Modifying Explicit Finite Difference Method by Using Radial Basis Function Neural Network

Omar S. Kasim
omar.saber@uomosul.edu.iq
College of Computer Science and Mathematics
University of Mosul, Mosul, Iraq

Received on: 03/03/2013 **Accepted on: 03/04/2013**

ABSTRACT

In this research, we use artificial neural networks, specifically radial basis function neural network (RBFNN) to improve the performance and work of the explicit finite differences method (EFDM), where it was compared, the modified method with an explicit finite differences method through solving the Murray equation and showing by comparing results with the exact solution that the improved method by using (RBFNN) is the best and most accurate by giving less error rate through root mean square error (RMSE) from the classical method (EFDM).

Keywords: *Artificial Neural Network; Finite Difference; Murray equation***.**

تعديل طريقة الفروق المنتهية الصريحة باستخدام الشبكة العصبية ذات دالة الأساس الشعاعي

عمر صابر قاسم
كلية علوم الحاسوب والرياضيات، جامعة الموصل

تاريخ استلام البحث: 2013/30/30 **تاريخ قبول البحث: 2013/04/30**

الملخص

تم في هذا البحث استخدام أسلوب الشبكات العصبية الاصطناعية وتحديدا شبكة (Radial Basis Function) لتطوير أداء وعمل طريقة الفروقات المنتهية الصريحة, حيث تم مقارنة الطريقة المطورة باستخدام شبكة (RBFNN) مع الطريقة الصريحة للفروقات المنتهية (Explicit Finite Differences Method) وذلك من خلال حل معادلة (Murray), وتبين من خلال مقارنة النتائج مع الحل المضبوط (Exact Solution) أن الطريقة المطورة باستخدام شبكة (RBFNN) هي الأفضل والأكثر دقة من خلال إعطاء اقل نسبة خطأ لمقياس (RMSE) من الطريقة الاعتيادية (EFDM).

الكلمات المفتاحية: الشبكات العصبية الاصطناعية ; الفروقات المنتهية; معادلة *Murray*.

1. Introduction:

Artificial neural networks (ANN) has been widely used in many researches as a very important member of computational intelligence and artificial intelligence. Neural networks which include radial basis function (RBF) networks, are to be very efficient in approximating nonlinear and multivariable functions when the sample training set is selected in a rigorous manner [12]. An (ANN) is applied for solving various problems in industrial applications, such as non-linear control, system diagnosis , data classification, pattern recognition and function approximation [13].

Finite difference method is one of the most popular methods of numerical solution of partial differential equations, which are used to solve many problems that involve unknown functions of several variables, where is distributed in space, or distributed in space and time [11]. In the finite difference method, we approximate the solution by using the numerical operators of the function's derivatives and finding the solution at specific preassigned grids[10].

The proposed or modified method is a hybrid method which is based on finite differences method and an artificial neural network . After we get the numerical solution from finite difference approximation, we get the first stage, and then enter the solution to the (RBFNN) as a second stage of process, which will be used to find the solutions for these points. Where the modified method gave errors less than from the classical method after using the explicit finite difference method in the first stage and the (RBFNN) in the second stage to solve Murray equation problem.

2.Artificial Neural Network (ANN) :

A neural network is a parallel system, which is capable of resolving paradigms that linear computing cannot. An artificial neural network (ANN) is a system based on the operation of biological neural networks [7]. It is composed of a large number of interconnected elements (neurons) working in parallel to solve specific problems. A given (ANN) is configured for the specific application, such as pattern recognition or data classification, through a learning process [8]. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. Neural networks with their ability to derive meaning from imprecise data can be thought as an "expert" and used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.[9] and [12].

2.1. Neural Network Structure : consistent

An artificial neural network is composed of several elements [12] and [13]:

1. **Input layer :** The role of the input units is to receive the raw information that is fed into the network.
2. **Hidden layers :** It is the processing unit for the network. Its activity is determined by the activities of the input units and the weights of the connections between the input and the first row of the hidden units, or between nodes of adjacent hidden layers.
3. **Output layer** : The behavior of the output units depends on the activity of the last row of the hidden units and the weights between the nodes in this row and the output units.
4. **Neuron :** It is the basic element of the neural network. It is a communication conduit that accepts inputs and produces outputs. In the case when a neuron produces output, it becomes active. A neuron will be active when the sum of its inputs satisfies the neuron’s activation function.

3. Radial Basis Function Neural Networks (RBFNN) :

A radial basis function network is an (ANN) that uses radial basis functions as activation functions. It is a linear combination of radial basis functions [6], (RBF) network provides a powerful alternative to multilayer perceptron (MLP) neural networks to approximate or to classify a pattern set. (RBFNN) differs from (MLP) in that the overall input output map is constructed from local contributions of Gaussian axons, require fewer training samples and train faster than (MLP). The exact interpolation of a set of N data points in a multi-dimensional space requires every one of the D dimensional input vectors $x^p = (x_i^p : i = 1,2,..D)$ to be mapped onto the corresponding target output t^p[3]. The radial basis function approach introduces a set of N basis functions, one for each data point, which takes the form $\varphi(\|x - x^p\|)$ where $\varphi(.)$ is non-linear function. The output of the mapping is then taken to be a linear combination of the basis functions, i.e.

