The Lagrange Interpolation Polynomial for Neural Network Learning

Khalid Ali Hussien

Mustansiriyah University, Educational College, Baghdad, Iraq

Abstract

One of the methods used to find this polynomial is called the Lagrange method of interpolation. In this research, the Lagrange interpolation method was used in a new neural network learning by develops the weighting calculation in the back propagation training. This proposed developing decrease the learning time with best classification operation results. Also, the Langrage interpolation polynomial was used to process the image pixels and remove the noise the image. This interpolation gives the effective processing in removing the noise and error in the image layers. One of the advantages of this method is reduce the noise to minimum value by replacing the noisy pixels (detected by Lagrange Back propagation Neural Network LBPNN) by results calculated by the Lagrange interpolation with high speed processing and best RMSE results.

Keyword:

Interpolation polynomial, Lagrange Interpolation, Lagrange neural network, neural learning, Lagrange learning, Denoising, Neural denoising.

1. Introduction

The numerical analysis is both a science and an art is a cliché to specialists in to the field but is often misunderstood by non-specialists. Is calling it an art and a science only a euphemism to hide the fact that numerical analysis is not a sufficiently precise discipline to merit being called a science? Is it true that "numerical analysis" is something of a misnomer because the classical meaning of analysis in is not applicable to numerical work? In fact, the answer to both these questions is no. The juxtaposition of science and art is due instead to an uncertainty principle which often occurs in solving problems, namely that to determine the best way to solve a problem may require the solution of the problem itself in other cases. The best way to solve u problem may depend upon a knowledge of the properties of the functions involved which is unobtainable either theoretically or practically. [1]

Numerical analysis is the study of algorithms that use numerical values (as opposed to general symbolic manipulations) for the problems of continuous mathematics (as distinguished from discrete mathematics). One of the earliest mathematical writings is the Babylonian tablet YBC 7289, which gives a sexagesimal numerical approximation of √2, the length of the diagonal in a unit square.[2,3] Being able to compute the sides of a triangle (and hence, being able to compute squares roots) is extremely important, for instance, in carpentry and construction.[4]

Numerical analysis continues this long tradition of practical mathematical calculations. Much like the Babylonian approximation to √2, modern numerical analysis does not seek exact answers, because exact answers are often impossible to obtain in practice. Instead, much of numerical analysis is concerned with obtaining approximate solutions while maintaining reasonable bounds on errors. [3]

Numerical analysis naturally finds applications in all fields of engineering and the physical sciences, but in the 21st century, the life sciences and even the arts have adopted elements of scientific computations. Ordinary differential equations appear in the movement of heavenly bodies (planets, stars and galaxies); optimization occurs in portfolio management; numerical linear algebra is important for data analysis; stochastic differential equations and Markov chains are essential in simulating living cells for medicine and biology.[3],[4]

Polynomials can be used to approximate more complicated curves, for example, the shapes of letters in typography, given a few points. A relevant application is the evaluation of the natural logarithm and trigonometric functions: pick a few known data points, create a lookup table, and interpolate between those data points. This results in significantly faster computations. Polynomial interpolation also forms the basis for algorithms in numerical quadrature and numerical ordinary differential equations. [5]

Polynomial interpolation is also essential to perform sub-quadratic multiplication and squaring such as Karatsuba multiplication and Toom–Cook multiplication, where an interpolation through points on a polynomial which defines the product yields the product itself. For example, given $a = f(x) = a_0x^0 + a_1x^1 + ...$ and $b = g(x) = b_0x^0 + b_1x^1 + ...$ then the product ab is equivalent to $W(x) = f(x)g(x)$. Finding points along W(x) by substituting x for small values in f(x) and g(x) yields points on the curve. Interpolation based on those points will yield the terms of W(x) and subsequently the product ab. In the case of Karatsuba multiplication this technique is substantially

Manuscript received March 5, 2011
Manuscript revised March 20, 2011

faster than quadratic multiplication, even for modest-sized inputs. This is especially true when implemented in parallel hardware.[5]

In this research, numerical analysis in proposed a new technique for learning the neural network. The Lagrange interpolation polynomial was used in back propagation neural learning in order to increase the decision accuracy of the neural network with decrease the learning time and more stability of neural network. Also, the Lagrange interpolation polynomial was used in image denoising process under control of the proposed neural network.

