An event-driven approach to biologically realistic simulation of neural aggregates

Enric T. Claverol

A thesis submitted for the degree of
Doctor of Philosophy

September 2000

An event-driven approach to biologically realistic simulation of neural aggregates

by

Enric T. Claverol

A thesis submitted for the degree of

Doctor of Philosophy

University of Southampton

September, 2000

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

An event-driven approach to biologically realistic simulation of neural aggregates

by
Enric T. Claverol

Biophysical simulation of neuronal aggregates typically utilizes analogue descriptions of the spatio-temporal dynamics of the membrane voltage in neurons. While this approach constitutes a convenient framework for realistic modelling of single neurons or small neural aggregates, the computational cost involved in the solution of the associated systems of non-linear differential equations has hampered its use in large scale simulations.

This thesis explores an emerging alternative to biophysical modelling which exploits the spike-based nature of inter-neuronal communication to replace the continuous simulation framework by a computationally more efficient event-driven technique.

A hierarchical finite state automaton neuron model suitable for message-based event-driven simulation (the MBED model) is described and discussed. It encapsulates various aspects of neuronal biophysics: synaptic/axonal latency, finite synapse activation duration, single spike and bursting behaviour, pace making, inhibition driven burst truncation and others.

The message-based event-driven simulator is designed to deliver efficient simulation of large aggregates of MBED neurons, incorporating a customized event queue management algorithm and a strategy for memory-efficient storage of synaptic parameter sets.

Two biological neural systems are tackled utilizing the MBED framework; the locomotory neural circuit of the nematode *C. elegans* and the mammalian olfactory cortex. The MBED model of the *C. elegans* locomotory system replicates experimental observations of normal, mutant and laser ablated animals and provides a *quantitative* description of a rich set of locomotory behaviours. Video recordings of active *C. elegans* behaviour, an automated image analysis system and a mechanical body model were developed to complement the neuronal simulation.

To further assess the validity of the MBED framework in biological simulations of neuronal aggregates, a model of the olfactory cortex incorporating 10^5 neurons of three cortical cell classes was developed. The model consistently replicates results obtained experimentally and with the less efficient compartmental technique, while retaining the computational efficiency inherent to event-driven simulation. The typical speed differential between the two techniques is a factor in the range 10-100. The response of the model to shock and random stimuli of various intensities is studied and shown to be in good agreement with previous results.

Finally, preliminary data on the scalability of the MBED framework utilizing Beowulf clusters is presented and further work is discussed.

ACKNOWLEDGEMENTS

I thank my supervisors Andrew Brown and John Chad for their support and advice.

I also thank my friends and colleagues Robert Cannon, Chris Franks, Darrel Pemberton, Daniel Milton, Zaher Baidas, Alan Williams, Simon Lowe, Ea, Panagiotis Melas, Andrew Perkins and others. They have all contributed to widen my knowledge of neuroscience and engineering but, more importantly, they have spiced three years of my life with their friendship.

Finally, I thank my wife, Maria Teresa, for her encouragement and my parents for their unconditional support.

Financial support for this project was provided by the Biotechnology and Biological Sciences Research Council (BBSRC) and the Dept. of Electronics and Computer Sciences (ECS), University of Southampton.

The Beowulf cluster was kindly made available by the Parallel and Distributed Computing Group of the ECS Dept., University of Southampton.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and structure of the thesis

In the last few decades, neuroscience has succeeded in providing explanations for a number of processes involved in information processing in the nervous system. Since the pioneering work of the Spanish scientist Ramón y Cajal, who described the nervous system as a network of cooperating neurons propagating information from dendrites to cell body and along the axon [1], intense research has been carried out aiming at increasing our knowledge of neural functions.

Much progress has been made at *the single cell level*, mainly as a result of increasingly sophisticated experimental techniques. In particular, the development of electrophysiology has allowed the characterization of the electrical properties of cellular membranes [2]. Less invasive imaging techniques (e.g. using voltage sensitive dyes) are also emerging as alternative methods to record neural activity [3].

Despite these advances, progress in the understanding of the cooperative behaviour of neurons at *the network level* has been hampered by the difficulties associated with the recording of activity from large numbers of cells for long periods of time. Several methods with potential to tackle this problem are gradually being developed: especially promising are multielectrode arrays, which aim at extending electrophysiological techniques to multicell recording [4], and photodiode arrays and high temporal resolution CCD imaging, which build on current optical techniques to allow imaging of activity in large aggregates of cells [5].

Simulation has emerged, concomitantly with the increase in computer power, as another useful tool for the neuroscientist [6]. The development of models of ion channels, dendrites, axons and synaptic communication between cells, has paved the

way to the construction of realistic models of single neurons [7] and aggregates of cells [8]. The availability of these models has facilitated the testing of hypotheses, while minimizing the amount of experimental data required, and has directed the design of new experiments for the validation of model predictions.

As the problem of network behaviour remains unsolved, there is an increasing need for techniques capable of simulating large aggregates of neurons. The brute force approach, the extension of classical models used for single cell simulation to large scale networks without fundamental changes in their design, has proved an arduous task due to the computational power required and the vast amount of experimental data needed to set model parameters.

This thesis aims at developing a framework where the simulation of large networks of neurons (in the order of 10^5 cells) is feasible with commodity computational resources while retaining the fundamental properties needed for realistic network activity. This goal is pursued with the development of the MBED (**M**essage-**B**ased **E**vent-**D**riven) neuron model, in an attempt to bridge the gap between classical biophysical models and oversimplified artificial neural network models. In providing a model with this common ground, some of the benefits of discrete abstract models (efficiency) and their analogue counterparts (direct mapping of biophysical parameters into the model) are retained.

In Chapter 2, background information on modelling of the nervous system is provided. Firstly, a description of the two main frameworks in computer simulation, continuous and discrete, is presented, highlighting issues relevant to neural simulation. Secondly, the levels at which models can be constructed (molecular kinetics, ion channels, single compartment neurons, multicompartment anatomically realistic neurons, small networks and large scale networks) are described. Network simulation is identified as the target level for this thesis.

Chapter 3 reviews neuron models and simulation tools from the perspective of network simulation. The selection of a particular type of neuron model affects dramatically the efficiency of a large scale network simulation. For this reason, model types are reviewed progressing from biophysically realistic compartmental models to highly abstract representations of neurons. Platforms available for the simulation of networks of these models are described.

Chapter 4 reviews previous work on modelling of the two biological systems which are studied in this thesis, the nematode *C. elegans* and the olfactory cortex of mammals. After providing background information on the invertebrate *C. elegans* and describing the experimental data available, previous work on computer models

is discussed. The lack of a quantitative model of the locomotory neural circuit which generates the observed patterns of muscle contraction is identified.

Similarly, the piriform cortex is first presented in the context of the biological olfactory system. Experimental data, including anatomical information and some activity recordings, are presented. Finally, network models of the olfactory cortex are reviewed. Two standing issues are identified as aspects to tackle with the MBED framework: firstly, the limitation of existing biophysical models to networks in the order of a few thousand neurons, when using the compartmental techniques, and secondly, the unsuitability of loosely biologically constrained models to replicate experimental data.

Chapter 5 presents the MBED neuron model which enhances oversimplified neuron models by including components which allow the direct mapping of several biophysical parameters (dendritic delay, axonal delay, synaptic latencies, finite synaptic activation duration, bursting, pace making) while retaining the computational efficiency of event-driven simulation.

Chapter 6 describes the MBED simulator. This is an object-oriented event-driven simulator implemented in the C++ programming language, integrated to a numerical package and optimized for the simulation of large networks of MBED models.

Chapter 7 extends the single cell simulations carried out in Chapter 5 to small size networks. In particular, the MBED concept is applied to the locomotory circuit of the nematode *C. elegans*. A network model is constructed which succeeds in replicating experimentally obtained patterns of muscle activation. The design of the MBED network model is complemented by a mechanical model of *C. elegans* and the use of an image processing algorithm developed with Matlab to analyse recordings of the behaving animal.

Chapter 8 extends the work presented in Chapters 5, 6 and 7 to large networks. A biologically constrained large scale MBED model of the olfactory (piriform) cortex is presented and validated by comparison with previous work carried out with classical models and with experimental data.

Chapter 9 explores the scalability of the MBED framework using Beowulf clusters of commodity computers. Experiments are carried out with an 8-node Beowulf in order to study the effects of the inter-process communication overhead on the performance of a distributed cortical simulation.

In Chapter 10, future work is proposed. In particular, enhancements to the MBED neuron model, its application to support emerging neural recording

technologies and the development of multimodule cortical models are discussed.

A number of appendices provide additional information. Appendix A contains mathematical material complementing Chapter 3. Appendices B and C contain the video recordings of *C. elegans* and a detailed description of the image processing algorithm developed to analyse them, respectively. Appendix D provides copies of papers based on this work.

Chapter 2

Background

The second half of the 20th century has seen an explosion in the amount of research aiming at a quantitative description of neural processes. This trend has been motivated by the maturation of the experimental techniques, in particular electrophysiological methods, which allowed for the first time direct recording of neuronal activity. During the 19th century and first decades of the 20th, neuroscience had been circumscribed by the limits of anatomy, where functional properties of neurons had to be inferred from the patterns of their anatomically observable features [1].

The increasing wealth of experimental data available in the decade of the 50s, sparked an interest in providing more quantitative descriptions of neurobiological phenomena by developing models of neural function. This trend was accentuated by the increasing availability of computers.

The increasing power of computing resources has made simulation a common tool for hypothesis validation in science and for system design in engineering. With an estimated 10^{11} neurons and 10^{14} synapses, the simulation of the human brain is a huge challenge, both algorithmically and computationally. Active research is underway to achieve functional models of brain modules whose simulation is feasible with the currently available technology.

Before delving into the details of the current applications of simulation in neuroscience, it is relevant to discuss the main simulation frameworks within which models are constructed.

2.1 Continuous vs discrete simulation

A general question facing the modeller in most simulation problems is the selection of the adequate level of abstraction for a particular system. The first watershed is the choice between analogue models, requiring continuous simulation, and discrete models, suitable for discrete simulation.

The distinction between continuous and discrete simulation lies in the nature of the way in which states change in the model throughout time. In continuous simulation, the granularity of time is typically several orders of magnitude smaller than the scale of the information in the system. For instance, in a continuous framework, an action potential, with a duration of approximately 10 ms, requires time steps of the order of 100 μs. Moreover, all components in the system are updated at each time step.

On the other hand, discrete simulation involves the identification of atomic information units or events. For example, if the information carried by an action potential is assumed to be captured by a pulse, the time course of the spike can be disregarded and the onset and falling edges of the equivalent pulse become the only significant events in the system. The update of the states of the components in the model is triggered by the occurrence of these events and can be restricted to those elements directly affected by them.

The different nature of the relationship between model and time leads to different frameworks of simulation and model specification methods. It also has implications in terms of efficiency which are relevant when aiming at the simulation of large aggregates of neurons.

2.1.1 Continuous simulation

Generalities

In continuous simulation, models are often specified as differential and algebraic equations. In the context of engineering, a block-oriented description of the system is common (e.g. a cascade of filters). A block is an entity characterized by its inputs, outputs and the mathematical relationships between them.

Models which describe physical phenomena do not possess such degree of modularity (e.g. diffussion processes). However, when the model attempts to capture the dynamics of a complex and physically heterogeneous object (e.g. an anatomically complex neuron) the physical entity may be represented by a model

constructed by repetition of a number of submodels which interact (e.g. chemical synapses, dendrites, axons, electrical synapses and so on). A certain degree of modularity arises from this approach.

Continuous simulation is characterized by the continuous update of the state variables in the model. As an example, consider the following partial differential equation,

$$\frac{d^2V(x,t)}{dx^2} = \alpha V(x,t) + \beta \frac{dV(x,t)}{dt} \tag{2.1}$$

This is the general form of the cable equation for passive dendrites and axons. Its application is described in Chapter 3. For the moment, it is a convenient case to illustrate the techniques of continuous simulation.

The solution of this equation is found by spatial and temporal discretization of the partial differential equation (PDE)[9],

$$\frac{V(x_{j+1},t_i) - V(x_j,t_i)}{\Delta_x^2} - \frac{V(x_j,t_i) - V(x_{j-1},t_i)}{\Delta_x^2} = \alpha V(x_j,t_i) + \beta \frac{V(x_j,t_i) - V(x_j,t_{i-1})}{\Delta_t}$$

Discretization turns the PDE into a set of algebraic relationships between a finite number of variables (in the previous example, voltages at different points in a one dimensional cable and at different points in time). Figure 2.1 shows the general algorithm involved in finding a solution for such a discretized PDE.

Note that all the voltages along the cable are updated at each time step (continuous change of state in the model). There are several numerical integration methods to realize each update. They are often classified in single step or multiple integration methods. Another classification distinguishes between implicit and explicit. The selection between methods is driven by considerations of stability and efficiency[10].

Efficiency issues

Of especial interest for the problem of simulation of aggregates of neurons is the issue of speed of numerical integration. Most commonly, the linear PDE in Equation 2.1 will be substituted by a more realistic non-linear version,

$$\frac{d^2V(x,t)}{dx^2} = \alpha I(x,t) + \beta \frac{dV(x,t)}{dt}$$

where $I(x,t)$ depends non-linearly on $V(x,t)$. The anatomical complexity of

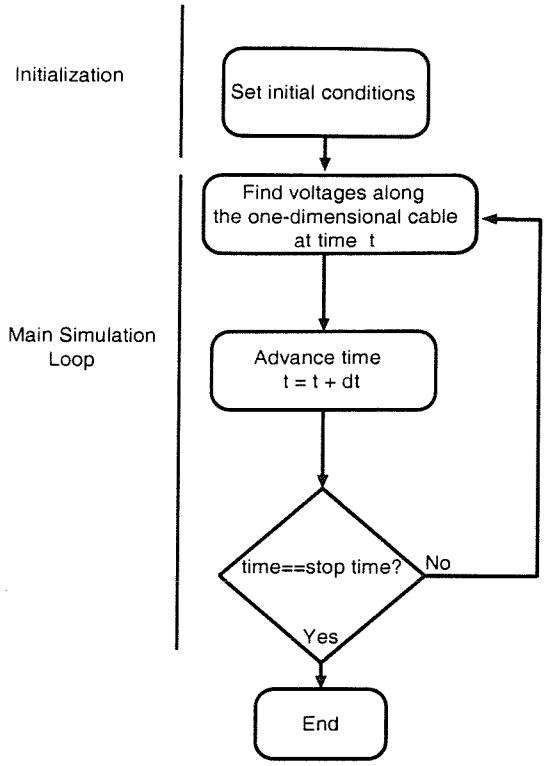


Figure 2.1: Continuous simulation execution flow

most neurons requires the use of multiple non-linear cable equations, which, after discretization, lead to a system of coupled non-linear differential equations.

Several techniques, especially designed for the problem of neural simulation, exist to reduce the computational cost of solving these equations. By exploiting the branched structure of dendrites and by casting the cable equations in a linear form, each time step requires O(N) arithmetic operations, N being the number of points obtained after discretization of the PDEs [9, 11]. Hence, the total number of arithmetic operations required for a simulation is O(NT) where T is the total number of time steps.

As an example, consider the continuous model of an hippocampal cell developed by Traub *et al.* [12]. The simulation of a network of 10000 of these neurons would require the numerical solution of approximately 250000 coupled differential equations [13]. Simulations of networks of hundreds of continuous neuron models are feasible. However, elapsed CPU times in the order of hours are typically required for simulation times in the order of hundreds of milliseconds.

2.1.2 Discrete simulation

Generalities

Traditionally, discrete simulation has been utilized in areas as diverse as telecommunications, operational science, and digital circuits. In general, discrete simulation deals with a class of problems known as *queueing* problems. These are mainly concerned with some of the following: the delays incurred by *entities* propagating through a system, the transformations applied to the *attributes* of the entities as they propagate and the occupancy of *resources* through which entities propagate.

In contrast with continuous simulation, the state of the system in discrete simulation does not change at all time steps. On the contrary, state changes only happen as a consequence of the displacement of an *entity* in the model environment, an *event*. No update of the state of the system is needed in between two consecutive events. This leads to the *discrete simulation* framework where the model evolves through time with *discrete* jumps triggered by *events*.

A fundamental concept associated with discrete simulation is the idea of the *event queue*. The displacement of entities in the discrete model happens with a certain delay. The dynamic behaviour of the model can be thought of as entities departing from certain points in the model and arriving to their destinations after a delay. The arrival would constitute an *event* and would trigger the simulation *time advance* and an update of the state of the system. The *event queue* holds a list of events sorted by time of arrival to their destinations in the system. Time advance is achieved by the extraction of the first event in the queue and its introduction back to the system. The *entities* undergoing displacements within the system, may carry associated attributes which affect the way in which the system interprets an event originated by this *entity*.

Figure 2.2 illustrates the concept of discrete simulation with a classical example taken from the problem of packet relay in a data communications network. *Entities* in this system are either data packets, which must be transmitted from an origin node to a destination node through a communication link, or notification entities. The attribute of a data packet entity is a single flag indicating its priority (H, high or L, low). Communication links and relay nodes introduce delays, labelled as d_x and D_x respectively. The origin node generates packets labelled with their priority level. The arrival of a packet to a free node changes its state to busy. The arrival of a packet to a busy node leads to the packet being discarded if its priority is lower

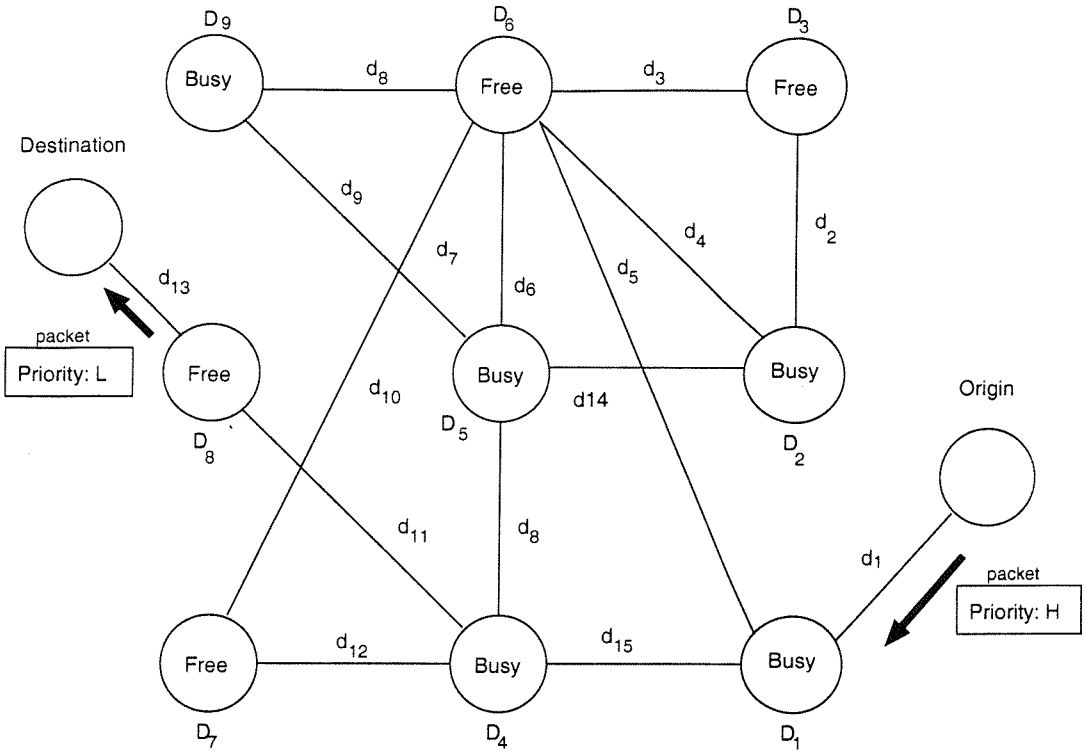


Figure 2.2: Packet communications network as a typical discrete system

than the packet being processed in the node. Otherwise, the packet in the busy node is discarded and the new packet takes its place.

Notification tokens are scheduled by nodes in order to introduce a delay between the acceptance of a new packet and its broadcasting to the following node in its route. In a typical sequence of events, a packet arrives at a free node at $t = t_0$. The node accepts the packet, changes its state to busy and schedules a notification token for $t = t_0 + D_x$, where D_x is the delay involved in the processing of a packet in a node. The notification token is inserted into the event queue. As time advances, this entity approaches the head of the queue. When it finally occupies the first position in the queue, it is popped out and delivered to the node which, upon the occurrence of this event, retransmits the packet and changes its state back to free.

The event queue stores the entities which have been scheduled for delivery to their destinations at a point in the future.

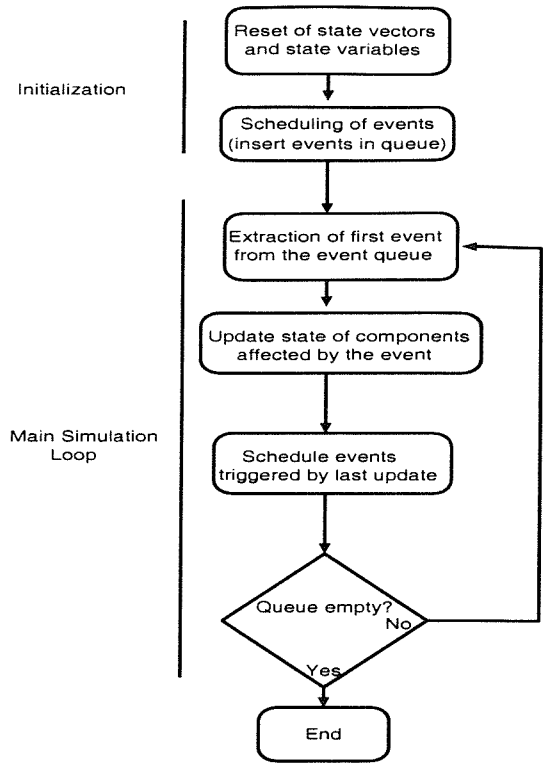


Figure 2.3: Discrete simulation execution flow

Efficiency issues: selective trace

Figure 2.3 shows a diagram of the execution flow in a typical discrete simulator.

As opposed to continuous simulation, the updating of states in the model is driven by the events scheduled and held in the queue until processing. In the example of a communications network, only those nodes which receive an event may change state (may become busy or free). Evaluation of the states of the rest of the nodes is superfluous as only those targeted by an event may need an update. This approach, based on following the events to determine which components must be updated (known as *selective trace)* minimizes the computation carried out by the simulator [14].

The fact that discrete models are often constructed at a higher level of abstraction than analogue models (implying fewer arithmetical operations in each state update) and that *selective trace* can be used (which reduces the number of updates), makes discrete simulation a more efficient framework than continuous simulation in those problems where equivalent discrete and continuous models exist.

This thesis applies the concepts of discrete simulation to the problem of realistic

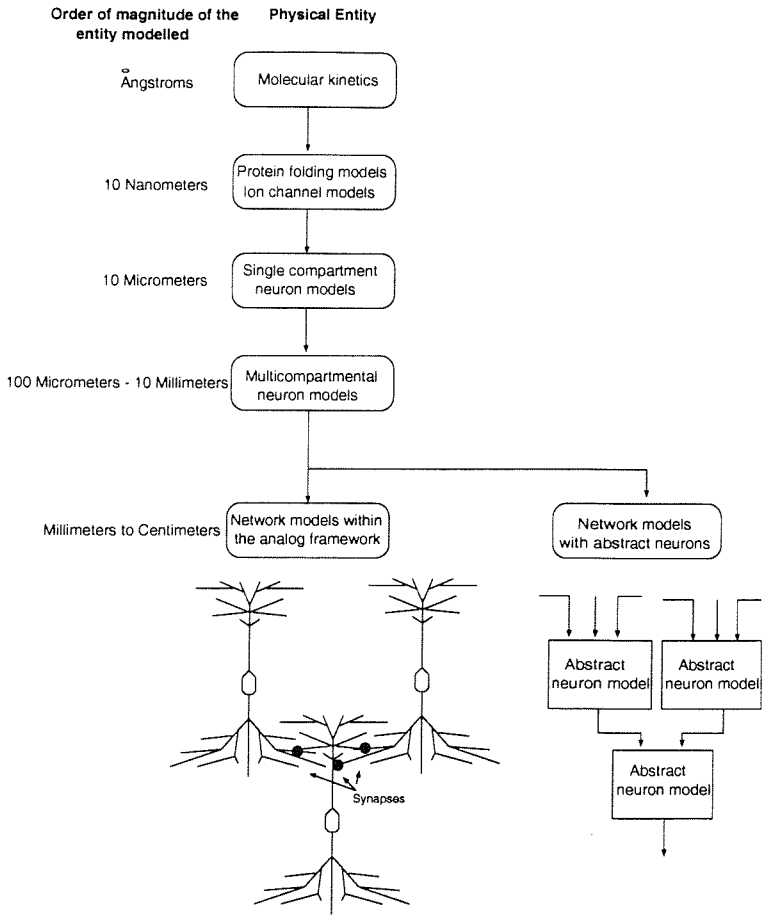


Figure 2.4: Modelling levels in neuroscience

neural simulation with the goal of increasing the efficiency of existing techniques and achieving a simplification of the parameter space. Selective trace will play a pivotal role in the efficiency increase.

2.2 Modelling in neuroscience: an overview

Models of relevance in neuroscience have been constructed ranging from the molecular level (e.g. molecule-molecule reaction kinetics models) to the neuron network level (e.g. networks with thousands of biophysically realistic neuron models). Figure 2.4 shows a diagram with several modelling levels and an approximate value for the dimensions of the physical entity modelled.

Two main approaches must be distinguished; biophysical models and abstract models. Biophysical models are based on physical descriptions of biological processes

and there is a direct mapping between the entities in the model and their physical counterparts. For instance, at the neural network level, biophysical models incorporate physical processes like ion diffusion through cellular membranes, intra-cellular ionic currents, effective capacity of the cellular membrane and so on [6]. This is the natural framework for electrophysiologists to develop quantitative descriptions based on experimental results. In figure 2.4, the biophysical approach to modelling is represented by the main trunk in the flow chart.

The rightmost branch below the bifurcation in figure 2.4 corresponds to an alternative approach to network modelling, based on highly simplified neuron models. Constraints imposed on the realism of the models are relaxed when the assumption is made that neural dynamics arise from the cooperation of functionally simple neurons. The direct use of electrophysiological data in the model is not an issue, rather, experimental results are abstracted to construct highly idealized models. Emphasis shifts, with respect to biophysical modelling, towards developing theories of neural population dynamics. This is an ideal framework to be used by engineers (targeting the construction of biologically inspired systems) and mathematicians (aiming at the discovery of mathematical laws underlying neuronal function).

2.2.1 Subnetwork biophysical models

Subcellular modelling

Neurobiological processes can be described in terms of interactions involving proteins, non-protein molecules and ions. Mathematical descriptions in this context, usually make use of a *kinetic* formalism. For instance, communication between neurons can be accomplished by means of a cascade of events originated in a presynaptic neuron which culminates in changes in a postsynaptic neuron. Each event involves the interaction between different molecules or ions. Figure 2.5 shows a schematic representation of a *kinetic* model of synaptic communication (modified from [15]). In particular, Calcium (Ca^{2+}) interacts with protein Xa to render it active. Xb, the active form of protein Xa, interacts with vesicles loaded with neurotransmitter (labelled Na) to trigger its release. The freed form of the neurotransmitter (Nb) interacts with receptors in the postsynaptic cell and triggers the transformation of protein Ga into its activated form Gb. This, in turn, will trigger further reactions.

Such detailed description of neural processes, involves the solution of systems of

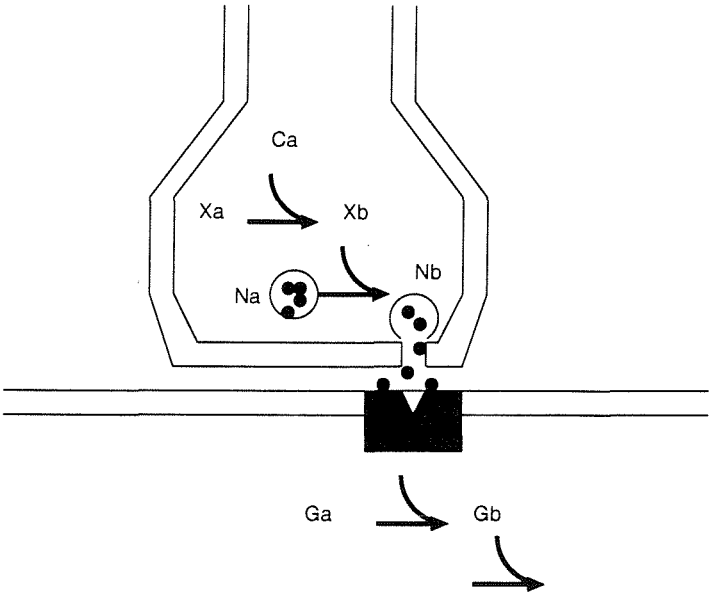


Figure 2.5: Kinetic description of synaptic transmission

differential equations where the time evolution of the concentrations of each participant in the cascade of events (e.g. Xa, Xb, Na, Nb, Ga, Gb in the example above) constitutes the solution of the equation.

Single cell modelling

Although, in principle, complex neuron level functions can be modelled considering all the molecular interactions involved, models of single neurons are often constructed with coarser granularity. A neuron is represented by a hierarchical structure where each component constitutes a model of a portion of the biological neuron. Components interact in a realistic way, by means of ionic currents flowing within the cell. Figure 2.6 shows a schematic model of a pyramidal cell.

Axons and dendrites have been segmented and each segment (known as a compartment) modelled and connected to other blocks following the anatomy of the real cell. The equations describing the dynamics of each compartment are typically systems of non-linear differential equations which make this model suitable for simulation within a continuous framework.

Single cell models of multiple neuron classes are available ranging from a thousand compartments (1600 in the Purkinje neuron model by DeSchutter *et al.* [16]) to a single compartment (see for instance the model of thalamic neurons used

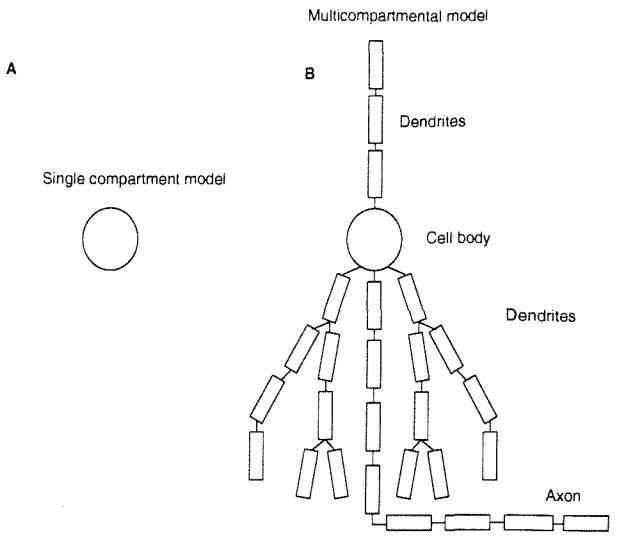


Figure 2.6: Single (A) and multicompartment (B) neuron models

in [17]). Models of intermediate size are used when anatomical characteristics are relevant but the computational cost must be minimized (for instance, Bower *et al.* developed a model of a pyramidal cell in the piriform cortex including 15 compartments [6]).

2.2.2 Network models: competing approaches

When the aim of the modelling effort is to construct system level models of the nervous system, the model is often viewed as a network of entities (neurons) interconnected through communication channels (synapses and gap junctions). Two main competing approaches have been taken; biophysical and abstract.

The biophysical approach extends the techniques used in biophysical subnetwork models to network level simulation. Realistic single cell models are interconnected by biophysical synapses and gap junctions and simulated within the framework of continuous simulation. Networks in the order of a few thousand neurons are attainable in single processor architectures. However, simplification of the single cell models making up the network is often required. Further increases in network size require parallelization of the simulation tools using clusters of workstations or massively parallel architectures.

An alternative approach is the substitution of the computationally demanding single cell models by more abstract representations which capture the main characteristics of neuron function while offering improved efficiency. Chapter 3

reviews specific neuron models belonging to biophysical and abstract techniques as well as alternatives in the interface between these two approaches.

2.3 Standing problems at the network level

Biophysical models have succeeded in providing a quantitative description of a number of processes in neurons; the generation of action potentials, the effect of different types of ion channels on the shape of the action potential, the effect of synapses on neuronal function, the regular firing of neurons involved in the creation of locomotor patterns and many others [7].

Despite these advances, the cooperation of neurons in a large network, in order to accomplish useful tasks, is still poorly understood. This is the area where more simplistic models were expected to provide the necessary insight. It is so because their simplicity reduces the number of parameters to be specified in the model and because the low amount of computation per neuron allows the simulation of large aggregates of neurons. Biophysical models are limited to networks of thousands of neurons on general purpose computer architectures. Highly parallel hardware is necessary for realistic simulation of tens to hundreds of thousands [18].

Simplified neuron models, however, have found little use in neuroscience due to their level of abstraction. The difficulties in mapping experimental data to these type of models have motivated their rejection by biologically orientated modellers.

2.4 Summary

Simulation techniques can be classified into continuous and discrete. The nature of the two approaches is fundamentally different. Continuous simulation requires the update of the state variables of the system at each time step, whereas discrete simulation is applicable to systems where the state evolves in discrete jumps, where updates are not required in between two consecutive events.

In the context of realistic simulation of the nervous system, models have been proposed ranging from the molecular level up to the level of large aggregates of neurons. Biophysical models have mainly made use of continuous simulation. Abstract models, however, are designed for mathematical tractability rather than realism, and are often suitable for discrete simulation. Their computational efficiency often comes at the expense of a lack of direct mapping between model

parameters and biophysical variables.

Chapter 3

Simulation of aggregates of neurons

In this Chapter, previous work on biologically motivated simulation of aggregates of neurons is reviewed. Firstly, background information on neuron models is provided. Secondly, representative tools available for the simulation of these models are described. Approaches based on general purpose single-processor architectures and special purpose parallel-architectures are compared.

Finally, the rationale for the selection of the message-based event-driven simulation framework on commodity architectures, as the technique to achieve low-cost efficient simulation of large scale neuronal aggregates, is discussed.

3.1 Neuron models

A wide range of models of individual neurons have been developed to serve as the atomic entity in neural network simulations. They are often classified with regard to their degree of biophysical realism [19, 20]. Compartmental models, utilized when the emphasis is on replication of physiological data, constitute the biologically accurate end of the spectrum. At the other end, abstract models, of which the binary Perceptron is an example, have traditionally been used in artificial applications and are considered oversimplified for biologically meaningful simulations. The following sections survey the main strategies for neuron modelling progressing from biophysical towards more abstract models.

3.1.1 Compartmental simulation and the Hodgkin-Huxley model

The cornerstones in biophysical models were the development of the Hodgkin-Huxley (HH) model of ion channels [21] and the extension of cable theory to dendrites and axons [22, 23, 24].

In this context, a neuron is modelled as a physical entity, often with a complex branched 3D structure, which separates two conductive media, the intracellular and the external spaces with its cellular membrane. Local differences in the concentrations of several types of ions (mainly Na^+, K^+, Cl^- and Ca^{2+}) between the external and internal sites of the membrane give rise to a transmembrane voltage. This is an important physical variable in the system; the current state of understanding of neuron function is based on a quantitative description of the time-space evolution of the membrane voltage.

The membrane behaves as a leaky capacitor, storing charges and letting them flow in and out in a controlled manner. Charge movement causes changes in transmembrane voltage which propagate along dendrites and axons. Ion channels play an important role in the control of this influx and outflux of charges.

Ion channels and single compartment neurons: the Hodgkin-Huxley model

Ion channel models are relevant to network modelling because ion-selective channels confer excitability to individual neurons in the network and constrain the patterns of neuronal activity.

Ion channels have been modelled as variable conductances across the cellular membrane [7, 21]. In the linear approximation, the current flowing through the channel is given by:

$$I_{chan_A} = G_{max} X(t, V_m, Ca^{2+}, \ldots)(V_m - E_{rev}) \tag{3.1}$$

where I_{chan_A} is the total current flowing through all channels of type A , G_{max} the maximal conductance achieved when all channels of this particular type are open, X a normalized time-changing variable expressing the degree of openness ($X = 1$ for all channels of type A open, $X = 0$ for all channels of type A closed), V_m the membrane voltage and E_{rev} the potential at which the current reverses its sign.

The variable X accounts for changes in the conformation of the proteins which

make up the channels and which affect their conductances. These changes may be induced by alterations in the membrane voltage (V_m), by changes in the concentration of an ion type to which the channel is sensitive (e.g. Ca^{2+}), by the release of neurotransmitter and others. These changes of conformation may render a channel closed (unable to conduct ions) or open (able to transport ions) [25], in addition to other intermediate states.

The current through ionic channels is responsible for the dynamic properties of neurons which lead to a local and abrupt change in the transmembrane voltage, the action potential. The Hodgkin-Huxley model succeeded in quantifying this phenomenon [21].

The Hodgkin-Huxley model considered two types of ion-selective channels, voltage-dependent Na^+-selective and K^+-selective channels. For both channels, the dynamic variable $X(t, V_m, Ca^{2+}, \ldots)$ was only dependent on time and membrane voltage, $X(t, V_m)$. In the case of the Na^+ channel, $X_{Na}(t, V_m)$ was given by (time and voltage dependency are not explicitly stated for clarity),

$$X_{Na} = mh^3$$

where m and h are dynamic variables described by first order ODEs,

$$\frac{dm}{dt} = \alpha_m(V_m)(1 - m) - \beta_m(V_m)m$$

$$\frac{dh}{dt} = \alpha_h(V_m)(1 - h) - \beta_h(V_m)h$$

where $\alpha(V_m)$ and $\beta(V_m)$ are the voltage dependent opening and closing rates. An alternative form for these equations clarifies their dynamic properties,

$$\frac{dm}{dt} = \frac{m - m_\infty(V_m)}{\tau_m(V_m)}$$

$$\frac{dh}{dt} = \frac{h - h_\infty(V_m)}{\tau_h(V_m)}$$

where m_∞ and h_∞ are the voltage dependent steady state values of m and h, respectively, and $\tau_m(V_m)$ and $\tau_h(V_m)$ their voltage dependent time constant.

As a result of a synaptically driven increase of V_m from resting potential to firing

threshold, m increases with a fast time constant, originating a Na^+ current influx, charging the membrane capacitance and rising V_m. This is the fast onset of an action potential.

The m (activation) variable is responsible for an increase in Na^+ permeability leading to the charge influx. Since this increase is seen experimentally to be transitory, a second (inactivation) variable, h, is added to the model. The variable h tends to decrease with a slow time constant within the voltage range where m takes its maximum.

The combination of the fast activating variable m and the slow inactivating variable h, attributes transitory activating properties to the Na^+ channel.

The activation variable for the K^+channel is given by:

$$X_K = n^4$$

where n is described by

$$n = \alpha_n(V_m)(1 - n) - \beta_n(V_m)n$$

The variable n shares some characteristics with m. A rising membrane voltage increases n, resulting in the opening of K^+-selective channels and in a charge outflux which decreases the voltage across the membrane capacitance. The K^+ current constributes to the ending of the sharp voltage spike.

When the 3D anatomy of the neuron does not need to be captured by the model, a neuron can be modelled as a single compartment (see figure 3.1). A compartment is defined as a section of membrane separating an isopotential intracellular volume from the isopotential external cell volume. The equation describing the electrical properties of a compartment is

$$C_m \frac{dV_m(x,t)}{dt} = -G_m(V_m(x,t) - E_{leak}) \tag{3.2}$$

where C_m is the membrane capacitance, V_m the membrane voltage, G_m the membrane conductance and E_{leak} the voltage at which the leakage current reverses its sign. This expression is only valid for a passive compartment, i.e. one that only includes ion channels whose conductances are fixed, unaffected by transmembrane voltage or other physiological variables (this excludes the Na^+ and K^+ channels described above).

When active conductances, i.e. channels affected by membrane voltage, are

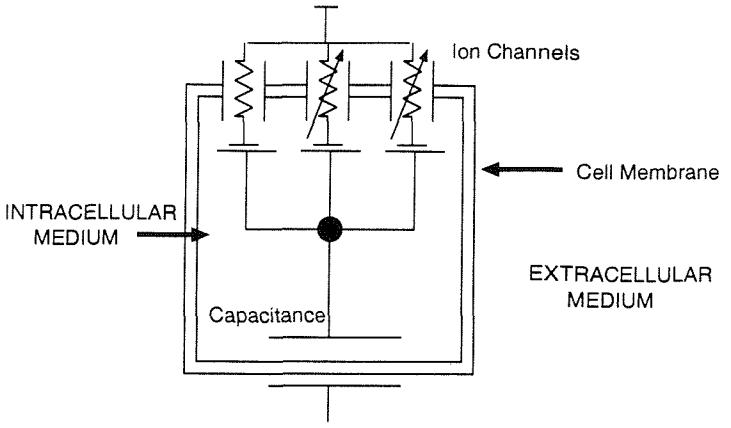


Figure 3.1: Components of a single compartment model

added, equation 3.2 becomes

$$C_m \frac{dV_m(x,t)}{dt} = -G_m(V_m(x,t) - E_{leak}) + I_{chan} \tag{3.3}$$

where I_{chan} is the total current flowing through active channels.

As the wealth of functionality implemented by a single neuron depends on the ion channels embedded in its membrane, most single compartment neuron models will not fall into the category of passive. They often include active conductances, making the magnitude of the transmembrane current dependent on membrane voltage (this is the case of voltage-dependent channels responsible for action potential generation), ion concentration (e.g. Ca^{2+}), the concentration of a particular neurotransmitter and so on.

I_{chan} typically accumulates currents through various types of ion channels,

$$C_m \frac{dV_m(x,t)}{dt} = -G_m(V_m - E_{leak}) - \sum_{j=1}^{J} G_{max_j} X_j (V_m - E_{rev_j}) + I_{inj} \tag{3.4}$$

where the summation is over J types of channels present in the compartment and I_{inj} has been added to account for current injected which an electrode.

Hodgkin and Huxley [21] demonstrated that, with the addition of Na^+ and K^+ channels to the passive single compartment model, setting I_{inj} to a positive value above a certain threshold is sufficient to trigger transient changes of membrane voltage. These replicated the axon potentials observed experimentally in the squid's

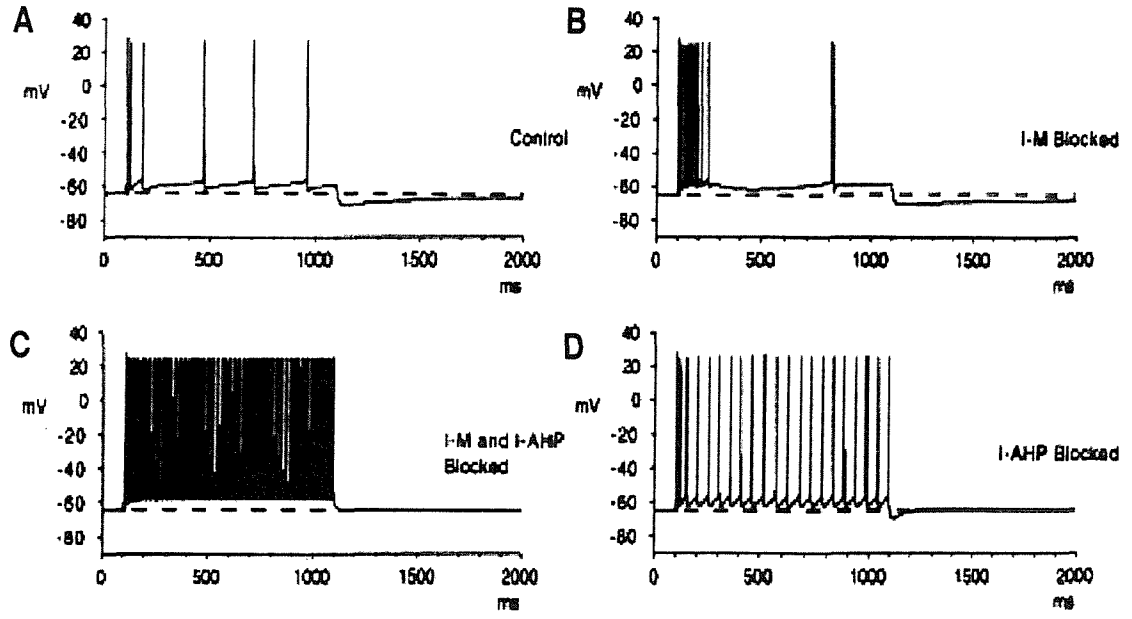


Figure 3.2: Spiking rate adaptation in an hippocampal neuron (from [26])

axon. Figure 3.2-A shows a train of action potentials obtained from a model of an hippocampal pyramidal cell by simulated current injection [26].

Single cell models incorporating several types of channels are capable of a rich set of responses to simple stimuli. Table 3.1 lists several important classes of ionic currents. Their notation (first column), ion type making up the current (second column), activation/inactivation characteristics (third column) and their effect upon neuronal activity (fourth column) are given.

Detailed models of hippocampal pyramidal cells were constructed in [26],

Current	Ion	Characteristics	Effect
I_{Na}	Na^+	Fast, transient	Onset of action potential
I_A	K^+	Transient	Spike repolarization
I_{Kd}	K^+	Non-transient	Spike repolarization
I_C	K^+	$V\&Ca^{2+}$ activated	Spike repolarization
I_M	K^+	Slow, non-transient	Spike freq. adaptation
I_{AHP}	K^+	Ca^{2+} activated	Spike freq. adaptation
$I_{T(LVA)}$	Ca^{2+}	Low threshold	Low threshold spikes
$I_{Ca(HVA)}$	Ca^{2+}	High threshold	Elongation of action pot.

Table 3.1: Several ionic currents which participate in patterning neuronal excitability

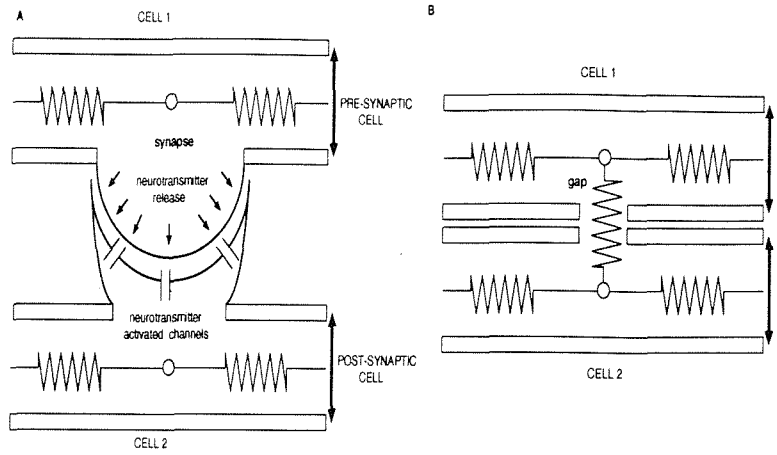


Figure 3.3: Compartmental models of chemical (A) and electrical (B) synapses

incorporating several of these channels. Figure 3.2 compares the response of a pyramidal neuron model to the injection of a 1 s current pulse with an intensity of 0.5 nA (taken from [26]). Four cases are shown, corresponding to I_M and I_{AHP} currents present (A), I_M absent (B), I_{AHP} absent (D) and both absent (C). The four responses are triggered by identical stimuli. However, the neuron shows markedly different patterns of activity as a consequence of the variations in ionic channel types.

Network models with single compartment neurons

For the creation of networks, compartmental models must incorporate synapses and/or gap junctions [27]. The former are introduced as neurotransmitter activated ion channels and the latter as resistive connections between the intracellular medium of the two cells participating in the gap junction (see figure 3.3) [15].

The neuron providing input to the synapse (presynaptic cell) releases neurotransmitter into the synaptic cleft, activating the synaptic channels in the receiving (postsynaptic) neuron. The opening of postsynaptic channels due to neurotransmitter release at $t = t_0$ leads to a transitory increase in synaptic conductance often modelled by

$$G_{syn} = \alpha t e^{-(t-t_0)} \tag{3.5}$$

where G_{syn} is the synaptic conductance and α a scaling factor related to the

synaptic efficacy. The resulting synaptic current is given by a similar expression to that of other ionic channel currents (expression 3.1) ,

$$I_{chan}(t) = G_{syn}(t)(V_m - E_{syn}) \tag{3.6}$$

where I_{syn} is the synaptic current and E_{syn} the potential at which the synaptic current reverses sign.

Single compartment neuron models and the interconnecting synapses are the building blocks upon which networks can be constructed. The computational cost of this framework is evident in, for instance, simulations of cerebellar [8] and thalamic [17] networks.

The model of the cerebellar granule cell layer developed by Maex *et al.* [8], incorporates single compartment Golgi and granule cell models. Its 30 Golgi cells receive in the order of a few thousand connections and approximately 10^4 granule cells receive 5 synapses each. The simulation ran for 18 h on a dedicated Sun UltraSparc workstation to simulate 10 s network time (6.48 CPU seconds per simulated ms).

Destexhe *et al.* [17] have constructed a network model of the thalamic reticular nucleus incorporating 100 single compartment neurons. The neuron model included the fast Na^+ and K^+current channels responsible for the generation of action potentials, in addition to the low threshold Ca^{2+} current (I_T) and the Ca^{2+}-activated $I_{K[Ca]}$ and I_{CAN} currents. The model was used to investigate the effect of network parameters on the 7-14 Hz spindle oscillations. The simulations where performed using NEURON and ran on a Sun Sparc 10 workstation. A typical simulation took 0.96 CPU seconds per simulated ms.

Cable theory of dendrites and axons. Multicompartment neuron models.

Following the generation of an action potential at a particular membrane location in the 3D structure of the cell, this local and transitory change in membrane voltage propagates along nearby branches. This process follows physical principles similar to those in classical cable theory and were successfully described by the cable theory of dendrites and axons developed by *Rall et al.* [22, 23, 24].

$$\lambda^2 \frac{d^2 V_m(x,t)}{dx^2} = (V_m(x,t) - E_{leak}) + \tau \frac{dV_m(x,t)}{dt} \tag{3.7}$$

where λ and τ are the space and time constants respectively, V_m is the membrane

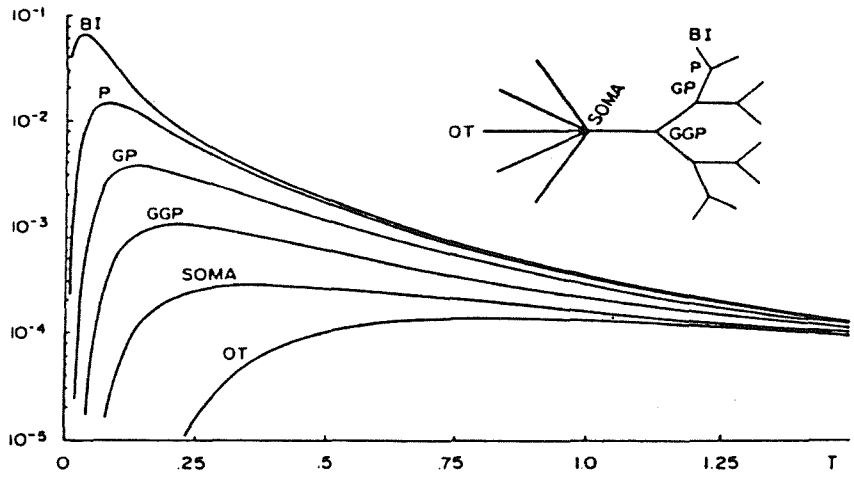


Figure 3.4: Signal attenuation in a dendritic tree (from [6])

voltage and E_{leak} is the voltage at which transmembrane leakage current ceases. This equation predicts that a change in membrane potential at one end of a dendritic terminal will propagate with a certain attenuation and velocity towards the other end.

Figure 3.4, taken from [6], shows the solution corresponding to the transient response to a current pulse injected at a single branch (labelled BI) in the neuronal dendritic tree shown in the figure. The length of each branch is $\frac{1}{4}\lambda$. Traces in the plot correspond to the voltage obtained at different points in the cell as a function of the dimensionless time variable $T = \frac{t}{\tau}$. The dendritic tree introduces a delay of 0.2 (0.2τ s) between the maximum at branch BI and the maximum seen at the soma and an attenuation by a factor of 0.5×10^3.

When the effects of the neuron anatomy must be captured by the model, the branched structure of the neuron is partitioned in isopotential compartments (see figure 2.6). Each one of these compartments is described by an equation similar to 3.4,

$$C\frac{dV_m}{dt} = -G_{max}X(V_m - E_{res}) + I_{i+1} - I_{i-1} - G_{leak}(V_m - E_{leak})$$

Note, however, two new terms, I_{i-1} and I_{i+1}. They correspond to the current drawn from the two adjacent compartments, $i+1$ and $i-1$. The magnitude of these currents is given by Ohm's law,

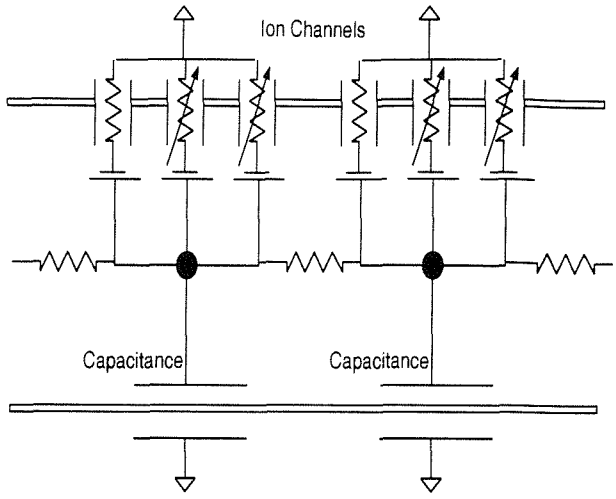


Figure 3.5: Cable model of a dendritic segment

$$I_{i-1} = \frac{(V_{m_i} - V_{m_{i-1}})}{R_a}$$

where V_{m_i} is the membrane voltage in compartment i, $V_{m_{i-1}}$ is the voltage in the adjacent compartment $i-1$ and R_a is the internal axial resistance.

Simulations with multicompartment models

The computational cost involved in the simulation of network models with multicompartmental neurons is exemplified by studies of cerebellar and cortical dynamics.

DeSchutter *et al.* [16] constructed a multicompartmental model of the cerebellar Purkinje cell with 1600 compartments. However, the computational requirements of the model made it unsuitable for network simulations; 550 ms simulated time required 1 h CPU time (6.53 CPU seconds per simulated ms).

A model of a pyramidal cell in the piriform cortex with 1089 compartments was constructed and simulated on a 200MHz PC running Linux (1.3 CPU seconds per simulated ms). With a reduced version of this model including 15 compartments, the computational requirements were reduced to 30 CPU ms per simulated ms)[6]. An even more simplified model with 5 compartments was used in a network simulation incorporating 4500 neurons requiring 100 CPU seconds per simulated ms

on a Sun 2/360 workstation [28].

3.1.2 Simplified Hodgkin-Huxley models

A reduction in the complexity of the HH model is possible by minimizing the number of dynamic variables [29]. The FitzHugh-Nagumo (FN) [30] neuron model includes two coupled variables, x (the excitation variable) and y (the inhibition variable),

$$\epsilon \frac{dx}{dt} = -y - g(x) + I$$

$$\frac{dy}{dt} = x - by$$

where $g(x) = x(x-a)(x-1)$ and a and b are parameters controlling the asymptotic or periodic response of the system and I provides the external input to the system.

Network models of oscillating neurons have been constructed to study object segmentation in the visual cortex [31] and sound recognition [32]. An FN neuron model was also used in a simulation of the olfactory bulb aiming at replicating experimentally obtained recordings [33].

Other simplifications of the HH model rely on approximations of the voltage dependence of the opening/closing rates (α and β). For instance, in [34], the commonly used sigmoidal form was substituted by a pulse function. This simplification allows an analytical solution of the channel gating equations (those describing m, h and n in the HH model). The obtained solutions are of the form

$$m(t) = A + Be^{-(t-t_0)}$$

Two sets of expressions must be found: one valid during an action potential and a second solution valid during inter-spike periods. As a result of the availability of an analytical solution, the fast changes in the rate constants during action potentials do not force a considerable reduction of the integration step in order to guarantee stability. These results in an increased efficiency of the model with respect to the original HH formalism while retaining an acceptable degree of realism [34].

3.1.3 The integrate and fire model

The integrate and fire (IF) model is a widely used further simplification of the HH formalism which exploits the invariability of the shape of action potentials. The assumption that the timing of the spike is the information carrier, rather than its shape (see for example Appendix A5 in [35]) is implicitly contained in IF models. Hence, the $Na^+$and K^+channels, responsible for the shaping of the spike, are excluded from the model. Such a simplification would lead to a complete absence of spikes, unless an alternative firing mechanism is added. This is the purpose of the threshold function incorporated into the model. An action potential is generated when the membrane voltage increases beyond a firing threshold, V_{th}, by setting V to V_{max}, the maximal voltage during a spike, for a short period of time (typically less than 1 ms) and the after-spike potential, V_{AS} (10 to 20 mV below the resting potential).

The refractory period is introduced by the time needed by the membrane capacitance to recover from V_{AS} to the resting potential V_{res}. The dynamics of the membrane voltage are described by an equation of the type

$$\tau \frac{dV}{dt} = (V_{res} - V) + I_{syn} \tag{3.8}$$

where I_{syn} is the current injected into the cell through synapses. Equation 3.8 applies only while the condition $V < V_{th}$ holds. Otherwise, if $V > V_{th}$ at $t = t_n$

$$V = \left\{ \begin{array}{ll} V_{max} & t = t_n + dt \\ V_{AS} & t = t_n + 2\ dt \end{array} \right\} \tag{3.9}$$

where dt is the integration step. Equation 3.8 holds in the interval $t \in \{t_n, t_{n+1}\}$ where t_{n+1} denotes the timing of the next spike.

Various versions of the IF model have been developed. For instance, equation 3.8 can be enriched with non-spiking conductances (such are rate adaptation K^+conductances) which modulate the effective time constant of the cell [35]. Also, to account for randomness in neuronal activity, noisy IF models have been proposed by introducing stochasticity in the generation of the spike or in the activation of synaptic current [36].

The computational cost of simulating IF models is greatly reduced with respect to compartmental models including HH channels since the integration step can take larger values. This is made possible by the absence of fast-changing action potential conductances. Multiple biologically motivated network models can be found in the

literature [37, 38, 36] which take advantage of the IF approach.

Wilson *et al.* [28] constructed a model of the piriform cortex with 4500 neurons. The neuron model was multicompartmental but the Na^+ and K^+ channels were substituted by the threshold principle used in IF neurons.

To study the variations in the patterns of activity in a generic cortical network as a result of changes in the neuronal excitation threshold and the membrane time constant, Hill *et al.* [39] used single compartment IF neurons. A network with 100×100 IF neurons, including excitatory and inhibitory cells, displayed network state transitions in certain regions of the threshold-time constant parameter space, as a result of alterations of the balance between excitatory and inhibitory synapses.

IF models have also been used in simulations of the visual system. Experimental evidence of image segmentation based on neuronal firing synchronization has motivated the creation of spiking network models which rely on this principle [40, 41, 42]. The conditions to be met by pairs and small networks of IF neurons to achieve synchronization have been studied using an event-driven simulator in [43].

In a further simplification of the IF model, specifically aimed at visual processing simulation, Thorpe *et al.* [44, 45] have developed a feedforward network with three layers of cells which sequentially propagate the activity triggered by an input image. The neurons were modelled as spiking units constrained to the generation of a single action potential throughout the entire simulation corresponding to the processing of one image. At the expense of biological realism, this simplification resulted in improved efficiency. A simulation of 100 ms of activity in a network of $7\,10^5$ neurons with $35\,10^6$ connections took 15 s of CPU time [45].

3.1.4 Biophysical continuous (non-spiking) models

In non-spiking models, the possibility of generating action potentials is removed altogether, the membrane voltage evolving smoothly over time. Biophysically motivated non-spiking models have been proposed for the study of both mammalian and invertebrate neural functions.

In [46], Wright *et al.* studied the generation of cortical electro-encephalographic (EEG) rhythms. In this work, individual neurons were not modelled, rather, entire cortical regions were represented by entities in an interconnected 20×20 lattice. Each node in the lattice had an associated continuously changing variable, $V(t)$, representing the average membrane voltage in the associated cortical volume,

$$V(t) = g \sum_{j=1}^{n} w_j Q_a(t - j\Delta t)$$

where g is the synaptic gain, $Q_a(t)$ is the density of afferent pulses, Δt is the discrete time step and w_j is the prototype synaptic response, and

$$w_j = b^2 j \Delta t e^{-bj\Delta t}$$

which is a discrete version of the alpha function in equation 3.5.

Time constants, relative magnitude of excitatory and inhibitory synapses and dendritic and axonal delays were set to realistic values. The effect of changes in network parameters (namely, dendritic delay, synaptic strength and synaptic reversal potential) on the spectral content of simulated EEG signal was explored. These studies concluded that realistic frequency components can be generated by the model and that the relative magnitude of each component is governed by the network parameters.

Wicks *et al.* [47] constructed a model of the tab-withdrawal circuitry in an invertebrate, the nematode *C. elegans*, with non-spiking neuron models. The membrane voltage for each neuron was given by

$$C_m \frac{dV_m(x,t)}{dt} = -G_m(V_m(x,t) - E_{leak}) + \sum I_{syn} + I_{ext}$$

where I_{syn} is the current due to synaptic events and I_{ext} is is the injected current. Note that the active conductances required for action potential generation were not included, allowing continuous membrane potential change in this model. The network incorporated 9 neurons and was designed following available anatomical data. The simulation aimed at determining the polarities (excitatory versus inhibitory) of the synapses in the network. For this purpose, all possible configurations were tested and likelihood values assigned.

3.1.5 Abstract models

Binary models

The binary neuron models developed by McCulloch and Pitts [48] in the 40s, constitute a landmark in abstract neuron representations. They considered the

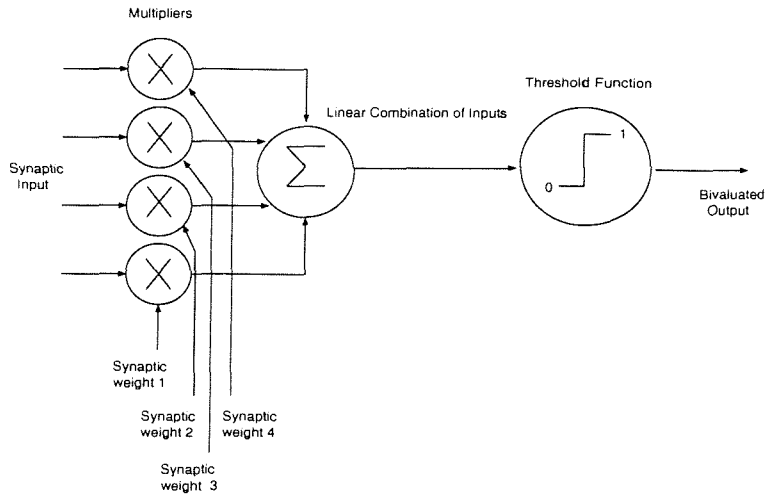


Figure 3.6: Perceptron with step function

neuron a bivaluated processing element and proposed these binary elements as the conceptual foundation upon which a formal logic of neural processing would be built.

A closely related model, the original Perceptron developed by Rosenblatt [49], is also a binary neuron abstraction with the added particularity of being suitable for pattern classification when associated to its learning rule. Figure 3.6 shows a diagram with its main blocks.

The output, o, of a Perceptron with J inputs is given by

$$o = sign(\sum_{j}^{J} w_j i_j) \tag{3.10}$$

where w_j is the synaptic weight associated to the j^{th}input, i_j is the analogue value provided by the j^{th} synapse and $sign(x)$ takes the value $+1$ if $x \geq 0$ and 0 if $x < 0$.

Similar simplified binary neurons have been used in multiple network models; for instance, Hopfield networks achieve associative memory [50] and Boltzmann machines are stochastic neural networks capable of solving certain types of combinatorial optimization problems [51].

Spiking rate models

Spiking rate models, on the other hand, assume that the relevant feature of neuronal activity is the spiking frequency and that the precise timing of individual action potentials is not fundamental to achieve functional neural activity.

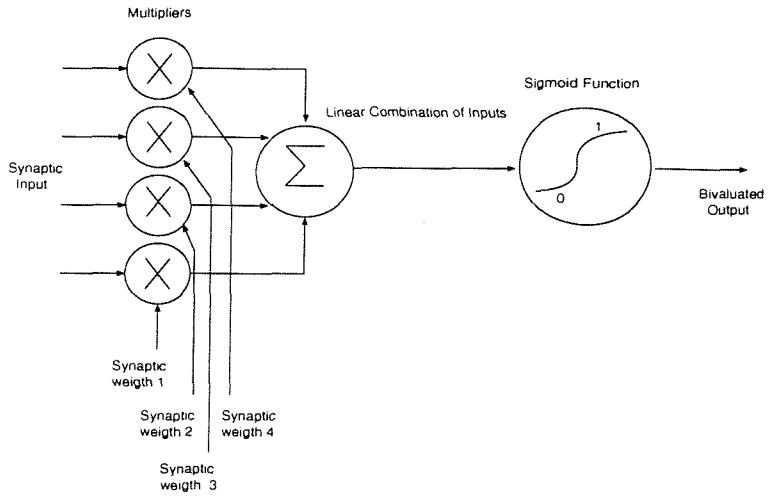


Figure 3.7: Perceptron with sigmoidal function

A modification of the binary Perceptron [52], shown in figure 3.7, is often used in this context where the output of the neuron is given by

$$o = sigm(\sum_{j}^{J} w_j i_j)$$

With respect to expression 3.10, the sign function has been substituted by $sigm(x)$, which is differentiable and monotonically increasing; typically, a sigmoid function,

$$sigm(x) = \frac{1}{1 + e^{-\lambda(x - x_0)}}$$

As a result of this change in the input-output relationship, the parallelism between the Perceptron and the biological neuron is modified with respect to binary models. Whereas, in a binary Perceptron, the activation of the neuronal output can be interpreted as a single action potential, the graded output of a continuous Perceptron lends itself to the interpretation in terms of spiking rate. The spiking rate of the neuron model is provided by the value at its output.

Multilayer feed-forward Perceptron networks (MLP) based on continuous output-input mapping were first developed and trained by Rumelhart *et al.* [53] who introduced the back-propagation learning rule. The suitability of these networks to perform classification and function approximation resulted in its wide-spread use in the field of articial neural network.

The linear Hopfield model [54], constitutes another example of rate neuron; it exhibits properties similar to its binary counterpart.

The field concerned with the use of abstract neuron models, both binary and rate based, partly to understand network behaviour in the nervous system and partly as an engineering technique suitable for the design of biologically inspired systems, is commonly referred to as *artificial neural networks* [55] and neurocomputing [56].

Cell automata

Cell automata models make use of the finite state automaton formalism to capture the functionality of a physical entity [57]. In the general case, a state vector is associated to each cell and its time evolution is determined by its state-history and the inputs received from other cells through cell-cell interactions.

Pytte *et al.* [13] have proposed a model of the CA3 hippocampal region based on this technique. It utilized a binary neuron model which undergoes a state transition, from *inactive* to *active*, as a result of incoming synaptic activity reaching a set threshold or by spontaneous firing if the time since the last activation exceeds a randomly chosen time τ_s.

For excitatory neurons the firing condition was given by

$$m_e K_e - (m_f K_f + m_s K_s) > h(n - \tau_r)$$

where m_e , m_f and m_s are the number of synaptic excitatory, fast inhibitory and slow inhibitory simultaneously active synaptic inputs. K_e , K_f and K_s denote their respective strengths. The function $h(n - \tau_r)$ corresponds to a monotonically decreasing firing threshold, taking its maximum at $n = 0$, the time of the last spike, 0 for $n > tau_r$. The firing threshold of inhibitory cells was set to a time-independent value of 0.

Upon firing, a neuron remains active for a duration equivalent to a burst of action potentials in the real neuron. Individual spikes in a burst were not modelled.

The hippocampal model included 10^4 neurons of three types; excitatory, fast inhibitory ($GABA_A$ mediated) and slow inhibitory ($GABA_B$ mediated) with an average of 38 synapses per neuron. Figure 3.8, taken from [13], shows the percentage of neurons firing in four simulations including 900 neurons where the strength of fast inhibitory synapses was progressively decreased ((a) 10.00, (b) 0.45, (c) 0.34, (d) 0.00). As a result of the decrease in total inhibition, a high percentage of the neurons in the network fired simultaneously (see (c)). A further decrease of

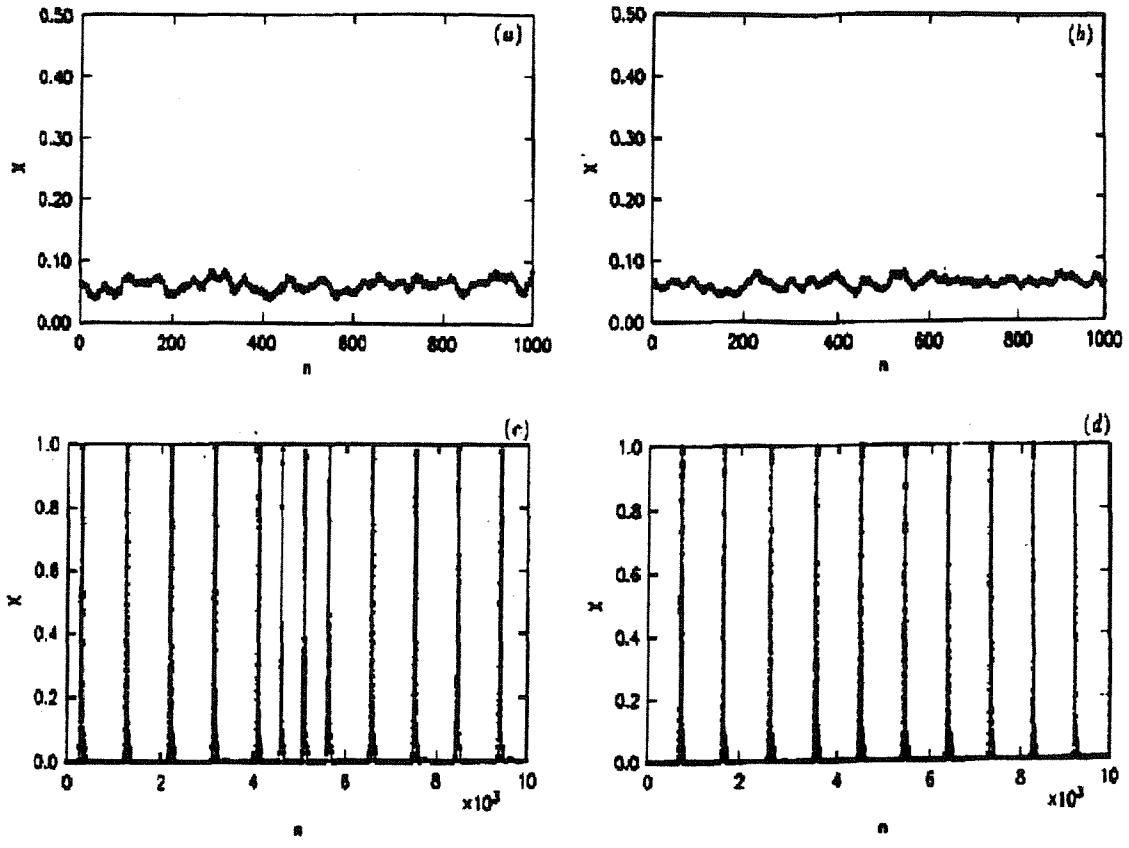


Figure 3.8: Percentage of neurons firing in a network with 900 automata neurons (from [13])

inhibition, resulted in an increase in the regularity of the population firing (see (d)).

Pytte *et al.* explicitly state that, in their implementation, neuronal states were continuously updated, rather than exploiting the discrete nature of the system by using an event queue to selectively update those neurons receiving synaptic activation (selective trace).

In another example of a cell automata network model, Axelrad, Guiraud *et al.* [58, 59] carried our simulations of the cerebellar cortex. In particular, the collateral inhibition between Purkinje cells was modelled by an array of tri-state (*silent*, *tonic* and *phasic*) automata. This study concluded that, in such a model, the spatio-temporal patterns of neuronal activity are characterized by Markov-type cyclic dynamics, where the network evolves through a number of global states; each new state depending exclusively on the previous one.

Moreover, the effects of general anaesthesia on cortical activity have been studied with an 80×80 lattice of cellular automata [60]. The total number of active

automata was taken to represent the EEG. These simulations concluded that an increase in synaptic strength induced low amplitude high frequency EEG components whereas a decrease (hypothesised to be equivalent to anaesthesia) gave rise to the appearance of high amplitude low frequency components.

Other instances of cell automata based neural simulation have been only loosely biologically motivated. Korkin *et al.* [61] and Gers [62] proposed the CoDi (collect and distribute) 1-bit model, targeted at a special purpose architecture (a cell automata machine) developed with FPGAs (Field-Programmable Gate-Arrays) capable of simulating 75 million neurons. In the simple CoDi model, communication between neurons is achieved through 1-bit buses (no concept of synaptic weight exists), where synaptic delays, synaptic activation duration times and long range connections are difficult to implement. The emphasis of the CoDi model is on efficiency and compactness for hardware implementation of genetic algorithms, rather than biological realism.

In addition to the dynamics of neural function, automata have also been proposed as adequate models for the simulation of other biological phenomena involving excitable cells.

Luthi *et al.* [63] simulated neurogenesis in Drosophila using cell automata. The model consisted of a 128×128 matrix of automata and was simulated on a parallel CM-200 (8000 processors) achieving a rate of 10^7 single automaton updates per second. Through inhibition between nearest neighbours in the matrix, a proportion of the initial pool of cells differentiated into neuroblasts while others remained undifferentiated. Quantitative results were shown to be consistent with values obtained experimentally.

Another developmental study was performed by Eddi *et al.* [64], who proposed a model for the establishment of synapses between climbing fibers and Purkinje cells during a transitory developmental phase, when the number of synapses is thought to reach a maximal level of redundancy.

Siregar *et al.* [65] proposed a model of the excitability of the heart consisting of a 3D matrix with $140 \times 120 \times 100$ automata which was able to simulate cardiac electrical activity.

3.2 Simulation platforms

Since the simulation of biophysical models is a computationally expensive task, there is active research aimed at the development of efficient simulation platforms. Both

Simulator	Type of neuron model	Simulation framework	Network Size (1 CPU)	Comments
GENESIS	Biophysical	Continuous	10^2	
NEURON	Biophysical	Continuous	10^2	
BIOSIM	Biophysical	Continuous	10^2	Teaching and research
NEOSIM	Biophysical	Mixed Mode	10^2	
NBC	Biophysical/IAF	Continuous	10^2	
SWIM	Biophysical	Continuous	10^3	Hybrid neuro-mechanical
XSIM	Multiple models	Continuous		
SPIKE/NEURALOG	Spiking neurons	Event-Driven	10^2	Not optimized for large nets
SURF-HIPPO	Biophysical	Continuous	10^2	
SYNOID	Biophysical	Continuous		
NANS	Biophysical	Continuous	10^2	GUI based
SPIEDERWEB	Multiple models	Continuous	10^2	C++ libraries
SLIM	Biophysical	Continuous	10^2	
SNNAP	Biophysical	Continuous	10^2	
NODUS	FN & IF	Continuous	10^4	
Nischwitz-Gluender	IF	Event-Driven	10^2	

Table 3.2: Software packages for biological neural simulation

single processor and parallel architectures have been exploited for this purpose [18].

3.2.1 Single processor architectures

Single processor architectures are commonly available and are the main platform for single cell and small network simulations. Table 3.2 includes some of the software tools available for network simulation with an emphasis on biological realism. The simulators GENESIS [66] and NEURON [9] are widely used in compartmental simulations, both packages offering similar features. However, NEURON provides cross-platform compatibility, running on Windows and Unix environments. Other simulators like SURF-HIPPO [67] and Biosim [68] offer similar capabilities, although their use is less wide spread. In general, these biophysical simulators are limited to networks of a few hundred neurons, reaching the thousands with reduced compartmental models. SWIM and the SPLI library [18] were also developed for biophysical simulation, however, they have been optimized for network simulations with thousands of neurons and tested on different platforms. Also, SWIM has been used in hybrid neuro-mechanical simulations of the lamprey [69].

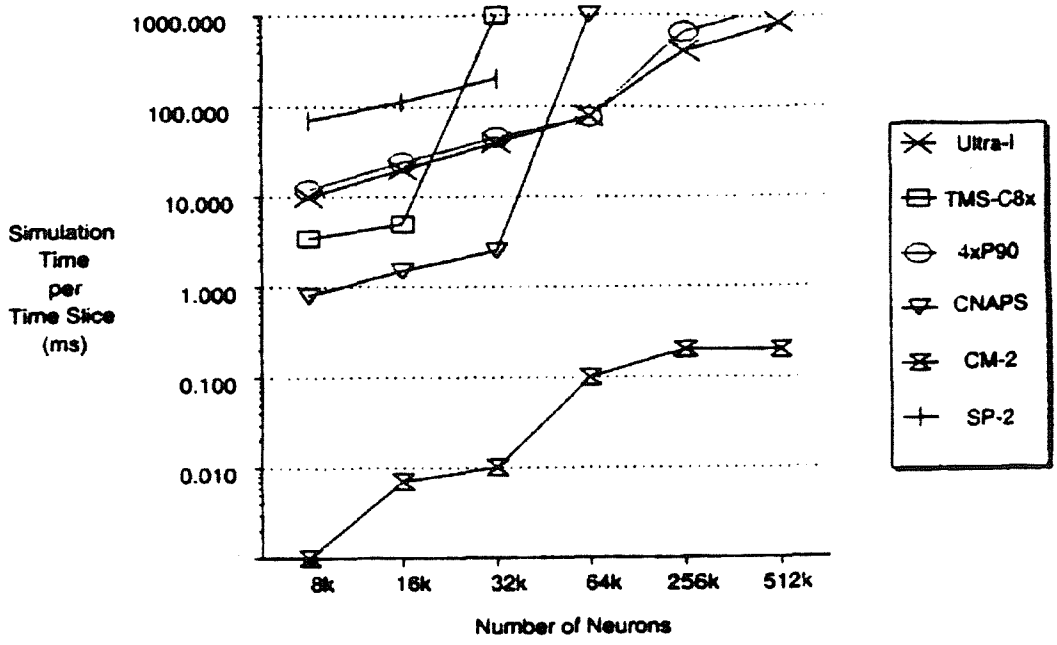


Figure 3.9: Comparison of the performance of several PAs (from [71])

3.2.2 Parallel architectures

Parallel architectures (PA) have been successfully used for the simulation of large networks. Brettle *et al.* [70] constructed a model of the primary visual pathway of the cat with 16000 single compartment neurons with 1000 synapses per neuron. The model ran on a *Connection Machine* CM-2 (a single instruction-multiple data, SIMD, machine) with 65536 one-bit processors and 1 Mbit local memory. The performance obtained was in the order of 10 μs CPU time per 1 ms simulated time.

The performance of several PAs has been studied in [71] for networks of IF neurons. The architectures explored were CM-2 (SIMD), TMS 320C80 (Texas Instruments), 4xP90 (with four Pentium P90) and SP2 (IBM, with 256 R6000 processing elements). The latter three platforms are all MIMD (multiple instruction-multiple data). Figure 3.9 shows the total computing time taken by a single time slice of 1 ms as a function of network size. The study concludes that massively parallel platforms (e.g. CM-2) are suitable for real time simulation of large networks of IF models. However, general purpose PAs are constrained by inter-processor communication bandwidth.

Special purpose parallel architectures (neurocomputers) have been proposed to overcome these limitations. For instance, toroidal lattice architectures (TLA) and

planar lattice architectures (PLA) were described in [72] for the simulation of abstract neuron models (Perceptrons and Hopfield neurons). It was estimated that the TLA architecture would allow high speed simulation of 10^6 million neurons with 10^4 synapses per neuron with 256×256 node processors and 128 Gb of memory. The CNAPS SIMD neurocomputer, developed by Hammerstrom [73], constitutes another example of a parallel hardware accelerator which has been used for efficient simulation of cortex-like networks [74].

Parallel versions of the simulators GENESIS (PGENESIS) and NEURON (PNEURON) have been developed to run on workstation clusters and supercomputers. By distributing the computing load amongst several processors and using a message-based synchronization procedure, thousands of neurons can be simulated. Parallelization of NEURON developed at the Pittsburgh Supercomputing Center (Carnegie Mellon University) used the Cray C90 architecture, achieving 100 fold increase in speed with respect to equivalent simulations ran on a Sparc II.

NEOSIM is under development to provide a parallel environment for mixed mode simulation which would achieve inter-operability between existing biophysical simulators [75].

3.3 Rationale for the use of message-based event-driven simulation on general purpose architectures

Compartmental models have been used in biophysical simulation of single neurons and small aggregates of neurons. However, the use of continuous simulation hampers its scalability.

Parallel architectures, in particular special purpose designs, offer high speed simulation of large networks. However, the cost of such systems makes the use of general purpose computing resources a more favorable alternative.

On the other hand, discrete simulation [76] results in a considerable increase in simulation performance when analogue models can be abstracted to their event-driven counterparts and the continuous framework can be substituted by a discrete framework [14]. Moreover, considerable research has been carried out on algorithms for efficient discrete simulation, in particular, to achieve an optimal management strategy for the event queue [77, 78, 79, 80]. Hence, discrete simulation was chosen as the framework to construct large scale and highly biologically

constrained models.

Classical biophysical neuron models are suitable for continuous simulation. A simplified description, adequate for discrete simulation, had to be developed. Although this task had been partially targeted by most abstract neuron models (Perceptron, Hopfield and others), which are compatible with discrete simulation, their loosely biologically based design limits direct use of physiological data in the model.

Message-based event-driven simulation is a natural framework to model distributed systems [81]. The message communication philosophy makes parallelization possible, if necessary for further improvements in performance, without fundamental changes in the algorithms.

The work by Pytte *et al.* [13] set a framework for the use of discrete neuron models in biologically realistic simulations. However, their implementation did not exploit some of the advantages of discrete simulation. In particular, the updating of states was continuous and selective trace was not implemented. Moreover, the neuron model used in this work could be further enhanced to account for physiologically measurable phenomena: delays introduced by the anatomical characteristics of dendritic trees, distance dependent delays in the axonal propagation of action potentials, pace making firing in CPGs (central pattern generators), inter-spike latencies within neuronal bursts and others.

Overall, the decision to adapt event-driven simulation techniques was taken because

- A reduction of the computational cost incurred by compartmental models is imperative to achieve large scale neural models
- Event-driven simulation is a well established field which has proven computationally efficient in areas like telecommunications and digital electronics
- The discrete nature of inter-neuronal communication motivates the use of neuron models suitable for an event-driven framework
- Previous work in this direction shows the feasibility of this approach but has not fully exploited its advantages

Chapter 4

C. elegans and the olfactory cortex

This Chapter provides, firstly, background information on the nematode *C. elegans* and reviews computer models of its neural circuits and mechanical properties. Secondly, the piriform cortex is introduced and previous network simulations are reviewed. MBED models of both biological systems, *C. elegans* and piriform cortex, will be constructed and discussed further in this thesis.

4.1 Biological targets of neural modelling

A wide range of biological targets can be found in the literature on neural modelling.

Invertebrates like the leech, the lobster and the mollusc *Aplysia* offer small-size neural aggregates subserving simple behavioural functions [82]. Their simplicity has motivated a large number of models in the hope of establishing links between neural activity and system level function. In this direction, the neural subsystems involved in locomotion have been the focus of a large amount of research effort. Experimental work on the leech using both standard microelectrode-based electrophysiological techniques [83, 84] and optical recording [85] have led to the development of computer models of its neuronal properties [86].

The stomatogastric system of the lobster and the associated neural ganglia are involved in food flow control and have also been the targets of network models. In a neuromechanical model [87] the connectivity space of the neural aggregate was searched and the conditions that maximized food flow found.

Neural control of locomotion in vertebrates offers a more complex problem than invertebrates [88]. A well understood system is the lamprey, whose locomotory nervous system has been extensively used in the study of fish swimming. A

neuromechanical model (neural aggregate + mechanical model) has been developed [69] based on experimental data [89]. It shows that swimming motion can be achieved by a chain of linked segments with the timing of motorneuron activation being regulated centrally by the brainstem and locally by intrasegmental interneurons.

Interest in the modelling of larger neural aggregates, in particular cortical modules, has been motivated by their role in high level functions in mammals. The cortical regions directly involved in the processing of incoming sensory information provide a somewhat easier case to link neural activity and function since controlled stimuli can be related to cortical activity patterns.

The visual cortex, for instance, has provided data on cortical image processing [41, 42, 90] and has motivated network models of image feature linking [41], direction selectivity [91], multiple topographic maps [92] and others. The auditory and somatosensory cortices have also been studied by network models [93], as well as extracortical areas like the cerebellum [8, 16].

The nervous system of the nematode *C. elegans* and the olfactory cortex of mammals constitute two examples of small and large scale neural aggregates, respectively; they are the target of the MBED network models developed in chapters 7 and 8. The following sections provide background information on these systems and review previous modelling work.

4.2 C. elegans

4.2.1 Background

C. elegans is a free living nematode of small size (1 mm long and approximately $80\mu m$ in diameter). It has a relatively rich set of behaviours which include feeding, temperature sensing, chemical sensing, mechanical sensing, defecation, mating, detection of changes in osmotic pressure, and others [94].

Its cylinder-like body is under considerable internal pressure which acts against an external cuticle [95]. The rigidity conferred by this internal pressure aids in the generation of wave-like locomotion. Figure 4.1-A shows an image taken with an optical microscope and figure 4.1-B is a diagram of the anatomy of *C. elegans*.

Two rows of electrically coupled muscles, situated in opposite sites of the body, constrain the nematode to movements in a plane. However, the head has an extra degree of freedom due to the grouping of its muscles in four electrically independent

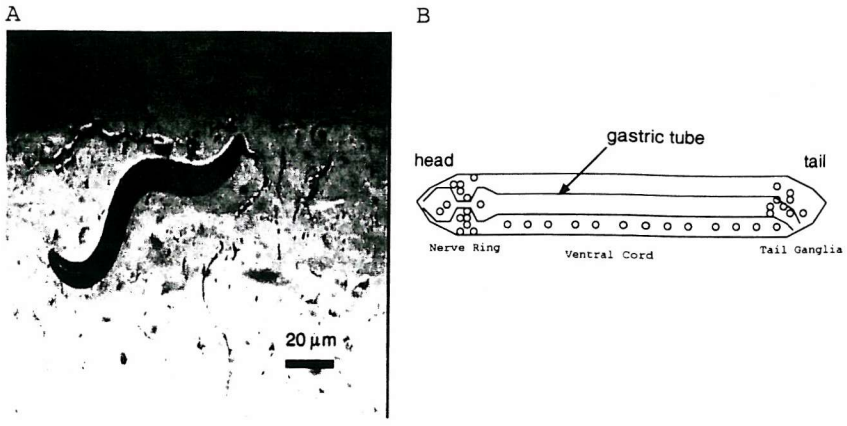


Figure 4.1: (A) Image of *C. elegans* taken with an optical microscope. (B) Schematic diagram of *C. elegans*.

quadrants.

The nervous system of *C. elegans* includes 302 neurons which can be found in three main locations; head, body and tail ganglia. Head ganglia work as information relay centers, the nerve ring being the most prominent, in addition to implementing paths of sensory input. Along the body, neurons extend longitudinally controlling the muscle contractions required for locomotion. Several neurons in the body constitute the egg laying subcircuit. In the tail, neurons sense their local environment and control functions like defecation [94, 96, 97].

C. elegans has a number of peculiarities which make its nervous system interesting from the modelling point of view; mainly, its known topology, its adequacy for the use of genetic tools and its transparency which allows laser ablation of individual neurons.

Regarding topology, the same neurons can be identified in different individuals, by their morphology and position with respect to the rest of the body. The connectivity can also be used as a means to identify neurons as it has been shown to be fairly constant [97]. In addition to this unusual invariability, the nematode is a very special case as the topology of its nervous system has been completely mapped using electron microscopy [96, 98, 97, 99].

Moreover, *C. elegans* is often used in genetic studies. Strains carrying mutations which affect genes required for the normal function of the nervous system are available. When those neurons affected by the mutation are involved in locomotion control, the result is an abnormal movement coordination.

These malfunctions may be due to erroneous connectivity of neurons or to

abnormal neural functionality [100, 101, 102]. Examples of both cases will be used in the validation of the MBED model of the locomotory neural circuit.

The body of the nematode is transparent, allowing the use of laser beams to ablate specific neurons. The functional elimination of identified neurons may induce abnormal behavioural patterns and provide clues on the role of the ablated cells on network level dynamics [103, 101, 104]. This technique has generated data which can be used in assessing the validity of neural models as it is possible to simulate ablation of a neuron in the model and compare the results with the experimental data. This approach has also been taken with the MBED model of *C. elegans* presented in this thesis.

Finally, histochemical experiments allow the identification of the neurotransmitters used in individual synapses and suggest a tentative classification of these connections into excitatory or inhibitory. This is possible by creating fluorescently labelled antibodies that specifically bind to a selected type of neurotransmitter receptor. The identification of fluorescent spots due to clustering of the antibodies indicates the presence of that particular receptor and neurotransmitter [105]. This type of information will also be introduced in the model.

Despite the abundance of topological and genetic data, electrophysiological recordings are limited so far to muscle cells [106], unidentified neurons (Lockery, personal communication) and chemical sensor neurons in the head [107]. This is due to the reduced size of its neurons (a few microns in diameter) and their lack of accessibility (due to the internal pressure, dissections easily damage the nervous system).

4.2.2 Models of C. elegans

C. elegans has been the target of several computer models dealing with the physics of nematode locomotion and the neural circuitry involved in its response to chemical compounds, touch stimuli and temperature changes.

Locomotion

The 2D mechanical model of the body developed by Niebur *et al.* [108] is relevant to the development of neural models since it provides a tool to predict the behavioural effect of a given pattern of neural activity. The body was modelled as a segmented

elastic cuticle with any one patch of this cuticle experiencing four force vectors; internal pressure, surface elastic force, environmental friction and muscle contraction.

The internal pressure force vector is given by

$$\vec{F}_{ip} = (\frac{p(0)V(0)}{V(t)})^a S\,\hat{n}$$

where $p(0)$ and $V(0)$ are the pressure and total volume of the body at $t = 0$, a is a positive integer in the range $(4 - 8)$, S is the surface of the patch of cuticle considered and $\hat{n}$ is a unitary vector normal to the surface.

The elasticity of the cuticle introduces a second force term. In the case of two points situated at opposite sites of the body, the elastic force is given by

$$\vec{F}_{ec} = k\,(\left|\frac{\vec{x}_i - \vec{x}_j}{d}\right| - 1)(\vec{x}_i - \vec{x}_j)$$

where k is a scaling constant, d is the diameter of the body and $\vec{x}_i$ and $\vec{x}_j$ are position vectors of the points. In the case of two contiguous points, $\vec{x_i}$ and $\vec{x}_{i+1}$, situated on the same site of the body, the elastic force is given by

$$\vec{F}_{ef} = f(l)(\frac{\vec{x}_i - \vec{x}_{i+1}}{l})$$

where l corresponds to the distance between the pair of points $\vec{x_i}$ and $\vec{x}_{i+1}$ $(|\vec{x}_i - \vec{x}_{i+1}|)$ and $f(l)$ is a non-linear function accounting for the non-linear elasticity of the cuticle.

The environment introduces a frictional force term acting on any one point on the surface of the body,

$$\vec{F}_{ff} = -c_1\,\vec{v}_t - c_2\,\vec{v}_n$$

where $\vec{v}_t$ and $\vec{v}_n$ are the tangential and normal components of the velocity vector at the considered point. c_1 and c_2 are positive scaling constants.

Finally, muscles introduce a fourth force component, collinear with a vector tangent to the body surface, given by

$$\vec{F}_{mf} = e(m,t)\,\eta\,\hat{t}$$

where $e(m,t)$ is a dimensionless number which depends on the state of the m^{th}

motorneuron at time t, η is a constant and $\hat{t}$ is a unitary vector tangent to the surface. Niebur *et al.* confirmed with this model that a pattern of muscle excitation consisting of waves propagating towards the tail and the head succeeded in generating realistic forward and backward locomotion respectively.

The focus of this work was the mechanical dynamics of locomotion rather than the neural circuitry underlying locomotory behaviour. However, in further work, the same author estimated the attenuation and velocity associated with passive propagation of activity in the axons of the motoneurons in *C. elegans* [95].

