Design of artificial neural oscillatory circuits for the control of lamprey- and salamander-like locomotion using evolutionary algorithms

Auke Jan Ijspeert

Ph.D.
University of Edinburgh
1998

Abstract

This dissertation investigates the evolutionary design of oscillatory artificial neural networks for the control of animal-like locomotion. It is inspired by the neural organisation of locomotor circuitries in vertebrates, and explores in particular the control of undulatory swimming and walking. The difficulty with designing such controllers is to find mechanisms which can transform commands concerning the direction and the speed of motion into the multiple rhythmic signals sent to the multiple actuators typically involved in animal-like locomotion. In vertebrates, such control mechanisms are provided by *central pattern generators* which are neural circuits capable of producing the patterns of oscillations necessary for locomotion without oscillatory input from higher control centres or from sensory feedback. This thesis explores the space of possible neural configurations for the control of undulatory locomotion, and addresses the problem of how biologically plausible neural controllers can be automatically generated.

Evolutionary algorithms are used to design connectionist models of central pattern generators for the motion of simulated lampreys and salamanders. This work is inspired by Ekeberg's neuronal and mechanical simulation of the lamprey [Ekeberg 93]. The first part of the thesis consists of developing alternative neural controllers for a similar mechanical simulation. Using a genetic algorithm and an incremental approach, a variety of controllers other than the biological configuration are successfully developed which can control swimming with at least the same efficiency. The same method is then used to generate synaptic weights for a controller which has the observed biological connectivity in order to illustrate how the genetic algorithm could be used for developing neurobiological models. Biologically plausible controllers are evolved which better fit physiological observations than Ekeberg's hand-crafted model. Finally, in collaboration with Jérôme Kodjabachian, swimming controllers are designed using a developmental encoding scheme, in which developmental programs are evolved which determine how neurons divide and get connected to each other on a two-dimensional substrate.

The second part of this dissertation examines the control of salamander-like swimming and trotting. Salamanders swim like lampreys but, on the ground, they switch to a trotting gait in which the trunk performs a standing wave with the nodes at the girdles. Little is known about the locomotion circuitry of the salamander, but neurobiologists have hypothesised that it is based on a lamprey-like organisation. A mechanical simulation of a salamander-like animat is developed, and neural controllers capable of exhibiting the two types of gaits are evolved. The controllers are made of two neural oscillators projecting to the limb motoneurons and to lamprey-like trunk circuitry. By modulating the tonic input applied to the networks, the type of gait, the speed and the direction of motion can be varied.

By developing neural controllers for lamprey- and salamander-like locomotion, this thesis provides insights into the biological control of undulatory swimming and walking, and shows how evolutionary algorithms can be used for developing neurobiological models and for generating neural controllers for locomotion. Such a method could potentially be used for designing controllers for swimming or walking robots, for instance.

Acknowledgements

I am very grateful to John Hallam and David Willshaw for their guidance and encouragements all along this project. I gratefully acknowledge their warm encouragements to transform an initial self-proposed MSc project into this doctoral work. They proved to be perfect supervisors, giving me a large amount of freedom while always being present for critical and useful comments.

Many thanks to the members of the Mobile Robots Group for making the lab a lively place. Special thanks to Richard Reeve for discussions on CPGs and fruitful comments on several of my papers.

Many thanks also to Jérôme Kodjabachian for the nice and fruitful time we had at the AnimatLab (Paris), and to Jean-Arcady Meyer for gracefully inviting me there. They, and the rest of the members of the Animatlab, made my stay in Paris very pleasant.

Warmest thanks to Aude for these wonderful three years spent together in Edinburgh and for sharing all the joys, difficulties, excitations, ... of our doctoral works. She made the long hours spent in the lab very enjoyable.

Finally a great thanks to my parents. I am greatly indebted to and enormously grateful for their constant support and multiple encouragements.

I gratefully acknowledge financial support from the Swiss National Research Fund and from the European Commission in the form of a grant for young researcher and a Marie Curie Fellowship, respectively.

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Auke Jan Ijspeert
Edinburgh
February 15, 1999

Contents

List of Figures

Chapter 1

Introduction

1.1 General overview

This thesis presents the development of neural controllers for locomotion using evolutionary algorithms. My approach is inspired by biological studies of animal locomotion, and takes inspiration from biology at three levels: 1) in the type of locomotion — swimming and walking; 2) in the type of control mechanisms — neuronal circuits; and 3) in the type of adaptation mechanisms which have led to animals — evolution and development. In particular, I investigate the anguiliform swimming of the lamprey and the swimming and walking of salamanders, and how to control these gaits using connectionist models with the organisation observed in vertebrates. My tools for this investigation are numerical simulation and evolutionary algorithms. Simulation is used to calculate the activity of networks of neurons modeled as leaky integrators and to model simple mechanical bodies made of rigid links and simple muscles. The evolutionary algorithms are used to design the configurations of the neural controllers for locomotion. Given a fixed body structure, neural controllers are evolved which can produce the patterns of oscillations necessary for locomotion, and which can modulate the speed and the direction of motion of the simulated body when simple inputs signals are varied.

The animat field

This work follows the *animat* approach to the development of autonomous agents. An animat is an autonomous agent which, similarly to animals, has to move, act and survive in an environment [Meyer & Guillot 94]. Although its long term objective is to be able to create systems with the cognitive capacities of humans, the animat approach is bottom-up, starting from simple survival skills. It takes inspiration from how animals function and behave, and gives a large importance to the sensing-acting loop within a dynamical environment.

Computational neuroethology

More precisely, this thesis is included in *computational neuroethology* [Beer 90, Cliff 95] (or also *synthetic psychology* [Braitenberg 84]). Computational neuroethology is a special field in the animat approach. It has the same aims as the animat approach, i.e. the study and creation of agents which present an adaptive behaviour, and studies specifically how this behaviour results from neural mechanisms. It is therefore very close to neuroethology, which is concerned with how animal behaviour is rooted in neuronal circuitries, and to computational neuroscience, which tries to decode and to simulate the neural mechanisms of the central nervous system found in animals. However, computational neuroethology differs from these two fields in that it not only studies (existing) animals but autonomous agents in general, where autonomous agents can be described as self-governing systems capable of operating (i.e. perceiving and acting) in environments which are complex, uncertain and dynamic [Cliff 95]. Therefore, computational neuroethology has not only a scientific or analytic aspect —the understanding of existing mechanisms— but also an engineering or synthetic aspect —the creation of artifical systems. It is particularly interesting when these two aspects interact, as creating artificial systems allows one to gain a good understanding of what are the necessary properties of a neural mechanism for a specific behaviour, and studying existing systems in animals allows one to study what mechanisms are sufficient (although not necessarily minimal) for a specific behaviour. The artificial systems can be either simulated and integrated in a simulated environment or physically, in the form of robots. This situatedness into an environment is important as having a complete agent-environment system lessens the chances of making untenable assumptions concerning issues of representation and processing [Cliff 95].

Animal-like locomotion

In this thesis, I study one particular aspect of neural control which is the control of locomotion. As mentioned, I shall study and take inspiration from examples found in nature both in the types of locomotion, such as swimming and walking, and in the types of control mechanisms, such as *central pattern generators*. Locomotion is a fundamental skill for animals as they need it for finding food, for encountering a mate for reproduction, for escaping predators, for moving to a more friendly environment,... For robotics as well, the capacity to move efficiently is essential, if we want robots to do something useful. Using wheeled robots has greatly simplified the task of the control of locomotion as it means that the direction and the speed of the motion is determined by only a few actuators, typically two. However, as wheeled robots are strongly limited in the kind of environments in which they can move, some engineers have turned to biological systems for inspiration and started to develop robots which move using more animal-like types of gaits. This means that more complex means of locomotion than powered wheels are needed, involving a greater number of actuators generally used in a rhythmic way. Very quickly, the engineer is then faced with the same control problems faced by biological systems, namely the control of multiple actuators which only produce the desired behaviour when appropriately coordinated. The problem is to find the right control mechanism which can translate commands concerning the speed and direction of motion into the set of signals sent to the multiple actuators. I therefore believe that robotics may not only gain from inspiration from biology for the structure of the robot (e.g. legged robots) but also for its control system (i.e. networks of neurons). Note that there is no reason that the structure and control systems found in nature should be optimal (Darwinian evolution just requires them to be good enough), but even the simplest animal is for the moment much more efficient and better adapted to its environment than the most complex robot.

Central pattern generators

The neural mechanisms used by animals to control complex motions are remarkably well adapted; they present several interesting features such as distributed control, flexibility, robustness against lesions, *etc.*. I take inspiration from the oscillatory neural circuitries found in vertebrates. Oscillatory networks play a very important role in animals, as they are used for many fundamental functions such as locomotion, breathing, chewing, blood circulation and even thinking. Many of these networks are based on circuits called *central pattern generators* (CPG) which can produce oscillations without oscillatory input either from the brain or from sensory feedback. CPGs are an interesting way to distribute control. They are located close to the muscles they control — the CPGs for the locomotion for vertebrates are, for instance, located in the spinal cord — and they produce, when receiving tonic input, the basic patterns of oscillation necessary for the function they control. These patterns form the templates for motion which can then be modulated by higher control and by sensory feedback depending on the external conditions.

1.2 Content and motivations of the thesis

In this dissertation, neural controllers for the swimming of the lamprey and the swimming and trotting of the salamander are developed using evolutionary algorithms. The thesis can be seen as an experiment in computational neuroethology with strong links with neurobiology. In a first part, it is inspired by findings of neurobiologists on the swimming circuitry of the lamprey, and develops alternative controllers using evolutionary algorithms. By doing so, I evaluate the quality of evolutionary algorithms as a design technique and investigate alternative neural configurations to the one found in the lamprey. In the second part, I develop potential controllers for the swimming and trotting of salamanders, whose locomotion circuitry has not been decoded yet. The dissertation therefore not only investigates a design method for locomotion controllers which could be useful for neurobiologists and roboticists, but also develops circuitries which, when analyzed, may give some insights into the functioning of biological networks. The tools in this investigation are simulation of the neural circuitries and the simplified mechanical bodies, and evolutionary algorithms.

1.2.1 The lamprey

A lamprey swims by creating a lateral undulation of its body which travels from head to tail. Because of its relative simplicity and because the isolated spinal cord can produce patterns of oscillation very similar to those of the intact animal (*fictive swimming*), the CPG of the lamprey has been studied in great detail by neurobiologists and it is one of the best known vertebrate CPGs. Physiological observations have allowed neurobiologists to make a model of the neural circuitry of the CPG whose capacity to produce most of the observed patterns of oscillations has been demonstrated through simulations. The lamprey constitutes an interesting starting point for this dissertation. It has the advantages of being one of the few animals whose CPG has been modeled in detail and of having a locomotion gait which is simple while presenting the basic features of any natural motion control, namely the creation and the modulation of sets of oscillations with characteristic frequencies and phase lags.

In this thesis, the existing models of the swimming CPG of the lamprey are studied, and alternative connectionist controllers are generated using evolutionary algorithms. There are several motivations for developing these alternative controllers. The first motivation is to evaluate whether evolutionary algorithms can be useful tools for developing connectionist locomotion controllers. Having one example of swimming controller, the biological model, is then very useful because it ensures that at least one solution exists in the search space of all possible solutions, and because it gives an example with which the performance of the evolved controllers can be compared. Another interest is to study whether there exist other solutions than that found in nature that can control swimming with the same efficiency. Studying the variety of solutions evolved may give indirect information on the characteristics needed for making good CPGs in general. Finally, because — in the case of Ekeberg's model —the CPG has been hand-crafted to fit physiological measurements, it is probably possible to slightly change Ekeberg's values to fit the measurements even better. In this thesis, I show that a GA could be a very useful tool for making neurobiological models when it is used to instantiate variables such as synaptic strengths which are difficult to measure. From a more general point of view, especially a robotics point of view, the technique of evolving swimming controllers is naturally attractive. Swimming is a more efficient way to move in the water than using propellers. Having a method which could automatically generate a swimming controller for any given body shape would therefore be very useful.

1.2.2 The salamander

This thesis also considers animals which can both swim and walk. Salamanders, for instance, swim like lampreys by creating a traveling wave along their trunk. On the ground, they switch to a trotting gait in which the trunk performs a standing wave with the nodes at the level of the girdles. Such a gait increases the reach of the limbs which are attached laterally to the trunk. Although neurobiologists have not yet decoded the CPG for locomotion of the salamander, they have hypothesised that, as salamanders have evolved from simpler vertebrates like the lamprey, their locomotor circuitries have a similar organisation. The main question is then to understand which kind of neural circuitry can produce both traveling waves for swimming and standing waves for trotting. In this thesis, based on lamprey-like swimming circuitries, neural controllers are evolved which can exhibit both types of gaits. Similarly to what has been hypothesised by neurobiologists, the controllers consist of two neural oscillators which project to limb motoneurons and to a lamprey-like trunk CPG.

The first motivation behind this experiment is to gain a better understanding of which kind of neural circuitries could exhibit the two gaits, and to develop potential controllers which could be compared with the actual circuitry of the salamander when it is finally decoded. The second motivation is that an autonomous agent that could both walk and swim would be able to move over most the surface of the earth. An amphibian robot could have numerous applications, and therefore being able to generate a controller which can produce the right patterns of oscillations for walking and swimming, and modulate them to vary the speed and direction of the locomotion would be most interesting.

1.2.3 Tools

Neuronal simulation

In the field of artificial neural networks and computational neuroscience, simulation has always been the main tool of investigation. Because networks of neurons (be it perceptron-like neurons or realistic biophysical models of neurons) usually form unsolvable sets of equations, numerical tools are used to calculate their activity.[1] As with any simulation of a biological neuronal circuit, one has to choose at which level of abstraction to simulate the system. Biologically-based neural networks can be simulated at a biophysical, a connectionist or a more abstract level like chains of oscillators. Biophysical models simulate neurons relatively realistically, taking into account the physical properties of neurons and therefore modeling dendritic and somatic compartments with ion channels. Connectionist models capture the main feature of neurons, that is, their ability to change the frequency of action potential spikes in their axon (the output) depending on the sum of the activity in their dendrites (the inputs). Finally some CPGs, like that of the lamprey, have been modelled as chains of abstract oscillators.

The connectionist level is chosen for this thesis. A connectionist model has the advantages of being abstract enough to be tractable, in the sense that the simulation of the CPG and of the body (see below) does not require too much computation time and too many parameter choices. At the same time, it is concrete enough to give some insights into the functioning of the CPG of the real lamprey and salamander. Finally, the connectionist models of the lamprey's CPG are very similar to those currently used for the control of some robots [Lewis *et al.* 93]. Therefore, learning how to design them may be directly applicable to the design of a controller for a swimming robot, for instance.

Mechanical simulation

Simulation is also used for representing a simplified mechanical body of a lamprey and of a salamander, in interaction with water or with the ground. I believe that it is important, especially in the study of neural controllers for locomotion, to investigate

[1] Alternatively, networks of neurons can be encoded directly into electronic circuits.

the effect of the neural activity on a mechanical body rather than simply analysing the neural activity. Analysing the motion of a mechanical body gives a direct evaluation of how good the neural controller is, since the production of efficient motion requires taking in account the inherent dynamics of the mechanical system and producing motoneuron signals with the right phases and amplitudes. It is also the only way to study the effect of sensory feedback on the controller. And finally, it allows one to integrate the CPGs in a more complete animat in order to study higher level behaviours (and make experiments in synthetic neuroethology). The main reason for my using simulation rather than a real robot is the absence of physical robots on which the CPGs could be tested. In addition, simulation has several interesting properties which make it a preferable choice, at least initially, for this thesis. Simulation gives a good control of all parameters involved which allows the experimenter to make perfectly reproductible experiments. This also allows (relatively) easy variation of the conditions under which experiments are made. Also, like any experiment in evolutionary robotics, it has the advantage of not risking any material damage of a physical robot due to wear or to bad controllers which are inevitably created in an evolutionary process. Note that, in contrast with many experiments in evolutionary robotics, simulation is not used to shorten the time to test robots as the time for running a mechanically realistic simulation with current computational power is not necessary shorter than that required by a real robot for evaluation of a controller.

Evolutionary algorithms

I use evolutionary algorithms, in particular *genetic algorithms* (GAs) and *genetic programming* (GPs), for the design of the neural controllers. These optimization algorithms are inspired by the process of Darwinian evolution found in Nature. The idea is to encode potential solutions to a problem into *chromosomes*, which are evaluated and given a *fitness value* depending on their capacity to solve the problem. New solutions are created by "breeding" the chromosomes depending on their fitness. The fitness of the population, i.e. the capacity of the solutions to solve the problem, increases gradually because of *selective pressure* applied to the population due to the selection of parent chromosomes and the rejection of the worst solutions at each generation. From a general point of view, design by evolutionary algorithms has the interesting property of moving the design process from the classical engineering approach where all the building bricks of a solution have to be specified, to a process which only needs the definition of basic building bricks (the representation) and of the problem (the evaluation function) and then uses the evolutionary algorithm to construct the system until the problem is satisfactorily solved. Applied to neural networks, evolutionary algorithms present a great flexibility that most traditional learning algorithms do not offer. This point will be discussed in more detail in the following chapters. Note that the evolutionary algorithms are used as design tools and not as simulations of natural evolution.

1.2.4 Original contribution

This thesis is not revolutionary, but rather a set of small new steps in different directions, in the relatively new field which is computational neuroethology. To the best of my knowledge, the original contributions of the thesis are the following:

- evolution of connectionist controllers for anguiliform swimming,
- study and development of neural controllers for the lamprey which are alternative to the biological circuitry,
- development of connectionist controllers which can exhibit both the swimming and the trotting gait of the salamander,
- mechanical simulation of a salamander-like animat.

The term connectionist is here important because the development of controllers for lamprey- and salamander-like locomotion and their application to robotics has been studied before by Lewis, but at a more abstract level [Lewis 96]. As we will see in the next chapter, Lewis developed an abstract representation of CPGs called Adaptive Ring Rules. He used this representation to develop "C" bending for a simulated lamprey, and hand-crafted a controller for the walking of a robot with a one-joint flexible spine. Although I also take inspiration from lampreys and salamanders, this work differs from Lewis' in the following ways: 1) it takes more inspiration from neurobiology, 2) it is situated at a connectionist level with networks of neurons rather than abstract structures, 3) it uses a mechanical simulation of the salamander with a trunk flexible at multiple points and which can both swim and walk, 4) it uses evolutionary algorithms rather than self-organising rules as design tool, and 5) it develops controllers which can modulate the speed and the direction of the locomotion (features which Lewis did not address).

Finally, this thesis is also among the first examples of the use of evolutionary algorithms for designing neurobiological models. In the next chapter (section 2.7), some other recent work will be presented in which evolutionary algorithms are used for setting variables in neurobiological models.

1.2.5 Summary of the objectives

The objectives of this thesis can be summarized as a series of questions I will try to answer:

- How can neural networks be used to control locomotion, and what can we learn from the neural circuitry found in vertebrates?
- How efficient are evolutionary algorithms for the design of neural controllers?
- Can evolutionary algorithms be used as design tools in neurobiology?
- What kinds of neural circuitry can produce the undulatory swimming of lampreys? In particular, are there alternative neural configurations to those found in Nature which can control locomotion with at least the same efficiency?
- How can controllers for undulatory swimming be extended to control both the swimming and the trotting of a salamander-like animat?

1.3 Summary of the thesis

Chapter 2: As I am interested in taking inspiration from animals, a significant part of the thesis is dedicated to the analysis of the mechanisms of control found in animals. I look in particular at the neural circuitries controlling the swimming of the lamprey and at the studies made of the locomotion of salamanders. Related research in the artificial neural network community is then presented, and an overview of biologically inspired swimming and walking machines is made. The mutual benefits of interactions between biology and artificial intelligence are discussed.

Chapter 3: After having made a review of the different models of the lamprey's CPG in chapter 2, this chapter presents in more detail the neuronal and mechanical simulation of the lamprey developed by Ekeberg [Ekeberg 93]. This connectionist model is an important inspiration of this thesis, and it is the model with which the evolved swimming controllers will be compared. I reproduce that model with both the neural and the mechanical simulation. The different properties of the model are described.

Chapter 4: This chapter presents the development of alternative controllers to the biological model using a GA. The alternative controllers are based on similar neurons to those of Ekeberg, although their sign (excitatory or inhibitory) can change. In this first research, a direct encoding and an incremental approach for developing swimming controllers are used. Solutions are developed in three stages, with first the evolution of segmental oscillators, then the evolution of the coupling between 100 copies of a segmental oscillator, and finally the evolution of sensory feedback from stretch sensitive cells.

Chapter 5: This chapter demonstrates how the GA can be used as a tool for neurobiological modeling, when used to instantiate variables which are difficult to measure. To illustrate this, the evolutions of swimming controllers are repeated and the possible solutions are restricted to solutions which present the observed biological connectivity. The evolutionary process is therefore used to automatically generate a part of the model Ekeberg has designed by hand.

Chapter 6: In order to test a more sophisticated encoding, I started a collaboration with Jérôme Kodjabachian to try the developmental encoding he designed. The idea is to evolve developmental programs which determine how neurons divide and grow connections on a 2-dimensional substrate. For this we use simpler neurons than those used in earlier chapters and also fewer neural segments. Swimming controllers are evolved in only one stage, with a fitness function only based on mechanical aspects such as the speed of swimming and the capacity to turn.

Chapter 7: This chapter presents how swimming controllers for the lamprey can be extended to control both the swimming and the trotting of a salamander-like animat. The mechanical simulation is an extension of that of the lamprey with the addition of simplified limbs. Similarly to hypotheses of neurobiologists, controllers are evolved which are made of limb oscillators which project to a trunk CPG similar to that of the lamprey. The fitness function is only based on mechanical aspects such as the control of speed and of direction. The same direct encoding scheme as for the first experiment on the lamprey is used, and the evolution of the walking CPG can be seen as a fourth evolutionary stage following the evolution of the swimming controller.

Chapter 8: Finally, in order to study how the CPGs could be used by higher control levels, I carry out, with both the lamprey and the salamander, a preliminary experiment in which the command signals sent to the evolved CPGs are determined by a simple visual system composed of two retinae. The visual system and the connections from it to the CPGs are hand-coded for the animats to exhibit a tracking behaviour of a randomly moving target.

Chapter 9: The dissertation concludes with a general discussion of the results.

Chapter 2

Background

Fields such as neurobiology, neuroethology, cognitive science, artificial life, robotics, artificial neural networks, artificial intelligence, are progressively coming closer as they are pushing further their respective barriers, which little by little leads to increasing overlap between the different fields.

In this chapter, I give an overview of the research carried out in these different fields on locomotion and its control. I start from biology and little by little move towards artificial intelligence — more specifically artificial neural networks and robotics —, and close the loop by looking at how artificial intelligence may help biology and *vice versa.*

First, some general considerations on animal locomotion are reviewed. Then the organization of locomotion control in animals and, in particular, the concept of *central pattern generation* is presented (section 2.2). As an example of locomotion controlled by central pattern generator, I summarize neurobiological findings on the control of the swimming in lampreys (section 2.3). Lampreys have been studied extensively by neurobiologists, whose research has inspired this thesis. The locomotion of salamanders is also presented. I then give an overview of the research carried out in the artificial neural network community on dynamical neural networks (section 2.5). This type of network has many similarities with connectionist models used in neurobiological modeling, and I especially look at the kind of design techniques developed in the ANN community. One of these techniques, evolutionary algorithms, is presented in more detail as it is the design method used in this thesis. In section 2.6, I give an overview of the research on locomotion carried out in the animat field, with a survey of swimming and legged robots. Section 2.6.2 presents different design methods for developing controllers for locomotion, especially neural controllers and evolutionary design. Finally, I present work which highlights the fruitful interaction between artificial intelligence and neurobiology with, in particular, the application of artificial intelligence techniques to neurobiology (section 2.7).

2.1 Animal-like locomotion: general considerations

A large variety of different types of locomotion have been developed in the animal kingdom, as animals adapted to different ecological niches. Animals, for instance, swim, fly, run, crawl, hop, *etc.* The different locomotion gaits are usually very well adapted to the environment of the animal, and animal locomotion presents fascinating agility and mechanical efficiency. These properties have made animal locomotion an interesting field to study by biologists and an interesting example to follow for robotics engineers.

Animal locomotion is characterized by a large number of actuators, a rhythmic activity, and the fact that efficient motion is only obtained when the actuators are well coordinated. This leads to several control problems. The main difficulty is how to transform commands concerning the speed and the direction of motion into the signals sent the different actuators. Such a transformation is significantly more difficult than for a powered two-wheeled robot, for instance, where a simple relation can be found between the voltages sent to the two motors and the direction and speed of motion. The difficulty arises from the fact that motion is obtained from the coordinated rhythmic activity of the different actuators. A signal sent to a single actuator has therefore a complex influence on the motion, as the effect of the signal depends on the state of the other actuators and of the state of the mechanical system which is controlled. The timing of the signals (i.e. the phase relation between the actuators) is crucial, especially for locomotion gaits which are not statically stable such as the gallop.

The dynamics of the mechanical system is also very important as it largely determines what movements are possible and it leads to delayed effects of a control signal. The control of locomotion has to take account (and take advantage) of the inherent dynamics of the mechanical system, such as its natural frequencies, its elasticity, *etc.* As Raibert puts it, "the mechanical system has a mind of its own, governed by the physical structure and the laws of physics. Rather than issuing commands, the nervous system can only make "suggestions" which are reconciled with the physics of the system and the task" [Raibert & Hodgins 93].

Finally some animals, such as cats, use different gaits (the walk, the trot and the gallop) for different speeds of motion and need therefore control mechanisms able to produce significantly different patterns of signals depending on the desired speed.

An important question in locomotion control is how to organize the control mechanism. For instance having a central control center responsible for coordinating and sending the signals to all individual actuators, or having a more distributed control with different centers responsible for a subgroup of actuators with some communication between the centers. Also the importance of sensory feedback must be considered, as it can either be used for refining the signals sent by a central control mechanism, or play a direct role in the generation of the signals.