2. The Lagrange Interpolation Polynomial

The problem of constructing a continuously defined function from given discrete data is unavoidable whenever one wishes to manipulate the data in a way that requires information not included explicitly in the data. The relatively easiest and in many applications often most desired approach to solve the problem is *interpolation*, where an approximating function is constructed in such a way as to agree perfectly with the usually unknown original function at the given measurement points. In the practical application of the finite calculus of the problem of interpolation is the following: given the values of the function for a finite set of arguments, to determine the value of the function for some intermediate argument. [6]

2.1 The Problem of Interpolation

The problem of interpolation consists in the following: Given the values yi corresponding to xi, i = 0, 1, 2, . . . , n, a function f(x) of the continuous variable x is to be determined which satisfies the equation:

yi = f(xi) for i = 0, 1, 2 . . . , n

and finally f(x) corresponding to x = x0 is required. (i.e. x0 different from xi, i = 1, n.)

In the absence of further knowledge as to the nature of the function this problem is, in the general case, indeterminate, since the values of the arguments other than those given can obviously assigned arbitrarily. [6]

If, however, certain analytic properties of the function be given, it is often possible to assign limits to the error committed in calculating the function from values given for a limited set of arguments. For example, when the function is known to be representable by a polynomial of degree n, the value for any argument is completely determinate when the values for n + 1 distinct arguments are given.

2.2 Lagrange Interpolation

Consider the function f: [x0, xn] → R given by the following table of values:

x_k	x_0	x_1	...	x_n
$f(x_k)$	$f(x_0)$	$f(x_1)$	...	$f(x_n)$

x_k are called *interpolation nodes*, and they are not necessary equally distanced from each other. We seek to find a polynomial P(x) of degree n that approximates the function f(x) in the interpolation nodes, i.e. [6]

$$f(x_k) = P(x_k);\ k = 0, 1, 2, \ldots, n.$$

The Lagrange interpolation method finds such a polynomial without solving the system.

Theorem: Lagrange Interpolating Polynomial [6]

The Lagrange interpolating polynomial *is the polynomial of degree* n *that passes through* (n + 1) *points* y0 = f(x0), y1 = f(x1), . . . yn = f(xn) . *Let:*

$$P(x) = \sum_{j=0}^{n} P_j(x)$$

Where

$$P_j(x) = y_j \prod_{k=0, k\neq j}^{n} \frac{x - x_k}{x_j - x_k}.$$

Written explicitly:

$$P(x) = \frac{(x-x_1)(x-x_2)\cdots(x-x_n)}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)}y_0 + \frac{(x-x_0)(x-x_2)\cdots(x-x_n)}{(x_1-x_0)(x_1-x_2)\cdots(x_1-x_n)}y_1 + \cdots + \frac{(x-x_0)(x-x_1)\cdots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\cdots(x_n-x_{n-1})}y_n.$$

Lagrange interpolating polynomials are implemented in *Mathematica* as Interpolating Polynomials[data,var]. For the case n = 4, i.e. interpolation through five points, we have:

$$P(x) = \frac{(x-x_1)(x-x_2)(x-x_3)(x-x_4)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)(x_0-x_4)}y_0 + \frac{(x-x_0)(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)(x_1-x_4)}y_1 + \frac{(x-x_0)(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)(x_2-x_4)}y_2 + \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)(x_3-x_4)}y_3 + \frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_0)(x_4-x_1)(x_4-x_2)(x_4-x_2)}y_4$$

