g_{\max}$.

4 The proposed hybrid GMDH-DE Algorithm

It is evident from the previous two sections that both modeling methods have many common features, but, unlike the GMDH, DE does not follow a pre-determined path for input data generation. The same input data elements can be included or excluded at any stage in the evolutionary process by virtue of the stochastic nature of the selection process. A DE algorithm can thus be seen as implicitly having the capacity to learn and adapt in the search space and thus allow previously bad elements to be included if they become beneficial in the latter stages of the search process. The standard GMDH algorithm is more deterministic and would thus discard any underperforming elements as soon as they are realized. Using DE in the selection process of the GMDH algorithm, the model building process is free to explore a more complex universe of data permutations. This selection procedure has three main advantages over the standard selection method. Firstly, it allows unfit individuals from early layers to be incorporated at an advanced layer where they generate fitter solutions. Secondly, it also allows those unfit individuals to survive the selection process if their combinations with one or more of the other individuals produce new fit individuals. Thirdly, it allows more implicit non-linearity by allowing multi-layer variable interaction.



E: entire point, S: selected GMDHs: preferred outputs in the i-th stage

Fig. 1 Overall architecture of the DE-GMDH

The new DE-GMDH algorithm that is proposed in this paper is constructed in exactly the same manner as the standard GMDH algorithm except for the selection process. The selected fit individuals are entered

in the GMDH algorithm as inputs at the next layer as shown in Figure 1. The whole procedure is repeated until the criterion for terminating the GMDH run has been reached.

4.1 Representation of representation of encoding strategy of each PD

In the standard GMDH, the issues to address are: (i) how to determine the optimal *number of input variables*; (ii) how to select the *order of polynomial* forming a partial descriptor (PD) in each node; and (iii) how to determine which *input variables* are chosen. Therefore, in the DE-GMDH design, the most important consideration is how to encode the key factors into the vector of solution (called individual of the population). While in genetically optimized polynomial neural network (gPNN) implementation, a binary coding has been used [11], [25], we employ a combinatorial DE coding approach [21-24] and a sequence of integer numbers is used in each vector of solution. In our DE-GMDH design, there are three system parameters (P_1, P_2 ,and P_3) which are now described. $P_1 \in [1,3]$ is randomly generated and represents the order of polynomial. $P_2 \in [1, r]$ is randomly generated and represents the number of input variables (where $r = \min(D, 5)$; D is the width of the input dataset; the default lower bound is $r = 2$). Designer must determine the maximum number (r) in consideration of the characteristic of system, design specification, and some prior knowledge of model. With this method the problem of conflict between over-fitting and generalization on one hand, and computation time on the other hand can be solved [25]. $P_3 \in [1, D]$ is the sequence of integers for each solution vector of the population of solutions, and it represents the entire candidates in the current layer of the network. The relationship between *vector of solution* and information on PD is shown in Figure 2, while the PD corresponding to the vector of solution is shown in Figure 3. Figure 2 shows the information for a PD for case where the width (number of columns) of the dataset is 5. Therefore, a population of initial vectors of solutions is randomly generated for initialization. For $P_1 = 2$ and $P_2 = 2$ (polynomial order of Type 2 [quadratic form], and two input variables to the node), only the first two columns of the population of solutions will be considered, corresponding to the selection of the following pair-wise combinations of the input variables: {1 3}, {5 4}, {2 1}, {4 3}, {1 5}, {3 4},…,{5 2}. For the fifth pair of selected input variables {1 5}, the PD output is

$$\hat{y} = f(x_1, x_5) = c_1 + c_2 x_1 + c_3 x_5 + c_4 x_1 x_5 + c_5 x_1^2 + c_6 x_5^2 \qquad (9)$$

where the coefficients $(c_1, c_2, c_3, c_4, c_5, c_6)$ are evaluated using the training dataset.

