

A Numerical Approach Based on Neuro-Fuzzy Systems for Obtaining Functional Inverse

Hadi Sadoghi Yazdi¹, Sohrab Effati²

1-Engineering Department, Tarbiat Moallem University of Sabzevar, Sabzevar, Iran

2-Department of Mathematics, Tarbiat Moallem University of Sabzevar, Sabzevar, Iran

Abstract: In this paper, a new and easy available numerical method is presented in calculation of functional inverse. As for ANFIS (Adaptive Network based Fuzzy Inference System) model is available without great difficulty in MATLAB software also due to important of obtaining functional inverse in wide range applications, we present ANFIS-based approach for the first time for obtaining inverse of mathematical functions. The proposed approach includes two main stages, in the first step; some limited points are sampled from desired function using Monte Carlo simulation and second step contains training of ANFIS system that input system is output points of function and ANFIS desired values are input values to function. One of notes in the proposed approach is calculation of inverse of time varying functions only with tuning of generated fuzzy model with the least square and the back propagation (BP) algorithms to form of hybrid learning algorithm which able into tracking of time varying functions. Experimental results over many functions show superiority of the proposed method relative MATLAB software.

Keywords: Functional inverse; Adaptive Network based Fuzzy Inference System; Monte Carlo method; Hybrid learning; MATLAB software.

1. Introduction

Functional inverse not only is used in mathematics but also have many applications in engineering and sciences. Finding sources from measured data is a common problem in a real world. From the mathematic point of view, solving the equation obtain inverse of function or an iterative approach give us functional inverse. If f is a continuous function defined in interval $[a, b]$, g is inverse of f if $(f \circ g)(x) = x$ or $f(g(x)) = x$; so g is f^{-1} or is a functional inverse. Functional inverse can study from different views:

Finding source of generated data:

The inverse problem of finding the source from the data is other form of functional inverse. In practice a process has two elements, the physical process and the measurement. In the measurement procedure noise is inserted to data. So it is needed a smoother is selected for noise reduction then backward mapping is required for finding source [1].

In the field of image processing, finding main image from degraded image is based on dealing with the problem as an inverse problem. Based on the observation model, our objective is to obtain a high

resolution image which is as close as possible to the original high resolution image subject to certain constraints [4].

Inverse learning in signal processing:

The developments of inverse learning was initiated from adaptive signal processing and control of linear systems approach [2, 3] and due to its potential applicability, until now several modifications and applications are still being developed in the literature.

In data communications, receiver equalization has become an essential building block to mitigate the inter-symbol interference problem, which is due to limited bandwidth of low cost channel materials. Several equalization methods have been developed which are types of functional inverse [5].

Constructing inverse signal:

Creating of inverse signal for reducing of undesirable signal is another view of functional inverse. Active noise control systems are based on the principle of superposition: an unwanted noise is cancelled out by the action of a secondary noise that generates a sound wave of equal amplitude and opposite phase [6]. This idea has evolved and new applications have been developed in which the

undesirable noise is not completely cancelled but manipulated to generate a desired residual noise field. One promising application in modern active noise control is the creation of desired acoustic sound fields inside cars, where the retained residual noise is applied to improve safety or create a comfortable acoustic sensation [7].

Creating device inverse:

A loudspeaker is a device that converts a signal from its electric form to audible sound wave. Noticeable distortions may be introduced in this conversion leading to a significant loss of sound quality. The technique of inverse loudspeaker has thus been developed to deal with this problem [8].

In this paper, a learning approach is selected to learn an inverse of a function based on ANFIS but how?

Learning based functional inverse:

Ordinarily, neural networks learn a function $\vec{y} = f(\vec{x})$, where \vec{y} is a vector of outputs, \vec{x} is a vector of inputs, and $f(\cdot)$ is the function to be learned. Sometimes, however, you may want to learn an inverse of a function $f(\cdot)$, that is, given \vec{y} , you want to be able to find \vec{x} such that $\vec{y} = f(\vec{x})$. In general, there may be many different \vec{x} that satisfy the equation $\vec{y} = f(\vec{x})$. For example, in robotics [9, and 10], \vec{x} might describe the positions of the joints in a robot's arm, while \vec{y} would describe the location of the robot's hand. There are simple formulas to compute the location of the hand given the positions of the joints, called the "forward kinematics" problem. But there is no simple formula for the "inverse kinematics" problem to compute positions of the joints that yield a given location for the hand. Furthermore, if the arm has several joints, there will usually be many different positions of the joints that yield the same location of the hand, so the forward kinematics function is many-to-one and has no unique inverse.