The estimated velocity in motorneurons ($\sim 8 - 30$ cm/s) was found to be much larger than the observed velocity of propagation of muscle contraction (~ 0.2 cm/s), ruling out the possibility of obtaining the observed patterns of muscle contraction solely as the result of passive propagation in motorneurons. These calculations suggested that the generation and propagation of waves along the body of *C. elegans* could rely on the use of stretch receptors. In agreement with these results, stretch receptors play an important role in the MBED model presented in Chapter 7.

The problem of the genesis of muscle contraction patterns was tackled experimentally by Stretton *et al.* [109] in the nematode *Ascaris*, which is thought to use mechanisms similar to those found in *C. elegans*. As a first approximation, a qualitative description of the propagation of the muscle contraction wave was put forward by Walrond *et al.* [110]. Experimentally, it was shown that the excitation wave propagates within motorneuron axons as opposite to a exclusively muscular propagation and further evidence for the presence of stretch receptors was put forward. These results are in agreement with Niebur's estimations. However, no computer simulation was provided for further quantitative studies.

Neural processing of sensory input

Detailed information on sensory circuits in *C. elegans* [111] has motivated several computer models of its response to sensory input (chemicals, light touch and temperature).

Chemotaxis refers to the ability of the worm to escape from damaging chemicals and to approach zones of high concentration of desirable compounds. The circuit involved in chemotaxis has been studied with laser ablation and a network model has been developed [112]. This model was constructed using neurons with relaxation dynamics, where the membrane voltage for any one neuron is given by

$$\tau_i \frac{dV_i}{dt} = -V_i + \sum w_{ji}(V_j - V_{ji_0}) + C(t)$$

where V_i is the membrane voltage of the i^{th} neuron, τ_i is its time constant, w_{ji} the synaptic weight between cells j and i, V_{ji_0} is the membrane voltage at which the synapse does not produce changes in the postsynaptic cell and $C(t)$ is a term only present in those cells that receive input from the environment. A number of cells in the network acted as steering signals feeding a black box model of the locomotory system. With appropriate selection of parameters, the model predicted movement towards increasing gradients of desirable chemical compounds. Thus, the network did not include a cell-level representation of the locomotory circuit.

Tab-withdrawal in *C. elegans* has also been modelled. It has been observed experimentally that the nematode locomotes backwards when its nose is touched and forwards when its tail is touched. The neural circuit involved in controlling this behaviour was identified by laser ablation [113, 103].

In [47], a continuous non-spiking neuron model was used in a simulation of the tab-withdrawal circuit. The aim of this work was to determine the sign of the synapses in the circuit (positive being excitatory and negative inhibitory) by testing multiple configurations and quantifying the probability of a certain parameter set generating the experimentally observed behaviour.

The dynamics of the membrane voltage in a neuron were given by

$$C_m \frac{dV}{dt} = \frac{1}{R_m}(V_{leak} - V) + \sum I_{syn} + I_{ext}$$

where C_m is the membrane capacitance, R_m its total leakage resistance, V_{leak} the membrane voltage at which no leakage current flows through the membrane, I_{syn} the synaptic currents and I_{ext} corresponds to the injected current.

The network model is shown in figure 4.2. It does not explicitly incorporate nematode locomotion. Rather, neurons AVA and AVB are assumed to be the main output to the locomotory circuit and to control locomotion as described by the *gearbox* analogy; the difference between the membrane voltage in neurons AVA and AVB, the steering signals, determines the speed and direction of locomotion

$$L \propto \int_{t_0}^{t} (V_{AVB}(\lambda) - V_{AVA}(\lambda))d\lambda$$

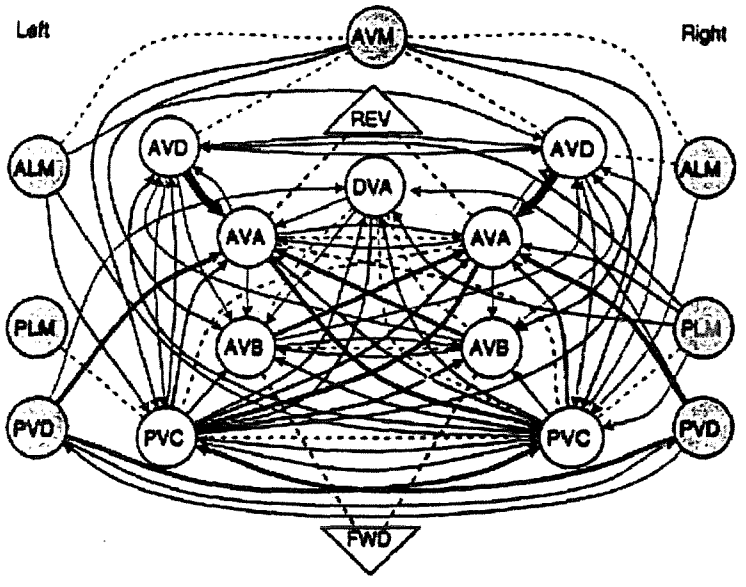


Figure 4.2: Network model as constructed by Wicks *et al.*[47]

where L is a quantitative measure of speed and direction of locomotion, V_{AVB} and V_{AVA} are the membrane potentials in neurons AVB and AVA respectively and t_0 is the last sign reversal. The sign of L distinguishes between forward and backward propagation and its magnitude sets the speed.

Laser ablation experiments were simulated with the model and the results quantitatively compared to experimental observations. For this purpose, a measure of the difference between the simulated and experimental values of L , based on least-squares error, was used to determine the fitness of each of the tested synaptic configurations.

An alternative approach to the modelling of the touch sensitivity circuit is being pursued by Osana *et al.* (personal communication) using the Boltzman machine formalism with a network of 66 neurons. Preliminary results indicate that, after training with Hebbian-like learning algorithm, the network converges to a configuration where the excitation of head and tail mechanical sensors generates backward and forward locomotion respectively.

The neural circuit involved in thermotaxis (the ability of the animal to move towards zones with an ideal temperature) has been identified with laser ablation. Tentative functions for some neurons were assigned after laser studies [114]. However, no mathematical model has been published. This is also the case with regard to the locomotory circuit, responsible for direct control of the body muscles.

Although several classes of neurons have been identified and generic functions proposed, no computer model is available.

The models of chemotaxis, thermotaxis and tab-withdrawal mentioned above did not explain the generation and control of the patterns of muscle contraction seen in *C. elegans*. They focused on the neural circuitry which maps the sensory input into a steering signal delivered to the locomotory system. The model proposed in Chapter 7 will deal with this untackled problem.

4.3 The piriform cortex

The neural structures involved in odour perception constitute a phylogenically old part of the mammalian brain. Several species have served as model systems to study their physiology and information processing capabilities; the cat [115], the rabbit [116], the rat [117] and the opossum [118] among others. A subset of the information obtained from these studies, specifically that relevant for the MBED model of the piriform cortex, is presented in the following sections.

4.3.1 Modules within the olfactory system

The first stage in the olfactory system (figure 4.3) corresponds to the transduction carried out by the chemical receptors located in the nasal cavity, within a layer known as olfactory epithelium. Each one of these chemical sensors detects the presence of a range of chemical compounds and transmits olfactory information to the first processing center, the olfactory bulb. Different odours generate spatially different patterns of input activity [119].

The olfactory bulb consists of several conglomerates of cells, glomeruli. By means of the interaction between excitatory (mitral) and inhibitory (granular) cells in these glomeruli, the sensory input from the smell receptors triggers the onset of oscillatory activity. It is believed that the olfactory bulb carries out an initial processing on smell information. However, the exact nature of this processing is still unclear [33].

In mammals, the olfactory bulb sends its output directly to several cortical modules. The biggest of these areas is the piriform cortex or primary olfactory cortex, which is involved in further processing of olfactory information. The connection from olfactory bulb to piriform cortex is provided by a bundle of axons termed the lateral olfactory track (LOT). There is also anatomical evidence of a feedback connection from piriform cortex to olfactory bulb [120].

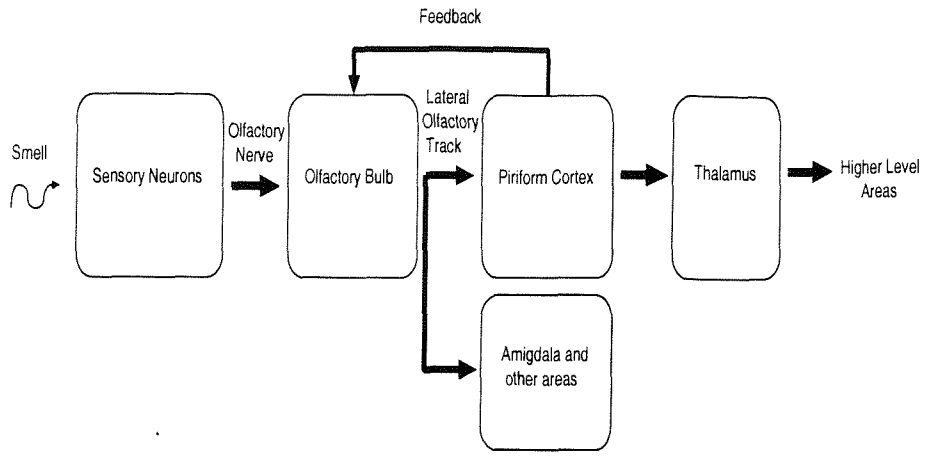


Figure 4.3: Block diagram of the olfactory system

4.3.2 Structure of the piriform cortex

The piriform cortex contains an estimated 10^7 neurons of 10 different types [121, 118, 120]. Its anatomical structure is often divided in three areas with distinctive characteristics (see figure 4.4). Layer I, the plexiform layer, includes few cell bodies and is mainly occupied by dendrites from neurons located in deeper layers. Layer I is further divided in Ia (superficial lamina) and Ib (deep sublamina). In sublayer Ia, the LOT establishes synapses with the dendritic trees which fill layer I. Sublayer Ib is characterized by the presence of synapses between pyramidal cells.

Layer II corresponds to the lamina with the highest density of cell bodies. Figure 4.5 shows a cross section of the piriform cortex stained with the Golgi technique (taken from [121]) where, due to its high density of cells, layer II appears as a dark band. The more superficial layer I, shows as a brighter band. The density of cells decreases gradually from layer II towards the deeper layer III, showing as a graded increase in brightness in figure 4.5.

Layer III contains both cell bodies (although at a lower density than layer II) and dendrites emanating from cells and directed downward in figures 4.4 and 4.5.

Layers II and III contain large numbers of pyramidal cells. These are anatomically similar to pyramidal cells in other cortical areas. Two dendritic trees originate in the cell body; an upper tree extends along layer I whereas the bottom tree extends along layers II and III. A typical pyramidal cell in the piriform cortex receives synaptic input from the olfactory bulb through the LOT and from other cells in the cortex. Axons from pyramidal cells establish excitatory local synaptic connections with nearby neurons and also long range excitatory connections with

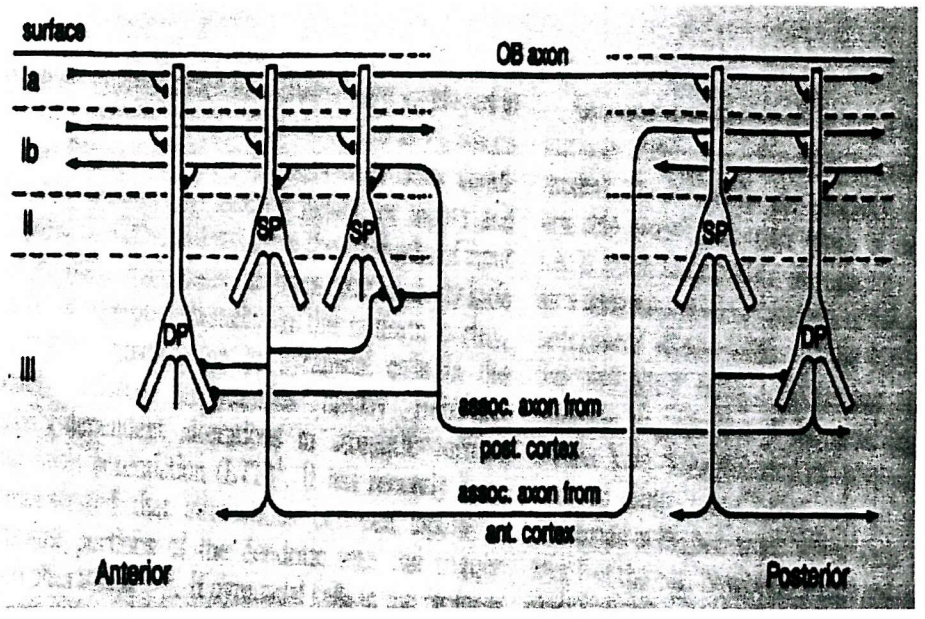


Figure 4.4: Layered structure of the piriform cortex (from [122])

distant neurons [122].

In addition to pyramidal cells, at least nine anatomically distinct classes of non-pyramidal neurons have also been identified in the piriform cortex. Some of these types are thought to be inhibitory cells. In particular, three classes of inhibitory connections have been observed physiologically; $GABA_A$ slow, $GABA_A$ fast and $GABA_B$ slow. $GABA_A$ synapses have a short onset latency (approximately 1 ms) and an activation duration of about 10 ms. On the other hand, $GABA_B$ synapses have an onset latency of 50 ms and an activation duration of 100 ms [28].

For modelling purposes, inhibitory cells have often been grouped into two classes;

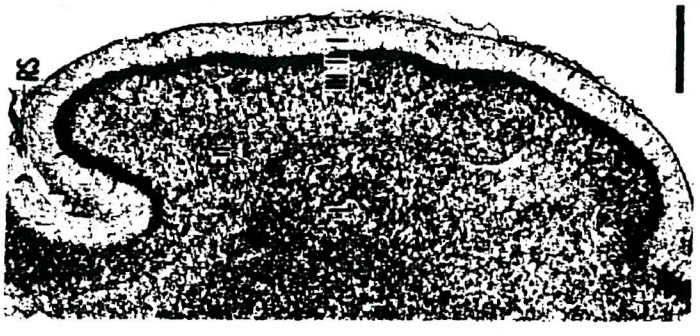


Figure 4.5: Cross section of the olfactory cortex stained with Golgi techniques, bar= 800 μm (from [121])

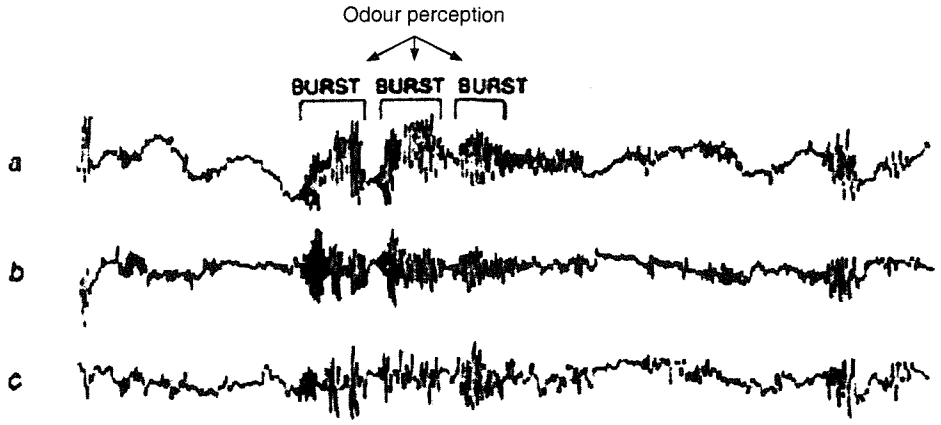


Figure 4.6: EEG recordings from olfactory bulb (a), anterior (b) and posterior (c) olfactory cortex (from [116])

slow feedforward inhibition and fast feedback inhibition. The former receives input from LOT and pyramidal cells and synapses back onto pyramidal neurons and the latter receives input from pyramidal neurons and synapses back onto them [28, 123].

4.3.3 Experimental data

Multiple electrophysiological and optical recordings of activity in the piriform cortex have been described in the literature. A limited number of these recordings aimed at the measurement of the cortical response to odour perception in-vivo. The main feature of these results was the presence of bursts of activity in the gamma range (at approximately 40 Hz) synchronized with sniffings [116]. Figure 4.6 shows EEG recordings obtained from the olfactory bulb (a) and from two distant locations in the olfactory cortex (b and c). During odour perception, bursts of high frequency and amplitude can be seen in the traces.

Due to experimental difficulties, the majority of recordings have been obtained in far less realistic setups. In most experiments, cortical activity is not evoked by odour perception, rather, electrical stimulation of the LOT causes the observed neural activation.

Ketchum *et al.* [124, 117] recorded the potentials generated by neuronal activity at several locations in the cortex after excitation of the LOT with a 0.1 ms current pulse. Two types of responses were obtained (see figure 4.7); a single peak and a damped oscillation responses could be triggered, the parameter determining the

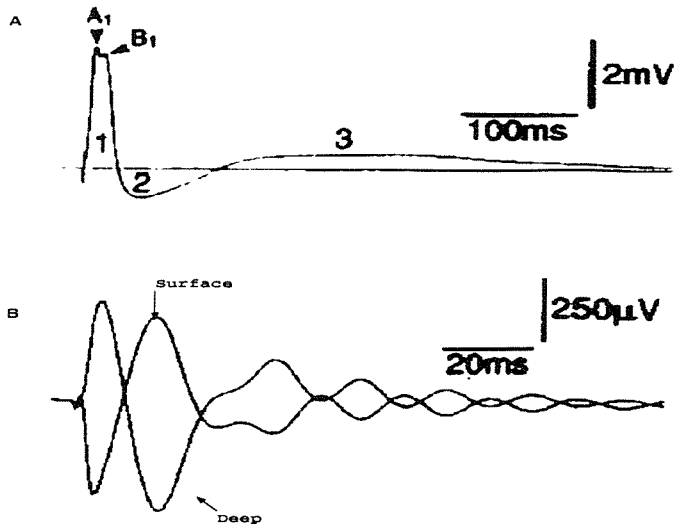


Figure 4.7: (A) Strong shock stimulus response, (B) Weak shock stimulus response (from [124])

response type being the intensity of the excitation current pulse. High intensity pulses evoked the single peak response whereas lower intensity pulses triggered damped oscillations.

Optical recordings using voltage sensitive dyes have also been carried out in the olfactory cortex. Curtis *et al.* [125] and Litaudon *et al.* [126] reported waves of activity across the cortex evoked by excitation of the LOT and olfactory bulb.

4.3.4 Network models

Several network models of the olfactory cortex have focused on its suspected role in odour recognition, proposing mechanisms for the achievement of an associative memory [122]. Freeman proposed, in 1987, such a network model of the olfactory cortex using an analytical approach and avoiding the modelling of individual neurons. The dynamics of the cortex were described in terms of ODEs and pointed at the resemblance between the cytoarchitecture of the olfactory cortex and the networks shown to implement associative memories [127, 128].

The Lynch-Granger model [129, 130] was proposed to study the function of the olfactory bulb and olfactory cortex system as an odour classifier system. The network model in [129] included 400 excitatory and inhibitory cells in the olfactory bulb model and 1000 excitatory and 50 inhibitory cells in the olfactory cortex model (see figure 4.8). Neurons were modelled as linear elements which summated the incoming inputs according to a matrix of weights W. A Hebbian learning algorithm

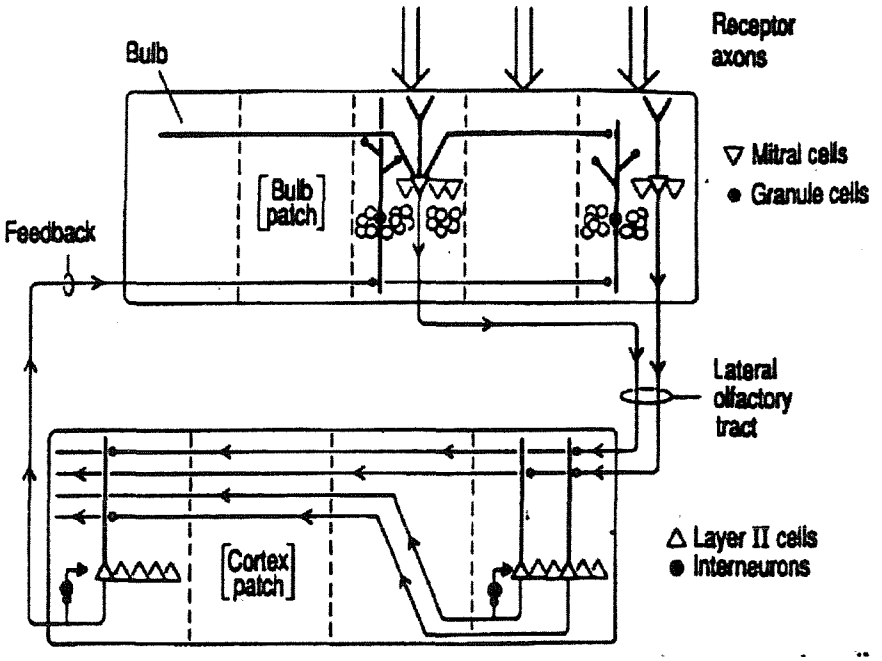


Figure 4.8: Bulb-cortex model developed by Lynch *et al.* [129]

was applied in conjunction with a "winner-take-all" mechanism to adjust synaptic weights as a result of the input generated by simulated sensing of odours. The aim of this model was to show that such a network was suitable for the implementation of an odour classifier. However, the activity of the model was only compared to psychophysical data. Comparisons with electrophysiological results were precluded by the level of abstraction of the model.

Li and Hertz [119] have recently proposed an alternative bulb-cortex model. In one of its implementations, the network included 200 neurons of four cell types (excitatory and inhibitory in bulb and cortex) in equal proportions. Formal excitatory and inhibitory neurons in the bulb model had an associated membrane potential, x and y respectively, whose dynamics are given by

$$\frac{dx_i}{dt} = -\alpha x_i - \sum H_{ij}\, g(y_j) + I_{sensors}$$

$$\frac{dy_i}{dt} = -\alpha y_i + \sum W_{ij}\, g(x_j) + I_{cortex}$$

where H_{ij} and W_{ij} are the synaptic weights between units i and j, and g is a

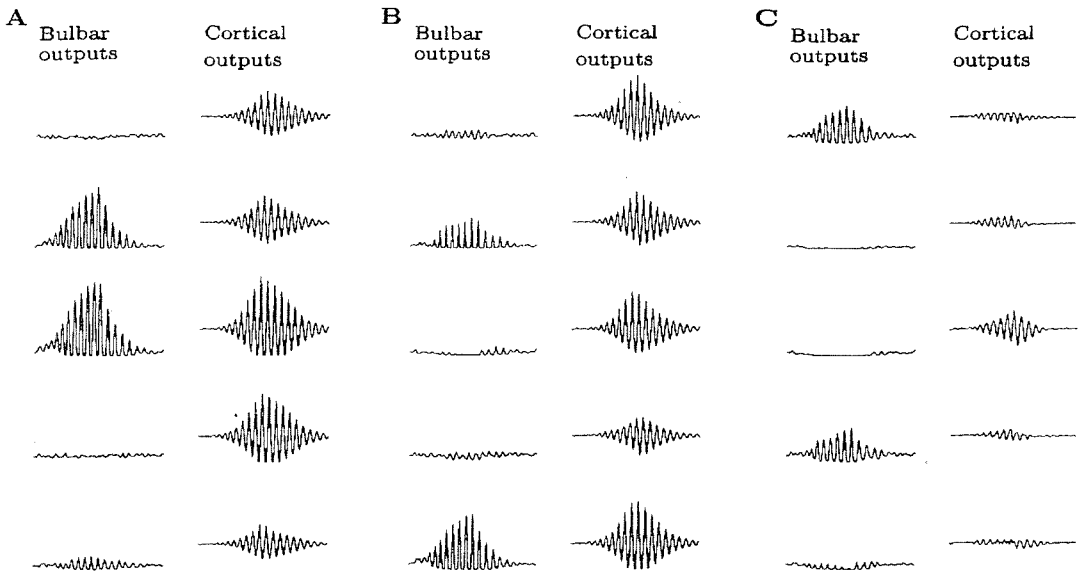


Figure 4.9: Activity in five neurons from a bulb-cortex model in response to three (A,B,C) different odours (from [119]).

function that maps membrane potential into firing rate.

The dynamics of the cortical model were described by similar expressions

$$\frac{du_i}{dt} = -\alpha u_i - \beta g(v_i) + \sum J_{ij} g(u_j) + I_i$$

$$\frac{dv_i}{dt} = -\alpha v_i + \beta g(u_i) - \sum D_{ij} g(u_j)$$

where u and v are the membrane potentials for excitatory and inhibitory cells and J and D are synaptic strength matrixes.

The aim of this model was to study how a biologically motivated neural network model could detect, recognize and segment odours. Figure 4.9 shows the temporal traces of the outputs from 5 excitatory cells in the bulb and cortex obtained for three different odour stimuli (A, B and C). A common feature in all responses is the oscillatory nature of the signal with a frequency of approximately 50 Hz and an intensity dependent on the number of previous exposures of the system to the odour. In figure 4.9, odours A and B (with which the model had been trained) trigger more generalized activity than odour C (completely unknown to the network).

A small network model with 10×10 neurons was constructed in [131] to

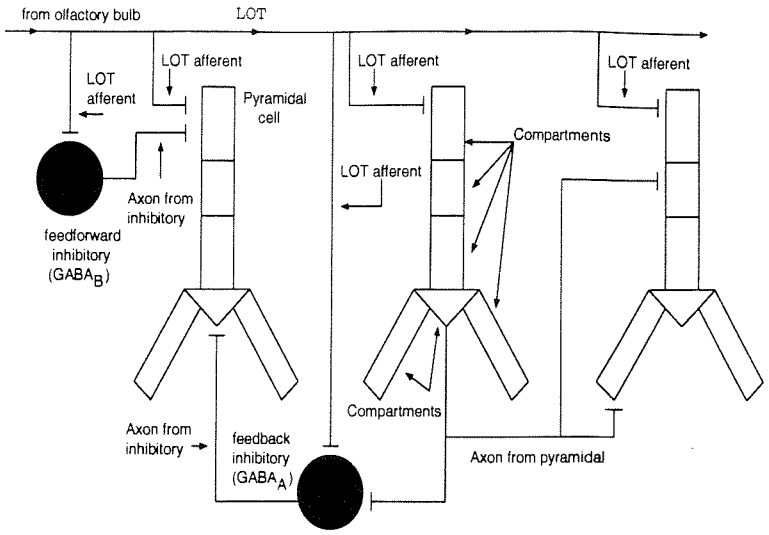


Figure 4.10: Schematic drawing of the structure simulated by Wilson *et al.* [28]

demonstrate associative memory properties of the piriform cortex. As in previous associative memory models, the proposed cortical mechanisms were based on anatomical data and psychophysical experiments with no direct reference to electrophysiological data.

Although this line of research provided qualitative understanding of cortical dynamics, the lack of direct mapping between biophysical parameters and parameters in the model limited its use as a tool for the neuroscientist.

Following an alternative approach based on realistic compartmental neurons, a model of the piriform cortex including 4500 neurons (effectively 405 neurons, given that, during data collection, the number of neurons had to be reduced) was constructed by Wilson *et al.* [28]. The network included three types of cells; slow inhibitory, fast inhibitory and pyramidals (excitatory).

Pyramidal neurons were modelled by a six compartment structure as shown in figure 4.10; the upper-most compartment received synapses from the LOT and slow inhibitory cells, the middle compartment received input from pyramidal cells, the cell body received synapses from fast inhibitory neurons and the bottom-most compartments received local connections from nearby pyramidal cells. Inhibitory cells were modelled with a single compartment.

Field potentials were simulated applying the following approximation

$$V(t) = \alpha \sum_{j}^{J} \sum_{k}^{K} \frac{I_{jk}(t)}{r_{jk}}$$

Experiment	Result
Strong shock stimulus	Single wave
Weak shock stimulus	Damped oscillation
Random input	40 Hz + 5 Hz frequency components

Table 4.1: Experimental results considered in the simulation by Wilson *et al.* [28]

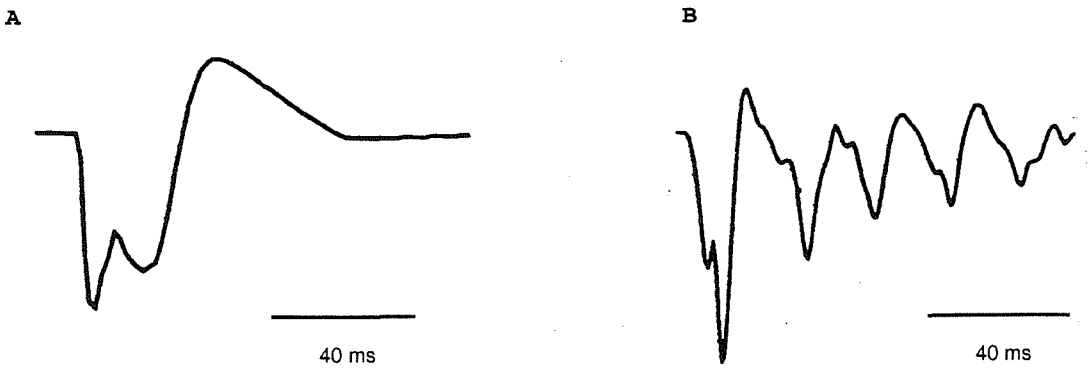


Figure 4.11: (A) Strong shock response, (B) Weak shock response

where

$$r_{jk} = \sqrt{[(x - x_{jk})^2 + (y - y_{jk})^2 + (z - z_{jk})^2]}$$

and (x_{jk}, y_{jk}, z_{jk}) and (x, y, z) are the position of the k^{th} compartment in the j^{th} neuron and the location of recording electrode, respectively. $I_{jk}(t)$ is the current flowing through the compartment and α is a scaling constant.

Table 4.1 summarizes the main results obtained. The response of the model was tested by providing three different types of stimulus through the LOT; low intensity shock stimulus, high intensity shock stimulus and random input.

Figure 4.11 shows the simulated field potential recordings obtained after low and high intensity shock stimuli. The response to the low intensity stimulus is a damped oscillation whereas the simulation predicts a single peak response to a strong shock stimulus.

With the same model, the frequency components present in simulated EEGs were also studied. The EEG recording was simulated by linearly adding the simulated field potentials obtained with regularly spaced virtual electrodes positioned in a grid over the cortical model

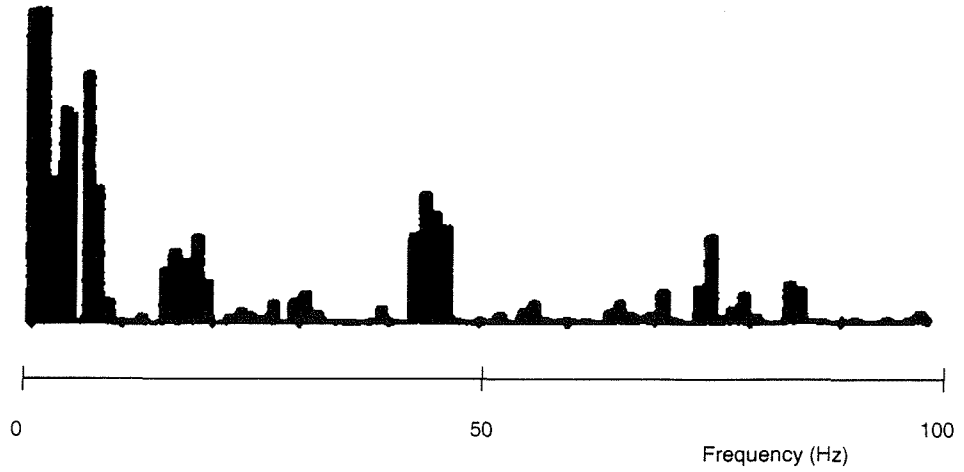


Figure 4.12: Simulated EEG power spectrum obtained by Wilson *et al.* [28].

$$V_{EEG}(t) = \sum_{x}^{X} \sum_{y}^{Y} v_{xy}(t)$$

where $v_{xy}(t)$ is the potential recorded by the electrode placed in position (x, y) and the summations are over the $X \times Y$ electrodes of the grid. The assumption contained in this approximation states that the EEG signal corresponds to a measurement of average activity in a large region of the cortex. This approximation was consistent with previous work on EEG [132].

EEG simulations were carried out providing random input to the model through the LOT. Figure 4.12 shows the estimated power spectrum of the EEG signal obtained. The main features of this result were a relatively high frequency component (40 Hz) and the theta-type activity peak (3-10 Hz). Both have been observed experimentally. Secondary peaks in frequency bands centered at 80 Hz and 20 Hz were also obtained.

Building on these simulations, Barkai *et al.* [123, 133] constructed a biophysical model of the piriform cortex including 240 pyramidal and 58 inhibitory cells of two classes, $GABA_A$ and $GABA_B$. Each compartmental neuron contained three compartments and several voltage and calcium dependent currents (I_{Na}, $I_{K(DR)}$, $I_{K(A)}$, $I_{K(M)}$, $I_{K(AHP)}$) responsible for shaping the trains of action potentials in

individual neurons. The simulation was carried out with the GENESIS simulation package and aimed at studying the effects of the neuromodulator acetylcholine on the postulated associative memory mechanism implemented by the piriform cortex. This work concluded that the increased rate of synaptic modification triggered by acetylcholine favours the learning phase whereas the suppression of the effects of acetylcholine facilitates the recall phase.

Ballain *et al.* [134] proposed an alternative approach to the compartmental modelling techniques based on relaxation dynamics to describe mathematically the time evolution of the optically recorded activity in the olfactory cortex. Since this type of recordings do not allow the measurement of activity from individual cells (each photodiode imaging average activity in approximately 2000 to 4000 neurons), the model consisted of a network of 54×24 units representing the ensembles of neurons imaged by individual photodiodes. For each ensemble, two subpopulations and the corresponding two associated state variables were introduced in the network model; V_{ij}, representing the average state of the excitatory variables in the pool, and U_{ij}, corresponding to the inhibitory subensemble.

The dynamics of each subensemble were given by

$$\frac{dV}{dt} = -\alpha V + (V_e - V)\sum_{k}^{K} g_1 B(t - t_{d1}) + (V_e - V)\sum_{w}^{W} g_2 \phi(V(t - t_{d2})) + (V_i - V) g_3 \phi(U(t - t_{d3}))$$

$$\frac{dU}{dt} = -\beta U + (U_i - U) g_4 \phi(V(t - t_{d3}))$$

where V_e and V_i are excitatory and inhibitory equilibrium potentials respectively, B represents the input activity from the olfactory bulb, ϕ is a transfer function, g_x are scaling factors and the summations are over the K input signals from the olfactory bulb and over the total number of ensembles, W.

Simulations were carried out to study the response of this model to strong and weak shock stimulus, obtaining single peak and oscillatory responses respectively, in accordance to experimental results. Moreover, propagation of waves and pacemaking activity between preferentially coupled ensembles was also successfully predicted by the model.

4.4 Summary

The nervous system of the nematode *C. elegans* contains 302 neurons and its connectivity has been mapped in its entirety with EM. Several network models have been constructed to study chemotaxis and tab-withdrawal.

A subcircuit containing approximately 100 neurons is directly involved in the control of locomotion. A model explaining the cooperation of the network to achieve locomotion has not been described in the literature. Progress has been hampered by the lack of electrophysiological data from these cells. However, tentative functions can be assigned to some neuron classes based on topological information and laser ablation experiments. This makes the locomotory circuit an interesting case from a modelling perspective.

The piriform cortex constitutes an opposite case, a large network where only statistical connectivity rules are known but where electrophysiological data are available. Several network models of the piriform cortex have been constructed. Those with emphasis on the replication of electrophysiological data have made use of compartmental models. The computational cost involved in the simulation of these types of models has limited the size of network models to approximately 5000 neurons. Other network models, with an emphasis on the suggested implementation of associative memory by the olfactory cortex, have relied on analytical descriptions of the dynamics of neuronal ensembles. Although this approach is potentially suitable for large scale simulations, its high level of abstraction complicates the correlation of simulation results and experimental data. Thus, there is a need for the development of large scale models, which allow the incorporation of biophysical parameters, such as synaptic timings, in order to explore the effects of these parameters on network dynamics.

Chapter 5

Message-based event-driven neuron model

In this Chapter, the message-based event-driven neuron model (MBED) is described. The blocks making up its internal structure (synapses, threshold subsystem, burst generator and oscillator) are explained and examples are provided both to illustrate their operation and to validate the correctness of the implementation. Functional similarities and differences between the event-driven model and compartmental models are shown. For this purpose, simulations of compartmental models have been carried out using the simulator Neuron [9].

5.1 Internal structure

The MBED neuron model is a finite state automaton. It is made up of several blocks (synapses, threshold subsystem, burst generator and oscillator), each of them capturing the functionality of a different component of the neuron (see figure 5.1).

Communication between blocks within a single neuron is achieved by message passing through unidirectional message channels (see table 5.1 for a complete list of message channels and legal message types for each channel). Message channels are depicted as solid line arrows in figure 5.1.

Each message is a data packet containing the following fields: delay, message type label and an optional parameter (see section 2.1.2 for a description of a discrete simulation framework based on packet exchange). The delay field contains the delay between message generation and message delivery to the destination. The message type label indicates the type of message which will determine the action taken by

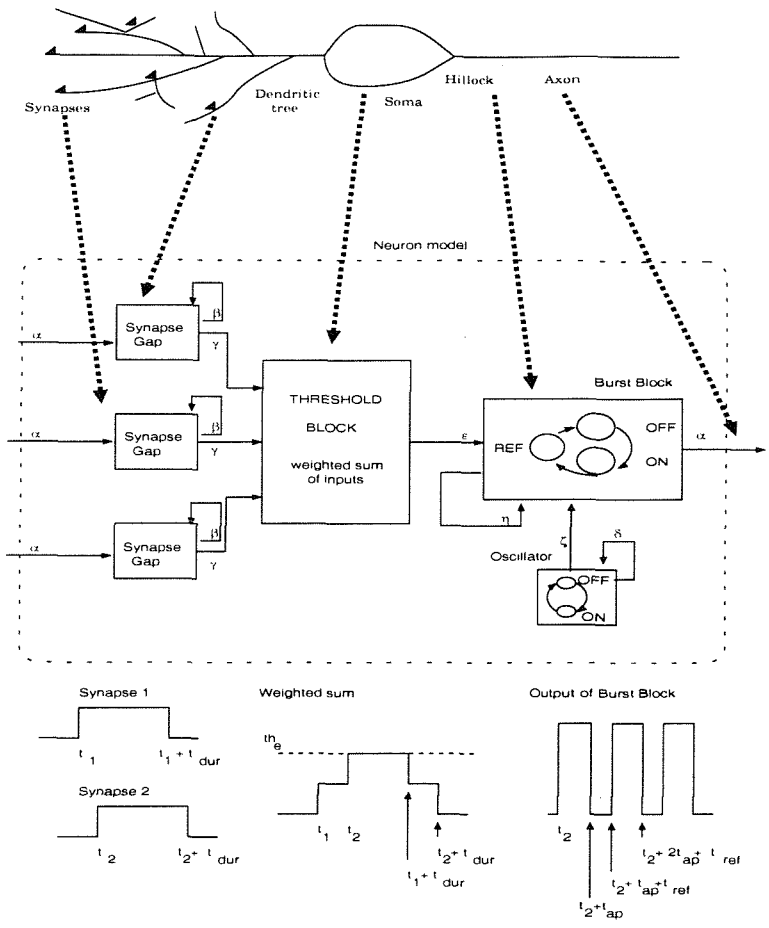


Figure 5.1: Message-based event-driven neuron model (solid arrows indicate channels for message broadcasting).

Channel	Message structure	Legal values of m	Legal values of p
α	{t,m,p}	*on*	Synapse type
β	{t,m}	*on*,*off*	
γ	{t,m,p}	*on*,*off*	Synapse type
δ	{t,m}	*change*	
ϵ	{t,m}	*on*,*off*	
ζ	{t,m}	*on*	
η	{t,m}	*off*, *ref*	

Table 5.1: Message channels in the neuron model

Parameter	Function
th_e	Excitation threshold
th_i	Inhibition threshold
t_{ap}	Duration of action potential
t_{ref}	Duration of refractory period
N_{burst}	Number of spikes per burst
t_{osc}	Period of pace maker
t_ϕ	Time offset of pace maker
t_{del}	Synaptic delay
t_{dur}	Duration of synaptic pulse
w_{syn}	Synaptic efficacy

Table 5.2: Parameters used in the model

the target block upon reception of the message. The optional parameter provides complementary information required by the destination block to process the message.

The message space of a neuron can be classified as external input events, external output events and internal events. External input events arrive through α channels and communicate to local synapses that a presynaptic neuron has fired. External output events correspond to outgoing messages broadcast to postsynaptic cells upon initiation of an action potential. Internal events are amenable to further categorization; inter-block messages (α, γ,ϵ, ζ) are communicated between internal blocks of a single neuron whereas intra-block messages (β, η, δ) are scheduled for the same block which generated them with the sole purpose of introducing a delay between two state changes. The former have a biological counterpart in the propagation of transient membrane voltage changes from synapses, along dendrites, to the cell body and proximal axonal segment (hillock zone, where the spike is initiated) and along axons to the next synapse. The latter are convenient abstractions to support the desired functionality.

Blocks in the neuron model are either state machines or combinational functions. In the state machines, the arrival of a message may trigger a change of state, an action (the update of internal state variables) and an output (the broadcasting of new messages). In combinational functions, the arrival of an input message triggers the broadcasting of one or more output messages.

Tables 5.2 and 5.3 show the complete set of parameters, states and variables in the MBED neuron. Their purpose is described further in the following sections simultaneously with the block to which they are associated.

Block	Allowed states	State variables	Parameters
Synapse	-	-	$t_{del}, t_{dur}, w_{syn}$
Threshold	-	w_{sum}	th_e, th_i
Burst generator	on, off, ref	n_{burst}	$t_{ap}, t_{ref}, N_{burst}$
Oscillator	on, off	-	t_{osc}, t_{ϕ}

Table 5.3: Allowed states, state variables and parameters for each block in the model

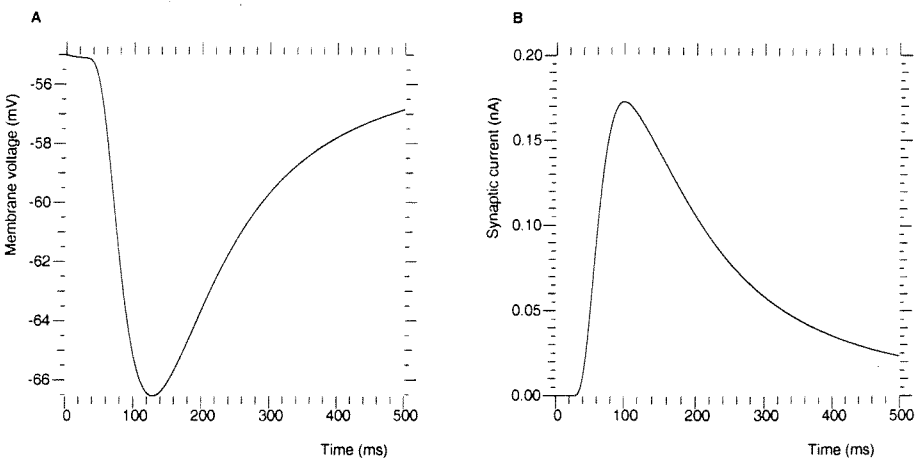


Figure 5.2: (A) Change in membrane voltage due to the activation of a $GABA_B$ synapse, (B) synaptic current

5.2 The synapse block

5.2.1 Compartmental models of synapses

Within the context of compartmental modelling, synapses are often modelled as time-varying conductances which transport current across the cell membrane, charging and discharging the membrane capacitance and altering the transmembrane voltage. Figures 5.2-A and 5.2-B show the transient change of membrane voltage and the synaptic current produced by an inhibitory $GABA_B$ synapse.

The magnitude of the synaptic current is given by equation 3.6, reproduced here for convenience,

$$I_{syn}(t) = G_{syn}(t)(V_m - E_{syn}) \tag{5.1}$$

where $I_{syn}(t)$ is the current across the synapse, $G_{syn}(t)$ the conductance of the synapse, V_m the membrane voltage and E_{syn} a voltage source whose value sets the

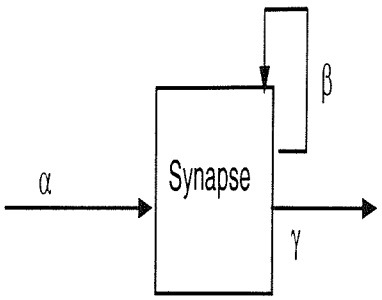


Figure 5.3: Synapse block and associated channels

Input	Output
$\alpha := on$	$\beta := \{t_{del}, on\}$
$\beta := on$	$\beta := \{t_{dur}, off\}$, $\gamma := \{0, on, synapsetype\}$
$\beta := off$	$\gamma := \{0, off, synapsetype\}$

Table 5.4: The synapse block function

membrane voltage for which no current flows across the synapse.

The sequence of events in a biological synapse is as follows; neurotransmitter is released by the presynaptic cell, introducing a delay of the order of ms and triggering the onset of I_{syn} by increasing $G_{syn}(t)$ and the current through the synapse increases the membrane voltage (EPSP, excitatory postsynaptic potential) or decreases it (IPSP, inhibitory postsynaptic potential). The duration of the change in membrane voltage depends on the type of synapse, varying from a few ms for fast (ionotropic) synapses up to hundreds of ms for slow (metabotropic) synapses. After the activation period, the neurotransmitter ceases its action on the synapse and $G_{syn}(t)$ returns to its initial value.

5.2.2 The discrete synapse model

The discrete synapse model is a combinational block with no internal state information. Three aspects of the biological synapse are captured; the *synaptic delay*, the *finite duration* of the synaptic activation and its *efficacy*.

Synaptic delay and activation duration

Figure 5.3 depicts the synapse block and table 5.4 shows the function that it implements.

A complete sequence of events in the MBED synapse is as follows:

- The synapse receives an *on* message (discrete equivalent of neurotransmitter release) at time t on channel α.
- The synapse schedules its delayed activation at $t + t_{del}$ by broadcasting a message *on* with a delay field set to t_{del} through channel β. This accounts for the synaptic delay.
- The synapse receives the message *on* at $t + t_{del}$, which triggers the broadcast of an *on* message through channel γ to notify its activation to the threshold block. The inactivation is scheduled by the broadcasting of an off message through channel β with the delay field set to t_{dur}.
- At $t + t_{del} + t_{dur}$ the synapse receives the off message through channel β and notifies the threshold block of its inactivation by broadcasting a message off through channel γ.

Figure 5.4 shows two examples of the function implemented by the synapse block. In figure 5.4-A, the synapse is activated at $t = 2\ ms$ and $t = 9\ ms$ by the reception of two *on* messages on the α channel. Due to the size of the delay between the two incoming messages, the two consecutive synaptic activations do not overlap in time. The synapse broadcasts the first off message, notifying the end of the first synaptic activation to the threshold block, before the broadcasting of the second *on* message on the γ channel, indicating the start of the second activation. Conversely, in figure 5.4-B, the *on* messages delivered to the synapse through the α channel, arrives with a delay of 3 ms, producing two overlapping synaptic activations.

Synaptic efficacy

In biological neurons, the simultaneous activation of multiple synapses may increase the membrane voltage above the firing threshold and trigger the generation of an action potential [27]. The contribution of each synapse to this change of membrane voltage depends on its functional characteristics (e.g. G_{syn} in expression 5.1) and also on its location in the dendritic tree. Transient changes in membrane voltage due to synapses located far from the cell body, undergo a distance dependent attenuation during its propagation along the dendrites. This effect is introduced in the MBED model by means of a synaptic efficacy factor.

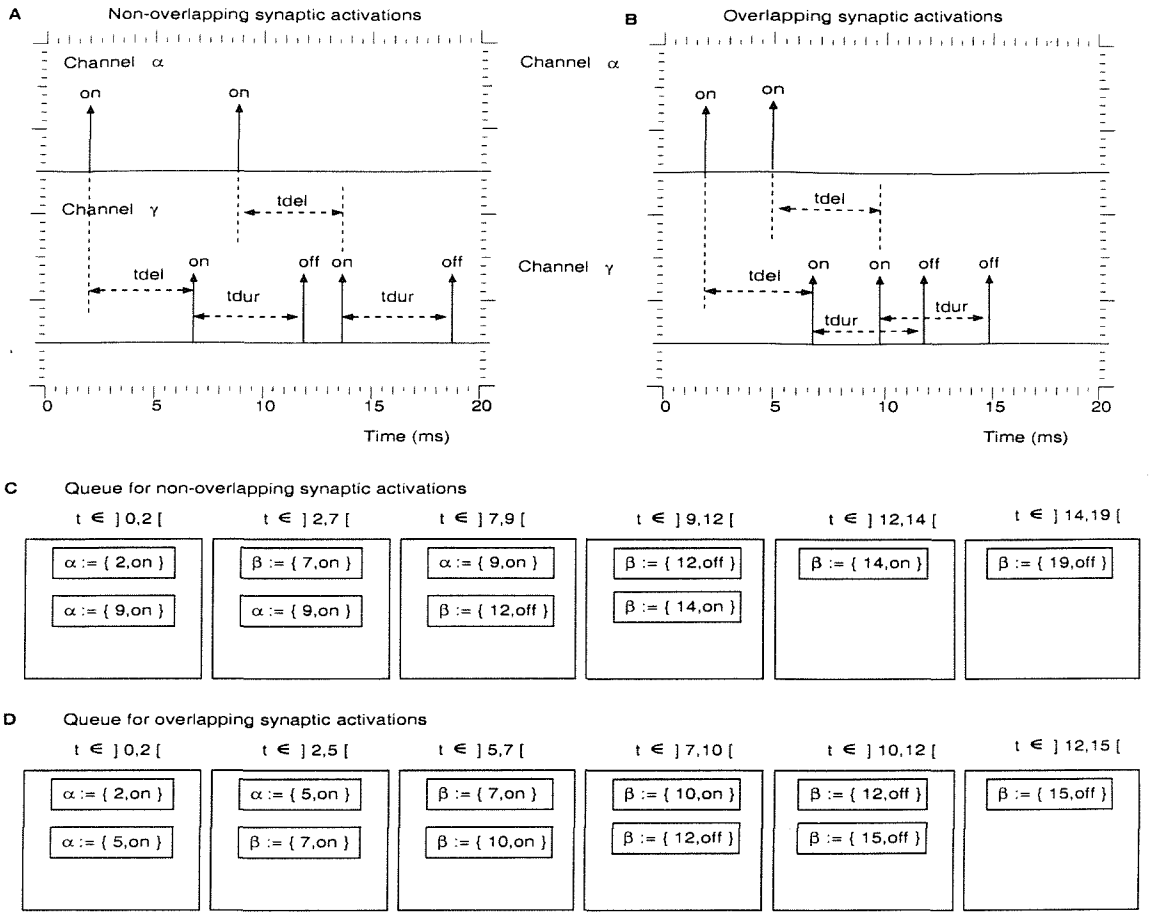


Figure 5.4: Examples of the function implemented by the synapse block. Nonoverlapping (A) and overlapping (B) activations and corresponding queues (C,D)

To demonstrate the impact of the anatomical location of the synapse on its efficacy in generating an action potential in the biological neuron, a standard compartmental model with spherical soma geometry and a single cylindrical dendrite (10 μm in diameter and 500 μm of longitude) was simulated using the Neuron simulator. Figure 5.5-A shows the membrane voltage in the cell body in the case of synaptic input received at the initial segment of the dendrite (synapse-soma distance of 0 μm). When the synaptic input is received at 250 μm from the cell body (figure 5.5-B) the number of action potentials is reduced from 33 to 24. This result indicates that the anatomical location of a synapse influences its synaptic efficacy in generating somatic action potentials.

The discrete synapse model captures the concept of synaptic efficacy by associating a synaptic weight (w_{syn}) to each synapse. Messages of type *on* and off

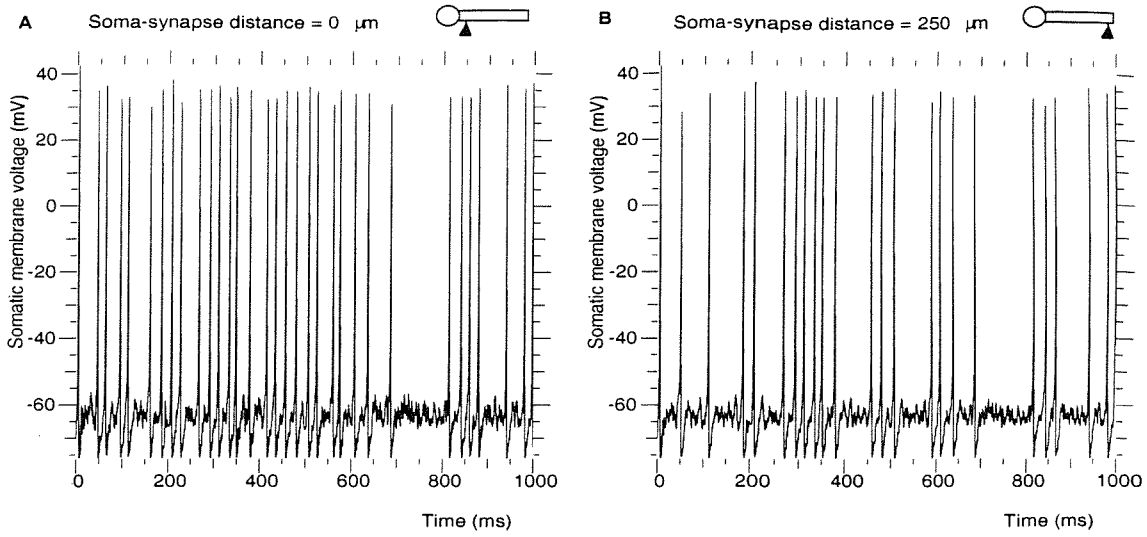


Figure 5.5: Effect of synapse position on synapse efficacy. (A) EPSPs at the initial segment of the dendrite, (B) EPSPs 250 μm away from the cell body

are broadcasted by the synapse to the threshold block to notify synaptic activation and inactivation respectively. The optional field in the message data packet (p in table 5.1) is used in these cases to communicate the efficacy of the synapse to the threshold block. Based on this efficacy, the synapse block updates its discrete estimation of the membrane voltage.

5.3 The threshold block

5.3.1 Nonlinear response of biological neurons

To illustrate the nonlinear neuronal input-output function, a single compartment model (spherical geometry with 100 μm^2 of total membrane surface) incorporating Hodgkin-Huxley Na^+ and K^+ channels was constructed. The transient neuronal response was probed with a set of injected current steps (100 ms in duration) of increasing magnitude within the range 0 - 80 pA. Figures 5.6-A and 5.6-B show the time course of the injected current and the membrane voltage, respectively.

Currents of 20, 40 and $60pA$ increase the membrane voltage from $-65\ mV$ to $-60\ mV$ following a linear current-voltage function. However, a current of 80 pA succeeds in triggering an action potential, leading to a nonlinear increase of the membrane voltage up to 30 mV.

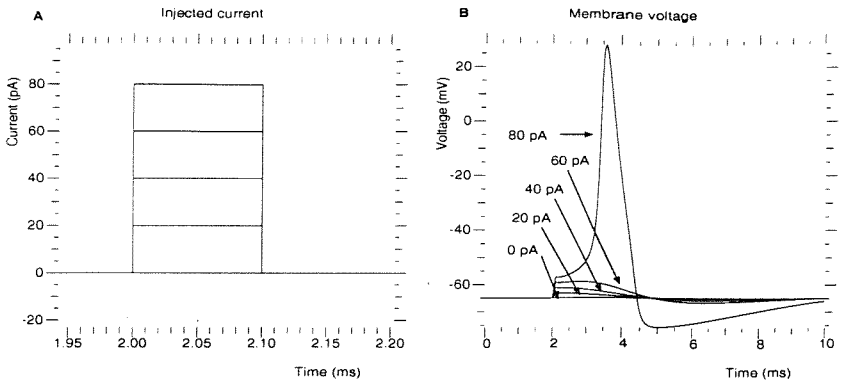


Figure 5.6: Simulation of the neuronal response triggered by pulse-shaped current injection. (A) Time course of the injected current, (B) Membrane voltage

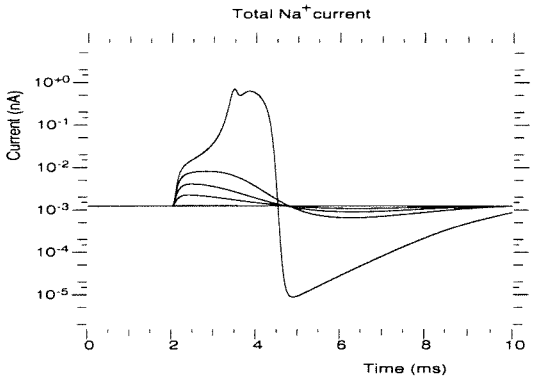


Figure 5.7: Time course of the Na^+ current responsible for the nonlinear onset of the action potential

Nonlinear voltage-gated Na^+ channels are responsible for this threshold effect. The Na^+ channels inject a cationic current into the cell, I_{Na},

$$I_{Na}(t) = G_{Na}(t, V_m)(V_m - E_{syn}) \tag{5.2}$$

where $G_{Na}(t, V_m)$ is the conductance of the channel, V_m the membrane voltage and E_{syn} the voltage at which no current flows through the channel. Note that the conductance of the channel $G_{Na}(t, V_m)$ is a function of time and membrane voltage.

The total current through the Na^+ channels, underlying the action potential, is shown in figure 5.7. It increases nonlinearly for an injected current greater than 60 pA.

Input	Action | Output
$\gamma := on$	$w_{sum} += w_{syn}$ $w_{sum} >= th_e$? true: | $\epsilon := \{0, on\}$ $w_{sum} <= th_i$? true: | $\epsilon := \{0, off\}$
$\gamma := off$	$w_{sum} -= w_{syn}$ $w_{sum} >= th_e$? true: | $\epsilon := \{0, on\}$ $w_{sum} <= th_i$? true: | $\epsilon := \{0, off\}$

Table 5.5: The threshold block state machine

5.3.2 The core threshold block

The threshold block in the MBED neuron model captures two experimentally observed effects; the integration of synaptic activity by the dendritic tree [22, 135, 136] and the threshold effect introduced by nonlinear voltage gated channels [21, 25].

It is a state machine and table 5.5 shows the transition table for the block. Its internal state variable, w_{sum}, stores a weighted sum of active synapses,

$$w_{sum_j} = \sum_{i}^{S} \alpha_i w_{syn_i} \tag{5.3}$$

$$\alpha_i = \left\{ \begin{array}{ll} n & if\ synapse\ i\ was\ activated\ n\ times \\ 0 & if\ synapse\ i\ is\ inactive \end{array} \right\} \tag{5.4}$$

where S is the number of synapses onto neuron j, w_{syn_i} is the synaptic weight of the i^{th} synapse and α_i takes the value 0 if the synapse is inactive and n if the synapse was activated by n incoming *on* messages.

Each synaptic event triggers an update of w_{sum} in the threshold block, after which, its value is compared against the excitation (th_e) and the inhibition (th_i) thresholds. If the condition ($w_{sum} \geq th_e$) holds, an *on* message is broadcast to the burst generator block through the ϵ channel in order to trigger a burst of action potentials. If, alternatively, the condition ($w_{sum} \leq th_i$) is evaluated true, an off message is broadcast to the burst generator block through the ϵ channel to truncate an ongoing burst. Note that these conditions are mutually exclusive given that, in

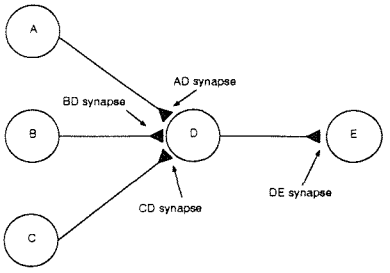


Figure 5.8: Small network used to illustrate the internals of the model

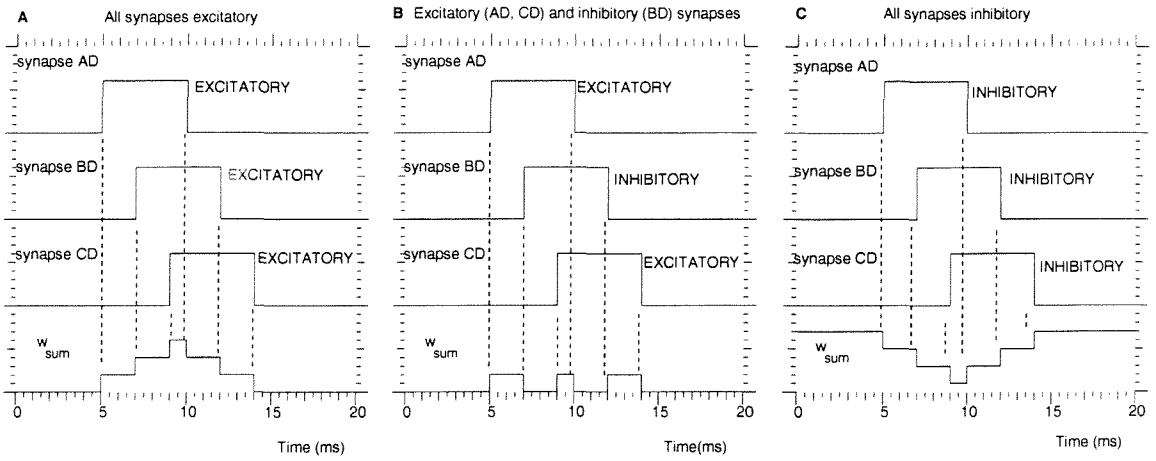


Figure 5.9: Integration of synaptic input by the threshold block. (A) All synapses excitatory, (B) mixed excitatory-inhibitory, (C) all inhibitory

order to confer biological realism to the model, the inequality $th_i < th_e$ must be observed when setting model parameters.

The circuit of four neurons shown in figure 5.8 is used to illustrate the function implemented by the threshold block. Figure 5.9 shows the synaptic events (three uppermost traces in each plot) and the value of the internal state variable w_{sum} in neuron D (bottom trace). Synaptic events are signaled by an assertion, indicating synaptic activation, and deassertion, to indicate deactivation.

Three cases are considered; in figure 5.9-A, all synapses onto neuron D (AD,BD and CD) were configured as excitatory with unit weight ($w_{syn} = +1$). To generate figure 5.9-B, synapses AD and CD remained configured as excitatory connections with unit weight, ($w_{syn} = +1$) but synapse BD was configured as inhibitory ($w_{syn} = -1$). For figure 5.9-C, all three synapses were configured as inhibitory connections ($w_{syn} = -1$).

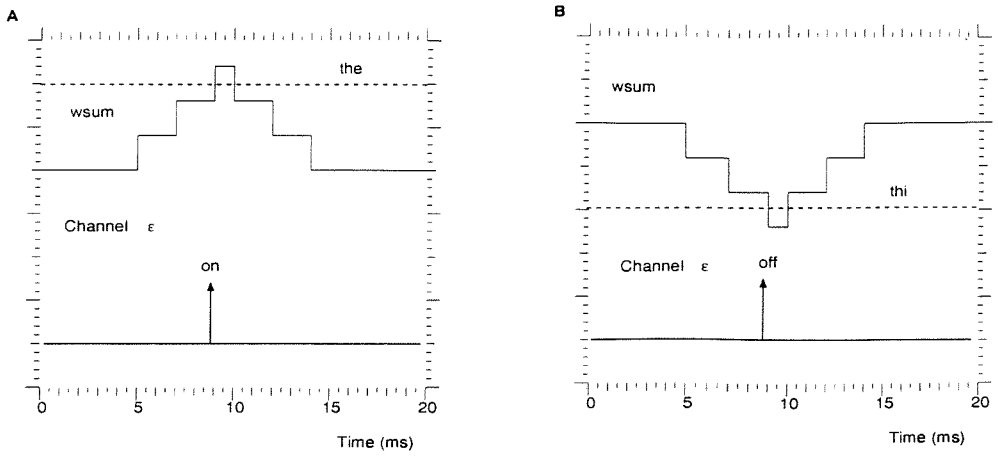


Figure 5.10: An increase of w_{sum} above th_e or a decrease below th_i triggers message broadcasting on channel ϵ.

Figure 5.10 shows the output of the threshold block corresponding to the examples of figures 5.9-A and 5.9-C. The upper traces reproduce the time evolution of w_{sum} whereas the bottom trace contains delta functions indicating the broadcasting of messages on channel ϵ. Their types are indicated by the associated labels.

5.4 The burst generator

Figure 5.6 shows the single spike obtained with a compartmental model incorporating voltage-gated Na^+ channels. An increase in the duration of the current pulse from 100 μs to 500 ms induces a change in the neuron response to a burst of 25 action potentials (see figure 5.11).

The burst generator block introduces the concept of burst in the event-driven model by implementing message streams and broadcasting them on the output channel α. These are interpreted at the receiving end (the synapse blocks of postsynaptic cells) as notifications of presynaptic action potentials. Upon reception of an *on* message either on the ϵ or ζ channels, the burst block outputs a stream of messages of type *on* through its output channel α, the number of messages per burst being determined by the parameter N_{burst}. The reception of an off message on the ϵ channel before the end of the output stream results in its premature truncation.

The burst generator consists of a three state (*on*, off and ref) automaton. It also contains the internal state variable n_{burst} which stores the number of remaining

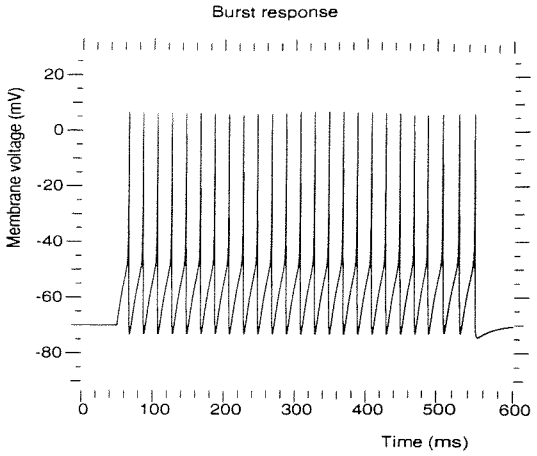


Figure 5.11: Burst of action potentials elicited by a current pulse of 500 ms

messages in an ongoing burst. Table 5.6 shows the state transition table for the burst generator block. Entries adhere to the format (next state | action | output).

A typical sequence of events leading to the generation of a burst is as follows:

- The state machine remains in its initial state, off and $n_{burst} = N_{burst}$, until the arrival of an *on* message input at time t. This causes its state to change to *on* (indicating the onset of an action potential) and triggers the broadcasting of the first *on* message of the outgoing burst.

- At $t + t_{ap}$, the state machine changes to state ref (the action potential has finished and it enters the refractory state).

- At $t + t_{ap} + t_{ref}$ the refractory state ends and the counter n_{burst} is decreased in one unit. The state machine returns to state *on* (initiating the next potential in the burst) if $n_{burst} > 0$. Alternatively, if $n_{burst} == 0$, it changes to off and resets $n_{burst} = N_{burst}$.

Note that, setting $N_{burst} = 1$, the neuron generates single action potentials rather than bursts.

Figure 5.12 shows two examples of the behaviour of the burst generator. Four traces are shown in each plot; they correspond, in succession from top to bottom, to a train of deltas indicating the sequence of message arrivals on channel ϵ, the time-evolution of the state of the burst block and the train of outgoing messages on channel α and value of the state variable n_{burst}. In 5.12-A, the arrival of an *on*

Burst generator					
Current state	Next state | Action | Output				
	Input				
	$\epsilon := on$	$\epsilon := off$	$\eta := off$	$\eta := r\_off$	$\zeta := on$
on	on | - | -	on | $n_{burst} = 0$ | -	ref | - | $\eta := \{t_{ref}, r\_off\}$	on | - | -	on | - | -
ref	ref | - | -	ref | $n_{burst} = 0$ | -	ref | - | -	$n_{burst} - 1 == 0$? true: off | $n_{burst} = N_{burst}$ | - false: on | $n_{burst} - = 1$ | $\eta := \{t_{ap}, off\}$	ref | - | -
off	on | - | $\alpha := \{0, on\}$, $\eta := \{t_{ap}, off\}$	off | - | -	off | - | -	off | - | -	on | - | $\alpha := \{0, on\}$, $\eta := \{t_{ap}, off\}$

Table 5.6: The burst generator state machine

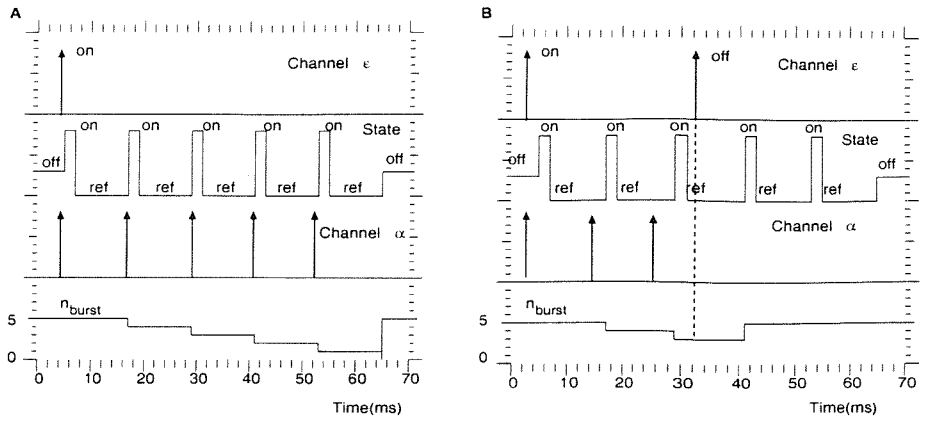


Figure 5.12: Example of burst generation (A) and truncation of a burst due to inhibition by an off message (B).

message at $t = 4\ ms$, triggers the broadcasting of a burst of 5 output messages ($N_{burst} = 5$). In 5.12-B, the burst is truncated to 3 messages upon the reception of an off message at $t = 35\ ms$ which resets the state variable n_{burst} to 0.

The burst block can also be configured to generate infinite length bursts. By setting the parameter N_{burst} to a negative value, the condition $n_{burst} == 0$ never holds. In this case, the train of outgoing messages can only be finalized by an inhibitory off message on channel ϵ which leads to burst truncation.

5.5 Axonal delay

Action potential generation in biological neurons is followed by the propagation along the neuronal axon. Figure 5.13-A shows the results of the simulation of an action potential propagating along an axon of 20 μm in diameter and 20 mm in length. The attenuation of the propagating spike is eliminated by the regenerative effect of homogeneously distributed HH Na^+ and K^+ channels.

The velocity of propagation, as calculated from figure 5.13-A, is 2.5 m/s. The reduction of the axonal diameter to 10 μm decreases the velocity of propagation to 1.6 m/s (from figure 5.13-B). For axons of a few mm in length, the arrival of the action potential to the most distal parts of the axon will introduce a latency of several ms.

Delays derived from the finite axonal velocity have been shown to be important in the generation of EEG oscillations, as indicated by the EEG models developed by

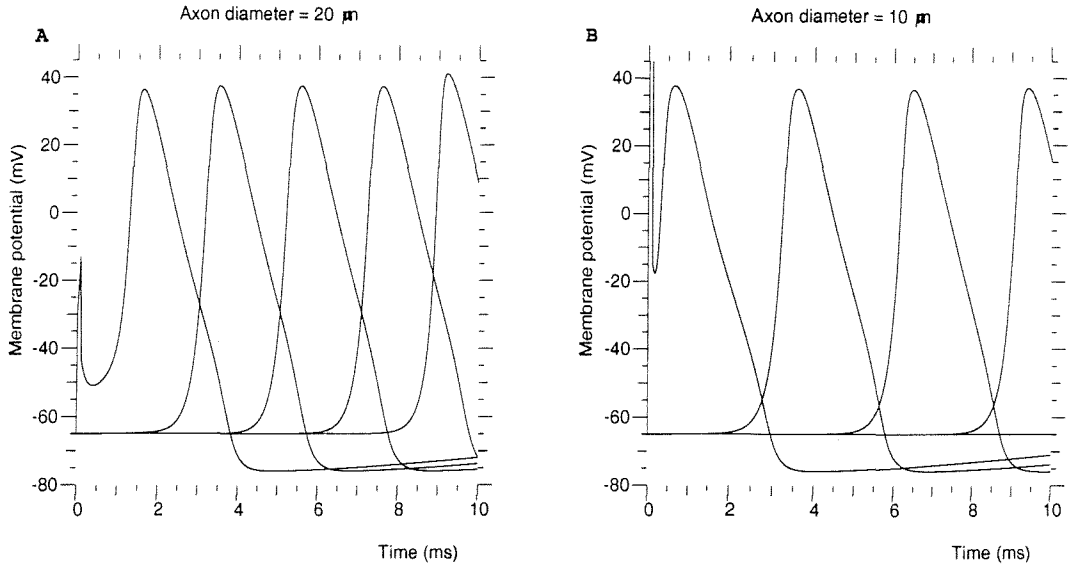


Figure 5.13: Action potential propagation in an axon of 20 μm (A) and 10 μm (B) in diameter

Nunez [132]. The axonal latency is captured in the event-driven model by the synaptic delay (t_{del}), which accumulates the delay involved in the release of the neurotransmitter and the latency due to axonal propagation of the action potential. Figure 5.14 shows an example based on the small circuit of figure 5.8. The four traces in 5.14-A correspond (top to bottom) to the the outgoing messages from neuron D, the DE synapse activation/deactivation state, the time-evolution of the w_{sum} state variable in cell E and the state of its burst block. Figure 5.14-B plots the output of neuron D and three traces corresponding to the output of cell E as obtained for three different values of t_{del}.

5.6 The oscillator

The oscillator block is a two state machine which implements a free-running oscillator. When activated by setting parameter $t_{osc} > 0$, it broadcasts an *on* message to the burst generator every t_{osc} time units. The first message in the sequence is broadcasted at $t = t_{\phi}$. Table 5.7 shows its state transition table.

Figure 5.15 demonstrates the function of the oscillator block when configured with $t_{osc} = 150\ ms$ and $t_{\phi} = 250\ ms$. The upper and middle traces show the state of the burst generator and oscillator blocks respectively. The bottom trace is the

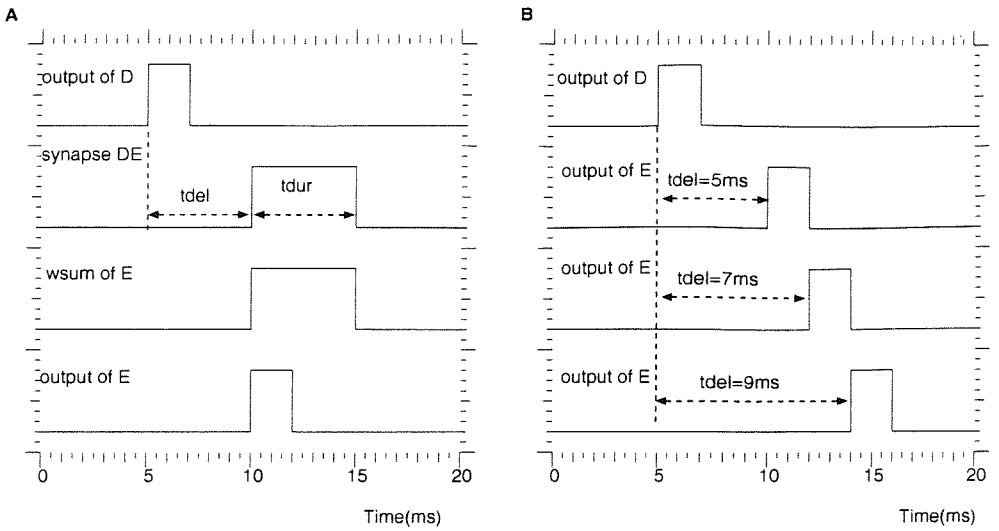


Figure 5.14: The axonal and synaptic delay introduced between the action potential in cell D and the action potential in cell E as set by parameter t_{del} of synapse DE

Current state	Next state | Output
	$\delta := \{t_{osc}, change\}$
on	$off \mid \delta := \{t_{osc}, change\}, \zeta := \{0, on\}$
off	$on \mid \delta := \{t_{osc}, change\}, \zeta := \{0, on\}$

Table 5.7: The oscillator state machine

sequence of *on* messages broadcasted by the oscillator on channel ζ. At intervals of 150 ms, it changes state and broadcasts a message to the burst generator which, when configured with $N_{burst} = 3$, generates a burst of three action potentials.

5.7 Coding schemes implementable with the MBED model

Several schemes for information coding in neural aggregates have been proposed [137, 138]. The two most prominent are *rate* codes and *temporal* codes [139]. The MBED model was simulated to validate its suitability for implementing these codes. In particular, it is shown that the sublinear summation of inputs ,often captured in rate coding models, and correlation detection, which underlies temporal codes, can be implemented with the MBED neuron.

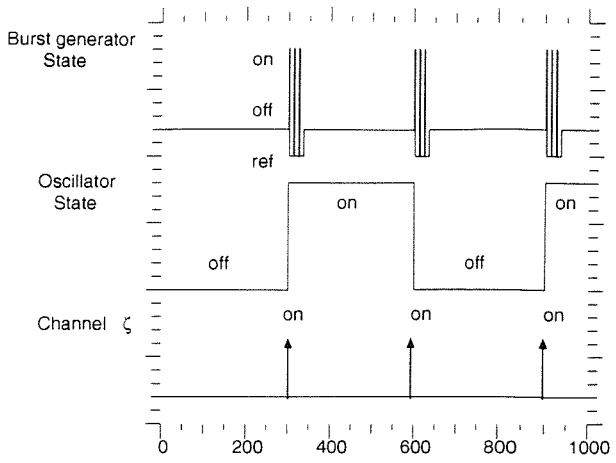


Figure 5.15: Oscillator block

5.7.1 Rate coding and sublinear synaptic summation

Rate coding is based on the assumption that the spiking *frequency* of the cell is the information carrier. Thus, the timing of individual spikes is considered irrelevant and usually not captured by these neuronal models. There is experimental evidence of this type of information coding in neurons located close to sensory inputs [27].

Neuron models used for rate coding map input firing rates into output firing rates. A common form for this mapping implements a nonlinear parameterized function of the input firing rates as shown in figure 5.16. This sigmoidal mapping was described in Chapter 3 within the context of the Perceptron model.

In biological neurons, such a sigmoidal input-output function results from the refractory period (approx. 10 ms), which limits the maximum firing rate of the cell. A biophysical neuron model constructed with a single compartment soma attached to two passive dendrites was constructed. The cell body included voltage-gated Na^+ and K^+ channels with Hodgkin-Huxley dynamics.

Figure 5.17-A shows the number of action potentials generated within a time window of 100 ms in response to a train of excitatory postsynaptic potentials. As the synaptic conductance is increased from 20 pS to 200 pS, the number of action potentials reaches a maximum of 8. This is the upper bound of the firing rate as imposed by the refractory period. The spiking rate versus synaptic conductance plot resembles the commonly used sigmoidal function.

Figure 5.17-B shows the results obtained in a similar experiment using the MBED model. A single neuron with an excitation threshold of $th_e = 100$ received 2000

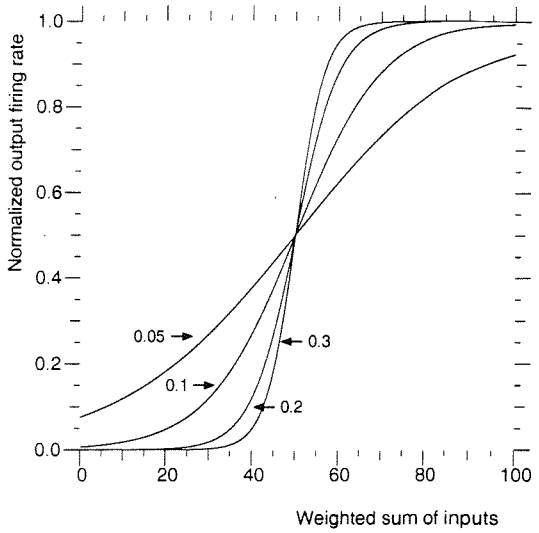


Figure 5.16: Sigmoid function as used for rate coding neuron models

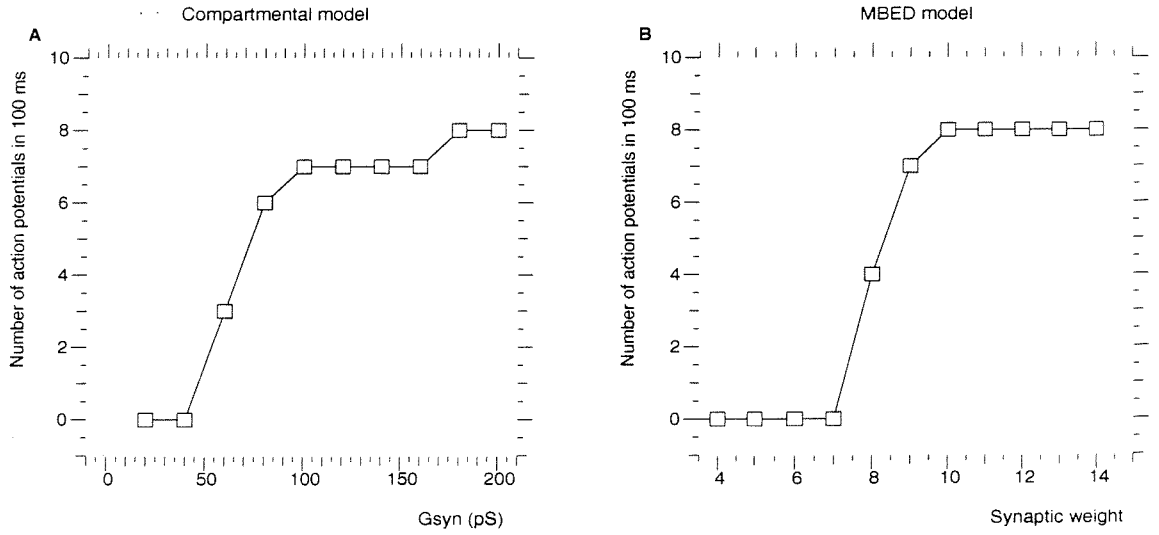


Figure 5.17: Sublinear increase of firing rate with an increase in synaptic input. Compartmental model (A) and event-driven model (B)

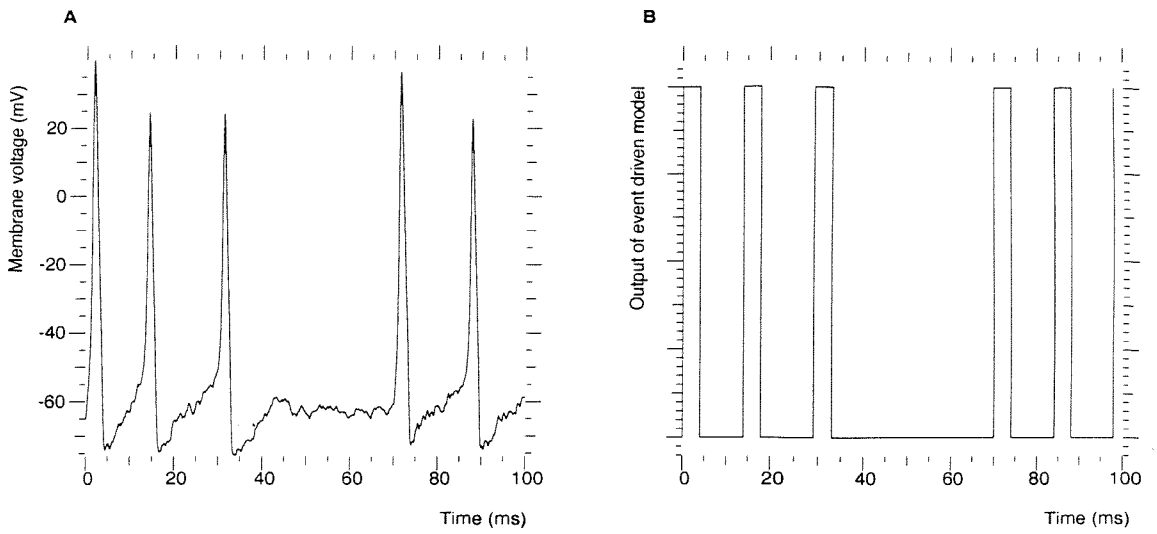


Figure 5.18: (A) Correlation detection in a compartmental model, (B) Correlation detection in the event-driven model

synaptic activations over an interval of 100 ms, with onset timing given by a uniform distribution in the range (0..100 ms). Synapses were configured with $t_{dur} = 5\ ms$.

Figure 5.17-B shows the number of action potentials obtained as a function of the synaptic weight (w_{syn}). As the synaptic efficacy increases, w_{sum} is more likely to reach the excitation threshold and generate an action potential. However, its maximum firing rate is limited by the refractory period ($t_{ref} = 10\ ms$) in the burst generator block. The result is a sigmoidal spiking rate versus synaptic weight function, analogous to the biological sublinear summation of inputs of figure 5.17-A.

5.7.2 Temporal coding

Temporal coding makes use of the timing of individual action potentials as the carrier of information between neurons. There is some experimental evidence of this type of neuronal coding in the central nervous system (e.g. in the visual cortex [41, 42]). In this context, neurons are often modelled as correlation detectors which generate action potentials when their inputs are correlated in time. The firing rate is no longer the relevant parameter, rather, the timing of individual spikes is thought to support neural function.

Figure 5.18-A shows the time course of the somatic membrane voltage in the two-dendrite compartmental single neuron model used in the previous Section. A train of 2000 EPSPs was generated, triggering several action potentials when the

subset of simultaneously active synapses was sufficient to increase the membrane voltage up to its firing threshold.

Figure 5.18-B shows the result of the simulation of an MBED neuron receiving the same train of excitatory synaptic activations. The neuron was configured with $th_e = 10$ and synapses with $w_{syn} = 1$ and $t_{dur} = 5\ ms$. The timing of individual spikes in the MBED simulation of figure 5.18-B coincides within the interval 0<t<95 ms with those in the compartmental model of figure 5.18-A, with a timing error $e < 5\ ms$.

However, at $t = 100\ ms$ the MBED model predicts an action potential not seen in the compartmental model. The extra spike is the result of the simplifications inherent to the discrete representation of a neuron when compared to biophysical models. The following sections describe some examples of these differences.

5.8 Comparative analysis

The simplifications involved in the construction of discrete neuron models are responsible for deviations from the dynamics of compartmental models. Studies dealing with large populations of simplified neuron models have demonstrated that collective dynamics observed in abstract representations of neural populations are capable of displaying realistic behaviour. For instance, Wright [46] has shown that 40 Hz gamma oscillations arise from pools of continuous neurons consisting of basic computational units implementing gain and lag operations. Nevertheless, several differences between the discrete and the continuous approaches were studied in order to asses the validity of MBED network simulations; voltage dependent efficacy of GABA synapses, NMDA channels and firing rate adaptation.

5.8.1 Efficacy of $GABA_A$ synapses

GABA is the major inhibitory neurotransmitter within the central nervous system [6]. Its release activates inhibitory synapses and triggers the onset of the synaptic current. As indicated in equation 5.1, the magnitude of the instantaneous synaptic current depends linearly, in a first approximation, on $(V_m - E_{res})$, E_{res} taking the value $-70mV$ for the $GABA_A$ synapse subtype.

A compartmental model was simulated to study the effect of the average membrane voltage on the efficacy of $GABA_A$ synapses. The model consists of a passive compartment incorporating two types of synapses: excitatory synapses and

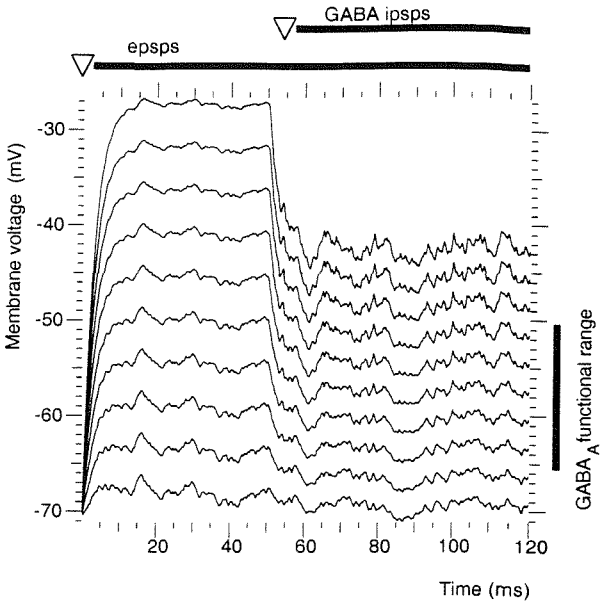


Figure 5.19: Effect of the average membrane voltage on the efficacy of GABA synapses.

$GABA_A$ inhibitory synapses. Due to the absence of Na^+ and K^+ channels, action potentials can not be generated.

The membrane resting potential was scanned in the range ($-70\ mV$ to $-25\ mV$) in 5 mV steps. At $t = 0\ ms$, a train of EPSPs was generated by random activation of the excitatory synapses. As a result, an increase of the average value of Vm within the interval $t = 0 - 50\ ms$ can be seen in figure 5.19. At $t = 50\ ms$, concomitantly with the train of EPSPs, a sequence of IPSPs was triggered by random activation of $GABA_A$ inhibitory synapses. The resulting effect was a decrease in membrane potential in the order of 0 to 20 mV within the interval $t = 50 - 120\ ms$.

For values of V_m close to E_{res} (bottom traces), the magnitude of the charge injected into the cell is close to 0 and the change in membrane voltage due to this charge is unnoticeable. As the resting potential was increased (upwards in figure 5.19) the voltage decrease induced by $GABA_A$ activation becomes more marked (up to 20 mV).

These simulations show that the efficacy of the inhibitory effect of a $GABA_A$ synapse depends on the value of V_m at the time of its activation. Such dependency was not introduced in the MBED model for the following reason; biological neurons incorporate non-linear conductances which effectively implement a threshold function. Values of the membrane voltage above this threshold, trigger an action

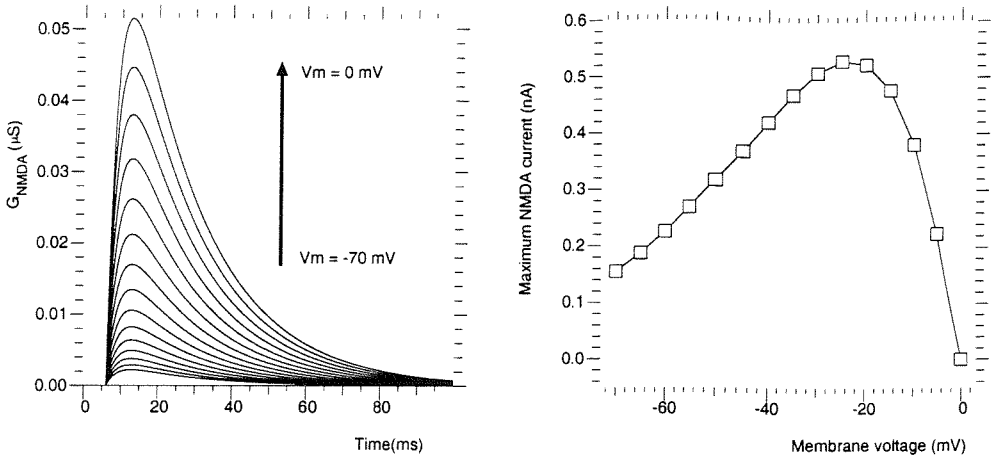


Figure 5.20: (A) Conductance of NMDA channels during voltage clamp, (B) Maximum current through NMDA channels during voltage clamp.

potential. During an action potential, synaptic activity has no effect on the stereotyped pattern of a spike. Thus, the functional interval of membrane values for $GABA_A$ synapses corresponds to the subset of the traces seen in figure 5.19 within the range limited by the resting voltage and the firing threshold . Within this subset, the variability of the efficacy of the train of $GABA_A$ IPSPs is reduced to approximately 5 mV.

If required, equivalent effects could be introduced in the MBED model (see Chapter 9) by making w_{syn} (the synaptic weight) of inhibitory synapses a function of w_{sum} (the weighted sum of inputs calculated by the threshold block). However, the limited range within which $GABA_A$ efficacy variations are physiologically realistic, led to the simplification that synaptic efficacy remained constant.

5.8.2 NMDA synapses

NMDA receptors constitute a class of neurotransmitter gated channels which are believed to be involved in learning [140]. The synaptic current for these channels is given by [6],

$$I_{syn}(t) = B(V_m)G_{syn}(t)(V_m - E_{NMDA}) \tag{5.5}$$

where $B(V_m)$ is an increasing function of the membrane voltage, V_m.

