MODIFIED RELAXATION METHOD FOR SOLUTION OF CONTINUOUS RECURRENT NEURAL NETWORKS

Bogdan M. Wilamowski and Stanley M. Kanarowski
Department of Electrical Engineering
University of Wyoming
Laramie, WY 82071

ABSTRACT

The derivation of a modified relaxation algorithm is presented followed by demonstration examples. The algorithm converges very well for continuous recurrent neural networks with both low and high gain neurons. This enables one to simulate recurrent Hopfield networks with both "soft" and "hard" continuous activation functions. The algorithm is suitable for large systems since the computational effort is proportional only to the system size, in contrast to the commonly used Newton-Raphson method where power relationships exist.

I. INTRODUCTION

Signals in recurrent neural networks can propagate in both directions, contrary to feed forward networks. This can create oscillations or upward latching in certain stages. One layer recurrent neural networks which have the ability to latch up are known as Hopfield networks. This concept is extended to single or multilayer associative memories. Recurrent neural networks such as Hopfield networks [1] are used for solving many practical problems including optimization [2], storing and retrieving patterns [3], solving linear equations systems [4], linear and nonlinear programming [5] and other related problems as listed in [6].

Such systems are usually difficult to analyze. There are several known approaches. One group of approaches describe networks by a set of differential equations and use one of the many methods for solving differential equation sets in the time domain. The simplest approach uses the forward or backward Euler formula. To avoid numerical instability, the time step in this method of transient analysis should be very small. This requires a large amount of computation before the steady state condition is reached. Another possible approach is to apply the Newton-Raphson algorithm, which is commonly used for analysis of electronic circuits. This method requires solving a large set of linear equations at each iteration step. Only a few iterations are needed before the solution is reached, however each iteration is very computationaly intensive. In contrast, the relaxation method is the simplest to use, but it will not converge for neural networks with high neuron gains when used as described in [7]. In addition, the convergence problem is more severe in larger networks.

In this paper the extension of "asynchronous" updating for continuous recurrent networks is discussed, followed by two modifications of the relaxation method. The paper then shows several numerical experiments. Finally, the paper concludes that with the introduced modifications to the relaxation method, the analysis of recurrent networks with both high and low neuron gains converge well.

II. ASYNCHRONOUS UPDATING

The relaxation method, as it was presented in [7], uses the following formulas for updating voltages in unipolar recurrent networks:

$$u_i^{k+1} = \frac{1}{G_i} \sum_{j=0}^{n} \left(w_{ij} v_j^k + i_i \right) \tag{1}$$

and

$$v_i^{k+1} = \frac{1}{1 + \exp\left(-\lambda u_i^{k+1} \right)} \tag{2}$$

where v_i and u_i are input and output voltages of i-th neuron. When these formulas are used for neurons with high λ coefficients (high neuron gains) then the computation process does not converge [7]. In this paper we have experimented with the asynchronous method of updating voltages. This means that at each iteration only a voltage on one neuron output was allowed to be changed, while the voltage values on all other outputs were maintained. Using this simple modification of the original relaxation formula, very fast convergence was possible for both low and high gain neural networks.

Figure 1 illustrates the convergency ratio for a two bit analog to a digital converter with x = 1.3 V. When λ = 1.0 (Fig. 1.a), the system converges well, but with λ = 10.0 (Fig.1.b) the solution is not reached.

Figure 2 illustrates the convergency for a two bit analog to digital converter to converge x = 1.3V with (a) λ = 1.0, (b) λ = 10.0 and (c) λ = 100.0 As one can see from Figure 2, the simple modification results in the good program convergency even for very large λ. This is because the Hopfield network was used in these experiments. The Hopfield network is a special case of the standard one layer recurrent neural networks. In the Hopfield networks the weight matrix has to be both symmetrical and have no connections between inputs and outputs of the same neuron [1]. Such networks will always converge to certain states which are located at the corners of the (-1,1) hypercubes.

