International Journal of
Science and Engineering Investigations vol. 7, issue 83, December 2018
ISSN: 2251-8843

Numerical Analysis Without Mesh Based on Artificial Neural Networks: A Comparison Between the Gradient Descent Method and Levenberg-Marquardt

Alcineide Pessoa[1], Davi Nascimento[2], Marta Barreiros[3], Thaís Alves[4], Gean Sousa[5]
[1,2,3,4]University of Maranhão
[5]Professor at the Federal University of Maranhão
([1]alcineidedutra@hotmail.com, [2]davi.nascimento777@gmail.com, [3]marta-barreiros@hotmail.com,
[4]thaishortiz3@gmail.com, [5]gean_mat@yahoo.com.br)

Abstract- Differential Equations are important for modeling physical problems. In some cases, these equations are solved by some kind of numerical method. The most known numerical methods are dependent on an early choice of a computational mesh. The definition of a mesh can be a constraint when the domain of the problem under analysis. The use of Artificial Neural Networks is an alternative to solve problems without depending on the mesh. However, it is important to deepen the analysis of this mechanism. In order to verify if there is superiority between two methods of optimization applied to the solution of an Ordinary Differential Equation (ODE), in this article we compare the Downward Gradient and Levenberg-Marquardt methods. We have verified that the Levenberg-Marquardt method is, both in terms of approach effectiveness and in terms of convergence time, superior to the Gradient Descent method.

Keywords- *Differential Equations, Neural Networks, Gradient Descent, Levenberg-Marquardt*

I. INTRODUCTION

Optimization problems pose several challenges to researchers in the areas of exact sciences [1-4]. The development of optimal algorithms or mathematical models capable of optimizing the solution of problems in engineering is, above all, the object of study in several researches [5-7]. Among the classical optimization algorithms is the Descending Gradient method [8]. This method has been applied in several optimization problems, for example in the solution of ODE with a mesh based domain [9] or even without mesh [10].

The Gradient Descent method, when used by an Artificial Neural Network (ANN), to solve an ODE was very effective if the domain was discretized in a pre-established mesh, but presented small errors when the domain was chosen randomly [10].]. Depending on the problem modeled, these errors can configure a relevant factor and thus represent an important approximation problem. Considering the above, we realized the need to improve these results.

In order to improve the results obtained by [10] it is that in this article we implemented the Levenberg-Marquardt algorithm in an ANN, in order to solve an ODE and obtain better results than those obtained by the Gradient Descent. The Levenberg-Marquardt algorithm was first published in 1944 by Kenneth Levenberg [11]. The main difference between the Gradient Descent and Levenberg-Marquardt lies in the fact that the first is based on the gradient vector and the second on the Jacobian matrix [12,13].

For the purpose of comparison, in this article we perform the ODE solution presented by [10], using the optimization methods mentioned above. We compute the errors generated by each method and present them in tabular form. The results presented show an important improvement when Levenberg-Marquardt is used, because the approximation error is considerably smaller.

Another problem encountered when using Gradient Descent is that the convergence of this method can be relatively slow, making the time to solve the problem an important issue [14,15]. In this context, we also computed the number of iterations each method needed to converge, and we also found a superiority of the Levenberg-Marquardt algorithm.

II. METHODS

The methodology to solve ODE using RNA is based on the use of two optimization algorithms applied to the minimization (or maximization) of a cost function. The topology of the RNA used, consists of two layers with five neurons in each layer. This cost function must be chosen in such a way that its minimization (or maximization) is equivalent to solving ODE.

Considering that our methodology will be applied to a second order ODE, and that this type of equation has the following formulation,

$$\frac{d^2\psi(x)}{dx^2} = \left(f(x, \psi, \frac{d\psi}{dx}\right), \quad (1)$$

then the approximate solution can be written in the form:

$$\psi_t(x) = A + A'x + x^2 N(x, p) \quad (2)$$

with the initial values given by $\psi(0) = A\ e\ \frac{d\psi(0)}{dx} = A1$. Consequently, the cost function to be minimized, either by the Downward Gradient method or by Levenberg-Marquardt, is $E[\vec{p}] = \sum_i \left\{ \frac{d^2\psi_t(X_i)}{dx^2} - f(x_i, \psi_t, \frac{d\psi_t(x_i)}{dx} \right\}^2$ where xi are the points of the problem domain. The points will be chosen randomly in order to test the effectiveness of the algorithms with respect to domain choice.