and

$$P'(x) = \frac{(x-x_2)(x-x_3)(x-x_4)+(x-x_1)(x-x_3)(x-x_4)+(x-x_1)(x-x_2)(x-x_4)+(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)(x_0-x_4)}y_0 + \frac{(x-x_2)(x-x_3)(x-x_4)+(x-x_0)(x-x_3)(x-x_4)+(x-x_0)(x-x_2)(x-x_4)+(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)(x_1-x_4)}y_1 + \frac{(x-x_1)(x-x_3)(x-x_4)+(x-x_0)(x-x_3)(x-x_4)+(x-x_0)(x-x_1)(x-x_4)+(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)(x_2-x_4)}y_2 + \frac{(x-x_1)(x-x_2)(x-x_4)+(x-x_0)(x-x_2)(x-x_4)+(x-x_0)(x-x_1)(x-x_4)+(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)(x_3-x_4)}y_3 + \frac{(x-x_1)(x-x_2)(x-x_3)+(x-x_0)(x-x_2)(x-x_3)+(x-x_0)(x-x_1)(x-x_3)+(x-x_0)(x-x_1)(x-x_2)}{(x_4-x_0)(x_4-x_1)(x_4-x_2)(x_4-x_2)}y_4$$

Note that the function P(x) passes through the points (xi, yi), i.e. P(xi) = yi.

For Examples: The Lagrange interpolating polynomial is given by[7]

$$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$$

where n in $f_n(x)$ stands for the n^{th} order polynomial that approximates the function $y = f(x)$ given at $n+1$ data points as $(x_0, y_0), (x_1, y_1), \ldots\ldots, (x_{n-1}, y_{n-1}), (x_n, y_n)$, and

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

$L_i(x)$ is a weighting function that includes a product of $n-1$ terms with terms of $j = i$

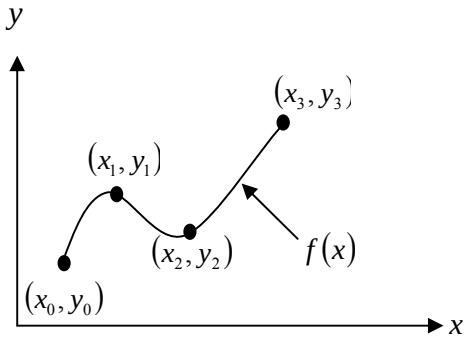

Figure 1 Interpolation of discrete data.[9]

Let Y = F(x) such that $y_o = f(x_o)$, $y_1 = f(x_1)$, $y_2 = f(x_2)$, … , $y_n = f(x_n)$, then to estimate value of f(x) we use :[8]

x	x_o	X_1	x_2	…	x_n
F(x)	y_o	Y_1	y_2	…	y_n

$$F(x^*) = \sum_{j=0}^{n} f(x_j) \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{(x^* - x_i)}{(x_j - x_i)}$$

…….. (1)

Example 1: By Lagrange formula , find the value of f(3) and f(5) frome the table .

X	0	1	2	4
F(x)	1	1	2	5

Solution:

$$F(3) = \sum_{j=0}^{3} f(x_j) \prod_{\substack{i=0 \\ i \neq j}}^{3} \frac{(3 - x_i)}{(x_j - x_i)}$$

$$= f(x_o) \frac{(3-x_1)(3-x_2)(3-x_3)}{(x_o-x_1)(x_o-x_2)(x_o-x_3)} + f(x_1) \frac{(3-x_o)(3-x_2)(3-x_3)}{(x_1-x_o)(x_1-x_2)(x_1-x_3)} +$$

$$f(x_2) \frac{(3-x_o)(3-x_1)(3-x_3)}{(x_2-x_o)(x_2-x_1)(x_2-x_3)} + f(x_3) \frac{(3-x_o)(3-x_1)(3-x_2)}{(x_3-x_o)(x_3-x_1)(x_3-x_2)}$$

F(3) = 3.5

So by the same way we have F(5)=6 .

