EXPLORING TEMPORAL COMPUTATION IN NEURONAL SYSTEMS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND ELECTRICAL ENGINEERING

OF GLASGOW UNIVERSITY

IN FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Øystein Haug Olsen

November 1993

'The scientist does not study nature because it is useful; he studies it because he delights in it, and he delights in it because it is beautiful. If nature were not beautiful, it would not be worth knowing, and if nature were not worth knowing, life would not be worth living.'

Henri Poincare.

Abstract

This thesis presents two approaches to identifying computational properties of networks of neurons. The first of these involves a bottom-up approach through detailed modelling of neuronal properties, while the second is a top-down approach using principal component analysis that quantifies the behaviour of neurons in complex networks from measurements.

An object-oriented modelling tool has been developed that is fast, flexible, and easy to use. A graphical user interface allows the user to manipulate neurons, synapses, and currents visually on the screen during prototyping. Subsequently, a systematic study of the model can be processed in batch mode, for example to tune it to a particular behaviour.

Small invertebrate neuronal circuits have been considered where the neuronal outputs can be related to the behaviour of the animal. The fundamental problem faced by experimental neurobiologists is that only one state variable per neuron is readily accessible, i.e. the membrane potential at the cell body. A neuron contains a vast number of state variables, but these are generally all hidden. The modelling tool enables one to "record" from hidden state variables and to manipulate inaccessible parameters of the real neurons.

The tool has been evaluated through an investigation involving a leech heart model. Significant findings include non-spiking oscillations, and the modelling has suggested further experimental work involving the real system. The modelling tool

provided close to real-time performance in this application which indicates its potential for its interactive use in an experimental environment, including dynamic voltage-clamp.

The top-down approach uses principal component analysis to quantify a trace of neuronal activity during a time interval. During this interval, the feedback loops within the neuron and through the neuronal network will affect the output of the neuron. Thus, the resulting measure of the neuronal output will indirectly include the states of "hidden" compartments away from the cell soma and other inaccessible state variables, like channel states. Although the technique offer no promise for tracing these hidden state variables, it will enable us to include them in quantifying the outputs of a neuron.

Further, the technique serves as an "objective critic" that display the largest sources of variance over the data set. Therefore, one avoids testing successively and explicitly through a large set of possible features. This also makes the measure low-dimensional and easy to analyse further.

Application of the Karhunen-Loève transform to the crayfish swimmeret showed that the principal components represented features like spike count, burst width, burst concentration, and burst latency. The burst latency proved to be significantly modulated.

Principal component analysis was also performed on the membrane potential after having removed the action potentials with a low pass filter. The membrane potential deviations did not correlate with those of the spikes from the same neuron. This demonstrates the point that the membrane potential at the cell body does not solely determine the spiking pattern, but other (unobservable) state variables influence the spiking dynamics. Recently developed imaging techniques show possibilities in terms of measuring the spatial distribution of the membrane potential and ion concentrations throughout a cell, as well as the temporal interaction between these state

variables. These imaging techniques are of potential value for validating theoretical models of a more complex kind for the study of which the object-oriented modelling tool could be readily applied.

This work has shown that the Karhunen-Loève transform can be used to quantify the relative coupling strength between two outputs. The coupling strength is estimated at the level of behaviour and relative to normal background activity in the system. This compares to the absolute estimate that at the level of individual synapses measures the coupling strength between two cells. The techniques above enables the effective coupling between two cells to be measured, where the coupling may occur through several pathways and through several cells. These results suggest that the long range intersegmental coupling is not much weaker than short range intrasegmental coupling and that the coupling performs phase locking only.

Acknowledgments

I would like to thank my supervisor Professor D.J. Murray-Smith for his advice and assistance throughout this project. I am also thankful for the advice from Dr. P. Connolly during the early phase of the project.

The generous support from the Mathematical Sciences Research Institute, that enabled me to attend the Neurons in Networks workshops at Berkeley during the summer 1992, is greatly appreciated. Thanks are also due to Professor W.O. Friesen and the Center for Biological timing at the University of Virginia for arranging a very pleasant and interesting stay during the summer of 1992. I would also like to thank Drs. R. Gray and C. Hocker for interesting discussions and suggestions on signal analysis.

I would not have been able to attend the Neural Systems & Behavior Course during the summer of 1993 at the Marine Biology Laboratory, Woods Hole, MA, U.S.A., without the financial aid from the L. Crocker Scholarship Fund, the E.G. Conklin Memorial Fund, and a grant from the Department of Electronics and Electrical Engineering, authorised by Professor P. Laybourn. I would also like to thank MBL for providing such an unique and exciting environment.

Thanks are due to Dr. A. Chrachri and Dr. C. Hocker for providing experimental data from the crayfish swimmeret and the leech swimming, respectively, and to Professor R.L. Calabrese for both experimental and modelling data on the leech heart system, as well as fruitful discussions.

Finally, the intellectual, mountaineering, and golf adventures with Dr. J. Cooper are highly appreciated.

This work was funded and supported by an Overseas Research Scholarship, the University of Glasgow, and the Norwegian Research Council.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Neurophysiological background

The neuron is a cell that has specialised in processing electrical signals. The entire neuron is enclosed in a semi-permeable membrane made up of hydrocarbons, phosphates, and proteins. The intracellular fluid has a high concentration of K^+, whereas the extracellular liquid has a high concentration of Na^+ and Ca^{2+}. Some proteins in the cellular membrane are leakage channels specific to Na^+, Ca^{2+}, or K^+ ions. These proteins come in many forms—some have constant permeability, some are gated by the membrane potential, and other channels are gated by ion concentrations. One protein, the Na^+/K^+ pump, pumps Na^+ out of and K^+ into the cell.

Electrical currents pass through the membrane carried by various ions. This ionic transport through the membrane is influenced by ion concentration gradients, the potential gradient across the membrane potential, the ion pumps in the membrane, and the number of open leakage channels. At equilibrium, the ionic currents through the membrane cancel and this results in a resting potential inside the cell that is lower than that of the surrounding fluid, typically -70 mV. The overall picture is analogous to a semiconductor pn-junction at equilibrium. The membrane has highest

permeability to Na^+ and K^+. These ions take the roles of holes and electrons in the pn-junction, but here both carriers have positive charge.

The membrane potential refers to the potential of the intracellular fluid relative to the external fluid. An increase in the membrane potential (towards $+\infty$) is called a *depolarisation*, and a decrease of the membrane potential is a *hyperpolarisation*.

When a neuron is depolarised to some threshold (typically 20 mV above the resting potential) some of the normally closed voltage gated Na^+ channels open. The sudden influx of positive Na^+ ions increases the depolarisation and thus opens more Na^+ channels. The positive feedback loop causes a sharp rise in the internal potential (from -70 mV to +30 mV). Within a few milliseconds, the potential is brought back to the resting level by delayed K^+ channels opening for a K^+ ion efflux. The Na^+/K^+ pump and the leakage channels restore the initial ionic concentrations. This potential spike, the *action potential* , is the basis for all computation in the vertebrate nervous system. Neurons in the vertebrate cortex fire action potentials at frequencies of typically 100 Hz. Conventionally, the action potential is thought to originate in the cell body and propagate through the *axon* out to the synapses (see figure 2). Figure 1 shows a stylised action potential.

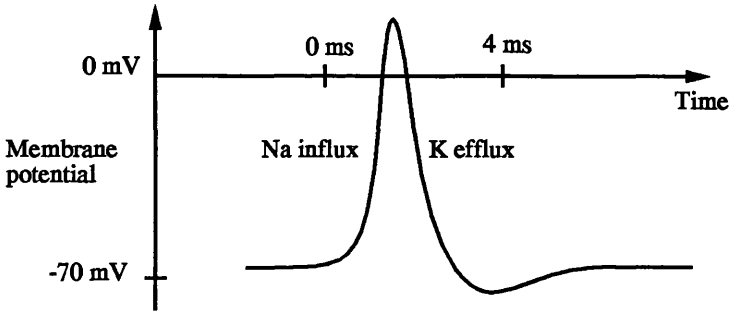

Figure 1: A typical action potential.

The layout of a typical neuron can be seen in Figure 2.

The link between neurons is the *synapse*, which is usually formed from an axon

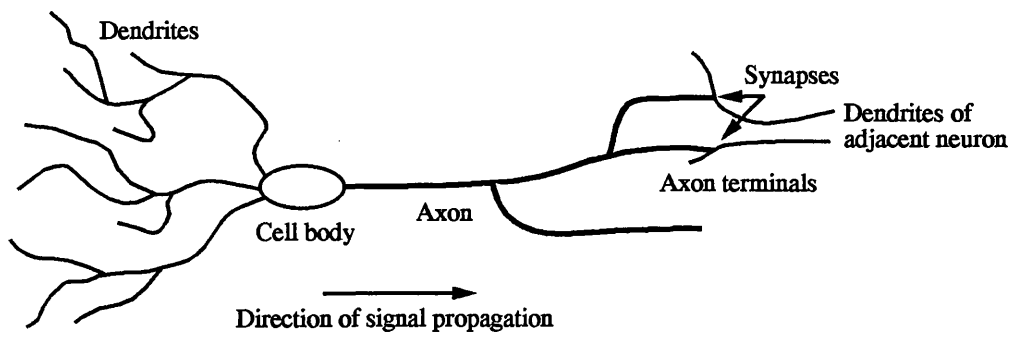

Figure 2: A typical biological neuron.

terminal onto a dendrite. There are two main types of synapses: (1) The *electrotonic* synapses let the pre-synaptic action potential ion currents flow through to the post-synaptic neuron and cause post-synaptic potentials (PSPs). (2) In the case of the *chemical* synapses, there is no physical contact between the two membranes. The neurotransmitter is stored in synaptic vesicles in the axon terminal and is released by the increased Ca^{2+} concentration during the pre-synaptic action potential. The neurotransmitter diffuses across to the post-synaptic membrane and opens chemical-gated ion channels. These induce ionic transport across the post-synaptic membrane and create post-synaptic potentials. The chemical synapses are either *excitatory* or *inhibitory* corresponding to depolarising or hyperpolarising post-synaptic potentials, respectively. These increase or decrease the probability of the post-synaptic membrane reaching threshold. The chemical synapse is always one-way, whereas the electrotonic synapse in principle can operate both ways. The respective electrical analogues are the diode and the resistor.

The effects of the neurotransmitter release cause short and small post-synaptic potentials to propagate up the dendrites into the cell body (or *soma*). The dendrite is commonly described by the cable equation (Rall [1989]), i.e. as a passive cable with distributed resistance and capacitance, but is likely to have active regions for amplification and processing. If enough of these small potentials arrive at the soma from the

many dendrites at about the same time, the cell reaches its threshold potential and an action potential is propagated down the axon. The threshold is now temporarily high during the refractory period, thus ignoring further input. The refractory period lasts typically a few milliseconds, after which the cell is ready to fire again. Information appears to be carried predominantly through the spike timing.

Invertebrate animals are much favoured by neurobiologists for their "simple" nervous systems. They contain relatively few neurons, i.e. some thousands, and the neurons are relatively large. Therefore the neurons can easily be impaled by microelectrodes, and the individual neurons can be identified, thus allowing repeatable experiments to be performed. The most intensely studied invertebrates are the marine molluscs *Aplysia californica* and *Tritonia*, the snail *Lymnea*, and the medicinal leech *Hirudo medicinalis.*

The vertebrate nervous systems are much more complicated. The neurons are much smaller and their morphology and inter-connectivity are more complex. The behaviour of the whole system is more sophisticated and the neural recordings are therefore harder to interpret. The favourite targets for vertebrate studies are slice preparations of the hippocampus, the olfactory bulb, the cerebellum, the visual cortex, and regions of motor co-ordination.

1.2 The remaining problems of neurobiology

Great progress has been achieved in both analytical and experimental techniques over the last decades. However, the fundamental questions of how the neurons assemble to form complex systems remain. A lot is known about some of the mechanisms of the cells, but there is no general theory of how these mechanisms produce an appropriate collective behaviour. The reasons for this include the following:

- We are studying the problem at a too small scale. Even though neurons are the obvious computational element, it may be that neurons assemble into larger elemental structures.
- We are focusing at a too large scale. We will have to know more about the smaller structures of the cells: dendrites, spines, and ion channel states.
- Neuronal modules have developed specific mechanisms to tailor its particular behaviour. It is not possible generalise these mechanisms into fundamental principles, and each module must be studied individually.
- The mathematical language and tools presently available are inadequate for describing neural computation.
- The number and complexity of neurons required for proper operation of nervous systems is beyond the resources of present computers.
- The field of neurobiology is so wide that it requires knowledge beyond the capacity of an individual. Ineffective inter-disciplinary collaborations hamper further progress.

The task of linking the behaviour of a network of neurons to the cellular mechanisms is made even harder by the vast number of state variables of a single neuron. At any given point of the cell membrane, there are tens of active ion channels of different characteristics. The channels transport selected ions and they are activated and inactivated at different membrane potentials. Hence, any point of the membrane represents a large number of state variables. These states will play a role in neural computation but are not uniquely represented in the membrane potential. Since the neuroscientists have only had access to the membrane potential, computation at the ionic level can only be traced by guesswork. Realistic models of single cells, including all the ionic currents, should therefore be used to assess the credibility of the guesses.

The analysis is further complicated by the spatial extent of the cell. Each point of the cell is in principle independent and contains a large number of states, although concentration and potential gradients will smooth the states spatially. If the distant regions of the cell, like the presynaptic terminals and the dendritic tree, are of importance, they have been inaccessible for experimental observation and hence not discovered.

Voltage and ion sensitive fluorescent dyes offer investigations of smaller areas than the soma. These dyes make the optical transparency at a point proportional to the membrane potential or an ion concentration at that point. By measuring the optical transparency of the cell, the membrane potential and the ion concentrations can be determined at any particular point. This allows distant processes and the roles of the membrane potential and ions to be analysed in great detail. The potential of these techniques is not yet realised, and they will soon contribute to extending our understanding of nervous systems (Ross, Arechiga & Nicholls [1987]).

1.3 Neuronal modelling

Eventually, theories regarding the computation of neural systems have to be quantified and tested. There is no general theory for the function of nerve cells so their operation must typically be verified by computer simulations. The approaches to nerve cell modelling are many and parallels the uncertainties about the crucial neural processes mentioned earlier.

W. Rall applied Kelvin's cable equation to neural modelling in the 1960s (Rall [1962]; Rall [1977]; Rall [1989]). Under particular restrictions, he obtained analytical solutions for passive membrane structures by analysing them as electrical cables with distributed capacitance and resistance. This is the basis for the present understanding of signal propagation in neurons, together with the active membrane properties

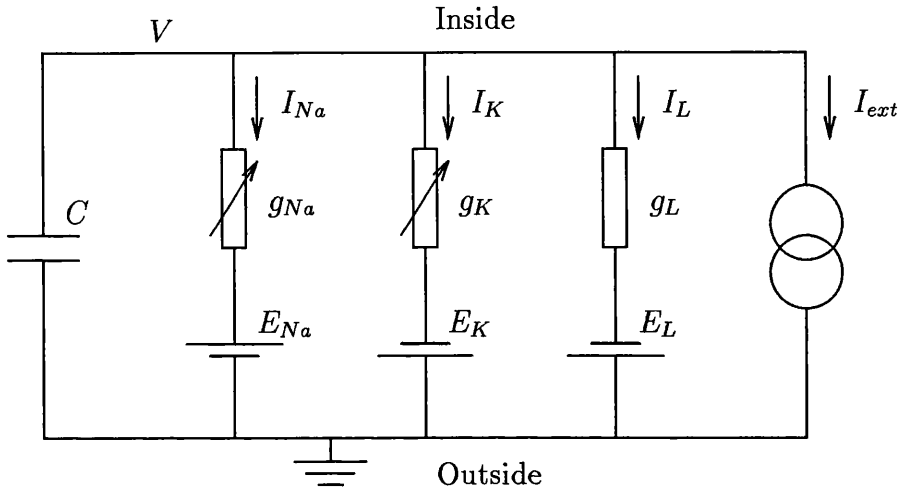

Figure 3: The Hodgkin-Huxley equivalent circuit of a membrane patch.

discovered by Hodgkin & Huxley [1952].

Hodgkin and Huxley measured the kinetics of the Na and K channels in the squid giant axon and calculated the shape of the action potential based on these measurements. Their formulation, which remains the standard expression of active channel dynamics, assumed the membrane is electrically equivalent to a capacitor with distinct leakage currents, as seen in the figure below.

They derived the following equations for a space-clamped axon, i.e. when the axon membrane potential is constant throughout its length and any external current is injected uniformly across the axon,

$$C\frac{dV}{dt} + I_{Na}(V,m,h) + I_K(V,n) + I_L(V) + I_{ext} = 0 \tag{1}$$

$$\frac{dm}{dt} = \alpha_m(V)(1-m) + \beta_m(V)m \tag{2}$$

$$\frac{dh}{dt} = \alpha_h(V)(1-h) + \beta_h(V)h \tag{3}$$

$$\frac{dn}{dt} = \alpha_n(V)(1-n) + \beta_n(V)n \tag{4}$$

where V is the membrane potential, C is the membrane capacitance, I_{Na} is the sodium current, I_K is the potassium current, I_L is the passive leakage current, and I_{ext} is the externally applied current.

The activation of the Na channels is represented by m and their inactivation by h. The activation of the K channels is n and they have no inactivation. The rate constants (αs and βs) are monotone sigmoids or exponentials. The αs are increasing with voltage for the activations m and n and decreasing with voltage for the inactivation h. The reverse applies to the βs.

The currents are given by

$$I_{Na}(V,m,h) = \overline{g}_{Na} m^3 h (V - E_{Na}) \quad (5)$$

$$I_K(V,n) = \overline{g}_K n^4 (V - E_K) \quad (6)$$

$$I_L(V) = \overline{g}_L (V - E_L) \quad (7)$$

where $\overline{g}_i$ is the maximum conductance and E_i is the reversal potential for ion i.

The (in)activation derivatives above are often equivalently expressed as (in the case of m)

$$\frac{dm}{dt} = \frac{m_\infty(V) - m}{\tau_m(V)}, \quad (8)$$

i.e. in terms of the steady-state (in)activation

$$m_\infty(V) = \alpha_m(V)/(\alpha_m(V) + \beta_m(V)) \quad (9)$$

and the time constant

$$\tau_m(V) = (\alpha_m(V) + \beta_m(V))^{-1}. \quad (10)$$

The propagation of action potentials is described by the diffusion equation, that is a second order partial differential equation in space and time. Due to the highly non-linear terms involved in this case, there is no analytic solution of the equation and one has to apply numerical techniques. The integration of general partial differential

equations is non-trivial. Hodgkin and Huxley managed to integrate their equation because they assumed the axon had uniform characteristics, e.g. axon radius, ion channel densities, etc. The solution is more commonly found by splitting the structure of a neuron into several compartments linked by an internal resistance and by diffusion coefficients. Each compartment is assumed to have uniform characteristics across its spatial extent and to be isopotential. This criterion determines the maximum size of the compartments and thus the number of compartments a particular neuronal morphology requires. Effectively, the original partial differential equation has been converted into many strongly coupled ordinary differential equations (Segev, Fleshman & Burke [1989]).

The Hodgkin-Huxley equations have become a framework for describing currents in nerve cells. Calabrese has discovered 10 currents in the leech heart interneurons that are all described by such equations (Calabrese & De Schutter [1992]). The task of integrating these equations is generally computationally expensive. Sometimes, these equations are stiff, i.e. the solution contains a wide range of characteristic time scales but only the slowest components are of interest. In order to maintain numerical accuracy, the system of equations has to be integrated with time steps on the scale of the fastest component. For example, the action potential currents operate on a fast time scale, whereas some other current components operate on time scales several orders of magnitude slower. All action potentials must be integrated to obtain the long-term behaviour of the system, although the shape of individual spikes on a long time scale appear insignificant.

Integrate-and fire models take advantage of the action potentials being all essentially identical. Since the action potential is nearly an impulse, its effects on post-synaptic cells do not depend on the dynamics of the Na and K channels involved. Getting [1983] implemented a particular variation: when the membrane potential exceeded a threshold, post-synaptic events were triggered. This included a recurrent

synapse that caused after-hyperpolarisation of the spiking cell. The spike induced post-synaptic conductance changes of fixed time course, where the effects of successive spikes accumulate. Following the spike, the threshold was temporally raised to mimic the refractory period of the action potential. An action potential produces effects in post-synaptic cells of much smaller bandwidth than itself. Thus, the integration of the system is speeded up by avoiding the integration of the fast action potential dynamics. This is obviously a great advantage if the behaviour of the system is to be observed at time scales larger than that of the action potential. The scheme still allowed Getting to include a Hodgkin-Huxley type current operating on a slow time scale.

Experimental evidence suggested that the neurons involved in the *Tritonia* escape response (Getting [1983]) were electrical compact, i.e. the cells were isopotential. Hence Getting modelled the neurons with single compartments. On the other hand, compartmental models have been created with thousands of compartments. Although the anatomy of such neurons can be accurately measured, little is known about the electro-physiological properties at the distant processes. This leaves large degrees of freedom in the choice of the parameters, and the models will inevitably become speculative.

The standard leaky integrate-and-fire neurons are even simpler models of neural function. The cell body is approximated by an RC circuit and pre-synaptic spikes produce current pulses that charge the capacitor. When the voltage exceeds a threshold, the cell fires and generates current pulses in the post-synaptic neurons. The charge is immediately removed from the capacitor and input current pulses are ignored during the refractory period. It has been shown that such a neuron can produce output frequencies that are either a weighted sum or a product of the input frequencies (Bugmann [1991]; Srinivasan & Bernard [1976]).

Further simplifications of the neuron have led to the field of artificial neural networks. Artificial neural networks are used to study higher cognitive functions, rather than the cellular details. Since the cell models are simple and analytically tractable it has been feasible to develop learning algorithms that adjust the parameters of the cells during a learning phase. Several applications within adaptive control, recognition, and classification systems have been based on artificial neural networks.

1.4 Tools for neuronal modelling

Several packages are freely available for modelling individual neurons and neurons in networks. The most popular packages include:

SPICE Bunow, Segev & Fleshman [1985] and Segev, Fleshman & Burke [1989] used SPICE to model active membrane mechanisms of compartmental neurons based on the Hodgkin-Huxley formulation. It proved to be awkward to express for example the rate constants in terms of SPICE elementary functions. Such a general simulation package does not achieve the same performance as a hand-coded version. Also, most good simulation programs are commercial and impose restrictions on the portability of the applications.

Genesis Wilson & Bower [1989] has developed an advanced simulation package for compartmental neurons called Genesis. Genesis runs on most UNIX platforms and is available for a small fee. Simulations of both hundreds of neurons as well as single neurons with hundreds of compartments have successfully been achieved with Genesis. It is designed for neurons and networks but still is a rather general simulation package and very hard to use, even though a graphical front end is available.

Nodus E. De Schutter has designed a tool for modelling compartmental neurons (De Schutter [1989]). His program is called Nodus and runs on MacIntosh computers. Its graphical user interface is particularly easy to use. Nodus supports hierarchical components, i.e. compartments are assembled from currents, which are again assembled from conductances. The sub-components are stored by name and can be included in future compartments. Neurons are assembled from compartments and interconnected by pre-defined synaptic interactions. Although easy to use, Nodus runs very slowly. It has therefore been proposed as a prototyping tool. During work on the leech heart model, the synaptic interactions had to be extended (De Schutter, Angstadt & Calabrese [1993]). This required recoding in Fortran. Since the source code is unavailable for public access, building new types of synaptic interactions, rate constants, etc., is infeasible for normal users.

NEURON M. Hines has developed NEURON which is a large package that chiefly models individual multicompartmental neurons or a few neurons in a network where the cable properties of the neurons play a crucial role. The program runs under several operating systems and there is a graphical user interface under X-windows. Several novel techniques have been used to speed up the numerical simulation of branching dendritic structures. There is a general configuration language to set up the model which is very flexible with respect to implementing membrane mechanisms. This language is translated into C code before being compiled and is therefore very fast.

Nemosys Nemosys has been developed at J. Miller's laboratory at Berkeley. It runs under Unix and X-windows and has a very nice user interface. Currently, it is limited to modelling a single multicompartmental neuron. The membrane potential distribution throughout the neuron is displayed in colour codes.

Neurodynamix W. O. Friesen has developed a modelling tool running on IBM PCs called Neurodynamix. This differs from the previous packages in that it has separate sub-programs for simulating patch clamping, somas, axons, neurons, and circuits. It has an intelligible graphical interface and good performance. However, the structure of the simulations is fixed. For example, 4 neurons are studied in the network part, where the parameters of the cells and the synapses can be varied. Friesen is currently running a project to develop a more flexible tool for studying leech swimming, and where, so far, I have contributed with the major work. He approaches the modelling from the network aspect in that he is interested in a network of 10-100 cells coordinating the leech swim pattern. The cell models are therefore simpler with the ionic currents approximating Hodgkin-Huxley kinetics.

SWIM Ö. Ekeberg and A. Lansner have developed SWIM running under Unix. It is centered around a specification language that allows compartments to be created that inherit parameter values from other compartments. It is therefore painless to create large networks with nearly identical neurons. The specification script is compiled, run, and then plotted. SWIM comes with a range of pre-defined mechanisms, but the advanced user will eventually reach a point where his particular synaptic transfer mechanism, say, may be impossible to implement in the SWIM specification language. To see the effects of a parameter change, the script has to be explicitly edited, compiled, run, and then plotted.

Other Getting [1989] simulated the escape response in *Tritonia* using the integrate-and-fire model describe earlier with his own system, MARIO. Earlier simulations were performed with programs provided by D. Perkel called MICKEY and MANUEL.

