ARTICLE Communicated by Ramamohan Paturi

Efficient Simplex-Like Methods for Equilibria of Nonsymmetric Analog Networks

Douglas A. Miller
Steven W. Zucker
Computer Vision and Robotics Laboratory, Research Centre for Intelligent Machines, McGill University, 3480 University Street, R. 410, Montréal, Canada H3A 2A7

What is the complexity of computing equilibria for physically implementable analog networks (Hopfield 1984; Sejnowski 1981) with arbitrary connectivity? We show that if the amplifiers are piecewise-linear, then such networks are instances of a game-theoretic model known as *polymatrix games*. In contrast with the usual gradient descent methods for symmetric networks, equilibria for polymatrix games may be computed by vertex pivoting algorithms similar to the simplex method for linear programming. Like the simplex method, these algorithms have characteristic low order polynomial behavior in virtually all practical cases, though not certain theoretical ones. While these algorithms cannot be applied to models requiring evolution from an initial point, they are applicable to "clamping" models whose input is expressed purely as a bias. Thus we have an a priori indication that such models are computationally tractable.

1 Introduction

A fundamental question is: Do biological or other physical systems solve problems that are NP-hard for Turing machines? Hopfield (1984) and Hopfield and Tank (1985) have provided evidence in the negative, suggesting that to the extent real or artificial neural systems *appear* to solve NP-hard problems (e.g., the traveling salesman problem), this is only illusory. Hopfield has taken the position that biological computation amounts to designing analog networks with appropriate asymptotically stable equilibria. What is really being solved are not NP-hard problems, but only much easier approximations, which amount to finding these equilibria. Thus, Hopfield seems implicitly to accept the Strong Church's Thesis of Vergis *et al.* (1986), which implies (accepting $P \neq NP$) that no analog computer (dynamical system) can solve NP-hard problems with less than exponential resources. (See Pour-El and Richards 1981 for an alternative though not contradictory view.)

A similar point of view has also been taken by Hummel and Zucker (1983) and Zucker *et al.* (1989) with regard to the solving of problems in

vision. Here there are instances, such as in the interpretation of line drawings (Kurousis and Papadimitriou 1988), where one might be tempted to think the brain is solving NP-hard problems, whereas what seems much more likely is that the brain finds only quick approximations.

However, even accepting this "stable state" view of biological computation, the question remains: How do we know that finding an asymptotically stable equilibrium of a dynamical system is computationally an easy problem in the Turing sense (e.g., polynomial)? This question seems especially important for nonsymmetric networks, where there is no descent function guaranteeing convergence. However, even in the symmetric case it is known to be NP-hard just to decide if a given point is a local minimum for a constrained quadratic! (Murty 1988, p. 170; Vergis *et al.* 1986).

In this paper we offer a partial answer to this question. We show that computing *an equilibrium* (not necessarily stable) for a Hopfield network with piecewise-linear amplifiers and arbitrary connectivity may be done with a type of vertex pivoting algorithm, *Lemke's algorithm*, which is very similar in its complexity to the simplex method for linear programming. The latter is strongly polynomial in practice although not necessarily so in theory. (An analysis of this phenomenon based on probability theory has become an outstanding mathematical question in recent decades, which has been partly answered — e.g., Adler *et al.* 1984.) Pivoting methods such as Lemke's algorithm would appear to be the only known algorithms for finding equilibria of nonsymmetric networks that are both guaranteed to work and have, at least in a probabilistic sense, polynomial complexity.

On the other hand Lemke's algorithm has two characteristics that sharply distinguish it from more traditional techniques of following integral curves through vector (usually gradient) fields. First, it has no sensitivity to initial conditions. Second, there is no guarantee it will produce a *stable* equilibrium. In many respects the algorithm could be expected to behave like a procedure that quickly picked an equilibrium at random.

These characteristics suggest a different approach to continuous dynamic models than has been taken so far in such applications as Hopfield and Tank's traveling salesman network and the vision relaxation network of Zucker *et al.* In these applications there are at all times an extremely large number of equilibria, and the evolution of the system is determined by the initial state. The supposition is that this initial state will evolve to a stable final state in its basin of attraction. There are, however, several possible problems with this kind of computation. First there is no guarantee, at least for a nonsymmetric network, than an attractive basin need exist (cf. Appendix A). Second, there is no guarantee that convergence, if it does occur, will be rapid, especially in numerical implementations. A well-known example of this kind of problem is zig-zagging behavior for steepest descent methods (Luenberger 1973). Third, an initial position

may be unstable in an especially bad way, by lying near the boundaries of a large number of different attractive basins, and thus requiring impractically large precision for a useful dynamic simulation. This appears to have been the case in the Wilson and Pawley (1988) simulations of the Hopfield and Tank traveling salesman network.

An alternative to this "initial position" view of computation is instead to express an input vector as a bias on some subset of the processing units, and then use Lemke's algorithm. If the bias is sufficiently large we get the kind of "clamping" described by Hinton and Sejnowski (1986) in the context of Boltzmann machines. The effect, for an appropriately designed or trained network, would be to eliminate the great mass of equilibria that exist in the unbiased state, leaving the system ideally with just one equilibrium state. The fact that we could use Lemke's algorithm would a priori indicate that the model was computationally tractable. Furthermore this kind of network computation would seem more consistent with the capabilities of low precision processing elements such as neurons, where it would appear difficult to specify accurately an initial position, or a consistent evolutionary path.

This bias/clamp approach raises the question, how much bias is necessary to constitute a clamp? Put another way, if c is a vector, δ a nonnegative scalar, and δc the bias, what is the *minimum* value of δ necessary for a subset of the processing units to be clamped into a given state? Indeed we may then ask how this minimum value would change with respect to changes in network connectivities resulting, say, from learning. These questions are similar in many respects to those that are efficiently handled in linear programming using *parametric sensitivity analysis* based on the simplex algorithm (Dantzig 1963). Our preliminary results suggest that Lemke's algorithm could provide the basis for a related kind of analysis for the networks considered here.

Of course, as a general procedure for computing equilibria, one may alternate between Lemke's algorithm and following integral curves, whichever is more appropriate. This approach would be analogous to the current situation in linear programming, where the simplex method provides an indispensable theoretical framework that may be supplemented by interior point methods such as Karmarkar's algorithm (Murty 1988).

We shall not concern ourselves directly with the earlier Hopfield (1982) model where the network is symmetric and the processors are binary valued. Hopfield computes equilibria for these networks with simple discrete descent methods not generally applicable to the later Hopfield (1984) model or to ours. Hopfield's (1982) problem fits into a very interesting class of *polynomial-time local search* (PLS) problems (Johnson *et al.* 1988) and in fact is known to be polynomially complete for this class (Papadimitriou *et al.* 1990). If we change this discrete problem by allowing the processors to assume a bounded range of real values, then it becomes one of those which we consider. However this *continuous* problem is easier since its solution set is always at least as large or

larger (cf. final example Appendix A). Thus the PLS-hardness results of Papadimitriou *et al.* would therefore not appear to apply to the kinds of continuous problems which we consider. We discuss the complexity of Lemke's algorithm further in Section 6.

2 Overview of Paper

This paper describes a correspondence between analog artificial neural networks similar to those described by Hopfield (1984) and the branch of mathematics known as game theory. An immediate result of this correspondence is the existence of a complementary vertex pivoting algorithm, known as *Lemke's algorithm*, similar in many respects to the well-known simplex algorithm for linear programming, which will compute an equilibrium for any instance of such an artificial neural network, regardless of interconnectivity, and in particular for cases where the interconnectivity is nonsymmetric. Such cases are in general avoided by Hopfield, yet are clearly of great interest for applications such as early visual processes (Zucker *et al.* 1989).