$f(x^q) = \sum_{p=1}^{N} w_p\, \varphi(\|x^q - x^p\|) = t^q$ …(1)

If we take $t = \{t^q\}$, $w = \{w_p\}$ and $\varphi = \varphi(\|x^q - x^p\|)$, then

$w = \varphi^{-1}\, t$ where $(\varphi \text{ is non} - \text{singular})$ …(2)

Because of the similar layer-by-layer topology, it is often considered that (RBF) networks belong to (MLP) networks. It was proved that (RBF) networks can be implemented by (MLP) networks with increased input dimensions [2]. A (RBF) neural network configured for software effort estimation has only one implement of the output-input relation in the eq.(1), which is indeed a composition of the non-linear mapping realized by the hidden layer with the linear mapping realized by the output layer output neuron.[11] and [8]

If we take $\varphi(x)$ is Gaussian radial basis function which is given by the following [6]:

$\varphi_j(x) = e^{\left(-\frac{\|x-c_j\|^2}{2\sigma_j^2}\right)}$ …(3)

where , $c_j \in R^p$ and σ_j $(\sigma > 0)$ are basis center and the width of j^{th} hidden neuron respectively and $\|.\|$ denotes the Euclidean distance.

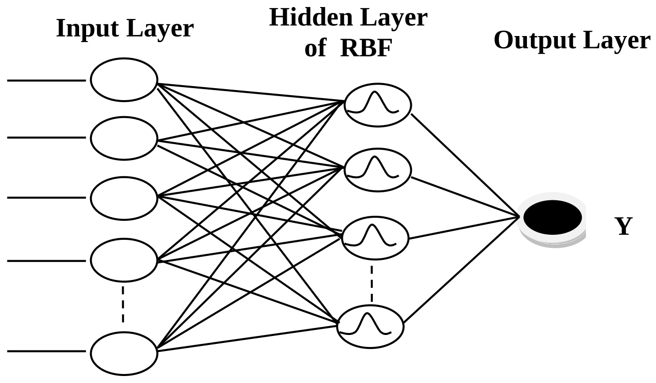

Figure. (1) : The General Shape of (RBF) Neural Network Architecture.

3.1. Determining the Weights of the (RBFNN):

From the general error we get [6][8]:

$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{C}\{y_k(x^n) - t_k^n\}^2 = \frac{1}{2}\|\varphi w^T - T\|^2$ …(4)

And deriving with respect to w

$\frac{\partial E}{\partial w} = \varphi^T(\varphi w^T - T) = \varphi^T \varphi w^T - \varphi^T T$ …(4a)

$\frac{\partial E}{\partial w} = \varphi^T \varphi w^T - \varphi^T T$ …(4b)

Setting to zero the derivative and finding w, we obtain:

$w^T = (\varphi^T \varphi)^{-1} \varphi^T T$ …(5)

(RBF) networks act as local approximation networks, because the network outputs are determined by the specified hidden units in certain local receptive fields a real-valued function whose value depends only on the distance from the origin, so that ; or alternatively on the distance from some other point c, called a center, so that any function that satisfies the property is a radial function. The norm is usually Euclidean distance, although other distance functions are also possible [3].

3.2. Training RBF Networks:

The training of a (RBF) network can be formulated as a non-linear unconstrained optimization problem given below [2] :

Let (u^k, y^k), represents the input and output training patterns, *k=1,2, ..K* , then choose w_{ij} and c_i , *i=1,2...L, j=1,2...M* , so as to minimize :

$$J(w,c) = \sum_{i=1}^{K} \|y - f(x)\|^2 \qquad \ldots(6)$$

Note that the training problem becomes quadratic once if c_i's (radial basis function centers) are known. In general, we have to pass through the following steps [8]:

1. Adjusting the widths.
2. Adjusting the centers.

4. Numerical Solution Using Finite Difference Methods :

Finite difference methods (FDM) are numerical methods for approximating the solutions to differential equations by using finite difference equations to approximate derivatives. Assuming the function whose derivatives are to be approximated is properly-behaved by Taylor's theorem [14].

The basic idea of how the finite differences method is the conversion equation partial differential to algebraic equation as they are discretization out function to a set of specific points and rounded derivatives in the equation partial differential formulas finite differences linking values function is known at this point specific. After order issue regular format as used this formula to approximate the function values at these points [10].

4.1.Finite Difference Formulas:

Below we list the commonly used finite difference formulas to approximate the first order derivative of a function u(x) using Taylor series as below [14] [5]:

$$u(x+h) = u(x) + hu'(x) + \frac{h^2}{2!}u''(x) + \frac{h^3}{3!}u'''(x) + ... \qquad \ldots(7)$$

$$u(x-h) = u(x) - hu'(x) + \frac{h^2}{2!}u''(x) - \frac{h^3}{3!}u'''(x) + ... \qquad \ldots(8)$$

Where, h is constant mesh spacing discretized.

Recall from calculus that the following approximations are valid for the first derivative of u(x). From eq.(7) and by discard the high orders of h, we get the forward difference approximation :

$$u'(x) = \frac{u(x+h) - u(x)}{h} + O(h)$$

$$\Rightarrow u'(x) = \frac{u(x+h) - u(x)}{h} \qquad \ldots(9)$$

And by the same way, we get the backward difference approximation from eq.(8) :

$$u'(x) = \frac{u(x) - u(x-h)}{h} \qquad \ldots(10)$$

By adding eq.(9) and eq.(10) we get a centered difference approximation:

$$u'(x) = \frac{u(x+h) - u(x-h)}{2h} \qquad \ldots(11)$$

Similarly by adding eq.(7) and eq.(8) with discard the high orders of h^2 we get the centered finite difference approximation for the second derivative of u(x) :

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} \qquad \ldots(12)$$

We will use these approximations to find a numerical solution to the non-linear reaction-diffusion equation.