Some are general simulation packages that have many features. Thus, most or all models can be implemented, but such packages tend to be inconvenient to use because they are very general. To implement a particular model involves therefore a lot of programming effort, often in a language that is specific to the package. The more specialised packages aimed at simulating neurons and networks tend to be easy to use but restricts the range of models they can simulate. The documentation tend to be poor and the programs require considerable programming skills to be adapted to new classes of problems.

Sheperd [1992] stresses the importance of using several programs in their research. The large, commercial general purpose simulation packages offer good documentation and support. The efficient neural simulation programs are used in the exploratory phases, and the display phase is done on the program with the best graphics. This approach also eases collaborations, since different collaborators prefer different packages.

In summary, the available neural modelling tools are either too general to be efficient or too specific to be applicable to new problems. There is also the issue of portability between different hardware platforms. Ideally, one would like an intuitive graphical user interface during the prototyping stage and then run the simulations on the platform with the highest performance. Only Neuron is to some degree portable, but none of the above packages run on parallel computers. Hence it was decided to implement a neuronal modelling tool that was fast, flexible, easy to expand, and platform independent. Chapter 2 describes the design of the tool, named *Neurolab.*

1.5 Methods of analysis

The objective of analysing nervous systems is to identify the processes involved in neural computation. The analysis involves identifying an activity measure of the

neuron, as well as a transfer function with respect to that measure that relates the output activity of the neuron to the activities of the input neurons. Learning can then be described in terms of changes in the transfer function. Note that the transfer function will not be a function in the mathematical sense, since it will include internal state variables.

The activity measure should represent the state of a neuron as seen by other neurons. One should therefore consider how the signals are detected and transferred at the synapses.

Definition 1 *The total activity of a neuron is a set of state variables of the neuron such that the only inputs to the inter-neuronal communication process between two neurons are the total activities of the two neurons.*

Definition 2 *The output activity is a set of state variables that represents the input from all converging neurons and from which the neurotransmitter release at the diverging synapses is uniquely defined.*

In a traditional vertebrate neuron, synapses converge onto the dendrites where synaptic potentials are integrated. Action potentials are triggered at the axon hillock and travel down the axon towards the diverging synapses. Vertebrate synapses release neurotransmitter only when action potentials appear at the pre-synaptic terminal, so the only input to the inter-neuronal communication process is the arrival of action potentials. Since there are no other external inputs that affect the state of the axon, the action potentials at the axon hillock are a good measure of the neuron's output activity. The action potentials are of short duration compared to the inter-spike intervals, and the output activity can therefore be considered as a sequence of point events, i.e. a series of Dirac delta-functions (e.g. Rosenberg et al, 1982).

Note that as long the neuron has the feed-forward architecture outlined above, the spike events at the axon hillock are a sufficient total activity measure. However, if

some post-synaptic state affects the transmission of a synapse and another converging synapse affects the same state, this state will have to be included in the total activity measure. For example, the membrane potential and ion concentrations at a node in the dendritic tree that is close to two converging synapses, where these states affect the synaptic transmissions, qualify to be included in the total activity set. The total activity measure will therefore be multi-dimensional for complex dendritic interactions. It follows that the output activity is a subset of the total activity. The state variables of the pre-synaptic calcium concentrations and neurotransmitter stores of the divergent synapses as well as the postsynaptic conductances of the convergent synapses must, in general, be included in the total activity.

However, invertebrates have graded synaptic transmission as well as high threshold spike mediated transmission (Calabrese, Angstadt & Arbas [1989]). In this case, the amount of neurotransmitter release depends directly on the presynaptic membrane potential and is effective even at low membrane potentials. The output activity therefore contains a continuous component in addition to the discrete spikes.

The continuous activity can not be transmitted over long distances, so the network must be spatially small. On the contrary, action potentials are regenerative and propagate without decay. This is a perfect mechanism for relaying signals over long distances although a discrete event code is not as effective to transmit information as a continuous code. To show this, consider the sequence of inter-spike intervals, which is equivalent to the spike train. This sequence can be thought of as a continuous inter-spike signal sampled at the occurrences of the spikes. Consequently, the sampled signal carries less information than the continuous signal. Spike coding has therefore developed where information has to be carried over long distances, but at some loss in efficiency, and graded synaptic transmission where there are closely connected neurons. It follows that one should be careful to infer that cellular properties discovered in vertebrates exist in invertebrates, and *vice versa.*

After having identified the state variables where the neurons interact and consequently conjectured an activity measure, one can study the transfer function of neurons. Several methods have been developed for studying the transfer function of neurons for both spiking, i.e. discrete events, and continuous signals.

1.5.1 Point process analysis

The outputs of a point process are events, i.e. points in time, rather than a continuous signal. Analyses of point processes applies to examining the action potentials generated by neurons, since the width of the action potentials is small compared to the inter-spike interval.

A typical experimental set up involves triggering a stimulus and recording the response in a neuron. Many stimulus-response pairs are collected and analysed in a peri- (or post-) stimulus histogram. The time of the spikes in a response are translated so that the time of the stimulus becomes the origin of time. The time axis is divided into bins where the number of spikes within a bin are accumulated. This will show the average distribution of spikes throughout the time just after a stimulus and has been the most common way to analyse the spikes of neurons.

The train of spike events is often transformed into a quasi-instantaneous frequency by inverting the interval between successive spikes. This gives an instantaneous frequency that is constant between spikes and is discontinuous at the spikes. Alternatively, Richmond et al. [1987] and MacPherson & Aldridge [1979] substituted each spike with a Gaussian impulse to obtain a continuous instantaneous frequency. This procedure can be viewed in the framework of density estimation as estimating the spike probability density (Silverman [1986]). Such a measure is continuous and is mathematically better behaved. The original spiking signal can then be studied with the continuous analyses described in the next section.

Brillinger has formalised the analysis of stochastic point processes (Brillinger

[1975]). Rosenberg *et al.* have developed this analysis further to study the muscle spindle transfer function in the frequency domain (Halliday, Murray-Smith & Rosenberg [1992]). A point process N is characterised by the number of events, $N(t)$, between time 0 and time t. The differential increment of the process, $dN(t) = N(t + dt) - N(t)$, is the number of events within a small time interval dt. The mean intensity, P_N, of the stochastic point process is defined as

$$P_N dt = E\{dN(t)\}$$

where $E\{\ \}$ is the expectation operator over the duration of the observation. The second order cross-product density at lag u, $P_{NM}(u)$, between the point processes N and M is defined as

$$P_{NM}(u)dudt = E\{dN(t+u)dM(t)\} \quad u \neq 0.$$

The cross-variance density, $q_{NM}(u)$, is given by

$$q_{NM}(u) = P_{NM}(u) - P_N P_M$$

and the cross-spectrum, $f_{NM}(\lambda)$, where λ is the frequency, is

$$f_{NM}(\lambda) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} q_{NM}(u) e^{-i\lambda u} du \quad u \neq 0$$

and the auto-spectrum of process M is

$$f_{MM}(\lambda) = \frac{P_M}{2\pi} + \frac{1}{2\pi} \int_{-\infty}^{+\infty} q_{MM}(u) e^{-i\lambda u} du \quad u \neq 0.$$

Brillinger [1975] has derived an identification procedure for a bivariate point process that is stationary for stationary input, M, and that has output N. The linear model of the process is characterised by the conditional probability of an output spike, given an input spike train,

$$E\{dN(t)/M\} = \left[s_0 + \int s_1(t-u) dM(u) \right] dt$$

where s_0 is the output in absence of any input, and $s_1(t-\beta)$ is the effect on the output of an input spike at time β.

The transfer function, $S_1(\lambda)$ can now be found in terms of the cross- and auto-spectrum as

$$S_1(\lambda) = \frac{f_{NM}(\lambda)}{f_{MM}(\lambda)}$$

and impulse response, $s_1(t)$, is the inverse Fourier transform of the transfer function

$$s_1(t) = \frac{1}{2\pi} \int S_1(\lambda) e^{i\lambda t} d\lambda.$$

The coherence

$$|R_{MN}(\lambda)|^2 = \frac{|f_{MN}(\lambda)|^2}{f_{MM}(\lambda) f_{NN}(\lambda)}$$

measures the degree of linear correlation between the two processes. The linear model of the relation between fusimotor input and Ia output had significance at frequencies up to 60 Hz (Rosenberg, Murray-Smith & Rigas [1982]). The model has been expanded to include higher order terms (Brillinger [1975]) as well as several inputs. Rosenberg, Murray-Smith & Rigas [1982] also included a continuous input, the muscle spindle length, in the framework. Halliday, Murray-Smith & Rosenberg [1992] present the computational methods for estimating the finite Fourier transform and the spectrum of a point process.

These point process techniques characterise the system using stationary random signals. In particular, the above studies used a Poisson process as input, whose spectrum has a constant value for all frequencies. For this input, a linear model for the neuromuscular system has been identified, and it was shown to reproduce a phenomenon found experimentally (Halliday, Murray-Smith & Rosenberg [1992]).

However, Optican & Richmond [1987] have shown that there exists information about visual stimuli in the temporal modulation of spike trains. The output of CPGs is oscillating, i.e. temporally modulated. Therefore, in general, neurons operate biologically with non-stationary signals. The neurons will comprise many activities

that will interact with different time-scales, some considerably longer than that of the action potential and shorter than the duration of the observation. Since nervous systems compute with non-stationary signals, one should search for methods for analysing non-stationary signals and systems. Two techniques are favoured below: wavelet analysis and principal component analysis.

1.5.2 Wavelet analysis

The Fourier transform, $X(f)$, of the signal $x(t)$ is given by

$$X(f) = \int x(t)e^{-2i\pi ft}dt \tag{11}$$

where the coefficients $X(f)$ represent the global frequency f in the signal. The transform works well if $x(t)$ is composed of a few sinewaves, but an abrupt change in a signal $x(t)$ is spread out over the whole frequency axis in $X(f)$. To analyse non-stationary signals, one can introduce time dependency in the Fourier analysis and map the signal onto a two-dimensional time-frequency plane. Usually, a local frequency parameter is introduced such that the local Fourier transform, i.e. local in time, transforms a window of the signal where the signal is approximately stationary. Equivalently, the sinewave basis functions of the Fourier transform can be modified to basis functions that are more concentrated in time (and thus less concentrated in frequency).

Gabor [1946] first adapted the Fourier transform to define the signal $x(t)$ in the time-frequency representation as $S(\tau, f)$. The signal is seen through a window $g(t-\tau)$ of limited extent that is centered at time τ. The Fourier transform of the windowed signal $x(t)g^*(t-\tau)$ becomes

$$S(\tau, f) = \int x(t)g^*(t-\tau)e^{-2i\pi ft}dt. \tag{12}$$

For any observation of a signal, the resolution in time and frequency is lower

bounded by the Heisenberg uncertainty principle,

$$\Delta t \, \Delta f \geq \frac{1}{4\pi}, \tag{13}$$

i.e. one can only trade time resolution for frequency resolution, and *vice versa.* For a given window $g(t-\tau)$, the time and frequency resolution of the Gabor transform are constant at all times and frequencies (Rioul & Vetterli [1991]), and Gaussian windows are favoured since they meet equation (13) with equality (Gabor [1946]). The Gabor transform allows a signal that is composed of small bursts and long quasi-stationary components to be analysed with good time resolution or good frequency resolution, but not both.

The wavelet transform sets the frequency resolution Δf proportional to f. The time resolution then becomes arbitrarily good at high frequencies and the frequency resolution arbitrarily good at low frequencies whilst still obeying the Heisenberg inequality (13). The family of the basis functions used in the transform depends on the signal. A typical simplification is to define the basis functions to be scaled versions $h_a(t)$ of the same prototype $h(t)$. The wavelet transform then becomes

$$S(\tau, a) = \int x(t) h_a^*(t-\tau) dt \tag{14}$$

$$h_a(t) = \frac{1}{\sqrt{|a|}} h(\frac{t}{a}) \tag{15}$$

where a is a scale factor and represents the local frequency. To see this, the wavelet prototype could be chosen as in the Gabor transform

$$h(t) = g(t) e^{-2j\pi f_0 t} \tag{16}$$

with the relation between local frequency f and scale a

$$a = \frac{f_0}{f}. \tag{17}$$

Equation (14) calculates the inner product of the signal $x(t)$ and the wavelets $h_a(t)$. The signal can be synthesised by adding up the projections of the signal on the wavelets as

$$x(t) = \int S(\tau, a) h_a(t - \tau) da d\tau. \tag{18}$$

Bartnik, Blinowska & Durka [1992] have applied a wavelet transform to evoked potentials generated by auditory stimuli. The responses were represented as an expansion of wavelets at different frequency scales. They were able to classify the stimulus based on only a few coefficients of the transformed signals. Thus the transformed signal space offer analysis that complements the original signal space. Similar analysis and dimensionality reduction has been performed on ECG signals (Crowe et al. [1992]). However, in general, it is not obvious which components at which scales contain important information. Therefore, the wavelet analysis tends to spread the signal out in the frequency-time plane which may provide extra information to the human eye but is undesired for quantifying the signal. On the contrary, principal component analysis significantly reduces the numbers required to quantify a signal.

1.5.3 Principal component analysis

Biological neural systems show complex behaviours that are generally difficult to relate to a particular phenomenon at the cellular level. Extensive feedback in neural systems contributes significantly to this difficulty. The observed membrane potential is coupled with the ionic concentrations and the channel states at the point of observation. Axonal propagation and diffusion through the regions of a cell provide spatial coupling of these states. Additional feedback loops will arise through synaptic interactions in a network of cells. Presently, only the membrane potential at the cell body is readily available for observation, although imaging techniques for displaying ionic concentrations in a cell as well as the membrane potential throughout the cell

are advancing (Ross, Arechiga & Nicholls [1987]). The multitude of interactions that are implicitly observed in the membrane potential makes it difficult to relate the instantaneous values of the membrane potential to the behaviour of the animal.

Recordings from both invertebrates and vertebrates demonstrate that neurons show great variation in spiking frequency and pattern. Generally, the activities of cells vary with time, and the activities as functions of time represent information. The time course of the activities will incorporate the effects of the feedback loops in the system. Thus, being able to quantify a pattern of activity over time, rather than just the instantaneous values, will greatly enhance our chances of understanding the computation in these systems.

Experimental recordings from neurons are generally difficult to quantify. Spike patterns and membrane potential cycles are sometimes compared qualitatively and are hence limited by human resolution and judgment. Conventionally, spike patterns have been quantified in terms of peri-stimulus histograms, Fourier transforms, and coherence analysis (e.g. Rosenberg, Murray-Smith & Rigas [1982]). These techniques all display the average behaviour of neurons over many repetitions and cannot compare single activity patterns.

The Karhunen-Loève transform is an alternative statistical method for describing temporally modulated patterns. Richmond et al. [1987] and Richmond & Optican [1987] used this approach to quantify the neuronal responses to visual stimuli in the monkey visual cortex. The responses of the neurons were sampled for 380 ms following a presentation of a stimulus. The Karhunen-Loève transform reduced the complex spike patterns to 3 numbers for each response whilst preserving the most significant features of the pattern. In contrast, a peri-stimulus histogram would only quantify the average of all the responses. Now, only having to work with 3 numbers per response, Optican & Richmond [1987] were able to further analyse the computation in the visual pathways.

Chapter 4 will present the mathematical background of the Karhunen-Loève transform. Chapter 5 will describe how the Karhunen-Loève transform was applied to quantify the bursts of spikes in the crayfish swimmeret and the leech swimming systems.

1.6 The goals of this thesis

This thesis aims to study the temporal interaction of neuronal state variables in small nervous systems. Since each neuron is very complex, the size of the networks has to be reduced to enable easier access to the neural functions. Central pattern generators (CPGs) have the advantage of producing repeated motor activity that can easily be correlated with behaviour. The following studies will therefore be concerned with CPGs. Chapter 3 will describe CPGs in further detail.

The thesis will reflect the multidisciplinary nature of the study of nervous systems. Techniques from a variety of disciplines will be applied to investigate the temporal interactions of state variables in the leech heart and the leech swimming systems.

Techniques from electrical engineering, mathematics, and statistics will be combined to allow the quantification of behaviour as well as the neuronal signals in a common signal space. This will allow the signals to be compared and correlated, and thus assign behavioural function to the neuronal signals.

Object-oriented techniques from computing science will be applied in the design of a new modelling tool that will reduce the complexity of the program, although maintaining flexibility. This tool will enable one to explore activity states within small networks and to study the behavioural relevance of the temporal variations in the states.

The tool will be aimed to be highly portable. In particular, it should be able to run on the parallel transputer-based computer in the department. Meanwhile, the

object-oriented design will allow a large fraction of general code to be written. Thus, it will be feasible to add new features with a minimum of effort. No program for neuronal modelling has been written using an object-oriented structure, so the design of the tool will be given attention accordingly.

Chapter 2

Neurolab—A neuronal modelling tool

As long as a large number of the neuronal state variables are experimentally inaccessible, more or less detailed computer models can be used to estimate these states and their interactions. The behaviour of the model can be compared to that of the real system and provide information about the accuracy of a particular model. If, for a given model, no set of parameters exists that approximate the real system, there are structural inadequacies in the model. On the contrary, if there is a range of parameters that equally well models the real system, it encompass the diversity commonly found in biological systems and provides a region in the parameter-space where the model works. In either case, new experiments can then be designed to address the structure or the actual parameters of the system, based on information from the state variables of the model.

This chapter describes the organisation of an object-oriented simulation tool that was designed for the modelling of 2–100 neuron networks, where each neuron incorporate several ionic currents. As mentioned in Chapter 1, there is no existing neuronal modelling tool that combines flexibility, speed, portability, and ease-of-use.

2.1 Object oriented programming and simulation

Object oriented programming is highly applicable to simulation, including the modelling of neurons in networks. In principle, the object oriented programming paradigm bundles data into objects, and the program sends messages to the miscellaneous objects.

The objects belong to particular classes, where objects of the same class receive the same set of messages. A new class can be built from an existing class, where the new class inherits all the properties of the class it was based on. The new class typically modifies its behaviour slightly from that of the base class.

In neuronal modelling, the neurons are similar, but not identical. Their function, as in receiving synaptic input and releasing neurotransmitter, is the same, but the implementation of these processes varies greatly between different neurons. When a large number of neurons are assembled in a network and a traditional programming language is used, many large if-then-else structures are required to assure the appropriate piece of code is executed for each type. As a modelling program increases in complexity it soon becomes prone to bugs and hard to maintain, since old pieces of code have to be updated when new neuronal models are incorporated into the program.

When using an object oriented programming language, one can define a base class for neurons and from that derive classes for particular types of neurons. As long as these classes of neurons are derived from the base class, the above if-then-else structures can be replaced with sending a single message to the base class, which then passes it on to the particular implementation of that class. Similar base classes can be defined for synapses, compartments, ion channels, and currents, which all appear in different forms. Object oriented design greatly simplifies the modelling program, which is processing a set of base class objects. It does not need to know which

particular derived class any of the objects belongs. The main modelling program does not have to be changed when new types (classes) of neurons, synapses, compartments, ion channels, and currents are included in the model.

It is important to distinguish between a class and an instance. A class is the abstract declaration of an object in terms of what data variables it comprises and what functions can be applied to an instance of the class. The instance is the physical realisation of the class in the memory of the computer. A class can have many instances which will differ in the values of their variables. However, the types of the variables are common, as defined in the class.

As far as I am aware, no other modelling tool for biological realistic neurons exists that benefits from the advantages an object-oriented language provides. However, a few neural network systems have already employed the paradigm (Ran & Karjalainen [1991]).

2.2 Integration

The standard cell model in Neurolab has a single compartment with user defined currents and synaptic currents. The membrane potential of the cell, V, is governed by

$$C\frac{dV}{dt} + \sum_i I_i(s_i) + \sum_j I_j(s_j) = 0 \tag{19}$$

where C is the membrane capacitance, and s_i is the state vector of current i. In the Hodgkin-Huxley formulation the state vector would be (V, m, h) for the Na^+ current and (V, n) for the K^+ current. The size of the state vector will vary for the different types of currents. The synapses are indexed by j and their states s_j may include pre-synaptic Ca^{2+} and neurotransmitter concentrations and the pre- and post-synaptic membrane potentials.

No other assumption about the structure of the model is made. Note that a

multi-compartmental cell model can be realised by linking "cells" with resistive synapses. This is entirely within the framework of the tool. However, models of multi-compartmental cells that include ion concentrations and diffusion is beyond this framework and require a (slight) modification to the implementation.

All states are contained in a central state vector for efficiency as well as for enabling the use of standard integration libraries. A cell contains a state reference to a particular location in the state vector where the value of its membrane potential is stored. Likewise, the currents use state references to index the central state vector to access for example the (in)activations and calcium concentrations. Such a mechanism also enables one to index the Jacobian matrix by state references.

Before performing the integration, each cell and current allocates the space they require in the central state vector. The simulation makes an instance of the class `NetOde` that keeps a list of the cells incorporated in the simulation. During the integration, the function that calculates the derivative of the state vector is redirected to the functions that calculate the derivative of each cell which again redirects the call to each of the currents in that cell. This allows the information required to calculate the derivative of a cell or a current to be stored locally. The code becomes modular, clear to read, and easy to expand with more cells or more currents, that can be of any type, as the binding between the various pieces of code is kept at a minimum.

Three methods of integration is currently implemented. The default method is LSODA developed by Alan C. Hindmarsh and Linda R. Petzold in FORTRAN (Hindmarsh [1983]; Petzold [1983]). This code was converted using a public domain FORTRAN to C converter and an interface in C++ was added. The package automatically switches between stiff (5th order BDF) and non-stiff (12th order Adams) integration methods. The stiff method is an improved method of Gear's algorithm (Gear [1971]).

The second choice is an adaptive step size Runge-Kutta method of 4th order (Press et al [1988]). It was extensively used during the development and now serves

as a reference since it is a standard and extensively used method in other simulation packages.

The third choice is a fixed step size method developed by W.O. Friesen that for all state variables calculates the steady state value and the time constant assuming all states remain constant. The next value is approximated by an exponential decay from the current value. As long as the time step is short enough, this is a stable method for integrating stiff differential equations. However, it is unable to detect when it safely can take longer steps without loss of accuracy.

Generally, the LSODA package proved to be the faster and more reliable of the three options, and it is the default method of integration.

2.3 Class hierarchy

Neurolab has three fundamental classes: `Cell`, `Synapse`, and `Current`. Of these, only `Current` is currently inherited by other classes that implement specific currents.

Typically, an operation is performed on a cell and then redirected within the cell to all currents and synapses. Such an operation could be calculating the derivative, registering parameters, and registering states. Figure 4 shows how the central parts of the tool views a cell, and how the function calls are directed to the derived classes.

This enables large parts of code to be written that handle objects (i.e. cells, currents, and synapses) in a general, abstract manner. The specific details about operations are handled within the objects themselves without the central parts of the program having to need in any detail how it was done.

Thus, the central core of the program can cope even with new classes that were unknown at time the program was written. The new classes specify *themselves* how the abstract operations are handled. This obviously requires the designer of the program to define a sensible interface between the central core and the various objects,

thus taking into account future requirements.

Since the main part of the program is independent of new classes, new types of currents and synapses can easily be incorporated into the tool. No changes have to be made in the central core, since all actions on the new class is defined within the class itself. Thus, it is very easy to extend the tool further, for example by defining new types of currents or modifying existing ones.

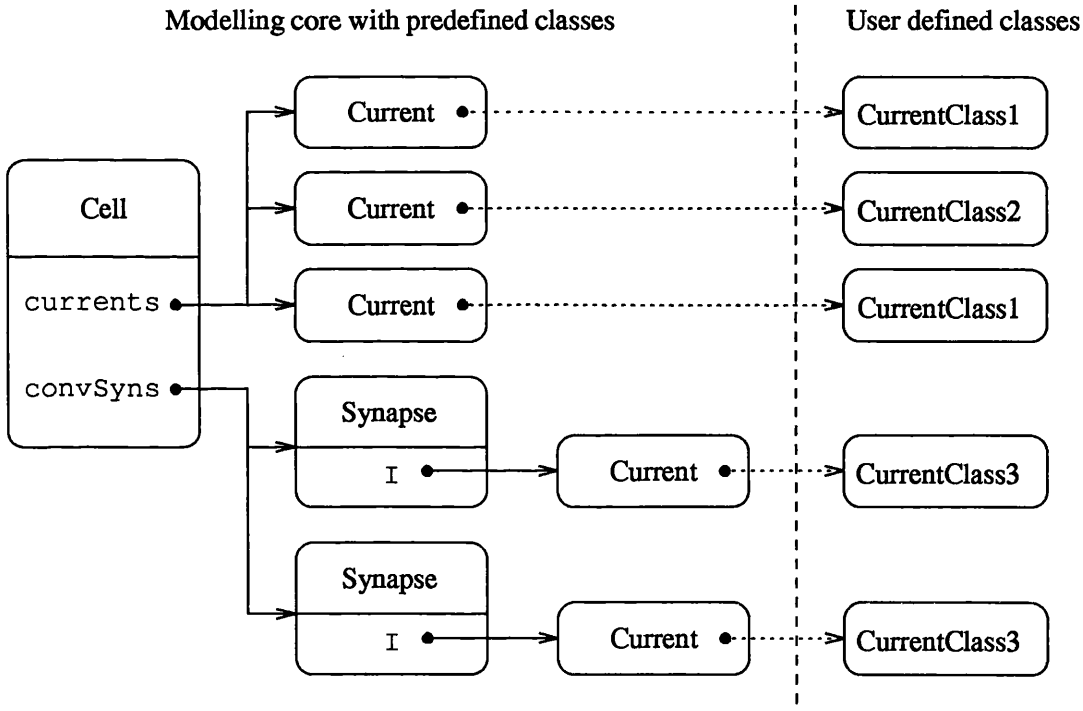

Figure 4: A cell's references to synapses and currents. A cell keeps a list of the currents operating in the cell and a list of synapses converging onto the cell. The solid arrows indicate references to instances of currents and synapses. In this example, 5 currents are operating in the cell: 3 recursive currents and 2 synaptic currents. The dotted arrows show how the virtual functions in the class `Current` are passed on and executed in the user defined classes. The area left of the vertical dashed line represents the scope of the central operations, and the area to the right represents instances of the user defined classes. The figure demonstrates that currents can be of many classes.