We will not be concerned with game theory as a whole, which originated with von Neumann and Morgenstern (1944), and for which the literature is now immense, but rather with a special branch known as *noncooperative n-person* games, and indeed a special branch of these known as *polymatrix* games.

To give an overview of this paper, in Section 3 we introduce the theory of polymatrix n-person games and noncooperative equilibrium strategies.

In Section 4 we describe a class of analog networks similar to those described in Hopfield (1984), the differences being (1) that our amplifiers are linear over their specified operating range, rather than asymptotic sigmoid over the real line, (2) that we make specific assumptions of lower and upper bounds on the voltages that our amplifier inputs can attain, and (3) that the interconnectivity ("synapses") of our amplifiers need not be symmetric. The first two modifications allow us to view these networks as polymatrix n-person games, each player representing a neuron, and each player/neuron having the two game strategies "depolarize" and "hyperpolarize." The third modification is a bonus, since polymatrix games do not require symmetry.

In Section 5 we show that the bounded linear voltage amplifiers of the previous section may be replaced with bounded piecewise linear voltage amplifiers, staying within the same model, by simply adding more linear amplifiers, approximately one per linear segment per amplifier. We argue that only a small number of linear segments may be necessary for biologically plausible response curves.

In Section 6 we discuss the complexity of computing equilibria for polymatrix games and hence for our analog networks. We state as a proposition the main result of the paper, that Lemke's algorithm will

compute an equilibrium for any bounded piecewise-linear Hopfield network.

While purely analytic results on the probabilistic efficiency of Lemke's algorithm (Todd 1983) are only suggestive, Lemke's algorithm is known *in practice* to be of low order polynomial complexity, its computational complexity being essentially that of the well-known simplex method for linear programming (Murty 1988). Thus the applicability of Lemke's algorithm to computing equilibria for nonsymmetric Hopfield networks strongly suggests such computations are tractable, and in particular, not NP-complete (Garey and Johnson 1979). This in turn reinforces our belief that such models may be able to capture important properties of biological computation. Indeed, as we have noted (Miller and Zucker 1991), polymatrix games with zero self-payoff terms are equivalent to *relaxation labeling* (Hummel and Zucker 1983), which has already been applied to modeling early visual systems (e.g., Zucker *et al.* 1989).

In Appendices A and B we compare two quite different methods for computing equilibria, which we refer to as *primal* and *dual*. The primal method (Appendix A) amounts to following the integral curves of the dynamical system defining the network. The dual method (Appendix B) is Lemke's algorithm, which we describe here in detail.

3 Polymatrix Games

An *n-person game* (Nash 1951) is a set of *n players*, each with a set of *m pure strategies*. Player i has a real-valued payoff function $s_i(\lambda_1, \ldots, \lambda_n)$ of the pure strategies $\lambda_1, \ldots, \lambda_n$ chosen by the n players. For each player there is an additional kind of strategy called *mixed*, which is a probability distribution on the player's m pure strategies. A player i's payoff for a mixed strategy is the expected value of i's pure strategy payoff given all players choose according to their mixed strategies. Notice that as with pure strategy payoffs, mixed strategy payoffs are only meaningful in terms of all players' simultaneous actions. A *noncooperative* or *Nash equilibrium* is a collection of mixed strategies for each player such that no player can receive a larger expected payoff by changing his/her mixed strategy given the other players stick to their mixed strategies. Nash showed using the Brouwer fixed point theorem that such equilibria always exist. However they need not be stable, as we shall discuss in Appendix A.

A *polymatrix game* is an n-person game in which each payoff to each player i in pure strategy is of the form

$$s_i(\lambda_1, \ldots, \lambda_n) = \sum_j r_{ij}(\lambda_i, \lambda_j)$$

where for all i, $r_{ii}(\lambda_i, \hat{\lambda}_i) = 0$. (Here we use $\hat{\lambda}_i$ to denote a strategy for i that is possibly different than λ_i.) We may interpret $r_{ij}(\lambda_i, \lambda_j)$ as i's payoff from j given their respective pure strategies λ_i and λ_j. This implies a

payoff to i in mixed strategies of the form

$$\sum_{\lambda_i, j, \lambda_j} p_i(\lambda_i)\, r_{ij}(\lambda_i, \lambda_j) p_j(\lambda_j) \tag{3.1}$$

where $p_i(\lambda_i)$ is the probability player i chooses strategy λ_i.

The quadratic form of 3.1 suggests a somewhat more general definition of polymatrix games due to Eaves (1973), in which for each player i we allow the $m \times m$ matrix corresponding to the r_{ii} terms in 3.1, instead of being zero, to be a symmetric negative semidefinite matrix $\frac{1}{2}\left[r_{ii}(\lambda_i, \hat{\lambda}_i)\right]$. In effect we include a penalty function in i's payoff 3.1 that is a convex quadratic function of i's mixed strategy. Henceforth we shall use this more general definition. Using a version of the Brouwer fixed point theorem (Kinderlehrer and Stampacchia 1980) one can show Nash equilibria always exist for these more general games.

Implicit in our definition of polymatrix games is the possibility of including a constant payoff term $c_i(\lambda_i)$ for each strategy λ_i of each player i, regardless of the other players' choices. [To see this, choose some other player j, and add $c_i(\lambda_i)$ to $r_{ij}(\lambda_i, \lambda_j)$ for each λ_j.] However, it will be more convenient to make the $c_i(\lambda_i)$ explicit, since these will correspond to Hopfield's bias terms. Therefore our payoff 3.1 to i shall henceforth be of the more general form

$$\begin{aligned} \frac{1}{2} \sum_{\lambda_i, \hat{\lambda}_i} p_i(\lambda_i) r_{ii}(\lambda_i, \hat{\lambda}_i) p_i(\hat{\lambda}_i) \;+\; & \sum_{\lambda_i,\, \lambda_j, j \neq i} p_i(\lambda_i)\, r_{ij}(\lambda_i, \lambda_j) p_j(\lambda_j) \\ & + \sum_{\lambda_i} c_i(\lambda_i) p_i(\lambda_i) \end{aligned} \tag{3.2}$$

At this point it will be useful to add some notation. Observe we may completely specify the elements of a polymatrix game with an $mn \times mn$ payoff matrix R and an mn bias vector c given by

$$R = \begin{bmatrix} \left[r_{11}(\lambda_1, \hat{\lambda}_1)\right] & \cdots & \left[r_{1n}(\lambda_1, \lambda_n)\right] \\ \vdots & \ddots & \vdots \\ \left[r_{n1}(\lambda_n, \lambda_1)\right] & \cdots & \left[r_{nn}(\lambda_n, \hat{\lambda}_n)\right] \end{bmatrix} \qquad c = \begin{bmatrix} [c_1(\lambda_1)] \\ \vdots \\ [c_n(\lambda_n)] \end{bmatrix} \tag{3.3}$$

The matrix R corresponds to the object/label *consistency matrix* in relaxation labeling (Hummel and Zucker 1983). For each player i it will also be convenient to refer to $[c_i(\lambda_i)]$ as c_i, to i's vector of mixed strategies $[p_i(\lambda_i)]$ as p_i, and to the vector of p_i's as p. It will also be useful to have a *direction vector* $d = p^1 - p^2$, for two strategy vectors p^1, p^2. Similiarly $d_i = p_i^1 - p_i^2$. If we let A be the $n \times mn$ matrix

$$\begin{bmatrix} -1 \ldots -1 & \cdots & 0 \ldots 0 \\ \vdots & \ddots & \vdots \\ 0 \ldots & \cdots & -1 \ldots -1 \end{bmatrix}$$

and let $q^\top$ be the n-vector $(-1,\ldots,-1)$, then p is a vector of all players' mixed strategies if and only if

$$Ap = q, \qquad p \geq 0 \tag{3.4}$$

Also in terms of this notation we may express the gradient of 3.2 as

$$\sum_{j=1}^{n} \left[r_{ij}(\lambda_i, \hat{\lambda}_j)\right] p_j + c_i \tag{3.5}$$

