PERGAMON

Microelectronics Reliability 41 (2001) 2051–2065

MICROELECTRONICS RELIABILITY

www.elsevier.com/locate/microrel

The Hi-NOON neural simulator and its applications ☆

R.I. Damper [a,*], R.L.B. French [a], T.W. Scutt [b]

[a] *Image, Speech and Intelligent Systems Research Group, Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK*

[b] *Core Design Ltd., 2 Roundhouse Road, Pride Park, Derby DE24 8JE, UK*

Received 20 July 2000; received in revised form 6 October 2000

Abstract

This paper describes the Hi-NOON (hierarchical network of object-oriented neurons) neural simulator, originally conceived as a general-purpose, computationally efficient, object-oriented software system for the simulation of small systems of biological neurons, as an aid to the study of links between neurophysiology and behaviour in lower animals. As such, the artificial neurons employed were spiking in nature; to effect an appropriate compromise between computational complexity and biological realism, modelling was at the transmembrane potential level of abstraction. Further, since real neural systems incorporate different types of neurons specialised to somewhat different functions, the software was written to accommodate a non-homogeneous population of neurons. The computational efficiency of Hi-NOON makes it eminently suitable for situated system studies (biological robotics, animats) where real-time operation is a pre-requisite. The flexibility which was a central design goal of Hi-NOON means that the system is also capable of modelling interconnections of non-spiking artificial neurons with continuous or piecewise linear activation functions. The efficacy of the simulator is illustrated with respect to some recent applications to situated systems studies. We also consider prospects for integrating Hi-NOON with a conventional circuit simulator in the future.

1. Introduction

In recent years, a convergence of two initially disparate threads of research exploring the links between neurophysiology and behaviour has occurred. In one particularly vibrant line of research, so-called parallel distributed processing [1], grossly simplified, artificial models of neural networks have been defined and studied, fuelled by the discovery of powerful learning algorithms such as error backpropagation [2,3]. The other thread has been the careful study of the functions of individual neurons within manageably small neural circuits in lower animals, such as the sea snail *Aplysia*, with a simple but interesting repertoire of behaviours [4,5]. The two threads – parallel distributed processing (PDP or 'connectionism') and systems neuroscience – meet in the relatively newer paradigm of computational neuroscience [6], which attempts to exploit their different strengths by linking some of the principles of connectionism with data from experimental neurophysiology. Such an approach allows an appropriate trade to be made between biological fidelity and computational expediency.

This paper describes a program originally designed to simulate small systems of neurons, within the computational neuroscience paradigm, and its more recent development and applications. The program is called Hi-NOON, which stands for hierarchical network of object-oriented neurons. As the name suggests, in Hi-NOON, synapses, neurons and networks are in principle represented as objects within an object-oriented hierarchy [7–9] at various levels of abstraction. (We say "in principle" because the latest version of the code is written in object-oriented style but not in an object-oriented language.) The lowest such level uses the membrane potential (strictly, transmembrane potential difference) as the observable parameter in the network model. This

☆ An earlier version of this paper was printed in the Proceedings of the 1st Small Systems Simulation Symposium (SSSS 2000), 4–5 September 2000, Niš, Yugoslavia.

* Corresponding author. Fax: +44-23-80-594-577.

E-mail address: rid@ecs.soton.ac.uk (R.I. Damper).

0026-2714/01/$ - see front matter
PII: S0026-2714(01)00097-X

is a much lower-level approach than the use of activation values very roughly corresponding to the rate of firing of individual neurons or collections of neurons as in PDP models. By contrast, Hi-NOON retains details of the generation of each individual 'spike' or action potential (AP) which is lost in the traditional connectionist approach. As well, Hi-NOON facilitates simulation of a non-homogeneous population of neurons. This allows different, higher levels of abstraction to be used also in a 'mixed mode'. Most obviously, PDP-type neurons modelled at the level of activation could be mixed with more biologically realistic spiking neurons, in which spike generation is stochastic. Thus, although one would be exercising only a proportion of its flexibility and power, one could even use Hi-NOON as a highly conventional artificial neural network simulator like PDP++ (see `http://www.cnbc.cmu.edu/PDP++/PDP++.html`).

Since the latter is a well-worn path, however, we concentrate in this paper on the less usual simulation of spiking behaviour. One might reasonably ask what advantages this might offer, i.e. what can a simulation based on spiking neurons achieve that cannot be achieved using more gross PDP-type model neurons? This is currently a vexed question in computational neuroscience, and a fully definitive answer cannot be given at this stage. It is likely that spikes evolved mainly as a means of regenerative signalling along the relatively long propagation paths of animal nervous systems [10]. Because of active regenerative processes, the action potential is propagated without loss of amplitude; hence, no information is carried by signal amplitude. Accordingly, time of firing plays a central role in neural coding, and this strongly suggests that a circuit of spiking neurons ought to be better able to handle temporal processing than PDP nets. The latter are most naturally thought of as discrete-time systems with synchronous update of neurons via connections involving a unit time delay. As a consequence, such nets are poorly suited to the processing of dynamic, temporal sequences [11] which assume extreme importance in many engineering applications. In the words of Port et al. [12]:

> "In standard feedforward backpropagation networks,... processing is seen as the sequential transformation, from one layer to the next, of static representations. Such networks are little more than ... devices for mapping static inputs into static outputs. No dynamics or temporal considerations are deployed in understanding the behavior of the network ..."

Clearly then, detailed timing information for individual spikes, and relative timing between spikes, offers an additional dimension to the neural code, as does the stochastic aspect. There is suggestive evidence that this sort of information is indeed important in biology. Citing Rieke et al. [13, p. 279]:

> "...under many conditions, behavioural decisions are made with of order one spike per cell, ... individual spikes can convey several bits of information about incoming sensory stimuli ... precise discriminations could ... be based on the occurrence of individual spikes ..."

Hence, one motivation for this work is to provide a simulation environment in which questions such as the relative merits of spiking and non-spiking neural nets can be explored.

The remainder of this paper is structured as follows. In the next section, we give an overview of the Hi-NOON implementation before presenting details of the model neurons and synapses. We then demonstrate the capability of Hi-NOON to deal with non-spiking neurons (described by an activation function) and outline some recent applications of the simulator to studies of real-time, situated systems. Finally, we consider the potential for integrating Hi-NOON with a conventional circuit simulator, before concluding.

2. An overview of Hi-NOON

The original simulation program was written in object-oriented PASCAL [7–9], but has subsequently been rewritten in C using the disciplines of object-oriented programming (OOP) [14,15]. C was used (rather than C++ with its explicit support of OOP features) to maximise portability among various realisations in different applications. The benefits of the OOP approach are twofold. First, the ability for objects to inherit properties from other objects means that it is easy to define more physiologically exact neurons in terms of simpler neurons. Thus, the system allows a simple threshold unit as the most basic type of object. More complex objects inherit certain properties from this object (e.g. the fact that it has weighted connections to other objects). The second benefit of OOP is polymorphism. This means that the network may contain many different types of neuron, at many levels of complexity, without the programmer having to be concerned with this.

Code for the C version of Hi-NOON is available by anonymous `ftp` from directory `pub/users/rid/hinoon` at `ftp.isis.ecs.soton.ac.uk`.

2.1. Neuron parameters

Basic neurophysiology suggests the attributes a model spiking neuron should have. The fixed parameters `BaseMP`, `Threshold` and `TimeConst` correspond to the resting potential, threshold and time constant of the

neuron, respectively. Dynamic parameters `MP`, `SynPot` and `fired` (a 1/0 predicate) model the actual membrane potential as it varies in time, accumulate the weighted sum of synaptic inputs which influence the updating of `MP` at the next time step, and indicate if the object is in the process of firing, respectively. This parameter system allows us easily to describe differences between neurons and to keep track of the changing states of neurons over time. It approximately satisfies Selverston's "minimum requirements" [16] for effective neural modelling.

