The Computational Power of Continuous Time Neural Networks (Preliminary Version)

Pekka Orponen
Department of Computer Science
University of Helsinki, Finland*

August 28, 1995

Abstract

We investigate the computational power of continuous-time neural networks with Hopfield-type units. We prove that polynomial-size networks with saturated-linear response functions are at least as powerful as polynomially space-bounded Turing machines.

1 Introduction

In a paper published in 1984 [11], John Hopfield introduced a continuous-time version of the neural network model whose discrete-time variant he had discussed in his seminal 1982 paper [10]. The 1984 paper also contains an electronic implementation scheme for the continuous-time networks, and an argument showing that for sufficiently large-gain nonlinearities, these behave similarly to the discrete-time ones, at least when used as associative memories.

The power of Hopfield's discrete-time networks as general-purpose computational devices was analyzed in [17, 18]. In this paper we conduct a similar analysis for networks consisting of Hopfield's continuous-time units; however we are at this stage able to analyze only the general asymmetric networks, and the very interesting subclass of continuous-time networks

*Address: P. O. Box 26, FIN–00014 University of Helsinki, Finland. E-mail: orponen@cs.helsinki.fi. This work was done during the author's visit to the Institute for Theoretical Computer Science, Technical University of Graz, Austria.

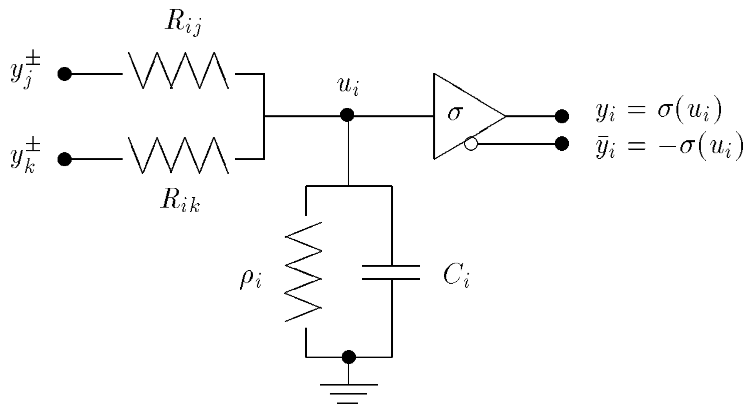

Figure 1: An electrical model of Hopfield's continuous-time neuron.

with symmetric interconnections has to wait for further research. Also, our analysis is restricted to networks with saturated-linear response functions, although computer experiments do indicate that our constructions work equally well for e.g. the standard sigmoid nonlinearities.

Under the above assumptions, we prove that sequences of networks of polynomially increasing size can compute all the functions in the class PSPACE/poly, i.e. the same functions as are computed by polynomially space-bounded nonuniform Turing machines. Such analyses of the computational power of continuous-time processes are at the moment relatively rare in the literature, but we expect their number to grow along with the current increase of interest in analog computation. (The only other analysis that explicitly addresses computational complexity issues seems to be that in [22]. Computability aspects have been studied more often; see, e.g. [2, 5, 6, 15, 16, 19, 20].)

2 A continuous-time neural network model

An electrical model of Hopfield's continuous-time neuron is shown in Figure 1. Here σ denotes the characteristic of the nonlinear amplifier, and ρ_i and C_i are its input resistance and capacitance, respectively. In the following analyses we shall consider only the saturated-linear characteristic

$$\sigma(z) = \begin{cases} -1, & \text{for } z < -1, \\ z, & \text{for } -1 \leq z \leq 1, \\ 1, & \text{for } z > 1. \end{cases}$$

The input voltage of the amplifier is denoted by u_i, and the output voltage by y_i. In order to establish inhibitory interconnections between such units, also the inverted output voltages $\bar{y}_i = -y_i$ are needed.

The unit i indicated in the figure draws input from units j and k via two resistors, whose resistances are denoted by R_{ij} and R_{ik}. The voltages $y_j^{\pm}$ and $y_k^{\pm}$ are obtained from the appropriate output terminals of units j and k, depending on whether the inputs are excitatory or inhibitory.

By Kirchhoff's current law, the circuit equations for a network of p such units can be written as

$$C_i \frac{du_i}{dt} + \frac{u_i}{\rho_i} = \sum_{j=1}^{p} \frac{1}{R_{ij}} (y_j^{\pm} - u_i), \qquad \text{for} \quad i = 1, \ldots, p.$$

By choosing the circuit parameters appropriately and normalizing the time constants to 1, we can use such a network to implement any system of first-order nonlinear differential equations of the form

$$\frac{du_i}{dt} = -u_i + \sum_{j=1}^{p} h_{ij} \sigma(u_j), \qquad i = 1, \ldots, p. \tag{1}$$

(We essentially choose $R_{ij} = 1/h_{ij}$ and normalize; for details see [11].) This is the formulation of the network we are going to use: i.e., we assume that the state u_i of each unit i, with input connections of weight h_{ij} from the other units, develops as described by equation (1).

3 Simulating Turing machines by networks

We shall consider classes of Boolean functions computed by networks of continuous-time units. The appropriate definitions may be framed in e.g. the following manner. Let N be a network of p units, including a special indicator unit u_{done}. Let $\iota : \{0,1\}^n \rightarrow \mathcal{R}^p$ and $\rho : \mathcal{R}^p \rightarrow \{0,1\}^m$ be two "simple" mappings, used to translate a binary input string of length n to an initial network state, and a final network state into a binary output string of length m. Given an input string $x \in \{0,1\}^n$, the network is initialized in state $\iota(x)$; this initial state should in particular satisfy $\sigma(u_{done}) = -1$. The network is then allowed to run until $\sigma(u_{done})$ achieves value 1: the computation is *well-behaved* if during its course the value of $\sigma(u_{done})$ increases monotonically from -1 to 1, and when $\sigma(u_{done}) = 1$, the value of $\rho(u)$ stays constant. The result of the computation is then the final stable value of $\rho(u)$.

Assuming the above notion of a well-behaved computation, a network N thus computes a partial mapping

$$f_N : \{0,1\}^n \to \{0,1\}^m.$$

The mapping is partial, because we take its value to be undefined for inputs that do not lead to a well-behaved computation.