For the synapse to modify V_m when activated by the release of neurotransmitter, $B(V_m) > 0$ must hold. Figure 5.20-A shows the value of the product $B(V_m)G_{syn}(t)$

as obtained from the simulation of a voltage clamp experiment in a single compartment model with NMDA channels. It illustrates the dependence of the NMDA conductance on V_m. At $t = 13\ ms$, the synaptic conductance is at its maximum. The peak conductance is shown to increase with V_m (upwards in figure 5.20-A). Figure 5.20-B plots the maximum synaptic current as a function of V_m. In an NMDA synapse, the magnitude of the current injected into the post-synaptic cell, which is directly related to its efficacy, is a function of V_m through the factors $B(V_m)$ and $(V_m - E_{NMDA})$.

As in the case of $GABA_A$ synapses, the MBED model could be modified to make the synaptic efficacy, w_{syn}, a function of w_{sum}. However, the network models studied in Chapters 7 and 8 do not incorporate learning. Thus, the implementation of NMDA channels was not necessary.

5.8.3 Firing rate adaptation

The MBED model is able to generate bursts of action potentials as observed in biological cells. The model assumes that the delay between two action potentials in a burst is not modified during the simulation and is specified by the parameter t_{ref}. Several types of classes of cells (e.g. pyramidal neurons) have been shown to adapt their firing rate as a function of past activity [141].

Figure 5.21 shows the results of the simulation of a compartmental model incorporating K^+ channels of the type I_{AHP} during a current pulse of 0.6 nA injected into the cell for an interval of 500 ms. Figures 5.21-A and 5.21-B show the membrane voltage and I_{AHP} conductance, respectively, for a model with no I_{AHP} channels. In figures 5.21-C and 5.21-D, I_{AHP} channels were added with a conductance density of 100 $\mu S/cm^2$. The conductance was increased in figures 5.21-E and 5.21-F to 200 $\mu S/cm^2$.

Comparison of figures 5.21-A, 5.21-C and 5.21-E indicates that, as the density of channels and the magnitude of I_{AHP} current increases, the number of action potentials per burst decreases while increasing their inter-spike delay.

For the MBED model to be used to simulate neurons incorporating I_{AHP} channels, the value of t_{del} should be adjusted during the simulation. However, there is no experimental evidence of spike rate adaption in the locomotory system of C. elegans. Moreover, for the network model of the piriform cortex presented in Chapter 8, neurons were configured as single spike cells. Thus, spike rate adaptation within bursts need not be implemented.

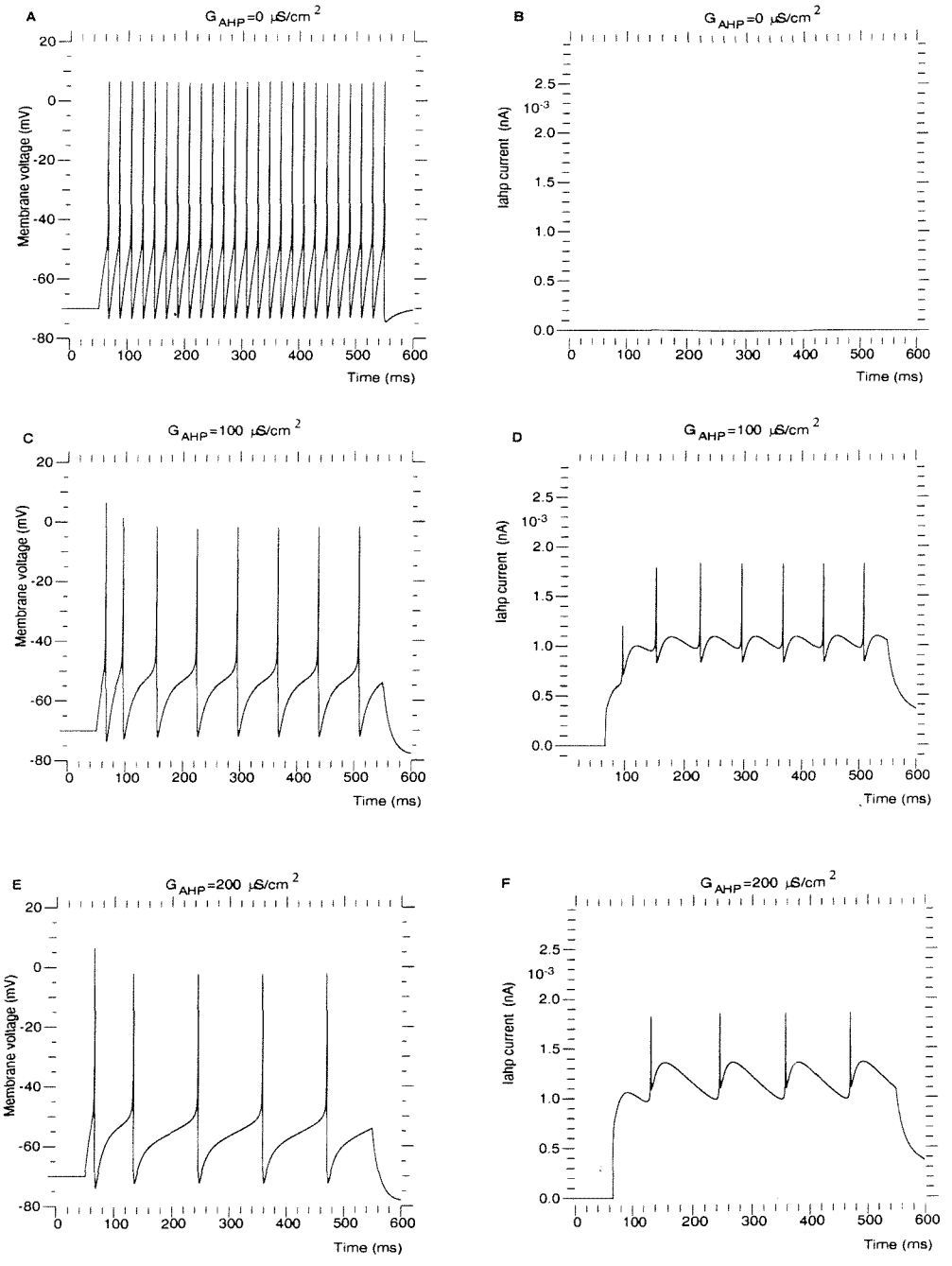


Figure 5.21: A,C,E - Membrane voltage for different values of G_{AHP}, B,D,F - Total I_{AHP} current.

Chapter 6

The MBED simulator

This Chapter describes the software developed for efficient message-based event-driven simulation of networks of MBED neurons. Firstly, an overview of the tool is provided, with emphasis on general design issues and the user interface. Secondly, the internal structure and algorithms are described. Several techniques have been utilized for improved simulation speed and memory use; a look-up-table (LUT) based priority queue provides O(1) queue insertion times irrespective of queue size and O(1) extraction latencies within the range of queue sizes typically encountered in large scale simulations. Memory efficient data structures for the storage of synaptic parameters and neuron identifiers, in addition to the use of an optimized algorithm for dynamic allocation of new messages, reduce memory consumption.

Finally, the performance of the simulator is tested using a uniformly connected network of MBED neurons including both excitatory and inhibitory synapses. The impact of several network parameters on simulation performance is studied. In particular, the effect of the relative proportion of the two types of synapses, the neuronal threshold, the number of synapses per neuron and the size of the network is explored. A more realistic topology, consisting of a model of the piriform cortex, is also considered.

6.1 Overview of the simulator

The simulator was implemented using the C++ programming language and embedded within Yorick [142]. This is a freely available numerical package with user interface and capabilities similar to Matlab. Yorick provides an interpreted and

mathematically orientated scripting language which can be used interactively and in batch mode. The default command set allows the creation of vectors and multidimensional matrices, implements a large number of operations on these mathematical structures and provides a variety of visualization routines. Yorick also supports functions and flow control statements following ANSI C syntax.

A typical interactive session starts invoking Yorick from the UNIX prompt.

```
$ yorick

 Copyright (c) 1996.  The Regents of the University of California.
 All rights reserved.  Yorick 1.4 ready.  For help type 'help'

>
```

The > prompt indicates that Yorick is ready to accept interactive commands. A 7-element vector and a 2×5 matrix can be created by

```
> v = [1,5,3,6,7]
> m = [[1,2,3,4,5],[6,7,8,9,10]]
```

Commands are invoked using the ANSI C syntax for function calls. In the first line of the following example, the command *sum* is used, returning a scalar value with the sum of the elements of the vector v. The return value is stored in the variable z and visualized with the *print* command. The session is finished with *quit* and returns the user to the UNIX prompt.

```
> z  = sum (v)
> print (z)
22
> quit

$
```

Yorick also supports a non-interactive (batch) mode invoked as

```
$ yorick -batch filename
```

where *filename* is a file containing the commands to be executed by Yorick.

The advantages obtained by the integration of a standard package and the simulator core are,

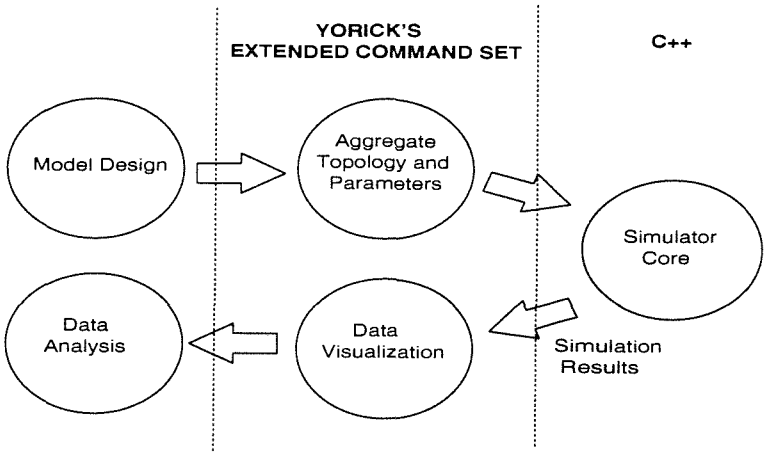


Figure 6.1: Overview of the simulation tool

- Simplification of the data visualization and analysis. The standard tools available in the package can be used to process the results of the simulation (e.g. the Fourier transform is a built-in function and allows frequency domain analysis of EEG simulations). Routines for 2D and 3D static and animated visualization of matrices are also provided by the package and are utilized for post-simulation data analysis.

- The integration of the simulator within a standard numerical package makes it possible to automate some of the tasks associated with the simulations. This is the case during studies of network dynamics, when a systematic search of a region of the parameter space is often needed. The possibility of using a scripting language to control the parameter search allows fast implementation of different search algorithms.

- The portability of the simulator is increased reusing the input/output functions included in the numerical package. As many packages have been ported to several operating systems, the programmer does not need to recode (e.g. the plotting routines) for cross-platform portability. In the case of the package Yorick, which has been enhanced with the addition of the simulator core, versions exist for the Unix, Windows and MacOS platforms.

Yorick was designed for easy customization of its command set. New commands can be implemented in C, C++ and Yorick's language to provide the functionality required by the user. Taking advantage of this feature, a number of commands were

Command	Purpose	Parameters	SNDS / LNDS
initevent(conf,con)	Topology creation	conf : neuron configuration con : connectivity matrix	SNDS
createpiriform(z)	Topology creation	z : Topology vector	LNDS
initmodels (m)	Synaptic models	m : vector of synaptic models	LNDS
simuevent(t)	Start simulation	t : stop time	Both
deleteall()	Deallocates memory	-	Both

Table 6.1: Command set provided by the simulator

implemented to extend the command set available in the default installation of Yorick. They constitute the user interface to the MBED simulator (figure 6.1) and support the specification of the topology of the network, initialization of neuronal and synaptic parameters and simulation control, providing access to functions within the C++ simulator core.

Table 6.1 lists these commands. As will be described in section 6.4.1, the simulator can be compiled with one of two data structures, the small networks data structure (SNDS) and the large network data structure (LNDS). Table 6.1 indicates the type of network appropriate for each command. The following sections describe the steps involved in a typical simulation.

Data visualization and analysis makes use of the commands provided by Yorick's command set.

6.2 Exemplar interactive session

An interactive session is started invoking Yorick from the UNIX prompt

```
$ yorick

 Copyright (c) 1996.  The Regents of the University of California.
 All rights reserved.  Yorick 1.4 ready.  For help type 'help'

>
```

and is followed by the specification of the topology of the network, the execution of the simulation and the visualization of the results.

6.2.1 Topology specification

The topology of the network and its neuronal and synaptic parameters can be specified in two ways:

- Two matrices can be created to describe the topology and the neuronal configuration. The topology of an N neuron aggregate is contained in a $N \times N$ connectivity matrix, each element corresponding to a possible connection in the network. Since the dynamics of a synapse are totally specified by three synaptic parameters ($t_{del}, w_{syn}, t_{dur}$), each entry in the matrix is a 3-element vector. A second matrix is necessary to contain neuronal parameters. The seven parameters needed by each neuron (th_e,th_i,t_{osc},N_{burst},t_{ap},t_{ref},t_{phi}) make up a $N \times 7$ matrix.

  The connectivity (*con* in the example below) and the configuration (*conf*) matrices for a 3 neuron network can be created using Yorick's environment as

```
> synapse1 = [1,1,1]
> nosynapse = [0,0,0]
> con = [[synapse1,nosynapse,nosynapse],
         [synapse1,nosynapse,nosynapse],
         [synapse1,nosynapse,nosynapse]]
> typicalneuron1 = [10,-10,0,1,1,10,2]
> typicalneuron2 = [20,-10,0,1,1,10,2]
> conf = [typicalneuron1,typicalneuron1,typicalneuron2]
```

  The command *initevent*() can now be used, with *conf* and *con* as its parameters, to instruct the event-driven simulator to instantiate its internal data structures according to the specified topology.

```
> initevent (conf,con)
```

  At this point the internal data structures have been created and the neurons are initialized. The network is ready to start the simulation.

- An alternative way to create the network avoids the specification of individual connections by describing the aggregate with a set of connectivity rules. This approach is specially suited for large scale simulations where the number of

connections is in the order of tens of millions. This is also consistent with the type of experimental data available from studies of cortical anatomy, where only the statistics of the connectivity are known. Typically, these rules will be parameterized and the creation of the network will only require the selection of values for these parameters (e.g. probability of establishing an excitatory synapse between two neurons at a distance d).

In the case of the MBED model of the piriform cortex which will be presented in Chapter 8, a new command was implemented, *createpiriform()*, which accepts the topological parameters and instantiates the network.

A vector describing the topology of the network can be created as

```
> network = [20, ... , 250,250,100,150,200, ... ]
```

The first entries of the vector specify the characteristics of the pool of pyramidal cells. The first seven elements correspond to the configuration of the neurons in the pyramidal pool, starting by the excitation threshold (set to 20 in the example above). The following parameters specify a grid of 250×250 neurons where each cell establishes 100 synapses with other pyramidal cells, 150 synapses with fast inhibitory ($GABA_A$) cells and 200 with slow inhibitory ($GABA_B$) neurons. The parameters needed by the cortical model will be described in Chapter 8.

The simulator is instructed to create its internal data structures according to the specified network topology by using the command $createpiriform()$ with the vector *network* as its parameter.

```
> createpiriform (network)
```

Of the two methods described for topology specification, the first approach (i.e. using the *initevent()* command) is adequate for small networks as it requires the specification of each connection as parameters. However, the implementation of simulations is accelerated by the fact that the generic command $initevent()$ can be used for any topology.

In the second alternative, the creation of new commands (e.g. $createpiriform()$) is required to support new network topologies. Since only the parameters of the connectivity rules have to be passed to the simulator, it provides a compact representation of the network which makes it adequate for large simulations.

Data	Visualization format
State of the burst generator	Waveform/ Matrix
Value of w_{sum} in the threshold block	Waveform/ Matrix
Instantaneous number of messages in the priority queue	Waveform

Table 6.2: Formats for the visualization of simulation results.

6.2.2 Simulation control

Following the creation of the network, the command *simuevent*() is used to initiate the simulation. Its parameter specifies the number of time steps to execute and the results are contained in the array returned by the function call.

```
> results = simuevent (100)
```

6.2.3 Visualization of results

Throughout the simulation, the occurrence of events leads to changes in the state vectors and variables of the neurons in the aggregate. These are logged and returned by the command *simuevent*() as a $M \times 3$ matrix, where M is the number of variable updates logged. Each item corresponds to a 3-element vector, whose elements are the time point when the variable change occurred (first vector element), a number which identifies the variable and the neuron to which it belongs (second element) and its new value (third element).

Table 6.2 lists the types of items in the results log (leftmost column) and the visualization formats which will be used in Chapters 7 and 8 for each type (rightmost column). The plots are generated using Yorick's command set.

The three types of data logged are:

- *The state vector of the burst block.* It is represented using a waveform view in figure 6.2 (traces are asserted when the burst block is in state *on* and deasserted in states *off* and *ref*) and as a coloured matrix in figure 6.3-A (neurons in state *on*, off and ref are represented by white, black and gray pixels respectively).

- *The state variable of the threshold block (w_{sum})* : It shows the total synaptic input received by a neuron at a point in time and can be viewed as a waveform and as a coloured matrix (figure 6.3-B) where blue areas correspond to regions

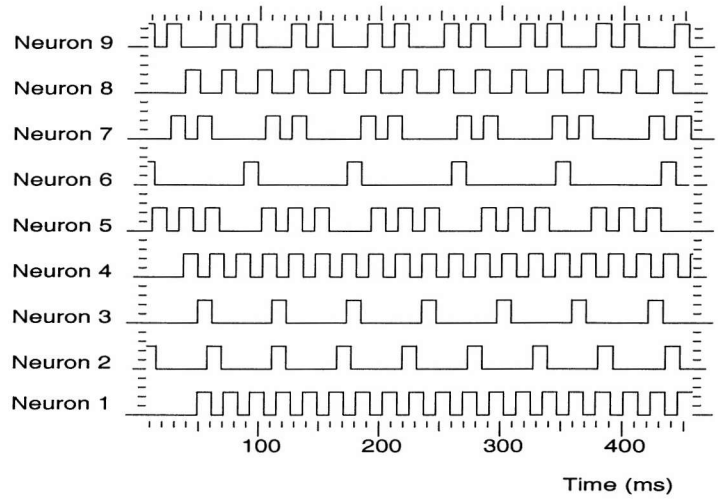


Figure 6.2: Waveform view of neuronal output

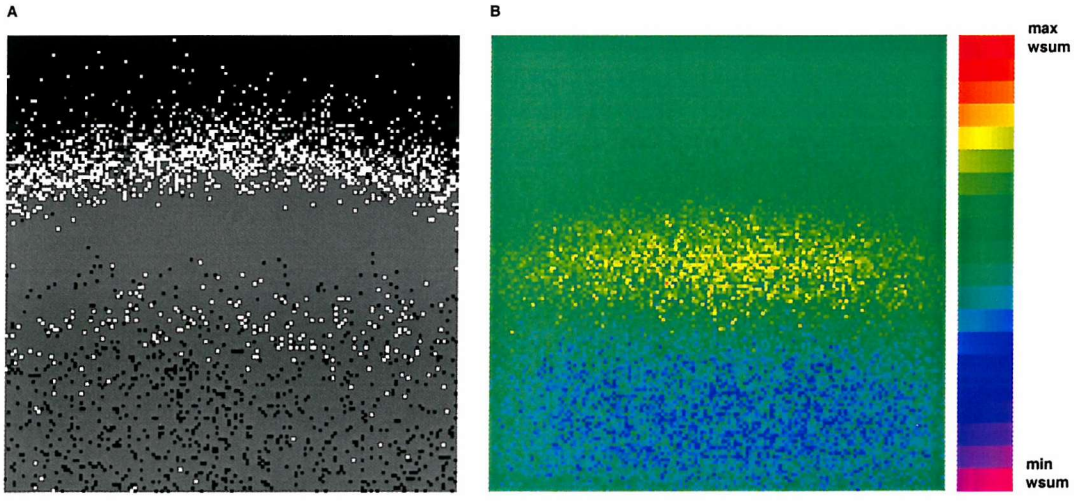


Figure 6.3: (A) Matrix representation of neuronal state (black, white and gray pixels correspond to neurons in off, on and ref states) (B) Matrix representation of the parameter w_{sum} in all neurons

receiving stronger inhibition than excitation ($w_{sum} < 0$) whereas yellow/red areas indicate greater excitation than inhibition ($w_{sum} > 0$).

The visualization of the contribution of individual classes of synapses to w_{sum} also facilitates the understanding of the dynamics of the network. In the case of the piriform cortex (Chapter 8), for instance, pyramidal neurons receive two types of inhibitory (fast $GABA_A$ synapses and slow $GABA_B$ synapses) and one type of excitatory synapses (fast Glutamate synapses). The measurement of the contribution to w_{sum_m} of each of these three types of synapses clarifies the role of each synaptic type in driving the dynamics of the neuronal population.

For a given neuron j, w_{sum_j} at time t can be expressed as

$$w_{sum_j}(t) = \sum_{m}^{M} \sum_{i}^{Sm} \alpha_{mi} w_{syn_{mi}} \tag{6.1}$$

where M is the number of types of synapses, S_m the number of synapses of the m^{th} type through which neuron j receives synaptic input, $w_{syn_{mi}}$ is the synaptic weight of the i^{th} synapse of the m^{th} class and α_{mi} is its number simultaneous synaptic activations at time t.

An alternative way of expressing w_{sum_j} is,

$$w_{sum_j}(t) = \sum_{m}^{M} pw_{sum_m} \tag{6.2}$$

where pw_{sum_m} is the partial contribution to w_{sum} of all synapses of type m and the summation is over the total number of synaptic types, M, through which the neuron receives its input.

Figure 6.4 shows a matrix plot of the partial contribution to w_{sum} (pw_{sum_m}) by the excitatory synapses (those with positive synaptic weight). A shift towards red indicates an increase of the total excitation.

- *The number of messages in the queue* : The time evolution of the instantaneous number of messages in the queue is used, later in this Chapter, in order to study the impact of queue size on the performance of the simulator.

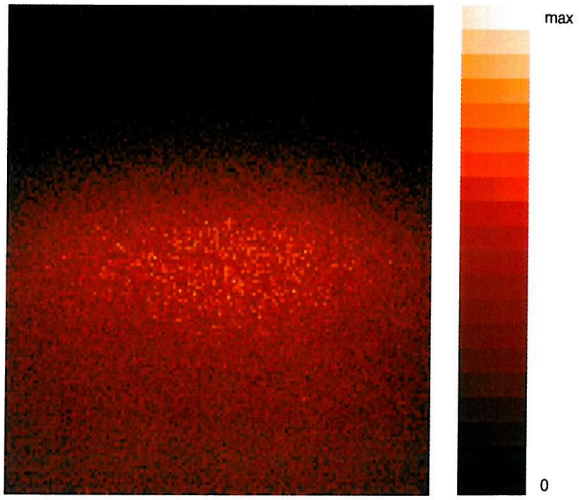


Figure 6.4: Matrix representation of the contribution of excitatory synapses to w_{sum}

6.3 Exemplar batch session for parameter space search

Parameter space search has been utilized in Chapter 8 to explore the effect of changes in network parameters on the dynamics of the model. Batch mode is more suited for this task than the interactive style shown in previous sections.

Figure 6.5 shows the listing corresponding to the file containing the commands to be executed by Yorick.

The first lines of code initialize vector z with a set of parameters used by the function *createpiriform()* to generate the network. A call to *eventstore()* and *initmodels()* sets the type of results to be retrieved and initializes the table of synaptic models respectively.

The parameter to be scanned is the first element in the vector, $z(1)$, and the range of the scan is (1..number_of_iterations). Within the loop, the parameter $z(1)$ is set to the value of the counter i and the topology is created by means of a call to the function *createpiriform()*. The simulation is run for 1000 time units (call to *simuevent()*) and the results returned in vector d can be visualized or stored before the next iteration.

```
//Set default value for non-scanned parameters
z=[1,-1000,0,0,...  ...1,0]

//Select data to retrieve
eventstore([0,0,1,1000000,2,2000000])

//main loop
for (i=1;i<number_of_iterations;i++)

{
  //set value of the scanned parameter
  //for this iteration
  z(1)=i

  //initialize synapse types
  initmodels(models)

  //create network
  //(z contains a vector of parameters for the
  // connectivity rules)
  createpiriform(z)

  //simulate for 1000 time units
  d=simuevent(1000)

  //Store/analyze results
  ...

}
```

Figure 6.5: An example script implementing parameter space search

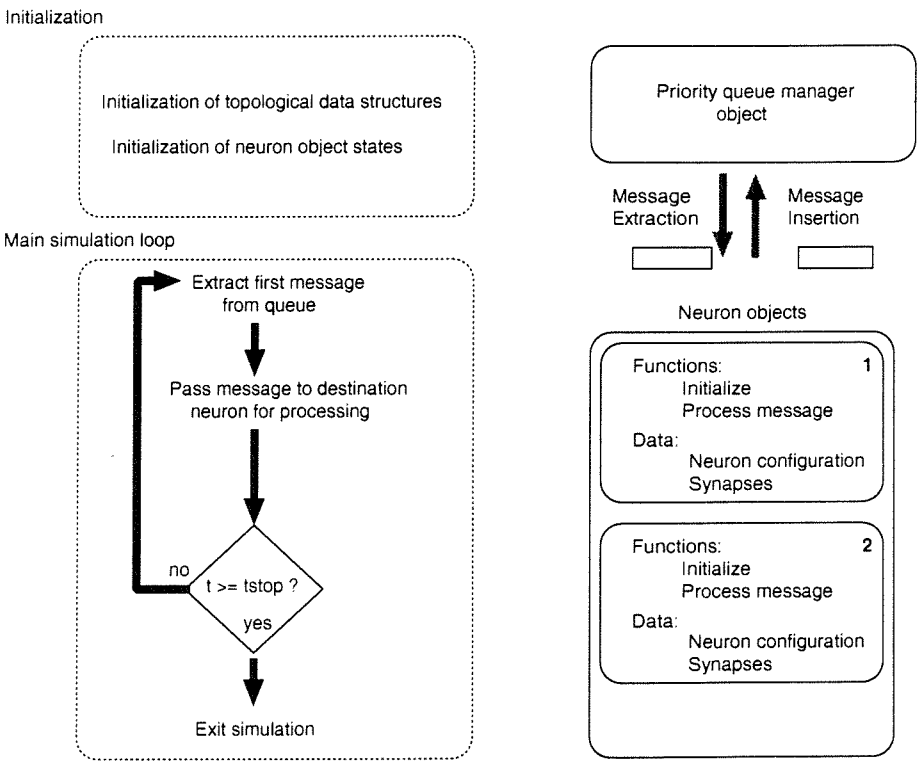


Figure 6.6: Block diagram of the internal structure of the simulator

6.4 Simulator internals

Figure 6.6 shows the main components of the simulator: the core of the simulation engine, which includes the initialization phase and the simulation loop, the event queue and the neuronal data structures. During the initialization stage (lefthand side of figure 6.6), the topological information provided by the user is used to create the network data structures (bottom-right in figure 6.6). These consist of the instantiated *neuron objects* and their synapses.

The simulator initializes the neuron objects calling their initialization function and enters the main simulation loop. Within each iteration, the message at the head of the priority queue (top-right in figure 6.6) is extracted and delivered to the destination device by a call to its *process_event* function. As a result of the arrival of a new message, the message processing routine within the *neuron object* updates its internal state vector and variables and, if needed, creates new messages to be inserted in the priority queue. Following the processing of the message, control is returned to the main simulation loop which continues with the next iteration until the condition $t \geq stop\_time$ is evaluated true.

6.4.1 Neuronal data structures

Two data structures have been implemented for the storage of the network: the large-network data structure (LNDS) is adequate for the problem of large scale simulation of networks in the order of 10^5 neurons, whereas an alternative implementation, the small-network data structure (SNDS), was developed for networks in the order of 100 neurons.

Data structure for large networks

The data structures used for the storage of large scale models are shown in figure 6.7.

The largest structure is the memory block allocated for the instantiation of *neuron objects* (center in figure 6.7). A *neuron object* contains the following fields: a neuron identifier (32 bits), the number of synapses from this neuron onto other neurons (32 bits), the state vector (32 bits), the state variables (three 32 bit-words), neuronal parameters (seven 32-bit words) and a list of synapses. The three state variables correspond to the sum of inputs, w_{sum}, the number of pending action potentials in an ongoing burst and a variable used for debugging purposes.

Each neuron object has an associated list of synapses. Each synapse is

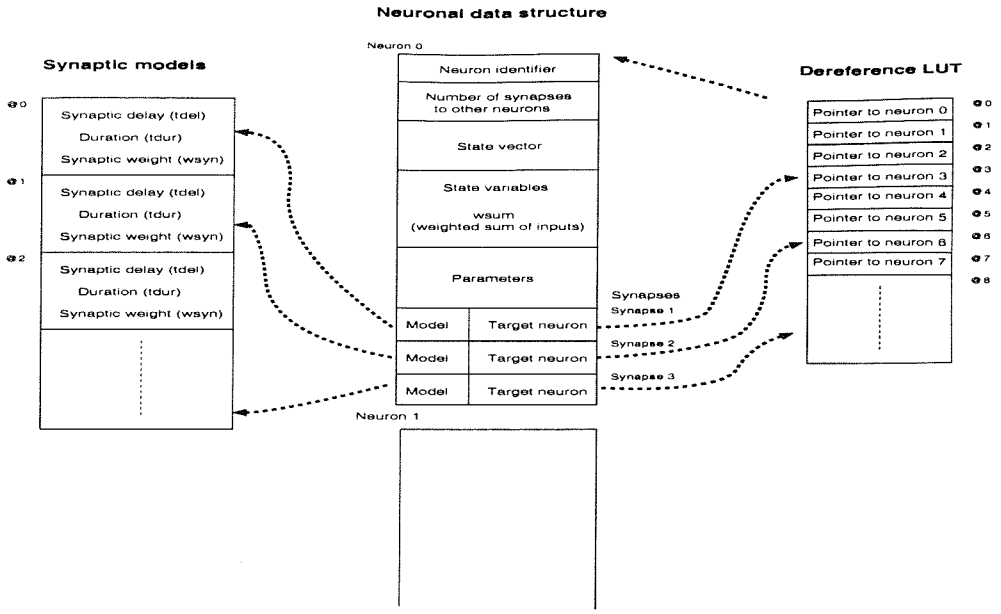


Figure 6.7: Data structures used for storage of neurons and synapses

functionally characterized by its parameter set (t_{del},w_{syn},t_{dur}) and the identifier of its target neuron. A major concern in deciding the most adequate data structures to store these data was to minimize their size, since the number of connections in a realistic model is expected to be two to four orders of magnitude higher than the number of neurons. Two strategies were implemented for this purpose:

- Synaptic parameters are not stored for each synaptic instantiation. Rather, a synapse type number is associated with each connection (8 bits). The actual parameters can be retrieved from a table containing synaptic parameter sets (seen on the lefthand side of figure 6.7) using the type number as an index into the table. Each entry in this table is a synaptic structure containing the parameters for one of the allowed types of synapses.

- The target neuron of a synapse is stored as a 24 bit identifier rather than a full 32 bit pointer. Considering that the MBED simulator run on a machine with a 32 bit-wide address bus, a 4 byte word would be needed to identify the target neuron if a pointer was to be stored in each synapse. Instead, an extra dereferencing level is introduced to minimize memory consumption. A neuron number is associated to any one synapse, identifying its postsynaptic cell. The actual memory address of the target neuron is found by accessing a table

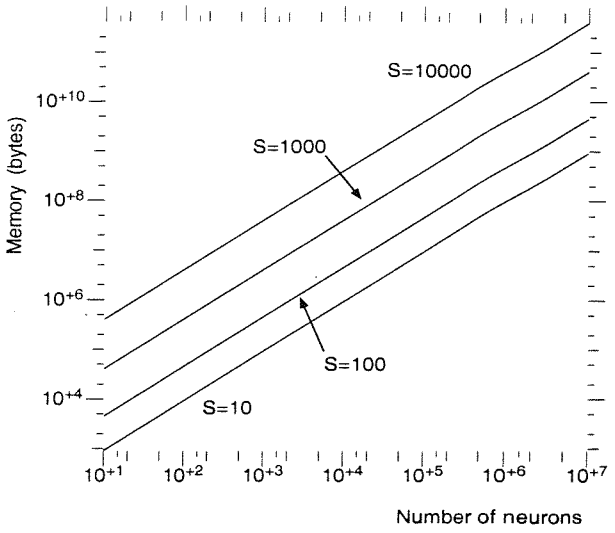


Figure 6.8: Memory space required for network storage as a function of the number of synapses per neuron (S)

(righthand side of figure 6.7), which contains actual pointers to the *neuron objects*, using the neuron number as an index into the table. Given that the neuron number does not correspond to a physical memory address, its size is not constrained to 32 bit (as required by a pointer) and can be reduced to a 24 bit identifier. The remaining 8 bits in a 32 bit synaptic word can allocate the synapse type number.

Hence, with the two strategies described above (a table of synaptic models and a dereferencing table of pointers to neurons), the parameters needed for a synapse can be masked into a single 32 bits word (24 bits for the target neuron identifier and 8 bits for the synapse type).

Efficient use of memory space was also achieved minimizing the overhead associated with dynamic allocation of large numbers of small objects [143]. Rather than allocating neurons individually, the simulator estimates the amount of memory required for the storage of all neurons and synapses in the network as

$$M = NCS + NP \tag{6.3}$$

where N is the number of neurons, C the number of connections per neuron and S and P the size of the neuronal structure (excluding synapse list) and the single synapse data structure respectively. A memory block of size M is requested from the

standard memory manager, avoiding dynamic allocation of individual neurons.

Overall, the described data structures implement a unidirectionally linked (presynaptic to postsynaptic neurons) network of *neuronal objects*. Linked structures were chosen because they are suitable for sparsely connected systems [55] whereas array based storage is memory-efficient for highly connected networks. This is a convenient design decision given that the expected degree of connectivity in a large scale network model incorporating experimentally obtained topological data is likely to be low and variable across different structures. For instance, the connectivity is estimated to be in the order of 4 % within the CA3 area but 0.005 % between dentate gyrus and CA3 pyramidal cells in hippocampus [35]. Unidirectionality of the inter-neuron connections (pre to postsynaptic) contributes to the memory efficiency of the data structures by eliminating the need for back-linking (post to presynaptic).

Figure 6.8 shows the memory space required for the storage of the network as a function of the number of neurons and synapses.

Data structure for small networks

Figure 6.9 shows the data structures implemented for small networks. With respect to those used for large aggregates (figure 6.7), two differences must be noted: the storage of a copy of the synaptic configuration for each connection and the addition of backlinking from postsynaptic to presynaptic neurons.

As depicted in figure 6.9, any one neuron has an associated set of parameters and state variables and two additional substructures: the table of synapses onto postsynaptic cells and the table of pointers to presynaptic cells. The entries in the table of synapses accommodate instantiations of synaptic parameter sets $(t_{del}, t_{dur}, w_{syn})$ in addition to the identifier of target neurons.

The table of pointers to presynaptic neurons was added to provide backlinking from postsynaptic to presynaptic neurons, transforming the unidirectionally linked network implemented for large scale simulations into a a doubly linked aggregate.

The advantage of this data structure is its adequacy for the implementation of the algorithms involved in the adaptation of synaptic parameters (e.g. weight adaptation for the simulation of LTP/LTD), if these mechanisms were to be modelled in the future. In a network incorporating learning algorithms, synaptic parameters would be adjusted during the simulation. Because the parameters associated to different synapses are likely to take different values, it is convenient to store a complete set of synaptic parameters for each connection. Moreover, activity

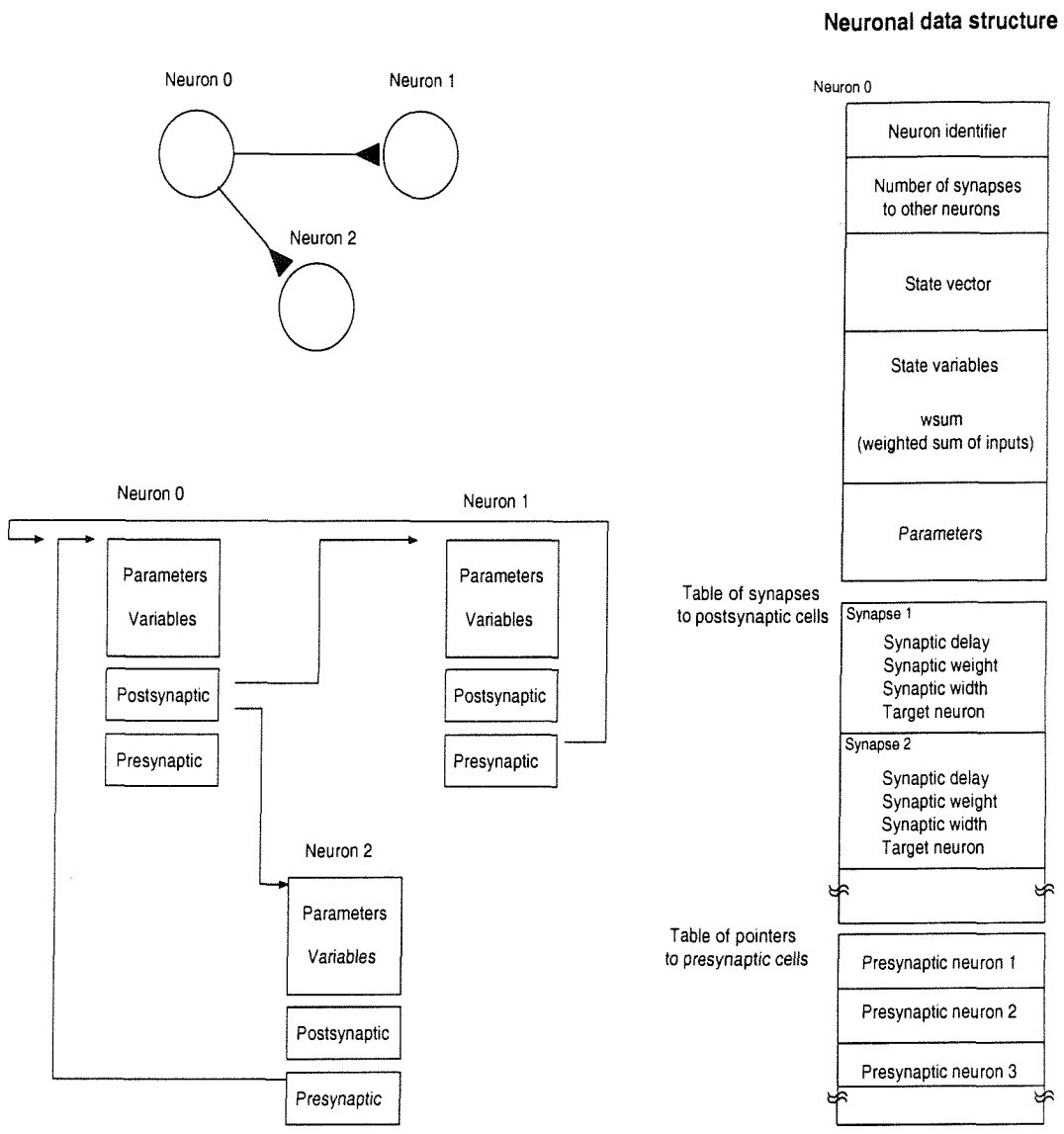


Figure 6.9: Data structures as implemented for small networks

in a postsynaptic neuron may induce changes in a presynaptic cell. For this reason, a doubly linked structure is preferred to a unidirectionally (presynaptic to postsynaptic) linked network.

The main disadvantage of this approach, when compared to the data structures for large scale models, is its less efficient use of memory resources. However, in the case of small networks (e.g. the *C. elegans* model described in Chapter 7) memory consumption is not an issue. For networks including in the order of 100 neurons and 100 synapses per neuron with an allocation of 20 bytes (five 32-bit words) per connection, the estimated memory space required exclusively for synaptic structures is $2 \cdot 10^5$ bytes. This value is well below the available memory in most desktop computers.

6.4.2 Priority queue

In a message-based event-driven framework, entities in the model communicate by message broadcasting [144, 145]. This is also the case with the neuronal objects in the MBED simulator, which transmit messages to their postsynaptic cells to communicate the occurrence of action potentials. As new messages are generated, those that do not carry an associated delay between generation and delivery to the target neuron object, are immediately processed by their destination neuron. On the other hand, those with non-zero latency between origin and destination are inserted in a time-sorted queue.

Since typical simulations of large network models (e.g. the cortical aggregates studied in Chapter 8) involve in the order of 10^8 messages, the design of the priority queue will affect the computational efficiency of the simulator. Two issues in this respect have been addressed: efficiency in terms of CPU time required for insertion/extraction of events into and from the queue and memory consumption by the queued messages themselves.

Efficient queue management

Numerous algorithms have been suggested for efficient queue management [78, 79, 146, 77]. Their performance is influenced by the insertion operations, usually the most costly operation in event-driven simulation, as events have to be time-sorted.

In general, insertion times in most queue management algorithms are affected by the size of the queue (the number of messages already queued); linear search

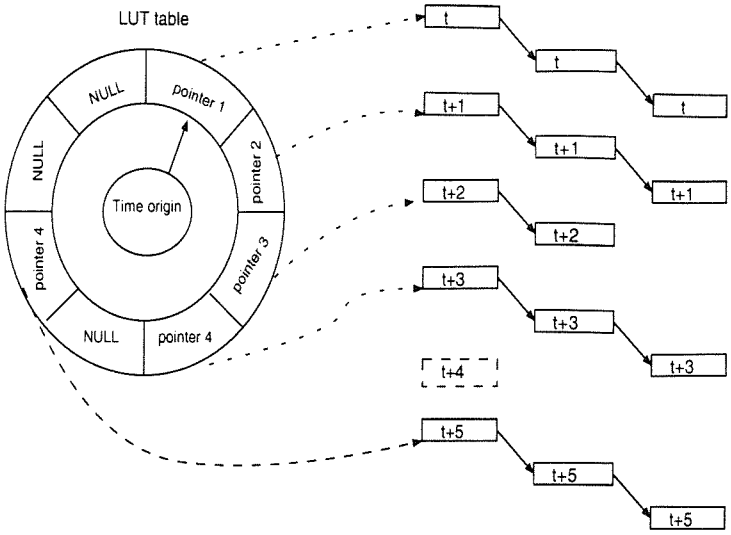


Figure 6.10: Priority queue managed with the aid of a circular lookup table

provides hold times (the latency involved in an extraction followed by an insertion) proportional to queue size, whereas binomial queues, pagodas, skew heaps, pairing heaps and splay trees run with O(log n) per hold operation [78].

An improved performance has been achieved with a caching technique [146]. This approach relies on the assumption that the insertion point of a new event is likely to be close to the insertion points of recent events and has proved efficient (O(1)) with five cache pointers. However, insertion latencies are expected to be dependent on the distribution of events in the queue, penalizing simulations in which events do not cluster in time.

The calendar queue, a multiple list scheme, also offers insertion latencies independent of queue size (O(1)) [77] and incorporates a dynamic adjustment of the internal structures which increases the robustness of the algorithm in the face of dynamic changes in the time distribution of events.

The calendar algorithm was inspired by the concept of a desk calendar. The priority queue, the calendar in the desk-calendar analogy, is partitioned into a number of sorted linked sublists. An array containing one pointer to the head of each sublist (page of the calendar) is used to find the appropriate sublist when an insertion is to be performed. A search for the insertion point is only necessary within a sublist. The implementation of an algorithm for dynamic adjustment of the total number of sublists was proposed in order to maintain an adequate mean sublist occupation. The algorithm was designed for robustness against transitory variations

in event distributions. With this mechanism, insertions O(1) in queue size were achieved.

There is, however, a computational cost associated to the dynamic adaption of the internal structures of the queue. This is aggravated in the case of a non-uniform distribution of events where most events cluster in a single sublist. This situation forces an increment in the number of sublists (pages in the calendar) to reduce the average number of events per sublist. As a consequence of event clustering, most sublists would remain empty throughout the simulation, at the expense of the computational efficiency of the algorithm.

Building on the calendar queue scheme, a new algorithm has been developed and used within the MBED simulator in order to further reduce the intra-sublist search cost and to simplify the algorithms involved in dynamic adjustment of the pagination of the queue. As a result, the developed algorithm provides O(1) insertion latencies and considerably simplifies the overall queue management. This has been possible migrating from a continuous time representation (as assumed in most priority queue algorithms) onto a discrete time representation.

The fine granularity of the time representation required by some event-driven problems (e.g. mixed-mode simulation) can be relaxed in the case of neural simulation. Neuronal activity consists of action potentials of, at least, $1 - 2\ ms$ of duration followed by a refractory period in the order of $10\ ms$. For discrete simulation of networks of neurons, time will be represented as a multiple of a time step in the range $100\mu s$ to $1\ ms$. Previous work on discretization of time in a realistic neural model [13] indicates that such an approximation is unlikely to compromise the usefulness of the simulation. Given this coarse granularity of time, a priority queue based on a LUT (look up table) and a multiple list scheme with one list per time point can be used (see figure 6.10). Since all messages within a given sublist are scheduled for the same point in time, no search is needed for an insertion.

Figure 6.10 illustrates this idea. The priority queue consists of a set of linked lists of messages. Each list links all the messages which have been scheduled for the same time in the future. An LUT stores pointers to the first message in each sublist. For an LUT with 10^6 entries, a maximum of 10^6 lists can be indexed. The first list links all messages scheduled for $t = 0$, and the last links all messages for $t = 10^6 - 1$. With a time step of $100\mu s$, messages cannot be scheduled further into the future than $100\ s$. The total amount of memory required for the storage of this array, using 32-bit pointers, is approximately 4 Mbytes. With the parameters above, an overflow occurs only if a neuron object introduces a delay greater than $100\ s$. Most single cell

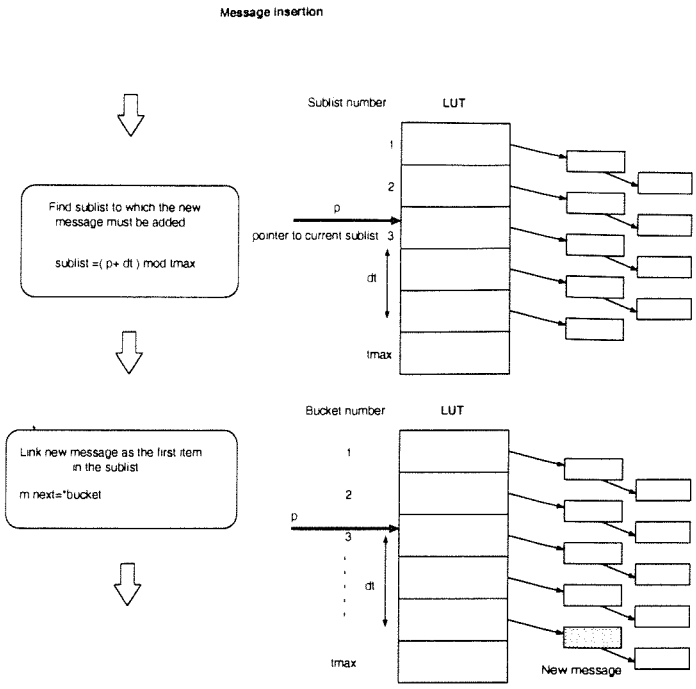


Figure 6.11: Algorithm for message insertion

neural processes occur within a time scale of 1 ms to 1 s, hence, an overflow is never to be expected in a realistic neural model. However, if longer delays have to be implemented, they can be introduced as a series of shorter delays.

The insertion of a new event (shown in figure 6.11) involves two steps; first, the entry in the LUT which keeps a pointer to the sublist where the event has to be added is calculated as

$$LUT_{entrynumber} = t_{delivery} - t_{current} \tag{6.4}$$

where $t_{delivery}$ is the delivery time for the message to be inserted and $t_{current}$ is the current time (both expressed in time steps).

Secondly, once a pointer to the appropriate sublist has been found, the actual insertion consists of the relinking of the chain of events with the new event being added to the head of the sublist.

The table of pointers is implemented as a cyclic buffer. A pointer to an entry in the table sets the time origin. As time advances, the pointer is moved forward in the buffer.

Figure 6.12 shows the steps involved in the extraction of the event at the head of

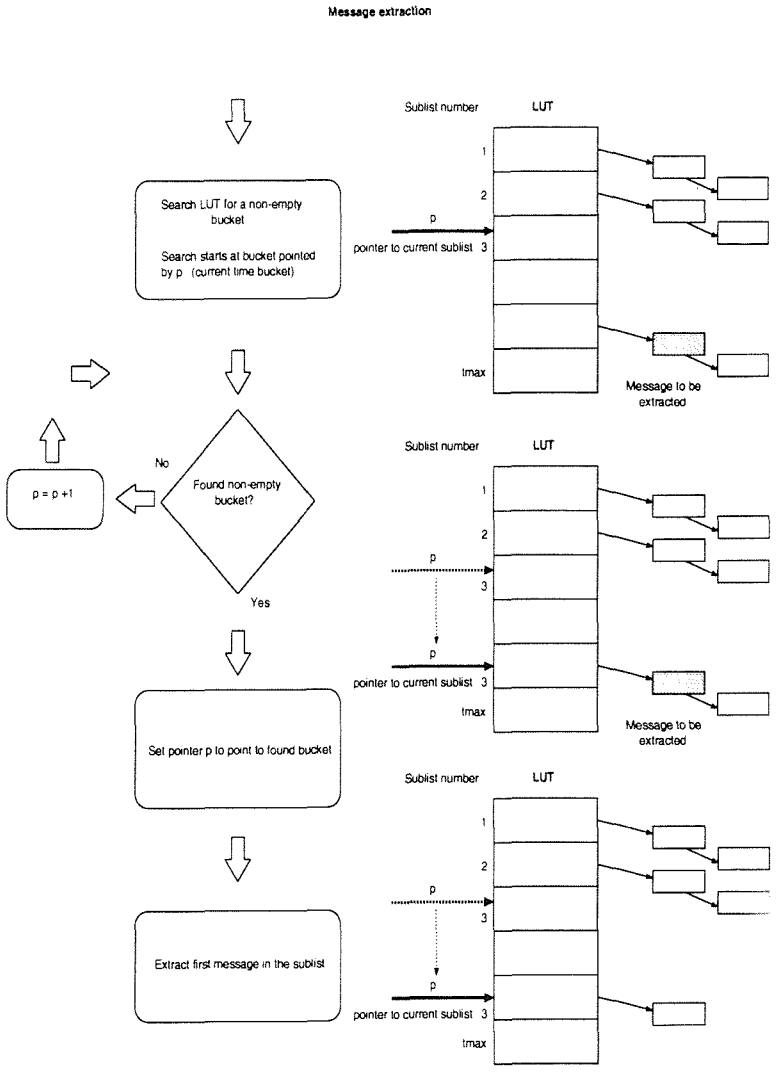


Figure 6.12: Algorithm for message extraction

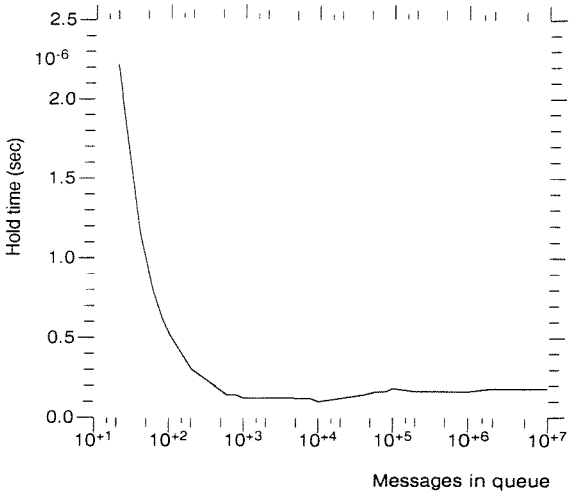


Figure 6.13: Hold latency as a function of queue size

the queue. First, the LUT slot corresponding to the current-time sublist is checked. If it points to a non-empty sublist, a message is extracted. If the sublist was empty, the current-time pointer (labelled p in figure 6.12) is incremented and the next sublist checked. This process is repeated until a non-empty list is found or until all slots in the LUT have been checked.

Figure 6.13 shows the latency associated with hold operations as a function of the size of the queue. It was generated by measuring the total CPU time taken by 10^7 hold operations, t_{hold}, on queues of several sizes, N.

For $N > 10^3$, the hold-time is independent with respect to queue size, whereas for $N < 10^3$, the hold-time increases as the size of the queue decreases. This result can be better understood by dissociating the latency involved in insertion from that incurred by extraction (see figure 6.14). Insertion latency is O(1) in queue size. Extraction times increase markedly as queue size decreases. A decrease in then number of messages in the queue leads to a sparse distribution of events and to a large number of sublists being empty. A penalization of the extraction operations was to be expected under this conditions, since it invòlves a linear search in order to find a non-empty bucket.

This does not compromise the performance of the algorithm in the context of neural simulation. Nearly empty queues constitute a highly unrealistic situation (as shown in simulations presented later in this Chapter) with queue sizes of the order of millions of messages being a more common situation. Within the limits of a typical large scale simulation, the simplicity of the modified calendar queue

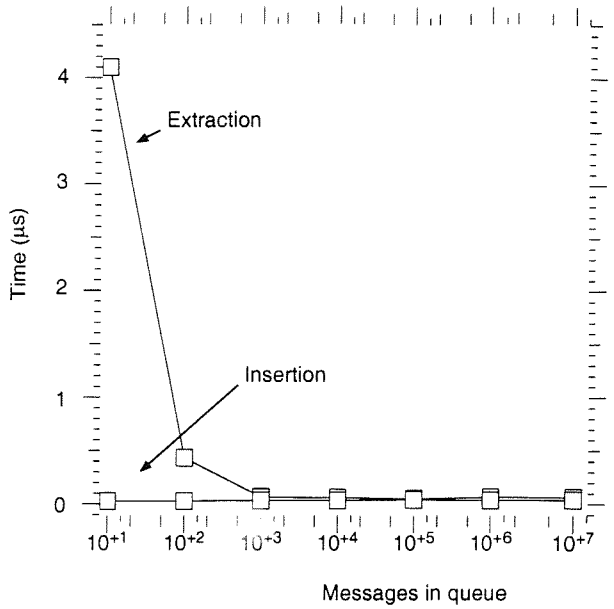


Figure 6.14: Insertion and extraction latencies as a function of queue size

presented here provides highly efficient queue management.

The memory required for the storage of the LUT is a function of the granularity of the representation of time and the maximum value allowed for the difference between current time and the time of message delivery, dt_{max}. Given a time step of t_{step}, the number of slots in the LUT is,

$$S_{LUT} = \frac{dt_{max}}{t_{step}} \tag{6.5}$$

In a typical case, with $t_{step} = 100\ \mu s$ and $dt_{max} = 1\ s$, the number of slots in the LUT is $S_{LUT} = 10^4$ and the memory required 39.06 Kbytes. This constitutes a 0.015% of the total 256 Mbytes of RAM used for the performance studies presented later in this Chapter.

Efficient dynamic memory allocation

The second major aspect to consider in the implementation of the event queue is the allocation and deallocation of memory for the storage of messages. A small block of memory is requested from the memory manager for the creation of each new message. Given the small size of the message data structure, it was found that dynamic allocation using the standard memory allocator provided by the C++

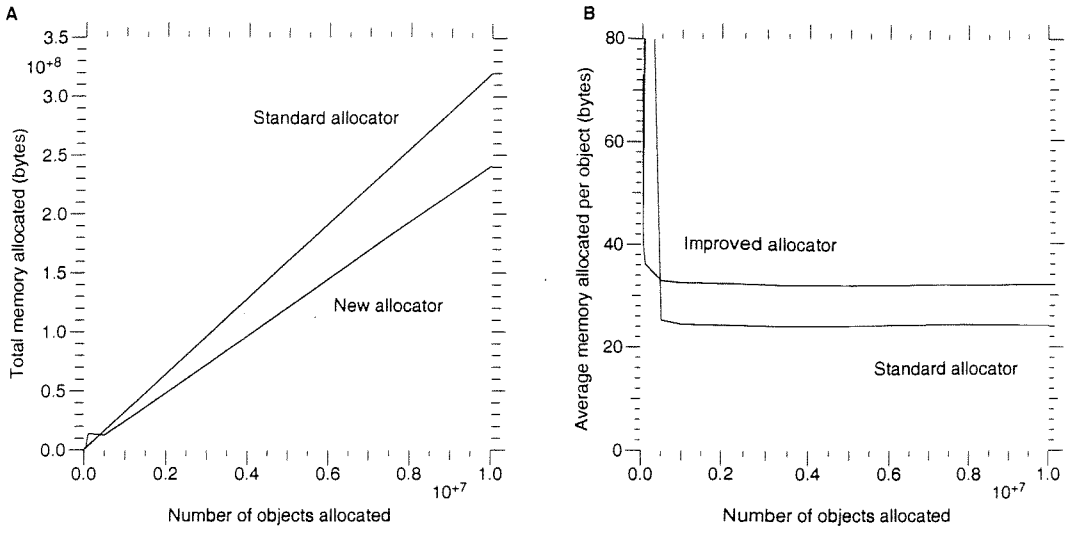


Figure 6.15: Comparison of overheads between standard dynamic allocation of messages and the improved algorithm

libraries constituted a rather inefficient alternative. Figure 6.15 shows the average memory taken by each dynamic allocation of a 24-byte object using the standard allocator. This was measured using a test program which dynamically allocates a large number of objects. The average memory taken by each object was estimated dividing the total memory requested by the process by the total number of objects instantiated. The average object size was 32 bytes, 8 bytes in excess of the actual dimensions of the data structure.

The amount of memory needed by the process of message creation and scheduling had to be minimized due to the size of the queue (typically millions of messages). Use of swap space degrades the performance of the simulator dramatically. Hence, this is not a viable alternative to the reduction of memory requirements.

The use of standard implementations of memory allocators is not adequate for the priority queue problem, because most standard memory managers have been designed for efficient memory allocation of heterogeneous objects. The empirical test shown in figure 6.15 identifies an overhead associated with dynamic allocation of small objects.

However, all message structures share a common size. This fact can be exploited to provide a more efficient memory allocator/deallocator. Gontmakher *et al.* [143] developed a memory management algorithm particularly suited for the allocation of homogeneous, fixed size, objects. This technique has been incorporated to the

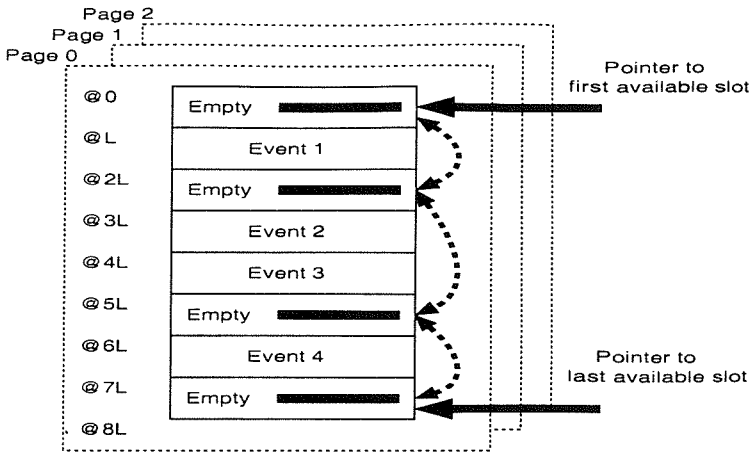


Figure 6.16: Efficient memory allocation for the priority queue

MBED simulator overriding the *new* operator for message objects.

Figure 6.15 shows that the improved implementation of the memory manager introduces an unnoticeable overhead when allocating homogeneous objects of 24 bytes in size. Note that the total memory requested by this improved algorithm (see 6.15-A) appears greater than that allocated by the standard manager only when the number of objects allocated is small. This is an expected result. Even when only a single object is allocated, the new memory manager requests a complete page and the average memory per allocated object is, apparently, high. As the number of objects increases, the advantage of using the new memory manager becomes more evident.

Figure 6.16 illustrates the algorithm. Upon initialization, the memory manager allocates a single page using the standard allocator. New messages are allocated in slots inside this page. Following the extraction of a message from the queue and its processing, it must be deleted from memory. Deallocation leaves empty spaces in the page. Free slots are added to a linked list of available slots.

Allocation of a message is a single step process when the free-slot list is not empty; the *new* operator removes the first slot from the free list. When this list is empty (no more free spaces in the current page), the standard memory allocator is requested to supply a new page and the first slot in this page is allocated.

Figure 6.17 compares the memory allocation times between the standard and the new memory allocators.

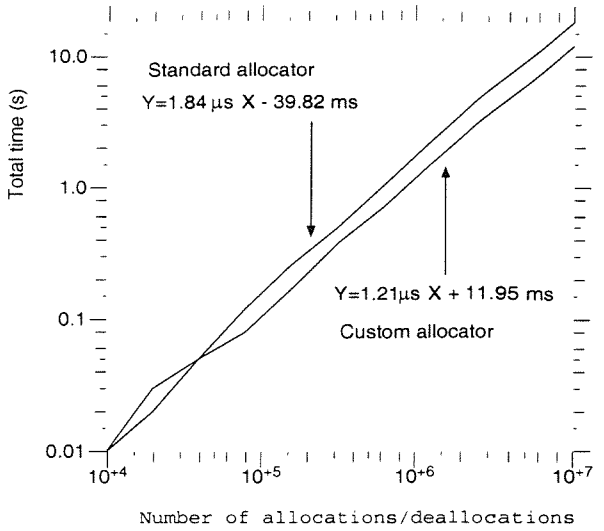


Figure 6.17: Comparison of allocation times for the standard dynamic allocation of messages and special purpose algorithm

6.4.3 Efficient computation of weighted synaptic input

To further increase the computational efficiency of the model, the update operation of the weighted sum of inputs (w_{sum}) in the threshold block can be optimized. An update of w_{sum} requires the computation of

$$w_{sum_j} = \sum_{i}^{S} \alpha_i w_{syn_i} \tag{6.6}$$

$$\alpha_i = \left\{ \begin{array}{ll} n & if\ synapse\ i\ was\ activated\ n\ times \\ 0 & if\ synapse\ i\ is\ inactive \end{array} \right\} \tag{6.7}$$

where S is the number of synapses providing input to neuron j, α_i is n if synapse i has been activated by n incoming messages and 0 if it remains inactive, and w_{syn_i} is the synaptic weight of synapse i.

The variable w_{sum} must be updated upon the arrival of *on* and off messages on channel γ. Complete recalculation of w_{sum}, as in expression 6.6, requires the weighted addition of S synaptic weights. In a typical neuron, the number of synapses (S) is expected to be in the range $10^2 - 10^4$.

Alternatively, w_{sum} can be recalculated as

$$w^{+}_{sum} = w^{-}_{sum} + \sum_{i}^{s} \alpha_i w_{syn_i} \tag{6.8}$$

where w^{-}_{sum} and w^{+}_{sum} are the weighted sums before and after an update respectively, s is the number of synapses which changed state simultaneously (typically $s << S$), α_i is 1 if synapse i has been activated and -1 if it has been inactivated and w_{syn_i} its weight. This requires the storage of the weighted sum as a state variable for each neuron but speeds up state recalculation of w_{sum} since the number of synapses which change state (s) is considerably lower than the total number of synapses S.

6.5 Performance evaluation with spatially uniform connectivity profiles

To study the performance of the simulator, a network of $5 \cdot 10^4$ neurons with random connectivity was simulated. Each neuron established C synapses with postsynaptic neurons chosen at random. Two types of synapses were included in the network; excitatory synapses with synaptic delay $t_{del} = 5\ ms$, efficacy $w_{syn} = 1$ and duration of activation $t_{dur} = 10\ ms$; and inhibitory synapses with $t_{del} = 5\ ms$, $w_{syn} = -1$ and $t_{dur} = 10\ ms$. The type of each synapse was chosen at random with probability p_e of being excitatory and $1 - p_e$ of being inhibitory. Multiple synapses of the same type from a given neuron to a target neuron were allowed, which is functionally equivalent to a single synapse of higher efficacy.

To provide input activity to the model, $5 \cdot 10^3$ out of the total $5 \cdot 10^4$ neurons were configured as pace makers which fired a single action potential (2 ms duration and 10 ms absolute refractory period) every 100 ms ($t_{osc} = 100\ ms$). The parameter t_ϕ (the time offset) was set according to a uniform distribution in the range $(0 - 60\ ms)$ to ensure that the subpopulation of pace makers did not fire in complete synchrony.

All neurons behaved as correlation detectors, firing a single action potential (also 2 ms duration and 10 ms refractory period) when the weighted sum of instantaneous synaptic inputs increases above the excitation threshold.

Several values for the percentage of inhibitory synapses, total number of synapses per neuron and the excitation threshold are explored in the following sections, to evaluate their impact on performance (simulation time and memory consumption).

All simulations of the randomly connected network were run for a simulated time

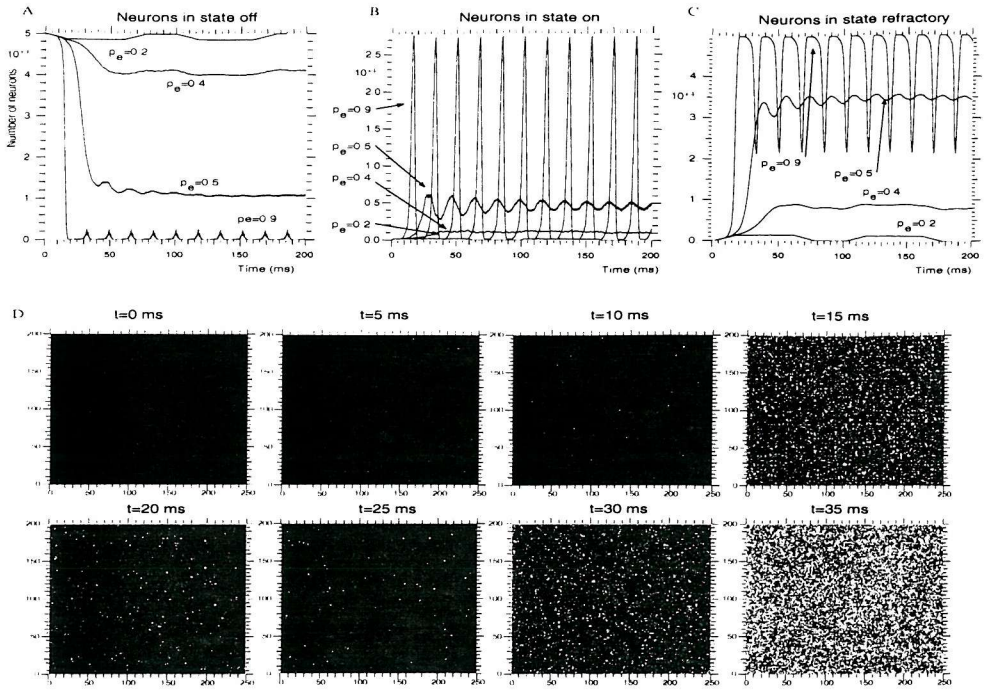


Figure 6.18: Total number of neurons in state *on* (A), *off* (B) and *refractory* (C), and time sequence of the neuron states for the network displaying epileptic-like activity (D) ($p_e = 0.9$). Neurons in state *on*, *off* and *refractory* are represented by white, black and gray pixels respectively

of 200 ms (2000 time steps) on a 233 MHz PC running Linux 2.2 with 256 Mb RAM.

6.5.1 Operation modes of the randomly connected network

During performance studies, the network displayed abrupt changes between different types of activity.

Figure 6.18 shows a set of simulations of the randomly connected model which contain both generalized and sparse network activity. Figures 6.18-A, 6.18-B and 6.18-C show the total number of neurons in states *on*, *off* and *refractory* respectively, for several values of the percentage of inhibitory synapses $(1 - p_e)$.

The sequence of peaks in figure 6.18-B, obtained with 10% of the synapses configured as inhibitory $(1 - p_e = 0.1)$, indicate that the neural ensemble becomes active and enters the refractory state nearly simultaneously. The lower trace in figure 6.18-A shows that few cells were in the off state since most neurons remained firing or refractory. This mode of operation resembles epileptic activity recorded with EEG and has been obtained previously with aggregates of cell automata models [13].

Figure 6.18-D shows the instantaneous state of the network ($1 - p_e = 0.1$) in matrix form. Neurons in states *on*, *off* and *refractory* are displayed as white, black and gray pixels respectively. The network-wide synchronization can be observed at $t = 15\ ms$, when a large proportion of the cell population is firing (*on* state). This corresponds to the first peak in figure 6.18-B for $p_e = 0.9$. At $t = 20\ ms$ and $t = 25\ ms$ most neurons are in $refractory$ state, which shows as a valley in figure 6.18-B and a maximum in figure 6.18-C.

As the percentage of inhibitory synapses is increased to 50% ($1 - p_e$=0.5), activation becomes less generalized and the global activity is characterized by a damped oscillation (middle trace in figure 6.18-B). Higher values of inhibition ($1 - p_e > 0.5$) result in neural activity confined to a small percentage of neurons, the maxima reaching 1000 in figure 6.18-B. Under these conditions, activity does not propagate in the network and, as a result, it is unable to trigger generalized synchronous firing.

These results are consistent with experimental results which have shown that the transition between generalized and low level activity can be triggered by the reduction in the total inhibition [147]. More generally, the global dynamics of brain tissue (as recorded by EEG) often shows different modes of operation. In an epileptic mode, a high percentage of the neurons in the ensemble fire simultaneously, whereas in normally behaving brain the overall activity is characterized by less generalized and less synchronous spiking and by the presence of characteristic frequency components [13].

The following sections provide quantitative results that demonstrate the effect of these modes of operation on simulator performance.

6.5.2 CPU time

The CPU time required for randomly connected networks has been studied running a set of simulations with several values for p_e (percentage of excitatory neurons), th_e (excitation threshold) and C (total number of connections per neuron). The region of the parameter space associated to these parameters includes configurations leading to near saturation network activity as well as sparse activation, allowing performance evaluation in a wide range of situations.

Figure 6.19-A shows the total number of messages processed during the simulations, which here serves as an indicator of processing load, versus p_e and th_e. The number of synapses per neuron was fixed to 200. Figure 6.19-B gives the CPU

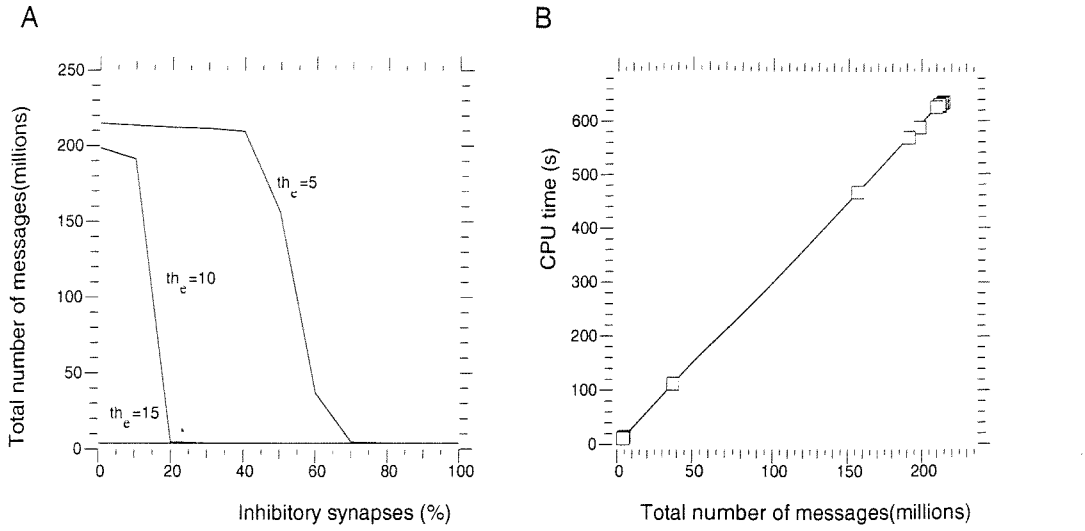


Figure 6.19: (A) Total number of messages versus percentage of inhibitory synapses and excitation threshold (number of synapses per neuron set to 200), (B) Simulation time versus total number of messages processed

time as a function of the total number of messages processed.

Figure 6.20-A shows the total number of messages as a function of the number of synapses per neuron and the value of the excitation threshold. The percentage of inhibitory synapses was set to 10% ($p_e = 0.9$). In Figure 6.20-B the simulation time is plotted as a function of the total number of messages processed.

The elapsed time can be found for any of the tested parameter sets in two steps: the number of processed messages can be looked up on the lefthand side plot and the CPU time required for this value is obtained from the plot on the righthand side.

Figures 6.19-A and 6.20-A indicate that, in the aggregates with the higher excitation threshold ($th_e = 15$), the activity remains comparatively sparse for all tested values of p_e and C. However, in those with lower thresholds, $th_e = 5$ and $th_e = 10$, the number of messages shows an abrupt increase indicating a transition in network dynamics from sparse activation into generalized firing. Generalized activity in the network decreases the performance of the simulator by increasing the total number of messages to process and, as a consequence, the CPU time required.

The relationship between the number of messages and the elapsed time is shown in figures 6.19-B and 6.20-B. The total CPU time taken by the simulations depends linearly on the total number of messages generated and processed during the simulation. This is explained by the fact that the two main tasks of the simulation

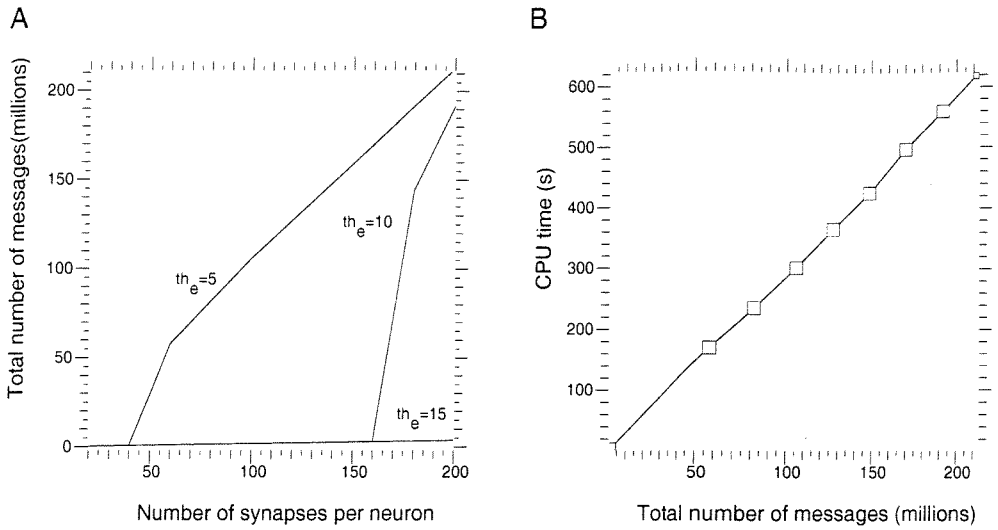


Figure 6.20: (A) Total number of messages versus number of synapses per neuron and excitation threshold (percentage of inhibitory synapses set to 10%), (B) Simulation time versus total number of messages processed

loop are the insertion of new messages into the priority queue and their extraction and processing.

An analytical expression can be found to relate simulation time and the number of messages processed

$$t = 2.9 \times 10^{-6} e \tag{6.9}$$

where e is the total number of messages and t the CPU time in seconds. Each message requires 2.9 μs for its processing.

Since it is useful to estimate, beforehand, the resources that will be required by a simulation, it is desirable to be able to predict the total number of messages which will be generated. However, this is difficult to anticipate as it depends not only on the topology of the network but also on its activity which will only be known after simulating.

The worst case scenario is produced by all the neurons in the aggregate firing simultaneously at their maximum firing rate throughout the entire simulation. In this case, the total number of messages processed is given by

$$e = E_{syn} + E_n \tag{6.10}$$

where E_{syn} is the total number of messages generated by synapses and E_n the number of messages generated by the rest of blocks in the neuron model. As the number of synapses is several orders of magnitude bigger than the number of neurons, the total number of messages processed can be approximated by

$$e \approx E_{syn} \approx \frac{2NC}{t_{ref}} t_{simu} \tag{6.11}$$

where N is the total number of neurons, C the average number of connections per neuron, t_{ref} the neuronal refractory period and t_{simu} the time of simulation. The factor 2 accounts for the two messages (activation and inactivation) inserted in the queue by a synapse.

In a typical simulation the average firing rate of a neuron is expected to be far from the maximum rate attainable. For this more realistic situation, expression 6.11 has to include a correction term β

$$e = \beta \frac{2NC}{t_{del}} t_{simu} \tag{6.12}$$

where β is the normalized average firing rate.

For the lowest values of p_e in figure 6.19-B, the total number of messages would be approximated by equation 6.12 with $\beta = 0.005$ whereas for high values of p_e a good match is achieved with $\beta = 0.5$.

Finally, figure 6.21-A plots the size of the queue versus time. When the generation (insertion) and processing (extraction) of messages is evenly distributed over time (see lower traces), the CPU time will also be evenly distributed throughout the simulation. However, the upper traces show that fluctuations of the queue size occur. Due to the event-driven nature of the simulator, the peaks in the number of messages to be processed will concentrate most of the CPU time whereas, in continuous-simulation, the processing load is evenly distributed over time. Moreover, variations in queue size lead to a dynamic demand of memory resources, as discussed below.

6.5.3 Memory requirements

The simulation of large networks is a demanding problem, not only with regards to CPU processing power, but also in terms of memory space.

To study the use of this resource by the MBED simulator, it is convenient to consider independently the two main sources of memory consumption:

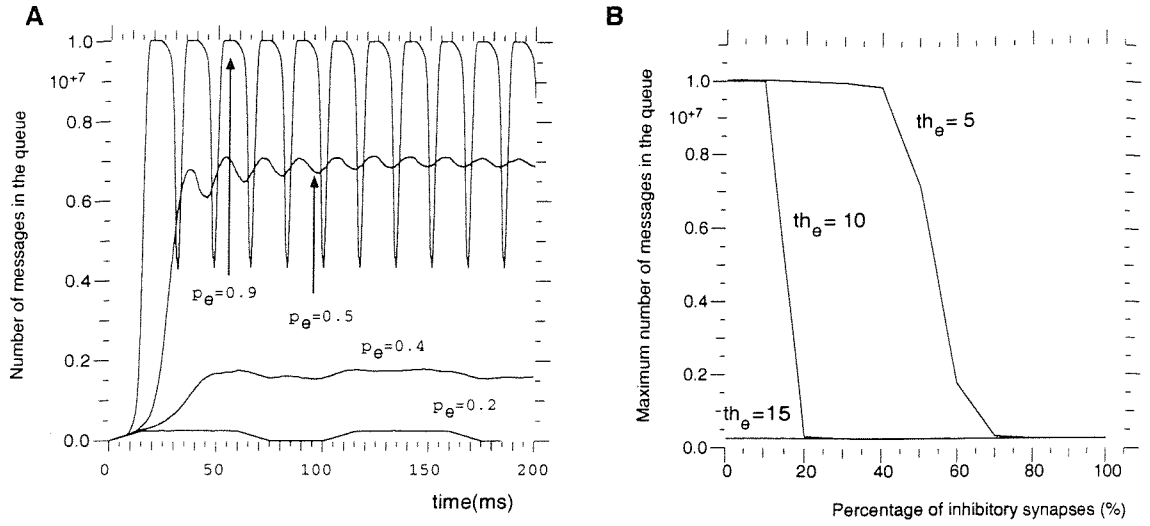


Figure 6.21: Instantaneous (A) and maximum (B) queue occupancy as a function of the percentage of inhibitory synapses and the excitation threshold (200 synapses per neuron)

topology/parameters, which are not dependent on the activity of the network, and the priority queue, whose size changes over time.

The amount of memory required to store topology and parameters (using the data structure described in section 6.4.1 for large scale models) can be estimated with equation 6.3 (reproduced for convenience)

$$M = NCS + NP \tag{6.13}$$

where N is the number of neurons, C the number of connections per neuron and S and P the space allocated for parameters and state variables for a single synapse and neuron respectively.

Figure 6.8 already gave the size of the network data structure for various aggregate sizes and numbers of synapses per neuron. It indicates, for example, that networks of 10^5 neurons with 100 connections per neuron required 100 Mb of memory.

Empirical measurements during the simulations of the network with $5 \cdot 10^4$ neurons used for figures 6.18, 6.19 and 6.20 yielded a value for the total memory allocated (including topology, parameters of the models, LUT of the priority queue and Yorick) of $53.5Mb$. This is consistent with the estimation provided by expression 6.3 taking $N = 5 \cdot 10^4$ neurons, $C = 200$ synapses, $S = 4$ bytes, $P = 52$ bytes).

In the case of models where the topology does not change during the simulation, the only uncertainty in terms of memory consumption lies in the size of the priority queue.

The queue, and the memory allocated for its storage, depends on the number of neurons and synapses simultaneously active. Figure 6.21-A shows superimposed traces with the instantaneous number of messages present in the queue during several simulations of the randomly connected network. Upper traces correspond to values of p_e close to 1 whereas lower traces correspond to values close to 0.

Synchronization of neuronal firing of large ensembles of neurons in the network causes oscillations in the size of the priority queue (as seen in figure 6.21-A for $p_e = 0.9$). These peaks in the number of firing neurons produce an accumulation of messages in the queue and the resulting increase of memory allocated to store it. Sufficient memory must be available in order to store the queue at any time during the simulation and avoid swapping, as this would have a negative impact on the performance of the simulator.

Since the maximum size of the queue during a simulation is the limiting factor, figure 6.21-B shows the maximum number of messages found in the queue during the simulations shown in figure 6.19. When the percentage of inhibitory synapses is small, the activity generated by the pace makers propagates in the network activating most neurons and flooding the event queue. As the percentage of inhibitory synapses increases, the network becomes only sparsely active and the maximum number of messages in the queue during a simulation decreases dramatically. This reduces the memory resources needed for the simulator.

The amount of memory allocated reached 120 Mbytes when the queue grew to its maximum size of 10^7 (message size of 12 bytes).

6.5.4 Effect of network size

The size of the randomly connected network (N) was set to values in the range 10^4 *to* $8 \cdot 10^4$ at intervals of 10^4 neurons, in order to study the effect of neural population size on ensemble dynamics and queue occupancy. The number of pace maker neurons was fixed to 5000. Four simulations were run for each network size, corresponding to four different values of the number of synapses per neuron (S=50,100,150 and 200).

Figure 6.22 shows the instantaneous number of neurons in state off as a function of the size of the network and the number of synapses per neuron. Traces

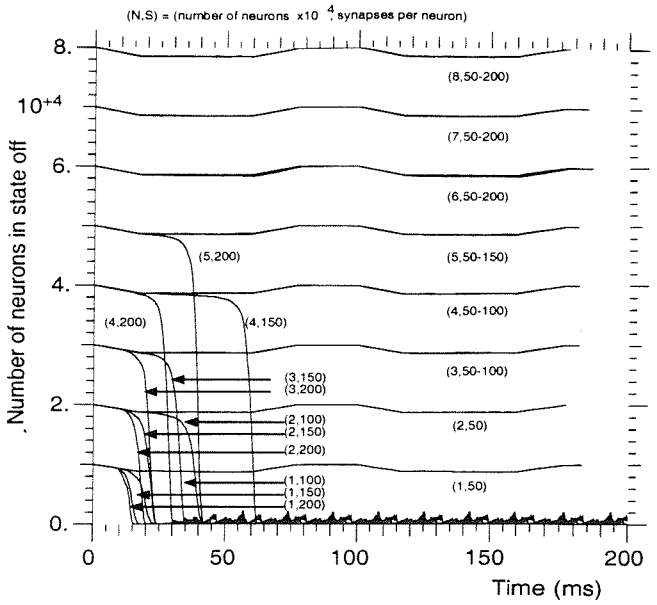


Figure 6.22: Instantaneous number of neurons in state off as a function of the size of the network and the number of synapses per neuron

are labelled with a pair of numbers, corresponding to N (expressed as multiples of 10^4) and S.

Because at time $t = 0$ all neurons are in state off, the value taken by the initial segment of each trace indicates the total number of neurons in each network. As the pace maker neurons start firing, some networks undergo the transition from a low activity mode to a whole-network spiking mode. This transition is indicated by the abrupt change of the traces towards a near-zero value, showing that very few cells remain inactive (in state off). This is a consequence of the initiation of a cycle where nearly all neurons exhibit periodic changes from *on* to $refractory$ state, avoiding the off state.

Some networks did not undergo generalized firing. Their corresponding traces curve slightly at around 50 ms and 125 ms as a result of the activation of the pace makers showing that most cells remained inactive.

As the size of the network increases (moving upwards in figure 6.22), the transition to a generalized firing mode occurs later in the simulation and for higher values of per neuron synapses. Compare, for example, the 10^4 neurons network (bottom trace) with the $5 \cdot 10^4$ neurons network (fifth trace from bottom). The former, undergoes transition for $S = 100, 150, 200$ at around $t = 10$ ms whereas the

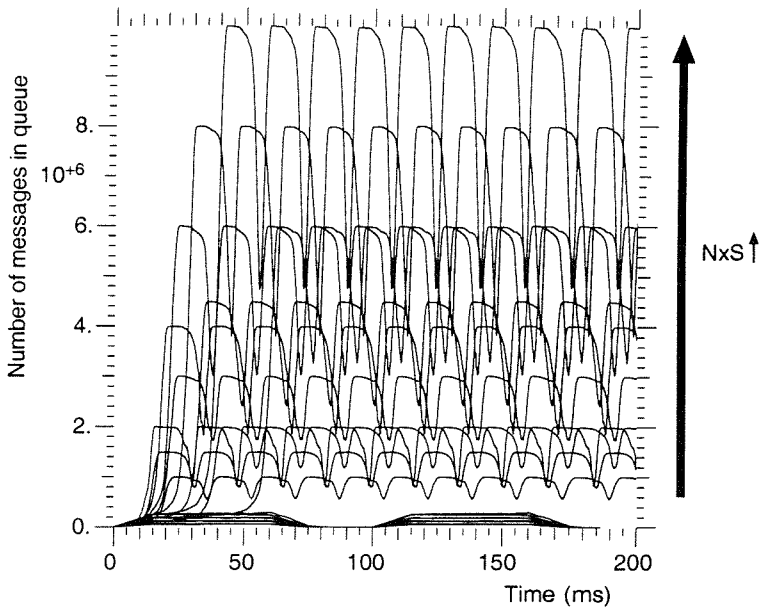


Figure 6.23: Instantaneous queue occupancy as a function of the product size of the network × number of synapses per neuron

latter exhibits generalized spiking only for $S = 200$ and at $t = 40\ ms$.

Figure 6.23 plots the number of messages in the queue as a function of the product network size × synapse density. Two subsets of traces can be readily identified; those corresponding to networks that did not undergo generalized spiking and those that did (analogous to the results shown in figure 6.21-A). Within the first subset, the average number of messages in the queue increases with the product $N \times S$. This indicates that, for a network displaying generalized epileptic-like activity, the CPU time taken by the simulation, which was shown in previous sections to depend linearly on the total number of messages processed, will decrease with increasing values of $N \times S$.

6.6 Performance evaluation with patterned connectivity

The performance of the simulator was also evaluated with a more realistic network; a model of the piriform cortex. The details of this model and its dynamics are provided in Chapter 8.

The network includes four pools of neurons; pyramidals ($4 \cdot 10^4$ cells), $GABA_A$

$(4 \cdot 10^4)$ and $GABA_B$ $(4 \cdot 10^4)$ inhibitory neurons and pace makers $(3 \cdot 10^4)$ totalling $1.5 \cdot 10^5$ neurons. Pace makers fire regularly (freq.= 10 Hz) introducing activity in the network. Pyramidal cells propagate excitation by pyramidal-pyramidal connections (100 synapses per neuron) and activate inhibitory GABA cells through pyramidal-inhibitory synapses (200 synapses per neuron). Inhibitory cells synapse back onto pyramidal cells (100 synapses per neuron). Pyramidal-pyramidal connections are long range whereas pyramidal-inhibitory are local. $GABA_A$ synapses were configured with $t_{del} = 5\ ms$, $w_{syn} = -10\ ms$ and $t_{dur} = 9\ ms$. $GABA_B$ connections have longer activation latencies and duration but less efficacy and were configured with $t_{del} = 10\ ms$, $w_{syn} = -1\ ms$ and $t_{dur} = 150\ ms$.

The plots on the lefthand side of figure 6.24 show the CPU time taken by 1 second of simulation as a function of the excitation threshold (th_e) of the pyramidal neurons (x axis) and the same parameter in $GABA_A$ and $GABA_B$ inhibitory neurons (indicated with a two element vector associated to each trace). The righthand side columns shows the total number of messages processed during the same simulations.

The CPU time decreases monotonically with increasing pyramidal excitation thresholds. Moreover, two regions can be distinguished in the x axis. Within the range $4 - 15$, the elapsed CPU time decreases approximately linearly from 400 s to 50 s. Smaller values of this parameter provoke a steep increase of the computational cost of the simulation, reaching 3500 s in the worst case (bottom-left plot).

The dependency of the elapsed time on the excitation threshold of the pyramidal cells is a consequence of its impact on the probability of firing: high thresholds lead to fewer spikes and, consequently, fewer synaptic activations and messages to process. Conversely, low excitation thresholds lead to more spikes and to a more computationally costly simulation. This interpretation of the results is confirmed by the plots in the righthand column, where the total number of messages are given as a function of the excitation thresholds. Moving along the x axis, the number of messages decreases towards higher values of the pyramidal threshold. Since the refractory period sets a maximum firing frequency, the number of messages reaches a plateau as the threshold approaches its minimum value.

The excitation thresholds of the inhibitory cells ($GABA_A$ and $GABA_B$ cell types) are given as a 2-element vector for each trace. As they are decreased, inhibitory cells are more likely to fire and the total inhibition in the aggregate increases. When the excitation threshold of the pyramidal cells is high (righthand half of the x axis), the activity of the pyramidal population is sensitive to the total

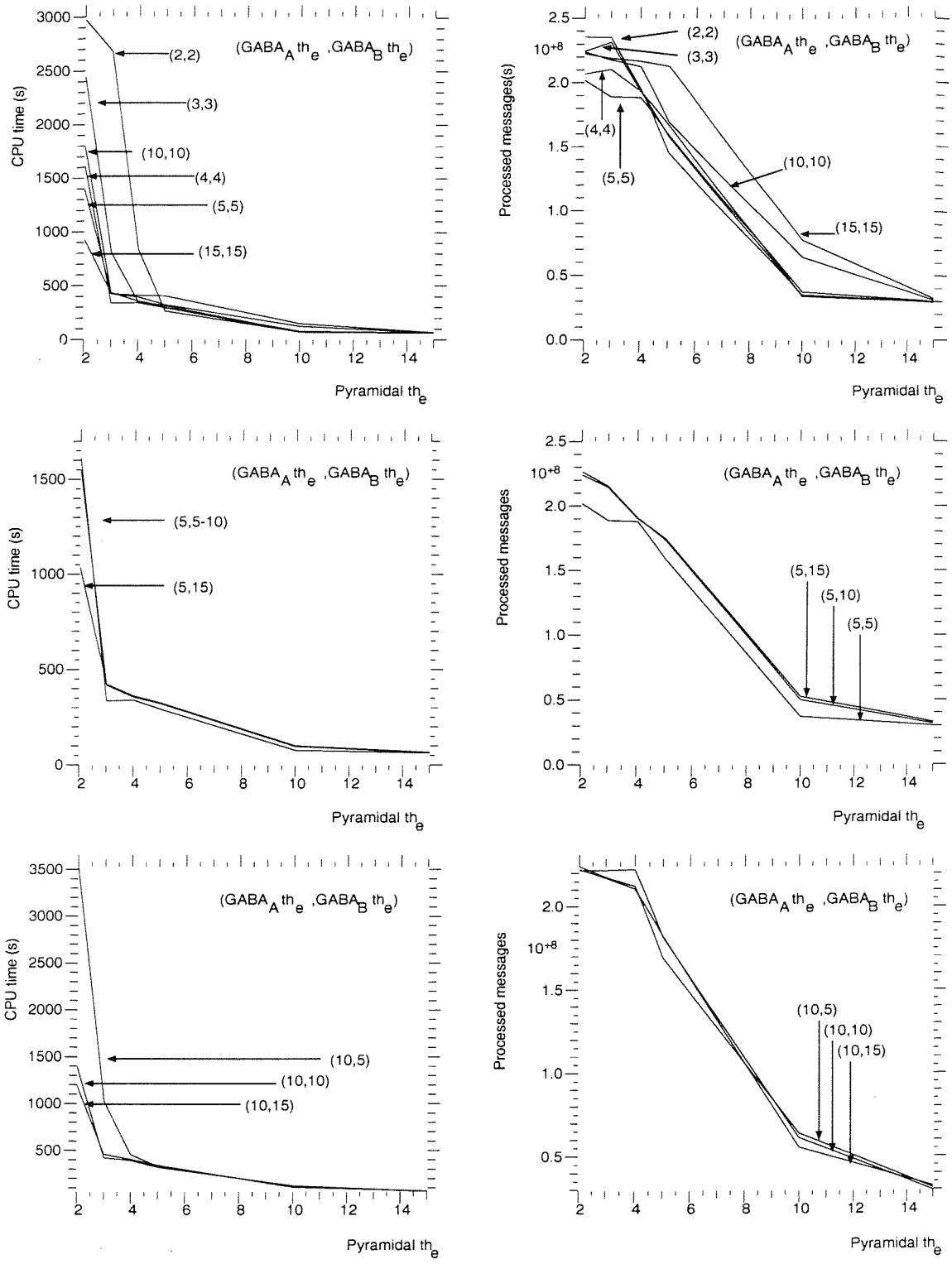


Figure 6.24: CPU time and number of processed messages as a function of the excitation threshold (th_e) of pyramidal and inhibitory neurons

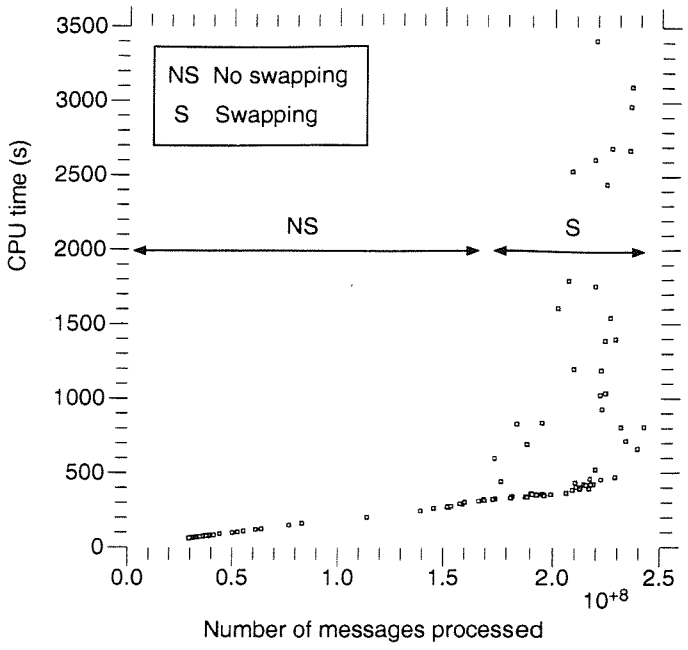


Figure 6.25: CPU time versus total number of messages processed

inhibition. This effect is seen as a decrease of the total number of messages in the top-right plot of figure 6.24. When the excitation threshold of the pyramidal cells is close to its minimum, the pyramidal population is firing at its maximum rate and is relatively insensitive to the inhibition in the network.