2.2. *Hi*-NOON objects

The neural network is held as a list of objects, where each such object corresponds to a single neuron and holds all the information about its state (see below) and about subsidiary objects. The information held in the neuron object is comprised of sets of:

- parameters which define the neuron;
- data structures which define the 'axon terminals' for the neuron, each of which is itself an object and has its own parameters;
- methods – pointers to functions – which access and alter parameter values and so determine exactly how the neuron functions.

The top-level list corresponds to the network object. This possesses two methods (called `h_access` and `add`) for accessing network objects and adding further objects onto the list, respectively. Simulation run length is handled by a global object. This stores the simulation and concurrent socket interface 'housekeeping' data, including a counter whose original value specifies the length of simulation. It decrements after each evaluation of the network object, and the simulation halts when the counter reaches zero.

As synapses are also objects, they too have fixed and dynamic parameters similar to those of neurons. Thus, `BaseWeight` is the default weight of the synapse and is a constant; `Weight` holds the present synaptic strength and is variable during simulation; `Recovery` is a constant (within each synapse) which determines how quickly `Weight` returns to `BaseWeight`. To prevent synaptic weights growing without limit, `Weight` is bounded during simulation (see Section 3.2.6).

2.3. Neuron types

Hi-NOON allows a non-homogeneous population of neurons to be simulated – reflecting the fact that neurons have specialised functions in real neurobiological systems – at the most appropriate level of abstraction. Modelling individual neurons at the level of membrane potential allows sub-threshold and spiking behaviours to be simulated in real time at low computational cost. The fixed parameters cater for differences between neurons which, in this work, are of the following types:

basic: tells its synapses to fire when its membrane potential crosses threshold from below.

noisy: similar to *basic*, but has an additional internal noise component determining the weighted synaptic input, and hence influencing the membrane potential at the next time step.

ramp: similar to *noisy*, but has the ability to ramp up spike generation rate. It is used as a test signal source in network development.

burst: similar to *noisy* but produces a short burst of spikes when its membrane potential crosses threshold.

sensor: similar to *basic*, but acts as a sensory neuron in a situated system, such as a mobile robot.

motor: similar to *basic*, but acts as a motor neuron in a situated system.

2.4. State system

Each neuron is treated as being in one of a number of six states depending on the present membrane potential, cell threshold and whether or not the cell has just fired, etc. For example, if the membrane potential of the basic cell is above threshold, and the cell has not just fired, then the neuron will start to generate a spike and will initiate synaptic transmission.

Fig. 1 (taken from a Hi-NOON simulation) shows the states passed through by a neuron during firing of an action potential. In the case illustrated, the minimum, resting and peak potentials of the neuron are set at −69, −60 and +45 mV respectively, and the threshold value was −45 mV. Note that actual values will under/overshoot these settings before state can change at the next iteration of simulation. The states are:

A: `MP` above resting potential and below threshold
B: above threshold and below peak
C: at peak
D: post-firing
E: at minimum
F: hyperpolarised

The equations governing the membrane potential in each of these states and the synaptic weights are given in Section 3.

The use of a state system for controlling the membrane potential facilitates the addition of new features to the program; it is only necessary to identify which of the states may trigger this feature and to add a procedure call at that particular state. This, coupled with OOP's inheritance, allows models to be developed and altered relatively easily. Also, the compromise between between

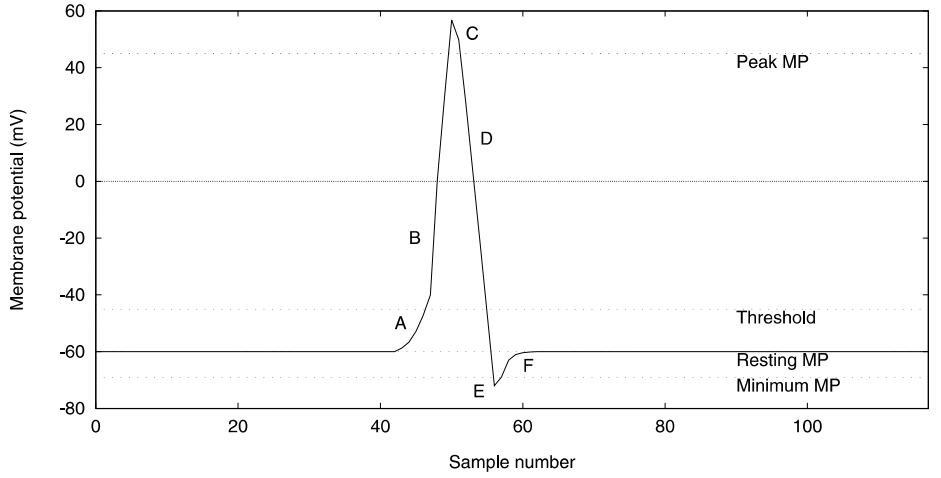


Fig. 1. Time evolution of typical action potential (spike) of a *basic* neuron in a Hi-NOON simulation. See text for specification of the states (A . . . F) passed through by a neuron during firing. Here, the sample period is approximately 4 ms (this varies with the machine on which the simulation runs).

computational complexity and biological realism implicit in the state-system algorithm allows for real-time operation of Hi-NOON. This is an essential pre-requisite for situated systems studies as described below (Section 4).

2.5. Axonal and synaptic transmission

Our neurons model sub-threshold behaviour but sub-threshold potentials are not propagated (from axon hillock to terminal fibres) in real neurons, only action potentials are. We do not attempt to model (regenerative) spike transmission along the axon realistically, as a spatio-temporal process. This, however, is not a serious concern because the model's behaviour depends entirely on how pre-synaptic activity is transformed into post-synaptic activity. It is only in supra-threshold states B, C and D (see Fig. 1 and Section 3.2) that synaptic communication can take place. Hence, it is irrelevant that we are, in some sense, modelling sub-threshold behaviour incorrectly. An alternative view is that, because we are not modelling axonal transmission, we have 'point' neurons as is common in neural modelling [17–20]. This is a very useful abstraction in which the model neuron is entirely localised to a point in space.

2.6. Learning in Hi-NOON

There is no specific support for learning in Hi-NOON. Thus, if PDP-type learning (e.g. backpropagation) is to be used, this must be implemented external to the simulator. In light of Hi-NOON's ability to model at the level of transmembrane potential, however, there is implicit support for biologically based forms of learning, such as habituation, sensitisation and classical conditioning [5,21,22]. Generally, these simple forms of learning are implemented using synapse-on-synapse connections in Hi-NOON.

Neurons and synapses can be connected together in five ways: as excitatory, inhibitory, habituating, sensitising, or conditioning connections.

Excitatory, inhibitory and habituating connections: The neural model for these is mono-synaptic and is implemented as a positive synaptic weight, a negative synaptic weight, and a synaptic weight whose magnitude decreases every time it fires, respectively. The mono-synaptic model is shown in Fig. 2.

Sensitising and conditioning connections: These are implemented with synapse-on-synapse connections as illustrated in Fig. 3. A sensitising synapse (from a facilitatory interneuron `I`) as in Fig. 3(a) increases the magnitude of the weight of its target synapse (from neuron `A` to neuron `B`), irrespective of whether the target has fired. For the case of the conditioning synapse shown in Fig. 3(b), the facilitatory interneuron codes the unconditioned stimulus, `US`. In this case, firing of the interneuron strengthens its target – the synapse between the conditioned stimulus neuron `CS` and the unconditioned response neuron `UR` – by an amount which is a function of the elapsed time since firing of the target (see Fig. 4

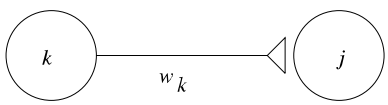


Fig. 2. Basic model of excitatory, inhibitory and habituating synaptic connections from parent neuron k to target neuron j, with weight w_k.

