Multi-Knowledge Electronic Comprehensive Journal For Education And Science Publications (MECSJ)
ISSUE (27), December (2019)
ISSN: **2616-9185**

www.mecsj.com

Using Multiple Network Approach for Solving Delay First Order Delay Differential Equations

Bayan Mohammed Najmi
Applied Mathematics, Vital mathematics
bayannajmi5@gmail.com

Abstract:

This paper focuses on the use of neural networks in solving first-order delay differential equation. First-order delay differential equation solved using the simulated annealing neural network. This was done functionally by using thermal minimization. These functional links were in the form of ChSANN and LSANN. The method chosen for the solution was the form of an algorithm which was run in steps of one to ten.

To understand the error analysis two examples were presented in this study; linear delay differential equations; and non-linear delay differential equations. After solving these examples, an error analysis was carried out.

The results showed that: there were variations in the values of error for both simulated annealing methods, ChSANN and LSANN; even with the same number of training points, the LSANN gave the least error and can thus be said to be the method providing results that are very close to the accurate results. Also, the accuracy of both simulated annealing methods were inversely proportional to the value of MSE that was obtained. Moreover, the simulated annealing neural network method was preferred to solve delay differential equation due to its high accuracy and less time as opposed to the Chebyshev simulated annealing neural network**.** Finally, the simulated annealing methods were applicable in the solution of linear delay differential equations as well as in solving non-linear high order delay differential equations.

Keywords: Network, Differential Equations, Neural, Delay, Mathematica, Simulated, Annealing.

Introduction

There are various types of differential equations. These include ordinary differential equation (ODE), functional differential equation (FDE), partial differential equation (PDE), Delay Differential Equation (DDE), and Stochastic differential equation (SDE). One of the most important components of modeling present-day dynamic activities into a mathematical model is the delay differential equations (DDE) which was derived from the larger class of functional differential equations (Balachandran, Kalmár-Nagy, & Gilsinn, 2009).

Dynamic processes in present-day take into account the history of a process. In the mostly used equations of ordinary differential equations, this is not taken into account. This makes these equations be seen as one why denies reality since the issue of history for dynamic processes is of importance and will determine the present and the future results.

For this reason, one has to note that unlike the commonly taught ordinary differential equations, delay differential equations provide a better representation of a dynamic process and understanding them makes it easier for one to understand the world and dynamic processes better. With the rise of automation and the need for automatic control of dynamic processes in nature, the use of delay differential equations has been on the rise. This has been the case since the end of the Second World War, which marked the beginning of the need for automation. This was after the formation of stability's main theory of delay differential equations in 1942 by Pontryagin (JUMAA, 2017). This is the theory that has been developed and led to the present day delay differential equation understanding and application.

Since the application of delay differential equations at the organizational level is very high, especially in automation systems, one must understand ways of solving these equations as it is from that they will develop the automation control systems. It is due to this that different methods have been developed for solving the equations. One of the most commonly used ways of solving these equations is the use of delay transform method. This provides both the numerical solution, the exact solution, and the analytical solution. In addition to this method, other methods have since been used. These include the monotone iterative technique, the domain decomposition method, and the spline functions method (Yadav, Yadav, & Kumar, 2015).

All these methods require that one has to do some manual calculation to obtain solutions for this, and that presents a complex approach. As with any activity in the world, there is the development of software which can handle all these problems.

In this case, the software developed for solving delay differential equation is one where a network is used. Multiple networks can be used in solving these equations. However, the neural network, which is built with artificial intelligence insights into it is the main network used for solving delay differential equations. The use of this network would be very viable in the instances that the delay differential equation is of high order or there are many differential equations of a lower order to be solved. To understand this better, this paper will be focusing on first order delay differential equations. One should note that the first-order delay differential equation represents a linear differential equation, and therefore understanding its solution will not present a problem. Since this paper will be focusing on the use of neural networks in solving delay differential equations, the use of a linear lower-order equation would be ideal as it will make it easier for one to understand. The equation used will be of the form;

$$u'(t) = \mu u(t) + \alpha u(t - \beta)$$ (Hartung, Krisztin, Walther, & Wu, 2006)

This is a simple first-order delay differential equation and its solution with neural networks is what the paper will be presenting.

Application of neural network in solving first order delay differential equations

Though it has been seen that there multiples networks ways of solving delay differential equations, this research paper will be focusing on neural networks, which is one of the technologies that will help define the future. The neural network is an integral part of artificial intelligence and understanding how it can be used to solve delay differential equations is necessary. This is because the future and the present of control systems apply delay differential equations to a larger magnitude. Since one can see that the two, neural network and delay differential equations have a huge part to play in the future, it is just logical that their relationship is researched and understood well.