5. Mathematical Model:

We consider the nonlinear reaction-diffusion equations with convection term of the form [1] and [4]:

$$\frac{\partial u}{\partial t} = A(u)\frac{\partial u^2}{\partial x^2} + B(u)\frac{\partial u}{\partial x} + C(u), 0 \le x < 1, 0 < t < 1 \qquad \ldots(13)$$

Where, $u(x,t)$ is an unknown function, *A(u), B(u)* and *C(u)* are arbitrary smooth functions. In eq.(13) , we generalized a great number of the well-known non-linear second-order evolution equations describing various processes in biology.

When A(u)=1, $B(u) = \lambda_1 u$, $C(u) = \lambda_2 u - \lambda_3 u^2$, where λ_1, λ_2 and $\lambda_3 \in R$, eq.(13) becomes:

$$\frac{\partial u}{\partial t} = \frac{\partial u^2}{\partial x^2} + \lambda_1 u \frac{\partial u}{\partial x} + \lambda_2 u - \lambda_3 u^2, 0 \le x < b, 0 < t < 1 \qquad \ldots(14)$$

which is called the non-linear Murray equation with initial condition:

$$u(x,0) = F(x) \text{ , } 0 \le x < b \qquad \ldots(14a)$$

and mixed boundary conditions: [4]

$$\left.\begin{aligned} u(0,t) &= G(t),\ 0 \le t < 1 \\ u_x(0,t) &= I(t)\ ,0 \le t < 1 \\ u_x(b,t) &= H(t)\ ,0 \le t < 1 \end{aligned}\right\} \qquad \ldots(14b)$$

Such that F(x) is prescribed space-dependent and G(t), I(t) are prescribed time-dependent.

6. Explicit Finite-Difference Method (EFDM):

We will apply the classical explicit finite-difference method to solve non-linear Murray equation in eq.(14). First we discretized the rectangle $R = \{(x,t) : 0 \le x < b \text{ , } 0 \le t \le 1\}$ to $(n-1)\times(m-1)$ of rectangles along each leg $(\Delta t = k)$ and $(\Delta x = h)$.

Compensate for the derivatives in eq.(14) by the finite difference approximate such that :

$$\frac{u_i^{j+1} - u_i^j}{k} = \left(\frac{u_{i-1}^j - 2u_i^j + u_{i+1}^j}{h^2}\right) + \lambda_1 u_i^j \left(\frac{u_{i+1}^j - u_{i-1}^j}{2h}\right) + \lambda_2 u_i^j - \lambda_3 (u_i^j)^2 \qquad \ldots(15)$$

Where, h and k are the space and time discretized . Now, multiply the eq.(15) by k and let $r = k / h^2$ we get:

$$u_i^{j+1} - u_i^j = r\left(u_{i-1}^j - 2u_i^j + u_{i+1}^j\right) + \frac{1}{2}\lambda_1 u_i^j r\, h\left(u_{i+1}^j - u_{i-1}^j\right) + k\lambda_2 u_i^j - k\lambda_3 (u_i^j)^2 \qquad \ldots(16)$$

Put the terms of level j+1 in the left and the terms of level j in the right we get:

$$u_i^{j+1} = u_i^j + r\, u_{i-1}^j - 2r\, u_i^j + u_{i+1}^j + \frac{1}{2}\lambda_1 u_i^j r\, h\, u_{i+1}^j - \frac{1}{2}\lambda_1 u_i^j r\, h\, u_{i-1}^j + k\lambda_2 u_i^j - k\lambda_3 (u_i^j)^2 \qquad \ldots(17)$$

By reordering the equation, we have :

$$u_i^{j+1} = (r - \frac{1}{2}\lambda_1 r\, h u_i^j\ )\ u_{i-1}^j + (1 - 2r + k\lambda_2 - k\lambda_3\ u_i^j)u_i^j + (r + \frac{1}{2}\lambda_1 r\, h u_i^j\ )\ u_{i+1}^j \qquad \ldots(18)$$

Which will be used to find the numerical solution for the points of the mesh as shown in the Figure (3) [1]:

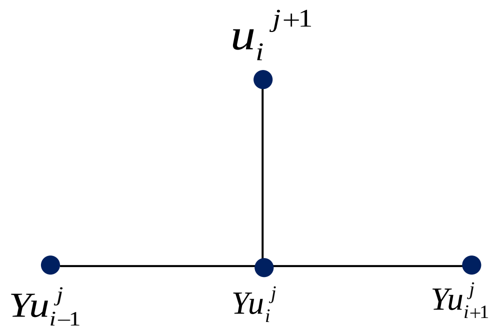

Figure (2): Computational Molecule for The Finite Difference Method for the Murray Equation.

Where, $T = r - \frac{1}{2}\lambda_1 r h u_i^j$,$Y = 1 - 2r + k\lambda_2 - k\lambda_3\ u_i^j$,$W = r + \frac{1}{2}\lambda_1 r h u_i^j$

Then, we get a set of linear equations that have numbers equal with the number of unknown in terms of which to obtain the desired numerical solution.

7. The Modified Method (RBFNN_EFDM):

The modified method is a proposed method that consists of two stages of processor are:

1- The training Stage.
2- The testing Stage.

The training stage consists of two main parts of the initial processing parts :

- Explicit Finite Difference Method (EFDM).
- Radial Basis Function Neural Network (RBFNN).

In the training stage, the output of (EFDM) becomes the input of (RBFNN) to get the optimal weight and bias. While, the testing stage requires only the optimal weight and bias to find the results for any level and any time step (k).