2.3 Inverse Interpolation

As shown, the equation of how to interpolation for function value corresponding to a given independent variable x was addressed. Suppose that, we have now reverse the equation so that we seek to determine on x value corresponding to a given functional value , then the problems becomes inverse interpolation , so we have : [8]

$$x^* = \sum_{j=0}^{n} x_j \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{(y^* - y_i)}{(y_j - y_i)} \quad \ldots\ldots\ldots(2)$$

Example: find the value of x^* , when $y^* = 2$,

Y	1	3	5
x	15	20	2

Solution:

$$x^* = \sum_{j=0}^{2} x_j \prod_{\substack{i=0 \\ i \neq j}}^{2} \frac{(y^* - y_i)}{(y_j - y_i)}$$

$$= x_o \frac{(2-y_1)(2-y_2)}{(y_o-y1)(y_o-y_2)} + x_1 \frac{(2-y_o)(2-y2)}{(y_1-y_o)(y_1-y_2)} + x_2 \frac{(2-y_o)(2-y_1)}{(y_2-y_o)(y_2-y_1)}$$

$$= 20.375.$$

3. Example 2 [7]

The upward velocity of a rocket is given as a function of time in Table 1 and Fig.(2).

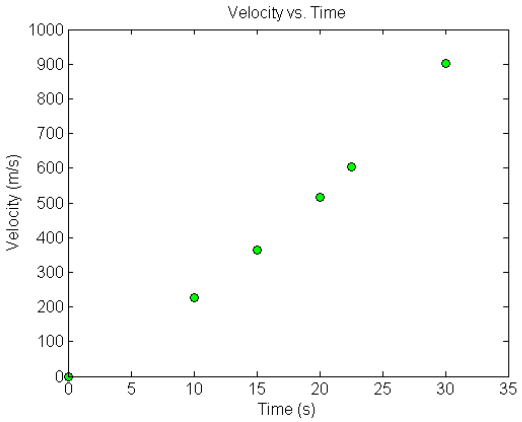

Fig. (2) Graph of velocity vs. time data for the rocket example.

Table 1: Velocity as a function of time.

t (s)	$v(t)$ (m/s)
0	0
10	227.04
15	362.78
20	517.35
22.5	602.97
30	901.67

Determine the value of the velocity at $t = 16$ seconds using a first order Lagrange polynomial.

Solution

For first order polynomial interpolation (also called linear interpolation as shown in Fig. (3)), the velocity is given by

$$v(t) = \sum_{i=0}^{1} L_i(t) v(t_i)$$
$$= L_0(t)v(t_0) + L_1(t)v(t_1)$$

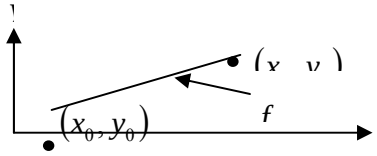

Fig.(3) Linear interpolation.

Since we want to find the velocity at $t = 16$, and we are using a first order polynomial, we need to choose the two data points that are closest to $t = 16$ that also bracket $t = 16$ to evaluate it. The two points are $t_0 = 15$ and $t_1 = 20$.

Then

$$t_0 = 15,\ v(t_0) = 362.78$$
$$t_1 = 20,\ v(t_1) = 517.35$$

gives

$$L_0(t) = \prod_{\substack{j=0 \\ j\neq 0}}^{1} \frac{t-t_j}{t_0-t_j} = \frac{t-t_1}{t_0-t_1}$$

$$L_1(t) = \prod_{\substack{j=0 \\ j\neq 1}}^{1} \frac{t-t_j}{t_1-t_j} = \frac{t-t_0}{t_1-t_0}$$

Hence

$$v(t) = \frac{t-t_1}{t_0-t_1} v(t_0) + \frac{t-t_0}{t_1-t_0} v(t_1)$$

$$= \frac{t-20}{15-20}(362.78) + \frac{t-15}{20-15}(517.35), \quad 15 \le t \le 20$$

$$v(16) = \frac{16-20}{15-20}(362.78) + \frac{16-15}{20-15}(517.35)$$
$$= 0.8(362.78) + 0.2(517.35)$$
$$= 393.69 \text{ m/s}$$

You can see that $L_0(t) = 0.8$ and $L_1(t) = 0.2$ are like weightages given to the velocities at $t = 15$ and $t = 20$ to calculate the velocity at $t = 16$.