For simplicity, we consider from now on only networks with a single-bit output (i.e. $m = 1$); the extensions to networks with multiple-bit outputs are straightforward. The *language recognized* by an n-bit input, single-bit output network N is defined as

$$L(N) = \{x \in \{0,1\}^n \mid f_N(x) = 1\}.$$

A *sequence* of networks $(N_n)_{n\geq 0}$ recognizes a language $A \subseteq \{0,1\}^*$, if each network N_n recognizes the language $A \cap \{0,1\}^n$. A sequence of networks has *polynomial size* if there is a polynomial $q(n)$ such that for each n, the network N_n has at most $q(n)$ units.

Let $\langle x, y\rangle$ be some standard pairing function mapping pairs of binary strings to binary strings (see, e.g. [4, p. 7]). A language $A \subseteq \{0,1\}^*$ belongs to the nonuniform complexity class PSPACE/poly ([3], [4, p. 100], [12]), if there are a polynomial space bounded Turing machine M, and an "advice" function $f : N \to \{0,1\}^*$, where for some polynomial q and all $n \in N$, $|f(n)| \leq q(n)$, and for all $x \in \{0,1\}^*$,

$$x \in A \quad \Leftrightarrow \quad M \text{ accepts } \langle x, f(|x|)\rangle.$$

It was shown in [17] that all languages in PSPACE/poly can be recognized by polynomial-size sequences of discrete Hopfield networks, even symmetric ones. (Recall that in a discrete Hopfield net the units have bipolar, i.e. ± 1 states, and each unit i updates its state from time t to time $t+1$ according to the rule $y_i(t+1) = \mathrm{sgn}(\sum_j h_{ij} y_j(t))$, where $\mathrm{sgn}(z) = 1$ if $z \geq 0$

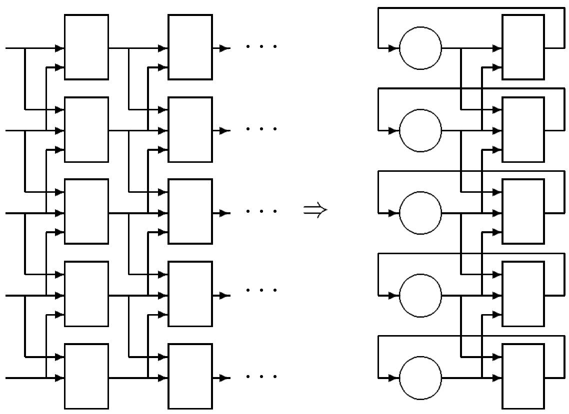

Figure 2: Simulation of a space bounded Turing machine by an asymmetric recurrent net.

and -1 if $z < 0$.) As the construction without the symmetricity restriction is not too difficult, we shall for completeness outline it here[1].

Let $A \in$ PSPACE/poly via a machine M and advice function f. Let the space complexity of M on input $\langle x, f(|x|)\rangle$ be bounded by a polynomial $q(|x|)$. Without loss of generality (see, e.g. [4]) we may assume that M has only one tape, halts on any input $\langle x, f(|x|)\rangle$ in time $c^{q(|x|)}$, for some constant c, and indicates its acceptance or rejection of the input by printing a 1 or a 0 on the first square of its tape.

Following the standard simulation of Turing machines by combinational circuits [4, pp. 106–112], it is straightforward to construct for each n a feedforward circuit that simulates the behavior of M on inputs of length n. (More precisely, the circuit simulates computations $M(\langle x, f(n)\rangle)$, where $|x| = n$.) This circuit consists of $c^{q(n)}$ "layers" of $O(q(n))$ parallel wires, where the tth layer represents the configuration of the machine M at time t (Figure 2, left). Every two consecutive layers of wires are interconnected by an intermediate layer of $q(n)$ constant-size subcircuits, each implementing the local transition rule of machine M at a single position of the simulated configuration. The input x is entered to the circuit along input wires; the

[1] In fact, this version of the result, i.e. Turing machine simulation by asymmetric networks with synchronously updated units, was already obtained by Lepley and Miller in the unpublished report [13]. The result was extended to symmetric networks in [17], and to networks with asynchronous updates in [18].

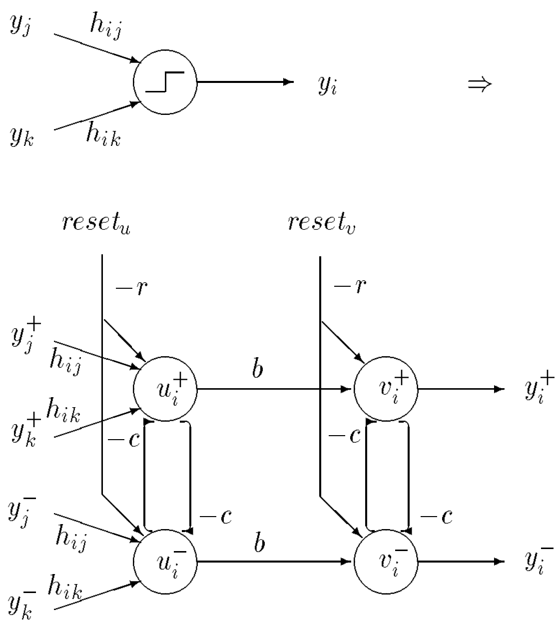

Figure 3: Simulation of a discrete-time unit by two bistable continuous-time unit pairs.

advice string $f(n)$ appears as a constant input on another set of wires; and the output is read from the particular wire at the end of the circuit that corresponds to the first square of the machine tape.

One may now observe that the interconnection patterns between layers are very uniform: all the local transition subcircuits are similar, with a structure that depends only on the structure of M, and their number depends only on the length of x. Hence we may replace the exponentially many consecutive layers in the feedforward circuit by a single transformation layer that feeds back on itself (Figure 2, right). The size of the recurrent network thus obtained is then only $O(q(n))$. When initialized with input x loaded onto the appropriate input units, and advice string $f(n)$ mapped to the appropriate initially active units, the network will converge in $O(c^{q(n)})$ update steps, at which point the output can be read off the unit corresponding to the first square of the machine tape. It is also easy to arrange for a separate unit u_{done} whose value flips from -1 to 1 when the simulated machine M enters a halting state.