These issues will be considered in the next sections. We will first consider the motor organization of animals, as far as it is known.

2.2 Motor organization in animals

2.2.1 Peripheral or central mechanism?

The locomotion of animals has been studied for a long time, but biologists had to wait for relatively modern techniques for gaining a better understanding of the different locomotion gaits and control mechanisms used by animals. These techniques include photography and video for the gait characterization, microscopes for the anatomy of the control circuitries, intracellular measuring techniques for the physiology of the circuitries, pharmacology for the understanding of the cellular properties, and more recently computer simulations for verifying hypotheses about the neural circuitries involved.

During this century, two theories have been proposed for the control mechanisms: *peripheral control* and *central control* [Delcomyn 80]. The peripheral control point of view is that the rhythmic sequences of actions linked with locomotion are due to chains of reflexes. Locomotor patterns could, for instance, be induced by a simple alternating signal, and signals from sensory feedback would then determine the complex sequence of actions necessary for motion, with each movement providing the trigger for the next one. Conversely, the central control point of view is that the central nervous system has the capacity for producing the patterns of oscillations in the absence of sensory feedback.

Although experiments in the '40s by Sir James Gray seemed to support peripheral control (as reported in [Delcomyn 80]), the current opinion is that most animals have a central control of rhythmic behaviour. Gray found out that sensory stimuli could initiate stepping in a toad whose spinal cord had been severed just behind the head, and that completely deafferented spinal cords of toads did not reveal any rhythmic activity. Gray therefore suggested that the stepping behaviour was due to a chain of reflexes. However more recent (and more careful) experiments contradicted his latter results by showing that completely deafferented toads were, in some cases, still able to produce rhythmic sequences and use their legs in the normal sequences, demonstrating therefore the existence of a central control mechanism. Similar oscillatory networks have nowadays been observed in many animals, both invertebrates and vertebrates, by making experiments on animals whose oscillatory circuits are isolated from sensory information[1] (see [Delcomyn 80] for a review). A central control mechanism does not mean, however that it does not integrate sensory feedback. Several experiments have, for instance, been done which demonstrate that sensory feedback plays an important role in "shaping" the output of the oscillatory circuits. In particular, sensory input can initiate oscillations in the control circuitry as demonstrated by Gray's first experiment. Such an experiment shows that sensory input plays an important role in the pattern generation, but it is not a proof that sensory feedback is *necessary* for the pattern generation.

[1] The sensory information is prevented to reach the oscillatory circuitries by either completely isolating the circuitry, or by cutting its afferent connections, i.e. deafferentating the circuitry, or by paralysing the animal, using curare for instance, which has the effect of eliminating movement-related feedback.

2.2.2 Central pattern generators

Once central control was commonly accepted by neurobiologists, another important question, especially for vertebrates, was to determine at which "level" the patterns of oscillations are generated and which kind of information has to travel from the brain to the motoneurons for the production of the locomotor sequence. In vertebrates, there is now strong evidence that the oscillatory circuits are located in the spinal cord. Completely isolated spinal cords of lampreys and frog embryos can for instance produce patterns of neural activity very similar to those used by the intact animal for swimming. Experiments by Shik, Orlovky and Severin in the 1960's (as reported by [Grillner 85, Grillner 96]) showed that the walking gait in a decerebrated cat could be induced by a very simple electric stimulus in a specific area of the brainstem. When the amplitude of the signal is increased, the speed of locomotion increases and the cat switches to the trot and to the gallop. Vertebrates have therefore circuits in their spinal cord which can produce the patterns of oscillations necessary for locomotion without oscillating input from the brainstem or from sensory feedback; such circuits are called *central pattern generators* (CPGs).

The characteristics of CPGs can be summarized as follows (see [Grillner 85, Getting 88, Cohen 88, Kleinfeld & Sompolinsky 89]):

1. CPGs can produce rhythmic neural output in the absence of sensory feedback from the muscles and structures controlled by the CPG and in the absence of control by higher neural centers.

2. Some CPGs function without a pacemaker cell, implying that the rhythmic output is a collective property of the network.

3. A CPG is capable of producing multiple patterns of rhythmic behaviours depending on the input to the CPG, with the same set of neurons being involved in different patterns.

4. Most locomotion CPGs are considered to be composed of coupled oscillators (or sub-CPGs). Tetrapods, for instance, have different CPGs for each limb, and probably different oscillators for each joint of a limb.

5. The output can be modulated by external output such as feedback from proprioceptors and from higher neural centers. For example modulation is used to both turn the CPG on and off and to control the period of its rhythm.

CPGs therefore provide the templates for the patterns of oscillations which can be shaped by sensory feedback or control commands from higher neural centers depending on the situation. This type of distributed control means that, during stationary locomotion (i.e. motion with constant speed and direction), only simple signals need to be sent to the CPG, carrying information about the speed of motion, for instance. It is only for specific, voluntary movements that more complex input signals are needed from the higher neural centers. A variety of CPGs have been identified, not only for locomotion, but also for breathing, swallowing, chewing, ..., both in invertebrates (see [Getting 88]) and in vertebrates (see [Grillner *et al.* 88, Cohen 88]). Among vertebrates, one of the most studied animals is the lamprey, which I shall present next.

2.3 Locomotion of lampreys

Lampreys are primitive fish (cyclostomes) which resemble eels, but which lack jaws. A lamprey's life cycle begins with a larval stage in the sand of rivers before living a parasitic life in the sea or in lakes. Lampreys have become famous by the nuisance they created to the fish population of the American Great Lakes, which they reached in the 1800s through shipping canals. Because of their simple swimming gait and because of their relatively simple control circuitries, lampreys have been extensively studied by neurobiologists for understanding the circuitries underlying vertebrate locomotion.

Figure 2.1: Picture of lamprey (New York State Department of Environmental Conservation).

Lampreys have an elongate body, with a smooth, scaleless skin. They have two dorsal fins, but no paired fins (Figure 2.1). A lamprey swims like an eel using an *anguiliform gait*, i.e. by propagating a traveling undulation of its body from head to tail. The undulation is lateral and propagates over the whole body with an approximately constant wavelength, which means that the whole body is activated for producing the swimming gait (Figure 2.2).

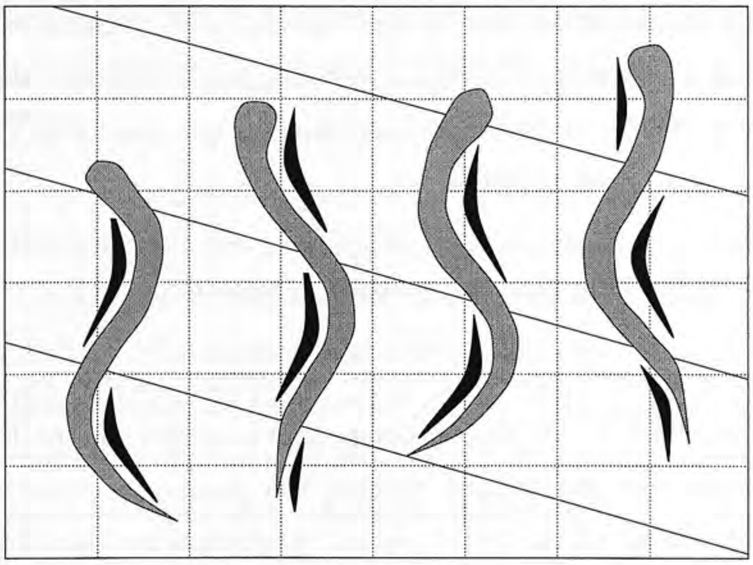


Figure 2.2: Anguiliform swimming of the lamprey. The traveling waves of muscle contraction are highlighted.

2.3.1 A good preparation for the study of vertebrate locomotion

The lamprey's locomotion circuitry is one of the most studied and well known vertebrate locomotor circuitries. It is a good subject for studying vertebrate locomotion for several reasons:

1. Its anatomy and type of locomotion are simple. The lamprey swims without using limbs and with only one type of gait, which makes it therefore significantly easier to analyse than tetrapods, for instance, which move by coordinating the muscular activity of several joints in the four limbs and which use several gaits (walking, trotting, galloping,..) depending on the speed of motion.

2. As it is one of the earliest vertebrates, its nervous system is relatively simple and

contains relatively few neurons compared to higher vertebrates, which makes the identification of the different cell types and their connectivity easier.[2]

3. Its isolated spinal cord can be maintained for several days in a glass dish, while producing patterns of oscillations which are almost identical to those of the intact animal. This allows *in vitro* cellular measurements of the locomotor circuitry.

4. Although simpler, the lamprey central nervous system presents an organization which has largely been conserved in more complex vertebrates. Findings on the locomotor circuitry of the lamprey may therefore give insights on the functioning of higher vertebrates as well.

2.3.2 The swimming central pattern generator

The circuitry which produces the neural oscillatory activity for swimming is located in the spinal cord. As mentioned above, studies on the isolated spinal cord have shown that the isolated circuitry can produce patterns of oscillations very similar to those of the intact animal, demonstrating that the circuitry is a CPG. This neural activity is called *fictive swimming*. I will here summarize neurobiological findings on the swimming CPG of the lamprey from papers including [Grillner *et al.* 88, Williams *et al.* 90, Grillner *et al.* 91, Ekeberg *et al.* 91, Matsushima & Grillner 92, Wallén *et al.* 92, Williams 92a, Traven *et al.* 93, Williams & Sigvardt 94, Grillner *et al.* 95, Sigvardt & Williams 96].

When the isolated spinal cord is immersed in an excitatory bath, oscillations of neural activity can be measured along the ventral roots of the spinal cord. The spinal cord is made of approximately 100 segments, with one ventral root per segment through which motoneurons project to the muscles (see below). Neural activity in each segment oscillates with contralateral sides out of phase, and each segment oscillates with a small phase lag compared to its rostral neighbour, which explains the rostro-caudal traveling undulation of the body. As the phase between 2 neighbour segments is approximately 1% of the oscillation period, the most rostral and caudal segments of the spinal cord

[2] The identification of cells and their connectivity is, however, more difficult that for invertebrates where single cells can be identified. In the lamprey, the identification concerns populations of functionally similar neurons, rather than single cells.

oscillate in phase, and the neural wave makes one complete wave over the spinal cord. An interesting property of the CPG is that it maintains, both in the intact animal and in the isolated spinal cord, a constant phase relation between segments even when the frequency of oscillation is varied. A lamprey changes its speed of swimming by changing the frequency of oscillation (between 0.25 and 10Hz), while maintaining a wavelength corresponding to the length of the body. The same behavior has been observed in the isolated spinal cord, where the phase lag between segments stays approximately constant while the frequency of oscillation is varied by changing the concentration of the excitatory bath.

As small parts (up to 2 segments) of any part of the spinal cord can be isolated and made to oscillate independently from neighbouring segments, it is believed that the whole circuit is made of an interconnection of segmental oscillators. It is not clear however, how distinct the oscillators are from each other. The spinal cord has distinct ventral roots through which the motoneurons project to the muscles, but the neurons in the spinal cord rather form a continuous column. In most models, the spinal cord is considered as a coupling of distinct neural oscillatory units.

Apart from motoneurons, three types of interneurons have been found to play a role in the generation of the swimming patterns. There are excitatory neurons which project to ipsilateral (i.e. on the same side) neurons (EIN), and two types of inhibitory neurons, one which projects ipsilaterally (LIN) and one which projects contralaterally (CIN). Based on anatomical studies, a segmental circuitry between these neurons has been proposed by Buchanan and Grillner [Buchanan & Grillner 87] as shown in Figure 2.3.

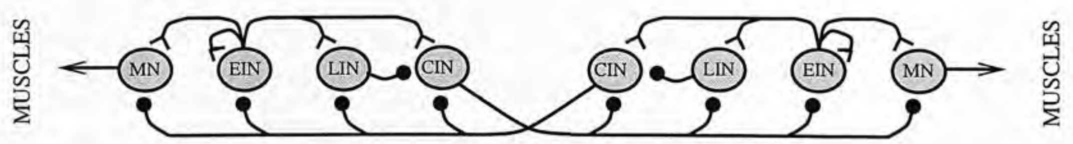


Figure 2.3: Segmental connectivity as proposed in [Buchanan & Grillner 87]. The lamprey's CPG is composed of four types of neurons: 3 types of interneurons EIN, CIN and LIN and the motoneurons MN. The neural units in the figure represent populations of neurons of the same type. The total number of neurons in a segment is approximately 1000.

The network is normally activated from the brainstem via reticularspinal neurons, but can also be activated experimentally by increasing the level of excitability pharmacologically in the isolated segments of the spinal cord. The behaviour of this segmental network is determined by the contralateral inhibition of the CIN neurons and burst terminating mechanisms which makes the neural activity switch from one side to the other. Once one side is slightly more active than the other, its activity will increase because of the ispilateral excitatory neurons, while the activity on the opposite side will lessen because of the contralateral inhibition of the CIN neurons. After a while a burst terminating mechanism stops the ipsilateral CIN activity, which therefore cease to inhibit the neurons on the opposite side. This means that the neurons on the other side become active, and among them the CIN neurons which will ensure that the activity on the initial side ceases completely. Half a cycle has then been performed, and the neural activity will similarly switch back and forth from one side to the other. Several burst terminating mechanisms have been proposed and I will review them below. One of them is the delayed activation of LIN neurons which could inhibit the ispilateral CIN (see Figure 2.3) late in the burst, allowing neurons on the other side to become active.

While the segmental connectivity has now been described in detail, the coupling of segments over the spinal cord is not perfectly known yet. A general observation is that when one type of neuron has connections to another type of neuron in the same segment, it can also project to corresponding neurons in neighbouring segments (*synaptic spread* [Williams 92a]). The projections vary from one type of neuron to another. EIN, for instance, project in both rostral and caudal directions for approximately 5 segments, LIN have long caudal projections to up to 50 segments and CIN mainly projects in the caudal direction to approximately 20 segments with some short rostral projections. There is, for the moment, very little information on the respective projections to different neurons types (for instance, whether CIN has similar projections to all neurons types or different projections depending on the postsynaptic cell type), and on the synaptic strength of the different projections.

Several hypotheses have been proposed for explaining the origin of the phase lags between segments. The phase lags are clearly not due to delays in conduction along descending axons. The fact that the phase lag does not vary with the frequency means indeed that the time delays between segments are kept at a fixed proportion of the cycle duration, and that they vary 40 fold when the frequency varies between 0.25 and 10 Hz. As conduction velocities along axons are more or less constant, conduction delays cannot be the origin of the phase lags.

Two other hypotheses are: a difference of the intrinsic frequencies of segments over the spinal cord; and a property of the coupling between segments. The difference of intrinsic frequencies is proposed by Sten Grillner in his *trailing oscillator* hypothesis [Matsushima & Grillner 92]. The idea is that some segments in the most rostral part of the spinal cord have a higher intrinsic frequency and therefore lead the other segments. As there is no evidence for systematic difference of intrinsic frequencies in isolated segments from a same spinal cord, it is proposed that higher intrinsic frequency is obtained through higher excitation from the brain stem. This hypothesis also explains how the lamprey can swim backwards, as it would correspond to providing more excitation to the most caudal segments. Experiments in which a spinal cord is maintained in a excitatory bath with different compartments and different concentrations shows that both caudally and rostrally directed waves can be obtained depending on the concentrations.

The other hypothesis is proposed by Thelma Williams and mathematicians modeling the spinal cord as a chain of oscillators [Williams *et al.* 90, Williams 92b](see also section 2.3.3) and comes from the observation that, on average, any part of the spinal cord oscillates with a caudally directed phase lag between segments while being submitted to a uniform excitatory bath. This tends to support the idea that the coupling has an asymmetrical functionality which leads to caudally directed waves.

Note that the two hypotheses are not mutually exclusive as one could imagine that the coupling between segments favours caudally directed waves, and that excitation from higher control centers determines the exact value of the phase lag, with the possibility to induce rostrally directed waves when the most caudal segments receive more excitation.

2.3.3 Models of the swimming circuitry

Based on anatomical and physiological findings, the swimming circuitry has been modeled at three levels of abstraction, namely at a biophysical, a connectionist and an abstract oscillator level. Biophysical models using relatively realistic neural models investigate whether the current state of knowledge of the lamprey is sufficient to produce models whose results agree with the physiological observations. Connectionist models of the CPG use less realistic neuron models which capture only the main feature of neurons, that is, their ability to change the frequency of action potential spikes in their axon (the output) depending on the sum of activity in their dendrites (the total input). These studies analyse the importance of the connectivity in the generation of swimming patterns. Finally, at the most abstract level, the swimming controller can be modeled as a chain of mathematical oscillators in order to study which kind of couplings can produce a phase relation between segments which is constant over the whole spinal cord and which remains a fixed proportion of a cycle when the frequency of the oscillations is changed.

Biophysical models

Several biophysical simulations based on the CPG's anatomy and physiology have been made, mainly by Sten Grillner and his colleagues at the Nobel Institute for Neurophysiology in Stockholm [Ekeberg *et al.* 91, Grillner *et al.* 91, Wallén *et al.* 92, Hellgren *et al.* 92, Traven *et al.* 93, Wadden *et al.* 97]. These simulations are composed of relatively realistic models of neurons of Hodgkin-Huxley type which are composed of several electrically coupled isopotential compartments, to represent the soma and the dendritic tree, equipped with ion channels.

These studies have progressively investigated the single cells implicated in the pattern generation [Ekeberg *et al.* 91], then the segmental circuitry [Grillner *et al.* 91, Wallén *et al.* 92, Hellgren *et al.* 92, Traven *et al.* 93], and have recently looked at the intersegmental coordination [Wadden *et al.* 97]. The simulations model experimentally established types of neurons with their specific membrane properties and synaptic interconnections. Except for [Hellgren *et al.* 92], they simplifiy the neural circuitry by representing whole populations of neurons of the same type by single neuron units. Simulations of the segmental circuitry as shown in Figure 2.3 demonstrated that it can produce the patterns of oscillations observed physiologically, and that several pharmacological experiments on the isolated spinal cord can be reproduced *in computo*. The simulations especially investigate the mechanisms of burst termination, i.e. the mechanisms which lead the neural activity to cease on one side of the spinal cord and to switch to the other.

Four possible burst terminating mechanisms have been proposed and investigated: 1) a frequency adaptation in CIN and EIN neurons, 2) the termination of the depolarized NMDA-evoked plateau in CIN and EIN neurons, 3) the synaptic inhibition from ipsilateral inhibitory neurons (LINs), and 4) the effect of sensory input from stretch sensitive cells. [Wallén *et al.* 92, Hellgren *et al.* 92] find that the first two (cellular) aspects play an important role for oscillations at low frequencies, while the synaptic inhibition from ipsilateral neurons (LINs) is mainly useful for high frequency oscillations. Similarly to physiological observations, they find out that, at low frequencies, NMDA receptors play an essential role in the rythmogenesis and evoke pacemaker-like membrane potential oscillations in individual neurons. [Hellgren *et al.* 92] also finds that duplicating the interneuron units of the segmental network, by using populations of network interneurons, leads to a more robust burst activity and covers a wider frequency range than simulation with single neuron units representing the interneurons.

In [Traven *et al.* 93] the effect of sensory input from stretch sensitive cells located on both sides of the spinal cord, the *edge cells*, is also investigated. The findings of experiments in which fictive swimming oscillations are entrained by moving the caudal end of the spinal cord from one side to the other with a manipulator [Grillner *et al.* 91], are reproduced by simulating rhythmic input from the stretch sensitive cells to the segmental network. Because of ipsilateral excitatory connections and contralateral inhibitory connections, the effect of the edge cells is to synchronize the neural activity with the movements of the body. The sensory input then acts as a burst terminating factor by inhibiting the activity on the contracting side and exciting the interneurons on the side about to become active.

[Wadden *et al.* 97] investigates the intersegmental coordination. Unlike the previous studies, this simulation includes more than one segmental network and models 60 segments of the spinal circuitry. The circuitry is also continuous in the sense that interneurons are spread out evenly along the length of the spinal cord with rostral and caudal projections which respect the biological projections as far as they are known. The simulation produces patterns of oscillations with phase lags between segments which are constant over most of the spinal cord. The model also reproduces effects of locally increasing the excitation in the spinal cord, similarly to the experiments reported in [Matsushima & Grillner 92]. However, unlike physiological observations, these simulations can not reproduce the independence between the phase lag and the frequency of oscillation, as, in these simulations, the phase lag varies with the frequency instead of maintaining a value correponding to 1% of the burst duration.

One problem with biophysical simulations is that they require the setting of many parameters defining the cellular and synaptic properties of the neurons. Many of these parameters can not be measured with current intracellular recording techniques, and have to be set by hand in order to fit the physiology of the neurons. As the number of unknown parameters grows rapidly with the number of neurons, biophysical simulations have difficulties to simulate large networks of neurons. The growing number of parameters which have to be artificially instantiated means that the larger the network, the less realistic the simulation and therefore the less motivation for using such a computationally expensive simulation. An interesting alternative for simulating networks of neurons is therefore to use connectionist models.

Connectionist models

Connectionist models investigate the dynamics of networks of neurons using less realistic neuron models than biophysical simulations. Rather than studying the detailed membrane activity of neurons, they study the effect of synaptic weights and synaptics delays on the overall activity of the network. These types of model are therefore very similar to dynamical recurrent neural networks from the artificial neural network community (see section 2.5).

The neuron model of connectionist simulations computes the mean firing rate of biological neurons (rather than spiking action potentials) depending on the synaptic input and depending on the neuron time constant(s). The mathematical models of neurons are either models in which the neuron activity is updated with discrete time steps depending on the current input and the previous activation, or differential equations representing leaky integrators. The important parameters of a network are the time constants of the neurons and the synaptic weights of the interconnections. The number of parameters is significantly lower than for biophysical simulations.

The lamprey's swimming circuitry has been modeled by different connectionist models [Buchanan 92, Williams 92a, Ferrar *et al.* 93, Ekeberg 93, Ekeberg *et al.* 95, Jung *et al.* 96]. These models simulate one or several segments of the swimming CPG, and are based on the segmental connectivity sketched in Figure 2.3. Because the neuron models do not include complicated neural mechanisms, the oscillations in these models are entirely due the synaptic connectivity (although the neuron model in [Ekeberg 93] includes a frequency adaptation mechanism). These models rely on the LIN neurons for acting as burst terminator, and they can therefore be considered to simulate the functioning of the biological network in the higher frequency range.

The simulations of [Buchanan 92, Williams 92a, Ekeberg 93, Jung *et al.* 96] demonstrate that the segmental connectivity is sufficient for producing oscillations, without the need for special neuronal properties such as action potentials and endogenous oscillations. The frequency of the oscillations can be increased by increasing the external excitation applied to the neurons, similarly to physiological experiments.

[Buchanan 92] and [Williams 92a] also study the effect of coupling different segmental oscillators. The couplings are obtained by projecting a segmental connection to the corresponding post-synaptic neuron in the neighbour segment. Different couplings, both unilateral and bilateral, are tested. Buchanan looks in particular at a variety of possible couplings between two oscillators which leads to stable phase locking between the two segments with one segment leading the other. Williams simulates a chain of 20 segments and uses a closest neighbour coupling in which each segmental connection is projected with a lower synaptic weight to the rostral and caudal neighbouring segments. Constant intersegmental phase lags can be obtained as long as the connections between oscillators are asymmetric. The neural activity travels from head to tail if the synaptic weights in the rostral direction are stronger than those in the caudal direction. Independence between frequency and phase lag is also obtained when the frequency is increased by increasing the tonic excitation the EIN cells alone. However, unlike physiological experiments, if the tonic excitation to all cells is increased, both the frequency and the phase lag are increased. She does not study the effect of varying the synaptic weight of the intersegmental coupling depending on the connection (e.g. increasing the inhibitory connections and decreasing the excitatory connections) or the effect of longer coupling than closest neighbour projections.

[Ferrar *et al.* 93] demonstrates that the current model of the segmental circuitry is remarkably robust against stochastic variations of the synaptic weights, and, similarly to [Hellgren *et al.* 92], that the network becomes even more robust when cells are duplicated.

[Ekeberg 93] presents a model combining a neural simulation of the whole spinal cord (100 segments) with a simple mechanical simulation of the body of the lamprey in interaction with water. This model has inspired our work on the lamprey and will be described in more detail in chapter 3. It uses more complex neuron models than the previous studies —integrators with three state variables which include features such as frequency adaptation. The model is able to produce swimming patterns similar to those observed in the lamprey, with a phase lag which is constant over the spinal cord and which is independent of the frequency. In [Ekeberg *et al.* 95], the effect of sensory feedback from the mechanical simulation is investigated with the simulation of the stretch sensitive edge cells. Ekeberg shows that these cells can be useful for crossing a simulated speed barrier. That paper also presents an extension of the CPG for controlling the swimming of a 3-dimensional simulation of a lamprey. An interesting aspect of Ekeberg's work is that, because both the neural circuit and a mechanical body are simulated, it enables study of control aspects such as the modulation of the speed and the direction of swimming.

Models using mathematical oscillators

The lamprey's swimming CPG can be studied at an even more abstract level than connectionist simulations, when regarded as a chain of mathematical oscillators [Rand *et al.* 88, Williams *et al.* 90, Kopell *et al.* 91, Williams 92b, Somers & Kopell 93, Nishii *et al.* 94a, Nishii *et al.* 94b, Kopell 95, Williams & Sigvardt 95, Sigvardt & Williams 96]. Such an approach is essentially taken to investigate the structure and the function of the intersegmental coupling. Each segment of the spinal cord is represented as a non-linear oscillator, without considering the cellular and connectivity of the segmental circuitry. The idea is to find general properties of the coupling between the oscillator units which can produce the swimming patterns of the lamprey, and which do not depend on the details of the individual oscillators.