Assume p satisfies 3.4 and is fixed except for player i, whose payoff is given by 3.2. Since this function is concave, and the constraint set (a simplex) is convex, a given mixed strategy for i will have a maximum payoff if and only if i's gradient 3.5 has a vanishing projection onto the constraint set 3.4. Equivalently, p_i is an optimal strategy for player i if and only if there exists no directional vector d such that $d_j = 0$ for $j \neq i$, and

$$\begin{aligned} d_i^\top \left(\sum_{j=1}^{n} \left[r_{ij}(\lambda_i, \hat{\lambda}_j)\right] p_j + c_i \right) > 0 \\ A_{i\cdot} d = 0 \\ \text{for all } \lambda_i, \quad p_i(\lambda_i) = 0 \text{ implies } d_i(\lambda_i) \geq 0 \end{aligned} \tag{3.6}$$

Now let all players be free to change their strategies. For p to be a Nash equilibrium, 3.6 must be simultaneously nonsatisfiable for each player i. It can be shown (e.g., Miller and Zucker 1991) that this set of simultaneous conditions is equivalent to there being no d satisfying the system

$$\begin{aligned} d^\top (Rp + c) > 0 \\ Ad = 0 \\ \text{for all } i, \lambda_i, \quad p_i(\lambda_i) = 0 \text{ implies } d_i(\lambda_i) \geq 0 \\ Ap = q, \quad p \geq 0 \end{aligned} \tag{3.7}$$

In view of 3.7 we now have an alternative characterization of the Nash equilibria of the polymatrix game 3.3 in terms of the equilibria of the dynamical system

$$\begin{aligned} p' = Rp + c \\ Ap = q, \quad p \geq 0 \end{aligned} \tag{3.8}$$

In other words, these equilibria are precisely the points at which the vector field of 3.8 vanishes. Notice if R is symmetric then p' is the gradient of

$$\frac{1}{2} p^\top R p + c^\top p \tag{3.9}$$

The first term in 3.9 corresponds to the *average local potential* in relaxation labeling (Hummel and Zucker 1983).

4 Analog Networks as Polymatrix Games

We take as our point of departure the class of analog networks defined by Hopfield (1984). These are dynamical systems defined by the equations

$$\begin{aligned} C_i \frac{du_i}{dt} &= \sum_{j \neq i} T_{ij} V_j - u_i/R_i + I_i \\ u_i &= g_i^{-1}(V_i) \end{aligned} \tag{4.1}$$

for $i = 1, \ldots, n$. Here u_i and V_i are interpreted as the input and output voltages of an instantaneous amplifier described by a continuous monotonic sigmoid function $g_i(u_i)$. In addition we define $|T_{ij}|$ as the conductance between the output of amplifier j and the input of amplifier i, we let C_i be the input capacitance of i, we let I_i be a fixed input bias current for i, and we define R_i by

$$1/R_i = 1/\rho_i + \sum_{j \neq i} |T_{ij}| \tag{4.2}$$

where ρ_i is the resistance across C_i. If T_{ij} is negative, then the input to amplifier i comes from an inverting amplifier $-g_j(u_j)$. Such a network is illustrated in Figure 1.

Suppose now g_i is a linear function on the real interval $[\alpha_i, \beta_i]$, $\alpha_i < 0 < \beta_i$, such that $g_i(\alpha_i) = 0$, $g_i(\beta_i) = 1$, and that α_i, β_i are also upper and lower bounds on the voltage that the input capacitor to amplifier i can attain. Thus a further input current to a saturated capacitor would produce no effect. (We shall show in the next section that this model actually includes *piecewise-linear* voltage amplifiers as well.)

Letting $\delta_i = (\beta_i - \alpha_i)$ be a unitless scalar these assumptions give us a new version of 4.1:

$$\begin{aligned} C_i \frac{du_i}{dt} &= \sum_{j \neq i} T_{ij} V_j - u_i/R_i + I_i \\ u_i &= \delta_i V_i + \alpha_i \\ \alpha_i &\leq u_i \leq \beta_i \end{aligned} \tag{4.3}$$

Rewriting this in terms of the output voltages V_i and dividing through by $\delta_i C_i$ we obtain

$$\begin{aligned} \frac{dV_i}{dt} &= \sum_{j \neq i} T_{ij} V_j/\delta_i C_i - V_i/R_i C_i + (-\alpha_i/R_i + I_i)/\delta_i C_i, \\ & 0 \leq V_i \leq 1 \end{aligned} \tag{4.4}$$

Notice the amplifier gain $1/\delta_i$ is inversely related to the influence of the capacitance term $-V_i/R_i C_i$.

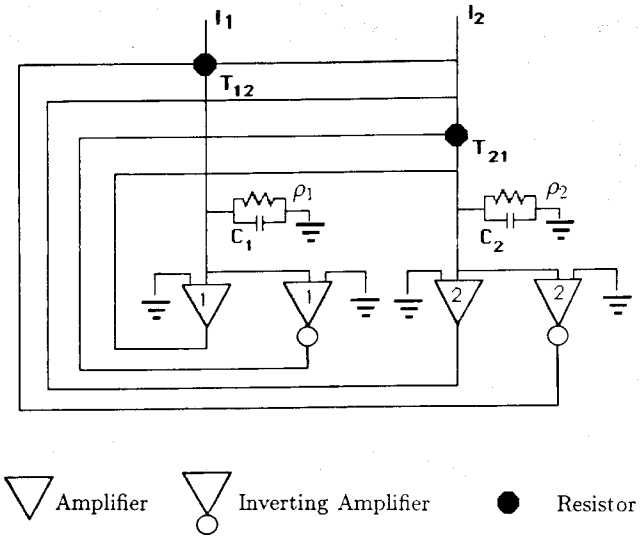

Figure 1: A two-node analog network. Each node or "neuron" i includes a noninverting and inverting voltage amplifier and a capacitor C_i with a parallel "membrane" resistor. Each other node j may connect ("synapse") onto i via a resistance $1/|T_{ij}|$ from j's noninverting or inverting amplifier, respectively, depending on whether T_{ij} is positive or negative. Node i may also have a constant input current I_i. In this example $T_{1,2}$ and $T_{2,1}$ are negative, so that the two nodes are mutually suppressing. (Adapted from Hopfield and Tank 1985.)

If the T_{ij} are symmetric, Hopfield (1984, p. 3090) gives a function for the dynamical system (4.1) of the form

$$-1/2\sum_i\sum_{j\neq i} T_{ij}V_iV_j + \sum_i 1/R_i \int_0^{V_i} g_i^{-1}(V)dV - \sum_i I_iV_i$$

which strictly decreases with the time evolution of the system unless an equilibrium is reached, thus showing the system cannot cycle.

Defining g_i^{-1} as in 4.3 and dividing through by δ_iC_i gives us

$$-1/2\sum_i\sum_{j\neq i} T_{ij}V_iV_j/\delta_iC_i + 1/2\sum_i V_i^2/R_iC_i - \sum_i(-\alpha_i/R_i + I_i)V_i/\delta_iC_i \quad (4.5)$$

which is actually a *potential* function for 4.4, that is, its negative gradient projected onto the constraint set of 4.4 is the vector field of 4.4.