A. *Update of the Synaptic Weights Using the Gradient Descent*

Considering that the RNA model we are going to use is composed of two layers, an input layer and a hidden layer, and what *w* are the weights of the input layer and the weights of the hidden layer, then by the optimization method known as Gradient Descent [16]

$$w(n+1) = w(n) - \frac{\partial E[\vec{p}]}{\partial w_{ki}} \tag{3}$$

and

$$v(n+1) = v(n) - \mu \frac{\partial E[\vec{p}]}{\partial v_k}. \tag{4}$$

If we start the values of w and v in a random manner then by the given conditions the method will converge to the minimum of the cost function presented [15].

Solving the derivates $\frac{\partial E[\vec{p}]}{\partial w_{ki}}$ and $\frac{\partial E[\vec{p}]}{\partial v_k}$ we obtain:

$$\frac{\partial E[\vec{p}]}{\partial v_k} = 2y \left\{ 2\ \frac{\partial N(x_1,\vec{p})}{\partial v_k} + 4x_i \frac{\partial \frac{dN(x_i,\vec{p})}{dx_i}}{\partial v_k} + {x_i}^2 \frac{\partial \frac{d^2N(x_i,\vec{p})}{dx_i^2}}{\partial v_k} - \frac{\partial f(x_i, \psi_t(x_i), \frac{d\psi_t(x_i)}{dx})}{\partial v_k} \right\} \tag{5}$$

At where $y = 2N\ (x_1, \vec{p}) + 4x_1\ \frac{dN(x_1,\vec{p})}{dx_i} + x_1^2 \frac{d^2N(x_1,\vec{p})}{dx_i^2} - f(x_1, \psi_t(x_1), \frac{d\psi_t(x_i)}{dx})$

The gradient of the cost function relative to the weights of the hidden layer is:

$$\frac{\partial E[\vec{p}]}{\partial w_{ki}} = 2y\{2v_k\sigma_k' x_i + 4x_i[v_k\sigma_k' + v_k w_{ki}\sigma_k'' x_i] + x_i^2\ (2v_k w_{ki}\sigma_k'' + v_k\ w_{kj}^2\sigma_k''' x_i) - \frac{\partial f(x_i, \psi_t(x_i), \frac{d\psi_t(x_i)}{dx}}{\partial \psi_t(x_i)}, x_i^2 v_k \sigma_k' x_i\} \tag{6}$$

More details on the calculation of the presented gradients can be found in [9].

B. *Update of the Synaptic Weights Using Levenberg-Marquardt*

The Levenberg-Marquardt optimization method requires the calculation of the Jacobian matrix of the error vector and $e(i) = \mathrm{E}[\vec{p}]$..

Given that error signal $e(i)$ is a function of the adjustable weight vector w (or v), and given yet an operating point $w(n)$, it is possible to linearize the dependence of $e(i)$ with respect to w by writing.

$$e'(i,w) = e(i) + \left[\frac{\partial e(i)}{\partial w}\right]_{w=w^k}^T (w - w^k), i = 1,2,\dots,k \tag{7}$$

Mathematically,

$$e'(k,w) = e(k) + j(k)(w - w^k) \tag{8}$$

where $e(k)$ is the vector

$$e(t) = [e(1), e(2), \dots, e(k)]^T \tag{9}$$

and $J(k)$ is the Jacobian k-by-m matrix of $e(k)$:

$$J(k) = \begin{bmatrix} \frac{\partial e(1)}{\partial w_1} & \dots & \frac{\partial e(1)}{\partial w_m} \\ \vdots & & \vdots \\ \frac{\partial e(k)}{\partial w_1} & \dots & \frac{\partial e(k)}{\partial w_m} \end{bmatrix}_{w=w^k} \tag{10}$$