The steep increase of CPU time associated to low excitation thresholds is the result of the exhaustion of memory resources and the start of intensive swapping. The scatter plot in figure 6.25 relates the CPU time and the total number of messages processed and contains all the simulations included in figure 6.24. It shows a linear increase within the range $\{0 - 200 \cdot 10^6\}$ messages, where swapping is not necessary because of sufficient free memory space. Beyond $200 \cdot 10^6$ messages, several simulations required CPU times that deviate from the straight line. In these cases, the event-queue grew reaching the limits of the available memory.

Overall, these results indicates that: (1) the simulation of cortical models with sizes in the order of $1.5\ 10^5$ can be executed with elapsed times bellow 500 s in realistic conditions where the excitation threshold of pyramidal cells are sufficiently high to avoid the saturation of the firing rate, (2) the amount of available memory has a marked impact on performance, and (3) an increase of memory resources should be considered for network sizes beyond $1.5\ 10^5$.

Chapter 7

Event-driven model of C. elegans

7.1 Introduction

In this Chapter, the MBED neuron is used to construct a network model of the small circuit (in the order of 100 cells) which makes up the locomotory nervous system of the nematode *C. elegans*. The aim is two fold; to provide insight on the control of muscle contraction patterns in *C. elegans*, a problem only partially tackled by previous work [47][108] and to illustrate the capabilities of the MBED approach in the modelling of small systems, bridging the gap between the single cell simulations of Chapter 5 and the large scale simulations of Chapter 8.

Firstly, experimental data upon which the model was developed is presented. Recordings of behaving animals and an automated image analysis algorithm were used to obtain the patterns of muscle excitation. Secondly, the capability of these patterns to generate locomotion was tested by developing a mechanical model of the body of *C. elegans*. Having obtained the patterns of muscle activity required for the control of locomotion, the problem of finding a neural circuit capable of originating this activity is tackled. A circuit using the MBED neuron model and based on the available experimental data is proposed and its capability to generate forward, backward, reversal and coiling motions is demonstrated. Further validation of the model is provided by comparison of predicted and experimental effects of mutations and laser ablation of neurons. The model also serves to propose a testable hypothesis to explain the capability of *C. elegans* to modify its propagation velocity as a result of external stimuli.

7.2 Experimental data

Previous work on models of small invertebrate networks (see for example [87]) have heavily relied on electrophysiological measurements to specify neuron parameters. Due to the limited electrophysiological information available in *C. elegans* [107], the model of its locomotory circuit is based on the following types of data: anatomical information obtained with electron microscopy, laser ablation, histochemistry, genetic mutations and analysis of CCD recordings of behaving animals.

Of the 302 neurons which make up the nervous system of *C. elegans*, around 80 are directly involved in the generation of forward and backward locomotion [108]. These neurons were first identified as participating in locomotion on anatomical grounds [96] and further studies employing laser ablation have confirmed these preliminary results [104].

Figure 7.1 shows the topology of the locomotory circuit, based on data from White *et al.* [97]. Neuron classes VA, VB, VD, DA, AS, DB and DD are located along the body of the nematode. Cell types AS and DA, which share several characteristics [97], were considered as a single class and labelled as DA in the figure. The two units labelled AVA and AVB in the figure, correspond to two pairs of cells in *C. elegans* which possess axons extending the full length of the body. The leftmost and rightmost columns of cells in figure 7.1 correspond to ventral (labelled MSCVx) and dorsal (MSCDx) muscles.

Table 7.1 summarizes the neuronal types incorporated into the model and experimental data relevant to identify their function in the circuit. Neurons have been grouped in classes by their morphology and connectivity patterns as described in [97]. Available data on the neurotransmitter secreted were included, since this information aids in assigning tentative polarity to synapses (acetylcholine corresponding to excitatory and GABA to inhibitory connections). The rightmost column includes the suspected neuronal functions.

In addition to the static information available, summarized in Figure 7.1 and Table 7.1, dynamic data regarding the activity of muscles and motoneurons during locomotion were needed to construct the network model. Due to the reduced number of classes of neurons involved in locomotion (eight types in the model), a number of predictions can be made about the underlying motoneurons through the analysis of the visually observable patterns of body movements and muscular activity while locomoting.

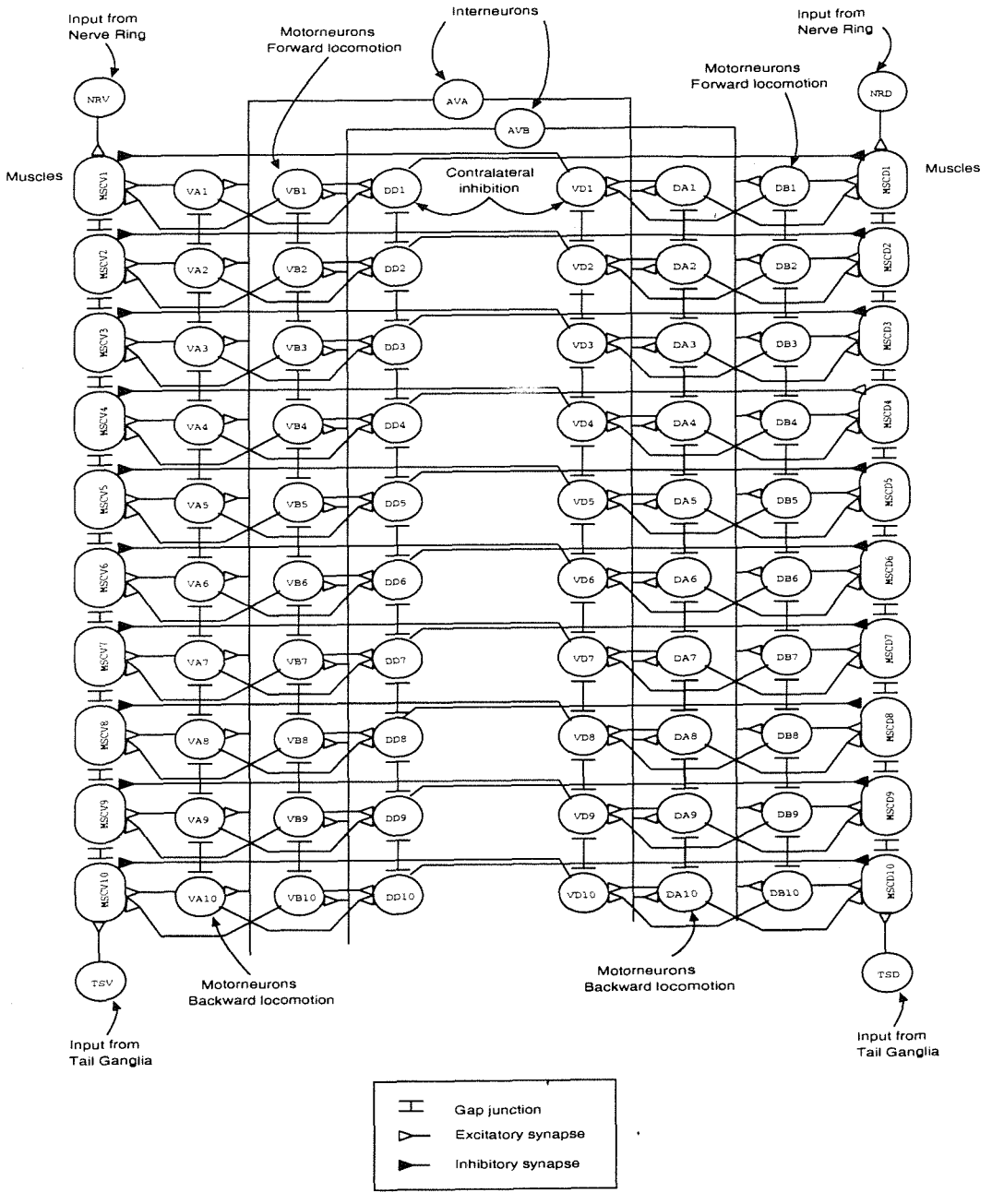


Figure 7.1: Simplified topology of the locomotory system

Neuron type	Number	Neurotransmitter secreted	Suspected function
VA	12	Acetylcholine	Motoneuron active in forward locom
VB	11	Acetylcholine	Motoneuron active in backward locom
VD	13	GABA	Inhibition of contralateral muscles
DA+AS	20	Acetylcholine	Motoneuron active in backward locom
DB	7	Acetylcholine	Motoneuron active in forward locom
DD	6	GABA	Inhibition of contralateral muscles
AVA	2		Control of backward locomotion
AVB	2		Control of forward locomotion

Table 7.1: Neuron classes involved in locomotion

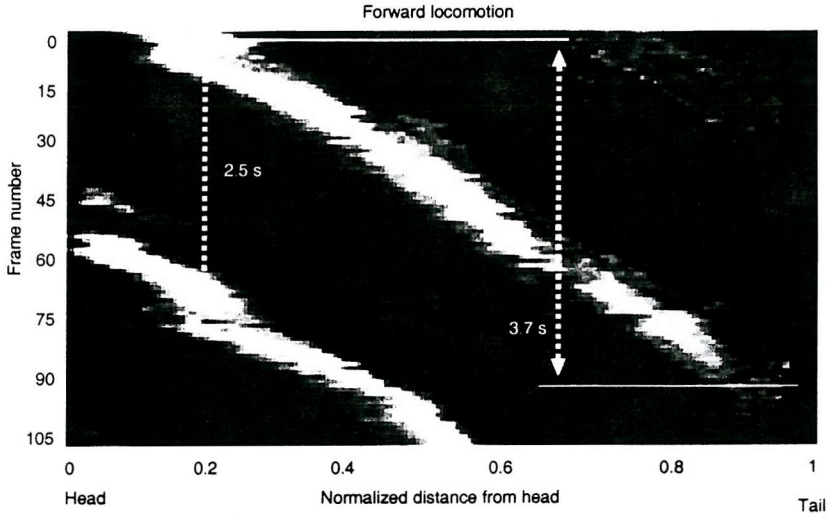


Figure 7.2: Local curvature of the body as a function of frame number and distance from the head during forward locomotion

7.3 Analysis of video recordings

Muscle contraction is required to generate and maintain the curvature of the body [101]. Hence, quantification of the body curvature can be used as an indirect way to measure muscle contraction without direct recording from muscle cells.

For this purpose, mature animals were transported to Petri dishes and imaged while locomoting on agar. An optical microscope (magnification x40) and standard interlaced video rate CCD camera (25 frames/sec, 50 fields/sec) were used. Figures B.1, B.2, B.3 and B.4 in Appendix B show sequences obtained during forward locomotion, backward locomotion, reversal and coiling respectively.

To allow further quantification of the temporal evolution of muscle states, the

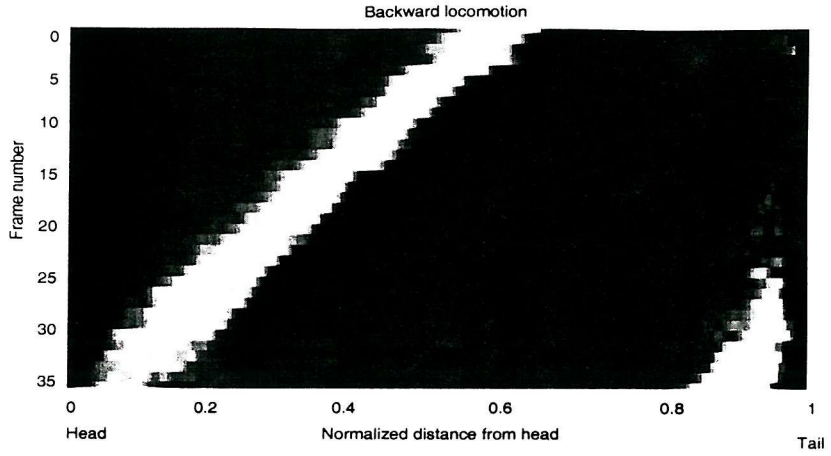


Figure 7.3: Local curvature of the body as a function of frame number and distance from the head during backward locomotion

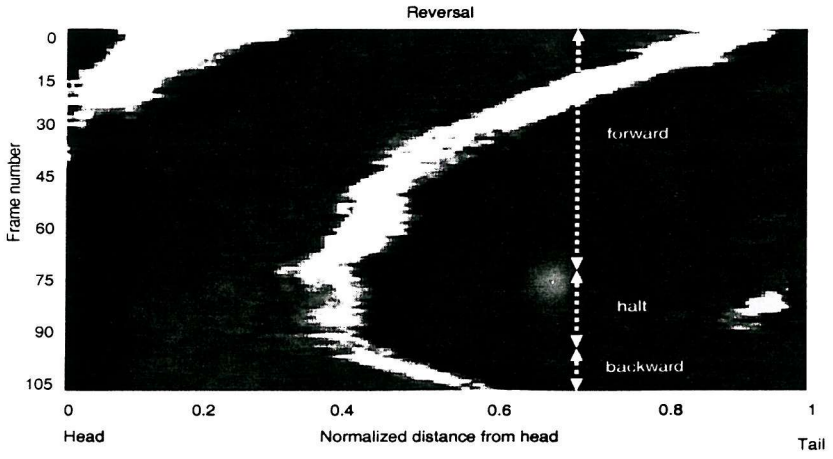


Figure 7.4: Local curvature of the body as a function of frame number and distance from the head during reversal

images were analysed with an automated image processing system. Details on the implementation of the algorithm are provided in Appendix C.

Briefly, sequences of images with locomoting animals were obtained with the CCD camera. Firstly, the body was identified in each image. Secondly, its contour was followed and stored as a sequence of two-coordinate position vectors. Finally, the second spatial derivative was calculated on the parameterized contour in order to estimate the local curvature of the body at different positions along its longitudinal axis.

The results are shown in figures 7.2, 7.3 and 7.4. They plot the temporal evolution of the local curvature of the body at different locations as it performs forward locomotion, backward locomotion and motion reversal, respectively.

The data are plotted as a coloured 2-D array where each row corresponds to a frame in the video sequence (time advances downwards) and columns to different positions along the body (leftmost and rightmost closest to the head and tail respectively). Curvature values were normalized to the maximum within the video sequence and plotted using a gray scale. Brighter segments indicate pronounced curvature whereas dark regions correspond to body segments remaining in a straight line position.

Figure 7.2 shows that forward locomotion was generated by propagation of muscle contraction from head to tail. Likewise, backward locomotion (figure 7.3) involved the propagation of waves of muscle contraction in the opposite direction (tail to head). Figure 7.4 shows the state of the muscles during motion reversal. Three phases can be distinguished: forward locomotion, halt and backward locomotion. During the halt phase, the spatial pattern of muscle contraction remains static.

From these CCD recordings, it was estimated that muscle contraction propagates at approximately 0.28 bodylengths/second. Considering ten neurons lined along the body in any one motoneuron class, it follows that the delay introduced between two contiguous neurons in the propagation of contraction is 360 ms. The latency between two excitations of any one muscle during forward movement is approximately 2.4 seconds. This is also the period of the head movement while propagating. The parameters in the MBED neuron model were set to obtain results consistent with these observations.

Having extracted the time-space evolution of body curvatures during locomotion, further proof of the equivalence between local curvature and muscle contraction was sought.

parameter	value	units
velocity of contraction propagation	0.28	bodylengths/sec.
delay between motoneuron	360	msec.
period of the body wave	2.4	sec.

Table 7.2: Quantitative data obtained from CCD recordings (average from $n = 4$ worms)

7.4 Mechanical model

A mechanical model of *C. elegans* was developed to confirm that the patterns of local curvature were indicative of muscle contraction by showing that they were capable of generating locomotion.

The model is based on previous work by Niebur *et al.* [108]. Their model was extended from two to three dimensions, allowing future work on out-of-plane head movements, and several force terms were simplified.

The body of the nematode was modelled as an elastic cylinder, a grid of $R \times C$ nodes (figure 7.5) with each node connected to its four closest neighbours by linear springs. The force acting on a node is the net contribution of its four neighbours

$$\vec{F}_s = \sum_{i=1}^{4} \vec{F}_i \tag{7.1}$$

where $\vec{F}_s$ is the net force and $\vec{F}_i$ the contribution of the i^{th} spring. The contribution of each spring is given by

$$\vec{F}_{r,c} = k(d_{r,c} - d_0)\hat{u}_i \tag{7.2}$$

where $\vec{F_{r,c}}$ is the force exerted by the spring located in row r and column c in the body grid, k is the spring constant, $d_{r,c}$ the distance between the nodes connected by the spring, d_0 the ideal length and u_i a unitary vector in the direction of the spring. The resting length, d_0, was set to confer stability to the model in a cylindrical shape,

$$\begin{aligned} d_{0t} &= \frac{2\,\pi\,rad}{C} \\ d_{0l} &= \frac{l}{R} \end{aligned} \tag{7.3}$$

where d_{0t} and d_{0l} are the resting lengths for transversal (perpendicular to the long axis) and longitudinal (oriented along the long body axis) springs respectively,

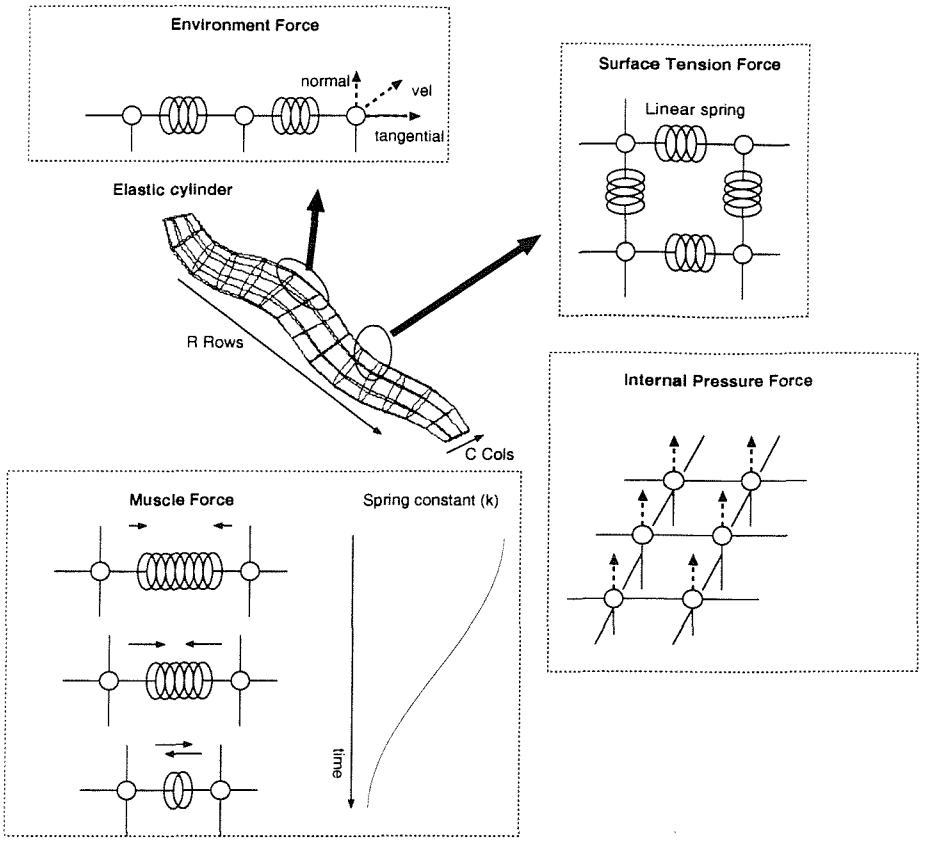


Figure 7.5: *C. elegans* mechanical model

rad is the radius of the cylinder, l its length and C and R the number of longitudinal and transversal bands of springs.

To maintain its cylindrical shape, the nematode requires the presence of internal pressure [108]. The internal pressure term is calculated in the model as,

$$\vec{F}_p = k_p \cdot \hat{n} \tag{7.4}$$

where k_p is a scaling factor and $\hat{n}$ is a unitary vector normal to the surface of the body.

The action of the environment is modelled as in equation 7.5.

$$\vec{F}_e = -k_r \cdot (\vec{v} \cdot \hat{n}) \cdot \hat{n} \tag{7.5}$$

where k_r is a scaling factor, $\hat{n}$ a unitary vector normal to the body and $\vec{v}$ is the velocity vector. The component of $\vec{v}$ tangential to the body surface is neglected,

since the nematode slips within a jelly groove with minimal friction. The normal component, however, encounters the resistance opposed by the jelly agar medium (analogous to the force acting on a free falling object in a fluid or gas). This force acts as a damping term to stop the mesh from oscillating in addition to providing the propulsion for body movement.

The net force acting on a point in the mesh is

$$\vec{F}_t = \vec{F}_s + \vec{F}_p + \vec{F}_e \tag{7.6}$$

where $\vec{F}_s$ is the force due to surface tension , $\vec{F}_p$ the internal pressure and $\vec{F}_e$ the resistance created by the environment.

Muscle contraction is simulated by changing the ideal length of the longitudinal springs in the body wall (equation 7.2). During contraction, the resting spring length is decreased. Conversely, during muscle relaxation, it is increased back to its initial value.

Figure 7.6 shows a sequence of images of the mechanical model performing forward locomotion. The spatio-temporal pattern of resting spring lengths corresponds to a travelling wave of contraction [108] and is given by

$$d_{r,c}(t) = d_0(1 + \alpha cos(\frac{2\pi c}{C})cos(\frac{2\pi r}{R/2} - \omega t)) \tag{7.7}$$

where $d_{r,c}(t)$ is the time changing ideal length of the spring in position (r, c), d_0 the length as in expression 7.3 and ω the frequency of the muscle contraction wave.

Expression 7.7 describes a wave of muscle contraction propagating from head to tail. At a point with maximal muscular contraction, it reaches its maximum, $d_0(1 + \alpha)$ whereas in regions with total muscle relaxation it yields a minimum of $d_0(1 - \alpha)$.

The first cosines in expression 7.7 introduces a π *radians* phase lag between springs within the same transversal section but in opposite sites of the body (i.e. between column n and $n + \frac{C}{2}$). This phase allows the bending of the body by ensuring that the contraction of the springs in one side coincides with relaxation of those in the opposite site. The second cosines takes the form $cos(kr - wt)$ corresponding to a wave propagating along the elastic cylinder.

Figure 7.7 shows the local curvature of the mechanical model as a function of time and distance from the head.

The capability of the model to move forwards with this pattern of muscle activity suggests that the neural circuitry controlling locomotion must be capable of

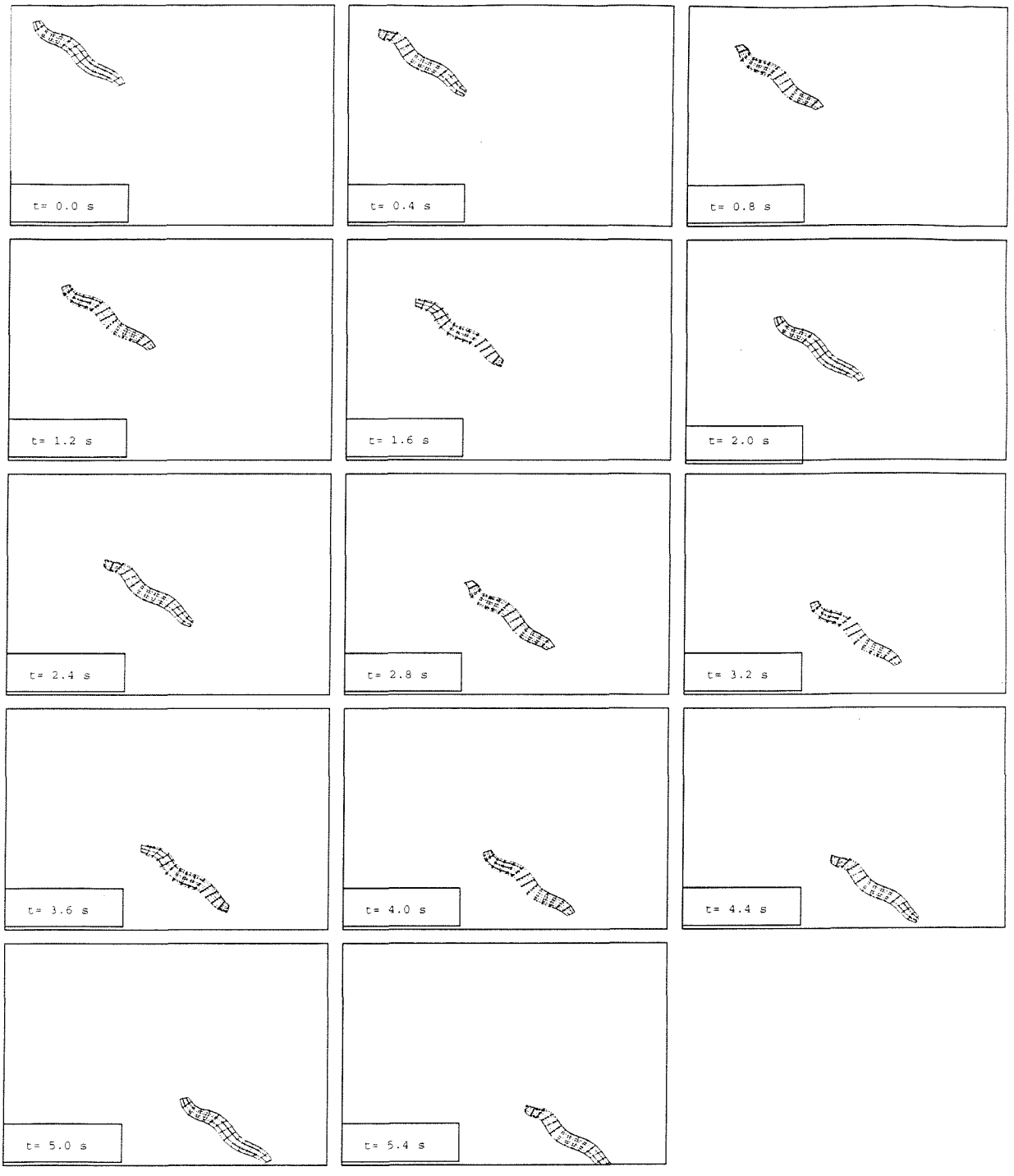


Figure 7.6: Sequence of images of the mechanical model during locomotion

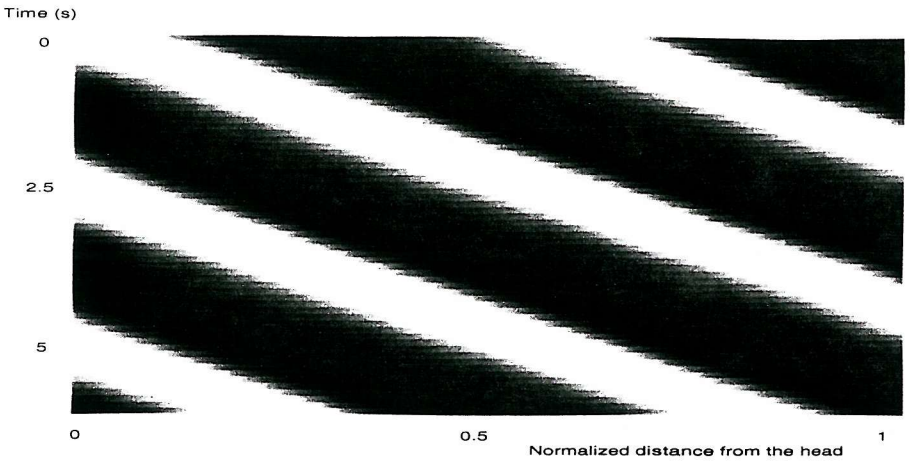


Figure 7.7: Local curvature of the mechanical model as a function of time and distance from the head

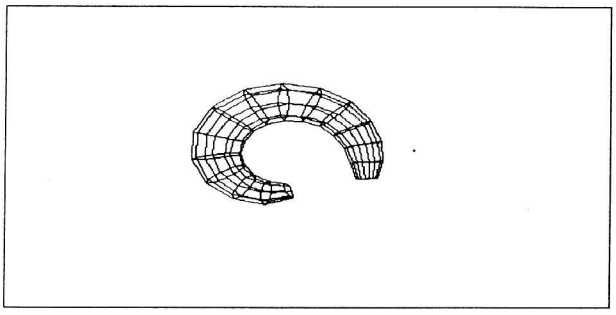


Figure 7.8: Complete body bending in the mechanical model

generating this output. It also confirms that the curvature data presented in figures 7.2, 7.3 and 7.4 are indicative of muscle contraction patterns, as they match those used in the mechanical model (figure 7.7) to generate forward locomotion.

These results have further constrained the network model presented in the following section to those configurations with motoneuron activity patterns consistent with the experimental observations.

The muscle states compatible with body coiling were also tested with the mechanical model. Figure 7.8 shows the mechanical model performing whole body bending.

The pattern of muscle excitation which induced whole body bending is given by,

$$d_{cr} = d_0(1 + \alpha cos(\frac{2\pi c}{C})) \tag{7.8}$$

Parameter	Value	Parameter	Value
C	10	l	30
R	8	rad	1
k	1	d_{0l}	3.3
k_p	0.03	d_{0t}	0.63
k_r	1	α	0.5
ω	$2\pi/2.5$	dt	0.8 s

Table 7.3: Numerical values used in the mechanical model

which corresponds to all muscles on one side being contracted while those on the opposite site are relaxed. Note that the time dependent term included in expression 7.7 has been left out in 7.8.

Table 7.3 lists the numerical values used for the simulations of locomotion and coiling. The number of rows and columns in the grid of springs (R and C) were chosen, in order to minimize the number of springs and the computational cost, as the lowest values giving smooth locomotion. The scaling factors k and k_r were arbitrarily set to 1 and k_p adjusted to confer realistic proportions to the resulting cylinder. The angular frequency w was set to $2\pi/2.5$, corresponding to a period of 2,5 s as obtained from image processing of the locomoting *C. elegans*. The length (l) and radius (rad) were set to match the physical proportions of the nematode and the inter-node distances (d_{0l} and d_{0l}) were calculated as in expression 7.3. The parameter α, the fractional change of spring length associated to maximal muscle contraction, was found by trial-error. The chosen value, 0.5, was found to achieve sufficient curvature for locomotion while maintaining the smoothness of the body surface. Finally, the time step dt was adjusted to ensure convergence and limit computational burden.

Having obtained the muscular states compatible with locomotion (the output of the locomotory neural circuitry), a network model consistent with topological and functional data and capable of generating such output was constructed.

7.5 Model of the locomotory nervous system

7.5.1 Topology

The starting point for the construction of the model is the complete topological map of the nervous system of *C. elegans* [97][96][148].

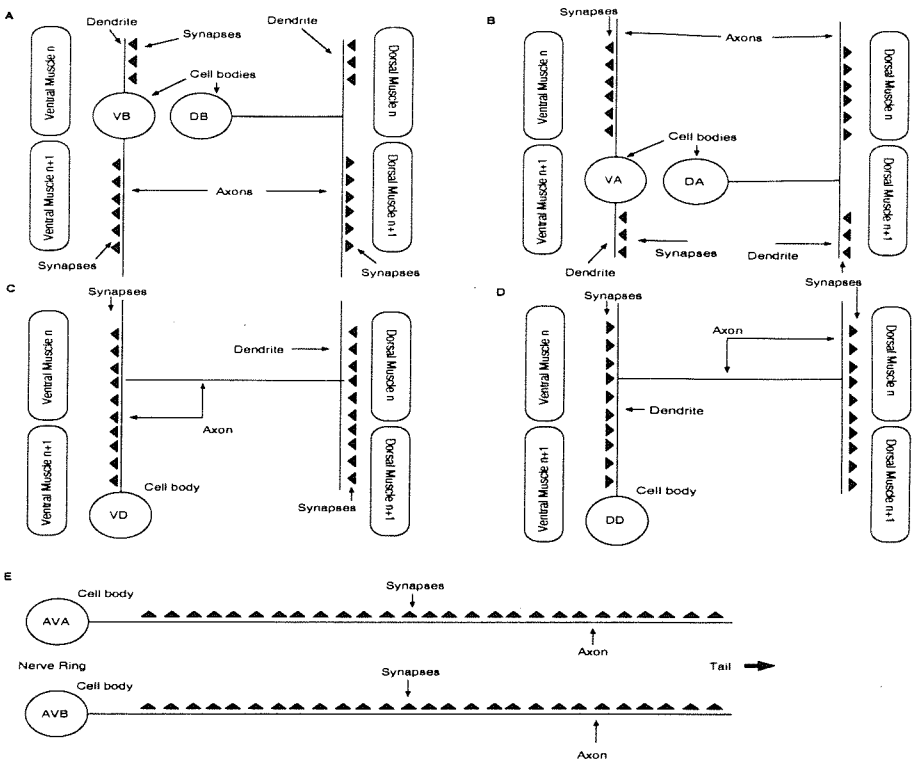


Figure 7.9: Sketch of the anatomy of the neuron classes in the locomotory circuit

Those classes of neurons no further than two levels of synapses away from muscle are identified. Several of these types of neurons have already been associated with functions other than locomotion, e.g. egg laying circuitry, and are excluded from the model [97]. Table 7.1 lists the remaining neuronal types, and a sketch of the connectivity amongst them is depicted in figure 7.1.

Figure 7.9 depicts their dendritic and axonal geometries, which aid the assignment of functional roles to individual neuron types.

The model includes ten cell types; VA, DA, VB, DB, VD, DD, AVA, AVB, MSCD and MSCV (cell notation as used by White *et al.* [97]). Classes VA, DA, and VB and DB have a short dendrite which receives synaptic input and a long axon synapsing onto nearby neurons. VA and DA neurons are a mirror image of VB and DB, with a symmetry axis perpendicular to the body. The former play the role of directional (tail to head) signal propagators whereas the latter have an analogous function for the propagation in the opposite direction [108].

VD and DD cells receive synaptic input from one side of the body and inhibit cells in the opposite site [110, 149]. AVA and AVB are interneurons which provide

input to VA-DA and VB-DB neurons respectively. MSCD and MSCV are muscle cells. The model includes ten neurons for each of the cell classes VA, VB, DA, DB, VD and DD and a single neuron for AVB and AVA. Ten muscles (MSCDx and MSCVx) are located in each side of the body.

Four units in the model, labelled NRD, NRV, TSD and TSV, act as signal sources encapsulating activity originated in other circuits and do not correspond to individual cells in *C. elegans*.

NRD and NRV account for dorsal and ventral input from the nerve ring, the main neuronal aggregate in the head. TSD and TSV correspond to dorsal and ventral neurons in the tail subsystem. Several mechanisms are capable of generating the activity modelled by these units in the animal. Proprioceptive sensors, mechanically activated cells, have been identified in Ascaris [109], a nematode with anatomical and functional similarities with *C. elegans*, and may provide direct input to the locomotory circuit. Passive gap junction propagation from neurons in the nerve ring and the tail may also provide a source of input (these connections have indeed been identified anatomically [97]).

In the model presented here, the following additional assumptions were made; that for classes VA, VB, DA and DB, the activity of the neurons is modulated by stretch receptors [108] (a similar hypothesis has been put forward and corroborated for other invertebrates, e.g. the leech [150]) and that gap junctions within classes VA, VB, DA and DB may be necessary for synchronization but are not essential to generate basic locomotion.

7.5.2 Neuronal parameters

Neurons and muscles were modelled with the MBED neuron described in Chapter 5. The excitation threshold (th_e) of neurons belonging to classes VA, VB, DA, DB, VD and DD was set as,

$$th_{e_j} = \sum_{i}^{N} \alpha(i,j) \tag{7.9}$$

where th_{e_j} is the excitation threshold of the j^{th}, N is the total number of neurons in the model and $\alpha(i,j) = 1$ if neuron i has an excitatory synapse ($w_{syn} > 0$) onto neuron j, otherwise $\alpha(i,j) = 0$.

As a consequence, all excitatory presynaptic neurons must be active to elicit an action potential in neuron j. For instance, VA neurons receive excitatory input from

Parameter	VA	DA	VB	DB	MSCD	MSCV	VD	DD	AVA	AVB
th_e	2	2	2	2	1	1	1	1	1	1
th_i	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
t_{osc}	0	0	0	0	0	0	0	0	-	-
N_{burst}	1	1	1	1	-1	-1	1	1	5	5
t_{ap}	1 ms	1 ms	1 ms	1 ms	10 ms	10 ms	1 ms	1 ms	1 ms	1 ms
t_{ref}	2 ms	2 ms	2 ms	2 ms	5 ms	5 ms	2 ms	2 ms	2 ms	2 ms
t_ϕ	0 ms	0 ms	0 ms	0 ms	0 ms	0 ms	0 ms	0 ms	-	-

Table 7.4: Fixed parameters of the model (I)

Parameter	Value	Parameter	Value
t_{del_1}	1 ms	t_{del_3}	1 ms
w_{syn_1}	1	w_{syn_3}	1
t_{dur_1}	300 ms	t_{dur_3}	1 ms
t_{del_2}	15 ms	t_{del_4}	1 ms
w_{syn_2}	1	w_{syn_4}	-1
t_{dur_2}	100 ms	t_{dur_4}	1 ms

Table 7.5: Fixed synaptic parameters of the network (II)

	Motion-type dependent parameter values					
	Forward		Backward		Coiling	
Neuron Type	t_{osc}	t_ϕ	t_{osc}	t_ϕ	t_{osc}	t_ϕ
AVA	0	0	360 ms	0	0	0
AVB	360 ms	0	0	0	360 ms	0
NRV	2.4 s	0	0	0	2.4 s	0
NRD	2.4 s	1.2 s	0	0	0	0
TSV	0	0	2.4 s	0	0	2.4 s
TSD	0	0	2.4 s	1.2 s	0	0

Table 7.6: Parameters of driving neurons for several types of locomotion

AVA and from their stretch receptors that sense the contraction of nearby muscles. Hence, $th_e = 2$ for all neurons in this class.

In muscle cells (MSCV and MSCD) the excitation threshold was set to $th_e = 1$. Consequently, the activation of any one of their presynaptic partners provides sufficient excitation to trigger the generation of action potentials.

Neuron classes AVA, AVB, NRV, NRD, TSV and TSD serve as activity sources and do not receive input from any other cells. For this reason, their excitation thresholds do not affect their operation.

The inhibition threshold was set to $th_i = -1$ for all neurons and muscles.

Neurons of classes AVA and AVB generate a burst of five pulses ($N_{burst} = 5$) as a result of an increase of input activity above the excitation threshold. The remaining neuron types generate a single pulse ($N_{burst} = 1$).

The oscillator block of the cells belonging to classes VA, VB, DA, DB, VD, DD was inactivated ($t_{osc} = 0$,$t_{phi} = 0$). These neurons fire action potentials as a result of the simultaneous activation of a sufficient number of excitatory synapses, and do not display pace maker behaviour. Neurons AVA and AVB are configured to generate rhythmic activity every 360 ms ($t_{osc} = 360ms$) during forward and backward locomotion respectively.

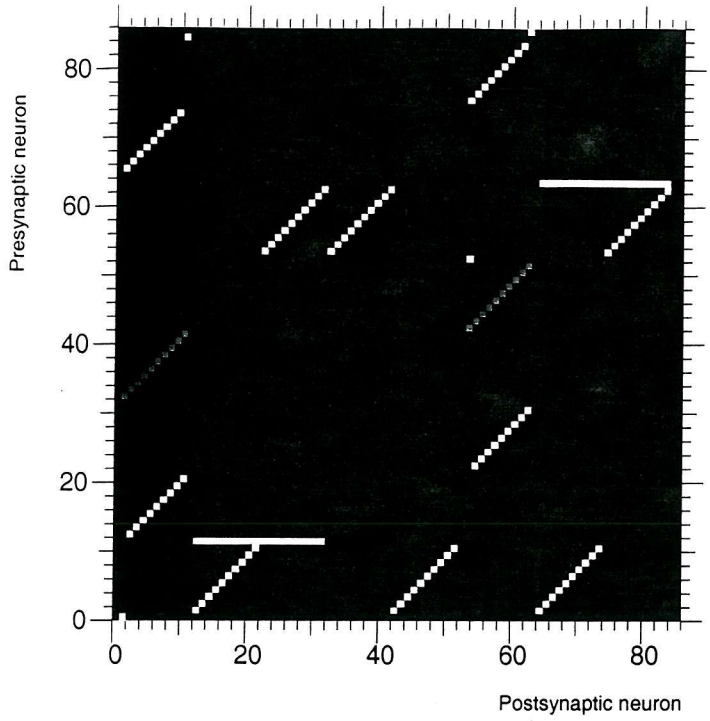
Muscles (MSC and MSD) fire continuously after the first action potential is triggered by input activity ($N_{burst} = -1$). Effectively, they are configured to generate bursts of infinite length. Their oscillator blocks were inactivated ($t_{osc} = 0$). After activation, muscle contraction continues until inhibition truncates the ongoing sequence of spikes.

The duration of the activity pulses (t_{ap}) and post-pulse refractory period (t_{ref}) is set to 1 and 2 ms respectively in neurons and to 10 and 5 ms in muscles.

Synapses in the model are configured with one of four possible combinations of synaptic delay, weight, and synaptic activation duration (see table 7.5 for complete listing). Three synaptic parameter sets with excitatory properties are defined (top three in table 7.5); the first configuration is used for synapses from NRV, NRD, TSV and TSD to muscles. The second corresponds to connections from VB, DB, VA and DA onto muscles. The third synaptic type is used for the remaining connections. All inhibitory synapses are configured with ($w_{syn} = -1$, $t_{del} = 1\,msec$, $t_{dur} = 1\,msec$).

Figure 7.10 shows the connectivity matrix of the network in figure 7.1, which follows the synaptic patterns described in [97]. White dots indicate excitatory synapses whereas gray dots mark inhibitory synapses.

To elicit different behaviours, the topology and parameters of the neurons in the



Wave number	Neuron Class
0	NRV
1-10	MSCV1-10
11	AVB
12-21	VB1-VB10
22-31	DB1-DB10
32-41	VD1-VD10
42-51	DD1-DD10
52	NRD
53-62	MSCD1-MSCD10
63	AVA
64-73	VA1-VA10
74-83	DA1-DA10
84	TSV
85	TSD

Figure 7.10: Connectivity matrix of the model

model remained unchanged. Only sources of signals driving the locomotory circuit (namely NRV, NRD, TSV, TSD and AVB and AVA) are activated or inactivated as necessary to elicit different motor patterns. These driving signals are generated, in the worm, by other subcircuits of the nervous system which have not been modelled here.

7.5.3 Forward locomotion

Driving signals

C. elegans locomotes forward by bending its head dorsally and ventrally and propagating the body curvature towards the tail [108].

Forward movement is generated in the model by the coordinated activation of the AVB neuron and units NRV and NRD. The timing in these units is set to the values determined experimentally from CCD recordings. AVB is configured to generate bursts of pulses with a period of $t_{osc} = 360\ ms$. The pattern of activity in the NRV and NRD units corresponds to two out of phase trains of pulses with an interpulse

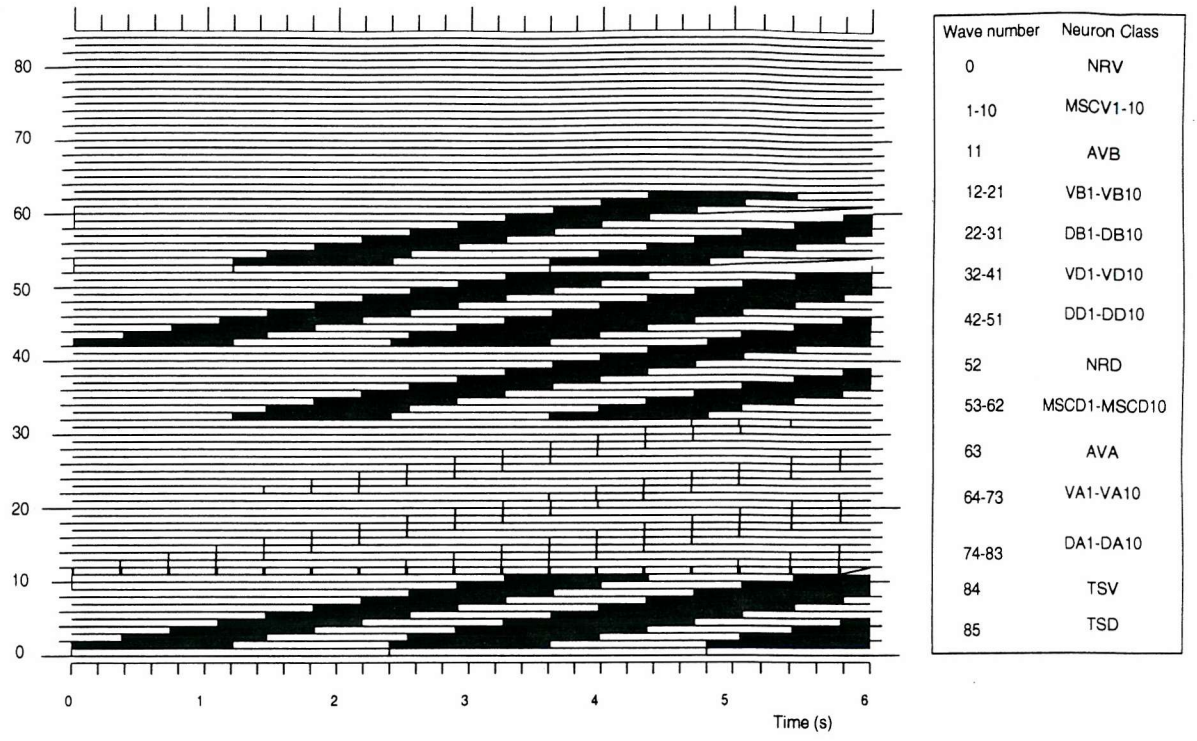


Figure 7.11: Neuron activity during forward locomotion

period of 2.4 s ($t_{osc} = 2.4\ s$). The time lag between NRV and NRD was set to 1.2 s.

NRV and NRD account for activity triggered by the bending of the head, which was observed concomitantly with forward locomotion (see figure B.1 in Appendix B). NRV generates pulses during a ventral bend of the head, whereas NRD fires during a dorsal bend. While locomoting forwards, neuron AVA and units TSD and TSV remain inactive ($t_{osc} = 0$).

The results of the simulation of forward locomotion are shown in figure 7.11.

Shift-register description of forward locomotion

The neuron classes participating in forward locomotion are VB, DB, VD, DD, AVB, NRV and NRD. Understanding of the mechanisms underlying forward locomotion in the model is aided by the shift-register analogy of figure 7.12. VB and DB neurons function as biological equivalents of AND gates in the model, which sense the state of nearby muscles (contracted/relaxed) and propagate the contraction towards the tail. In this analogy, muscles (MSCV and MSCD) are the counterparts of the flip-flops in the shift-register. They contract as a result of the onset of an action

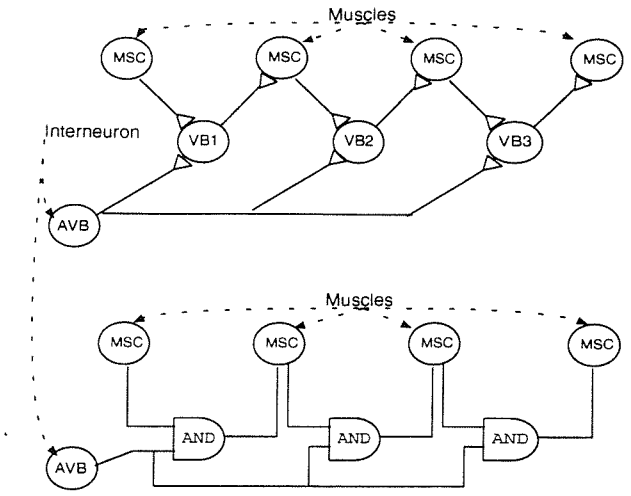


Figure 7.12: Schematic representation of the combined function of AVB and VB cells in the model

potential in the motoneurons VB and DB, and remain contracted until inhibition causes their relaxation.

Inhibitory neurons VD and DD, provide cross-inhibition between contralateral muscles. VD neurons become active whenever the muscles in the ventral side are activated and inhibit the muscles at the opposite (dorsal) site of the body, forcing them to stop the contraction. DD neurons have a similar effect but are activated by the dorsal side and inhibit the ventral muscles.

Clocking in this biological shift-register is provided by the AVB interneuron. Each burst of pulses generated by AVB triggers the propagation of muscle contraction further towards the tail.

The overall sequence of events during forward locomotion as seen in figure 7.11 is as follows:

- The head bends ventrally, activating the NRV unit, contracting the first ventral muscle (MSCV1) and triggering EPSPs in neuron VB1.
- A burst in AVB (central clock in the model), activates neuron VB1 which, in turn, contracts its nearby ventral muscle MSCV2, effectively propagating the state of MSCV1.
- Subsequent bursts from AVB, further propagate muscle states towards the tail.
- VD neurons, sensing muscle contraction in the ventral site, are activated and

inhibit dorsal muscles, post-synaptic to VD.

- After 1.2 s, the head bends dorsally, activating the NRD unit, contracting the first dorsal muscle (MSD1) and silencing NRV.
- A burst in AVB, activates neuron DB1, which propagates the state of MSD1 to MSD2.
- DD neurons are activated by dorsal muscle contraction and inhibit the ventral muscles located in the initial segment of the body.
- Subsequent bursts of AVB propagate dorsal and ventral muscle contraction towards the tail.
- After 2.4 s the head bends ventrally and the cycle restarts.

This model assumes that VB and DB neurons provide unidirectional propagation of activity (from head to tail). This functional polarization was suggested by the observation of the anatomical particularities of neurons VB and DB [96] (see the diagram in figure 7.9).

VB and DB neurons are bipolar, having all synapses onto postsynaptic neurons in a long axon extended towards the tail, whereas all synapses from presynaptic neurons are located in a short dendrite extended towards the head [97]. This pattern of synapses supports the idea of a directional propagation of excitation, which is also functionally consistent. In addition to the anatomical evidence, laser ablation experiments have confirmed that VB and DB are required for the propagation of head-to-tail muscle contraction waves [113].

The capability of VB and DB neurons to sense contraction in nearby muscles was introduced in the model by the addition of connections from muscles to VB and DB neurons. These are not anatomically identifiable connections, rather, a convenient way to simulate the stretch receptors. There is genetic evidence of the importance of stretch receptors in the generation of locomotion. Mutant unc-8, which is thought to carry a mutation affecting mechanosensation, shows abnormal locomotion [151].

7.5.4 Backward locomotion

Driving signals

C. elegans locomotes backwards by bending its tail dorsally and ventrally and propagating the curvature towards the head [95].

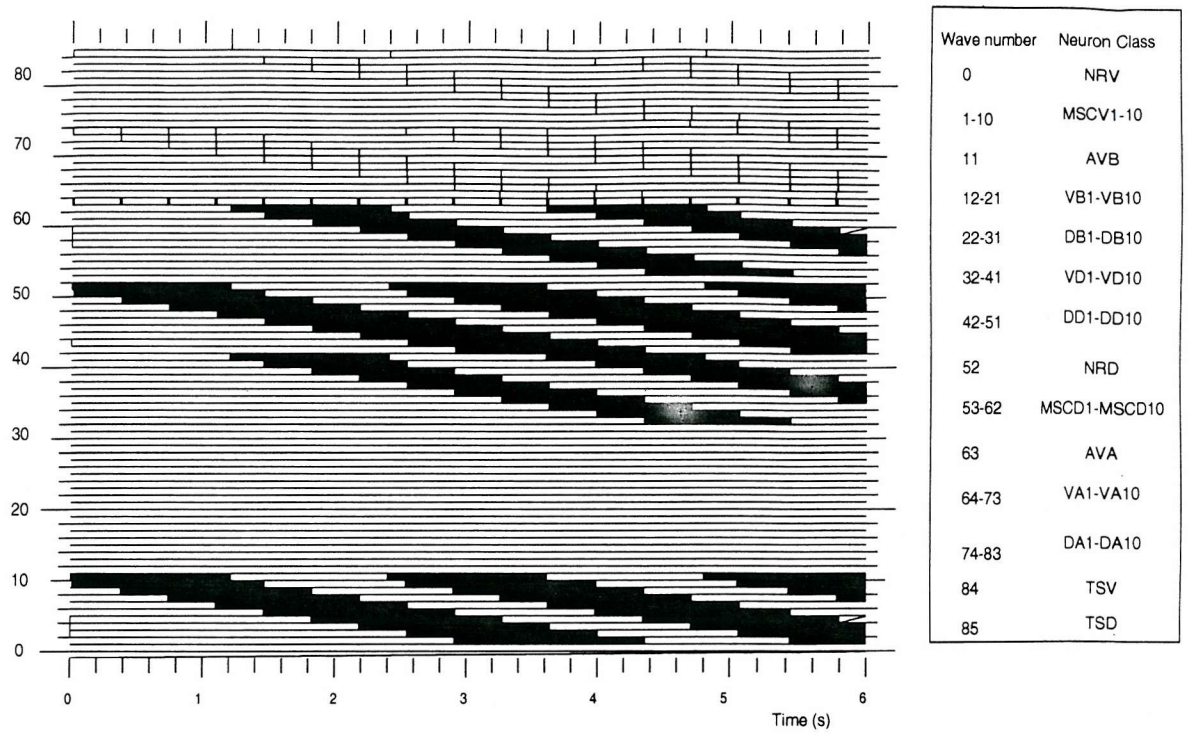


Figure 7.13: Neuron activity during backward locomotion

The results obtained in a simulation of backward locomotion are shown in figure 7.13. The parameters of AVA and AVB in forward locomotion have been exchanged to generate backward locomotion. AVA generates a burst of pulses every 360 ms ($t_{osc} = 360\ ms$) whereas AVB remains inactive ($t_{osc} = 0$).

NRV and NRD units have also exchanged configuration parameters with TSR and TSD with respect to forward locomotion. TSV and TSD model the activity associated to dorsal bending of the tail, respectively. NRV and NRD remain inactive during backward locomotion.

Events during backward locomotion

The neuron classes participating in backward locomotion are VA, DA, VD, DD, AVB, TSV and TSD. Figure 7.9 shows that, anatomically, VA/DA can be seen as a 180 degrees rotation of VB/DB neurons. Neurons VA,DA,VB and DB share the same spatial arrangement of synapses (long axon with synapses to postsynaptic neurons and short dendrite with synapses from presynaptic neurons). In the model, the function of neurons VA and DA is equivalent to that attributed to VB/DB

neurons during forward locomotion, they propagate muscle contraction. However, as suggested by their anatomy, neurons VA and DA propagate contraction towards the head whereas neurons VB and DB propagate muscle activity towards the tail.

Neurons VD and DD carry out the same function during forward and backward locomotion; they inhibit contralateral sites to allow body torsion.

Neuron AVA takes on the role of central clock during backward locomotion and AVB (which acted as clock during forward locomotion) remains inactive.

The circuit controlling the movement of the tail, which activates the signal source units TSV and TSD, is not included in the model. As the worm propagates backwards, it often follows the groove created during forward motion. The curvature induced by the groove could activate stretch receptors which provide the input to TSV and TSD, but this possibility has not been confirmed experimentally.

The sequence of events during backward locomotion is similar to that seen during locomotion,

- The tail bends ventrally, activating the unit TSV unit, contracting the ventral muscle closest to the tail (MSCV10) and triggering EPSPs in the VA10 neuron.
- A burst in AVA (central clock in the model), triggers VA10 which, in turn, contracts its nearby ventral muscle MSCV9.
- Subsequent bursts from AVA, propagate muscle contraction further towards the head.
- VD neurons, sensing muscle contraction in the ventral site, inhibit dorsal muscles.
- After 1.2 s, the tail bends dorsally, activating the unit TSD, contracting the dorsal muscle closest to the tail (MSD10) and inactivating TSV.
- A burst of pulses in AVA, trigger the activation of DA10 which contracts muscle MSD9.
- DD neurons are activated by dorsal muscle contraction and inhibit the ventral muscles located in the final segment of the body.
- Subsequent bursts of AVA propagate dorsal and ventral muscle contraction towards the head.
- Upon ventral bending of the tail, the sequence of events restarts.

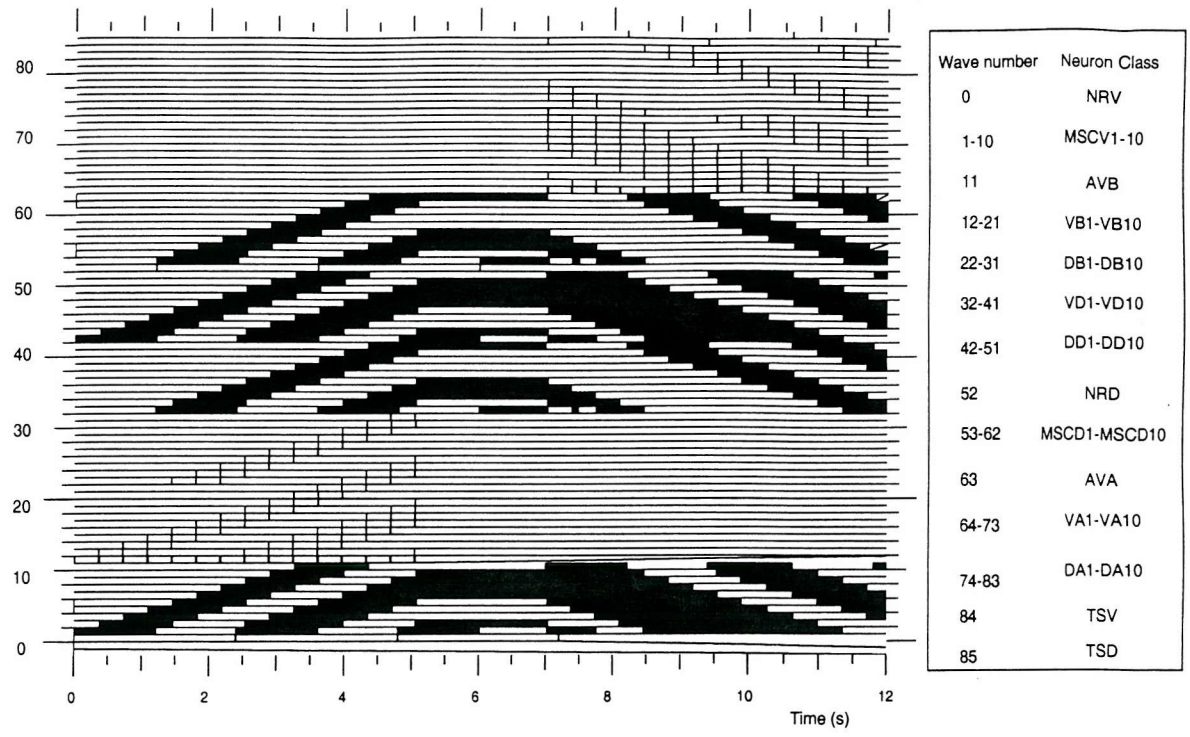


Figure 7.14: Neuron activity during reversal

7.5.5 Reversal

C. elegans is able to switch between forward and backward locomotion without abrupt changes in the sinusoidal shape of its body. Figure B.3 in Appendix B shows a reversal.

Three phases can be distinguished in figure 7.4; forward locomotion, halt and backward locomotion. Up to frame number 75 ($t < 3\ s$), the nematode propagates forward. Between frames 75 and 90 ($3\ s < t < 3.6\ s$), it pauses mantaining the curvature of the body. From frame 90 onwards ($t > 3.6\ s$), it locomotes backwards.

Figure 7.14 shows the waveforms corresponding to reversal. The forward locomotion phase is achieved by configuring the driving signals in the model as in section 7.5.3. The halt phase is started by the inactivation of AVB ($t_{osc} = 0$). The lack of activity in the central clock, AVB, stops the propagation of the muscular contraction wave and takes the system into a static phase.

To reverse, AVB remained inactivated ($t_{osc} = 0$) and AVA was activated ($t_{osc} = 360\ ms$). Units TSV and TSD were also activated as in section 7.5.4, to

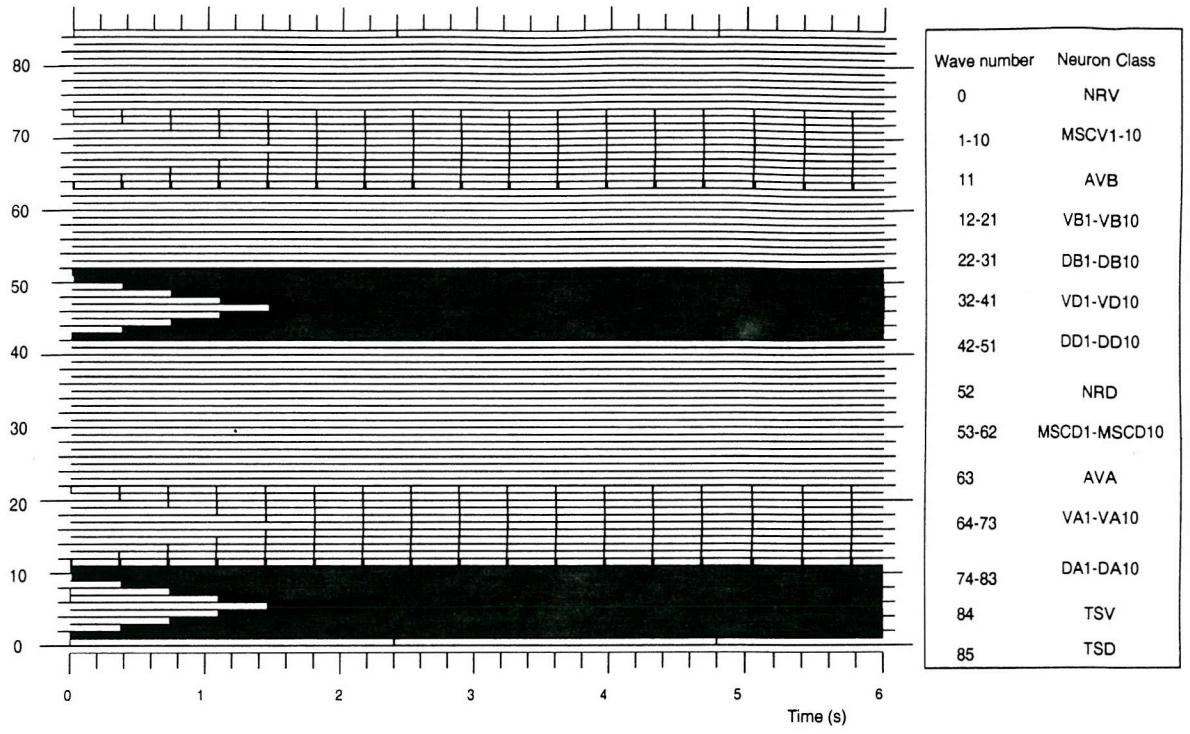


Figure 7.15: Neuron activity during coiling

account for ample tail bending.

7.5.6 Whole body bending

During periods of absence of locomotion, *C. elegans* is able to perform a whole body bending, adopting a "U" position. To achieve this shape, the spatial pattern of muscle contraction must be substantially different from that required for normal locomotion. For a ventral bending, all ventral muscles must be contracted whereas dorsal muscles should be relaxed (the opposite is the case for dorsal bending)

$$MSCV_n = \overline{MSCD_n} \tag{7.10}$$

where $MSCV_n$ denotes the state of ventral muscles which, to achieve whole body bending, must be in the negated state of dorsal muscles, $MSCD_n$. This was confirmed by the mechanical model (figure 7.8) which coils as a result of such a pattern of muscle activation.

Figure 7.15 shows the results of the simulation of ventral bending. For this

Experiment	TSx	NRx	AVA	AVB
forward loc.	pace maker	silent	silent	pace maker
backward loc.	silent	pace maker	pace maker	silent
direc. change	pace maker	pace maker	pace maker	pace maker
bend	silent	NRV pace maker, NRD silent	silent	pace maker

Table 7.7: Configuration of the stimulus units

purpose, both AVB and AVA neurons are configured to generate rhythmic activity.

Coiling is initiated by a ventral bend of the head and the tail. This is introduced in the model by the activation of NRV (signal 0 in figure 7.15) and TSV (signal 85 in figure 7.15) at $t = 0\,msec$. The simultaneous activation of NRV/TSV and rhythmic bursting of AVB/AVA (signals 11 and 63 in figure 7.15) propagates muscle contractions from opposite ends of the body towards the center. After 1.5 s all muscles in the ventral side have been contracted, corresponding to a ventrally curved body. Relaxation of all muscles in the dorsal side is guaranteed by the inhibition produced by DD neurons.

The time required for a bending to be completed can be controlled by adjusting the period of bursting of neurons AVB and AVA, in an analogous way to the control of speed in forward locomotion.

Table 7.5.6 summarizes the configuration of the units driving the network, TSx, NRx, AVA and AVB, to elicit the four types of behaviour treated in the previous sections.

7.6 Velocity control

C. elegans is able to adapt its propagation velocity in response to external stimuli. A similar behaviour is displayed by the model; since the firing frequency of AVB determines the speed of propagation of the contraction wave along the body, changes in the firing rate lead to predictable variations of the locomotion velocity.

Figure 7.16 shows the waveforms obtained for several values (50 ms, 150 ms, 250 ms and 350 ms) of the inter-burst period, t_{osc}, of neuron AVB. Figure 7.17 plots the velocity of muscle contraction propagation as a function of the AVB neuron inter-burst period. An increase of the AVB firing frequency leads to a faster propagation of muscle contraction towards the tail.

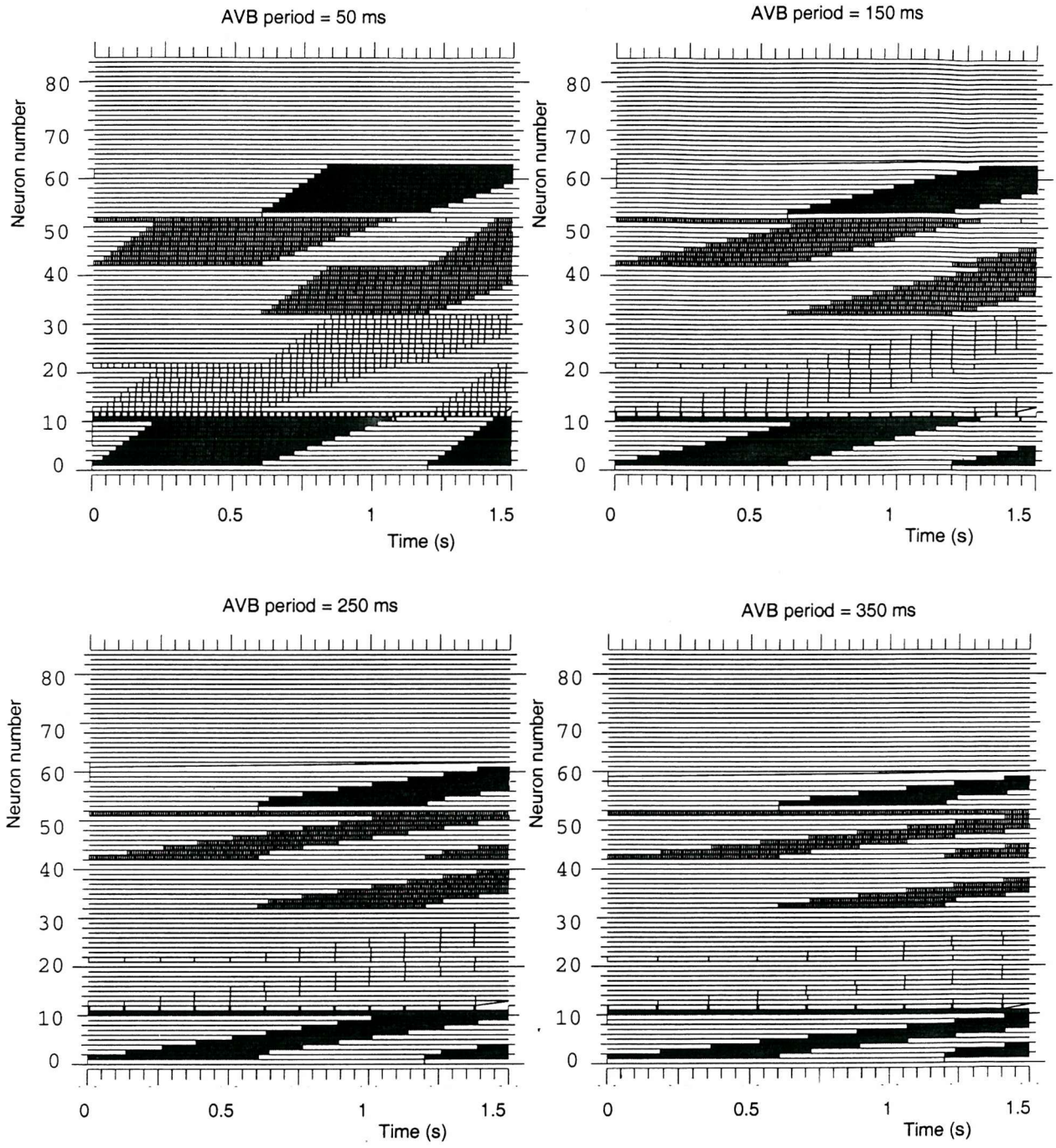


Figure 7.16: Forward locomotion for several values of the inter-burst period in the AVB neuron

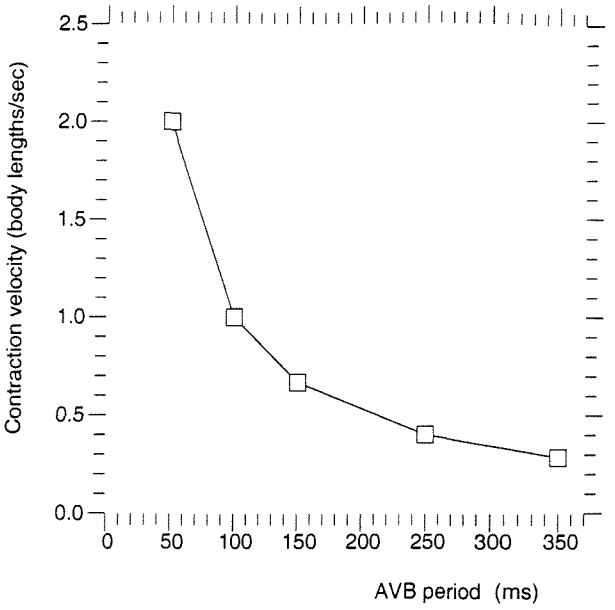


Figure 7.17: Velocity versus inter-burst period in neuron AVB

Type of Defect	Abnormal behaviour
Ablation of AVB	No forward movement. Backward movement still possible
Ablation of AVA	No backward movement. Forward movement still possible
Decreased GABA synthesis	Contraction of contralateral muscles. Impaired locomotion
Abnormal connection of AVA	Functional forward movement. Coiling when backward

Table 7.8: Several ablation and mutation experiments considered for validation of the model

7.7 Model validation with laser ablated and mutant worms

It has been shown in previous sections that the model is able to replicate four common types of locomotory behaviours (forward, backward, reversal and whole body bending).

Additional experimental data, including observations of individuals having undergone laser ablation of identified neurons [103] and mutations affecting either the topology or the neurotransmitters in the locomotory nervous system [100], were compared to model predictions for further validation. Table 7.8 summarizes the experimental results considered in the following sections.

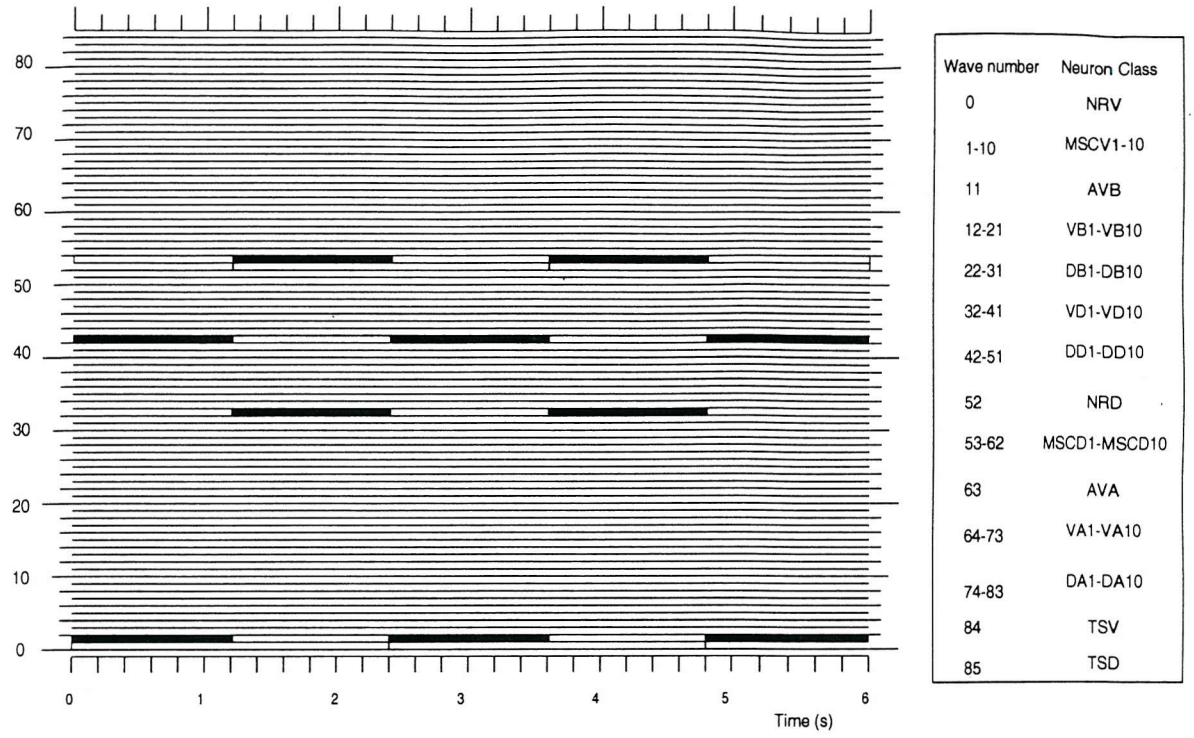


Figure 7.18: Effect of laser ablation of AVB on forward locomotion

7.7.1 Ablation of AVB/AVA neurons

Laser ablation of AVB and AVA neurons in *C. elegans* impairs forward and backward locomotion respectively [152].

The parameters of the oscillator block in the AVB neuron in the model were modified, $t_{osc} = 0$, in order to simulate its ablation. The inactivation of its oscillator, renders AVB inactive, which is functionally equivalent to its removal with a laser beam.

Figure 7.18 shows the results of the simulation of the forward locomotion configuration after the inactivation of AVB. Units NRV and NRD fire action potentials (waveforms 0 and 52), accounting for the effect of ventral and dorsal head bending. They trigger contraction in the first segment of ventral and dorsal muscle cells (signals 1 and 53) onto which NRV and NRD synapse.

Due to the absence of AVB, the contraction of these muscles is not propagated along the body, a spatial wave is not created and forward locomotion is impaired.

An analogous result is obtained when backward locomotion is simulated after

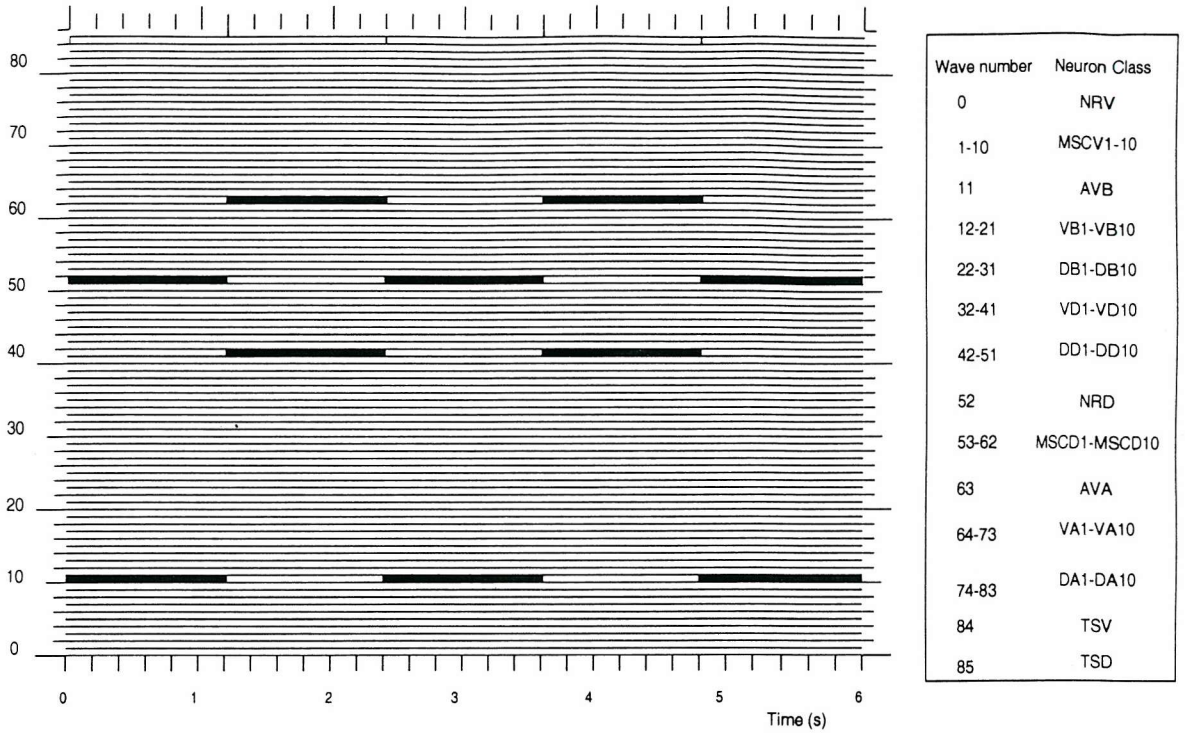


Figure 7.19: Effect of laser ablation of AVA on backward locomotion

inactivation of AVA (simulation results shown in figure 7.19). When AVA is inactivated, $t_{osc} = 0$, bending of the tail triggers contraction in the segment of muscles closest to the tail, cells MSCV10 and MSD10 (waveforms 10 and 62 in figure 7.19). However, due to the absence of activity in the interneuron AVA, this contraction is not propagated towards the head and backward locomotion is impaired.

In good accordance with experimental results, the ablation of AVB and AVA neurons disrupts totally forward and backward locomotion in the model.

7.7.2 Defective GABA synthesis mutation

Neurons belonging to classes VD and DD are thought to provide GABA mediated contralateral inhibition [153] [108] to allow body bending.

Muscle contraction in the body during locomotion obeys the condition imposed by expression 7.10, i.e. the state of the n^{th} ventral muscle (relaxed/contracted) is opposite to the state of the n^{th} dorsal muscle.

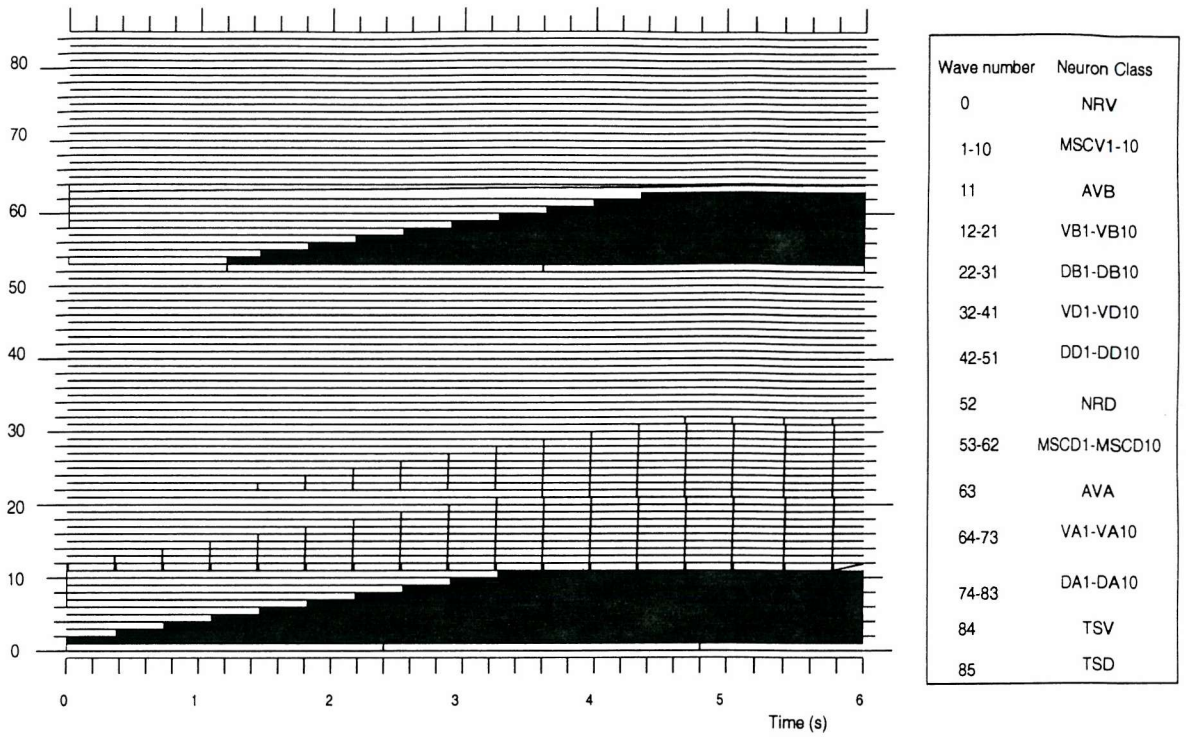


Figure 7.20: Simulation of forward locomotion in a model lacking VD and DD neurons

Abnormal locomotion has been reported in mutant worms which are defective in the production of GABA due a mutation affecting the synthesis and release of this neurotransmitter [101]. As a result of the reduced release of GABA in VD and DD neurons, contralateral muscles contract simultaneously, making propagation impossible and causing a shrinkage of the body [101]. The condition 7.10 does not hold in this case, instead

$$MSCV_n = MSCD_n \tag{7.11}$$

where $MSCV_n$ is the state of the n^{th} ventral muscle and $MSCD_n$ that of the n^{th} dorsal muscle.

Figure 7.20 shows the results obtained in a simulation of forward locomotion with a model lacking contralateral GABA mediated inhibition. The loss of function of neurons VD and DD has been introduced in the model by rising their excitation threshold ($th_e = 100$). With a sufficiently high value of th_e, VD and DD neurons

never reach the firing threshold and they remain inactive throughout the simulation.

Ventral bending of the head, at $t = 0\ s$, activates unit NRV and muscle MSCV1. Subsequent activation of interneuron AVB propagates ventral contraction (waveforms 1 to 10 in figure 7.20). At $t = 1.2\ s$, dorsal bending of the head activates NRD and the dorsal muscle MSCD1. Interneuron AVB causes propagation of dorsal contraction (waveforms 53 to 62).

In the model incorporating VD and DD neurons, simultaneous contraction of muscles at opposite sites of the body is not possible due to mutual inhibition. Thus, a spatial periodic pattern of activated and inactivated muscles is created in each body side. In the VD/DD inactivated model, both ventral and dorsal muscles become simultaneously contracted.

After 5 s, all muscles in the dorsal and ventral sides are contracted. The release of the contraction is not possible since there is no functional inhibitory mechanism. The network remains in this state indefinitely, forcing the worm to remain static.

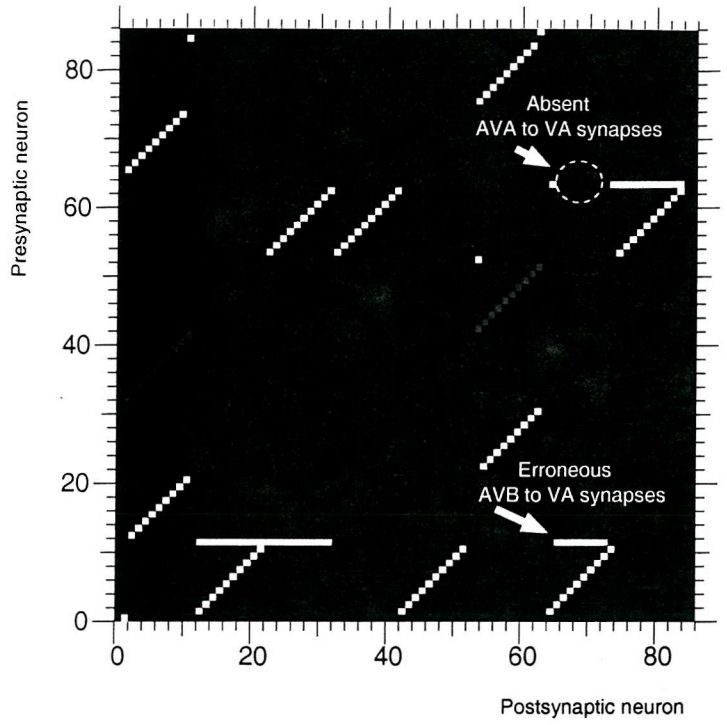
7.7.3 Mutation induced abnormal topology

Several mutations in *C. elegans* have been reported to result in abnormal connectivity between neurons in the locomotory nervous system [100]. The result of the anomalous topology is an abnormal locomotion.

In the unc-4 mutant, the connections from the interneuron AVA to the motorneurons VA are absent. Instead, the AVB interneuron provides synaptic input to VA neurons [100].

The observable effect in locomotion is the impossibility of generating backward movement. When the head of the worm is touched to elicit backward locomotion, the body coils instead of locomoting backwards. However, touching its tail triggers an abnormal (but still functional) forward locomotion [100].

To simulate this effect, the connectivity matrix of the model was altered, accommodating the new AVB-VA connections and removing the AVA-VA synapses. Figure 7.21 shows the new connectivity matrix, highlighting the synaptic changes in the model with respect to those present in the wild-type animal. All parameters, with the only exception of the new connections, were set as in the simulation of forward and backward locomotion described in sections 7.5.3 and 7.5.4.



Wave number	Neuron Class
0	NRV
1-10	MSCV1-10
11	AVB
12-21	VB1-VB10
22-31	DB1-DB10
32-41	VD1-VD10
42-51	DD1-DD10
52	NRD
53-62	MSCD1-MSCD10
63	AVA
64-73	VA1-VA10
74-83	DA1-DA10
84	TSV
85	TSD

Figure 7.21: Connectivity matrix of the model of the unc-4 mutant with abnormal topology

Impaired backward locomotion

Figure 7.22 shows the results of the simulation of backward locomotion with the modified model. At $t = 0\ s$, the two ventral muscles closest to the tail (MSCV10 and MSCV9, waveforms 9 and 10) contract. This contraction, however, does not propagate further towards the head (downwards from waveforms 10 and 9), as required for backward locomotion. At $t = 1.2\ s$, dorsal muscle MSD10 contracts and its contraction progressively propagates towards the head. At $t = 5\ s$, 8 out of the 10 dorsal muscles (MSCD1 to MSCD8) are simultaneously contracted and remain in this state indefinitely. The generalized activation of the dorsal muscles added to the relaxation of their ventral counterparts, causes a dorsal coiling.

The reason for the inactivity of ventral muscles is the abnormal connectivity between VA neurons and AVA and AVB neurons. In the wild type animal, activation of neuron AVA during backward locomotion triggers activity in VA neurons which propagate ventral muscle contraction towards the head. In the unc-4 mutant, VA neurons do not receive input from AVA but from AVB. Neuron AVB

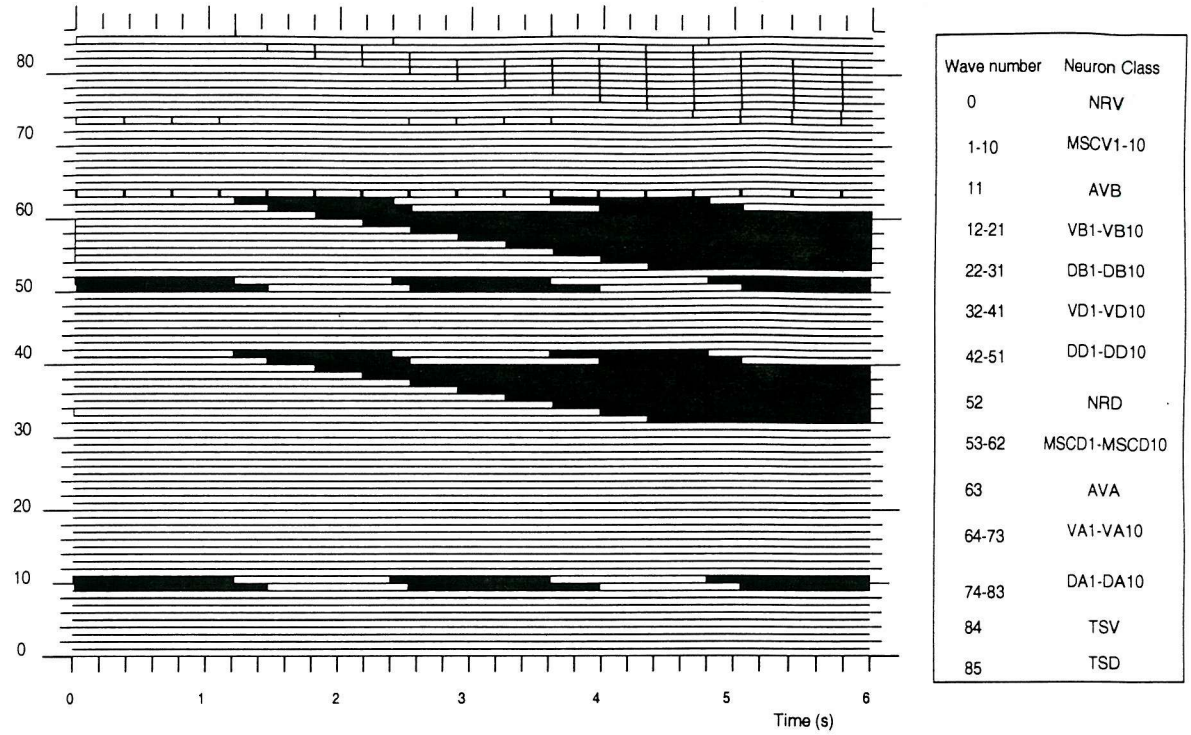


Figure 7.22: Simulation of backward locomotion in a model of the unc-4 mutant worm

remains inactive during backward locomotion and, as a result, ventral muscles fails to contract. A reduced activity in the ventral side leads to a decreased inhibition to the dorsal side through the contralateral inhibition carried out by neurons VD. As a consequence, dorsal muscles are not inhibited and the periodic spatial pattern of contraction, typical of normal locomotion, is not generated, resulting in a complete contraction of all muscles in the dorsal side.

