Stable Solutions of Continuous Recurrent Neural Networks

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

Abstract

An expanded contraction mapping theorem to single layer feedback neural networks of a gradient-type is discussed. The derivation of a modified relaxation algorithm is presented followed by a brief demonstration example.

Introduction

There are currently many different types of recurrent neural networks in use; Hopfield, Bidirectional Associative Memories and Multidirectional Associative Memories (BAM and MAM). These recurrent networks are used to solve problems such as optimization [1], storing and retrieving patterns [2] or circuit generations (analog to digital converters) [2]. In such cases a numerical simulation is always required before going to a hardware implementation.

An exact transient solution requires the set of differential equations (1) to be solved in traditional ways. The most commonly used solution is the forward or backward Euler formula. Such an approach typically requires a large number of iteration steps with small time steps before a steady-state solution is reached. The Newton-Raphson method can be used when a steady-state solution is required. This method is commonly used for electronic circuit analysis. Although this method requires only a few iteration steps, it is computationally very intensive. Another possible method is to use the relaxation method. It is much easier to implement, but it converges only for neurons with small gains. The gain limitations become more severe for larger systems [3]. In many applications, high gain neurons are required and thus this method can not be used.

In this paper the modified incremental relaxation method is presented. Similar to the relaxation method no equations need to be solved. This method converges very well for all neuron gains, even for extremely large gains as required for hard threshold discrete neurons.

Algorithm Principles.

The basic system diagram is shown in Fig. 1. A typical activation function $v_i = f(u_i) = [\ 1 + \exp(-\lambda u_i)\ ]^{-1}$ is used in this network. The input conductances and capacitances are equal to g_i and C_i, respectively. The network transient response is described by the following set of differential equations [2][3]:

$$C_i \frac{du_i}{dt} = \sum_{j=0}^{n} w_{ij} v_j - G_i u_i + i_i = cur_i \qquad (1)$$

where

$$G_i = \sum_{j=1}^{n} w_{ij} + g_i \qquad (2)$$

Each side of equation (1) is also equal to the unbalanced sum of the currents cur_i flowing toward the node. This also corresponds to the appropriate i-th component of the energy function gradient. When these unbalanced currents cur_i are equal to zero, the node capacitors are not charging and the system is in a steady-state condition. Furthermore the system energy reaches a minimum when all components of the gradient vector are equal to zero.

The relaxation algorithm requires the iterative computation of successive output values using the formula:

$$u_i^{k+1} = \frac{1}{G_i} \sum_{j=0}^{n} \left(w_{ij} v_j^k + i_i \right) \qquad v_i^{k+1} = f\left(u_i^{k+1}\right) \tag{3}$$

The convergence condition is satisfied when the sum of the closed loop gains through all possible paths is smaller than one. This can be written as:

$$n \left| \frac{\partial f_i(v_i)}{dv_i} \right| < 1 \tag{4}$$

This leads to the requirement given by [3]:

$$\lambda \leq \frac{4}{n} \left| \frac{G_i}{w_{ij}} \right| \tag{5}$$

A different approach to obtaining a steady-state solution, is to use the forward Euler formula for solving the differential equations as given in **(1)**. This algorithm follows the physical phenomena of the circuit very strictly. After each iteration step the unbalanced current of every node is computed using equation **(1)**. This current is either 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:

$$\Delta u_i = \frac{\Delta t \ cur_i}{C_i} \tag{6}$$

The next logical step is to update the values of all the voltages using the relationships:

$$u_i^{k+1} = u_i^k + \Delta u_i^k \qquad v_i = f(u_i) \tag{7}$$

The above algorithm will converge if the time increment Δt is chosen small enough. The value of Δt can be chosen to be equal to the time constant of the input node:

$$\Delta t = \tau = \frac{C_i}{G_i} \tag{8}$$

Equation **(3)** of the relaxation method can easily be derived by combing **(1)(6)(7)** and **(8)**. Thus the relaxation method as described in [3] is a special case of the forward Euler formula for solving a set of differential equations with the assumption that Δt is equal to the time constant given by **(8)**. Therefore the algorithm given by **(1)(6)(7)** and **(8)** will not converge for large values of λ, just like the relaxation method. To assure convergence, Δt in equation **(6)** should be reduced so that the voltage changes at each iteration step are small. By doing this the incremental gain of the close loop of equation **(4)** is below 1.0 and thus stabilizes the algorithm.