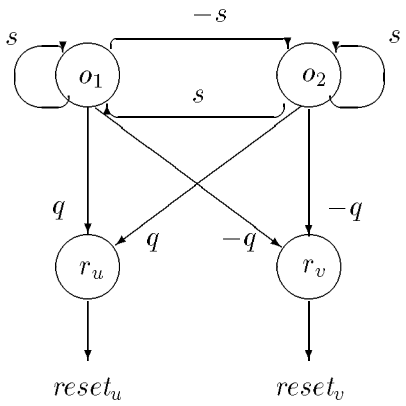

Figure 4: Deriving reset pulses from an oscillating unit pair.

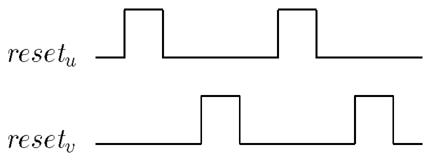

Figure 5: Reset pulse sequence.

Let us then turn to the continuous-time simulation. We shall simply take the discrete network of Figure 2 (or any other discrete network, for that matter), and replace it unit by unit with a computationally equivalent continuous-time system. Each discrete-time unit i is replaced by two *bistable pairs* of continuous-time units as indicated in Figure 3[2]. The units in the bistable pairs are periodically reset by alternating "clock pulses" derived from an *oscillating pair* of units as shown in Figure 4. The sequencing of the reset signals is schematically depicted in Figure 5. As regards the values of the various parameters, we shall just note here that they should be chosen so that $r \gg c \gg b > 0$, and $s > 1$; we shall discuss appropriate choices in more detail later. For simplicity, we are always assuming that the connection weights in the simulated discrete network are integers.

The computational idea underlying this construction is the following. The pair of units labeled $v_i^{\pm}$ in Figure 3 represents the state of the simulated discrete unit i at each discrete time t in a redundant manner, so that $y_i^+ = \sigma(v_i^+) = 1$ if $y_i(t) = 1$, and $y_i^- = \sigma(v_i^-) = 1$ if $y_i(t) = -1$. As long as no reset signals arrive from the clock subnetwork this representation is stable because of the inhibitory connections of weight $-c \ll -b$ between the units v_i^+ and v_i^-.

Assume then that a $reset_u$ signal from the clock subnetwork drives the states of all of the u units in the network close to the values $u^{\pm} = -r$. When the signal falls off, the u units start to compete for activation, based on the inputs they receive from the v units (which have not been reset, and remain stable). It can be seen that the net input to unit u_i^+ (resp. u_i^-) is positive if and only if in the discrete network $y_i(t+1) = 1$ (resp. -1). Thus, as a result of this biased competition, the u units will converge to values that represent the states of the discrete units at time $t+1$: $\sigma(u_i^+) = 1$ if $y_i(t+1) = 1$, and $\sigma(u_i^-) = 1$ if $y_i(t+1) = -1$.

When the u pairs have stabilized, a $reset_v$ signal from the clock similarly drives all the v units close to $-r$. When this signal falls off, the values of the u units are simply copied into the v units by the competitive mechanism, biased by connections of weight b from the u units to the v units. After the v pairs have stabilized, the network is ready for another step of the simulation. (Thus, we could actually double the speed of the simulation by having the u and v units represent the discrete units at even and odd times

[2]Note that Figures 3, 4, and 6 are really just pictorial representations of the equations (1). Correspondingly, we shall in the sequel use the terms "unit" and "variable" completely interchangeably. The actual physical implementation of these "units" is rather more complicated, as indicated in Section 2.

t, and computing new values into the v units instead of just copying the u values.)

We shall analyze the simulation in a general way in Section 4, but let us first look at an example. Figure 6 shows the continuous-time implementation of a single discrete-time unit with a self-connection of weight -1, i.e. a discrete oscillator. The relevant parameter values are indicated in the figure. Note that each of the u and v units has an internal bias of weight -8: this has the effect of rescaling the *reset* signals arriving at them from range $[-1, 1]$ with weight -8 to range $[0, 1]$ with weight -16. Also the r units, used for deriving the *reset* signals from the oscillating o_1/o_2 pair, have internal biases of weight -8, in order to drive their outputs to -1 when they receive zero input (i.e., when the inputs arriving from the o_1 and o_2 units cancel each other).

Figures 7–14, obtained from a MATLAB [14] numerical integration of the corresponding differential equations, illustrate the behavior of the system. Consider for instance Figure 7, which shows the time development of the internal states of the two oscillating units o_1 and o_2. (The solid line represents o_1, the dashed line o_2.) The differential equations governing the corresponding variables are:

$$\begin{aligned} \dot{o}_1 &= -o_1 + s\sigma(o_1) + s\sigma(o_2), \\ \dot{o}_2 &= -o_2 + s\sigma(o_2) - s\sigma(o_1), \end{aligned} \tag{2}$$

where we have for brevity adopted the Newtonian dot notation for the time derivatives, and dropped explicit references to the time variable. Here σ is the saturated-linear response function, and $s = 16$. The initial conditions are $o_1(0) = -31$, $o_2(0) = 1$. A state space plot of o_1 vs. o_2 appears in Figure 8. (To show the emergence of the limit cycle more clearly, we have here chosen the initial conditions $o_1(0) = -1$, $o_2(0) = 1$.) Figure 9 shows the reset pulses derived from the oscillator, i.e. the values $reset_u = \sigma(r_u)$ and $reset_v = \sigma(r_v)$, where r_u and r_v are governed by the equations

$$\begin{aligned} \dot{r}_u &= -r_u + q\sigma(o_1) + q\sigma(o_2) - q, \\ \dot{r}_v &= -r_v - q\sigma(o_1) - q\sigma(o_2) - q, \end{aligned} \tag{3}$$

for $q = 8$. (The solid line corresponds to $reset_u$, the dashed one to $reset_v$.)

