The Hi-NOON Neural Simulator and its Applications

R.I. Damper*, R.L.B. French* and T.W. Scutt†

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

†Core Design Ltd.,
55 Ashbourne Road,
Derby DE22 3FS, UK.

Abstract

This paper describes the Hi-NOON neural simulator, originally conceived as a general-purpose, 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 efficacy of the simulator is illustrated with respect to some recent applications to situated systems studies.

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 [13], grossly simplified, artificial models of neural networks have been defined and studied, fuelled by the discovery of powerful learning algorithms such as error back-propagation. 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 repertoire of behaviours [8, 9]. The two threads – parallel distributed processing (PDP or 'connectionism') and systems neuroscience – meet in the relatively newer paradigm of computational neuroscience [16], 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 **Hi**erarchical **N**etwork of **O**bject-**O**riented **N**eurons. As the name suggests, in Hi-NOON, synapses, neurons and networks are in principle represented as objects within an object-oriented hierarchy [14, 15] at various levels of abstraction. The lowest such level uses the membrane potential (strictly, transmembrane potential difference) as the observable parameter in the network model. This is a much lower-level approach than the use of activation values roughly corresponding to the spike or action potential rate of individual neurons or collections of neurons as in PDP models. By contrast, Hi-NOON retains details of individual spike generation which is lost in the traditional connectionist approach. As well, Hi-NOON facilitates simulation of a non-homogeneous population of neurons. In principle, 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 the simulator for a highly conventional PDP-type artificial neural network.

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. Clearly, 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. [12, p. 279]:

> "... under many conditions, behavioral 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 ..."

In the section immediately following, we give an overview of the Hi-NOON implementation before presenting details of the model neurons and synapses. We then outline some recent applications of the simulator before concluding.

2 An Overview of Hi-NOON

The original program was written in object-oriented Pascal, but has subsequently been rewritten in C using the disciplines of object-oriented programming (OOP) [2, 5]. C was used (rather than C++ with its explicit support of OOP features) to maximize portability among various realisations in different applications. The benefits of the OOP approach are two-fold. 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 to `mun.ecs.soton.uk` from directory `pub/users/rid/hinoon`.

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 [18] "minimum requirements" 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:

- a set of parameters which defines the neuron;
- a set of data structures which defines the 'axon terminals' for the neuron, each of which is itself an object and has its own parameters;
- a set of 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. This models the finite stores of neurotransmitter in the synaptic terminals of biological neurons.

2.3 Neuron Types

Hi-NOON allows a non-homogeneous population of neurons to be simulated – reflecting the fact that neurons have specialized functions in real neurobiological systems – at the most appropriate level of abstraction. Modeling individual neurons at the level of membrane potential allows sub-threshold and spiking behaviors to be simulated 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 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 Approximate State System

Each neuron is treated as being in one of a number (or occasionally more than one) 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.

Figure 1 (taken from a Hi-NOON simulation) shows the states passed through by a neuron during firing. 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 below. The state system is 'approximate' – there is some overlap between states. For instance, a neuron may be sub-threshold, but `fired` may still be true, indicating that the membrane potential is undergoing its post-firing hyperpolarisation.

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.

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. 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 Figure 1 and Section 3.2) that synaptic communication can take place. Hence, it is irrelevant that we are, in some sense, modeling sub-threshold behaviour incorrectly. An alternative view is that we are not modeling axonal transmission, i.e. we have 'point' neurons as is common in neural modeling [11, pp. 21-4].

2.6 Learning in Hi-NOON

There is no specific support for learning in Hi-NOON. Thus, if PDP-type learning (e.g. back-propagation) 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 [8, 4, 10]. Generally, these simple forms of learning are implemented using synapse-on-synapse connections in Hi-NOON.

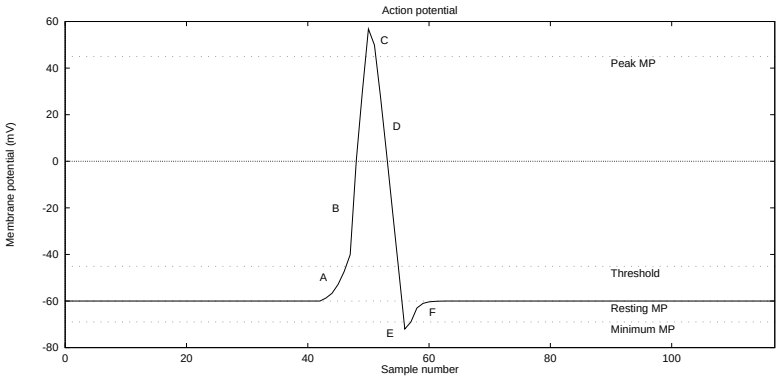

Figure 1: Time evolution of typical action potential (spike) of a **basic** neuron in a Hi-NOON simulation. See text for specification of the states (A..F) passed through by a neuron during firing. Here, the sample period is approximately 4 ms (this varies with the machine on which the simulation runs.)

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 millivolts) for this neuron type are:

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

where:

$$S(t) = \sum_i w_i \kappa \, (MP_i(t) - BaseMP_i)$$

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 learning 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.