As another example, consider an industrial process in which \vec{x} represents settings of control variables imposed by an operator, and \vec{y} represents measurements of the product of the industrial process. The function $\vec{y} = f(\vec{x})$ can be learned by neural networks using conventional training methods. But the goal of the analysis may be to find control settings \vec{x} that yield a product with specified measurements \vec{y} , in which case an inverse of $f(\vec{x})$ is required. In industrial applications, financial considerations are important, so not just any setting \vec{x} that yields the desired result \vec{y} may be acceptable. Perhaps a function can be specified that gives the cost of \vec{x} resulting from energy consumption, raw materials, etc., in which case you would want to find

\vec{x} that minimizes the cost function while satisfying the equation $\vec{y} = f(\vec{x})$.

The obvious way to try to learn an inverse function is to generate a set of training data from a given forward function, but designate \vec{y} as the input and \vec{x} as the output when training the network. Using a least-squares error function, this approach will fail if $f(\cdot)$ is many-to-one. The problem is that for an input \vec{y} , the net will not learn any single \vec{x} such that $\vec{y} = f(\vec{x})$, but will instead learn the arithmetic mean of all the \vec{x} s in the training set that satisfy the equation [11]. One solution to this difficulty is to construct a network that learns a mixture approximation to the conditional distribution of \vec{x} given \vec{y} [11]. However, the mixture method will not work well in general for a \vec{x} vector that is more than one-dimensional, such as $y = x_1^2 + x_2^2$, since the number of mixture components required may increase exponentially with the dimensionality of \vec{x} . And you are still left with the problem of extracting a single output vector from the mixture distribution, which is nontrivial if the mixture components overlap considerably. Another solution is to use a highly robust error function, such as a re-descending M-estimator, that learns a single mode of the conditional distribution instead of learning the mean [12, and 13]. Additional regularization terms or constraints may be required to persuade the network to choose appropriately among several modes, and there may be severe problems with local optima.

Another approach is to train a network to learn the forward mapping $f(\cdot)$ and then numerically invert the function. Finding \vec{x} such that $\vec{y} = f(\vec{x})$ is simply a matter of solving a nonlinear system of equations, for which many algorithms can be found in the numerical analysis literature [14]. One way to solve nonlinear equations is turn the problem into an optimization problem by minimizing sum $\sum_{i=1}^N (y_i - f(x_i))^2$ where N is number of training samples. This method fits in nicely with the usual gradient-descent methods for training neural networks [15]. Since the nonlinear equations will generally have multiple solutions, there may be severe problems with local optima, especially if some solutions are considered more desirable than others. You can deal with multiple solutions by inventing some objective function that measures the goodness of different solutions, and optimizing this objective function under the nonlinear constraint $\vec{y} = f(\vec{x})$ using any of numerous algorithms for nonlinear programming [16]. The power and flexibility of the nonlinear programming approach are offset by possibly high computational demands. If the forward mapping $f(\cdot)$ is obtained by training a

network, there will generally be some error in the network's outputs. The magnitude of this error can be difficult to estimate. The process of inverting a network can propagate this error, so the results should be checked carefully for validity and numerical stability. Some training methods can produce not just a point output but also a prediction interval [11, and 17]. You can take advantage of prediction intervals when inverting a network by using nonlinear programming methods. For example, you could try to find an \bar{x} that minimizes the width of the prediction interval under the constraint that the equation $\bar{y} = f(\bar{x})$ is satisfied. Or instead of requiring $\bar{y} = f(\bar{x})$ be satisfied exactly, you could try to find \bar{x} such that the prediction interval is contained within some specified interval while minimizing some cost function.

Novelty of this paper appropriate to calculation of inverse function using adaptive network based fuzzy inference system. But why a neural network is not enough for this purpose?

1.1. Brief review over advantages of adaptive network based fuzzy Inference system

Artificial neural network is a favorable technique to solve optimization problems because it can simulate the operations of the brain and uses parallel processing to save computational time [18]. Fuzzy logic approach is another intelligent computing tool which is competent for applying to wide variety of problems. Primary aim of Lotfi Zadeh which introduced the notion of a "fuzzy set" in 1965 was to set up a formal framework for the representation and management of vague and uncertain knowledge. Neural Networks are demonstrated to have powerful capability of expressing relationship between input–output variables. In fact it is always possible to develop a structure that approximates a function with a given precision. However, there is still distrust about the neural networks identification capability in some applications [19]. Fuzzy set theory plays an important role in dealing with uncertainty in plant modeling applications.

Recently, there has been a growing interest in combining both these approaches, and as a result, neuro-fuzzy computing techniques have been evolved. Neuro-fuzzy systems are fuzzy systems, which use neural networks theory in order to determine their properties (fuzzy sets and fuzzy rules) by processing data samples [20]. Neuro-fuzzy integrates to synthesize the merits of both neural networks and fuzzy systems in a complementary way to overcome their disadvantage. The fusion of neural network and fuzzy logic in neuro-fuzzy models possess both low-level learning and computational power of neural networks and advantages of high-level human like thinking of fuzzy systems. ANFIS (Adaptive Network based Fuzzy Inference System) model combined the neural network adaptive capabilities and the fuzzy logic qualitative approach.

