Journal of University of Babylon for Pure and Applied Sciences,Vol.(26), No.(7): 2018

Modification Of Levenberg-Marquardt Algorithm For Solve Two Dimension Partial Differential Equation

Khalid. Mindeel. M. Al-Abrahemee Rana T. Shwayaa

Department of Mathematics, College of Education, University of AL-Qadisiyhah

Khalid.mohammed@qu.edu.iq

Keywords: Partial Differential Equations; Neural network; Levnberg- Marquardt, signaler value decompositions

Abstract

In this paper we presented a new way based on neural network has been developed for solutione of two dimension partial differential equations . A modified neural network use to over passing the Disadvantages of LM algorithm, in the beginning we suggest signaler value decompositions of Jacobin matrix (J) and inverse of Jacobin matrix(J^{-1}), if $J(\mathrm{w})$ *is* a matrix rectangular or singular . Secondly, we suggest new calculation of μ_k , that is$\mu_k=\|E(w)\|^2$. look the nonlinear execution equations E(w) = 0 has not empty solution W^* and we refer $\|\cdot\|$ to the second norm in all cases ,whereE(w): $R^n \rightarrow R^m$ is continuously differentiable and E(x) is Lipeschitz continuous, that is=$\| E(w_2)- E(w_1)\| \leq L\| w_2- w_1\|$,where L is Lipeschitz constant.

1. Introduction

We have been many methods developed so far to solve differential equations. a few of them generate a result in the type of an range that contain the value of the result at a chosen grouping of points. The result in analytic type and make over the novel problem generally in a system of linear equations. the majority of the preceding work in solving differential equations using Nearul networks is constrained to the case. The result of a linear structure of equations is mapped onto the manner of a Hopfield Neural network. The decreasing of the network's force function provide the solution to the structure of equations.

Here object we outlook the problem from a special angle. We at in attendance a universal manner for solve partial differential equations (PDEs), that depends on function approximate capability of feedforword Nearul networks and outcome in the configuration of a solutione write in a differentiable, near exact form. This figure employ a feedforword nearul network as the basic approximate element, whose parameters' (weights and biases) are use to decreasing an suitable error function. in the direction of teach the network we use optimization technique, which in twirl want the calculation of the grade of the error with value to the network parameters. In the projected move toward the mold function is articulated as the sum of two requisites: the first part satisfy the initial / boundary conditions and contain no modifiable parameters'. The second part involve a feedforward nearul network to be train so as to suit the differential equation by Lee, H. & Kang, I.,1990 , Wang, L. & Mendel J.M.,1990 and Yentis, R. and Zaghoul, M.E.,1996.

2. Levnberg-Marquardt algorithm (LM)

The algorithm of Levnberg-Marquardt (LM) is an purified procedure that locate the least of a more than variable function that is uttered as the sum of squares of non-linear real-valued functions by K. Levenberg ,1994 and D.W. Marquardt ,1963.It has happen to a typical procedure for non-linear least-squares problems, extensively adopt in a wide range of discipline.

For LM algorithm, the performance index to be optimized is defined by H.D. Mittelmann ,2004

$$E(w) = \sum_{P=1}^{P}[\sum_{k=1}^{K}(d_{KP} - O_{KP})^2] \qquad (1) .$$

Where $w = [w_1\, w_2 \dots\dots w_N]$ consistes of all weights of the network, d_{kp} is the desired value of the k^{th} output and the p^{th}pattern, o_{th} is the actual value of the k^{th} output and the p^{th} pattern, N is the number of the weights, P is the number of pattern, and K is the number of the network output .

Equation (1) can be written its

$$E(w) = E^T E \qquad (2) .$$

Where $E = [e_{11} \dots\dots e_{K1} e_{12} \dots\dots e_{K2} \dots\dots e_{1p} \dots\dots e_{KP}]$

$$e_{kp} = d_{kp} - o_{kp} \qquad k = 1, \dots\dots, K \qquad p = 1, \dots\dots, P$$