2.4 The database mechanism

A database mechanism for assigning values to parameters was designed for general programs, not only the modelling tool. This avoids recompiling programs when runs with different sets of parameter values are required. It also serves as an interface to reading and assigning the state values of the simulation.

A class `DataBase` was implemented to store a list of associations between a name and an instance of the class `DBentry`. The class `DBentry` defines an interface to assignments and reading values, and the actual implementation is deferred to derived classes. Thus the `DataBase` class does not have to be changed when new types of entries are defined and stored in the database. The following two functions have to be redefined in derived classes of `DBentry`.

`virtual int Assign( const char *value );` This function assigns the character string `value` to the entry.

`virtual void Report( ostream& s );` This function reports the value of the entry to the output stream `s`.

Entry classes are defined for all the fundamental types of C/C++: double, float, int, and string, and some other entries take advantage of the call-back mechanisms of C++. For example, a class is derived from the string entry to represent the name of a cell in the graphical interface (see section 2.6). The `Assign` function is redefined so that it first performs the inherited string assignment and then changes the graphical label displayed in the cell. Likewise, the return value of a function can be an entry whose `Assign` function does nothing and the `Report` function outputs the return value of the function. This mechanism is used to output the instantaneous values of the conductances and currents. These values are not stored in states but are functions of the states. Two classes are defined for this purpose. They both store a reference to the current and the central state vector and redirect the `Report` call to the current's

`g(state)` and `I(state)` functions, respectively.

An entry is registered in the database with a name and a type. The type may be either a parameter, state, or a function. Typical source lines in the application is

```
int n; Database db; db.Register("N", new IntEntry(&n),
DBentry::param);
```

which registers an integer parameter referenced with the name `N` in the database `db`. The value of the integer is stored in the variable `n`.

Assignments can be made on the command line when invoking the modelling program. Assignments have the form `<name>=<value>` or `<filename>`. In the latter case, assignments are read from the file specified. Typical commands would be

```
tool cfg N=4
tool cfg N=7
```

where the program `tool` is run with default configuration assignments in the file `cfg`, followed by assignments particular to each run. These runs would assign the values 4 and 7, respectively, to the variable `n` in the preceding example. Note that default values of `N` can be specified in the file `cfg` and then modified on the command line.

The parameters, states, and functions are accessed in an hierarchical manner. Database entries defining a cell are referenced by the name of the cell and the name of the entry, separated by a ":". Entries defining a current are referenced by the cell name, then the current name, and finally the name of the entry. For example,

```
HNL:Vm
HNL:ICa:h
```

describe references to the membrane potential of cell "HNL" and the activation "h" of the current "ICa" in the same cell, respectively.

2.5 Derived current classes

In order to implement a new current class, one must first define the state references and the parameters of the current. Then the functions `ODEregister`, `RegisterDB`, `ClassName`, `Init`, `g`, `CalcDeriv`, and possibly `Step` must be written. These changes to the tool are all local to the new class and require no update of any other parts of the code. Finally, the new class must be registered in the current loader by adding one line in that module. This is the only change to the modelling core. The new current class can now be used as any other current in the modelling core as well as the graphical user interface.

Currently, there are 4 pre-defined current classes and 2 pre-defined synaptic current classes. Two current classes implement the standard Hodgkin-Huxley currents with one and two activation states, respectively, as shown in equations (1–4). To each activation state there are two rates α and β that each have the form used in Nodus

$$\frac{aV+b}{c+\exp\frac{V+d}{e}}. \tag{20}$$

A base class `Rate` was created to provide the interface to derived classes that perform clipping if the rate is negative and approximation at the potential singularity. This allows none, one, or both operations to be performed on the value of equation (20). By choosing the right class in the initialisation of the simulation, only a minimum of extra operations are executed. Further gain in execution speed can be achieved by deriving classes that approximate equation (20) and experimental activation curves by Chebyshev polynomials, interpolation tables, etc. Again note the benefit of the object-oriented design: the implementations of the two current classes are invariant; it suffices to initialise the instances of the currents with instances of the new rate classes.

The `g(state)` function returns the product of the peak conductance and an integer power of the activation state(s).

Two additional current classes implements a slightly simpler model of the Hodgkin-Huxley currents, again with one or two activation states. Instead of being based on the rates, it is parameterised by the steady state value of the activation(s)

$$m_\infty = \frac{1}{1 + \exp \frac{V+V_h}{V_s}} \tag{21}$$

where V_h is the half point of the sigmoid and V_s defines the slope of the curve at the half point. Note that deactivation curves can be obtained by setting V_s negative.

The activation then progresses as in equation 8 but with the time constant τ being constant. The `g(state)` function returns the product of the peak conductance and the activation state(s). It uses `Current`'s `I` function. The function `RegisterDB` adds V_h, V_s, and τ for each activation state to the database.

This form requires fewer parameters to be specified and is easier to relate to biological data. A change in one of the parameters is easy to picture mentally, whereas the curve of equation 20 must be graphically verified. However, the simpler form may not as accurately describe the biological data where such data are available.

Two forms of synaptic currents are pre-defined. The synaptic currents are derived from the class `SynCurrent`, which is a `Current` with an additional variable: a reference to the synapse it is representing.

The first synaptic current is used by W.O. Friesen in Neurodynamix. It calculates the synaptic drive, d, as

$$d = \Theta(V_{pre} - V_{th}) \tag{22}$$

where V_{pre} is the pre-synaptic membrane potential, V_{th} is the threshold for neurotransmitter release, and Θ is defined by

$$\Theta(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

The post-synaptic current is modulated by an activation state, a, that implements synaptic fatigue. The synaptic conductance is the product of the peak conductance,

the synaptic activation, and the synaptic drive. The synaptic current is the product of the conductance and $V_{post} - E_{rev}$, where V_{post} is the post-synaptic membrane potential, and E_{rev} is the synaptic reversal potential.

The dynamics of the activation is defined by

$$\frac{da}{dt} = \alpha(1 - a) - \beta a \tag{24}$$

where α is the rate of recovery and β is the rate of decay. During prolonged high synaptic drive, the activation will fall and thus reduce the post-synaptic current, and for low synaptic drive, the synapse will recover as the activation is increased. The `RegisterDB` function adds the parameters V_{th}, α, and β to the database. E_{rev} and the peak conductance are registered by explicitly calling the function of the same name belonging to `Current`.

The second form of synaptic current is a particular mechanism discovered in the leech heart (De Schutter, Angstadt & Calabrese [1993]). It is very specific to the leech heart and will be described in Appendix B.

2.6 The graphical user interface

Of existing neuronal modelling tools, several have graphical user interfaces. However, the specification of networks in Nodus is performed using menus and dialog boxes, whereas Genesis has a complex language to define neurons and networks. None of the existing tools combine the simulation with a graphical display of the neurons and synapses that can be interactively manipulated.

A graphical user interface to the modelling core was developed using the InterViews library for X-windows. The design goal was to minimise the binding between the graphical interface and the modelling core but still achieve a functional user interface. The modelling core remains the same, and the graphical interface uses only a

minimum of information about the modelling core. It is therefore convenient to prototype a model in the graphical interface and then proceed to the textual interface when many runs with different parameters are desired.

The user interface, called Ned, is illustrated in Figure 5 as seen on the computer screen. The tools provided along the right side are implemented as radio buttons, i.e. when one tool is pressed the previously selected tool is unselected. The action of a click with the left mouse button on the drawing area to the left depends on which tool is selected:

Cell A new cell is created centered at the position of the click. The name of the cell automatically becomes "Cx", where x is the number of cells in the simulation.

Move If there is a cell under the cursor, the cell is dragged while the button on the mouse is pressed and placed at the new position when the button is released.

Param If there is a cell or a synapse under the cursor, the entries registered as parameters are shown in a list, and the value of the parameters can be edited.

State If there is a cell or a synapse under the cursor, the entries registered as states are shown in a list, and the value of the parameters can be edited. There is a bug in that the values, as shown in the list, are not updated when the simulation is run.

Probe If there is a cell under the cursor, the cell's membrane potential is added to the group of cells that are plotted in the graph window. If the cell is already plotted, it is removed from the plot.

Delete If there is a synapse under the cursor, it deletes the synapse. If there is a cell under the cursor, it deletes all synapses onto the cell and then the cell itself. This tool is currently not yet implemented.

Current If there is a cell under the cursor, a pop-up dialog box appears asking for the name and type of the current. A new current is then created and added to the cell selected.

Synapse If there is a cell under the cursor, it is selected as the pre-synaptic cell. While keeping the mouse button, the user can move the cursor to a post-synaptic cell, release the button, and a new synapse is created between the two cells. Currently there is no choice of the type of synaptic current; the Friesen synaptic current is used. The symbol appearing next to the post-synaptic cell is determined by the 2 radio buttons below the Synapse tool. These can select whether the synapse symbol is inhibitory, represented with a filled circle, or excitatory, represented with a filled rectangle. However, in the simulation, the sign of the synapse is determined by the reversal potential. This can be changed using the Param tool, but note that this will not update the symbol of the synapse.

A click with the right mouse button on the drawing area displays a menu where one can choose to quit the application, save the simulation including the graphical picture, load a simulation and picture, run the simulation, and stop it. The simulation is run for 1000 time units whilst plotting the probed cells in the graph window.

2.7 A parallel implementation

Neural modelling usually involves running many simulations with different parameters. The task of interactively setting the parameters, then running the simulation, checking the results, and choosing new parameters is time consuming and tedious. This applies particularly to complex models with many parameters that are inaccessible by experiments.

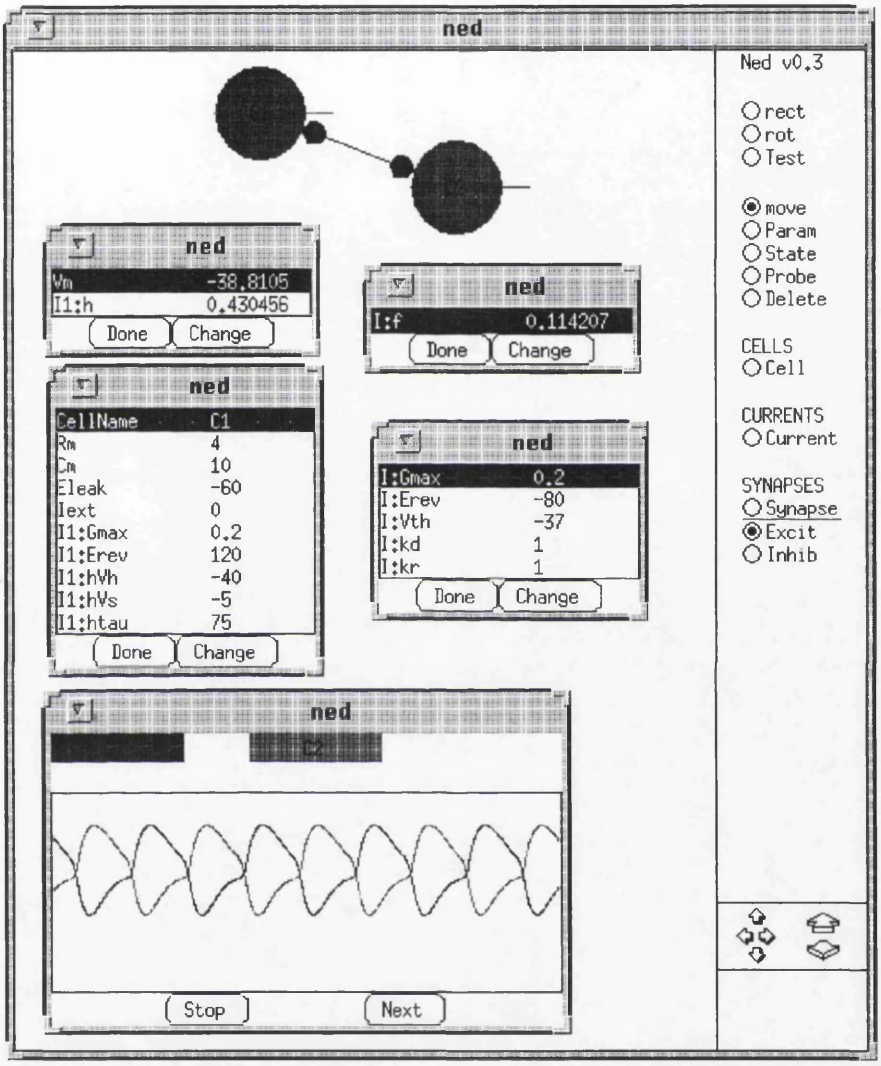

Figure 5: A snapshot of the graphical user interface in use. At the top, a network of two recurrent inhibitory cells can be seen. The cells are the large circles and the inhibitory synapses are the smaller circles. The short horizontal lines on the right of the cells are paths for possible synapses from a cell directly onto itself. The sub-windows show the *state* menus of the left cell (upper left) and synapse (upper right) and the *parameter* menus of the left cell (middle left) and synapse (middle right). The window at the bottom shows the membrane potential of the two cells in user selectable colours. Along the right side, the actions can be selected with radio buttons. At the bottom right corner there are facilities for panning and zooming. The oscillations were obtained by simplifying an existing model of 2 recurrent inhibitory cells with post-inhibitory rebound (Wang & Rinzel [1992]).

It would be highly desirable to have an integrated process that chooses the parameters, runs the simulations, and analyses the results. For example, the process could perform a minimisation of the difference between the results of the model and a typical experimental recording. The minimisation could be performed by any of the available numerical algorithms for optimisation. Since optimisation algorithms often use line searches in many directions, and the evaluation value in each direction is obtained by an independent simulation, such a process would make efficient use of a parallel computer. All processors would be heavily loaded and communicate only small amounts of information.

With more computer power one can investigate different configurations, for example using multi-compartmental models or adding more currents. Optimising several such models, one could determine which topology describes the experimental results best. This could then guide further experimental studies.

Biological systems prove to be remarkably fault tolerant. Further insights could be obtained by running simulations with random noise on the parameters. This would indicate whether the model is insensitive to the particular point in parameter space or it is more realistic. Again, such an investigation would take full benefit of a parallel computer.

The modeling core was compiled on the 64-processor transputer system in the department. Each slave processor runs its own copy of the modeling tool and is independent of all the other processors. A single master processor distributes the jobs as a database assignment file to the slaves. A slave starts by asking the master for a job, processes it, returns the results of the job, asks for a new job, and so on. The master has a set of jobs to be performed and distributes the next job to the first free slave. The processors were arranged in a pipeline with the master at one end. Organising the processors in a tree would reduce the communication overhead, but this proved unnecessary as the time spent running the simulations by far exceeds the

time spent on relaying the job assignments and results.

Each transputer proved to roughly equal a 50 MHz 486DX/2 PC or a Sun Sparcstation. In this application, the power of the parallel machine scales linearly with the number of transputers used. With a maximum of 96 transputers available, the machine achieves almost 100 times the performance of a PC.

2.8 Summary

The object-oriented design allows general base classes to be written. Specialist objects are derived from these base classes and inherit some or all of their properties. The amount of code that has to be written is therefore small, and the tool is easy to expand since new objects can inherit existing object's properties. This is important since currents and synapses show wide ranges of behaviours. For example, the development of the leech heart synapse showed that it can be hard to fit a current to a general framework. For example, one can envisage future classes for NMDA-receptors and calcium-activated potassium channels.

General simulation languages like Genesis or SPICE will infer performance reductions and will eventually reach a point where a mechanism becomes inconvenient to implement. However, this tool implements higher level operations in a more efficient language and achieves better performance and more flexibility.

Not only is the source code relatively small and easy to understand, it also makes it run faster. A speed-up of approximately 50–100 times has been accomplished over an existing implementation of the leech heart beat oscillator (Calabrese & De Schutter [1992]). The tool achieves this by being compact and using better integration methods. This is a significant improvement and will speed up the development of better models of neural systems. It will also make experimentalists more keen on combining experiments with modelling studies.

The database mechanism for accessing parameters, states, and functions has proven very versatile. It allows one to assign values to parameters and states at the command line, as well as using files of assignments. A configuration can be saved to a file as assignments and then recovered at some later time by applying the assignments in the file. On a parallel computer, a central job scheduler can define jobs as sets of assignments and then send those assignments to a free slave processor that performs the job and reports the result back to the scheduler. It also allows a set of jobs to be predefined and then executed in batch on a single processor while the user can do something else. Thus, the user does not have to interactively set the parameters at the start of each run.

The source code has proven to be easy to compile on a variety of machines and operating systems. Currently, the code works without modifications under Unix, MS-DOS, OS/2, and with 3L C++ on transputers.

Chapter 3

Leech heart modelling

A central pattern generator (CPG) is defined as a single anatomically defined network of neurons that generate timing and phasing cues for simple rhythmic movements. Inputs from higher command centres modulate the pattern of the CPG but do not participate in the rhythmic pattern themselves. The outputs of the CPG typically project to motorneurons that innervate the muscles.

The study of CPGs offer several advantages. CPGs produce motor patterns that are closely related to physical behaviour of the animal. It is therefore possible to directly link cellular mechanisms to behaviour. This compares favourably to the study of for example the vertebrate cortex, where there is little or no information about the behavioural relevance of the signals the cortex receives and produces.

The neurons of CPGs are usually large and easy to record from with microelectrodes. Furthermore, the neurons come in a relatively small number and they are possible to identify on the basis of size, position, and function.

As described earlier in Chapter 2, modelling enables one to study the experimentally inaccessible state variables of a system. This chapter will focus on investigating the leech heart timing oscillator with the modelling tool described in Chapter 2, Neurolab. This oscillator is a part of a CPG that controls the leech heart.

De Schutter, Angstadt & Calabrese [1993] have developed a model of a single oscillator involving 2 neurons and an ensemble of ionic currents. The model is implemented in *Nodus* and runs very slowly: simulating 20 seconds of data may take 5 hours. The model has therefore limited use in exploring parameters and incorporating more cells.

Their model was implemented in Neurolab, thus enabling a performance comparison to be made. It will also be shown that the model can account for other biological behaviours with a minor change in the parameters.

3.1 The leech heart

The leech is composed of 21 nearly identical segments, where each segment contains a ganglion with approximately 400 cells that mostly occur in bilateral pairs. The ganglia are connected through a nerve cord along the ventral side of the body.

Two bilateral heart tubes run the length of the leech. Each tube contains muscles that intrinsically contract in a periodic rhythm. A central pattern generator in the anterior 7 ganglia entrain the heart tubes to oscillate in phase with the CPG.

The timing circuit of the leech heart beat is located in the front 4 ganglia. The main components of the circuit are two oscillators, one in ganglion 3 and one in ganglion 4. Each oscillator consists of a pair of reciprocally inhibitory neurons. Figure 6 shows the activity of a pair of HN interneurons. The two oscillators are phase locked and have a typical cycle period of 10–15 s. The intersegmental coordination between the two oscillators is carried by neurons whose cell bodies are located in ganglia 1 and 2, but whose axons extend both to ganglion 3 and 4.

Distributed in ganglia 5–7 there is a switching circuit that gates the oscillating activity to the left and the right heart. The heart motorneurons, located throughout the segments of the animal, coordinate a wave running down the length of one heart

while the other heart contracts synchronously throughout the animal. Approximately every 50 cycles the switching circuit swap the behaviour of the two hearts. These behaviours are relevant for the pressure differences throughout the circulatory system.

3.2 Model of normal heart beat

The model comprises a bilateral pair of the heart interneurons (HN) in either ganglion 3 or 4. The HN(3) and HN(4) pairs are only different in that the HN(4) pair oscillate with a shorter period than the HN(3) pair.

Each cell incorporate a passive leakage current and 9 active currents, i.e. currents with separate state variables, including the synaptic current from the other cell (De Schutter, Angstadt & Calabrese [1993]). The synaptic release is governed by the presynaptic calcium concentration which represents a separate state variable. Na^+, K^+, and Ca^{2+} currents are each modelled by a fast and a slow current. Furthermore there is an A-current that typically implements a delay, and a h-current that provides an escape mechanism from prolonged inhibition by the opposite cell. There is a total of 30 state variables for the system of both cells and synapses. Almost all currents have been characterised by voltage-clamp experiments, except the fast Na^+ and K^+ currents.

Figure 7 shows the activity of the model.

3.3 Slow oscillations

Experiments in low external Na^+ have been performed to determine whether spiking is required for the system to oscillate (Arbas & Calabrese [1987b]). In order to get oscillations, the external Ca^{2+} concentration has to be raised. Figure 8 shows the slow membrane potential oscillations of the leech heart in 25% Na^+ and 1111% Ca^{2+}.

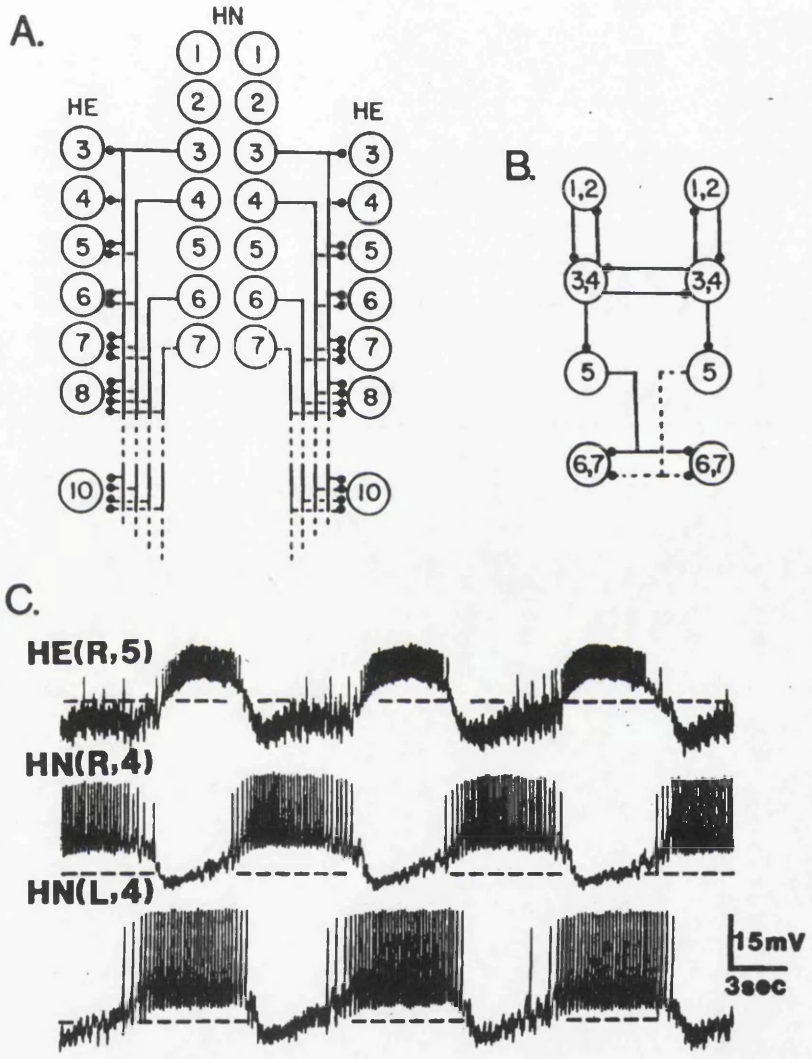

Figure 6: The organisation of the leech heart and recorded heart beat from contralateral heart interneurons. From Arbas & Calabrese [1987a].

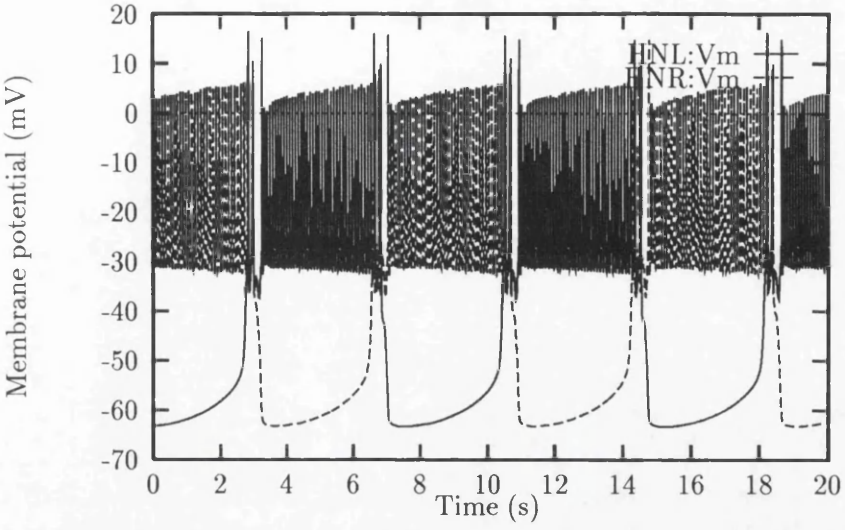

Figure 7: Simulated leech heart beat.

The thermal equilibrium potential, E, for an ion of charge Z with concentrations $[P]_o$ and $[P]_i$ on the outside and inside of the cell membrane, respectively, is given by the Nernst equation

$$E = \frac{kT}{Ze} \ln \frac{[P]_o}{[P]_i} \tag{25}$$

where $kT/e = 25.9$ mV at room temperature. The situation is analogous to that of an pn-junction where the equilibrium potential arises from drift and diffusion of charged particles.