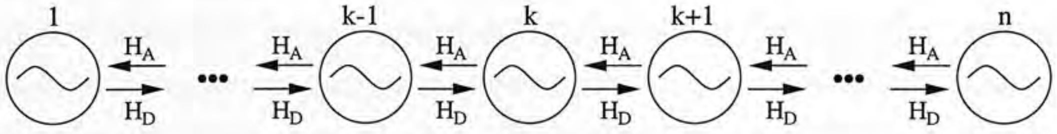


Figure 2.4: Chain of coupled oscillators

These studies use non-linear oscillators usually coupled with a closest neighbour coupling (Figure 2.4). Most of the mathematical framework is due to Nancy Kopell and Bard Ermentrout (see for instance [Ermentrout & Kopell 91, Kopell *et al.* 91]). All these studies rely on an *averaging technique*, which states that if the effect of one oscillator on the instantaneous frequency of another is averaged over the entire cycle, then this effect can be expressed as a function of only the phase difference between the two oscillators, rather than on the state of an oscillator at each time step. This averaging technique is shown to be valid if the coupling signals are dispersed around the cycle rather than occurring at only one or two points within the cycle [Ermentrout & Kopell 91], and following [Williams *et al.* 90], this can be considered true for the lamprey. The chain of oscillators is then mathematically described as:

$$\begin{array}{lcll} \Omega & = & \omega_1 + H_A(\phi_1) & k = 1 \\ \Omega & = & \omega_k + H_A(\phi_k) + H_D(-\phi_{k-1}) & 1 < k < n \\ \Omega & = & \omega_n + H_D(-\phi_{n-1}) & k = n \end{array} \qquad (2.1)$$

where ω_k is the intrinsic frequency of the oscillator k (i.e. the frequency at which it would oscillate when isolated), Ω is the ensemble frequency of the coupled oscillators (i.e. the frequency of all oscillators when they are phase-locked due to the coupling), ϕ_k is the phase difference between oscillator k and oscillator k-1, and $H_A(\phi)$ and $H_D(\phi)$ are the ascending and descending coupling functions respectively. Because of the averaging technique, the effect of one oscillator on the frequency of oscillation of the other depends only on the phase lag between them.

The aim is to find the charateristics of the coupling functions $H_A(\phi)$ and $H_D(\phi)$ which, similarly to the lamprey's spinal cord, can produce a phase lag between segments which is constant over the chain and which is independent of the ensemble frequency Ω of the oscillators. As there is no biological evidence that there is a systematic variation of the intrinsic frequencies of the oscillators along the spinal cord, most studies assume that all oscillators have the same intrinsic frequency, which means that the phase lags are due only to the coupling. [Williams *et al.* 90] gives a summary of the properties the coupling functions H_A and H_D should possess for the chain to produce lamprey-like oscillations. In particular, she points out that the value taken by the intersegmental phase lag ϕ_k is near the zero crossing of either the ascending or descending coupling (i.e. ϕ_A or ϕ_D where $H_A(\phi_A) = 0$ and $H_D(-\phi_D) = 0$). Which value it is near depends on the relative magnitudes of each coupling function at the zero crossing of the other. For instance, if $|H_A(\phi_D)| > |H_D(-\phi_A)|$, the chain will phase-lock with a phase lag of ϕ_A for most of the chain, and the ascending coupling H_A is then said to *dominate.*

The mathematical predictions have been compared with several biological experiments in order to determine which coupling, ascending or descending, dominates:

1. Experiments in which mechanical entrainment is applied to the extremities of the spinal cord, have shown that caudal forcing can induce a larger range of frequencies than rostral forcing [Williams *et al.* 90]. When compared to the mathematical framework, this indicates that the ascending coupling is dominant.

2. Analysis of the phase lags between segments over the spinal cord shows that the phase lag is constant for most of the spinal cord except at the rostral boundary [Williams & Sigvardt 94]. This is also evidence that the ascending coupling dominates.

3. When the spinal cord is immersed in an excitatory bath with two different concentrations between the rostral and the caudal half, the phase lag changes significantly within the rostral but not the caudal compartment, indicating a dominating ascending coupling [Sigvardt & Williams 96].

4. But in the same experiment, it is also observed that the ensemble frequency of the spinal cord with different activations in the rostral half and the caudal half is always closer to the intrinsic frequency of the rostral segments rather than the caudal ones [Sigvardt & Williams 96]. In the framework, this would mean that the descending coupling is dominant, which is opposite to the previous observations.

[Sigvardt & Williams 96] conclude that although the mathematical theory agrees with most of the biological observations, it should be extended to explain the last observation. The authors suggest introducing another type of coupling which affects an oscillator's frequency independently of its phase lag, and to include both short and long range couplings.

One problem with this mathematical model is that it does not give a clear relation between the meaning of dominance of a coupling and neural properties such as the synaptic strength, the length and the sign (inhibitory or excitatory) of connections between neurons. In the theory, a coupling is said to dominate when it gives the main contribution to the determination of the phase lag between segments. This does not mean that the neural wave should travel in the direction of the dominant coupling, i.e. an ascending coupling can potentially lead to both positive or negative phase lags between segments. Also, although it is clear that for a coupling in one direction to exist there must be synaptic connections in that direction, a dominant coupling does not necessary mean that the synaptic connections in that direction have stronger excitatory or inhibitory synapses or longer projections.

In an attempt to bring the mathematical theory closer to neural properties, [Williams 92a] presents a modification of the mathematical theory and provides a link between the mathematical theory and a connectionist model similar to that developed in [Buchanan 92]. In particular, she proposes a variant of the coupling function which can be measured from the interaction of two neural oscillators, and shows that the mathematical chain of oscillators correctly predicts the behaviour of the simulated connectionist model. Further work in that direction should provide a better understanding of the relation between the mathematical coupling functions and the synaptic properties of a simulated or biological neural network.

2.4 Locomotion of salamanders

Salamanders (and newts) are legged amphibians which are capable of swimming and walking. This property makes an interesting inspiration for robotics, as a salamander-like amphibian robot would be able to move in a large variety of environments.

Salamanders swim like lampreys using an anguiliform swimming gait and, on the ground, they move with a trotting gait in which the body performs a standing wave, rather than the travelling wave of the anguiliform swimming. The standing wave leads to an S-bending of the body with the nodes at the girdles [Frolich & Biewener 92]. The movement of the body is coordinated with the movements of the limbs in order to increase their reach, as the limbs are fixed laterally to the body. An interesting question is therefore to find out which kind of neural circuitry can produce both types of gaits, and especially both the traveling and standing wave of the body.

The locomotor circuitry of the salamander is much less known than that of the lamprey. Kinematic studies of locomotion as well as elecromyographic (EMG) studies of muscle activity have given a fairly clear idea of the output of the locomotion circuitry [Frolich & Biewener 92, Carrier 93, Ashley-Ross 94a, Gillis 97, Ashley-Ross & Lauder 97, Delvolvé *et al.* 97], but the interneurons and the connectivity of that circuitry is for the moment undecoded.

Based on EMG measurements of the swimming and trotting motor patterns, [Delvolvé *et al.* 97] have shown that the muscles of the body are activated with a traveling wave for swimming and with a standing wave for trotting (Figure 2.5). The authors have distinguished three different traveling waves during the swimming, along the neck, the mid-trunk (the part of the body between the girdles) and the tail. The three waves travel in the caudal direction with slightly different speeds. During trotting, the muscles of the mid-trunk express single synchronous bursts which lead to the body forming a standing wave. The muscles of the neck and the tail display a double bursting pattern in the form of two waves of EMG activity propagating in opposite directions (see Figure 2.5). The authors therefore suggest that the motor patterns may be created by a lamprey-like CPG, receiving tonic input from the limb CPGs during swimming and phasic input during trotting. This would mean that, during

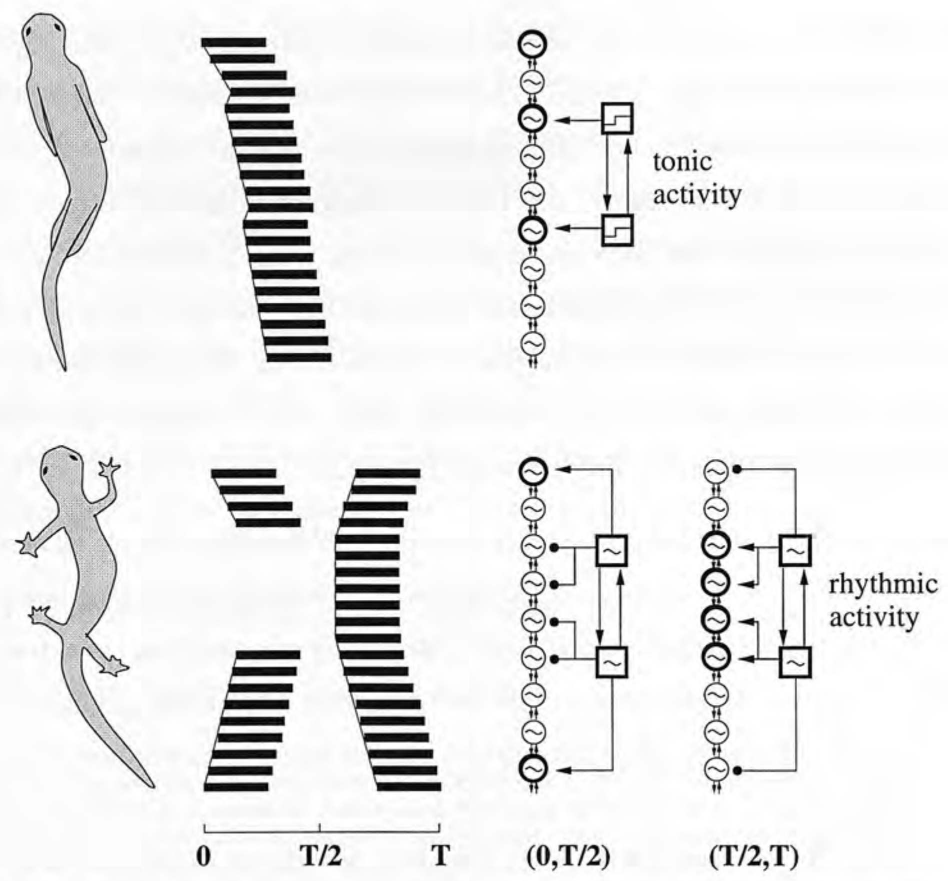


Figure 2.5: Neural activity of body muscles during swimming (*top*) and trotting (*bottom*), and proposed neural configuration (adapted from [Delvolvé *et al.* 97]). *Middle:* The thick black lines represent EMG bursts in the left muscles along the body. During trotting (bottom), midtrunk muscles contract in synchrony while tail and neck muscles present a double bursting pattern in a complete stepping cycle T. *Right:* Hypothesised neural organisation and schematic neural activity during a cycle. Inhibitory and excitatory effects are represented by filled circles and arrows, respectively.

swimming, three segments of the body CPG would receive extra excitation and, following Grillner's "trailing oscillator hypothesis", would generate the observed three traveling waves. During trotting the phasic input from the limb oscillators would force the mid-trunk muscles to oscillate in phase, while rhythmicaly exciting and inhibiting the most rostral and caudal segments leading to the double traveling waves observed in the neck and the tail. The model would also indicate that the intersegmental coupling of the body is asymmetrical and favours the caudal direction, as the three traveling waves measured during swimming only propagate in the caudal direction.

Similarly, by studying transspecies similarities, Avis Cohen has hypothesized that the locomotor circuitry may have several similarities with that of the lamprey CPG [Cohen 88]. Amphibians are believed to have evolved from simpler vertebrates similar to the lamprey and, based on both phylogenetic and ontogenetic comparisons, she proposes a pathway between them made of simple transformations. The idea is mainly that morphological changes have seen segments of the body becoming fins and then limbs, and that similarly some segments of a lamprey-like spinal cord have specialized to control them. In chapter 7, I will evolve controllers for the swimming and the trotting gaits of the salamander based on a similar idea, with two segmental oscillators specializing for controlling the motoneurons of the limbs.

Finally the capacity of a lamprey-like CPG to produce standing S-waves as observed during the trotting of the salamander has been studied in the mathematical framework of chains of oscillators [Ermentrout & Kopell 94b]. Long coupling from the extremities to segments next to the middle of a chain with otherwise only closest neighbour coupling can produce S-waves, when the local coupling is designed to produce synchrony and the long coupling is designed to produce anti-phase behaviour (i.e. an abstraction of inhibition). By performing a bifurcation analysis on different parameters of the network, the authors find that the S-wave is a stable solution in a region of the parameter space.

2.5 Dynamical neural networks

The fascinating computational power of neuronal circuitries found in animals has led researchers in artificial intelligence to develop the *artificial neural network* (ANN) field [Hertz *et al.* 91, Arbib 95]. In that field, abstract neuron units are used in a network for distributed computation tasks. The mathematical neurons typically calculate a weighted sum of their inputs and produce an output through a *transfer* function. The main interest of the field is to develop *learning* algorithms for setting the parameters of the network, such as the weights of the connections between neurons, in order to perform some computation. A large number of algorithms and network structures have thus been developed for a variety of applications such pattern recognition and classification, associative memories, *etc.*

More recently, potential applications such as control and signal processing have led researchers in artificial neural networks to develop networks which present temporal behaviour. These networks exhibit some dynamics because of their operation and structure by incorporating recurrent connections and/or because of the dynamics of their neuron models. Because of their temporal behavior, these types of neural networks present many similarities with biological networks and the connectionist models of neurobiologists. The field of dynamical neural networks and connectionist models therefore represents a field in which ANNs and computational neurobiology meet. I will here give a brief overview of the research on dynamical neural networks (for more detail see [Hertz *et al.* 91, Haykin 94, Arbib 95], for instance).

2.5.1 Neural networks which deal with time

Dynamics in the operation of the network

Recurrent networks, such as Jordan's or Elman's networks [Jordan 86, Elman 90], introduce time-dependent behaviour by containing recurrent connections in an otherwise feedforward network which feed back the states of some neurons in the network as inputs to the network at the next time step (the feedback is provided through units called *context units*). In this way, the operation of the network introduces time-dependent output with discrete time steps. The neurons themselves are not time dependent and are typical neurons of a feedforward network, where the output a_i of a neurons is calculated from its weighted input sum, $a_i = g(\sum w_{ij}a_j)$, through a saturating function such as the *sigmoid function* ($g(z) = 1/(1+e^{-z})$).

More generally, recurrent networks, in which each connection transmits the state of its pre-synaptic neuron to the post-synaptic at the next time step, can be trained with a variation of the back-propagation of feedforward networks called back-propagation through time [Williams & Peng 90]. Because of the discrete time steps, the recurrent network can be "unfolded" over time into a multilayer feedforward network, with a new layer added at each time step. A backpropagation algorithm can then be applied with the restriction that the synaptic weights of the unfolded connections corresponding to the same connection in the recurrent network are constrained to have the same value for the whole unfolded network.

A Hopfield network is another example of a recurrent network which presents a dynamics because of the way it is updated [Hopfield 82]. The network is a fully recurrent network with neurons whose ouputs are either $+1$ or -1 depending whether their weighted inputs are positive or negative, respectively. The connectivity is symmetric, without self connections ($w_{ji} = w_{ij}$ and $w_{ii} = 0$). The update rule has the particularity that neurons are updated randomly and asynchronously (i.e. one at a time). Because of the symmetric connectivity, the update rule means that the network always converges to a stable state, which corresponds to a minimum of a *Lyapunov function.* This function depends on the weights of the network, and the stable states of the network can be defined by a "storage" rule without iterative learning. Note that although this network exhibits a dynamics before reaching a stable state, its main interest is in the states to which it converges for applications such as associative memory or constraint satisfaction problems.

Dynamics in the neuron models

Time dependence can also be explicitely incorporated in the neuron models through time-delayed synaptic connections. Finite-duration impulse response (FIR) neurons have, for instance, been used to introduce time in feedforward networks [Wan 90] (see also the *time-delay neural network* [Lang & Hinton 88]). These neurons have synapses which are modeled as linear, time-invariant filters. The activation of each synapse depends not only on the current input but also on a fixed number of steps M in the past (finite memory). The linear filter computes the activation of the synapse by making a *convolution sum* on the memorized activity of its input. A synapse is therefore represented by M weights rather than a single weight. The output of a neuron at a time step is then calculated depending on the activity of the different synapses through a saturating function such as the sigmoid function. Feedforward networks with FIR neurons can be trained with a variation of the back-propagation algorithm called *temporal back-propagation*, and have showed very good performance at time-series prediction (see [Wan 94], for instance, for an example of prediction of a chaotic time series).

Continuous-time recurrent networks

The models described so far work with discrete time steps, either in the operation of the network or in the synaptic filters of the neurons. Continous-time neurons have been developed in which the neurons' behaviour is expressed in first-order differential equations. The activity of a network is then described as a nonlinear dynamical system of coupled differential equations.

The types of neurons used for dynamical recurrent networks are very similar, and often identical, to models of neurons used for connectionist models of biological systems. It is therefore not surprising that terms such as mean firing rate and membrane potentials are used in these kinds of models. A popular neuron model is the *leaky integrator*. According to this model, the mean membrane potential m_i of a neuron N_i is governed by the equation:

$$\tau_i \cdot dm_i/dt = -m_i + \sum w_{i,j}x_j + I_i$$

where $x_j = \varphi(m_j)$ represents the neuron's short-term average firing frequency, τ_i is a time constant associated with the passive properties of the neuron's membrane, $w_{i,j}$ is the synaptic weight of a connection from neuron N_j to neuron N_i, and I_i is an external input applied to the neuron. The function $\varphi(m_j)$ is the *activation* or *transfer* function, i.e. a nonlinear saturating function such as the sigmoid function $\varphi(m_j) = (1 + e^{(m_j+b_j)})^{-1}$ where b_j is the neuron's bias, or the hyperbolic tangent function $\varphi(m_j) = (1 - e^{-g_j m_j})/(1 + e^{-g_j m_j})$ where g_j is the gain.

Depending on the application, several types of network structures and learning algorithms have been developed. For instance, some applications require networks which always converge to stable states, such as the continuous Hopfield network. For these applications, the state space trajectories followed to reach the stable states are not important and specific "fixed point" learning algorithms have been developed for them. Other applications require the recurrent network to produce specific trajectories, and need therefore more general learning algorithms.

Hopfield extended his discrete model and made it continous using leaky-integrator neurons with the hyperbolic tangent function as activation function [Hopfield 84]. The output of neurons is then a real number between -1 and 1 rather than the binary output of the discrete net's neuron. The network has symmetric weights similarly to the discrete model. The evolution of the states of the neurons is determined by the system of coupled differential equations, instead of the asynchronous update rule. Similarly to the discrete model, it can be shown that because of the symmetry of the connections, the network will always converge to stable attractors, i.e. the network is globally asymptotically stable. When the gain of the neurons is high, these attractors correspond to those of the discrete model. With lower gains, these attractors can no longer be predicted as easily, and a fixed point learning algorithm must be used instead of the discrete learning rule (see [Pearlmutter 95] for instance).

The continous Hopfield network can be used as a *content addressable memory* or for optimization problems. These applications rely on the fact that the network is asymptotically stable, and are not concerned with the details of the dynamics of the network, except for determining the *basins of attraction* of the different stable states, i.e. the different initial states from which the network converges to a given attractor.

For many applications such as control or time series predictions, the details of the state space trajectories are important. These applications take advantage of the fact that continous-time recurent networks are *universal dynamics approximators* [Funahashi & Nakamura 93]. Dynamical recurrent networks can exhibit complex dynamics before reaching stable states; they can also converge to limit cycles or exhibit chaos. The aim is then to define the synaptic weights and time constants such that the network produces specific desired trajectories. This is done by minimizing a function which computes the difference between the actual trajectory $\mathbf{y}(t)$ and a desired trajectory $\mathbf{D}(t)$, such as $E = \sum_i \int_{t0}^{t1} (y_i(t) - D_i(t))^2 dt$. Several algorithms have been proposed to calculate $\partial E / \partial w_{ij}$ and $\partial E / \partial \tau_i$ for updating the synaptic weights and the time constants and minimizing E, such as an extension of backpropagation through time to continous neurons or the *real time recurrent learning* algorithm (see [Pearlmutter 95] for a review).

One problem with these learning algorithms, based on a gradient descent on E, is that they require a desired trajectory $\mathbf{D}(t)$ to be provided by the user. For many problems, especially in control, the desired output over time of the network is not known in advance. The performance of the network then depends on the effect of its output on the system it controls. The parameters of the network have to be updated depending on an evaluation of the performance of the system, rather than a desired trajectory. In these cases, especially if the system which is controlled by the network has its own complex dynamics and if it is difficult to provide a differentiable evaluation of the system, an interesting alternative to gradient-based learning algorithms is to use evolutionary techniques for designing the networks. In the next section, I will give a short presentation of evolutionary algorithms and, in section 2.5.3, I will present how they can be applied to evolve neural networks.

2.5.2 Evolutionary algorithms

Evolutionary algorithms are parallel and stochastic optimization algorithms which are inspired by natural evolution. The idea is to somehow encode solutions of a problem into *chromosomes*, to evaluate potential solutions through a *fitness function*, which returns a scalar depending on how well the solution solves a given problem, and to *evolve* (see below) the population of chromosomes until satisfactory solutions are generated. Three types of evolutionary algorithms have been developed: genetic algorithms [Holland 75], evolution strategies [Schwefel 95] and genetic programming [Koza 92]. These algorithms differ in the type of encodings and genetic operators they use, but they share the same basic functioning. I give here a brief overview of the algorithms; for a more formal description see [Goldberg 89, Bäck 96, Michalewicz 96], for instance.

Once the encoding and the fitness function are specified, the algorithms start with a randomly generated population of chromosomes which are evaluated and given a fitness value depending on the fitness function. The algorithms then start a loop through three operations: *selection, variation* and *rejection.* Selection consists of randomly choosing parent chromosomes with a probability depending on their fitness value. This probability is typically somehow proportional to the fitness value of the chromosome. Then, new chromosomes are created which resemble the parents except for some variations introduced by one or several genetic operators (see below). These new chromosomes are evaluated and given a fitness value. Finally the size of the population is kept constant by rejecting some of the chromosomes with the lowest fitness values. When looping through these operations, the algorithms little by little improve the mean fitness of the population because of the *selective pressure* which favours genetic material of good chromosomes to be copied within the population and which rejects the worst chromosomes.

In genetic algorithms, the chromosomes are composed of binary gene strings [Goldberg 89]. Pairs of child chromosomes are created from pairs of parent chromosomes by a *crossover* operator followed by a *mutation* operator. Crossover consists of swapping parts of gene strings between chromosomes which are copies of the parent chromosomes. Researchers have used single-point, two-point or uniform crossover. The mutation operator is then applied on each child chromosome and consists of switching the binary genes with some probability.

In evolutionary strategies, chromosomes are composed of a string of real numbers between 0 and 1 [Schwefel 95]. The first part of the string represents the variables of the problem to solve, and the second part represents parameters defining gaussian distributions of probability for the variables of the problem. In their first applications, evolutionary strategies were used with populations consisting of one individual only and with constant mutation parameters. They were later improved to use larger populations and a mutation operator which keeps a memory of the success of previous mutations. This is done by first making small mutations of the parameters of the probability distributions for problem variables, and then defining new values for the problem variables by mutating them following these (new) probability distributions. Parameters which have led to good mutations are therefore transmitted with the values of the problem variables.

Genetic Programming uses a more sophisticated encoding than GAs and ES. Instead of evolving a string of scalars, GP evolves trees with nodes made of functions of a programming language such as LISP, and of scalars. The genetic operators are similar to those used in GAs, but are modified to be applied to tree structures. Crossover for instance consists of exchanging two subtrees between two trees and mutation is applied both on a tree structure, by replacing a subtree by another randomly created one, or on a scalar by changing it by some random value. An interesting aspect of GPs is that they can perform symbolic regression rather than only parametric regression.

As there exist many variations on the types of encodings and on the type of genetic operations applied, these three types of algorithms little by little overlap to form a continous group of evolutionary algorithms. Many GAs, for instance, applied to optimizing problems with real parameters are now used with a real number encoding rather than binary encodings. The real number GA I shall use is described in detail in chapter 4.

2.5.3 Evolution of neural networks

Evolutionary algorithms have been used to design neural networks for approximately ten years (for reviews see [Schaffer *et al.* 92, Yao 93, Balakrishnan & Honavar 95, Whitley 95]). They have been used either to optimize the topology of the network, or as the learning algorithm in a fixed network structure, or for defining a network completely (both the structure and the synaptic weights).

The interest of using GAs for defining the topology for neural networks is that for feedforward networks, for instance, there exists no method for determining the number of layers and neurons per layers for obtaining the best performance for a specific problem. For the moment, the configuration of such networks is determined by trial and error and depends on the experience of the user. A GA can be used to optimize the topology when used with learning algorithms (see [Miller *et al.* 89] for instance). The fitness of a topology is determined from the performance of the network on its task after training.

GAs have also been used with networks with a fixed structure as an alternative learning algorithm to backpropagation, for instance [Whitley & Hanson 89, Montana & Davis 89]. Their parallel and stochastic nature makes them less likely to be trapped in a local minimum than gradient-based algorithms. However, they tend to be slower and may have problems to adequately tackle the numerous symmetries of most neural networks. Some researchers therefore use GAs for making a first search and then use a gradient based algorithm for finishing the search [Belew *et al.* 91].

The main interest for using evolutionary algorithms as learning algorithms is when the details of the desired output are not known in advance. As mentioned earlier, most learning algorithms developed for neural networks minimize an error function based on a desired output or state trajectory which has to be specified by the user. These algorithms are then not adequate for control problems where the performance of a network depends on the output of the system it controls. These types of problems require a kind of reinforcement learning, in which the network can only be rewarded after a complete control sequence. Evolutionary algorithms can then provide a very useful learning algorithm as they can optimize any function based on the behaviour of the system rather than a specific error function based on the output of the network. They have furthermore the advantage that the function does not need to be differentiable or even continous, because an evolutionary search is not gradient based.