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

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

$$\texttt{fired}(t) = \begin{cases} \text{TRUE} & \text{if } \texttt{cond1} \\ \text{FALSE} & \text{otherwise} \end{cases}$$

where `cond1` is (state B, C, D), as for the noise-free synapse, ANDed with:

$$\frac{MP_p - \theta_p}{h - \theta_p} \times 100 \geq \text{rand mod100}$$

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

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

where:

$$k = \frac{nT}{\psi} e^{\left(\frac{-nT}{\varsigma}\right)}$$

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.

4 Applications

The Hi-NOON simulator has been used to design and implement the 'nervous systems' of two rather different situated systems. One is the ARBIB autonomous robot [3] which has been implemented on a variety of hardware and software platforms. ARBIB learns from and adapts to its environment, which consists of hard objects and light sources casting shadows. 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 [8] that (associative) classical and 'higher order'conditioning might be based on an elaboration of these (non-associative) forms of learning. Accordingly, ARBIB's 'nervous system' has a non-homogeneous population of spiking neurons, its drive to explore its environment was provided by a simple central pattern generator neural circuit [17], 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 behaviors.

One problem with the initial ARBIB implementation was that its learning was almost entirely plastic. That is, it rapidly 'forgot' 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 [6], according to which new synapses may be created. The generation process was constrained by introducing a new predicate into Hi-NOON: a new synapse is only created, parallel to an existing conditioned synapse, 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 this stabilises the learning to a useful degree, so offering a practical remedy to the stability-plasticity dilemma [7, 1]

Webb and Scutt [19] have used Hi-NOON to simulate and then implement the auditory system of the cricket within a mobile robot, to study the neurophysiological underpinnings of phonotaxis, i.e. the movement of the female towards the male's mating song. The robot produces behaviour closely similar to the cricket in most situations. In their words: "No alternative models have as yet been presented with a comparable detail or evaluation".

5 Conclusions

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) rather than using the continuous activation functions typical of PDP-style artificial neural nets. The main simplifications are that it treats each neuron as a single compartment, with inputs modelled as added voltage. In place of differential equations, an approximate 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. 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 reproduces the phonotaxis behaviour of the real animal.

References

[1] G. A. Carpenter and S. Grossberg. The ART of adaptive pattern recognition by a self-organizing neural network. *IEEE Computer*, 21(3):77–88, 1988.

[2] P. Coad and E. Yourdon. *Object Oriented Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1991. Second Edition.

[3] R. I. Damper, R. L. B. French, and T. W. Scutt. ARBIB: an autonomous robot based on inspirations from biology. *Robotics and Autonomous Systems*, in press.

[4] N. H. Donegan, M. A. Gluck, and R. F. Thompson. Integrating biological and behavioral models of classical conditioning. In R. D. Hawkins and G. H. Bower, editors, *Computational Models of Learning in Simple Neural Systems*, pages 109–156. Academic, San Diego, CA, 1989.

[5] A. Eliëns. *Principles of Object-Oriented Software Development*. Addison-Wesley, Wokingham, UK, 1994.

[6] R. L. B. French and R. I. Damper. Stability of learning in the ARBIB autonomous robot. submitted to *Sixth International Conference on Simulation of Adaptive Behavior*, Paris, France, September 2000.

[7] W. Grey Walter. A machine that learns. *Scientific American*, 185(5):60–63, 1951.

[8] R. D. Hawkins and E. R. Kandel. Is there a cell biological alphabet for simple forms of learning? *Psychological Review*, 91:375–391, 1984.

[9] E. R. Kandel. Small systems of neurons. *Scientific American*, 241:61–70, 1979.

[10] D. A. Lieberman. *Learning: Behavior and Cognition (2nd edition)*. Brooks/Cole, Pacific Grove, CA, 1993.

[11] R. J. MacGregor. *Neural and Brain Modeling*. Academic, London, UK, 1987.

[12] F. Rieke, D. Warland, R. de Ruyter van Steveninck, and W. Bialek. *Spikes: Exploring the Neural Code*. Bradford Books/MIT Press, Cambridge, MA, 1997.

[13] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1 – Foundations*. Bradford Books/MIT Press, Cambridge, MA, 1986.

[14] T. W. Scutt and R. I. Damper. Computational modelling of learning and behaviour in small neuronal systems. In *Proceedings of International Joint Conference on Neural Networks*, pages 430–435, Singapore, 1991.

[15] T. W. Scutt and R. I. Damper. Designing a nervous system for an adaptive mobile robot. In A. Browne, editor, *Neural Network Perspectives on Cognition and Adaptive Robotics*, pages 220–250. Institute of Physics Press, Bristol, UK, 1997.

[16] T. J. Sejnowski, C. Koch, and P. S. Churchland. Computational neuroscience. *Science*, 241:1299–1306, 1988.

[17] A. I. Selverston. A consideration of invertebrate pattern generators as computational databases. *Neural Networks*, 1:109–117, 1988.

[18] A. I. Selverston. Modeling of neural circuits – What have we learned? *Annual Review of Neuroscience*, 16:531–546, 1993.

[19] B. Webb and T. Scutt. A simple latency-dependent spiking-neuron model of cricket phonotaxis. *Biological Cybernetics*, in press.