Changing the ion concentration in the external solution to $[P]'_o = x[P]_o$ will add the following term to the reversal potential of Na^+ ($Z = 1$) and Ca^{2+} ($Z = 2$)

$$\Delta E = E' - E = \frac{25.9}{Z} \ln x \text{ mV}. \tag{26}$$

For 25% Na^+, $x = 0.25$, and $\Delta E_{Na} = -36$ mV, and for 1111% Ca^{2+}, $x = 11.11$ and $\Delta E_{Ca} = +31$ mV.

Figure 9 shows the result of changing the parameters of the model according to

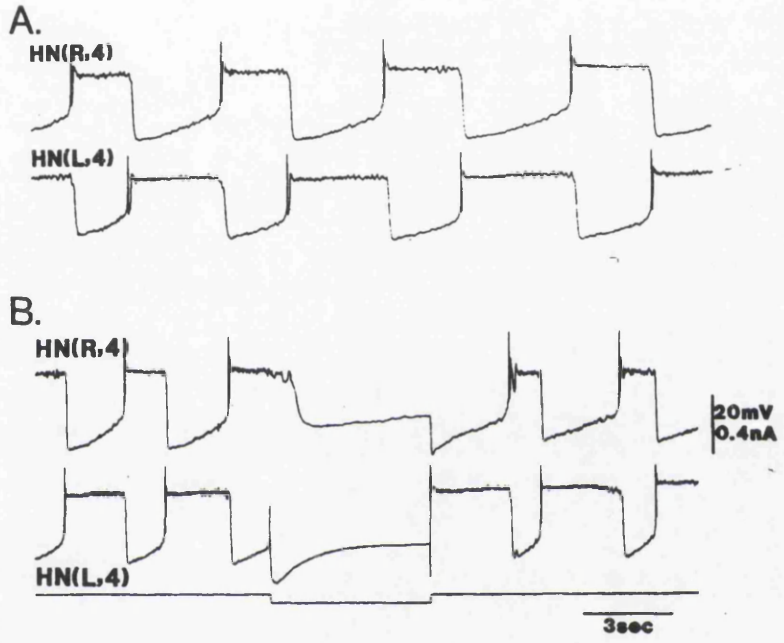

Figure 8: Slow oscillations of the leech heart in 25% Na^+ and 1111% Ca^{2+}. From Arbas & Calabrese [1987b].

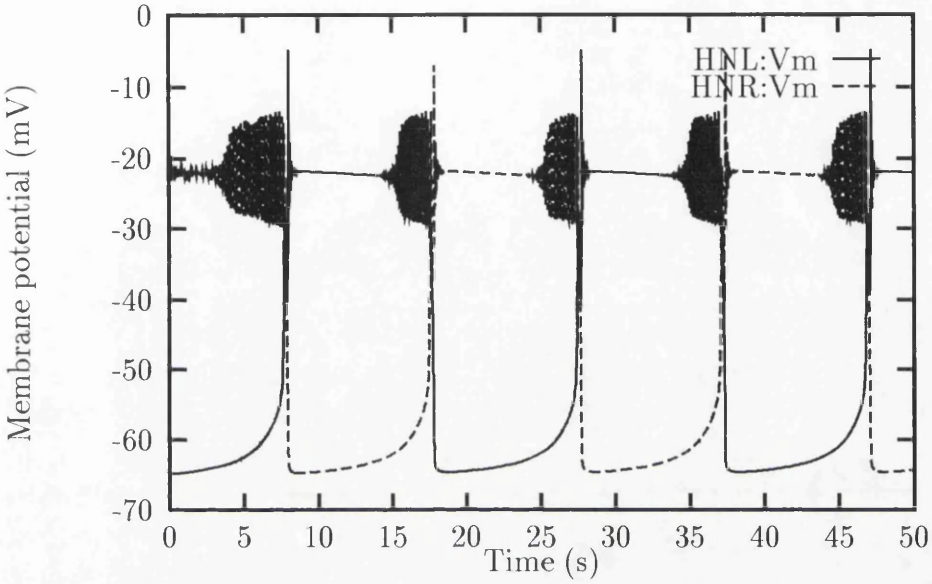

Figure 9: Unstable slow oscillations after modifying Na^+ and Ca^{2+} reversal potentials.

ΔE_{Na} and ΔE_{Ca} above. The model oscillates in principle like the real system, but it has large fast oscillations superimposed on the slow oscillations. The oscillations differ from actionpotentials in that the amplitude is modulated, but closer inspection reveals that the fast oscillations are shaped like actionpotentials. This instability is highly likely due to the fast Na^+ current since it is the fastest component within the model. In fact, the equations for the fast Na^+ current and the delayed rectifier K^+ current are taken from the literature since the dynamics of the actionpotentials are too fast to measure with single electrode voltage clamp. It is highly likely therefore, that these currents are somewhat misrepresented in the model. No actionpotentials were produced when the conductance of the fast Na^+ current was reduced by 33% (Figure 10). It was confirmed that this value of the conductance still induced actionpotentials in normal external saline, as well as the normal pattern of alternating bursting of the two neurons. The system is therefore relatively insensitive to the fast Na^+ conductance, and since it has not been possible to directly measure its value or the fast Na^+ dynamics in the HN cells, this is a valid modification of the model.

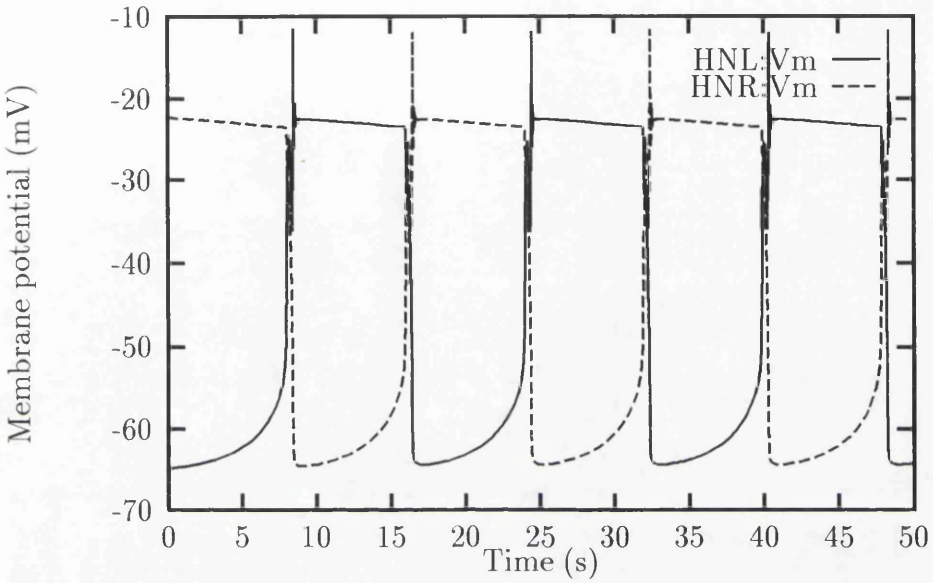

Figure 10: Stable slow oscillations after modifying Na^+ and Ca^{2+} reversal potentials and reducing the fast Na^+ conductance. The system has oscillated for 100 s since the change in parameters.

During the simulations, it was realised that the error tolerance for the numeric integration was critical for triggering the fast oscillations on top of the slow oscillations. In order to examine if the source of the fast oscillations was eradicated, the eigenvalues of the Jacobian of the system were calculated. Instabilities of the system is characterised by positive real parts of the (possibly complex) eigenvalues. The system has 30 state variables and will have 30 eigenvalues. In Figure 11, the maximal real part of the 30 eigenvalues was plotted as the system proceeded through time.

Theoretical studies of reciprocal inhibitory neurons have identified two mechanisms causing the transitions from the inhibited state to the active state (Wang & Rinzel [1992]). If the active cell somehow cannot maintain a prolonged depolarised phase, it may *release* the inactive cell from inhibition. On the contrary, the inhibited cell itself may *escape* from inhibition by not sustaining a prolonged hyperpolarised

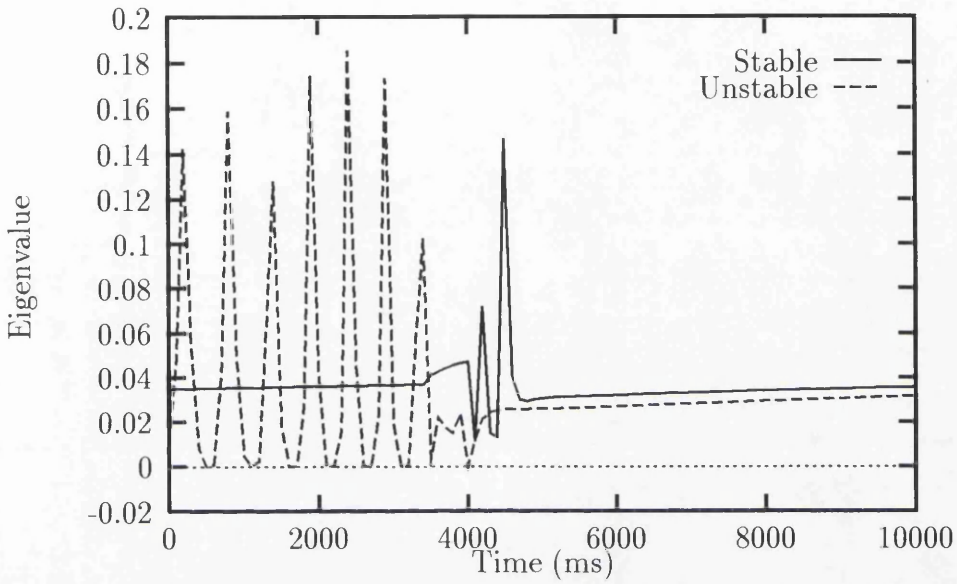

Figure 11: Maximal real part of the eigenvalues of the Jacobian.

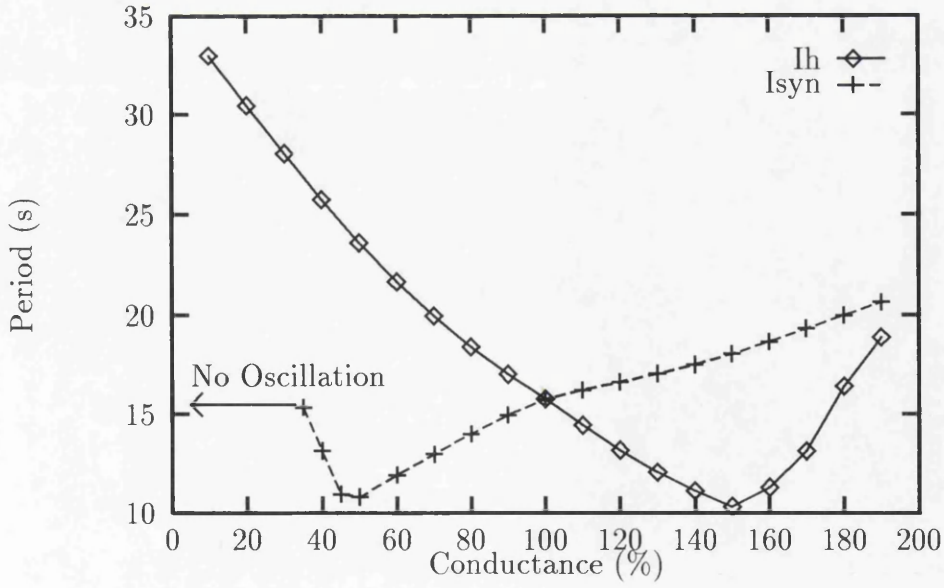

Figure 12: Period of slow oscillation as function of conductances of the synaptic current and the h-current. 100% corresponds to the default values.

phase. The spiking model as well as the real system have been found to operate predominantly in the escape mode. In Figure 12 the period of the slow oscillations are shown as function of the h-current conductance and the synaptic conductance. At the 100% point, the period is sensitive to changes in both the synaptic strength as well as the h-current conductance. The slope is steeper for the h-current, thus suggesting the period is more sensitive to a fractional change in the h-current conductance than the synaptic strength. Since the h-current implements the escape mechanism, the system is biased towards the escape mode also under slow oscillations. Note that the period decreases as the h-current conductance is increased, thus enabling a faster escape from inhibition. The synaptic current is the antagonist of the h-current and it is their relative strengths that control the period. The period is therefore increased as the synaptic strength is increased.

3.4 Performance comparisons

This implementation of the model in Neurolab performed roughly 50 times faster than in *Nodus*. The original model with spikes ran slower than real-time on a 50 MHz 80486 DX/2 computer under an Unix operating system. A typical simulation covering 20 s of simulated time took approximately 200 s of run time. The run time to heart time ratio then becomes roughly 10.

The slow oscillations ran faster than real-time on average. A 50 s simulation consumed 33.5 s of run time, thus achieving a run time to heart time ratio of 0.67. However, the model ran slightly slower than real-time at the transitions. This is due to the adaptive step size integration and the swapping between stiff and non-stiff methods for integration. At the transitions there are rapid changes in the state variables, and the integration has to be carried out with small time steps.

With increased throughput of simulations, one is able to develop models more

quickly since one is able to interactively adjust the parameters. Consequently, it will appeal more to the experimental biologist to use modelling as an integral part of the laboratory.

The issue of speed has also experimental relevance. Recently, dynamic voltage clamp was developed for assessing the effects of artificial introduced currents in neurons (Sharp et al. [1993a]). A single electrode alternates at a few kHz between sensing the membrane potential and injecting a current into the neuron. The current injected is controlled by a model running on a fast computer. The model is usually the standard Hodgkin-Huxley equations (1-4) with suitable dynamics, conductance, and reversal potential (Sharp et al. [1993b]).

In the leech heart, this technique has been used to implement artificial synapses between a pair of HN(3-4) cells (R.L. Calabrese, personal communication). The inhibitory synapses between a cell pair are blocked by bicuculline, and when injecting suitable currents into the two cells, effectively, the reciprocal inhibitory synapses are reintroduced. Now, one can manipulate parameters of the model synapses that are impossible to study in the real synapses. The existing synapse model is very simple, so eventually one would like to use the detailed model used in the simulations (De Schutter, Angstadt & Calabrese [1993]). However, this requires the simulation of 2 synapses to run in real-time. Since the full model includes 18 active currents and only 2 currents are required for the dynamic voltage-clamp experiments, and the full model runs 10 times slower than real-time, simulating 2 currents should achieve close to real-time performance. However, the Ca^{2+} currents will also have to be simulated in order to estimate the Ca^{2+} influxes at the presynaptic terminals. On the other hand, the expensive simulation of the action potentials is avoided. It is therefore possible to implement detailed artificial synapses between two real decoupled HN(3-4) interneurons.

3.5 Further work

Further work will address 3 inadequacies of the model.

When comparing the behaviour of the model with that of the real heart, it can be seen from Figures 6 and 7 that in the model, the transitions between the spiking and inhibited phases are too rapid. The real system has a smoother transition with some overlapping of spikes from the two cells.

The real system has both inhibitory graded synaptic transmission and inhibitory spike-induced postsynaptic potentials. However, the model only includes the graded synaptic transmission. The model of the presynaptic calcium dynamics operates as a low-pass filter and produces no postsynaptic potentials in the other neuron. Secondly, the model synapse will therefore be improved to include the postsynaptic spike-mediated potentials.

Thirdly, voltage clamp data of a single neuron indicate that a multicompartmental model of the HN cells is necessary.

Further developments will connect the HN(3) pair with the HN(4) pair via the HN(1-2) neurons and aim to establish phase-locked bursting between the HN(3) pair and the HN(4) pair. When more experimental data prevails, the switching between the synchronous activity and the wave running the length of the heart tube will also be included in the model

3.6 Summary

The standard model of the leech heart beat was tested with the original *Nodus* output and found to agree. The model ran approximately 50 times faster using Neurolab than *Nodus* on an equivalent computer. Moreover, up to 96 simulations can be run in parallel on the transputer-based computer in the department. Totally, this gives

a throughput that is several thousands times that earlier achieved. This becomes significant when searching the parameter space for optimal behaviour of the model, and it makes studies of more complex models of the heart feasible.

The significant increase in processing speed necessitates the possibility of setting up batch sequences of simulations. With the existing programs one will have to sit at the computer and adjust the parameters before every simulation. This becomes tedious when each simulation runs quickly. The database mechanism makes it easy to define parameter values on the command line, and sequences of runs can be predefined and processed in batch mode. The mechanism also prepares Neurolab to allow the parameters to be chosen by a program that optimises the output of the model to experimental data.

Slow oscillations, previously found experimentally in low Na^+ and high Ca^{2+} concentrations, were modelled by adjusting the reversal potentials for Na^+ and Ca^{2+} accordingly. The fast Na^+ conductance had to be reduced by 33% to avoid spiking.

The flexibility of the program and the advantages of being in control of the source code were demonstrated by adding the extraction of the Jacobian matrix to the program. This allowed the stability of the slow oscillations to be studied.

The modelling described in this chapter took advantage of being able to control parameters of the system that in an experimental situation are inaccessible. It also proved invaluable to be able to study experimentally unobservable state variables during the development of the model. Thus, with a model one achieves greater insight into the temporal operation of the system.

Chapter 4

Principal component analysis

This chapter introduces the mathematical detail of principal component analysis. The theory will be applied in Chapter 5 to quantify a trace of activity over an interval of time. The following section does not intend to be complete but only to generate a feeling for what the transform does. Later sections will show that the transform can be computed by a simple neural network. The nature of these networks and the advantage of such processing of sensory input suggest that the transform might be performed in real neuronal systems.

4.1 The discrete Karhunen-Loève transform

Let $\mathbf{s}$ be an n-dimensional column vector. In the context of this thesis, the vector will be the samples over some interval of a signal. The vector $\mathbf{s}$ can be expressed as a linear expansion of the form

$$\mathbf{s} = \sum_{i=1}^{n} c_i \Phi_i = \Phi \mathbf{c} \tag{27}$$

where

$$\Phi = [\Phi_1 | \Phi_2 | \ldots | \Phi_n] . \tag{28}$$

The matrix Φ is made up of n linearly independent column vectors, Φ_i, that span the n-dimensional space. If the Φ_i are orthonormal, i.e.

$$\Phi_i^T \Phi_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}, \tag{29}$$

the c_i are given by

$$c_i = \Phi_i^T \mathbf{s}. \tag{30}$$

Such an expansion is analogous to a Fourier series. The Fourier components are mutually orthogonal and are added up, weighted by suitable coefficients, to reproduce the original function. The original function is uniquely defined by those coefficients. Here, the coefficients c_i will uniquely define the vectors $\mathbf{s}$, but the basis vectors will not be pre-defined sines and cosines. The basis vectors, Φ_i, will be extracted from a set of input vectors $\{\mathbf{s}^k\}$.

Assume only $p(< n)$ components of $\mathbf{c}$ are used in the expansion, and the remaining coefficients, b_i, are preselected. The original vector can be estimated as

$$\hat{\mathbf{s}} = \sum_{i=1}^{p} c_i \Phi_i + \sum_{i=p+1}^{n} b_i \Phi_i. \tag{31}$$

The error in this estimate is

$$\Delta\mathbf{s} = \mathbf{s} - \hat{\mathbf{s}} = \sum_{i=1}^{n} c_i \Phi_i - \sum_{i=1}^{p} c_i \Phi_i - \sum_{i=p+1}^{n} b_i \Phi_i = \sum_{i=p+1}^{n} (c_i - b_i)\Phi_i. \tag{32}$$

Use the mean square magnitude of $\Delta\mathbf{s}$ to measure the effectiveness of the estimate

$$\langle |\Delta\mathbf{s}|^2 \rangle = \langle \sum_{i=p+1}^{n} \sum_{j=p+1}^{n} (c_i - b_i)(c_j - b_j)\Phi_i^T \Phi_j \rangle = \sum_{i=p+1}^{n} \langle (c_i - b_i)^2 \rangle. \tag{33}$$

The mean over the set of vectors is denoted by $\langle \cdot \rangle$. The basis vectors Φ_i and the coefficients b_i are the parameters of the error measure and will be chosen so to minimise the error. The optimal choice of b_i is $b_i = \langle c_i \rangle$ and using (30) the error reduces to

$$\langle |\Delta\mathbf{s}|^2 \rangle = \sum_{i=p+1}^{n} \Phi_i^T \mathbf{C} \Phi_i, \tag{34}$$

where the covariance matrix, $\mathbf{C}$, is given by

$$\mathbf{C} = \langle (\mathbf{s} - \langle \mathbf{s} \rangle)^T (\mathbf{s} - \langle \mathbf{s} \rangle) \rangle, \tag{35}$$

or equivalently

$$C_{ij} = \langle (s_i^k - \langle s_i \rangle)(s_j^k - \langle s_j \rangle) \rangle, \quad i, j \in [1, n]. \tag{36}$$

It can be shown (Fukunaga [1972]) that the optimal choice of basis vectors in the mean square error sense are the eigenvectors of the covariance matrix

$$\mathbf{C}\Phi_i = \lambda_i \Phi_i \tag{37}$$

where λ_i is the eigenvalue.

The covariance matrix is symmetric which leads to real eigenvalues and positive semi-definite which further restricts the eigenvalues to be non-negative.

The error becomes

$$\langle |\Delta \mathbf{s}|^2 \rangle = \sum_{i=p+1}^{n} \lambda_i, \tag{38}$$

so an eigenvector Φ_i will contribute with its eigenvalue, λ_i, towards the square error, if it is left out of the expansion (27). Therefore, if the eigenvectors are ordered such that

$$\Phi = [\Phi_1 | \Phi_2 | \ldots | \Phi_n], \quad \lambda_1 > \lambda_2 > \cdots > \lambda_n, \tag{39}$$

an expansion using only the first p basis vectors will contain a fraction

$$\frac{\sum_{i=1}^{p} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \tag{40}$$

of the total variance of the set of vectors $\mathbf{s}^k$.

Here, the basis vectors Φ_i will be referred to as principal components (PCs) and the coefficients c_i^k as the principal component coefficients (PCCs). In the literature, the coefficients are sometimes confusingly referred to as the principal components.

Typically, one will set the desired fractional error and use the minimal number of basis vectors satisfying the error criterion. If it suffices to consider 95%, say, of the

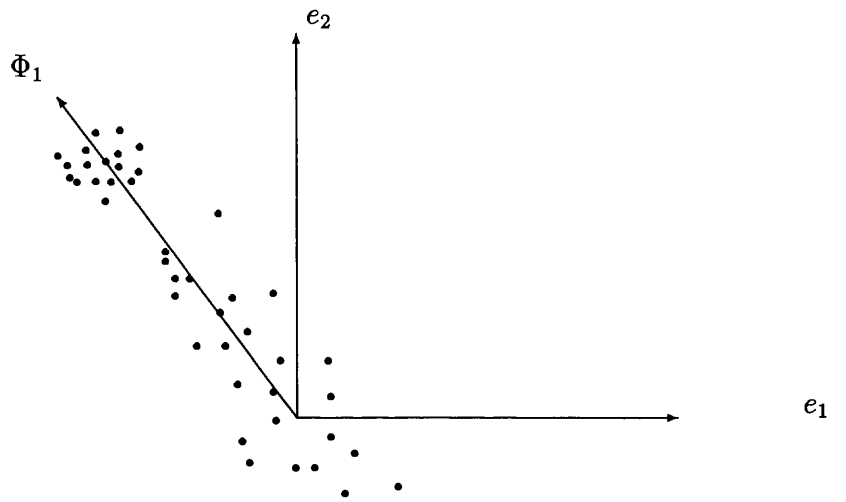

Figure 13: Principal component of a simple distribution. The basis vectors of the input vectors are e_1 and e_2, and Φ_1 is the first principal component. The second principal component is perpendicular to the first by the orthogonality requirement.

total variance of the input vector set, it may be that only 3, say, principal components are required in the expansion. The number of dimensions of the input vectors has thus been significantly reduced. Figure 13 illustrates a simple 2-dimensional distribution that can be approximated in one dimension.

The principal component coefficients are statistically uncorrelated, i.e. the covariance matrix of the coefficients c_i is diagonal. Of all linear transforms, the Karhunen-Loève transform is the only one that guarantees uncorrelated coefficients. In particular, the Fourier transform will produce correlated coefficients in the frequency domain.

4.2 Artificial neural networks

In the field of artificial neural networks, several network topologies and learning algorithms have been devised to perform principal component analysis. The learning algorithms are developed from the concept introduced by Hebb [1949] that states

> When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

An algorithm based on this principle is local. The information required to adjust the state of the network is supplied locally—there is no global error or reinforcement signal. The connection between two units is modified according to the states of the two units and the state of the connection only. Many other learning algorithms also require to know the states of other connections and other cells in order to modify a particular connection. This is unlikely to happen in nature and will complicate any implementation, irrespective if it is with biological neurons, in VLSI hardware, or on parallel computers.

Hebbian learning algorithms are also unsupervised. Many of the artificial neural network learning algorithms require an external teacher that supplies either the correct answer or a reinforcement signal that provides feedback to the network how well the current parameters works. The learning algorithms described here do not require any such external signal, which is likely to be the case in most biological neural systems.

It can therefore be argued that these models are biologically plausible and that principal component analysis is performed in neural systems. This would be particularly relevant to sensory systems, where the input is typically high-dimensional and contains large amounts of redundancy. Further processing of the sensory input would benefit substantially from reducing the dimensions to a more compact representation that is tuned in to the typical input experienced by the sensory system.