Where E is the cumulative error vector (for all pattern) from equeation (2) the jacobian matrix is define as=
$$\begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & \cdots & \frac{\partial e_{1,1}}{\partial w_N} \\ \frac{\partial e_{2,1}}{\partial w_1} & \frac{\partial e_{2,1}}{\partial w_2} & \cdots & \frac{\partial e_{2,1}}{\partial w_N} \\ \cdots & \cdots & \cdots\cdot & \cdots \\ \frac{\partial e_{k,1}}{\partial w_1} & \frac{\partial e_{k,1}}{\partial w_2} & \cdots & \frac{\partial e_{k,1}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{1,p}}{\partial w_1} & \frac{\partial e_{1,p}}{\partial w_2} & \cdots & \frac{\partial e_{1,p}}{\partial w_N} \\ \frac{\partial e_{2,p}}{\partial w_1} & \frac{\partial e_{2,p}}{\partial w_2} & \cdots & \frac{\partial e_{2,p}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{k,p}}{\partial w_1} & \frac{\partial e_{k,p}}{\partial w_2} & \cdots & \frac{\partial e_{k,p}}{\partial w_N} \end{bmatrix}$$
(3) .

And the weights are calculated using the following equation

$$w_{t+1} = w_t - (J_t^T J_t)^{-1} J_t E_t \qquad (4) .$$

Where I is identite unit matrix, μ is the learing parameter and J is jacobian of m output error with respect to n weights of the Neural network. The μ parameter is automatically adjusted at each iteration in order to secure convergence, the LM algorithm requires computation of the jacobian J matrix at each iteration step and the inversion of $J^T J$ square matrix, the dimension of which is N*N by B. M. wilamowski and S. Iplikei ,2001.

3. Modification Of High Performance Training Algorithms:

In this part we will improve the training algorithm LM by overcoming some of the disadvantages that have emerged in the LM standard training algorithm, in the beginning we suggest singalur value decompositions of Jacobin matrix (J) and inverse of Jacobin matrix(J^{-1}), if $J(w)$ *is* a matrix rectangular or singular and secondly we suggest new calculation of parameter μ_k that is $\mu_k=||E(w)||^2$ and we refer $||\cdot||$ to the second norm in all cases ,whereE(w): $R^n \to R^m$ is *continuously differentiable* and E(x) is Lipeschitz continuous, that is $||E(w_2)-E(w_1)||\leq L||w_2-w_1||$, where L is Lipeschitz constant.

4. Illustration of the Method

We will consider a two - dimensional of partial defferential equeation (P.D.E).

$$\frac{\partial^2 \Psi(x,\ y)}{\partial x^2} = f(\frac{\partial \Psi(x,\ y)}{\partial x}, \frac{\partial \Psi(x,\ y)}{\partial y}, \frac{\partial^2 \Psi(x,\ y)}{\partial y^2}, \frac{\partial^2 \Psi(x,\ y)}{\partial x \partial y}, x, y) \qquad (5)$$

, belongs to the closed period 0 and 1 with Direchlet B C:

"$\Psi(0, y) = f_0(y), \Psi(1, y) = f_1(y), \Psi(x, 0) = g_0(x) and\ \Psi(x, 1) = g_1(x)$ "

$, where\ f_0, f_1, g_0\ and\ g_1$ are continuous function . The trial solutione is written as

$\Psi_t(x, y) = A(x, y) + x(1 - x)\, y(1 - y)\, N(x, y, \vec{p})$ (6)
where A(x ,y) is chosen so as to satisfy the BC, namely:

$$A(x, y) = (1 - x) f_0(y) + x f_1(y) + (1 - y)\{g_0(x) - [(1 - x) g_0(0) + x g_0(1)]\} + y\{g_1(x) - [(1 - x) g_1(0) + x g_1(1)]\} \qquad (6)\ .$$

For mixed boundary conditions of the form:

$\Psi(0, y) = f_0(y), \Psi(1, y) = f_1(y), \Psi(x, 0) = g_0(x)\ and\ (\partial\Psi(x, 1)/\partial y) = g1(x)$ (i.e., Dirichlet on part of the boundary and Neumann elsewhere), the trial solutione can be written as

$$\Psi_t(x, y) = B(x, y) + x(1 - x) y[N(x, y, \vec{p}) - N(x, 1, \vec{p}) - [\frac{\partial N(x, 1, \vec{p})}{\partial y}] \quad . \qquad (7)$$

And B(x ,y) is again chosen so as to satisfy the BC's:

$$B(x, y) = (1 - x) f_0(y) + x f_1(y) + g_0(x) - [(1 - x) g_0(0) + x g_0(1)] + y\{g1(x) - [(1 - x) g1(0) + x g1(1)]\} \qquad (8)$$

Note that the second term of the trial solutione does not affect the boundary conditions .