Where, the architecture of the proposed method (RBFNN_EFDM) in the training stage is given by Figure (4), while the proposed method (RBFNN_EFDM) in the testing stage is given by Figure (5), and also the general representation of the modified method (EFDM_RBFNN) can be taken as the following form:

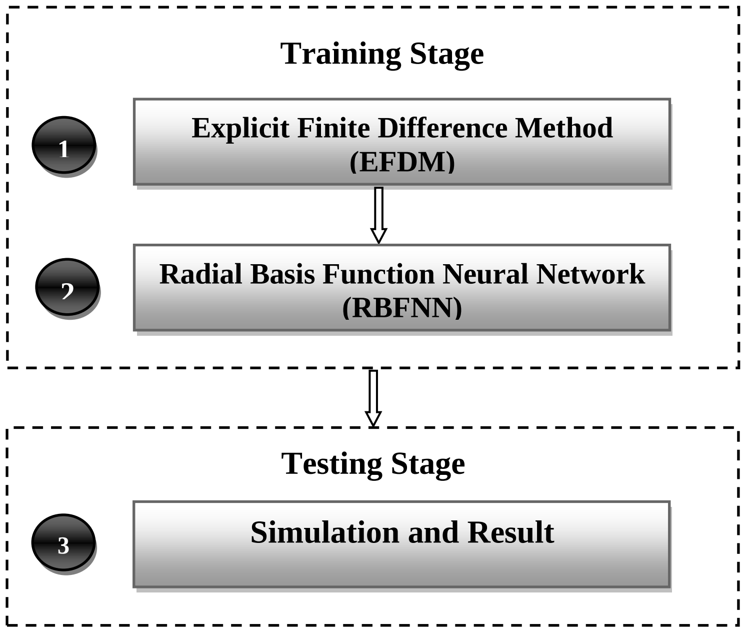
Training Stage
1
Explicit Finite Difference Method (EFDM)
2
Radial Basis Function Neural Network (RBFNN)
Testing Stage
3
Simulation and Result

Figure (3): General Representation of the Proposed Method Architecture.

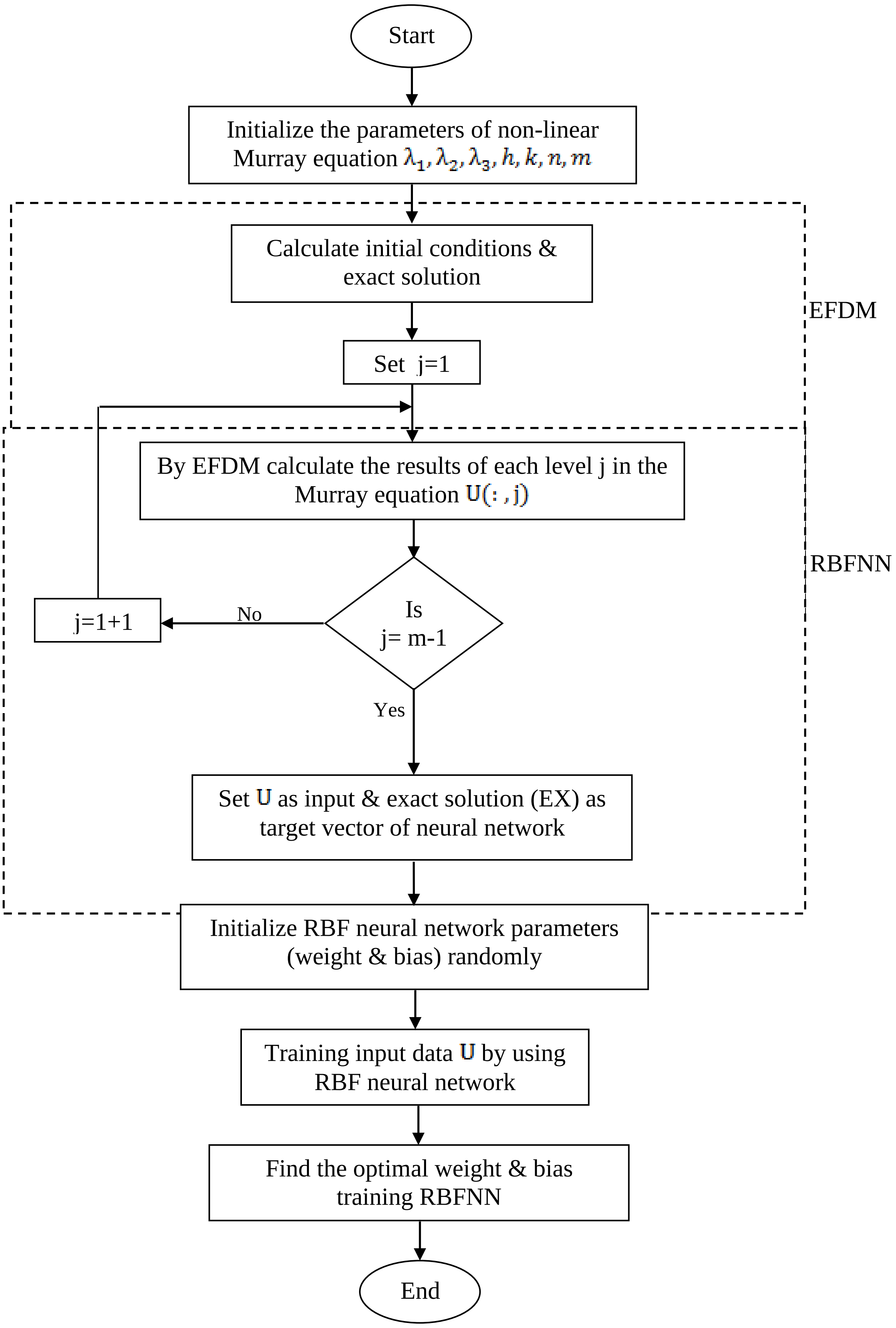
Start
Initialize the parameters of non-linear Murray equation $\lambda_1, \lambda_2, \lambda_3, h, k, n, m$
Calculate initial conditions & exact solution
Set j=1
EFDM
By EFDM calculate the results of each level j in the Murray equation U(: , j)
RBFNN
Is j= m-1
No
j=1+1
Yes
Set U as input & exact solution (EX) as target vector of neural network
Initialize RBF neural network parameters (weight & bias) randomly
Training input data U by using RBF neural network
Find the optimal weight & bias training RBFNN
End

Figure (4) : General Representation of the Proposed Method Architecture in the Training Stage.