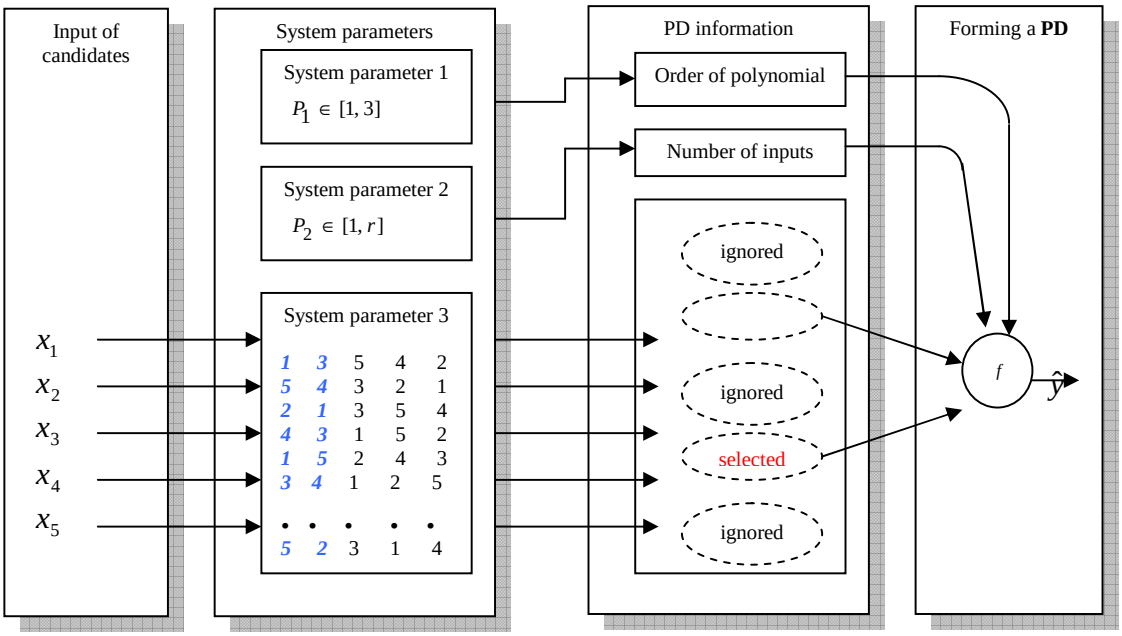


Fig. 2 relationship between *vector of solution* and information on PD

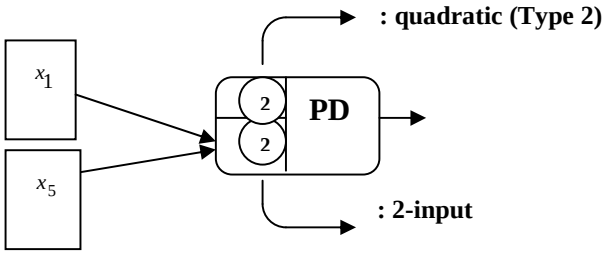


Fig. 3 Relationship between *vector of solution* and information on

4.2 Fitness function

The objective function (performance index) is a basic instrument guiding the evolutionary search in the solution space [26]. The objective function includes both the training data and the testing data and comes as a convex sum of two components:

$$f(PI, EPI) = \theta \times PI + (1-\theta) \times EPI \tag{10}$$

where *PI* and *EPI* denote the performance index for the training data and testing data (or validation data), respectively. Moreover θ is a weighting factor.

4.3 Framework of the design procedure of the DE-GMDH

The framework of the design procedure of the *DE-GMDH* consists of the following steps.
Step 1: *Determine system's input variables*: Define the input variables of the system as $\boldsymbol{x}_i$ $(i = 1, 2, ..., n)$ related to output variable *y*.
Step 2: *Form training and testing data*: The input-output data set $(\boldsymbol{x}_i, y_i) = (x_{1i}, x_{2i}, ..., x_{ni}, y_i), i = 1, 2, ..., n$ (n: the total number of data) is divided into a training and a testing dataset. Their sizes are denoted by n_{tr} and n_{te} respectively, and $n = n_{tr} + n_{te}$. The training data set is used to construct the *DE-GMDH* model. Next, the testing data set is used to evaluate the quality of the model.
Step 3: *Determine initial information for constructing the DE-GMDH structure*: We determine initial information for the *DE-GMDH* structure: i) The termination method. ii) The maximum number of input variables used at each node in the corresponding layer. iii) The value of the weighting factor of the aggregate objective function.
Step 4: *Determine polynomial neuron (PN) structure using DE design*: Determining the polynomial neuron (PN), is concerned with the selection of the number of input variables, the polynomial order, and the input variables to be assigned in each node of the corresponding layer. The PN structure is determined using DE design. The DE design available in a PN structure uses a solution vector of DE is the one illustrated in Fig. 2 in which the design of optimal parameters available within the PN (viz. the number of input variables, the order of the polynomials, and input variables) at last leads to a structurally and parametrically optimized network, which is more flexible as well as simpler in architecture than the conventional *GMDH*. Each sub-step of the DE design procedure of three kinds of parameters available within the PN has already been discussed. The polynomials differ according to the number of input variables and the polynomial order. Several types of polynomials can be used such as Bilinear, Biquadratic, Modified biquadratic, Trilinear, Triquadratic, and Modified triquadratic.
Step 5: *Coefficient estimation of the polynomial corresponding to the selected node (PN)*: The vector of the coefficients of the PDs is determined using a standard mean squared error by minimizing the following index:

$$E_r = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} (y_i - z_{ki})^2, \qquad k = 1, 2, ..., r, \tag{11}$$

where z_{ki} denotes the output of the *k*-th node with respect to the *i*-th data, r is the value in the second system parameter $P_2 \in [1, r]$ and n_{tr} is the number of training data subsets. Evidently, the coefficients of the

PN of nodes in each layer are determined by the standard least square method. This procedure is implemented repeatedly for all nodes of the layer and also for all *DE-GMDH* layers starting from the input layer and moving to the output layer.

Step 6: *Select nodes (PNs) with the best predictive capability, and construct their corresponding layer*: As shown in Fig. 2, all nodes of the corresponding layer of *DE-GMDH* architecture are constructed by DE optimization. The generation process of PNs in the corresponding layer is described in detail as the design procedure of 4 sub-steps. A sequence of the sub-steps is as follows:

***Sub-step 1*)** We determine initial DE information for generation of the *DE-GMDH* architecture. That is, the number of generations and populations, mutation rate, crossover rate, and the length of a solution vector.

Sub-step 2*)** The nodes (PNs) are generated by DE design as many as the number of populations in the 1st generation. Where, one population takes the same role as one node (PN) in the *DE-GMDH* architecture and each population is operated by DE as shown in Fig. 2. That is, the number of input variables, the order of the polynomials, and the input variables as one individual (population) are selected by DE. The polynomial parameters are produced by the standard least squares method. ***Sub-step 3*)** Evaluate the performance of PNs (nodes) in each population using (10) as already discussed. ***Sub-step 4*)** To produce the next generation, we carry out *mutation*, *crossover*, and *selection* operations using DE initial information and the fitness values obtained from ***sub-step 3. Generally, after these DE operations, the overall fitness of the population improves. We choose several PNs characterized by the best fitness values. Here, we select the node that has the highest fitness value for optimal operation of the next iteration in the *DE-GMDH* algorithm. The outputs of the retained nodes (PNs) serve as inputs in the subsequent layer of the network. The iterative process generates the optimal nodes of a layer in the *DE-GMDH* model.

Step 7: *Termination criterion:* The termination method exploited here the maximum number of generations predetermined by the designer to achieve a balance between model accuracy and its complexity.

The *DE-GMDH* algorithm repeats steps 4-6 consecutively. After the iteration process, the final generation of population consists of highly fit solution vectors that provide optimum solutions. The *DE-GMDH* algorithm is implemented in C++ language using a Pentium PC, with at least 65MB, 733MHz processor.

5 Experimentation

We present two classes of problems: one modeling problem based on end-milling tool wear, and one prediction problem based on US Federal Funds Rate (%).

5.1 DE-GMDH for modeling the end-milling tool wear problem:

The end-milling tool wear experiment was carried out on the Seiki Hitachi milling machine at the University of the South Pacific. A 16mm Co-high speed (HSS) Kobelco Hi Cut brand new end mill cutter was used to machine the work-piece. The end mill cutter had four flutes and the flute geometry was 30 degrees spiral. The key process input variables were spindle speed, feed, and depth-of-cut, while the key output variable is tool-wear. The range chosen for the experiments are $v \in \{27, 39, 49\}$, $f \in \{0.0188, 0.0590, 0.1684\}$, $d_t \in \{0.5, 1.0, 1.5\}$ for speed (v) , feed (f), and depth-of-cut (d_t) respectively. The end-milling tool wear can be modeled as