GAs have also been used for defining both the topology and the synaptic weights. In the animat field, in particular, GAs are a popular method for defining neural controllers for locomotion and behaviours. I will review several examples of this approach in section 2.6.2.

Encodings Researchers have used several types of encoding of the neural network into the chromosome, which are either *direct* or *indirect*. In direct encodings, there is a one-to-one mapping between one gene (or string of genes if the encoding is binary) of a chromosome and one parameter of the network. Miller, for instance, used a direct encoding of the topology of a feedforward network into a chromosome by having the chromosome represent a binary matrix of all possible feedforward connections [Miller *et al.* 89]. Note that a chromosome with direct encoding does not necessarily need to be of a fixed size [Cliff *et al.* 93].

Direct encodings have the advantage of being simple and easy to implement, but have the drawback that chromosomes grow rapidly with the size of the network. Large populations of chromosomes are then needed to cope with the number of dimensions, making the evolutionary process very computationally costly. For evolving large structures some constraints for reducing the search space may be necessary.

In indirect encodings, a chromosome encodes some rules of how the network is constructed, and therefore determines the network indirectly. These type of encodings have more similarities with the type of encoding found in biology, with a *genotype* being decoded into a *phenotype* through some developmental process. Several indirect encodings, with the aim of encoding neural networks in a more compact way, have been developed. In many approaches a GP-like evolutionary algorithm is used. In such approaches (for a review see [Kodjabachian & Meyer 95]), some kind of grammars, e.g. grammars with rewriting rules [Kitano 90, Gruau 95], or grammars determining axonal growth [Nolfi & Parisi 92], are evolved which determine how the neural network is constructed. These types of encoding present therefore several similarities with biological development and some types of encoding come very close to natural genetic encoding [Kitano 95, Eggenberger 97].

Indirect encodings have the advantages, compared to direct encodings, of being more compact and potentially allowing modularity. A more compact encoding means that there are fewer dimensions in the search space, which makes the search easier for the evolutionary algorithm. Modularity is an interesting property especially in problems with symmetries, as it allows for the same substructure to be used several times in the complete solution.

2.6 Swimming and walking animats

I will give here a short survey of the animat field, especially related to locomotion. As mentioned in the introduction, the remarkable agility of animals to move and survive in an environment has led several researchers to take inspiration from animals for the creation of autonomous agents. An *animat* is a simulated or physical agent whose rules of behavior are inspired by those of animals [Meyer & Guillot 94]. An animat is therefore equipped with sensors and actuators and with a control mechanism which enables it to survive and to act in a dynamic environment, i.e which leads to an *adaptive behaviour*. An important aspect of the animat approach (or the behavior-based approach) is the strong interaction of the agent and the environment, and the belief that human-like intelligence can not be reached without first solving the problems of sensing, moving, and reacting in an environment. Researchers in the animat field are therefore naturally interested in analysing the structure and control mechanisms of animals and trying to take inspiration from them for the building of artificial agents.

Note that the interest in simulated agents or robots which move using an animal type of gait is not limited to researchers from the animat field. Many engineers have, for instance, built machines which move with an animal-like gait for some specific purpose without aiming to build agents showing adaptive behaviour. Researchers in biomechanics are also interested in replicating legged systems. Finally, in graphics animation there is now a strong interest for developing realistic characters, and physically based simulations of articulated bodies are used for more realism.

In the two next sections, I give an overview of different machines (both physical and simulated) which use animal-like types of gait, and of the locomotion controllers which have been developed for them.

2.6.1 Morphologies

Swimming animats

Swimming robots are relatively rare, compared to walking robots for instance, because of the technical difficulties of building a system with many degrees of freedom, which is water-tight and which has to move in an environment with difficult access (it is difficult for humans to follow the robot, it is difficult to stay in connection with the machine,...). However, there are several motivations for building swimming robots. The main motivation is that the swimming gaits of fishes and swimming mammals are remarkably adapted to the hydrodynamics. Swimming is for instance more mechanically efficient than propeller propulsion and it is significantly more agile in terms of accelerations and change of direction. Building swimming machines may therefore be interesting for underwater robotics and for gaining a better understanding of the hydrodynamics of fish swimming. These considerations have led to the construction of swimming robots, such as the Robotuna at the MIT. The Robotuna is swimming machine made of 8 rigid segments which are driven through a system of pulleys and cables by motors situated above the waterline. The carangiform swimming movements of the tuna, in which a wave travels from head to tail with increasing amplitude, are reproduced by control software producing an analytical wave defined by seven parameters. Researchers at MIT have found that Robotuna, similarly to fishes, presents Gray's paradox which is that the drag on the swimming fish appears to be less than the drag on the straight fish, demonstrating that fish take advantage of the hydrodynamics to reduce their drag while swimming (see http://web.mit.edu/towtank/www/tuna/index.html).

Anguiliform swimming is also currently studied on a robotics platform at Northeastern University [Jalbert *et al.* 95]. In this project, shape memory metal wires are used as linear actuators, and a prototype body of a lamprey-like robot has been built with four segments. The prototype is fixed and can produce undulatory movements when the wires are activated with a lamprey-like activation pattern. The control system is inspired from that of the lamprey and is implemented as a set of finite-state machines. The long term aim of the project is to extend the fixed prototype into an autonomous swimming machine (for a description of this project see: http://www.dac.neu.edu/msc/lamprey.html).

Terzopoulos and his colleagues have developed an interesting example of a physically-based simulated 3D underwater world [Terzopoulos *et al.* 94, Terzopoulos *et al.* 96]. The appearance, movement, and behaviour of autonomous artificial fishes are simulated, with algorithmic controllers for producing the motor sequences. The fish body is a spring-mass model with some springs with variable spring constant representing muscles (with some similarities to Ekeberg's model of the lamprey's body [Ekeberg 93]).

Other examples of simulated swimming creatures are the works by Sims [Sims 94b] and Ventrella [Ventrella 98]. An interesting aspect of these works is that the morphologies (and the control mechanisms) are evolved. The evolved bodies are made of articulated rigid parts, with several types of simplified joints (in 3 dimensions for Sims and 2 dimensions for Ventrella). The evolutionary process for building the bodies and the control mechanisms will be discussed in section 2.6.2.

Legged animats

Many legged machines have been designed and built, with several motivations. The first motivation is to build machines which can move in a larger variety of environments than wheeled robots. These environments include man-made environments such as factories, offices with uneven floors, stairs,... or rough natural environments. Another motivation is to reproduce and analyse the biomechanics and locomotion of legged animals. Finally, as mentioned earlier, there is also the motivation of designing realistic legged agents for computer graphics.

A variety of physical legged machines with one, two, four, six or eight legs have been constructed (for reviews see [Raibert & Hodgins 93, Reeve 98]). The machines also vary in the type of gaits they use; motion can, for instance, be either statically or dynamically stable. In statically stable motion, the center of gravity of the machine remains over the base of support at all times, which means that the machine is always in static equilibrium. This is obtained either through using large feet or by having enough legs to always keep three legs on the ground. The forward velocity must also be kept low enough to minimize the effect of kinetic energy on stability.

Most legged animals use dynamically stable motion for fast locomotion. This type of motion is stable over time, rather than at all times, with the gait going through some periods with limited (e.g. during trot or biped walking), or even no, support (e.g. during gallop). Dynamically stable legged machines which walk, hop, trot, run, *etc.* have been constructed (see [Raibert & Hodgins 93]), with the motivation, similarly to animals, of having faster and more agile motion. These machines require more complex control mechanisms because of the need for active balance.

Although trunk movements seem to be important for running in quadruped animals and for increasing the reach of limbs for animals such as the salamander, which have the limbs attached laterally to the body, few robots have been built with flexible spines. Lewis studied the body movements of salamander and built a robot, GEO, with an articulated spine made of two parts connected by a 3 degree of freedom joint [Lewis 96]. As this thesis presents several similarities with his work, it will be described in more details in section 2.6.2.

Several walking machines have been simulated, mainly as a basis for studying the control mechanism needed for legged gaits. Beer, for instance, developed a kinematic simulation of a six-legged insect [Beer 90], which has been reproduced by several researchers as a test bed for evaluating design methods for locomotion controllers (see section 2.6.2).

Simulation is also used to study the control of biped locomotion with, for instance, the

2D stick biped model [deGaris 90, Taga *et al.* 91, Reeve & Hallam 95]. In [deGaris 90], the simulation is only kinematic, but in the other works, forces on each joint are calculated, and the simulation is therefore dynamic.

Other animats

Although swimming and legged animats are the most commun, other animal-like locomotion systems have also been designed such as snake-like robots [Paap *et al.* 96], robots capable of hanging and swinging like monkeys [Saito & Fukuda 96], ornithopters (flying machines which flapping wings have also been built, see the University of Toronto ornithopter http://www.utias.toronto.edu/lowsped.htm), *etc.*

2.6.2 Design of locomotion controllers

As discussed in section 2.1, animal locomotion is characterized by a large number of actuators, a rhythmic activity, and the fact that efficient motion is only obtained when the actuators are well coordinated. An efficient control mechanism has to make the right transformation from commands concerning the direction and speed of motion into the rhythmic signals sent to the different actuators. Key issues are the correct timing of the different signals and taking in account the dynamics of the controlled mechanical system. The complexity also depends on the type of gait. Controlling statically stable walking is not too difficult, for instance, because the correct control signal is time independent and only depends on the current state of the machine. For swimming and dynamically stable walking, the control signals are not only state-dependent but must also be coordinated over time for efficient motion.

Researchers have taken different approaches for solving this control problem, using different implementations and different design techniques. Locomotion controllers have, for instance, been implemented as explicit control algorithms [Raibert & Hodgins 93], finite state machines [Brooks 89], classifiers [Bull *et al.* 95], neural networks [Beer 90, Quinn & Espenschied 93]. They have been designed either completely by hand, or by learning algorithms, or with evolutionary algorithms.

In the next sections, I will present different results with an emphasis on works which are biologically inspired (either in the organization or because they are neural based) and on approaches which use evolutionary algorithms as a design technique.

Biologically inspired locomotion controllers

Brooks applied the *subsumption architecture* he developed for controlling the motion of hexapod robots [Brooks 89]. His architecture distributes control in different modules. An important aspect of the architecture is that control is divided into separate behaviours rather than different functions. In the architecture, each module (made of a collection of finite state machines) connects sensor to actuators and is responsible for one behaviour such as avoiding obstacles, wandering about,... This approach was a reaction to the traditional robotics approach in which control is divided into functional modules such as a perception module, a planner and inference module and an actuator module. Brooks' aim was to build robust control systems which do not rely on a representation of the world in order to perform some task in an environment. This type of distributed and decentralized control has been pointed out to have several similarities with the neural organization found in animals such as the locust, in which inputs are integrated in several neural centers with output being a consensus of different centers [Altman & Kien 89].

Inspiration from insects, especially stick insects and cockroaches, has been used more directly for the control of a robot. For instance, Cruse and his colleagues have used analyses of the leg patterns and the interleg coordinating mechanisms in stick insects for developing a control algorithm for an hexapod robot. They have also developed a neural network model (with some parts hand-coded and other designed with learning algorithms) satisfying these observed mechanisms for controlling a simulated stick insect (see [Cruse *et al.* 95] for a review). As the neural circuitry underlying locomotion in the insect is not decoded yet, Cruse believes that such a neural model could be used to guide electrophysiological studies.

As mentioned above, Beer has simulated an artificial insect which is capable of exhibiting some basic survival behaviours in an artificial environment, including locomotion, wandering, recoil, edge-following, and feeding [Beer 90]. The insect is a six-legged animat whose locomotion and behaviour are entirely generated by artificial neurons.

The design of the neural controllers was done by hand and is inspired from several natural animals (mainly the cockroach) but without copying one. The leg-controller is a CPG whose rhythm is generated by the properties of pace-maker cells. The coordination between legs is obtained through inhibitory coupling between the pacemakers of each leg. The behaviour controller connects sensors such as antenna tactile sensors, antenna chemical sensors, energy sensors, and so on, to the locomotion controller. This type of locomotion controller was later implemented in a hexapod robot similar to the simulation, with 2 degree of freedom legs which can swing and retract [Quinn & Espenschied 93].

Evolution of locomotion controllers

One of the first examples of evolutionary design of a walking controller is [deGaris 90], in which a controller is evolved for a simplified kinematic simulation of a biped system. The controller is a two layer network whose inputs are the angles and the angular velocities of the joints and whose outputs are the angular accelerations. The weights and the signs of the connections are evolved with a binary GA. Efficient walking is obtained when the population of solutions is evolved in three evolutionary phases (i.e. a population is first evolved with a fitness function A which rewards the swing of a single leg, then with a function B which rewards a correct complete step and finally with a function C which measures the distance covered over a fixed time period).

Beer spent a long time (his doctoral study) hand-coding the neural controller for his six-legged insect [Beer 90]. Following his experiment, several researchers, including himself, used simulations inspired by his kinematic simulation as a testbed for the evolutionary design of locomotion controllers. The motivation was to develop a method for automatically generating and optimizing such a controller.

In [Beer & Gallagher 92], a neural locomotion controller is evolved with a genetic algorithm and a staged-evolution approach. First, a leg controller is generated by evolving the synaptic weights of a fully interconnected five-neuron dynamical neural network. The complete controller is then obtained by evolving the coupling between six copies of the leg controller. Successful locomotion with a tripod gait is obtained. Unlike [Beer 90], the oscillations are due to the dynamics of the network rather than to pace-maker cells. Another experiment where the whole controller is evolved from scratch (without the leg controller stage) results in even more efficient walking; but the oscillations in each leg depend on the whole system and disappear when the leg controller is isolated.

Gruau evolved similar neural controllers using his indirect, cellular encoding [Gruau 95]. In this work the cellular encoding is extended in order to allow modularity and reuse of sub-networks. This allowed him to evolve controllers for the six-legged insect in one stage only, without fixing in advance the symmetries of the network as Beer did. The modularity allows the evolutionary process to find its own symmetries, and therefore is less reliant on the *a priori* assumptions of the experimenter.

Kodjabachian also evolved controllers for Beer's insect with a developmental encoding which is similar to cellular encoding but which also incorporates geometrical aspects [Kodjabachian & Meyer 98a, Kodjabachian & Meyer 98b]. Unlike Gruau, the symmetries of the problem are fixed in advance, but the encoding has the interesting property of being context-dependent (each precursor cell will develop following the same developmental program, but the effect of the program will depend on their place on the geometrical substrate), which enables the evolutionary process to take advantage of side-effects in the development for introducing asymmetries. This encoding scheme will be described in more detail in chapter 6, where it is used for evolving swimming controllers for the lamprey in a collaboration with Jérôme Kodjabachian.

Alternatively to neural networks, controllers for Beer's hexapod insect can developed using genetic programming [Spencer 94]. The GP evolves programs using a few simple mathematical functions, with terminals which either provide constants, or an oscillatory input or sensory input. The GP is successful in evolving programs for walking patterns either based on these intrinsic oscillators or on sensory feedback. The advantage of these controllers is that they are "readable", as they resemble conventional computer programs, but with the disadvantage that they are not adaptive and distributed as neural networks are.

[Lewis *et al.* 93] present experiments in which dynamical neural networks for the walking of a real hexapod robot are evolved on-line. It is the first example of evolutionary robotics in which each evaluation is carried out on the real robot rather than in simulation. Similarly to [Beer & Gallagher 92], controllers are evolved in two stages, with first the evolution of two-neuron oscillators for a single leg, and then the evolution of the coupling between the oscillators of the legs for coordinating their movement. The fitness of a leg oscillator is defined by visual inspection of the experimenter. As this thesis presents several similaties with Lewis' doctoral research, that work is described in more detail in the next section.

Locomotion controllers for swimming have also been developed using evolutionary algorithms, but, to the best of my knowledge, never with dynamical neural networks as controllers. Sims demonstrated a methodology for jointly evolving morphologies and controllers for artificial simulated creatures [Sims 94b]. The creatures have a 3D body and move in a physically-based simulation, either on the ground or in water. The morphology of a creature and neurons controlling its behaviour are encoded into a directed graph of nodes and connections. The morphology and the neural controller are evolved simultaneously. The neurons used have little in common with biological neurons or the usual artificial neurons. They represent a function such as sum, product, min, max, sin, cos, integrate, memory, oscillate,... The resulting creatures present impressive realistic behaviour and are able to perform all kind of locomotions, such as swimming, jumping, crawling,... Sims' work is interesting as it shows the kind of complex results that can be achieved with evolutionary methods. In another work, controllers and morphologies are evolved for catching a food resource by making two individuals compete for a common resource and rewarding the individual which performs best [Sims 94a].

Ventrella also develops both the bodies and control mechanisms of swimming creatures in a 2-dimensional world [Ventrella 98]. The bodies can be made of up to eight polygons, and the evolved controllers are algorithmic oscillators, defining the movements of the segments, whose amplitude and phase are determined genetically. An interesting aspect of this work is that there is no explicit fitness function for rewarding swimming behaviour. A population of "swimbots" is allowed to evolve and new creatures are created when two swimbots meet and mate. The child then inherits a mixture of genes from each parent using genetic crossover with some mutation. As the swimbots are dispersed in the 2D world, solutions which are good swimmers have a higher probability to reproduce. Note that Ventrella not only studies the emergence of swimming behaviours, but also the effect of mate preference on the evolution, as the swimbots are geneticaly driven to preferentially search out mates exhibiting specified phenotypic features. For another example of evolution of algorithmic swimming controllers (but without evolution of the morphology) see [Usami *et al.* 98].

Although, I have here concentrated on the evolution of controllers for locomotion, a significant amount of work has also been carried out on the evolution of behaviour controllers for robots, in particular neural controllers. See [Floreano 97, Floreano 98] for reviews.

Lewis' doctoral research

Anthony Lewis' thesis [Lewis 96] addressed two issues which are directly revelant to my research: the evolution of trunk motion and its importance in robotics design, and the automatic generation of locomotion controllers using principles of vertebrate development and evolutionary computational techniques. I will therefore summarize here the main results of his research.

As already mentioned, Lewis used a genetic algorithm for the evolution of a neural controller for the walking of a hexapod robot. He then addressed the control of the swimming of the lamprey and the trotting of the salamander. Rather than using connectionist models, he tackled these problems by using mathematical oscillators and a graphical representation called a *ring-rule* (as proposed in [Winfree 80]). A ring rule represents the activity of the oscillator by unfolding it on a circle whose radius corresponds to the period of oscillations. Unit CPGs are modeled as a ring stack, with one output ring and several *adaptive rings.* The adaptive rings determine how the output ring is altered by input from other unit CPGs or from sensory input. Several adaptive rings are developed for rules such as in-phase synchronization, out-of-phase synchronization, burst length adaptation, gain adaptation,...

Lewis developed a mechanical simulation of a lamprey similar to that of Ekeberg [Ekeberg 93], and used a control mechanism made of two ring stacks per mechanical segment (the simulated model has 5 segments in the reported experiments). He designed the ring stacks to have one output ring and 4 adaptive rings (contralateral phase adaptation, ipsilateral phase adaptation, burst duration adaptation and burst height adaptation). He reports an experiment for the development of a C-bending of the body (i.e. a bending in which all muscles on one side contract in synchrony). By sequentially turning on the adaptive rules, a similar development as observed in real lamprey is obtained. No experiment is reported on the intersegmental coordination and on how phase lags between segments can be induced for swimming.

For testing the salamander locomotion, Lewis constructed a quadruped robot with a flexible spine (one joint with 3 DOF). An experiment with a handcrafted set of output rings is conducted (no adaptive rules). The experiment is unfortunately scarcely reported (summarized in one page). It seems that for being able to move forward, the robot needs both a phase lag between segments and a twist of the spine (synchronized with bending). Walking without phase lag between segments was not possible, which is in opposition with the recent findings that the midtrunk segments of salamanders oscillate in synchrony [Delvolvé *et al.* 97].

A general comment about Lewis' thesis is that it presents several very interesting ideas, but that the experiments on the development of swimming controllers for the lamprey and walking controllers for the salamander are far from conclusive (they are described in only 8 pages for the lamprey and 2 for the salamander). One would like to know how the adaptive ring rules can be used for setting the intersegmental coordination for the lamprey's controller for the creation of the swimming gait, rather than just the C-bending. As Lewis' aim was to develop a method for automatically generating locomotion controllers, one would also like to know what adaptive rules could lead to the walking gait of the salamander robot. Futhermore, it would be interesting to understand why the robot cannot move forward without twisting and without phase delays between segments. Finally, although Lewis considered the issue of pattern generation, he did not address in any of his experiments the issue of control, namely how the speed and direction of motion can be modulated by input signals to the pattern generator.

In this thesis, I will try, similarly to Lewis, to address the issues related to lamprey and salamander locomotion but at a connectionist level and using evolutionary algorithms.

The connectionist level has the advantage of being less abstract than ring rules, and the evolved neural controllers can therefore be directly compared with biological neural configurations. Designing the controllers with evolutionary algorithms will allow design with significantly fewer *a priori* assumptions than the adaptive ring rules, but it will not allow study of the biological development of the lamprey's controller as Lewis' approach did.

2.7 Interactions between artificial intelligence and biology

Advances in biology and artificial intelligence (AI) have led these two fields to come closer. Improved knowledge of the functioning of animals has brought an increasing number of models and mechanisms from which AI can take inspiration. Especially, the increasing use of computer simulation in biology, for instance in computational neuroscience, in biomechanics,..., has led to models and simulations which can readily be integrated into, compared with, and inspire artificial intelligence research.

In the previous sections, we have mainly looked at how AI has taken inspiration from biology. For instance, inspiration from biology has led to neural networks, legged machines, the animat approach, evolutionary algorithms, *etc.* I will here summarize a few aspects in which the interaction between AI and biology may bring something to biology in return.

From a general point of view, the synthetic approach of AI towards adaptive behaviour and intelligence might provide some clues and hypotheses to test about potential functionings in animals. Similary, research in artificial neural networks on applications such as pattern recognition or control, for instance, can give some hints on the computation performed in the central nervous system.

More concretely, implementing a biological control model into a robot (or a realistic mechanical model) is a useful exercise for testing the completeness of the model, that is, to verify whether all elements necessary for the production of the observed behaviour have been taken in account. It is also useful for analysing the effect of having a real body in terms of sensory feedback and body dynamics. Sensory feedback can be an essential part of the control system, and the body dynamics can play a significant role in determining the effect of a neural command on a motor action. If the model does not produce the desired behaviour, or if the implementation in the robot relies on several construction hypotheses, the robotic experiment may point to new biological experiments for gaining a better understanding of the model. Potentially, experiments such as lesions can be carried out both on the animal and on the robot to test how closely the artifical implementation and the real animal match.

Techniques developed in AI can also be useful for biology, and for neuroscience in particular. I will here look at how computational neuroscience may benefit from techniques developed in the ANN field. Computational neuroscience and ANN have different objectives and different approaches to the simulation of neuronal networks. Computational neuroscience tries to model existing neuronal circuits found in animals in order gain some insight into their functioning. Its main priority is designing models which produce all the detail of the physiological measurements. A significant task is therefore to set the parameters of the individual neuron units such that their functioning corresponds to cellular recordings, and then to see if the different units put together can produce the overall behaviour of the network of neurons studied[3] (see [Getting 89] for a nice example of the modelling methodology applied to the swimming CPG of Tritonia).

The aim of ANN researchers is not to simulate a particular part of an animal central nervous system, but to develop distributed computation systems for solving particular problems. Therefore most of ANN research is focussed on developing powerful algorithms for automatically designing the network for a particular task. These algorithms can be interesting tools for neurobiology for specifying parts of biological models for which there exists unsufficient physiological data. The algorithms are then not used as a model of the learning process (they are usually not biological plausible), but as a tool to define some parameters, such as synaptic weights, which are difficult or impossible to measure. [Lockery & Sejnowski 93a, Lockery & Sejnowski 93b] give, for instance, an interesting example of how a version of the backpropagation algorithm

[3] Ideally none of the parameter setting should be based on the overall behaviour of the system, as the capacity of the model to produce that behaviour from lower level elements is the main factor to determine whether the model accurately simulates the corresponding neuronal circuitry.

can be applied to define some synaptic weights of a connectionist model representing the escape reflex in a leech.

Because they have fewer restrictions on the functions they minimize than learning algorithms, evolutionary algorithms are particularly useful for defining parameters in neurobiological modeling. They have recently been used for determining parameters of biophysical models of single cells, for instance [West & Wilcox 97, West *et al.* 98, Vanier & Bower 98]. Evolution strategies have also been applied for defining the synaptic weights in a neural model of the salamander's visual system [Eurich *et al.* 95, Eurich *et al.* 97]. In chapter 5, I will present how a GA can be used for defining and optimizing a connectionist model of the lamprey's swimming CPG.

2.8 Summary

This chapter has presented the study of locomotion and its control in different fields such as biology, neurobiology, artificial neural networks and robotics. The main task in controlling animal-like locomotion is to be able to transform general commands, concerning directions and speeds of motion, into commands to the numerous actuators typically involved in the generation of motion. The difficulty of this transformation resides, firstly, in the necessary coordination between all the actuators for effective locomotion and, secondly, in the strong time-dependence of the signals sent to the actuators; the signals are typically rythmic and their timing (their phase relation) is crucial for the motion, especially in dynamically stable gaits.

In animals, the control mechanisms are provided by networks of neurons. Natural evolution has led to a motor organisation which is distributed and which relies on central control for the generation of the rhythmic patterns. Rather than relying on chains of reflexes, the rhythmic patterns are generated by central pattern generators, which are networks of neurons which can produce rhythmic activity without rhythmic input either from higher control centers or from sensory feedback. CPGs provide the templates for stationary motion which can be modulated by higher control and sensory feedback. I have presented the findings of neurobiologists on the CPG for the swimming of the lamprey which is one of the vertebrates whose locomotion circuitry is best understood. This thesis takes strong inspiration of one of the connectionist simulations developed from the neurobiological findings and which will presented in more details in the next chapter. I have also presented current knowledge about the locomotion of salamanders. Because salamanders are believed to have evolved from simpler vertebrates like the lamprey, and because the salamander uses a swimming gait very similar to that of the lamprey, neurobiologists believe that the salamander locomotion circuitry is based on a similar organization than that of the lamprey. The circuitry is however not known for the moment, and one of the aims of this thesis is to study which kind of neural circuitries could exhibit the typical swimming and trotting gaits of the salamander.