General recurrent neural networks with one or more layers usually do not converge when the asynchronous updating is applied. From our numerical experiments we have concluded that the

CH 3381-1/93/$01.00 ©1993 IEEE

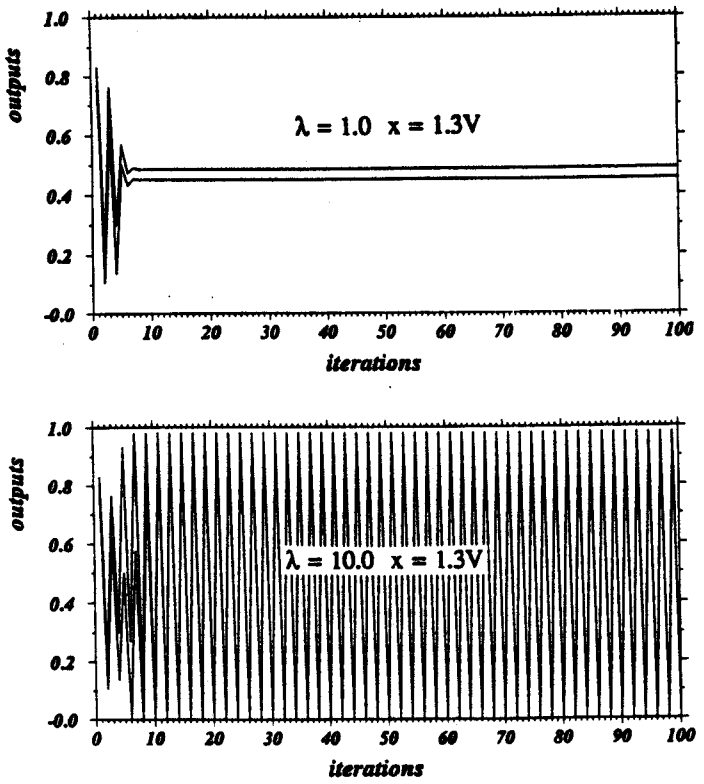

Fig. 1. Output results for a 2-bit A/D converter with x = 1.3 converging to the binary state *01* using the standard relaxation technique: (a) λ = 1.0; (b) λ = 10.0.

asynchronous updating does not work when the solution vector is located far from the corners of the (-1,1) hypercube.

III. MODIFICATION OF RELAXATION ALGORITHM

The charge conservation principle [8], implemented in the CHARCO program [9], is one of the most efficient methods for circuit analysis. This algorithm for transient analysis of integrated circuits does not require solutions of equations at each iteration step and thus only the explicit computations are required. For example, in the case of medium size circuits, (20 MOS transistors), the algorithm is about 100 times faster than the algorithm used in the SPICE program. For large circuits the comparison is even more favorable. Obviously this algorithm has some other drawbacks.If the charge conservation algorithm is applied for recurrent neural networks, then at each iteration step the unbalanced current of each capacitor is computed using equations:

$$cur_i = C_i \frac{du_i}{dt} = \sum_{j=0}^{n} w_{ij} v_j - G_i u_i + i_i \qquad (3)$$

where

$$G_i = \sum_{j=1}^{n} w_{ij} + g_i \qquad (4)$$

This current cur_i is charging or discharging the corresponding capacitor depending upon it's sign. Assuming a small time increment, the corresponding voltage change on the input nodes can be found using:

$$\Delta u_i = \frac{\Delta t \; cur_i}{C_i} \qquad (5)$$

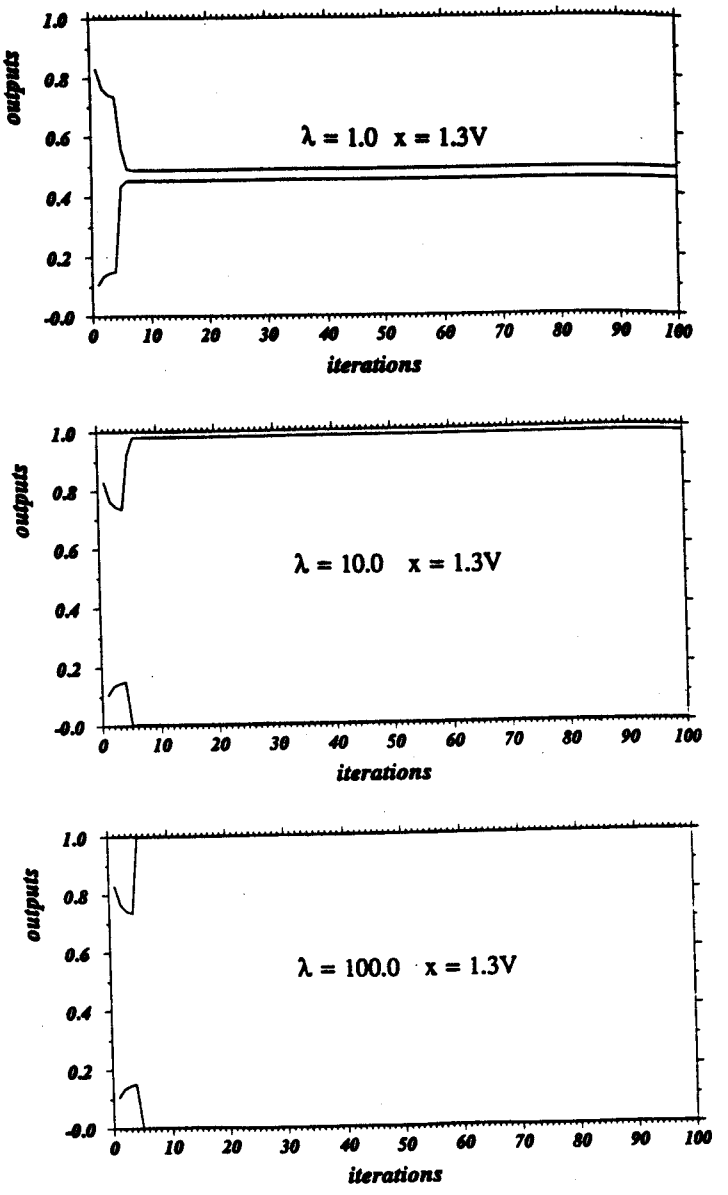

Fig. 2. Output results for a 2-bit A/D converter with x = 1.3 converging to the binary state *01* using the asynchronous updating technique: (a) λ = 1.0; (b) λ = 10.0; (c) λ = 100.0.

The above algorithm will converge if the time increment Δt is chosen small enough. One can prove that if the value of Δt is equal to the time constant of the input node than the computation is equivalent to the relaxation algorithm. Thus, the relaxation method, as described in [6], is a special case of the method, based on the charge conservation principle. Therefore, like the relaxation method, the algorithm with Δt=τ will not converge for large values of neuron gain λ. To assure convergence, Δt in equation (5) should be reduced so that the voltage change at each iteration step is small. In the case of recurrent neural network, a further simplification of the algorithm is possible since only a steady state solution is required. In addition, one can prove that the relaxation formula can be useful if the results obtained from equation (1) are than modified in the following way:

$$u_i^{new} = u_i^{old} + \frac{1}{A_i + 1} \left(u_i^{relax} - u_i^{old} \right) \qquad (6)$$

where A_i is the gain of the corresponding neuron and u_i^{relax} is obtained from the relaxation formula (1). The one in the denominator of (6) is required to limit the voltage change for neurons with very small

gains. Neurons with the unipolar activation functions (2) will always have a gain A_i smaller than $\lambda/4$. Instead of using the maximum gain of $A_i = \lambda/4$, it is possible to speedup the computation process and still guarantee convergence. This is done by computing the gain of each neuron at every iteration step using the derivative of the activation function:

$$A_i = \frac{\partial v_i}{\partial u_i} = \lambda_i v_i (1 - v_i) \qquad (7)$$

Two modification of the basic relaxation formula are possible. Both use the equation (6), the first modification uses the fix neuron gain $A_i = \lambda/4$, while the second uses the adjustable gain, which depends on the signal value as it is given by equation (7).

IV. EXAMPLE RESULTS

The algorithm was proven to converge by using the same example as in [6]. Figure 3 shows the results when the fixed gain approach was used. Figure 4 shows similar results for the adjustable gain approach when equation (7) was used. In both cases the algorithm converges very well. The faster convergence is obviously with adjustable gain.

In addition to two-bit, four- and eight-bit analog to digital converters were tested with single-layer recurrent Hopfield networks. Fig. 5 shows the results for the 4-bit A/D converter, with x = 9.0, λ = 100 and using both the fixed and the adjustable gain approaches. One can see from this figure that the algorithm with adjustable gain converges in approximately 20 iterations while algorithm with fixed gain requires more than 100 iterations to reach a solution. The difference is even more significant in the case of the 8-bit AD converter. Thus one can see that when using the formula (7) the system converges more rapidly.

