Appl. Math. Lett. Vol. 6, No. 3, pp. 85–89, 1993
Printed in Great Britain

0893-9659/93 $6.00 + 0.00
Pergamon Press Ltd

ACCELERATING NEURAL NET DYNAMICS BY BOUNDARY LAYER METHODS

WILLARD L. MIRANKER
Mathematical Sciences Department, IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598, U.S.A.
ERIC MJOLSNESS
Computer Science Department, Yale University
New Haven, CT 06520, U.S.A.

(Received and accepted July 1992)

Abstract—A boundary layer method for accelerating the solution of the differential equations representing the dynamics of an analog relaxation neural net in a high gain limit is presented. The inverse of the gain parameter in an analog neuron's transfer function is used as a small parameter, in terms of which the net dynamics may be separated into two time scales. This separation leads to economies in the numerical treatment of the associated differential equations, i.e., the acceleration in question. Illustrative computations are presented.

1. INTRODUCTION

Analog neural networks can be designed very effectively by using optimization principles together with a descent dynamics that can be realized as a circuit composed of analog neurons. This approach was suggested independently by Grossberg [1] and Hopfield [2], and the resulting neural networks, analog relaxation neural networks (ARNN), are often referred to as *Hopfield nets*. Neural networks for a variety of computational tasks have been designed this way. Networks that are derived directly by this method may be slow to converge to an optimal answer, for example, because the condition number of the coefficient matrix of the associated circuit dynamics' equations (a Jacobian in the nonlinear case) can be large. The same problem arises throughout scientific computing and may often be addressed by acceleration methods.

Here, we develop a *boundary layer method* for accelerating the solution of the differential equations representing the dynamics of an ARNN. We have previously adapted multigrid methods to the acceleration of such neural networks' dynamics [3]. Boundary layer methods are used in solving systems of differential equations and partial differential equations because they are powerful methods for stiff or singularly perturbed problems. They work by identifying some small parameter in terms of which a separation of the problem into two distinct but coupled time or space scales (a fine scale and a coarse scale) may be made. The fine scale dynamics is trivial except in a small region of space-time called the *boundary layer*. Given such a small parameter, the separation of scales can often be achieved in a standard way, described in [4].

The differential equations representing an ARNN's dynamics, like any analog circuit dynamics, have a special form which must be taken into account in order to develop an effective boundary layer method. Since these nets are generally run at high gain, we use the inverse of the *gain parameter* in an analog neuron's transfer function as the small parameter that separates two time scales. The boundary layer will consist of a subset of the cross product of the set of neurons and the time axis: namely those time intervals, for each neuron, during which the neuron is in the high-gain region of its transfer function. (For a small network, one could more simply say that the boundary layer is the union, on the time axis, of all such intervals over all neurons, but for a large network, this subset would not be small.) In Section 2, we explain the technique

for separation into scales, including a description of the numerical treatment of the separated problem. (The formalism of this scale separation is given in the Appendix.) In Section 3, a computation illustrates the new technique.

2. APPLICATION OF THE BOUNDARY LAYER FORMALISM

The dynamics of an ARNN are specified by a system of the form

$$\begin{aligned} \dot{\boldsymbol{u}} + \boldsymbol{u} &= \boldsymbol{F}(\boldsymbol{v}), \qquad t > 0, \\ \boldsymbol{v} &= \boldsymbol{g}(\boldsymbol{u}), \end{aligned} \tag{1}$$

with $\boldsymbol{u}(0)$ being given. $\boldsymbol{g}$ is the *sigmoid*, which we take in the form

$$\boldsymbol{g}(\boldsymbol{z}) = \boldsymbol{g}(\boldsymbol{z}, \varepsilon) = \frac{1}{1 + e^{-\boldsymbol{z}/\varepsilon}},$$

where the definition is componentwise when $\boldsymbol{z}$ and $\boldsymbol{g}$ are vectors. Here, ε is a scale parameter. The solution of this system is sought, so that the equilibrium value of $\boldsymbol{u}$ may be determined. The numerical solution of the initial value problem (1) may be obtained by introducing a mesh of t values $\{j\,\Delta t,\ j = 0, 1, \dots\}$ and employing some numerical method. We seek to accelerate this solution process when ε is a small positive parameter. In this case, the boundary layer methods of singular perturbation theory will come into play.