ANFIS presented by Jang [21]. It has attained its popularity due to a broad range of useful applications in such diverse areas in recent years as optimization of fishing predictions [22], vehicular navigation [23], identify the turbine speed dynamics [24], radio frequency power amplifier linearization [25], microwave application [26], image denoising [27, and 28], prediction in cleaning with high pressure water [29], sensor calibration [30], fetal electrocardiogram extraction from ECG signal captured from mother [31], identification of normal and glaucomatous eyes [32].

All above works show that ANFIS as a good universal approximator, predictor, interpolator and estimator. They demonstrate each non-linear function of many inputs and outputs can be easily constructed with ANFIS. Dynamic behavior of ANFIS motivates us in this study to use it in finding time variable functional inverse. The paper is organized as follows. The architecture of adaptive neuro-fuzzy inference system is explained in Section 2. Section 3 is devoted to present of proposed structure. Experimental results are discussed in section 4 and in final section conclusions are presented.

2. Adaptive neuro-fuzzy inference system (ANFIS) Architecture

Neural networks (NNs) are demonstrated to have powerful capability of expressing relationship between input–output variables. In fact it is always possible to develop a structure that approximates a function with a given precision. However, there is still distrust about NNs identification capability in some applications [19]. Fuzzy set theory plays an important role in dealing with uncertainty in plant modeling applications. Neuro-fuzzy systems are fuzzy systems, which use NNs to determine their properties (fuzzy sets and fuzzy rules) by processing data samples [20]. Neuro-fuzzy integrates to synthesize the merits of both NN and fuzzy systems in a complementary way to overcome their disadvantage. The fusion of a NN and fuzzy logic in neuro-fuzzy models possess both low-level learning and computational power of NNs and advantages of high-level human like thinking of fuzzy systems. For identification, hybrid neuro-fuzzy system called ANFIS combines a NN and a fuzzy system together. ANFIS has been proved to have significant results in modeling nonlinear functions. In ANFIS, the membership functions (MF) are extracted from a data set that describes the system behavior. The ANFIS learns features in the data set and adjusts the system parameters according to given error criterion. In a fused architecture, NN learning algorithms are used to determine the parameters of fuzzy inference system. Below, we have summarized the advantages of the ANFIS technique.

- Real-time processing of instantaneous system input and output data's. This property helps using of this technique for many operational researches problems.

- Offline adaptation instead of online system-error minimization, thus easier to manage and no iterative algorithms are involved.
- System performance is not limited by the order of the function since it is not represented in polynomial format.
- Fast learning time.
- System performance tuning is flexible as the number of membership functions and training epochs can be altered easily.
- The simple if-then rules declaration and the ANFIS structure are easy to understand and implement.

A typical architecture of ANFIS is shown in Fig 1 for modeling of function $f(x, y)$, in which a circle indicates a fixed node, and a square indicates an adaptive node. For simplicity, we consider two inputs x, y and one output z in the fuzzy inference system (FIS). The ANFIS used in this paper implements a first-order Sugeno fuzzy model. Among many fuzzy inference systems, the Sugeno fuzzy model is the most widely used for its high interpretability and computational efficiency, and built-in optimal and adaptive techniques. For example for a first-order Sugeno fuzzy model, a common rule set with two fuzzy if-then rules can be expressed as:

$$\begin{aligned} \text{Rule 1: If } x \text{ is } A_1 \text{ and } y \text{ is } B_1, \\ \text{then } z_1 = p_1x + q_1y + r_1 \end{aligned} \quad (1)$$

$$\begin{aligned} \text{Rule 2: If } x \text{ is } A_2 \text{ and } y \text{ is } B_2, \\ \text{then } z_2 = p_2x + q_2y + r_2 \end{aligned} \quad (2)$$

where $A_i, B_i (i = 1, 2)$ are fuzzy sets in the antecedent, and $p_i, q_i, r_i (i = 1, 2)$ are the design parameters that are determined during the training process. As in Fig 1, the ANFIS consists of five layers:

Layer 1, every node i in this layer is an adaptive node with a node function:

$$\begin{aligned} O_i^1 &= \mu_{A_i}(x), & i = 1, 2 \\ O_i^1 &= \mu_{B_i}(y), & i = 3, 4 \end{aligned} \quad (3)$$

where x, y are the input of node i , and $\mu_{A_i}(x)$ and $\mu_{B_i}(y)$ can adopt any fuzzy membership function (MF). In this paper, Gaussian MFs are used:

$$\text{gaussian}(x, c, \sigma) = e^{-\frac{1}{2} \left(\frac{x-c}{\sigma} \right)^2} \quad (4)$$

where c is center of Gaussian membership function and σ is standard deviation of this cluster. Layer 2, every node in the second layer represents the ring strength of a rule by multiplying the incoming signals and forwarding the product as:

$$O_i^2 = \omega_i = \mu_{A_i}(x) \mu_{B_i}(y), \quad i = 1, 2. \quad (5)$$

Layer 3, the i th node in this layer calculates the ratio of the i th rule's ring strength to the sum of all rules' ring strengths:

$$O_i^3 = \bar{\omega}_i = \frac{\omega_i}{\omega_1 + \omega_2}, \quad i = 1, 2 \quad (6)$$

where $\bar{\omega}_i$ is referred to as the normalized ring strengths.