One of the methods through which a delay differential equation can be solved using neural networks is providing a link of the neural network with the optimization. This can be done functionally by using thermal minimization. These functional links will be in the form of ChSANN and LSANN (Shaikh, Jamal, Hanif, Khan, & Inayatullah, 2019). These two are revised versions for the functional link that is present artificially for NN coupled. This is evident in the optimization strategy of learning in the neural network. The designing of this is chosen because it conforms well to the linearity provided by the first-order delay differential equation chosen in this case. The design allows for the building of a good connection between the linearity that is present for single layer NN and any challenges which are present for a multilayer NN. For this reason, thought the use of the neural network will be researched using first-order delay differential equations, higher-order delay differential equations will be used. This presents the introduction of the method that will be used in the solution of first-order delay differential equations.

Method of solution

The method chosen for the solution will be in the form of an algorithm which will run in steps of one to 10. The steps are as described below. One should not the two steps are developed from the combination of both ChSANN and LSANN, which form the basis of this development.

Step 1: The first step in the solution would be through the initialization of the first-order delay differential equation. This will be done using the Legendre polynomial or the Chebyshev polynomial. This would be used for the independent variable in the differential equation. It will be for all k=0 to k=n.

Step 2: This would include the development of network adaptive coefficient for each of the polynomials. This would be an important step in the solution of the equation in the neural network platform.

Step 3: In this step, there will be the summation of the products for the network adaptive coefficient and the polynomial that was chosen in set one, either ChSANN or LSANN. This value will then be stored in the ϕ or ψ respectively, as per the polynomial is chosen.

If the one chooses ChSANN, it will be stored in the ϕ but if they choose LSANN, it will be stored in the ψ.

Step 4: This would include the activation of the value store in the ϕ and ψ. The activation will be achieved through the Taylor Series expansion of the tanh function whereby only the first three terms will be chosen (Shaikh et.al. 2019).

Step 5: Initial conditions and the activated ϕ and ψ will be used to generate a trial solution of the delay differential equation.

Step 6: This involved the calculation of the delay trail solutions. This will be done by repeating all the steps from 1 to 5. This will be done with the delay independent variable.

Step 7: This would be followed by the calculation of the mean square error (MSE) of the delay differential equation. This will be done by discretization of the domain in several points β.

Step 8: In this step focus would be on the achievement of minimized error by setting the tolerance for a minimized value of the MSE.

Step 9: This will include the minimization of the MSE. It will be done using thermal minimization methodology. For one to do this, they will use the setting from Mathematica 11.0. These will be as listed below by Shaikh et.al. (2019).

i. ➢. Level iterations→50
ii. ➢. Perturbation Scale→1.0
iii. ➢. Probability function→ $e^{\frac{-Log(i+1)\nabla MSE}{T}}$
iv. ➢. Random seed→ 0
v. ➢. Tolerance for accepting constraint violations→ 0.001

Step 10: This is the final step and will involve the checking of the MSE to ascertain that it falls within a range of criteria that has been predetermined. Once it falls within this, it is followed by the substitution of the value for the NAC for the trial solution so that one can get the output. If this is not the case, the MSE does not fall within the pre-determined criteria, one will be forced to go to step 1 and repeat the process altogether. This will be done until the MSE obtained is within the required criteria from where no one can obtain the output which will represent the solution of the delayed differential equation.

These are the general 10 steps that will be implemented in the delay differential equations to ensure that a solution is generated with the use of neural networks. The flow chart of the above steps should be as follows;