Without loss of generality, $w(k+1)$ is defined by

$$w(k+1) = \arg\ min \left\{ \frac{1}{2}\ \|e'(n,w)\|^2 \right\} \tag{11}$$

Calculating the quadratic euclidean norm of $e'(k,w)$ gives

$$\frac{1}{2}\ \|e'(k,w)\|^2 = \frac{1}{2}\ \|e(k)\|^2 + e^T(k)J(k)\Delta w + \frac{1}{2}\Delta w^T J^T(k)J(k)\Delta w \tag{12}$$

Where $\Delta w = (w - w(k))$.

Differentiating with respect to w and equalizing the result to zero obtains

$$J^T(k)e(k) + J^T\ (k)J(k)\Delta w = 0 \tag{13}$$

Solving the equation for w, one can then write,

$$w(k+1) = w(k) - \left(J^T(k)J(k)\right)^{-1} J^T\ (k)e(k) \tag{14}$$

The equation appeared with the same pure form of the Gauss-Newton method, but for the interaction of this method to be computable, the product matrix $J^T(k)J(k)$ should be non-singular. So to ensure this non-singularity of the product matrix $J^T(k)J(k)$ we have as current practice the addition of the diagonal matrix μI . Where I is the identity matrix and μ is a parameter that adjusts the convergence rate of the algorithm.

Then the method is implemented in a slightly modified way known as the Levemberg-Marquadt method:

$$w(k+1) = w(k) - (J^T(k)J(k) + \mu I)^{-1} J^T\ (k)e(k) \tag{15}$$

In our methodology, this same procedure is applied to update v. This method has convergence in fewer iterations, but requires more calculations by interaction due to the calculation of inverse matrices. More details can be found in [9].

III. Result

The neural model presented previously combined with the described optimization methods and respective synaptic weights update algorithms was applied to a second order problem, in order to verify the behavior of the solutions when choosing a domain with random points. The problem chosen to support the simulation is given by [16]:

$$\frac{d^2y}{dt^2} = -b\frac{dy}{dt} - cy + w(t), \tag{16}$$

Where $b = \frac{6}{3t} + 2$, $c = \frac{1}{t} + 1$ and $w(t) = t^2 - 2$. The choice of these parameters was made in a random manner.

The initial conditions $y(0) = 0$ and $dy\frac{(0)}{dt} = 0$, were satisfied. The parameters used in the neural network were as follows:

a) An input layer and a hidden layer with weights w and v;

b) Number of neurons in the hidden layer equal to the number of neurons in the input layer equal to 5;

c) Sigmoid activation function;

d) Learning rate 0.5

To make it possible to compare the results, in all simulations the synaptic weights were started with the same values:

$w = [0.157 \quad 0.970 \quad 0.957 \quad 0.485 \quad 0.800];$

And

$v = [0.142 \quad 0.421 \quad 0.915 \quad 0.792 \quad 0.959].$

Figures 1 and 2 are comparisons between the analytical solution and that obtained by the neural network when the domain was determined by choosing 10 random points in the interval [0,1]. In figure one was used the method of the Gradient Descent and in figure 2 Levemberg Marquardt.

Figures 3 and 4 are comparisons between the analytical solution and that obtained by the neural network when the domain was determined by choosing 20 random points in the interval [0,1]. Figure 3 shows the downward gradient method and figure 4 Levemberg Marquardt.

Figures 5 and 6 are comparisons between the analytical solution and that obtained by the neural network when the domain was determined by choosing 30 random points in the interval [0,1]. In Figure 5, the Gradient Descent method was used and in Figure 6 Levemberg Marquardt.