4. Back –Propagation Neural Network

The back propagation neural is a multilayered, feed forward neural network and is by far the most extensively used. Back Propagation works by approximating the non-linear relationship between the input and the output by adjusting the weight values internally. A supervised learning algorithm of back propagation is utilized to establish the neural network modeling. A normal back-propagation neural (BPN) model consists of an input layer, one or more hidden layers, and output layer. There are two parameters including learning rate ($0 < \alpha < 1$) and momentum ($0 < \eta < 1$) required to define by user. The theoretical results showed that one hidden layer is sufficient for a BP network to approximate any continuous mapping from the input patterns to the output patterns to an arbitrary degree freedom. The selection and nodes of hidden layers primarily affect the classification performance. The following figure shows the topology of the black-propagation neural network that includes and input layer, one hidden layer and output layer. [9]

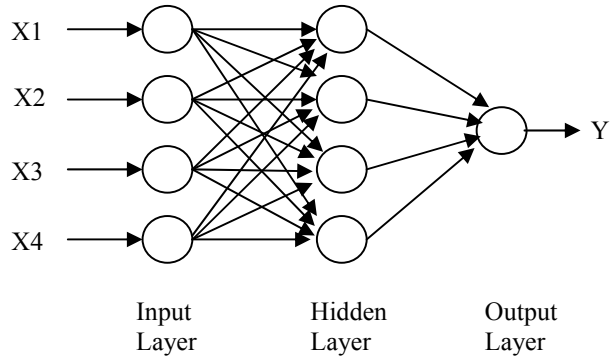

Fig. 4 Two category model of back-propagation

5. Proposed Lagrange BPN network

A new learning for BPN network was proposed to achieve more stability of the neural network and increase the accuracy of the BPN network results using the modification of Lagrange interpolation polynomial. The Lagrange interpolation used to change the ratio of learning weights for each layer of neural network to add more stability in each neural network layer and to reduce the minimum and global energy loops of back propagation neural learning. Below the proposed Lagrange BPN network learning algorithm:

Step 1: Design the structure of neural network and input parameters of the network.

Step2: Get initial weights W and initial (threshold values) from randomizing.

Step 3: Input training data matrix X and output matrix Y.

Step 4: Compute the output vector of each neural units.

(a) Compute the output vector H of the hidden layer

$$net_k = \sum w_{ik} x_i - \theta_k$$

$$H_k = f(net_k)$$

(b) Compute the output vector Y of the output layer

$$net_j = \sum w_{kj} H_i - \theta_j$$

$$Y_j = f(net_j)$$

Step 5: Compute Lagrange interpolation for Hidden and Output layer

$$L_k(n) = \sum_{k=0}^{n} net_k \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x^* - x_i)}{(x_k - x_i)}$$

$$L_j(n) = \sum_{j=0}^{n} H_j \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{(x^* - x_i)}{(x_j - x_i)}$$

Step 6: Compute the modification of W and θ ($\eta$is the learning rate, α is the momentum coefficient)

(a) Compute the modification of W and θ of the output layer

$$\Delta w_{kj}(n) = \eta H_K + \alpha \Delta w_{kj}(n-1) + \eta \alpha L_j(n)$$
$$\Delta \theta_j(n) = \eta L_j(n) + \alpha \Delta \theta_j(n-1)$$