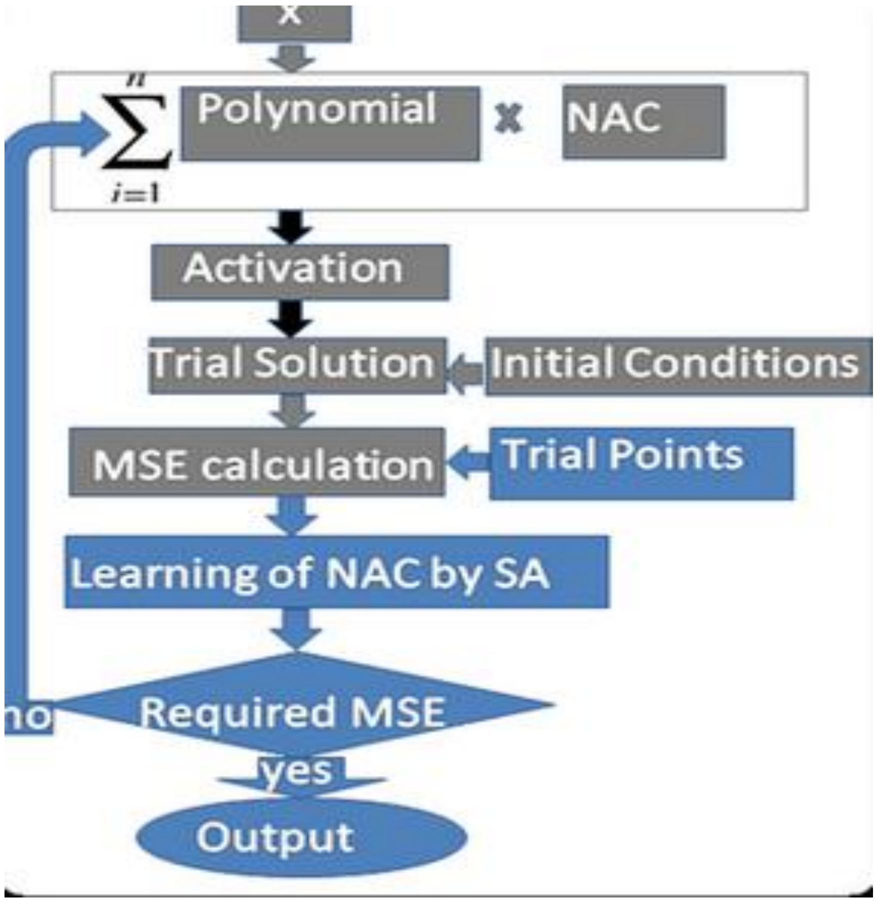


Figure 1: Flow chart of the methodology used

In the solution, one of the most important detail is the initial conditions. This comes in step five and is important in providing the history of the delay differential equation, which is a great determinant of the solution that one gives to the delayed differential equation. For this reason, there is a need for developing a standard form of initial conditions that will be used in the solution. The initial conditions taken for these case will be of the form;

$$\begin{cases} \forall i_1 = 1,\ldots,p_1 : D_{i_1}\left[t, x, \ldots, \dfrac{\partial^{\alpha_0+\alpha_1+\cdots+\alpha_n}}{\partial t^{\alpha_0}\partial x_1^{\alpha_1}\cdots\partial x_n^{\alpha_n}} y_i(t,x), \ldots\right] = 0, \quad t \in [t_0, t_{\max}], \quad x \in \Omega \\ \forall i_2 = 1,\ldots,p_2 : I_{i_2}\left[t_0, x, \ldots, \dfrac{\partial^{\alpha_0+\alpha_1+\cdots+\alpha_n}}{\partial t^{\alpha_0}\partial x_1^{\alpha_1}\ldots\partial x_n^{\alpha_n}} y_i(t_0,x), \ldots\right] = 0, \quad x \in \Omega, \quad 1 \le i \le m \\ \forall i_3 = 1,\ldots,p_3 : B_{i_3}\left[t, x, \ldots, \dfrac{\partial^{\alpha_0+\alpha_1+\cdots+\alpha_n}}{\partial t^{\alpha_0}\partial x_1^{\alpha_1}\ldots\partial x_n^{\alpha_n}} y_i(t,x), \ldots\right] = 0, \quad t \in [t_0, t_{\max}], \quad x \in \Omega \end{cases}$$

In this equation of initial values, D_{i1}, I_{i2}, and B_{i3} are all real-valued multivariable functions. These three are a representation of the non-linear time-dependent system of the equation, in the cases that the equations are not linear, are of third order and so on. However, these will not be used in the case that the equation is of first-order since it will not be non-linear but will be a linear equation. In the equations above, t represents the time variable, x represents the real-valued spatial variable, $\Omega \subseteq R^n$ is the bounded domain of the equation while $(\alpha_0, \alpha_1, \ldots, \alpha_n) \in N_0^{n+1} (N_0 = N \cup \{0\})$ is the multi-index variable of the equation. For this, a trial solution will then be prepared and will be presented as. This comes in step six of the algorithm above. It will be as presented below;

$$y_T(t, x, P) = [y_{T_1}(t, x, P_1), \ldots, y_{T_m}(t, x, P_m)]$$

This includes layered feed-forward neural networks. In addition to this, it also had adjustable parameters in the form of weights and biases which are useful in obtaining the solution. Since the parameters are very important in obtaining the solution, one should try to obtain their proper values. In obtaining this, the problem is first transformed into an unconstrained optimization problem.