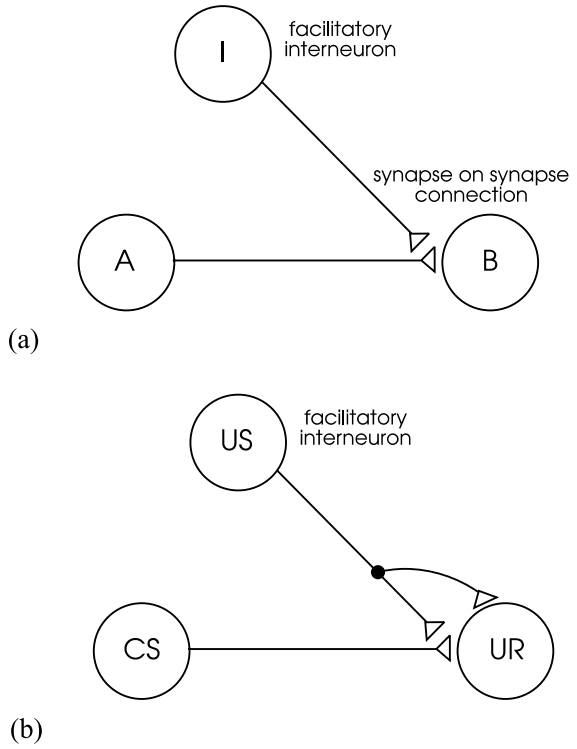


Fig. 3. (a) Sensitisation and (b) classical conditioning are modelled using synapse-on-synapse connections within the Hi-NOON simulation.

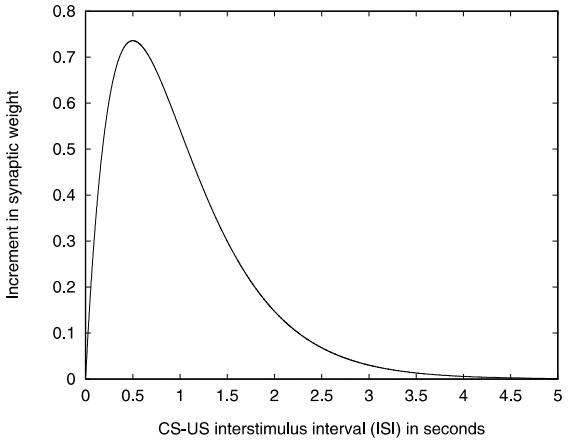


Fig. 4. Increment in synaptic weight through conditioning as a function of the interval between conditioned and unconditioned stimuli.

and Eq. (2)). The maximum strengthening occurs when a target has fired half a second before the conditioning synapse.

These are the building blocks from which we can construct a nervous system.

3. Neurons and synapses

In this section, we present more detailed descriptions of neurons and synapses within Hi-NOON. Since Hi-NOON is intended for (amongst other things) applications in situated robotics studies, there is provision for sensory and motor neurons which connect to the environment, as well as for more prosaic 'basic' (information processing) neurons.

3.1. Neurons

The 'basic' neuron type has the state system functionality which is subsequently embedded in all derivatives, such as the sensory and motor cells.

3.1.1. Basic neurons

Updating equations for the membrane potential (MP in mV) for this neuron type are:

$$\begin{aligned}
&\text{state A}: && \mathrm{MP}(t+1) = \mathrm{MP}(t) - \tau + S(t)\\
&\text{state B}: && \mathrm{MP}(t+1) = \mathrm{MP}(t) - \alpha + S(t)\\
&\text{state C}: && \mathrm{MP}(t+1) = h + S(t)\\
&\text{state D}: && \mathrm{MP}(t+1) = \mathrm{MP}(t) - \mu + S(t)\\
&\text{state E}: && \mathrm{MP}(t+1) = l + S(t)\\
&\text{state F}: && \mathrm{MP}(t+1) = \mathrm{MP}(t) + \frac{\mathrm{BaseMP} - \mathrm{MP}(t)}{\eta} + S(t)
\end{aligned}$$

where:

$$S(t) = \sum_i w_i \kappa(\mathrm{MP}_i(t) - \mathrm{BaseMP}_i) \qquad (1)$$

is the synaptic potential (`SynPot`), i is a counter which counts over active pre-synaptic cells, w_i is the synaptic weight from a pre-synaptic neuron, τ is the neuron time constant, $\eta = 1.5$ is the post-undershoot increment rate, $\mu = 25$ is the post-action potential peak-MP decrement, $\kappa = 1/450$ is a heuristically set constant, $\alpha = 20$ is the post-threshold attack increment, $h = 45$ is the post-threshold maximum MP, and $l = -69$ is the pre-undershoot minimum MP.

Certain of the above parameters (e.g. τ, η) are time dependent and have been set empirically to suit a range of processor speeds and implementations. However, they may be inappropriate in some circumstances (as when implementing a real-time robotic system using a fast processor).

3.1.2. Sensory and motor neurons

These neuron types are important in the specific case of a robotic system which needs input and output from/to its environment. Since, in this paper, we are principally concerned with more general principles, we omit details of these neuron types here.

3.2. Synapses

The basic synapse (which is noise free) has functionality which is subsequently embedded in all derivatives such as the habituating, sensitising and conditioning types used in our ARBIB robot work (see below). These allow us to implement a simple, biologically based form of learning.

$$w(t) = \begin{cases} w(t) - \beta & \text{if } w(t) > w_{\text{base}} \\ w(t) + \beta & \text{if } w(t) \leqslant w_{\text{base}} \\ w_{\max} & \text{if } w(t) > w_{\max} \\ w_{\min} & \text{if } w(t) < w_{\min} \\ w_{\min} & \text{otherwise} \end{cases}$$

where β is the MP recovery parameter and w_{base} is the base weight (typically 0). These are individually set (together with $w_{\min}$ and $w_{\max}$, typically ± 16) for each neuron.

3.2.1. Noise-free synapse

$$\texttt{fired}(t) = \begin{cases} \text{TRUE} & \text{if state B, C, D} \\ \text{FALSE} & \text{otherwise} \end{cases}$$

3.2.2. Noisy synapse

$$\texttt{fired}(t) = \begin{cases} \text{TRUE} & \text{if } \texttt{condl} \\ \text{FALSE} & \text{otherwise} \end{cases}$$

where `condl` is state B, C, D, as for the noise-free synapse, ANDed with:

$$\frac{\text{MP}_{\text{p}} - \theta_{\text{p}}}{h - \theta_{\text{p}}} \times 100 \geqslant \text{rand mod100}$$

and p denotes a parent (pre-synaptic) neuron.

3.2.3. Habituating type

$$w(t+1) = \begin{cases} w(t) - d & \text{if state C} \\ w(t) & \text{otherwise} \end{cases}$$

where d is a constant decrement (typically ~ 1).

3.2.4. Sensitising type

$$w(t+1)_{\text{targ}} = \begin{cases} w(t)_{\text{targ}} + w(t)_{\text{sos}} & \text{if } \texttt{cond2} \\ w(t)_{\text{targ}} & \text{otherwise} \end{cases}$$

where `cond2` is $\texttt{fired}_{\text{targ}} \wedge \texttt{fired}_{\text{sos}}$, 'targ' denotes the target synapse (to be sensitised) and 'sos' denotes the synapse-on-synapse influence.