Hancock, Baddeley & Smith [1992] demonstrated that calculating the principal components with an artificial neural network is more efficient than calculating the eigenvectors with standard methods. The input vectors were 64 by 64 pixel images

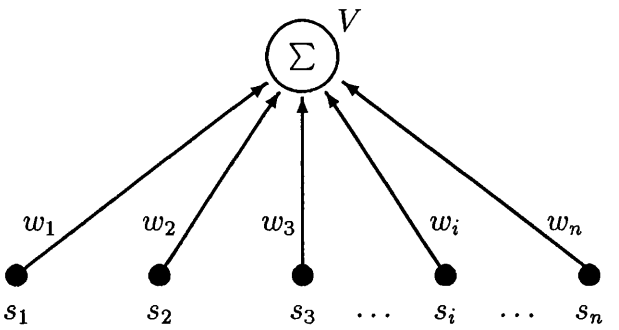

Figure 14: Architecture of Oja's network.

totalling 4096 pixels per image. Thus the input vectors have 4096 components and the covariance matrix has 4096 by 4096 elements. It is an unreasonable task to calculate the eigenvectors of such a matrix by normal numerical methods. A 16 unit neural network finds an approximation to the first 16 principal components within a few hours on a normal workstation.

4.2.1 Oja's rule

Consider a single linear McCulloch-Pitts unit with output V, weights $\mathbf{w}$, and input vector $\mathbf{s}$, such that

$$V = \sum_{i=1}^{n} w_i s_i = \mathbf{w}^T \mathbf{s} = \mathbf{s}^T \mathbf{w}. \tag{41}$$

The architecture is illustrated in Figure 14.

Plain Hebbian learning updates the weights according to

$$\Delta w_i = \eta V s_i \tag{42}$$

where η is the learning rate and the weight change is applied after each presentation of an input vector. This rule strengthens the output for frequent input vectors, and the output becomes a measure of familiarity. However, plain Hebbian learning will never cause the weights to converge—they will grow indefinitely.

However, suppose there were an equilibrium point for $\mathbf{w}$. The average of the ith weight change would then be

$$0 = \langle \Delta w_i \rangle = \eta \langle V s_i \rangle = \eta \langle \sum_j w_j s_j s_i \rangle = \sum_j C_{ij} w_j = \mathbf{Cw} \tag{43}$$

where the correlation matrix $\mathbf{C}$ is defined by

$$C_{ij} = \langle s_i s_j \rangle \tag{44}$$

or

$$\mathbf{C} = \langle \mathbf{s}\mathbf{s}^T \rangle. \tag{45}$$

The correlation matrix differs from the covariance matrix that was described earlier, where the vectors were translated by the means $\langle s_i \rangle$. The two become equivalent if the input vectors have zero means.

$\mathbf{C}$ is symmetric and positive semi-definite and hence have real non-negative eigenvalues. At the potential equilibrium point (43), $\mathbf{w}$ is an eigenvector of $\mathbf{C}$ with eigenvalue 0. The positive eigenvalues cause the equilibrium point to be unstable, and the weights would grow exponentially in the direction of the eigenvector of $\mathbf{C}$ with the largest eigenvalue. The weight vector would therefore approximate the first principal component with an increasingly large norm.

The weight vector can be normalised after each update, so that $|\mathbf{w}| = 1$. However, Oja [1982] modified the plain Hebbian rule by adding a weight decay proportional to V^2

$$\Delta w_i = \eta V (s_i - V w_i). \tag{46}$$

It can be shown that Oja's rule converges to a weight vector of unit length, i.e. $|\mathbf{w}| = 1$, with $\mathbf{w}$ being an eigenvector corresponding to the maximum eigenvalue of $\mathbf{C}$, and $\mathbf{w}$ being such that $\langle V^2 \rangle$ is maximised (Hertz, Krogh & Palmer [1991]; Oja & Karhunen [1985]). For zero mean input vectors, the latter results correspond to maximising the variance of the output and, equivalently, finding the first principal component.

Alternative rules to (46) have been used to keep the weight vector bounded. Linsker [1986] and Linsker [1988] simply clipped the weights so that $w_- \leq w_i \leq w_+$, whereas other rules that are more mathematically tractable tend to be non-local (Pearlmutter & Hinton [1986]; Yuille, Kammen & Cohen [1989]).

4.2.2 Principal components

Oja's rule extracts the first principal component coefficient. The theory has been developed to obtain p principal components by using one-layer p-unit networks, where the output of unit i, V_i, is linear

$$V_i = \sum_j w_{ij} s_j = \mathbf{w}_i^T \mathbf{s}. \tag{47}$$

Note that each unit has its own weight vector with the same dimension as the input vectors, n.

Sanger [1989] introduced the learning rule

$$\Delta w_{ij} = \eta V_i \left(s_j - \sum_{k=1}^{i} V_k w_{kj} \right) \tag{48}$$

and Oja [1989] developed his p-unit rule

$$\Delta w_{ij} = \eta V_i \left(s_j - \sum_{k=1}^{n} V_k w_{kj} \right). \tag{49}$$

The only difference is in the upper limit of the summation.

For Sanger's rule, the p weight vectors develop exactly to the first p principal components of the Karhunen-Loève transform, and the output values give the principal component coefficients of the transform for a given input vector. Oja's rule spans the same subspace but does not obtain the exact principal components. They both reduce to Oja's original rule for $p = 1$.

In practise, Sanger's rule is advantageous since it provides the coefficients ordered by variance, whereas Oja's rule is more likely to appear in biological systems. However, both rules are non-local. A change in weight w_{ij} require information beyond

what is available at units i and j. Sanger has suggested a reformulation of his learning rule so that it becomes local.

There are several other architectures that perform principal component analysis. Some networks use lateral connections, u_{ij}, from V_j to V_i that obey anti-Hebbian learning, where

$$\Delta u_{ij} = -\gamma V_i V_j. \tag{50}$$

This rule decorrelates the outputs of the network. Like with Sanger's rule, the network learns to extract the first p principal components (Földiák [1989]; Rubner & Schulten [1990]; Rubner & Tavan [1989]).

4.3 Quantification of temporal activity

In the following chapter, principal component analysis will be used to quantify a trace of neuronal activity. The particular activities to be studied are the spike density (or spiking frequency) and the membrane potential after removing the spikes. The activity over a period of time will be sampled and form a set of input vectors. The analysis will show in what way and how much the individual activity traces differ from the average of all the traces. Therefore, it allows one to quantify the behaviour of a neuron over a period of time. During this time, the feedback loops within the neuron and through the neuronal network will affect the output of the neuron. Thus, the resulting measure of the neuronal output will indirectly include the states of "hidden" compartments away from the cell soma and other inaccessible state variables, like ion concentrations. These state variables are all inaccessible through a microelectrode recording from the cell soma when only the instantaneous values of the membrane potential are analysed. Although the technique offer no promise for tracing these hidden state variables, it will enable us to include them in quantifying the outputs of a neuron.

4.4 Summary

The Karhunen-Loève transform performs a dimensionality reduction of the input vector space. Each input vector is decomposed into features that are common among the set of input vectors, i.e. principal components. The principal components span the same space as the input vectors, but when there is much redundancy in the input vectors, only a few features (principal components) are necessary to approximate the input vectors. The principal components are ordered by the variance they represent over the set of input vectors.

Simple artificial neural networks can perform principal component extraction and transformation. They are particularly useful in high-dimensional input spaces, and tend to be biologically plausible. This form of signal processing may be performed in biological sensory systems.

Chapter 5

Analysing oscillations in neuronal systems

The Karhunen-Loève transform described in Chapter 4 provides a measure of a signal over some period of time. The time course of the signal will incorporate the effects of the feedback loops in the system. The output of the Karhunen-Loève transform should therefore shed light on features that are not apparent in the instantaneous values of the signal.

The technique can be applied to any neuronal system that generates repeated activity patterns. In previous studies, these patterns were triggered by visual stimuli (Richmond & Optican [1987]). Here, the investigated patterns are bursts of spikes from the crayfish swimmeret and the leech swimming that are generated periodically.

5.1 Crayfish swimmeret

The crustacean swimmeret is located on the ventral side of the abdominal region. Each abdominal ganglion control a pair of small limbs that are controlled by alternative bursts of spikes to the agonist and antagonist muscles generating the power-stroke

and the return-stroke. The period of swimming ranges from 1–2 seconds and can be controlled by applying drugs.

The ganglia are sequentially connected by the ventral cord. The sequence of ganglia maintain mutually fixed phase lags over the range of swim frequencies, where the phase lag per ganglion is approximately 14 degrees. The inter-ganglia connections are therefore no simple time delays but incorporate complex phase locking mechanisms.

Each ganglion contains thousands of neurons and are therefore hard to study in great detail. However, experiments have shown that there are both spike-mediated synaptic transmission as well as graded synaptic transmission, where the post-synaptic potential depends directly on the membrane potential of the pre-synaptic neuron.

The following study analyses the bursts of spikes from a central interneuron and a motor neuron using the Karhunen-Loève transform. The method quantifies the progress of the bursts through time, and it can compare the signal mediated by the spikes with that of the base membrane potential, on which the spikes are imposed.

5.1.1 Signal processing

The analysis was performed on intracellular recordings from two unidentified cells from a single ganglion in the crayfish swimmeret, stored on tape in analogue form. Each of the two channels was sampled at 1000 Hz. Given a typical action potential width of 5 ms, this gives 5 samples per spike, which is sufficient to recognise the spikes from the background noise and post-synaptic potentials. Figure 15 shows a sample record.

The amplitude of the slow membrane oscillation was of the same order as the spikes (Figure 15). A simple threshold algorithm would therefore not suffice to extract the spikes. The membrane base potential was found by passing the recorded signal through a simple low-pass filter. Let us denote the sequence of membrane potential

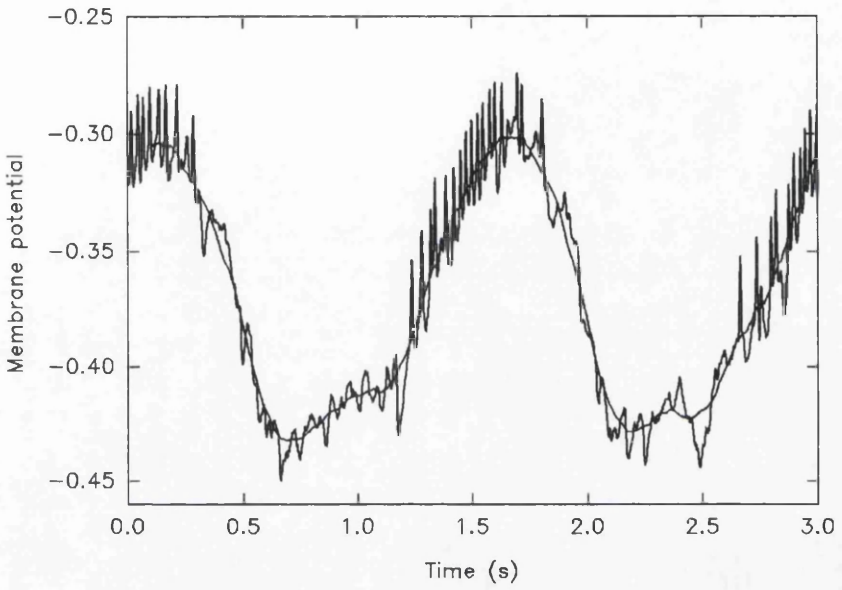

Figure 15: Intracellular recording of membrane potential (with spikes) and the slow membrane potential obtained by a low-pass filter (without spikes). (The potential has arbitrary origin and scale.)

samples by v_i and the sample interval by Δ. The filtered output at sample i was the average of the $m = 2k + 1$ samples $\{v_{i-k}, \ldots, v_i, \ldots, v_{i+k}\}$. This low-pass filter was chosen for its simplicity and has the transfer function

$$H(f) = \frac{\sin m\pi\Delta f}{m \sin \pi\Delta f} \tag{51}$$

where f is the frequency of the Fourier components in the signal. This filter has the advantage of having no time delay. The transfer function is 1 for small f, and falls to 0 at $f_c = (m\Delta)^{-1}$, and then oscillates with relatively small amplitude at high frequencies. Note that setting the cut-off frequency f_c defines m. Figure 15 shows that the filtered signal with $f_c = 5$ Hz succeeds in representing the slowly oscillating membrane potential.

A simple threshold algorithm was used to extract the spikes from the recorded potential minus the slowly oscillating membrane potential. Some ambiguous spikes were situated between the main spikes. These artifacts had the same shape as the

main spikes but had smaller amplitude and sometimes merged. The threshold was chosen as to extract the main spikes only, but to include all the main spikes a few secondary spikes had also to be included (the ratio of main to secondary spikes was higher than 20:1).

Spike density

The spikes are a series of discrete events and can be mapped into a continuous spike density function by superimposing a Gaussian of unit area, $g(t)$, on each of the N_s spikes within a burst (MacPherson & Aldridge [1979]; Richmond et al. [1987]; Silverman [1986]). The spike density function $s(t)$ representing the set of spikes at $\{t_i\}$, is estimated by

$$g(t) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{t^2}{2\sigma^2}\right) \tag{52}$$

$$s(t) = \sum_{i=1}^{N_s} g(t - t_i) \tag{53}$$

where σ is the width of the Gaussian and $s(t)$ is normalised to integrate to N_s, the number of spikes in the burst. The spike density function can be interpreted as the instantaneous spike frequency throughout the bursts. An example is illustrated in Figure 16.

The width of the Gaussian could be varied in the range 35–200 ms without significant effects on the transform. This range of values gave smooth spike densities for the bursts. Richmond et al. [1987] used a shorter value of the width (10 ms) in the visual cortex of monkeys where the spike frequencies may reach 300 Hz. Those neurons process more information and hence require a higher bandwidth than the crayfish swimmeret neurons. The value to be chosen in a particular application has to compromise between giving individual spikes too much influence (too small width) and smoothing out important characteristics (too large width). A method for choosing the value is given below.

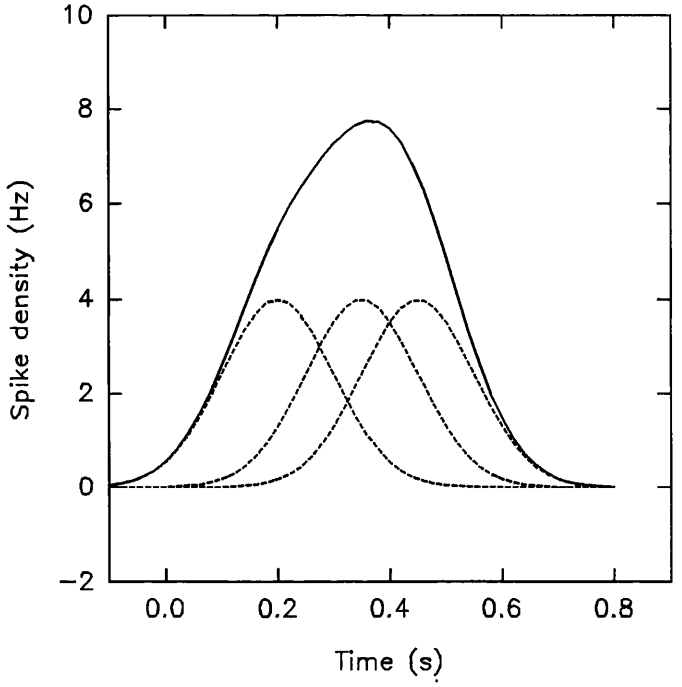

Figure 16: Example of spike density function composed of spikes at $s_1 = 0.2$ s, $s_2 = 0.35$ s, and $s_3 = 0.45$ s ($\sigma = 100$ ms). The total spike density is shown in the full line and the contribution from individual spikes in dashed lines.

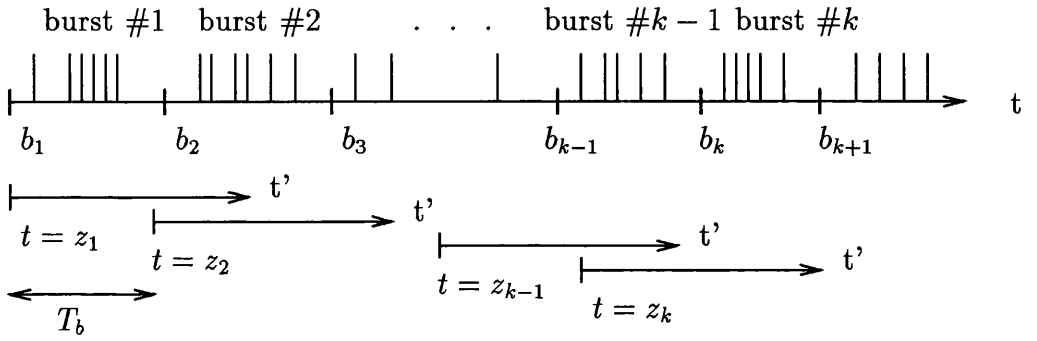

Figure 17: The spike train and the burst separators $\{b_k\}$ at the top, and the periodic burst windows shown as arrows below the time axis. The burst separators are irregular whilst the windows are periodic. The windows start at $\{z_k\}$ every T_b seconds where T_b is the average burst period, and the length of the window (arrow) is T_w. Note that windows will overlap when $T_w > T_b$, but they only include spikes from the corresponding burst. The overlap is caused by large burst displacements (see text). The first two bursts are illustrated at the start of the windows and the last two bursts at the end of the windows. This corresponds to minimum and maximum burst displacements, respectively. Bursts succeeding the last illustrated will move back towards the start of the windows.

The spikes from the oscillating swimmeret were associated with clearly separated bursts. Burst separators, as shown in Figure 17, were obtained by setting a low threshold on the spike density of the spike train. The burst separator events were calculated as the mid-points between the point when the spike density fell below the threshold and the successive point when the spike density rose above the threshold. For this purpose, σ was relatively large so to smooth the bursts, and the threshold was a few of $1/\sqrt{2\pi}\sigma$.

Initially, it was assumed that the bursts appeared periodically. The periodic events closest to the burst separators were determined by linear regression. Denoting the periodic events by $z_k = z_0 + kT_b$, where k is the burst index, the average period T_b and the offset z_0 are the results of the linear least squares fit between the burst separators and burst indices. The origin of time for burst number k was taken to be

z_k, so a spike occurring at time t and belonging to burst k is translated to the local time $t' = t - z_k$ within that burst (see Figure 17). Fitting the median or mean spike in each burst instead of the burst separators would give similar results.

The spike density function s(t) for the spikes within burst number k was sampled at N_v points over a window of duration T_w to form an N_v-dimensional column vector $\mathbf{s}^k$, where

$$s_i^k = s(z_k + \frac{i}{N_v} T_w), \quad i \in [0, N_v - 1]. \tag{54}$$

The index i therefore represents the local time t' within the burst window. For N_b bursts, the set of N_b N_v-element spike density vectors, $\{\mathbf{s}^k\}$, was input to the Karhunen-Loève transform.

As seen in Chapter 4, the Karhunen-Loève transform involves taking the average of the spike density vectors. The offset z_0 was adjusted so that the average vector appeared in the center of the window, and the window width T_w was selected so that it contained all, but no more than, the non-zero parts of the average and the principal components. These are efficiency considerations as the cost of computing the Karhunen-Loève transform increases with N_v^2. Note that when the burst displacements (see later) are larger than T_b, the window length T_w will have to be increased so that the tail of a window will overlap the head of the succeeding window, as illustrated in Figure 17.

Choosing a value for σ

The parameter σ determines the amplitude and width of each spike's contribution to the total spike density. A lower limit on the value for σ can be determined simply by considering the smoothness of the spike density function. There is no reason to expect a spike at a certain instant as the technique is model free.

The value of σ can be estimated by considering the width of the bursts and the number of spikes in the bursts. The worst case for forming single spike peaks is when

the spikes are evenly distributed throughout the burst. Then, the corresponding spike density function would be approximately constant. A lower bound on σ would therefore be the effective width of the burst divided by the number of spikes in the burst. Note that the burst period here is 1.5 s, but that the spikes occur over a shorter period, approximately 1.2 s. Given approximately 17 spikes in a burst, this imposes a lower bound on σ of 70 ms. Increasing the value of σ beyond this will cause information to be thrown away, since the features of the spike train are smoothed. Figure 18 shows some spike density functions for various values of σ. In the following studies, a value of $\sigma = 100$ ms was used.

Slow membrane potential

The low-pass filtered membrane potential was split into bursts covering the same interval as the spike density. The membrane potential within each burst was sampled to form membrane potential vectors.

5.1.2 Implementing the Karhunen-Loève transform

The eigenmatrix Φ was extracted using the Householder routine as presented in Wilkinson & Reinsch [1971]. The algorithm was implemented in the public domain matrix package *newmat07* written in C++ that is available on the Internet news group *comp.sources.misc*.

5.1.3 Results

Figure 19 shows the Karhunen-Loève transform of 100 successive bursts from a swimmeret motor neuron. The wide spread in the burst positions within the window is the most significant variance over the data set and is likely to appear in the first principal component. Indeed, a positive coefficient of the first principal component

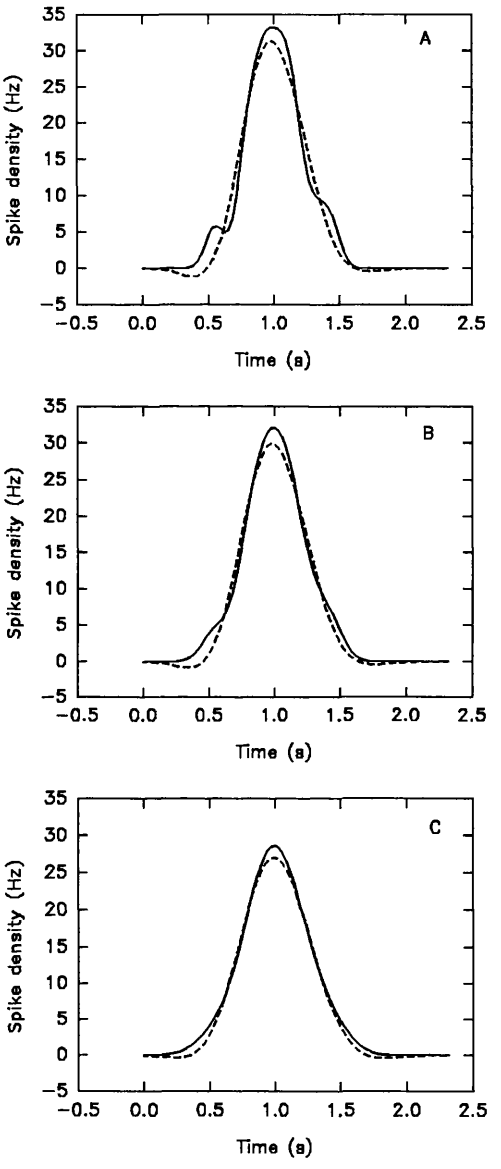

Figure 18: A random spike density vector (solid line) and its Karhunen-Loève approximation (dotted line) for 3 principal components and different σ. A: $\sigma = 70$ ms. B: $\sigma = 100$ ms. C: $\sigma = 150$ ms. The width of the windows is $T_w = 2.5$ s, the principal components are extracted from $N_b = 100$ bursts, the spike density of a burst is sampled at $N_v = 80$ points, and $p = 3$ principal components are used to approximate each spike density vector. The set of 3 components contained from 94.4% (A) to 96.8% (C) of the variance of the 100 bursts.

(PC1) added to the average spike density shifts the burst spike density forward in time, and a negative coefficient shifts the burst spike density backward. This suggests that the first principal component represents displacements along the time axis.

Figure 20 verifies that the first principal component coefficient (PCC1) can be correlated with the burst displacement. The arithmetic means of the spike times of the bursts were used to establish the position of the bursts in the window. The correlation is good over a large range apart from the extremities, where the displacement variation exceeds the capability of a single linear term. Higher order terms, or, as here, further principal components, have to be included to express the large variation in the non-linear (Gaussian) shape.

For long sequences of bursts, any sloping trend in the latency, defined as the mean spike minus the start of the window, would suggest that the estimated period of bursting, T_b, is wrong. A positive divergence would mean T_b is too small and a negative divergence would mean T_b is too large. Here the latency initially grows and then decreases back to the starting point. This suggests T_b is a good estimate of the mean burst period.

The other principal components were affected by the wide spread in the burst latency. The windows were therefore redefined to be centered at the mean spike. The window width could be decreased to focus directly on each burst, and not in a neighbourhood surrounding the burst to cover the displacements. The transform could therefore fully quantify the shape of the bursts, unaffected by the displacements. The shape of original principal components can be recognised in the new components, and the interpretation of the new components applies to the old ones, although the new components bring out the features of the burst more clearly.