Layer 4, the node function in this layer is represented by:

$$O_i^4 = \bar{\omega}_i z_i = \bar{\omega}_i (p_i x + q_i y + r_i), \quad i = 1, 2 \quad (7)$$

where $\bar{\omega}_i$ is the output of layer 3, and $\{p_i, q_i, r_i\}$ is the parameter set. Parameters in this layer are referred to as the consequent parameters.

Layer 5, the single node in this layer computes the overall output as the summation of all incoming signals:

$$O_1^5 = \sum_{i=1}^2 \bar{\omega}_i z_i = \frac{\omega_1 z_1 + \omega_2 z_2}{\omega_1 + \omega_2} \quad (8)$$

It is seen from the ANFIS architecture that when the values of the premise parameters are fixed, the overall output can be expressed as a linear combination of the consequent parameters:

$$z = (\bar{\omega}_1 x) p_1 + (\bar{\omega}_1 y) q_1 + (\bar{\omega}_1) r_1 + (\bar{\omega}_2 x) p_2 + (\bar{\omega}_2 y) q_2 + (\bar{\omega}_2) r_2 \quad (9)$$

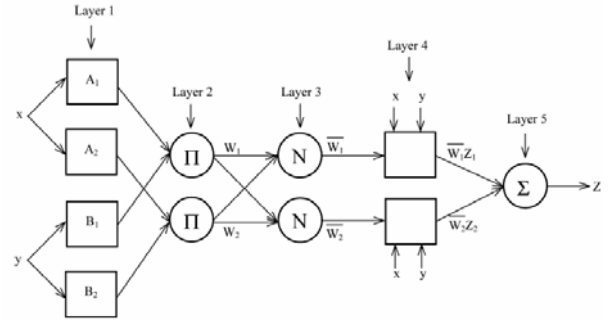


Fig.1: ANFIS architecture. Π, N, Σ are defined in (5), (6), (8) respectively.

The hybrid learning algorithm [21, and 33] combining the least square method and the back propagation (BP) algorithm can be used to solve this problem. This algorithm converges much faster since it reduces the dimension of the search space of the BP algorithm. During the learning process, the premise parameters in layer 1 and the consequent parameters in layer 4 are tuned until the desired response of the FIS is achieved. The hybrid learning algorithm has a two-step process. First, while holding the premise parameters fixed, the functional signals are propagated forward to layer 4, where the consequent parameters are identified by the least square method. Second, the consequent parameters are held fixed while the error signals, the derivative of the error

measure with respect to each node output, are propagated from the output end to the input end, and the premise parameters are updated by the standard BP algorithm.

3. Neuro-fuzzy based functional inverse approach

It is noted that MATLAB evaluates functional inverse using "finverse" instruction but fail in many functions as $y = x^4 - x^2 - 4x - 3$ and so on. This subject motivates us for presentation of new numerical approach based on combining of artificial neural network and fuzzy logic and present suitable and easy available way in MATLAB software. In this section a neuro-fuzzy based functional inverse is studied. The structure of the proposed adaptive system is shown in Fig 2. In this method, we have substituted neural network with ANFIS with better features.

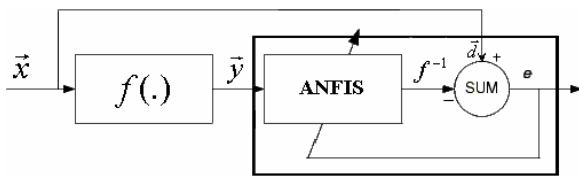


Fig.2: The structure of the proposed neuro-fuzzy based functional inverse

By placing the desired function in series with ANFIS, ANFIS becomes the inverse of the required function when e gets very small. As shown in the figure the process requires input signal \vec{x} as the desired signal \vec{d} path to keep the data at the summation synchronized. Error e tune parameters of FIS with hybrid learning. With putting of a function which its inverse is required, ANFIS is trained for finding functional inverse. Hereon a note is important about generating of learning data. In the proposed scheme, \vec{x} is generated using Monte Carlo simulation and ANFIS interpolates points that don't exist in the learning set.

ANFIS have powerful capability of expressing relationship between input-output variables. In fact they constitute a powerful interpolation tool and it is always possible to develop a structure that approximate a function with a given precision. It can approximate a function and interpret the result due to concept of desired function. It has capability to explain about the system.

ANFIS contains two stages, in the first stage; fuzzy inference system (FIS) is generated using fuzzy subtractive clustering and accomplishes this by extracting a set of rules that models the data behavior and then uses linear least squares estimation to determine each rule's consequent equations. Tuning of FIS parameters is performed using hybrid learning. So in the proposed approach can calculate inverse of time varying functions only with tuning of generated fuzzy model with the least square and the back

propagation (BP) algorithms to form of hybrid learning algorithm which able into tracking of time varying functions.