3.2.5. Conditioning type

$$w(t+1)_{\text{targ}} = \begin{cases} w(t)_{\text{targ}} + kw(t)_{\text{sos}} & \text{if } \texttt{cond2} \\ w(t)_{\text{targ}} & \text{otherwise} \end{cases}$$

where:

$$k = \frac{nT}{\psi} \exp\left(\frac{-nT}{\varsigma}\right) \tag{2}$$

and nT is a count of sample periods initiated by encountering state C for the target neuron, ψ (=250) is an empirically set scaling factor and ς (=500) is a constant chosen to maximise the effect of conditioning when the conditioning stimulus precedes the unconditioned stimulus by 0.5 s, as depicted in Fig. 4.

3.2.6. Autonomous decay of synapses

To prevent synaptic weights growing without limit, `Weight` is bounded during simulation. This models the finite size of stores of neurotransmitter in the synaptic terminals of real biological neurons. Also, according to Sutton and Barto [23]:

> "If it is assumed that synaptic strength slowly decays in the absence of a reinforcement signal, then a bound on weight size is imposed that is a function of reinforcement level and the decay rate ...In system theoretic terms, the adaptive element has *definite memory*: it cannot remember anything that occurred arbitrarily far in the past."

Sutton and Barto call this *autonomous decay*. This decay mechanism is also implemented here (via `Recovery`).

4. Applications

In this section we examine three examples of Hi-NOON in action. First, the well-known XOR problem is considered, and we show that Hi-NOON can be used to implement its solution with spiking as well as non-spiking neurons. Second, we describe work with two situated systems: our autonomous robot called ARBIB [24–26], and finally the 'cricket' robot of Webb and Scutt [27].

4.1. Spiking and non-spiking neurons

This subsection demonstrates the ability of Hi-NOON to support both spiking and non-spiking neuron models. As an example, we give two solutions to a well-known benchmark: the XOR problem.

4.1.1. Non-spiking XOR circuit

Fig. 5 shows the non-spiking circuit. Here, the flexibility of Hi-NOON has been exploited to define a (non-

standard) activation function – see below – that lends itself to solution of the XOR problem with a minimal number of neurons.

Two input neurons A and B make excitatory connections of weight 6 with an output neuron C, whose activation function is:

$$\mathrm{MP}(t+1) = \begin{cases} 0 & \text{if } S(t) < a \\ 0 & \text{if } S(t) > b \\ \frac{S(t)}{b} & \text{otherwise} \end{cases} \quad (3)$$

where $\mathrm{MP}(t+1)$ is the notional 'membrane potential' of this non-spiking neuron at the next time step, the parameters a and b are here set to 2 and 10 respectively, and $S(t)$ is the post-synaptic potential (the summed input) of the neuron as in Eq. (1). This activation function is depicted in Fig. 6.

If both A and B are inactive, then $S(t) = 0 < a$, so that C is set to zero activation by Eq. (3). If both A and B are active, then $S(t) = 12 > b$, again setting C to zero activation. However, if A and B have different activation values, then $S(t) = 6$. Hence, neither of the inequalities in Eq. (3) applies and C is given the activation value $6/10 = 0.6$. The operation of the circuit is summarised in the truth table of Table 1. Parameters a and b can be regarded as threshold and cut-off values respectively for neuron activation.

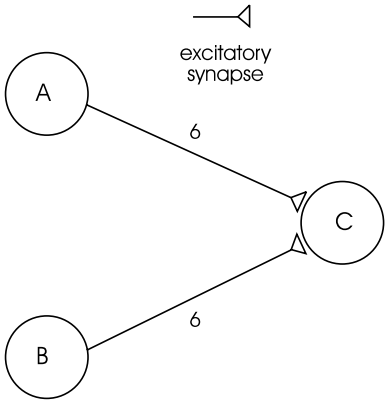


Fig. 5. XOR circuit using non-spiking neurons.

4.1.2. Spiking XOR circuit

Fig. 7 shows the spiking neuron circuit. Input neurons A and B make inhibitory connections with neuron C and thence to output neuron D. On first consideration, it might appear that this circuit is incapable of any active behaviour because of the inhibitory connections from input neurons to the output unit. However, the operation of this circuit depends upon a phenomenon called *post-inhibitory rebound* in cell C.

This simply means that after the membrane potential of C has been hyperpolarised for a short period of time, the cell becomes more excitable than usual. When the membrane is then allowed to return toward its normal resting potential, one or more action potentials may result [10]. In the Hi-NOON *basic* neuron model, post-inhibitory rebound is simply provided by setting a flag when the membrane potential becomes hyperpolarised. Hence, if the membrane potential returns to its resting level and this flag is set, then an action potential may be generated.

Fig. 8(a)–(c) shows the activity of input neurons A and B and of cell C, respectively. Because of the post-inhibitory rebound, cell C fires when one of A or B fires.

Table 1
XOR activation table

Inputs		Neuron activation
A	B	C
0	0	0
0	1	0.6
1	0	0.6
1	1	0

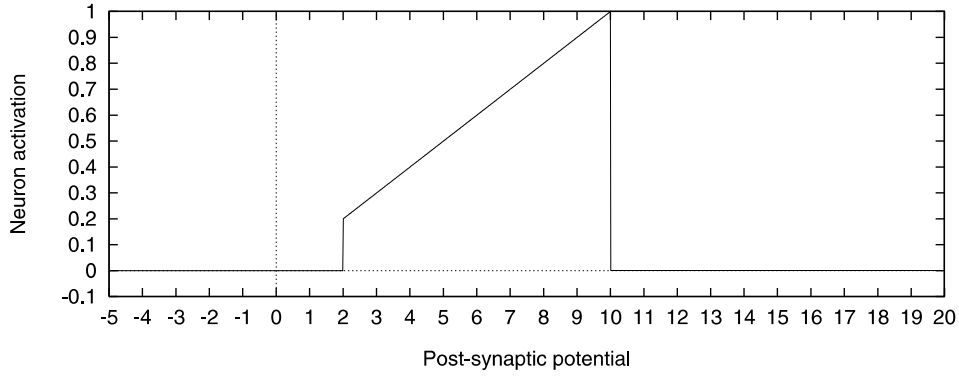


Fig. 6. Activation function for the non-spiking solution.

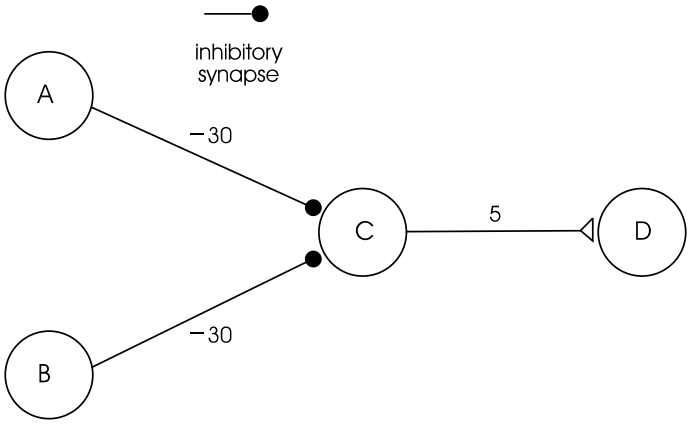


Fig. 7. XOR circuit using spiking neurons with post-inhibitory rebound.

However, it does not fire when both A and B are active because of the increased inhibition which prevents sufficient recovery of the membrane potential towards its resting value.