The average spike density vector in Figure 21 shows that typical burst spike densities (or frequencies) follow a Gaussian-shaped time-course. If the spike count in a burst increased above the average, the area under the burst spike density curve would

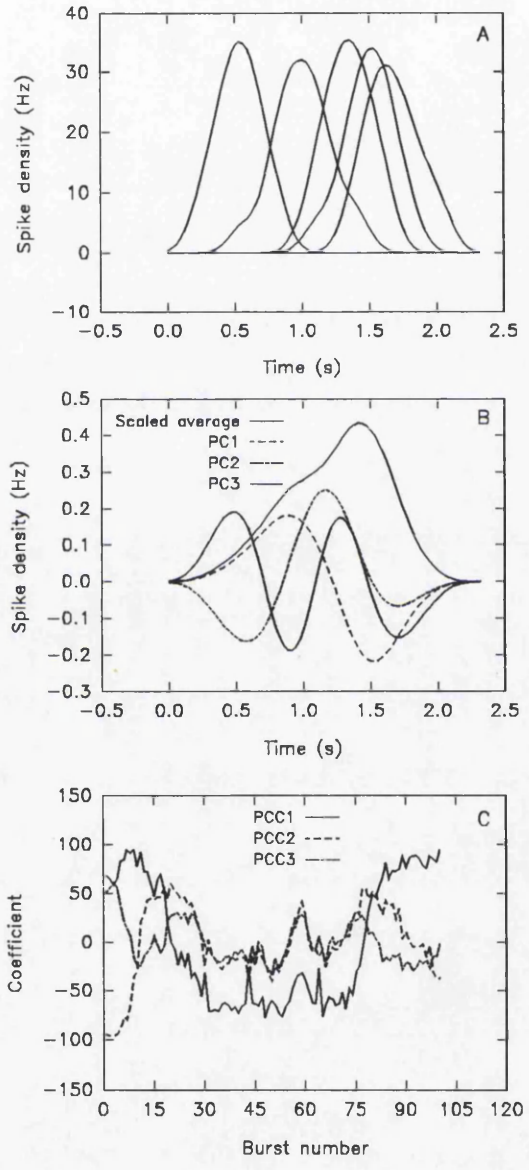

Figure 19: Karhunen-Loève transform of swimmeret motor neuron. A: Some sample spike density vectors. B: The first 3 principal components and the average scaled down for reference. C: The principal component coefficients. (T_b = 1.49 s, T_w = 2.35 s, σ = 100 ms, N_b = 100, N_v = 64). (PC: principal component, PCC: principal component coefficient)

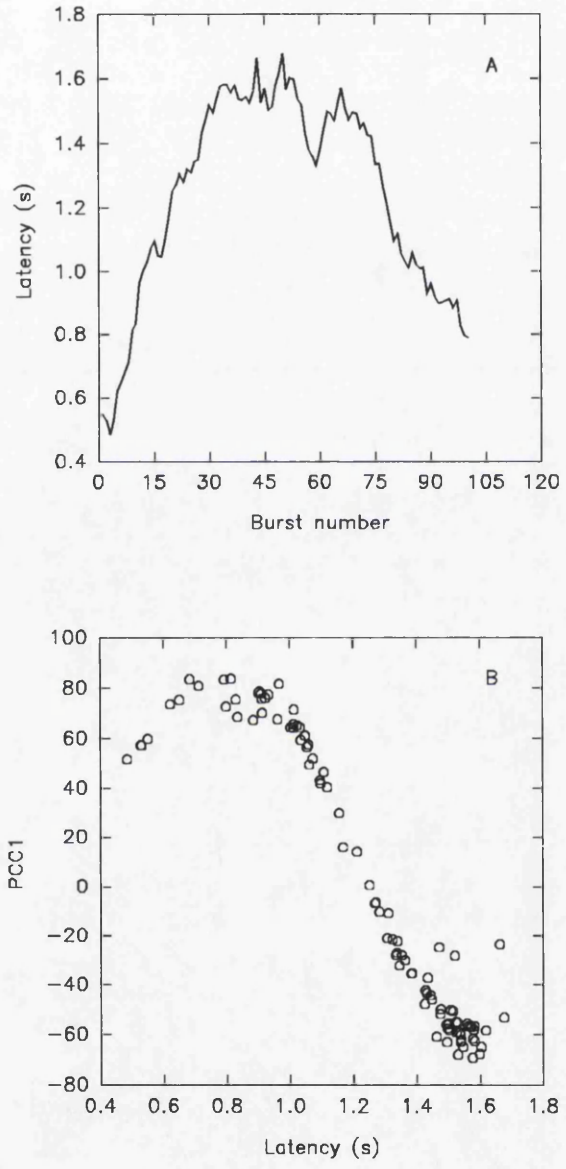

Figure 20: A: The mean spike latency in window for 100 successive bursts. B: Correlation of first principal component coefficient with the mean spike. (σ = 100ms, $r = 0.930$).

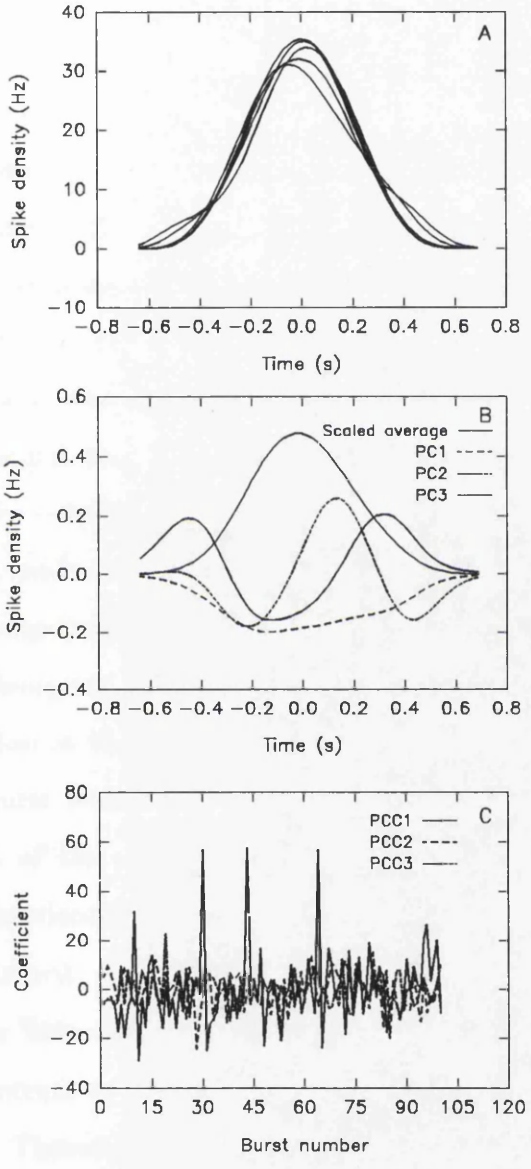

Figure 21: Karhunen-Loève transform of swimmeret motor neuron with windows centered at mean spike. A: Some sample spike density vectors. B: The first 3 principal components and the average scaled down for reference. C: The principal component coefficients. ($T_w = 1.35$s, $\sigma = 100$ms, $N_b = 100$, $N_v = 64$, $p = 3$). (PC: principal component, PCC: principal component coefficient)

increase proportionately. If the deviating spikes also carry some temporal information, the shape of the spike density would be distorted. The first principal component is of the same sign throughout the burst and therefore accounts for the spike count. However, it also carries some information about where the times at which the spikes occur or do not occur since it does not have the same shape as the average. Figure 22 shows that it correlates well with the burst spike count.

Applying the same procedure to an unidentified interneuron showed that the phase difference with respect to the motor neuron remained constant throughout a sequence of bursts (Figure 23), i.e. they both participated in the modulation of the burst period. The first principal component, shown in Figure 24, is all the same sign and is therefore likely to correlate well with the number of spikes in the burst ($r = 0.849$), since more area under a particular spike density curve is equivalent to more spikes. However, the component is narrower than the average spike density, so it also contains information about the burst width. One combination of these two features is the spike concentration in the burst. The concentration was defined as the spike count divided by the burst width, where the burst width was calculated as the root mean square deviation of the spikes in the burst. The first principal component coefficient correlated excellently with the spike concentration ($r = 0.963$, Figure 24), significantly better than with the spike count. The second principal component for the interneuron had the 'Mexican hat' shape which again reflects its representation of the burst width. In contrast to the first component, it had comparable positive and negative contributions. Therefore it does not contain any information about the spike count and seems to focus on the burst width. The correlation between the second principal component coefficient and the burst width was good ($r = 0.815$, Figure 24).

The result of applying the Karhunen-Loève transform to the slow membrane oscillations of the motor neuron is shown in Figure 25. The shapes of the principal components can be seen to resemble those of the spike density transforms. There was

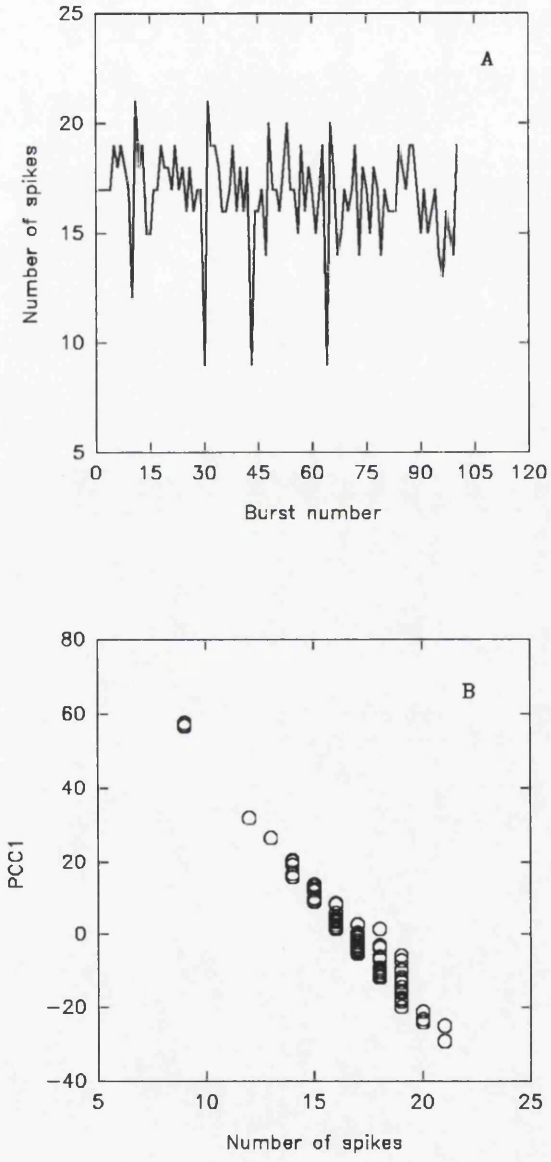

Figure 22: A: The number of spikes in the bursts. B: Correlation of number of spikes in burst and the first principal component coefficient (PCC1) ($r = 0.980$, $N_b = 100$).

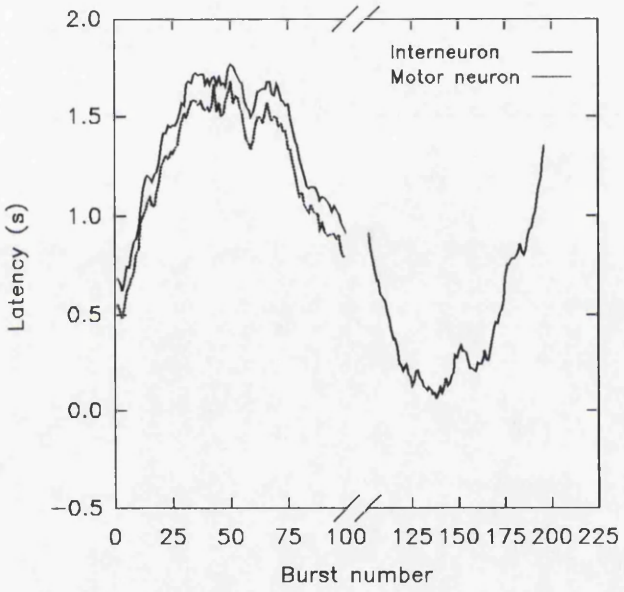

Figure 23: Left half: Latency comparison of a motor neuron (MN) and a interneuron (IN). Mean spike offset versus burst number. Mean latency difference = 132 ± 31 ms. Right half: The latency of the interneuron continued from the left half. The halfway break on the time axis is an approximately 0.5 s discontinuity in digitising the recorded data.

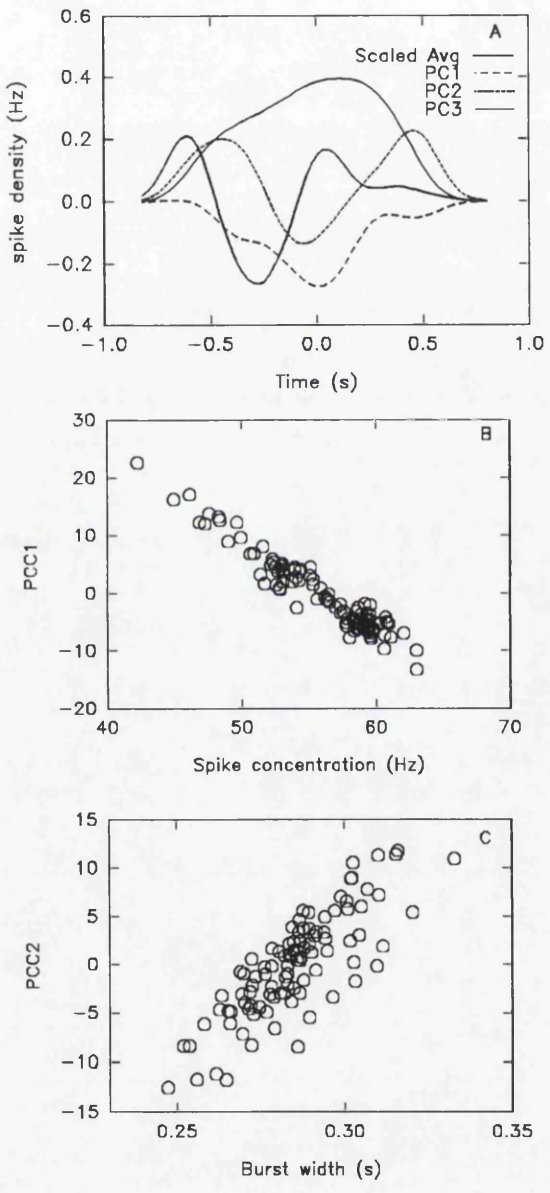

Figure 24: Karhunen-Loève transform of the interneuron. A: The principal components. B: Correlation of the first principal component coefficient and the spike concentration ($r = 0.963$) C: Correlation of the second principal component and the burst width ($r = 0.815$). (PC: principal component)

Signal	Windows	Variance
Spike density	periodic windows	2.142
Spike density	centered windows	0.033
Membrane potential	centered windows	5.07×10^{-5}

Table 1: The relative size of the deviations of the motor neuron.

no clear correlation between the coefficients of the slowly oscillating membrane potential and the corresponding coefficients of the spiking pattern ($r = 0.58$ for PCC1, $r = 0.56$ for PCC2). The obvious deviations ("spikes") observed in both PCC1s are caused by a few bursts being significantly different from the other bursts. Omitting these relatively large common deviations will reduce the correlation between the smaller deviations in the other bursts.

The Karhunen-Loève transform picks up any variance in the data set, even if it is very small. Hence, it would be useful to have a measure of the size of the deviations. The sum of all the n eigenvalues λ_i gives the total variance about the average for the whole input set. However, this variance depends on the scale of the input set, and it does not allow a comparison across different signal spaces, e.g. between spike density and membrane potential. The variance of the elements of the average vector $\mathbf{a}$, $\langle a_i^2 \rangle - \langle a_i \rangle^2$, provides the scale of the input vectors. The relative size of the deviations becomes

$$\frac{\frac{1}{n}\sum \lambda_i}{\langle a_i^2 \rangle - \langle a_i \rangle^2}. \tag{55}$$

The relative variances of the various signals are shown in Table 1.

5.1.4 Discussion

The non-random burst displacements were discovered without any a priori suspicion of their existence. This shows that principal component analysis is an "unbiased critic" that extracts the regularity and consistent deviations in the input data. These

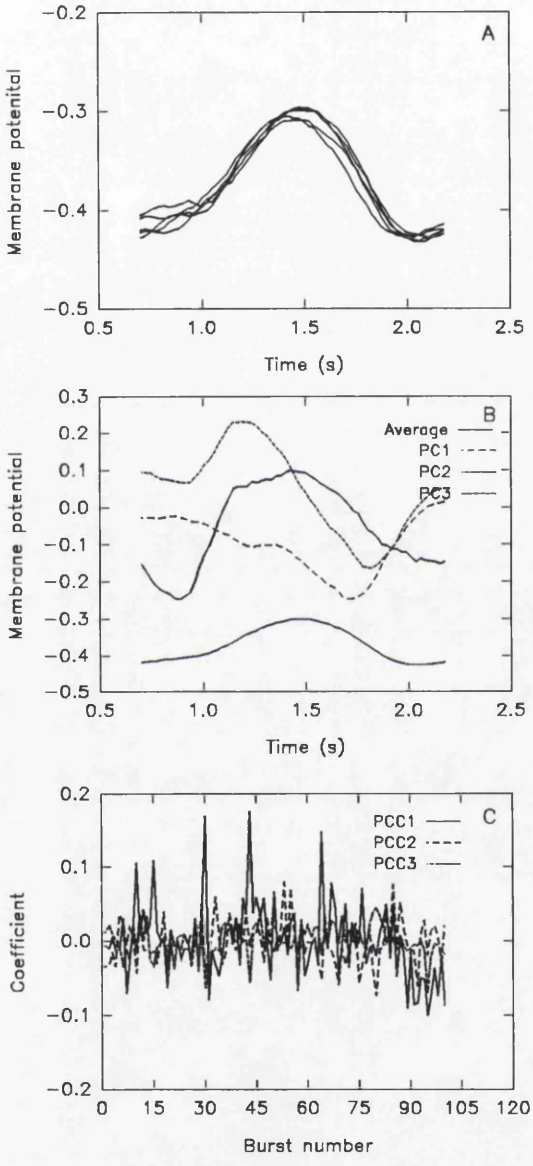

Figure 11

Figure 25: Karhunen-Loève transform of the slow membrane oscillations of the motor neuron. The windows were centered on the mean spike. A: Some sample potential vectors. B: The first 3 principal components and the average potential. C: The principal component coefficients. ($N_b = 100$, $N_v = 64$). (PC: principal component, PCC: principal component coefficient. The potential has arbitrary origin and scale.)

features may not be clear in a stream of spike events viewed by a human interpreter, who would have to search explicitly and successively for possible features.

The technique quantifies individual bursts in the context of the similarity and dominant variations of a set of bursts, whereas conventional analysis techniques, like post-stimulus histograms and coherence analysis, can only quantify the average behaviour of the data set as a whole or in sections. The PCA technique therefore has advantages when the bursts differ significantly. For example, the burst displacements could not have been identified by the conventional techniques.

The significant components of the progress of bursts through time were readily extracted. The activity of neurons as a function of time is unlikely to be easily expressed in terms of classical mathematics. However, the method enables the activities to be related to intuitive concepts by graphically displaying the extracted features.

When the windows were centered on the mean spike there was no systematic displacement modulation. This implies that the arithmetic mean spike is a good measure for the phase offset of the bursts. Having eliminated the burst deviations, there was no apparent pattern in the principal component coefficients, even though the components were identified as representing meaningful quantities like spike count and spike concentration.

The results showing that the slow membrane oscillations and the spiking patterns are uncorrelated suggest that these measures represent different activities of the system. The spiking pattern is not fully determined by the slow membrane potential. There must be other state variables of the system than the membrane potential that modulates the spiking on time-scales larger than the spike width. Since the deviations of the slow membrane potential were much smaller than the spike density deviations, there are other processes that control the spiking than the almost stationary membrane potential.

Richmond & Optican [1987] correlated their first principal component with the

spike count in the response but were unable, because of the signal complexity, to say anything else about the interpretation of the principal components. Here, the initial first principal component was correlated with the burst displacement, and further principal components were correlated with spike count, spike concentration, and burst width.

Further applications of the transform include determining the altered processing when drugs are applied and where hyperpolarising and depolarising intra-cellular currents are injected. It could also be a useful tool for classifying and comparing distinct neuronal classes, particularly at the sensory and motor flanks of the nervous system.

5.2 Leech swimming

The leech is composed of 21 nearly identical segments. Each segment contains a ganglion with approximately 400 cells that mostly occurs in contralateral pairs. The swimming incorporates the 16 midbody ganglia. The body maintains one wavelength regardless of the swimming frequency. Neighbouring ganglia have a phase difference of approximately 15 degrees and the cycle period is typically 0.8–1.5 seconds. This behaviour is therefore similar to that of the crayfish swimmeret. The phase locking cannot be achieved by simple fixed delay connections, since that would not work over a large range of cycle periods. There must be a mechanism that adjusts the effective coupling to match the swim frequency. Kopell & Ermentrout [1988] have proposed a mechanism for this phenomenon although it has not verified that the leech actually implements it.

The motor output is recorded by suction electrodes from the right and left nerves connecting the ganglion to the corresponding muscles in the body wall. The spikes are recorded extra-cellularly, and a simple threshold algorithm extracts the occurrence of

the spikes. Typically 3–4 recording electrodes will measure the activity at different ganglia along the length of the leech. A single ganglion has recently been shown to produce the swimming rhythm on its own. The leech swims in the vertical plane, so the right and left motor outputs of a ganglion are similar. Experiments suggest that each half-ganglion has oscillatory circuitry, although independent swimming activity has not been observed in isolated half-ganglia. Anatomical and physiological studies show extensive coupling between the two half-ganglia, but the strength of the intersegmental coupling between ganglia is largely unknown. Axons are known to run the length of the animal, but the position and the strength of the synapses onto the various ganglia are not known.

W. Otto Friesen is running a project at the University of Virginia that aims to determine the intersegmental coordination of the swimming. Initially, it would be instructive to compare the strength of the intrasegmental coupling between the two half-sides in a ganglion to the intersegmental coupling between ipsilateral half-sides in different ganglia.

The principal components encompass the highest variance of the activity. The coupling strength will determine how well the burst deviations causing the variances correlate. If the coupling between the two channels is strong, the burst deviations from the average burst of two channels correlate well. If the coupling is small, it means other processes, like e.g. other synaptic input and noise, determine the activities of the two channels. The correlation would therefore be small.

Craig G. Hocker at the University of Virginia provided simultaneous records of spikes from the right nerve of ganglia 2, 10, and 16, and from the left nerve of ganglion 10. These records will be labelled 2R, 10R, 16R, and 10L, respectively. The recording lasted 100 seconds and covered 74 bursts. The burst separators were obtained from the spike density using $\sigma = 200$ ms and a threshold of 5. The subsequent analysis used $\sigma = 100$ ms.

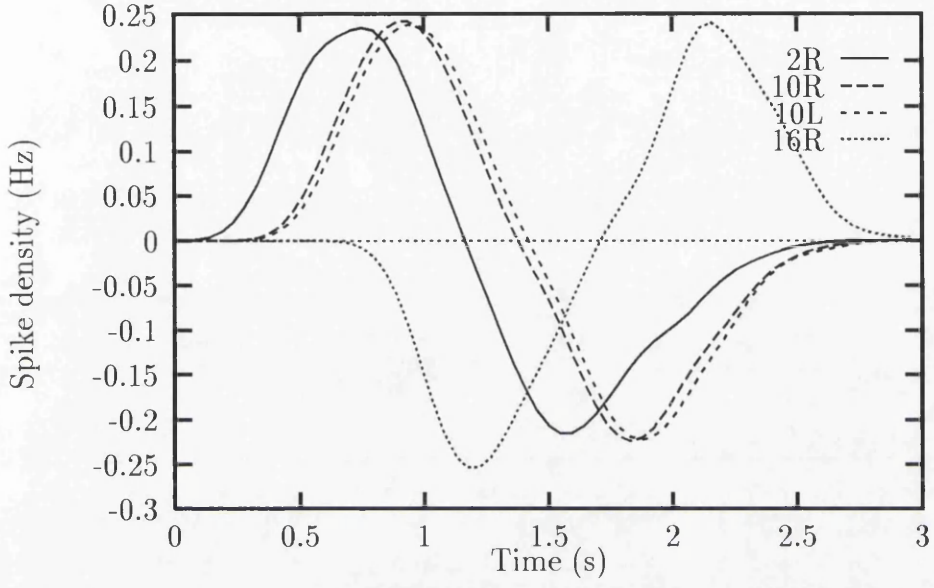

Figure 26: Principal component one of the leech swimming with periodic windows. Like for the crayfish, it represents the latency of the bursts with respect to the periodic windows.

5.2.1 Results

The mean periodic windows were generated by linear regression from the average of the 4 sets of burst separators. This creates common windows for all channels. The Karhunen-Loève transform produced very similar results to those of the crayfish swimmeret (Figure 26).

The shape of the first principal components suggests that they represent the burst latency. The coefficients correlate well with the latency of the 4 channels ($r \approx 0.9$ for each channel). Figure 27 shows the latencies of the mean spike in the 4 channels relative the periodic windows, as well as the average latency. The modulation of the burst latency as seen in Figure 27 resembles that of the crayfish. However, in the leech, this behaviour had already been discovered.

The mutual correlations between the first principal component coefficients are

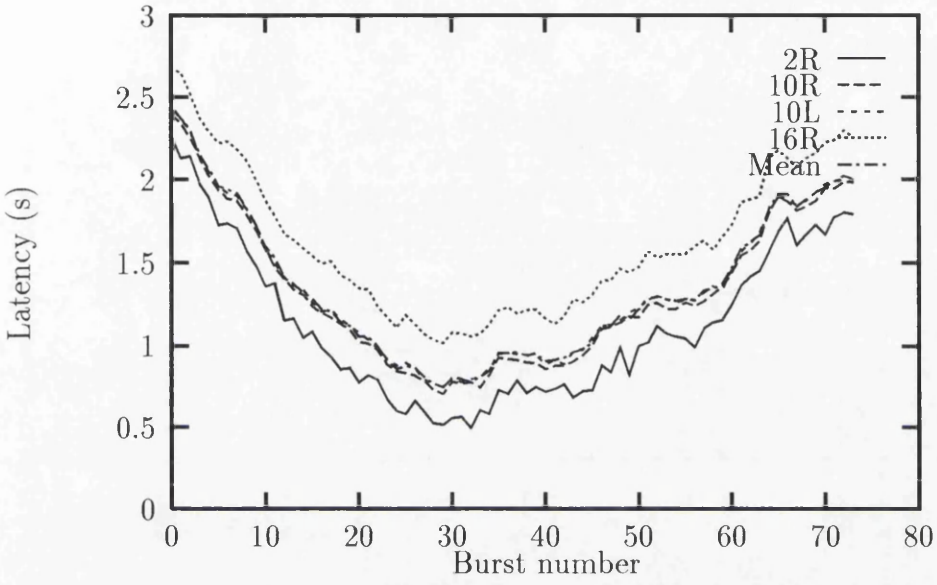

Figure 27: Latencies and average latency of mean spike with respect to the mean periodic windows. The window period was 1.33 seconds.

high ($r > 0.95$), as seen in Table 2. All the channels therefore participate in the burst latency modulation, and the phase locking is strong. The correlation between 10R and 10L is not significantly higher than the other correlations. Any difference in the coupling strengths therefore does not appear through this analysis. This is due to the large latency deviations that dominate the variance and therefore the principal components. All ganglia participate in the phase-locked swim pattern, so a difference in the coupling strength will only appear on a smaller scale.