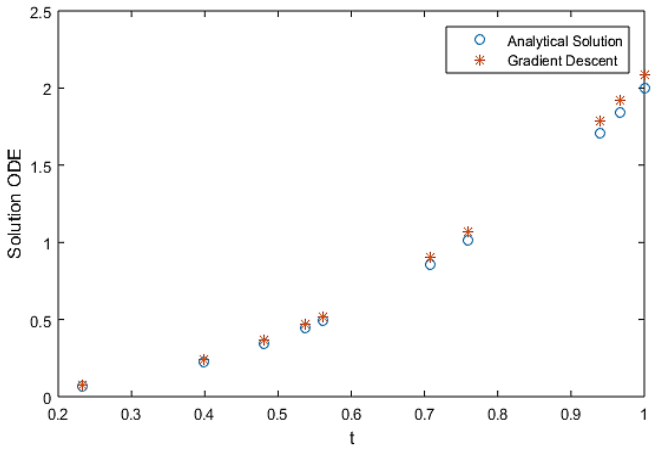

Figure 1. Comparison between exact solution and ANN (Gradient Descent) for a 10-point domain

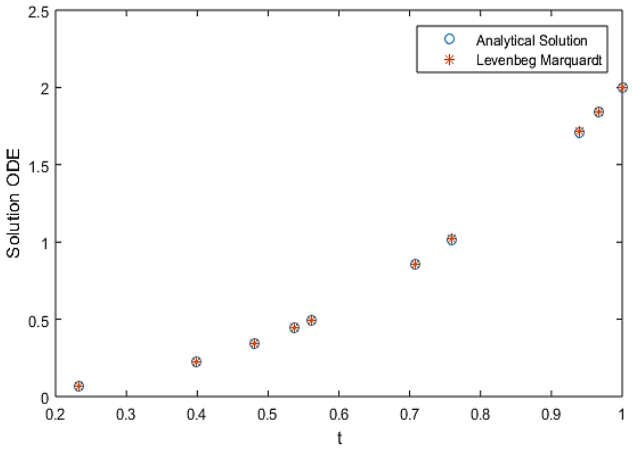

Figure 2. Comparison between exact solution and ANN (Levemberg Marquardt) for a 10-point domain.

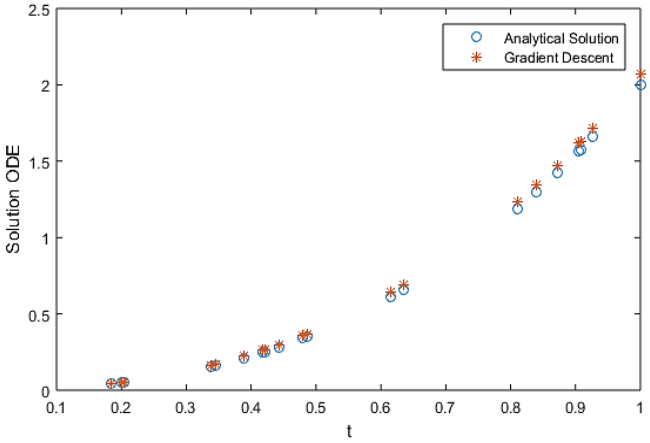

Figure 3. Comparison between exact solution and ANN (Gradient Descent) for a 20-point domain.

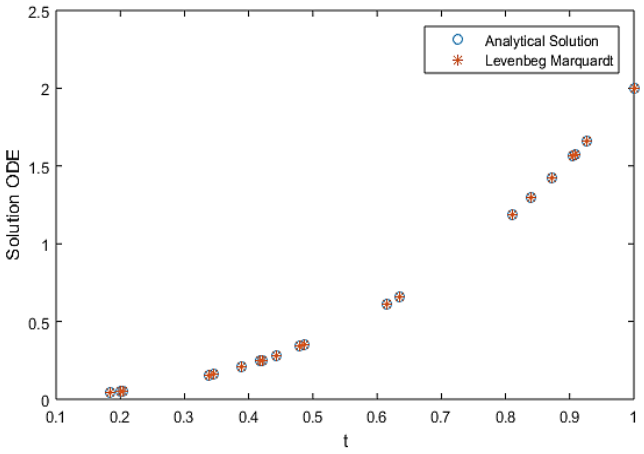

Figure 4. Comparison between exact solution and ANN (Levemberg Marquardt) for a 20-point domain.