Instead of using the algorithm described by equations **(1)(6)(7)** and **(8)** it is also possible to change the relaxation formula **(2)** in the following way:

$$u_i^{new} = u_i^{old} + \frac{1}{A_i + 1}\left(u_i^{computed} - u_i^{old} \right) \quad (9)$$

where A_i is the gain of the corresponding neuron and $u_i^{computed}$ is obtained from the relaxation formula (2). The assumed activation function will always have a gain smaller than $\lambda/4$. Therefore the modified relaxation algorithm

$$u_i^{new} = u_i^{old} + \frac{1}{\frac{\lambda_i}{4} + 1}\left(u_i^{computed} - u_i^{old} \right) \quad (10)$$

will insure convergence. Instead of using the maximum gain of $A_i = \lambda/4$, it is possible to speedup the computation process and still guarantee convergence by computing the gain of each neuron after every iteration step using:

$$A_i = \frac{\partial v_i}{\partial u_i} = \lambda_i v_i(1 - v_i) \quad (11)$$

Example results

The algorithm was proven to converge by using the same example as in [3]. In addition to two-bit, four- and eight-bit analog to digital converters were tested with single-layer recurrent Hopfield networks. The value of the circuit components were obtained using the equations:

$$w_{ij} = -2^{i+j} \qquad i_i = 2^{ix} - 2^{2i-1} \quad (12)$$

Fig. 2(a) shows the results for the 2-bit A/D converter, with x = 1.3 and λ = 0.3,1,3,10,30,100. This circuit converges to a binary state of *01*. The results for the same 2-bit A/D converter using equations (9) and (11) are depicted in Fig. 2(b). Fig. 3 illustrates the convergence of an 8-bit A/D converter with an input value equal to 182.3. This converter converges to the binary state of *11000000*. Fig. 3(a) illustrates the results for a network with λ = 100, while Fig. 3(b) is for λ = 10,000. Thus one can see that when using the modified formulas **(9)** and **(11)** system converges more rapidly.

Conclusion

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. Thus the use of the modified relaxation method is now applicable to both small and large size networks regardless of neuron gain.

References

[1] 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

[2] 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

[3] 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

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

[5] 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

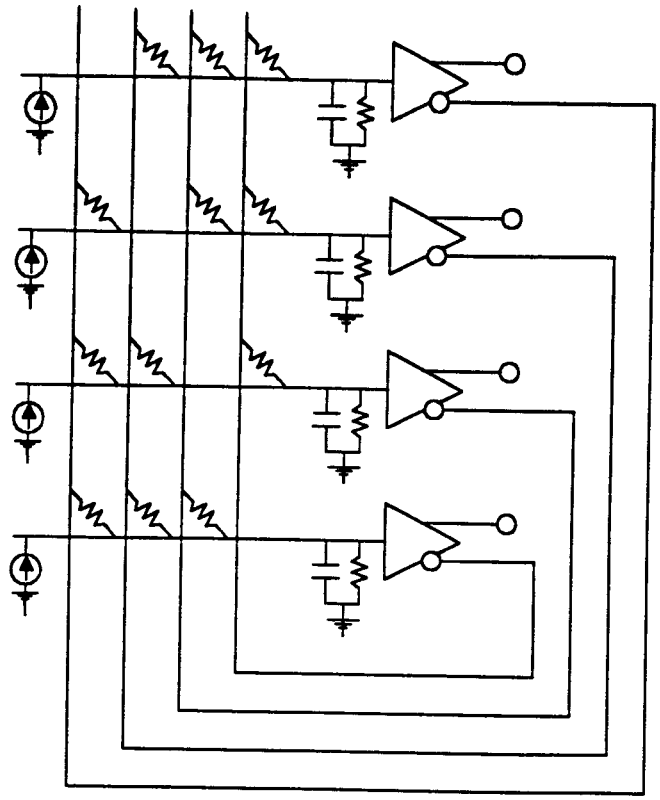

Fig. 1. One layer Recurrent Neural Network with four neurons.