On that account, the windows were centered on the mean spike of each burst. The mean spike was calculated from the corresponding bursts of the 4 channels. Like the case of the crayfish swimmeret, this will focus the Karhunen-Loève transform on the finer detail of the bursts after having subtracted out the latency variations. The windows are again common to all channels. Figures 28 and 29 show the first 2 principal components of the 4 channels with the windows centered on the mean spike.

channel	2R	10R	10L	16R
2R	1	0.9639	0.9615	0.9546
10R	·	1	0.9975	0.9940
10L	·	·	1	0.9930
16R	·	·	·	1

Table 2: Correlation between the first principal component coefficients with periodic windows. The table is symmetric about the diagonal. The correlation between 10R and 10L is not significantly higher than the other correlations.

Table 3 shows the variance included in the first principal components, as well as the distribution of the variance across the significant principal components.

Channel 10L and 10R have different shapes. The interpretations that were developed for the crayfish suggest that the first principal component for channel 10R represents the latency and for channel 10L it represents the spike concentration. That means that the main component of the variances are due to different behaviours. The variance of channel 10R is mostly caused by latency deviations, and channel 10L mainly alters its spike concentration. If there was strong coupling between the two contralateral channels, one would expect this to result in similar principal components since the two halves play the same rôle in the generation of the swim pattern.

Channel 16R also displays dominant latency deviations, whereas the shape of principal component 1 of channel 2R suggests the bursts mainly differ in the spike count.

The mutual correlation coefficients between the four channels are given in Table 3. The correlation is higher between 10R and 10L than any other pair of channels. However, it is not high enough to suggest a great difference in coupling strength. It was realised that the correlation between PCC1 of 10R and PCC2 of 10L was higher ($r = 0.5165$). Since the variance represented by the first 2 principal components are approximately equal (Table 4), it could be argued that the correlations between the

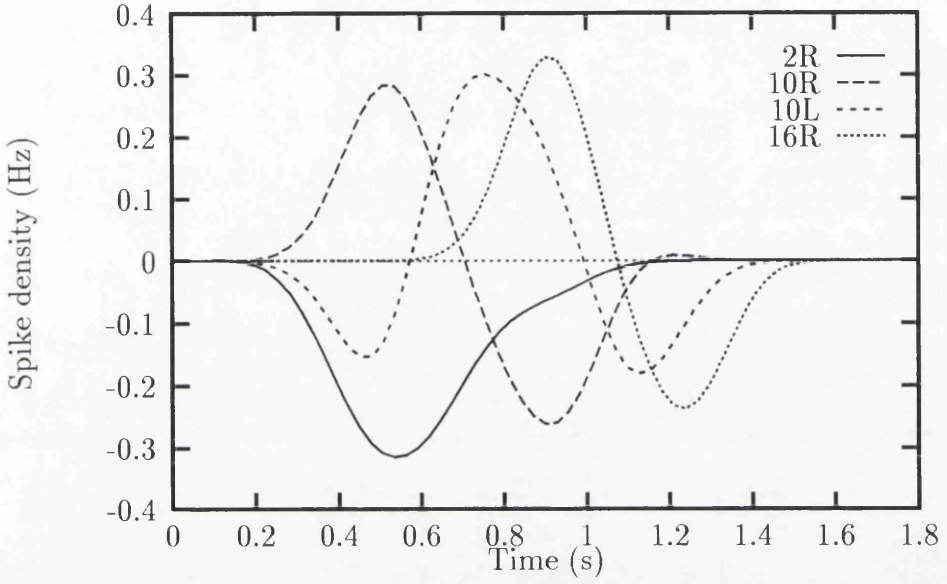

Figure 28: Principal component 1 with the windows centered at the mean spike. 10R and 10L have different shapes.

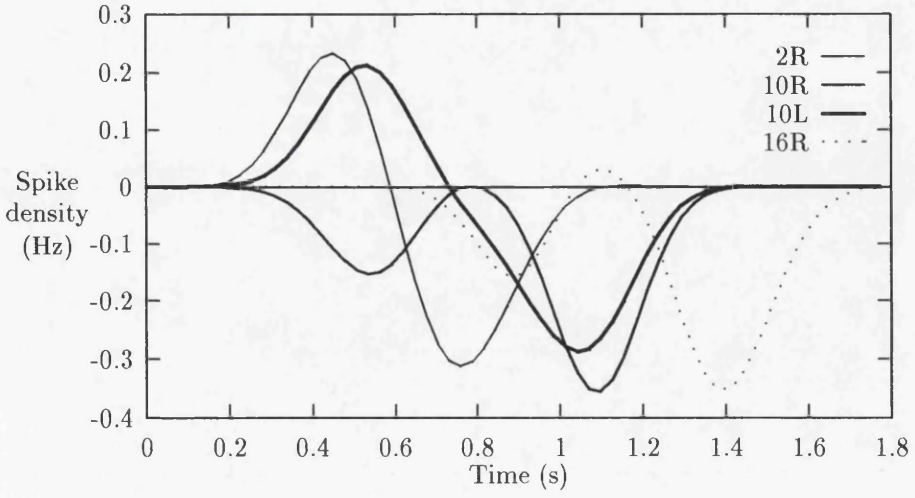

Figure 29: Principal component 2 with the windows centered at the mean spike.

channel	2R	10R	10L	16R
2R	1	0.1808	0.0248	0.2388
10R	·	1	0.3780	0.0763
10L	·	·	1	0.2140
16R	·	·	·	1

Table 3: Correlation coefficients between the first principal component coefficients with windows centered at the mean spike. The table is symmetric about the diagonal.

Channel	2R	10R	10L	16R
Total Variance	347.8	106.8	104.3	116.5
Variance in PC 1 (%)	52.0	34.3	34.9	38.1
Variance in PCs 1–2 (%)	88.0	64.7	65.1	66.4
Variance in PCs 1–3 (%)	92.8	84.2	86.6	86.7

Table 4: The efficiency of the Karhunen-Loève transform using windows centered on the mean spike. The variance is in units of Hz^2 times the sample interval. It represents the total area between the input vectors and the average vector. The principal component transformations cover the given fractions of that area. (PC = principal component)

multi-dimensional coefficient spaces ought to be tested (see section 6.2). However, Figure 30 shows that the correlation using the first 2 principal component coefficients is still too small to claim any strong coupling between 10L and 10R. Since the other correlation coefficients are even smaller, there is no evidence for any further intersegmental coordination beyond the strong phase locking.

The normalised deviations, as measured by equation (55), are shown in Table 5. The size of the deviations of 10L and 10R are almost equal, but 2R have a much larger variation. Although the shapes of the bursts do not correlate, this shows that the contralateral outputs from 10L and 10R are more tightly coupled than for example 10R and 2R. There is obviously other input to 2R that causes the extra variance.

Channel	Periodic	Centered
2R	5.78	0.279
10R	4.50	0.020
10L	4.31	0.022
16R	5.48	0.056

Table 5: The size of the deviations relative to the average signal for periodic windows and windows centered on the mean spike. The table shows the mean variance of the deviations per variance of the average.

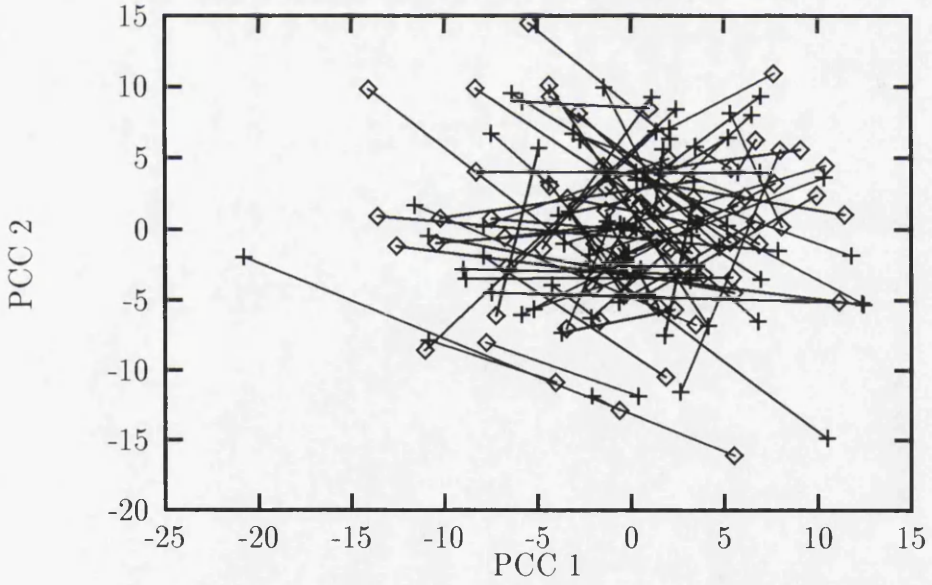

Figure 30: The first 2 principal component coefficients (PCCs) for 10L (diamonds) and 10R (plus). The points of 10L and 10R that represent the same burst are joined by a line. In general, neighbouring points of 10L diverge to different regions of activity of 10R, and vice versa. There is therefore no significant correlation between the two sets of points.

5.2.2 Conclusions

There are two ways to measure the coupling strength between two half-ganglia. The obvious way is to study the postsynaptic potentials in one half-ganglion produced by action potentials in the other. This gives an *absolute* measure on the coupling strength. The second approach is to characterise the coupling in terms of behavioural correlations. Behavioural correlation exists if a deviation in behaviour of one half-ganglion corresponds to a change in behaviour of the other half-ganglion. This gives a *relative* measure of the coupling strength, since other sources that produce activity are taken into account. At the motor output, any spike produces muscle contractions regardless the source of the spike. The coupling is therefore only as strong as its ability to compete with other sources of activity. Such sources could be synaptic input from parts not participating in the swim generation, noise within a neuron, and noise emerging from the network of neurons in the half-ganglia.

In the leech, it is very hard to find pairs of cells in different ganglia that have direct synaptic connections. Thus, the measurement of absolute coupling strength is impractical. Principal component analysis allows the quantification of behaviour. Hence, the analysis can be expanded from studying the effects of single spikes to quantifying a sequence of spikes that represent behaviour over time. The Karhunen-Loève transform can also estimate the relative coupling of behaviour.

The results of the analysis show that all channels feature equally strongly in the phase locking of the swim pattern. The contralateral channels within the same ganglion are no more phase locked than any other pair of channels.

After removing the modulation of the swim period, there was no sign of any significant correlation between any two channels. The system therefore appears to do no more than efficiently phase lock the segments along the body of the leech. Beyond that, the channels are dominated by other sources of activity. The contralateral outputs from ganglion 10 had only a very small variance with respect to the size of

the signal. This shows that these two outputs are very tightly phase locked. Ganglion 2 had a slightly larger variance. However, this variance still remains relatively small compared to the average signal. So even though the ganglion is located in a very anterior position, it remains relatively tightly phase locked. The variance of the midbody ganglion 16 was of the same order as those from ganglion 10, which shows that the phase coupling is very strong along the midbody ganglia.

Finally, the behavioural correlations suggests that the relative magnitudes of the intrasegmental and intersegmental physical coupling are indistinguishable.

Intuitively, short-range, and hence short-delay, graded synaptic transmission poses as a better mechanism for coupling two oscillators. However, these results show that long range spike mediated communication is as reliable and efficient in phase locking the bursts as short range synaptic transmission.

5.3 Summary

The Karhunen-Loève transform was developed to quantify periodic and quasi-periodic signals. It allowed the output of central pattern generators to be quantified in behavioural terms rather than at the level of individual spikes. The computation of neural systems is hard to detect when considering single spikes. The proper computation is only exposed when the analysis account for the statistical nature and sequence of events found in biological signals.

In particular, the Karhunen-Loève transform enabled the discovery of the latency modulation in the crayfish swimmeret, the comparison of spiking activity with the slow membrane potential, and, through correlation of behaviour, the study of coupling strengths in the leech.

Chapter 6

Conclusions and further work

6.1 Conclusions

The fundamental problem faced by experimental neurobiologists is that only one state variable per neuron is readily accessible, i.e. the membrane potential at the cell body. Therefore, it is unclear how even a single neuron operates; to trace the computation of a population of neurons poses an even grander challenge. This thesis has presented an approach to the investigation of neuronal systems: small invertebrate neuronal circuits are considered, the experiments are complemented with modelling, and the analysis accounts for the unobservable state variables.

The first part of the thesis presented a tool for modelling neuronal circuits. Its object-oriented design has made it possible to write most of the code in a general manner without referring to special cases for different classes of currents and synapses. Only a very small fraction of the total code is particular to the various classes of currents and synapses, and the small amount of specific code is isolated from the main part of the program. This guarantees that the general code can handle an expansion of the tool with new synapses and currents without any modification. The tool is therefore very flexible and expandable, and contrary to many existing

simulation tools, this flexibility has been achieved without a loss of performance.

The graphical user interface allows the user to manipulate neurons, synapses, and currents visually on the screen. This is an important step towards making simulation tools easy to use and thus enabling experimental neurobiologists to complement experiments with modelling in the laboratories.

The tool has been equipped with a database mechanism that allows the parameters and state variables to be controlled at the invocation of the program. A series of runs can be set up and then the program goes sequentially through the simulations, possibly with some analysis after each run. This even allows another program to set up the simulations. The modeller is free therefore to do other things as the computer performs the simulations. On the contrary, with a graphical user interface the modeller has to stay at the computer to select the parameters for each run.

The tool combines the advantages of both the graphical user interface and the latter batch runs. The initial prototyping makes use of the graphics interface, after which the model is stored as a default configuration for later batch runs, when the parameters are typically tuned to a particular behaviour.

The modelling tool has been implemented on a parallel computer. A central processor distribute the model configurations to many slaves, all running the tool. This allows many simulations to be run simultaneously. The communication overhead proved to be negligible so the output scales linearly with the number of processors available.

The performance of the tool is approximately 50 times that of an existing tool, *Nodus*, on a comparable computer. A single transputer in the department's parallel computer has the same performance, so with 100 transputers, 5000 times the speed of *Nodus* is achieved.

The tool was tested on the leech heart simulation and reproduced exactly the bursting activity of the *Nodus* implementation. By altering the reversal potentials of

Na^+ and Ca^{2+} according to the changes in extracellular concentrations, the model produced slow oscillations like the real system. The eigenvalues of the Jacobian confirmed that the fast Na^+ conductance had to be reduced by 33% to obtain stable oscillations. The period of oscillation was demonstrated to depend on both the synaptic strength and the conductance of the h-current.

The normal simulations of the leech heart with actionpotentials ran approximately 10 times slower than real time, and the slow oscillations ran roughly at real time. This is close to the speed required to use dynamic voltage clamp. Dynamic voltage clamp allows the synaptic interaction between two real cells to be modelled, or alternatively between a real and a model cell. The technique will become important for assessing temporal interactions between cells now that the modelling achieves near real time performance.

The application of the Karhunen-Loève transform to the crayfish swimmeret showed that the principal components represented features like spike count, burst width, burst concentration, and burst latency. The principal component analysis tells you which features that dominate the deviations from the average. Therefore, one avoids testing successively and explicitly through a large set of possible features. The technique serves as an "objective critic" that display the largest sources of variance over the data set.

The burst latency proved to be significantly modulated with a period of 200 bursts. The latency is related to the instantaneous burst period by the period being the derivative of the latency. Consequently, the burst period is modulated by a sine wave of period 200 bursts. A similar observation has been done in the leech swimming system, as well as the lamprey spinal cord. It seems like that long-term modulation of the output of a chain of coupled oscillators is common across swimming in different species.

The analysis was also performed on the membrane potential after having removed

the actionpotentials with a low pass filter. The membrane potential deviated with similar features from the average, but the deviations did not correlate with those of the spikes from the same neuron. This demonstrates the point that the membrane potential at the cell body does not solely determine the spiking pattern, but other (unobservable) state variables influence the spiking dynamics. The recent imaging techniques promise to measure the spatial distribution of the membrane potential and ion concentrations throughout a cell, as well as the temporal interaction between these state variables.

The leech swimming was shown to have burst latencies that are modulated similarly to that of the crayfish swimmeret. This modulation was the most significant variance and was effective throughout the length of the animal. The anterior and posterior ganglia as well as between left and right were closely coupled. Hence, the phase locking is strong for all outputs.

After removing the common latency deviations, there was very little coupling between the outputs. Also, the highest variance features differed for the 4 outputs. Therefore, there is only very little coupling between the outputs apart from the latencies. This is even the case for contralateral outputs from the same ganglion. Hence, these results suggest that the long range intersegmental coupling is no weaker than short range intrasegmental coupling, and the coupling performs phase locking only.

This work has shown that the Karhunen-Loève transform can be used to quantify the relative coupling strength between two outputs. The coupling strength is estimated at the level of behaviour and relative to normal background activity in the system. This compares to the absolute estimate that at the level of individual synapses measures the coupling strength between two cells. The techniques above enables the effective coupling between two cells to be measured, where the coupling may occur through several pathways and through several cells.

In general, the modelling tool and the principal component analysis techniques

have both been used to explore temporal computation in small nervous systems. The modelling tool has enabled one to "record" from hidden state variables and to manipulate inaccessible parameters of the real neurons. The principal component analysis has enabled neuronal output to be quantified in such a way that it incorporates feedback loops through hidden state variables in the activity measure, yet the measure is easy to analyse further because it is low-dimensional. As a result, the study of temporal interactions between the state variables in small central pattern generators has brought some insight into the mechanisms of computation in neuronal systems.

In order to trace the computation in a single neuron, the thesis has argued the need to look for the state variables that are inaccessible by conventional recording techniques. The interactions between these states are so rich and extensive that it is a tremendous task to identify the states and the interactions only from the membrane potential in the cell body. Therefore, more effort should be directed towards developing the optical imaging techniques that promise to record the membrane potential and the ion concentrations throughout the spatial extent of the cell. However, these techniques will not determine every state variable and interaction to the greatest detail, so modelling will continue to play a crucial role, but would benefit from a few more fixed points determined experimentally.

Likewise, the optical imaging techniques have been used to record from thousands of neurons (Falk et al. [1993]; Morton et al. [1991]). The experimentalist now faces the rather strange situation where one has large amounts of data but does not quite know how to analyse it. As shown with the principal component analysis of the crayfish swimmeret and the leech swimming, that analysis is capable of determining the coupling strength between any two neurons based on their correlating spiking patterns. Therefore, the Karhunen-Loève transform would quantify aspects of the large-scale computation of neurons in a network.

6.2 Further work

Further developments of the leech heart model will incorporate spike mediated synaptic transmission and multicompartmental neurons so that more realistic behaviour is achieved. In particular, the current model produces too abrupt transitions. Also, a real neuron cannot be properly voltage-clamped. A multicompartmental model is required therefore to understand the spatio-temporal interactions during voltage-clamp. In the long term, the current model will form the foundation for an extensive model of the leech heart that incorporates all the interneurons from the 7 anterior ganglia. This model will demonstrate modulation of the heart rhythm by neuromodulators and the switching behaviour between alternating rhythms in the left and right heart tube.

Large amounts of literature exists on the identifiability of model parameters particularly in compartmental systems. Although some of these methods apply more to systems with linear or weakly nonlinear properties, many of the techniques can be applied to neuronal modelling. System identification concerns about designing experiments that best determine the model parameters. Conversely, it gives the error bounds on the parameters for a particular experiment. In particular, the extension of the leech heart model to multi-compartmental neurons would take great advantage of these techniques. Godfrey [1983] gives a clear account of the techniques involved in system identification of compartmental models.

One technique that is particularly relevant is sensitivity analysis. It would provide valuable extra insight to vary the parameters slightly while observing the change in behaviour. For example, the sensitivity of the period of the slow oscillations to the various conductances may determined this way. If the period is highly dependent on a particular parameter, it is likely that the parameter is well defined. However,

if the period varies very little with a parameter, that parameter cannot be determined within low error bounds. Lehman & Stark [1982] applies this technique to the occu-motor-vestibular system. The method can be extented to look at the instantaneous sensitivity of the state variables to the parameters. This involves integrating a cosystem of differential equations. Both these methods of sensitivity analysis are easy to implement and interpret. The slow oscillations and the extension to the multi-compartmental leech model should both benefit from these.

In general, the parameters differ significantly between individual neurons. The parameters presented of the leech model presented in Appendix B are averages over several neurons. The sensitivity analysis would show how robust the model is with respect to these parameters. If the model is insensitive to parameter changes, the large spreads are random, and the model is very robust. On the other hand, if the model is very sensitive there must exist mechanisms within the neurons that tune the parameters to the suitable values.

When extracting the variance from a data set with the Karhunen-Loève transform, the variance included in the first 2, say, principal components may be be comparable. If one wants to correlate the coefficients with some other activity, it would be desirable to correlate multidimensional vectors, rather than single numbers. For example, this would enable the 3-dimensional outputs from each of the 4 channels of the leech swimming to be correlated with each other, and thus obtaining a better measure of the coupling between the channels.

It can be tested if two multi-dimensional vectors $\mathbf{u}$ and $\mathbf{v}$ with zero mean and possibly different dimensions are related with the linear transform,

$$\mathbf{v} = \mathbf{A}\mathbf{u}.$$

The matrix $\mathbf{A}$ can easily be obtained with a least squares fit. However, a measure for the degree of correlation is not obvious. A literature search for the correlation

coefficient of a multi-dimensional linear fit should be undertaken. On the basis of such a correlation coefficient one would be able to tell whether a set of principal component coefficients from one channel are approximately scaled, translated, and rotated coefficients from another channel.

Appendix A

Neurolab class specifications

The following description aims to document the object-oriented structure of Neurolab. The variables and the functions of each of the main base classes are explained. The interface listing is presented in section A.4.

A.1 Class `Cell`

Variables

The class contains the following variables:

`PList convSyns, divSyns;` These are lists of references to converging and diverging synapses to the cell, respectively.

`PList currents;` This is a list of references to the currents operating in the cell, excluding the synaptic currents.

`int Vm;` This is the state reference of the membrane potential of the cell.

`real Eleak, gm, Cm, Iext;` These are the parameters of the passive leakage current of the cell, as well as the membrane capacitance and the externally injected current.

`char* name;` This is the name of the cell that identifies the cell in the user interface, e.g. in parameter assignments and state recordings.

Functions

The class is operated on by the following functions:

`void AddConvSyn(Synapse*);` Adds a converging synapse to the cell.

`void AddDivSyn(Synapse*);` Adds a diverging synapse to the cell.

`void AddCurrent(Current*);` Adds a current to the cell.

`Current* CurrentAt(int i);` Returns the ith current.

`Synapse* ConvSynAt(int i);` Returns the ith synapse.

`real I_syn(const real *state);` Returns the total synaptic current for the given state by adding the currents from all converging synapses.

`real I_active(const real *state);` Returns the total active current for the given state by adding all currents, excluding synaptic currents.

`real I_stim(real t);` Returns the external applied current at time `t`. In the current version, it is simply `I_ext`. The stimulus can therefore only be changed at each step of integration, rather than continuously. Note that such a change would have to be implemented centrally in the integration method and is an example of non-object-oriented design. It would be much better to have a pointer to a stimulus that calculates the current at any time instead of simply returning `I_ext`.

`real pot(const real* state);` Returns the membrane potential of the cell. This is found at the `Vmth` position in `state`.

`void Init( real* state, real v0 );` Initialises the states of the cell and the active currents to the steady state values at a membrane potential of `v0`.

`void InitSyns( real* state, real v0 );` Initialises the states of the synapses and the corresponding currents. Note that this has to be called after

`Init(real*,real)` above, since it requires *both* cells that are linked to a synapse to be initialised.

`void CalcDeriv( const real* state, real t, real* dydt );`
Calculates the derivative of the state vector at time `t` and returns it in `dydt`. It calculates all the currents in the cell to find the derivative of the membrane potential and then redirects the call to the currents, including the synaptic currents.

`void Step( const real* from, real* new, real dt );` Takes a step of length `dt` from the state vector `from` and saves the new state in `new` using the method due to Friesen. It first calculates the change in the membrane potential and the redirects the call to the currents, including the synaptic currents.

`void ODEregister( int& N );` Registers the states of the cell. On input, `N` is the number of states already registered. On return, `N` is incremented to reflect the states required by the current cell, its currents, and its synapses. The state references are assigned their values as `N` is incremented.

`void RegisterDB( DataBase *db, real *state );` Registers all states, parameters, and functions that are to be accessed externally using the following function.

`void DBregister( DataBase*, int type, const char *name, DBentry* );`
This is the function that registers individual database entries and makes sure the name of the entry is unique and hierarchical. `Type` states whether the entry is a state, parameter, or function. The database mechanism is explained in section 2.4.