Computationally, the most interesting graphs are those in Figures 10 and 11: these show the development of the states of the u and v units, starting from the initial conditions $u^+(0) = v^+(0) = 3$, $u^-(0) = v^-(0) = -3$. (The solid lines in both figures represent the "+"-units, the dashed lines the

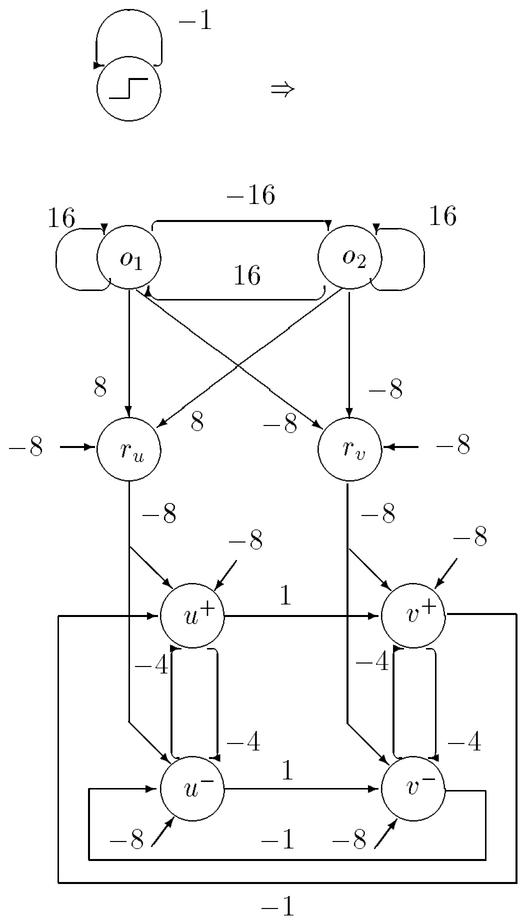

Figure 6: Continuous-time simulation of a discrete oscillator.

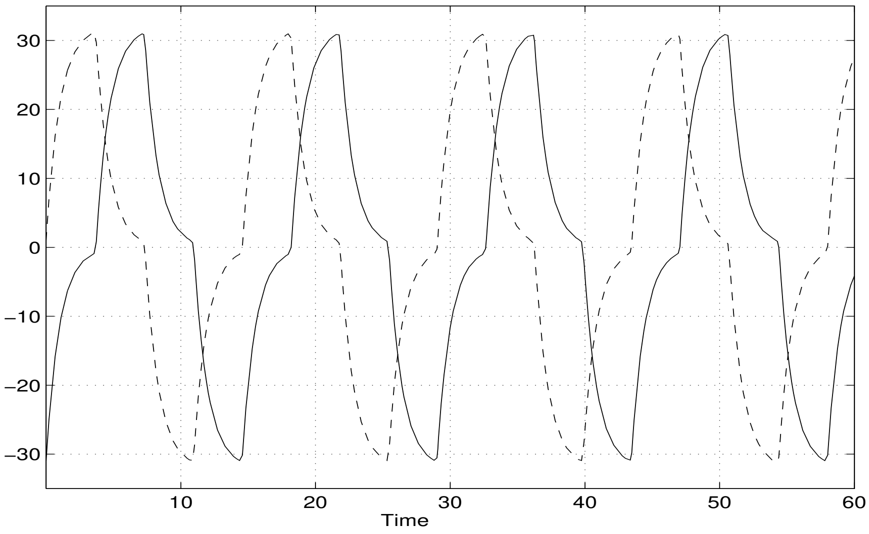

Figure 7: Internal states of the units o_1 and o_2.

"−"-units.) The corresponding output signals, i.e. the values $\sigma(u^{\pm})$ and $\sigma(v^{\pm})$ are shown in Figures 12 and 13. For completeness, let us write down also the equations for the u and v units, as inferred from the diagram in Figure 6:

$$\begin{aligned}
\dot{u}^+ &= -u^+ + h\sigma(v^+) - c\sigma(u^-) - r \cdot reset_u - r, \\
\dot{u}^- &= -u^- + h\sigma(v^-) - c\sigma(u^+) - r \cdot reset_u - r, \\
\dot{v}^+ &= -v^+ + b\sigma(u^+) - c\sigma(v^-) - r \cdot reset_v - r, \\
\dot{v}^- &= -v^- + b\sigma(u^-) - c\sigma(v^+) - r \cdot reset_v - r,
\end{aligned} \tag{4}$$

where $h = -1$, $b = 1$, $c = 4$, and $r = 8$.

One can observe in Figure 10 first the resetting of the u^+ and u^- units in response to the $reset_u$ signal at about time $t = 4$, and then the emergence of the competition between the two units as the reset signal is switched off, at about time $t = 8$. The competition is eventually won by the u^- unit, because it receives a weighted input signal of strength 1 from unit v^-, whereas unit u^+ receives a weighted input signal of strength -1 from unit v^+. (This initial part of the computation is shown in more detail in Figure 14.) In Figure 11 one may then observe how the new states of the u units are copied into the v units, after these have been cleared by the $reset_v$

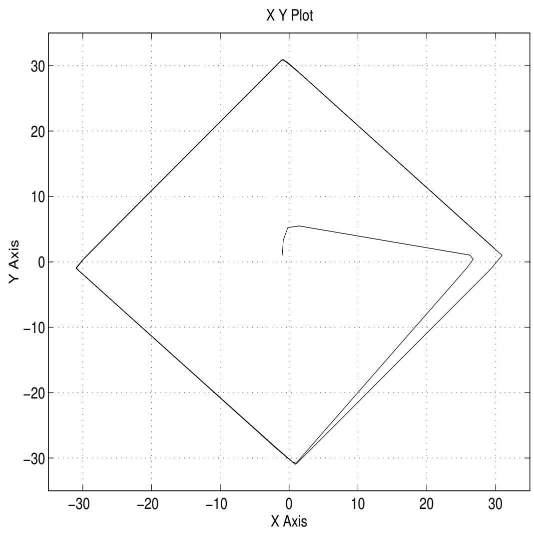

Figure 8: State space plot of o_1 vs. o_2.

signal, at about time $t = 15$. The cycle starts again, but from inverted initial conditions, at about time $t = 19$ with the next $reset_u$ signal.