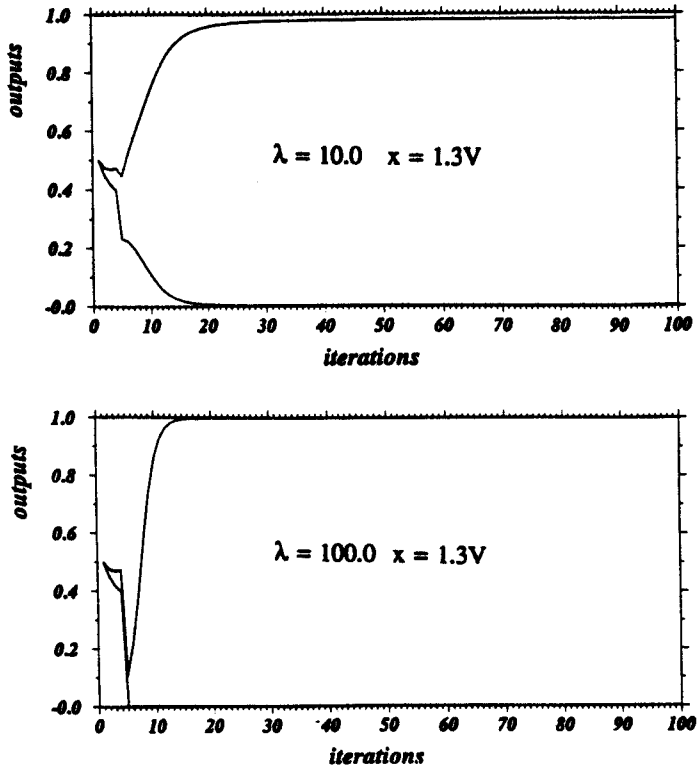

Fig. 3. Output results for a 2-bit A/D converter with x = 1.3 converging to the binary state *01* using the modified relaxation technique with the fixed gain approach: (a) λ = 10.0; (b) λ = 100.0.

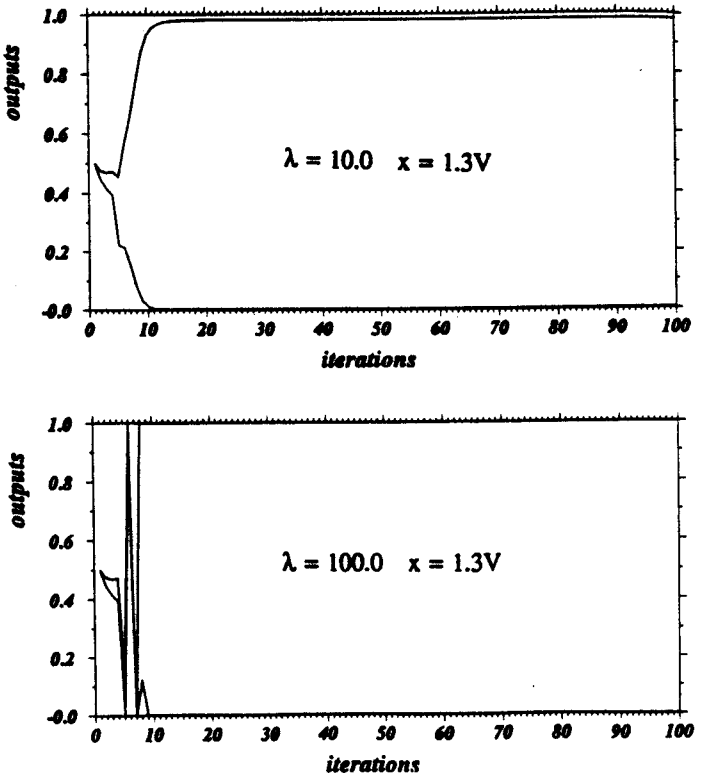

Fig. 4. Output results for a 2-bit A/D converter with x = 1.3 converging to the binary state *01* using the modified relaxation technique with the adjustable gain approach (Eq. (7)): (a) λ = 10.0; (b) λ = 100.0.