This happens for any order of the differential function, meaning that it captures both linear and non-linear delay differential equations. The trial solution in this will then be obtained as;

$$y_{T_i}(t, x, P) = \gamma_i[t, x, N_i(t, x, P)]$$

The two, both trial solution and the initial conditions are a very important part of the algorithm above and helps in the determination of the MSE, which also helps in determining the solution. The setting of a predetermined value for the MSE helps in the evaluation of the solution to the equation since it determines its error and, as such, accuracy. This shows why the two have been explained in detail in this case.

Employment or inputting the delay differential equation into the neural network

The application of the two, ChSANN and LSANN will be conducted for any delay differential equation. This can be an equation of the following form;

$$f^{(n)}(x) = F(f(x - \tau x), f'(x), f''(x), f'''(x), \ldots, f^{(n-1)}(x)) + g(x).$$

In this case, n will be taken as one as the equation will be a first-order delay differential equation. However, one has to note that we are providing a general case where any order equation can be solved in the neural network, though the focus will be directed to how the first-order delay differential equation will be solved. The initial conditions of the above case will be $f(0) = \alpha_{0,}\, f'(0) = \alpha_{1,}\ \ldots f^{(n-1)}(0) = \alpha_{n-1}.$ represented as;

For the first order, the initial conditions will be represented as;

$$f(0) = \alpha_{0,}$$

In the implementation of the two, ChSANN and LSANN, the above equation for any delay differential equation will be rearranged. This rearrangement will focus on making the right-hand side of equation zero by taking everything on that side to the left-hand side. It will be as shown below;

$$f^{(n)}(x) - F(f(x-\tau x), f'(x), f''(x), f'''(x), \dots, f^{(n-1)}(x)) - g(x) = 0.$$

As said above, for first-order delay differential equation, n will be replaced with 1, and higher-order terms will not be present.

To obtain the trial solution and the delayed trial solution, one will be required to consider the following equation that will be very vital if one is to get this right.

$$\phi = \sum_{k=1}^{m} w_k T_{k-1}(x)$$

In this equation, it is important to note that the term T_k represents the Chebyshev polynomial whose recursive formula is defined as;

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}, \quad k \geq 2$$ (Shaikh et.al. 2019)

In this case, one has to note that when k=0, the value of T_k will be 1 while when k=1, this value will be x. These two are described as the fundamental values for the Chebyshev polynomial. All this described when one chooses the Chebyshev polynomial in their solution.

Now let's move to what happens when one chooses the Legendre polynomial in the place of the Chebyshev polynomial described above. The representation of the Legendre polynomial will be as in the equation below;

$$L_{j+1}(x) = \frac{1}{(j+1)}(2j+1)\, x\, L_j(x) - \frac{1}{(j+1)} j\, L_{j-1}(x),\ j \geq 2,$$

As with the Chebyshev equation, the fundamental values for this will be $L_0(x) = 1$ and $L_1(x) = x$. This will all happen when j=0 and 1.

As with the Chebyshev, one will be required to use the following equation to obtain the trial solution and the delay trial solution.

$$\psi = \sum_{j=1}^{n} w_k \, L_{j-1}(x)$$

From this, one is required to use the first three terms of the tanh series for the activation of both the ϕ and ψ as they are represented in the two equations above. The activated equation which represents the trial equation will be as represented below;

$$f_{trial}(x, w) = \alpha_0 + \alpha_1 x + \frac{x^2}{2!}\alpha_2 + \ldots + N(x, w)\frac{x^k}{k!}.$$

This forms the trial equation for the method or algorithm described above. In this, N is a representation of the activated ϕ and ψ. The above equation represents the trial solution. If one wants to obtain the delayed trial solution, they will then be required to replace the variable x with x−τx (Wang, Yang, & Hu, 2015).

From the trial solution and the delayed trial solution as represented in the algorithm, the next step would be to obtain the mean square error (MSE). This is represented as the difference between the trial solution and the initial solution. Its equation is as represented below.

$$E_r(w) = \sum_{l=1}^{\beta} \left(f_{trial}{}^{(n)}(x_l, w) - F\begin{pmatrix} f_{trial}(x_l, w), f_{trial}(x_l - \tau x_l, w), f'_{trial}(x_l, w), f''_{trial}(x_l, w) \\ , \ldots, f^{(n-1)}{}_{trial}(x_l, w) \end{pmatrix} - g(x_l)\right)^2$$

In this case, β will be a representation of the number of trials to be carried out for the predetermined MSE to be obtained. The above equation can be said to be the fitness function for the learning of the NAC.

Understanding required MSE through error analysis

To understand the required MSE, the neural system should be capable of carrying out error analysis. One point that one has to note is that the network approach used in this case is dependent on simulated annealing, whereby both Legendre and Chebyshev simulated annealing of neural networks will be used, giving rise to both ChSANN and LSANN methods described above.