$$VB = c_1 + c_2 x_1 + c_3 x_2 + c_4 x_1 x_2 + c_5 x_1^2 + c_6 x_2^2 \tag{12}$$

where x_1 = spindle speed, x_2 = feed, and x_3 = depth of cut. The methodology presented in Section 4 was applied to the output of the *DE-GMDH* to determine model coefficients {0.00262059, 0.0495905, 0.00054459, -0.0329972, 0.00303677, 0.00187647} and an output sequence {1, 3, 2} for the quadratic equation given in equation (13) for the milling operation, leading to a predictive model given as

$$VB = 0.002620 + 0.049590 x_1 + 0.000544 x_3 - 0.032997 x_1 x_3 + 0.003036 x_1^2 + 0.001876 x_3^2 \tag{13}$$

Figure 4 shows the output results with the mean square error (MSE) for testing for DE-GMDH. Results of comparative study for four modeling approaches are shown in Table 1. The results show that the proposed DE-GMDH algorithm outperforms other approaches for both training (PI) and generalization (EPI) errors.

Tab. 1. Performance index of identified mode

Model	*PI*	*EPI*
PNN [27]	4.345000	3.694000
gPNN [11]	0.277014	2.029077
e-GMDH [28]	0.033419	0.154649
DE-GMDH [this paper]	0.028946	0.005274

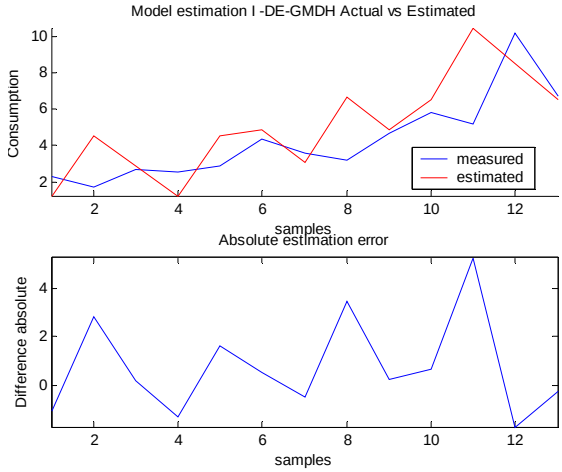


Fig. 4. The DE-GMDH actual and predicted figures for testing the tool wear problem

5.2 DE-GMDH for modeling US Federal Funds Rate (%)

In our second experimentation, we used the US Federal Funds Rate (%), monthly average published by Ohashi of the Washington World Bank in [5] (pp.209) using a delay period of 3. The DE-GMDH used for the work reported in this paper found an output sequence {1, 2, 3} and coefficients to be as follows {0.0414144, 0.31091, 0.305793, -0.00216837, 0.0245347, -5.76E-07} leading to a predictive model given as

$$FFR = 0.041414 + 0.310910x(t-1) + 0.305793x(t-2) - 0.002168x(t-1)x(t-2) + 0.024534x(t-1)^2 - 0.0000005x(t-2)^2 \quad (14)$$

Figure 5 shows the output results with the mean square error (MSE) for testing for this problem.

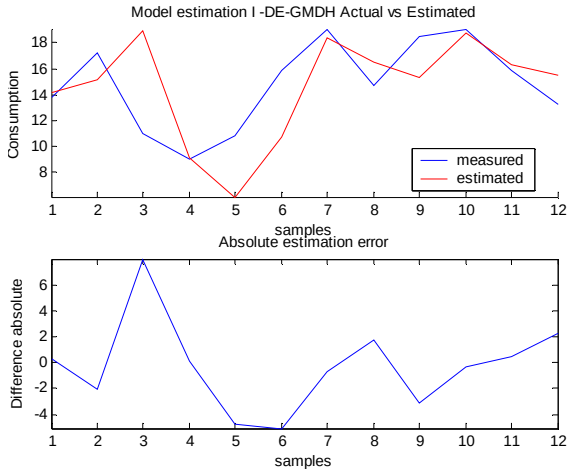


Fig. 5. The DE-GMDH actual and predicted figures for testing the Federal Funds Rate (%) problem

For the proposed DE-GMDH, the testing error (PI) = 0.0153948, while the testing error (EPI) = 0.0481683. For the same problem, PNN realized testing error (PI) = 8.505876, and testing error (EPI) = 2.869034.