The computation performed by networks of neurons is also studied in the artificial neural network community. I have presented an overview of the subfield of ANN which is concerned with the temporal behaviour of neural networks. Some of the network models developed come very close to the connectionist models used by neurobiologists, but with a particular emphasis on the algorithms for automatically designing the network for a specific task. One of these techniques, design by evolutionary algorithms, is used in this thesis and is described in more detail.

There is a strong motivation for understanding and reproducing animal locomotion in several fields such as robotics, the animat field, biomechanics, graphical animation,... I have presented a brief overview of walking and swimming machines, physical or simulated, developed in these fields. Of direct interest to this thesis are the different control mechanisms and the different design methods used for controlling animal-like locomotion. I reviewed in particular the combination of neural networks and evolutionary algorithms.

Finally, although the interaction between biology and artificial intelligence seems to be mainly in one direction, with the artificial intelligence researchers taking inspiration from biological findings, there are more and more examples where research and techniques from artificial intelligence can be directly applied to biology for gaining a better understanding of the functioning of animals.

is best understood. This thesis takes strong inspiration of one of the connectionist simulations developed from the neurobiological findings and which will presented in more details in the next chapter. I have also presented current knowledge about the locomotion of salamanders. Because salamanders are believed to have evolved from simpler vertebrates like the lamprey, and because the salamander uses a swimming gait very similar to that of the lamprey, neurobiologists believe that the salamander locomotion circuitry is based on a similar organization than that of the lamprey. The circuitry is however not known for the moment, and one of the aims of this thesis is to study which kind of neural circuitries could exhibit the typical swimming and trotting gaits of the salamander.

The computation performed by networks of neurons is also studied in the artificial neural network community. I have presented an overview of the subfield of ANN which is concerned with the temporal behaviour of neural networks. Some of the network models developed come very close to the connectionist models used by neurobiologists, but with a particular emphasis on the algorithms for automatically designing the network for a specific task. One of these techniques, namely evolutionary algorithms, is used in this thesis and is described in more details.

There is a strong motivation for understanding and reproducing animal locomotion in several fields such as robotics, the animat field, biomechanics, computer animation,... I have presented a brief overview of walking and swimming machines, physical or simulated, developed in these fields. Of direct interest to this thesis are the different control mechanisms and the different design methods used for controlling animal-like locomotion. I reviewed in particular the combination of neural networks and evolutionary algorithms.

Finally, although the interaction between biology and artificial intelligence seems to be mainly in one direction, with the artificial intelligence community taking inspiration from biological findings, there are more and more examples where research and techniques from artificial intelligence can be directly applied to biology for gaining a better understanding of the functioning of animals.

Chapter 3

Ekeberg's neuronal and mechanical model of the lamprey

This chapter presents the reproduction of Ekeberg's two-dimensional connectionist model [Ekeberg 93] of the lamprey swimming circuitry. This model has inspired this thesis and it is the biological model with which the evolved controllers will be compared. I will refer to it as the *biological model.* The model combines a simulation of the neural controller with a mechanical simulation of a lamprey in water allowing direct evaluation of how neural activity is transformed into mechanical movements. It also offers the possibility of investigating the effect of sensory feedback from stretch sensitive cells. I present here the model and analyze it quantitatively (Ekeberg's paper presents mainly qualitative behaviours of the model). This quantitative analysis will give the basis with which the performances of the evolved swimming controllers of chapters 3 and 4 will be compared.

3.1 Neural controller

The neural controller represents the complete circuitry of the lamprey and is composed of 100 interconnected segmental networks (Figure 3.1). Each segmental network corresponds to the connectivity observed in the lamprey and is based on the model proposed by Grillner and Buchanan (1987, see Figure 2.3, pp 21). The segmental network is composed of the four types of neurons discovered to play a role in the pattern generation: two motoneurons (MN), two excitatory interneurons (EIN), two contralat-

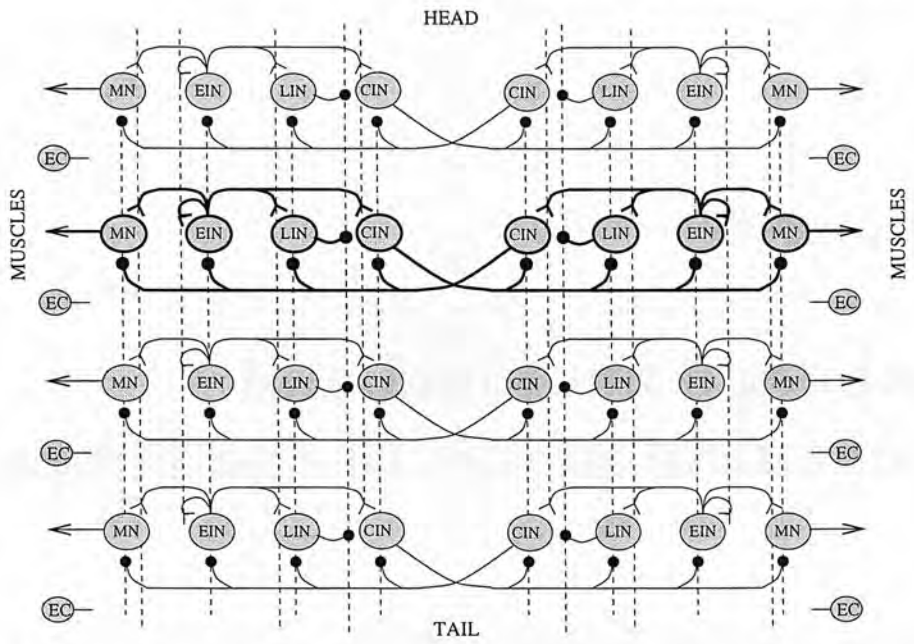


Figure 3.1: Configuration of the biological controller. The controller is made of 100 interconnected segmental oscillators (only 4 segments shown) composed of 8 neurons each. Four types of neurons are present in the oscillators: 3 types of interneurons EIN, CIN and LIN and the motoneurons MN. The controller can receive feedback from the stretch sensitive edge cells EC. (See text for explanation.)

from: / *to:*	EINl	CINl	LINl	EINr	CINr	LINr	BS
EINl	0.4 [2, 2]	-	-	-	-2.0 [1, 10]	-	2.0
CINl	3.0 [2, 2]	-	-1.0 [5, 5]	-	-2.0 [1, 10]	-	7.0
LINl	13.0 [5, 5]	-	-	-	-1.0 [1, 10]	-	5.0
MNl	1.0 [5, 5]	-	-	-	-2.0 [5, 5]	-	5.0
EINr	-	-2.0 [1, 10]	-	0.4 [2, 2]	-	-	2.0
CINr	-	-2.0 [1, 10]	-	3.0 [2, 2]	-	-1.0 [5, 5]	7.0
LINr	-	-1.0 [1, 10]	-	13.0 [5, 5]	-	-	5.0
MNr	-	-2.0 [5, 5]	-	1.0 [5, 5]	-	-	5.0

Table 3.1: Biological configuration, as given in [Ekeberg 93]. Excitatory and inhibitory connections are represented by positive and negative weights respectively. Left and right neurons are indicated by *l* and *r*. *BS* stands for brain stem. The extension of the segmental connection to neighbour segments is given in brackets (extensions to the rostral and caudal direction, respectively).

Neuron type	Θ	Γ	τ_D	μ	τ_A
EIN	-0.2	1.8	30 ms	0.3	400 ms
CIN	0.5	1.0	20 ms	0.3	200 ms
LIN	8.0	0.5	50 ms	0.0	-
MN	0.1	0.3	20 ms	0.0	-

Table 3.2: Neuron parameters, as given in [Ekeberg 93]. Θ is the threshold, Γ the gain, τ_D the time constant of the dendritic sums, μ the coefficient of frequency adaptation and τ_A, the time constant of the frequency adaptation.

eral inhibitory interneurons (CIN) and two lateral inhibitory interneurons (LIN). Each neuron unit in the model represents a population of functionally similar neurons in the real lamprey. Neurons receive excitatory input from the brainstem. The weights of the segmental connections are given in Table 3.1.

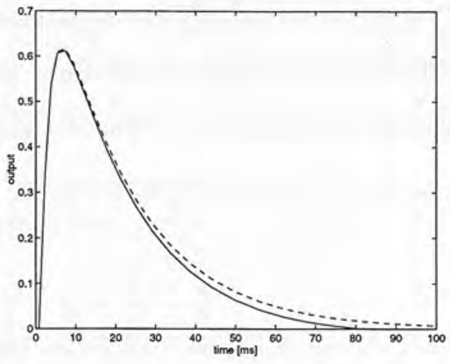


Figure 3.2: Typical output of a CIN neuron. From time $t = 0$ constant inputs: excitatory input = 10, inhibitory input = 1. The dashed line shows the same behaviour but without the frequency adaptation (μ=0).

A neuron unit is modeled as a leaky integrator with a saturating transfer function. Its output u corresponds to the mean firing frequency of the population it represents ($\in [0, 1]$) and is calculated as follows:

$$\dot{\xi_+} = \frac{1}{\tau_D}(\sum_{i\in\Psi_+} u_i w_i - \xi_+) \tag{3.1}$$

$$\dot{\xi_-} = \frac{1}{\tau_D}(\sum_{i\in\Psi_-} u_i w_i - \xi_-) \tag{3.2}$$

$$\dot{\vartheta} = \frac{1}{\tau_A}(u - \vartheta) \tag{3.3}$$

$$u = \begin{cases} 1 - \exp\{(\Theta - \xi_+)\Gamma\} - \xi_- - \mu\vartheta & (u > 0) \\ 0 & (u \leq 0) \end{cases} \tag{3.4}$$

where w_i are the synaptic weights, Ψ_+ and Ψ_- represent the groups of pre-synaptic excitatory and inhibitory neurons respectively, ξ_+ and ξ_- are the delayed 'reactions' to excitatory and inhibitory input and ϑ represents the frequency adaptation observed in some real neurons [1] [Ekeberg 93]. The parameters of each type of neuron are given in Table 3.2. Figure 3.2 illustrates the kind of dynamics the neuron models exhibit. The

[1] Frequency adaptation means that the firing rate of a neuron is not constant for a constant input, with, typically, a slight decrease of the firing rate over time. This is a neural property which should not be mistaken for an effect on the oscillation frequency of the network (although frequency adaptation may indirectly affect it).

neuron parameters, as well as the connection weights of Table 3.1, have been defined so that the simulation of the model fits physiological observations [Ekeberg 93]. The activity of a segmental network or the complete controller is calculated by integrating the system of coupled differential equations given by equations 3.1 to 3.4. I use the fourth-order Runge-Kutta method for solving the equations [Press *et al.* 94], with a fixed time step of 5 ms (Ekeberg uses the first order Euler method with a step size of 10 ms). The code is written in C, and Matlab is used for visualising the results.

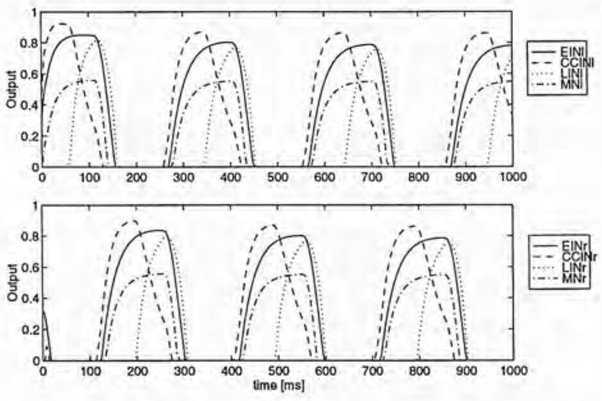


Figure 3.3: Activity of the segmental network with an excitation level of 0.4.

When a segmental network with asymmetric initial conditions[2] receives adequate excitation from the brainstem (i.e. the activity of the brainstem is set to some value, which therefore leads to an excitation of the neurons through the brainstem-neurons connections), it oscillates regularly with the left and right neurons out of phase (Figure 3.3). Burst termination is provided by the LIN neurons which become active late in the cycle because of their high threshold and the high time delay. Once active, the LIN neurons inhibit the ispilateral CIN neurons which allows activity on the contralateral side to start. Increasing the external excitation has the effect of increasing the frequency of oscillation and the amplitude of the motoneurons (Figure 3.4). The frequency varies nearly linearly with the excitation, and ranges from 1.7 to 5.6 Hz.

The complete controller is formed by interconnecting 100 copies of this segmental network. An interconnection consists of extending the connection from one neuron to another in one segment to the corresponding neuron in neighbouring segments (Table 3.1).

[2] For instance, $\xi_+(0) = 1$ and $\xi_-(0) = 0$ for all the left neurons and $\xi_+(0) = \xi_-(0) = 0$ for all the right neurons.

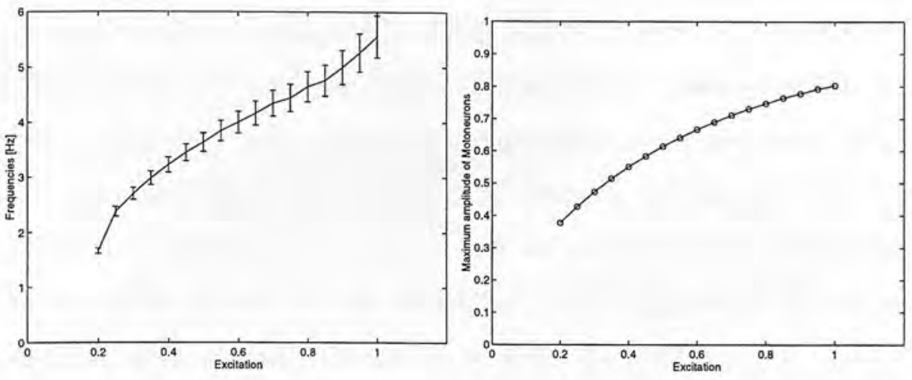


Figure 3.4: Effect of the external excitation on the frequency of oscillation (*left*) and the amplitude of the motoneurons (*right*).

As the projections between segments in the lamprey are not well known, Ekeberg has chosen a simplified coupling in which each segmental connection extends symmetrically in the caudal and rostral direction, except the connections from CIN neurons which have longer projections in the caudal direction. In order to limit the input of each neuron and to compensate neurons located in segments close to the extremities of the spinal cord, synaptic weights are rescaled and the weights of the connections to a neuron are divided by the number of segments it receives input from.

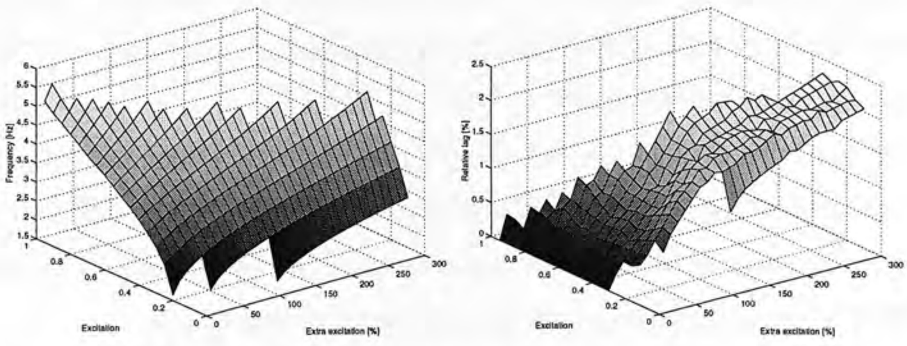


Figure 3.5: Effect of the global excitation and the extra excitation on the frequency of oscillation (*left*) and on the lag between segment relative to the cycle duration(*right*).

When external excitation is applied to the complete network through the connections from the brainstem, all segments start to oscillate in synchrony, with the same frequency. Similarly to the single segment, the frequency of oscillation increases monotonically with the excitation. Applying extra excitation to the 5 most rostral segments leads to small phase lags between segments, and therefore to a wave of neural activity that travels from head to tail. The lag between segments relative to the period of oscillation is then constant over the whole spinal cord, except for the first and last ten segments at the extremities where it decreases towards zero. The value of the lag can be varied with the amount of extra excitation, and the higher the extra excitation, the larger the lag (or the shorter the wavelength). Interestingly, the frequency of oscillation and the wavelength of the undulation can be changed nearly independently (Figure 3.5). The model can therefore reproduce the capacity of the real lamprey to cover a whole range of different frequencies of oscillations while keeping the wavelength of the undulation constant. The simulations show that frequencies between 1.6 and 5.5 Hz and phase lags per segment between 0.0% (no travelling wave) and 2.4% of the period can be obtained (as the wavelength corresponds to the inverse of the total phase lag between the first and the last segment, the wavelength is then smaller than 50% of the 100-segment body).

3.2 Mechanical simulation

I have also reproduced Ekeberg's mechanical model of the lamprey's body interacting with water [Ekeberg 93]. The body is made of 10 rigid links connected through one-degree of freedom joints, with each mechanical link corresponding therefore to 10 neural segments. Muscles are connected to each link (Figure 3.6) and are modeled as a combination of springs and dampers. Each link of the body is assumed to be 30 mm long, with an elliptical cross section of constant height (30 mm) and variable width (see Table 3.3). The masses and moments of inertia of the links are calculated by assuming the density of the lamprey to be constant and equal to that of water.

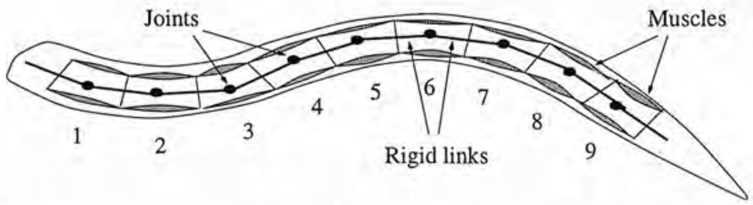


Figure 3.6: Biomechanical simulation of the lamprey. A two dimensional body of a lamprey is modeled as a set of rigid links connected through one-degree of freedom joints. The muscles are simulated as a combination of dampers and springs. Neural activity is transformed into muscular activity by the motoneurons changing the spring constants of the muscles.

The acceleration of each link i depends on the torques due to the muscles T, the forces exerted by the water F_{wat}, and the inner forces due to the constraints F_{in}:

$$\begin{aligned} m_i \ddot{x}_i &= F_{wat,i,x} + \sum F_{in,i,x} && (3.5) \\ m_i \ddot{y}_i &= F_{wat,i,y} + \sum F_{in,i,y} && (3.6) \\ I_i \ddot{\varphi}_i &= \sum T_i - \sum F_{in,i,x} \frac{l_i}{2} \sin \varphi_i + \sum F_{in,i,y} \frac{l_i}{2} \cos \varphi_i && (3.7) \end{aligned}$$

where m_i and I_i are the mass and the moment of inertia of a link, x_i and y_i are the position of the middle of the link and φ_i is its angle.

link	w_i [mm]	m_i [g]	I_i [g mm^2]	$\lambda_\perp$ [Ns^2/m^2]	$\lambda_\parallel$ [Ns^2/m^2]
1	20.0	14.1	1414	0.45	0.3
2	20.0	14.1	1414	0.45	0.2
3	20.0	14.1	1414	0.45	0.1
4	20.0	14.1	1414	0.45	0.0
5	17.2	12.2	1137	0.45	0.0
6	15.0	10.6	944	0.45	0.0
7	11.7	8.3	691	0.45	0.0
8	8.3	5.9	465	0.45	0.0
9	5.0	3.5	271	0.45	0.0
10	1.7	1.2	90	0.45	0.0

Table 3.3: Corrected parameters for the mechanical simulation (see [Ekeberg 93]).

The motoneurons can 'contract' muscles by increasing their spring constant, thus reducing their resting length. The motoneuron signals for determining the torques on the 9 joints of the body come from 9 equally spaced neural segments (from neural segment 5 close to the head, to neural segment 95). The torque acting at a particular joint is therefore determined by the left/right motoneuron activities (M_l and M_r) of the corresponding parallel muscles:

$$T = \alpha(M_l - M_r) + \beta(M_l + M_r + \gamma)\Delta\varphi + \delta\Delta\dot{\varphi} \tag{3.8}$$

where $\Delta\varphi$ is the difference between the actual angle of the joint and the default angle. The different coefficients α, β, γ, and δ determine, respectively, the gain, the stiffness gain, the tonic stiffness, and the damping coefficient of the muscles. I have used larger values for these parameters than Ekeberg (see below), and in my simulations $\alpha = 9.4$ [N mm], $\beta = 0.94$ [N mm], $\gamma = 10.0$ and $\delta = 94.0$ [N mm ms].

It is assumed that the speed of the water relative to the body is sufficiently high for the forces exerted by the water to be mainly inertial forces (high Reynolds number).

It is also assumed that the water is stationary and that the parallel and perpendicular components of that force on each segment can be calculated separately. The components of the force can therefore be calculated as: $F_{env\|} = \lambda_{\|} v_{\|}^2$ and $F_{env\perp} = \lambda_{\perp} v_{\perp}^2$ where $v_{\|}$ and $v_{\perp}$ are the components of the speed of the link relative to the water and $\lambda_{\|} = \frac{1}{2} C_{\|} S \rho$ and $\lambda_{\perp} = \frac{1}{2} C_{\perp} S \rho$ are coefficients which depend on the density of the fluid ρ, the area perpendicular to the movement S and the drag coefficient C dependent on the shape of the link (here $C_{\perp} = 1$ and $C_{\|} = 0$ for all links except those close to the head. See Table 3.3).

The inner forces are forces which correspond to the constraints due to the joints and which ensure that segments stay connected at all times. These constraints are: $x_i + \frac{l}{2} \cos \varphi_i = x_{i+1} - \frac{l}{2} \cos \varphi_{i+1}$ and $y_i + \frac{l}{2} \sin \varphi_i = y_{i+1} - \frac{l}{2} \sin \varphi_{i+1}$ for $i \in \{1, ..., 9\}$. These constraints can be rewritten in compact form as $\mathbf{g}(\mathbf{p}) = 0$, where $\mathbf{p}$ is a column vector composed of the position coordinates of all the links ($\mathbf{p} = \{x_1, ..., x_9, y_1, ..., y_9, \varphi_1, ..., \varphi_9\}$). Based on the jacobian of $\mathbf{g}(\mathbf{p})$, a system of linear equations can be derived for calculating the forces necessary for keeping the constraint equations true (see [Ekeberg 93] for the details of the calculation).

The equations for the mechanical simulation are solved with the fourth order Runge-Kutta method with a fixed time step of 0.5 ms. This means that, between each neural step, 10 mechanical integration steps are realised, similarly to Ekeberg's simulation.

My simulation is identical to Ekeberg's, except for some mechanical parameter values (and for the integration method: I use the fourth order Runge-Kutta method, instead of the first order Euler method). It appeared that, given the geometry of the body assumed by Ekeberg [*ibid*], the masses of the links m_i should be increased by a factor of π in order to satisfy the assumption that the density of the body is the same as that of the water. Also, the coefficients of the drag forces ($\lambda_{\perp}$,$\lambda_{\|}$) seem to be a factor 10 too small compared to the assumptions of a near cylindrical body (drag coefficient C close to 1). Table 3.3 gives the mechanical values I used in my simulations. Note that I increased the muscles' coefficients by a factor of π to compensate for the larger masses and the stronger water forces. Because of these new values, I obtain speeds of swimming which are approximately 40% smaller for the two quantitative examples given in [Ekeberg 93]. However, the qualitative behaviours of the two simulations are identical with, namely, identical influence on the speed of swimming of the global excitation and the extra excitation on the first segments.

3.2.1 Characterisation of the mechanical simulation with a sinusoidal wave

Propulsion through water is obtained when a traveling wave of neural activity is transformed into a traveling undulation of the body. The head to tail propagation of the undulation leads to forces from the surrounding water which propel the fish forward.

In order to test the swimming capacities of the mechanical simulation, I carried out a few simple swimming tests with a sinusoidal controller (instead of the neural controller). The output of the left and right motoneurons along the spinal cord is then determined by the following sinusoidal wave:

$$MN_l(t, seg) = \begin{cases} \alpha \sin(\omega t - \omega \Delta\tau (seg - 1)) & \text{if positive} \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

$$MN_r(t, seg) = \begin{cases} \alpha \sin(\pi + \omega t - \omega \Delta\tau (seg - 1)) & \text{if positive} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

where $MN_l(t, seg)$ and $MN_r(t, seg)$ are, respectively, the outputs of left and right motoneurons in segment seg at time t ($seg \in [1, 100]$). The maximal amplitude of the signal is given by α; and ω and $\Delta\tau$ correspond to the frequency of oscillation and the relative lag per segment, respectively.

With this simple function, it is possible to investigate how the speed of swimming is affected by muscular signals with different frequencies of oscillation, relative phase lags and amplitudes. Figure 3.7 shows the dependency of the speed of swimming on the frequency and relative lag (maximum amplitude $\alpha = 0.8$). The maximum speed of swimming is obtained with a frequency of approximately 10.0Hz and a relative lag per segment of 2.0%. The swimming speed decreases with higher frequencies and lags because the elasticy and the damping of the muscles prevent them to make large contractions at higher frequencies and because the nine-segment body can not propagate an undulation with too short a wavelength.