In fact, 4.5 is really just an instance of 3.9, for at this point it is trivial to show 4 equivalent to a dynamical system of the form 3.8, that is,

to a polymatrix game. The idea is, first, to associate amplifiers with players, and then, for each i, to let player i have exactly two strategies d ("depolarize") and h ("hyperpolarize"), and to associate $p_i(d)$ with V_i.

To do this, let

$$\begin{aligned} r_{ij}(d,d) &= 2T_{ij}/\delta_i C_i \\ r_{ii}(d,d) &= -2/R_i C_i \\ c_i(d) &= 2[(-\alpha_i/R_i) + I_i]/\delta_i C_i \end{aligned} \tag{4.6}$$

and all other entries of R and c be zero. Using the first line of 3.8 we find

$$\begin{bmatrix} p_i(d)' \\ p_i(h)' \end{bmatrix} = \tag{4.7}$$
$$\begin{bmatrix} 2\left(\sum_{j\neq i}(T_{ij}/\delta_i C_i)p_j(d) - (1/R_i C_i)p_i(d) + [(-\alpha_i/R_i) + I_i]/\delta_i C_i\right) \\ 0 \end{bmatrix}$$

We may then compute the magnitude $\pi_i(d)'$ of the projection of 4.7 onto the subspace $p_i(d) + p_i(h) = 1$ in the $p_i(d)$ direction by taking the dot product of 4.7 with $(1/\sqrt{2}, -1/\sqrt{2})$, obtaining

$$\pi_i(d)' = \sqrt{2}\left\{\sum_{j\neq i}(T_{ij}/\delta_i C_i)p_j(d) - (1/R_i C_i)p_i(d) + [(-\alpha_i/R_i) + I_i]/\delta_i C_i\right\}$$

Using the simple geometrical relation $(d/dt)p_i(d) = \pi_i(h)'/\sqrt{2}$, we see that each $(d/dt)p_i(d)$ satisfies precisely the same equation as dV_i/dt in 4. Thus we have an instance of a polymatrix game of the form 3.8 which is equivalent to 4.

5 Extension to Piecewise-Linear Amplifiers

In this section we show that the bounded linear amplifiers described previously can, within the same model, be used to construct, to an arbitrary degree of accuracy, *any* piecewise-linear amplifier with bounded input.

To consider the simplest case, suppose we want a voltage amplifier $\hat{g}_i$ (Fig. 2) whose input $\hat{u}_i$ is bounded between α_i and β_i, and which linearly maps the interval $[\bar{\alpha}_i, \bar{\beta}_i]$ onto $[0,1]$, where $\alpha_i < \bar{\alpha}_i < \bar{\beta}_i < \beta_i$. We can construct $\hat{g}_i$ with two bounded linear amplifiers

$$\begin{aligned} &g_i : [\alpha_i, \beta_i] \to [0,1] \\ &g_{\bar{i}} : [g_i(\bar{\alpha}_i)/2, g_i(\bar{\beta}_i)/2] \to [0,1] \end{aligned} \tag{5.1}$$

using the circuit in Figure 3. With respect to Figure 3, let ρ_i and C_i have the desired values for the input to amplifier $\hat{g}_i$, and let $T_{\bar{i},i} = \rho_{\bar{i}} = 1$.

Observe ρ_i and C_i act as a low-pass frequency filter for amplifier i. It follows that if we choose $C_{\bar{i}}$ sufficiently small with respect to C_i, we may neglect its impedance, treating the output of i as going instantaneously

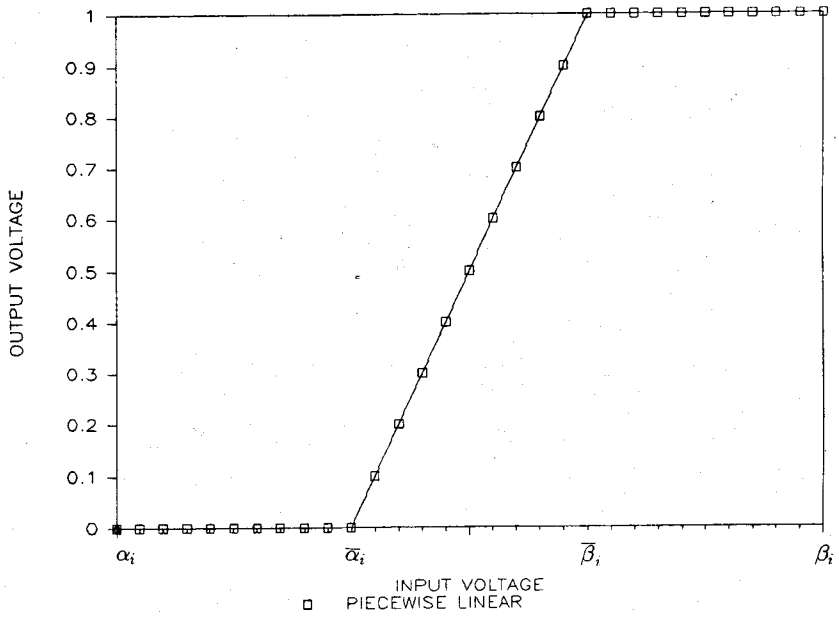

Figure 2: Piecewise-linear amplifier $\hat{g}_i$ with one nonzero slope.

through a pure voltage divider. In that case

$$V_i(t)T_{\bar{i},i} - u_{\bar{i}}(t)(1/\rho_{\bar{i}} + T_{\bar{i},i}) = 0 \tag{5.2}$$

and hence

$$u_{\bar{i}}(t) = V_i(t)/2 \tag{5.3}$$

It follows from 5.1 and 5.3 that in response to any input voltage to amplifier g_i, $g_{\bar{i}}$ and hence $\hat{g}_i$ will have (as $C_{\bar{i}}$ goes to zero) the output given in Figure 2.

This procedure may be extended to more complicated piecewise-linear amplifiers $\hat{g}_i$ such as in Figure 4, where we have two distinct nonzero slopes. Here the lower and upper bounds are α_i and β_i, the first nonlinearity occurs at $\alpha_{i(1)}$, the second at $\beta_{i(1)}$ (which we also label $\alpha_{i(2)}$), and the third at $\beta_{i(2)}$. To create $\hat{g}_i$, we may use the circuit in Figure 5. As before, ρ_i and C_i have the desired values associated with the input to $\hat{g}_i$. Other values are

$$\begin{aligned} \rho_{i(1)} &= \rho_{i(2)} = \rho_{\bar{i}} = 1 \\ T_{i(1),i} &= T_{i(2),i} = 1 \end{aligned}$$

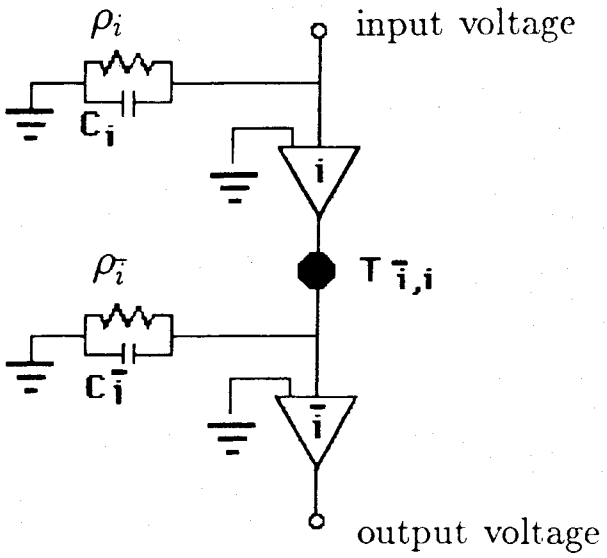

Figure 3: Circuit giving piecewise-linear response of Figure 2. Note amplifiers g_i and $g_{\bar{i}}$ are same type as in Figure 1.