4 General analysis

Let us then look more generally into the behavior of the equations governing the various components of the simulation, starting with the oscillating pair of units o_1 and o_2. From equation (2) one can see that this system has a fixed point at origin — in fact, some amount of tedious algebra, considering separately the cases $o_1 \lesseqgtr 0$, $o_2 \lesseqgtr 0$, shows that this is the *only* fixed point. The Jacobian matrix of the system at origin is

$$J = \begin{pmatrix} -1+s & s \\ -s & -1+s \end{pmatrix},$$

with eigenvalues $(s-1) \pm si$, so the fixed point at origin is repelling if and only if $s > 1$. It is easy to see that if the initial conditions satisfy $-2s \leq o_1, o_2 \leq 2s$, then the state of the system stays in this region; thus, by the the Poincaré–Bendixson theorem [9] this region contains a limit cycle of the system.

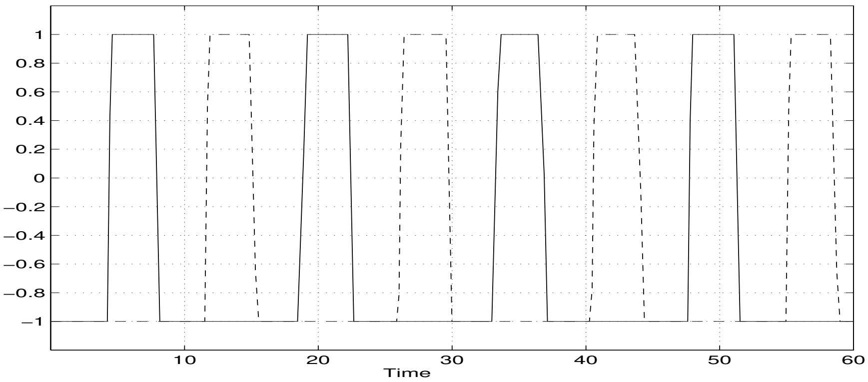

Figure 9: Reset pulses $reset_u$ and $reset_v$ derived from the oscillating pair.

While determining the exact location and period of the cyclic trajectory, as a function of s, is difficult, a tedious iterative solution of the equations (2), made feasible by the computer algebra system Maple [8], shows that for large s, the trajectory passes close to the points $\pm(2s-1, 1)$, $\pm(1, -2s+1)$. and its period grows as $4 \ln 2s + O(s^{-1})$. In particular, then, the oscillation of o_1 and o_2 may be made arbitrarily slow by increasing the parameter s — although the period does grow only logarithmically in s.

In the analysis of the discrete network simulation we shall proceed piecewise, by considering the behavior of the continuous-time system separately within each linear region of the response function σ. The large amounts of calculation required by this brute-force approach have again been performed with the help of the Maple system. Even so, we shall make two simplifying assumptions. First, we only consider a single update step of a single pair of u units, where the units move, as a response to their net inputs, from an initial state of $\sigma(u^+) = 1$, $\sigma(u^-) = -1$ to the state $\sigma(u^+) = -1$, $\sigma(u^-) = 1$ (as in the example considered above). This simplification is justified, because (i) all the u units change state synchronously, so looking at one pair suffices; (ii) the opposite move from $\sigma(u^+) = -1$, $\sigma(u^-) = 1$ to $\sigma(u^+) = 1$, $\sigma(u^-) = -1$ is symmetric, and thus does not need to be considered; (iii) any move where the states of the units stay unchanged is more robust than the one considered, because then the unit with output 1 stays continually ahead in the competition: it both has a larger initial value, and receives excitatory, as opposed to inhibitory, input from other units; and finally (iv)

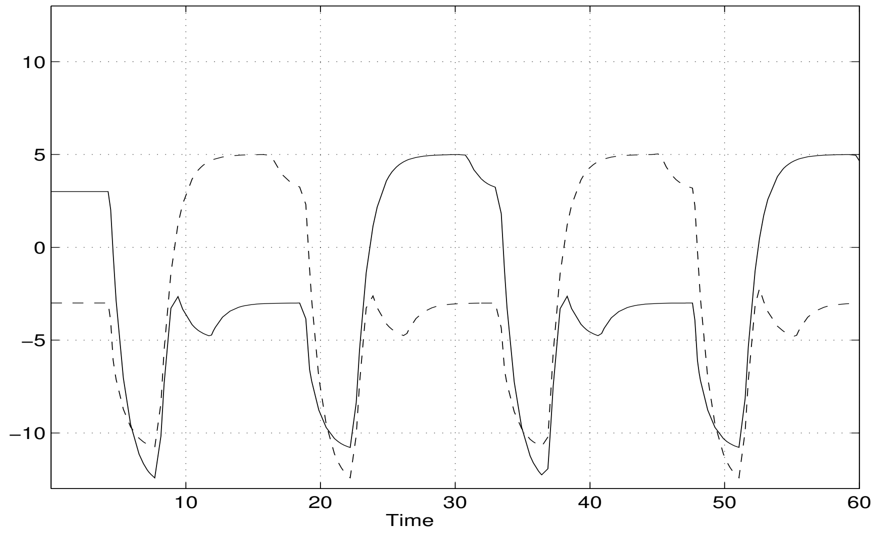

Figure 10: Internal states of the units u^+ and u^-.

the v units are similar to u units with a single excitatory input connection, so they need not be considered separately.

The second assumption we make is that the *reset* pulses are sharp, i.e. each pulse switches from -1 to 1 and back at precisely defined moments, instead of making a continuous transition. This simplification can be justified by observing that (i) slow rise times don't actually matter, as long as the pulse stays high for sufficiently long to effect the intended reset; and (ii) although the critical competition between $+/-$ unit pairs is initiated during the falling phase of the reset pulse, the tail of the pulse affects both members of a pair uniformly, and so a slow fall only slows down the competition without affecting its outcome. What *is* important, however, is that the quiescent period between the *reset* signals is sufficiently long for the $+/-$ unit pairs to converge sufficiently close to their new limiting values. A Maple analysis of the oscillator and reset signal equations (2) and (3) shows that the rise and fall times of the reset pulses are asymptotic to $2/q + O(q^{-2})$, and the high and quiescent times are asymptotic to $\ln 2s + O(s^{-1} + q^{-1})$. Thus, the rise and fall times can be made arbitrarily short and the high and quiescent times arbitrarily long by adjusting the parameters s and q appropriately.