If one substitutes the values of network adaptive coefficients (NAC) after the learning from the simulated annealing into the trial solution above, they will obtain solutions for the ChSANN or the LSANN depending on the method they chose to use (Shaikh et.al. 2019). This will then be substituted in the equation below for one to obtain an analysis for the accuracy of the equation in the domain of [0, 1]. The equation for this is shown;

$$E(x) = |f^{n}(x) - F(f(x-\tau x), f'(x), f''(x), f'''(x), \ldots, f^{(n-1)}(x)) - g(x)| \cong 0$$

The value of the MSE should be predetermined such that the $E_r(x_i)$ value will tend to be zero once the predetermined value of MSE is obtained. This convergent is dependent on the learning methodology used, and in this case, the methodology used is simulated annealing (SA). The use of simulated annealing methodology for NN architecture in the neural network will be the only determinant of the accuracy of the solution obtained and the convergent which will determine if there is a solution to the delayed differential equation or not.

To understand the error analysis better, some examples can will be presented in this case. There will be two examples, one of linear delay differential equations and the other of non-linear delay differential equations. One these examples are solved, an error analysis will be carried out and from it, a conclusion will be drawn from where now the best simulation annealing method for solving delay differential equations using neural networks will be determined and presented.

Example 1: Linear delay differential equation.

This will be done using a second-order delay differential equation since it is linear and solved in the same way that a first-order delay differential equation will be solved.

Consider the following equation;

$$y^{(\alpha)}(x) = \frac{3}{4}y(x) + y\left(\frac{x}{2}\right) - x^2 + 2,\ y(0) = 0; y'(0) = 0; \alpha = 2.$$

This is a second-order equation since the value of α is 2 as stated in the conditions. Now let's start by getting the exact solution of the equation when α = 1.
The exact solution for this will be as shown below:

$$y(x) = x^2$$

To solve the second-order linear delay differential equation, the two simulation annealing methodologies have to be used. This will be done on the domain [0, 1]. With the two methods employed, one can then understand which one provided a better solution. In obtaining the solution, the division will be carried out by dividing the training distance 10 times. This will also be accompanied by 6 NAC. Upon carrying out the all this division, and at α = 2, the value of MSE was found to be 1.899×10−11 for ChSANN and 1.323×10−14 for the case of LSANN (Shaikh et.al. 2019). In this case, the trial solution was obtained as follows.

$$y_{trial} = \frac{x^2}{2}N$$

In this case, N described the structural output of the neural network used.

$$y_{dtrial} = \frac{(x/2)^2}{2}M.$$

Similarly, M, in this case, is a description of the structural output of the neural network used in the solution.

The final value of the NAC after training using simulated annealing method is shown in the table below.

Number	**NAC**	**ChSANN value**	**LSANN value**

1	W1	1.70683523	1.710066834
2	W2	-0.00101189	-0.00004054
3	W3	-0.00220052	-0.00372515
4	W4	-0.00028488	-0.00002618
5	W5	0.000058430	-0.00003209
6	W6	-0.00001175	-0.00002803

(Shaikh et.al. 2019)