Forward locomotion

Forward locomotion is anomalous but still functional in the abnormally connected unc-4 mutant [100]. Due to the lack of a quantitative description of this abnormalities, it was assumed that an abnormal forward locomotion involved a deviation of the pattern of muscle contraction from the wild-type animal but, to allow forward movement, a head to tail contraction wave should be present.

Figure 7.23 shows the simulation of forward locomotion. The results show a comparatively increased degree of muscle contraction in ventral muscles (waveforms 1 to 10) with respect to dorsal muscles (waves 53 to 62). This is the result of

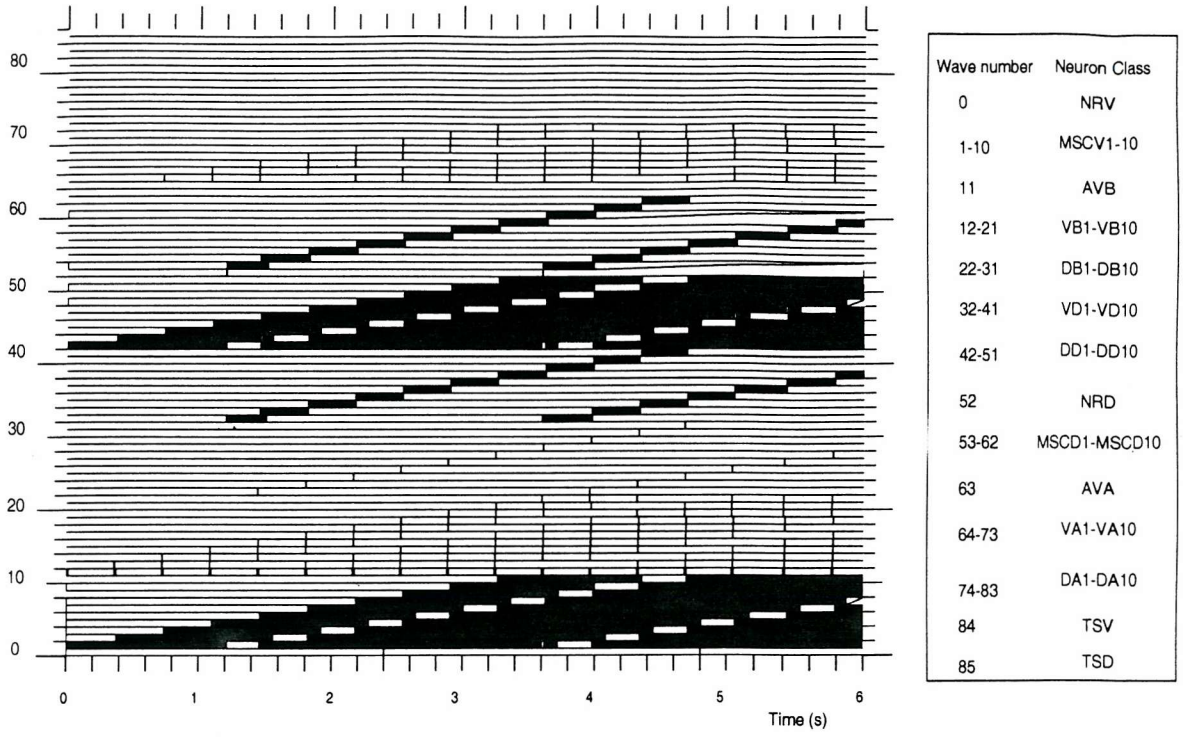


Figure 7.23: Simulation of forward locomotion in a model of the unc-4 mutant worm

synaptic activity provided by VA neurons. In the wild-type animal, during forward locomotion, ventral muscles receive excitatory synaptic input exclusively from VB neurons. In the unc-4 mutant, excitatory input is relayed by both VB and VA neurons, which increases the degree of activation of ventral muscles with respect to dorsal muscles.

However, head to tail waves of muscle contraction can be seen, indicating that forward locomotion, though with an uncompensated dorsal-ventral excitation, is possible in the model. This is in accordance with experimental observations.

7.8 Conclusions

A model for the locomotory nervous system of *C. elegans*, based on the event driven neuron model, has been proposed. This model is able to generate normal locomotion (forward and backward locomotion, forward/backward switching and whole body bending). It successfully predicts experimental observations with mutant and laser ablated worms (ablation of AVA/AVB, abnormal GABA release in VD/DD neurons

and anomalous topology in unc-4 mutant).

As electrophysiological data of *C. elegans* are still scarce, the model has aimed at reproducing the gross locomotion behaviour.

Constraining the model by the synaptic topology and accepting some assumptions on the parameters have sufficed to create a working model. Due to the simplistic nature of discrete models of neurons, the parameter space can be searched more easily than those required for Hodgkin-Huxley models [154].

Even with the parameter uncertainty, the model displays a rich set of realistic locomotion behaviours which require a limited set of constraints in the parameters. It is clear, though, that even with a subcircuit of a relatively small nervous system, experimental data are required for the creation of a working model with realistic parameters.

Chapter 8

Event-driven model of the piriform cortex

8.1 Introduction

The MBED model was utilized in simulations of a small network of neurons in Chapter 7. In this Chapter, it is used to construct a large scale model (10^5 neurons) of the piriform cortex. The aim is to demonstrate the adequacy of the MBED framework as an alternative to compartmental models in large simulations, providing an improvement in computational efficiency while retaining the capability of incorporating electrophysiological data and producing realistic results.

To this end, an MBED model of the piriform cortex including 10^5 neurons is constructed. The effect of network parameters on the dynamics of the aggregate are explored and the results compared with available experimental data and previous theoretical work. Table 8.1, reproduced for convenience from table 4.1, summarizes the three types of aggregate activity obtained experimentally by Ketchum *et al.* [124, 117] and theoretically with compartmental models [28].

The MBED model of the piriform cortex and the procedures employed to simulate field potentials and EEG recordings will be described first. Secondly, a set of test simulations are carried out which explore the dynamics of partially connected instantiations of the model in order to rule out implementation errors. Thirdly, the responses of the model to shock and random stimuli are studied, and several synaptic parameters and spatial patterns of LOT afferents are tested. Finally, the neural population sizes are modified. The balanced pyramidal-inhibitory neural pools, as proposed in [28], are substituted by a more realistic 4:1 excitatory to

Experiment	Results
Strong shock stimulus	Single wave
Weak shock stimulus	Damped oscillations
Random input	40 Hz + 5 Hz frequency components

Table 8.1: Experimental [124, 117] and compartmental modelling [28] results

inhibitory cells ratio, following Hasselmo *et al.* [133].

8.2 The piriform cortex model with homogeneously sized neural pools

The MBED model of the piriform cortex is based on the compartmental model by Wilson *et al.* [28]. Four types of cells have been included: fast excitatory pyramidal cells, fast inhibitory ($GABA_A$) cells, slow inhibitory ($GABA_B$) cells and stimulus (LOT) cells (figure 8.1).

Each one of the first three cell populations (pyramidal, $GABA_A$ and $GABA_B$) consists of a grid of 150 × 150 neurons. For clarity, these layers are depicted in separate planes in figure 8.1. However, when topological information is needed (e.g. to simulate EEG and field recordings, as will be described in section 8.3), the 3-D model of figure 8.1 is collapsed into a 2-D model where the grids with pyramidal and inhibitory cells ($GABA_A$ and $GABA_B$) are positioned in the same z-plane.

The LOT layer models the afferent activity which originates in the olfactory bulb and reaches the piriform cortex through the lateral olfactory track. The number of cells in this pool has been adjusted for each simulation in order to provide the desired degree of excitation.

The LOT units in the model represent axonal bundles rather than a fourth cell type within the olfactory cortex. They are assumed to be situated at a distant point location with respect to the rest of neurons. As a consequence, the simulation of EEG recordings incorporates exclusively the contribution of the three cell classes, pyramidals and fast and slow inhibitory, physically located in the cortical region.

Pyramidal cells possess local and long range intralayer excitatory connections (amongst pyramidal cells) and local interlayer connections (exciting nearby neurons in the $GABA_A$ and $GABA_B$ layers). Inhibitory cells ($GABA_A$ and $GABA_B$ layers) do not have intralayer connections in the model. Instead, they inhibit pyramidal

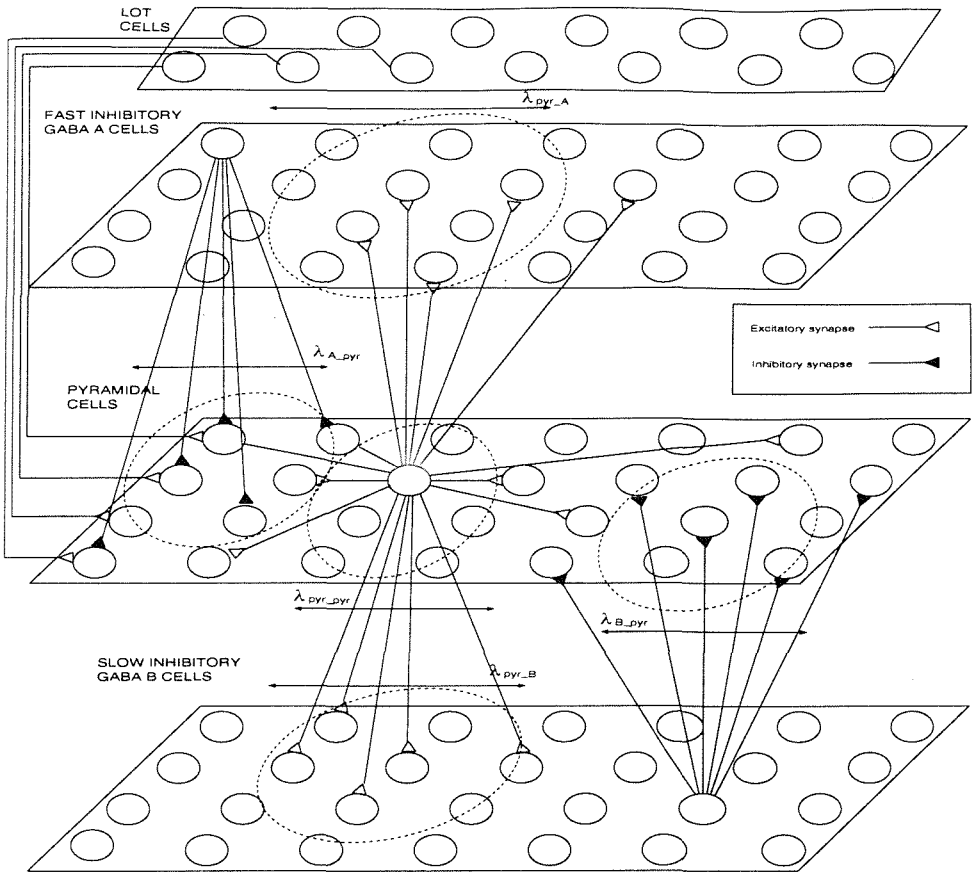


Figure 8.1: Piriform cortex model

cells by means of local connections.

The number of connections established by any one neuron is deterministically fixed and listed amongst other network parameters in table 8.2. The target neuron, j, of a synapse from neuron i, is chosen generating a random vector $\vec{d}$ of components $\{\rho, \phi\}$ (in polar coordinates), where ρ is an exponential variable and ϕ a uniformly distributed value in the range $0 - 2\pi$. The target neuron is chosen as the closest cell to,

$$\vec{p_j} = \vec{p_i} + \vec{d} \tag{8.1}$$

where $\vec{p_i}$ is the position vector for neuron i. This expression is valid for all intracortical connections (pyramidal-pyramidal, pyramidal-inhibitory and inhibitory-pyramidal)

LOT cells synapse onto pyramidal cells and introduce an external stimulus into

Neuronal parameters	
th_e	5
th_i	-1000 (burst truncation inactivated)
t_{ap}	1 ms
t_{ref}	10 ms
N_{burst}	1
t_{osc} (pyramidals and inhibitory)	0 (*inactive oscillator*)
t_ϕ (pyramidals and inhibitory)	0 (*inactive oscillator*)
t_{osc} (LOT cells, all stimuli)	3000 ms
t_ϕ (LOT cells, shock stimulus)	0 ms
t_ϕ (LOT cells, random input)	Uniform$(0 - t_{stop})$
Number of synapses per neuron	
pyramidal to pyramidal	180
pyramidal to fast inhibitory	12
pyramidal to slow inhibitory	10
fast inhibitory to pyramidals	12
slow inhibitory to pyramidals	5
Synaptic parameters	
t_{del} (pyramidal to pyr./inh.)	$(3 - 12\ ms)$
t_{dur} (pyramidal to pyr./inh.)	5 ms
w_{syn} (pyramidal to pyr./inh.)	1
t_{del} (fast inh. to pyramidal)	5 ms
t_{dur} (fast inh. to pyramidal)	9 ms
w_{syn} (fast inh. to pyramidal)	-10
t_{del} (slow inh. to pyramidal)	10 ms
t_{dur} (slow inh. to pyramidal)	150 ms
w_{syn} (slow inh. to pyramidal)	-1
t_{del} (LOT to pyramidal)	$(1 - 4\ ms)$
t_{dur} (LOT to pyramidal)	5 ms
w_{syn} (LOT to pyramidal)	1
Connection range, λ (normalized distance)	
pyramidal to pyramidal	2
pyramidal to fast inhibitory	10
pyramidal to slow inhibitory	10
fast inhibitory to pyramidals	10
slow inhibitory to pyramidals	10
LOT to pyramidals	2

Table 8.2: Numerical values of parameters in the homogeneously sized piriform cortex model

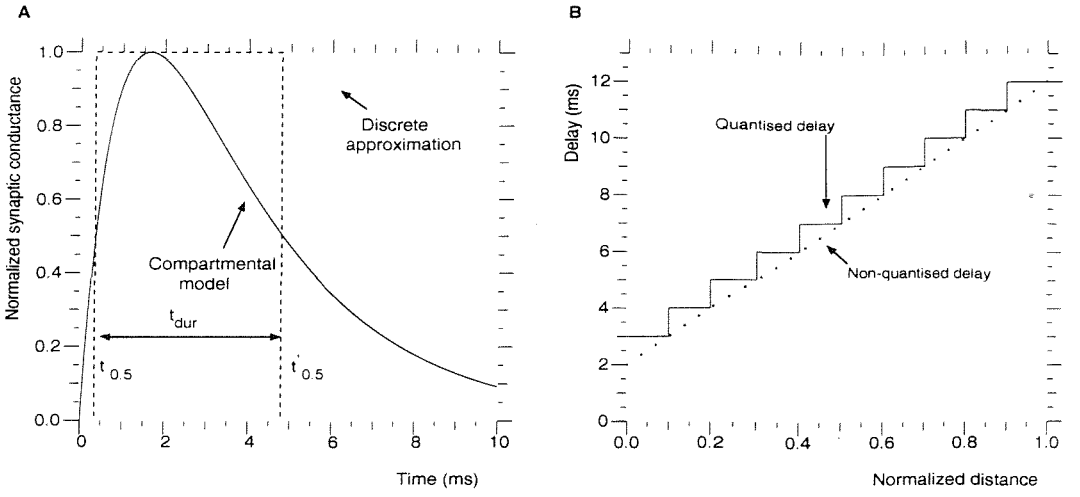


Figure 8.2: (A) Excursion of a compartmental excitatory synaptic conductance and its discrete approximation. (B) Pyramidal-pyramidal synaptic latency (t_{del}) as a function of pre to postsynaptic cell distance.

the model. The density of connections from LOT to pyramidal cells decreases exponentially from left to right in the pyramidal layer of figure 8.1. The target in the pyramidal layer of a connection from a LOT cell, is chosen as in expression 8.1 but, in this case, vector $\vec{d} = \{d_x, d_y\}$ (in cartesian coordinates) where d_x and d_y are exponential and uniform (range $0 - 1$) random variables respectively.

Numerical values for the λs of the exponential distributions are given in table 8.2 in units of normalized distance (position $(0, 0)$ corresponding to the top-left corner of the layers in figure 8.1 and $(1, 1)$ to the bottom-right corner).

Note that connections from pyramidal to inhibitory neurons and back from inhibitory to pyramidals are short range ($\lambda = 10$). The probability of establishing a connection between a pyramidal and an inhibitory neuron at a distance of 0.1 (normalized to the dimensions of the model) decreases by a factor of $1/e$ with respect to the probability of setting connections with nearby (distance ≈ 0) neurons. Pyramidal to pyramidal connections are long range ($\lambda = 2$). For this type of connections, the probability decreases to $1/e$ times the value for close neurons for a distance of 0.5, which corresponds to half the distance between opposite ends of the cortical model.

The duration of the synaptic activation (t_{dur}) and the synaptic delays (t_{del}) were set in accordance with experimentally determined values as in [28]. Figure 8.2-A compares the excursions of an excitatory synaptic conductance during a synaptic

event as modelled in a compartmental framework (solid line) and the discrete approximation used in the MBED model (dotted line). The parameter t_{dur} corresponds to the width of the pulse with onset at $t_1 = t_{0.5max}$ and falling edge at $t_2 = t'_{0.5max}$ ($t_{0.5max}$ and $t'_{0.5max}$ are the half conductance times) and rounded to the closest integer.

The parameter t_{del} was calculated as,

$$t_{del} = t_{syn} + t_{axon} \tag{8.2}$$

where the term t_{syn} accumulates the delay involved in the chemical activation of a synapse and the propagation of activity in the dendritic tree, whereas t_{axon} accounts for the delay due to the propagation of the action potential along the presynaptic axon.

Axonal delays are distance dependent. For an axon of length l, the delay is obtained as

$$t_{axon} = \frac{1}{v_{axon}} l \tag{8.3}$$

where v_{axon} is the velocity of an action potential propagating along the axon. In order to make use of the synaptic model strategy (see section 6.4.1), the number of allowed synaptic parameter sets was reduced. For this purpose several approximations were introduced. In the case of pyramidal to pyramidal and pyramidal to GABA connections, the parameter t_{del} was quantized and the number of allowed values limited to 10. The maximum delay corresponds to connections ranging the complete length of the cortex and the minimum delay to connections between nearby neurons. The remaining 8 values are equally spaced as a function of distance, totalling ten synaptic models for pyramidal-pyramidal and pyramidal to inhibitory connections. Figure 8.2-B plots t_{del} as a function of interneuronal distance. A similar quantification was used for the delay in synapses from LOT to pyramidal cells, allowing four distance dependent values for t_{del}.

The third simplification applies to connections from GABA to pyramidal cells. Since these are short range connections, the small difference between the delay introduced by the shortest and the longest axons allows a distance independence approximation. Thus, synapses from $GABA_A$ and $GABA_B$ neurons have a fixed, distance independent, t_{del} parameter (see table 8.2).

As a result of the delay quantification described above, sixteen different synaptic parameter sets (synaptic models) are used.

The synaptic efficacy, w_{syn}, of the various synaptic types in the model was set to make their relative strengths consistent with the maximal conductance of synaptic channels in compartmental models. It takes negative values for inhibitory connections ($GABA_A$ and $GABA_B$) and positive for excitatory (pyramidals).

The MBED neurons in the network are configured to fire one-spike bursts ($N_{burst} = 1$) of a duration of $t_{ap} = 1\ ms$ followed by a refractory period of $t_{ref} = 10\ ms$. Pyramidal and $GABA_A/GABA_B$ neurons fire whenever w_{sum} (weighted sum of inputs) increases above the excitation threshold. Suitable values for the excitation threshold were determined by parameter space search. Units in the LOT pool, which provide input to the model, were configured as pace makers and the firing frequency modified for each simulation to provide the various types of stimuli required. -

8.3 Simulation of field potential recordings, EEGs and power spectra

Field potentials [33, 155] and EEGs [132] are measurements of time changing electric potentials generated by neuronal activity. Field potentials are recorded with a pair of microelectrodes, one serving as a reference and the second located close to the pool of neurons under study, whereas EEGs make use of arrays of electrodes placed on the scalp. For the purpose of model validation, it is desirable to obtain simulated field potentials and EEG recordings associated to MBED network simulations. In this way, the patterns predicted by the discrete model can be compared to those obtained with compartmental models.

For the simulation of EEG recordings, a procedure similar to that described by Wilson *et al.* [28] has been followed. A number of virtual electrodes are spatially distributed forming a grid of $E \times E$ recording sites (figure 8.3). Each one of these simulated electrodes records a field potential calculated as,

$$S_{FP_i} = \sum_{j}^{J} \sum_{k}^{K} \frac{1}{d_{ij}} \delta_j(t - t_k) * h(t) \tag{8.4}$$

where S_{FP_i} is the field potential signal recorded by the i^{th} electrode, d_{ij} is the distance between the i^{th} electrode and neuron j, $\delta_j(t - t_k)$ is a delta function indicating that neuron j fired an action potential at $t = t_k$ and $h(t)$ is the field potential function recorded from a group of neurons firing nearly simultaneously.

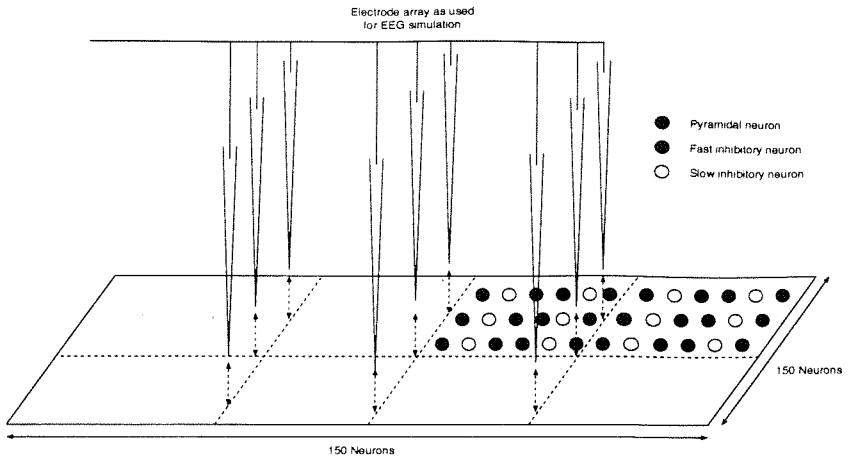


Figure 8.3: Setup used for the simulation of field recordings and EEGs

This impulse response is convolved with the train of weighted deltas to obtain the total measured potential. The summations are over the number of action potentials K generated by neuron j and over all neurons J in the network.

The impulse response, $h(t)$, as utilized for EEG and field potential estimation is given by (t in ms),

$$h(t) = \begin{cases} 0 & t < 0 \\ -5 & 0 < t < t_1 \\ 2 & t_1 < t < t_2 \\ 0 & t_2 < t \end{cases} \tag{8.5}$$

where the negative segment accounts for the negative potential, recorded experimentally during the onset of action potentials and the subsequent positive segment corresponds to the positive potential seen during repolarization of the neuronal membrane (the return to resting voltage) [24, 155].

The shape of the field potentials depends strongly on the time sequence of neuronal activations, the anatomical characteristics of the tissue and the position of the recording electrode [33]. It was found that $t_1 = 5$ and $t_2 = 12$ provided the best match between the recordings predicted by the MBED model and those obtained experimentally and with compartmental models.

Consistently with studies on the linearity of the electrical properties of living tissue [132], the EEG signal is obtained by linear combination of the field potentials,

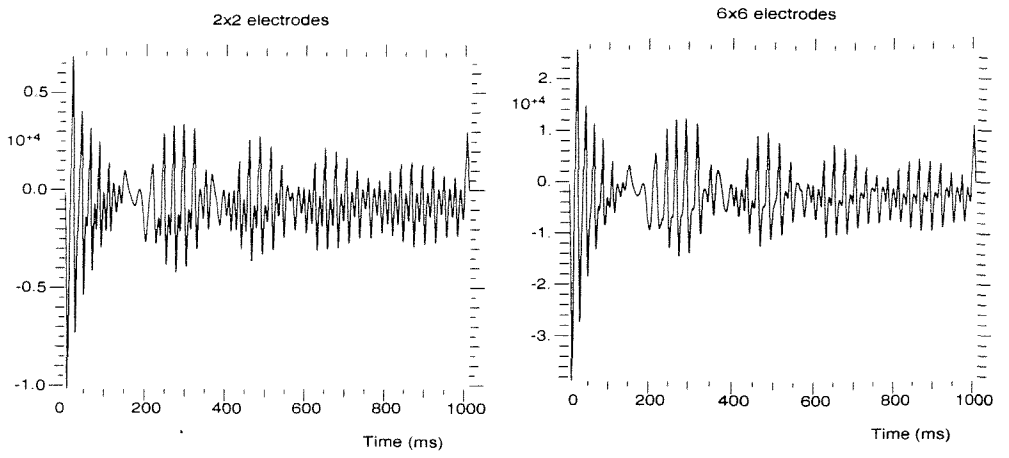


Figure 8.4: EEGs obtained with grids of 2×2 (A) and 6×6 (B) electrodes

$$S_{EEG} = \sum_{i}^{ExE} S_{FP_i} \tag{8.6}$$

where S_{EEG} is the EEG signal, S_{FP_i} the field potential recorded by the i^{th} electrode and the summation is over all the electrodes forming the $E \times E$ grid. The effect of the grid, as opposed to the single electrode recordings, is to provide a measurement of the average activity in the network.

Figure 8.4 shows the EEG calculated as in equation 8.6 for two values of E. Setting $E = 2$ results in a low spatial sampling frequency and a single peak propagating in the cortex generates several peaks in the EEG (figure 8.4-A). For $E = 6$, the sampling effect is reduced and single waves generate single peaks in the EEG (figure 8.4-B). In the following sections, EEG measurements will be carried out using grids with 10×10 electrodes.

Estimations of the EEG power spectrum were carried out following the procedure described in [156], with a Hanning window, segments of 512 samples and an equivalent sampling frequency of 1 KHz.

8.4 The partially connected model

Partially connected versions of the piriform cortex model were simulated. Firstly, the simplified topology allows better understanding of the dynamics of the model. This will be required in this Chapter to comprehend the origin of the responses of

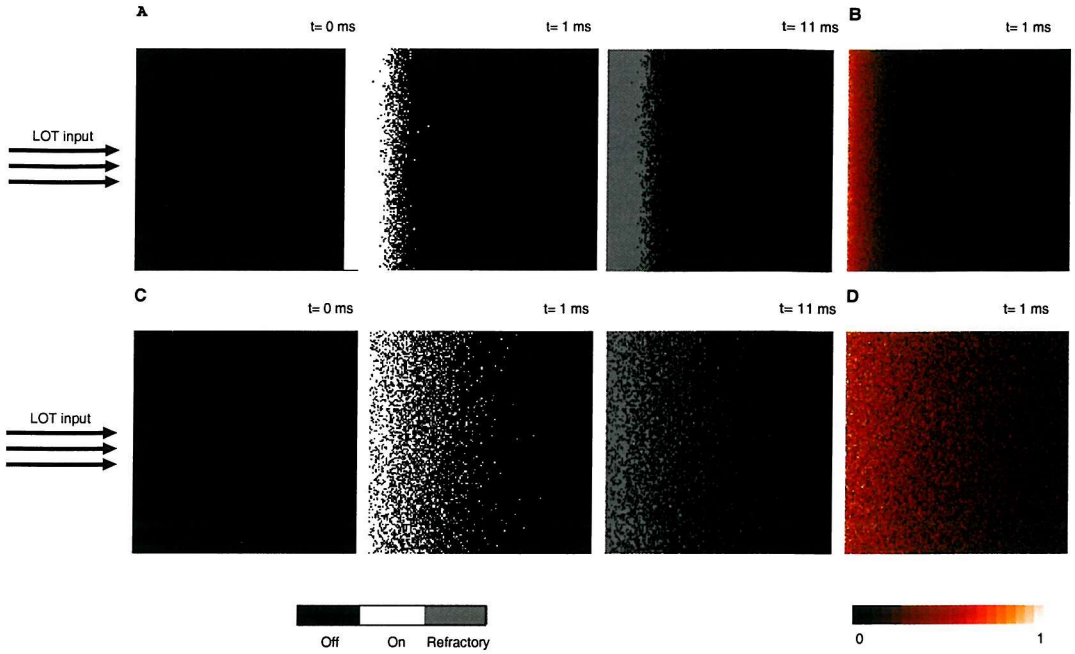


Figure 8.5: State and w_{sum} of pyramidal neurons in a partially connected model for two exponentially distributed LOT to pyramidal spatial patterns; $\lambda = 10$ (A,B) and $\lambda = 2$ (C,D)

the fully connected model to several kinds of stimuli. Secondly, implementation errors would be manifested and more easily identified in simulations of the simplified topology whereas they could be misinterpreted in the fully connected model.

8.4.1 LOT-pyramidal interactions

Figure 8.5 shows the time evolution of the states and w_{sum} of the pyramidal cells in a partially connected model. Single cells are represented as individual dots in the 2-D coloured arrays. In the state sequences (A and C), black, gray and white dots indicate neurons in state off, $refractory$ and on, respectively. Panels B and C show the normalized value of w_{sum} in all pyramidal cells, a shift from black to red indicating an increase of w_{sum}.

For these simulations, all connections with the exception of those from LOT cells to pyramidal neurons have been removed. Through these synapses, the LOT units, which provide the input stimulus in the model, are able to excite the pyramidal cells. This excitation, however, can not propagate in the pyramidal layer through

pyramidal-pyramidal connections or affect inhibitory cells by pyramidal-$GABA_A$ or pyramidal-$GABA_B$ connections, since these are not present.

Figures 8.5-A and 8.5-B show the activation of the pyramidal cells due to the simultaneous firing of 150 LOT cells at $t = 0\ ms$ each establishing 10^3 excitatory connections with pyramidal neurons. The probability of choosing a given pyramidal neuron as the target for an LOT-pyramidal synapse decreases exponentially with its distance from the LOT entry region (area on the lefthand side) with space constant $\lambda = 10$ (figures 8.5-A and 8.5-B) and $\lambda = 2$ (figures 8.5-C and 8.5-D).

At $t = 0\ ms$, all pyramidal cells remain in the initial (off) state and all LOT cells (not shown in the figures) simultaneously fire an action potential. Since the propagation delay between LOT and pyramidal cells was set in this example to a distance independent value, $t_{del} = 1\ ms$, the pyramidal layer receives the excitatory synaptic activations from the LOT cells at $t = 1\ ms$. The excitatory synaptic input to the pyramidal cells leads to the change of state depicted in the figures.

The area activated by the LOT input in the pyramidal cell layer increased in figures 8.5-C,D with respect to figures 8.5-A,B. This was the result of an increase of the mean of the exponentially distributed pattern of LOT-pyramidal connections. This heterogeneous excitation of the pyramidal layer will be relevant for the understanding of the origin of waves in the fully connected model. The higher level of excitatory input received by leftmost areas of the cortex with respect to rightmost areas, will result in waves propagating from left to right during shock and random stimulus, as described later in this Chapter.

8.4.2 Pyramidal-pyramidal interactions

To illustrate the mechanisms involved in the generation and propagation of waves in the pyramidal cell layer, the interlayer connections from/to inhibitory cells to/from pyramidal cells have been removed. This renders open the feedback loop pyramidal->inhibitory->pyramidal, ensuring that the activity seen in the pyramidal layer is the result of the LOT stimulus and the intralayer interactions, with no inhibition present. To aid in the visualization of wave genesis, the pool of LOT cells was collapsed into a single neuron with connections to one randomly chosen pyramidal cell. Figures 8.6-A and 8.6-B show the state and the value of the weighted sum of inputs (w_{sum}) of the pyramidal cells, respectively, at selected times.

At $t = 8\ ms$, only one pyramidal neuron fires, due to the single LOT cell generating its excitatory input. At $t = 11\ ms$, local excitatory pyramidal to

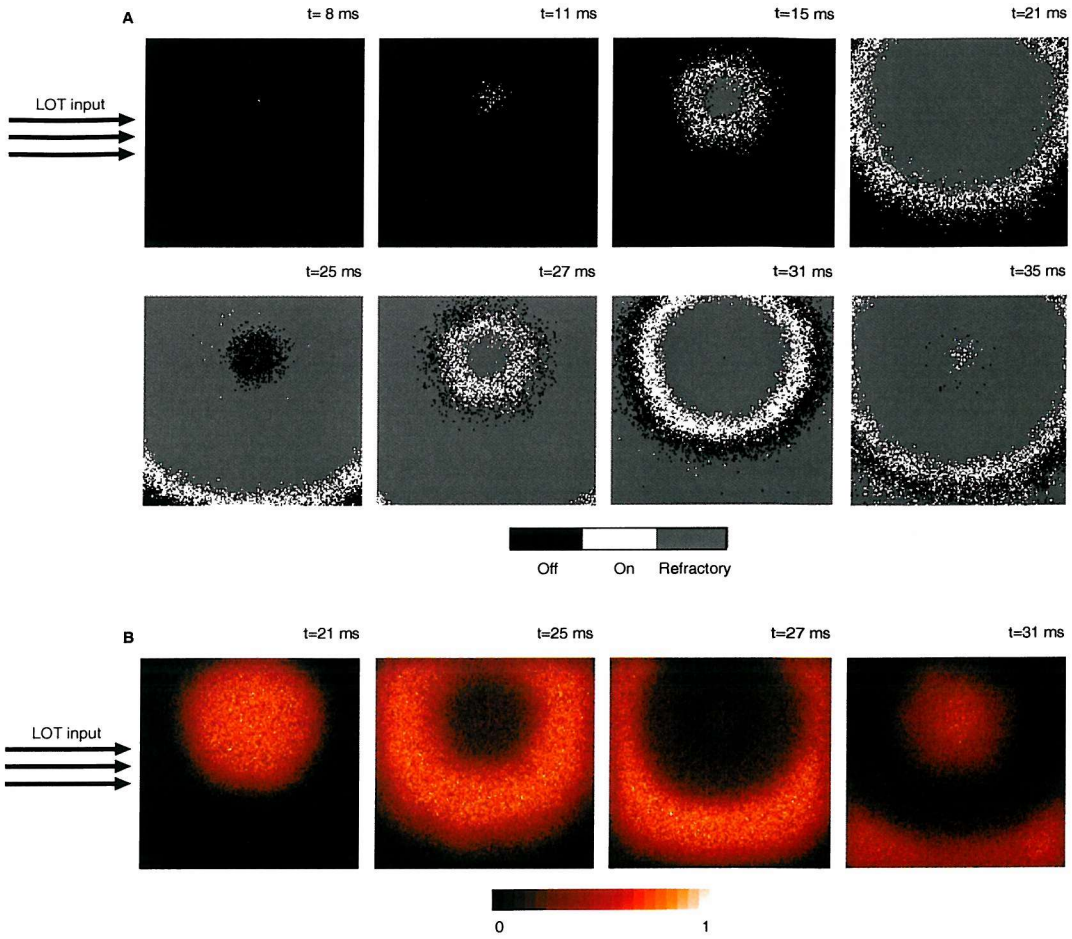


Figure 8.6: State (A) and w_{sum} (B) of pyramidal neurons in the cortical model after removal of inhibition

pyramidal connections spread the excitation. At $t = 15\ ms$, the cells located at the core of the excited patch become refractory (gray area). At $t = 25\ ms$, the wave of firing neurons has reached the borders of the cortex. Most pyramidal cells are still in refractory state. However, those which originated the wave at $t = 11\ ms$ have finished their refractory period and enter the off state. They can now be re-excited by long range axonal connections carrying action potentials from the distant wave (now at the boundary of the cortex) towards the core. In this way, a second wave is initiated.

A comparison of figures 8.6-A and 8.6-B, for instance at $t = 21\ ms$, indicates that the finite axonal propagation velocity introduces a delay in the propagation of

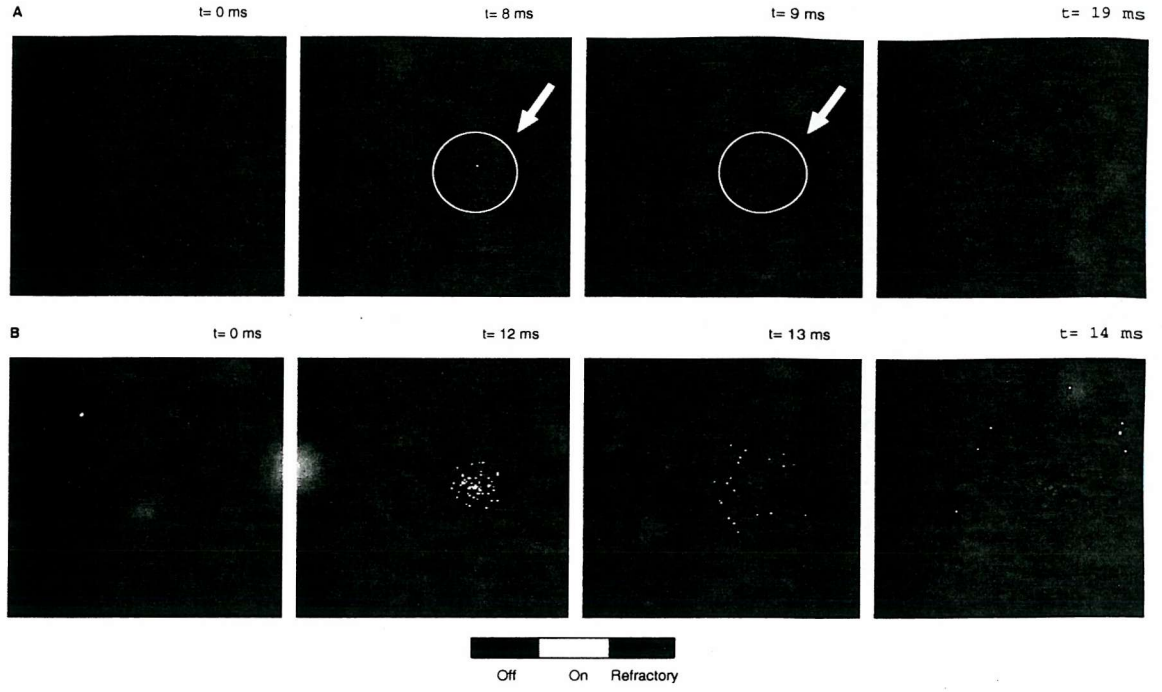


Figure 8.7: Sequence of images representing the state of pyramidal and $GABA_A$ neurons. (A) Pyramidal neurons, (B) $GABA_A$ cells

excitation between pyramidal cells. Although cells are firing in the outer shell at $t = 21\ ms$, the maximal synaptic excitation is being received at the core (see first image in 8.6-B) where the neurons have already entered refractory period and are unable to fire.

In this partially connected model, the absence of inhibition allows continuous wave generation. The addition of inhibitory connections, described in following sections, introduces attenuation and limits the number and intensity of the waves.

8.4.3 Pyramidal-GABA interactions

As in the previous section, the LOT pool has been collapsed to a single unit which excites one randomly chosen neuron in the pyramidal layer. Pyramidal-pyramidal, pyramidal-$GABA_B$ and connections from $GABA_A$ to pyramidals have been removed. Only the synapses from LOT to pyramidals and from pyramidals to $GABA_A$ remain in the model. Figures 8.7-A and 8.7-B show the states of the neurons in the pyramidal and $GABA_A$ layers, respectively.

At $t = 8\ ms$, the activity from the LOT pool arrives at the pyramidal cell layer,

activating only one neuron (see figure 8.7-A). At $t = 9\ ms$, the pyramidal neuron enters refractory state. At $t = 12\ ms$, connections between pyramidal cells and $GABA_A$ neurons trigger action potentials in those $GABA_A$ cells closest to the activated pyramidal (see figure 8.7-B). Due to the distance dependent axonal latency from pyramidals to $GABA_A$ neurons, more distant $GABA_A$ cells fire at $t = 13\ ms$ and $t = 14\ ms$.

Note that the lack of $GABA_A$ to $GABA_A$ synapses eliminates the possibility of wave generation within the $GABA_A$ layer. Only the pyramidal layer is capable of generating and sustaining cortical waves.

8.4.4 GABA-pyramidal interactions

Figure 8.8 shows the time evolution of the states and the normalized w_{sum} state variables of the pyramidal neurons for a model incorporating only LOT-pyramidal and pyramidal-$GABA_A$ connections. The pyramidal-pyramidal and pyramidal-$GABA_B$ synapses have been removed. Note that the inhibitory loop pyramidal->inhibitory->pyramidal is closed. Thus, pyramidal excitation triggers activity in the $GABA_A$ layer and eventually results in the inhibition of pyramidal neurons.

As in previous sections, the LOT pool has been collapsed into a single unit which excites a single neuron in the pyramidal layer. This simplication allows easier visualization of the $GABA_A$-pyramidal interactions. Further, the number of connections from each pyramidal cell to $GABA_A$ neurons has been reduced to one. This ensures that the inhibition seen in the pyramidal layer from the $GABA_A$ layer is due to the activation of a single $GABA_A$ cell.

At $t = 0\ ms$, the pyramidal cell layer remains in its initial inactive state. At $t = 4\ ms$, excitatory activity arrives to a single pyramidal unit, triggering an action potential. The pyramidal to $GABA_A$ connections trigger action potentials in the $GABA_A$ layer (not shown in the figure). In turn, at $t = 12\ ms$, the feedback loop consisting of inhibitory connections from $GABA_A$ cells to pyramidal cells, generate IPSPs (inhibitory postsynaptic potentials) in the pyramidal layer. This effect can be seen in the fourth panel of Figure 8.8-B as a local shift towards blue indicating that the state variable w_{sum} has decreased below 0. The last panel in 8.8-B shows a magnified image of the core of the inhibited area in the pyramidal layer.

Given that the duration of the activation of $GABA_A$ synapses was set to $t_{dur} = 5\ ms$, at $t = 17\ ms$ the inhibition ends, having started at $t = 12\ ms$.

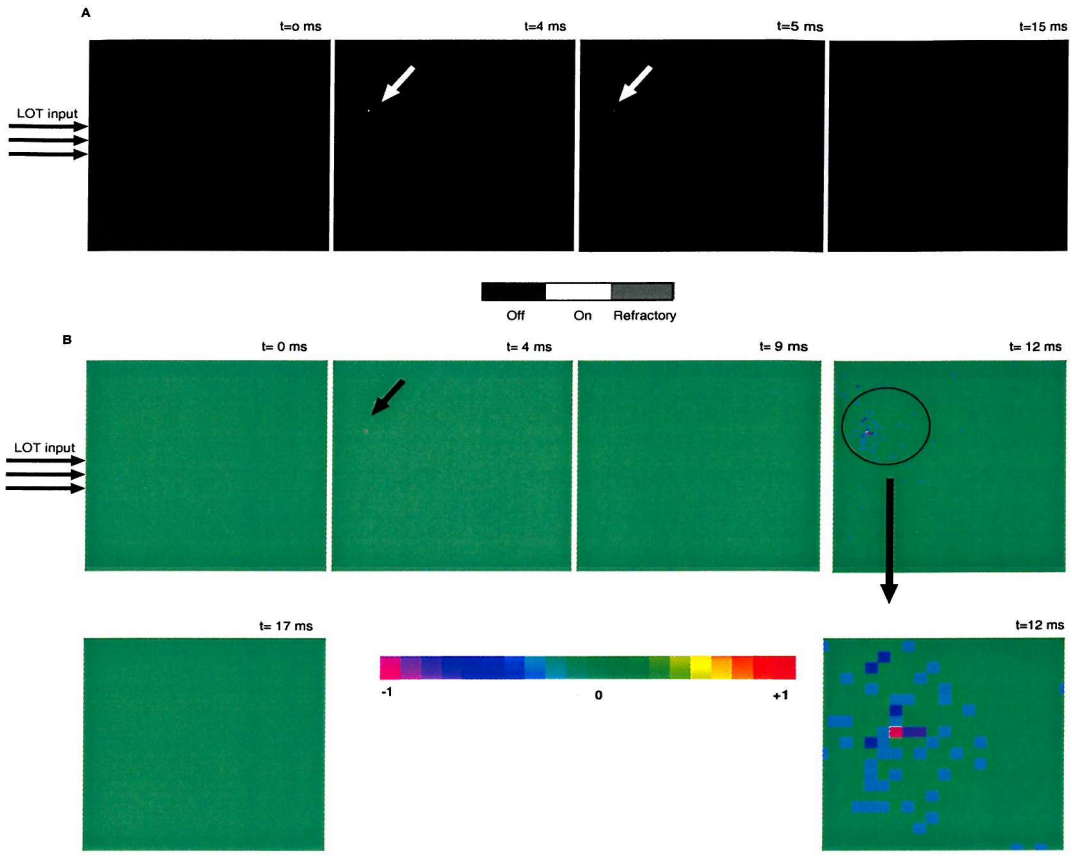


Figure 8.8: States (A) and w_{sum} (B) of pyramidal neurons showing the effect of the $GABA_A$ inhibitory loop

The interaction between pyramidal cells and $GABA_B$ neurons follows the same pattern. However, $t_{dur} = 150\ ms$ for $GABA_B$ inhibitory synapses, which results in a long lasting inhibition, as opposite to the short (5 ms) inhibition generated by $GABA_A$ synapses.

8.5 Shock stimulus response

Having tested the functionality of partially connected networks, this section addresses the issue of adequacy of the MBED framework to replicate biological aggregate dynamics. To this end, the responses of the network to shock stimuli are compared to previous theoretical and experimental data.

In a shock stimulus experiment, the LOT is stimulated with a short duration

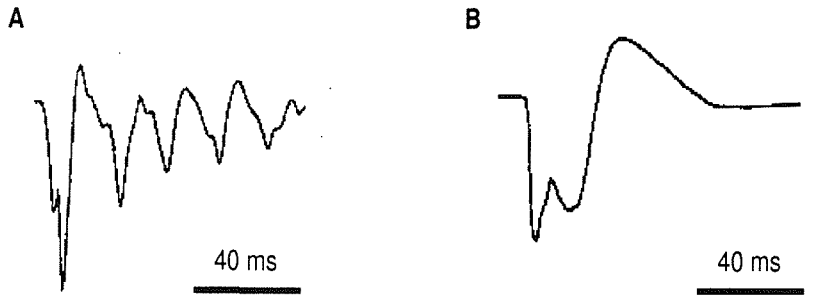


Figure 8.9: Piriform cortex response to weak shock stimulus (A) and strong shock stimulus (B) as obtained with a compartmental model (from Wilson *et al.* [28])

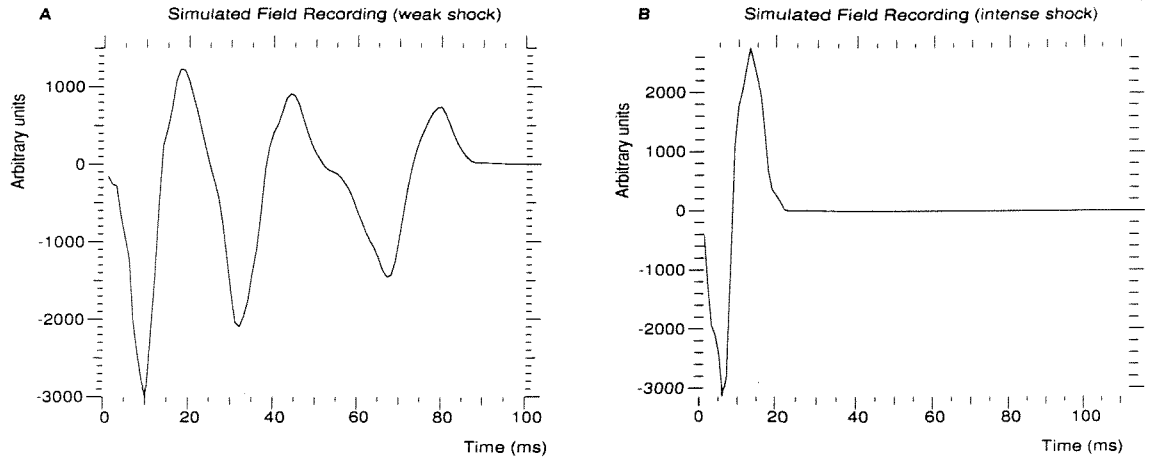


Figure 8.10: (A) Simulated field potential after weak shock stimulus, (B) Simulated field potential after strong shock stimulus

($< 1\ ms$) current pulse while monitoring the field potentials elicited in the olfactory cortex. Two types of recordings have been obtained with such experimental setups; single wave responses and damped oscillations [124, 117, 157]. High intensity current pulses generate single peak extracellular recordings whereas lower intensity pulses produce long lasting responses consisting of several damped peaks [115]. Figures 8.9-A and 8.9-B show the simulated field potentials obtained by Wilson *et al.* [28] with a compartmental model of the piriform cortex stimulated with weak and strong shock stimuli.

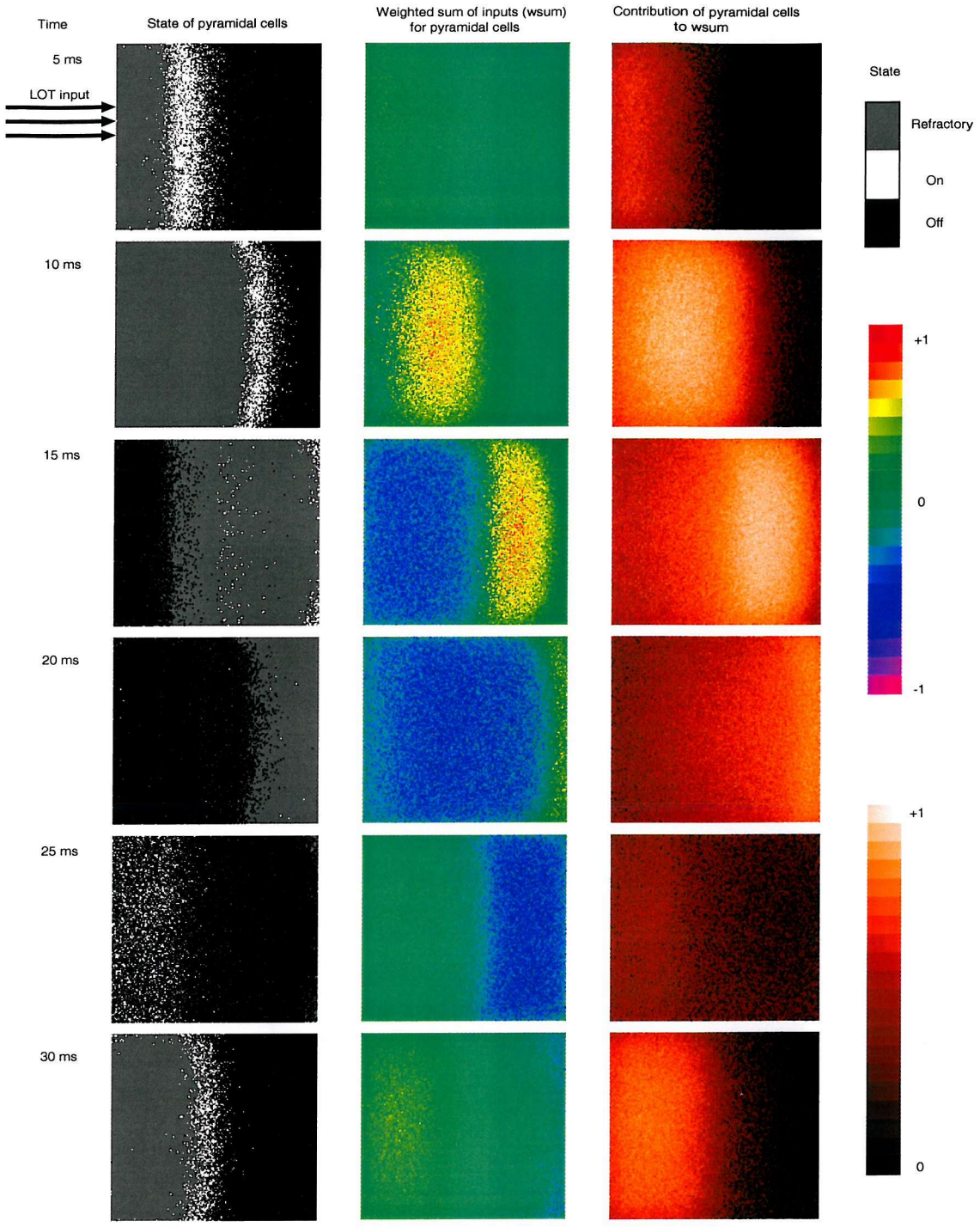


Figure 8.11: States, w_{sum} and pyramidal-pyramidal excitation of pyramidal neurons after weak shock stimulus

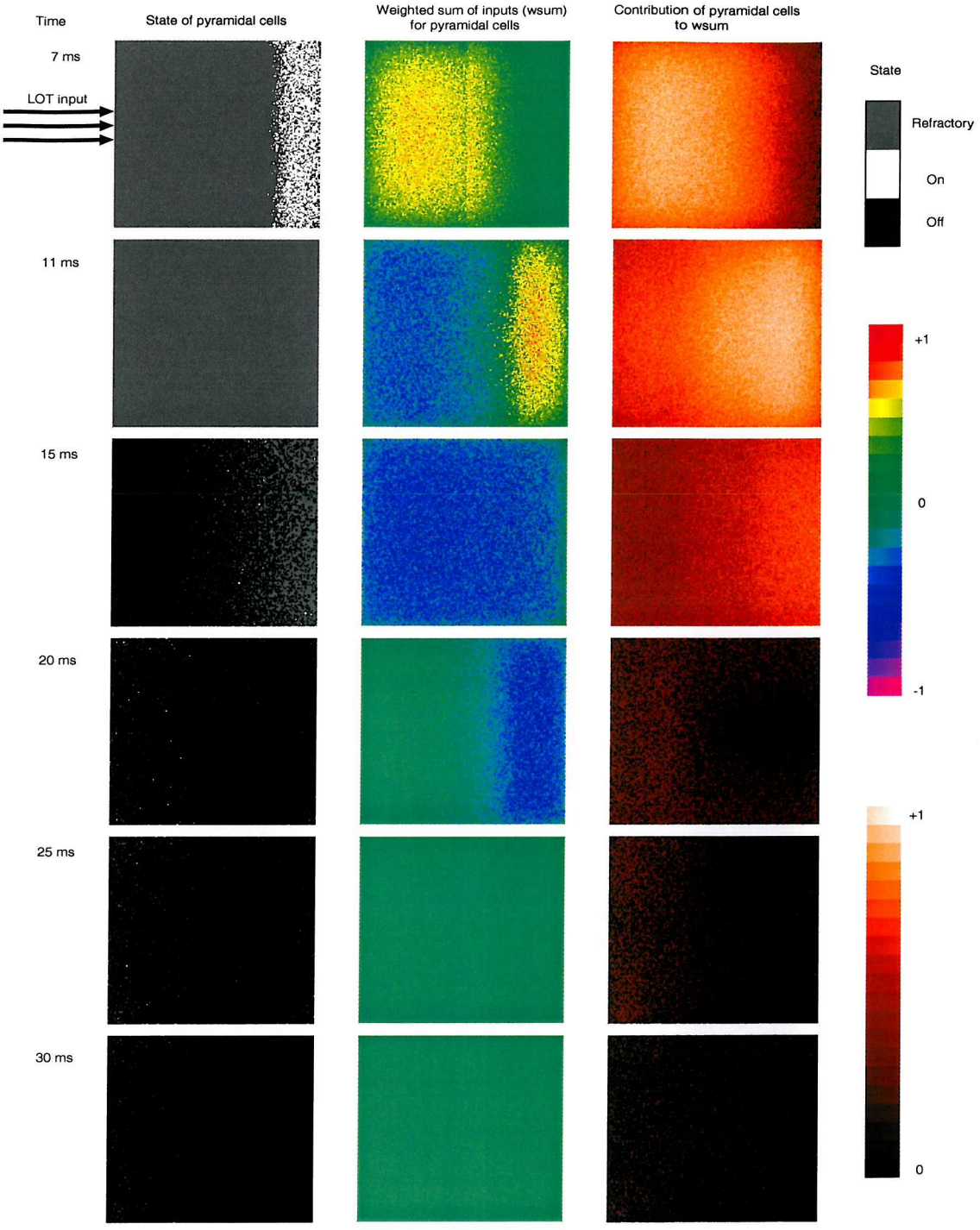


Figure 8.12: States, w_{sum} and pyramidal-pyramidal excitation of pyramidal neurons after strong shock stimulus

Shock stimuli were simulated in the MBED network model by the simultaneous firing of all LOT cells at $t = 0$. For this purpose, the LOT neurons were configured as pace makers with the parameters of the oscillator block set to $t_\phi = 0$, in order to generate the first spike exactly at $t = 0$ and $t_{osc} > t_{max}$, to ensure that the second action potential in the sequence was triggered beyond the finishing time of the simulation, t_{max}.

The intensity of the shock was adjusted changing the total number of cells in the LOT pool and the weights of LOT-pyramidal synapses. Figure 8.10 shows the field potential obtained with a centered virtual electrode and calculated as in expression 8.4. In accordance with experimental observations, a weak stimulus generated by 600 LOT cells (shown in figure 8.10-A) generates a damped oscillation whereas an stimulus created by 2500 LOT cells (shown in figure 8.10-B) produces a single peak in the simulated field recording.

Understanding of the mechanisms underlying these two responses can be gained studying the panels shown in figures 8.11 and 8.12 (only pyramidal cells are shown) and the plots of figures 8.13 and 8.14. The coloured images in 8.11 and 8.12 correspond to the state (leftmost column), normalized value of w_{sum} (middle column) and normalized partial contribution to w_{sum} by other pyramidal cells (rightmost column) at selected points in time. Each neuron is represented by a dot in the matrix using the corresponding colour palette.

Figures 8.13-A,B and C show the total excitatory, $GABA_A$ and $GABA_B$ synaptic input received by pyramidal cells during the course of the first wave after weak shock stimulus. These are presented as the average over the pyramidal cells located in the same column in the panels of figures 8.11 and 8.12 (column numbers increasing from left to right). Figure 8.13-D plots the temporal evolution of the three synaptic input types calculated as an average across the leftmost third (column 1 to 50) of the pyramidal cell layer, where the cortical waves originate. Figures 8.14-A,B,C and D plot analogous data for the strong shock stimulus experiment.

Figure 8.11 shows how the weak stimulus triggers a wave of activity, its front reaching the distant region of the cortex after 15 ms (leftmost column). In the middle panels, plotting the spatial distribution of w_{sum}, it is seen that the wave of neuronal spiking causes a wave of excitatory synaptic activations, its front reaching the far cortical end at 20 ms. In the wake of the excitatory wave, appears a region of inhibited cells (identified by a shift towards blue) due to the activation of inhibitory cells in the $GABA_A$ and $GABA_B$ layers. However, at $t = 25$ ms, the area where the first wave was originated (leftmost region in all panels) is returning to the

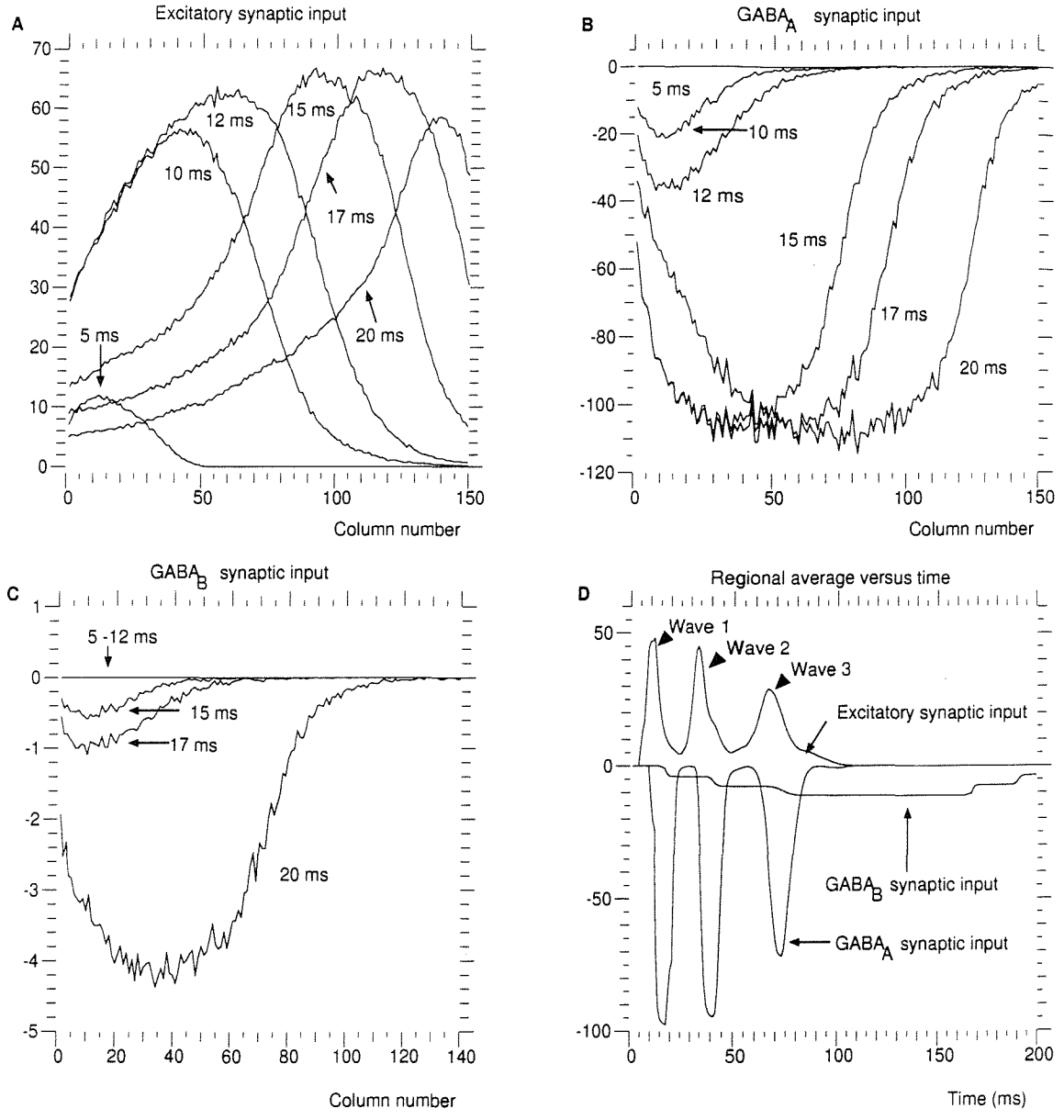


Figure 8.13: Weak stimulus induced synaptic input to pyramidal cells versus column number (A,B,C) and time (D)

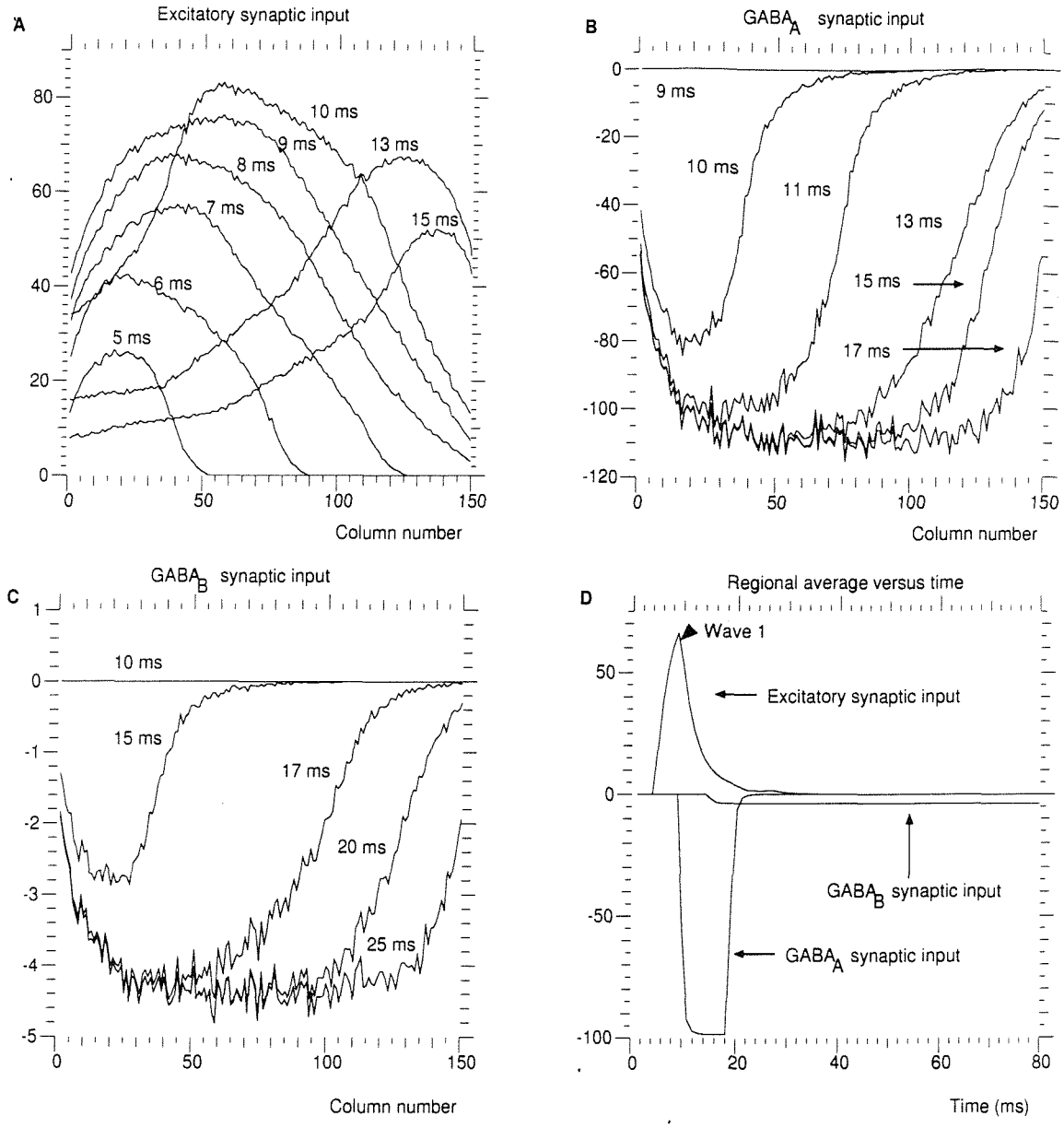


Figure 8.14: Strong stimulus induced synaptic input to pyramidal cells versus column number (A,B,C) and time (D)

initial state (indicated by the shift from blue to green). The disappearance of inhibition makes it possible to generate a second wave.

This wave is triggered by excitation carried by long axons from the front of the first wave back to the leftmost region in the panels. At $t = 20\ ms$, the image in the first column shows how the initial wave has already disappeared and only the wake of neurons in $refractory$ state are left. However, also at $t = 20\ ms$, the third column of images shows a marked excitation reaching pyramidal cells. The intensity decreases from right to left, indicating that it was generated by the first wave while approaching the rightmost end of the cortex. This remaining excitation originates the second wave seen at $t = 25\ ms$ and $t = 30\ ms$.

Figure 8.12 shows the results for the high intensity shock stimulus. In comparison with figure 8.11, the initial wave of excitation propagates faster, arriving at the far end of the cortex at $t = 7\ ms$ (see leftmost panel) in contrast with the $15\ ms$ needed by the weak shock. $15\ ms$ after the strong stimulus, the entire pyramidal layer remains still inhibited (middle panel) and long range excitatory connections between pyramidal cells (see rightmost column) are unable to generate a second wave. Simultaneously with the decrease of inhibition at $t = 20\ ms$ and $t = 25\ ms$, the pyramidal to pyramidal excitation has also decreased (compare rightmost panels at $t = 15\ ms$ and $t = 20\ ms$). The remaining excitation is only able to trigger sparse action potentials (see leftmost regions of the panels in the leftmost column at $t = 20\ ms$) and insufficient to promote the genesis of a new wave.

These latency differences are also manifested in figures 8.13-A and 8.14-A. The wave of excitatory synaptic activation reaches the rightmost end of the cortex (neuronal column number 150) five milliseconds later after a weak stimulus (8.13-A) than after strong shock (8.14-A) . The unequal efficacy of the overall inhibition to decrease the excitability of the cortex can be further understood studying figures 8.13-B,C and 8.14-B,C. The region affected by the $GABA_A$ and $GABA_B$ inhibition triggered by weak stimulus covers a larger range of columns than that obtained with a strong shock. This can be explained by the differences in timing of the excitatory wave and the finite duration of inhibitory synaptic activation. The slower wave after weak stimulus reaches the far end of the cortex before the inhibition resulting from its progression disappears. Conversely, the strong shock travels across the entire cortex faster, reaching the rightmost end before the inhibition in its wake vanishes.

Figure 8.15 plots the temporal evolution of the spatial average of w_{sum} after weak and strong stimuli. Only the subset of neurons confined to the region $1 < column < 50$ was considered, since this region originates the rebound activity.

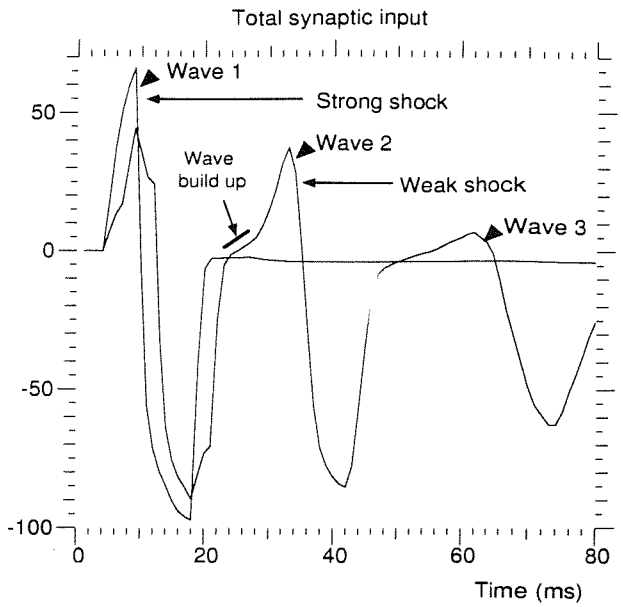


Figure 8.15: Average w_{sum} per neuron across leftmost region of the pyramidal layer (rows 1 to 150, col. 1 to 50)

Both traces have maxima at $t = 9\ ms$ but their time courses exhibit a time lag of 4 ms. The weak stimulus succeeds, with posteriority to the end of the negative (inhibited) phase of w_{sum}, in generating a build up interval which, eventually, triggers a second wave of excitation. Subsequent waves show a decrease in their peak value, consistent with a progressive build up of the cortex-wide $GABA_B$ inhibition (see figure 8.13-D).

A similar sequence of events was observed by Wilson *et al.* [28] in their compartmental cortical model.

8.6 Random input response

Further validation of the MBED model is sought in this section studying the network response to continuous rather than shock stimulation. In experimentally obtained EEGs from the olfactory cortex, theta-type (3-10 Hz) and gamma-type (40 Hz) components have been identified [158]. Previous theoretical work has shown that continuous stimulation triggers EEG oscillations at similar frequencies in compartmental models [28]. The MBED model is used to explore the mechanisms underlying the generation of these patterns of activity.

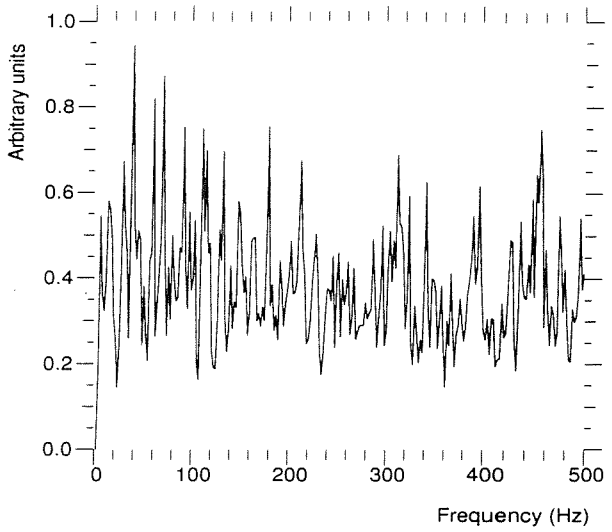


Figure 8.16: Power spectrum of a typical random input stimulus

8.6.1 LOT stimulus

The dynamics of the model was studied using random input stimuli. These were generated by spreading the firing times of the LOT neurons throughout the entire simulation. LOT neurons functioned as pace makers and were configured to ensure that the second spike would occur beyond the simulation stop time, so that only one action potential was generated ($t_{osc} > t_{stop}$). Their firing times, t_{ϕ}, were given by an uniform distribution in the range $(0 - t_{stop})$. Hence, the intrinsic firing frequencies of the LOT neurons can be ruled out as the cause of emergent temporal or spatial patterns that are be observed in the cortex.

The average number of activations of excitatory synaptic connections from LOT cells to pyramidal neurons per unit of time is given by,

$$R = \frac{N_{LOT} C_{LOT-to-pyr}}{t_{stop}} \tag{8.7}$$

where N_{LOT} is the number of LOT cells and $C_{LOT-to-pyr}$ the number of connections to pyramidal cells from a single LOT cell. The stimulus obtained in this way, provides a temporally unstructured and spectrally broad input signal for the cortical model. Figure 8.16 shows the power spectrum of such an unpatterned signal for a stimulus intensity of 97.65 epsps/ms.

No changes, other than the modification of the stimulus type, were made to the model with respect to the parameters used for the study of the shock stimulus

response. All EEGs in this section were obtained using a grid of 10x10 electrodes.

8.6.2 Results with nominal parameter values

In order to explore the activity patterns generated by LOT stimuli of various intensities, a range of input firing rates was tested (see figures 8.17 and 8.18). The plots at the top row of figure 8.17 show the EEGs obtained for 500 epsp/ms and 1000 epsp/ms. Their amplitudes, three to four orders of magnitude lower than subsequent EEGs, indicate that excitation was confined to isolated neurons and that activation was not generalized. As expected from an unpatterned input, no structured signal was seen in the EEG. However, an increase of the input intensity to 1500 epsp/ms and 3500 epsp/ms (second row of figure 8.17) originates bursts in the EEG. The number of waves per burst increases with the stimulus intensity whereas the interburst delay decreases from 300 ms, corresponding to 1500 epsp/ms, down to the 150 ms obtained with 20000 epsp/ms.

Figure 8.18 shows both temporal and spectral EEG patterns. At 16000 epsp/ms (top row) the pattern in the EEG presents characteristics similar to those obtained with lower intensity. Its power spectrum presents a double peak at about 40 Hz (gamma oscillations) and a low frequency peak (theta oscillations). Subsequent increases of the stimulus to 17000 epsp/ms and 21000 epsp/ms, bring about the loss of burst patterned oscillations while retaining a high frequency oscillation. Concomitantly with the disappearance of the burst-like pattern, the associated power spectra shows an emerging peak at 75 Hz.

More detailed analysis of these results indicate that the high frequency component seen during an EEG burst is caused by the propagation of cortical waves similar to those observed after shock stimuli. EEG peaks are associated with individual waves propagating across the cortex. Figure 8.19-A shows a fragment of an EEG during a burst and figure 8.20 shows the activity in the pyramidal layer occurring simultaneously. Note that waves originate on the lefthand side in the pyramidal layer (closest to the LOT input) and propagate to the distant end. The same pattern had been observed as a result of single shock stimuli.

An example of the period of diminished activity between EEG bursts is shown in figure 8.19-B and its corresponding sequence of activity in the pyramidal cell layer depicted in figure 8.21. Large regions of the the pyramidal layer are inhibited (seen as a shift towards blue in the panels) during the interburst period (see t = 160 ms). This inhibition reduces the probability of neuronal firing resulting in a absence of

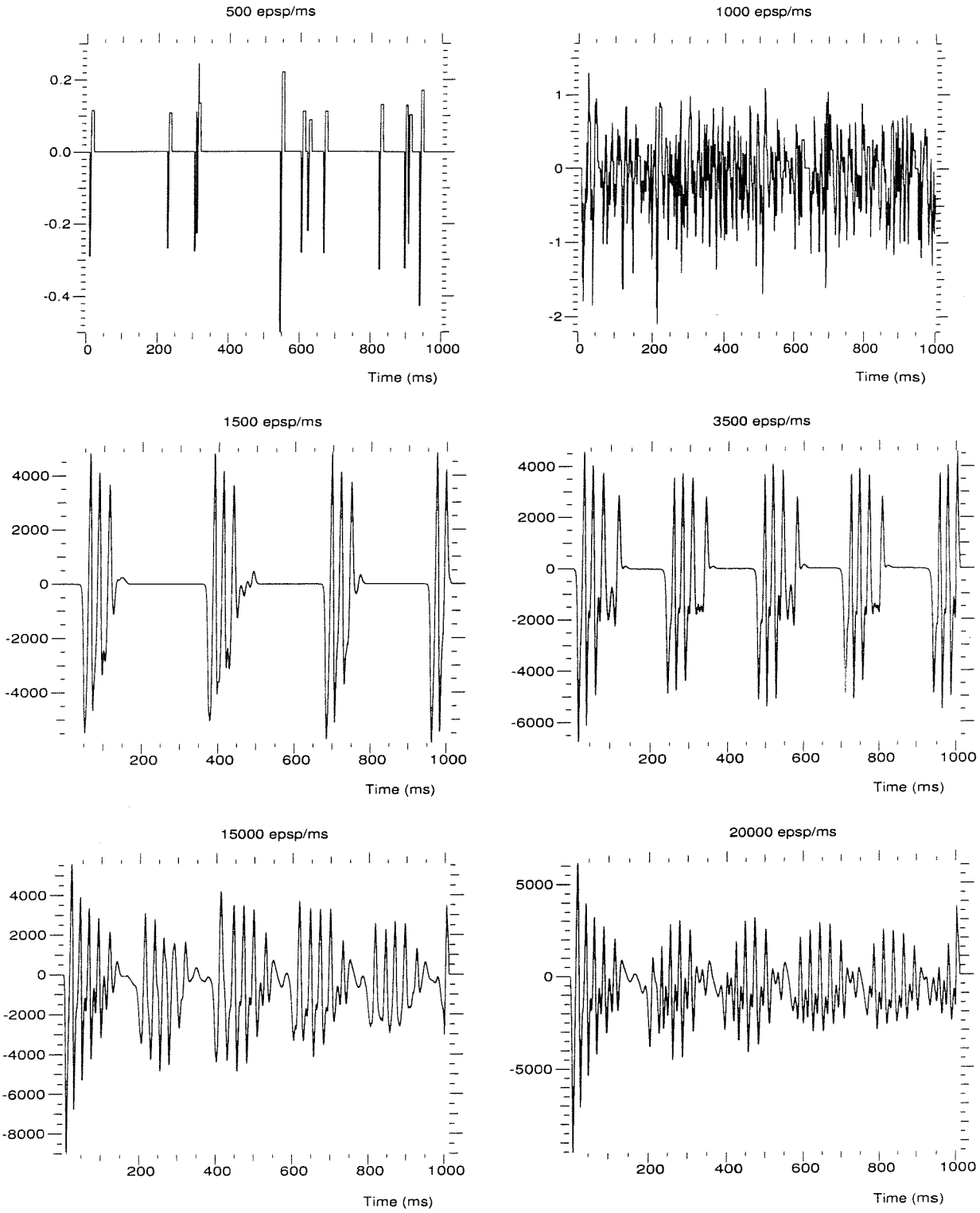


Figure 8.17: EEGs obtained with random input(I)

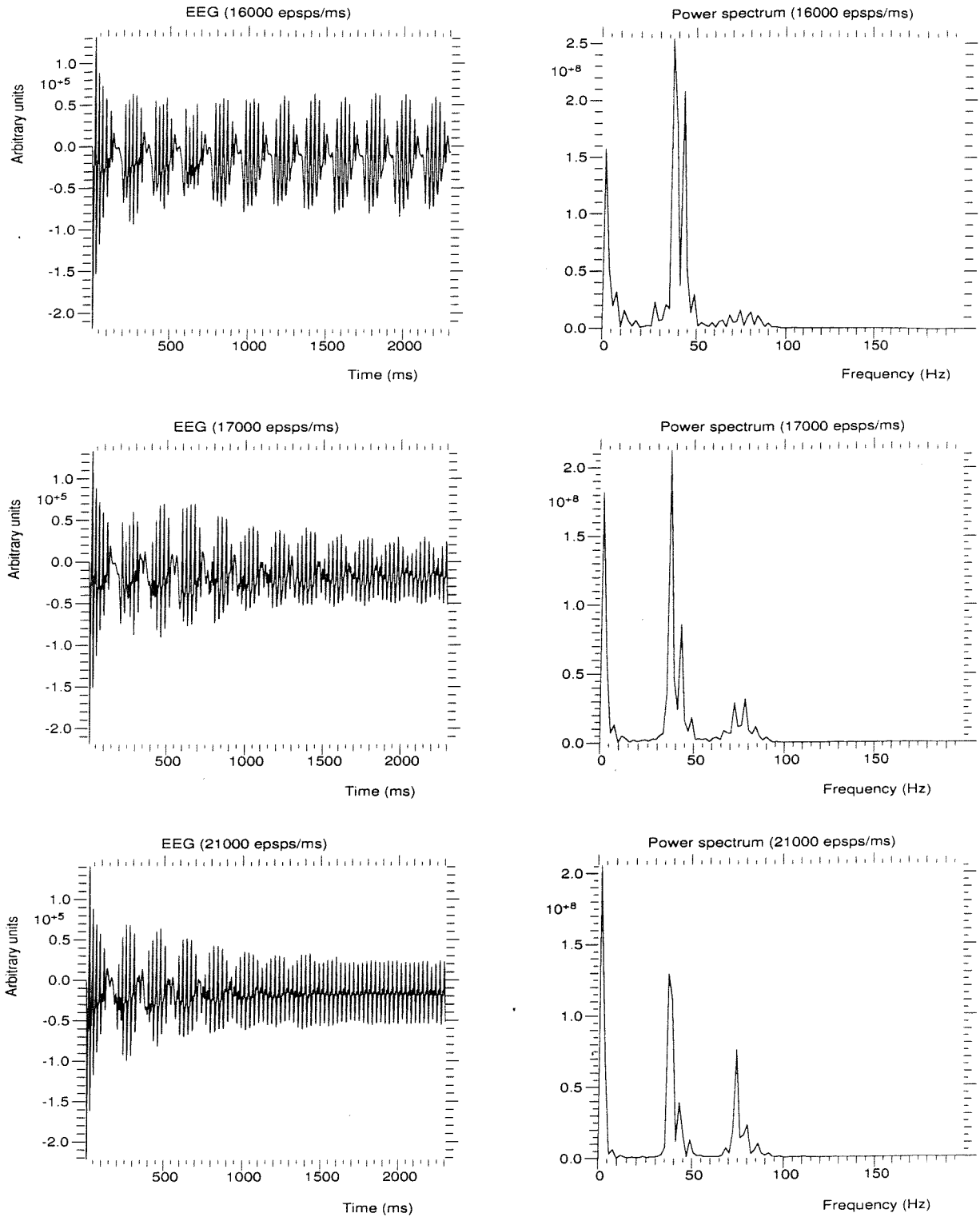


Figure 8.18: EEGs obtained with random input(II)

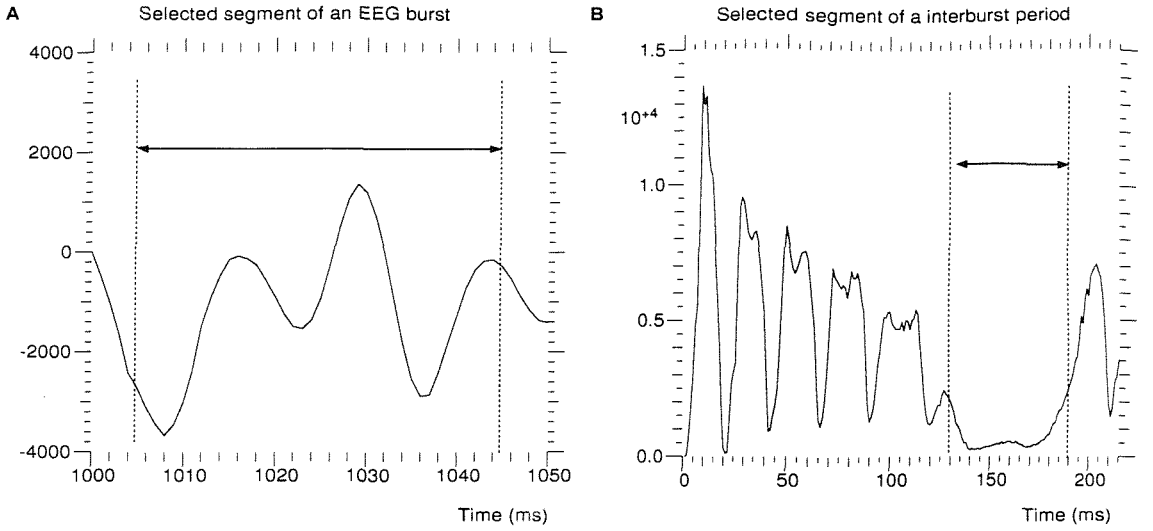


Figure 8.19: EEG burst (A) and interburst period (B)

cortical waves. As the inhibition vanishes, a new wave is generated.

The bursting component is controlled by the slow $GABA_B$ inhibitory cells. This is hinted at by the fact that the duration of the synaptic activation in the $GABA_B$ synapses of the model lasts 150 ms, which is the only parameter in the model of the same order of magnitude as the period between bursts in the EEG. In previous experiments with shock stimuli, the appearance of consecutive waves led to a progressive build up of the overall $GABA_B$ inhibition (figure 8.13-D). Further, this assumption was confirmed by the fact that a decrease in the duration of $GABA_B$ synapse activation to $t_{dur} = 50\ ms$ produces an EEG consisting of a high frequency oscillation without bursting components.

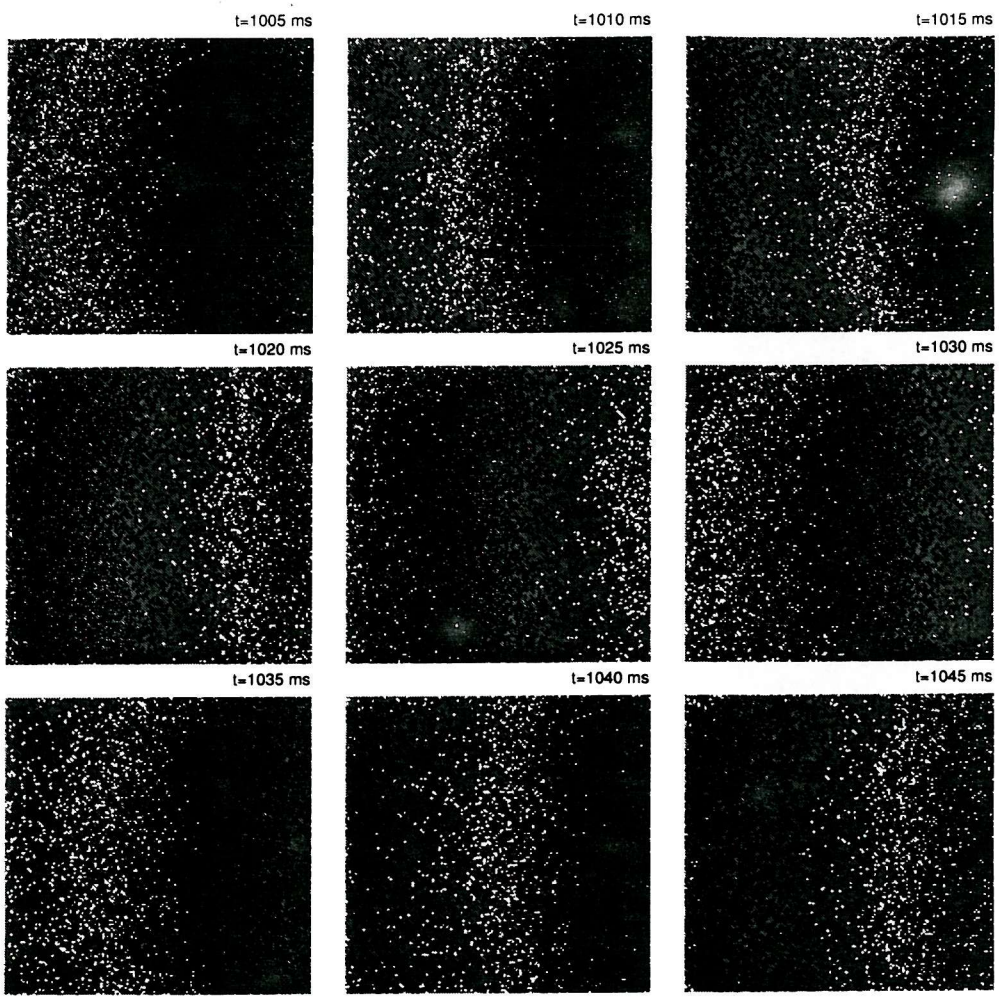


Figure 8.20: Activity in the pyramidal layer during an EEG burst

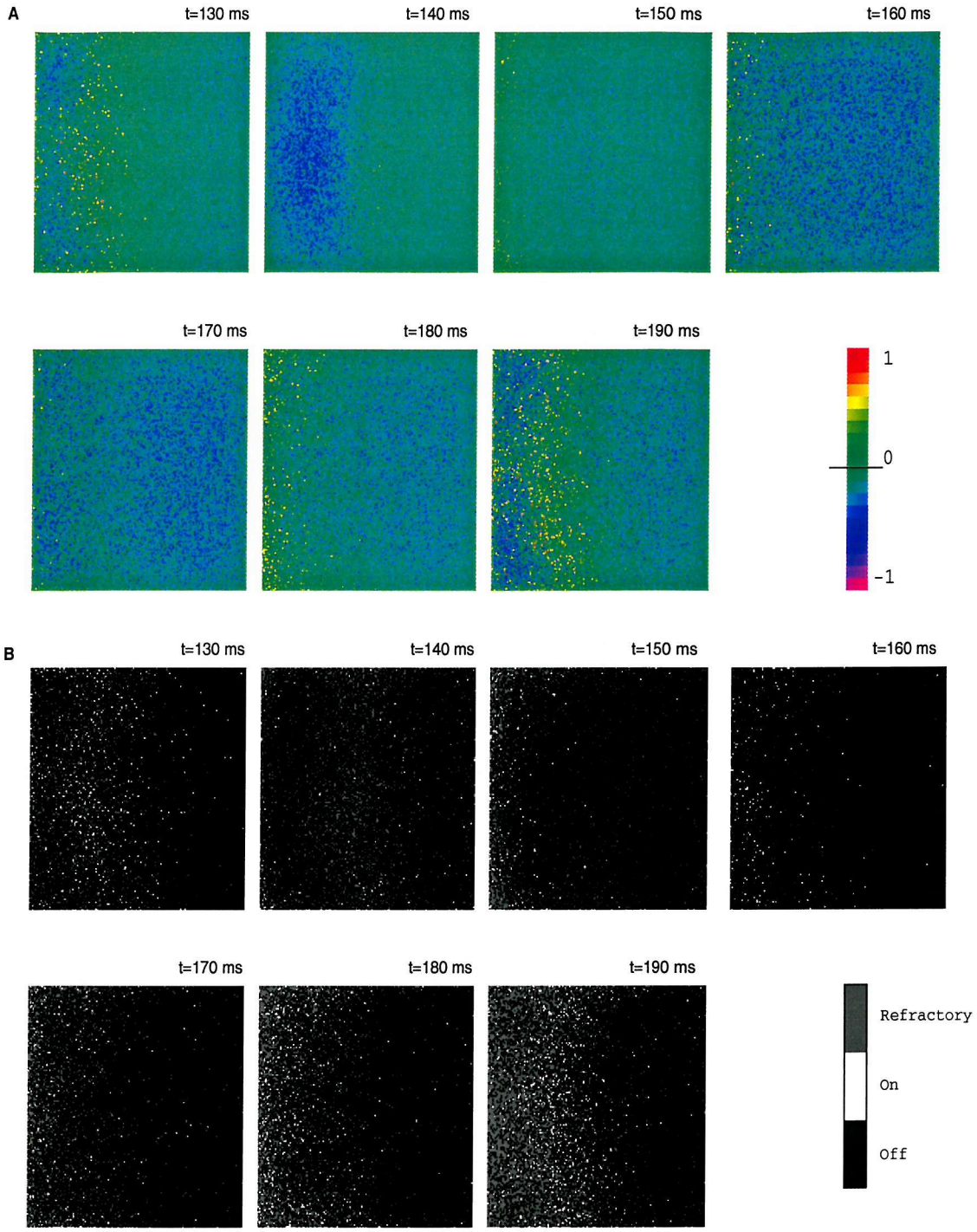


Figure 8.21: Spatial profile of w_{sum} (A) and neuronal states (B) in the pyramidal cell layer during an inter-burst interval

8.6.3 Impact of synaptic parameter variation on the temporal and spectral contents of the EEGs

Previous work with compartmental models has predicted a decrease in the frequency of the main spectral peak in the EEG power spectrum as a result of an increase of the activation times, t_{dur}, of the fast inhibitory ($GABA_A$) synapses [28]. A similar parameter variation was tested with the MBED cortical model in order to establish the consistency of its predictions.

