A Continuous-Time Hopfield Net Simulation of Discrete Neural Networks

Šíma, Jiří
1999

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

A Continuous-Time Hopfield Net Simulation of Discrete Neural Networks

Jiří Šíma Pekka Orponen

Technical report No. 773

January 1999

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+4202) 6605 3030 fax: (+4202) 85 85 789
e-mail: sima@uivt.cas.cz.

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

A Continuous-Time Hopfield Net Simulation of Discrete Neural Networks

Jiří Šíma [1] Pekka Orponen [2]

Technical report No. 773
January 1999

Abstract

We investigate the computational power of continuous-time symmetric Hopfield nets. Since the dynamics of such networks are governed by Liapunov (energy) functions, they cannot generate infinite nondamping oscillations, and hence cannot simulate arbitrary (potentially divergent) discrete computations. Nevertheless, we prove that any *convergent* fully parallel computation by a network of n discrete-time binary neurons, with in general asymmetric interconnections, can be simulated by a symmetric continuous-time Hopfield net containing $14n+6$ units using the saturated-linear sigmoid activation function. In terms of standard discrete computation models this result implies that any polynomially space-bounded Turing machine can be simulated by a polynomially size-increasing sequence of continuous-time Hopfield nets. Similar techniques as here yield corresponding results on the convergence time and computational power of discrete-time Hopfield nets.

Keywords
neural networks, continuous-time Hopfield nets, analog computation, computational power

[1]Department of Theoretical Informatics, Institute of Computer Science, Academy of Sciences of the Czech Republic, Research supported by GA ČR Grant No. 201/98/0717.

[2]Department of Mathematics, University of Jyväskylä, P. O. Box 35, FIN-40351 Jyväskylä, Finland, E-mail: orponen@math.jyu.fi. Research partially supported by the Academy of Finland.

1 Introduction

In his 1984 paper [5], John Hopfield introduced a continuous-time version of the very influential associative memory model whose discrete-time version he had analyzed two years earlier in [4]. Part of the appeal of Hopfield's continuous-time model stems from its efficient implementations in analog electrical [5] and optical [16] hardware. Besides associative memory, proposed uses of continuous-time Hopfield nets include, e.g., fast approximate solution of combinatorial optimization problems such as the traveling salesman problem [6].

In this paper, we study the power of continuous-time Hopfield nets as *general computational devices*. At first sight it would appear that the computational capabilities of this model must be severely limited, because the dynamics of any continuous-time Hopfield net $\mathcal{N}$ is governed by a Liapunov, or "energy" function E defined on its state space. The values of E are bounded from below, and they are properly decreasing along any nonconstant computation path of $\mathcal{N}$. A consequence of the Liapunov property is that such a network $\mathcal{N}$ always converges from any initial state towards some stable final state, and in particular global nondamping oscillations of the network state are impossible. The existence of a Liapunov function is a fundamental property of networks whose interconnection weight matrix is *symmetric*, as required for both continuous- and discrete-time Hopfield nets. More general asymmetric networks usually do not behave in the simple manner guaranteed by this property.

Because of the Liapunov property, not even a single oscillating discrete-time neuron can faithfully be simulated by a symmetric continuous-time Hopfield net. However, we shall show that oscillations are the *only* feature that cannot be reproduced, in the sense that any *converging* fully parallel computation by a network of n discrete-time binary neurons, with in general asymmetric interconnections, can be simulated by a symmetric continuous-time Hopfield net containing $14n + 6$ units using the saturated-linear sigmoid activation function.

Observe, namely, that any converging computation by a discrete-time deterministic network of n binary neurons must terminate within 2^n steps. A basic technique used in our proof is then the construction of an $(n+2)$-bit symmetric continuous-time *clock* network (a simulated binary counter) that, using $5n + 6$ units, produces a sequence of 2^n well-controlled oscillations (generated by the second least significant counter bit) before it converges. This sequence of clock pulses is used to drive the rest of the network where each discrete neuron is simulated by a symmetrically interconnected subnetwork of 9 continuous-time units.