The problem with this as it stands, however, is that C *can* spike (weakly) when the inhibitory inputs from A and B are removed. This unwanted effect can be filtered out using an additional neuron. This is output cell D, which has a weak excitatory connection from C. Hence, D will only fire if activity in C is sufficiently intense, which is not so here (Fig. 8(d)).

4.1.3. Using spiking and non-spiking neurons together

Hi-NOON was principally designed to deal with spiking neural models. Obviously, the non-spiking model functions at a much higher level of abstraction than the spiking type: it has lost the ability to model individual action potentials as they develop through time. Hence, the state system algorithm requires less time to compute the activation of the non-spiking type than the activity in a spiking neuron, simply because the latter is a more detailed model. However, modelling spiking behaviour allows Hi-NOON to take advantage of the interspike interval for encoding and transmitting information between neurons in associative learning at the synapse level. Consequently, efficient systems can be conceived in which computational complexity and information coding ability (or biological realism if that is important in a specific case) are traded using both spiking and non-spiking neurons, depending upon their function in a network.

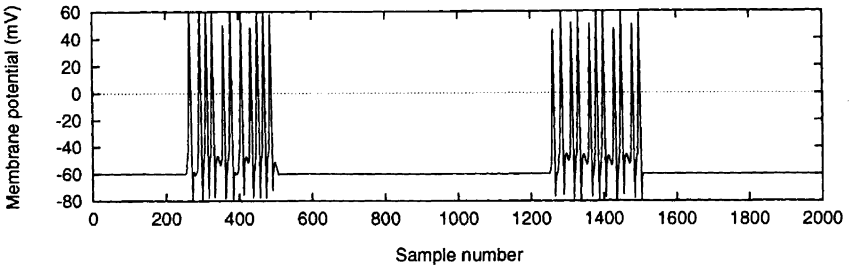
4.2. Situated systems

The Hi-NOON simulator has been used to design and implement the 'nervous systems' of two rather different situated systems. The first is the ARBIB autonomous robot [24–26] which has been implemented on a variety of hardware and software platforms. The second is a robot model of cricket phonotaxis behaviour by Webb and Scutt [27]. These can be considered as simple artificial creatures, or 'animats' to use Wilson's [28] term.

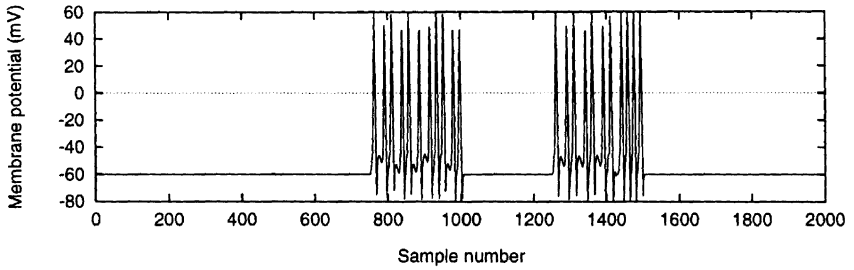
4.2.1. ARBIB

ARBIB is an autonomous robot based on inspirations from biology. It learns from and adapts to its environment, which consists of hard objects and light sources casting shadows. ARBIB's 'nervous system' was simulated in real time on Hi-NOON, as it explored its environment. A primary goal of this work was to test the notion that effective robot learning can be based on neural habituation and sensitisation, so validating the suggestion of Hawkins and Kandel [5] that (associative) classical and 'higher order' conditioning might be based on an elaboration of these (non-associative) forms of learning. Accordingly, the nervous system has a non-homogeneous population of spiking neurons. ARBIB's drive to explore its environment was provided by a simple central pattern generator neural circuit [29], and learning was by modification of a basic, pre-existing ('hard-wired') reflex to reverse and turn on hitting an obstruction. By monitoring firing rates of specific neurons and synaptic weights between neural connections as ARBIB learns, we have confirmed that both classical and higher-order conditioning occur, leading to the emergence of interesting and ecologically valid, obstacle-avoidance behaviours.

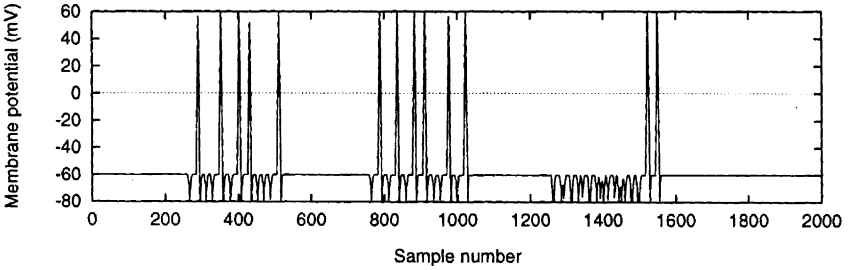
Synaptogenesis for a mobile robot: One problem with the initial ARBIB implementation was that its learning was almost entirely plastic. That is, it rapidly 'forgot' (by autonomous decay – see Section 3.2.6) what it had learned about its environment, which then had to be relearned. More recently, we have implemented a simple form of synaptogenesis within Hi-NOON [30], according to which new synapses may be created as shown in Fig. 9. These new synapses are created parallel to an existing conditioned synapse, so effectively increasing the strength of the latter and increasing the time that it



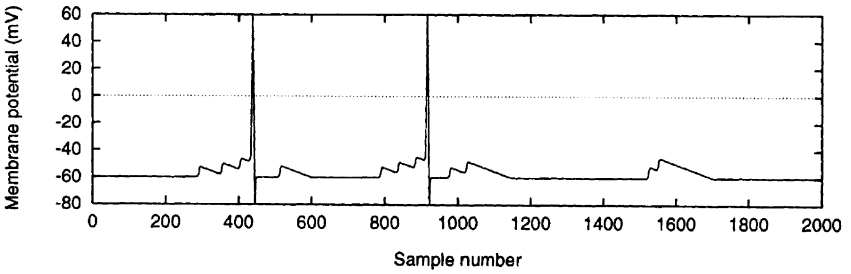
(a) Activity in spiking input neuron A.



(b) Activity in spiking input neuron B.



(c) Activity in spiking neuron C.



(d) Activity in spiking neuron D.

Fig. 8. (a–d) Firing activity in neurons A, B, C and D of the spiking solution, respectively. Neuron D fires when either A or B is active, but not both.

takes for it to decay autonomously. The creation process was constrained by introducing a new predicate into Hi-NOON: a new synapse is only created once the conditioned synaptic strength reaches some percentage of the allowed maximum. The newly created synapse has a strength calculated from the difference between the elapsed time of post-synaptic cell firing and elapsed time of conditioned synapse firing. Experiments showed that

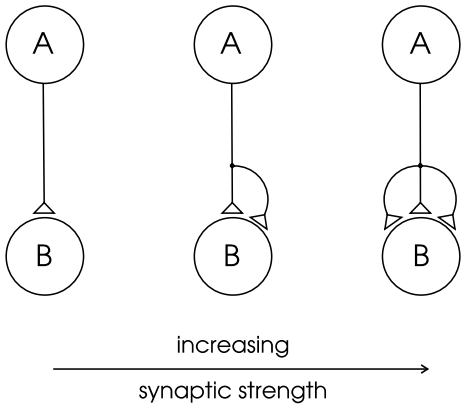


Fig. 9. Synaptogenesis leads to the growth of new synaptic connections, so strengthening the overall A → B connection weight.

this stabilises the learning to a useful degree, so offering a practical remedy to the stability–plasticity dilemma [31,32].