The MATLAB¹ (2004) toolbox function Genfis2 performs cluster analysis in order to generate fuzzy sets due to projections of clusters in the multidimensional space onto the corresponding axis. This function uses the subtractive clustering algorithm to determine the number of rules and antecedent MFs and then uses global linear LSE to determine each rule's consequent equations. Also instruction of "anfis" from MATLAB uses a hybrid learning algorithm to identify the membership function parameters of single-output, Sugeno type fuzzy inference systems (FIS). A combination of least-squares and back propagation gradient descent methods are used for training FIS membership function parameters to model a given set of input/output data.

The proposed algorithm is described as follows,

Comment: Functional inverse using ANFIS

Comment: $f(\cdot)$ is desired function and $[x_{\min}, x_{\max}]$ is duration of variable which functional inverse needs.

Initialize:

Sustainable error in each sub-duration is $Thre_Err$,

Maximum number of sub-duration is $MaxNumDuration$,

Maximum number of samples which Monte Carlo simulation uses: $N_MonteCarlo$

Initial Error (Err) is defined large value

While $Err > Thre_Err$

Generate $N_MonteCarlo$ samples include $x, f(x)$ in duration $L = x_{\max} - x_{\min}$;

Using Genfis2 MATLAB function for obtaining of f^{-1}

Tuning of FIS structure using Anfis MATLAB function

Measurement of error in duration L over test samples (Err)

If $Err > Thre_Err$

Duration is divided to two parts until number of splitting reach to $MaxNumSplit$.

Operation continue over each parts independently,

End

End While

¹ Fuzzy Logic Toolbox for use with MATLAB, 2000. The Math Works Inc., USA.

In the above algorithm Err is obtained as follow which is normalized error between $f(g(x))$ and x in duration $[x_{min}, x_{max}]$.

$$Err = \frac{\|f(g(x)) - x\|}{|x_{min} - x_{max}|} \quad (10)$$

Also with small variation in above algorithm, we can find inverse of time varying functions. Of course variations of function must be little. In the tracking phase, a reasonable assumption is that the variation of function in time is according to a first-order Markov process. The following relations shows these variation

$$f_{n+1}(x) = af_n(x) + \omega_n \quad (11)$$

where, a is a constant and ω_n is the noise vector in the n^{th} step, which has zero mean with correlation matrix Φ . We add following algorithm for tracking of functional inverse with time varying function.

Comment: This part is added to above part

Comment: Time varying functional inverse; Tracking part

Comment: $f(x,t)$ is time varying desired function.

In each slice time Δt

While $Err > Thre_Err$

FIS structure is tuned using "Anfis" MATLAB function

Measurement of error in duration L over test samples (Err)

If $Err > Thre_Err$

Duration is divided to two parts until number of splitting reach to $MaxNumSplit$.

Operation continue over each parts independently,

End

End While

4. Experimental results

Three examples are considered as follows.

Example 1: Consider the following function for obtaining functional inverse in duration of $[1, 2]$.

$$f(x) = x^4 - x^2 - 4x - 3 \quad (12)$$

MATLAB software has function of "finverse" which use MAPLE can not find f^{-1} . For applying the proposed method, we put $N_MonteCarlo$ equal to 2000 points, $L = x_{max} - x_{min} = 1$, $Thre_Err = 0.001$, $MaxNumSplit = 10$. In the first step, $f(g(x))$ per x is obtained as illustrated in Fig 3. We expect it is to form of beeline but obtained

error is $Err = 0.0018$ which is bigger than $Thre_Err$ also f and f^{-1} are shown in Fig 4, 5. Because of Err is bigger than $Thre_Err$ duration split to two parts.

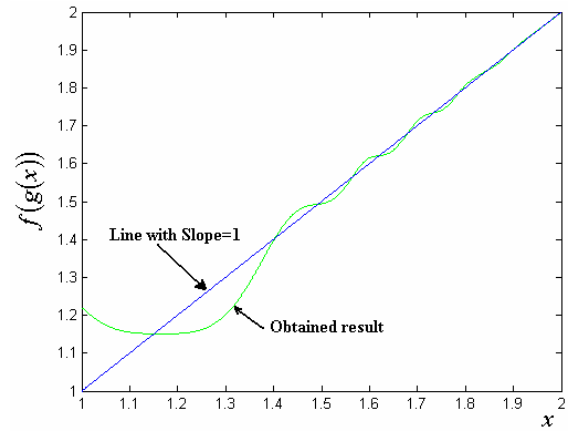


Fig.3: $f(g(x))$ per x for $f(x) = x^4 - x^2 - 4x - 3$ in $[1, 2]$.