Setting $\boldsymbol{h}(\boldsymbol{z}) = \frac{1}{2}(1 + \operatorname{sig} \boldsymbol{z})$, we have

$$\boldsymbol{g}(\boldsymbol{z}, \varepsilon) = \boldsymbol{h}(\boldsymbol{z}) + O\left(e^{-\min_i |z_i|/\varepsilon}\right).$$

So when no component of $\boldsymbol{u}$ is small in magnitude compared to ε, the solution $\boldsymbol{u}$ of (1) may be approximated to good accuracy by the solution of

$$\dot{\boldsymbol{u}} + \boldsymbol{u} = \boldsymbol{F}(\boldsymbol{h}(\boldsymbol{u})), \tag{2}$$

which is simpler (cheaper) to solve than (1). There are at least two reasons for this:

(i) Equation (1) is a stiff equation for small ε, whereas (2) is not stiff. Thus, far fewer time steps are needed for numerically solving (2) than (1).

(ii) The limiting form $\boldsymbol{F}(\boldsymbol{h}(\boldsymbol{u}))$ is usually simpler than $\boldsymbol{g}(\boldsymbol{z}, \varepsilon)$, so that less arithmetic is involved in evaluating $\boldsymbol{F}$ compared to $\boldsymbol{g}$. Equation (2) may be used until one or more components of $\boldsymbol{u}$ becomes $O(\varepsilon)$.

Say all such components are less than $c\varepsilon$ in magnitude, c being some fixed positive constant. We take this as an indication that these components are in process of changing sign, and we call such a state of the solution a *transition* (of sign). Call $\boldsymbol{u}_2$ the block of components in transition, and call $\boldsymbol{u}_1$ the remaining block. Call the time at which this state commences $t = L^-$. Corresponding to the block decomposition of $\boldsymbol{u}$, compose a block decomposition of $\boldsymbol{F} = (\boldsymbol{F}_1, \boldsymbol{F}_2)^\top$. In applications,

$$\boldsymbol{F}(\boldsymbol{v}) = \boldsymbol{l} + \boldsymbol{T}\boldsymbol{v}, \tag{3}$$

where $\boldsymbol{l}$ is a fixed vector and $\boldsymbol{T}$ a fixed matrix. Denote the corresponding block decomposition of $\boldsymbol{l}$ and $\boldsymbol{T}$ by

$$\boldsymbol{l} = \begin{pmatrix} \boldsymbol{l}_1 \\ \boldsymbol{l}_2 \end{pmatrix} \quad \text{and} \quad \boldsymbol{T} = \begin{pmatrix} \boldsymbol{T}_{11} & \boldsymbol{T}_{12} \\ \boldsymbol{T}_{21} & \boldsymbol{T}_{22} \end{pmatrix}. \tag{4}$$

Now let us introduce new variables by scaling, as follows:

$$(\boldsymbol{u}_1, \boldsymbol{u}_2)^\top = (\boldsymbol{x}, \varepsilon\,\boldsymbol{y})^\top;$$

$\boldsymbol{y}$ may be approximated within $O(\varepsilon)$ by a function $\boldsymbol{z}(\tau)$ [4, Chapter 5], $\tau = (t - L^-)/\varepsilon$ is the so-called *fast time*, and $\boldsymbol{z}(\tau)$ is a solution of the following initial value problem (cf. (A.4) in the Appendix):

$$\boldsymbol{z}' = \boldsymbol{B} + \boldsymbol{T}_{22}\frac{1}{1+e^{-\boldsymbol{z}}}, \qquad \tau > 0,$$
$$\boldsymbol{z}(0) = \frac{\boldsymbol{u}_2(L^-)}{\varepsilon},$$
$$\boldsymbol{B} = \boldsymbol{l}_2 + \boldsymbol{T}_{21}\,\boldsymbol{h}(\boldsymbol{x}_0(L^-)).$$

Here $'$ denotes $\frac{d}{d\tau}$. While this initial value problem can be solved in closed form (see [5]), it is cheaper to solve it numerically on a τ mesh, $\tau_j = j\,\Delta\tau$, $j = 0, 1, \ldots$. We follow this numerical solution until the transition is complete, say when all components of $\boldsymbol{z}$ exceed c in magnitude. We suppose this occurs at the N^{th} mesh point, τ_N. Thus the transition occurs on the τ interval $[0, \tau_N]$, or equivalently, on the t interval

$$I_t = [L^-,\ L^- + \varepsilon\, N\, \Delta\tau].$$

With $\boldsymbol{x}(L^-) = \boldsymbol{u}_1(L^-)$ being prescribed, $\boldsymbol{x}(t)$ is approximated to within $O(\varepsilon)$ by the solution of an initial value problem (cf. (A.2) (i)). In particular,

$$\dot{\boldsymbol{x}} + \boldsymbol{x} = \boldsymbol{l}_1 + \boldsymbol{T}_{11}\,\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{T}_{12}\,\boldsymbol{g}(\varepsilon\,\boldsymbol{z}), \qquad t \in I_t. \tag{5}$$