In all the above PDE problems the error that should be minimized is given by:

$$E[\vec{p}] = \sum_{i=1}^{n} \{ \frac{\partial^2 \Psi(x,y)}{\partial x^2} + \frac{\partial^2 \Psi(x,y)}{\partial y^2} - f\left(\frac{\partial \Psi(x,y)}{\partial x}, \frac{\partial \Psi(x,y)}{\partial y}, x, y\right) \qquad (9)$$

Where(x_i, y_i) are points in [0,1] × [0,1] .

5. Numerical examples

In this section we reporte some numerical result and the solution of two model problems. In all cases we used a multi layer FFNN having one hiden layer with 7 hidden units (neurons) and one linear output unit. The sigmoid activation of each hidden unit is logsig that is $\sigma(x) = \frac{1}{1+e^{-x}}$. For each test problem the analytic solutione $\Psi_a(\vec{x})$ was known in advance. Therefore we test the accureacy of the obtained solutions by computing the deviation $\Delta\Psi(\vec{x}) = |\Psi_t(\vec{x}) - \Psi_a(\vec{x})|$ (10)

Example 1:

Consider the following 2nd order of two-dimensional of partial defferential equeation for the function $\nabla^2\Psi(x,y) = (2-\pi^2y^2)\sin(\pi x)\ with\ x,y \in [0,1]$ and BC's (Mixed case)

$\Psi(0,y) = 0$, $\Psi(1,y) = 0$, $\Psi(x,0) = 0, \frac{\partial}{\partial y}\Psi(x,1) = 2\sin(\pi x)$, the equeation has the analytic solutione $\Psi_a(x,y) = y^2\sin(\pi x)$ by Abdul-Majid Wazwaz , 2009.

The network treale use a net of ten equidistant points in x∈[0,1] , y∈[0,1] . Figure (1) expand the analytic and nearul solutions activation functions. The neural results with f activation functions with modify Levnberg-Marquardt algorithmic rule (trainmlm) introduced in Table (1) , In table (2) Accurace of solutions, execution of the trail with period and time insert in table (3) and table (4) gave the parameter of initial values.

Table 1: Exact and Approximate solutione of example

input		Exact	$Out\ of\ propose\ FFNN\ yt(x) for\ modify\ of\ LM\ and\ stander$	
x	y	$\Psi_a(\vec{x})$	Modify Trainlm	Trainlm(Standard)
0.0	0.0	0	−0.00202206927449109	$3.10855168354629e-05$
0.1	0.1	0.00309016994374900	0.00138905123743776	0.00297032870364780
0.2	0.2	0.0235114100916989	0.0235114100916989	0.0237062214932224
0.3	0.3	0.0728115294937453	0.0728115294937454	0.0726212142017721
0.4	0.4	0.152169042607225	0.152169042607225	0.152314664724119
0.5	0.5	0.250000000000000	0.250000000000000	0.249894950885798
0.6	0.6	0.342380345866255	0.342380345866255	0.342452052996387
0.7	0.7	0.396418327243724	0.396418327243724	0.396380079895659
0.8	0.8	0.376182561467183	0.366268424395609	0.376193022157707
0.9	0.9	0.250303765443707	0.218693601738362	0.291730607114354
1.0	1.0	$1.22464679914735e-16$	$1.98050739430218e-16$	0.201043695741907

Table 2 : Accurace of solutions for example

$\boldsymbol{The\ error\ \Delta\Psi(\vec{x}) = \lvert\Psi_t(\vec{x}) - \Psi_a(\vec{x})\rvert\ where\ \Psi_t(\vec{x})\ approximate\ values}$	
Modify of lm	**Trainlm(Standard)**
0.00202206927449109	$3.10855168354629e-05$
0.00170111870631171	0.000119841240101675
$3.46944695195361e-18$	0.000194811401523508
$5.55111512312578e-17$	0.000190315291973248
$2.77555756156289e-17$	0.000145622116893923
$5.55111512312578e-17$	0.000105049114202144
0	$7.17071301321037e-05$
0	$3.82473480647905e-05$
0.00991413707157351	$1.04606905236282e-05$
0.0316101637053458	0.0414268416706468
$7.55860595154824e-17$	0.201043695741907