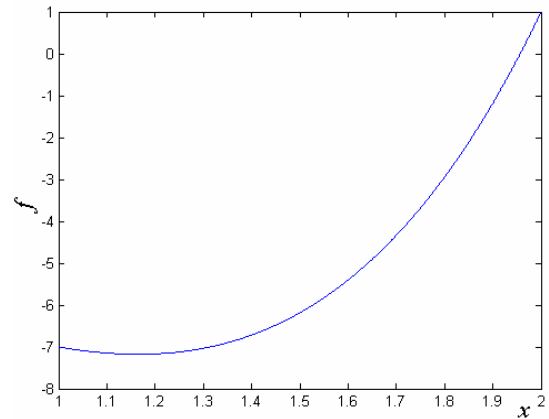


Fig.4: $f(x) = x^4 - x^2 - 4x - 3$

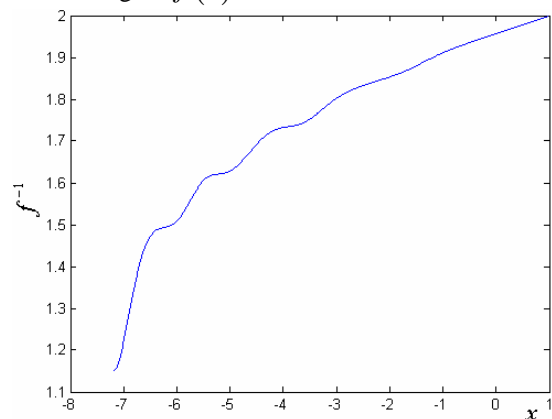


Fig.5: Obtained f^{-1} for example 1.

Now, two duration are mentioned $[1, 1.5]$ and $[1.5, 2]$. In duration $[1.5, 2]$ obtained error is $1.1373e-005$ and result is acceptable but in duration $[1, 1.5]$ $f(g(x))$ per x is shown in Fig 6 and error is 0.0033 , so this duration is splitted to $[1, 1.25]$, $[1.25, 1.5]$ and

this work continues until functional inverse is obtained properly.

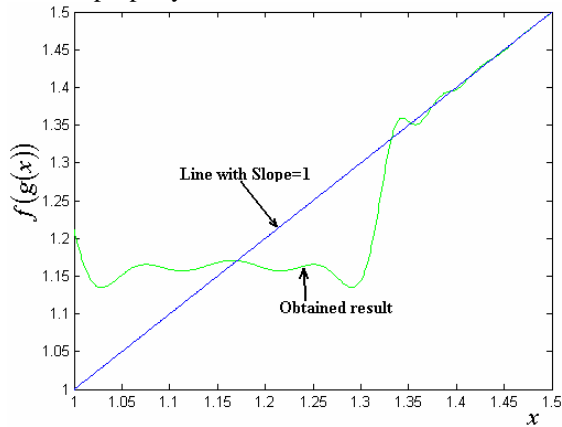


Fig.6: $f(g(x))$ per x for $f(x) = x^4 - x^2 - 4x - 3$ in $[1, 1.5]$

Example 2: Consider the following complex function for obtaining functional inverse in duration of $[1, 10]$.

$$f(x) = x^2 \sin(x) + x^3 \log(1+x) \tanh(x) - 3x + 5 \quad (13)$$

In this function, obtained result in total of desired duration is admissible with normalized error equal to $9.8153e-004$ which $f(g(x))$ per x is shown in Fig 7.

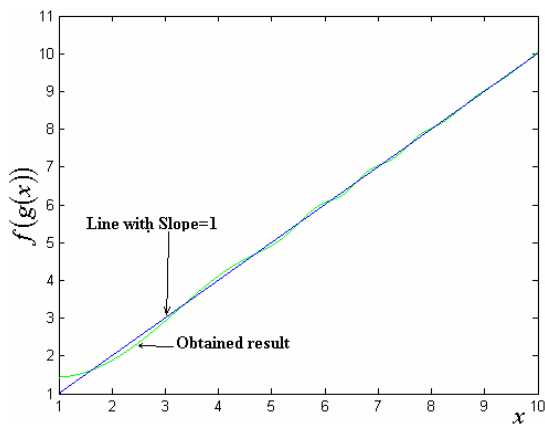


Fig.7: $f(g(x))$ per x in example 2 in $[1, 10]$

Example 3: This example appropriate to time varying function. Consider the following time varying function in duration of $[1, 3]$ for x and t is between $[1, 1.75]$.

$$f(x, t) = tx^2 + t^2. \quad (13)$$

We selected Δt in this example 0.25 and after ending of first phase of the proposed algorithm, tracking process is performed for each time slice only with tuning of FIS parameters using "Anfis" MATLAB function. Time varying function for $\Delta t = 0.25$ and $t \in [1, 1.75]$ and $x \in [1, 3]$ is shown in Fig 8. Obtained error in each stage is smaller than

$Thre_Err = 0.001$ and functional inverse is shown in Fig 9.