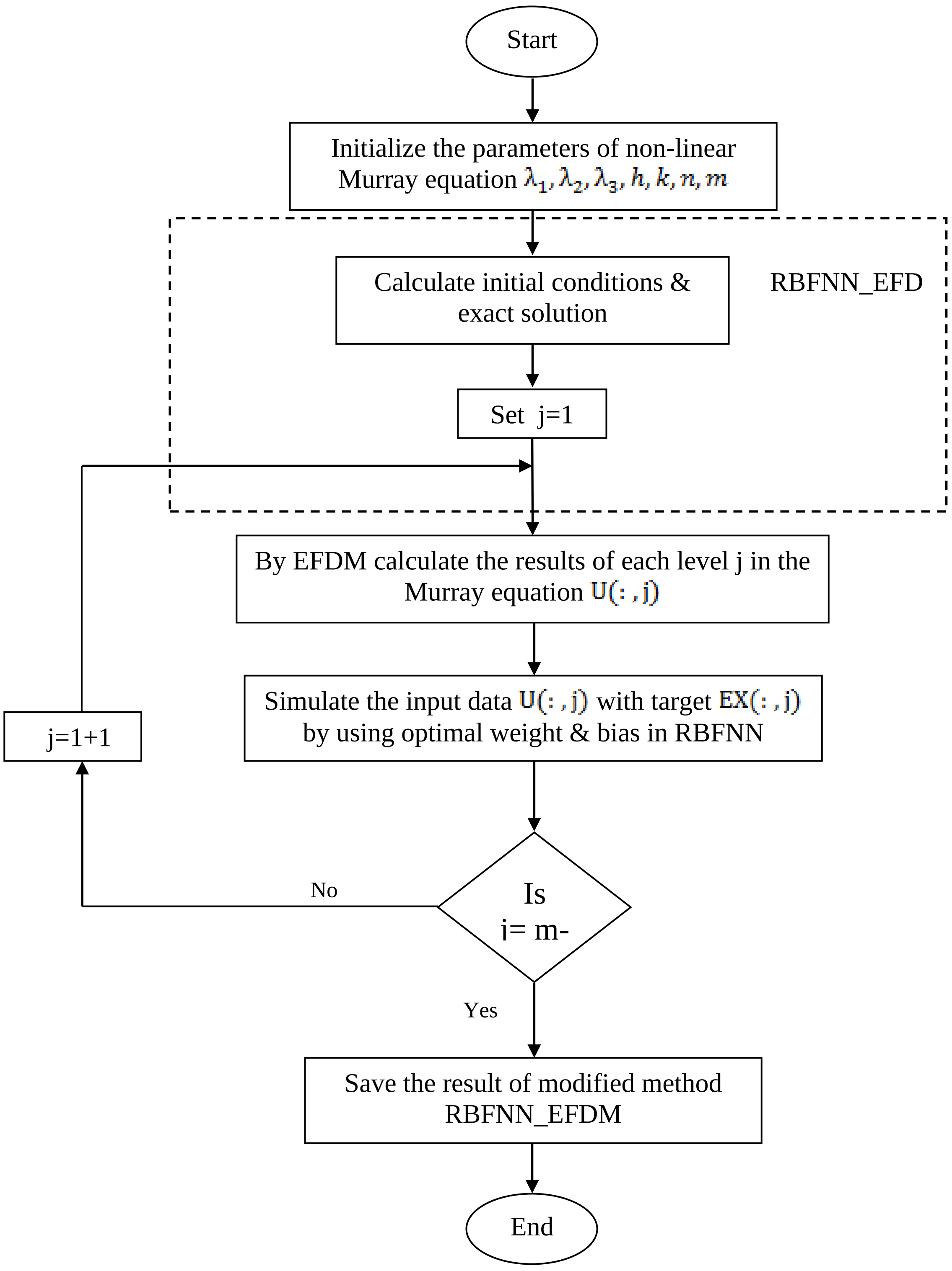
Start
Initialize the parameters of non-linear Murray equation $\lambda_1, \lambda_2, \lambda_3, h, k, n, m$
Calculate initial conditions & exact solution
RBFNN_EFD
Set j=1
By EFDM calculate the results of each level j in the Murray equation U(: , j)
Simulate the input data U(: , j) with target EX(: , j) by using optimal weight & bias in RBFNN
j=1+1
No
Is i= m-
Yes
Save the result of modified method RBFNN_EFDM
End

Figure (5): General Representation of the Proposed Method Architecture in the Testing Stage.

8. Numerical Results and Discussion:

In this section, we present the results of non-linear Murray equation by the classical explicit finite difference method and compare those with modified method by using (RBFNN). Where, we apply these methods to compute solutions numerically and compare these solutions with the exact solutions at various times.