Increasing the amplitude α leads to an increase of speed, without significantly influencing the speed(frequency,lag) relation showed in Figure 3.7. The maximum speed of

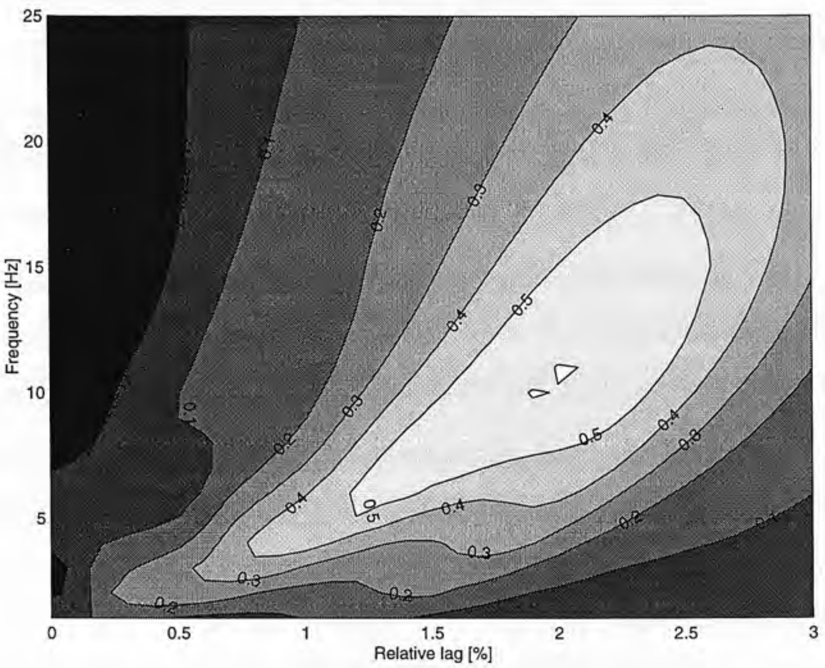


Figure 3.7: Effect of different frequencies and relative lags on the speed of swimming (contour plot), in swimming controlled by a sinusoidal function (maximum amplitude $\alpha = 0.8$).

swimming in all the tests I realised was of approximately 0.60m/s.

3.3 Neuromechanical simulation: simulated swimming

Figure 3.8 shows a simulation of both the neural controller and the mechanical model of the body. The traveling wave of motoneuron activity leads to an anguiliform swimming similar to that obtained with the simple sinusoidal controller. The speed of swimming can be modulated by varying the frequency and the relative phase lag of the oscillations depending on the how excitation is applied through the connections from the brainstem: the frequency of oscillations can be modulated by the global excitation applied to the whole CPG, and the phase lag between segments can be varied with the extra excitation applied to the most rostral segments. I have simulated the biological model over the whole range of frequencies and phase lags it can produce, and the highest speed it can reach is 0.50 m/s (global excitation of 0.65 with 70% extra excitation on first segments producing oscillations at 5.5 Hz and relative lags between segments of 1.2%). Figure 3.9 shows how the speed depends on the global and local excitation applied to the CPG.

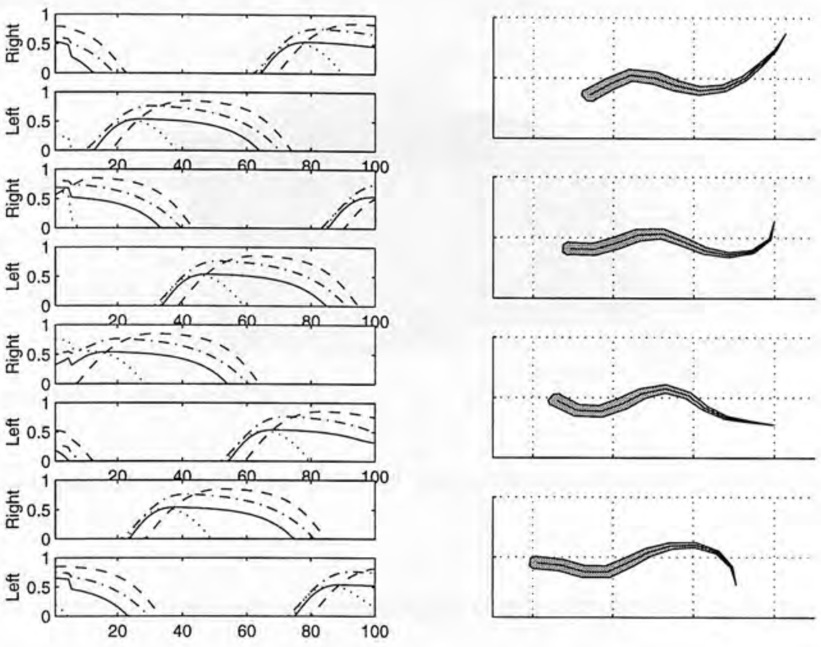


Figure 3.8: Simulated swimming of the lamprey. The level of excitation is 0.4 (arbitrary value) over the spinal cord, with 70% extra on the 5 first segments. The segments oscillate at 4.0 Hz and the speed of swimming is 0.40 m/s. A) Neural activity of the 800 neurons along the spinal cord. B) Mechanical simulation. There is 50ms between each snapshot, and the doted vertical lines are separated by 100mm.

Note that the speed of swimming abruptly drops when both the global and the extra excitations are large because the CPG saturates and does not oscillate for those levels of excitation.

The direction of swimming can be changed when asymmetrical excitation is applied to the left and right sides of the spinal cord. Exciting more one side leads to an increase of the amplitudes of the motoneurons and therefore to an increased curvature on that side. If the asymmetry is brief, this leads to a turn in the direction of the extra excitation followed by straight swimming. If the asymmetry is maintained, the lamprey will continue to swim in a circle.

The mechanical simulation allows a direct evaluation of the efficiency of the CPG for controlling swimming (in terms of speed of swimming, for instance). It also allows study of the effect of sensory feedback from edge cells, which are stretch sensitive cells located on both sides of the spinal cord [Viana Di Prisco *et al.* 90]. These cells

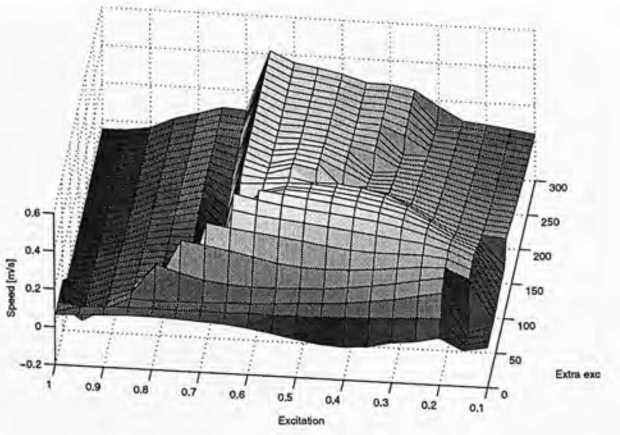


Figure 3.9: Effect of the global excitation and the extra excitation on the speed of swimming.

can be modeled as outputing a signal proportional to the local curvature of the body. While in [Ekeberg 93] the effects of the edge cells on the controller are negligible (very low synaptic weight), in [Ekeberg *et al.* 95] it is shown that feedback can help the lamprey to cross a speed barrier (water with local speed opposite to the direction of swimming) by coordinating the neural activity with the actual movements of the body and preventing a change of direction.

3.4 Summary

This chapter presented my reproduction of Ekeberg's neuronal and mechanical simulation of the lamprey. The model is based on the connectivity observed in the lamprey, with a segmental connectivity similar to Buchanan and Grillner's model [Buchanan & Grillner 87], and a simple intersegmental coupling in which segmental connections project symmetrically to several neighbouring segments caudally and rostrally, except for the connections from the CIN neurons which project asymmetrically and have longer extensions in the caudal direction. The model is able to reproduce several features observed in the real lamprey: 1) segmental networks can be made to oscillate independently when isolated, 2) the frequency of oscillation increases with the level of external excitation, 3) phase lags between segments can be obtained which are almost constant over the spinal cord, and which do not depend on the global level of excitation. Unlike the real lamprey, however, all segments oscillate in phase when no extra excitation is applied to the most rostral segments, while caudally directed waves are usually observed in the real lamprey [Grillner *et al.* 91].

When the neural simulation is used to control the mechanical simulation, an anguiliform swimming is obtained which is very similar to that observed in the real lamprey. By playing with the levels of global and local excitation of the spinal cord, swimming with different frequencies of oscillation and wavelengths can be obtained, which allows modulation of the speed of swimming. The direction of swimming can also be changed when asymmetrical excitation is applied.

extra excitation is applied to the most rostral segments while caudally directed waves are usually observed in the real lamprey [Grillner et al. 41].

When the neural simulation is [illegible] mechanical simulation, an anguilliform swimming is obtained [illegible] very similar [illegible] observed in the real lamprey. By playing with the levels [illegible] and [illegible] spinal cord, swimming with different frequencies [illegible] obtained, which allows modulation of the speed of [illegible] direction of swimming can also be changed when asymmetrical excitation is applied.

[illegible] of the extra excitation on the speed of swimming.

[illegible] proportional to the local curvature of the body. [illegible] on the controller are negligible (very [illegible] is shown that feedback can help the [illegible] speed opposite to the direction of [illegible] with the actual movements of the body [illegible]

[illegible]

[illegible] neuronal and mechanical [illegible] observed in the [illegible] model [illegible] coupling in which segmental [illegible] model is able to reproduce [illegible] can be made to [illegible] increase with the [illegible]

Chapter 4

Evolving swimming CPGs using a direct encoding scheme

In this chapter, I present how swimming controllers based on similar neurons to those of Ekeberg's model can be evolved with a genetic algorithm. Swimming controllers are developed in three evolutionary stages, with a simple direct encoding scheme.

The motivations are to show how a genetic algorithm can be used as a tool for designing neural controllers and to visit the space of possible controllers for undulatory swimming. Having the biological controller as an example possible controller gives us the assurance that at least one interesting solution exists in the search space, and also gives us an example with which evolved controllers can be compared.

By visiting the space of possible solutions and generating alternative controllers, my purpose is not only to evaluate the GA as a design method, but also to gain some insight into the general characteristics of the control of undulatory swimming and to investigate how unique the lamprey's neural configuration is. In this chapter, the search space will however be limited to solutions which possess several similarities with the biological configuration. A larger space will be visited in chapter 6.

The next sections describe the three stages of the evolution, namely the evolution of segmental oscillators in a first stage, followed by the evolution of the coupling between 100 copies of a segmental oscillator, and finally the evolution of sensory feedback connections from stretch-sensitive cells. The research presented here has been published in [Ijspeert *et al.* 98b], and follows preliminary experiments published in [Ijspeert *et al.* 97].

4.1 Methods

I develop alternative swimming controllers by using a real number GA to define suitable connections and synaptic weights between neurons similar to those of Ekeberg's model. The similarity of the neuron models will enable a direct comparison of the performances of the evolved solutions with those of the biological model. In this chapter, a direct encoding scheme is used in which each gene corresponds to one parameter of the neural configuration. The controllers are evolved in three stages. First, segmental oscillators are evolved; then multi-segmental controllers are generated by evolving the couplings between copies of a chosen segmental oscillator; and, finally, connections providing sensory feedback from stretch sensitive cells are added. The first stage requires only the simulation of the neural activity within a segment, while the two last are realised with a simulation of the whole controller together with the mechanical simulation of the body interacting with water.

This decomposition into stages reduces the search space of possible solutions and is motivated by properties of the biological controller I would like to reproduce: 1) the capacity of a segment to oscillate independently from the other segments and from sensory feedback, 2) the capacity of the whole (multi-segmental) controller to produce traveling waves of oscillations without the need for sensory feedback for intersegmental coordination, and 3) the capacity to integrate sensory feedback for coordinating the neural activity with the actual movements of the body when these are disturbed by the environment. The third stage therefore includes the simulation of a speed barrier which can only be crossed when sensory feedback is provided to the CPG.

The controllers are evaluated depending on their ability to control swimming. In order to obtain similar performance to the biological controller, I define the following goals for the evolved controllers:

- production of stable oscillations over the spinal cord,
- generation of phase lags between segments,
- frequency of oscillation dependent on global excitation,
- lag between segments dependent on the proportion of extra excitation of the most

rostral segments,

- independence of lag and frequency,
- large ranges of frequencies, lags and speeds of swimming,
- capacity to incorporate sensory feedback from stretch sensitive cells for crossing a speed barrier.

4.1.1 Genetic Algorithm

I present here the basic real number genetic algorithm which is used throughout this thesis. The algorithm is a variation of the standard GA (see for instance [Goldberg 89]) with the usual binary encoding being replaced by a real number encoding. Potential solutions are encoded into *chromosomes* made of fixed-length strings of *genes* which are real numbers $\in [0, 1]$. Starting with a randomly generated initial population, the GA is a loop through the *selection*, *variation* and *rejection* operations (Figure 4.1).

Selection At each generation, a fixed number of *parent* chromosomes are chosen with a *rank-based probability.* The selection consists of choosing chromosomes with a probability linearly proportional to their position in the fitness rank. For instance, the fittest of a population of N chromosomes has a probability $P(1) = \frac{N}{1+2+...+N}$ to be chosen, while the j^{th} fittest has a probability $P(j) = \frac{N-(j-1)}{1+2+...+N}$. Only pairs made of different chromosomes are taken, but the same chromosome can be chosen in several pairs.

Variation New chromosomes (*children*) are created by a *crossover* and a *mutation* operator. The crossover operator creates pairs of children from pairs of parents, by either applying a two-point crossover (probability *Prob_Xover*) or by simply copying the two parents. The two-point crossover consists of randomly choosing two locations in the string of genes and creating children by swapping substrings of the parent chromosomes (Figure 4.2). If the two locations happen to be the same, a single-point crossover is applied.

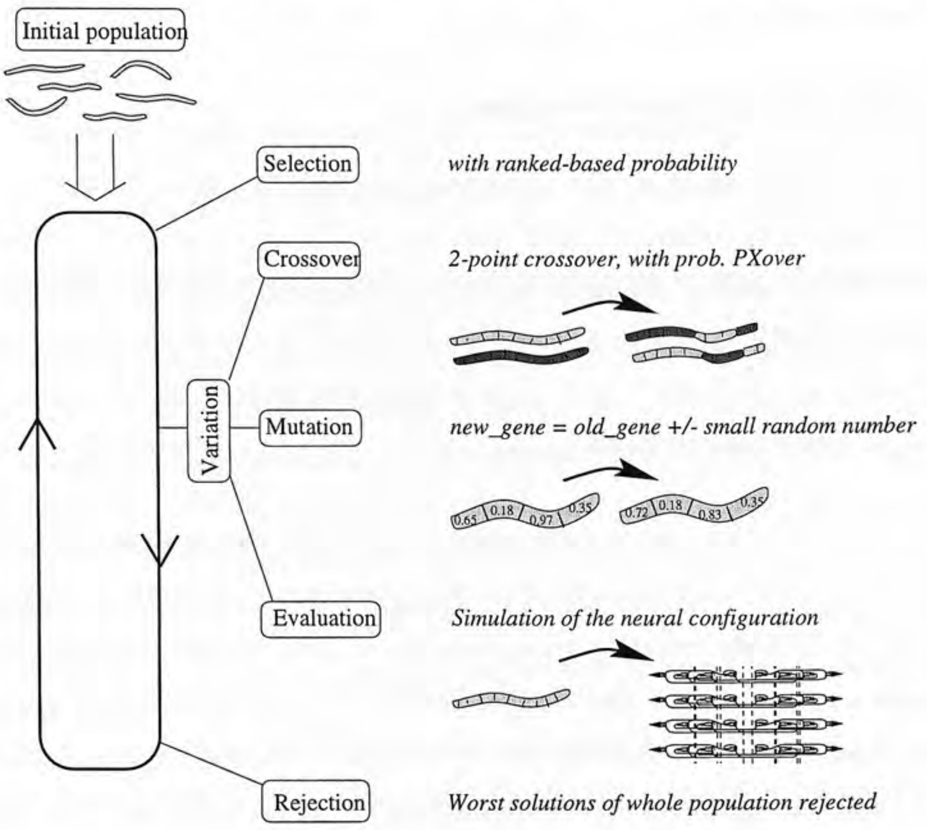

Initial population
Selection
with ranked-based probability
Crossover
2-point crossover, with prob. PXover
Variation
Mutation
new_gene = old_gene +/- small random number
0.65
0.18
0.97
0.35
0.72
0.18
0.83
0.35
Evaluation
Simulation of the neural configuration
Rejection
Worst solutions of whole population rejected

Figure 4.1: Schematic view of the genetic algorithm used throughout this thesis.

The *mutation* operator mutates each gene of the children with a probability *Prob_mut* and the mutation consists of adding or substracting a small random number within a mutation range:

$$new\_value = old\_value + Mut\_Range \cdot rand$$

where $rand$ is a random number $\in [-0.5, 0.5]$. If the new value falls outside the [0,1] range, it is set to the closest boundary. In some applications, a *pruning* operator is applied as an extra mutation. This operator is problem-specific and is used to randomly prune synaptic connections by setting, with a probability *Prob_prune*, a gene to the value corresponding to a null weight. The last step in the variation operation consists of evaluating the newly created chromosomes and assigning to them a fitness value.

Rejection The size of the population is kept constant by rejecting, at each generation, the worst solutions of the increased population (old population plus children). This rejection procedure leads to a relatively "aggressive" search as it means that the

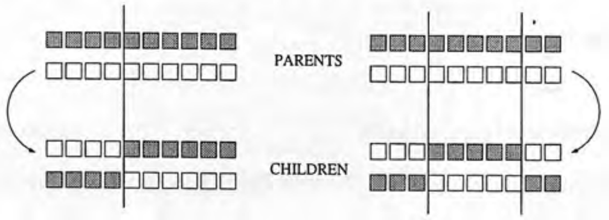


Figure 4.2: Single- and two-point crossover.

mean fitness of the population can never decrease.

Ending of the algorithm The evolution of the fitness of the population is monitored and the algorithm is arbitrarily stopped when a satisfactory level of fitness is obtained. As my experiments always consist of several evolutions with different initial populations, all evolutions are stopped after the same number of generations, unless stated otherwise.

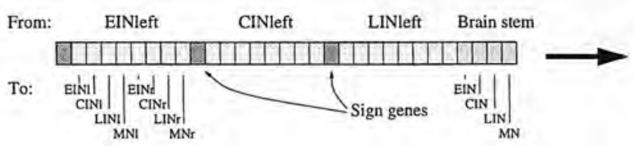
Note that this implementation of an evolutionary algorithm involves several arbitrary choices concerning the selection, variation and rejection operations and their parameters. These choices were partly motivated by observations found in the evolutionary algorithms literature and by a few initial tests. As will be further discussed in Chapter 9, the search performed by the algorithm is therefore not necessary optimal.

4.2 First stage: Evolution of segmental oscillators

The first stage consists of the evolution of segmental oscillators. Segmental networks are developed by evolving the synaptic weights of all possible connections between 8 neurons similar to those of Ekeberg. The same neuron models are used, with the same 4 types of neurons, but without specifying in advance the *sign* of the neuron, i.e. whether it is inhibitory or excitatory. The quality of a solution is evaluated depending on its ability to produce regular oscillations whose frequency can be varied with the level of external excitation.

4.2.1 Encoding

A chromosome represents the connectivity between the 8 neurons of the biological model. The number of neurons and their dynamics (defined by the 5 parameters of equations 3.1-3.4, pp61) are fixed and only the sign of the neurons and the synaptic weights of the connections are evolved. I impose a left-right symmetry. A chromosome is thus a string of 31 genes which are real numbers between 0 and 1 (Figure 4.3). There are three *sign genes* which define whether the interneurons are inhibitory or excitatory.[1] The motoneurons are excitatory. The other genes directly encode the synaptic weights and correspond (via a linear transformation) to a real value between -5 and 0 or between 0 and 15, depending on the value of the sign gene (the interneuron is inhibitory if the sign gene is smaller than 0.5 and excitatory otherwise). Finally the four last genes of a chromosome determine the synaptic weights of the connections coming from the brain stem and correspond to a real value between -5 and 15. The weight boundaries (-5 and 15) have been defined so that they include the range of weights of the biological model (-2 and 13). Note that, although the denominations EIN, CIN and LIN are kept to distinguish between the different types of interneurons (the different neuron dynamics given by equations 1-4), they lose their meaning as a description of the function or even the sign these neurons had in the biological model.



	EINl	CINl	LINl	EINr	CINr	LINr	Brain Stem
EINl	1.3	-0.8			-2.7		4.3
CINl		-3.5	-4.2				10.8
LINl	7.4						3.0
MNl	6.0				-3.0		5.8
EINr				1.3	-0.8		4.3
CINr		-2.7			-3.5	-4.2	10.8
LINr				7.4			3.0
MNr		-3.0		6.0			5.8

Figure 4.3: Encoding of a segmental network.

4.2.2 Genetic Algorithm

The GA described in section 4.1.1 is used, with the pruning operator. The parameters used for the evolution of segmental oscillators are given in Table 4.1.

[1] I chose to have neurons which are either inhibitory or excitatory (rather than neurons which could both inhibit and excite other neurons) in order to develop networks under the same constraints as the biological model, in the sense that biological neurons do not usually emit both excitatory and inhibitory neurotransmitters.

Population size	100
Number of children	30
Crossover probability	0.5
Mutation probability	0.4
Mutation range	0.2
Pruning probability	0.1

Table 4.1: GA parameters for evolving segmental oscillators

4.2.3 Evaluation

I define a fitness function that rewards three desired characteristics for segmental oscillators:

1. The production of regular oscillations of the motoneurons, with one peak of activity per period and with the left and right neurons out of phase.
2. A frequency of oscillation which can be varied and which increases monotonically with the level of external excitation.
3. A minimal set of connections. In [Ijspeert *et al.* 97], I evolved fully connected solutions, the weakest connections of which proved unnecessary for the creation of oscillations and could be removed without affecting the neural activity.

Note that the evaluation of the fitness of a segmental oscillator is only based on the neural activity of the motoneurons because they are the only neurons influencing the muscular activity. This means that the interneurons can have any activity and that solutions with fewer than 8 active neurons can be developed.

Solutions are evaluated by fixed-duration simulations (simulated time of 3000 ms) with asymmetric initial conditions (all left neurons excited). Several simulations are carried out with different levels of excitation (starting from 1.0 and making steps of ± 0.1), in order to determine, for the networks which oscillate, the range of frequencies they can cover.

The mathematical definition of the fitness function is the following:

$$\mathit{Fitness\_1} = \underbrace{\mathit{fit\_var} \cdot \mathit{fit\_reg} \cdot \mathit{fit\_anti\_phase}}_{\mathit{fit\_oscil}} \cdot \mathit{fit\_freq} \cdot \mathit{fit\_connectivity} \in [(0.05)^5, 1]$$

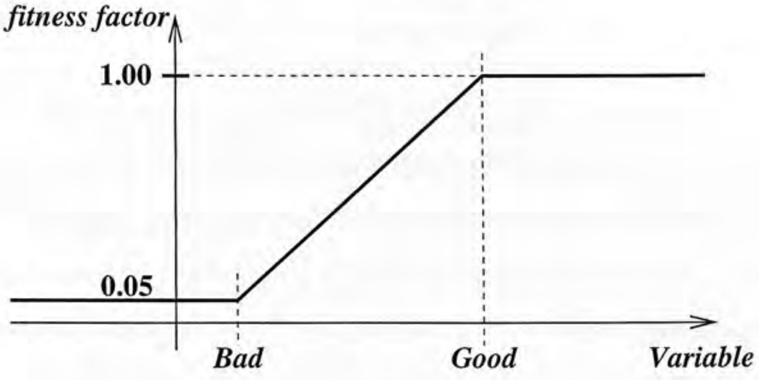


Figure 4.4: Transformation function used for calculating the different fitness factors. Note that the *good* boundary need not necessarily be greater than the *bad* boundary.

where

$$\begin{aligned} \mathit{fit\_var} &= \frac{f_1 + f_2}{2} \\ \mathit{fit\_reg} &= \frac{f_3 + f_4 + f_5 + f_6}{4} \end{aligned}$$

Each fitness factor is limited between 0.05 and 1.00. The factors *fit_anti_phase, fit_freq, fit_connectivity*, and f_i vary linearly between 0.05 and 1.00 when their corresponding variables vary between two boundaries, a *bad* and a *good* boundary, respectively (Figure 4.4). This transformation is obtained by the following function:

$$F(x) = 0.95 \cdot \frac{x - G}{G - B} + 1$$

where B and G are the *bad* and *good* boundaries. Note that the *good* boundary need not necessarily be greater than the *bad* boundary. The variables of these factors and their boundaries are given in Table 4.2.

The function *fit_oscil* is made of three factors which reward activity of the motoneurons which is varying (*fit_var*), regular (*fit_reg*) and out of phase between left and right motoneurons (*fit_anti_phase*). These factors are functions of statistical measures of the motoneuron signals (Tables 4.2 and 4.3), and the boundaries for the transformation function for each variable have been determined by hand on 40 examples of different behaviours from initial experiments. It is possible to fix these boundaries such that *fit_oscil* clearly makes the difference between interesting solutions and the others: a limit of 0.45 is determined above which a solution is certain to oscillate regularly with

Function	Variable	[*bad,good*] Boundaries
f_1	Mean number of zeros	[3,8]
f_2	Mean standart deviation	[0.1,0.5]
f_3	Left-right period difference	[0.15, 0.00]
f_4	Consecutive period difference	[0.15, 0.00]
f_5	Signal difference in consecutive cycles	[0.40, 0.05]
f_6	Signal difference between left and right bursts	[0.40, 0.05]
fit_anti_phase	Left-right difference	[0.0,0.8]
fit_freq	Oscillation frequency	[1.0,12.0] Hz
fit_connectivity	Connectivity ratio	[1.0,0.3]

Table 4.2: Variables and boundaries for the fitness function. See Table 4.3 for the mathematical definition of some of these variables.