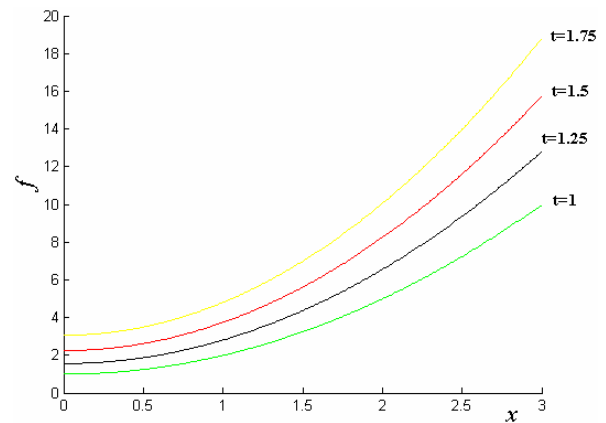


Fig.8: $f(x, t) = tx^2 + t^2$

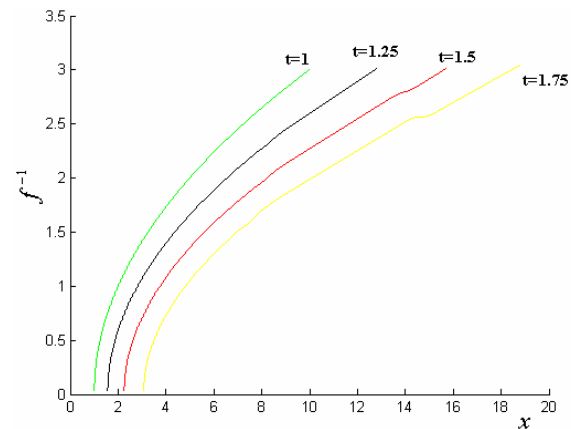


Fig.9: Fig.5: Obtained f^{-1} for example 3.

5. Conclusions

Functional inverse not only is used in mathematics but also have many applications in engineering and sciences. The inverse function may be interested for finding the source from the data. The development of inverse learning was initiated from adaptive signal processing and control of linear systems. One way to try to learn an inverse function is to generate a set of training data. In this paper, a new and easy available numerical method is presented in calculation of functional inverse based on ANFIS (Adaptive Network based Fuzzy Inference System) model. The ANFIS is available without great difficulty in MATLAB software. We presented ANFIS-based approach for the first time for obtaining inverse of mathematical functions. The proposed approach includes two main stages, in the first step; some limited points were sampled from desired function using Monte Carlo simulation and second step contained training of ANFIS system that input system is output points of function and ANFIS desired values are input values to function. One of notes in the proposed approach is calculation of inverse of time varying functions only with tuning of generated fuzzy model with the least square and the back propagation (BP) algorithms to form of hybrid learning algorithm

which able into tracking of time varying functions. Experimental results over many functions showed superiority of the proposed method relative MATLAB software.

References:

[1] M. Hori, "Application of spectral decomposition of Green's function to linear inverse problem," *Engineering Analysis with Boundary Elements* 28 (2004) 183–193.

[2] Widrow, B., and Stearns, S.D., "Adaptive Signal Processing", Prentice-Hall, Englewood Cliffs, NJ, 1984.

[3] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[4] S.E. El-Khamy, M.M. Hadhoud, M.I. Dessouky, B.M. Salam, F.E. Abd El-Samie, "Efficient implementation of image interpolation as an inverse problem," *Digital Signal Processing* 15 (2005) 137–152.

[5] X. Lin, J. Liu, H. Lee, and H. Liu, "A 2.5- to 3.5-Gb/s Adaptive FIR Equalizer with Continuous-Time Wide-Bandwidth Delay Line in 0.25- μ m CMOS," *IEEE Journal of Solid-State Circuits*, Vol. 41, no.8, pp.1908-1918, Aug. 2006.

[6] A. Gonzalez, M. de Diego, M. Ferrer, and G. Piñero, "Multichannel Active Noise Equalization of Interior Noise," *IEEE Trans. on Audio, Speech, and Language Processing*, vol.14, no.1, pp.110-122, Jan. 2006.

[7] A. González, M. Ferrer, M. de Diego, G. Piñero, and J. J. Garcia-Bonito, "Sound quality of low-frequency and car engine noises after active noise control," *J. Sound Vibr.*, vol. 265, no. 3, pp. 663–679, 2003.

[8] W. Ser, P. Wang, M. Zhang, "Loudspeaker Equalization with Post-Processing," *EURASIP Journal on Applied Signal Processing* 2002:11, 1296–1300.

[9] DeMers, D., and Kreutz-Delgado, K. (1996), "Canonical Parameterization of Excess motor degrees of freedom with self organizing maps", *IEEE Trans Neural Networks*, 7, 43-55.

[10] DeMers, D., and Kreutz-Delgado, K. (1997), "Inverse kinematics of dexterous manipulators," in Omidvar, O., and van der Smagt, P., (eds.) *Neural Systems for Robotics*, San Diego: Academic Press, pp. 75-116.