After being trained (RBFNN) and get the optimal parameters, we enter (EFDM) outputs in each level t to the trained (RBFNN) in testing phase, where all numerical computations in training stage, we performed with the space step h=0.1 and the time step k=0.001, while the numerical computations in testing stage, we performed with the space step h=0.1 and at various times, and we take the parameters $\lambda_1 = 1, \lambda_2 = 1 \; and \; \lambda_3 = 1$ in eq(14). where :

$$u(x,0) = F(x) = \frac{\lambda_2 + c_1 e^{(\gamma x)}}{\lambda_3 + c_0} \qquad \ldots(19)$$

With the mixed boundary conditions as follows:

$$u(0,t) = G(t) = \frac{\lambda_2 + c_1 e^{(\gamma^2 t)}}{\lambda_3 + c_0 e^{(-\lambda_2 t)}} \qquad \ldots(20)$$

$$\frac{\partial u(0,t)}{\partial x} = I(t) = \frac{c_1 e^{(\gamma^2 t)}}{\lambda_3 + c_0 e^{(-\lambda_2 t)}} \qquad \ldots(21)$$

$$\frac{\partial u(b,t)}{\partial x} = H(t) = 0 \qquad \ldots(22)$$

And with the exact solution for eq(14) in the form :

$$u(x,t) = \frac{\lambda_2 + c_1 e^{(\gamma^2 t + \gamma x)}}{\lambda_3 + c_0 e^{(-\lambda_2 t)}} \qquad \ldots(23)$$

Where, $\gamma = \frac{\lambda_3}{\lambda_1}$, $\lambda_1 \neq 0$, and c_0 is constant such that $\lambda_3 + c_0 e^{(-\lambda_2 t)} \neq 0$ and c_1 is arbitrary constant [4][14].

If we take the root mean square error (RMSE) as a scale, which is defined in the following :

$$RMSE = \sqrt{MSE} = \sqrt{\frac{\sum_{i=1}^{n} (\bar{y}_i - y_i)^2}{n}}$$

Assuming that y_i represents the exact solution for ith value and $\bar{y}_i$ is the resulting value from solution, we get:

Table (1): Comparison with (RMSE) Scale between (RBFNN_EFDM) and (EFDM) for Solve the Murray Equation at t=0: 0.001: 1

The Test Results	**Modified Method (RBFNN_EFDM)**	**Classical Method (EFDM)**
RMSE	5.940007650089429e-15	0.002085439777498

Table (2): Comparison with (RMSE) Scale between the Modified (RBFNN_EFDM) and (EFDM) Method for Solving Murray Equation at t=0.1.

x	Classical Method (CFDM)	Modified Method (RBFNN_FDM)	Exact Solution
0	1.105332965990298	1.105170918075645	1.105170918075648
0.1	1.166346935537792	1.166190215042397	1.166190215042402
0.2	1.233787142736782	1.233626967491479	1.233626967491481
0.3	1.308326104347700	1.308156105107666	1.308156105107670
0.4	1.390707497572141	1.390523540550333	1.390523540550339
0.5	1.481753435156384	1.481553634798044	1.481553634798050
0.6	1.582372539234274	1.582157447630305	1.582157447630306
0.7	1.693568903061077	1.693341855820037	1.693341855820042
0.8	1.816452035328295	1.816219630294785	1.816219630294788
0.9	1.952247887324579	1.952020573122134	1.952020573122137
1	2.102311070019368	2.102103825782169	2.102103825782176
RMSE	**1.966967966730e-4**	**4.516450977248e-15**	

Table (3): Comparison with (RMSE) Scale between the Modified (RBFNN_EFDM) and (EFDM) Method for Solving Murray Equation at t=0.5.

x	Classical Method (CFDM)	Modified Method (RBFNN_FDM)	Exact Solution
0	1.650079164442094	1.648721270700132	1.648721270700129
0.1	1.757999257814407	1.756654181063259	1.756654181063257
0.2	1.877295631565723	1.875938494699855	1.875938494699852
0.3	2.009154227434270	2.007768049113636	2.007768049113631
0.4	2.154887135898868	2.153462238794615	2.153462238794611
0.5	2.315945502834315	2.314479220162632	2.314479220162627
0.6	2.493933860329751	2.492430505286891	2.492430505286887
0.7	2.690626017954850	2.689097090440411	2.689097090440406
0.8	2.907982667663615	2.906447280909331	2.906447280909324
0.9	3.148170873564575	3.146656390453782	3.146656390453775
1	3.413585637231621	3.412128512579156	3.412128512579149
RMSE	**0.001445013805584**	**5.175895635334e-15**	

Table (4): Comparison with (RMSE) Scale between the Modified Method (RBFNN_EFDM) and the Classical Method (EFDM) for Solving Murray Equation at t=1.

X	Classical Method (CFDM)	Modified Method (RBFNN_FDM)	Exact Solution
0	2.722374084330388	2.718281828459050	2.718281828459047
0.1	2.931343053538322	2.927279922064840	2.927279922064840
0.2	3.162359652465431	3.158258537051214	3.158258537051212
0.3	3.417708991838341	3.413529385031545	3.413529385031545
0.4	3.699923709244395	3.695647302451920	3.695647302451919
0.5	4.011808133495100	4.007435820252983	4.007435820252981
0.6	4.356465404357867	4.352015422716629	4.352015422716628
0.7	4.737327759724735	4.732834778321521	4.732834778321518
0.8	5.158190248642745	5.153705255176351	5.153705255176350
0.9	5.623248174182395	5.618839066472944	5.618839066472940
1	6.137138616017499	6.132891427731622	6.132891427731618
RMSE	**0.004290919146601**	**2.256488105455e-15**	