Table 3: execution of the trail with period

$\boldsymbol{Function}$	$\boldsymbol{Pereformance\ of\ teach}$	$\boldsymbol{period}$	$\boldsymbol{Time}$	$\boldsymbol{MSE}$
$Modify\ of\ Train\ lm$	$1.18e-33$	16	0:00:00	$1.004068302142286e-04$
$Trainblm(Standard)$	$1.43e-8$	869	0:00:11	0.003830443569392

Table 4 : weight and bias of initial values for modify Lm teaching algorethm

$\boldsymbol{initial\ values\ for\ trainlm\ (modify)}$			
$\boldsymbol{Net.IW\{1,1}$	$\boldsymbol{Net.IU\{1,1}$	$\boldsymbol{Net.LW\{2,1}$	$\boldsymbol{Net.B\{1\}}$
0.3507	0.9390	0.8443	0.4357
0.8759	0.5502	0.1948	0.3111
0.6225	0.5870	0.2259	0.9234
0.2077	0.3012	0.1707	0.4302
0.4709	0.2305	0.2277	0.1848

$\boldsymbol{initial\ values\ for\ trainlm\ (Standard)}$			
$\boldsymbol{Net.IW\{1,1}$	$\boldsymbol{Net.IU\{1,1}$	$\boldsymbol{Net.LW\{2,1}$	$\boldsymbol{Net.B\{1\}}$
0.2242	0.6525	0.8416	0.2653
0.6050	0.3872	0.7342	0.9246
0.1422	0.0251	0.5710	0.2238
0.4211	0.1841	0.1769	0.3736
0.7258	0.3704	0.9574	0.0875

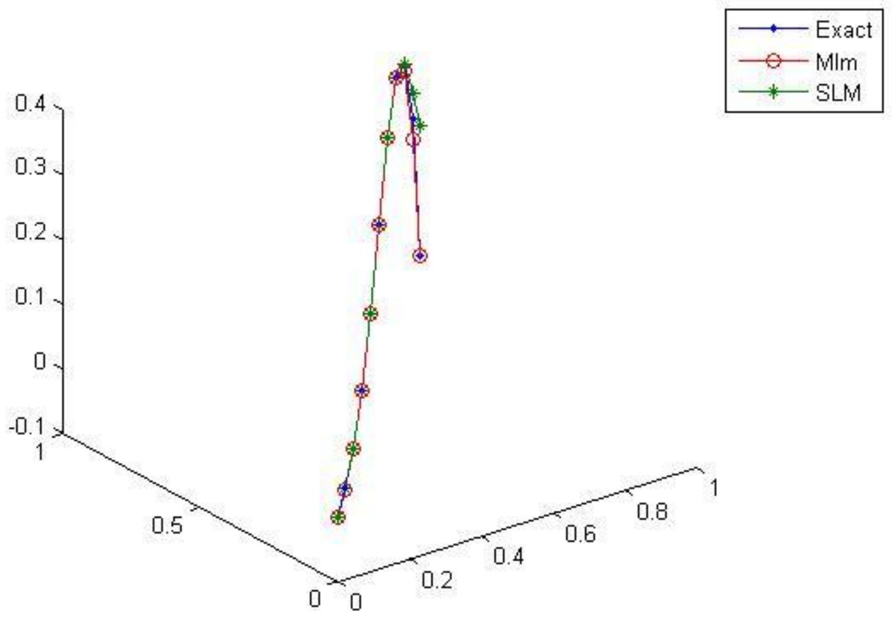

Figure 1:exacat and approximatsolution of example1 using : modify trainlm and stander trainlm algorithms

Example 2:

Consider the following 2nd order of two-dimensional of partial defferential equeation for the function

$$\nabla^2\Psi(x,y) = e^{-\frac{ax+y}{5}}\{\left[-\frac{4}{5}a^3x - \frac{2}{5} + 2a^2\right]\cos(a^2x^2+y) + \left[\frac{1}{25} - 1 - 4a^2x^2 + \frac{a^2}{25}\right]\sin(\mathrm{a}^2x^2+y)\}$$

$with\ a = 3\,, x, y \in [0,1]$ and BC's (Direchlet case)

$\Psi(0,\mathrm{y}) = e^{-\frac{y}{5}}\sin(y)\,,\ \Psi(1,\mathrm{y}) = e^{-\frac{a+y}{5}}\sin(\mathrm{a}^2+y)\,, \Psi(\mathrm{x},0) = e^{-\frac{ax}{5}}\sin(\mathrm{a}^2x^2)\,, \Psi(\mathrm{x},1) = e^{-\frac{ax+1}{5}}\sin(\mathrm{a}^2x^2+1),$ the equeation has the analytic solutione $\Psi_a(x,y) = e^{-\frac{ax+y}{5}}\sin(\mathrm{a}^2x^2+y).\,)$ by Abdul-Majid Wazwaz , 2009.