and

$$\begin{aligned} T_{\bar{i},i(1)} &= 2\mu \\ T_{\bar{i},i(2)} &= 2(1-\mu) \end{aligned} \tag{5.4}$$

where $\mu = \hat{g}_i(\beta_{i(1)}) = \hat{g}_i(\alpha_{i(2)})$ will be used as a weighting factor between the two nonzero slopes. Further let

$$\begin{aligned} &g_i : [\alpha_i, \beta_i] \to [0,1] \\ &g_{i(1)} : [g_i[\bar{\alpha}_i(1)]/2, g_i[\bar{\beta}_i(1)]/2] \to [0,1] \\ &g_{i(2)} : [g_i[\bar{\alpha}_i(2)]/2, g_i[\bar{\beta}_i(2)]/2] \to [0,1] \\ &g_{\bar{i}} : [0, 2/3] \to [0,1] \end{aligned} \tag{5.5}$$

If all other capacitors are small relative to C_i, then as before we have in the limit

$$u_{i(1)}(t) = V_i(t)/2, \qquad u_{i(2)}(t) = V_i(t)/2 \tag{5.6}$$

and in addition

$$V_{i(1)}(t)T_{\bar{i},i(1)} + V_{i(2)}(t)T_{\bar{i},i(2)} - u_{\bar{i}}(t)(1/\rho_{\bar{i}} + T_{\bar{i},i(1)} + T_{\bar{i},i(2)}) = 0$$

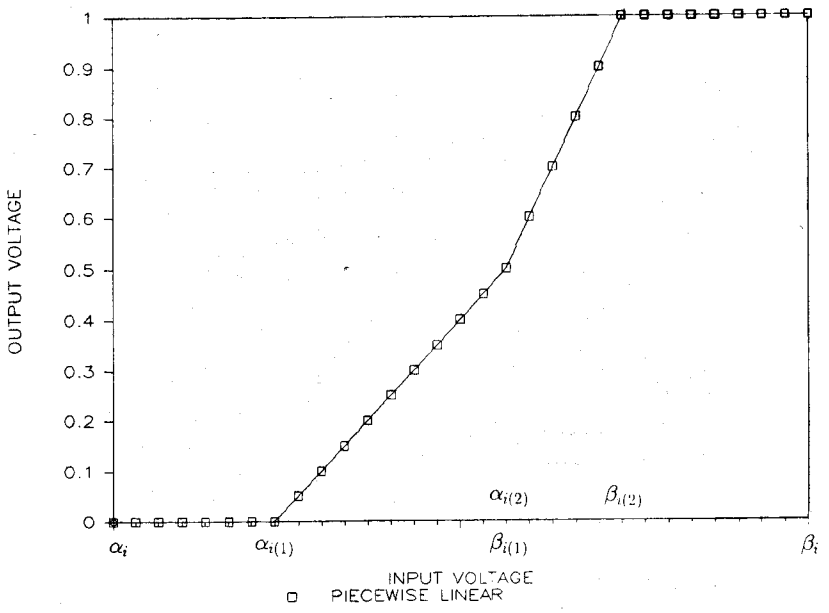

Figure 4: A piecewise-linear amplifier $\hat{g}_i$ with two nonzero slopes.

hence

$$u_i(t) = [V_{i(1)}(t) + V_{i(2)}(t)]/3 \tag{5.7}$$

It follows from 5.4–5.7 that $\hat{g}_i$ will have the desired form. Notice that 5.5 uniquely determines the abscissa values of the nonlinear points of Figure 4, and that 5.4 uniquely determines the ordinate value for the transition from the first nonzero slope to the second.

The reader may verify that in general this procedure can be extended to n nonzero slopes using $n+2$ bounded linear amplifiers.

Although there is a complexity cost in this procedure (see the next section and Appendix B), the low precision of individual neurons as processing units implies that piecewise-linear approximations with just a few linear segments are often likely to suffice, in which case the extra computational cost would be minimal. For instance in Figure 6 we compare a piecewise-linear response curve with a smooth asymptotic sigmoid $g(u) = \gamma/[1 + \exp(-u/\lambda)]$. If we assume (cf. Hopfield 1984) that the smooth sigmoid represents the mean firing rate of a spiking neuron for a 50 msec interval, that the neuron's upper firing rate is 200/sec, and that the number of firings in a time interval has a Poisson distribution,

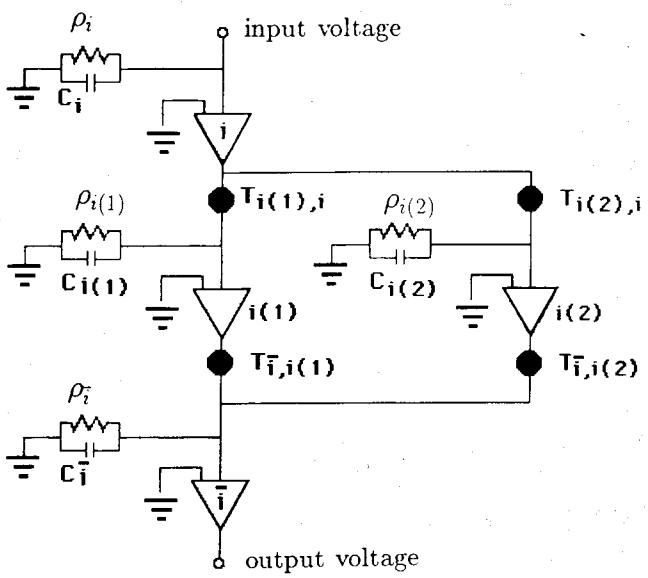

Figure 5: Circuit giving piecewise-linear response of Figure 4. In general n nonzero slopes require $n+2$ bounded linear amplifiers.

then the upper and lower curves give the corresponding values of the asymptotic sigmoid mean plus and minus a standard deviation. It seems hard to imagine, within the kind of time periods in which neural systems compute (e.g., a few hundred msec), that one could distinguish between the dynamical behavior of a system composed of one or the other kind of low-precision amplifier.

6 Complexity of Computing Equilibria

In the previous sections we have reduced the problem of finding an equilibrium for a general analog network with bounded piecewise-linear amplifiers to that of finding an equilibrium of a polymatrix game. We therefore now address the question of how we can find such an equilibrium, and do so in a computationally efficient manner.

Certainly the question is far from trivial since, as is shown in Appendix A, a primal approach such as a generalized gradient descent technique that simply follows integral curves, while potentially useful, can fail badly even in very simple nonsymmetric cases.