Variable	mathematical definition
Mean standart deviation	$\frac{\sum_{t=1}^{N}(U_l(t)-\bar{U_l})^2}{2N} + \frac{\sum_{t=1}^{N}(U_r(t)-\bar{U_r})^2}{2N}$
Left-right period difference	$\frac{\sum_{cycle=1}^{C} P_l(cycle)}{2C} + \frac{\sum_{cycle=1}^{C} P_r(cycle)}{2C}$
Consecutive period difference	$\frac{\lvert P_l(C)-P_l(C-1)\rvert}{2\bar{P}} + \frac{\lvert P_r(C)-P_r(C-1)}{2\bar{P}}$
Signal difference in consecutive cycles	$\frac{\sum_{t\in\text{last\_cycle}} \lvert U_l(t)-U_l(t-\bar{P})\rvert}{\sum_{t\in\text{last\_cycle}}(U_l(t)+U_l(t-\bar{P}))}$
Signal difference between left and right bursts	$\frac{\sum_{t\in\text{last\_cycle}} \lvert U_l(t)-U_r(t-\bar{P}/2)\rvert}{\sum_{t\in\text{last\_cycle}}(U_l(t)+U_r(t-\bar{P}/2))}$
Left-right signal difference	$\frac{\sum_{t=1}^{N} \lvert U_l(t)-U_r(t)\rvert}{\sum_{t=1}^{N} \lvert U_l(t)+U_r(t)\rvert}$
Connectivity ratio	$\frac{\text{N\_Connections}}{\text{N\_Max\_Connections}}$

Table 4.3: Mathematical definition of variables. N and C are the numbers of integration steps and simulated cycles, respectively. U_l and U_r are the outputs of left and right motoneurons. $P_l(j)$ is the period of cycle j for the left motoneuron (cycles start at the onsets of the burst).

opposite behaviour between left and right motoneurons.[2] The value of *fit_oscil* of the fitness function is measured at the default excitation level 1.0.

The factor *fit_freq* rewards solutions which can oscillate over a large range of frequencies when the excitation from the brain stem is varied. The range of frequencies is only measured if the solution oscillates regularly and with opposite behaviour between left and right at the default level of excitation (1.0), i.e. if *fit_oscil* > 0.45. The frequency range, the ratio between maximum and minimum frequencies, is measured by making a set of simulations at different levels of excitation (steps of ± 0.1). Only ranges in which the frequency and the amplitude of the motoneuron signals increase monotonically with the excitation level are rewarded.[3] A function similar to that used to calculate *fit_oscil* is then used to check whether the oscillations are regular at these different excitation levels. Note that because the duration of a simulation is fixed and that a minimum number of oscillations is required for verifying the stability of the signals, the lowest measurable frequency is approximately 1.0 Hz.

The function *fit_connectivity* rewards solutions with reduced connectivity. It depends on the connectivity ratio (*con_ratio*), the ratio between the number of connections and the maximum number of possible connections (56). The smaller the connectivity the higher the reward, with a maximum reward of 1 for solutions with a connectivity ratio smaller than 0.3 (fewer than 18 connections).

Note that this fitness function is the result of an incremental design of the fitness function with new terms and factors being added for correcting small flaws of initial evolutions (see [Ijspeert 96] for a description of that incremental design). This has led to a complexity of the fitness function which is partly unnecessary and initial tests show that simpler, more compact, fitness functions could be designed yielding similar results. The overcomplex fitness function is here described because it was used to develop the oscillators on which the next evolutionary stages are based.

[2] Of the 40 examples of neural activities, 17 correspond to interesting behaviours. With the chosen fixed bounds of *fit_var*, *fit_reg* and *fit_anti_phase*, all the good behaviours have a *fit_oscil* value higher than 0.6 and all the others have a value lower than 0.25, with most lower than 0.1.

[3] The condition that the amplitude of the neural activity should not decrease when the frequency increases is added to prevent an antagonistic effect on the speed of swimming, as tests with the mechanical simulation showed that, in general, increases of the frequency or the amplitude of the motoneuron signals increase the speed of swimming.

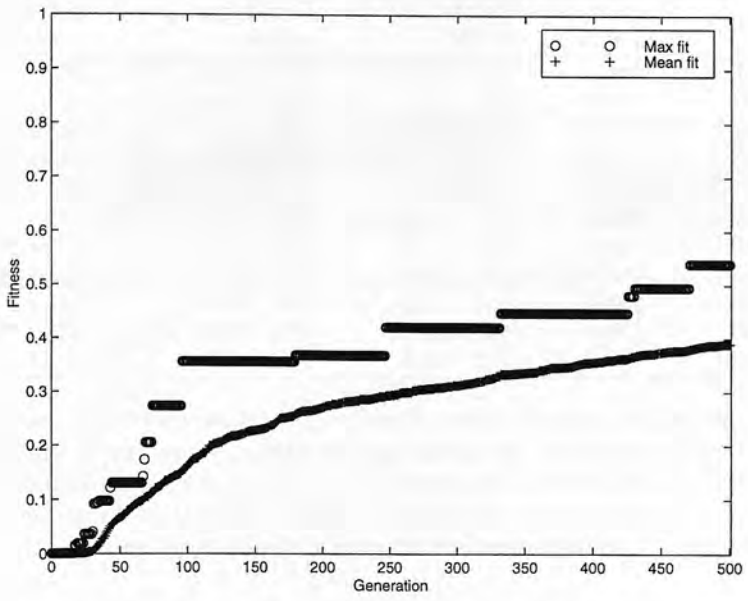


Figure 4.5: Typical evolution of a segmental oscillator (run7).

4.2.4 Results

I carried out 10 evolutions with populations of 100 chromosomes. The evolutions were stopped after 500 generations, an arbitrarily chosen limit which was sufficient for all the runs to produce interesting oscillators (see Figure 4.5 for an example of evolution). Each evolution started with a different randomly generated initial population. When the neurons are randomly connected, there is little chance of the resulting network oscillating. In my case, only 3 solutions within the 1000 initially generated solutions produced regular oscillations, and this only at a fixed frequency. Within 500 generations, all evolutions converged to final populations composed of networks producing regular oscillations with variable frequency. Similarly to the biological model, the frequency of oscillations and the amplitude of the motoneuron signals increase with the level of tonic excitation (applied through the connections from the brainstem). When the oscillator receives too little excitation, the motoneurons stay inactive, and when the excitation level is too high they stabilize, after a few oscillations, in an asymmetrical state with one active and one inactive motoneuron. The fittest solutions of all runs cover a larger range of frequencies than the biological model, reaching lower and higher frequencies. The results are summarized in Table 4.4.

	Fitness value	Frequency range in [Hz]	Oscillating neurons	Number of connections
Biol. model	0.18	[1.7, 5.6]	EIN+,CIN−,LIN−	26
run1	0.44	[1.0, 8.4]	EIN−,CIN+,LIN−	32
run2	0.39	[1.0, 6.8]	EIN−,LIN−	34 → 24
run3	0.43	[1.1, 7.7]	EIN−,CIN−	34 → 26
run4	0.47	[1.2, 9.9]	EIN+,CIN−,LIN−	38
run5	0.40	[1.1, 7.9]	EIN−,CIN−,LIN−	34
run6	0.51	[1.1, 11.3]	EIN−,CIN−	40 → 24
run7	0.54	[1.2, 8.0]	EIN−,CIN−	32 → 24
run8	0.31	[1.1, 5.8]	EIN−,CIN−	36 → 22
run9	0.60	[1.0, 10.4]	EIN−,CIN+,LIN−	32
run10	0.44	[1.3, 8.6]	EIN−,CIN−	28 → 22

Table 4.4: Summary of results for the evolved segmental networks. The table gives, for the best solution of each population, the lowest and the highest frequency at which the segmental network can oscillate, the interneurons active in the oscillator with the sign of their influence, and the number of connections of the oscillator. When fewer than 6 interneurons are active, the number of connections after complete removal of the inactive neurons are given.

The ten runs did not all converge to the same network configuration, but rather to several different solutions with similar fitness values (see Appendix A for a description of the evolved configurations). These best solutions differ not only in their connectivity, but also in the sign of the interneurons and in the number of neurons active in the creation of oscillations. Six evolutions converged to networks with only 6 rather than 8 active neurons. These solutions are composed of the two motoneurons and four inhibitory interneurons of two different types which create the oscillations. This demonstrates that excitatory interneurons are not necessary (at least in a connectionist model) for the production of oscillations over a large frequency range. Most often the inactive neurons are the LINs, which is probably due to their high threshold: they have more chance to be inactive unless they receive some strong excitatory input. Amongst these six runs, four (runs 3,6,8 and 10) converged to similar configurations based on an identical oscillator structure (when the possible permutations due to the left-right symmetry are taken into account) composed of inhibitory EIN and CIN and the motoneurons (Figure 4.6). Although there are some variations of the values of the connection weights, the solutions have all the same connections with similar strengths and present the same behaviour. Figure 4.7 shows the simulation and the configuration of one of them.

Although most of the evolved oscillators differ significantly from the biological segmental network, one controller (the best solution of run4) presents strong similarities

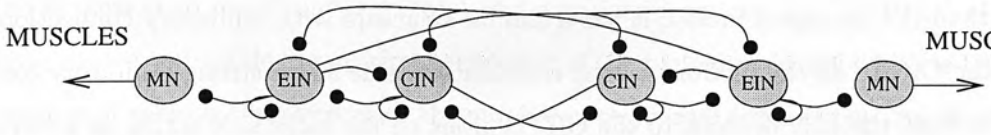
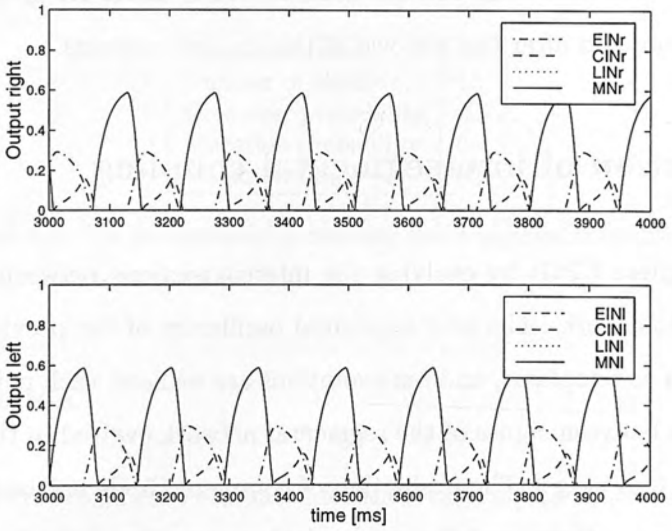


Figure 4.6: Segmental network configuration to which four out of ten runs converged (runs 3,6,8 and 10).



from: / *to:*	EINl	CINl	LINl	EINr	CINr	LINr	BS
EINl	−1.9	−5.0	-	−4.2	−0.1	-	12.1
CCINl	−2.8	−1.7	-	−5.0	−2.1	-	4.7
LINl	-	-	-	-	-	-	-
MNl	−5.0	-	-	-	-	-	4.8
EINr	−4.2	−0.1	-	−1.9	−5.0	-	12.1
CCINr	−5.0	−2.1	-	−2.8	−1.7	-	4.7
LINr	-	-	-	-	-	-	-
MNl	-	-	-	−5.0	-	-	4.8

Figure 4.7: Evolved segmental network (run 6). *Top:* Neural activity, *bottom:* Neural configuration.

with it. That solution is composed of the same interneurons — same number and with the same sign — as the biological model. Furthermore, although it has more connections than the biological model, it has a similar structure with inhibitory connections from the CIN to all the neurons on the contralateral side and a strong inhibitory connection from the LIN neurons to the CIN neurons on the same side acting as a burst terminator (as with the biological model, the LIN becomes active later in the cycle and inhibits the ispilateral CIN allowing the activity on the other side to start).

The pruning operator of the GA has led to solutions using significantly fewer than the 56 possible connections. The solutions have therefore few weak (low weight) connections which do not play an important role in the creation of oscillations. In the case of solutions which use fewer than 8 neurons, the majority of the connections to and from the inactive neurons have been cut; for these solutions Table 4.4 gives the number of connections before and after the removal of the inactive neurons.[4]

4.3 Evolution of intersegmental coupling

I generate complete CPGs by evolving the interconnections between 100 copies of a fixed segmental network. The best segmental oscillators of the previous evolutionary stage are chosen as templates, and ten evolutions are realised, each being the evolution of the couplings between copies of the segmental network evolved in the corresponding ten runs of the first stage. The evaluation of complete CPGs is based on simulations of both the neural activity and the mechanical movements of the body. The quality of the solutions will depend on their capacity to control swimming at different speeds, frequencies of oscillation, and lags between segments.

4.3.1 Encoding and GA

A chromosome encodes how the segments are interconnected. Similarly to the biological model, interconnections are extensions of segmental connections to the corresponding

[4] The fact that there are still some weak connections which could be removed without affecting the oscillation is probably due to the fact that the mutation operator can at any time give a non-zero value to a weight which had previously been set to zero. The mutation operator therefore slows down convergence to the minimal set of connections.

post-synaptic neurons in neighbouring segments. For each segmental connection, the extensions in each direction are encoded into a chromosome with values between 0 (no extension to neighbour segments) and a maximal extension of 12 (this value has been chosen to include the maximal extension of the biological model which is 10). The number of genes is therefore twice the number of segmental connections. As previously, a left-right symmetry is assumed. The connection weights of the segmental networks are fixed except for a rescaling depending on the extension of the interconnection: the weight of a connection to a neuron is divided by the number of segments it receives input from.

The GA described in Section 4.1.1 is used, without the pruning operator. Although the genes are transformed into integers when the chromosome is decoded, they are real numbers in the GA. The GA parameters for the evolutions are given in Table 4.5.

Population size	40
Number of children	12
Crossover probability	0.5
Mutation probability	0.4
Mutation range	0.2

Table 4.5: GA parameters for evolving multi-segmental controllers.

4.3.2 Evaluation

Multi-segmental controllers are evaluated for their capacity to control swimming in the mechanical simulation. The required features for the controllers are:

1. generation of stable oscillations in the 100 segments with coordinated phase differences for the creation of traveling undulations of the body,
2. ability to change the speed of swimming by changing either the frequency of oscillation or the wavelength of the undulation,
3. ability to change the frequency and the wavelength independently, by changing, respectively, the global excitation level and the amount of extra excitation on the most rostral segments.

In order to allow comparisons with the biological controller, special emphasis is given to the capacity to swim with a wavelength corresponding to the length of the body (phase lag per segment of 1%), and therefore the capacity to change the frequency over a large range is only measured for lags close to 1%.

The mathematical definition of the fitness function is the following:

$$\mathit{Fitness\_2} = \mathit{min\_fit\_oscil} \cdot \mathit{fit\_lagcontrol} \cdot \mathit{fit\_freqcontrol} \cdot \mathit{fit\_speed}$$

where

- *min_fit_oscil* is the minimum *fit_oscil* value (see section 4.2.3) between segments 1,10,20,..,100 when the controller is simulated (simulations of 3000 ms) with an excitation level of *excit0*, corresponding to the middle of the range of excitations with which the segmental oscillator can oscillate, and without extra excitation on the first segments.
- $\mathit{fit\_lagcontrol} = \begin{cases} 0.05 + \frac{\mathit{lag\_range1}}{1+\mathit{freq\_range1}} & \text{if} < 1 \\ 1 & \text{otherwise} \end{cases}$
  Lag_range1 and *freq_range1* are measured by making several simulations at a fixed level of excitation (*excit0*) and with an increasing amount of extra excitation. The lag range is non-zero only if the oscillations are regular in all segments (*min_fit_oscil*>0.45) and if the lag increases monotonically with the amount of extra excitation. The lag range is divided by the frequency range in order to reward solutions which can change the lag between segments independently of the frequency.
- $\mathit{fit\_freqcontrol} = \begin{cases} 0.05 + \frac{\mathit{freq\_range2}}{1+\mathit{lag\_range2}} & \text{if} < 1 \\ 1 & \text{otherwise} \end{cases}$
  Freq_range2 and *lag_range2* are measured by making several simulations with excitation levels varying around *excit0* and with a fixed amount of extra excitation *extra0*. The value *extra0* is calculated from the previous lag measurement if *lag_range1* includes a lag of 1%. *Extra0* is then defined by taking the value of extra excitation corresponding to the lag closest to 1%. The frequency range is non-zero only if the value *extra0* exists, and if the frequency increases monotonically with the level of excitation.

- $fit\_speed = \begin{cases} 0.05 + speed\_range & \text{if} < 1 \\ 1 & \text{otherwise} \end{cases}$
  Speed_range corresponds to the range of speeds covered by all the simulations made for the definition of *lag_freqcontrol* and *fit_freqcontrol*.

The different value ranges are measured as the difference between the maximum and the minimum value and are normalised by a value corresponding to a target value. The targets have been fixed to 2.5% for the lag ranges, 10.0 Hz for the frequency ranges and 0.8 m/s for the speed range.

4.3.3 Results

I carried out 10 evolutions based on the best evolved segmental oscillators of the previous evolutionary stage (i.e. run 1 in this stage evolves the couplings between copies of the best segmental oscillator of run 1 of the previous stage). Each population is made of 40 chromosomes and starts with randomly created chromosomes (i.e. random couplings between the fixed oscillators). Evolutions are stopped when the fitness of the fittest solution of the population ceases to increase significantly (between 75 and 300 generations depending on the run).

The random coupling of the initial populations has a different effect depending on the segmental network. Some segmental networks are quite robust and produce stable oscillations despite the random couplings, but most of them fail to produce stable oscillations. A general observation is that, among the randomly connected segmental oscillators which can produce stable oscillations, very few can have the phase between segments varied by the extra excitation on the first segments.

All runs successfully converged to controllers capable of generating stable oscillations with phase lags between segments for relatively large ranges of frequency and phase lags. Similarly to the biological model, the evolved CPGs produce lags between segments which are constant over the whole spinal cord, except for the extremities (because of the lack of extensions from the boundaries, the segments at the extremities tend to oscillate in phase). The lags increase when extra excitation is applied to the most rostral segments. After the evolutions, I tested the best solutions of each run over a whole range of levels of excitation and extra excitation in order to determine

	Fitness value	Frequency range in [Hz]	Relative lag range in [%]	Speed range in [m/s]
Biol. model	0.08	[1.6, 5.6]	[0.0, 2.4]	[−0.03, 0.50]
run1	0.07	[1.9, 8.5]	[0.0, 1.9]	[−0.04, 0.59]
run2	0.09	[1.0, 6.8]	[0.1, 3.1]	[0.06, 0.58]
run3	0.07	[1.1, 7.8]	[0.0, 2.0]	[0.08, 0.49]
run4	0.13	[0.8, 10.1]	[0.0, 3.8]	[−0.03, 0.53]
run5	0.26	[1.1, 8.4]	[0.0, 3.6]	[−0.10, 0.55]
run6	0.02	[1.1, 7.0]	[0.0, 1.3]	[0.07, 0.51]
run7	0.02	[1.2, 7.5]	[0.0, 2.0]	[−0.03, 0.52]
run8	0.10	[1.1, 5.8]	[0.0, 3.0]	[−0.03, 0.56]
run9	0.13	[2.0, 11.0]	[−1.7, 3.5]	[−0.07, 0.50]
run10	0.18	[3.3, 8.7]	[0.3, 2.2]	[0.18, 0.60]

Table 4.6: Summary of results for the evolved multi-segmental controllers. The table gives, for the best solution of each population, the ranges of frequency, phase lag and speed it can produce.

their exact ranges of speed, frequency and phase lag. The results are summarized in Table 4.6. Figure 4.8 shows the effect of different levels of global and extra excitation on the frequency, phase lag and speed for one of them (best of run9).

The multisegmental controllers cover approximately the same range of frequency as the corresponding isolated segmental networks. Five evolved controllers cover a larger range of phase lags than Ekeberg's solution, with a maximum lag of 3.8% for the solution of run4. The speed range covered by the evolved solutions is also generally larger than the range of Ekeberg's model (8 solutions reach higher speeds). Note that some solutions swim slowly backwards when all segments oscillate in phase (no extra excitation on the first segments), because of the kind of wriggling they then perform. Also, some solutions produce lags even without extra excitation of the first segments.

Although most solutions cover larger ranges of frequencies and phase lags than the biological model, the independence of control of these variables is usually less good. For several solutions, for instance, increasing the global level of excitation not only increases the frequency of oscillations but also influences the relative lag between segments (usually decreasing it). This problem is illustrated by the best solution of run9 (see Figure 4.8) which can reach high frequencies and high lags, but not both together, which explains why its maximum speed is relatively low (a combination of high frequency and high lag is necessary for high speeds in my mechanical simulation).[5] My

[5] Note that the best solution of run9 also has the idiosyncrasy of producing, for some levels of global excitation, a caudal to rostral traveling wave when it receives no extra excitation. As soon as some

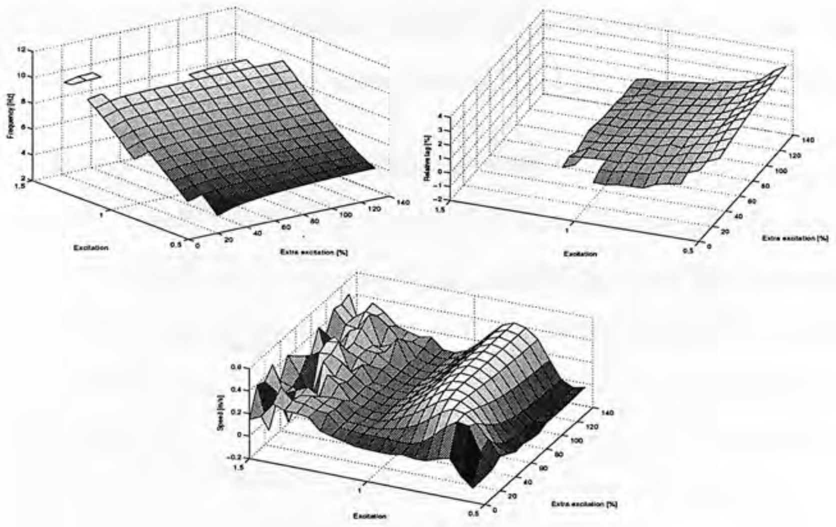


Figure 4.8: Effect of the excitation on the controller evolved in run9. *Top:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

fitness function has not prevented this relative lack of independence because, although it rewards independence, it does it only for the small area of the global excitation versus extra excitation space that is visited for the evaluation. The dependence occurs mainly at high levels of global excitation and extra excitation which are not visited for the evaluation. This relative problem could probably be solved by sampling more combinations of both types of excitation.

There are no clear conclusions to make from the way the segmental networks are coupled (see Appendix B for a description of the evolved configurations). The extents of the interconnections vary considerably from one solution to another, and there is, for instance, no systematic coupling favouring one direction. The extents of the coupling and the asymmetries vary depending on the types of the connected neurons, the influence of the connection, the phase differences between the activities of the neurons within the segmental oscillator, *etc.* A few general observations can, however, be made. On average, the ipsilateral connections between neurons of the same type (the self-connections) extend more rostrally than caudally while the contralateral connections between neurons of the same type extend more caudally. Also a large majority of the inhibitory connections sent to the motoneurons extend more caudally. Finally

extra excitation is given, the undulation travels again from head to tail.

the CPGs based on similar segmental oscillators have also similar couplings, with the same asymmetries of coupling for most segmental connections.

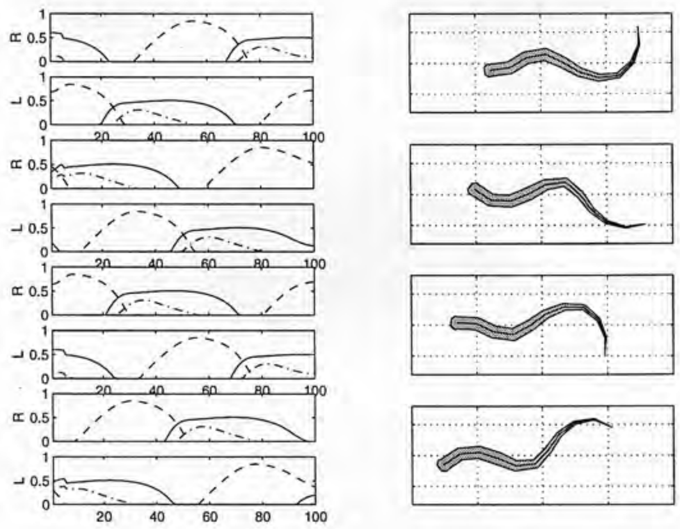


Figure 4.9: Simulation of one of the evolved swimming controllers (best of run7). The frequency of oscillations is 4.0 Hz, the phase lag between segments is 1.1% of a cycle duration and the speed of swimming is 0.41 m/s. *Left:* Neural simulation, the continuous line correspond to the MN, the dashed line corresponds to the CIN and dotted-dashed line to the EIN. *Right:* Mechanical simulation, there are 60 ms between each snapshot, and the dotted vertical lines are separated by 100 mm.

Figure 4.9 shows a simulation of one of the evolved multi-segmental controllers. Interestingly, the evolved CPGs can also perform turning although this aspect has not been taken into account in the evaluation functions. Similarly to the biological controller, giving a higher excitation level to one side of the spinal cord leads to a difference of the motoneuron amplitude and, because of stronger muscle contractions on one side, to a change in the direction of swimming. In the case of the evolved controllers, some left-right permutations of the interneurons may be necessary for optimising the turning ability (because of the left-right symmetry, these permutations have no effect on straight swimming). For instance, for avoiding antagonistic effects on the motoneuron amplitude when extra excitation is applied, the neurons sending inhibitory connections to a motoneuron should be on the contralateral side. Finally, initial tests on some of the evolved controllers show that it is possible to produce backwards swimming by giving more excitation to the most caudal segments and therefore inducing a tail to head undulation, but only very limited lags and speeds can then be obtained.