With these assumptions in place, we shall now proceed to consider the

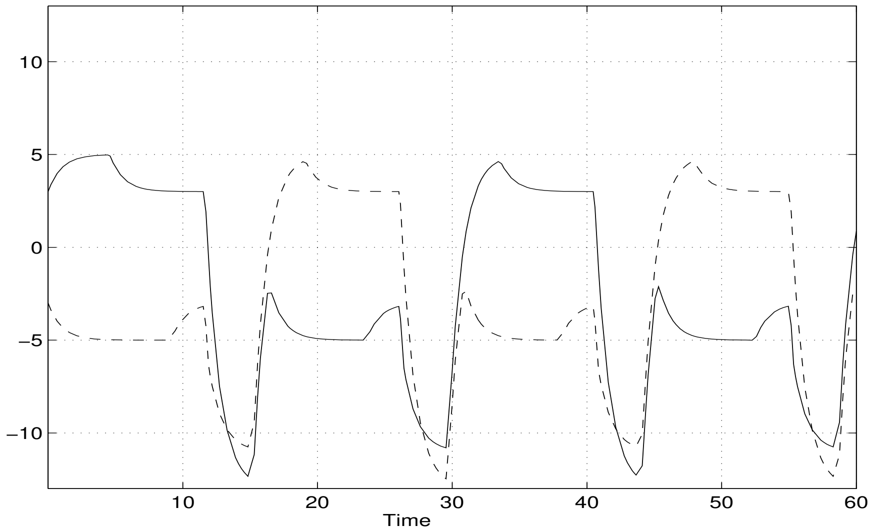

Figure 11: Internal states of the units v^+ and v^-.

unit equations (1). As discussed above, initially $\sigma(u^+) = 1$, $\sigma(u^-) = -1$, and unit u^+ receives from the rest of the network some total input of $-h < 0$, whereas u^- receives a net input of $h > 0$. Assuming that both of the reset signals are initially off, the equations governing the pair of units are:

$$\begin{aligned} \dot{u}^+ &= -u^+ - h + c, \\ \dot{u}^- &= -u^- + h - c. \end{aligned}$$

One can thus see that if the system is not perturbed, and $c \gg h$, unit u^+ tends to value $-h + c \gg 0$, and unit u^- tends to value $h - c \ll 0$. To simplify the formulas, let us fix a specific value for the competition strength parameter c, say $c = 4w$, where $w \geq 1$ is the maximum total net input to any of the u units. Also, we might just as well assume that $h = 1$, because this is the minimum possible bias for u^- over u^+: for any $h > 1$ the system behaves even more robustly than analyzed below.

Recapitulating, then, in the initial condition the unit u^+ tends to value $c - h = 4w - 1$, and the unit u^- tends to value $-c + h = -4w + 1$. We shall assume that initially the system has had sufficient time to stabilize so that $4w - 2 \leq u^+(0) \leq 4w - 1$ and $-4w + 1 \leq u^-(0) \leq -4w + 2$; and we shall show that after a sequence of six update phases, the corresponding

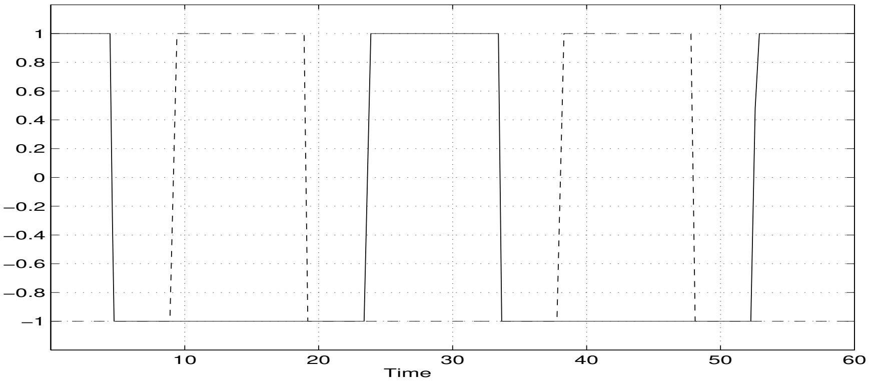

Figure 12: Output signals of the units u^+ and u^-.

opposite situation will be reached, so that $-4w - 1 \le u^+(T_6) \le -4w$ and $4w \le u^-(T_6) \le 4w + 1$.

Phase 1: At some time $T_0 \ge 0$ the $reset_u$ signal turns on. Again, to simplify the equations, we shall fix a definite value $r = 2c = 8w$ for the strength of the reset connection. Thus, at time T_0 the equations governing the unit pair change to (cf. equation (4)):

$$\begin{aligned} \dot{u}^+ &= -u^+ - h + c - 2r &&= -u^+ - 1 - 12w, \\ \dot{u}^- &= -u^- + h - c - 2r &&= -u^- + 1 - 20w. \end{aligned}$$

Solving this system of linear first-order o.d.e.'s (with the help of the Maple system) with the initial conditions $4w - 2 \le u^+(T_0) \le 4w - 1$, $-4w + 1 \le u^-(T_0) \le -4w + 2$ shows that the value of u^+ reaches 1 at some time $T_1 = T_0 + t_1$, where $\ln \frac{16w-1}{12w-2} \le t_1 \le \ln \frac{16w}{12w-2}$. At this time variable u^- has a value bounded by $-8w + 3 \le u^-(T_1) \le -\frac{128w^2-68w-1}{16w-1}$. Approximately, then, u^+ decreases from about $4w$ to 1 in roughly time $t_1 = \ln 4w$, and in this time u^- decreases from about $-4w$ to between $-8w+3$ and $-8w+5$. Because we wish to show that in the eventual competition between the units, u^- will win and obtain a positive value, and u^+ will obtain a negative value, we are mainly interested in lower bounds on u^- and upper bounds on u^+: to keep from cluttering the presentation we shall in the sequel list only these bounds.