Figure 8.22 shows the EEGs and power spectra for several values of t_{dur} in $GABA_A$ synapses. The high frequency peak decreases from 80 Hz for $t_{dur} = 1\ ms$ to 40 Hz for $t_{dur} = 9\ ms$ (nominal value) and, further, to 30 Hz for $t_{dur} = 13\ ms$.

In addition to this shift in the main frequency component, the EEG indicates marked changes in the dynamics of the model. For $t_{dur} <= 9\ ms$, the EEGs show bursts of activity whereas for $t_{dur} > 9\ ms$, the bursts disappear leaving an unmodulated high frequency oscillation.

Further experiments were carried out to study the effect of t_{dur} in $GABA_B$ synapses on the temporal and spectral characteristics of the EEG; figure 8.23 shows the results obtained. The increase of $GABA_B$ activation times, from 150 ms to 250 ms, results in elongated interburst latencies. This is consistent with the measurements obtained in previous sections which identified the prominent role of $GABA_B$ synapses on oscillation damping after shock stimuli, and burst generation during a random stimulus.

The two main peaks of the power spectra (corresponding to theta and gamma oscillations) remain unchanged. However, a third component, with a frequency between 70 and 80 Hz appears for $t_{dur} > 200\ ms$.

Figure 8.24 shows the results obtained for several values of the pyramidal-pyramidal synaptic strength, w_{syn}. The sequence of EEG bursts obtained for $w_{syn} = 1$, are discontinued after the second burst for $w_{syn} = 2$. They reappear with more irregular temporal and wider spectral components for $w_{syn} = 3$. Further increases of the synaptic strength lead to a damped oscillation in the EEG and to the concentration of the spectral contents in two frequency bands; $1 - 5\ Hz$ and $80 - 90\ Hz$. These results are consistent with the existence of phase transitions as observed in more abstract lattices of Boolean automata [159, 160] and indicate that such network mode switchings are possible in cortical structures as a result of synaptic parameter variation.

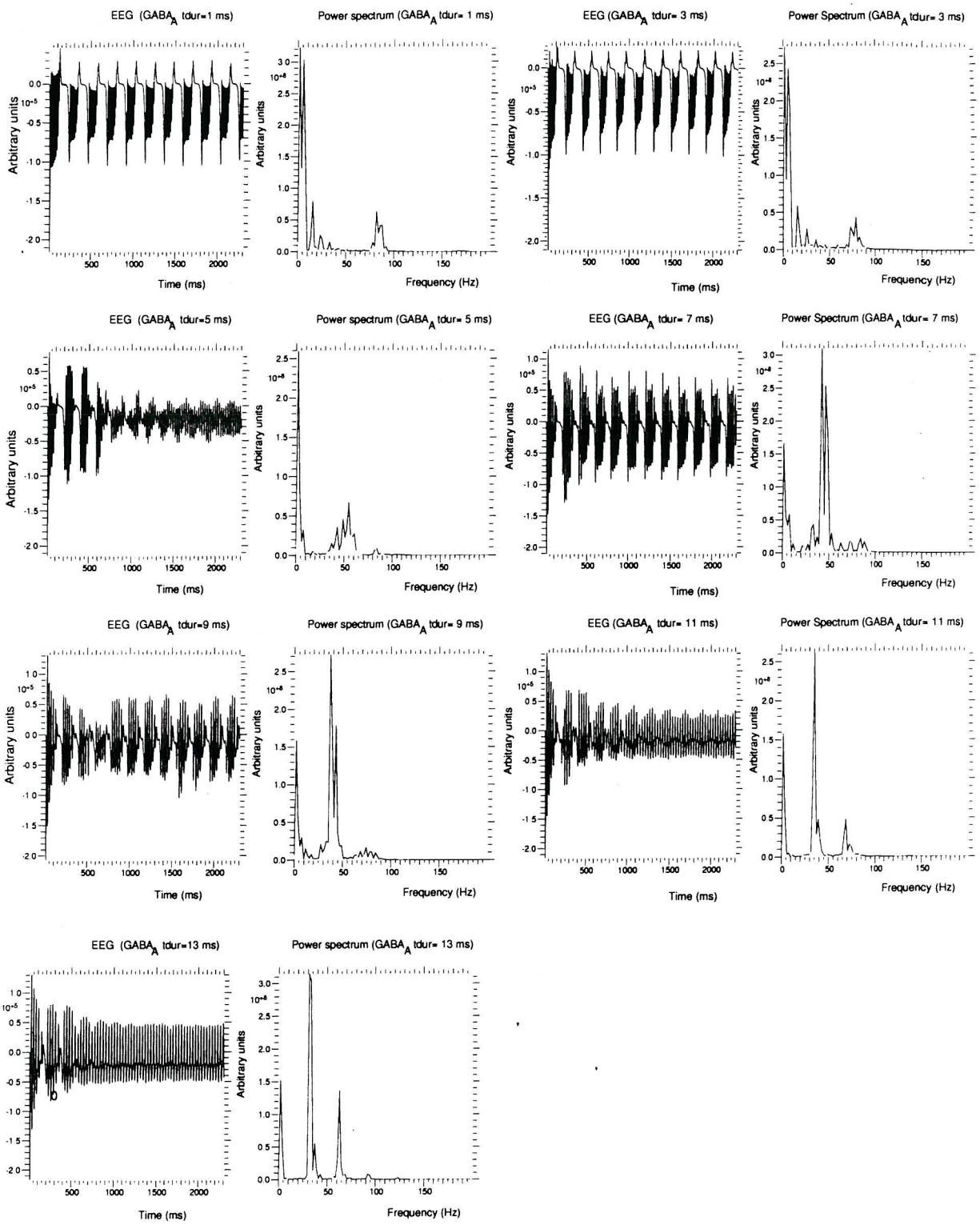


Figure 8.22: EEGs and power spectra obtained for several values of $GABA_A$ activation times (t_{dur})

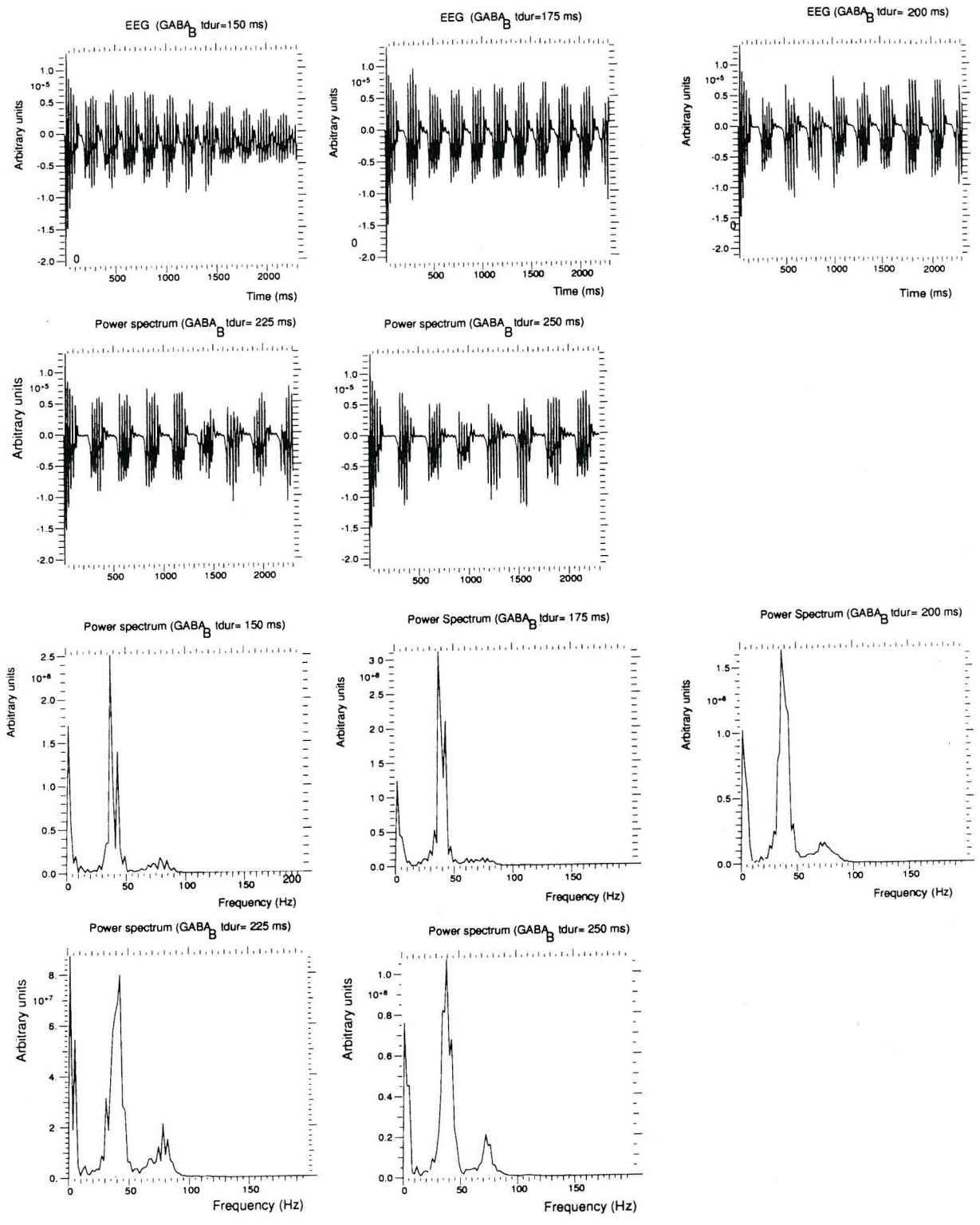


Figure 8.23: EEGs and power spectra obtained for several values of $GABA_B$ activation times

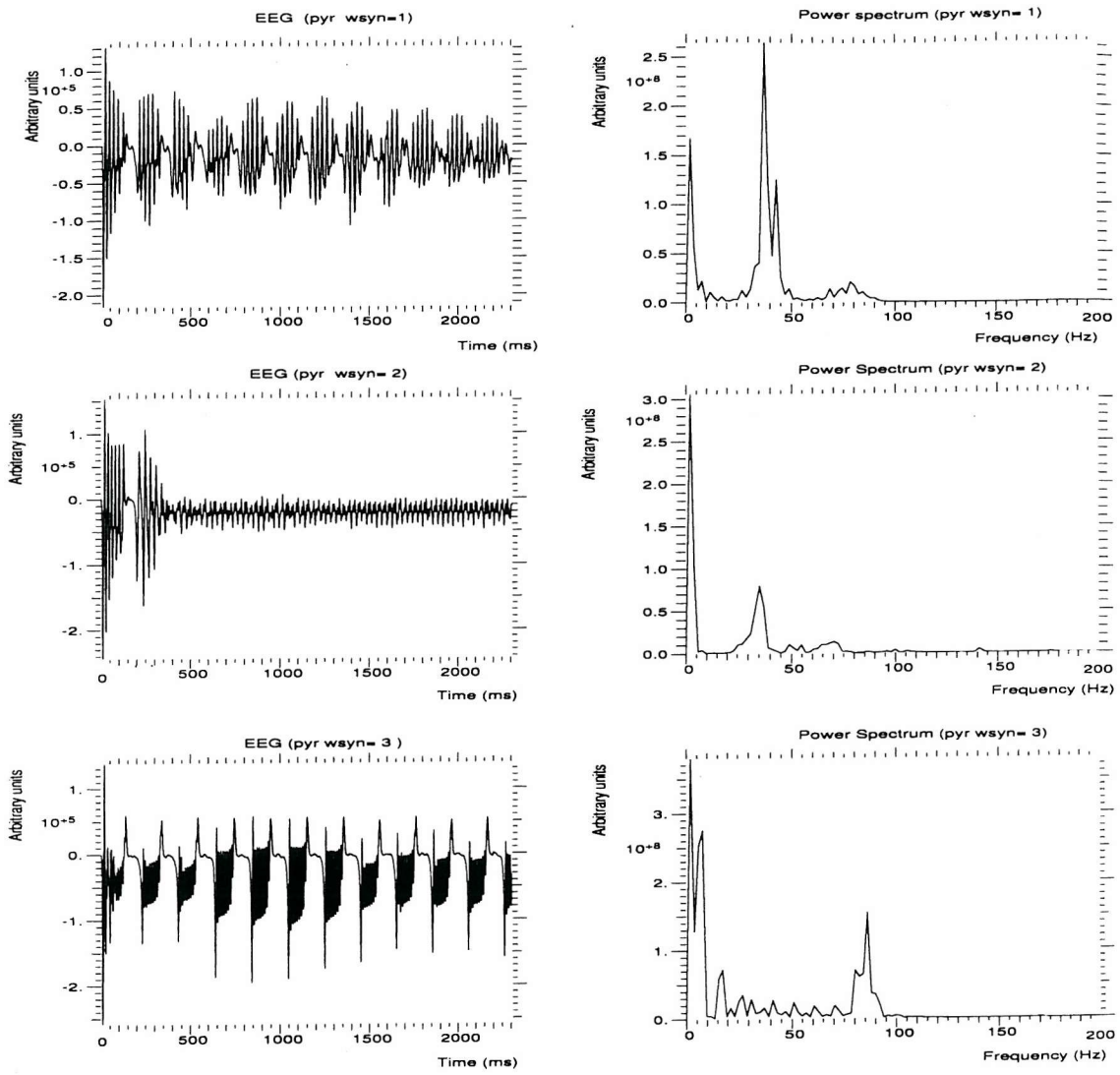


Figure 8.24: Effect of pyramidal-pyramidal synapse strength, w_{syn}, on temporal and spectral EEG characteristics (I)

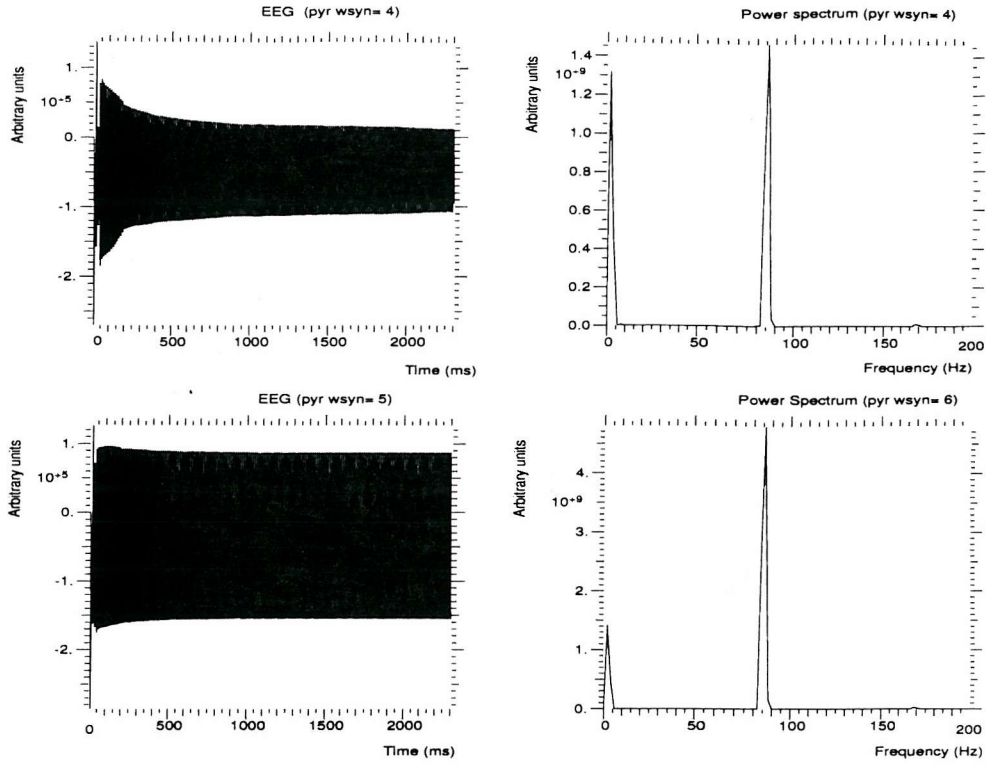


Figure 8.25: Effect of pyramidal-pyramidal synapse strength, w_{syn}, on temporal and spectral EEG characteristics (II)

8.6.4 Spatially uniform LOT stimulus

The previous sections, have used LOT stimuli with an exponentially decreasing spatial profile of intensity towards the righthand side of the panels. Such a spatial pattern of stimulus activity led to waves being generated in the leftmost cortical area where the intensity of the input activity was higher (see for example figure 8.20).

Cortical waves, however, have also been experimentally observed in cortical regions where the input stimuli are more uniformly distributed across the entire area [132].

To study the effect of the spatial profile of the stimulus on wave generation, the exponentially decreasing density of connections from LOT to pyramidal neurons was substituted by an uniform distribution. Hence, the target for a synapse from an LOT neuron was chosen by generating a random vector $\vec{p} = \{x, y\}$ (given in normalized rectangular coordinates) where x and y are uniform random variables within the limits of the model. The target neuron for the connection is chosen as the pyramidal neuron closest to the vector $\vec{p}$. Further, the delay introduced by the axons

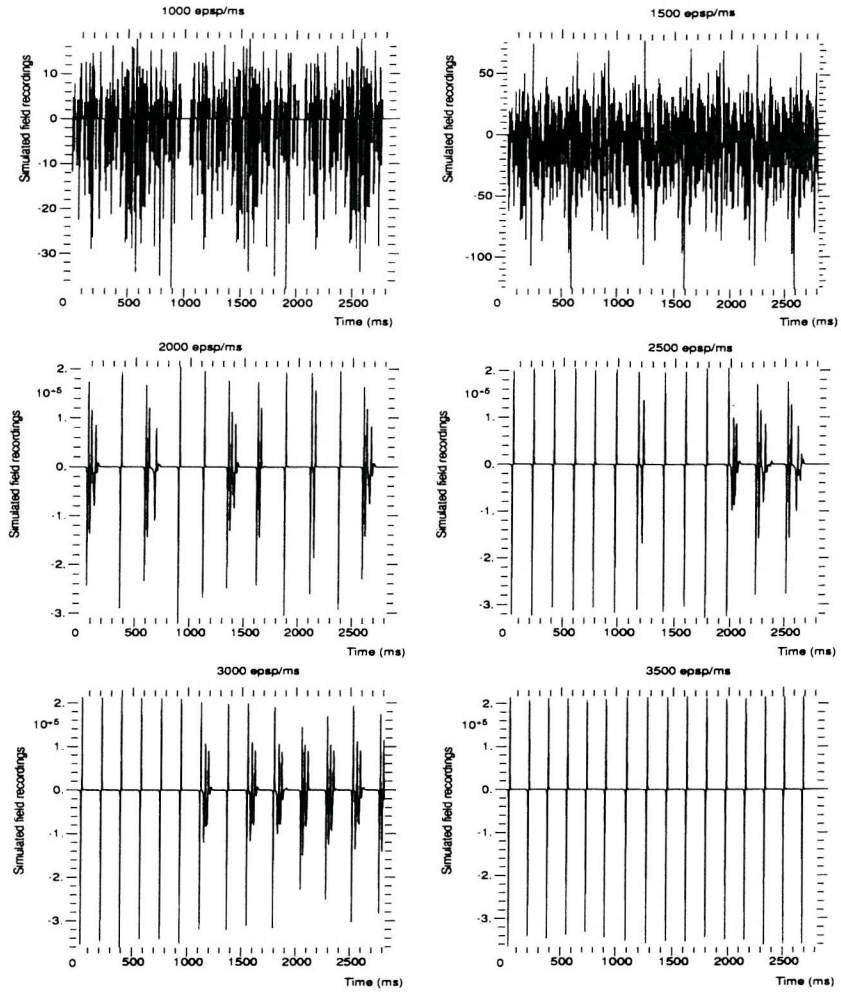


Figure 8.26: EEGs obtained with uniformly distributed and fixed delay LOT to pyramidal connections (I)

from LOT cells to pyramidal neurons (t_{del} of LOT-pyramidal synapses) was fixed to a distance independent value of 1 ms. These modifications to the model ensured that the input stimuli did not favour a specific region in the cortex due to an spatially heterogeneous number of LOT-pyramidal connections (as it was the case with an exponential spatial profile) or by exciting different areas with different latencies (as would occur with a distance dependent axonal delay).

Figures 8.26 and 8.27 show the simulated EEGs obtained for several values of the stimulus intensity, given as the number of activations of excitatory LOT-pyramidal synapses per unit of time.

The magnitude of the EEGs shown in the two upper plots of figure 8.26

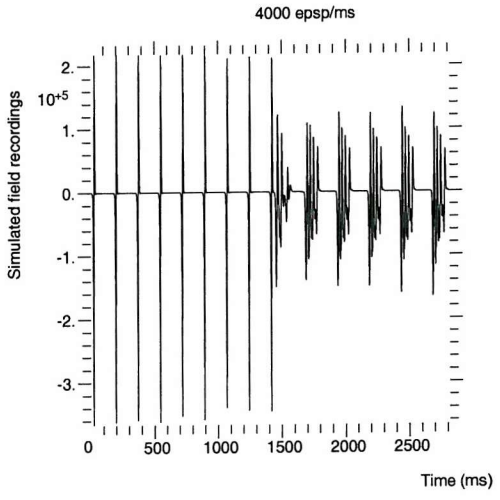


Figure 8.27: EEGs obtained with uniformly distributed and fixed delay LOT to pyramidal connections (II)

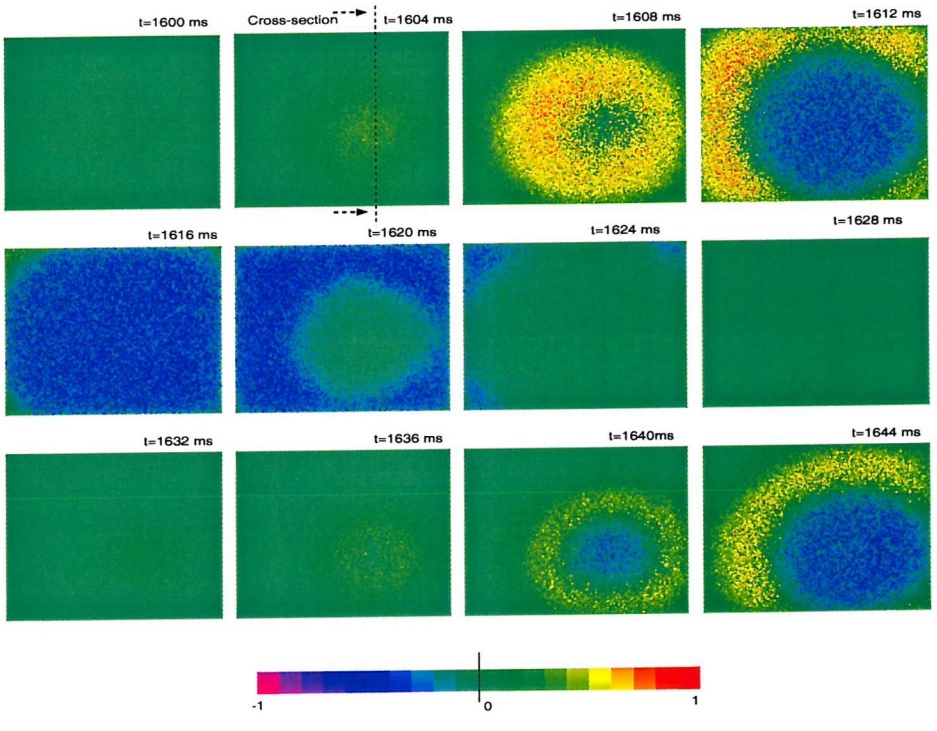


Figure 8.28: Sequence of images showing the time evolution of the normalized w_{sum} for the pyramidal cell layer

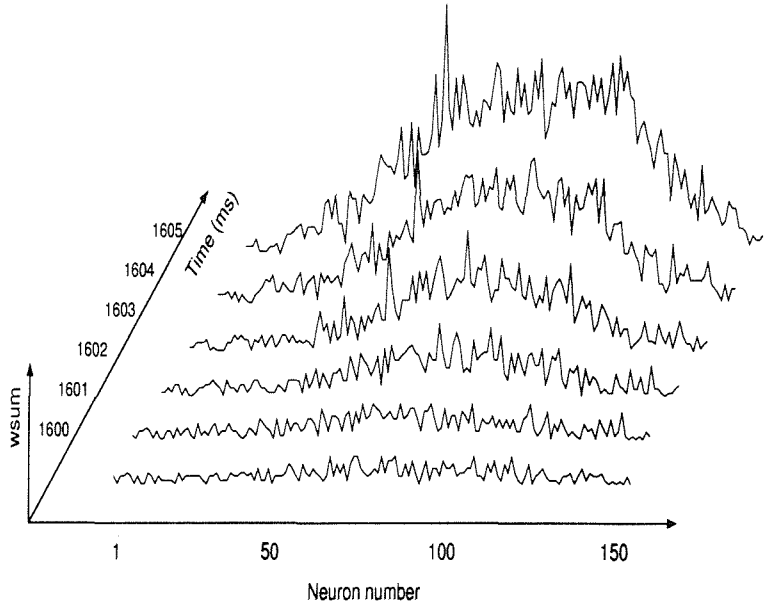


Figure 8.29: Spatio-temporal evolution of w_{sum} in the pyramidal neurons located in the cross-section marked in figure 8.28

(corresponding to stimuli of 1000 epsps/ms and 1500 epsps/ms) is four orders of magnitude lower than that of the EEGs obtained for input intensities higher than 1500 epsps/ms. This fact indicates that activity was limited to sporadic firings and that coherent waves, involving generalized synchronization, were not generated in the first two simulations. For values greater than 1500 epsps/ms, two types of responses were obtained: isolated waves and bursts with multiple waves. Both types of responses are alternated during most of the simulations. However, figure 8.27 shows an spontaneous switch between single wave and burst response at 4000 epsps/ms.

The fact that the EEG peaks indicate spatial waves of cortical activity is confirmed by the images shown in figures 8.28, which display the time evolution of the state variable w_{sum} in the pyramidal cell layer. The sequence corresponds to the two-wave burst generated at approximately $t = 1600\ ms$ during the simulation with an stimulus intensity of 2000 epsps/ms shown in figure 8.26.

Figure 8.28 shows, at $t = 1604\ ms$, an increase of w_{sum} confined to a circular area. Figure 8.29 provides an alternative view of the creation of this focus of activity. It plots w_{sum} at six time points coinciding with the generation of the new wave for the cross-section indicated in figure 8.28 (second panel in the top row). At $t = 1608\ ms$, a wave front had been formed which propagated radially from the center of the

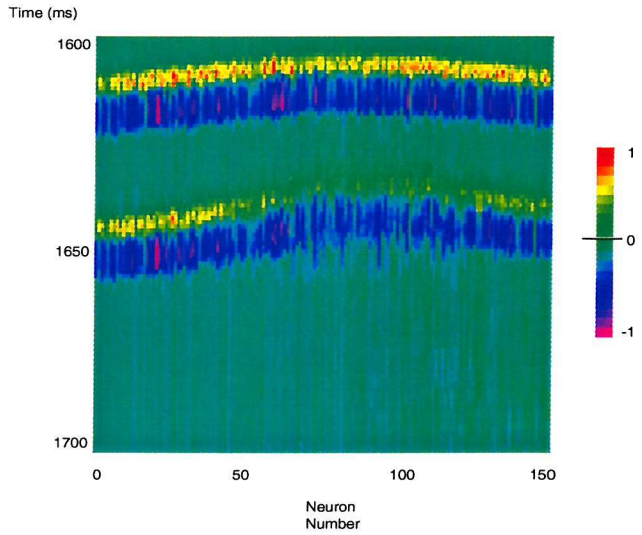


Figure 8.30: Sequence of cross-sections showing the normalized w_{sum} during a two wave burst

region excited at $t = 1604\ ms$. At $t = 1612\ ms$, the wave of excitation reaches the limits of the layer whereas the core area is inhibited. Four milliseconds later, the totality of the piriform layer remained inhibited by the inhibitory interneurons. At $t = 1620\ ms$, the core returns to the initial state and at $t = 1632\ ms$, a second wave starts. Note, however, that the excitation corona of the second wave shows less intensity than that of the first wave (compare for example $t = 1640\ ms$ and $t = 1608\ ms$ in figure 8.28). This attenuation is apparent in figure 8.30, which shows a time extended colour-scale representation of the sequence of cross-sections plotted in figure 8.29. Each row in the matrix corresponds to the value of w_{sum} for the neurons (150) located in the cross-section at a particular point in time. As time progresses (downwards in figure 8.29), excitation propagates from the centre of the section towards its limits. Note that the shift towards red is clearly more marked in the first wave than in the second indicating an attenuation in the last wave.

The main finding of these experiments is that cortical waves with radial propagation can occur in a network with an evenly distributed input stimulus where the temporal and spatial heterogeneities of the LOT stimuli have been removed.

8.7 Heterogeneous neural pools

The network model presented in previous sections was constructed with a 1:1:1 population size ratio for the three cortical cell types; pyramidal, fast and slow inhibitory. The response to shock and random input followed the patterns obtained by [28] with a compartmental network model incorporating equal size neural populations.

Barkai *et al.* [133] have also constructed a compartmental network model, akin to that described in [28], including 298 neurons distributed among the three neuron classes with a biologically realistic proportion of neuron types; 20% inhibitory and 80% pyramidals. Other researchers have also constructed cortical models with these excitation-inhibition proportions [39].

The relative size of the neural populations in the MBED model was modified to adhere to the above percentages. The following sections describe the simulations carried out in order to confirm that a parameter set exists which displays realistic responses to shock and random stimuli in such a model.

8.7.1 Network parameters

Table 8.3 lists the parameters of the model found to replicate the experimental responses to shock and continuous stimuli. The network includes a layer of 250×250 pyramidal neurons and two layers of 80×80 inhibitory cells each, with an average of 285 synapses per neuron. The spatial patterns of connectivity have not been altered with respect to the model used in previous sections. The excitation threshold, th_e, was raised from 5 to 7 in the pyramidal layer. Likewise, the threshold of inhibitory cells was increased to 30, to limit their level of activity. This was necessary as a consequence of an overall increase of the average excitation received per inhibitory cell. Stronger excitation was due to the larger number of excitatory pyramidal cells (from 150×150 to 250×250) and the reduction in the number of inhibitory neurons (from $150 \times 150 \times 2$ to $80 \times 80 \times 2$) with respect to the homogeneously sized model.

8.7.2 Shock and random stimuli

Figure 8.31 shows the single electrode measurement obtained after a weak (A) and a strong (B) stimulus. The weak shock stimulus was generated with a pool of 1000 LOT units firing simultaneously at $t = 0\ ms$ whereas the strong stimulus response was obtained with a pool of 6000 LOT units. Figure 8.32 shows the EEG obtained

Neuronal parameters	
th_e (pyramidal)	7
th_e (fast inh.)	30
th_e (slow inh.)	30
th_i	-1000 (burst truncation inactivated)
t_{ap}	1 ms
t_{ref}	10 ms
N_{burst}	1
t_{osc} (pyramidals and inhibitory)	0 (*inactive oscillator*)
t_ϕ (pyramidals and inhibitory)	0 (*inactive oscillator*)
t_{osc} (LOT cells, all stimuli)	3000 ms
t_ϕ (LOT cells, shock stimulus)	0 ms
t_ϕ (LOT cells, random input)	Uniform$(0 - t_{stop})$
Number of synapses per neuron	
pyramidal to pyramidal	300
pyramidal to fast inhibitory	20
pyramidal to slow inhibitory	10
fast inhibitory to pyramidals	70
slow inhibitory to pyramidals	60
Synaptic parameters	
t_{del} (pyramidal to pyr./inh.)	$(3 - 12\ ms)$
t_{dur} (pyramidal to pyr./inh.)	5 ms
w_{syn} (pyramidal to pyr./inh.)	1
t_{del} (fast inh. to pyramidal)	5 ms
t_{dur} (fast inh. to pyramidal)	12 ms
w_{syn} (fast inh. to pyramidal)	-15
t_{del} (slow inh. to pyramidal)	10 ms
t_{dur} (slow inh. to pyramidal)	150 ms
w_{syn} (slow inh. to pyramidal)	-1
t_{del} (LOT to pyramidal)	$(1 - 4\ ms)$
t_{dur} (LOT to pyramidal)	5 ms
w_{syn} (LOT to pyramidal)	4
Connection range, λ (normalized distance)	
pyramidal to pyramidal	2
pyramidal to fast inhibitory	10
pyramidal to slow inhibitory	10
fast inhibitory to pyramidals	10
slow inhibitory to pyramidals	10
LOT to pyramidals	2

Table 8.3: Numerical values of parameters in the heterogeneously sized piriform cortex model

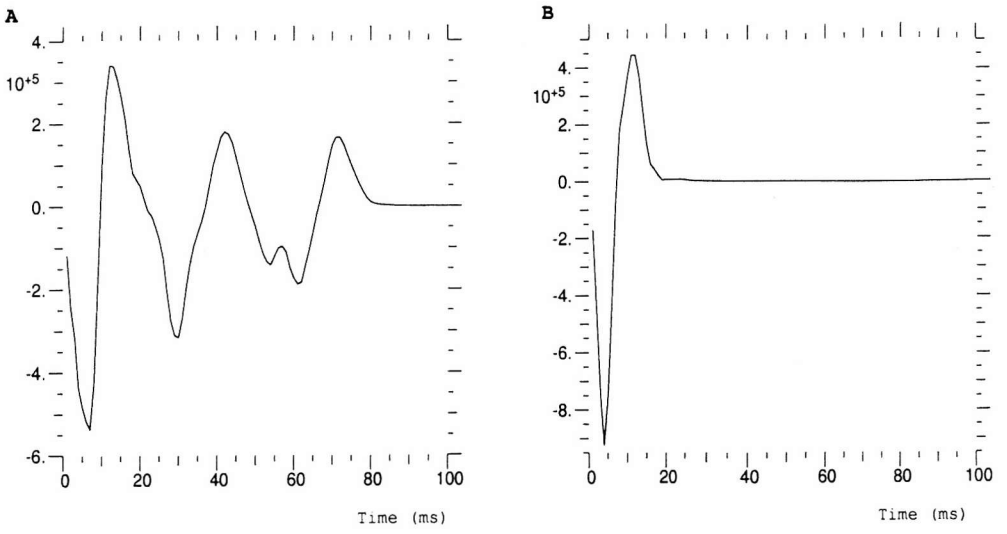


Figure 8.31: Response to weak (A) and strong (B) shock stimuli

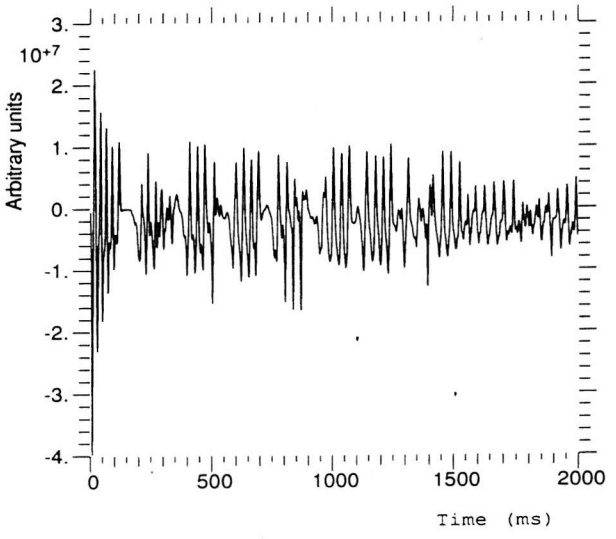


Figure 8.32: EEG obtained with a random stimulus

with a random stimulus consisting of $10^4 epsps/ms$ generated by LOT units onto pyramidal cells and uniformly distributed in time throughout the entire simulation. These responses, single peak, damped oscillation and gamma-type EEG components, are equivalent to those already described for homogeneous neuronal populations and confirm that both, homogeneous and heterogeneous MBED cortical models, are capable of replicating experimentally obtained measurements.

8.8 Conclusions

In this Chapter, the cortical model based on the MBED framework has been shown to replicate results obtained both experimentally and with compartmental models, which substantiates the hypothesis that automata models are suitable for quantitative, in addition to qualitative, biologically motivated simulation. Moreover, the computational efficiency of the MBED model compares favourably with competing modelling approaches. The heterogeneous MBED network model incorporating 75300 cortical neurons and 10^5 LOT units with an average of 285 synapses per neuron, took 10-20 minutes to simulate 2000 ms of the random stimulus experiments (the actual CPU time varied amongst simulations with changes in the spatio-temporal patterns of cortical activity). In contrast, the existing compartmental models by Hasselmo, Barkai *et al.* [123, 133] and Wilson *et al.* [28] were limited to 298 and 4500 neurons, respectively, due to their computational cost. More efficient integrate and fire cortical models incorporating 10^4 neurons and 100 synapses per neuron running on a set of workstations (Digital Alpha, Sun Sparc 20 and HP 712/100) took 1-3 CPU hours to simulate 5000 ms of activity in the network [39]. A quantitative comparison between the MBED simulator and the efficient cortical simulations based on integrate and fire models is possible by applying correction factors to account for the difference in network size and computer architecture.

Considering a performance ratio of 2 between an AMD-K6 350 computer and the workstation Sun Sparc 20 as measured with the LINPACK benchmark (data obtained from the *Performance Database Server* [161]) and a ratio of 30 between the total number of synapses in the MBED cortical model and that in [39], the event-driven framework provides a scaled reduction in the CPU time corresponding to a factor of 45.

A similar comparison with the cortical model in [28], considering a factor of 60 between the performance of an AMD K6 350 and the Sun 3/260 [161] and a three

fold increase in network size, the MBED framework results in a decrease of the CPU time in a factor of 11.

Chapter 9

MBED simulations on Beowulf architectures

9.1 Introduction

Previous Chapters in this thesis have made use of commodity PC-based single processor architectures as the preferred hardware platforms for MBED neural simulations. Work on large scale neural simulations has often resorted to parallel architectures to achieve the necessary processing power [18, 70, 72, 162]. Although substantial performance increases have been demonstrated with hypercube architectures [163], the cost of these platforms and the considerable development involved in the customization of the simulation environments have limited the impact of parallel architectures in the field of neural simulation. Beowulfs constitute an emerging technology aiming at delivering parallel processing power at a reasonable cost by interconnecting commodity single processor PC-based architectures with high speed data links [164, 165, 166].

The message-passing nature of the MBED model and simulator makes them suitable for this platform. In order to assess the performance increase and scalability achievable with Beowulfs, the simulator was modified to allow the partitioning of the simulated neural aggregate and the concurrent simulation of individual partitions on different nodes of the cluster.

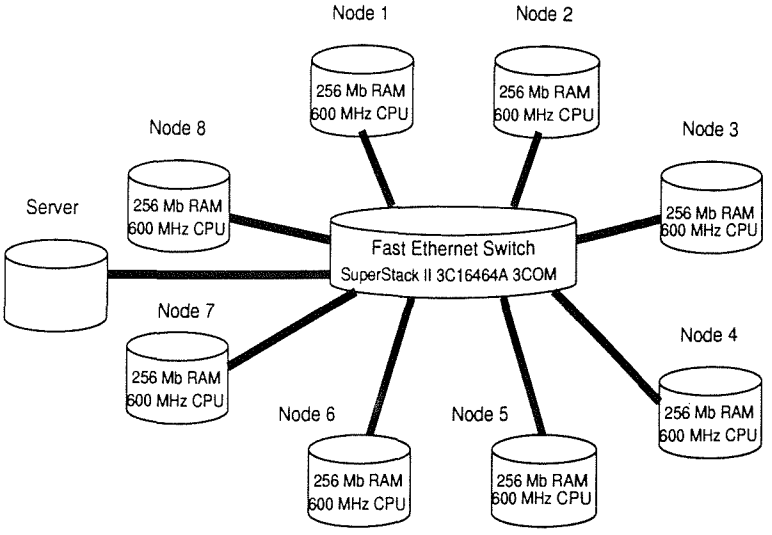


Figure 9.1: Schematic diagram of the cluster

9.2 The Beowulf platform

The Beowulf platform under test is illustrated in figure 9.1. It consists of 8 single processor Athlon (AMD-K7) machines running Linux RedHat 6.0 with an aggregate peak performance of 900 MegaFlops, with 2 Gigabytes of memory (256 Mbytes per node) and 100 GigaBytes of disk space. Two SuperStack II 3C16464A 3COM Fast Ethernet switches interconnect the nodes in a star-like topology. An extra node (totalling 9 nodes) functions as a server in charge of job scheduling across the 8-node architecture and other maintenance tasks. For this purpose, the PBS (Portable Batch System) software package is available. This computer does not participate in distributed computations.

9.3 Parallelization of the MBED simulator

Concurrent execution of the MBED simulator can be achieved with the command *qsub*, part of the PBS package, which schedules the execution of multiple images of the program across the cluster. It takes one command line argument,

```
$ qsub job_description_script
```

where *job_description_script* is a text-based file containing the information necessary for job scheduling. The script used for the simulations in this Chapter

follows,

```
#! /bin/sh
#PBS -N MPI_TEST
#PBS -m be
#PBS -l nodes=8
# executable including any command line arguments
EXE="simulator -batch script.i"
#to run a LAM MPI job
mpidistr="lam"
. /usr/local/mpi_scripts/pbs.function
```

The most relevant entries are the number of nodes allocated in the cluster, specified with the directive *PBS -l nodes=node_number* and the command line to be executed, specified by *EXE=command_line*. In the code above, the command line *"simulator -batch script.i"* invokes the simulator with the script file *script.i* as the input stream (in place of the keyboard) in all eight nodes.

The strategy followed to distribute the simulation across the cluster was to partition the neural aggregate in sub-aggregates, each one being assigned to a single node in the cluster.

Several enhancements were made to the MBED simulator to make it suitable for this parallel environment; mainly, the implementation of a synchronization mechanism between concurrent sub-aggregate simulations and the addition of inter-process (across-cluster) neuronal communications to account for inter-aggregate axonal bundles.

Both additions made use of the LAM 6.3.1 [167] free implementation of the Message Passing Interface (MPI) [168] libraries .

9.3.1 Process synchronization

Process synchronization is necessary in the context of distributed message-based event-driven simulation [169] in order to compensate for the unavoidable workload imbalance between simulation processes. Without a synchronization mechanism, a time lag would arise between the simulation clocks of different processes, which could lead to the loss of inter-process messages when the simulation clock at the receiving end is advanced with respect to that of the process originating the message.

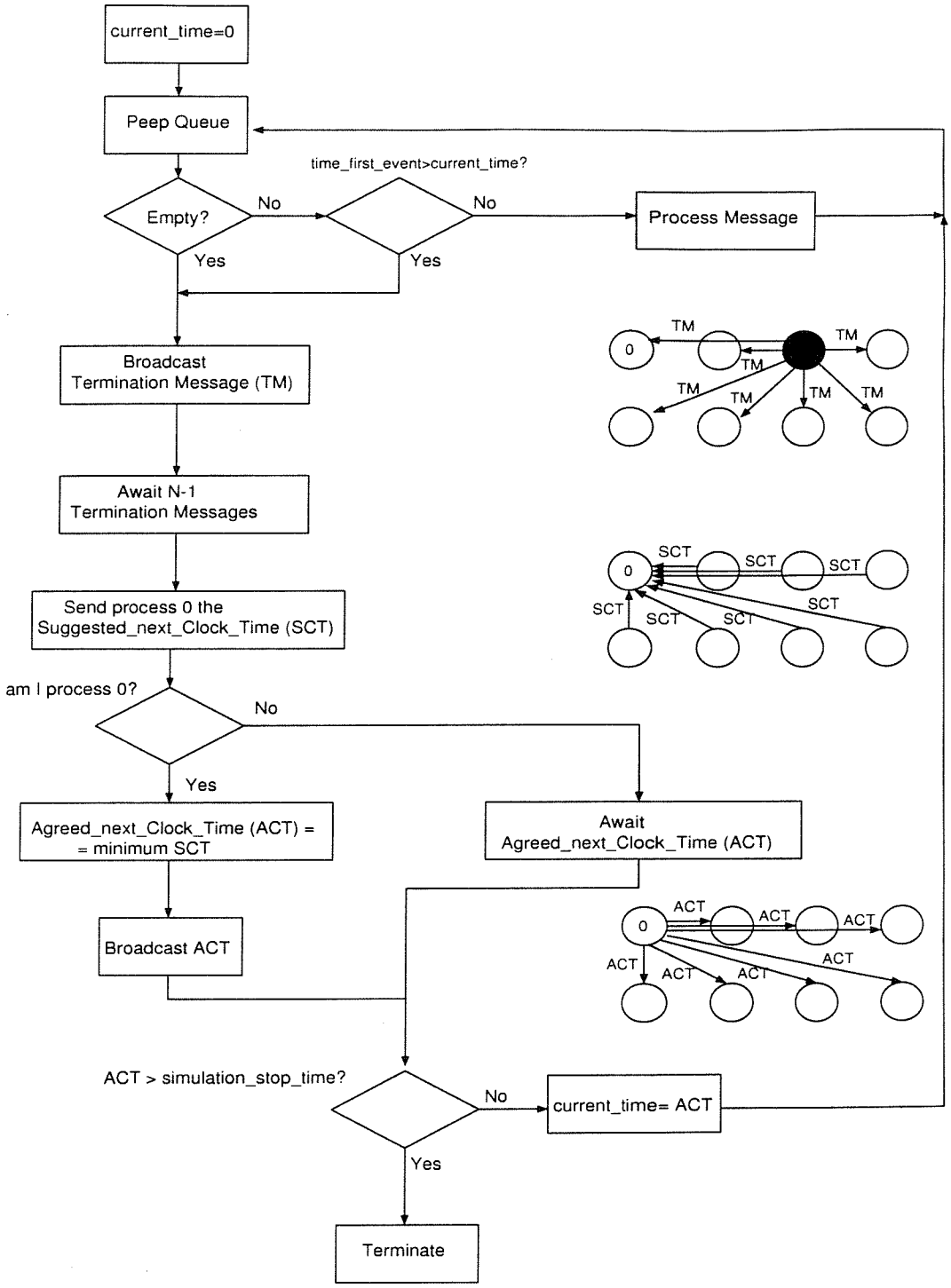


Figure 9.2: Synchronization algorithm to achieve a cluster-wide coordinated simulation clock

The MBED simulator was modified to incorporate such a synchronization mechanism. Figure 9.2 provides a complete flow-chart description of the algorithm.

Upon initialization, all processes are synchronized to the time slice $t = 0\ ms$. At the end of each of the subsequent time steps, a *termination* MPI message is sent to all processes. At this point, the process awaits the reception of the corresponding N-1 *termination* messages (N being the number of nodes) from the remaining nodes.

All processes having finished the current time slice, they propose a next value for the global simulation clock as the scheduling time of the first message in their respective priority queues. Each process broadcasts its proposed value to process 0, which acts as the coordinator. Of the proposed times, process 0 chooses the minimum and broadcasts the value to the rest of the processes, which set their respective local simulation clocks to the agreed values and start processing the messages in their queues scheduled for this consensued clock time.

The simulation finishes when the agreed global simulation clock takes a value beyond the specified simulation stop time, in which case all processes terminate.

9.3.2 Inter-module communication

Inter-module communication accounts for axonal bundles which carry action potentials across sub-aggregates.

The MBED simulator minimizes the number of MPI messages for action potential propagation by means of a buffer. Throughout the simulation, the firing of a neuron (a transition in its burst block to state *on*) triggers the addition of its neuron identifier, a 4-byte integer, to the buffer. Upon finishing a time slice or whenever the buffer is full, its contents, a list of neuron identifiers and a 4-byte header set to the actual number of entries in the transmitted data structure, are sent as an MPI message to those nodes which, in the previously specified neuronal topologically, directly receive axonal tracks. The experiments carried out in this Chapter made use of a 10 Kbyte buffer.

9.4 Results

The cortical model described in Chapter 8 was chosen as the atomic sub-aggregate (the portion of the network simulated by one node) because previous studies had shown that it was capable of replicating experimental data on cortical dynamics. Thus, the results of the benchmarking are likely to be representative of the

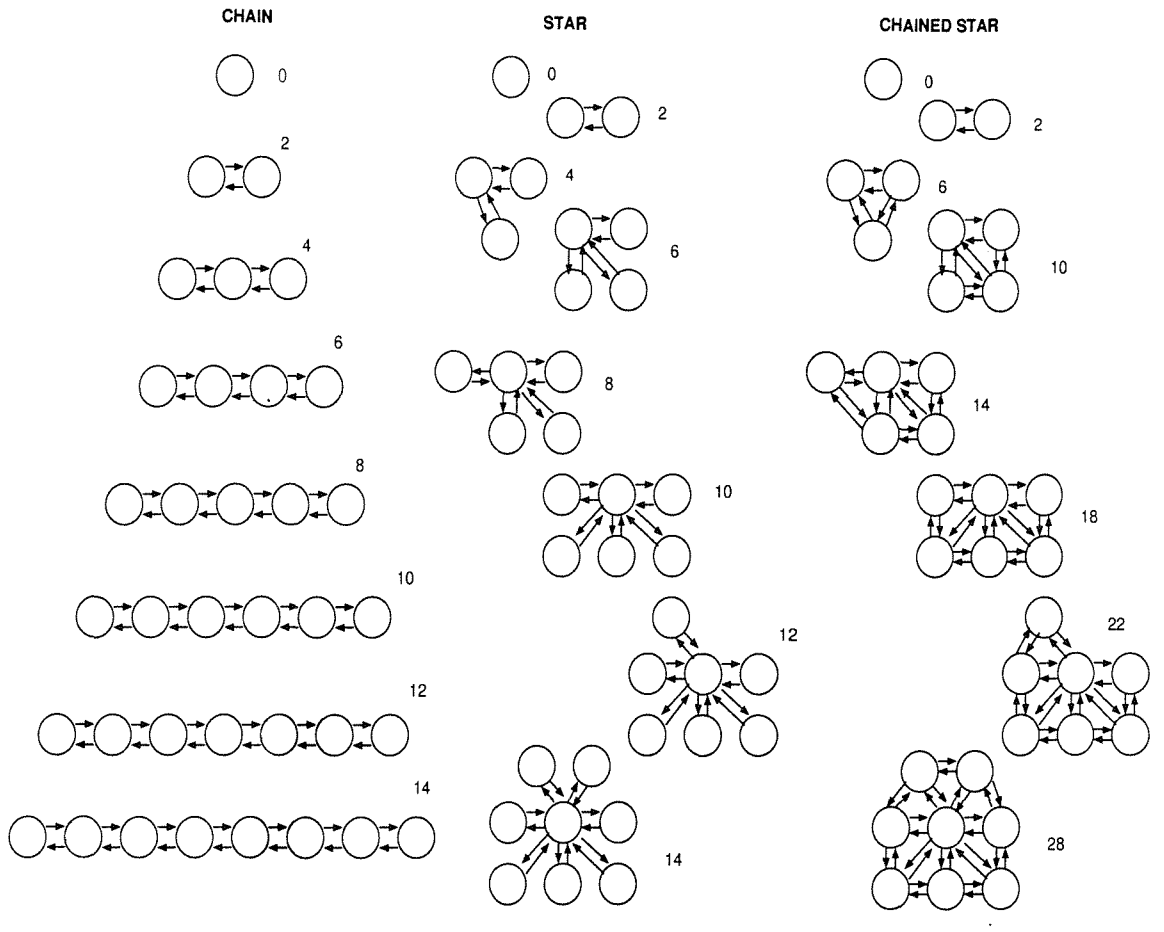


Figure 9.3: Chain, Star and Chained-star topologies used for performance benchmarking

performance attainable with a wide range of biologically realistic neural simulation problems.

Given an arbitrarily chosen set of brain areas, only a subset of all the possible pairs would be directly connected by axonal bundles. Assuming a one-to-one mapping between nodes in the cluster and modelled brain regions, it follows that several logical cluster topologies are possible. For performance evaluation, models with various numbers of modules (1-8) and patterns of axonal bundles were simulated in order to explore the effect of these parameters on the elapsed time. Figure 9.3 depicts the simulated topologies. In addition to those contained in the figure, a completely unconnected aggregate, without inter-module axonal bundles altogether, was also simulated.

Further, special care was taken to ensure that the performance evaluation was

carried out with realistic neural simulations; since the performance of an MBED simulation engine is strongly affected by the network activity pattern, misleading performance studies can result from simulations with exceedingly low or high neuronal activity. Parameter space search is needed to find the configuration that results in realistic activity in all the nodes conforming the cluster. This is a computationally costly problem in itself, and aggravated by the fact that a new set of parameters has to be found for each one of the logical cluster topologies under test.

A convenient simplification to the network model was put in place to achieve realistic activity and inter-node communication overhead for all nodes while eliminating the need for computationally intensive parameter space searches. Each sub-aggregate includes a pool of neurons which provides stimulation to the local model. The inter-module neuronal spikes transported by afferent bundles (and implemented by means of MPI messaging) is actually transmitted to retain the performance degradation caused by communication overhead. This guarantees the validity of the performance results. However, the receiving end disregards the incoming trains of action potentials, and takes its input from the stimulus neuronal pool. The dynamics of such a network is simpler and the parameter space search needs to be carried out once and with a single sub-aggregate rather than with the entire network.

In this way, (1) all sub-aggregates display a realistic level of intra and inter-aggregate activity irrespective of network size and topology, (2) the inter-node data are actually transmitted to evaluate the effect on the performance and (3) computationally expensive parameter space searches are avoided.

Figure 9.4 plots the time taken by simulations of 1 s of network activity. The lower trace corresponds to the measured elapsed times averaged over the four topologies tested: unconnected, chain, star and chained-star. For comparison, the upper trace represents a linear estimation of the time taken to simulate equivalent network sizes on a single-processor architecture. Actual measurements of single-processor times could not be performed given that individual sub-aggregates, totalling $1.753\ 10^5$ neurons, were already at the limit of memory resources. The estimated values for a single-processor platform were calculated with a linear approximation; the time taken by the simulation of a network aggregate of N sub-aggregates running on a single node (assuming enough memory space) was approximated by N times the measured time taken by a single sub-aggregate running on a single node.

The flat profile of the Beowulf system indicates that, within the measured range

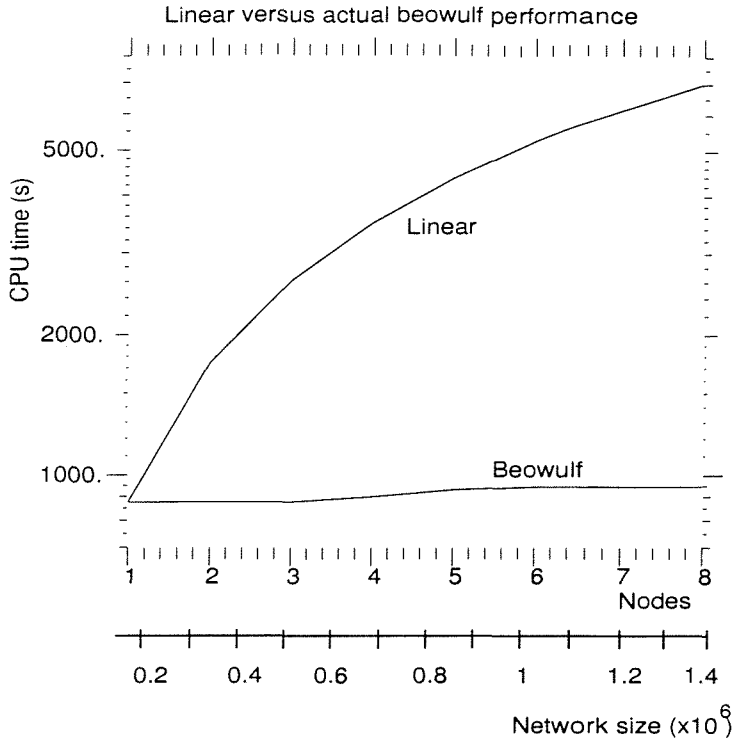


Figure 9.4: Elapsed time for single node and Beowulf architectures versus number of nodes and network size

of 1-8 nodes, network size can be increased with nearly constant elapsed times. Quantification of Beowulf results is possible with figure 9.5, which shows the elapsed time for the four network layouts tested. Considering the shortest (874.14 s) and longest simulations (947.4 s), an 8-fold increase in network size (from 1 to 8 nodes) results in a mere 8.3% in elapsed time in the worst case.

The low overhead incurred by the migration from single node to Beowulf distributed processing results from the low communication requirements when compared to the computation part. Further, the used inter-node bandwidth represents a small fraction of the available bandwidth: The measured average size of an inter-node packet carrying the contents of the spike buffer described in the previous section was 8915.76 bytes (2227.94 spikes $\times$ 4 bytes per spike + 4 bytes header). The number of packets travelling through the switch during a 1 s simulation was measured to be $C \times 10^3$, where C is the number of inter-node unidirectional channels (arrows in figure 9.3) in the topology under test. For instance, 28000 packets were transmitted for the 8 node chained-star network which results from 28 inter-node channels and 1000 time slices of 1 ms per simulated

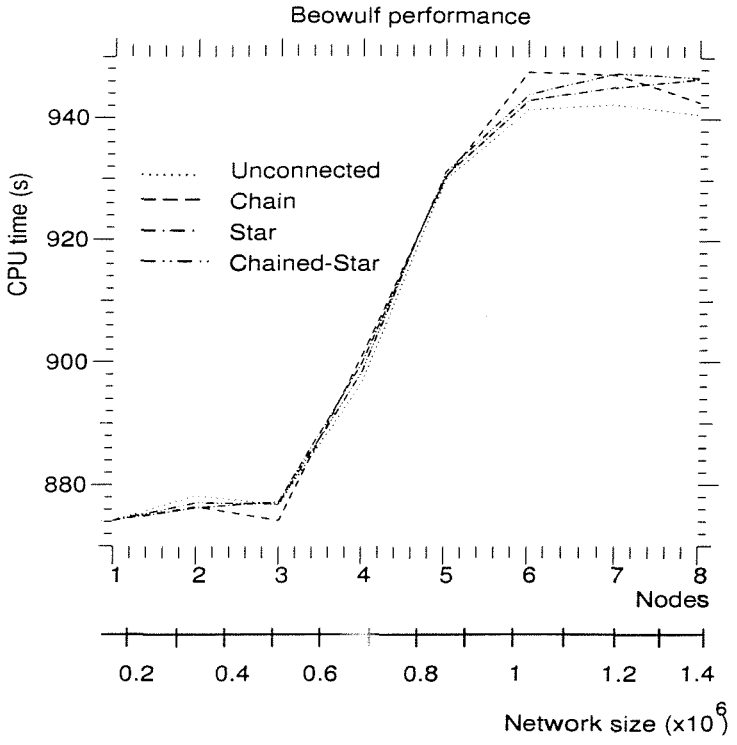


Figure 9.5: Elapsed time versus number of nodes and network size for various network topologies

second. It follows that the total amount of data communicated between nodes throughout the entire simulation was approximately 28000 × 10 Kb, 280 Mbytes. This corresponds to 320 Kbytes/s (considering an elapsed time of 874.14 s), which is well below the approximately 40 Mbytes/s of available bandwidth (estimated with in-house benchmarking tools).

9.5 Conclusions

This Chapter has presented preliminary results concerning the scalability of a message-based event-driven framework for biologically motivated neural simulation on Beowulf architectures. The experiments carried out with an 8-node Beowulf indicate that the migration from a single node to this parallel environment results in an 8-fold increase in aggregate size with an 8.3% increase in elapsed time; the total size of the distributed aggregate reached 10^6 neurons with an average of 179 synapses per cell.

Further tests are needed with Beowulfs in excess of 8 nodes to explore the scalability to larger simulations. Nevertheless, the results already obtained with an 8-node cluster indicate that low communication overhead can be achieved with an event-driven framework, resulting in efficient scalability.

The cortical model used for the benchmarking purposes has been developed as part of ongoing research on the dynamics of the piriform olfactory cortex. This cortical region contains approximately 10^7 neurons with several thousand synapses per cell [118]. Further code optimization and an increase in the number of nodes promise to make such problem sizes tractable using clusters of commodity computers.

Chapter 10

Further work and final comments

Further work is proposed in this Chapter dealing with three aspects of the MBED framework developed in the thesis; at the single cell level, the enhancement of the functionality of the MBED model. At the small network level, the application of the MBED paradigm to multielectrode-array (MEA) data and, in the domain of large scale networks, the extension of the cortical model presented in Chapter 8 to a multi-module model of the olfactory system.

Finally, the main outcomes of the thesis are outlined.

10.1 The single cell MBED model

Three enhancements to the model contained in Chapter 5 would extend the capabilities of the MBED framework to reproduce biological data; the introduction of dendritic delay, t_{den}, the substitution of fixed parameters, e.g. the synaptic efficacy and inter-spike delay in bursts, w_{syn} and t_{ref}, by variable values dependent on w_{sum} and the addition of synaptic plasticity.

Dendritic trees introduce a propagation delay between the activation of a synapse and its resulting effect at the cell body [24]. The addition of a delay on the γ channel from synapses to threshold block and its associated parameter t_{den} (see model depicted in figure 10.1) would account for this latency. This modification is necessary in order to accurately replicate the timing of synaptic events present in field potential recordings as those obtained, for instance, by Ketchum [124]. Figure 4.7 in Chapter 4 showed the response of the piriform cortex to shock stimuli. The trace corresponding to the strong shock response can be segmented in intervals according to the synaptic type whose contribution to the recorded potential is

maximal. The modelling of this temporal sequence of synaptic activations will be possible after the addition of the dendritic delay to the MBED model.

A synaptic weight, w_{syn}, obtained as a function of w_{sum} would account for the variability of the synaptic efficacy due to changes in membrane voltage. The substitution of the fixed refractory period, t_{ref}, by a variable parameter dependent on w_{sum} would also allow the implementation of adaptation within bursts. The inter-spike delay could be made dependent on previous activity of the cell.

Finally, synaptic plasticity, the proposed mechanism of learning in the nervous system, can be implemented in the MBED neuron model. The execution of learning algorithms (e.g. Hebbian learning rules [170]) would involve the update of synaptic parameters throughout the simulation. This gives rise to a potential problem; since different synapses experience different sequences of activation, their learning algorithms will most likely adjust their respective synaptic parameters to different values. This is not a problem when each synapse has its associated copy of the complete synaptic parameter set. However, the data structure proposed in Chapter 6 for memory-efficient large scale simulations, relies on a table of synaptic models. Each synapse stores an index into this table. Thus, one synaptic parameter set is shared by multiple synapse instantiations. Learning is still possible by providing a sufficiently large set of possible synaptic models and implementing the adaptation algorithms to act on the synaptic type index associated to each synapse rather than on the synaptic parameters themselves. For instance, long term potentiation (LTP) would be implemented as a change of the synaptic type index of the potentiated synapse, taking the value of an existing synaptic model with identical synaptic delay (t_{del}) and activation duration (t_{dur}) but with increased synaptic weight (w_{syn}).

10.2 MEA data modelling

The above mentioned modifications to the MBED neuron model would extend its time-constant parameter set and also add time-dependent adaptation algorithms. For the goal of the MBED framework is two fold, efficient and biologically realistic modelling, experimental characterization of these parameters and algorithms will be of fundamental interest.

Multielectrode-array technology (MEA) is emerging as a tool for quantitative study of the cooperation of cells in small networks. This technology relies on microfabricated structures, using silicon as their substrate, which accommodate a number of electrodes and neurons. Figure 10.2 illustrates such a set up, including

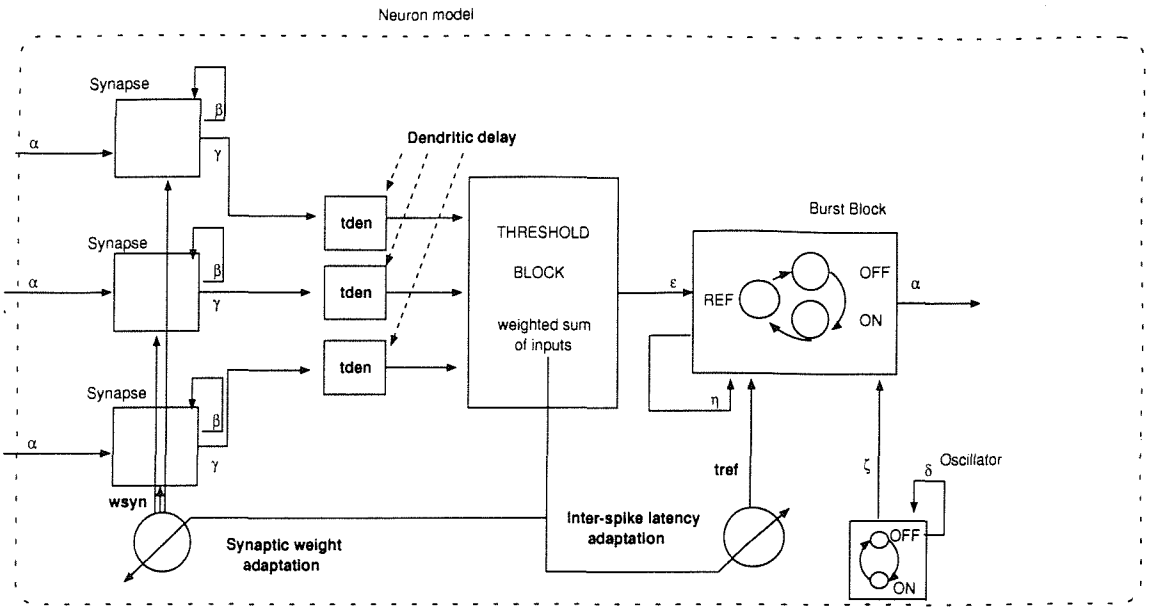


Figure 10.1: MBED model enhanced by the addition of dendritic delay and membrane voltage dependent w_{syn} and t_{ref}

optical imaging. A system for recording and stimulation of all cells in a 16 neuron network has been developed [171].

The MBED model of the piriform cortex described in Chapter 8 relied on EEG and field potential data, which typically correspond to the average activity of populations of neurons [132]. It is desirable, however, to base the network models on quantitative information regarding the interaction of individual cells making up the network. MEA technology is the ideal candidate for such studies in-vitro, since it provides independent monitoring of all cells integrating a small network.

The MBED framework is suitable for the problem of modelling MEA recordings. The parameter set associated to the MBED neuron model matches the type of data readily available with MEAs (axonal delay, burst duration, refractory period duration and others). On the other hand, MEAs do not provide detailed information as required for the development of compartmental models (e.g. ion channel types, conductance distribution, and so on).

The proposed experimental procedure would consist of two phases; in phase 1, the MEA would be used to record from 16 neurons obtained at early developmental stages which would establish connections, as is frequently the case with cultures of developing cells. In phase 2, parameter space search would be run on the MBED

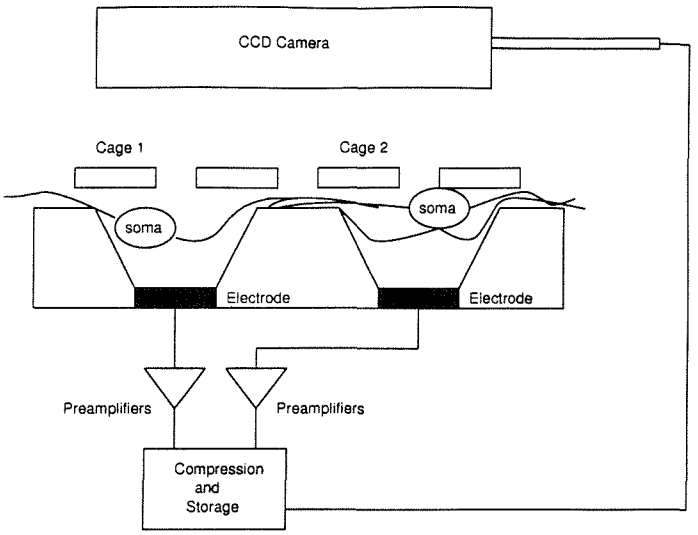


Figure 10.2: Multielectrode array setup as used in the recording/stimulation of small networks of neurons

network model in order to fit the experimental data. Successful match of experimental data will indicate an adequate modelling of small network dynamics and will lead to the extension of the model to larger (multi-module) networks.

10.3 Multi-module models

The understanding of the neural processing carried out on incoming sensory information would be facilitated by the construction of models including several of the brain modules participating in a particular sensory pathway.

Parallel architecture are capable of providing the computational power necessary for such simulations. Niebur *et al.* [70, 162] and Jahnke *et al.* [163] have demonstrated that the supercomputer CM-2 is suitable for the simulation of networks of spiking neurons of up to $4 \cdot 10^6$ cells. Such highly parallel architectures guarantee wide inter-neuron communication bandwidth and prevent quick performance degradation with increasing network sizes. Preliminary results, presented in Chapter 9, indicate that the MBED framework can be ported to low cost parallel architectures based on Beowulf clusters to achieve increases in tractable problem sizes.

The availability of a computational framework capable of simulations in the order of millions of neurons would allow the construction of a multi-module model of

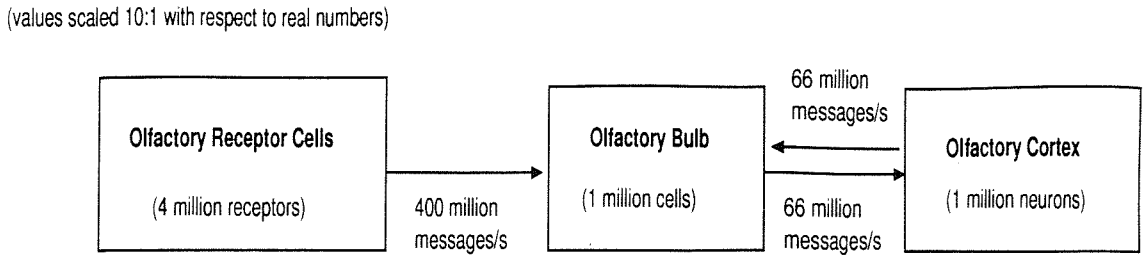


Figure 10.3: Proposed multi-module MBED model of the olfactory system

the olfactory system. Firstly, an MBED model of the olfactory bulb including three populations of cells, mitral, tufted and granular, would be constructed. Secondly, a pool of sensory cells accounting for olfactory receptors would be added to the model. Finally, sensory receptors, olfactory bulb and piriform cortex would be connected to form a multi-module model to support studies of the dynamics of smell recognition and segmentation.

Figure 10.3 depicts the proposed model, including estimations of the inter-module communication overhead. All values have been scaled down with an approximate real/model ratio of 10:1. To estimate the overhead caused by inter-module communication, a worst-case scenario is considered; all neurons firing at 100 Hz and contributing with one message per spike. Note that, in the olfactory bulb, only mitral and tufted cells, taken to be two thirds of the total bulbar cell population, contribute to the communications towards the piriform cortex. Equally, in figure 10.3, two thirds of the piriform cortex cells are assumed to synapse back onto the olfactory bulb.

The network size already achieved on Beowulf clusters (above 10^6 neurons) indicates that further increases in the number of nodes (up to approximately 50 PCs) would be sufficient to construct and simulate the described model.

10.4 Final comments

The main outcomes of this thesis are:

- Development of a neuron model based on the finite state automaton formalism incorporating synaptic timing (activation latency and duration and synaptic efficacy), axonal delays, single spike, bursting and pace making dynamics and excitation and burst truncation inhibition thresholds.

- Implementation of an efficient MBED simulator incorporating synaptic model structures for memory efficiency, and an LUT based queue and a low-overhead memory manager for improved CPU and memory efficiency.
- The application of the MBED framework to a small network model of the locomotory circuit of *C. elegans*, consistent with quantitative experimental data obtained from wild-type, mutated and laser ablated animals.
- The development of a large scale model of the piriform cortex including 10^5 neurons with physiologically realistic properties and its validation by comparison with field potential recordings and EEGs.
- Preliminary tests on the scalability of MBED cortical models utilizing commodity Beowulf architectures. The low inter-process communication overhead made possible an 8-fold increase in problem size with an 8% increase CPU time.

Page 220
missing from
this Thesis

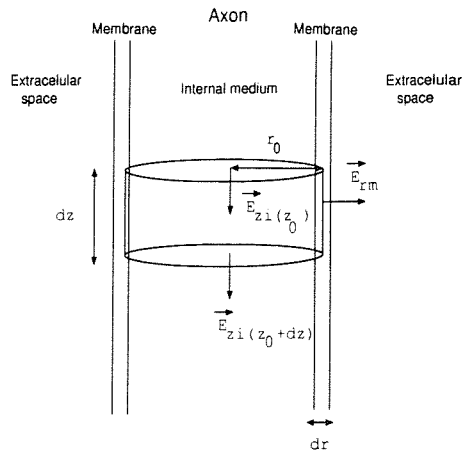


Figure A.1: Schematic representation of an axon

- The core current, assumed to flow axially inside the axon, is considered constant within a cross-section.
- The membrane is a partly conductive partly capacitive shield which completely seals the infinite length cylindrical axon.
- The external medium, an purely conductive open space, provides a low resistance path that can be approximated by a zero resistance equipotential ($\Phi_e = 0$) conductor.

When the surface integral of equation A.4 is applied to the differential volume of figure A.1 it yields,

$$(\sigma_i \mid \vec{E_{zi}}(z_0) \mid -\sigma_i \mid \vec{E_{zi}}(z_0 + dz) \mid)\pi r^2 - 2\pi r\, dz\, (\sigma_m \mid \vec{E}_{rm}(r_0) \mid -\epsilon_m \frac{d \mid \vec{E}_{rm}(r_0) \mid}{dt}) = I \tag{A.5}$$

where the subindices i, e and m correspond to intracelular, extracelular and membrane and z and r indicate axial (alongside the z axis) and radial components. The righthand term, I, accounts for the current injected through a microelectrode.

Since the cable equation for axons and dendrites is often expressed in terms of the transmembrane voltage, V, the electric field term, $\vec{E}$, in A.5 can be substituted taking into account the following equalities,

$$\mid \vec{E}_{zi} \mid = \frac{d\Phi_i}{dz} \tag{A.6}$$

and

$$\mid \vec{E}_{rm} \mid \approx \frac{\Phi_i - \Phi_e}{dr} = \frac{\Phi_i}{dr} \tag{A.7}$$

to yield,

$$\sigma_i \, dz \, \frac{d^2\Phi}{dz^2}\pi r^2 - 2\pi r \, dz \, \sigma_m \frac{\Phi}{dr} - 2\pi r \, dz \, \epsilon_m \frac{d\Phi}{dr \, dt} = I \tag{A.8}$$

Further arrangements lead to,

$$dr \, \frac{\sigma_i \pi r^2}{\sigma_m 2\pi r} \frac{d^2\Phi}{dz^2} - \Phi - \frac{\epsilon_m}{\sigma_m} \frac{d\Phi}{dt} = \frac{I dr}{\sigma_m 2\pi \, r \, dz} \tag{A.9}$$

which, with the aid of the following definitions,

$$\lambda = \sqrt{\frac{r_m}{r_i}} = \sqrt{\frac{g_i}{g_m}} = \sqrt{\frac{\sigma_i \pi r^2}{\frac{\sigma_m 2\pi \, r}{dr}}} \tag{A.10}$$

$$\tau = \frac{\epsilon_m}{\sigma_m} \tag{A.11}$$

$$V = \Phi_i - \Phi_e = \Phi_i \tag{A.12}$$

takes the usual form of the cable equation for neuronal branches

$$\lambda^2 \frac{d^2V(t,z)}{dz^2} - V(t,z) - \tau \frac{dV(t,z)}{dt} = \frac{I dr}{\sigma_m 2\pi \, r \, dz} \tag{A.13}$$

Appendix B

CCD imaging of locomotion in C. elegans

Video recordings of behaving C. elegans were obtained with a standard IMT-2 Olympus microscope and a $\times 4$ objective. Worms were grown on agar-filled Petri dishes. Adults were identified by their body dimensions, isolated and imaged while performing forward and backward locomotion, reversal and body coiling. A subset of the 16-bit depth 320 by 240 pixel gray scale images is included in this Appendix.

Details on C. elegans protocols can be found in [94].

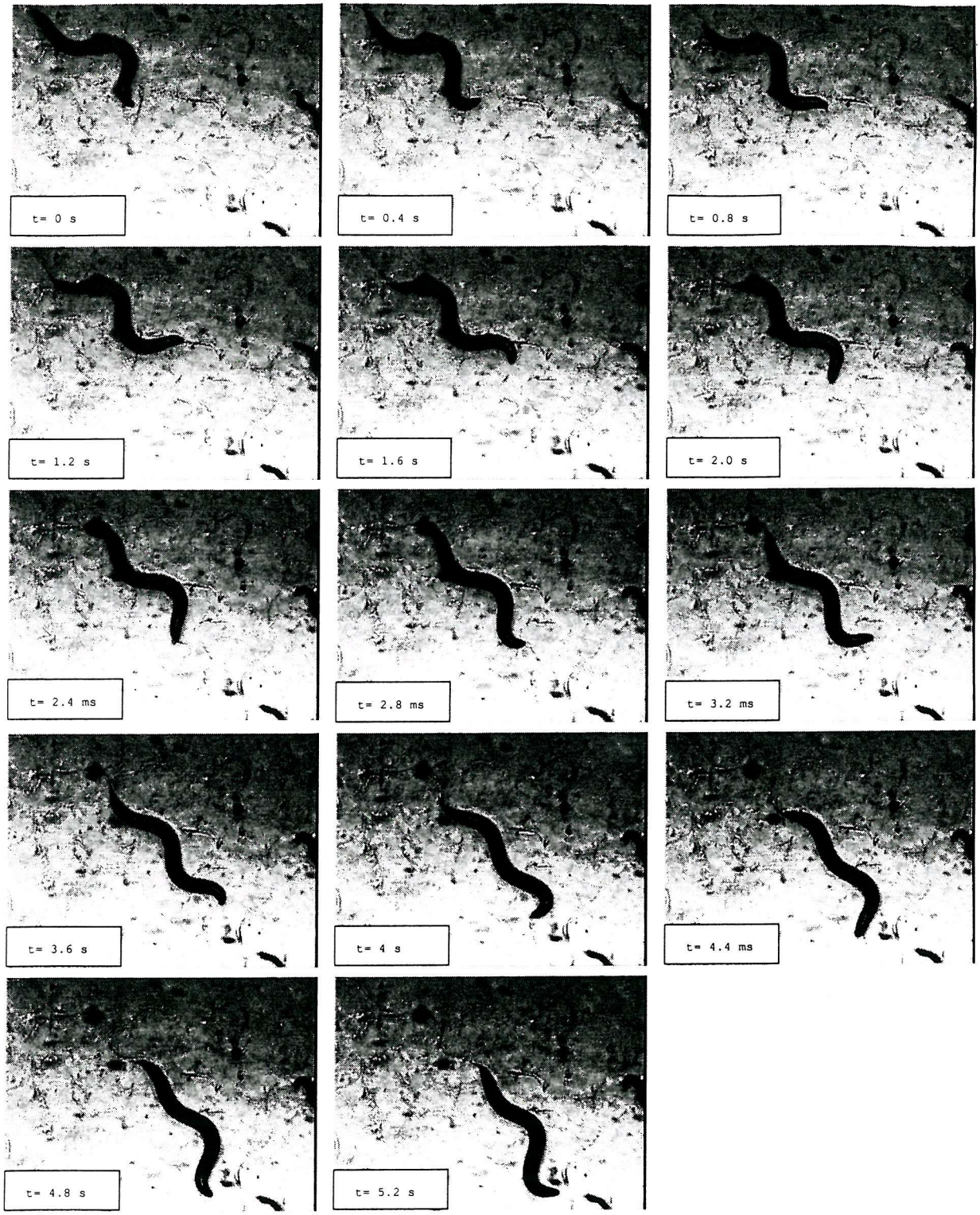


Figure B.1: Forward locomotion

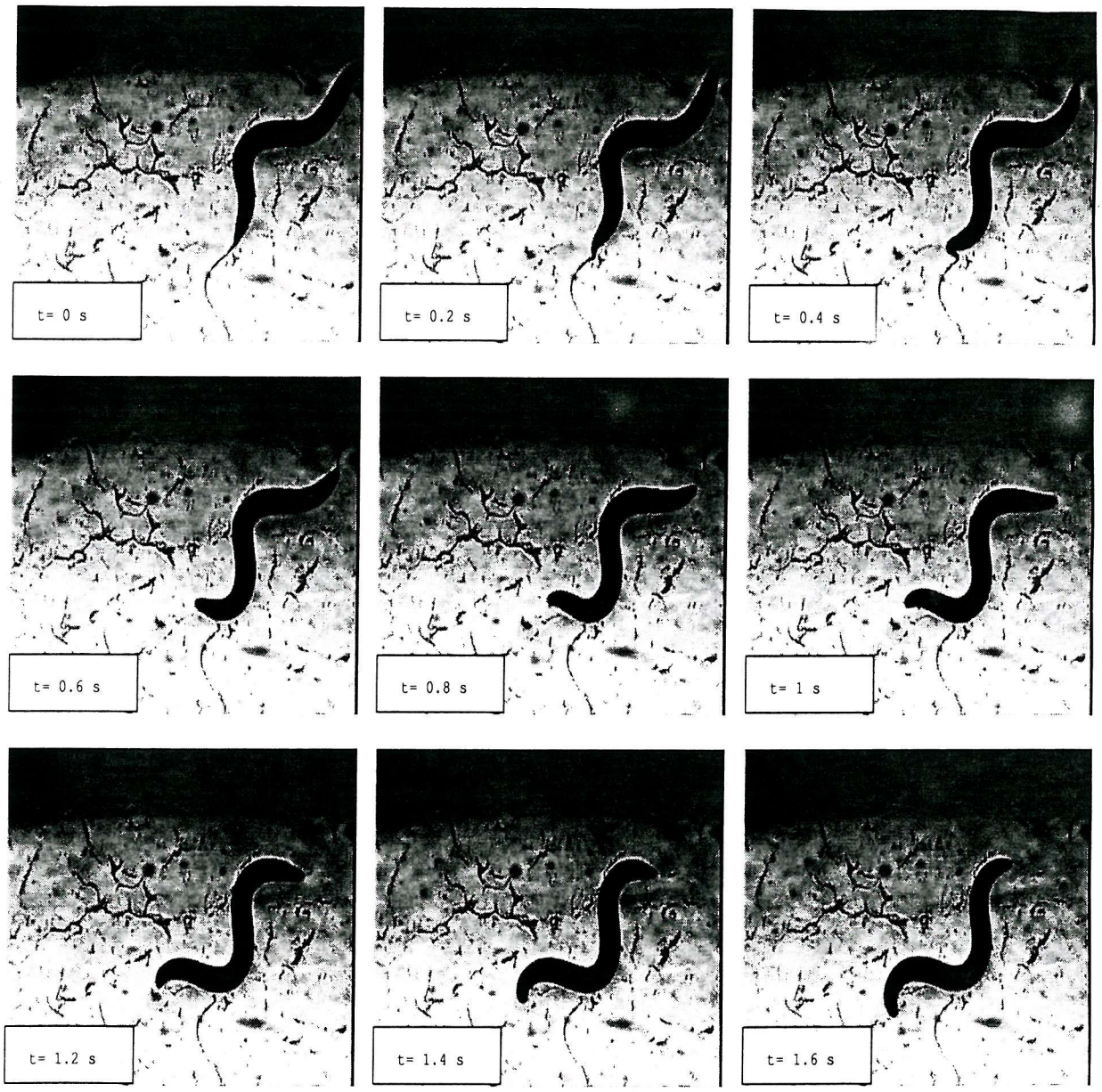


Figure B.2: Backward locomotion

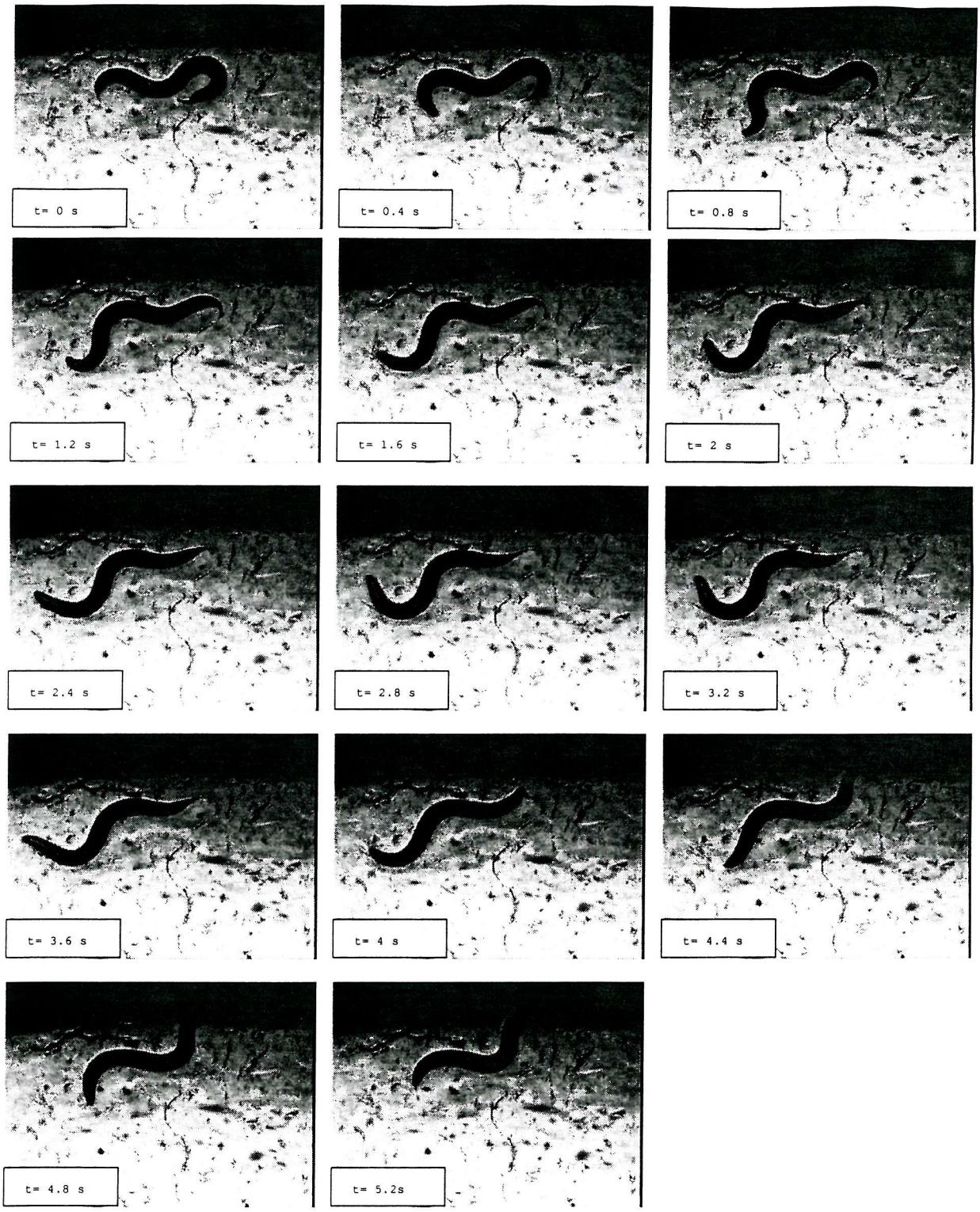


Figure B.3: Reversal

Figure B.4: Whole body bending

Appendix C

The image processing algorithm

The algorithm used to process the video recordings of C. elegans is summarized in the flow chart of figure C.1.

A threshold operation is applied first to the unprocessed frames (figure C.2-A),

$$im_{thres}(x,y) = \begin{cases} 1 & if\ im_{unproc}(x,y) > th \\ 0 & if\ im_{unproc}(x,y) <= th \end{cases} \quad \text{(C.1)}$$

The resulting image shows the worm as a dark stain on white background (figure C.2-B).

A contour closing algorithm is then used to eliminate the bright stains which appear inside the body as a result of light reflections (figure C.2-C).

Next, a sliding window of size NxN pixels is used to find a region of the image which includes a segment of the body edge. The windows scans the image until a region is found where the number of dark pixels cover between 30% and 90% of its surface. This condition ensures that the window overlaps partially with the body of the worm and contains a segment of its edge (figure C.2-D),

```
for (i=0;i<=imax;i++)
for (j=0;j<=jmax;j++)
{
if   ( CountBlackPixels(i,i+N,j,j+N) > 0.3*N*N ) &&
           ( CountBlackPixels(i,i+N,j,j+N) < 0.9*N*N)
  break;
}
```

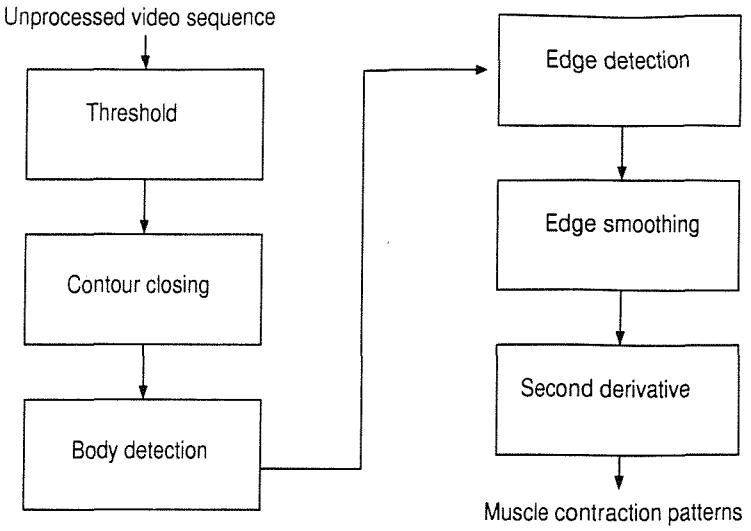


Figure C.1: Flow chart of the image processing algorithm

where CountBlackPixels(x1,x2,y1,y2) is a function which counts the number of dark pixels in the image area delimited by (x1,x2,y1,y2).

The selected region is scanned to find a pair of contiguous pixels with one point inside the body and one outside (figure C.2-E),

```
for (i=x1;i<=x1+N;i++)
for (j=y1;j<=y1+N;j++)
{
if   ( image(x1,y1) != image(x1+1,y1))
break;
}
```

The complete edge of the body is then detected, using a contour following algorithm which operates starting at the pair of pixels. The extracted contour is stored as a list of position vectors (figure C.2-F).

The local curvature is obtained from this parametric representation calculating the second derivative of the contour vector sequence (figure C.3-A). Low pass filtering is required previous to the determination of the curvature for smoothing of the data. The tips of the head and the tail show as the two maxima in the second derivative of the contour.

Finally, the sequence of second derivative functions obtained from multiple consecutive images are aligned and shown in matrix format in figure C.3-B.