The clock network is already by itself of some interest from a dynamical systems perspective, because it provides an explicit example of a Liapunov-type continuous-time system whose convergence time grows exponentially in the system dimension. More precisely, we shall show in Section 3 that the convergence time for an $(n+1)$-bit clock network, which consists of $r = 5n + 1$ units, is $\Omega(2^n/\varepsilon) = \Omega(2^{r/5}/\varepsilon)$, where ε is a parameter controlling the convergence rate of the system. In terms of bit representations this bounds translates to a convergence time of $2^{\Omega(g(M))}$ for a network with an encoding size of M bits, where $g(M)$ is an arbitrary continuous function such that $g(M) = o(M)$, $g(M) = \Omega(M^{2/3})$, and $M/g(M)$ is increasing. This can be compared to a general convergence time upper bound of $2^{O(\sqrt{N})}$ for discrete Hopfield networks with N-bit representations. Thus, the continuous-time implementation actually yields better bounds than the discrete-time one, assuming that the time interval between two subsequent discrete updates corresponds to a continuous time unit.

The predecessors of the present work are: a similar, but considerably simpler, construction used in [10] to prove the computational equivalence of symmetric and asymmetric *discrete-time binary* networks[3], and the simulation of discrete-time networks by *asymmetric* continuous-time networks in [12]. The original idea for the discrete-time clock network used in [10], and on which our current construction is based, stems from [3]. A general survey of topics in continuous-time computation is presented in [11].

As pointed out in [10], polynomial-size increasing sequences of discrete networks are computationally equivalent to (nonuniform) polynomially space-bounded Turing machines (more precisely, they compute the complexity class PSPACE/poly [1, p. 100]). By the result in the present paper, we now know that continuous-time symmetric networks are at least as powerful, i.e. given any polynomially space-bounded Turing machine, we can construct a polynomial-size sequence of continuous-time Hopfield nets for simulating it.

A related line of study concerns the computational power of *finite discrete-time analog-state* neural networks. Here it is known that the computational power of asymmetric networks using the saturated-linear sigmoid activation function increases with the Kolmogorov complexity of the weight parameters [2].[4] On the other hand, it is known that any amount of analog noise reduces the computational power of this model to that of finite automata [9].

In the present abstract we outline our proof construction, and give a simulation example witnessing its validity. The formal verification of the correctness of the construction requires a lengthy and tedious case analysis (similar, but more involved than that in [12]), and will thus be deferred to the full version of the paper.

2 Constructing the Continuous-Time Network

First, we will briefly specify the model of a finite *discrete recurrent neural network*. The network consists of n simple computational *units* or *neurons*, indexed as $1, \ldots, n$, that are connected into a generally cyclic oriented graph or *architecture*, in which each edge (i, j) leading from neuron i to j is labelled with an integer *weight* $w(i, j) = w_{ji}$. The absence of a connection within the architecture corresponds to a zero weight between the respective neurons, and vice versa.

The synchronous computational dynamics of the network, working in *fully parallel mode*, determines the evolution of the *network state* $\mathbf{y}^{(t)} = (y_1^{(t)}, \ldots, y_n^{(t)}) \in \{0, 1\}^n$

[3]Our present construction can actually also be used to improve the discrete-time simulation in [10], which requires a symmetric network of $\Omega(n^2)$ units to simulate a convergent asymmetric network of size n. Using the technique presented in Section 2, the simulation overhead can be reduced to $6n + 2$ units in the discrete case.

[4]With integer weights such networks are equivalent to finite automata [15], while with rational weights arbitrary Turing machines can be simulated [14, 7]. With arbitrary real weights the networks can even have "super-Turing" computational capabilities [13].

for all discrete time instants $t = 0, 1, \ldots$ as follows. At the beginning of a computation the network is placed in an *initial state* $\mathbf{y}^{(0)}$ which may include an external input. At discrete time $t \geq 0$, each neuron $j = 1, \ldots, n$ collects its binary *inputs* from the *states* (*outputs*) $y_i^{(t)} \in \{0, 1\}$ of incident neurons i. Then its integer *excitation* $\xi_j^{(t)} = \sum_{i=0}^{n} w_{ji} y_i^{(t)}$ $(j = 1, \ldots, n)$ is computed as the respective weighted sum of inputs including an integer *bias* w_{j0} which can be viewed as a weight of the formal constant unit input $y_0^{(t)} = 1$. At the next instant $t+1$, an *activation function*, which in this case is the *hard limiter* or *threshold function* s, is applied to $\xi_j^{(t)}$ for all neurons $j = 1, \ldots, n$ in order to determine the new network state $\mathbf{y}^{(t+1)}$ by the following rule:

$$y_j^{(t+1)} = s\left(\xi_j^{(t)}\right) \quad j = 1, \ldots, n \tag{2.1}$$

where

$$s(\xi) = \begin{cases} 1 & \text{for } \xi \geq 0 \\ 0 & \text{for } \xi < 0\,. \end{cases} \tag{2.2}$$