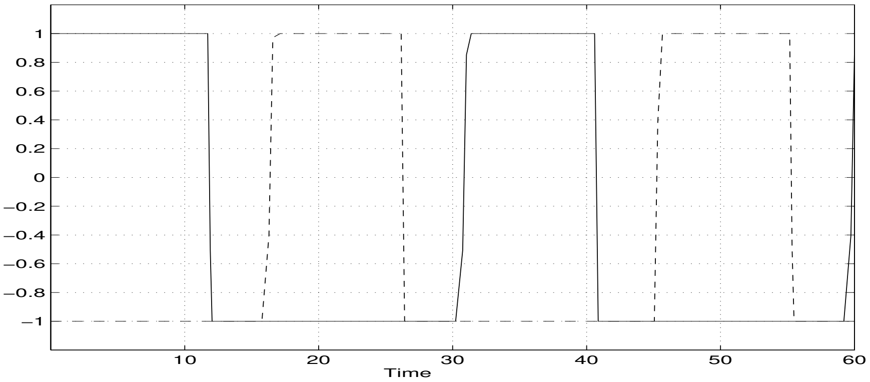

Figure 13: Output signals of the units v^+ and v^-.

Phase 2: At time T_1 the system equations change again, because the value of u^+ enters the region where the response function is $\sigma(u^+) = u^+$. The new equations are

$$\begin{aligned} \dot{u}^+ &= -u^+ - 1 - 12w, \\ \dot{u}^- &= -u^- + 1 - 4wu^+ - 16w. \end{aligned}$$

Solving this system again with the help of Maple, one obtains for the time when u^+ reaches -1 the value $T_2 = T_1 + t_2$, where $t_2 = \ln \frac{6w+1}{6w} \approx \frac{1}{6w}$. At this time $u^-(T_2) \geq -8w + 1$.

Phase 3: At time T_2 the system equations change to

$$\begin{aligned} \dot{u}^+ &= -u^+ - 1 - 12w, \\ \dot{u}^- &= -u^- + 1 - 12w, \end{aligned}$$

so that as long as the $reset_u$ signal stays on, the unit states approach exponentially the values $u^+ = -12w - 1$, $u^- = -12w + 1$. We shall assume that the clock subnetwork oscillates so slowly that the reset signal stays on for at least an additional time of $t_3 = \ln 2r = \ln 16w$. One obtains then at time $T_3 = T_2 + t_2$ the bounds $u^+(T_3) \leq -12w$, $u^- \geq -12w + 1$.

Phase 4: At time T_3 the $reset_u$ signal switches off, and the system equations change to

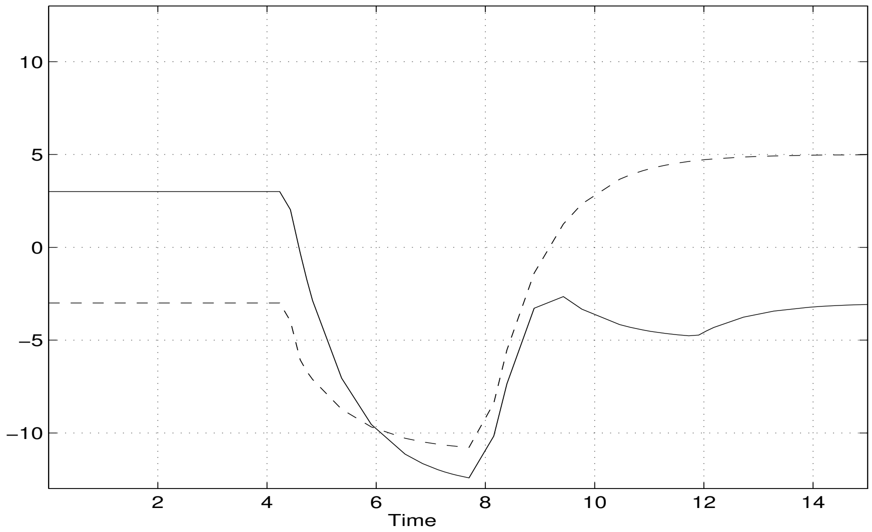

Figure 14: Expanded view of the internal states of the units u^+ and u^-.

$$\begin{aligned}\dot{u}^+ &= -u^+ - 1 + 4w,\\ \dot{u}^- &= -u^- + 1 + 4w.\end{aligned}$$

Thus the values of both u^+ and u^- start to increase, and one can see that u^- reaches -1 sooner than u^+. (It both has a larger initial value, and rises more steeply than u^+.) More precisely, $u^-(T_4) = -1$ at some time $T_4 = T_3 + t_4$, where $t_4 \leq \ln(4 - \frac{2}{w+1})$, and at this time $u^+(T_4) \leq -\frac{5}{2}$.

Phase 5: It is now already clear that unit u^- will win the competition, but let us nevertheless follow the system for a few more phases. At time T_4 the system equations become

$$\begin{aligned}\dot{u}^+ &= -u^+ - 1 - 4wu^-,\\ \dot{u}^- &= -u^- + 1 + 4w,\end{aligned}$$

and they have this form until either u^- reaches 1 or u^+ reaches -1. Actually, one can verify that the latter never happens, and when u^- reaches 1, at time $T_5 = T_4 + t_5$, where $t_5 = \ln(w + \frac{1}{2w})$, the value of u^+ is still less than -2.

Phase 6: At time T_5 the system equations change to

$$\begin{aligned}\dot{u}^+ &= -u^+ - 1 - 4w,\\ \dot{u}^- &= -u^- + 1 + 4w,\end{aligned}$$

so that in the limit u^- converges to the value $4w+1$, and u^+ converges to $-4w-1$. Assuming again that the system has at least time $t_6 = \ln 2r = \ln 16w$ to stabilize before the $reset_v$ signal turns on, the values at time $T_6 = T_5 + t_6$ satisfy $u^-(T_6) \geq 4w$, $u^+(T_6) \leq -4w$, as desired. (In fact, in time $\ln 16w$ both variables will converge to within $\frac{1}{4}$ of their asymptotic values.)