Solving this equation on the t mesh

$$t_i = L^- + i\,\Delta t, \qquad i = 0, \ldots, m,$$

we see that the values of $\boldsymbol{z}$ needed for the solution process of (5) are at the following values of τ:

$$\tau_{(j)} \equiv \frac{N}{m}\,j\,\Delta\tau, \qquad j = 0, \ldots, m;$$

m is the smallest integer exceeding $\varepsilon\, N\, \Delta\tau/\Delta t$. If m divides N, these values of τ lie on the τ mesh. If m does not divide N, we must interpolate. The very simplest interpolation is to choose the j^{th} τ needed as

$$\tau_{(j)} = \left[\frac{N}{m}\,j\right]\Delta\tau.$$

That is, we take the following approximations

$$\boldsymbol{z}(\tau_{(j)}) \sim \boldsymbol{z}\left(\left[\frac{N}{m}\,j\right]\Delta\tau\right) \sim \boldsymbol{z}\left(\left[\frac{j\,\Delta t}{\varepsilon\,\Delta\tau}\right]\Delta\tau\right), \qquad j = 0, \ldots, m.$$

The transition completed, we return to (2) to continue the solution development.

Why the Boundary Layer Method is Advantageous

The transition stage is a boundary layer region for $\boldsymbol{u}$. This region of rapid development requires small mesh increments in the numerical development (the τ mesh). The point is that the boundary layer development identifies that part of the system (in particular, $\boldsymbol{u}_2$) which undergoes the rapid variation in the boundary layer (namely $\boldsymbol{z}$) and invests in a fine mesh (the τ mesh) calculation only on that part. In the example in Section 3, there are 60 coarse mesh points and a total of 300 evaluations at such points. There are 500 fine mesh points and 550 corresponding evaluations. Without the boundary layer methodology, this number of evaluations would increase from 550 to 2500. Thus, the boundary layer methodology eliminates 1950 of 2800 evaluations, or 70%. This fraction eliminated should grow with the order of the system. (Note that all figures given here are approximate.)

3. A NUMERICAL EXPERIMENT

In this section, we illustrate the methods proposed in Section 2 by means of a computation performed on a special example.

EXAMPLE. We treat a fifth order system of the form (1), (3), (4), with

$$\boldsymbol{l} = -\begin{pmatrix} 1.5 \\ 1.5 \\ 1.5 \\ 0.5 \\ 0.5 \end{pmatrix} \quad \text{and} \quad \boldsymbol{T} = -\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

which is the standard k winners ARNN [6]. We take $\boldsymbol{u}(0) = (0.9, 1.5, 1.1, 1, 0.75)$. Other numerical values are $c = 3$, $\Delta t = 0.005$, $\Delta\tau = 0.025$, $\varepsilon = 0.01$. The results are illustrated in Figure 1. We see that there are 4 transitions. First the block of unknowns $(\boldsymbol{u}_1, \boldsymbol{u}_5)$ transits at $t = 0.135$ (approx.), then $\boldsymbol{u}_3$, $\boldsymbol{u}_4$, and $\boldsymbol{u}_2$, respectively, transit separately at $t = 0.175$, 0.195, 0.285 (approx.), respectively. As is evident from the figure, the trajectories are smooth throughout. The transition regions I_t are marked with blocks.

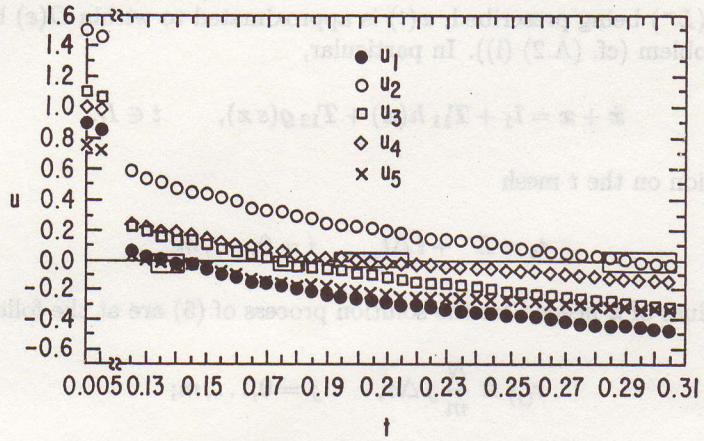

Figure 1. The numerical experiment illustrating transitions.