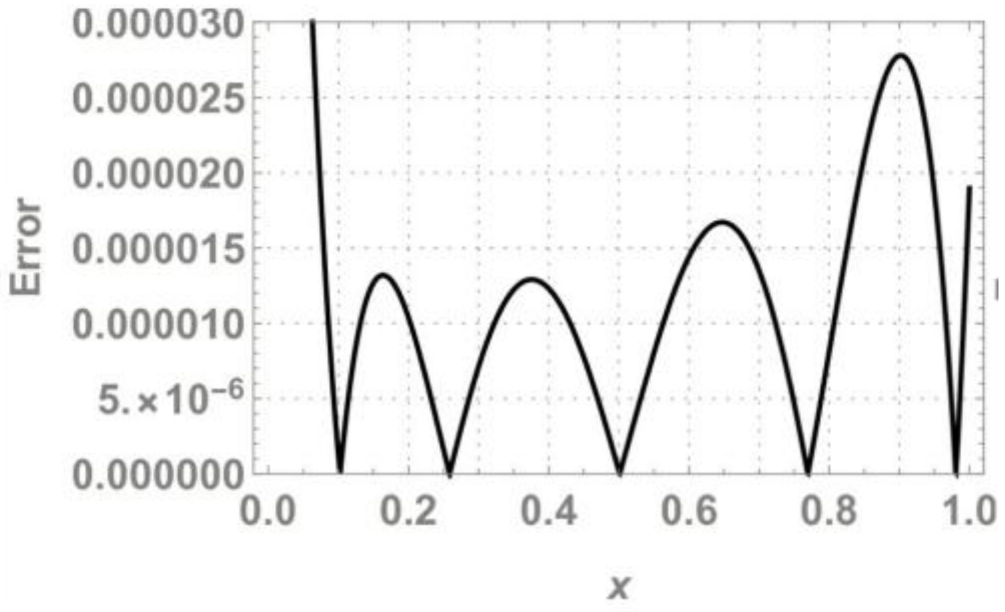


Figure 2: ChSANN error analysis graphical solution for the above linear DDE

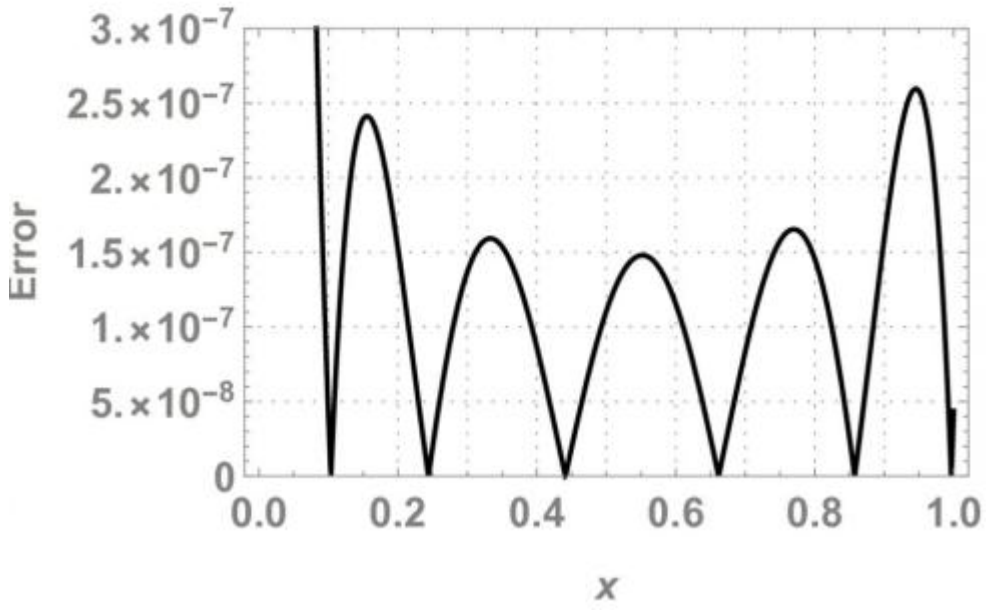


Figure 3: LSANN error analysis graphical solutions

Example 2: Non-linear delay differential equation

The equation used for this is a third-order delay differential equation. This is the simplest non-linear delay differential equation and understanding its solution would be helpful for anyone interested in solving these types of equations using the two simulation annealing methods. Consider the following third-order nonlinear delay differential equation.

$$y^{(\alpha)}(x) = 2y^2\left(\frac{x}{2}\right) - 1,\ y(0) = 0,\ y'(0) = 1, y''(0) = 0,\ \alpha = 3$$

It is a third-order delay differential equation since the value of α is 3.

This would require one to find the exact solution when α=3. The solution to these can be found to be as shown below.