Similarly, a finite *continuous-time analog neural network* is composed of m analog units which operate (in our case) with the *saturated-linear* sigmoid activation function

$$\sigma(\xi) = \begin{cases} 1 & \text{for } \xi > 1 \\ \xi & \text{for } 0 \leq \xi \leq 1 \\ 0 & \text{for } \xi < 0\,. \end{cases} \tag{2.3}$$

Hence, the states of analog units are real numbers within the interval $[0, 1]$, and the weights (including biases), denoted by $v(p, q)$ (for units p, q) are reals as well. In particular, we shall consider *Hopfield* (*symmetric*) *networks*, whose architecture is an undirected graph with symmetric weights $v(p, q) = v(q, p)$ for every p, q. The computational dynamics of a continuous-time network is defined for every real $t > 0$ by the following system of differential equations, with the initial network state $\mathbf{y}(0)$ providing the initial conditions:

$$\frac{dy_p}{dt}(t) = -y_p(t) + \sigma(\xi_p(t)) = -y_p(t) + \sigma\left(\sum_{q=0}^{n} v(p, q) y_q(t)\right) \quad p = 1, \ldots, m\,. \tag{2.4}$$

By a Liapunov function argument [5], it can be shown that a Hopfield network converges from any initial state $\mathbf{y}(0)$ to some stable state satisfying $dy_p/dt = 0$ for all $p = 1, \ldots, m$. The set of stable states of the continuous-time system (2.4) coincides with that of the discrete system (2.1).

Now, given a convergent discrete asymmetric neural network with n neurons, we shall construct a computationally equivalent analog Hopfield network with $m = 14n+6$ continuous-time units. The analog network will be composed of an $(n + 2)$-bit binary counter (clock) subnetwork consisting of $5n + 6$ units, each starting at the zero initial state, and n other subnetworks, each containing 9 analog units for the purpose of simulating one discrete neuron.

The initial construction for a 2-bit counter is presented in Figure 2.1, where the symmetric connections between units are labelled with the respective weights, and the biases are indicated by the edges drawn without an originating unit.

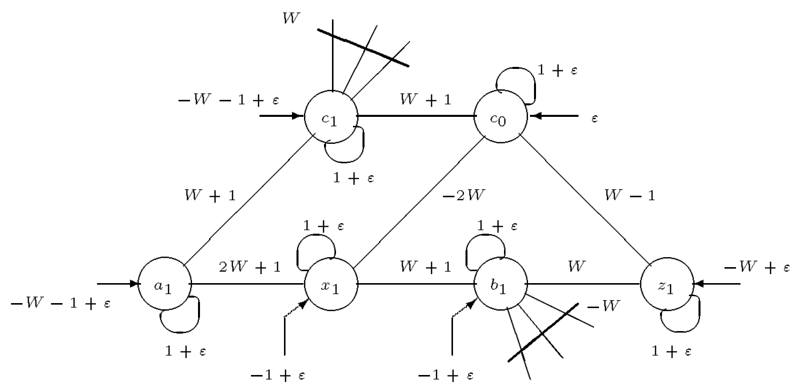

Figure 2.1: A continuous-time 2-bit counter network

The counter bit c_0 of order 0 starts its excitation with a bias $v(0, c_0) = \varepsilon > 0$ which is a small (e.g. $\varepsilon \leq 0.1$) optional parameter that also determines the time overhead of the simulation. Because of its feedback weight $v(c_0, c_0) = 1+\varepsilon$ the state of c_0 gradually grows towards saturation at value 1, at which time (more precisely, when the state of c_0 is "sufficiently close" to 1) we say that the unit is *active* or *fires*. This trick of gradual transition from 0 to 1 is used repeatedly throughout the analog network construction. The operation of the remaining units in Figure 2.1, which are of order 1, is the same as that of the units of a higher order $k > 1$ whose inductive description and explanation follows (although the definition of weights is slightly different).