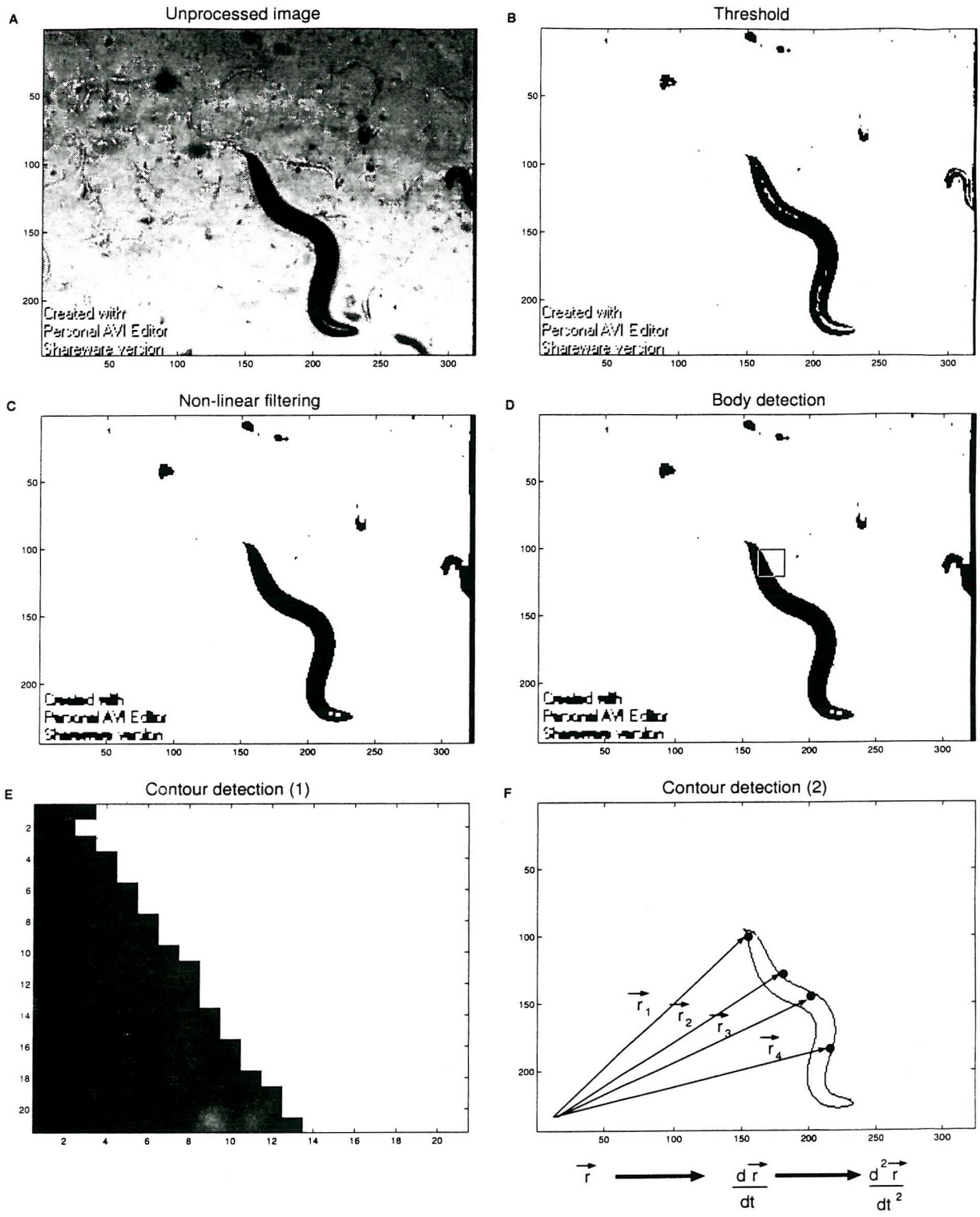


Figure C.2: Image processing algorithm (I)

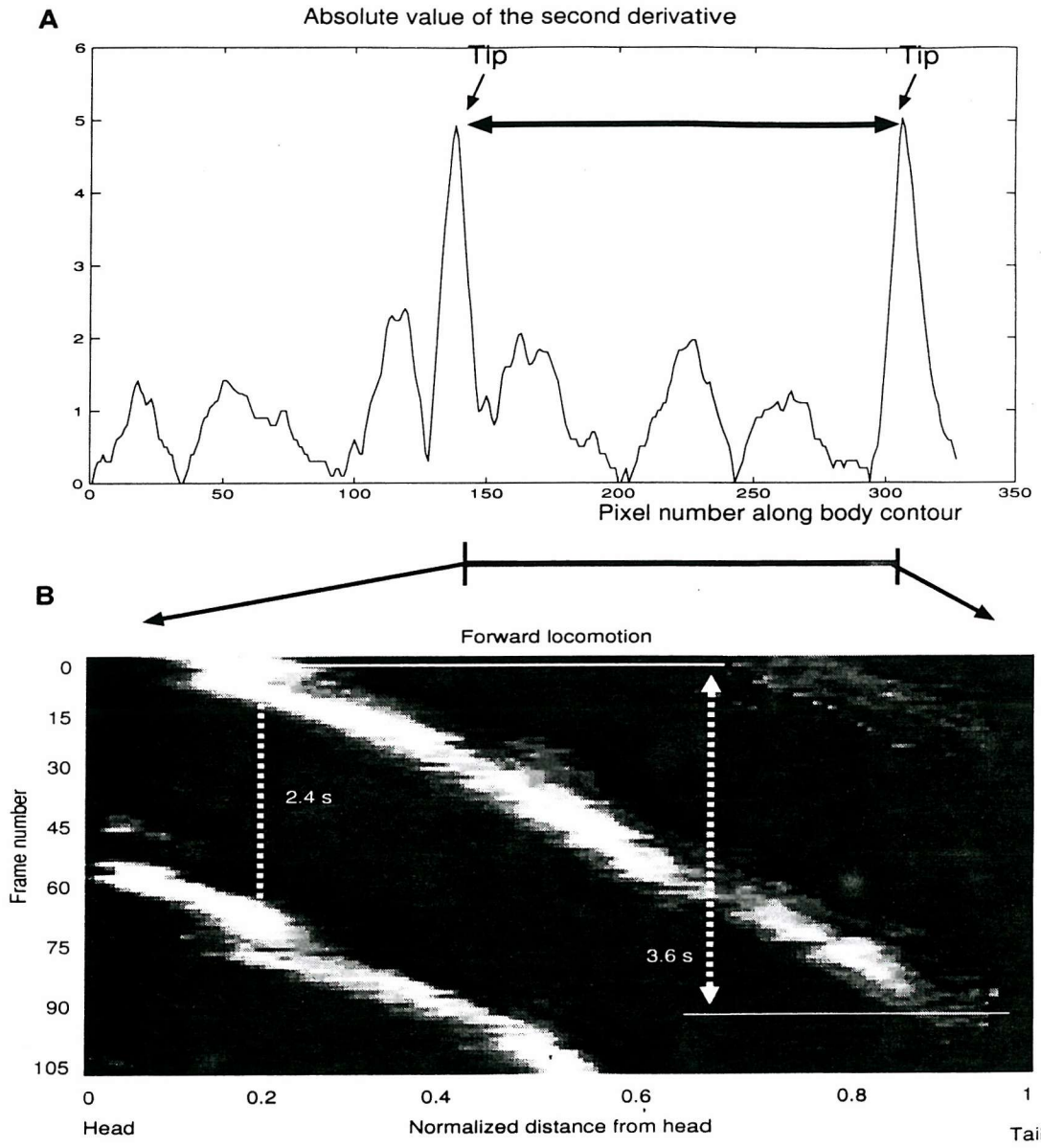


Figure C.3: Image processing algorithm (II)

Appendix D

Papers

Event based neuron models for biological simulation. A model of the locomotion circuitry of the nematode C. elegans. Claverol E.T., Cannon R.C., Chad J.E. and Brown A.D. Computational Intelligence and Applications. N.E. Mastorakis (Ed.). World Scientific Engineering Society Press. 1999

Discrete simulation of large aggregates of neurons. Claverol E.T., Brown A.D. and Chad J.E. Submitted to Neurocomputing.

A large scale simulation of the piriform cortex by a cell automaton-based network model. Claverol E.T., Brown A.D. and Chad J.E. To be submitted to IEEE Trans. on Biomedical Engineering.

Scalable cortical simulations on Beowulf architectures. Claverol E.T., Brown A.D. and Chad J.E. Submitted to Neurocomputing.

Event based neuron models for biological simulation. A model of the locomotion circuitry of the nematode C.Elegans.

E.T.Claverol† R.C.Cannon‡ J.E.Chad ‡ A.D.Brown†

†Dept. of Electronics and Computer Science. University of Southampton
Mountbatten Building. Salisbury Road
Southampton SO17 1BJ
United Kingdom
‡ School of Biological Sciences. University of Southampton
Biomedical Sciences Building. Bassett Crescent East
Southampton SO16 7PX
United Kingdom
etc97r@ecs.soton.ac.uk

Abstract: - C.Elegans is a nematode whose nervous system has 302 neurons. Its swimming motion is controlled by a subsystem of 80 neurons, which are able to generate both forward and backward locomotion at variable speed. We present both a model of this circuitry based upon event-driven models of neurons and a model of the nematode's body. We test its capability of generating forward/backward locomotion. The final aim of this work is to demonstrate the feasibility of using event-based models of neurons to reproduce the fundamental behaviour of circuits of neurons not only in locomotion but also in sensory signal processing.

Key-Words: - neuronal simulation, discrete simulation, C. elegans, locomotion

1 Introduction

C.Elegans is a nematode of small dimensions (1 mm long and 80 micrometers wide) which is found in soil. It lives on bacteria which it must locate and ingest. Despite the reduced size of its nervous system,with 302 neurons [1], it still has a relatively rich behaviour. Its locomotion is based on crawling, both forward and backward and its speed of propagation may be changed depending on stimulation from environment.It also bends in an elaborate manner when mating.

Several attempts have been made to model and simulate subcircuits of C.elegans' nervous system [2] [3] [4]. Functional data is presently limited to observation of behaviour due to difficulties in electrophysiological recordings.

In this paper we present a model of the locomotion neural circuitry which accounts for a variety of behaviours. We have extracted the relevant features of it and simulated a simplified version. A mechanical model for the nematode has been proposed [2]. We extend this model from two to three dimensions to allow the evaluation of the modeled nervous system at the behavioural level.

Different types of models have been used for the simulation of single neurons or small aggregates of neurons [5]. We have chosen an event-driven neuron model which combines the rich behaviour of real neurons with efficient simulation.

2 Event based simulation versus compartmental models

Traditionally, modeling of realistic circuits of neurons has been based on compartmental mod-

els. The simulation of these models usually involves the numerical solution of non-linear differential equations (due to the non-linearity of the Hodgkin-Huxley ion channel equations). In addition to the computational requirements of the simulation, compartmental models tend to be highly sensitive to the many parameters required. This makes them specially difficult to tune and difficult to use in simulating large aggregates of neurons. On the other hand, integrate and fire models use leaky capacitors and threshold functions. This solution reduces the computational requirements for simulation, allowing simulation of large aggregates, but limits the functionality of the neurons [6].

We use event driven models of neurons, whose computation requirements allow fast simulation but maintain a relatively high complexity in the functionality of each model neuron. Discrete simulation allows exploitation of latency in biological neurons to speed up simulation.

As an example, consider the simulation of integrate and fire versus event based. In an event driven model, the number of events depends on the number of action potentials. If no action potential is generated, the processor spends no time in that neuron whereas in integrate and fire models the membrane voltage is still updated for every time step.

The drawback of discrete models is their limitation in reproducing the neuron dynamics at the membrane voltage level. Our working hypothesis is that the behaviour of a neuron can be captured by pulse based models.

3 Event-driven neuron model

In Figure 1 the different blocks making up the model neuron are presented. It is an asynchronous system based on pulse modulation. Signals (pulses) are evaluated and propagated in the direction of the arrows in the diagram.

Signals originating in chemical synapses and electrical junctions (gaps) enter the neuron from the left hand side of Fig. 1. Blocks marked as gaps/synapses behave as monostable oscillators triggered by the incoming pulses (which model biological action potentials). They are stretched (thereby implementing a low pass filter) and a propagation delay is also introduced.

Delayed and stretched pulses reach the multifunctional block. This stage is responsible for evaluating different combinational functions with the inputs from the gaps/synapses. The specific function implemented depends on the functionality of the neuron.

For example, for a neuron working as a correlation detector, this stage would calculate the usual weighted sum of inputs and apply a thresholding function to it.

$$out = TH(\sum_{i=1}^{N} x_i * w_i) \qquad (1)$$

Where TH is the thresholding function, N is the number of synapses, x_i is the value of input i (1 or 0) and w_i is the weight of the synapse i. Negative weights account for inhibitory inputs whereas positive weigths account for excitatory inputs.

For a bypass neuron, the model behaves as a D flip-flop. Inputs are flagged with type ids therefore grouping input synapses into two classes (clocking synapses and D synapses setting the future state). When the neuron is in its asserted state, its output is a train of pulses. In the deasserted state its output is silent.

Both correlation detectors and bypass neurons are used in our model of the locomotion system in C.Elegans.

Two outputs drive the burst generator block from the multifunctional block. The line labeled "exc" is asserted when input activity is exciting the neuron. The " inh" line is asserted only when input activity gives rise to sufficient inhibition (as opposite to excitation). Both lines may be deasserted indicating that the input pulses do not force neither excitation nor inhibition.

To account for spontaneous activity an astable oscillator drives the bursting block. In our locomotion circuitry this block is only active for AVB and AVA inter neurons which control the speed of locomotion.

The last block, AP shaper, generates pulses of variable duty cycle when the output of the burst

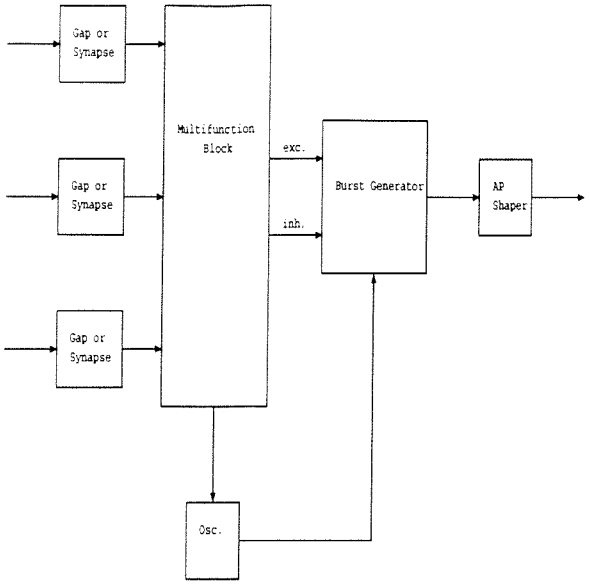


Figure 1: Blocks diagram of the neuron model.

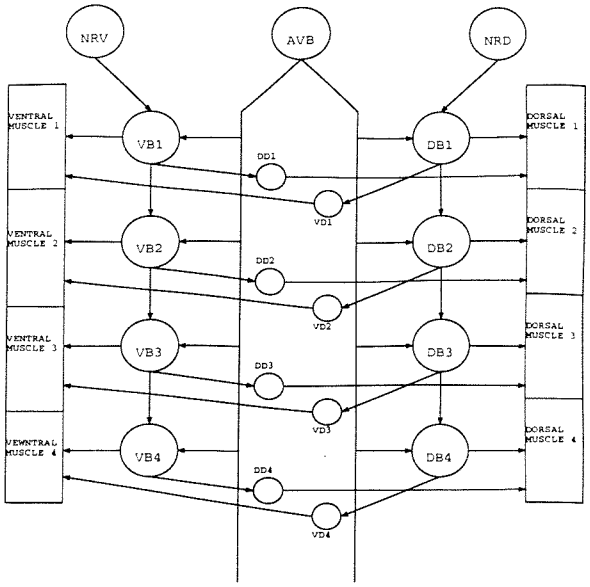


Figure 2: Locomotion circuit of C. elegans

block is asserted. The refractory period is modeled by silent periods between two pulses within which no action potential can be generated. This stage drives the output

4 Existing data about the locomotion circuitry

The nervous system of C. elegans has been mapped completely using electron microscopy [1]. In addition to this topological information, several techniques have provided insight to the functionality of specific neurons (immunochemistry allows staining cells which release a particular type of neurotransmitter, genetic studies allow identification of malfunctioning cells in mutants,and so on) [7].

Laser ablation allows the elimination of identified neurons and the study of the effect on locomotion [8].

Based on this data we propose the model presented in Fig. 2. Circles represent neurons and arrows represent synapses/electrical junctions.The top part of the diagram is closest to the head. On both sides square boxes represent body muscles. NRV and NRD stand for nerve ring ventral and dorsal excitation. AVB, DB and AVB generate and propagate contractions down the body while DD and VD inhibit antagonistic muscles.

Only a part of the forward locomotion circuit is included in the figure. The backward locomotion circuit uses a separate set of cells which is symmetrical to the forward locomotion circuit but rotated 180 degrees.

5 Mechanical model

To test the usefulness of the proposed neural circuit in the generation of forward and backward movement, it is advantageous to interface the control circuitry with the mechanical model, to allow the gross interaction of the system to be easily viewed. Our mechanical model is based on previous work on modeling the body movement of the nematode[2]. We have extended this model to three dimensions to allow further studies of the head movement (which has an extra degree of freedom when compared to the body). We have also simplified some force terms as explained below.

In summary, the model is based on an elastic cylinder made of an array of linear springs. Each point of the body mesh is connected to its four closest neighbours by a spring. The force acting on this point is the net contribution of all four springs. The resting length of the springs has

been set to force the mesh to be stable in cylindrical shape.

$$F_s = k * (d_i - d_0) * u_i \quad (2)$$

where k is the spring constant, d_i the distance to the point i in the mesh, d_0 the ideal length and u_i the unitary vector towards neighbouring point i.

To maintain its cylindrical shape, the nematode requires a high internal pressure [2]. The internal pressure term is calculated as,

$$F_p = k_p * n \quad (3)$$

where k_p is a scaling factor and n is a unitary vector normal to the surface of the body.

The action of the environment is modeled as in Eq.3 Only inertial forces are considered; viscous forces are neglected as in [2].

$$F_e = -k_r * (v * n) * n \quad (4)$$

where k_r is a scaling factor, n a unitary vector normal to the body and v is the velocity vector.

This force acts as a damping term to stop the mesh from oscillating in addition to provide the propulsion for body movement.

Finally, all types of forces acting on a point in the mesh are,

$$F_t = F_s + F_p + F_e \quad (5)$$

where F_s is the force by neighboring point, F_p the internal pressure and F_e the resistance created by the environment.

Muscle contraction is simulated by changing the ideal length of the springs in the body wall (Eq. 1). Those springs located at the position where the contracted muscle is, will see their ideal lengths reduced until relaxation.

Fig. 3 shows the resulting body shape.

The cylindrical body of the nematode is covered by muscles which are organised in longitudinal stripes . Contraction of these muscles generates the bending of the body required for locomotion, mating, and so on.

In C.Elegans there are twelve muscles per row and eight rows in the body.

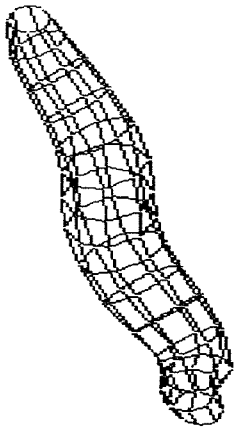


Figure 3: Screen shot of the mechanical model.

Sets of muscles are connected by electrical junctions and controlled by a single neuron. In our model, we have collapsed the body muscles into two rows (ventral and dorsal).

6 Simulation results

6.1 Normal forward/backward propagation

In Fig. 5 the results of the simulation of the forward locomotion circuit are shown. Only 4 muscles from the ventral side of the body have been included in the plot.

As mentioned before no electrophysiological experiments have been conducted so far on neurons from the locomotion system. Action potentials in mammalian neurons last for a few milliseconds. On the other hand, action potentials in pharyngeal muscle of C.Elegans has been shown to last hundreds of milliseconds and in Ascaris (a nematode similar to C. Elegans) pulse-like signals in neurons of up to a few hundreds of msec have been recorded [9].

In our model we have long pulses (action potentials) in muscles but we use pulses of a few msec in neurons.

The equivalent of the simulated circuit in the digital electronics domain would be a shift register. NRV and NRD act as inputs to the register, VB and DB behave as flip-flops making up the shift register and AVB takes the place of the

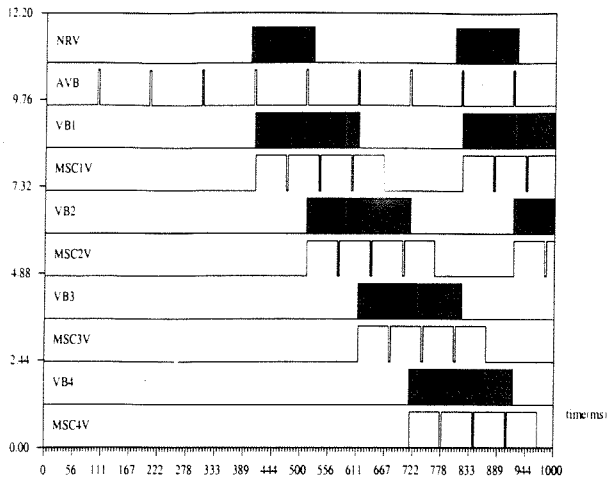


Figure 4: Simulation of forward locomotion circuit.

clock.

At t=0 all muscles in the body (labeled MSCxV) are relaxed and the animal remains still in a straight line. When it initiates forward locomotion, its muscles in the head contract forcing the bending of the tip of the body.

The head is driven by a neural circuit situated in the nerve ring which is independent from the forward/backward locomotion circuit. We have not simulated that circuit, hence, we assume that the head circuitry has forced the contraction of muscles in the head.

AVB is the inter neuron which controls the speed at which contraction propagates along the body. Each pulse generated by this inter neuron forces a displacement of the contraction pattern in the body muscles towards the tail.

Muscle cells close to the head (MSC1V and MSC2D) become active as a result of activity arriving from the AVB inter neuron and from the head (NRV and VRD). They generate a train of pulses and the muscle in the mechanical model contracts.

VB motor neurons behave as flip-flops making up the shift register. When AVB is asserted, the state of VB_n is propagated to VB_{n+1}.

The output of the VB cell activates the adjacent muscle, creating a propagating wave of contraction.

The ability of the nematode to change speed can be accounted for by changing the frequency at which AVB works. An increase in frequency forces faster propagation of the contraction, increasing the speed of the animal through the medium.

VD and DD inhibitory neurons (not shown in Fig.5) will fire whenever the motorneuron they are connected to becomes active. Their output inhibits the muscle in the opposite side of the body preventing simultaneous contraction of two muscles in opposite positions (ventral and dorsal) in the same segment.

Backward locomotion follows the same mechanism. AVA is the inter neuron responsible for speed control and VA and DA are the bypass neurons which propagate contraction along the body muscles towards the head.

DD and VD act as inhibitory neurons preventing simultaneous contraction in antagonistic muscles in the same way they acted for forward locomotion.

6.2 Defective locomotion

Laser ablation of neurons in the locomotion circuit generates nematodes with locomotion defects.

When the AVA inter neuron is laser ablated [10] , backward locomotion is never observed. AVA is also required in our model to trigger the propagation of contraction down the body. If it is forced to be silent, though the head muscles contract, no contraction wave propagates in the body and backward locomotion is impossible.

When the AVB inter neuron is ablated, the effect in both animal and model is the opposite; no forward locomotion is seen though backward locomotion is possible.

The ablation of the VB and DB neurons also perturbs normal locomotion in the nematode. If these motor neurons are forced to be silent, contraction cannot be propagated correctly and normal locomotion is impaired.

7 Conclusions and future work

The presented work is part of ongoing research studying the possibility of using event-driven models of neurons in simulations of biological neural circuits. If the behaviour of biological neural nets can be captured by a set of event driven neuron models, the simulation of aggregates of hundreds of thousands of neurons would become feasible. In the case of the locomotion system, correlate detector neurons were not enough to generate the locomotion pattern. Bypass neurons (VB, VA, DA and DB) had to be added.

It is likely that more complex functionality will have to be added to the model neuron as more complex neural circuits are simulated.

C.Elegans has been chosen as the first target system. The simulation of the locomotion circuitry is being currently extended to other neural circuits. In particular, ongoing work is focusing on thermo taxis. C.Elegans is able to steer its locomotion towards a suitable temperature. Once the ideal temperature zone has been reached, the nematode ensures it does not move out of that region.

References

[1] White J.G., Southgate E., Thomson J.N., and Brenner S. The structure of the nervous system of caenorhabditis elegans. *Philos. Trans. R. Soc. Lond. [Biol]*, 314:1–340, 1986.

[2] Niebur E. and Erdös P. Theory of the locomotion of nematodes: control of the somatic motor neurons by interneurons. *Mathematical Biosciences*, 118:51–82, 1993.

[3] T.C. Ferrée and Lockery S.R. Chemotaxis control by linear recurrent networks. *Computation and Neural Systems Meeting*, 1997.

[4] Wicks S.R., Roehrig C.J., and Rankin C.H. A dynamic network simulation of the nematode tap withdrawal circuit: predictions concerning synaptic function using behavioral criteria. *The Journal of Neuroscience*, 16(2):4017–4031, 1996.

[5] Segev I. Single neurone models: oversimple, complex and reduced. *Trends in Neurosciences*, 15:414–421, 1992.

[6] Wilson M. and Bower J.M. Cortical oscillations and temporal interactions in a computer simulation of piriform cortex. *Journal of neurophysiology*, 67(4):981–995, 1992.

[7] McIntire S.L., Jorgensen E., and Kaplan J. Horvitz H.R. The gabaergic nervous system of caenorhabditis elegans. *Nature*, 364:337–341, 1993.

[8] Rankin C.H. Interactions between two antagonistic reflexes in the nematode c. elegans. *J. Comp. Physiol.*, 169:59–67, 1991.

[9] Davis R.E. and Stretton A.O.W. The motornervous system of ascaris: electrophysiology and anatomy of the neurons and their control by neuromodulators. *Parasitology*, 113:97–117, 1996.

[10] Stephen R.W. and Rankin C.H. Integration of mechanosensory stimuli in caenorhabditis elegans. *The Journal of Neuroscience*, 13(3):2434–2444, 1995.

Discrete simulation of large aggregates of neurons

Enric T. Claverol, Andrew D. Brown, John E. Chad

Abstract

Realistic simulation of aggregates of neurons often utilises compartmental models which limit the scope of the simulations in single processor architectures to small or medium size networks (typically hundreds of neurons). An alternative approach, based on cell automata models, allows efficient simulation of nervous tissue by modelling neurons as finite state automata. In this paper, data structures and algorithms appropriate for efficient simulation of message based event driven models of neurons in single processor architectures are presented. With these techniques, the simulation of large networks (of the order of 10^5 neurons with 10^2 synapses per neuron) becomes feasible.

Keywords

Neuronal simulation, discrete simulation, pulse coded neuron models, cell automata

I. Introduction

Simulation of the nervous system is one of the techniques available to the neuroscientist to help understand the way in which neurons cooperate to process information. Realistic simulation of brain tissue often relies on compartmental models of neurons. In this context, the dynamics of a neuron are captured by a set of non-linear differential equations describing the changes in the voltage across the cellular membrane [1,2]. Each of these equations describes the voltage in a section of the cell which is assumed to be isopotential.The membrane is modelled as a leaky capacitor which draws and delivers current to nearby compartments,

$$\frac{\partial V_m}{\partial t} = \frac{1}{C_m}(\sum_{i}^{I}(V_i - V_m)g_i - \sum_{j}^{J}(V_m - E_j)g_j(t, V_m))$$

E.T.Claverol and A.D.Brown are with the Electronic Systems Design Group, Dept. of Electronics and Computer Science, University of Southampton, Southampton, UK. E-mail: E.T.Claverol@ecs.soton.ac.uk and adb@ecs.soton.ac.uk.

J.E.Chad is with the Neuroscience Research Group, School of Biological Sciences, University of Southampton. E-mail: jchad@neuro.soton.ac.uk

where V_m is the membrane voltage in the m^{th} isopotential compartment, C_m the capacitance of the membrane in that compartment, V_i the membrane voltage in one of I contiguous compartments which draw and inject current into the compartment through an internal conductance g_i and E_j and g_j are the voltage source and conductance modelling one of J ion channels which may also draw and inject current to the membrane capacitance. The value of the conductance g_j is usually a non-linear function of time, voltage or concentration of neurotransmitter and ions.

Extensive work has been done on optimization of the simulation of compartmental models (for a review see [1]). However, the simulation of compartmental models is still inherently computationally demanding and limited for this reason to networks of modest size (a few thousand neurons for models with a few compartments).

In addition to their computational complexity, compartmental models have extensive experimental requirements. Modelling ion channels following the classical Hodgkin-Huxley approach, as required for most compartmental models, involves the isolation of the different types of ion channels present in the cell (sometimes impossible) and determination of its kinetic properties (usually through voltage clamp and similar techniques).

Several more simplified models (e.g. integrate and fire) have been suggested as an alternative to compartmental modelling for large scale simulations [4-6]. In these models, the differential equations have been simplified for improved efficiency by eliminating some of the non-linear contributions to the membrane current.

Drawing from the techniques used in the simulation of discrete systems, neurons can also be modelled as complex finite state machines. Neurons are described as automata with a finite number of possible states which interact by communicating action potentials. This approach has been successfully used for the simulation of the hippocampus and has proven to be efficient enough for the simulation of tens of thousands of neurons [2].

In this paper, efficiency issues relating to the simulation of a discrete neuron model are presented. This model allows neurons with diverse types of behaviour (correlation detection, spontaneous bursting activity, single action potentials and pulse width modulation) and can be extended easily to incorporate other types of properties.

First, the discrete neuron model is described. Secondly, issues arising from the implementation of an efficient simulator for these models are discussed.

Finally, the performance of the implementation of the discrete simulator is studied, showing the suitability for simulations of networks in the order of 10^5 or greater neurons.

II. Message based event driven neuron model

Message based event driven neural simulation is a generic concept which may refer to a large number of models, all of them sharing the same principles, but with different levels of complexity. An example is briefly described for the sole purpose of clarifying the principles underlying message based event driven simulation of neurons. The same issues dealt with here are applicable to other message based event driven models. For a review of event driven simulation techniques see [8,9]. The model presented here to illustrate the basics of the message based simulator is being used for realistic simulation of circuits of neurons [3].

The message based event driven neuron model is a finite state automaton. It is made up of several blocks, each of them capturing the functionality of a different component of the neuron (see Fig. 1).

Communication between neurons and blocks within a single neuron is achieved by message passing. Each message is a data packet containing the time at which the message will be delivered to its destination (expressed as the difference between delivery time and current time), a label field indicating the type of message and a third optional field with an extra parameter used by the target to process the message. Arrows with solid lines in Fig. 1 indicate message paths. Note that some solid arrows are originated and terminate in the same block.

A change of state in a block is always triggered by the arrival of a message. After a change of state, new messages may be scheduled for broadcasting to other blocks or to the same block.

Table I and Table II list the parameters which characterise the model and the channels for message broadcasting respectively. Tables III,IV,V and VI show the state transition tables and combinational functions implemented by the blocks in the neuron model. When a message is delivered to a block, the automaton will be in one of a finite number of states and may change to a new state. This change of state may be accompanied by the scheduling of a new message

Parameter	Function
th_e	Excitation threshold
th_i	Inhibition threshold
t_{ap}	Duration of action potential
t_{ref}	Duration of refractory period
N_{burst}	Number of spikes per burst
t_{osc}	Period of pace maker
t_{ϕ}	Time offset of pace maker
t_{del}	Synaptic delay
t_{dur}	Duration of synaptic pulse
w_{syn}	Synaptic efficacy

TABLE I

PARAMETERS USED IN THE MODEL

(an output) and the update of state variables in the block (an action). For purely combinational functions (e.g. the synapse block) the output is only a function of its input.

A. The synapse block

Synapses receiving the *on* message at t, become activated and, after introducing a synaptic delay, deliver an *on* message to the threshold block (at $t + t_{del}$). At $t + t_{del} + t_{dur}$, the synapse inactivates and sends an off message to the threshold block. Synapses are combinational functions which schedule new messages depending on the last message received (they do not need memory of their current state).

In real neurons the consequence of an action potential is the release of neurotransmitter after a certain delay. The release of neurotransmitter affects postsynaptic neurons by increasing (excitatory) or decreasing (inhibitory) its membrane potential. In the message based model, the neurotransmitter is substituted by a message and its release by the broadcasting of the message to the target neuron.

Channel	Message structure	Legal values of m	Legal values of p
α	{t,m,p}	*on*	efficacy(w_{syn})
β	{t,m}	*on*,*off*	
γ	{t,m,p}	*on*,*off*	efficacy(w_{syn})
δ	{t,m}	*change*	
ϵ	{t,m}	*on*,*off*	
ζ	{t,m}	*on*	
η	{t,m}	*off*, *r_off*	

TABLE II

MESSAGE CHANNELS IN THE NEURON MODEL

B. The threshold block

This block is responsible for calculating a weighted sum of the active synapses (the weight being their efficacies, w_{syn}). Whenever the weighted sum goes above the excitation threshold (th_e), an *on* message is sent to the burst generator block which generates a burst of action potentials. If the weighted sum goes below the inhibition threshold (th_i), the threshold block sends an *off* message to the burst block to stop the ongoing burst.

The update of the weighted sum of inputs and its comparison with the excitation and inhibition thresholds is triggered by the reception of *on* and *off* messages from synapses.

C. The oscillator block

An oscillator block has been added to the model which sends *on* messages to the burst generator block every t_{osc} time units starting at $t = t_\phi$. This block simulates rhythmic activity in neurons.

D. The burst generator block

The burst generator block generates a burst upon reception of an *on* message. The arrival of the *on* message triggers the start of a cycle of state changes. The sequence starts with a change from state *off* to state *on* (onset of the first action potential). After t_{ap} time units, the state

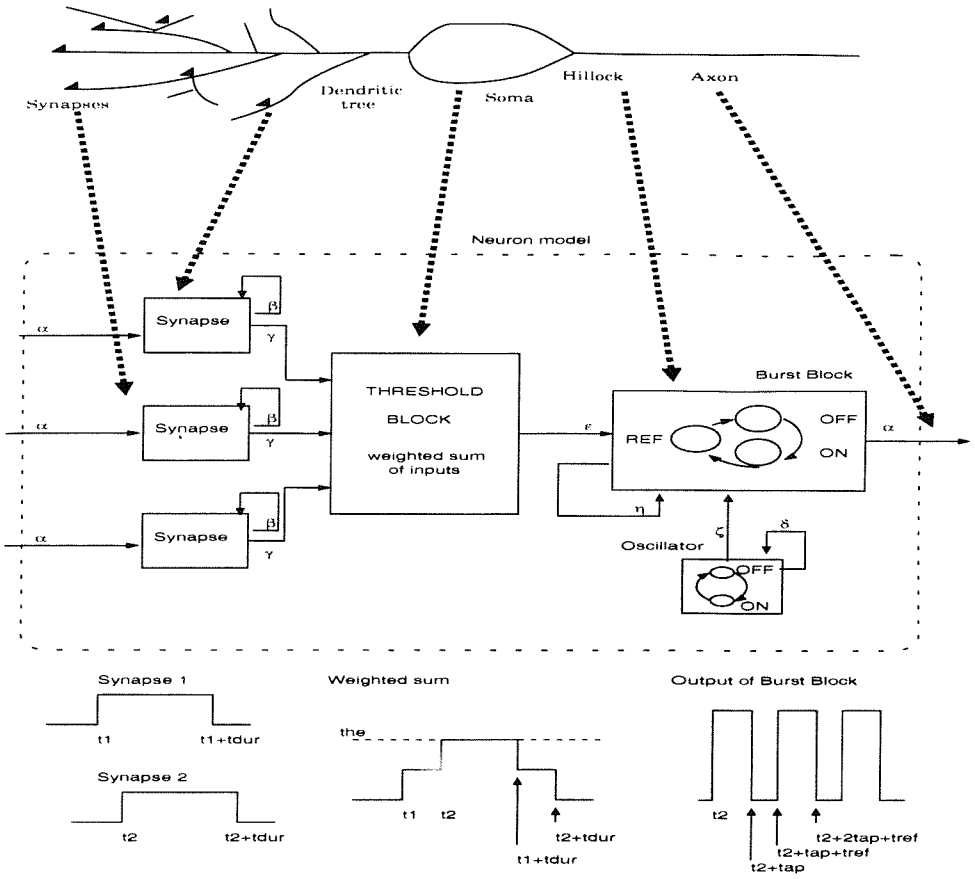


Fig. 1. Message based event driven neuron model. Solid arrows indicate origins and destinations of messages. Thick dashed arrows indicate the correspondence between parts of the real neuron and blocks in the model.

changes from *on* to ref (beginning of the refractory period) and, after t_{ref} time units, back to *on* (start of the second action potential in the sequence). This cycle is repeated N_{burst} times (making up a burst of N_{burst} action potentials). An *on* message is broadcasted to all synapses driven by the burst block when its state changes from off to *on* in order to communicate neurotransmitter release.

III. Implementation of the event driven simulator

A. Overview

The simulator has been implemented as an extension of the commands provided by the scripting language of a standard numerical package (see Fig. 2 for an overview of the system developed). This approach provides flexibility in the storage of topology, data analysis, automation of parameter space search and portability across platforms.

Input	Output
$\alpha := on$	$\beta := \{t_{del}, on\}$
$\beta := on$	$\beta := \{t_{dur}, off\}$, $\gamma := \{0, on, w_{syn}\}$
$\beta := off$	$\gamma := \{0, off, -w_{syn}\}$

TABLE III

THE SYNAPSE BLOCK FUNCTION

Input	Action | Output
$\gamma := on$	$w_{sum} += w_{syn}$ $w_{sum} >= th_e$? true: - | $\epsilon := \{0, on\}$ $w_{sum} <= th_i$? true: - | $\epsilon := \{0, off\}$
$\gamma := off$	$w_{sum} -= w_{syn}$ $w_{sum} >= th_e$? true: - | $\epsilon := \{0, on\}$ $w_{sum} <= th_i$? true: - | $\epsilon := \{0, off\}$

TABLE IV

THE THRESHOLD BLOCK STATE MACHINE

Current state	Next state | Output
	$\delta := \{t_{osc}, change\}$
on	off | $\delta := \{t_{osc}, change\}$,$\zeta := \{0, on\}$
off	on | $\delta := \{t_{osc}, change\}$,$\zeta := \{0, on\}$

TABLE V

THE OSCILLATOR STATE MACHINE

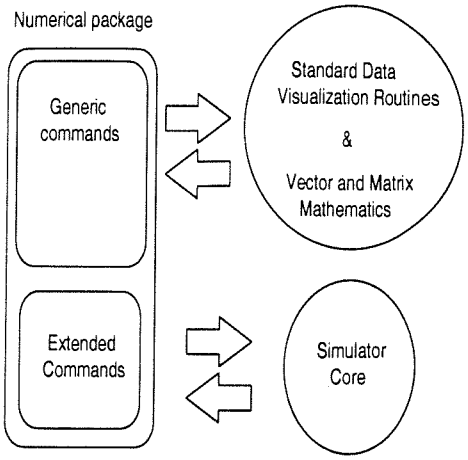


Fig. 2. Overview of the simulation tool

When aiming at large scale simulations of the nervous system, the simulation tool must provide an efficient way of storing the topology and parameters of the network. Storage of individual connections and neurons in files is not efficient (files in the order of hundreds of Megabytes would be needed for networks of 10^5 neurons and 10^7 synapses).

The simulator described here offers two alternatives. A program written using the scripting language of the numerical package creates a vector with each entry declaring a synapse. A new command, part of the language extension, accepts the vector of connections as a parameter and creates the data structures required for the simulation. With this approach, only the code needed to create the vector of synapses (and not the synapses themselves) has to be stored.

A second alternative is to provide special purpose higher level commands which do not accept vectors of synapses but parameters which characterize the connectivity rules of the topology. As an example, for the randomly connected network of Section IV, the parameters would include the number of neurons, number of connections per neuron and the probability of establishing an excitatory or an inhibitory connection. Although this approach offers an efficient use of memory (no intermediate lists of synapses must be created) it requires the implementation of a new command for each family of topologies.

The integration of the simulator within a standard numerical package also simplifies the analysis of the data. The standard tools available in the package can be used with the results of the simulation

(e.g. the Fourier transform is implemented in most numerical packages and allows frequency domain analysis of EEG simulations). The results of interest are typically parameters which summarize properties of the network dynamics (see Fig. 6-A,B,C) for comparison with experimental bulk measurements like EEG and external field potentials. The knowledge of the state of individual cells is also important to understand the bulk measurements. Cell states are plotted in matrix form and used to create mpeg movies with the activity of the network (see Fig. 6-D).

The integration of the simulator within a standard numerical package makes also possible the automation of some of the tasks associated with the simulations. For example, when exploring the dynamics of a network, a search of a region of the parameter space is likely to be needed. The possibility of using a scripting language to control the parameter search allows fast implementation of different search algorithms. Finally, relying on the numerical package for input/output increases the portability of the simulator. As many packages have been ported to several operating systems, the programmer does not need to recode (e.g. the plotting routines) for cross-platform portability.

B. Data structures

The data structures which store topology, parameters and state variables of neurons and synapses are shown in Fig. 3 (see [4] for a comparison of data structures for the simulation of neural networks).

Upon initialization, the simulator estimates and allocates the total amount of memory required for the storage of all neurons and synapses. Dynamic allocation of individual neurons and synapses must be avoided to reduce the overhead associated with dynamic allocation.

The data structure for a neuron contains the following fields: a neuron id, the number of synapses from this neuron onto other neurons, the state vector, the state variables, parameters and a list of synapses.

The number of synapses in realistic simulations will be at least two orders of magnitude higher than the number of neurons. Hence, the minimization of the memory allocated for each synapse is important.

Each synapse is characterized by its parameters (t_{del},t_{dur},w_{syn}) and the identifier of its target neuron. The parameters are not stored for each synapse. Instead, only an index into a table of

Burst generator					
Current	Next state | Action | Output				
state	Input				
	$\epsilon := on$	$\epsilon := off$	$\eta := off$	$\eta := r\_off$	$\zeta := on$
on	on | - | -	on | $n_{burst} = 0$ | -	ref | - | $\eta := \{t_{ref}, r\_off\}$	on | - | -	on | - | -
ref	ref | - | -	ref | $n_{burst} = 0$ | -	ref | - | -	$n_{burst} - 1 == 0$? true: off | $n_{burst} = N_{burst}$ | - false: on | $n_{burst} - = 1$ | $\eta := \{t_{ap}, off\}$	ref | - | -
off	on | - | $\alpha := \{0, on\}$, $\eta := \{t_{ap}, off\}$	off | - | -	off | - | -	off | - | -	on | - | $\alpha := \{0, on\}$, $\eta := \{t_{ap}, off\}$

TABLE VI

THE BURST GENERATOR STATE MACHINE

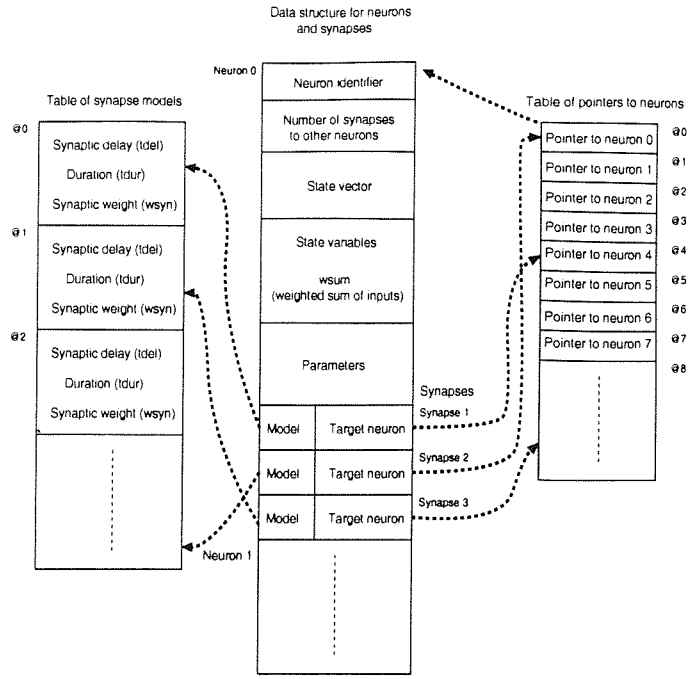


Fig. 3. Data structures used for storage of neurons and synapses.

types of synapses (seen on the lefthand side of Fig. 3) is stored.

Entries in this table are model structures which contain the parameters for one of the allowed types of synapses. For our implementation, the table of synaptic models may have up to 64 entries, allowing 64 types of synapses.

In addition to its parameters, each synapse structure needs to identify the target neuron. Our simulator, running on a machine with a 32 bits wide address bus, would need a 4 bytes word to identify the target neuron if a pointer was to be stored in each synapse. Instead, an index into a table of pointers is stored.

Given an addressable memory space of 2^{32} bytes, it is unlikely that a simulation of 2^{24} neurons (16 million) or more would fit in the available memory. Hence, neurons are labeled with a 24 bit identifier which is used as an index into the table of pointers to neurons (righthand side of Fig. 3) to locate the neuron in memory.

With these two strategies (a table of models and a table of pointers to neurons), the parameters needed for a synapse can be masked into a single 32 bits word (24 bits for the target neuron identifier, 6 bits for the synapse type and 2 bits unused). Memory consumption for the storage of

synapses is reduced in this way.

C. Priority queue

Neurons communicate by message broadcasting. As new messages are generated, those that are scheduled for delayed delivery to their destinations are inserted in a time-sorted queue. The main loop of the simulator is responsible for extracting the messages and for their delivery to their target devices at the appropriate time.

Two main issues have to be considered when implementing the priority queue. Efficiency in terms of CPU time required for insertion/extraction of new events into the queue and memory consumption.

C.1 Efficient insertion of new messages

The insertion of messages in the queue is usually the most costly operation in event driven simulation, as messages have to be sorted by time of delivery. Several algorithms have been suggested for queue management (for a review see [12,13]). In most cases, the insertion time is affected by the number of messages in the queue. Calendar queues deserve special attention as this approach offers insertion latencies independent of the size of the queue (O(1))[5].

Neuronal activity consists of action potentials of, at least, $1 - 2\ ms$ of duration. For discrete simulation of a network of neurons, time can be represented as a multiple of a basic time step of $100\mu s$ without compromising the usefulness of the simulation. Given this coarse granularity of time, a priority queue based on an LUT (look up table) for fast insertion can be used (see Fig. 4-A).

The priority queue is made up of a set of linked lists of messages. Each list containing all the messages which have been scheduled for the same time in the future. An LUT stores pointers to the first message in each sublist. For an LUT with 10^6 entries, a maximum of 10^6 lists can be indexed. The first list links all messages scheduled for $t = 0$, and the last links all messages for $t = 10^6 - 1$. With a time step of $100\mu s$, messages could be scheduled no further into the future than 100 s. The total amount of memory required for the storage of this table, when using 32 bit pointers, would be approximately 4 Mbytes.

Pages 252-253
missing from
this Thesis

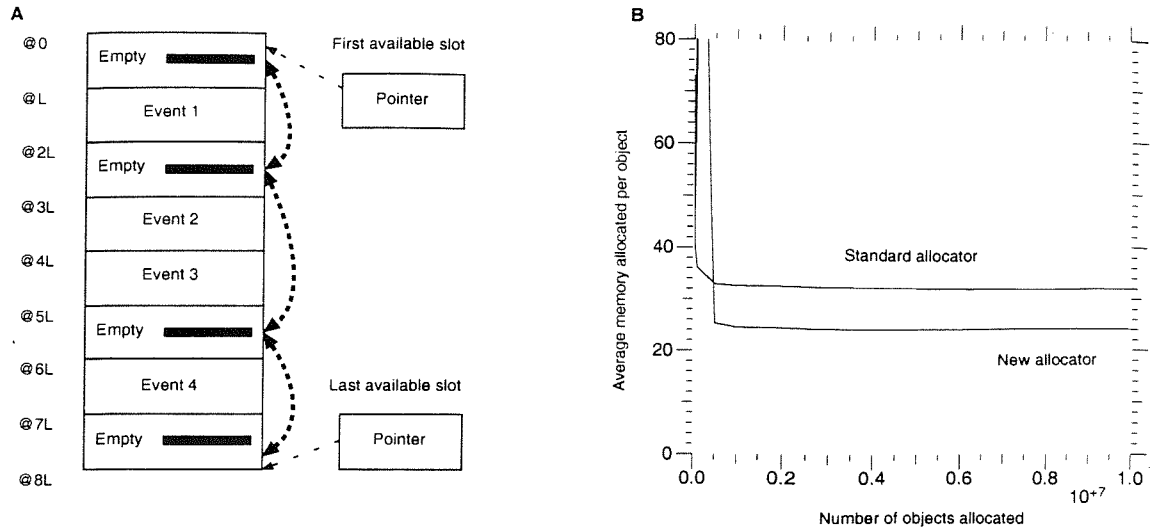


Fig. 5. Efficient memory allocation is possible for priority queues due to the constant size of message structures.A - Memory allocation algorithm, B - Comparison of standard and new allocators.

and there is no extra computation involved in finding a free slot of the appropriate size. Fig. 5-B shows the average memory required per object (overhead+data) as a function of the number of dynamically allocated objects (each one consisting of a 24 bytes long data structure). There is no overhead with the new allocator whereas, with the standard allocator, there is an overhead of 8 bytes per object.

D. Other implementation issues

It is important to note that the high number of connections per neuron may slow down the update of the weighted sum of inputs (w_{sum}) in the threshold block. Each update of w_{sum} requires the computation of,

$$w_{sum_j} = \sum_{i}^{S} \alpha_i w_{syn_i} \tag{2}$$

$$\alpha_i = \left\{ \begin{array}{ll} 1 & if\ synapse\ i\ is\ active \\ 0 & if\ synapse\ i\ is\ inactive \end{array} \right\} \tag{3}$$

where S is the number of synapses providing input to neuron j, α_i is 1 if synapse i has been activated and 0 if it remains inactive and w_{syn_i} is the synaptic weight of synapse i.

w_{sum} must be updated each time one of the synapses becomes active or inactive (the threshold block is notified by the arrival of an *on* or off messages). Complete recalculation of w_{sum} requires the weighted addition of S synaptic weights and, for a typical neuron, the number of synapses (S) is in the range $10^2 - 10^4$. However, this is not needed if w_{sum} is updated as,

$$w^+_{sum} = w^-_{sum} + \sum_i^s \alpha_i w_{syn_i} \tag{4}$$

where w^-_{sum} and w^+_{sum} are the weighted sums before and after an update respectively, s is the number of synapses which changed state simultaneously (typically $s << S$), α_i is 1 if synapse i has been activated and -1 if it has been inactivated and w_{syn_i} its weight. This requires the storage of the weighted sum as a state variable for each neuron but speeds up state recalculation of w_{sum} as the number of synapses which change state (s) is considerably lower than the total number of synapses S.

Regarding the overall implementation of the neuron, it is convenient to follow an object oriented approach. The neuron object offers an interface which allows the simulation engine to initialize the automaton at the beginning of the simulation and to deliver the messages to be processed.

This object oriented approach reduces programming time whenever the behaviour of the finite state machine has to be modified. New device objects can be created without forcing any modification in the rest of the code as long as the interface with the simulator does not change.

The cost of modifying the functionality of the model and the simulator is an important factor to consider. Simulation of the nervous system with cell automata is still in its infancy. It will often be necessary to modify the model to incorporate new types of behaviours. An object oriented implementation considerably reduces the time involved in these changes.

IV. Performance of the simulator

To study the performance of the simulator, a network of 5 10^4 neurons with random connections has been simulated.

Each neuron has exactly C synapses with postsynaptic neurons chosen at random. Two types of synapses are included in the network; excitatory synapses with synaptic delay $t_{del} = 5\ ms$, efficacy $w_{syn} = 1$ and duration of activation $t_{dur} = 10\ ms$; inhibitory synapses with $t_{del} = 5\ ms$,

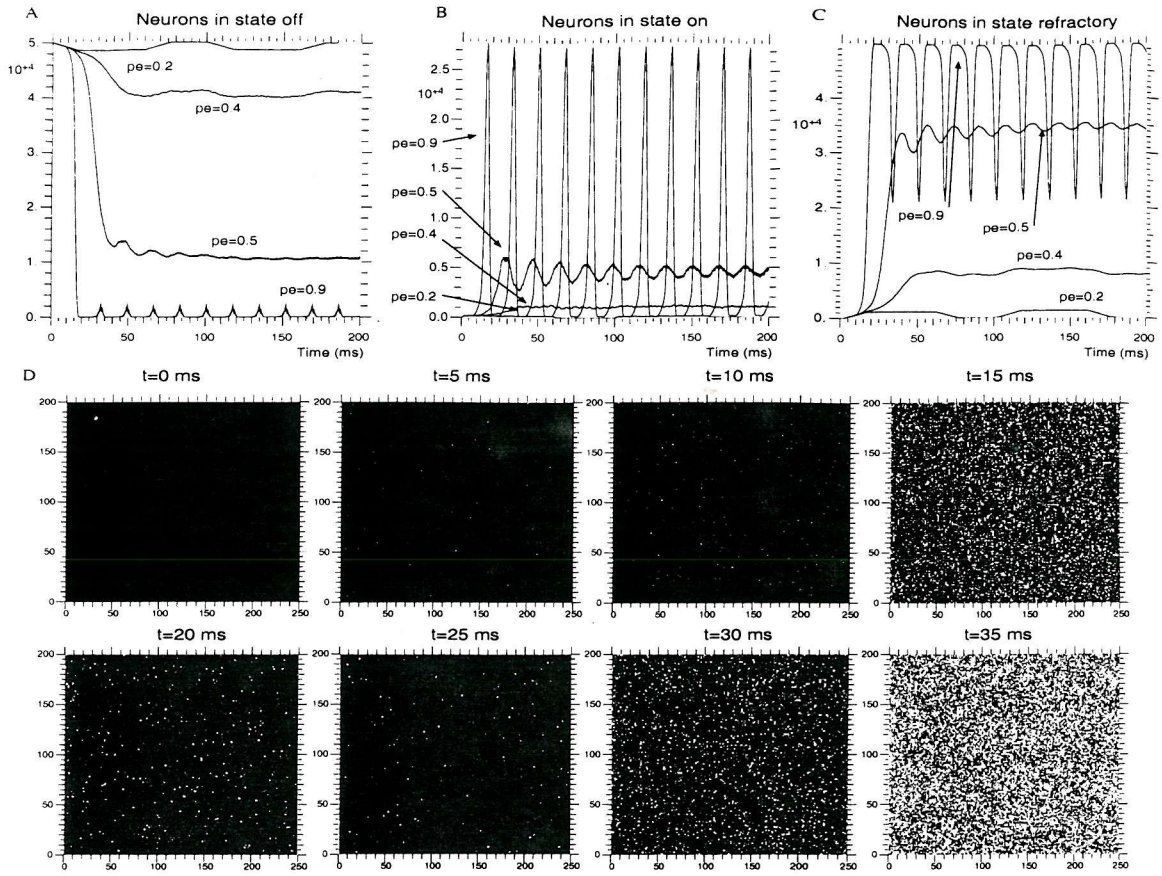


Fig. 6. A,B,C - Total number of neurons in state *off* (A), *on* (B) and *refractory* (C), D - Time sequence of the neuron states for the network displaying epileptic-like activity ($p_e = 0.9$). Matrix of 200x250 neurons. Those in state *on*, *off* and *refractory* are represented by white, black and gray dots respectively.

$w_{syn} = -1$ and $t_{dur} = 10\ ms$. The type of each synapse is chosen at random with probability p_e of being excitatory and $1 - p_e$ of being inhibitory. Multiple synapses of the same type from a given neuron to a target neuron are allowed (equivalent to a single synapse of increased efficacy).

5 10^3 out of the 5 10^4 neurons have been configured as pace makers which fire a single action potential (2 ms duration and 10 ms absolute refractory period) every 100 ms ($t_{osc} = 100\ ms$) with a time shift of $t_\phi\ ms$, where t_ϕ is a random variable given by a uniform distribution in the range (0..60 ms).

All neurons behave as correlation detectors, firing an action potential (also 2 ms duration and 10 ms refractory period) when the weighted sum of instantaneous synaptic inputs goes above the excitation threshold.

Several values for the percentage of inhibitory synapses, total number of synapses per neuron

Page 257
missing from
this Thesis

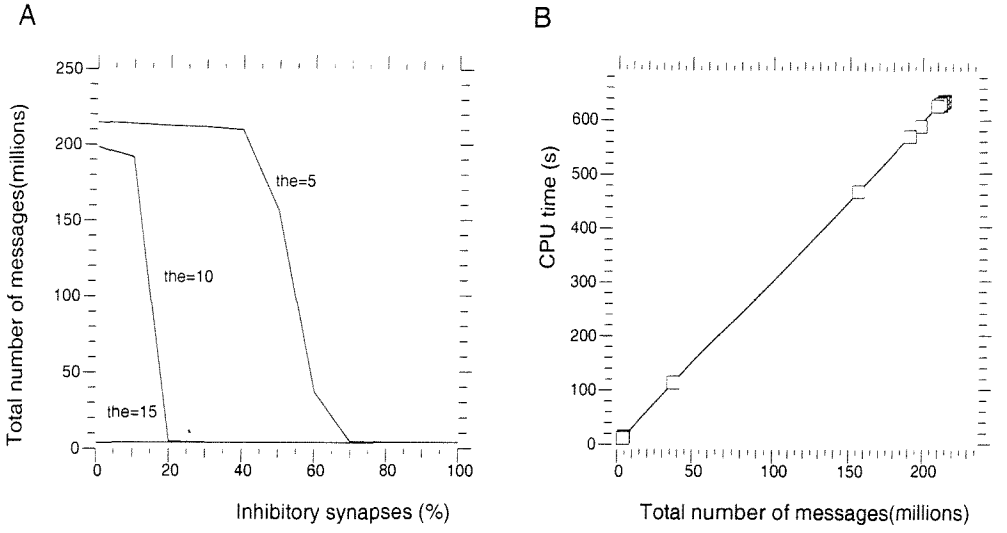


Fig. 7. A - Total number of messages versus percentage of inhibitory synapses and excitation threshold (number of synapses per neuron set to 200), B - Simulation time versus total number of messages processed

(total number of connections per neuron).

In Fig. 7-A the total number of messages processed during the simulations is shown as a function of p_e and th_e. The number of synapses per neuron has been set to 200. It can be seen in the plot that, as p_e is increased, there is a transition from sparse activity into generalized firing made evident by the increase in the total number of messages generated. Fig. 7-B shows the CPU time as a function of the total number of messages processed.

Fig. 8-A shows the total number of messages as a function of the number of synapses per neuron and the value of the excitation threshold. The percentage of inhibitory synapses has been set to 10% ($p_e = 0.9$). In Fig. 8-B the simulation time is plotted as a function of the total number of messages processed.

Note that in Figs. 7-A and 8-A, for $the = 15$, the network activity remains sparse for all tested values of p_e and C. However, for $the = 5$ and $the = 10$ the number of messages generated shows an abrupt increase indicating the switch of the network dynamics from sparse activation into generalized firing. Generalized activity in the network decreases the performance of the simulator by increasing the total number of messages to process.

As seen in Figs. 7-B and 8-B, the simulation time depends linearly on the total number of messages generated and processed during the simulation. This is because the two main tasks of

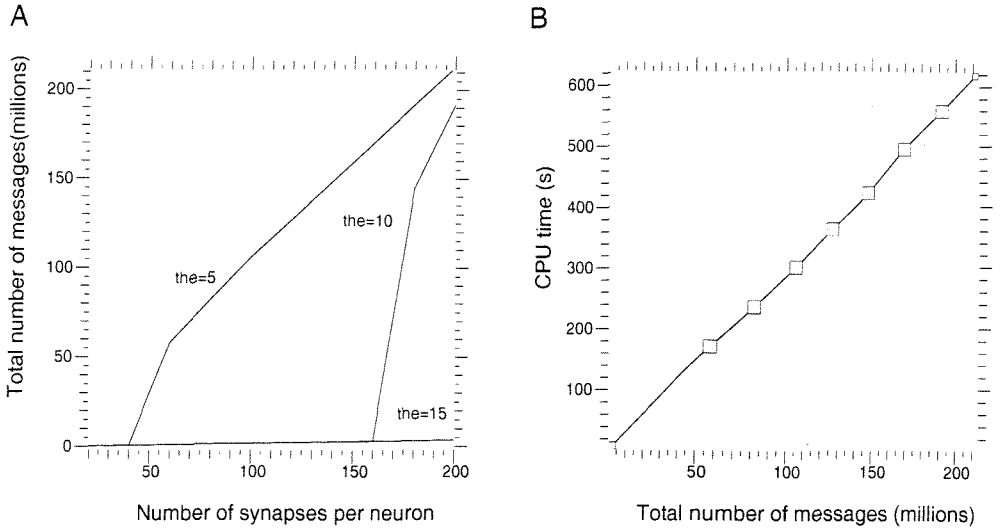


Fig. 8. A - Total number of messages versus number of synapses per neuron and excitation threshold (percentage of inhibitory synapses set to 10%), B - Simulation time versus total number of messages processed

the simulation loop are the insertion of new messages into the priority queue and their extraction and processing. The simulation time in seconds is given by,

$$t = 2.9\ 10^{-6} e \tag{5}$$

where e is the total number of messages. Each message requires 2.9 μs for its processing.

As it is useful to estimate the resources that will be required by a simulation, it is desirable to be able to predict the total number of messages which will be generated. However,this is difficult to anticipate as it depends not only on the topology of the network (known beforehand) but also on the activity which will be known only after simulating.

Considering a worst case scenario, all neurons could fire simultaneously at their maximum firing rate during the entire simulation. In this case, the total number of messages processed is given by,

$$e = E_{syn} + E_n \tag{6}$$

where E_{syn} is the total number of messages generated by synapses and E_n the number of messages generated by the rest of blocks in the neuron model. As the number of synapses is several orders of magnitude bigger than the number of neurons, the total number of messages

processed can be approximated by,

$$e \approx E_{syn} \approx \frac{2NC}{t_{ref}} t_{simu} \tag{7}$$

where N is the total number of neurons, C the average number of connections per neuron, t_{ref} the neuronal refractory period and t_{simu} the time of simulation. The factor 2 accounts for the two messages (activation and inactivation) inserted in the queue by a synapse.

In a typical simulation the average firing rate of a neuron is expected to be far from the maximum rate attainable. In this more realistic situation, expression 7 has to include a correction term β,

$$e = \beta \frac{2NC}{t_{del}} t_{simu} \tag{8}$$

where β is the normalized average firing rate.

For the lowest values of p_e in Fig. 7-B , the total number of messages would be approximated by Eq. 8 with $\beta = 0.005$ whereas for high values of p_e a good match is achieved for $\beta = 0.5$.

Finally, note that if the generation and processing of events is evenly distributed throughout time (see lower traces in Fig. 9-A) the CPU time per time step will be also evenly distributed throughout the simulation. However, in contrast with non-event driven simulation, when oscillations occur in the total number of neurons firing in the network (see upper traces in Fig. 9-A), the peaks in the oscillations (large number of messages being broadcasted) will concentrate most of the CPU time.

C. Memory requirements

Two factors have to be considered regarding memory consumption: topology/parameters and the priority queue. The amount of memory required to store topology and parameters can be estimated by,

$$M = NCS + NP \tag{9}$$

where N is the number of neurons, C the number of connections per neuron and S and P the space allocated for parameters and state variables for a single synapse and neuron respectively.

For the simulations of Figs. 6, 7 and 8, the total memory allocated for topology, parameters of the models, LUT of the priority queue and numerical package was $53.5Mb$ ($N = 5\,10^4$ neurons, $C = 200$ synapses, $S = 4$ bytes, $P = 52$ bytes).

In the case of simulations where the topology does not change online, the only uncertainty in the memory consumption lies in the size of the priority queue.

The instantaneous number of events in the queue, and the memory allocated to store them, depends on the number of neurons and synapses simultaneously active. Fig. 9-A shows superimposed traces with the instantaneous number of messages present in the queue during several simulations of the randomly connected network. Upper traces correspond to values of p_e close to 1 whereas lower traces correspond to values close to 0.

Synchronization of neuronal firing of large ensembles of neurons in the network causes oscillations in the size of the priority queue (as seen in Fig. 9-A for $p_e = 0.9$). These peaks in the number of neurons firing produce an accumulation of messages in the queue and the resulting increase of memory allocated to store it. Enough memory has to be available in order to store the queue at any time during the simulation and avoid swapping, as this would have a negative impact on the performance of the simulator.

As the maximum size of the queue during a simulation is the limiting factor, Fig. 9-B shows the maximum number of messages found in the queue during the simulations shown in Fig. 7. When the percentage of inhibitory synapses is small, the activity generated by the pace makers propagates in the network activating most neurons and flooding the event queue. As the percentage of inhibitory synapses increases, the network becomes only sparsely active and the maximum number of messages in the queue during a simulation decreases dramatically. This reduces the memory resources needed for the simulator.

For a maximum size of the queue of 10^7 messages and messages of size 12 bytes, 120 Mb were allocated for the queue.

V. Conclusion

In this paper, a tool is presented for the simulation of event driven models of neurons which are being explored as alternatives to more detailed but less efficient compartmental models.

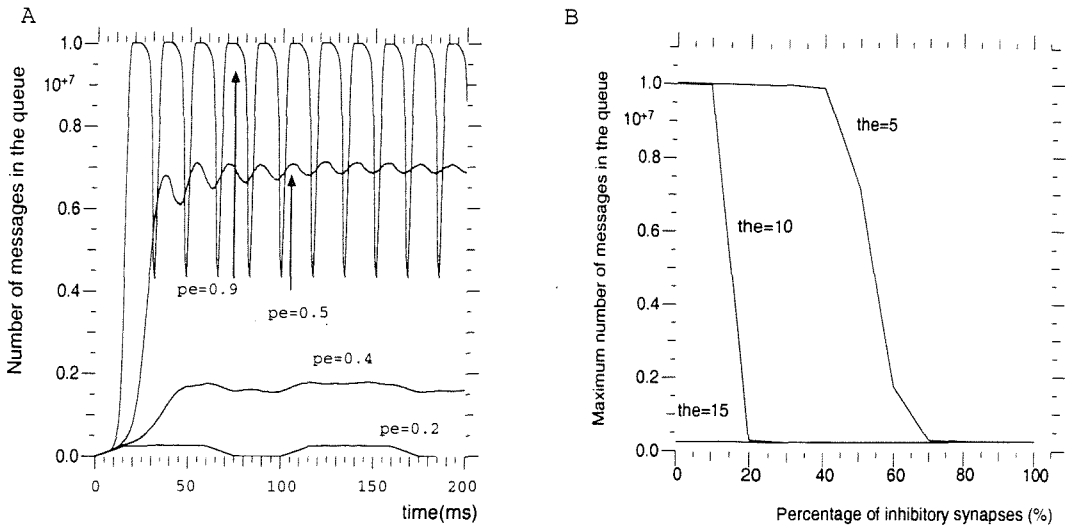


Fig. 9. A - Instantaneous queue occupancy for increasing values of the percentage of inhibitory synapses. B - Maximum queue occupancy as a function of the percentage of inhibitory synapses and the excitation threshold (200 synapses per neuron).

Neuronal models based on finite state automata provide a framework where the functionality of a real neuron can be captured while avoiding the modelling of molecular details.

Several techniques for efficient simulation of these models have been shown. The use of the LUT based priority queue allows O(1) queue insertion times, minimizing CPU time for message processing. Model structures, an LUT for device indexing and the use of an optimized algorithm for dynamic memory allocation of new messages reduce memory consumption.

Changes between different modes of operation in a network of randomly connected neurons, qualitatively similar to those observed in EEG recordings, have been triggered by changes in the total number of inhibitory synapses.

Simulations of hundreds of thousands of discrete neurons on a desktop computer are feasible with this approach.

Acknowledgements : We thank our colleagues in the ESDG group for helpful discussions. This work has been funded by the BBSRC and the Dept. of Electronics and Computer Science of the University of Southampton, UK.

REFERENCES

[1] Hammarlund P. and Ekeberg O. Large neural network simulations on multiple hardware platforms. *Journal of Computational Neuroscience*, 5:443–459, 1998.

[2] Pytte E. and Grinstein G. Traub R.D. Cellular automaton models of the ca3 region of the hippocampus. *Network: computation in neural systems*, 2(2):149–167, 1991.

[3] Claverol E.T., Cannon R.C., Chad J.E., and Brown A.D. Event based neuron models for biological simulation. a model of the locomotion circuitry of the nematode c. elegans. In Mastorakis N.E., editor, *Computational intelligence and applications*, pages 217–222. World Scientific Engineering Society Press, 1999.

[4] Freeman J.A. and Skapura D.M., editors. *Neural networks : Algorithms, applications and programming techniques*. Addison-Wesley, 1992.

[5] Brown R. Calendar queues: A fast o(1) priority queue implementation for the simulation event set problem. *Communications of the ACM*, 31(10):1220–1227, 1988.

[6] Gontmakher S. and Horn I. Efficient memory allocation. *Dr. Dobb's Journal*, pages 116–119, January 1990.

LIST OF FIGURES

LIST OF TABLES

A Large Scale Simulation of the Piriform Cortex by a Cell Automaton-Based Network Model

Enric T. Claverol, Student Member, IEEE, Andrew D. Brown, Senior Member, IEEE, John E. Chad

Abstract

An event-driven framework is used to construct a physiologically motivated large scale model of the piriform cortex containing in the order of 10^5 neurons. This approach is based on a hierarchically defined neuron model consisting of finite state machines. It provides computational efficiency while incorporating components which have identifiable counterparts in the neurophysiological domain. The network model incorporates four neuron types and their main electrophysiological features.

The spatio-temporal patterns of cortical activity and the temporal and spectral characteristics of simulated EEGs are studied. In line with previous experimental and compartmental work, 1) shock stimuli elicit EEG profiles with either isolated peaks or damped oscillations, the response type being determined by the intensity of the stimuli, and 2) temporally unpatterned input generates EEG oscillations supported by model-wide waves of excitation.

Keywords

Piriform olfactory cortex, discrete simulation, pulse coded neuron model, cell automata, EEG oscillations

I. Introduction

The simulation of the mechanisms implicated in information processing in the nervous system is an area of active research [1], [2], [3], [4], [5], [6], [7]. It provides a tool for the understanding of brain functions which are difficult to study experimentally due to the large number of cells involved and the difficulties arising from the execution of *in-vivo* experiments.

The techniques used for the simulation of large aggregates of neurons can be grouped into two categories: biophysically detailed models [1] and artificial neural networks [2].

2

Biophysically detailed models are based on cable theory applied to dendrites and axons and make use of ion channel models which are usually described using the Hodgkin-Huxley formalism [3]. In this context, neurons are described by systems of nonlinear differential equations which must be solved numerically. Two undesirable properties of this approach are the computational cost of numerical integration and the amount of experimental data required to set the parameters in the model. As a result of these limitations, the simulation of large aggregates of neurons (more than 10^4) is unfeasible or requires parallel architectures [4]. Artificial neural networks, in general, do not allow direct mapping of biophysical parameters into model parameters and are considered unrealistic. However, they provide a computationally efficient alternative to biophysical models.

During the past decade, several software tools have been developed for the realistic simulation of single cells and small aggregates of neurons (e.g. GENESIS [5], NEURON [6]). There is ongoing research to develop simulators capable of handling large networks [4] by means of parallel architectures.

We have chosen an alternative approach, based on the adaptation of event-driven simulation techniques to the problem of neural simulation These allow direct use of biophysical parameters while permitting large scale simulations with the available computing resources.

Drawing from the methods used in discrete simulation, neurons can be modelled as complex finite state machines [7]. By describing the automaton as a hierarchical structure where each component has a counterpart in the biological neuron, biophysical parameters can be introduced in the model. By introducing the concept of an event-driven neuron, the efficiency inherent to discrete simulation is retained [8].

The availability of both experimental data [9] [10] and simulations based on biophysically detailed models [11] [12], makes the piriform cortex an ideal cortical module to validate this approach. The piriform cortex is thought to be involved in smell recognition [13]. It receives input from the olfactory bulb, which performs the first stages in smell identification [14], through the lateral olfactory track (LOT). After carrying out certain computations on the input data (the nature of which is still unclear) it relays the results to higher level cortical modules. Previous simulations

have been confined to networks of 4500 neurons in [11] and 292 neurons in [12], far from the approximately 10^7 neurons found in the piriform cortex.

A cortical model including 10^5 discrete neurons of four types (fast glutamate excitatory, fast $GABA_A$ inhibitory, slow $GABA_B$ inhibitory and LOT) is presented in this paper.

The message-based event-driven neuron model will be described first. Secondly, the piriform cortex model and the calculations involved in the estimation of field potentials and EEGs will be discussed. Thirdly, the responses of the cortical model to shock stimulus and random input will be studied and shown to share the main characteristics of experimental data and results obtained with compartmental models.

Issues regarding the implementation of an efficient simulator for this type of models and networks in the order of 10^5 neurons are discussed elsewhere [15].

II. Methods

A. Message-based event-driven neuron model

The message-based event-driven neuron model is a hierarchically defined finite state automaton [16]. It is made up of several blocks, each of them capturing the functionality of a different component of the neuron (see Fig. 1).

[Figure 1 about here.]

Message passing is the method used for communication between neurons and between blocks within a single neuron. Each message is a data packet containing the time at which the message will be delivered to its destination, a label field indicating the type of message and a third optional field with extra information used by the target neuron to process the message (see Table I). Arrows with solid lines in Fig. 1 indicate message paths.

The delivery of a message to a block, triggers the update of its state, which may be accompanied by the broadcasting of new messages (an output) and the update of state variables in the block (an action). For purely combinational functions (e.g. the synapse block) the output is only a function of the input.

Table II lists the parameters required for the configuration of the model and Appendix A contains the state transition tables and combinational functions implemented by the blocks in the neuron model.

A.1 The synapse block

Synapses receiving the *on* message at time t, which notifies of the firing of a presynaptic neuron, introduce a synaptic delay and become activated at $t + t_{del}$. An *on* message is then broadcasted to the threshold block.

At $t + t_{del} + t_{dur}$, the synapse inactivates, having remained activated for t_{dur} time units, and sends an off message to the threshold block.

Synapses are combinational functions which schedule new messages depending on the last message received (they do not need memory of their current state).

[Table 1 about here.]

[Table 2 about here.]

A.2 The threshold block

The threshold block computes a weighted sum of inputs (w_{sum}) where the weights are the synaptic efficacies (w_{syn}). The arrival of *on* and off messages from synapses, triggers the update of w_{sum}. After an update, its value is compared against the excitation (th_e) and inhibition (th_i) thresholds. An *on* message is sent to the burst generator block (which generates a burst of action potentials) if w_{sum} increases beyond th_e. Conversely, if the weighted sum becomes more negative than the inhibition threshold (th_i), the threshold block sends an off message to the burst block to stop an ongoing burst. Note that, since neurons in the cortical model presented in this paper were configured to fire single spikes, burst truncation does not apply. Thus, in order to avoid the unnecessary generation of off messages, th_i was set to -1000, a value never reached by w_{sum}.

A.3 The oscillator block

The oscillator block simulates rhythmic activity in neurons. It sends an *on* message to the burst generator block every t_{osc} time units starting at $t = t_\phi$.

A.4 The burst generator block

The burst generator block generates a burst of action potentials upon reception of an *on* message. The arrival of the *on* message triggers the start of a cycle of state changes. The sequence starts with a change from state off to state *on* (onset of the first action potential). After t_{ap} time units, the state changes from *on* to ref (beginning of the refractory period). After t_{ref} time units it returns to state *on* (start of the second action potential in the sequence). This cycle is repeated N_{burst} times (making up a burst of N_{burst} action potentials). An *on* message is broadcasted to all synapses driven by the burst block whenever its state changes from off to *on* in order to communicate the start of the propagation of an action potential along its axon.

B. Piriform cortex model

This discrete model of the piriform cortex is based on the compartmental simulations by Wilson *et. al* [11] and Barkai *et al.* [12]. Four types of cells have been included: fast excitatory pyramidal cells, fast inhibitory ($GABA_A$) cells, slow inhibitory ($GABA_B$) cells and stimulus (LOT) cells (Fig. 2).

[Figure 2 about here.]

The pyramidal cell layer consists of a grid of 250 × 250 neurons whereas $GABA_A$ and $GABA_B$ inhibitory cells are arranged in two layers of 80 × 80 cells each. For clarity, these are depicted in separate planes in Fig. 2. However, they occupy the same plane in the actual model.

The layer labeled LOT, models the input activity which arrives at the piriform cortex by the lateral olfactory track. The number of cells in this pool has been adjusted for each simulation in order to provide the desired rate of excitation. As they are topologically far from the rest of the cells, these neurons do not contribute to the simulated field potential recordings.

Pyramidal cells possess local and long range intralayer excitatory connections (amongst pyramidal cells) and local interlayer connections (exciting nearby $GABA_A$ and $GABA_B$ cells). Inhibitory cells ($GABA_A$ and $GABA_B$ layers) do not have intralayer connections in this model. Instead, they locally inhibit pyramidal cells by means of local connections.

The target neuron, j, of a synapse from neuron i is chosen by generating a random vector $\vec{d}$ of components $\{\rho, \phi\}$ (in polar coordinates), where ρ is an exponential random variable (λ given in Appendix B) and ϕ is a uniform variable in the range $0 - 2\pi$. The target neuron is chosen as the closest cell to the vector,

$$\vec{p_j} = \vec{p_i} + \vec{d} \tag{1}$$

where $\vec{p_i}$ and $\vec{p_j}$ are the position vectors for neurons i and j respectively.

LOT cells synapse onto pyramidal cells and introduce external stimulus into the model. The density of connections from LOT to pyramidal cells decreases exponentially from left to right in the pyramidal layer of Fig. 2.

Values for the duration of synaptic activation and synaptic delay have been chosen to reproduce experimentally determined values. Variations of synaptic efficacies across different synapse types are analogous to the maximal conductance of synaptic channels as used in compartmental models. Delays due to axonal propagation have been estimated from experimentally determined values [11]. Further, to reduce memory consumption, the axonal delay has been quantised and the number of allowed values limited to 10.

Neurons fire a single action potential (with a duration of 1 ms and followed by a 10 ms refractory period) whenever w_{sum} (weighted sum of inputs) increases above the excitation threshold. Suitable values for the excitation threshold were determined by parameter space search.

Numerical values for the parameters of the model are provided in Appendix B. Note that, in order to reduce the memory resources required by the simulation, the synaptic delays (t_{del}) were constrained to integer values within the ranges contained in Appendix B. [1]

[1]The quantisation of the synaptic delays causes artifacts in shock stimulus simulations consisting of precisely delimited cortical bands showing homogeneous neuronal states (Fig. 6,t = 13 ms; Fig. 7,t = 14 ms). This effect is not present in realistic long-lasting simulations of EEGs (e.g. Fig. 8(c)), for which reason the computational advantages of delay quantisation motivated the introduction of this simplification in the model

C. Simulation of field potential recordings, EEGs and power spectra

Field potentials and EEGs are measurements of time changing potentials generated by neuronal activity. Field potentials are recorded with single microelectrodes located close to the pool of neurons under study whereas EEGs are recorded with electrodes placed on the skull. For the purpose of model validation, it is desirable to compare the characteristics of the recordings predicted by the model with those seen in experimentally recorded signals.

[Figure 3 about here.]

For the simulation of EEG recordings, a procedure similar to that described by Wilson *et al.* [11] has been followed. A number of virtual electrodes are spatially distributed forming a grid of $E \times E$ recording sites (Fig. 3). Each one of these simulated electrodes obtains a field potential calculated as,

$$S_{FP_i} = \sum_{j}^{J} \sum_{k}^{K} \frac{1}{d_{ij}} \delta_j(t - t_k) * h(t) \tag{2}$$

where S_{FP_i} is the field potential signal recorded by the i^{th} electrode , d_{ij} is the distance between the i^{th} electrode and neuron j, $\delta_j(t - t_k)$ is the delta function indicating that neuron j fired an action potential at $t = t_k$ and $h(t)$ is the prototype field potential recorded from a group of neurons firing nearly simultaneously. The summations are over the total number of action potentials K generated by neuron j and over all the neurons J in the network.

The prototype field potential, $h(t)$, is shown in the box in Fig. 3 and given by (t in ms),

$$h(t) = \begin{cases} t < 0 & 0 \\ 0 < t < 5 & -5 \\ 5 < t < 12 & 2 \\ 12 < t & 0 \end{cases} \tag{3}$$

where the negative segment accounts for the negative field potential recorded experimentally during the onset of action potentials and the positive segment corresponds to the positive field potential seen during repolarization (its return to resting voltage) [17] [18].

Pages 273-275
missing from
this Thesis

second wave. Simultaneously, the pyramidal to pyramidal excitation has also decreased (compare rightmost panels at $t = 14 - 15\ ms$ and $t = 18\ ms$). The remaining excitation is only able to trigger sparse action potentials in a few cells (see leftmost region in the first column of panels at $t = 15 - 18\ ms$).

In contrast with the weak stimulus, the strong stimulus causes fast excitation which leads to the disappearance of excitatory input before the GABA inhibition following the passing wave deactivates.

C. Random input response

Experimentally recorded EEGs often display oscillations with well delimited frequency bands [21]. These pseudo-periodic EEG profiles are thought to be supported by spatial waves of excitation [22], [23] sweeping across the cortex. To study these phenomena with our cortical model, a long-lasting random input stimulus was used. Random excitation is more closely related to the normally functioning piriform cortex than the shock stimulus. It was generated by spreading the firing times of the LOT neurons throughout the entire simulation. Each neuron in the LOT pool was configured to fire once and its firing time is given by a uniform distribution in the range $(0 - t_{stop})$, where t_{stop} is the duration of the simulation. Hence, the stimulus intensity, expressed as the average number of excitatory synaptic connections from LOT cells to pyramidal neurons activated per unit of time, is given by

$$R = \frac{N_{LOT}\ C_{LOT-to-pyr}}{t_{stop}} \tag{5}$$

where N_{LOT} is the number of LOT cells and $C_{LOT-to-pyr}$ the number of connections to pyramidal cells from a single LOT cell. Fig. 8(a) shows the simulated EEG obtained using a grid of 10x10 electrodes and a stimulus of $R = 10^4$ activations/ms. Fig. 8(b) is the corresponding power spectrum.

[Figure 8 about here.]

The unstructured random input generates an structured activity pattern in the cortical model.

The simulated EEG (Fig. 8(a)) shows an initial transitory phase of approximately 150 ms, followed by sequences of peaks with intervals of diminished activity. In its power spectrum (Fig. 8(b)), two main frequency bands appear; 0-10 Hz and a higher frequency range 30-35 Hz. A secondary harmonic peak is also present at 60-65 Hz. Fig. 8(c) shows a state map of the pyramidal layer. It is characterized by cortex-wide excitation waves, reminiscent of those observed for shock stimulus. Each peak of the main frequency component (30 Hz) in the EEG can be associated to a single wave propagating across the cortex.

D. Effect of synaptic parameters on EEG profiles

The impact of model parameters on the characteristics of the EEG was explored. In particular, variations of t_{dur} in inhibitory and excitatory synapses were found to trigger EEG profile transitions. Fig. 9 shows the EEG traces for several values of t_{dur} in $GABA_A$ (leftmost column), $GABA_B$ (middle column) and excitatory synapses (rightmost column). The middle trace in all columns corresponds to nominal values.

Only minor profile alterations were induced by variations in $GABA_A$ synapses. Close analysis of the leftmost column in Fig. 9 shows, however, that the EEG corresponding to $t_{dur} = 14\ ms$ displays higher regularity than those seen for $t_{dur} = 10\ ms$ and $t_{dur} = 11\ ms$.

On the other hand, a decrease of t_{dur} in $GABA_B$ synapses from 150 ms to 50 ms produces marked changes in the EEG, leading to the absence of bursts and nearly sinusoidal traces. Conversely, an increase to 250 ms results in an EEG with longer interburst latencies.

[Figure 9 about here.]

An opposite effect results from changes in the activation duration of excitatory synapses (see rightmost column in Fig. 9). For a value of $t_{dur} = 3\ ms$ (top), the EEG corresponds to a biphasic sequence of bursts. Progressive increases (towards bottom) lead to a steady state consisting of nearly sinusoidal profile, for $t_{dur} = 6\ ms$ and $t_{dur} = 7\ ms$.

IV. Discussion

A model of the piriform cortex has been constructed by means of a hierarchically defined finite state automaton neuron. It aims at demonstrating the usefulness of an event-driven framework where fundamental features of neuronal function can be captured while avoiding the computational complexity inherent to analog models. The model with 10^5 and 3×10^7 synapses represents an increase of two to three orders of magnitude in problem size with respect to previous simulations [11][12].

The standard approach to realistic simulation of large neural aggregates makes use of an analog paradigm based on core-conductor theory of axons and dendrites [24] and Hodgkin-Huxley ion channel models [3]. Computational complexity can be reduced in various ways; minimizing the number of isopotential segments (compartments), limiting the number ion channel types or even substituting those responsible for action potential generation by a threshold function (e.g. the integrate and fire model [25]). Uncoupling of the equations belonging to different neurons is possible by exploiting the discrete nature of the spike. In this case, analog models describe neurons whereas an event-driven engine manages the inter-neuron communication at the synaptic level [4].

We have used a completely event-driven description of the neuron itself, eliminating the need for a continuous simulation engine altogether. Within this framework, computational efficiency arises from the state update scheme, where only those neurons receiving messages at a particular time point must be re-evaluated. The simplicity of the update operation also contributes to diminishing the need for processing resources.

To investigate the dynamics of a large scale cortical model, the cell types and the connectivity patterns in the network were constrained by anatomical studies. The three synaptic classes considered, $GABA_A$, $GABA_B$ and excitatory were configured with relative efficacies and time constants in accordance with experimental findings. The following issues have been addressed: 1) wave generation in a pyramidal layer deprived of inhibition; 2) response of the model, including pyramidal and inhibitory inter-neurons, to pulse-like excitation (shock stimuli); 3) the genesis of EEG oscillations as a result of unpatterned long-lasting stimuli; 4) the modulation of the temporal

EEG profiles by variations in the synaptic time constants.

A. Shock stimulus

The predicted response to shock stimuli shows non-linear properties in agreement with experimental results [10]. Namely, a low intensity pulse stimulus elicits a sequence of model-wide excitation waves and a ringing EEG whereas high intensity stimuli lead to single waves and single peak EEGs. More subtle experimental results are also accounted for by the model. For instance, current source density analysis shows that different synaptic types are maximally active at different points in time during shock response [26]. This effect can be seen in Fig. 5(a) where two secondary peaks can be distinguished, the first corresponding to excitatory synapses from LOT to pyramidal neurons and the second to the delayed activation of pyramidal to pyramidal synapses. Similar double-bumped shock responses were obtained with compartmental [11] and relaxation models [22].

B. Random stimulus

Waves of excitation have been proposed as the physical phenomena underlying EEG oscillations and are thought to arise throughout the cortex [23]. Experimental *in-vivo* studies of piriform cortex and olfactory bulb activity have indeed confirmed that EEG oscillations occur and are especially regular during odour inhalation [27].

To investigate the generation of cortical waves, a temporally unstructured input stimulus was used. The cells in the LOT neuronal pool, which project to the piriform cortex from the olfactory bulb in the actual cortex, were configured to fire randomly throughout the simulation. This setup aimed at producing an input stimulus more closely related to the real pattern of activations than that utilized in shock experiments. Its random nature guarantees that the derived cortical spatio-temporal patterns arise from the intrinsic anatomical and dynamical properties of the model rather than the pre-arranged structure of its input.

This activity leads to cortex-wide waves in our model, in line with compartmental simulations [11]. Further, each EEG peak can be related to a particular cortical wave sweeping across the model.

These waves are preferentially originated in the leftmost region in all panels (corresponding to the rostral end in the actual cortex). This result is the consequence of the decrease of LOT-pyramidal connections when moving from left to right in the panels (dorsal to caudal in the actual cortex), an anatomical feature already observed in early experimental studies [28].

There is experimental and theoretical evidence supporting the hypothesis that global changes of network parameters trigger a switch between functionally different aggregate states. In the olfactory cortex, such a mechanism has been reported [12], [29]. The generalized release of Acetylcholine (Ach) is thought to alter synaptic dynamics and neuronal excitability [30], triggering a mode transition from memory recall to memory acquisition. More generally, abrupt EEG transitions are a well known phenomenon, often associated with changes of conscious states (sleep, walk, anesthesia and so on).

The variations in the temporal and frequencial profiles of the EEG resulting from parameter changes were investigated. In particular, transitions between nearly periodic, bursting and irregular EEGs can be achieved by means of changes in the synaptic activation duration (t_{dur}) of the different synaptic types.

The emergence of spatially coherent patterns and the transition between modes of operation by means of network parameter variation is also in agreement with previous cortical simulations utilizing integrate and fire models [31].

Additional simulations were carried out with equal size cell populations, with three 150×150 grids corresponding to the pyramidal, fast and slow inhibitory cell types (results not shown). This was done to compare our results with those obtained by Wilson *et al.* [11] who included 1500 neurons in each pool. With this configuration, a parameter set was also found which produced single peak and damped ringing in response to shock stimuli and continuous oscillations in response to unpatterned long-lasting input. In the latter case, the main spectral component was centered at 40 Hz, which coincides with the results in [11]. This constitutes a shift of approximately 10 Hz towards higher frequencies with respect to the spectra described in the results section, obtained for more realistic relative population sizes. The large parameter space presented by the model makes

it likely that multiple configurations exist whose EEG share a number of temporal and frequencial characteristics. As already shown, the synaptic parameter t_{dur} affects EEG profiles and suggests that parameter tuning would induce further shifts in the main frequency components.

The simulations carried out have aimed at comparing the model response with experimentally obtained recordings and theoretical investigations with analog neuron descriptions. It has not been attempted, however, to link network dynamics to the suspected functionality of the piriform cortex within the olfactory system. Several studies have tackled this problem: the Lynch-Granger model [32], [33] suggests that the system olfactory bulb-olfactory cortex performs hierarchical clustering of the cue environment; the Li-Hertz model [14] proposes that odour recognition is achieved by a resonance phenomenon between cortex and bulbar oscillations; the Wilson-Bower model [34] implements an associative memory able to store and retrieve odour information.

A feature shared by this model is the use of synaptic modification algorithms or other types of network plasticity which must me activity-dependent to allow the storage of new odours. The model described here does not incorporate plasticity. However, the striking similarities between the physiological data and the results obtained with the simple event-driven model supports the possibility of utilizing the same framework for investigations of the functional role of synaptic plasticity.

V. Concluding remarks

In summary, the approach presented in this paper allows large scale neural simulations on single processor desktop computers and the exploration of the effects of physiological parameters on neural population dynamics. This is achieved by devising a completely event-driven framework where the need for computationally costly continuous simulation engines is eliminated altogether.

The piriform cortex model provides a tool for the testing of theories related to the nature of the computations carried out by this cortical area. The results obtained so far demonstrate that the main features of the olfactory cortex response to short and long-lasting stimuli can be accounted for by a simple event-driven cortical model.

Moreover, the techniques presented here can also be applied to other areas of the nervous system. The reduction of computational complexity in comparison with alternative strategies suggests the adequacy of the proposed approach for large-scale models incorporating multiple cortical areas. Research is underway to exploit Beowulf clusters in order to provide the necessary resources for such computationally costly simulations. Preliminary results indicate that problem size could be increased at least one order of magnitude by means of a proportional increase in the number of processing nodes with a mere 10% simulation time overhead due to internode communication.

APPENDIX

I. STATE TRANSITION TABLES

[Table 3 about here.]

[Table 4 about here.]

[Table 5 about here.]

[Table 6 about here.]

II. MODEL PARAMETERS

[Table 7 about here.]

Acknowledgements : This work was supported by the Biotechnology and Biological Sciences Research Council, UK, and the Dept. of Electronics and Computer Science, University of Southampton, UK.

REFERENCES

[1] C. Koch and I. Segev, editors. *Methods in neural modeling: From ions to networks.* MIT Press, Cambridge, 1998.

[2] J.A. Freeman and D.M. Skapura, editors. *Neural networks : Algorithms, applications and programming techniques.* Addison-Wesley, 1992.

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

[4] P. Hammarlund and O. Ekeberg. Large neural network simulations on multiple hardware platforms. *Journal of Computational Neuroscience*, 5:443–459, 1998.

[5] M. Bower and D. Beeman, editors. *The Book of Genesis.* TELOS/Springer-Verlag, 1998.

[6] M.L. Hines and N.T. Carnevale. The neuron simulation environment. *Neural Computation*, 9(6), 1997.

[7] E. Pytte, G. Grinstein, and R.D. Traub. Cellular automaton models of the CA3 region of the hippocampus. *Network: Computation in Neural Systems*, 2(2):149–167, 1991.

[8] R. Mansharamani. An overview of discrete event simulation methodologies and implementation. *Sadhana*, 22(5):611–627, 1997.

[9] L.B. Harberly and S.L. Feig. Structure of the piriform cortex of the opossum. I. fine structure of cell bodies and neuropil. *J. of Comp. Neurology*, 216:69–88, 1983.

[10] K.L. Ketchum and L.B. Haberly. J. of Neurosc. *Synaptic events that generate fast oscillations in piriform cortex*, 13(9):3980–3985, 1993.

[11] M. Wilson and J.M. Bower. Cortical oscillations and temporal interactions in a computer simulation of piriform cortex. *Journal of Neurophysiology*, 67(4):981–995, 1992.

[12] E. Barkai, R.E. Bergman, G. Horwitz, and M.E. Hasselmo. Modulation of associative memory function in a biophysical simulation of rat piriform cortex. *J. of Neurophysiol.*, 72(2):659–677, 1994.

[13] L.B. Harrberly and J.M. Bower. Olfactory cortex: model circuit for study of associative memory? *TINS*, 12(7):258–264, 1989.