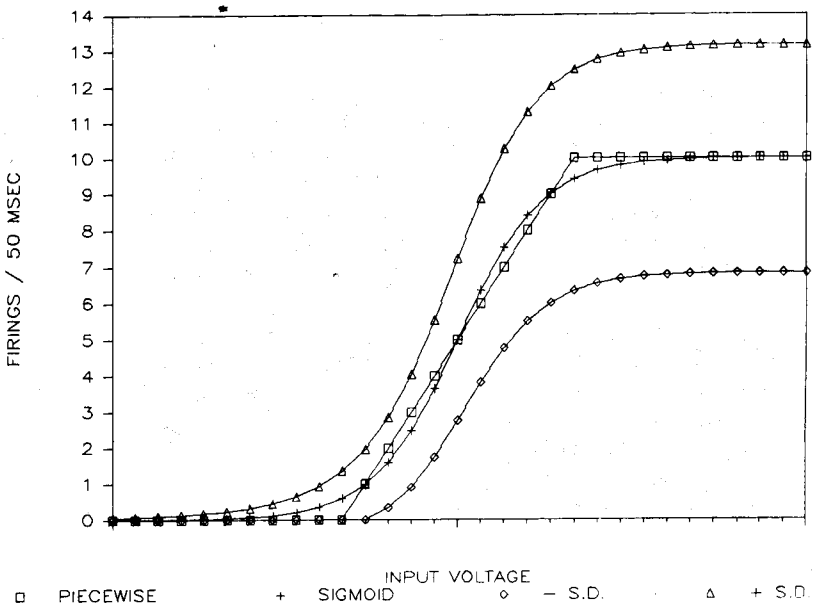

Figure 6: A biological frame of reference for comparing a smooth and a piecewise-linear response curve. We assume (cf. Hopfield 1984) that the sigmoid represents the mean firing rate of a spiking neuron for a 50 msec interval, that the upper firing rate is 200/sec, and that the number of firings in a time interval has a Poisson distribution. The upper and lower curves give the corresponding values of the asymptotic sigmoid mean plus and minus a standard deviation. Thus the piecewise linear curve is well within the likely statistical behavior of a neuron with the given asymptotic sigmoid mean firing rate.

In this section we describe an alternate view of this problem, a *dual* approach, to borrow from mathematical programming terminology, which will provide us with an algorithm for computing an equilibrium for *any* problem instance, in a time complexity which, while not deterministically polynomial, appears to be polynomial at least in a very strong probabilistic sense.

Let us first redefine R and c by adding a sufficiently small negative constant k to each term so that

$$R < 0, \qquad c < 0 \tag{6.1}$$

Notice this does not alter the Nash equilibria, since each player receives an identical penalty $(n+1)k$ regardless of strategy. On the other hand,

6.1 implies each player's payoff gradient 3.5 is negative. This permits us to relax 3.4 and replace it with

$$Ap \leq q, \qquad p \geq 0 \tag{6.2}$$

(To see this, observe that for some i, letting $A_{i\cdot}$ be the ith row of A, if $A_{i\cdot}p^1 < q_i$, then for some $\epsilon > 0$, $p_i^2 = (1-\epsilon)p_i^1$ is a feasible preferred strategy for i given the other players' strategies remain unchanged. Thus all equilibria must still satisfy 3.4.)

We now describe our principal analytic tool, a variant of the well-known theorem of Kuhn and Tucker (1951). The idea is to replace the requirement of the nonexistence of d in 3.7 with the requirement of the existence of a pair of vectors of *dual variables* or *Kuhn–Tucker multipliers* y, u. The n multipliers y correspond to the n constraints $Ap \leq q$, and the mn multipliers u correspond to the mn constraints $p \geq 0$.

Now it can be shown (e.g., Miller and Zucker 1991) that finding p and v satisfying our original equilibrium conditions 3.7 is equivalent to finding p, y, u, v that satisfy the system

$$\begin{bmatrix} R & -A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} I_{mn} & 0 \\ 0 & I_n \end{bmatrix} \begin{bmatrix} p \\ y \\ u \\ v \end{bmatrix} = \begin{bmatrix} -c \\ q \end{bmatrix}$$
$$p, y, u, v \geq 0 \tag{6.3}$$
$$p^\top u + y^\top v = 0$$

where v is an n-vector, I_{mn} and I_n are identity matrices of size mn and n, and R, p, c, q are as in Section 3.

We refer to v as a vector of *slack* variables for the constraints 6.2. This is because 6.3 implies

$$Ap + v = q, \qquad p, v \geq 0 \tag{6.4}$$

which is of course equivalent to 6.2. Similarly p may be viewed as a set of slack variables for the n constraints $p \geq 0$. The third line of 6.3 may then be interpreted as stating that if a slack variable is positive, its Kuhn–Tucker multiplier is zero.

The first two lines of 6.3 represent a set of linear equalities and linear inequalities of the kind found in the well-known *linear programming* problem (e.g., Dantzig 1963). Although solving such problems is far from trivial, they are known to be of polynomial computational complexity in terms of the problem specification size, as opposed NP-complete problems, which are for all practical purposes of exponential complexity (Garey and Johnson 1979).

The situation changes dramatically, however, when we consider the third line of 6.3. The problem then becomes an instance of the *linear complementarity problem*, which in general *is* NP-complete (Garey and Johnson

1979). Furthermore, although the theory of NP-completeness applies to digital computation (specifically, Turing machines) the intuitively attractive thesis has been proposed that any analog machine (such as Hopfield and Tank 1985) for solving NP-complete problems would necessarily consume exorbitantly large physical resources (Vergis *et al.* 1986). (Note Hopfield and Tank do not actually claim to *solve* the NP-complete traveling salesman problem with an analog device, but merely to find approximate solutions.) Thus it seems very difficult to accept the idea that any biologically plausible system could be based on solving an NP-complete problem. In particular, if finding an equilibrium for the kinds of polymatrix games we are interested in is an NP-complete problem, it is hard to imagine such a model could be biologically useful. (See also Kirousis and Papadimitriou 1988 and Tsotsos 1988.)

There is a possible way out of this impasse, in that 6.3 has a special structure, and linear complementarity problems with special structure may still be solvable in polynomial time.

For instance if R is negative semidefinite, then 6.3 may be solved in polynomial time by an algorithm similar to the ellipsoid algorithm for linear programming (Adler *et al.* 1980). This class of Rs has two important subclasses. The first consists of those R that are also symmetric, in which case solving 6.3 amounts to solving a convex quadratic program. The second subclass consists of those R for which the diagonal submatrices $\left[r_{ii}(\lambda_i, \hat{\lambda}_i)\right]$ are zero for each i, and for which $R = -R^\top$. Such an R is trivially negative semidefinite, and defines a *zero sum* polymatrix game. It follows that equilibria for such games may be computed in polynomial time.

To the authors' knowledge, the complexity of computing general polymatrix game equilibria, that is, linear complementarity problems of the form 6.3, is an open question. There are, however, results that at least make it appear unlikely this problem is NP-complete. In particular, 6.3 may be solved by an algorithm belonging to a family of vertex pivoting algorithms (the most well-known member being the simplex method for linear programming) that tend to be extremely fast in practice but that can, in certain artificial cases, require exponential time (Murty 1988, p. 162; Cottle 1980).

This algorithm, known as *Lemke's algorithm*, is described in detail in Appendix B. It was not originally used for polymatrix games, and the fact that it could be was first recognized by Eaves (1973), although in a different form than 6.3 (see also Appendix B for a discussion of Eaves' method).

Independently of Eaves' result, we showed Lemke's algorithm could also be applied in a form known as *copositive-plus*, of which 6.3 together with the assumption 6.1 is an example (Miller and Zucker 1991).

Given that we may use Lemke's algorithm for polymatrix game equilibria, if we extrapolate from linear programming, for which true polynomial algorithms were found after decades of experience with the simplex

method, the latter invariably polynomial in practice, then it seems reasonable to hope that true polynomial algorithms exist for polynomial game equilibria as well.

Moreover, from the point of view of the present paper, what is significant is that *in practice* 6.3 is solvable in polynomial time, just as with NP-complete problems it is only known that *in practice* they are not solvable in polynomial time. [To know the latter with certainty would mean solving the famous "$P = NP$?" problem (Garey and Johnson 1979).]