A.2 Class Current

The cell class is not intended to be inherited by user defined classes. In contrast, the class `Current` only serves as a template for user defined classes that implement the particular currents. Below, the C++ keyword `virtual` appears in the function declarations to provide a link to the actual implementation. The derived classes will

implement a function with the same name and argument list, and C++ automatically redirects the calls to the derived class. The compiler achieves this by adding a variable to the class `Current` that points to the function in the derived class. The virtual functions in class `Current` implement a default behaviour. If this behaviour is desired, it must be explicitly called from the functions of the derived class.

Variables

`real gmax,Erev;` These two parameters are common to all currents and are the maximum conductance and the reversal potential, respectively.

`char *name;` This is the name of the current for identification within the database mechanism.

`Cell *cell;` This is a reference to the cell in which the current is operating.

`int Vm;` The membrane potential of the cell in which the current is operating is state number `Vm` in the central state vector.

Functions

`virtual char* ClassName();` This returns the class name of the current. This is required when saving and loading cells.

`virtual void ODEregister( int& N );` Registers the state variables of the current and updates the state counter `N`.

`virtual void Init(real *state, real v0);` Initialises the states of the current to the steady states at membrane potential `v0`.

`virtual void Step(const real*,real*,real dt );` Performs a step of Friesen's integration method.

`virtual void CalcDeriv( const real *state, real t, real *deriv );` Calculates the derivative of the state variables belonging to the current. Note that

`Vm` does not belong to the current but to the cell in which the current operates.

`virtual real I(const real *state);` Returns the value of the current for the given state. By default it returns `g(state)*(state[Vm]-Erev)`. This is adequate for the current set of current classes since they redefine `g(state)`.

`virtual real g(const real *state);` Defines the conductance of a derived current in terms of its states. By default, it returns `gmax`.

`virtual void RegisterDB( DataBase *db, real *state );` Registers the parameters, states, and functions in the database. By default, it adds the parameters `gmax` and `Erev` and the functions `I` and `g` to the database. Each registration should be performed by the following function.

`void DBregister( DataBase*, int type, const char *ename, DBentry* );` Registers an entry with a hierarchical name defined by the name of the cell in which the current operates, the name of the current, `name`, and the base name of the entry, `ename`.

A.3 Class Synapse

Variables

`Cell *from,*to;` These are references to the cell which control the conductance of the synapse, `from`, and the cell in which the synaptic current operates, `to`.

`Current *I;` This is a reference to the synaptic current.

Functions

`Synapse( Cell* from, Cell* to );` This creates a directional synapse between the two cells and adds it to the lists of synapses kept in the cells.

A.4 Interface listing

This is the file `cell.h` that defines the interface to the 3 base classes.

```
#include "oho/plist.h"

class Cell;
class Synapse;
class DataBase;
class DBentry;
class Current;

/*
UNITS:
  V   I   t   R     g   C
  mV  nA  ms  MOhm  uS  nF
*/


class Cell {
public:
  PList convSyns, divSyns; // lists of Synapse*
  PList currents;          // list  of Current*

  int Vm;   // state pointer

  real   Eleak, gm, Cm;  // parameters
```

```
  real   Iext;
  char*  name;

  Cell();

  void AddConvSyn(Synapse*);
  void AddDivSyn(Synapse*);
  void AddCurrent(Current*);

  Current* CurrentAt(int i);
  Synapse* ConvSynAt(int i);

  real I_syn(const real* state);
  real I_active(const real* state);
  real I_stim(real time);

  real  pot(const real* state);

  virtual void CalcDeriv( const real* state, real t, real* dstate );
  virtual void Init( real* state, real v0 );
  virtual void InitSyns( real* state, real v0 );
  virtual void Step( const real*, real*, real ); // from,new,dt

  virtual void ODEregister( int& );
  virtual void RegisterDB( DataBase *db, real *state );
  void DBregister( DataBase*, int type , const char *name, DBentry* );
};
```

```
class Current {
public:
  real gmax,Erev;  // parameters
  char *name;
  Cell *cell;    // in which current is operating
  int Vm; // state pointer of Vm in 'cell' above, as in g*(Vm-Erev)

  Current( real g=0, real E=-80 );
  virtual char* ClassName();

  real I( const real *state );
  virtual real g(const real * state);

  virtual void Init(real *state, real v0);
  virtual void Step(const real*,real*,real dt );
  virtual void CalcDeriv( const real *state, real t, real *deriv );

  virtual void ODEregister( int&, int _Vm );
  virtual void RegisterDB( DataBase *db, real *state );
  void DBregister( DataBase*, int type , const char *name, DBentry* );
};


class Synapse {
```

```
public:
  Cell *from,*to;
  Current *I;

  Synapse( Cell *from, Cell *to );
};
```

Appendix B

Leech heart model

This appendix gives the parameters for the standard leech heart model as used by Calabrese & De Schutter [1992]. These differ slightly from those presented by De Schutter, Angstadt & Calabrese [1993]. Table 6 shows the units that are used in the Neurolab and in which the parameters are given.

Dimension	Unit
Potential	mV
Current	nA
Time	ms
Conductance	μS
Capacitance	nF

Table 6: The units of physical dimensions.

Table 7 shows the parameters of the ionic currents. These currents have the form

$$I_i = \bar{g}_i s_i (V - E_i)$$

where s_i is the state expression for each current. These expressions are also shown in Table 7 where the subscripts in the m and h are avoided for clarity. Table 8 shows the rate constants that all have the form

$$\frac{x_1 + x_2 V}{x_3 + \exp \frac{x_4 + V}{x_5}}$$

and are additionally clipped at 0 if they go negative. The derivative of the state m is given by

$$\frac{dm}{dt} = \alpha(V)(1-m) - \beta(V)m$$

and equivalently for h if it is present. The derivative of the membrane potential of a cell is given by

$$C\frac{dV}{dt} + \sum_i I_i(s_i) + I_{syn} = 0 \tag{56}$$

where i indexes the currents in Table 7. The current I_{syn} is given by

$$I_{syn} = \bar{g}_{syn} P_{pre}^3 (V - E_{syn})$$

where P_{pre} is a measure of the Ca^{2+} concentration in the *presynaptic* cell, and E_{syn} is the reversal potential of the postsynaptic current. The derivative of P is expressed as

$$\frac{dP}{dt} = 0.001 I_{Ca,pre} - \beta(V_{pre})P$$

where the subscript pre denotes the presynaptic cell, and

$$I_{Ca,pre} = \max(0, -I_{fastCa,pre} - I_{slowCa,pre} - \alpha(V_{pre})) \tag{57}$$

$$\alpha(V) = \max(0, \min(0.29, 0.66 + 0.012V)) \tag{58}$$

$$\beta(V) = \max(0, -0.000101V + 0.011\exp(-0.1(V+49)^2)). \tag{59}$$

With $C = 0.5$, $\bar{g}_{syn} = 700$, $E_{syn} = -65$, and two cells with reciprocal synapses, the model description is completed.

The initial conditions used to test the output of Neurolab with respect to Nodus are given in Table 9. These values produce a short transient before the model enters periodic bursting. This transient is shown in Figure 31, and the stable limit cycle was shown in Figure 7.

Current	s_i	$\bar{g}_i$	E_i
Fast Ca^{2+}	mh	0.020	122.5
Slow Ca^{2+}	mh	0.0075	122.5
Fast Na^{+}	m^3h	0.750	45.0
Persistent Na^{+}	m	0.001	45.0
Delayed rectifier (K^{+})	m^2h	0.075	-80.0
Slow K^{+}	m^2	0.100	-80.0
A current	m^2h	0.150	-80.0
h current	m^2	0.0125	-21.0
passive leak	1	0.0015	-40.0

Table 7: The ionic currents operating in a cell.

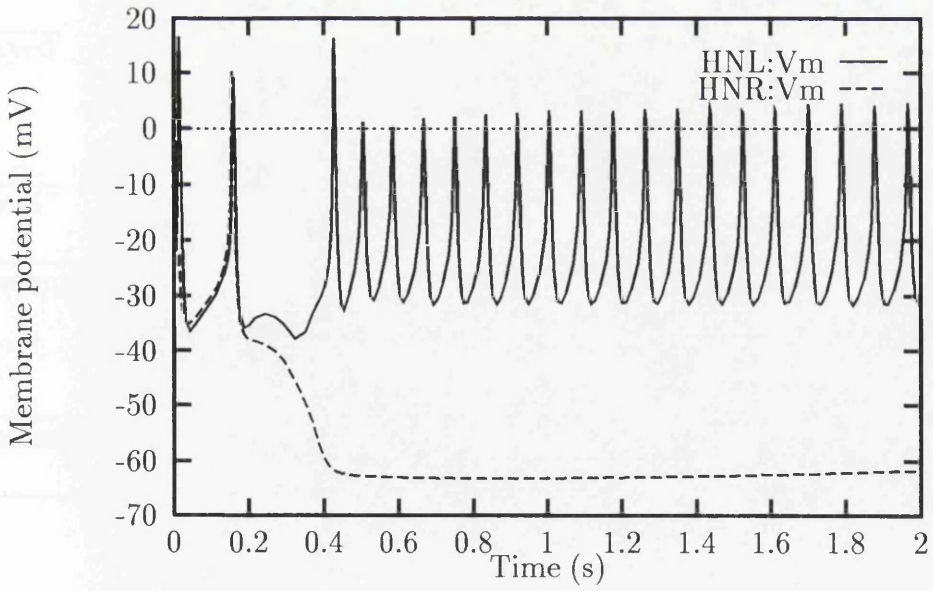

Figure 31: The transient of the leech heart model for the above initial values before a stable limit cycle (bursting pattern) is reached.

Current	State	Rate	x_1	x_2	x_3	x_4	x_5
Fast Ca^{2+}	m	α	0.04	0	1	39	-2.8
		β	7.41	0.13	-1	57	2
	h	α	0.005	0	0	61	5.6
		β	0.028	0	1	45	-3
Slow Ca^{2+}	m	α	0.0072	0	1	51	-3.5
		β	0.19	0	1	58	3.5
	h	α	0.0027	0	1	58	2
		β	0.04824	0.00072	1	67	6.5
Fast Na^+	m	α	-0.52365	-0.06982	-1	7.5	-5
		β	0.14	0	1	5.5	6
	h	α	0.01	0	1	23	2
		β	0.255	0	1	8	-5
Persistent Na^+	m	α	0.1	0	1	25	-6
		β	0.015	0	1	40	8
Delayed rectifier (K^+)	m	α	1	0	1	-10	-7.7
		β	1	0	8.5	72	28.6
	h	α	0.002	0	1	19	9.1
		β	0.00144	0	1	24	-5
Slow K^+	m	α	0.2	0	20	2	-5.9
		β	0.2	0	20	15	6.7
A current	m	α	0.335	0	0.86	32.5	-7.7
		β	2.48	0	7.5	50	8.3
	h	α	0.03	0	1	50	4.2
		β	0.029	0	1	56	-5
h current	m	α	-0.000783	-0.000018	1	43.5	10
		β	0.0015	0	1	86	23

Table 8: The rate parameters for the ionic currents.

State		HNL	HNR
Potential	V	-23.6669	0.2895
Fast Ca^{2+}	m	0.9473	0.999956
	h	0.0348	0.000371
Slow Ca^{2+}	m	0.6068	0.993821
	h	0.3851	0.299800
Fast Na^{+}	m	0.2066	0.8019
	h	0.8271	0.1952
Persistent Na^{+}	m	0.8463	0.9758
Delayed rectifier (K^{+})	m	0.1022	0.4342
	h	0.9762	0.6614
Slow K^{+}	m	0.0796	0.2575
A current	m	0.7572	0.9515
	h	0.0344	0.0042
h current	m	0.3668	0.2477
Ca^{2+} concentration	P	0.0156521	0.0211216

Table 9: The initial values of the model. The Ca^{2+} concentrations P for HNL controls the postsynaptic current in HNR and *vice versa.*

Appendix C

Parallel simulation

The overall configuration of the parallel implementation of Neurolab can be seen in Figure 32. A master processor sends out parameter and state values, i.e. a configuration, to a one of a series of slave processors. All slave processors run their own copy of Neurolab. When they have performed the simulation—possibly with some post-processing—they output the result to the master processor.

The protocol for distributing configurations works as follows: When a slave have finished its run, it outputs the result to the multiplexer (M in Figure 32) and signals to the demultiplexer (D) that it is ready to receive a new configuration. This causes the next configuration that arrives at that D to be routed to the Neurolab process. The following configuration will be passed on down the chain of demultiplexers to the next available Neurolab process, given that the first process is still computing. When the master process receives the result of a simulation, it knows that a slave processor is waiting for a new configuration. It saves the result for later post-processing and sends out a new configuration down the chain of demultiplexers. The configuration contains an identification parameter that is also sent back together with the result. The master process can thus identify which configuration the result belongs to.

This scheme works very well as long as each slave processor spends a relatively

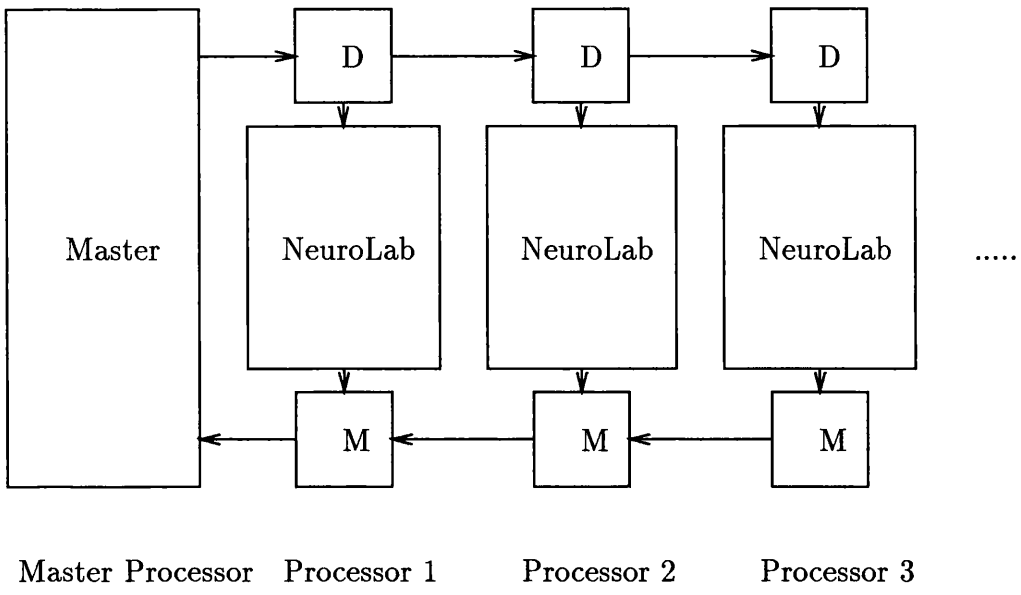

Figure 32: The organisation of the parallel implementation of Neurolab. Each slave processor has 3 processes running in parallel: its own copy of Neurolab, D—a demultiplexer of input data to Neurolab, and M—a multiplexer of output data from Neurolab.

long time (a minute) on simulating a single configuration. If there is a large set of configurations to be simulated, this scheme achieves true parallel processing that scales linearly with the number of processors available. The communication overhead is very small, since the only information that has to be transmitted is the configuration and the result. This represents only a very small fraction of the resources required to simulate the model.

The master process can read the set of configurations from a file or a set of files, it can generate the configurations itself by randomly shaking some of the parameters to study the robustness of the model, or it could in principle perform some sort of gradient descent (not implemented). It effectively serves as a center of computation where a large number of configurations can be dispatched simultaneously, and the

results are returned relatively quickly.

Bibliography

Földiák, P. [1989], "Adaptive Network for Optimal Linear Feature Extraction," in *International Joint Conference on Neural Networks* #1, IEEE, New York, Washington 1989, 401–405.

Press et al, W. H. [1988], *Numerical Recipes in C*, Cambridge University Press.

Arbas, E. A. & Calabrese, R. L. [1987a], "Ionic conductances underlying the activity of interneurons that control heartbeat in the medicinal leech," *J Neurosci* 7, 3945–3952.

Arbas, E. A. & Calabrese, R. L. [1987b], "Slow oscillations of membrane potential in interneurons that control heartbeat in the medicinal leech," *J Neurosci* 7, 3953–3960.

Bartnik, E.A., Blinowska, K.J. & Durka, P.J. [1992], "Single evoked-potential reconstruction by means of wavelet transform," *Biological Cybernetics* 67, 175–181.

Brillinger, D. R. [1975], "The identification of point process systems," *Ann Probab* 3, 909–929.

Bugmann, G. [1991], "Summation an multiplication: two distinct operation domains of leaky integrate-and-fire neurons," *Network* 2, 489–509.

Bunow, B., Segev, I. & Fleshman, J. W. [1985], "Modeling the electrical properties of anatomically complex neurons using a network analysis program: excitable membrane," *Biological Cybernetics* 53, 41–56.

Calabrese, R. L., Angstadt, J. D. & Arbas, E. A. [1989], "A neural oscillator based on reciprocal inhibition," in *Perspectives in neural systems and behavior*, T. J. Carew & D. Kelley, eds., Alan R. Liss, 33–50.

Calabrese, R. L. & De Schutter, E. [1992], "Motor-pattern-generating networks in invertebrates: modeling our way toward understanding," *TINS* 15, 439–445.

Crowe, J.A., Gibson, N.M., Woolfson, M.S. & Somekh, M.G. [1992], "Wavelet transform as a potential tool for ECG analysis and compression," *J Biomed Eng* 14, 268–272.

De Schutter, E. [1989], "Computer Software for Development and Simulation of Compartmental Models of Neurons," *Comput Biol Med* 19, 71–81.

De Schutter, E., Angstadt, J. D. & Calabrese, R. L. [1993], "A model of graded synaptic transmission for use in dynamic network simulations," *J Neurophysiol* 69, 1225–1235.

Falk, C.X., Wu, J.Y., Cohen, L.B. & Tang, A.C. [1993], "nonuniform expression of habituation in the activity of distinct classes of neurons in the Aplysia abdominal-ganglion," *J Neurosci* 13, 4072–4081.

Fukunaga, K. [1972], *Introduction to Statistical Pattern Recognition*, Academic Press, New York and London.

Gabor, D. [1946], "Theory of communication," *J of the IEE* 93, 429–457.

Gear, C. W. [1971], "The automatic integration of ordinary differential equations," *Comm ACM* 14, 176–179.

Getting, P. A. [1983], "Mechanisms of pattern generation underlying swimming in *Tritonia*. II. Network reconstruction," *J Neurophysiol* 49, 1017–1035.

Getting, P. A. [1989], "Reconstruction of Small Neural Networks," in *Methods in Neuronal Modeling*, C. Koch & I. Segev, eds., MIT Press, Cambridge, USA.

Godfrey, K. [1983], *Compartmental models and their application*, Academic Press, London.

Halliday, D., Murray-Smith, D. J. & Rosenberg, J. R. [1992], "A frequency-domain identification approach to the study of neuromuscular systems—a combined experimental and modelling study," *Trans Inst M C* 14, 79–90.

Hancock, P. J. B., Baddeley, R. J. & Smith, L. S. [1992], "The principal components of natural images," *Network* 3, 61–70.

Hebb, D. O. [1949], *The Organization of Behaviour*, Wiley, New York.

Hertz, J., Krogh, A. & Palmer, R. G. [1991], *Introduction to the theory of neural computation*, Addison-Wesley, Redwood City, CA.

Hindmarsh, A. C. [1983], "Odepack, a systematized collection of ode solvers," in *Scientific Computing*, R. S. Stepleman *et al.*, ed., North-Holland, Amsterdam, 55–64.

Hodgkin, A. L. & Huxley, A. F. [1952], "A quantitive description of membrane current and its application to conduction and excitation in nerve," *J Physiol London* 117, 500–544.

Kopell, N. & Ermentrout, G.B. [1988], "Coupled oscillators and the design of central pattern generators," *Mathematical Biosciences* 90, 87–109.

Lehman, S.L. & Stark, L.W. [1982], "Three algorithms for interpreting models consisting of ordinary differential equations: sensitivity coefficients, sensitivity functions, global optimization," *Math Biosci* 62, 107–122.

Linsker, R. [1986], "From basic network principles to neural architecture," *Proc Natl Acad Sci USA* 83, 7508–7512, 8390-8394, 8779-8783.

Linsker, R. [1988], "Self-organization in a perceptual network," *Computer* March, 105–117.

MacPherson, J. M. & Aldridge, J. W. [1979], "A quantitative method of computer analysis of spike train data collected from behaving animals," *Brain Res* 175, 183–187.

Morton, D.W., Chiel, H.J., Cohen, L.B. & Wu, J.Y. [1991], "Optical methods can be utilized to map the location and activity of putative motor neurons and interneurons during rhythmic patterns of activity in the buccal ganglion of aplysia," *Brain Research* 564, 45–55.

Oja, E. [1982], "A Simplified Neuron Model As a Principal Component Analyzer," *Journal of Mathematical Biology* 15, 267–273.

Oja, E. [1989], "Neural Networks, Principal Components, and Subspaces," *International Journal of Neural Systems* 1, 61–68.

Oja, E. & Karhunen, J. [1985], "On Stochastic Approximation of the Eigenvectors and Eigenvalues," *Journal of Mathematical Analysis and Applications* 106, 69–84.

Optican, L. M. & Richmond, B. J. [1987], "Temporal encoding of two-dimensional patterns by single units in primate inferior temporal cortex. III. Information theoretic analysis," *J Neurophysiol* 57, 162–178.

Pearlmutter, B. A. & Hinton, G. E. [1986], "G-Maximization: An Unsupervised Learning Procedure for Discovering Regularities," in *Neural Networks for Computing*, J. S. Denker, ed., American Institute of Physics, New York, 333–338.

Petzold, L. R. [1983], "Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations," *Siam J Sci Stat Comput* 4, 136–148.

Rall, W. [1962], "Theory of physiological properties of dendrites," *Ann NY Acad Sci* 96, 1071–1092.

Rall, W. [1977], "Core conductor theory and cable properties of neurons," in *Handbook of physiology: the nervous system, Vol 1*, E.R. Kandel, J.M. Brookhardt & V.B. Mountcastle, eds., Williams and Wilkins, Baltimore, Maryland, 39–98.

Rall, W. [1989], "Cable Theory for Dendritic Neurons," in *Methods in Neuronal Modeling*, C. Koch & I. Segev, eds., MIT Press, Cambridge, USA.

Ran, A. & Karjalainen, M. [1991], "High level object-oriented programming environment for efficient neural computations," in *Artificial Neural Networks*, T. Kohonen, K. Mäkisara, O. Simula & J. Kangas, eds., Elsevier, North-Holland, 597–602.

Richmond, B. J. & Optican, L. M. [1987], "Temporal encoding of two-dimensional patterns by single units in primate inferior temporal cortex. II. Quantification of response waveform," *J Neurophysiol* 57, 147–161.

Richmond, B. J., Optican, L. M., Podell, M. & Spitzer, H. [1987], "Temporal encoding of two-dimensional patterns by single units in primate inferior temporal cortex. I. Response characteristics," *J Neurophysiol* 57, 132–146.

Rioul, O. & Vetterli, M. [1991], "Wavelets and signal processing," *IEEE Signal Processing Magazine* October, 14–38.

Rosenberg, J. R., Murray-Smith, D. J. & Rigas, A. [1982], "An introduction to the applications of system identification techniques to elements of the neuromuscular system," *Trans Inst M C* 4, 187–201.

Ross, W.N., Arechiga, H. & Nicholls, J.G. [1987], "Optical recording of calcium and voltage transients following impulses in cell bodies and processes of identified neurons in culture," *J Neurosci* 7, 3877–3887.

Rubner, J. & Schulten, K. [1990], "Development of Feature Detectors by Self-Organization," *Biological Cybernetics* 62, 193–199.

Rubner, J. & Tavan, P. [1989], "A Self-Organizing Network for Principal-Component Analysis," *Europhysics Letters* 10, 693–698.

Sanger, T. D. [1989], "Optimal Unsupervised Learning in a Single-Layer Linear Feed-forward Neural Network," *Neural Networks* 2, 459–473.

Segev, I., Fleshman, J. W. & Burke, R. E. [1989], "Compartmental models of complex neurons," in *Methods in Neuronal Modeling*, C. Koch & I. Segev, eds., MIT Press, Cambridge, USA.

Sharp, A. A., ONeil, M. B., Abbott, L. F. & Marder, E. [1993a], "The dynamic clamp — A new approach for understanding the role of ionic conductances in neural network regulation," *Biophysical J* 64, 103.

Sharp, A. A., ONeil, M. B., Abbott, L. F. & Marder, E. [1993b], "Dynamic clamp — Computer-generated conductances in real neurons," *J Neurophys* 69, 992–995.

Sheperd, G. M. [1992], "Canonical neurons and their computational organization," in *Single neuron computation*, T. McKenna, J. Davis & S. F. Zornetzer, eds., Academic Press, San Diego, 27–60.

Silverman, B. W. [1986], *Density estimation for statistics and data analysis*, Chapman and Hall, London and New York.

Srinivasan, M. V. & Bernard, G. D. [1976], "A proposed mechanism for multiplication of neural signals," *Biological Cybernetics* 21, 227–236.

Wang, X.-J. & Rinzel, J. [1992], "Alternating and synchronous rhythms in reciprocally inhibitory model neurons," *Neural Comp.*

Wilkinson, J. H. & Reinsch, C. [1971], *Handbook for Automatic Computation, Vol II: Linear Algebra*, Springer, Berlin and New York.

Wilson, M. A. & Bower, J. M. [1989], "The simulation of large-scale neural networks," in *Methods in Neuronal Modeling*, C. Koch & I. Segev, eds., MIT Press, Cambridge, USA.

Yuille, A. L., Kammen, D. M. & Cohen, D. S. [1989], "Quadrature and the Development of Orientation Selective Cortical Cells by Hebb Rules," *Biological Cybernetics* 61, 183–194.