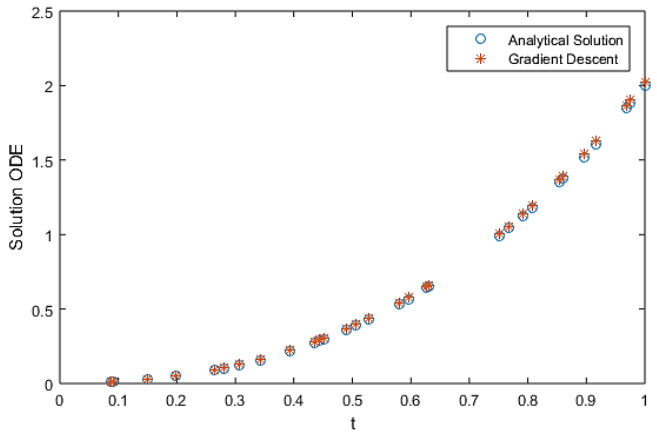

Figure 5. Comparison between exact solution and ANN (Gradient Descent) for a 30-point domain.

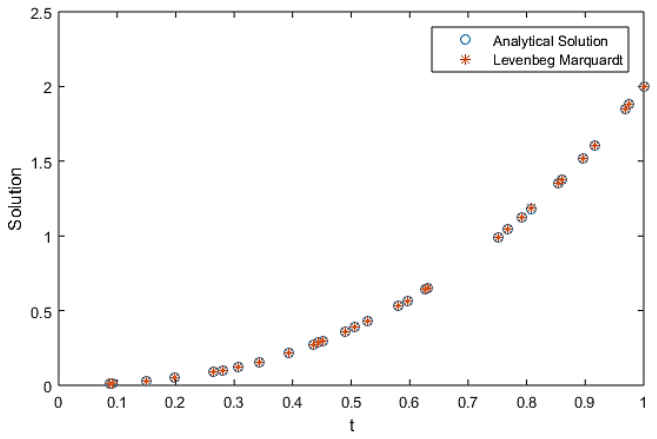

Figure 6. Comparison between exact solution and ANN (Levemberg Marquardt) for a domain of 30 points.

The experiments were repeated ten times for each case (10, 20 and 30 points) in Table 1 are the averages of the 10 repetitions of the Mean Square Error for each experiment. And in Table 2 the means of the number of times of the 10 experiments.

TABLE I. MEAN SQUARE ERROR MEAN

Number of Points	Gradient	Levemberg
10	2.9×10^{-4}	1.8×10^{-7}
20	2.3×10^{-4}	3.5×10^{-8}
30	2.2×10^{-4}	2.9×10^{-8}

TABLE II. MEAN OF THE NUMBER OF EPOCHS

Number of Points	Gradient	Levemberg
10	1709	9
20	20	7
30	47	7

IV. DISCUSSION

After analyzing the presented graphs, we noticed that the optimization method Levemberg Marquardt showed, in general, better than the Gradient Descent. This superiority can be verified by comparing Figures 1 to Figure 2, Figure 3 to Figure 4, and Figure 5 to Figure 6. As each figure shows only the result for a simulation, there was a need to perform of several simulations. The results obtained in 10 simulations varying the number of random points in the domain of analysis, confirm what was realized graphically.

In terms of precision, the best efficacy of the Levemberg Marquardt method is confirmed when Table 1 is evaluated, because for all the experiments (10, 20 or 30 random points) the Mean of the Mean Squared Error was always lower for this optimization method, when compared to the Gradient Descent method. This fact means that Levemberg Marquardt's method results in better approximations for this type of problem.

In addition to a better approximation, another advantage of the Levemberg Marquardt method is the fact that its convergence is, on average, much faster than the Gradient Descent. This fact is important because when working with engineering problems, the time to solve a problem is an important factor when choosing the solution.

V. CONCLUSION

In the light of the above discussion, we can show that despite the two methods presented, converge to an approximation of the solution of the problem, there is a superiority of the optimization method Levemberg Marquardt, in the solution of ODE through Artificial Neural Networks. This superiority occurs both with regard to a smaller approximation error and the time of convergence.