Adding up all the transition times one obtains for the total time $T_6 - T_0$ the bound

$$\begin{aligned}T_6 &\leq \ln 4w + \frac{1}{6w} + \ln 16w + \ln(4 - \frac{2}{w+1}) + \ln(w + \frac{1}{2w}) + \ln 16w\\ &\approx 4\ln 8w.\end{aligned}$$

To guarantee correct behavior, the clock network should oscillate with a period of at least twice this bound (recall that in our somewhat inefficient simulation technique we must update both the u and v units during a single clock cycle). Thus, we should choose for the parameter s a value such that $4\ln 2s \geq 8\ln 8w$, i.e. $s \geq 32w^2$. However, this is assuming the reset pulses are perfectly sharp: theoretically one should choose for s a somewhat larger value, to compensate for the nonzero rise and fall times. On the other hand, in the example simulation in Section 3, we used successfully the value $s = 16$ instead of a larger value $s \geq 32$, as would be suggested by these calculations.

5 Conclusion and Open Problems

We have shown that continuous-time neural networks based on Hopfield-type units with saturated-linear amplifiers are universal computational devices, in the sense that (sequences of) networks with polynomially many units are capable of simulating polynomially space-bounded Turing machines. Some of the many open questions suggested by this work are the following.

Our present simulation construction is somewhat unsatisfying because of its heavy reliance on the clock pulses provided by the oscillator subnetwork. It would be most interesting to develop computation and analysis techniques for nonoscillating networks. One especially interesting case where at least infinite undamped oscillations are precluded are networks with symmetric interconnections. In the discrete-time case also symmetric networks are capable of efficient Turing machine simulation [17], but the continuous-time case seems to be rather more complicated. Apparently not even simple convergence time bounds for symmetric continous-time networks are known. (For discrete symmetric networks such bounds were first obtained in [7], and the convergence behavior of such networks is by now well understood.)

A technical issue concerns more general response functions. Our computer simulations indicate that networks with, e.g., *tanh* nonlinearities have qualitatively the same behavior as networks using the saturated-linear response function. However the piecewise-linear analysis of network behavior used above obviously doesn't extend to this case.

Finally, we have only obtained a *lower* bound on the computational power of continuous-time networks, and one may also inquire about the corresponding *upper* bounds. For comparison, note that in the discrete-time case Siegelmann and Sontag [21] have shown that any Turing machine can be simulated uniformly for all input lengths by a *single* network with saturated-linear response functions and arbitrary-precision real number states. Also, several simulations of arbitrary Turing machines by continuous-time and finite-dimensional, but non-network-like systems are known (e.g. [2, 5, 15, 16]).

References

[1] Anderson, J. A., Rosenfeld, E. (eds.) *Neurocomputing: Foundations of Research.* The MIT Press, Cambridge, MA, 1988.

[2] Asarin, E., Maler, O. On some relations between dynamical systems and transition systems. *Proc. 21st Internat. Colloq. on Automata, Languages, and Programming (Jerusalem, Israel, July 1994),* 59–72. Springer-Verlag, Berlin, 1994.

[3] Balcázar, J. L., Díaz, J., Gabarró, J. On characterizations of the class PSPACE/poly. *Theoret. Comput. Sci. 52* (1987), 251–267.

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

[5] Branicky, M. Analog computation with continuous ODEs. *Proc. Workshop on Physics and Computation 1994 (Dallas, Texas, Nov. 1994),* 265–274. IEEE Computer Society Press, Los Alamitos, CA, 1994.

[6] Branicky, M. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoret. Comput. Sci. 138* (1995), 67–100.

[7] Fogelman, F., Goles, E., and Weisbuch, G. Transient length in sequential iterations of threshold functions. *Discr. Appl. Math. 6* (1983), 95–98.

[8] Char, B. W, Geddes, K. O., Gonnet, G. H., Leong, B. L., Monagan, M. B., Watt, S. M. *First Leaves: A Tutorial Introduction to Maple V.* Springer-Verlag, New York, NY, 1992.

[9] Hirsch. M. W., Smale, S. *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, San Diego, CA, 1974.

[10] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci. USA 79* (1982), 2554–2558. Reprinted in [1], pp. 460–464.

[11] Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Nat. Acad. Sci. USA 81* (1984), 3088–3092. Reprinted in [1], pp. 579–583.

[12] Karp, R. M., Lipton, R. J. Turing machines that take advice. *L'Enseignement Mathématique 28* (1982), 191–209.

[13] Lepley, M., Miller, G. Computational power for networks of threshold devices in an asynchronous environment. Unpublished manuscript, Dept. of Mathematics, Massachusetts Inst. of Technology, 1983.

[14] *MATLAB Reference Guide.* MathWorks Inc., Natick, MA, 1992.

[15] Moore, C. Unpredictability and undecidability in physical systems. *Phys. Rev. Lett. 64 (1990),* 2354–2357.

[16] Moore, C. Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity 4* (1991), 199–230.

[17] Orponen, P. The computational power of discrete Hopfield nets with hidden units. To appear in *Neural Computation.* Preliminary version in *Proc. 20th Internat. Colloq. on Automata, Languages, and Programming (Lund, Sweden, July 1993),* 215–226. Springer-Verlag, Berlin, 1993.

[18] Orponen, P. Computing with truly asynchronous threshold logic networks. To appear in *Theoret. Comput. Sci.* Preliminary version available as Project NeuroCOLT Report NC-TR-95-038, Royal Holloway College, Univ. of London, Dept. of Computer Science.

[19] Pour-El, M. B., Richards, I. *Computability in Analysis and Physics.* Springer-Verlag, Berlin, 1989.

[20] Rubel, L. A. Digital simulation of analog computation and Church's Thesis. *J. Symb. Logic 54* (1989), 1011–1017.

[21] Siegelmann, H. T., Sontag, E. D. On the computational power of neural nets. *J. Comput. System Sciences 50* (1995), 132–150.

[22] Vergis, A., Steiglitz, K., Dickinson, B. The complexity of analog computation. *Math. and Computers in Simulation 28* (1986), 91–113.