Thus, suppose that the counter has been constructed up to the first $k < n+2$ counter bits $c_0, \ldots, c_{k-1}$, and denote by P_k the set of all its $m_k = 5k-4$ units, including the auxiliary ones labelled $a_\ell, x_\ell, b_\ell, z_\ell$, for $\ell = 1, \ldots, k-1$. Then, the counter unit c_k with a feedback weight $v(c_k, c_k) = 1+\varepsilon$, is connected to all m_k units $p \in P_k$ via weights $v(p, c_k) = 1$ which, together with its bias $v(0, c_k) = -m_k + \varepsilon$, make c_k to fire shortly after all these units are active (including the first k counter bits $c_0, \ldots, c_{k-1}$ which means that counting from 0 to $2^k - 1$ has been accomplished). Further, the unit c_k is connected to a sequence of 4 auxiliary units a_k, x_k, b_k, z_k (all having feedbacks $1+\varepsilon$) which are being, one by one, activated after c_k fires. This is implemented by the following weights $v(c_k, a_k) = m_k$, $v(a_k, x_k) = V_k$ (specified below), $v(x_k, b_k) = 1$, $v(b_k, z_k) = V_k - m_k$, and biases $v(0, a_k) = -m_k + \varepsilon$, $v(0, x_k) = v(0, b_k) = -1 + \varepsilon$, $v(0, z_k) = m_k - V_k + \varepsilon$. The units a_k, b_k only slow down the continuous-time state flow in order to synchronize the computation. The unit x_k resets all the units in P_k to their initial zero states. For this purpose, x_k is further connected to each $p \in P_k$ via a sufficiently large negative weight $v(x_k, p) < 0$ such that $-v(x_k, p) > 1 + \sum_{q \in P_k : v(q,p) > 0} v(q, p)$ exceeds their mutual positive influence (including the weight $v(c_k, p) = 1$). This also determines the above-mentioned large positive weight parameter $V_k = 1 - \sum_{p \in P_k} v(x_k, p)$ that makes the state of x_k (similarly for z_k) independent of the outputs from $p \in P_k$. Finally, the unit z_k balances the influence of x_k on P_k so that the first k counter bits can again count from 0 to $2^k - 1$ but now with c_k being active. This is achieved by the weight $v(z_k, p) = -v(x_k, p) - 1$ for each $p \in P_k$ in which -1 compensates $v(c_k, p) = 1$. This completes the induction step.

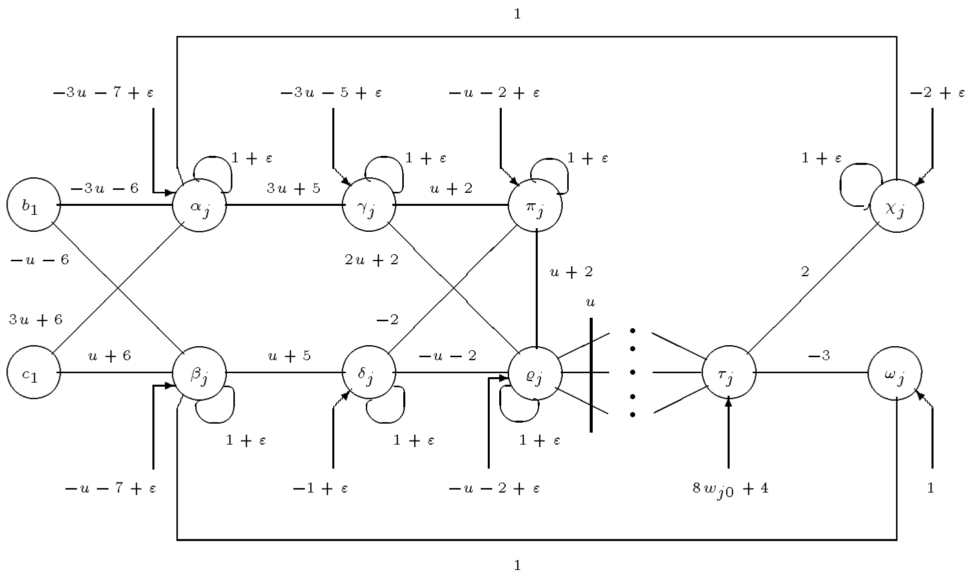

Figure 2.2: A continuous-time simulation of a discrete neuron

Furthermore, in Figure 2.2 a 9-unit symmetric analog subnetwork is depicted for simulating one neuron j from the discrete network. The output $y_j^{(t)}$ of the binary-state neuron at discrete time instant $t \geq 0$ is represented by the state of analog unit ϱ_j, whose state is momentarily stabilized by the support from unit π_j that doubles this state.