The network treale use a gridiron of ten points and the difference between points is equal. Figure (2) expand the analytic and nearul solutions activation functions. The neural results with f activation functions with modify Levnberg-Marquardt algorithmic rule (trainmlm) introduced in Table (5) , In table (6) Accurace of solutions, execution of the trail with period and time insert in table (7) and table (8) gave the parameter of initial values.

Table 5: Analytic and Nearul solutione of example

input		Exact solution	$Out\ of\ propose\ FFNN\ yt(x) for\ modify\ of\ and\ stander$	
x	**y**	$\Psi_a(\vec{x})$	**Modify Trainlm**	**Trainlm(Standard)**
0.0	0.0	0	0	0
0.1	0.1	0.204588398543426	0.204588398567885	0.200895773658833
0.2	0.2	0.623352777780964	0.623352777790694	0.598329868276009
0.3	0.3	1.138656193114760	1.138609685746877	1.138656164886924
0.4	0.4	1.327527748349300	1.327527748359200	1.327527775936198
0.5	0.5	0.569371294069937	0.569371294067855	0.906499362901860
0.6	0.6	−1.039133807675550	−1.039133807695874	−1.175970379137568
0.7	0.7	−1.614100223368030	−1.614100223378600	−1.614768239056286
0.8	0.8	0.518294939421891	0.518289600564874	0.887352778509732
0.9	0.9	1.939539059282460	1.939539059749785	1.936573987243657
1.0	1.0	−1.210741248248230	−1.210741247676904	−1.210741248248875

Table 6 : Accurace

$The\ error\ \Delta\Psi(\vec{x}) = \lvert\Psi_t(\vec{x}) - \Psi_a(\vec{x})\rvert\ where\ \Psi_t(\vec{x})\ approximate\ values$	
Modify Trainlm	**Trainlm(Standard)**
0	0
$2.45E-11$	0.003693
$9.73E-12$	0.025023
$4.65E-05$	$2.82E-08$
$9.9E-12$	$2.76E-08$
$2.08E-12$	0.337128
$2.03E-11$	0.136837
$1.06E-11$	0.000668
$5.34E-06$	0.369058
$4.67E-10$	0.002965
$5.71E-10$	$6.38E-13$

Table 7: execution of the trail with period and time

$Function$	$Pereformance\ of\ teach$	$period$	$Time$	MSE
$Modify\ of\ Trainlm$	$2.40e-38$	16	0: 00: 00	$1.00486368693728e-07$
$Trainblm(Standard)$	$5.22e-5$	869	0: 00: 11	0.0287563073606448

Table 8: weight and bias of initial values for modify Lm training algorethm

$parameter\ of\ initial\ values\ for\ modify\ lm$			
$Net.IW\{1,1$	$Net.IU\{1,1$	$Net.LW\{2,1$	$Net.B\{1\}$
0.4563	0.4472	0.1567	0.4887
0.5789	0.2960	0.1237	0.3986
0.2356	0.4598	0.5743	0.7954
0.6686	0.2398	0.9835	0.3986
0.0236	0.9347	0.4629	0.7698

$parameter\ of\ initial\ values\ for\ Standardlm$			
$Net.IW\{1,1$	$Net.IU\{1,1$	$Net.LW\{2,1$	$Net.B\{1\}$
0.3785	0.6973	0.4787	0.4893
0.7065	0.2765	0.1298	0.5209
0,0365	0.0875	0.6578	0.8763
0.3678	0.6783	0.5398	0.7843
0.0549	0.1857	0.2987	0.7987