Testing synaptogenesis: A Nomad Scout 2 robot (see `http://www.robots.com`) controlled through the Nomadic Technologies *Nserver* software, via the Scout's *host* port, was used for testing the effectiveness of the synaptogenesis model. The Scout has six bump sensors (used as the unconditioned stimulus US) and 16 sonar devices (used as the conditioned stimulus CS) arranged around its circumference, shown in Fig. 10. In this instantiation of ARBIB, the infrared sensory neurons have been replaced by sensory cells that are coupled to the Scout's forward facing sonar devices and the delta light sensory neurons have been disabled.

A total of 14 runs was carried out. Each 11 min run consisted of ARBIB having free range to travel around the robot laboratory, negotiating obstacles in its path. The first seven runs were made with synaptogenesis enabled in the Hi-NOON model. For comparison, the remaining runs were made with synaptogenesis disabled. By interaction with the environment, ARBIB learns (by stimulus substitution of a CS for a US) to elicit its avoidance reflex (the UR). Hence, the measure of bump sensory neuron activity is a useful indicator of how successfully it has learned to avoid direct contact with obstacles. Fig. 11 shows the average action potential count for bins of 100 sample points throughout the tests. Results of the first seven runs are shown in Fig. 11(a) and (b). Comparing these results with Fig. 11(c) and (d), we see a decrease in activity during the runs with synaptogenesis enabled. An interesting observation is that Fig. 11(b) still shows activity at the end of the test. This means that stimulus substitution during the synaptogenesis runs has improved the obstacle avoidance skill of ARBIB, but seems not to have saturated the network with an overwhelming population of new parallel synapses.

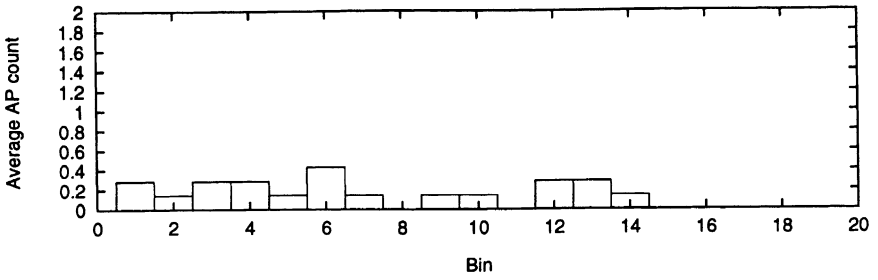
Fig. 10. The Nomad Scout 2 instantiation of ARBIB used in the synaptogenesis experiments.

4.2.2. Cricket phonotaxis

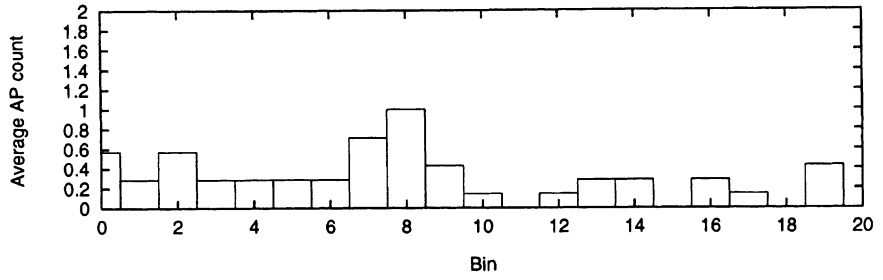
Neuroethology is a new and broad field, centring on an understanding of how nervous systems produce natural animal behaviours. Hi-NOON is a potentially valuable simulation tool in this new field. Here, we outline the use of the simulator in the neuroethological study of phonotaxis in the cricket *Gryllus bimaculatus*.

Phonotaxis is the movement of the female cricket towards the male's mating song. Lund et al. [33] have modified a Khepera robot for their work with cricket phonotaxis by adding an audio input to the robot via three additional interface boards: two analogue audio input and signal processing boards, and a Khepera bus interface board. This hardware enhancement (Fig. 12) implements an electronic model of a cricket's auditory system [34, p. 159], comprised of cross-coupled auditory spiracles (openings either side of the animal's thorax) and tympani (pressure detectors) as shown in block diagram form in Fig. 13.

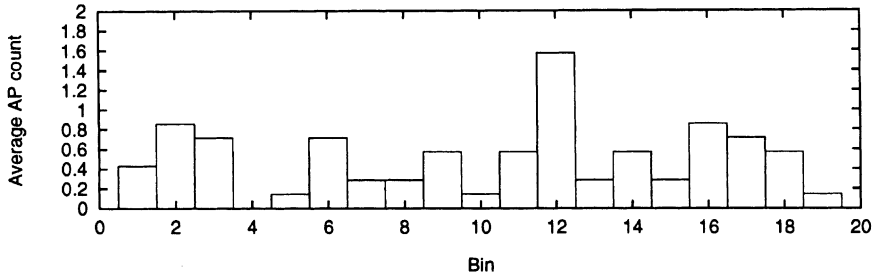
Scutt and Webb [35] have used Hi-NOON to simulate the auditory system of the cricket within this modified mobile robot, to study the neurophysiological underpinnings of phonotaxis. Here, the Hi-NOON simulator executes on the on-board processor of the Khepera



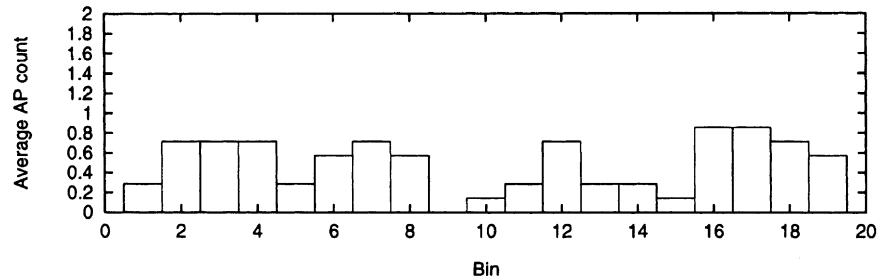
(a) Left bump sensory neuron firing activity with synaptogenesis.



(b) Right bump sensory neuron firing activity with synaptogenesis.



(c) Left bump sensory neuron firing activity without synaptogenesis.



(d) Right bump sensory neuron firing activity without synaptogenesis.

Fig. 11. Neural activity in left and right bump sensory neurons (a, b) with and (c, d) without synaptogenesis.

robot and simulates Webb and Scutt's four-neuron model of cricket phonotaxis behaviour, shown in Fig. 14. This receives input from the electronic model of the cricket's auditory system mentioned above. In particular, we note that it was necessary to introduce (as an essential element) synaptic depression on repeatedly

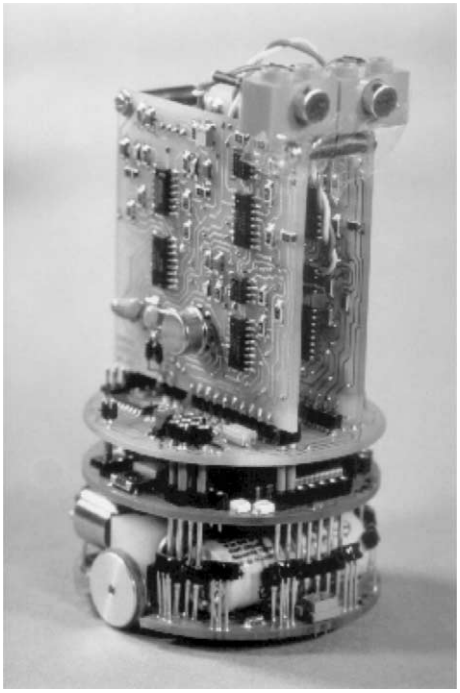


Fig. 12. A modified Khepera robot used for the study of cricket phonotaxis by Lund et al. [33].

activated synapses. Although this was not part of the original Hi-NOON, the ease with which it was added illustrates the flexibility of the simulator.