Indeed the above probabilistic view of the efficiency of Lemke's algorithm has been given a rigorous form. Todd (1983) has shown, under an extremely broad class of joint probability distributions on the numerical components of the linear complementarity problem, that the expected number of pivots (see Appendix B) of a particular form of Lemke's algorithm is bounded above by $N(N+1)/2$, where N is the number of linear equations, and hence the total expected computation is polynomial. While problems of the form 6.3 can only be regarded as a subpopulation of a population of problems for which this result holds, still Todd's analysis is highly encouraging.

We conclude this section with two propositions. First, since we have reduced our modified Hopfield network (4.3) to a two-strategy polymatrix game (3.8) we may state

Proposition 1. *The complexity of computing an equilibrium for an analog network with piecewise-linear amplifiers is bounded by the complexity of computing an equilibrium for a polymatrix game.*

Finally, since Lemke's algorithm solves the linear complementarity problem 6.3, which is equivalent to finding an equilibrium for 3.8, we conclude

Proposition 2. *Lemke's algorithm computes an equilibrium for an analog network with piecewise-linear amplifiers, for arbitrary interconnections, amplifier gains, and input biases.*

7 Appendix A: Computing Equilibria with Integral Curves

In this appendix we shall discuss a *primal* algorithm for computing polymatrix game equilibria, that is, an algorithm which attempts to solve 3.8 directly.

The primal method we shall be concerned with is in a sense the most obvious method, and the only one plausible biologically. It amounts to following the integral curves of the dynamical system 3.8 defining the network (cf. the *relaxation labeling algorithm* of Hummel and Zucker 1983). Although Hopfield and Tank (1985) have built physical devices to imitate

such systems, this algorithm can of course be attempted numerically, for instance by a piecewise smooth version of Euler's method, each "piece" corresponding to a new face of the polytope constraint set. If R is symmetric this is similar to the *projected gradient* method of mathematical programming (Luenberger 1973, p. 247).

Specifically, given a point p^k, the algorithm may be described as follows: Iteratively compute p^{k+1} by letting $\bar{d}^k + p^k$ be the projection of $Rp^k + c$ onto the constraint set $Ap = q$, and letting $d^k + p^k$ be the projection of $Rp^k + c$ onto $Ap = q$ together with all constraints $p_i = 0$ for which both $p^k = 0$ and $\bar{d}^k < 0$. Then let $p^{k+1} = p^k + \alpha^k d^k$ for some scalar $\alpha^k > 0$. If $d^k = 0$, the algorithm terminates and p^k is an equilibrium. There are many critical issues in implementing this method, such as the choice of α^k, the manner of computing the projection (trivial in the two-state case), and the choice of a stopping rule, since in general $d^k = 0$ will be a numerical impossibility, unless, as is common in optimization procedures, we use the proximity of the p^k to a particular vertex to correctly guess that that vertex is an equilibrium.

As Luenberger (1973, p. 251) notes, a major source of potential difficulty with the primal algorithm is the discontinuous behavior of the vector field on the polytope boundary, although he also states this is not a problem in practice for the projected gradient method. However, from a theoretical standpoint the absence of a global Lipschitz condition

$$||\pi(p^1) - \pi(p^2)|| \leq \lambda ||p^1 - p^2||$$

where $\pi(p)$ is the projected vector field at p and $\lambda \geq 0$ a constant, makes it difficult to say much about the complexity of the primal method, except for a given smooth segment belonging to the interior of a given polytope face. For such a segment $\phi(t)$, $t_0 \leq t \leq t_1$ (where the Lipschitz condition does hold) we can show (cf. Vergis *et al.* 1986, p. 108) that an approximation to the actual curve can be computed to any accuracy $\epsilon > 0$ with a number of steps which is polynomial in $1/\epsilon$. However, at least with Euler's method, the number of steps required for a given accuracy ϵ may be exponential in $(t_1 - t_0)$.

If R is not symmetric there is perhaps a deeper problem than discontinuity and numerical approximation, namely the question of convergence of the integral curve itself. Consider for example the zero sum game given by

$$R = \begin{bmatrix} [0] & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} & [0] \end{bmatrix}, \qquad c = [0] \tag{7.1}$$

We can compute the magnitude of the projection of the vector

$$[p_1(d)', (p_1(h)'] = [p_2(d), 1 - p_2(d)] \tag{7.2}$$

onto the subspace $p_1(d) + p_1(h) = 1$ by taking the dot product of 7.2 with $(1/\sqrt{2}, -1/\sqrt{2})$, giving $\sqrt{2}p_2(d) - 1/\sqrt{2}$. Dividing by $\sqrt{2}$ gives the projected derivative of $p_1(d)$ as a function of $p_2(d)$, namely

$$\frac{d}{dt}p_1(d) = p_2(d) - 1/2$$

Similarly we have

$$\frac{d}{dt}p_2(d) = -p_1(d) + 1/2$$

Substituting, we get the second order linear equation

$$\frac{d^2}{dt^2}p_2(d) + p_2(d) = 1/2$$

whose solutions are of the form

$$\alpha \sin(t) + \beta \cos(t) + 1/2$$

Similarly for solutions of $p_1(d)$. Thus 7.1 has just one convergent solution, namely the constant curve corresponding to the unique equilibrium $p_1(d) = p_2(d) = 1/2$. All other curves not touching the polytope boundary will follow a closed elliptical contour and never converge.

Thus the primal algorithm need not find an equilibrium for 3.8 in its general form. As with the discontinuity and numerical questions, however, we interpret this as implying that one should choose biologically plausible instances of 3.8, rather than reject the primal method.

We conclude this section by noting that if we change the -1s to 1s in 7.1, we have the solution

$$\alpha \exp(t) + \beta \exp(-t) + 1/2 \tag{7.3}$$

which gives rise to an unstable saddle equilibrium at $p_1(d) = p_2(d) = 1/2$. Thus from a purely analytic viewpoint we also have the possibility ($\alpha = 0$) of the solution curve reaching an unstable equilibrium. However, in practice the first term in 7.3 eventually will dominate, and the system will reach one of the two stable equilibria $[p_1(d), p_2(d)] = (1, 1)$ or $[p_1(d), p_2(d)] = (0, 0)$.

8 Appendix B: Lemke's Algorithm

We now describe a dual algorithm for 3.8, that is, an algorithm that directly solves the equivalent dual problem 6.3. This procedure, known as Lemke's algorithm, is one of a family of vertex pivoting algorithms

that has been developed for the *linear complementarity problem*, of which 6.3 is an example. (See Murty 1988 for a comprehensive survey. Also Cottle and Dantzig 1968.)

The central algebraic operation of all these algorithms is a *pivot* or *basis change*. The operation acts on a system linear equations

$$\bar{A}x = \bar{b} \tag{8.1}$$

where $\bar{A}$ is an $M \times N$ matrix ($M < N$), and presupposes an $M \times M$ identity matrix I_M distributed among the columns of $\bar{A}$ (the *basis* columns). A pivot is then a choice of a nonbasic column r and a basic column s, and a corresponding multiplication of $\bar{A}, \bar{b}$ by an $M \times M$ matrix B such that $B\bar{A}$ contains an identity matrix in column r and the $m-1$ columns of the former basis excluding s. The significance of a basis is that a solution to $\bar{A}x = \bar{b}$ is trivial since if I_M is, say, the first M columns of $\bar{A}$, then we need just let $x_i = \bar{b}_i$ for $i = 1, \ldots, M$, and $x_i = 0$ for $i = M+1, \ldots, N$.

A key feature of pivoting is that once a column is chosen to enter the basis, the requirement that $\bar{b}$ and $B\bar{b}$ be nonnegative will (under certain mild nondegeneracy conditions) uniquely determine which column will leave.