The new output $y_j^{(t+1)}$ of j at discrete time $t+1$ is computed in a unit τ_j that is connected to the appropriate units ϱ_i from the other subnetworks, as required by the original discrete network, via slightly adjusted weights $v(\varrho_i, \tau_j) = 8w(i,j)$ (including the bias $v(0,\tau_j) = 8w(0,j)+4$).

The parameter u is chosen as the maximum value of $\sum_{i=1;v(\varrho_j,\tau_i)>0}^{n} v(\varrho_j,\tau_i)$ and $-\sum_{i=1;v(\varrho_j,\tau_i)<0}^{n} v(\varrho_j,\tau_i)$ for all $j=1,\ldots,n$, in order to keep the unit ϱ_j from being affected by the units τ_i. Further, the unit χ_j receives the state $y_j^{(t+1)}$ from τ_j while the unit ω_j computes its negation. In reverse, χ_j, ω_j cannot influence τ_j since the respective weights $2, -3$ are too small in comparison with bias $v(0,\tau_j) = 8w_{j0}+4$ (recall that the original asymmetric weights w_{ji} are integers). In the meantime, the remaining units $\alpha_j, \beta_j, \gamma_j, \delta_j$ are *passive* (have states close to 0), and the underlying subnetwork is temporarily stable.

The update of the simulated discrete state in the continuous-time network, i.e. replacing the old state $y_j^{(t)}$ in ϱ_j (and π_j) by the new state $y_j^{(t+1)}$ from τ_j, is controlled by pulses from the clock. This process is initiated by the activity of counter bit c_1, and concluded by the subsequent activity of b_1 which compensates the influence of c_1 on α_j, β_j. The size of the parameter $W = (3u+6)n$ in Figure 2.1 ensures that the states of the counter units c_1, b_1 are not affected by the weights originating in the α_j or β_j units active in the n subnetworks. Note that the units c_1 and b_1 in the above-described $(n+2)$-bit counter fire 2^n times, which is sufficient to simulate any convergent computation on a discrete neural network of size n.

Thus, for $y_j^{(t+1)} = 1$ the unit χ_j gets activated and this, together with the support from c_1, induces the unit α_j to fire. The signal from α_j is further propagated following the non-increasing sequence of weights and biases via the synchronizing unit γ_j up to ϱ_j, π_j making them active as required.

In the opposite case when $y_j^{(t+1)} = 0$, unit ω_j gets activated and this, together with the support from c_1, induces the unit β_j to fire. Further, the unit β_j sends the signal to δ_j, and this inhibits ϱ_j, π_j by means of sufficiently negative weights so that they are passive, as required.

In the meantime, b_1 locks the channels via α_j, β_j. Finally, the new discrete state $y_j^{(t+2)}$ is computed by τ_j and the subnetwork is stable until b_1 fires again.

3 Convergence Time Analysis

The $(n+1)$-bit continuous-time clock network from Section 2, which consists of $r = 5n+1$ units, can be exploited to achieve a lower bound on the convergence time of continuous-time networks. For this purpose, the duration of a gradual state transition from 0 to 1 of the unit c_0 will be estimated. During its state growth the influence of the remaining units on c_0 is balanced, and may be neglected in order to simplify our analysis. Thus, the state evolution of c_0 in continuous time, denoted by $y(t)$, can be described by the following differential equation with the initial condition $y(0) = 0$:

$$\frac{dy}{dt}(t) = -y(t) + \sigma(\varepsilon + (1+\varepsilon)y(t)) \tag{3.1}$$

whose solution can explicitly be expressed as follows:

$$y(t) = \begin{cases} e^{\varepsilon t} - 1 & \text{for } 0 \le t \le t_1 \\ 1 - \varepsilon e^{(1+\varepsilon)t_1 - t} & \text{for } t \ge t_1 \end{cases} \tag{3.2}$$

where $t_1 = (1/\varepsilon)\ln(2/(1+\varepsilon))$ and $y(t_1) = (1-\varepsilon)/(1+\varepsilon)$. Hence, for a small $\varepsilon < 1$ the respective state transition takes time at least $t_1 = \Omega(1/\varepsilon)$ which, together with the fact that the unit c_0 fires 2^n times before the $(n+1)$-bit clock converges, provides the desired lower bounds $\Omega(2^n/\varepsilon) = \Omega(2^{r/5}/\varepsilon)$ on the convergence time.