REFERENCES

1. S. Grossberg, Nonlinear neural networks: Principles, mechanisms, and architecture, *Neural Networks* **1**, 17–61 (1988).
2. J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, *Proceedings of the National Academy of Sciences, USA* **81**, 3088–3092 (May 1984).
3. E. Mjolsness, C. Garrett and W.L. Miranker, Multiscale optimization in neural nets, IBM T.J. Watson Research Center, RC 15910, (July 9, 1990).
4. W.L. Miranker, *Numerical Methods for Stiff Equations and Singular Perturbation Problems*, D. Reidel, Dordrecht, (1981).
5. W.L. Miranker and E. Mjolsness, Accelerating neural net dynamics by boundary layer methods, IBM T.J. Watson Research Center, RC 16962, (1991).
6. J.J. Hopfield and D.W. Tank, Neural computation of decisions in optimization problems, *Biological Cybernetics* **52**, 141–152 (1985).

APPENDIX

After scaling, equation (1) takes the form (we use the form (3) for $\boldsymbol{F}(\boldsymbol{v})$ with the block decomposition given in (4)),

$$\dot{\boldsymbol{x}} + \boldsymbol{x} = \boldsymbol{l}_1 + \boldsymbol{T}_{11}\,\boldsymbol{g}(\boldsymbol{x}, \varepsilon) + \boldsymbol{T}_{12}\,\boldsymbol{g}(\varepsilon\,\boldsymbol{y}, \varepsilon),$$
$$\varepsilon\,\dot{\boldsymbol{y}} + \boldsymbol{y} = \boldsymbol{l}_2 + \boldsymbol{T}_{21}\,\boldsymbol{g}(\boldsymbol{x}, \varepsilon) + \boldsymbol{T}_{22}\,\boldsymbol{g}(\varepsilon\,\boldsymbol{y}, \varepsilon).$$

We take $t = 0$ as the crossing point of the components y. That is, the value of t where the components in transition vanish, i.e., $y(0) = 0$. Now [4, Chapter 5] with $\tau = t/\varepsilon$ and $' = \frac{d}{d\tau}$,

$$\boldsymbol{x}(t) = \boldsymbol{x}_0(t) + O(\varepsilon),$$
$$\boldsymbol{y}(t) = \boldsymbol{y}_0(t) + \boldsymbol{Y}_0(\tau) + O(\varepsilon), \tag{A.1}$$

where

$$\text{(i)} \quad \dot{x}_0 + x_0 = l_1 + T_{11}\, h(x_0) + T_{12}\, \frac{1}{1+e^{-y_0}},$$

$$\text{(ii)} \quad 0 = l_2 + T_{21}\, h(x_0) + T_{22}\, \frac{1}{1+e^{-y_0}}, \tag{A.2}$$

$$\text{(iii)} \quad (y_0(0) + Y_0)' = l_2 + T_{21}\, h(x_0(0)) + T_{22}\, \frac{1}{1+e^{-(y_0(0)+Y_0)}},$$

with $x_0(0) = x(0)$ and $y_0(0) + Y(0) = 0$.

Combining (i) and (ii) in (A.2), we find

$$\dot{x}_0 + x_0 = l_1 + T_{11}\, h(x_0) - T_{12}\, T_{22}^{-1}\, (l_2 + T_{21}\, h(x_0)). \tag{A.3}$$

Since x_0 doesn't change sign in the transition (in a neighborhood of zero), the right member of (A.3) is time-independent in the transition. Denoting this right member by A, a constant, we have

$$x_0 = (x_0(0) - A)\, e^{-t} + A.$$

We also have

$$y_0 = -\log\left(-1 - \frac{1}{T_{22}^{-1}\, (l_2 + T_{21}\, h(x_0))}\right).$$

Note that y_0 is also a constant in the transition, and so, $y_0 = y_0(0)$, say. Let

$$z(\tau) = y_0 + Y_0(\tau),$$
$$B = l_2 + T_{21}\, h(x_0(0)).$$

Then from (A.2) (iii),

$$z' = B + T_{22}\, \frac{1}{1+e^{-z}}, \quad z(0) = 0. \tag{A.4}$$