With Lemke's algorithm we associate 8.1 with the first line of 6.3 and then add to $\bar{A}$ a temporary "artificial column" z of negative numbers (say -1s) corresponding to a new variable z_0. A special initial pivot brings z_0 into the basis and causes the right-hand side of 8.1 to become nonnegative. All subsequent pivots will maintain this nonnegativity, thus satisfying the second line of 6.3 and also determining which column can leave the basis. Satisfying the third line of 6.3 (the "complementarity" condition) will determine which column can enter, and thus a unique pivoting sequence is specified. The algorithm terminates in a solution when the first line of 6.3 is satisfied as well, which coincides with z_0 leaving the basis.

In general Lemke's algorithm may not terminate in a solution. This occurs when the new pivoting column is nonpositive (geometrically an infinite ray), thus making it impossible to pivot into that column and preserve the nonnegativity constraints. A critical issue in Lemke's algorithm is therefore specifying sufficient conditions on the structure of the linear complementarity problem being solved to ensure that

1. either a termination in a solution occurs, or
2. termination in a ray implies there exists no solution.

[In the case of polymatrix game equilibria a solution always exists, so the above conditions (1) and (2) imply that a solution is computed.] In Miller and Zucker (1991) we have observed that polymatrix games may be put in such a form, a special case of a class defined by Lemke (1965) and later known as *copositive-plus* (Cottle and Dantzig 1968). With respect to 6.3, the essential condition for this result is that we may assume $R < 0$.

A result by Eaves (1973) shows that polymatrix games fit into another class of linear complementarity problem of the same general form as 6.3, for which Lemke's algorithm is also guaranteed to terminate successfully. In this case there is no requirement that $R < 0$. However it is necessary that $Ap \leq q$ include the special constraint

$$e^\top p \leq \kappa$$

where e is a vector of 1s, and κ is a variable that is treated during each pivoting operation as though it were arbitrarily large in relation to the absolute values of any numbers used to specify the problem.

As with the simplex method for linear programming, Lemke's algorithm may take an exponential number of pivots to solve certain linear complementarity problems (e.g., Cottle 1980). However, as far as we are aware, all cases where exponential behavior has been demonstrated were specifically created for that purpose. When applied to real world or simulated random problems, the typical number of pivots needed for Lemke's algorithm to terminate is $O(M)$ (Murty 1988, p. 162), or in the present case $O(mn)$. Since each pivot may be accomplished in $O(M^2)$ arithmetic operations, this implies an empirical bound of $O(m^3n^3)$ arithmetic operations. If, as in the present case, m is fixed, then we get $O(n^3)$.

Acknowledgments

The authors thank Frank Ferrie and David Jones for valuable criticism and suggestions. This research was supported by grants from NSERC and AFOSR. S. W. Z. is a Fellow, Canadian Institute for Advanced Research.

References

Adler, I., McClean, R. P., and Provan, J. S. 1980. An application of the Khachiyan-Shor algorithm to a class of linear complementarity problems. Cowles Foundation Discussion Paper **549**, Yale University, New Haven, Connecticut.

Adler, I., Megiddo, N., and Todd, M. J. 1984. New results on the average behavior of simplex algorithms. *Bull. Am. Math. Soc. (N.S.)* **11**, 378–382.

Cottle, R. W. 1980. Observations on a class of nasty linear complementarity problems. *Discrete Appl. Math.* **2**, 89–111.

Cottle, R. W., and Dantzig, G. B. 1968. Complementary pivot theory of mathematical programming. In *Mathematics of the Decision Sciences*, G. B. Dantzig and A. F. Veinott, Jr., eds., Part I, pp. 115–136. AMS.

Dantzig, G. B. 1963. *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ.

Eaves, B. C. 1973. Polymatrix games with joint constraints. *SIAM J. Appl. Math.* **24**, 418–423.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability.* W. H. Freeman, San Francisco.

Hinton, G. E., and Sejnowski, T. J. 1986. Learning and relearning in Boltzmann machines. In *Parallel Distributed Processing,* D. E. Rumelhart and J. L. McClelland, eds., Vol. I, pp. 282–317. MIT Press, Cambridge, MA.

Hopfield, J. J. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. U.S.A.* **79,** 2554–2558.

Hopfield, J. J. 1984. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci. U.S.A.* **81,** 3088–3092.

Hopfield, J. J., and Tank, D. W. 1985. 'Neural' computation of decisions in optimization problems. *Biol. Cybernet.* **52,** 1–12.

Hummel, R. A., and Zucker, S. W. 1983. On the foundations of relaxation labeling processes. *IEEE PAMI* **5,** 267–287.

Johnson, D. S., Papadimitriou, C. H., and Yannakakis, M. 1988. How easy is local search? *J. Comput. Syst. Sci.* **26,** 79–100.

Kinderlehrer, D., and Stampacchia, G. 1980. *An Introduction to Variational Inequalities and Their Applications.* Academic Press, New York.

Kirousis, L. M., and Papadimitriou, C. H. 1988. The complexity of recognizing polyhedral scenes. *J. Comput. Syst. Sci.* **37,** 14–38.

Kuhn, H. W., and Tucker, A. W. 1951. Nonlinear programming. In *Second Berkeley Symposium on Mathematical Statistics and Probability,* J. Neyman, ed., pp. 481–492. University of California Press, Berkeley, CA.

Lemke, C. E. 1965. Bimatrix equilibrium points and mathematical programming. *Management Sci.* **11,** 681–689.

Luenberger, D. G. 1973. *Introduction to Linear and Nonlinear Programming.* Addison-Wesley, Reading, MA.

Miller, D. A., and Zucker, S. W. 1991. Copositive-plus Lemke algorithm solves polymatrix games. *Operations Res. Lett.* **10,** 285–290.

Murty, K. G. 1988. *Linear Complementarity, Linear and Nonlinear Programming.* Heldermann Verlag, Berlin.

Nash, J. F. 1951. Noncooperative games. *Ann. Math.* **54,** 286–295.

Papadimitriou, C. H., Schäffer, A. A., and Yannakakis, M. 1990. On the complexity of local search. *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing,* Baltimore, Maryland, May, 438–445.

Pour-El, M. B., and Richards, I. 1981. The wave equation with computable initial data such that its unique solution is not computable. *Adv. Math.* **39,** 215–239.

Sejnowski, T. J. 1981. Skeleton filters in the brain. In *Parallel Models of Associative Memory,* G. E. Hinton and J. A. Anderson, eds., pp. 189–212. Lawrence Erlbaum, Hillsdale, NJ.

Todd, M. J. 1983. *Polynomial expected behavior of a pivoting algorithm for linear complementarity and linear programming problems.* Tech. Rep. 595, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York.

Tsotsos, J. 1988. A 'complexity-level' analysis of intermediate vision. *Int. J. Comput. Vision* **1,** 303–320.

Vergis, A., Steiglitz, K., and Dickinson, B. 1986. The complexity of analog computation. *Math. Comput. Simulation* **28,** 91–113.

von Neumann, J., and Morgenstern, O. 1944. *Theory of Games and Economic Behavior.* Princeton University Press, Princeton, NJ.

Wilson, G. V., and Pawley, G. S. 1988. On the stability of the travelling salesman problem algorithm of Hopfield and Tank. *Biol. Cybernet.* **58,** 63–70.

Zucker, S. W., Dobbins, A., and Iverson, L. 1989. Two stages of curve detection suggest two styles of visual computation. *Neural Comp.* **1,** 68–81.

Received 23 October 1990; accepted 10 May 1991.