(b) Compute the modification of W andθ of the hidden layer

$$\Delta w_{jk}(n) = \eta X_j + \alpha \Delta w_{jk}(n-1) + \eta \alpha L_k(n)$$
$$\Delta \theta_\kappa(n) = -\eta L_k(n) + \alpha \Delta \theta_\kappa(n-1)$$

Step 7: Renew W and θ

(a) Renew W and _ of the output layer

$$w_{kj}(p) = w_{kj}(p-1) + \Delta w_{kj}$$
$$\theta_j(p) = \theta_j(p-1) + \Delta \theta_j$$

(b) Renew W and θ of the hidden layer

$$w_{jk}(p) = w_{jk}(p-1) + \Delta w_{jk}$$
$$\theta_k(p) = \theta_\kappa(p-1) + \Delta \theta_k$$

Step 8: Repeat step 3 to step 7 until converge

6. The proposed Lagrange Back Propagation Neural Network System (LBNPNN)

The proposed Lagrange back propagation neural network was used to develop the image denoising system explains in [10]. In begin; we must explain the denoising operation. Being a simple inverse problem, the denoising is a challenging task and basically addresses the problem of estimating a signal from the noisy measured version available from that. A very common assumption is that the present noise is additive zero-mean white Gaussian with standard deviation *ó*. Many solutions have been proposed for this problem based on different ideas, such as spatial adaptive filters, diffusion enhancement, statistical modeling, transfer domain methods, order statistics and yet many more. Among these methods, a method based on with sparse and redundant representations has recently attracted lots of attentions. Many researchers have reported that such representations are highly effective and promising toward this stated problem [11].

There are many different cases of distortions. One of the most prevalent cases is distortion due to additive white Gaussian noise which can be caused by poor image acquisition or by transferring the image data in noisy communication channels. Early methods to restore the image used linear filtering or smoothing methods. These methods where simple and easy to apply but their effectiveness is limited since this often leads to blurred or smoothed out in high frequency regions.[12]

All denoising methods use images artificially distorted with well defined white Gaussian noise to achieve objective test results. Note however that in real world images, to discriminate the distorting signal from the "true" image is an ill posed problem since it is not always well defined whether a pixel value belongs to the image or it is part of unwanted noise.[12]

The proposed system used to reduce the noise in the image and applying the Lagrange interpolation method with the proposed LBPNN for the denoising operation with the new denoises calculation technique.

After loading image, the proposed system starts calculation of The Lagrange interpolation by using 2D mask and initial the proposed LBPN network with a neural configuration build from (16, 25,1) 16 input nodes, 25 hidden nodes, and one output node. This mask take two rows for each time calculation and repeat calculation until the image completes. From these rows, the calculated The Lagrange values for each row depend on the previous row. In the same time, the proposed LBPN network will apply to the same 2D mask pixels to distinguish between normal pixel and noisy pixel. Where, the proposed LBPN network learned using some types of the training image sets with different types of noise distribution.

For each pixel in the selected 2D mask, the Lagrange interpolation resulted values will replace the pixel value if proposed LBPN network flagged the tested pixel is a noisy pixel. The proposed system steps are:

1. Loading the image.
2. Divide the image into squares with size 16x16 pixels.
3. Applying The Lagrange interpolation calculation on the loaded image squares parts.
4. Applying the proposed LBPN network.
5. if proposed LBPN network flags the pixel is noisy then applying the denoising operation by replace noisy by Lagrange interpolation resulted value.
6. Calculate RMS
7. Save the resulted image. The block diagram of the proposed system is as shown below:

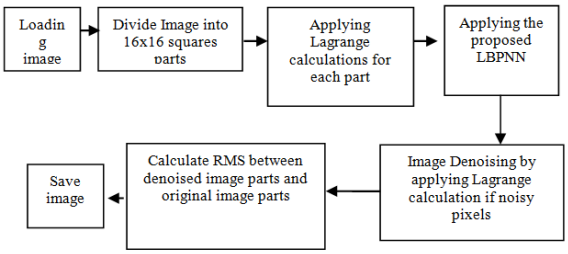

Fig.(6) The proposed system block diagram

7. Experimental Results & Discussions

The performance of the Lagrange interpolation method and Proposed Lagrange Interpolation polynomial Back Propagation Neural Network that have been proposed in this research is investigated with simulations and give a very good results for some types of noise like Gaussian noise and Laplace noise. Also, denoising is carried out for image with noise of variance types. The proposed mask has the best nonlinear filtering possible due to the LBPNN and Lagrange interpolation polynomial math calculations, the mask replacing noisy pixels depend on LBPNN decision and the content of the neighbor pixels values in the suggested mask. Fig(7) shows some proposed system results.

(S1) before (S1) After

(S2) before (S2) after

Fig (7) some results of the proposed system.

The performance of the different denoising samples is compared in Table 2.

Table 2 The RMSE for the Test Samples after denoising

Sample name	RMSE	Size
S1	43.90	512x512
S2	48.54	640x480
S3	44.98	600x800
S4	47.12	512x 512
S5	45.89	512x480

8. Reference

[1] Ralston, A., & Rabinowtiz, P. A First Course in Numerical Analysis (2nd Edition Ed). (Paperback (Feb. 6, 2001) ed.): Dover Publications.2001.

[2] Gilat, A.. MATLAB: An Introduction with Applications (2nd edition Ed.). John Wiley & Sons. ISBN 0-471-69420-7. (2004).

[3] Leader, J. J. (2004). Numerical analysis and scientific computation: Pearson Addison Wesley.

[4] Hildebrand, F. B. Introduction to numerical analysis: Dover Pubns. (1987).

[5] Crandall, R. E., & Pomerance, C. (2005). Prime numbers: a computational perspective: Springer Verlag.

[6] KovJcs, A., KovJcs, L., & Fazekas, F. The Lagrange Interpolation Formula in Determining the Fluid's Velocity Potential through Profile Grids (pp. 26-29). (2005).

[7] Kaw, A., & Paul, J. Computer Engineering Example on the Lagrange Method of Interpolation . Retrieved 16 Feb 2010, from http://numericalmethods.eng.usf .edu/topics/lagrange_method.html.

[8] Al-Khafaji, A. W., Tooley, J. R., & Al-Khafaji, A. W. Numerical methods in engineering practice: Holt, Rinehart and Winston. (1986).

[9] Lee, M. C., & To, C. Comparison of Support Vector Machine and Back Propagation Neural Network in Evaluating the Enterprise Financial Distress. International Journal of Artificial Intelligence & Applications (IJAIA), 1(3), (2010).

[10] Hussien, A. K., & Naif, J. R. Lagrange Interpolation Polynomial For Images. Iraqi Jorunal of Information Technology , 3(1) , (2010) .

[11] Mathews, J. H., & Fink, K. D. Numerical methods using MATLAB: Prentice Hall. (4th edition Ed.). (2004).

[12] Barthel, K. U., Cycon, H. L., & Marpe, D. (Eds.). (Vols. 5266). (2004).

[13] Anthony, M., & Bartlett, P. L. Neural network learning: Theoretical foundations: Cambridge Univ Pr. (1999).

[14] Wilamowski, B. M., & Yu, H. Neural network learning without backpropagation (Vol. 21, pp. 1793-1803): IEEE.(2010).

[15] Archer, N. P., & Wang, S. Application of the Back Propagation Neural Network Algorithm with Monotonicity Constraints for Two Group Classification Problems* (Vol. 24, pp. 60-75): Wiley Online Library. (1993).

Khalid A. Hussein Bs.c. in mathematics from Iraq , Ms.c. in applied mathematics from Jordan in Al-al Byte university. Senior lecturer in numerical methods ,al-Mustansiryah University / Educational college / computer Dpt.