The two AN auditory neurons receive sound input. Each has an excitatory connection to the ipsilateral motor neuron (MN), and an inhibitory connection to the contralateral AN–MN synapse. This means that a motor neuron will tend to increase in potential only if its ipsilateral AN started to fire before the contralateral AN. Because of synaptic depression, only the first few spikes in a burst in AN will contribute to increased potential in MN. The weight of the AN–MN connection and the time constant of MN have been tuned so that a single (or continuous) burst in AN will not bring MN above threshold, but a gap in AN sufficient to allow synaptic recovery will enable a subsequent burst to produce a spike in MN, provided it occurs before MN has decayed back to resting level. This is so that the response of the robot becomes selective to the temporal pattern in the auditory signal, i.e. it recognises the male cricket song. The robot produces behaviour closely similar to the cricket in most situations. In the words of Webb and Scutt [27]: "No alternative models have as yet been presented with a comparable detail or evaluation".

5. Potential for integrating neural and electronic circuit simulation

So far, this paper has concentrated upon the applications of the Hi-NOON neural simulator. Apart from its other capabilities, we have shown that Hi-NOON is able to simulate both spiking and non-spiking (PDP-type)

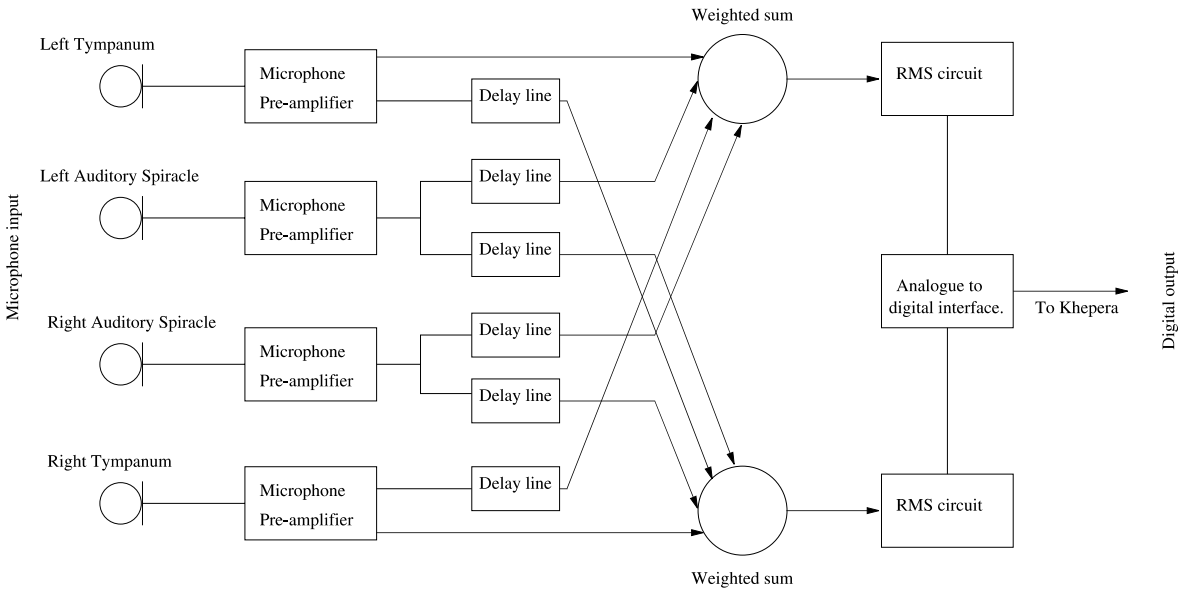


Fig. 13. Block diagram of the cricket's auditory system as realised on Khepera by Lund et al. [33].

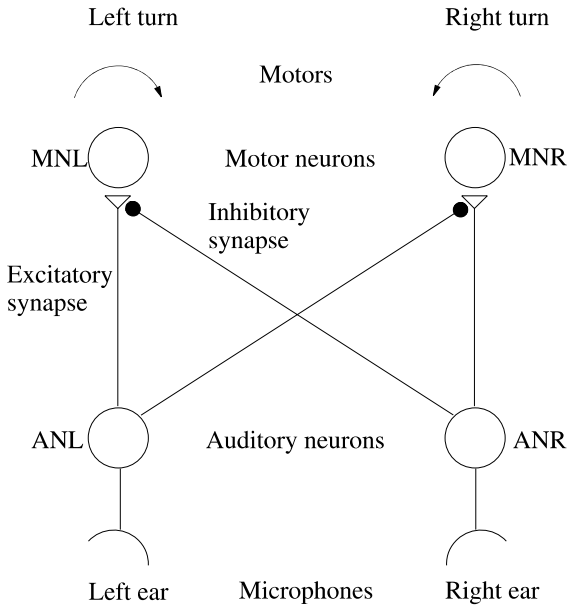


Fig. 14. Proposed neural circuit for cricket phonotaxis from Webb and Scutt [27].

artificial neurons. An early demonstration of the possible benefits of mixing neural and non-neural models was provided by Litovski et al. [36], who used a PDP-type neural network as a 'black-box' model of electronic circuit components (specifically an MOS transistor) whose characteristics were unknown but could usefully be learned from example devices. Given the benefits of mixing neural and conventional electronic circuit components within the same simulation, this requirement has been catered for in the Alecsis package [37,38]. So it seems that there is value in mixing spiking and PDP-style (non-spiking) neural models as in Hi-NOON, and in mixing PDP-style neural models and electronic circuit components as in Alecsis. This raises the question: would it be worth integrating the whole into a single simulation package?

The two simulators have very different goals and philosophies. With Hi-NOON, real-time operation is essential if it is to be used to model animat or robot neural circuitry in situated systems and neuroethological studies; hence the use of a computationally efficient state system working at the level of membrane potential. Alecsis is intended primarily as a circuit design aid and so real-time operation is less of a concern, although the huge size of modern electronic circuits means that computational efficiency during simulation remains paramount if the iterative design process is to be supported effectively. Circuit designers are interested in currents and voltages so this is appropriate level of abstraction; hence, the core simulation engine is based on nodal analysis [39,40]. These fundamental differences mean that integration would not be straightforward.

Two alternatives for using Hi-NOON models with Alecsis are possible:

1. If it is desirable to incorporate a complete Hi-NOON simulator into Alecsis, two things must be done:
   - the creation of high-level commands in the Alec++ language for describing Hi-NOON objects, and
   - interfacing a Hi-NOON simulator to the simulation engine of Alecsis.
2. If only the spiking neuron models of Hi-NOON are required for use in Alecsis, the Alec++ language can be used to describe the state system algorithm.

What would we gain by merging both systems? There is every possibility that such a hybrid simulator would contribute to the advancement of a new and exciting area of research – the interfacing of brain tissue to robots [41]. In such a simulation, the Hi-NOON neuron models would interact with the electrode interface circuitry modelled by Alecsis, thus aiding the development of hybrid organic/inorganic animats.

6. Conclusion

Hi-NOON is an object-oriented neuronal circuit simulator specifically developed for studying the neurophysiological basis of behaviour in real animals and in situated artificial systems. It simulates changes in membrane potential (including spiking behaviour) but is also capable of modelling the continuous or piecewise linear activation functions typical of PDP-style artificial neural nets. The main simplifications for neurobiological modelling are that it treats each neuron as a single compartment (a point neuron), with inputs modelled as added voltage. In place of differential equations, a state system is used, along with a flexible parameter system to cater for differences between neuron types and to keep track of the changing state of each neuron over time. This allows circuits of heterogeneous neurons modelled on real neurophysiological data to be constructed with a minimum of effort and processed with relative ease in real time. These practical advantages are illustrated by the use of Hi-NOON to simulate and implement the 'nervous systems' of the ARBIB mobile robot and a robot/cricket which closely mimics the phonotaxis behaviour of the real animal. Some possible advantages of integrating Hi-NOON with a conventional electronic circuit simulator like Alecsis have been identified and the practical difficulties of doing so have been outlined.