Results obtained in the Table (1) represent the root mean square error (RMSE) for the Murray equation at times t=0: 0.001 :1 and h=0.1, while the results obtained of the problem are displayed in the Tables (2), (3) and (4) for times t=0.1 , t=0.5 and t=1 respectively, with h=0.1 represents results at those times, and the modified method by using (RBFNN) at all times have (RMSE) better than the classical explicit finite difference method (EFDM). We see an excellent agreement between the modified method (RBFNN_EFDM) and exact solution in results for all tables, where the error of the solution rapidly decreases as shown in the table (1) and Figures (5) as the following:

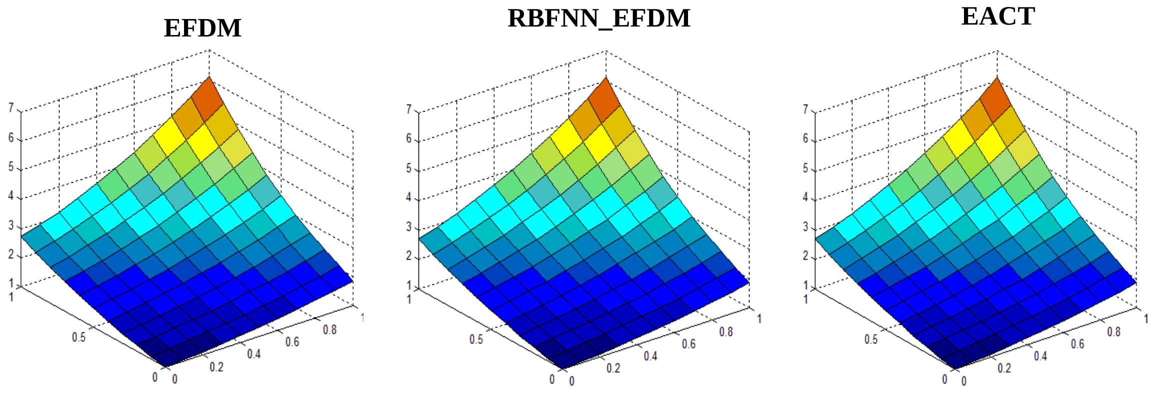

Figure (6): An Illustration the Numerical Results of non Linear Murray Equation with (t=0:0.1:1) by Using: (a) (CFDM) Method; (b) (RBFNN_FDM) Method; (c) Exact Solution.

REFERENCES:

[1] AL-Rawi E.S. and Qasem A.F.,(2010),**"Numerical Solution for Non-Linear Murray Equation Using The Operational Matrices of The Haar Wavelets Method"**, College of Computer Sciences and Mathematics , University of Mosul , Mosul , Iraq.

[2] B. M. Wilamowski, R. C. Jaeger, (1996) **"Implementation of RBF Type Networks by MLP Networks",** IEEE International Conference on Neural Networks, Washington DC,, pp. 1670-1675.

[3] Castellanos A., Blanco A.M. and Palencia V., (2007), **" Applications of Radial Basis Neural Networks for Area Forest"**, International Journal "Information Theories & Applications" Vol.14.

[4] Cherniha, R. M., (1997), **"New Ansatze and Exact Solutions for Nonlinear Reaction-Diffusion Equations Arising in Mathematical Biology",** Institute of Mathematics of the National Academy of Sciences of Ukraine, E-mail: chern@apmat.freenet.kiev.ua.

[5] Everstine G.C.,(2010),**"Numerical Solution of Partial Differential Equations",**Gaithersburg, Maryland.

[6] Goyal S. and Goyal G.K., (2012),**" Radial Basis (Exact Fit) Artificial Neural Network Technique for Estimating Shelf Life of Burfi",** Advances in Computer Science and its Applications (ISSN 2166-2924).

[7] Goyal S. and Goyal G.K., (2012),**"Radial Basis (Exact Fit) and Linear Layer (Design) ANN Models for Shelf Life Prediction of Processed Cheese "**, International Journal of u- and e- Service, Science and Technology, Vol. 5, No. 1.

[8] L. Oukhellou & P. Aknin, (2001) **" Hybrid Training of Radial Basis Function Networks in A Partitioning Context of Classification"**, Neurocomputing . Vol. 28. Nos. 1-3 pp. 165-175.

[9] Lukaszyk, S. (2004), "A New Concept of Probability Metric and its Applications in Approximation of Scattered Data Sets", Computational Mechanics, 33, 299-3004.

[10] Malek A., Beidokhti R.S,(2006)**,"Numerical Solution for High Order Differential Equations Using A Hybrid Neural Network Optimization Method"**, Applied Mathematics and Computation 183 (2006) 260–271.

[11] Parida P.K.,(2012),**"Artificial Neural Network Based Numerical Solution of Ordinary Differential Equations"** , Master of Science in Mathematics, Department of Mathematics, National Institute of Technology, Rourkela-769008 Odisha ,India.

[12] Sitouah M., (2009)," **Estimation of Reservoir Properties from Seismic Attributes and Well Log Data using Artificial Intelligence ",** A Thesis Presented to the Deanship of Graduate Studies King Fahd University of Petroleum & Minerals, Saudi Arabia.

[13] Suzuki K.,(2011)," **Artificial Neural Networks - Methodological Advances and Biomedical Applications" ,**ISBN 978-953-307-243-2, Hard cover, 362 pages, Publisher InTech.

[14] Young T.Y. and Mohlenkamp M.J.,(2012),**"Introduction to Numerical Methods and Matlab Programming for Engineers"**, Department of Mathematics ,Ohio University.