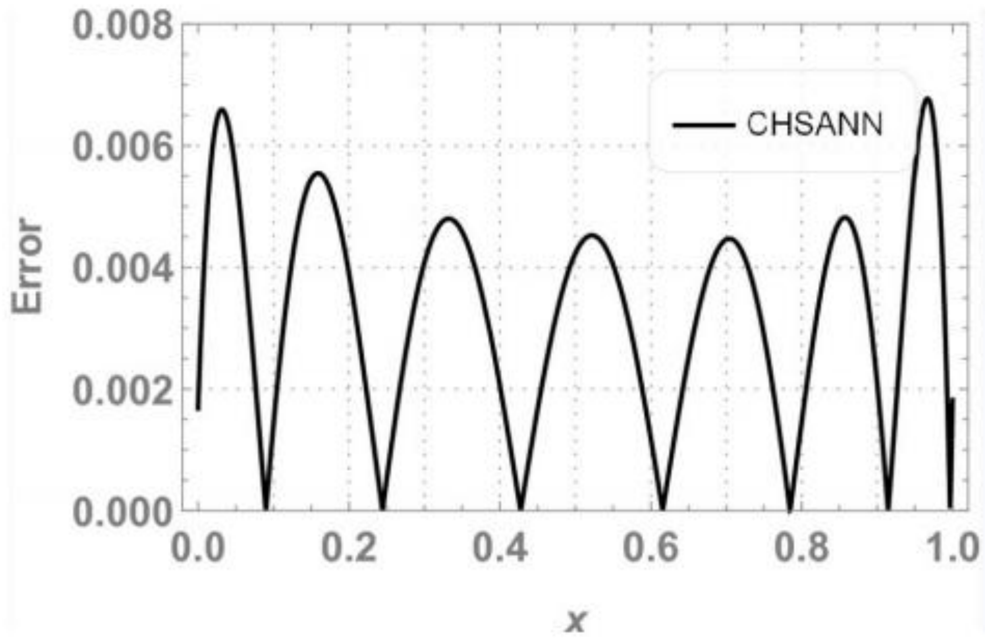
$$y(x) = Sin(x)$$

As we did with the linear delay differential equation above, the solution of this equation will be obtained by using the continuous domain of [0, 1]. The same procedure of running both simulation annealing neural network for 10 NAC for the Chebyshev and 6 NAC for the Legendre. The training method used in this case was increased from 10 to 20. With all these conditions, it was found that the mean square error for the ChSANN was $2.57\times10-5$ and LSANN was found to be $5.48\times10-7$. The final values of the weights of each method after training with the algorithm of simulated annealing areas represented in the following table.

Number	**NAC**	**ChSANN value**	**LSANN value**
1	W1	-0.874541110	-1.2104577306
2	W2	2 *w*2 -0.478671752	-0.0384815109
3	W3	0.0165454550	0.06652279388
4	W4	0.4425599703	-0.0172612429
5	W5	- 0.594007418	0.00543949906
6	W6	0.4767150261	-0.00064910894
7	W7	-0.261651890	0.9618236111
8	W8	0.0976657350	-1.0110550149
9	W9	-0.022615861	2.3259303584
10	W10	0.0024649552	-2.0381716440

(Shaikh et.al. 2019)

The graphical



representation of the error analysis for the two methods is as shown

Figure 4: CHSANN nonlinear DDE error analysis

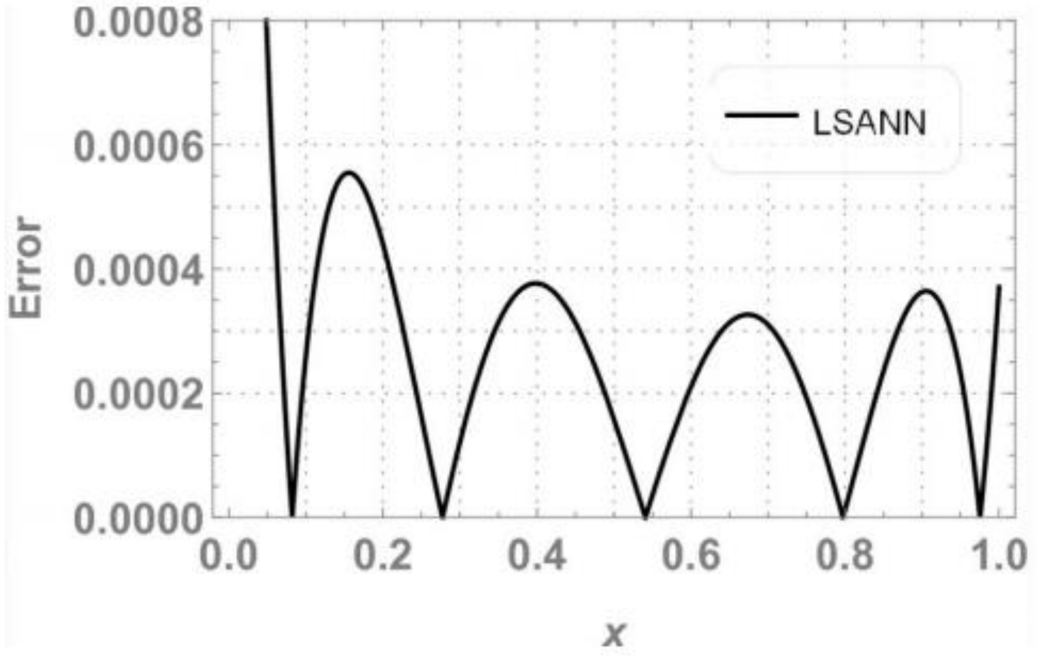


Figure 5: LSANN nonlinear DDE error analysis

Analysis from the examples

From the two examples above, one can see that there are variations in the value of error for both simulated annealing methods, ChSANN and LSANN. Even with the same number of training points, the LSANN gives the least error and can thus be said to be the method providing results that are very close to the accurate results. Also one has to note that the accuracy of both simulated annealing method is inversely proportional to the value of MSE that is obtained. This helps in understanding the results which were obtained in solving the two examples using the simulated annealing neural network for the solution.