Acknowledgements

The authors wish to thank Barbara Webb for her contribution to the successful use of Hi-NOON in modelling cricket phonotaxis, and for critical comments on relevant aspects of this paper.

References

[1] Rumelhart DE, McClelland JL, editors. Parallel distributed processing explorations in the microstructure of cognition, vol. 1 – Foundations. Cambridge, MA: Bradford Books/MIT Press; 1986.
[2] Rumelhart DE, Hinton GE, Williams R. Learning representations by back-propagating errors. Nature 1986; 323:533–6.
[3] Chauvin Y, Rumelhart D, editors. Backpropagation: theories, architectures and applications. Hillsdale, NJ: Lawrence Erlbaum; 1995.
[4] Kandel ER. Small systems of neurons. Sci Am 1979; 241:61–70.
[5] Hawkins RD, Kandel ER. Is there a cell biological alphabet for simple forms of learning? Psychol Rev 1984;91:375–91.
[6] Sejnowski TJ, Koch C, Churchland PS. Computational neuroscience. Science 1988;241:1299–306.
[7] Scutt TW, Damper RI. Computational modelling of learning and behaviour in small neuronal systems. Proceedings of International Joint Conference on Neural Networks, Singapore, 1991. p. 430–5.
[8] Scutt TW. Synthetic neural networks: a situated systems approach. PhD Thesis, Department of Electronics and Computer Science, University of Southampton, UK. 1995.
[9] Scutt TW, Damper RI. Designing a nervous system for an adaptive mobile robot. In: Browne A, editor. Neural network perspectives on cognition and adaptive robotics. Bristol, UK: Institute of Physics Press; 1997. p. 220–50.
[10] Levitan IB, Kaczmarek LK. The neuron: cell and molecular biology. New York, NY: Oxford University Press; 1997.
[11] Port RF. Representation and recognition of temporal patterns. Connect Sci 1990;2:151–76.
[12] Port RF, Cummins F, McAuley JD. Naive time, temporal patterns and human audition. In: Port RF, van Gelder T, editors. Mind as motion: explorations in the dynamics of cognition. Cambridge, MA: Bradford Books/MIT Press; 1995. p. 339–71.
[13] Rieke F, Warland D, de Ruyter van Steveninck R, Bialek W. Spikes: exploring the neural code. Cambridge, MA: Bradford Books/MIT Press; 1997.
[14] Coad P, Yourdon E. Object oriented analysis. 2nd ed. Englewood Cliffs, NJ: Prentice-Hall; 1991.
[15] Eliëns AS. Principles of object-oriented software development. Wokingham, UK: Addison-Wesley; 1994.
[16] Selverston AI. Modeling of neural circuits – what have we learned? Annu Rev Neurosci 1993;16:531–46.
[17] MacGregor RJ. Neural and brain modeling. London, UK: Academic; 1987.
[18] Koch C, Segev I, editors. Methods in neuronal modeling: from synapses to networks. Cambridge, MA: MIT Press; 1989.
[19] MacGregor RJ. Theoretical mechanics of biological neural networks. London, UK: Academic; 1993.
[20] O'Reilly RC. Six principles for biologically based computational models of cognition. Trends Cognit Sci 1998; 2:455–62.
[21] Donegan NH, Gluck MA, Thompson RF. Integrating biological and behavioral models of classical conditioning. In: Hawkins RD, Bower GH, editors. Computational models of learning in simple neural systems. San Diego, CA: Academic; 1989. p. 109–56.
[22] Lieberman DA. Learning: behavior and cognition. 2nd ed. Pacific Grove, CA: Brooks/Cole; 1993.
[23] Sutton RS, Barto AG. Towards a modern theory of adaptive networks: expectation and prediction. Psychol Rev 1981;88:135–70.
[24] Damper RI, Scutt TW. Biologically-based learning in the ARBIB autonomous robot. Proceedings of IEEE International Symposia on Intelligence and Systems, Washington, DC, 1998. p. 49–56.
[25] Damper RI, Scutt TW. Biologically-motivated neural learning in situated systems. In: Proceedings of IEEE International Symposium on Circuits and Systems, ISCAS'98, Monterey, CA, 1998. p. III-115–8.
[26] Damper RI, French RLB, Scutt TW. ARBIB: an autonomous robot based on inspirations from biology. Robot Autonom Syst 2000;31(4):247–74.
[27] Webb B, Scutt T. A simple latency-dependent spiking-neuron model of cricket phonotaxis. Biol Cybernet 2000; 82(3):247–69.
[28] Wilson SW. Knowledge growth in an artificial animal. Proceedings of 1st International Conference on Genetic Algorithms and their Applications. Hillsdale, NJ: Lawrence Erlbaum; 1985. p. 16–23.
[29] Selverston AI. A consideration of invertebrate pattern generators as computational databases. Neur Netw 1988; 1:109–17.
[30] French RLB, Damper RI. Stability of learning in the ARBIB autonomous robot. SAB2000 Proceedings Supplement, 6th International Conference on Simulation of Adaptive Behavior: From Animals to Animats, Paris, France, 2000. p. 150–9.
[31] Walter WG. A machine that learns. Sci Am 1951; 185(5):60–3.
[32] Carpenter GA, Grossberg S. The ART of adaptive pattern recognition by a self-organizing neural network. IEEE Comput 1988;21(3):77–88.
[33] Lund HH, Webb B, Hallam J. A robot attracted to the cricket species *Gryllus bimaculatus*. In: Husbands P, Harvey I, editors. Proceedings of Fourth European Conference on Artificial Life, ECAL'97, Brighton, UK. Cambridge, MA: MIT Press/Bradford Books; 1997. p. 246–55.
[34] Bradbury JW, Vehrencamp SL. Principles of animal communication. Sunderland, MA: Sinauer Associates; 1998.
[35] Scutt T, Webb B. Real neurons in real networks. Proceedings of European Symposium on Artificial Neural Networks, Brussels, Belgium, 1997. p. 33–8.
[36] Litovski VB, Radjenović J, Mrčarica ŽM, Milenković S. MOS transistor modelling using neural networks. Electron Lett 1992;28(18):1766–8.

[37] Mrčarica Ž, Glozić D, Litovski V, Detter H. Simulation of microsystems using a behavioural hybrid simulator Alecsis. First International Conference on Microsystems and Microstructures, MICROSIM'95, Southampton, UK, 1995. p. 129–36.

[38] Mrčarica Ž, Glozić D, Litovski V, Maksimović D, Ilić T, Gavrilović D. Alecsis 2.3: the simulator for circuits and systems, User's Manual. Technical Report LEDA-1/98, Faculty of Electronic Engineering, University of Niš, Yugoslavia, 1998.

[39] Ho CW, Ruehli AI, Brennan PA. The modified nodal approach to network analysis. IEEE Trans Circ Syst 1975;22(6):504–9.

[40] Litovski V, Zwolinski M. VLSI circuit simulation and optimization. London, UK: Chapman and Hall; 1997.

[41] Fleming KM, Reger BD, Sanguineti V, Alford S, Mussa-Ivaldi FA. Connecting brains to robots: an artificial animal for the study of learning in vertebrate nervous systems. 6th International Conference on Simulation of Adaptive Behavior, Paris, France, September 2000 (in press).