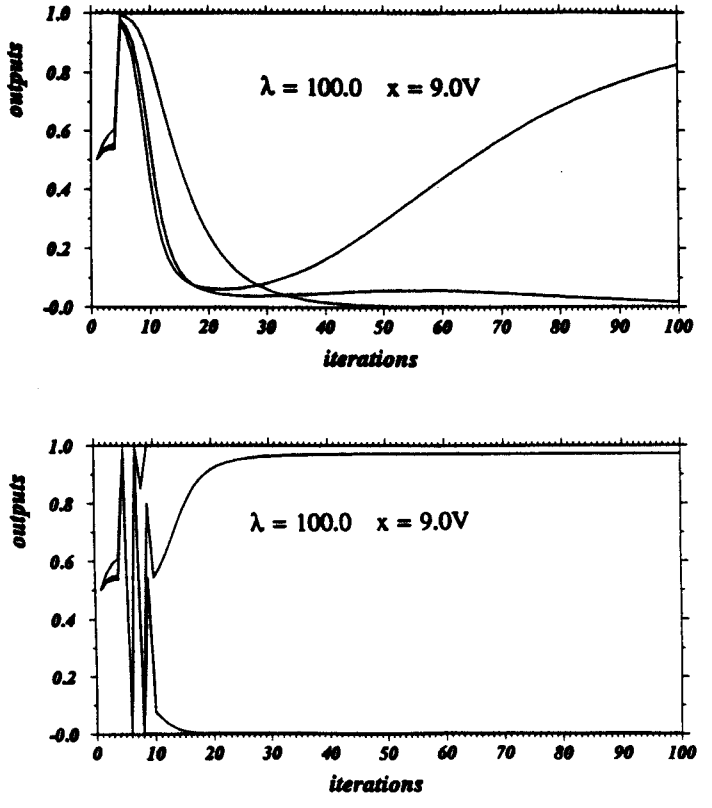

Fig. 5. Output results for a 4-bit A/D converter with λ = 100 and x = 9.0 converging to the binary state *1001* using the modified relaxation technique with: (a)fixed gain approach and (b) the adjustable gain approach.

V. CONCLUSION

It was demonstrated that through a slight modification of the relaxation algorithm it is possible to insure convergence for any neuron gain (λ in the activation function). This enables one to simulate recurrent Hopfield networks with both "soft" and "hard" continuous activation functions. Furthermore the proposed algorithm requires far less computation effort than applying the Newton-Raphson to equation set (1) or any method of solving differential equations. It should be noted that algorithm is suitable for large systems since the computational effort is proportional only to the system size, in contrast to the commonly used Newton-Raphson method where power relationships exist. Thus the use of the modified relaxation method is applicable to both small and large size networks regardless of neuron gain.

REFERENCES

[1] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two state neurons," in *Proc. Nat. Acad. Sci.*, vol. 81, pp. 3088-3092, 1984

[2] J.M. Zurada, "Gradient-type neural systems for computation and decision-making," in *Progress in Neural Networks, Vol. II*, O. M. Omidvar, Ed. Norwood, NJ:Ablex, 1991

[3] D.W. Tank. and J.J. Hopfield, "Simple neural optimization networks: An A/D converter, signal decision circuit and a linear programming circuit, " *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 533-541, May 1986

[4] A. Cichocki and R. Unbehauen, "Neural Networks for Solving System of Linear Equations and Related Problems" IEEE Trans. on Circuits and Syst. I, CASI-39, pp. 124-138, 1992.

[5] M.P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," IEEE Trans. Circuits Sys., vol CAS-35, pp. 554-562, May 1988.

[6] J.M. Zurada and M.J. Kang, "Computational circuits using neural optimization concepts," *Int. J. Electron.*, vol. 67, no. 3, pp. 311-320

[7] J.M. Zurada and W. Shen, "Sufficient Condition for Convergence of a Relaxation Algorithm in Actual Single-Layer Neural Networks" *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 300-303, Dec. 1992

[8] B. M. Wilamowski, D. J. Hamilton, Z. J. Staszak, A. J. Majewski, "Analysis of digital integrated circuits using charge conservation principle," Electron Technology, vol 19, no. 1/2 pp.59-73, 1986

[9] B. M. Wilamowski, Z. J. Staszak, D. J. Hamilton, "CHARCO - IC transient analysis program," IEEE International Circuit and System Conference, San Jose, California, pp. 548-551, May 5-7, 1986