REFERENCES

[1] P. Khargonekar, "Mixed H/sub 2//H/sub infinity / control: a convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 36, nº 7, pp. 824 - 837, 1991.

[2] H. Sasaki, "An interior point nonlinear programming for optimal power flow problems with a novel data structure," IEEE Transactions on Power Systems, vol. 13, nº 3, pp. 870 - 877, 1998.

[3] K. Xie e Y.-H. Song, "Decomposition model and interior point methods for optimal spot pricing of electricity in deregulation environments," IEEE Transactions on Power Systems , vol. 15, nº 1, pp. 39 - 50, 2000.

[4] G. Irisarri, “Maximum loadability of power systems using interior point nonlinear optimization method,” IEEE Transactions on Power Systems , vol. 12, nº 1, pp. 162 - 172, 1997.

[5] V. Quintana, “Interior-point methods and their applications to power systems: a classification of publications and software codes,” *IEEE Transactions on Power Systems*, vol. 15, nº 1, pp. 170 - 176, 2000.

[6] A. Fischer, “A special newton-type optimization method,” *Optimization: A Journal of Mathematical Programming and Operations Research* , vol. 24, nº 3, pp. 269-284 , 2007.

[7] S. N. A. Sofer, “Linear And Nonlinear Programming,” 25 Dez 2017. [Online]. Available: http://pdfhole.info/stephen-nash-ariela-sofer-reading-online-linear-and-nonlinear-programming.pdf. [Acesso em 12 2018 2018].

[8] Y. H. Dai, “A Nonlinear Conjugate Gradient Method with a Strong Global Convergence Property,” *SIAM Journal on Optimization*, vol. 10, nº 1, p. 177–182., 2006.

[9] A. Pessoa, “Artificial Neural Networks: A Meshfree Numerical Method for,” *International Journal of Science and Engineering Investigations* , vol. 7, nº 82, pp. 2251-8843, 2018.

[10] K. M. Brown, “Derivative free analogues of the Levenberg-Marquardt and Gauss algorithms for nonlinear least squares approximation,” *Numerische Mathematik*, vol. 18, nº 4, p. 289–297, 1971.

[11] J. D. Jr., Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Philadelphia: Library of Congress, 1996.

[12] B. Batista, “Soluções de Equações Diferenciais Usando Redes Neurais de Múltiplas camadas com os métodos da Descida mais íngreme e Levenberg-Marquardt.,” Ppgme.Ufpa.Br, 2012.

[13] Levenberg, Kenneth (1944). "A Method for the Solution of Certain Non-Linear Problems in Least Squares". Quarterly of Applied Mathematics. 2: 164–168.

[14] F. A. Gomide, “Redes neurais artificiais para engenharia e ciências aplicadas: curso prático,” Sba Control. Automação Soc. Bras. Autom., 2012.

[15] L. Fleck, M. H. F. Tavares, E. Eyng, and A. HELMANN, “Redes Neurais Artificiais: Princípios Básicos,” Rev. Eletrônica Científica Inovação e Tecnol., 2016.

[16] B. A. Fillenwarth and T. M. Washington, “Improving the Mathematical Model of the Tacoma Narrows Bridge,” vol. 8, no. 2, 2007.

Alcineide Dultra Pessoa, Bachelor in science and technology and Bachelor in Civil Engineering at the Federal University of Maranhão.

Davi Costa Nascimento, Undergraduate student of the Bachelor in Science and Technology, Federal University of Maranhão.

Marta de Oliveira Barreiros, Bachelor in Information Systems from the Faculty of Imperatriz, Master in Biotechnology from the Federal University of Pará. PhD student in the Graduate Program in Electrical Engineering, Federal University of Maranhão.

Thaís Hortiz Santos Alves, Graduated in Science and Technology from the Federal University of Maranhão.

Gean Carlos Lopes de Sousa, Graduated and Master in Mathematics from the Federal University of Pará, with experience in Numerical Analysis of Differential Equations. He is currently a professor at the Federal University of Maranhão and PhD student in the Graduate Program in Electrical Engineering, Federal University of Maranhão.