[14] Z. Li and J. Hertz. Odor recognition and segmentation by coupled olfactory bulb and cortical networks. *Neurocomputing*, 26-27:789–794, 1999.

[15] E.T. Claverol, A.D. Brown, and J.E.Chad. Discrete simulation of large aggregates of neurons. *Submitted to Neurocomputing*, 2000.

[16] R.L. Bagrodia, K.M Chandy, and J. Misra. A message-based approach to discrete-event simulation. *IEEE Trans. on Soft. Eng.*, 13(6):654–665, 1987.

[17] W. Rall. Electrophysiology of a dendritic model. *Biophys. J.*, 2:145–167, 1962.

[18] M. Klee and W. Rall. Computed potentials of cortically arranged populations of neurons. *J. of Neurophysiology*, 40:647–666, 1977.

[19] W.H. Press, W.T. Vetterling, S.A. Teukolsky, and B.P. Flannery. *Numerical recipes in C*, chapter 13. Cambridge Univ. Press, 2th edition, 1992.

[20] W.J. Freeman. Relations between unit activity and evoked potentials in prepyriform cortex in cats. *J. Neurophysiol.*, 31:337–348, 1968.

[21] S.L. Bressler. Spatial organization of EEGs from olfactory bulb and cortex. *Electroencephalogr. Clin. Neurophysiol*, 57:270–276, 1984.

[22] T. Ballain, P. Litaudon, J. Martiel, and M. Cattarelli. Role of the net architecture in piriform cortex activity: analysis by a mathematical model. *Biological Cybernetics*, 79(4):232–336, 1998.

[23] P.L. Nunez. *Electric fields of the brain: the neurophysics of EEG.* Oxford Univ. Press, New York, 1981.

[24] W. Rall. Core conductor theory and cable properties of neurons. In Kandel E.R., editor, *Handbook of Physiology.*, pages 39–97. American Physiological Society, 1977.

[25] W. Gerstner. Time structure of the activity in neural network models. *Physical Rev. E*, 51:738–758, 1995.

[26] K.L. Ketchum and L.B. Haberly. J. of Neurophys. *Membrane currents evoked by afferent fiber stimulation in rat piriform cortex I. Current source-density analysis*, 69(1):248–260, 1993.

[27] W.J. Freeman. The physiology of perception. *American Scientific*, 264(2):78–85, 1991.

[28] J.E. Schwob and J.L. Price. The cortical projection of the olfactory bulb: development in fetaland neonatal rats correlated with quantitative variations in adult rats. *Brain Research*, 151:369–374, 1978.

[29] M.E. Hasselmo and E. Barkai. Cholinergic modulation of activity-dependent synaptic plasticity in the piriform cortex and associative memory function in a network biophysical simulation. *J. of Neurosc.*, 15(10):6592–6604, 1995.

[30] E. Barkai, G. Horwitz, R.E. Bergman, and M.E. Hasselmo. Modulation of the input/output function of rat piriform cortex pyramidal cells. *J. of Neurophysiol.*, 72(2):644–658, 1994.

[31] S.L. Hill and A.E.P. Villa. Dynamic transitions in global network activity influenced by the balance of excitation and inhibition. *Network: Computation in Neural Systems*, 8(2):165–184, 1997.

[32] J. Ambros-Ingerson, R. Granger, and G. Lynch. Simulation of paleocortex performs hierarchical clustering. *Science*, 247:1344–1348, 1990.

[33] E. Barnard. Analysis of the lynch-granger model of olfactory cortex. *Biol. Cybern.*, 62(2):151–155, 1989.

[34] M. Wilson and J. Bower. A computer simulation of olfactory cortex with functional implications for storage and retrieval of olfactory information. In Anderson D.Z., editor, *Neural Information Processing Systems*, pages 114–126. American Institute of Physics, New York, 1988.

List of Figures

Figures 21

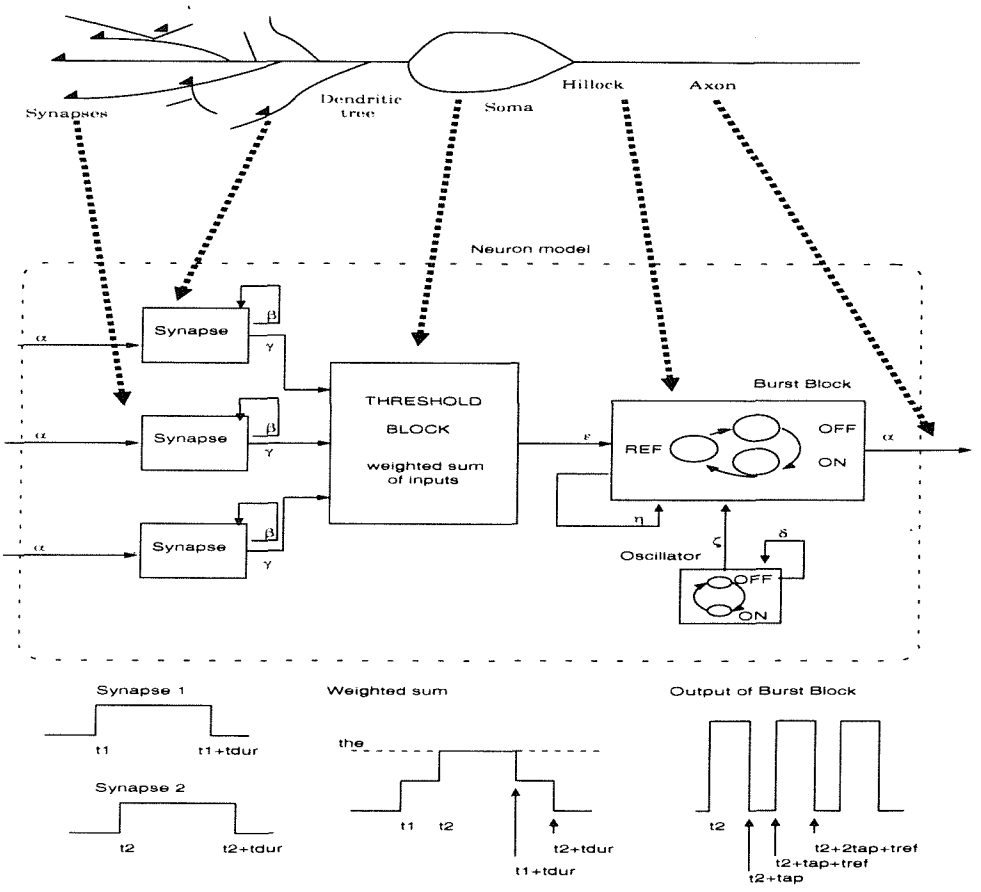


Fig. 1. Message-based event-driven neuron model. Solid arrows indicate origins and destinations of messages. Thick dashed arrows indicate the correspondence between parts of the real neuron and blocks in the model.

Figures 22

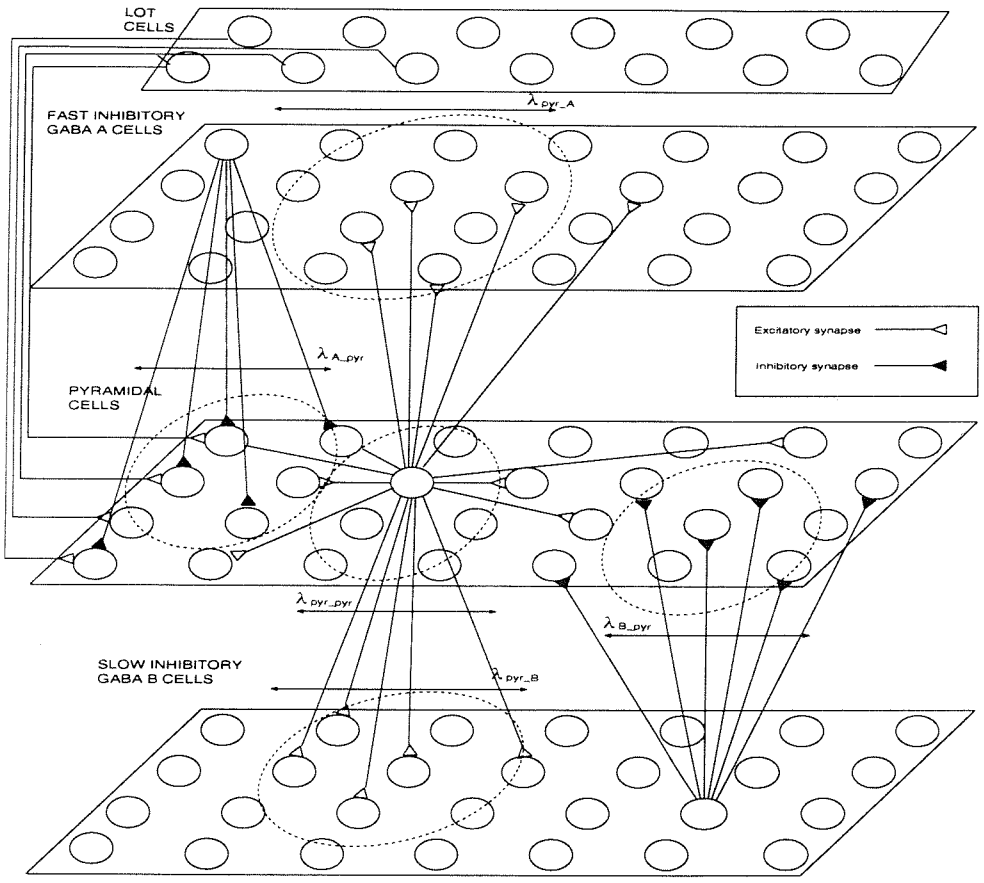


Fig. 2. Model of the piriform cortex.

Figures 23

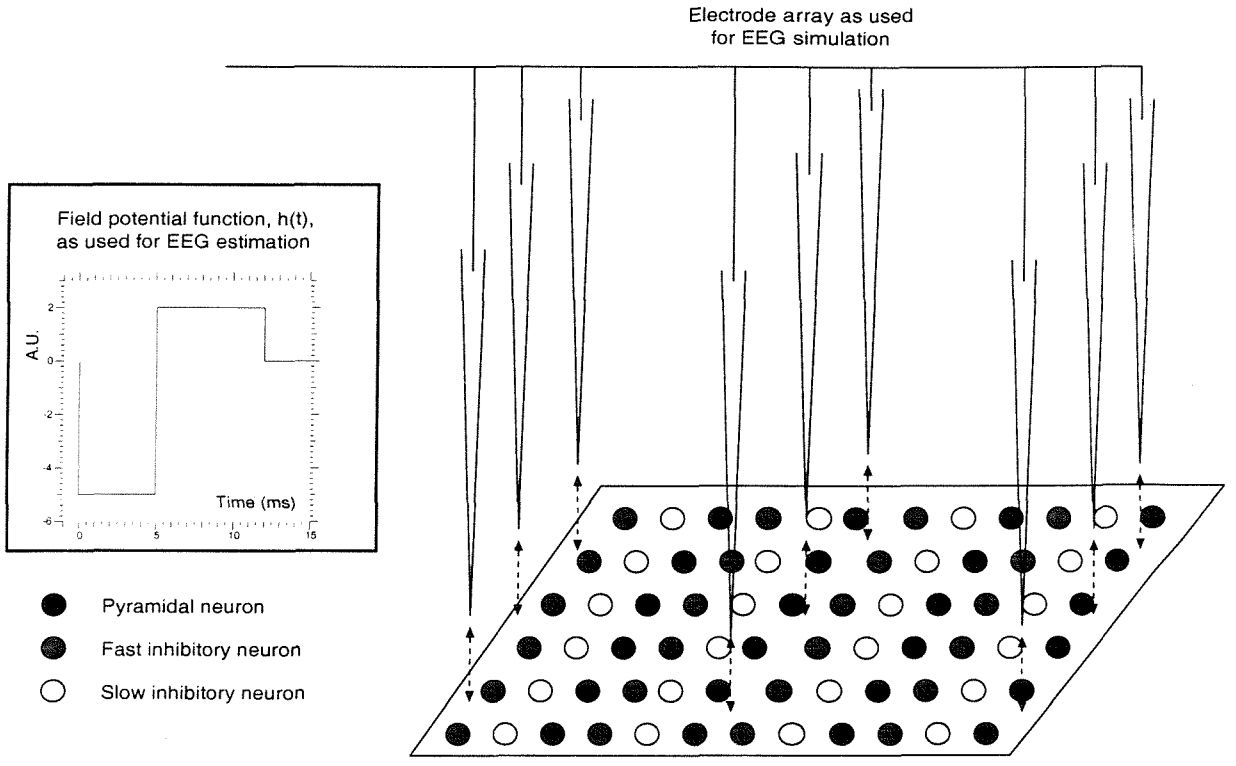


Fig. 3. Setup used for the simulation of field recordings and EEGs.

Figures 24

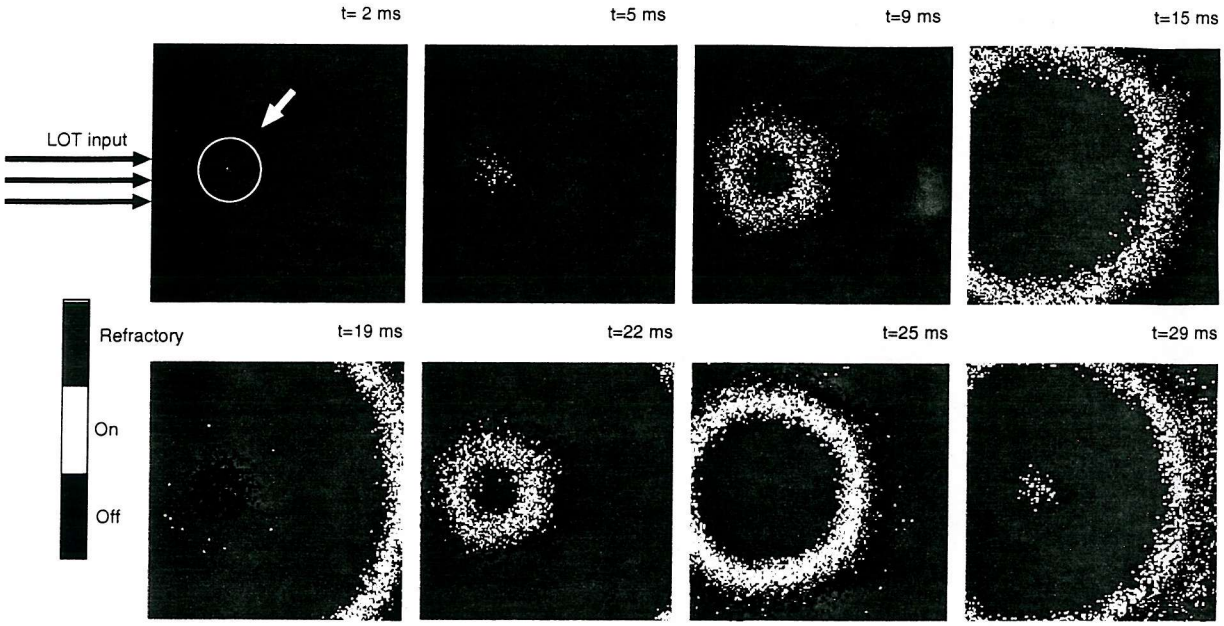


Fig. 4. Sequence of images representing, (a) the state and (b) the weighted sum of inputs (w_{sum}) of pyramidal neurons in a partially connected model (each pixel in the arrays corresponds to an individual neuron)

Figures 25

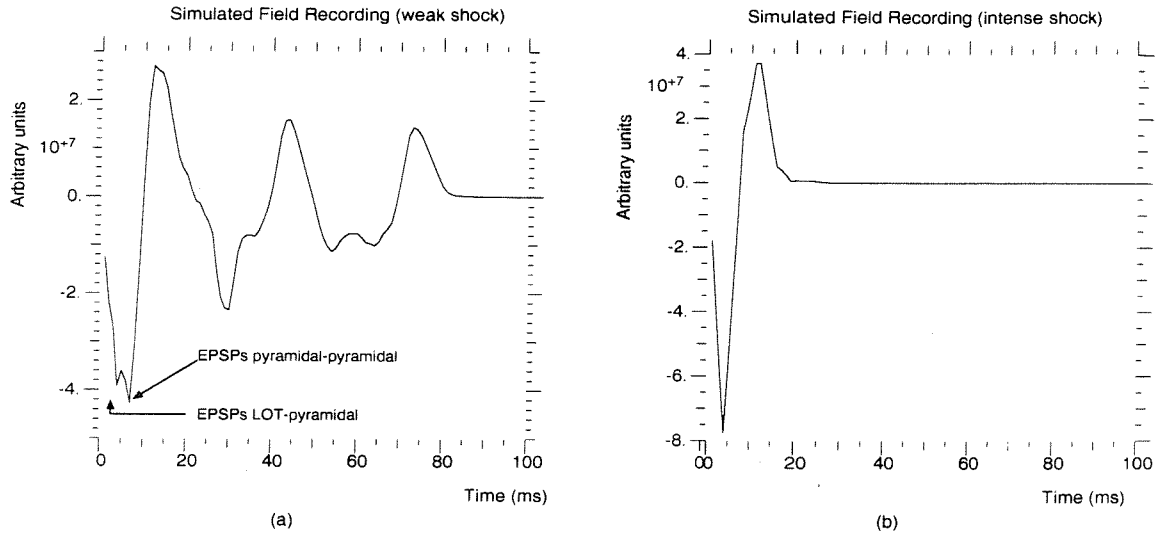


Fig. 5. (a) Simulated field potential after weak shock stimulus. (b) Simulated field potential after strong shock stimulus

Figures 26

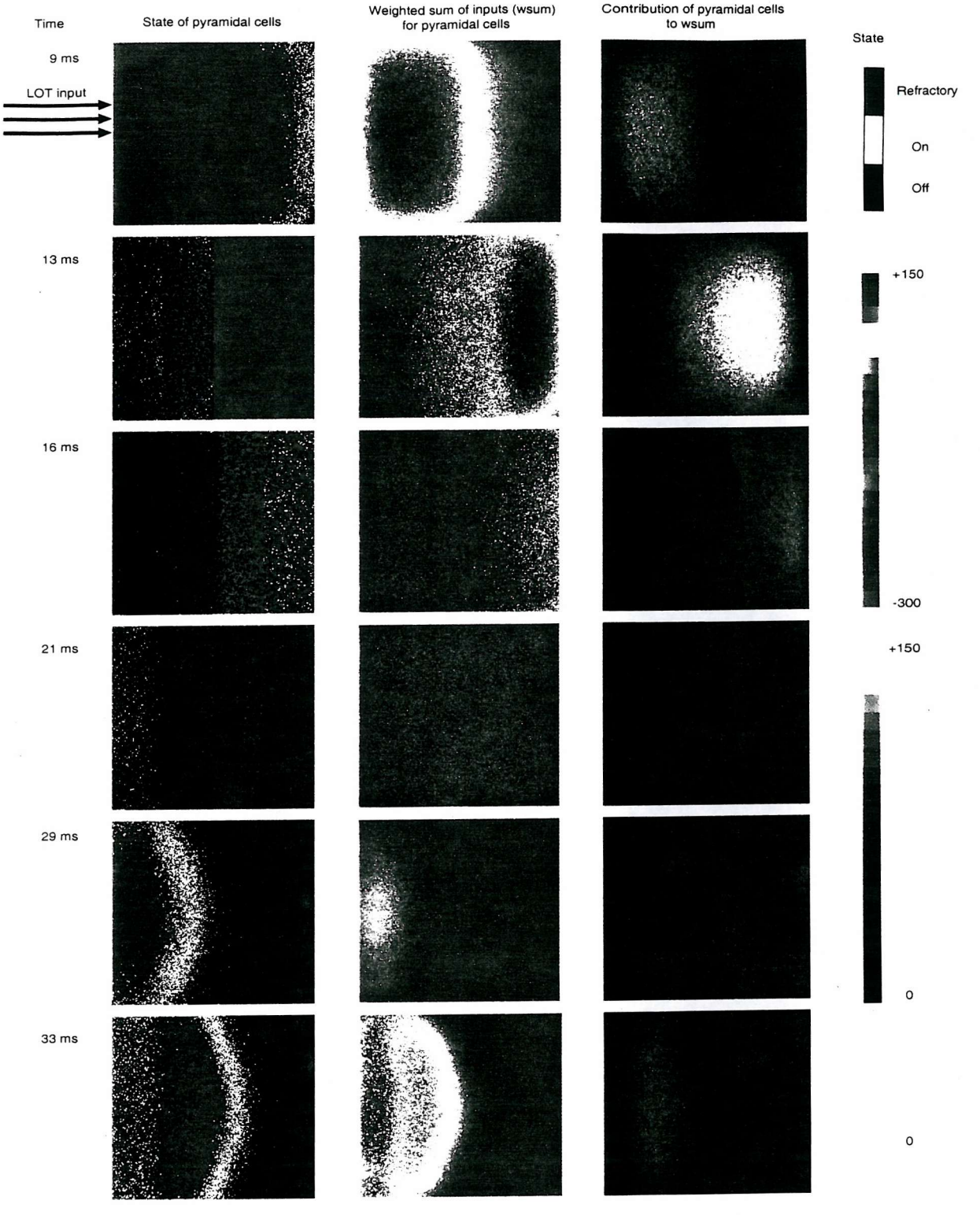


Fig. 6. States, w_{sum} and pyramidal-pyramidal excitation for pyramidal neurons after weak shock stimulus

Figures 27

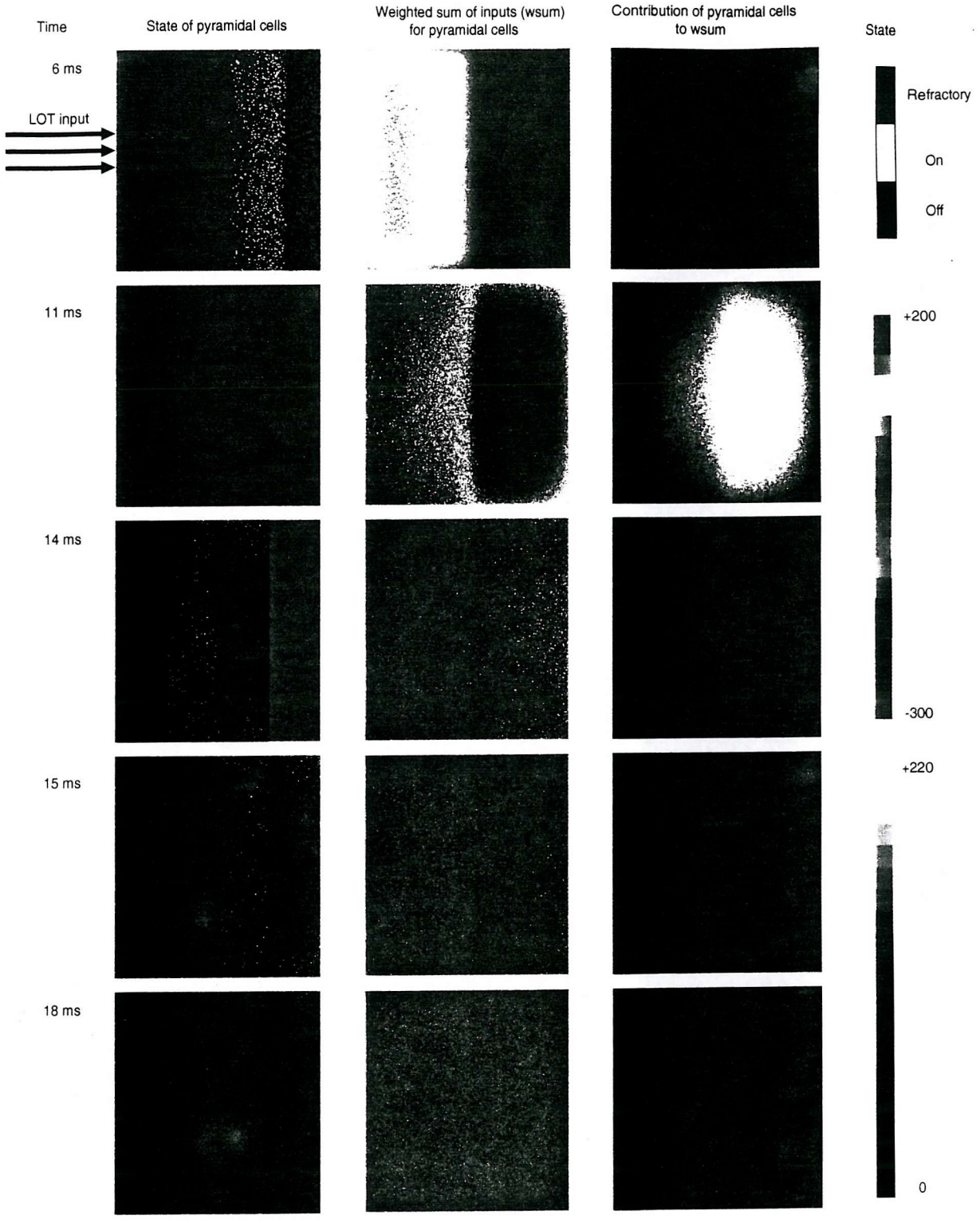


Fig. 7. States, w_{sum} and pyramidal-pyramidal excitation for pyramidal neurons after strong shock stimulus

28

Figures

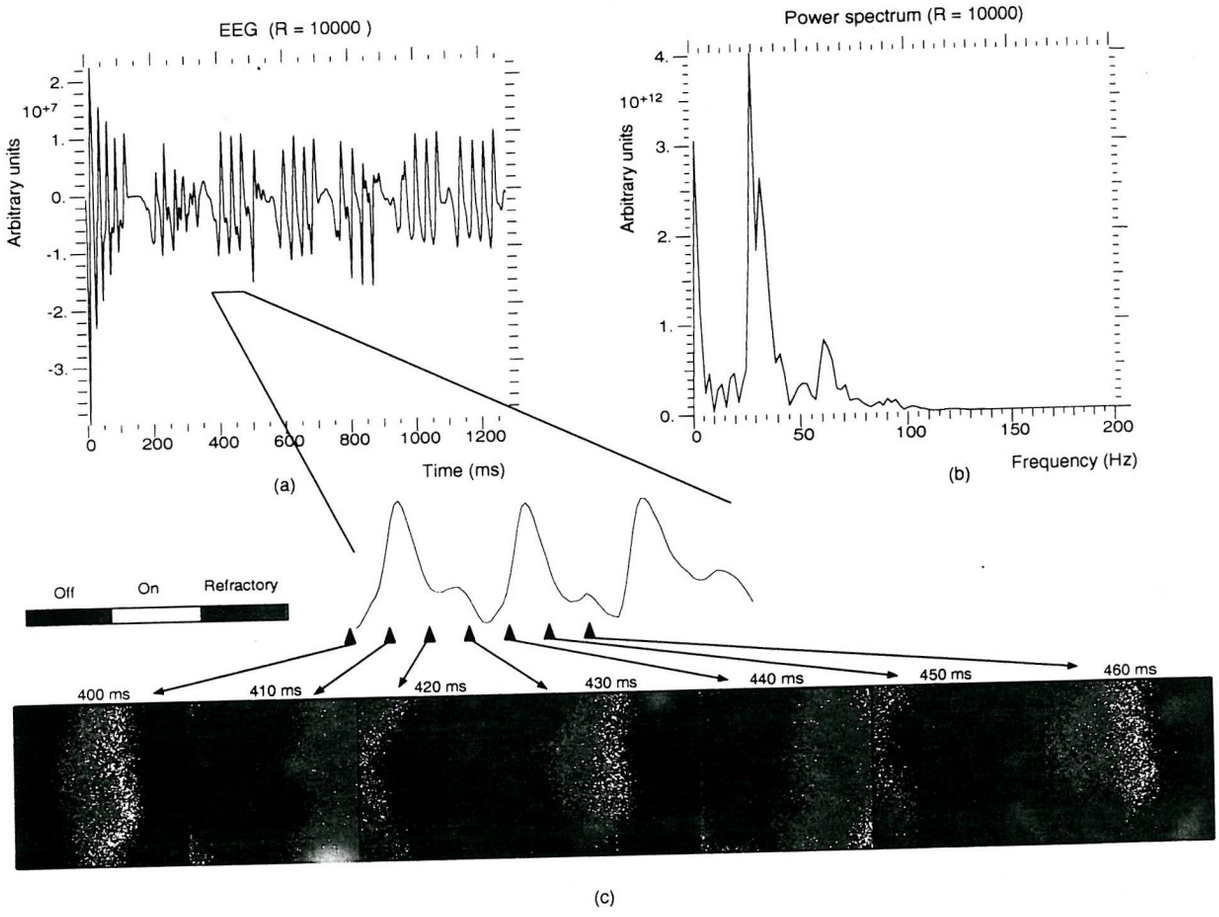


Fig. 8. (a) Simulated EEG. (b) Power spectrum of the EEG. (c) Cortical waves underlying EEG oscillations

Figures 29

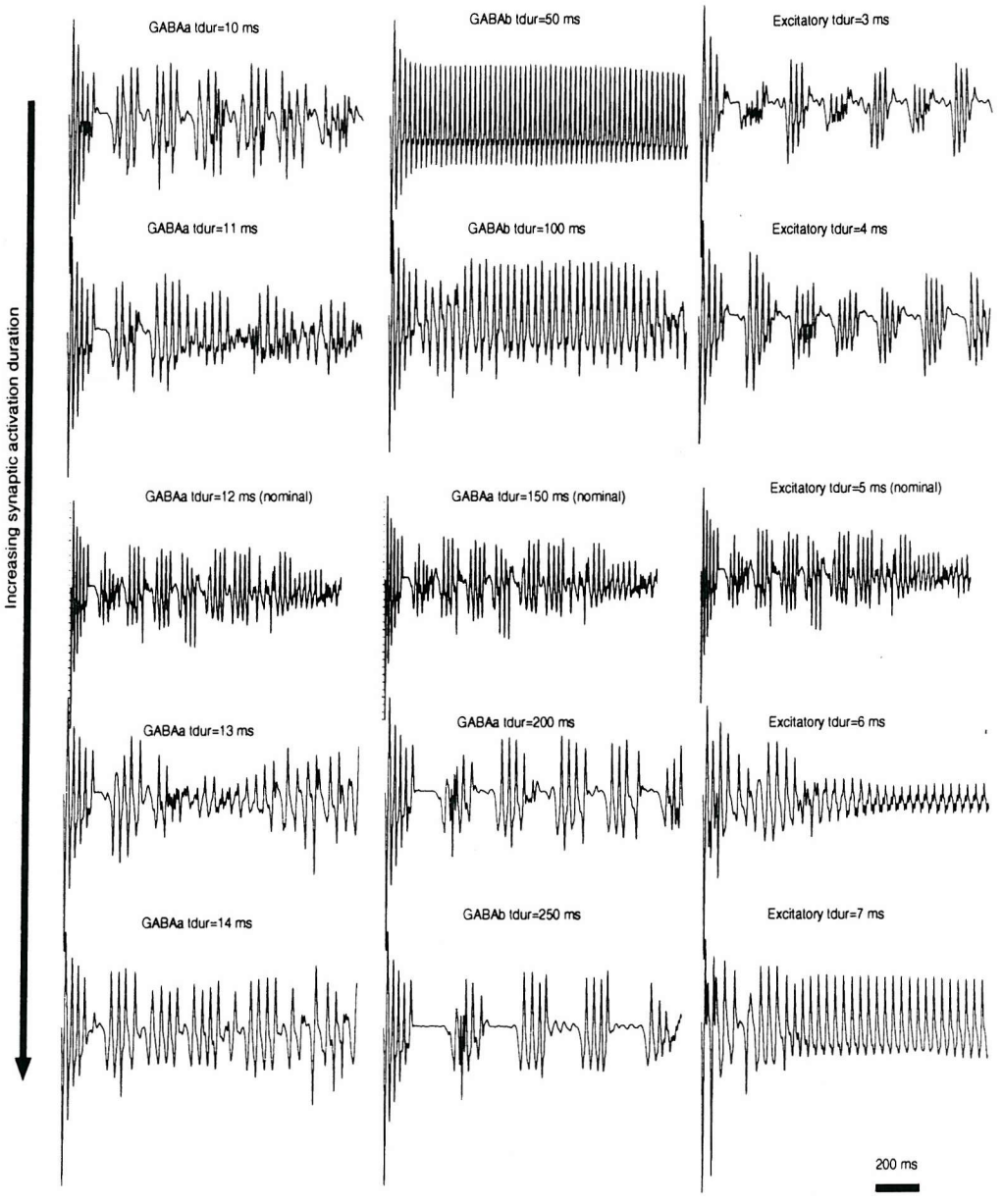


Fig. 9. EEG profile for several values of t_{dur} in $GABA_A$, $GABA_B$ and excitatory synapses

List of Tables

Channel	Message structure	Legal values of m	Legal values of p
α	{t,m}	*on*	
β	{t,m}	*on*,*off*	
γ	{t,m,p}	*on*,*off*	efficacy(w_{syn})
δ	{t,m}	*change*	
ϵ	{t,m}	*on*,*off*	
ζ	{t,m}	*on*	
η	{t,m}	*off*, *r_off*	

TABLE I

MESSAGE CHANNELS IN THE NEURON MODEL

Parameter	Function
th_e	Excitation threshold
th_i	Inhibition threshold
t_{ap}	Duration of action potential
t_{ref}	Duration of refractory period
N_{burst}	Number of spikes per burst
t_{osc}	Period of pace maker
t_ϕ	Time offset of pace maker
t_{del}	Synaptic delay
t_{dur}	Duration of synaptic pulse
w_{syn}	Synaptic efficacy

TABLE II

PARAMETERS IN THE NEURON MODEL

Synapse block	
Input	Output
$\alpha := on$	$\beta := \{t_{del}, on\}$
$\beta := on$	$\beta := \{t_{dur}, off\}$, $\gamma := \{0, on, w_{syn}\}$
$\beta := off$	$\gamma := \{0, off, -w_{syn}\}$

TABLE III

THE SYNAPSE BLOCK FUNCTION

Threshold block	
Input	Action | Output
$\gamma := on$	$w_{sum} + = w_{syn}$ $w_{sum} >= th_e$? true: - | $\epsilon := \{0, on\}$ $w_{sum} <= th_i$? true: - | $\epsilon := \{0, off\}$
$\gamma := off$	$w_{sum} - = w_{syn}$ $w_{sum} >= th_e$? true: - | $\epsilon := \{0, on\}$ $w_{sum} <= th_i$? true: - | $\epsilon := \{0, off\}$

TABLE IV

THE THRESHOLD BLOCK STATE MACHINE

Oscillator	
Current state	Next state | Output
	Input
	$\delta := \{t_{osc}, change\}$
on	$off \mid \delta := \{t_{osc}, change\}, \zeta := \{0, on\}$
off	$on \mid \delta := \{t_{osc}, change\}, \zeta := \{0, on\}$

TABLE V
THE OSCILLATOR STATE MACHINE

Burst generator					
Current state	Next state | Action | Output Input				
	$\epsilon := on$	$\epsilon := off$	$\eta := off$	$\eta := r\_off$	$\zeta := on$
on	on | - | -	on | $n_{burst} = 0$ | -	ref | - | $\eta := \{t_{ref}, r\_off\}$	on | - | -	on | - | -
ref	ref | - | -	ref | $n_{burst} = 0$ | -	ref | - | -	$n_{burst} - 1 == 0$? true: off |$n_{burst} = N_{burst}$ | - false: on |$n_{burst} - = 1$ | $\eta := \{t_{ap}, off\}$	ref | - | -
off	on | - | $\alpha := \{0, on\}$, $\eta := \{t_{ap}, off\}$	off | - | -	off | - | -	off | - | -	on | - | $\alpha := \{0, on\}$, $\eta := \{t_{ap}, off\}$

TABLE VI

THE BURST GENERATOR STATE MACHINE

Neuronal parameters	
th_e (pyramidal)	7
th_e (fast inh.)	30
th_e (slow inh.)	30
th_i	-1000 (burst truncation inactivated)
t_{ap}	1 ms
t_{ref}	10 ms
N_{burst}	1
t_{osc} (pyramidals and inhibitory)	0 (*inactive oscillator*)
t_ϕ (pyramidals and inhibitory)	0 (*inactive oscillator*)
t_{osc} (LOT cells, all stimuli)	3000 ms
t_ϕ (LOT cells, shock stimulus)	0 ms
t_ϕ (LOT cells, random input)	Uniform$(0 - t_{stop})$
Number of synapses per neuron	
LOT to pyramidal	100
pyramidal to pyramidal	300
pyramidal to fast inhibitory	20
pyramidal to slow inhibitory	10
fast inhibitory to pyramidals	70
slow inhibitory to pyramidals	60
Synaptic parameters	
t_{del} (pyramidal to pyr./inh.)	$(3 - 12\ ms)$
t_{dur} (pyramidal to pyr./inh.)	5 ms
w_{syn} (pyramidal to pyr./inh.)	1
t_{del} (fast inh. to pyramidal)	5 ms
t_{dur} (fast inh. to pyramidal)	12 ms
w_{syn} (fast inh. to pyramidal)	-15
t_{del} (slow inh. to pyramidal)	10 ms
t_{dur} (slow inh. to pyramidal)	150 ms
w_{syn} (slow inh. to pyramidal)	-1
t_{del} (LOT to pyramidal)	$(1 - 4\ ms)$
t_{dur} (LOT to pyramidal)	5 ms
w_{syn} (LOT to pyramidal)	4
Mean connection range, $1/\lambda$ (normalized distance)	
pyramidal to pyramidal	2
pyramidal to fast inhibitory	10
pyramidal to slow inhibitory	10
fast inhibitory to pyramidals	10
slow inhibitory to pyramidals	10
LOT to pyramidals	2

TABLE VII

NUMERICAL VALUES OF PARAMETERS IN THE PIRIFORM CORTEX MODEL

Scalable cortical simulations on Beowulf architectures

E.T.Claverol[†] A.D.Brown[†] J.E.Chad [‡]

[†]Dept. of Electronics and Computer Science. University of Southampton

Mountbatten Building. Salisbury Road

Southampton SO17 1BJ

United Kingdom

[‡] School of Biological Sciences. University of Southampton

Biomedical Sciences Building. Bassett Crescent East

Southampton SO16 7PX

United Kingdom

Corresponding author: E.T.Claverol@ecs.soton.ac.uk

Abstract

Biologically motivated simulation of large scale neural networks is a computationally costly task. In this paper, a commodity 8-node Beowulf architecture is proposed as a scalable low cost environment for studies of cortical dynamics. By means of a distributed message-based event-driven framework, the size of memory-limited tractable problems increased 8-fold, resulting in a mere 8.3% increase in elapsed CPU time, attributable to inter-process communication overhead. The attainable network size reached over 10^6 neurons and 2.5 10^8 synapses, with a typical performance of 900 s, Beowulf processing time, per simulated second.

Key-Words

Parallel Event-driven Neuronal Simulation, Beowulf Cluster, Spiking Neurons, Cell Automata

1 Introduction

Biophysical neuron models rely on analogue descriptions of the spatio-temporal patterns of electrical activity in living cells [1]. Although physically accurate, these models are computationally intensive, requiring the numerical integration of systems of non-linear differential equations.

Work on large scale neural simulations has often resorted to parallel architectures to achieve the necessary processing power [2, 3, 4, 5]. Although substantial performance increases have been demonstrated with hypercube architectures (see for example [6]), the cost of these platforms and the considerable development involved in the customization of the simulation environments, have limited the impact of parallel architectures in the field of neural simulation.

Cell automata models can substantially decrease the demand for processing power [7], since they are suitable for event-driven rather analogue simulation frameworks. Moreover, distributed computation can take advantage of the efficiency inherent to event-driven simulation to achieve further increases in the size of the tractable problems.

Beowulfs constitute an emerging technology aiming at delivering parallel processing power at a reasonable cost by interconnecting commodity single processor PC-based architectures with high speed data links [8, 9, 10]. Their application to simulation of neuronal dynamics is still in its infancy [11].

This paper presents results regarding the application of Beowulf systems to the study of cortical dynamics. The emphasis is on problem scalability employing a distributed event-driven framework.

The cortical model will be briefly described first. Next, the Beowulf under test and the inter-node communication algorithms will be presented. Finally, its performance will be evaluated for various cortical network sizes and topologies.

2 Cortical model

The cortical model consists of a set 2-D lattices of automaton-like neurons suitable for message-based event-driven (MBED) simulation. Action potentials are represented by pulses propagating across the network, a mechanism implemented by message broadcasting between the entities in

the model. Synapses introduce a time lag, implement pulse stretching and have associated weight terms. Neurons generate a spike when the weighted sum of simultaneously active synaptic inputs yields a value above an specified threshold. This is followed by an absolute refractory period, after which, the neuron returns to its initial (excitable) state. Time is represented by integer multiples of a basic simulation time unit of $1ms$.

Three classes of cells were included; excitatory (pyramidal), fast inhibitory (GABAa) and slow inhibitory (GABAb), with synaptic parameters in accordance with electrophysiological data.

The connectivity was guided by previous experimental and modelling work of the olfactory cortex [12]. Excitatory-excitatory connections are long-range (network-wide) whereas excitatory-inhibitory connections are local. No inhibitory-inhibitory synapses were included.

Within each 2-D lattice, input activity was randomly distributed over time and cortical area and generated by a fourth pool of neurons. Its intensity, as well as firing thresholds across the network, were adjusted to replicate previously described cortical waves. Each individual 2-D lattice, a neural sub-aggregate, establishes connections through axonal bundles with other lattices to make up the multi-lattice aggregate.

The strategy followed to distribute the simulation across the cluster was to assign each one of the equal size sub-aggregates to a Beowulf node, yielding a maximum attainable size of 8 sub-aggregates with 1.75 10^5 neurons and 31.45 10^6 synapses each.

3 Beowulf platform

The Beowulf platform under test is illustrated in figure 1-A. It consists of 8 single processor Athlon (AMD-K7) machines running Linux RedHat 6.0 with an aggregate peak performance of 900 MegaFlops, with 2 Gigabytes of memory (256 Mbytes per node) and 100 GigaBytes of disk space. Two SuperStack II 3C16464A 3COM Fast Ethernet switches interconnect the nodes in a star-like topology. An extra node (totalling 9 nodes) functions as a server in charge of job scheduling across the 8-node architecture and other maintenance tasks. This computer does not participate in distributed computations. Inter-process communications make use of the LAM 6.3.1 [13] free implementation of the Message Passing Interface (MPI) [14] libraries.

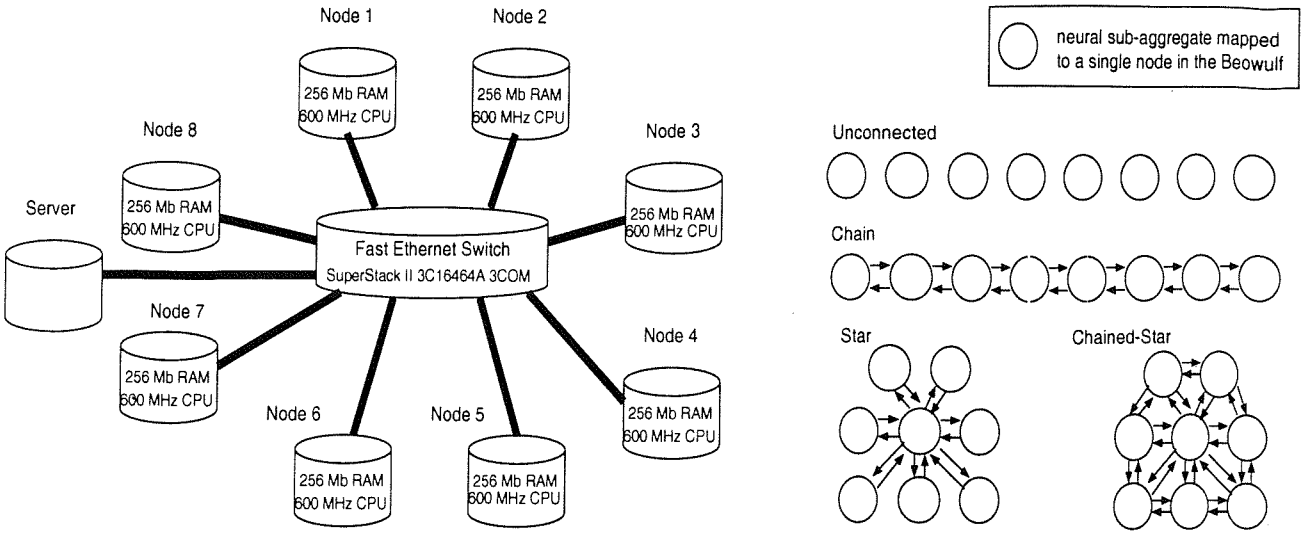


Figure 1: A - Schematic diagram of the Beowulf, B - Neural aggregate topologies under test

4 Inter-aggregate communications

Process synchronization is necessary in the context of distributed message-based event-driven simulation [15] in order to compensate for the unavoidable workload imbalance between simulation processes. Without a synchronization mechanism, a time lag would arise between the simulation clocks of different processes, which could lead to the loss of inter-process messages when the simulation clock at the receiving end is advanced with respect to that of the process originating the message.

Figure 2 provides a complete flow-chart description of the implemented synchronization mechanism. Upon initialization, all processes are synchronized to the time slice $t = 0\ ms$. At the end of each of the subsequent time steps, each process broadcasts a *termination* message (TM) to the others. At this point, the process awaits the reception of the corresponding N-1 TM's (N being the number of nodes) from the rest of the cluster. In the meantime, it continues receiving action potential messages from those nodes where the current time step is still under execution.

All processes, having finished the current time slice, propose the next value for the global simulation clock; this is taken as the scheduling time of the first message in their respective local priority queues. Each process broadcasts its proposed value to process 0, which acts as the coordinator. Of the proposed times, process 0 selects the minimum and broadcasts the value to the rest of the nodes, which set their respective local simulation clocks to the agreed value

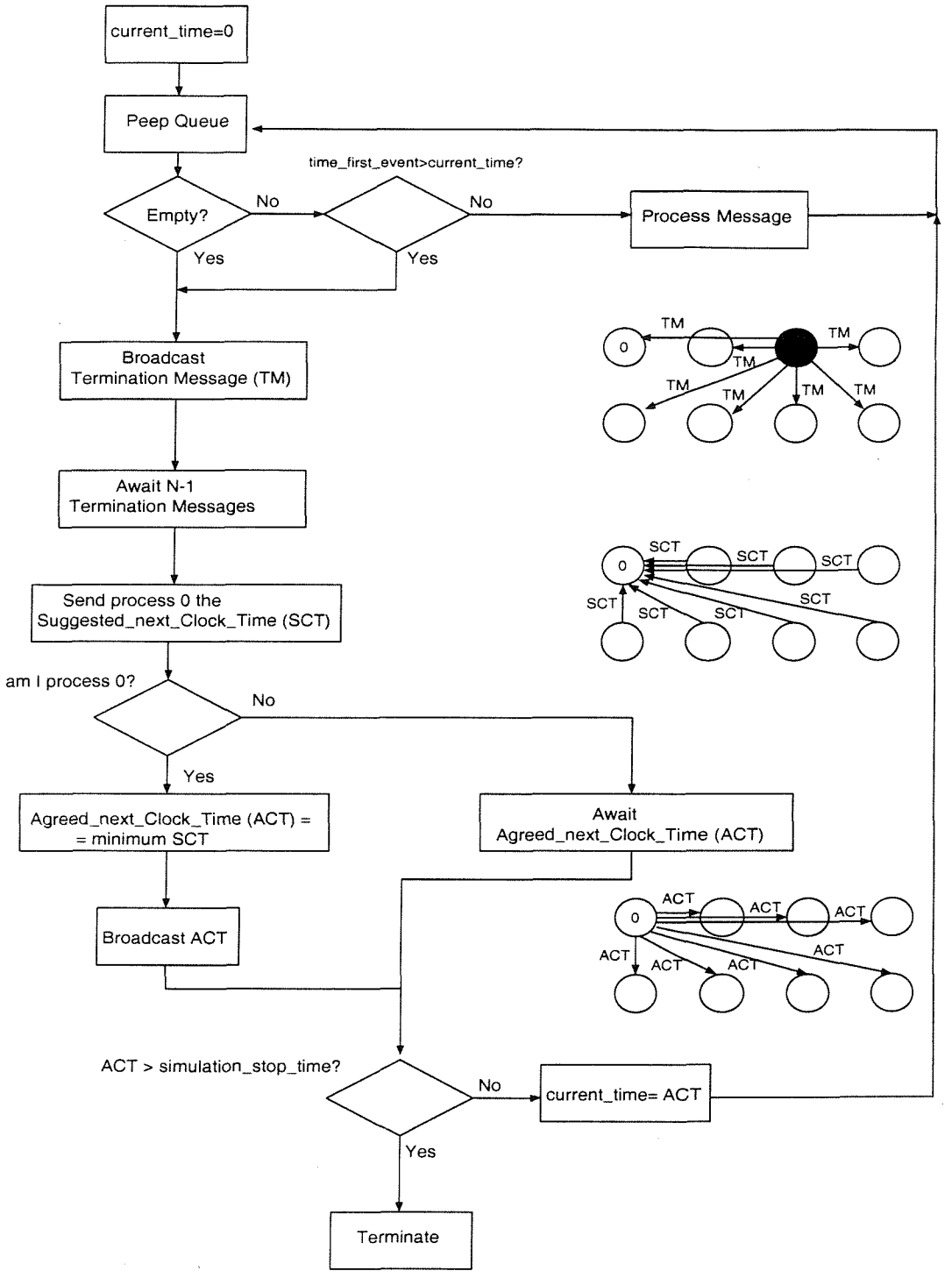


Figure 2: Synchronization algorithm to achieve a cluster-wide coordinated simulation clock

and start processing the messages in their queues (if any) scheduled for this consensued clock time. The simulation finishes when the agreed global simulation clock takes a value beyond the specified simulation stop time.

Another issue regarding inter-node communications, the mapping between action potentials and MPI messages, bears relevance to the overall simulation efficiency. A one-to-one relationship between inter-node MPI messages and propagated action potentials would constitute an inefficient strategy, resulting in a large number of small size messages quickly exhausting the available communication bandwidth. The number of MPI messages for inter-node communication can be minimized, however, by means of a buffering mechanism. The firing of a neuron triggers the addition of its neuron id, a 4-byte integer, to the buffer. Upon finishing a time slice or whenever the buffer is full, its contents, the list of identifiers of firing neurons and a 4-byte header set to the actual number of entries in the data structure, are sent as a single MPI message. The destination nodes are those machines which, in the previously specified neuronal topologically, directly receive axonal tracks. The experiments carried out in this paper made use of a 10 Kbyte buffer which proved sufficiently large to avoid buffer overflow in all the tested cases.

5 Results

The cortical model described in section 2 was chosen as the atomic sub-aggregate (the portion of the network simulated by one node) because previous studies had shown that it was capable of replicating experimental data on cortical dynamics. Thus, the results of the benchmarking are likely to be representative of the performance attainable with a wide range of biologically realistic neural simulation problems.

Given an arbitrarily chosen set of brain areas, only a subset of all the possible pairs would be directly connected by axonal bundles. Assuming a one-to-one mapping between nodes in the cluster and modelled brain regions, it follows that several logical cluster topologies are possible. For performance evaluation, models with various numbers of modules (1-8) and patterns of axonal bundles were simulated in order to explore the effect of these parameters on the elapsed time. Figure 1-B depicts the simulated topologies.

Further, special care was taken to ensure that the performance evaluation was carried out

with realistic neural simulations; since the performance of an MBED simulation engine is strongly affected by the network activity pattern, misleading performance studies can result from simulations with exceedingly low or high neuronal activity. Parameter space search is needed to find the configuration that results in realistic activity in all the nodes conforming the cluster. This is a computationally costly problem in itself, and aggravated by the fact that a new set of parameters has to be found for each one of the logical cluster topologies under test.

A convenient simplification of the network model was put in place to achieve realistic activity and inter-node communication overhead for all nodes while eliminating the need for computationally intensive parameter space searches. Each sub-aggregate includes a pool of neurons which provides stimulation. The inter-module neuronal spikes transported by afferent bundles (and implemented by means of MPI messaging) is actually transmitted to retain the performance degradation caused by communication overhead. This guarantees the validity of the performance results. However, the receiving end disregards the incoming trains of action potentials, and takes its input from the stimulus neuronal pool. The dynamics of such a network is simpler and the parameter space search needs to be carried out once and with a single sub-aggregate rather than with the entire network.

In this way, (1) all sub-aggregates display a realistic level of intra and inter-aggregate activity irrespective of network size and topology, (2) the inter-node data are actually transmitted to evaluate the effect on the performance and (3) computationally expensive parameter space searches are avoided.

Figure 3-A plots the time taken by simulations of 1 s of network activity. The lower trace corresponds to the measured elapsed times averaged over the four topologies tested: unconnected, chain, star and chained-star. For comparison, the upper trace represents a linear estimation of the time taken to simulate equivalent network sizes on a single-processor architecture. Actual measurements of single-processor times could not be performed given that individual sub-aggregates, totalling $1.75\ 10^5$ neurons, were already at the limit of memory resources. The estimated values for a single-processor platform were calculated with a linear approximation; the time taken by the simulation of a network aggregate of N sub-aggregates running on a single node (assuming enough memory space) was approximated by N times the measured time taken by a single sub-aggregate running on a single node.

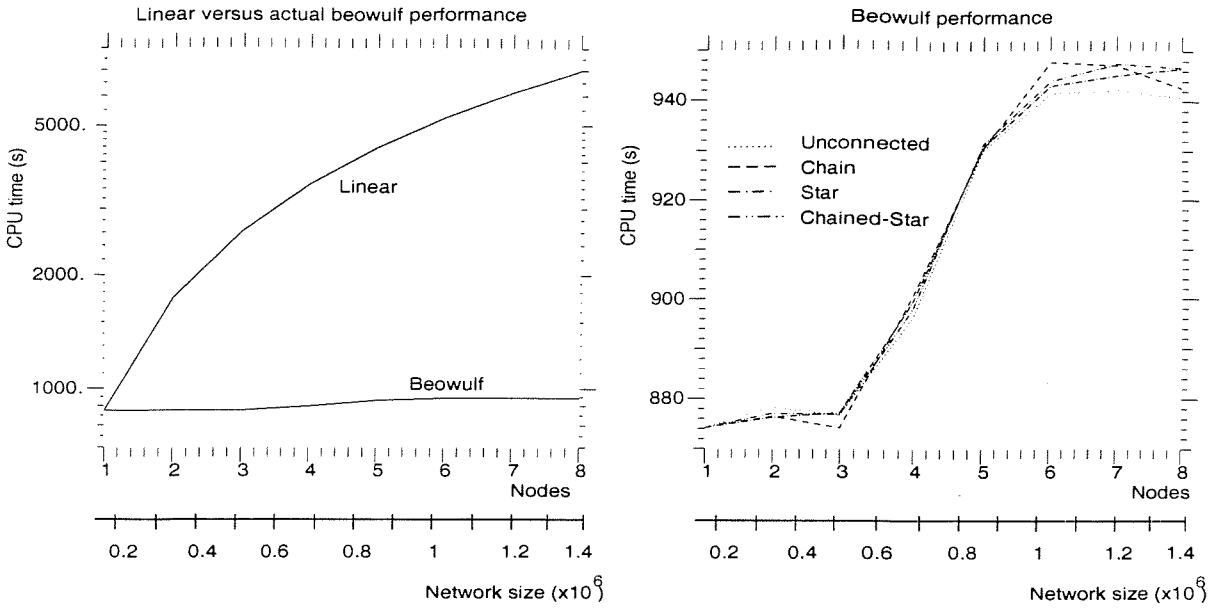


Figure 3: Elapsed time versus number of nodes and network size, A - Beowulf averaged over tested topologies compared with linear estimation, B - Beowulf for various network topologies

The flat profile of the Beowulf system indicates that, within the measured range of 1-8 nodes, network size can be increased with nearly constant elapsed times. Quantification of Beowulf results is possible with figure 3-B, which shows the elapsed time for the four network layouts tested. Considering the shortest (874.14 s) and longest simulations (947.4 s), an 8-fold increase in network size (from 1 to 8 nodes) results in a mere 8.3% in elapsed time in the worst case.

The low overhead incurred by the migration from single node to Beowulf distributed processing results from the low communication requirements when compared to the computation part. Further, the used inter-node bandwidth represents a small fraction of the available bandwidth. The measured average size of an inter-node packet carrying the contents of the spike buffer described in the previous section was 8915.76 bytes (2227.94 spikes $\times$ 4 bytes per spike + 4 bytes header). The number of packets travelling through the switch during a 1 s simulation was measured to be $C \times 10^3$, where C is the number of inter-node unidirectional channels (arrows in figure 1-B) in the topology under test. For instance, 28000 packets were transmitted for the 8 node chained-star network which results from 28 inter-node channels and 1000 time slices of 1 ms per simulated second. It follows that the total amount of data communicated between nodes throughout the entire simulation was approximately 28000 $\times$ 10 Kb, 280 Mbytes.

This corresponds to 320 Kbytes/s (considering an elapsed time of 874.14 s), which is well below the approximately 40 Mbytes/s of available bandwidth (estimated with in-house benchmarking tools).

6 Conclusions

This paper has presented preliminary results concerning the scalability of a message-based event-driven framework for biologically motivated neural simulation on Beowulf architectures. The experiments carried out with an 8-node Beowulf indicate that the migration from a single node to this parallel environment results in an 8-fold increase in aggregate size with an 8.3% increase in elapsed time; the total size of the distributed aggregate reached 10^6 neurons with an average of 179 synapses per cell.

Further tests are needed with Beowulfs in excess of 8 nodes to explore the scalability to larger simulations. Nevertheless, the results already obtained with an 8-node cluster indicate that low communication overhead can be achieved with an event-driven framework, resulting in efficient scalability.

The cortical model used for the benchmarking purposes has been developed as part of ongoing research on the dynamics of the piriform olfactory cortex. This cortical region contains approximately 10^7 neurons with several thousand synapses per cell [16]. Further code optimization and an increase in the number of nodes promise to make such problem sizes tractable using clusters of commodity computers.

Acknowledgements : We thank Dr. P. Melas for his advice on parallelization. The Beowulf cluster was kindly provided by the PDC group of the ECS Dept., University of Southampton, UK. This project was funded by the Biotechnology and Biological Sciences Research Council (BBSRC) and the ECS Dept., University of Southampton, UK.

References

[1] Koch C. and Segev I., editors. *Methods in neural modeling: From ions to networks.* MIT Press, Cambridge, 1998. ISBN 0-262-11231-0.

[2] Brettle D. and Niebur E. Detailed parallel simulation of a biological neuronal network. *Computational Biology*, pages 31–43, winter 1994.

[3] Hammarlund P. and Ekeberg O. Large neural network simulations on multiple hardware platforms. *Journal of Computational Neuroscience*, 5:443–459, 1998.

[4] Fujimoto Y. and Akabane T. Massively parallel architectures for large scale neural network simulations. *IEEE Trans. on Neural Networks*, 3(6):876–888, 1992.

[5] Niebur E. and Brettle D. Efficient simulation of biological neural networks on massively parallel supercomputers with hypercube architecture. In *Advances in Neural Processing Systems*, volume 6, pages 904–910, 1993.

[6] Jahnke A., Roth U., and Klar H. Towards efficient hardware for spike-processing neural networks. In *Proc. World Congress on Neural Networks*, pages 460–463, 1995.

[7] Pytte E. and Grinstein G. Traub R.D. Cellular automaton models of the CA3 region of the hippocampus. *Network: Computation in neural systems*, 2(2):149–167, 1991.

[8] Sterling T., Salmon J., Becker D., and Savarese D. *How to build a Beowulf. A guide to the implementation and application of PC clusters.* Scientific and Engineering Computation Series. The MIT Press, January 1999. ISBN 0-262-69218-X.

[9] Sterling T., Savarese D., Becker D. J., Dorband J. E., Ranawake U. A., and Packer C. V. Beowulf : A parallel workstation for scientific computation. In Banerjee P., editor, *Proc. of the International Conference on Parallel Processing, Vol.1: Architecture*, pages 11–14, Boca Raton, USA, August 1995. CRC Press.

[10] Sterling T., Cwik T., Becker D., Salmon J., Warren M., and Nitzberg B. An assessment of Beowulf-class computing for NASA requirements: Initial findings from the first NASA

workshop on beowulf-class clustered computing. In *Proceedings IEEE Aerospace Conference*, March 1998.

[11] Orellana C.J.G., Aligue F.J.L., Velasco H.M.G., Macias M.M., and Sotoca M.I.A. Large neural net simulation under beowulf-like systems. *Lecture Notes in Computer Science*, 1607:30–39, 1999.

[12] Wilson M. and Bower J.M. Cortical oscillations and temporal interactions in a computer simulation of piriform cortex. *Journal of Neurophys.*, 67(4):981–995, 1992.

[13] Nicholas J. Nevin. The performance of LAM 6.0 and MPICH 1.0.12 on a workstation cluster. Technical Report OSC-TR-1996-4, Ohio Supercomputing Center, Columbus, Ohio, 1996.

[14] Gropp W., Lusk E., and Skjellum A. *Using MPI. Portable Parallel Programming with the Message Passing Interface*. MIT Press, Cambridge, 1999. ISBN 0-262-57132-3.

[15] Das S.R. Adaptive protocols for parallel discrete event simulation. *J. of Operational Research Society*, 51(4):385–394, 2000.

[16] Harberly L.B. and Feig S.L. Structure of the piriform cortex of the opossum. I. Fine structure of cell bodies and neuropil. *J. of Comp. Neurology*, 216:69–88, 1983.

Bibliography

[1] Ramón y Cajal S. Conexión general de los elementos nerviosos. *Medicina Práctica*, 1:479–487, 1889. (In Spanish).

[2] Bekkers J., Bookman R., Delay M., Finkel A., Fox A., Gallegos D., Griffiths R., Hilgemann D., Kramer R., Lester H., Levis R., Levitan E., McKay M., Perkel D., Thompson S., Rae J., White M., and Wonderlin W. The Axon Guide. Technical report, Axon Instruments Inc., 1993.

[3] Grinvald A. Real-time optical mapping of neuronal activity: From single growth cones to the intact brain. *Ann. Rev. Neurosci.*, 8:263–305, 1985.

[4] Maher M.P., Pine J., Wright J., and Tai Y. The neurochip: a new multielectrode device for stimulating and recording from cultured neurons. *J. of Neurosc. Methods*, 87:45–56, 1999.

[5] Potter S.M., Mart A.N., and Pine J. High-speed CCD movie camera with random pixel selection for neurobiology research. In Paisley D.L., editor, *Proc. SPIE Vol. 2869*, pages 243–253, Bellingham WA, 1997. SPIE Press.

[6] Koch C. and Segev I., editors. *Methods in neural modeling: From ions to networks*. MIT Press, Cambridge, 1998. ISBN 0-262-11231-0.

[7] Koch C. *Biophysics of Computation: Information processing in single neurons*, chapter 14. Oxford University Press, New York, 1999. ISBN 0-195-10491-9.

[8] Maex R. and deSchutter E. Synchronization of Golgi and Granule cell firing in a detailed network model of the cerebellar Granule cell layer. *J. of Neurophys.*, 80(5):2521–2537, 1998.

[9] Hines M.L. and Carnevale N.T. The NEURON simulation environment. *Neural Computation*, 9(6):1179–1209, 1997.

[10] Bennet B.S. *Simulation fundamentals.* Prentice Hall, London, 1995. ISBN 0-13-813262-3.

[11] West R.M.E. and Wilcox G.L. A renumbering method to decrease matrix banding in equations describing branched neuron-like structures. *Journal of Neurosc. Methods*, 68(1):15–19, 1996.

[12] Traub R.D.and Wong R.K.S., Miles R., and Michelson H. A model of a CA3 hippocampal pyramidal neuron incorporating voltage-clamp data on intrinsic conductances. *Journal of Neurophys.*, 66:635–650, 1991.

[13] Pytte E. and Grinstein G. Traub R.D. Cellular automaton models of the CA3 region of the hippocampus. *Network: Computation in neural systems*, 2(2):149–167, 1991.

[14] Litovski V. and Zwolinski M. *VLSI. Circuit simulation and optimization.* Chapman & Hall, London, 1996. ISBN 0-412-63860-6.

[15] Destexhe A., Mainen Z.F., and Sejnowski T.J. Synthesis of models for excitable membranes, synaptic transmission and neuromodulation, using a common kinetic formalism. *J. of Computational Neuroscience*, 1:195–230, 1994.

[16] deSchutter E. and Bower J.M. An active membrane model of cerebellar purkinje cell i. simulation of current clamps in slice. *J. of Neurophys.*, 80(2):504–519, 1998.

[17] Destexhe A., Contreras D., and Sejnowski T.J.and Steriade M. A model of spindle rhythmicity in the isolated thalamic reticular nucleus. *J. of Neurophys.*, 72:808–818, 1994.

[18] Hammarlund P. and Ekeberg O. Large neural network simulations on multiple hardware platforms. *Journal of Computational Neuroscience*, 5:443–459, 1998.

[19] Abbot L.F. and Kepler T.B. Model neurons: from Hodgkin-Huxley to Hopfield. In Garrido L., editor, *Statistical mechanics of neural networks*, pages 5–18, Berlin, 1990. Springer. ISBN 3-540-53267-6.

[20] Segev I. Single neurone models: oversimple, complex and reduced. *Trends in Neurosciences*, 15:414–421, 1992.

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

[22] Rinzel J. and Rall W. Transient response in a dendritic neuron model for current injected at one branch. *Biophys. J.*, 14:759–790, 1974.

[23] Rall W. Core conductor theory and cable properties of neurons. In Kandel E.R., editor, *Handbook of Physiology.*, pages 39–97. American Physiological Society, 1977. ISBN 0-195-20658-4.

[24] Rall W. Electrophysiology of a dendritic model. *Biophys. J.*, 2:145–167, 1962.

[25] Hille B. *Ionic channels of excitable membranes.* Sinauer, Mass, 1992. ISBN 0-878-93323-9.

[26] Borg-Graham L.J. Interpretations of data and mechanisms for hippocampal pyramidal cell models. In Ulinski P.S., Jones E.G., and Peters A., editors, *Cerebral cortex, Volume 13: Cortical Models.* Plenum Press, New York, 1998.

[27] Kandel E.R., Schwartz J.H., and Jessell T.M. *Essentials of neural science and behavior.* Prentice Hall, 1996. ISBN 0-838-52245-9.

[28] Wilson M. and Bower J.M. Cortical oscillations and temporal interactions in a computer simulation of piriform cortex. *Journal of Neurophys.*, 67(4):981–995, 1992.

[29] Kepler T.B., Abbott L.F., and Marder E. Reduction of conductance-based neuron models. *Biological Cybernetics*, 66:381–387, 1992.

[30] FitzHugh R. Impulses and physiological states in models of nerve membrane. *Biophys. J.*, 1:445–466, 1961.

[31] Labbi A., Milanese R., and Bosch H. Gray-level object segmentation with network of Fitzhugh-Nagumo oscillators. *Lecture Notes in Comp. Sc.*, 1240:1075–1084, 1997.

[32] Liu F., Yamaguchi Y., and Shimizu H. Flexible vowel recognition by the generation of dynamic coherence in oscillator neural networks. *Biol. Cybern.*, 71(2):105–114, 1994.

[33] Rall W. and Shepher M. Theoretical reconstruction of field potentials and dendrodendritic synaptic interactions in olfactory bulb. *J. of Neurophys.*, 31:884–915, 1968.

[34] Destexhe A. Conductance-based integrate and fire models. *Neural Computation*, 9:503–514, 1997.

[35] Rolls E.T. and Treves A., editors. *Neural networks and brain function.* Oxford University Press, Oxford, 1998.

[36] Christodoulou G., Bugmann G., and Clarkson T.G. The temporal noisy-leaky integrator neuron model. In Beale R. and Plumbley M.D., editors, *Recent advances in neural networks.* Prentice Hall, 1993.

[37] Smith L.S. A one-dimensional frequency map implemented using a network of integrate-and-fire neurons. In *Proceedings of the 8th International Conference on Artificial Neural Networks*, pages 991–996. Springer, 1998.

[38] Smith L.S., Nischwitz A., and Cairns D.E. Synchronization of integrate-and-fire neurons with delayed inhibitory lateral connections. In Marinaro M. and Morasso P.G., editors, *Proceedings of ICANN94*, pages 142–145. Springer-Verlag, 1994.

[39] Hill S.L. and Villa A.E.P. Dynamic transitions in global network activity influenced by the balance of excitation and inhibition. *Network: Computtation in neural systems*, 8:165–184, 1997.

[40] Engel A.K., Kong P., and Singer W. Direct physiological evidence for scene segmentation by temporal coding. *Proc. Natl. Acad. Sci. USA*, 88:9136–99140, 1991.

[41] Eckhorn R. Dynamic bindings by real neurons: Arguments from physiology, neural network models and information theory. *Behavioral and Brain Sciences*, 16:457–458, 1993.

[42] Eckhorn R. Oscillatory and non-oscillatory synchronizations in the visual cortex and their possible roles in associations of visual features. In Pelt J. van, Corner M.A. Uylings H.B.M., and Lopes de Silva F.H., editors, *Progress in brain research.* Elsevier Science, 1994.

[43] Nischwitz A. and Gluender H. Local lateral inhibition: a key to spike synchronization? *Biol. Cybern.*, 73(5):389–400, 1995.

[44] Thorpe S.J. Face processing using one spike per neurone. *Biosystems*, 48:229–239, 1998.

[45] Thorpe S.J. and Gautrais J. Rapid visual processing using spike analysis. In Jornal M., editor, *NIPS*, pages 901–907, Cambridge, 1997. MIT Press.

[46] Wright J.J. Simulation of EEG: dynamic changes in synaptic efficacy, cerebral rythms, and dissipative and generative activity in cortex. *Biol. Cybern.*, 81:131–147, 1999.

[47] Wicks S.R., Roehrig C.J., and Rankin C.H. A dynamic network simulation of the nematode tap withdrawal circuit: predictions concerning synaptic function using behavioral criteria. *J. of Neurosc.*, 16(2):4017–4031, 1996.

[48] McCulloch W.S. and Pitts W. A logical calculus of the ideas immanent in nervous activity. *Bull. of Math. Biophysics*, 5:115–133, 1943.

[49] Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.*, 65:384–408, 1958.

[50] Sole R.V. and Manrubia S.C. *Orden y caos en sistemas complejos.* Ediciones UPC, Barcelona, 1996. ISBN 84-89636-10-9 (in Spanish).

[51] Hinton G.E. and Sejnowski T.J. Optimal perceptual inference. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 71, Washington, 1983.

[52] Robinson D.A. Implications of neural networks for how we think about the brain function. *Brain and Behavioral Sciences*, 15(4):644–655, 1992.

[53] Rumelhart D.E., Hinton G.E., and Williams R.J. Learning internal representations by error propagation. In Rumelhart D. E. and McClelland J. L., editors, *Explorations in the Microstructure of Cognition*, Parallel Distributed Processing. MIT Press, Cambridge, 1986.

[54] Hopfield J.J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci.*, 81:3088–3092, 1984.

[55] Freeman J.A. and Skapura D.M., editors. *Neural networks : Algorithms, applications and programming techniques.* Addison-Wesley, 1991. ISBN 0201513765.

[56] Hecht-Nielsen R. *Neurocomputing.* Addison-Wesley, 1997. ISBN 0201093553.

[57] Ermentrout G.B. and Edelstein-Keshet L. Cellular automata approaches to biological modeling. *J. Theor. Biol.*, 160(1):97–133, 1993.

[58] H. Axelrad, C. Bernard, B. Giraud, and M. E. Marc. Cerebellar cortex - simulation of the recurrent collateral inhibition by means of a neuronal automata assembly .1. Model and global analysis. *Comptes Rendus de LÁcademie des Sciences Serie III-Sciences De La Vie-Life Sciences*, 301:465 et seq., 1985.

[59] B. Giraud, C. Bernard, and H. Axelrad. Cerebellar cortex - simulation of the recurrent collateral inhibition of purkinje-cells by means of a neuronal automata assembly .2. Temporal-modulation and informational content. *Comptes Rendus de LÁcademie des Sciences Serie III-Sciences De La Vie-Life Sciences*, 301:565–570, 1985.

[60] Sleigh J.W. and Galletly D.C. A model of the electrocortical effects of general anaesthesia. *Brit. J. Anaesth.*, 78(3):260–263, 1997.

[61] Korkin M., Nawa N. E., and deGaris H. Spike interval information coding representation for ATR's CAM-brain machine (CBM). *Lecture Notes in Computer Science*, 1478:256–267, 1998.

[62] Gers F., deGaris H., and Korkin M. Codi-1bit: A simplified cellular automata based neuron model. *Lecture Notes in Computer Science*, 1363:315–333, 1998.

[63] Luthi P. O., Chopard B., Preiss A., and Ramsden J. J. A cellular automaton model for neurogenesis in drosophila. *Physica D*, 118:151–160, 1998.

[64] Eddi F., Mariani J., and Waysand G. Transient synaptic redundancy in the cerebellum and isostatic random stacking of hard spheres. *Biol. Cybern.*, 74(2):139–146, 1996.

[65] Siregar P., Sinteff J.P., Julen N., and LeBeux P. An interactive 3D anisotropic cellular automata model of the heart. *Computers and Biomedical Research*, 31:323–347, 1998.

[66] M. Bower and D. Beeman, editors. *The Book of Genesis.* TELOS/Springer-Verlag, 1998. ISBN 3540940197.

[67] Borg-Graham L. Additional efficient computation of branched nerve equations: adaptive time step and ideal voltage clamp. *J. Comput. Neurosc.*, 8(3):209–226, 2000.

[68] Bergdoll S. and Koch U.T. Biosim - A biological neural network simulator for research and teaching, featuring interactive graphical user interface amd learning capabilities. *Neurocomputing*, 8:93–112, 1995.

[69] Ekeberg O. A combined neuronal and mechanical model of the fish swimming. *Biological Cybernetics*, 69:363–374, 1993.

[70] Brettle D. and Niebur E. Detailed parallel simulation of a biological neuronal network. *Computational Biology*, pages 31–43, winter 1994.

[71] Maass W. and Bishop C., editors. *Pulsed neural networks.* The MIT Press, Cambridge, 1998. ISBN 0-262-13350-4.

[72] Fujimoto Y. and Akabane T. Massively parallel architectures for large scale neural network simulations. *IEEE Trans. on Neural Networks*, 3(6):876–888, 1992.

[73] Hammerstrom D. and Rehfuss S. Neurocomputing hardware - present and future. *Artificial Intelligence Rev.*, 7(5):285–300, 1993.

[74] Moller R. and Paschke P. Simulation of cortex-like neural networks on a CNAPS SIMD neurocomputer. *Neural Processing Letters*, 4(2):67–74, 1996.

[75] Goddard N., Hood G., Howell F., Hines M., and deSchutter E. NEOSIM: Portable large-scale plug-and-play neuronal modelling. Proceedings of the CNS2000, Computational Neuroscience: Trends in Research 2000, in press.

[76] Mansharamani R. An overview of discrete event simulation methodologies and implementation. *Sadhana*, 22(5):611–627, 1997.

[77] Brown R. Calendar queues: A fast O(1) priority queue implementation for the simulation event set problem. *Communications of the ACM*, 31(10):1220–1227, 1988.

[78] Jones D.W. An empirical comparison of priority-queue and event-set implementations. *Communications of the ACM*, 29(4):300–311, 1990.

[79] Ronngren R. and Ayani R. A comparative study of parallel and sequential priority queue algorithms. *ACM Trans. on Modeling and Computer Simulation*, 7(2):157–209, 1997.

[80] Vaucher J.G. and Duval P. A comparison of simulation event list algorithms. *Communications of the ACM*, 18(4):223–230, 1975.

[81] Bagrodia R.L. A message-based approach to discrete-event simulation. *IEEE Trans. on software engineering*, 13(6):654–665, 1987.

[82] Kuffler S.W., Nicholls J.G., and Martin A.R. *From neuron to brain*, chapter 18. Sinauer Assoc. Inc. Pub., 5th edition, 1984. ISBN 0878934448.

[83] Brodfuehrer P.D., Debski E.A., O'Gara B.A., and Friesen W.O. Neuronal control of leech swimming. *J. of Neurobiol.*, 27:403–418, 1995.

[84] Lockery S.R. and Kristan W.B. Distributed processing of sensory information in the leech. II. identification of interneurons contributing to the local bending reflex. *J. of Neurosc.*, 10:1816–1829, 1990.

[85] Cacciatore T.W., Brodfuehrer P.D., Gonzalez J.E., Jiang T., Adams S.R., Tsien R.Y., Kristan W.B. Jr, and Kleinfeld D. Identification of neural circuits by imaging coherent electrical activity with fret-based dyes. *Neuron*, 23:449–459, 1999.

[86] Synchronized oscillatory activity in leech neurons induced by calcium channel blockers. Angstadt j.d. and friesen w.o. *J. of Neurophysiol.*, 66:1958–1873, 1991.

[87] Huerta R., Sanchez-Montanes M.A., Corbacho F., and Siguenza J.A. A central pattern generator to control a pyloric-based system. *Biol. Cybern.*, 82(1):85–94, 2000.

[88] Grillner S. Neural networks for vertebrate locomotion. *Scientific Americam*, January:48–53, 1996.

[89] Grillner S., Wallen P., and Brodin L. Neural network generating locomotor behavior in lamprey: circuitry, transmitters, membrane properties, and simulation. *Annu. Rev. Neurosc.*, 14:169–199, 1991.

[90] Eckhorn R., Bauer R., Jordan W., Brosch M., Kruse W., Munk M., and Reitboeck H.J. Coherent oscillations: A mechanism of feature linking in the visual cortex. *Biol. Cybern.*, 60:121–130, 1988.

[91] Anderson J.C., Binzegger T., Kahana O., Martin K.A., and Segev I. Dendritic asymmetry cannot account for directional responses of neurons in visual cortex. *Nature Neurosc.*, 2:820–824, 1999.

[92] Swindale N.V. How many maps are there in visual cortex? *Cereb. Cortex*, 10:633–643, 2000.

[93] Mazza M.B., dePinho M., and Roque A.C. Biologically plausible models of topographic map formation in the somatosensory and auditory cortices. *Int. J. Neural Syst.*, 9:265–271, 1999.

[94] Riddle D.L., Blumenthal T., Meyer B.J., and Priess J.R., editors. *C. elegans II.* Cold Spring Harbor Laboratory Press, 1997. ISBN 0879694882.

[95] Niebur E. and Erdös P. Theory of the locomotion of nematodes: Dynamics of undulatory progession on a surface. *Biophys. J.*, 60:1132–1146, 1991.

[96] White J.G., Southgate E., Thomson J.N., and Brenner S. The structure of the ventral nerve cord of C. elegans. *Phil. Trans. R. Soc. Lond. B*, 275:327–348, 1976.

[97] White J.G., Southgate E., Thomson J.N., and Brenner S. The structure of the nervous system of Caenorhabditis elegans. *Phil. Trans. R. Soc. Lond. [Biol]*, 314:1–340, 1986.

[98] Ware R.R., Clark D., Crossland K., and Russell R.L. The nerve ring of the nematode C. elegans: sensory input and motor output. *J. Comp. Neurol.*, 162:71–110, 1975.

[99] Albertson D.G. and Thomson J.N. The pharynx of Caenorhabditis elegans. *Phil. Trans. R. Soc. Lond. B*, 275:299–325, 1976.

[100] White J.G., Southgate E., and Thomson J.N. Mutations in the C. elegans unc-4 gene alter the synaptic input to ventral cord motor neurons. *Nature*, 355:838–841, 1992.

[101] McIntire S.L., Jorgensen E., and Kaplan J. Horvitz H.R. The GABAergic nervous system of Caenorhabditis elegans. *Nature*, 364:337–341, 1993.

[102] Jin Y., Jorgensen E., Hartwieg E., and Horvitz H.R. The C. elegans gene unc-25 encodes glutamic acid decarboxylase and is required for synaptic transmission but not synaptic development. *J. of Neurosc.*, 19(2):539–548, 1999.

[103] Wicks S.R. and Rankin C.H. Integration of mechanosensory stimuli in Caenorhabditis elegans. *J. of Neurosc.*, 15(3):2434–2444, 1995.

[104] Wicks S.R. and Rankin C.H. The integration of antagonistic reflexes revealed by laser ablation of identified neurons determines habituation kinetics of the C. elegans tab withdrawal response. *J. Comp. Physiol.*, 179:675–685, 1996.

[105] Alberts B., Bray D., Lewis J., Raff M., Roberts K., and Watson J.D. *Molecular Biology of the Cell.* Garland Publishing Inc., New York, 1994. ISBN 0815316194.

[106] Richmond J.E. and Jorgensen E.M. One GABA and two acetylcholine receptors function at the C. elegans neuromuscular junction. *Nature Neuroscience*, 2(9):791–797, 1999.

[107] Goodman M.B., Hall D.B., Avery L., and S.R. Lockery. Active currents regulate sensitivity and dynamic range in C. elegans neurons. *Neuron*, 20:763–772, 1998.

[108] Niebur E. and Erdös P. Theory of the locomotion of nematodes: control of the somatic motor neurons by interneurons. *Mathematical Biosciences*, 118:51–82, 1993.

[109] Davis R.E. and Stretton A.O.W. The motornervous system of Ascaris: electrophysiology and anatomy of the neurons and their control by neuromodulators. *Parasitology*, 113:97–117, 1996.

[110] Walrond J.P. and Stretton A.O. Excitatory and inhibitory activity in the dorsal musculature of the nematode Ascaris evoked by single dorsal excitatory motorneurons. *J. of Neurosc.*, 5(1):16–22, 1985.

[111] Kaplan J.M. Sensory signaling in C. elegans. *Curr. Opin. Neurobiol.*, 6(4):494–499, 1996.

[112] Ferrée T.C. and Lockery S.R. Chemotaxis control by linear recurrent networks. In Bower J., editor, *Computational Neuroscience: Trends in Research.* Plenum Press, 1998.

[113] Chalfie M., Sulston J.E., Southgate E., and Thomson J.N. The neural circuit for touch sensitivity in C. elegans. *J. of Neurosc.*, 5:956–964, 1985.

[114] Mori I. and Ohshima Y. Neural regulation of thermotaxis in C. elegans. *Nature*, 376:344–348, 1995.

[115] Freeman W.J. Relations between unit activity and evoked potentials in prepyriform cortex in cats. *J. of Neurophys.*, 31:337–348, 1968.

[116] Freeman W.J. The physiology of perception. *American Scientific*, 264(2):78–85, 1991.

[117] Ketchum K.L. and Haberly L.B. Membrane currents evoked by afferent fiber stimulation in rat piriform cortex i. current source-density analysis. *J. of Neurophys.*, 69(1):248–260, 1993.

[118] Harberly L.B. and Feig S.L. Structure of the piriform cortex of the opossum. I. Fine structure of cell bodies and neuropil. *J. of Comp. Neurology*, 216:69–88, 1983.

[119] Li Z. and Hertz J. Odor recognition and segmentation by coupled olfactory bulb and cortical networks. *Neurocomputing*, 26-27:789–794, 1999.

[120] Harberly L.B.and Behan M. Structure of the piriform cortex of the opossum. III. Ultrastructural characterization of synaptic terminals of association and olfactory bulb afferent fibers. *J. of Comp. Neurology*, 219:448–460, 1983.

[121] Harberly L.B. Structure of the piriform cortex of the opossum. II. Description of the neuron types with Golgi methods. *J. of Comp. Neurology*, 213:163–187, 1983.

[122] Harberly L.B. and Bower J.M. Olfactory cortex: model circuit for study of associative memory? *TINS*, 12(7):258–264, 1989.

[123] Barkai E., Bergman R.E., Horwitz G., and Hasselmo M.E. Modulation of associative memory function in a biophysical simulation of rat piriform cortex. *J. of Neurophys.*, 72(2):659–677, 1994.

[124] Ketchum K.L. and Haberly L.B. Synaptic events that generate fast oscillations in piriform cortex. *J. of Neurosc.*, 13(9):3980–3985, 1993.

[125] Curtis M., Takashima I., and Iijima T. Optical recording of cortical activity after in vitro perfusion of cerebral arteries with a voltage-sensitive dye. *Brain Res.*, 837:314–319, 1999.

[126] Litaudon P., Datiche F., and Cattarelli M. Optical recording of the rat piriform cortex activity. *Progress in Neurobiol.*, 52:485–510, 1997.

[127] Freeman W.J. Simulation of chaotic eeg patterns with a dynamic model of olfactory system. *Biol. Cybern.*, 56:139–150, 1987.

[128] Hopfield J. Neural networks and physical systems with emergent computational habilities. *Proc. Natl. Acad. Sci. USA*, 79:2554, 1982.

[129] Ambros-Ingerson J., Granger R., and Lynch G. Simulation of paleocortex performs hierarchical clustering. *Science*, 247:1344–1348, 1990.

[130] Barnard E. Analysis of the Lynch-Granger model of olfactory cortex. *Biol. Cybern.*, 62:151–155, 1989.

[131] Wilson M. and J. Bower. A computer simulation of olfactory cortex with functional implications for storage and retrieval of olfactory information. In Anderson D.Z., editor, *Neural Information Processing Systems*, pages 114–126. America Institute of Physics, New York, 1988.

[132] Nunez P.L. *Electric fields of the brain: the neurophysics of EEG.* Oxford Univ. Press, New York, 1981. ISBN 0195027965.

[133] Hasselmo M.E. and Barkai E. Cholinergic modulation of activity-dependent synaptic plasticity in the piriform cortex and associative memory function in a network biophysical simulation. *J. of Neurosc.*, 15(10):6592–6604, 1995.

[134] Ballain T., Litaudon P., Martiel J., and Cattarelli M. Role of the net architecture in piriform cortex activity: analysis by a mathematical model. *Biol. Cybern.*, 79(4):323–336, 1998.

[135] Rapp M., Yarom Y., and Seggev I. Modeling back propagating action potentials in weakly excitable dendrites of neocortical pyramidal cells. *Proc. Natl. Acad. Sci. USA*, 93:11985–11990, 1996.

[136] Yuste R. and Tank W. Dendritic integration in mammalian neurons, a century after Cajal. *Neuron*, 16:701–716, 1996.

[137] Gerstner W. Spiking neurons. In Mass W. and Bishop C., editors, *Pulsed Neural Networks*, chapter 1. The MIT Press, Cambridge, 1998. ISBN 0-262-13350-4.

[138] Gerstner W. Time structure of the activity in neural network models. *Physical Rev. E*, 51:738–758, 1995.

[139] Shadlen M.N. and Newsome W.T. Noise, neural codes and cortical organization. *Current Opinion in Neurobiology*, 4:569–579, 1994.

[140] Shors T.J. and Matzel L.D. Long-term potentiation: what's learning got to do with it? *Behavioral and Brain Sciences*, 20(4):597–614, 1997.

[141] Madison D.V. and Nicoll R.A. Control of repetitive discharges of rat CA1 pyramidal neurons in vitro. *J. Physiol.*, 354:319–331, 1984.

[142] Munro D.H. Using the Yorick Interpreted Language. *Computers in Physics*, 9(6):609–615, 1995.

[143] Gontmakher S. and Horn I. Efficient memory allocation. *Dr. Dobb's Journal*, pages 116–119, January 1990.

[144] Bolte J.P., Fisher J.A., and Ernst D.H. An object-oriented, message-based environment for integrating continuous, event-driven and knowledge-based simulation. In *Application of advanced information technologies: Effective management of natural resources.* ASAE. June 18-19, Spokane, WA, 1993.

[145] Bagrodia R.L., Chandy K.M, and Misra J. A message-based approach to discrete-event simulation. *IEEE Trans. on Soft. Eng.*, 13(6):654–665, 1987.

[146] Brown A.D., Nichols K.G., and Zwolinski M. Issues in the design of a logic simulator: an improved caching technique for event-queue management. *IEE Proc. Circuits Devices Syst.*, 142(5):293–298, 1995.

[147] Ben-Ari Y., Cherubini E., Corradetti R., and Gaiarsa J.L. Giant synaptic potentials in immature rat CA3 hippocampal neurones. *J. of Physiol.*, 416:303–325, 1989.

[148] Hall D.H. and Russell R.L. The posterior nervous system of the nematode Caenorhabditis elegans: serial reconstruction of identified neurons and complete pattern of synaptic interactions. *J. of Neurosc.*, 11(1):1–22, 1991.

[149] Walrond J.P. and Stretton A.O. Reciprocal inhibition in the motor nervous system of the nematode Ascaris: direct control of ventral inhibitory motorneurons by dorsal excitatory motornuerons. *J. of Neurosc.*, 5(1):9–15, 1985.

[150] Yu X., Nguyen B., and Friesen W.O. Sensory feedback can coordinate the swimming activity of the leech. *J. of Neurosc.*, 19:4634–4643, 1999.

[151] Tavernakaris N. and Driscoll M. Molecular modeling of mechanotransduction in the nematode Caenorhabditis elegans. *Ann. Rev. Physiol.*, 59:659–689, 1997.

[152] Rankin C.H. Interactions between two antagonistic reflexes in the nematode C. elegans. *J. Comp. Physiol.*, 169:59–67, 1991.

[153] Rand J.B. and Nonet M.L. Neurotransmitter assignment for specific neurons. In Riddle D.L., Blumenthal T., Meyer B.J., and Priess J.R., editors, *C. elegans II.* Cold Spring Harbor Laboratory Press, 1997. ISBN 0879694882.

[154] Vanier M.C. and Bower J.M. A comparative survey of automated parameter-search methods for compartmental neural models. *J. of Computational Neurosc.*, 7(2):149–171, 1999.

[155] Klee M. and Rall W. Computed potentials of cortically arranged populations of neurons. *J. of Neurophys.*, 40:647–666, 1977.

[156] Press W.H., Vetterling W.T., Teukolsky S.A., and Flannery B.P. *Numerical recipes in C*, chapter 13. Cambridge Univ. Press, 2th edition, 1993. ISBN 0521431085.

[157] Ketchum K.L. and Haberly L.B. Membrane currents evoked by afferent fiber stimulation in rat piriform cortex ii. analysis with a system model. *J. of Neurophys.*, 69(1):261–281, 1993.

[158] Bressler S.L. Spatial organization of EEGs from olfactory bulb and cortex. *Electroencephalogr. Clin. Neurophysiol*, 57:270–276, 1984.

[159] Derrida B. and Stauffer D. Phase transitions in two-dimensional Kauffman cellular automata. *Europhysics Letters*, 2(10):739–745, 1986.

[160] Stauffer D. Random boolean networks: analogy with percolation. *Philosophical Magazine B*, 56(6):901–916, 1987.

[161] Dongarra J.J. Performance of various computers using standard linear equations software. Technical Report BS 1629, University of Tennessee, 2000.

[162] Niebur E. and Brettle D. Efficient simulation of biological neural networks on massively parallel supercomputers with hypercube architecture. In *Advances in Neural Processing Systems*, volume 6, pages 904–910, 1993.

[163] Jahnke A., Roth U., and Klar H. Towards efficient hardware for spike-processing neural networks. In *Proc. World Congress on Neural Networks*, pages 460–463, 1995.

[164] Sterling T., Salmon J., Becker D., and Savarese D. *How to build a Beowulf. A guide to the implementation and application of PC clusters.* Scientific and Engineering Computation Series. The MIT Press, January 1999. ISBN 0-262-69218-X.

[165] Sterling T., Savarese D., Becker D. J., Dorband J. E., Ranawake U. A., and Packer C. V. Beowulf : A parallel workstation for scientific computation. In Banerjee P., editor, *Proc. of the International Conference on Parallel Processing, Vol.1: Architecture*, pages 11–14, Boca Raton, USA, August 1995. CRC Press.

[166] Sterling T., Cwik T., Becker D., Salmon J., Warren M., and Nitzberg B. An assessment of Beowulf-class computing for NASA requirements: Initial findings from the first NASA workshop on beowulf-class clustered computing. In *Proceedings IEEE Aerospace Conference*, March 1998.

[167] Nicholas J. Nevin. The performance of LAM 6.0 and MPICH 1.0.12 on a workstation cluster. Technical Report OSC-TR-1996-4, Ohio Supercomputing Center, Columbus, Ohio, 1996.

[168] Gropp W., Lusk E., and Skjellum A. *Using MPI. Portable Parallel Programming with the Message Passing Interface.* MIT Press, Cambridge, 1999. ISBN 0-262-57132-3.

[169] Das S.R. Adaptive protocols for parallel discrete event simulation. *J. of Operational Research Society*, 51(4):385–394, 2000.

[170] Hebb D.O. *The organization of behavior.* Wiley, New York, 1949.

[171] Maher M.P, Pine J., Wright J., Wright J., and Tai Y. The neurochip: a new multielectrode device for stimulating and recording from cultured neurons. *J. of Neurosc. Methods*, 87:45–56, 1999.