6 Conclusions

In this paper, both methods of DE and GMDH are explained and a new design methodology which is a hybrid of *GMDH* and *DE*, which we refer to as *DE-GMDH*, is proposed. The architecture of model is not predefined, but can be self-organized automatically during the design process. We have applied the *DE-GMDH* approach to the problem of developing predictive model for tool-wear in end-milling operations and the Federal Funds Rate problem. The predictive models based on *DE-GMDH* network give reasonably good solutions. Realizing the model for our tool wear in milling operation is a useful and practical tool in industrial applications since we can predict the wear level of the milling tool once we know the spindle speed, feed, and material depth-of-cut during operation. In our approach, we first present a methodology for modeling, and then develop predictive model(s) of the problem being solved in form of second-order equations based on the input data and coefficients realized. One major conclusion resulting from the studies carried out in implementing hybrid *DE-GMDH* network is that population-based optimization techniques (genetic programming [GP], genetic algorithm [GA], differential evolution [DE], scatter search [SS], ant colony system [ACS], particle swarm optimization [PSO], etc.) are all candidates of hybridization with GMDH. In the past, only the use of GA and GP has been studied for hybridization with GMDH. Further research activities include incorporating more design features to improve the modeling solutions and to realize more flexibility.

References

[1] Ivakhnenko A. G., 1968. The Group Method of Data Handling-A rival of the Method of Stochastic Approximation. *Soviet Automatic Control*, *vol 13 c/c of avtomatika*, 1, 3, 43-55.

[2] Ivakhnenko, A. G., 1971. Polynomial theory of complex systems, *IEEE Trans. on Systems, Man and Cybernetics*, vol. SMC-1, pp. 364-378.

[3] Ivakhnenko, A. G., Ivakhnenko, G. A. and Muller, J.A. 1994. Self-organization of neural networks with active neurons, *Pattern Recognition and Image Analysis*, vol. 4, no. 2, pp. 185-196.

[4] Ivakhnenko, A. G. and Ivakhnenko, G. A. 1995. The review of problems solvable by algorithms of the group method of data handling (GMDH), *Pattern Recognition and Image Analysis*, vol. 5, no. 4, pp. 527-535.

[5] Farlow, S. J. (ed.) 1984: *Self-organizing Methods in Modeling. GMDH Type Algorithms*. Marcel Dekker. New York, Basel

[6] Madala, H.R.; Ivakhnenko, A.G. 1994: *Inductive Learning Algorithms for Complex Systems Modeling*. CRC Press Inc., Boca Raton, Ann Arbor, London, Tokyo

[7] Mueller, J-A., and Lemke, F.: *Self-Organizing Data Mining: An Intelligent Approach to Extract Knowledge From Data*, Dresden, Berlin, 1999.

[8] Robinson, C. 1998. *Multi-objective optimization of polynomial models for time series prediction using genetic algorithms and neural networks*, PhD Thesis in the Department of Automatic Control & Systems Engineering, University of Sheffield, UK.

[9] Hiassat M., Abbod M., and Mort N. 2003. Using Genetic Programming to Improve the GMDH in Time Series Prediction, *Statistical Data Mining and Knowledge Discovery*, edited by Hamparsum Bozdogan. Chapman & Hall CRC, pp257-268.

[10] Iba, H., de Garis, H., Sato, T., Genetic programming using a minimum description length principle, In Advances in Genetic Programming, Kinnear, K. E. Jr (ed), Cambridge: MIT, 1994, pp 265-284.

[11] Park, H.-S., Park, B.-J., Kim, H, -K., and Oh, S.-K. 2004. Self-organizing polynomial neural networks based on genetically optimized multi-layer perceptron architecture, International Journal of Control, Automation, and Systems, 2(4), pp. 423-434.

[12] Oh, S.-K., Park, B.-J., and Kim, H.-K, Genetically optimized hybrid fuzzy neural networks based on linear fuzzy inference rules, International Journal of Control, Automation, and Systems, 3(2), pp. 183-194, 2005.

[13] Press, W. H., Teukolsky, S A, Vetterling, W. T. and Flannery, B. P. 1992. "*Numerical recipes in C: The art of scientific computing*". Cambridge University Press.