Appendix A

% (First Program) : Represent Training Stage of Murray Equation

```
% Exact Solution
clc;clear all;close all
format long
c0=1;c1=1;L1=1;L2=1;L3=1;
z=L1/L2;h=0.1 ;k=0.001 ;n=11; m=1001;
EX=zeros(n,m);x=0;t=0;
for j=1:m
    x=0;
  for i=1:n
      R1=L2+c1*exp(((z^2)*t)+(z*x));R2=L3+c0*exp(-L2*t);
      EX(i,j)=R1/R2;x=x+h;
   end
t=t+k;
end
r=k/(h^2);U=zeros(n,m);
% Intial Condition
x=0;t=0;
    for i=1:n
        j=1;
      R1=L2+c1*exp(((z^2)*t)+(z*x));
      R2=L3+c0*exp(-L2*t);
      U(i,j)=R1/R2;
  x=x+h;
    end
t=k;
for j=1:m-1
   for i=1:n
   if i==1
       x=0;
B1=((c1*z*exp((z^2)*t))/(L3+c0*exp(-L2*t)));
U(i,j+1)=((r-(0.5*L1*r*h*U(i,j)))*(U(i+1,j)-(2*h*B1)))
+((1-2*r+k*L2-k*L3*U(i,j))* U(i,j)) + ((r+(0.5*L1*r*h
*U(i,j))) *U(i+1,j));
        elseif i==n
            x=1;
 B2=(c1*z*exp(((z^2)*t)+z))/(L3+c0*exp(-L2*t));
U(i,j+1)=((r-(0.5*L1*r*h*U(i,j)))*U(i-1,j))+ ((1-2*r+k* L2-
k*L3*U(i,j))*U(i,j))+ (r+(0.5*r*L1*h*U(i,j)))*(U(i-
1,j)+2*h*B2);
   else
       M2=(U(i+1,j)-(2*U(i,j))+U(i-1,j))/(h^2);
       M1=(U(i+1,j)-U(i-1,j))/(2*h);

U(i,j+1)=U(i,j)+(k*M2)+(k*L1*U(i,j)*M1)+(L2*k*U(i,j))-
(L3*k*(U(i,j)^2));
    end
end;
```

```
t=t+k;
end
% RBF Neural Network
xx=U;yy=EX;
rand('twister',0);randn('state',0);
net=newrbe(xx,yy,0.01);
aa=sim(net,xx);
om_FDM_training=sqrt(mse(EX-U))
om_NN_train=sqrt(mse(aa-EX))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```
% (Second Program): Represent Testing Stage of Murray Equation
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
h=0.1 ;k=0.001 ;
n=11;m=1001;
EX=zeros(n,m);
x=0;t=0;
for j=1:m
    x=0;
  for i=1:n
      R1=L2+c1*exp(((z^2)*t)+(z*x));
      R2=L3+c0*exp(-L2*t);
      EX(i,j)=R1/R2;
  x=x+h;
   end
t=t+k;
end
x=[0:h:1];t=[0:k:1];
[t,x]=meshgrid(t,x);
figure;surf(x,t,EX);title('Exact')
r=k/(h^2);U=zeros(n,m);
% Intial Condition
x=0;t=0;
    for i=1:n
        j=1;
      R1=L2+c1*exp(((z^2)*t)+(z*x));
      R2=L3+c0*exp(-L2*t);
      U(i,j)=R1/R2;x=x+h;
    end
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++
%%%%% ((((((((((((((RBFNN_EFDM)))))))))))))))
%+++++++++++++++++++++++++++++++++++++++++++++++++++++++
kk=1;bb1=[];t=k;
for j=1:m-1
   for i=1:n
 % (((First Stage))) == EFD Method
   if i==1
       x=0;
    B1=((c1*z*exp((z^2)*t))/(L3+c0*exp(-L2*t)));
```

```
U(i,j+1)=((r-(0.5*L1*r*h*U(i,j)))*(U(i+1,j)-(2*h*B1)))
+((1-2*r+k*L2-k*L3*U(i,j))* U(i,j)) +((r+(0.5*L1*r*h*
U(i,j)))*U(i+1,j));
        elseif i==n
            x=1;
B2=(c1*z*exp(((z^2)*t)+z))/(L3+c0*exp(-L2*t));
U(i,j+1)=((r-(0.5*L1*r*h*U(i,j)))*U(i-1,j))+((1-2*r+k*L2-
k*L3*U(i,j))*U(i,j))+ (r+(0.5*r*L1*h*U(i,j)))*(U(i-1,j)
+2*h*B2);
   else
       M2=(U(i+1,j)-(2*U(i,j))+U(i-1,j))/(h^2);
       M1=(U(i+1,j)-U(i-1,j))/(2*h);
U(i,j+1)=U(i,j)+(k*M2)+(k*L1*U(i,j)*M1)+(L2*k*U(i,j))-
(L3*k*(U(i,j)^2));
    end
   end;
t=t+k;
%(((Second Stage)))== Testing in RBF Neural network
kk=kk+1;
input_test=U(:,kk);
om1=net.IW{1,1};
om2=net.b{1};
om3=dist(om1,input_test);
om4=netprod(om3,om2);
a{1} = radbas(om4);
om5=net.LW{2,1};
om6=net.b{2};
aa=[om5 om6] * [a{1}; ones];
bb1=[bb1 aa];
end
bb2=[U(:,1) bb1];
 %%%% Calculate Root Mean Square Error
om_FDM_testing=sqrt(mse(EX-U))
om_NN_test=sqrt(mse(bb2-EX))
x=[0:0.1:1];t=[0:0.1:1];[t,x]=meshgrid(t,x);
figure;surf(x,t,bb2(:,1:100:1001));
title('Modified Method2')
```