In addition to this, it is also important for one to compare this will other methods of solving delay differential equations. One thing that one has to learn in this is that the time elapsed in using the two methods, both ChSANN and LSANN. This is one point that the simulated annealing method beats all other methods of solving delay differential equations through the use of neural networks.

Other methods of representation of the solution in graphical form are mainly in collocation points (Şaylı, & Yılmaz, 2016). This is a problem that is solved by using simulated annealing neural network methods since they will provide a continuous solution. This represents one superior of this method.

In other methods, the solution of the equation is presented is very large and difficult to understand. This happens in the case that the delay differential equation is non-linear and some complexities present in its operation. For the method presented in this case, one can see that it handles non-linear complex and high order delay differential equations in simple terms. It is for this reason that the time it takes to complete computation is less.

From this method, and using the examples that have been provided in this case, one can see that first-order delay differential equations can be solved using the network approach method whereby the network chosen is a neural network. The same can be applied to a system of first-order delay equations. Using numerical methods would have provided a solution, however, this would have taken much time to be completed. The fact that most dynamic processes in present-day are not described by a single delay differential equation but a system of this. Since it is presented in a system the provision of a method that can view these equations at once and solve them with minimum time would be very useful. The approach to this would be using the network approach method. Since there are various ways of using the network approach method, the one that was chosen for the study is the neural network one, and especially the simulated annealing neural network. With this method, time taken obtaining the solution to the equation will be minimized and, as such, a system of similar equations will be solved. This explains the reason as to why the chosen methodology is viable and the best in the solution of not only first-order delay differential equation, but any order delay differential equation. One should not that this method will be implemented by the use of mathematical software. One of the best software to be used is MATLAB or Mathematica (Kumar, & Yadav, 2011). One can choose either of the two and with this knowledge, they will end up understanding the solution of delay differential equations better using neural network simulated annealing.

Conclusions

The network approach is one of the best ways of solving delay differential equations. Though various networks can be used, one network which stands out is the neural network, which is an integral component of Artificial Intelligence. Since the future is expected to be built upon artificial intelligence, this method should be extensively used and understood. Simulated annealing neural network method is very useful and beats other methods of solving delay differential equations by saving time thus being able to solve a system of first-order delay differential equations. Though the simulated annealing methods are two in number, based on the polynomial chosen, Legendre simulated annealing neural network method is preferred due to its high accuracy and less time as opposed to the Chebyshev simulated annealing neural network. This explains why the LSANN method is preferred and should be used for the solution of these equations. Finally, it is important to note that the simulated annealing methods are not only applicable in the solution of linear delay differential equations but can also be used in solving non-linear high order delay differential equations. This makes the two methods of ideal methods for solving any system of delay differential equations.

References

Balachandran, B., Kalmár-Nagy, T., & Gilsinn, D. E. (2009). *Delay differential equations*. Berlin: Springer.

Hartung, F., Krisztin, T., Walther, H. O., & Wu, J. (2006). Functional differential equations with state-dependent delays: theory and applications. In *Handbook of differential equations: ordinary differential equations* (Vol. 3, pp. 435-545). North-Holland.

JUMAA, S. I. (2017). *Solving linear first-order delay differential equations by MOC and steps method comparing with MATLAB solver* (Doctoral dissertation, Dissertação (Mestrado) Near East University-Turkish).

Kumar, M., & Yadav, N. (2011). Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey. *Computers & Mathematics with Applications*, *62*(10), 3796-3811.

Şaylı, M., & Yılmaz, E. (2016). State-dependent impulsive Cohen–Grossberg neural networks with time-varying delays. *Neurocomputing*, *171*, 1375-1386.

Shaikh, A., Jamal, M. A., Hanif, F., Khan, M. S. A., & Inayatullah, S. (2019). Neural minimization methods (NMM) for solving variable-order fractional delay differential equations (FDDEs) with simulated annealing (SA). *PloS one*, *14*(10).

Wang, F., Yang, Y., & Hu, M. (2015). Asymptotic stability of delayed fractional-order neural networks with impulsive effects. *Neurocomputing*, *154*, 239-244.

Yadav, N., Yadav, A., & Kumar, M. (2015). Neural Network Methods for Solving Differential Equations. *An Introduction to Neural Network Methods for Differential Equations* (pp. 43-100). Springer, Dordrecht.