[11] Bishop, C.M. (1995), *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press.

[12] Huber, P.J. (1981), *Robust Statistics*, NY: Wiley.

[13] Rohwer, R., and van der Rest, J.C. (1996), "Minimum description length, regularization, and multimodal data," *Neural Computation*, 8, 595-609.

[14] Dennis, J.E. and Schnabel, R.B. (1983) *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall.

[15] Kindermann, J., and Linden, A. (1990), "Inversion of Neural Networks by Gradient Descent," *Parallel Computing*, 14, 277-286, <ftp://icsi.Berkeley.EDU/pub/ai/linden/KindermannLinden.IEEE92.ps.Z>

[16] Bertsekas, D. P. (1995), *Nonlinear Programming*, Belmont, MA: Athena Scientific.

[17] White, H. (1992), "Nonparametric Estimation of Conditional Quantiles Using Neural Networks," in Page, C. and Le Page, R. (eds.), *Proceedings of the 23rd Symposium on the Interface: Computing Science and Statistics*, Alexandria, VA: American Statistical Association, pp. 190-199.

[18] A. Cichocki, R. Unbehauen, K. Weinzierl and R. Holzel, "A new neural network for solving linear programming problems," *European J. of Operational Research* 93, 244-256, (1996).

[19] Castro, A., Miranda, V., 2002. Mapping neural networks into rule sets and making their hidden knowledge explicit application to spatial load forecasting. In: *Proceedings of the 14th Power System Computation Conference*.

[20] Mitra, S., Hayashi, Y., 2000. Neuro-fuzzy rule generation: survey in soft computing framework. *IEEE Transactions on Neural Networks* 3, 748–768.

[21] Jang, J.-S.R. (1993). ANFIS: adaptive-network-based fuzzy inference system. *IEEE Transaction on System Man and Cybernet*, 23(5), 665–685.

[22] A. I. Nuno, B. Arcay, J.M. Cotos, J. Varela, "Optimization of Fishing Predictions by Means of Artificial Neural Networks, ANFIS, Functional Networks and Remote Sensing Images," *Expert Systems with Applications*, vol.29, pp.356–363, 2005.

[23] A. Noureldin, A. El-Shafie, M. R. Tahab, "Optimizing Neuro-Fuzzy Modules for Data Fusion of Vehicular Navigation Systems Using Temporal Cross-Validation," *Engineering Applications of Artificial Intelligence*, vol.20, pp.49–61, 2007.

[24] N. Kishor, S.P. Singh, A.S. Raghuvanshi, "Adaptive Intelligent Hydro Turbine Speed Identification with Water and Random Load Disturbances," *Engineering Applications of Artificial Intelligence* (2007), doi:10.1016/j.engappai.2006.11.014.

[25] K. C. Lee, and P. Gardner, "Adaptive Neuro-Fuzzy Inference System (ANFIS) Digital Predistorter for RF Power Amplifier Linearization," *IEEE Trans. on Vehicular Technology*, vol..55, no.1, pp.43-51, Jan. 2006.

[26] E. D. Ubeyli, I. Guler, "Adaptive Neuro-Fuzzy Inference System to Compute Quasi-TEM Characteristic Parameters of Micro Shield Lines with Practical Cavity Sidewall Profiles," *Neurocomputing*, vol.70, pp.296–304, 2006.

[27] H. Qin, S. X. Yang, "Adaptive Neuro-Fuzzy Inference Systems Based Approach to Nonlinear Noise Cancellation for Images," *Fuzzy Sets and Systems* (2007), doi: 10.1016/j.fss.2006.10.028

[28] P. Çivicioglu, "Using Uncorrupted Neighborhoods of the Pixels for Impulsive Noise Suppression with ANFIS," *IEEE Trans. On Image*

Processing, (2007), Digital Object Identifier 0.1109/TIP.2007.891067.

[29] G. Daoming, C. Jie, "ANFIS for High-Pressure Water jet Cleaning Prediction," Surface & Coatings Technology, vol.201, pp.1629–1634, 2006.

[30] A. Depari, A. Flammini, D. Marioli, and Andrea Taroni, "Application of an ANFIS Algorithm to Sensor Data Processing," IEEE Trans. On Instrumentation and Measurment, vol.56, no.1, pp.75-79, Feb. 2007.

[31] K. Assaleh, "Extraction of Fetal Electrocardiogram Using Adaptive Neuro-Fuzzy Inference Systems," IEEE Trans. On Biomedical Engineering, vol.54, no.1, pp.59-68, Jan. 2007.

[32] M-L. Huang, H-Y. Chen, J-J. Huang, "Glaucoma Detection Using Adaptive Neuro-Fuzzy Inference System," Expert Systems with Applications, vol.32, pp.458–468, 2007.

[33] Jang J-SR, Sun CT, Mizutani E, "Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence. Prentice-Hall, Englewood Clis, NJ, 1997.