[14] Nariman-Zadeh, N., Darvizeh, A., and Ahmad-Zadeh, G. R., Hybrid genetic design of GMDH-type neural networks using singular value decomposition for modeling and predicting of the explosive cutting process, Nariman-Zadeh, Proc. Instn Mech. Engrs Vol 217 Part B: Nariman-Zadeh, 779-790.
[15] Storn, R. and Price, K., 1995, Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, *Technical Report TR-95-012, ICSI, March 1999 (Available via ftp from ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012.ps.Z).*
[16] Storn, R. M. and Price, K. V., 1997, Differential evolution – a simple evolution strategy for fast optimization. *Dr. Dobb's Journal*, April 1997, 18–24 and 78.
[17] Price, K. V, 1999, An introduction to differential evolution. *New Ideas in Optimization*, (Eds.) Corne, D., Dorigo, M., and Glover, McGraw Hill, International (UK), 79-108.
[18] Storn, R. M., 1999, Designing digital filters with differential evolution, *New Ideas in Optimization*, (Eds.) Corne, D., Dorigo, M., and Glover, McGraw Hill, International (UK*)*, 109-125.
[19] Price, K. V., and Storn, R. M., 2001, Differential evolution homepage (Web site of Price and Storm) as at 2001. http://www.ICSI.Berkeley.edu/~storn/code.html
[20] Price, K. V., Storn, R. M, 2006, and Lampinene, J. A., Differential Evolution: A Practical Approach to Global Optimization, Springer-Verlag, Berlin 2005.
[21] Onwubolu, G. C., 2001, Optimization using differential evolution, *Institute of Applied Science Technical Report, TR-2001/05*.
[22] Davendra, D., *Hybrid Differential Evolution and Scatter Search*, MSc Thesis, The University of the South Pacific, 2003.
[23] Onwubolu, G. C., and Davendra, D., Scheduling flow shops using differential evolution algorithm, *European Journal of Operational Research*, 171, 674-692, 2006.
[24] Davendra, D., and Onwubolu, G. C., Scheduling flow shops using enhanced differential evolution algorithm, *European Conference on Modeling and Simulation (ECMS)*, Prague, Czech, 2007.
[25] Kim, D., and Park, G.-T, GMDH-type neural network modeling in evolutionary optimization, Lecture Notes in Computer Science, Springer, Berlin, 3533, pp. 563-570, 2005.
[26] Oh, S.-K and Pedrycz, W., Identification of fuzzy systems by means of an auto-tuning algorithm and its application to nonlinear systems, Fuzzy Sets and Systems, 115(2), pp. 205-230, 2000.
[27] Park, H.-S., Park, B.-J., Kim, H, -K., and Oh, S.-K. 2004. Self-organizing polynomial neural networks based on genetically optimized multi-layer perceptron architecture, International Journal of Control, Automation, and Systems, 2(4), pp. 423-434.
[28] Buryan, P., and Onwubolu, G. C., 2007, Design of Enhanced MIA-GMDH Learning Networks, (submitted)

This paper proposes a new design methodology which is a hybrid of differential evolution (DE) and Group Method of Data Handling (GMDH) for self-organizing data mining. The new hybrid implementation is applied to the data mining activity of prediction of soil moisture, which is an aspect of hydrology. Experimental results indicate that the proposed approach is useful for data mining technique for forecasting hydrological data. Keywords: Inductive modeling; Self-Organizing Data Mining, DE, GMDH 1 Introduction Data Mining (DM) is an important component of the emerging field of knowledge discovery Modeling simulation using the ELMOD program for interpretation of a deep induction Phasor log in a North Sea well with. apparent dip of 38Â°. The initial trial model was refined in two steps, left to right, until agreement was reached between the model-computed and field logs. Simulation Three was consistent with the log analystâ€™s knowledge of the field.Â Numerical methods, such as 2D-and 3D-FEM codes, can solve differential equations in almost any geometry. The FEM is widely used in research and engineering, from the design of automobile bodies to the study of diffusion over corrugated surfaces.Â Earlier studies of hybrid techniques 1 describe other methods for modeling induc-. tion tool responses.15 Numerous codes and. Hybrid method. Finite-element method Rt.