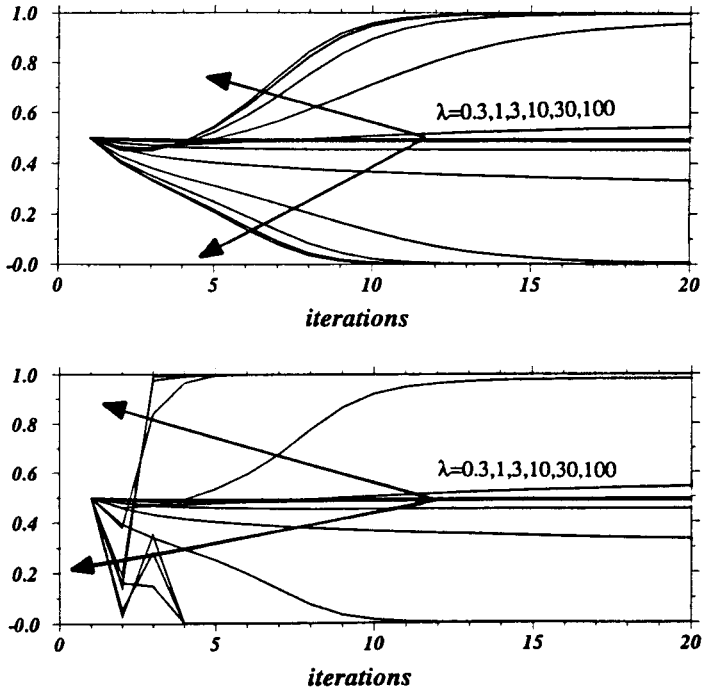

Fig. 2. Output results for a 2-bit A/D converter with x = 1.3 and λ = 0.3, 1, 3, 10, 100 which converges to the binary state *01*; (a) algorithm uses equation (10); (b) algorithm uses equations (9) and (11)

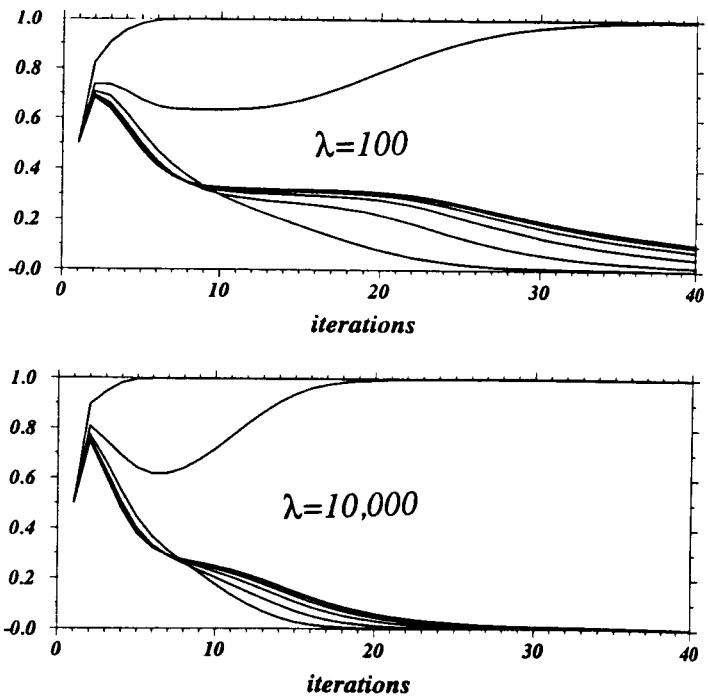

Fig. 3. Output results using the algorithm given by (9) and (11) for an 8-bit A/D converter with x = 182.3 which converges to the binary state *11000000*; (a) λ = 100; (b) λ = 10,000.

Volume I

WORLD CONGRESS ON NEURAL NETWORKS

1993 INTERNATIONAL
NEURAL NETWORK SOCIETY
ANNUAL MEETING

OREGON CONVENTION CENTER
PORTLAND, OREGON
JULY 11-15, 1993

LEA