Now, we will express this bound in terms of the size M in bits of the network representation. First, consider the integer part of the weight parameter representation excluding fractions ε. By induction, the maximum integer weight parameter in the clock is of order $2^{O(r)}$. This corresponds to $O(r)$ bits per weight that is repeated $O(r^2)$ times, and thus yields at most $O(r^3)$ bits in the representation. In addition, the biases and feedbacks of the r units include the fraction ε, and taking this into account requires $\Theta(r \log(1/\varepsilon))$ additional bits, say at least $\kappa r \log(1/\varepsilon)$ bits for some constant $\kappa > 0$.

By choosing $\varepsilon = 2^{-f(r)/(\kappa r)}$ in which f is a continuous increasing function whose inverse is defined as $f^{-1}(\mu) = \mu/g(\mu)$, where g is an arbitrary function such that $g(\mu) = \Omega(\mu^{2/3})$ (implying $f(r) = \Omega(r^3)$) and $g(\mu) = o(\mu)$, we get $M = \Theta(f(r))$, especially $M \ge f(r)$ from $M \ge \kappa r \log(1/\varepsilon)$. Finally, the convergence time $\Omega(2^{r/5}/\varepsilon)$ can be translated to $\Omega(2^{f(r)/(\kappa r) + r/5}) = 2^{\Omega(f(r)/r)}$ which can be rewritten as $2^{\Omega(M/f^{-1}(M))} = 2^{\Omega(g(M))}$ since $f(r) = \Omega(M)$ from $M = \Theta(f(r))$ and $f^{-1}(M) \ge r$ from $M \ge f(r)$.

On the other hand, by the Liapunov property the discrete-time binary Hopfield network with n neurons and an $N = \Omega(n^2)$-bit representation converges after at most $O(2^n)$ update steps, which gives a convergence time upper bound of $2^{O(\sqrt{N})}$. The continuous-time implementation actually yields better bounds $2^{\Omega(g(M))}$ for any $g(M) = \Omega(M^{2/3})$ up to $g(M) = o(M)$ than the discrete-time one, assuming that a time interval between two subsequent discrete updates corresponds to a continuous time unit.

4 A Simulation Example

A computer program HNGEN has been created to automate the construction from Section 2. The input for HNGEN is a text file containing the asymmetric weights and biases of the discrete neural network, as well as its initial state. The program generates the corresponding system (2.4) of differential equations, together with the respective initial conditions in the form of a FORTRAN subroutine which describes the continuous-time dynamics of the analog Hopfield net that simulates the given discrete network. This FORTRAN procedure is then presented to a powerful numerical solver UFO [8] that provides the user with a numerical solution for the respective system (2.4), i.e. it draws the graphs of the state evolution in time for selected analog units.

By using the program HNGEN, the construction from Section 2 has been successfully tested on several examples. Consider e.g. a simple discrete asymmetric neural network which is an oriented cycle of 3 neurons with all the weights 1 and biases -1. Now, for the initial state including exactly one active neuron, the signal is propagated through the cycle in a circle. Eight (2^3) steps of this computation were simulated by the respective continuous-time Hopfield network with 48 units and $\varepsilon = 0.1$. The state evolution of the corresponding 3 analog units $\varrho_1, \varrho_2, \varrho_3$ together with the counter bit c_1 is shown in Figure 4.1.

5 Conclusions and Open Problems

We have proved that an arbitrary discrete-time binary network can be simulated by a symmetric continuous-time network with only a linear increase in the network size. The existence of a Liapunov function for symmetric networks precludes the existence of undamping oscillations in the continuous-time system, but nevertheless our construction relies heavily on the finite sequence of clock pulses generated by the continuous-time counter subnetwork.

From the point of view of understanding analog computation in general this technique is somewhat unsatisfying, since we are still basically discretizing the continuous-time computation. It would be most interesting to develop some theoretical tools (e.g. complexity measures, reductions, universal computation) for “naturally” continuous-time computations that exclude the use of discretizing oscillations.