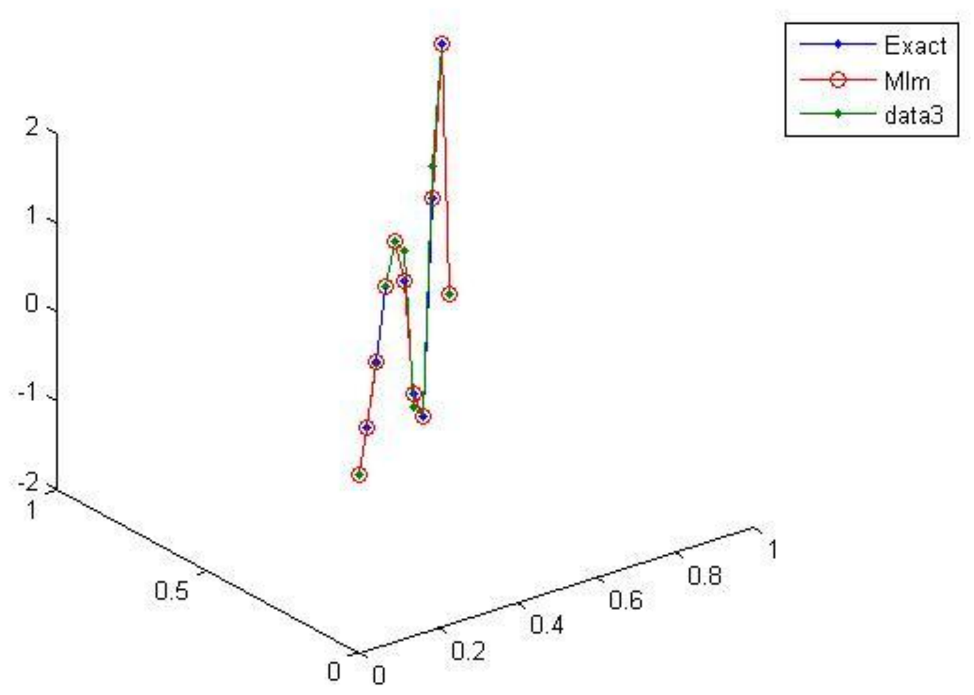

Figure 1:exact and approximate solution of example2 using : modify trainlm and stander trainlm algorithms

References

[1] H. Lee, I. Kang, Nearul algorithms for solving defferential equeations, Journal of Computational Physics 91 (1990) 110–117.
[2] L. Wang, M. J.M., Structured trainable networks for matrix algebra, IEEE Int. Joint Conference on Nearul networks 2 (1990) 125–128.
[3] R. Yentis, M. Zaghoul, Vlsi implementation of locally connected nearul network for solving paretial defferential equeations, IEEE Trans on Circuits and Systems-I 43 (8) (1996) 687–690.
[4] K. Levenberg, A method for the solutione of certain non-linear problems in least squares, Quarterly of Applied Mathematics 2 (2) (1944) 164–168.
[5] D. W. Marquardt, An algorithm for the least-squares estimation of nonlinear parameters, SIAM Journal of Applied Mathematics 11 (2) (1963) 431–441.
[6] H. D. Mittelmann, The least squares problem (2004). URL http://plato.asu.edu/topics/problems/nlolsq.html
[7] B. M. wilamowski, S. Iplikei, an algorithm for fast convergence in training nearul network (2001).
[8] A.-M. Wazwaz, Partial differential equations and solitary waves theory (2009).

الخلاصة

في هذه الدراسة تم تطوير طريقة جديدة تقوم على الشبكة العصبية من أجل حل المعادلات التفاضلية الجزئية البعدين. استخدام الشبكة العصبية المعدلة لتجنب عيوب خوارزمية التدريب لـقنبرك – ماركوادت. أولا نقترح SVD تحليل القيمة المنفردة إلى J و J^{-1} إذا كانت المصفوفة J(w) مستطيلة او منفردة . ثانيا نقترح حساب جديد إلى μ بحيث ان $\mu_k = ||E(w)||^2$. نعتبر ان دالة الهدف الغير خطية E(w) تملك مجموعة غير خالية من الحلول W^* ونشير أن || || هو من المعيار ٢ و $E(w): R^n \rightarrow R^m$ هي مستمرة وقابلة للاشتقاق وتحقق شرط $||E(w_2) - E(w_1)|| \leq L||w_2 - w_1||$ حيث ان L هو ثابت ليبشيتز.

الكلمات المفتاحية: المعادلات التفاضلية الجزئية, الشبكات العصبية الصناعية, خوارزمية التدريب لـقنبرك – ماركوادت , تحليل القيمة المنفردة.