Another challenge for further research is to prove *upper bounds* on the power of continuous-time networks. Note that in the case of discrete-time analog-state networks a single fixed-size network with rational-number parameters can be computationally universal, i.e. able to simulate a universal Turing machine on arbitrary inputs [14].

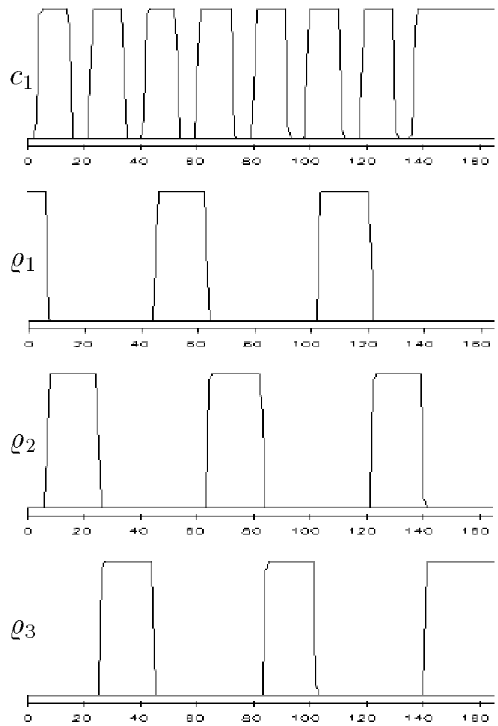

Figure 4.1: Simulation of a discrete 3-neuron cycle network

For example, can this strong universality result be generalized for continuous-time networks? Also, we have established an exponential lower bound on the convergence time of symmetric continuous-time networks: can a matching upper bound be proved, or the lower bound be increased?

Bibliography

[1] Balcázar, J. L., Díaz, J., Gabarró, J. *Structural Complexity I.* Springer-Verlag, Berlin Heidelberg, 1988.

[2] Balcázar, J. L., Gavaldà, R., Siegelmann, H. T. Computational power of neural networks: A characterization in terms of Kolmogorov complexity. In *IEEE Transactions of Information Theory*, **43**, 1175–1183, 1997.

[3] Goles, E., Martínez, S. Exponential transient classes of symmetric neural networks for synchronous and sequential updating. *Complex Systems*, **3**, 589–597, 1989.

[4] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, vol. **79**, 2554–2558, 1982.

[5] Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. In *Proceedings of the National Academy of Sciences*, vol. **81**, 3088–3092, 1984.

[6] Hopfield, J. J., Tank, D. W. "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, **52**, 141-152, 1985.

[7] Indyk, P. Optimal simulation of automata by neural nets. In *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science*, vol. **900** of LNCS, 337-348, Springer-Verlag, Berlin, 1995.

[8] Lukšan, L., Tůma, M., Šiška, M., Ramešová, N. Interactive system for Universal Functional Optimization (UFO) version 1988. Technical report V-766, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, 1998.

[9] Maass, W., Orponen, P. On the effect of analog noise in discrete-time analog computations. *Neural Computation*, **10**, 1071–1095, 1998.

[10] Orponen, P. The computational power of discrete Hopfield nets with hidden units. *Neural Computation*, **8**, 403–415, 1996.

[11] Orponen, P. A survey of continuous-time computation theory. In D.-Z. Du and K.-I. Ko, editors, *Advances in Algorithms, Languages, and Complexity*, 209–224, Kluwer Academic Publishers, Dordrecht, 1997.

[12] Orponen, P. The computational power of continuous time neural networks. In *Proceedings of the SOFSEM Seminar on Current Trends in Theory and Practice of Informatics*, Milovy, Czech Republic, vol. **1338** of LNCS, 86–103, Springer-Verlag, Berlin, 1997.

[13] Siegelmann, H. T., Sontag, E. D. Analog computation via neural networks. *Theoretical Computer Science*, **131**, 331–360, 1994.

[14] Siegelmann, H. T., Sontag, E. D. Computational power of neural networks. *Journal of Computer System Science*, **50**, 132–150, 1995.

[15] Šíma, J., Wiedermann, J. Theory of neuromata. *Journal of the ACM*, **45**, 155–178, 1998.

[16] Stoll, H. M., Lee, L.-S. A continuous-time optical neural network. In *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, vol. **II**, 373–384, 1988.