4.4 Evolution of sensory feedback from stretch sensitive cells

The CPGs evolved so far receive no feedback from the mechanical simulation. In the real lamprey, sensory feedback is provided to the CPG by stretch sensitive cells (the edge cells) situated on both side of the spinal cord. There are both inhibitory and excitatory edge cells and they emit signals proportional to their elongation [Viana Di Prisco *et al.* 90]. Ekeberg showed [Ekeberg *et al.* 95] that this sensory feedback is necessary for swimming in unstationary water and, in particular, for crossing a barrier of water with local speed opposite to the direction of swimming. I show here a preliminary experiment on how to evolve this sensory feedback for the artificial controllers in order to allow them to cross a similar speed barrier.

4.4.1 Encoding and GA

A chromosome encodes how each segmental oscillator receives sensory feedback from two stretch sensitive cells situated at the sides of the corresponding part of the spinal cord. It encodes the synaptic weights of the connections from one edge cell to the 8 (or 6) neurons of the corresponding segmental network. I impose symmetry between both sides of the segmental network as well as over the whole spinal cord (the 100 segments). Each connection can be either excitatory or inhibitory (this is to represent the fact that there are both inhibitory and excitatory edge cells) and genes are decoded into synaptic weights between -2 and 5. Note that I have fixed these boundaries to be relatively small in order to prevent the sensory feedback disturbing the production of oscillations. The same GA with the same parameters as for the first evolutionary stage is used (Table 4.1), but without the pruning operator.

4.4.2 Evaluation

Solutions are rewarded depending on their ability to cross a speed barrier, that is, an area with an increase of the speed of the water. Such a situation could, for instance, correspond to a lamprey swimming up a river and crossing an shallow area (Figure 4.10). The barrier is 150 mm wide (half the length of the simulated lamprey) and the speed of the water flow opposite to the direction of swimming is set to be 40% higher than the speed of swimming of the lamprey.

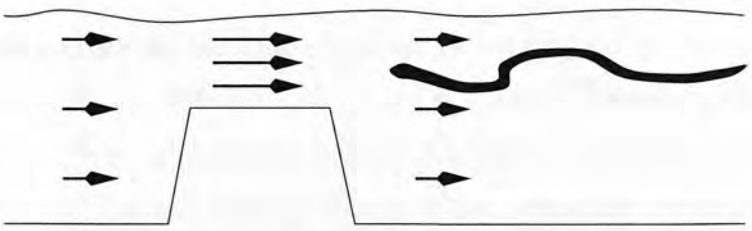


Figure 4.10: Speed barrier due to a shallow passage in a river. Following Bernouilli's equation, a reduction of the cross section of the water flow leads to an increase of the water speed for keeping the flow constant.

The fitness function is defined to reward:

1. progression through the barrier in the direction of swimming,
2. minimal deviation in the direction,
3. minimal difference of speed between swimming with and without sensory feedback.

This last point was added to prevent important changes in the swimming patterns when sensory feedback is given to the controller. Evaluations are realised at a chosen combination of frequency of oscillations and phase lag between the segments (approximately in the middle of their respective ranges).

The mathematical definition of the fitness function is:

$$\mathit{Fitness\_3} = \mathit{fit\_progression} \cdot \mathit{fit\_deviation} \cdot \mathit{fit\_speed\_constancy}$$

where

- *fit_progression* varies linearly between (and is limited to) 0.05 and 1 when the maximum progression of the head of the lamprey varies between 0 and 450 mm (width of the speed barrier plus the body length). The progression is measured as the distance (in the initial direction of swimming) between the head of the lamprey and the entry point to the barrier,

- *fit_deviation* varies linearly between 0.05 and 1 as the cosine of the deviation angle (measured between the initial direction of swimming and the direction after entry into the barrier) varies between 0 and 1,

- *fit_speed_constancy* varies linearly between 0.05 and 1 when the relative difference between the speeds with and without sensory feedback ($|speed\_with - speed\_without|/speed\_without$) varies between 50% and 0%.

4.4.3 Results

I carried out two sets of 5 runs (populations of 40 chromosomes) using the best controller of run7 of the previous evolutionary stage[6]. In the first set of runs, sensory feedback is evolved to all 6 neurons forming the segmental oscillator (2 EINs, 2 CCINs and 2 MNs), while in the second set sensory feedback is only provided to the interneurons. The chosen levels of excitation produce swimming (without sensory feedback) at a frequency of oscillation of 4.0Hz, a relative lag of 1.1% and a corresponding speed of 0.41 m/s. The speed of the barrier was set to 0.57 m/s (40% higher than the speed of swimming). Without sensory feedback, the lamprey is not able to cross the speed barrier (Figure 4.11, top). The increase in local forces at the entrance to the barrier leads to extra bending of the rostral part of the body and forces the lamprey to change its direction of swimming. The lamprey eventually swims perpendicularly to its initial direction.

In the first set of runs (sensory feedback to both interneurons and motoneurons), all runs converged within 80 generations to controllers able to cross the speed barrier (see Appendix C for a description of the evolved configurations). The connections from the stretch sensitive cells are not identical in all the evolved controllers. All solutions have, however, evolved excitatory connections from the stretch sensitive cells to the interneurons inhibiting the motoneurons on the contralateral side, therefore preventing excessive bending. The solutions have also developed strong excitatory connections from the stretch sensitive cells to the motoneurons. Interestingly, these connections are not only ipsilateral (to prevent excessive bending) but also contralateral. The

[6] This controller was chosen because it combines relative large ranges of frequencies, lags and speeds with an independence of control of the frequency and the lag similar to the biological model.

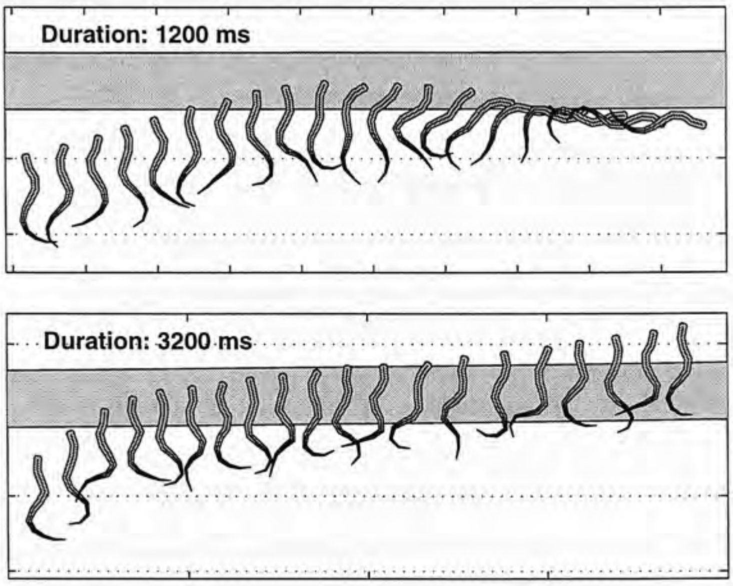


Figure 4.11: Effect of a speed barrier on swimming without (*top*) and with (*bottom*) evolved sensory feedback.

overall effect is that the amplitudes of the motoneuron signals are approximately 20% larger than without sensory feedback. The excitatory feedback to the motoneurons leads thus to a stiffer body of the lamprey, which is therefore less sensitive to the extra forces due to the barrier.[7]

In the second set of runs, this direct feedback from the stretch sensitive cells to the motoneuron is not allowed. In a first try with the speed barrier of 0.57m/s, none of the runs converged (within 50 generations) to controllers capable of crossing the barrier. The speed was therefore reduced to 0.53m/s (30% larger than the speed of swimming without sensory feedback). After 50 generations, 2 of the 5 runs converged to solutions capable of crossing the barrier[8]. The controllers of the two successful runs are able to cross the speed barrier without increase of the amplitude of the motoneurons. These controllers therefore manage to cross the barrier without increasing the stiffness

[7] Note that the lamprey is able to cross a barrier with a higher speed than its swimming speed, firstly because the barrier's width is only half the length of the lamprey, and secondly because the increase of inertial forces due to the water (i.e. the negative acceleration) needs some time to "brake" and invert the speed of the lamprey.

[8] The three unsuccessful runs had a premature convergence to controllers which, although they crossed most of the barrier, were unable to cross it completely. The loss of diversity in the chromosomes meant that the GA could not improve those local maxima.

of the body but only by correcting the phases of the motoneuron oscillations. The best controllers of both runs have, for instance, stretch sensitive cells which inhibit the inhibitory interneuron connected to the motoneuron on the same side, and which excite the inhibitory neuron connected to the motoneuron on the contralateral side. The stretch senstive cells therefore act as burst terminators which tend to switch the motoneuron activity from one side to the other when the latter is too much extended.

4.5 Discussion of the results

By developing efficient CPGs which can produce coordinated oscillations for the swimming of a simulated lamprey, the GA has proved to be an interesting design technique. I will here review the method and discuss the evolved controllers.

4.5.1 Discussion of the Method

Controllers were evolved in different stages in order to simplify the problem by decomposing it into subproblems. The decomposition was motivated by similar decompositions in the real lamprey and gave the possibility of direct comparison between evolved solutions and the biological model. One advantage of such a decomposition is to reduce the time needed to evolve a complete controller by avoiding evaluation of a whole controller made of segmental oscillators which do not oscillate correctly even when isolated. Decomposition presents the risk, however, of misleading the evolution as the fittest solutions of one stage may not necessarily be the best starting elements for the next stage. The segmental oscillators evolved in the first stage, for instance, were not rewarded for their capacity to be interconnected, which may have constituted a handicap for the next stages. The pros and cons of staged evolution will be further discussed in chapter 9.

The simple encoding scheme which directly encodes synaptic weights and extents of interconnections into a chromosome meant that search was performed in a well defined and limited space of possible solutions (fixed neuron models, fixed maximum number of neurons and left-right symmetry). Despite these limitations, the multidimensional search space was still large enough to generate a variety of interesting central pattern generators. One way to enlarge the search space could be to include the parameters of the neuron dynamics into the encoding. Initial tests show that the GA is able to evolve both these neuron parameters and the connectivity and generate working CPGs. A larger search space will also be visited in chapter 6, where swimming controllers will be evolved with a developmental encoding.

Defining the fitness functions probably constitutes the most important part of the method. My objectives were to develop CPGs capable of producing stable oscillations and of modulating the speed of swimming by varying, through a few external signals, the frequency and the phase lag of the oscillations. Once the objectives were defined, relatively little iteration was needed to define a suitable set of fitness functions. The functions are products of factors between 0.05 and 1.0 rewarding qualitative and quantitative aspects, which means that solutions which perform evenly in all aspects receive the highest fitness value. Both neural (e.g. the motoneuron frequency) and mechanical (e.g. the speed of swimming) variables are rewarded, with the importance of neural aspects decreasing and that of mechanical aspects increasing over the evolutionary stages (in the first stage only neural aspects are rewarded, in the second stage, both neural and mechanical aspects are considered and in the third stage only mechanical aspects are taken in account). Ideally, only mechanical aspects should be examined as they alone effectively determine the quality of a swimming controller; rewarding neural activities presents the risk of biasing the search to particular network architectures. However, in this case, taking in account neural variables was necessary for allowing the staged evolutionary approach discussed above.

The interesting properties of using evolutionary algorithms for designing neural controllers compared to more traditional learning algorithms will be discussed in chapter 9.

4.5.2 Discussion of the evolved controllers

Several interesting CPGs have been successfully evolved. A first observation is thus that there exists a variety of possible connectivities which produce the neural activities necessary for swimming.[9] I have demonstrated that there exist solutions alternative

[9] A similar observation was made by Lockery for the leech bending reflex, where many different networks could produce a physiologically accurate local bending input-output function

to the biological CPG which can control swimming with the same efficiency. These solutions differ from Ekeberg's biological model in terms of which neurons are interconnected, of the excitatory or inhibitory influence of the interneurons and even of the number of neurons involved in the rhythmogenesis.

A diversity of artificial segmental oscillators were evolved with significantly larger ranges of frequencies than Ekeberg's segmental oscillator. Although one oscillator has a similar structure to the biological one, most evolved networks have interneurons that play other roles than in the biological network and have other signs. There is therefore no preferred role and sign for each dynamics of the different neuron models. A majority of oscillators are based on only four inhibitory interneurons, showing that, at least at the connectionist level, excitatory interneurons are not necessary for the production of oscillations over large frequency ranges.

The evolution of multi-segmental oscillators showed that segmental networks could be interconnected in several ways and produce stable oscillations with the necessary phase lags for swimming. Similary to the biological model, swimming at different speeds and with different wavelengths of undulations can be produced by varying simple external signals (the excitation applied to the CPG). Turning can also be induced when one side of the controller receives more excitation than the other. The evolved controllers can generally cover larger ranges of frequency, phase and speed than Ekeberg's model but with less independence in the control of the frequency and the phase between segments. A general observation on the coupling between segments is that there is no systematic asymmetry of interconnection favouring one direction but rather asymmetries in both directions which depend on the segmental connection. Note that the coupling of both Ekeberg's and the evolved controllers is a little bit crude in the sense that, except for the segments at the extremities, the synaptic weights of the extensions of a segmental connection are the same for all connected segments. It could be interesting to evolve, for each coupling, a normalising function which would determine the synaptic strengths depending on the length of the projection, in order to allow, for instance, a decrease of the influence of a segment on another with the distance separating them.

The evolutions of connections from stretch sensitive cells illustrates how sensory feed-

[Lockery & Sejnowski 93a].

back can be used by the evolved CPGs for coordinating the neural activity with the actual movements of the body and allowing the lamprey to cross a speed barrier. The feedback increases the stiffness of the body and acts as a burst terminator which switches the neural activity to the side which is excessively bent. These preliminary experiments were realised with fixed external commands to the CPG (swimming straight) and the crossing of the barrier was therefore due to the passive properties of the CPG with sensory feedback. As active corrections in the commands may also help the lamprey to cross the barrier, further experiments should include these in order to determine the respective importance of sensory feedback and higher commands from the brain for crossing a barrier. This is an aspect which Ekeberg did not consider, and which I will address with a preliminary experiment in chapter 8.

The variety of different evolved controllers could be studied in more detail and, for instance, it would be interesting to study how robust the evolved controllers are against lesions (destruction of connections and/or of neurons). The fact that a variety of different controllers with similar fitness values have been evolved also rises the question of why the biological configuration is as it is, i.e. if it is just due to the randomness of natural evolution and the little changes compared to previously existing CPGs, or if it is the result of an optimisation of criteria which have not been taken in account in my very simple fitness functions, such as robustness against lesions, stability, energy efficiency, ability to incorporate sensory feedback,... Further evolutions with more elaborate fitness functions may be an interesting way to investigate these questions.

4.6 Summary

This chapter presented how swimming controllers alternative to Ekeberg's biological model of the lamprey, can be evolved using a staged evolution approach and a simple direct encoding scheme. The space of possible solutions was designed to include the biological configuration. It was restricted to alternative solutions which, similarly to the biological model, are made of segmental oscillators which can be made to oscillate when isolated and consist of 100 segmental networks coupling in a way which allows the production of traveling waves of neural activity without needing sensory feedback for intersegmental coupling.

The genetic algorithm could successfully generate a variety of different controllers for producing an anguiliform swimming very similar to that of the lamprey. The controllers differ from the biological configuration and from each other in terms of which neurons are interconnected, of the excitatory ot inhibitory influence of the interneurons and of the number of neurons involved in the rhythmogenesis.

Similarly to the biological controller, the frequency and the wavelength of the traveling undulations, and therefore the speed of swimming, can be modulated by varying the tonic input applied to the network. Turning can also be induced when left-right asymmetrical input is applied. Finally sensory feedback from stretch sensitive cells is evolved which enables the lamprey to cross a speed barrier it would not have crossed without.

The genetic algorithm could successfully generate a variety of different controllers for producing an anguilliform swimming very similar to that of the lamprey. The controllers differ from the biological configuration, and from each other in terms of which neurons are interconnected, of the excitatory or inhibitory influence of the interneurons and of the number of neurons involved in the rhythmogenesis.

Similarly to the biological controller, the frequency and the wavelength of the traveling undulations, and therefore the speed of swimming, can be modulated by varying the tonic input applied to the network. Turning can also be induced when left-right asymmetrical input is applied. Finally, sensory feedback from stretch sensitive cells is evolved which enables the lamprey to cross a speed barrier it would not have crossed without.

The variety of different evolved [illegible] be studied in more detail and, for example, it would be [illegible] the evolved controllers are against noise [illegible]. The fact that a variety of different [illegible] have been evolved also rises the question of why the biological [illegible] is just due to the randomness of natural evolution, [illegible] existing CPGs, or if [illegible] have not been taken in account in [illegible] stability, energy efficiency [illegible]. Further evolutions with more [illegible] investigate these questions.

4.8 Summary

This chapter presented [illegible] Ekeberg's biological model [illegible] and a simple [illegible] scheme. The genetic [illegible] was designed to include the biological configuration. [illegible] which, similarly to the biological model, are [illegible] made to oscillate [illegible] which allows [illegible] sensory feedback [illegible]

Chapter 5

Using a GA for neurobiological modelling

Modeling a biological neural system requires the setting of many parameters linked with the cellular and synaptic properties of the system. With current recording techniques, many of these parameters are difficult or impossible to measure. One important difficulty of modeling is therefore to instantiate all these parameters of the simulation to fit cellular and network properties of the system. This is a hard problem to tackle by hand due to the typical non-linearity of the system and to the fact that the effect of a parameter generally depends significantly on the values of the others. There is therefore a strong motivation for having methods which could instantiate the parameters automatically, given a quantitative or qualitative desired behaviour. I believe that evolutionary algorithms can be powerful tools for solving these kinds of problems, and I demonstrate in this chapter how a GA can used for developing a biologically plausible connectionist model of the swimming CPG of the lamprey.

The same staged approach and direct encoding scheme of the previous chapter are employed with a focus on how they can be used for automatically generating a part of the biological model Ekeberg has designed by hand. Ekeberg spent significant time in setting the different synaptic weights and intersegmental couplings in order to develop a controller which could produce the patterns of oscillation observed in the lamprey (Ekeberg, personal communication). Here the GA is used to automatically instantiate these variables for obtaining a similar behaviour.

Biologically plausible solutions are generated by reducing the search space to solutions which have the biological segmental connectivity observed in the lamprey, and which are made of the same neurons as Ekeberg's model (i.e. the same dynamics and the same sign). The same three stages are used as in the previous chapter. The evolutionary process is used to set the synaptic weights of the segmental network, to visit the space of possible intersegmental coupling between biological segmental networks, and to investigate possible sensory feedback connections to the biological CPG. The research presented here has been partially published in [Ijspeert *et al.* 98b].

The GA is therefore used as a synthetic tool for generating a part of a neurobiological model for fitting physiological observations, and as an analytical tool for investigating possible configurations (e.g. possible intersegmental couplings) which optimise some specified criteria.

5.1 Methods

In this chapter, the same staged evolution and the same fitness functions are used as in the previous chapter. The evolutions are also based on the same genetic algorithm. Compared to the previous chapter, the search space is reduced such that the evolved controllers are composed of interneurons with the same sign (inhibitory or excitatory) as the biological neurons, and present the same segmental connectivity as the one observed in the real lamprey.

In the first stage, only synaptic weights are evolved, and the connectivity and neuron signs are fixed to correspond to those of the lamprey. In the second evolutionary stage, the coupling between 100 copies of Ekeberg's segmental network (not one of the evolved segmental oscillators of the previous stage) are evolved and rewarded for their capacity to produce swimming gaits with variable frequency and phase lags. Finally, in the last stage, the synaptic weights of sensory feedback connections from simulated stretch sensitive cells are evolved for Ekeberg's CPG of the lamprey.

I chose, in the second and third stage, to use elements of Ekeberg's model instead of the corresponding evolved elements in order to allow an easier comparison of the evolved controllers with Ekeberg's model.

5.2 Evolution of segmental oscillators

5.2.1 Encoding, GA, and evaluation

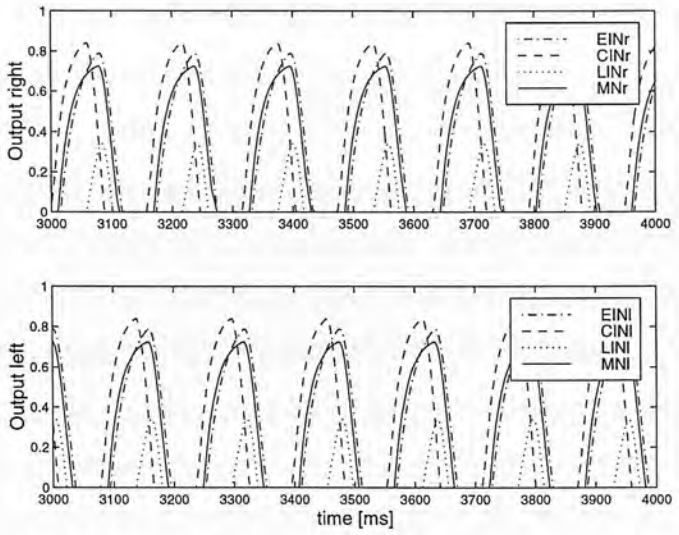
Segmental oscillators with the biological connectivity are evolved. The fitness function (*Fitness_1*) and the GA (with the same parameters, as given in Table 4.1) used for evolving artificial segmental oscillators are reused here with the exception that the chromosome only encodes the connections observed in the real lamprey, i.e. the 26 connections of Ekeberg's model, and that no pruning operator is used. The type of each connection (excitatory or inhibitory) is also fixed and corresponds to the physiological observation. Ten evolutions with populations of 100 chromosomes were carried out for 500 generations.

	Fitness value	Frequency range in [Hz]
Biol. model	0.18	[1.7, 5.6]
run1	0.58	[1.0, 8.6]
run2	0.70	[0.9, 10.5]
run3	0.63	[0.9, 9.5]
run4	0.56	[1.0, 8.3]
run5	0.71	[0.9, 10.8]
run6	0.71	[1.0, 10.3]
run7	0.54	[1.4, 11.6]
run8	0.71	[0.9, 11.0]
run9	0.67	[0.9, 9.8]
run10	0.70	[1.0, 10.6]

Table 5.1: Summary of results for the evolved segmental oscillators with biological connectivity.

5.2.2 Results

All populations converged to best solutions covering larger ranges of frequency than the segmental oscillator with Ekeberg's values (Table 5.1). These evolved segmental oscillators cover most of the observed frequencies of the real lamprey (between 0.25 and 10 Hz), with, for instance, the solution with highest fitness oscillating at frequencies between 0.9 and 11.0 Hz. Figure 5.1 shows the simulation and the configuration of that best oscillator. None converged to the same set of connection weights as Ekeberg's segmental oscillator. Interestingly, five other runs converged to a network similar to that of the best run, with very similar weights (almost identical inhibitory weights)



from: / *to:*	EINl	CINl	LINl	EINr	CINr	LINr	BS
EINl	2.4	-	-	-	−2.3	-	9.6
CINl	10.0	-	−4.1	-	−1.3	-	10.3
LINl	4.2	-	-	-	−4.1	-	6.6
MNl	0.0	-	-	-	−1.8	-	3.4
EINr	-	−2.3	-	2.4	-	-	9.6
CINr	-	−1.3	-	10.0	-	−4.1	10.3
LINr	-	−4.1	-	4.2	-	-	6.6
MNr	-	−1.8	-	0.0	-	-	3.4

Figure 5.1: Simulation (*top*) and configuration (*bottom*) of the best evolved oscillator with biological connectivity (to be compared with Ekeberg's values Table 3.1 pp60). This oscillator can oscillate at frequencies between 0.9 and 11.0 Hz. Five (out of ten) other runs converged to very similar configurations.

and similar fitness values. The corresponding segmental network seems thus to be an important local fitness maximum.

A general observation from these runs is that the increase of the frequency range compared to the original segmental oscillator is due to stronger inhibitory links, especially the inhibitory connections from CIN to contralateral LIN and from LIN to ipsilateral CIN (see Appendix D for the configurations of all evolved oscillators). In many cases (7 out of 10 runs), the excitatory connection from EIN to ipsilateral MN evolved to a zero weight, indicating that that connection can be removed without disturbing the oscillatory capacities of the network (see the configuration of Figure 5.1, for instance).

Interestingly, one of the evolved segmental networks had non-oscillating EINs showing that, even with the biological connectivity, it is possible to remove the excitatory interneurons and still produce oscillations over a large range of frequency. A similar observation was made by Jung [Jung *et al.* 96], who observed that, in a connectionist model similar to this one, stable oscillatory output without EIN neurons could be obtained with appropriate changes in the tonic drives to the LIN and CIN neurons. Also in a biophysical simulation of the segmental network with populations of neurons rather than single neuron units [Hellgren *et al.* 92], it was found that oscillations could be obtained without EIN but which were less regular than when the EINs were included. EINs therefore appear not to be necessary for the rhythmogenesis, but to play a role in stabilizing the oscillatory behaviour of the network.

5.3 Evolution of intersegmental coupling

5.3.1 Encoding, GA, and evaluation

I also evolved the interconnectivity between fixed segmental oscillators corresponding to the segmental network of Ekeberg's biological model. As mentioned in chapter 2, the existing interconnections between segments in the real lamprey are not perfectly known yet and there is some controversy in the mechanisms which lead to phase lags which are constant over the spinal cord and independent of the frequency of oscillations (see discussion). Evolving the couplings between the biological segmental network may therefore give some insight into which kind of interconnections can produce this type of phase lag.

I use the same methods (same GA and same fitness function) as for the second evolutionary stage of the previous chapter. The fixed segmental oscillator corresponds to Ekeberg's segmental network and a chromosome encodes the extensions to neighbouring segments of the segmental connections.

5.3.2 Results

I realised 5 runs with populations of 40 chromosomes. All five runs converged, in fewer than 75 generations, to complete controllers with similar performances to Ekeberg's biological model (see Table 5.2). The solutions cover larger ranges of lags and can reach slightly higher speeds. The best solution, for instance, can have its lag per

	Fitness value	Frequency range in [Hz]	Relative lag range in [%]	Speed range in [m/s]
Biol. model	0.08	[1.6, 5.6]	[0.0, 2.4]	[-0.03, 0.50]
run1	0.22	[1.7, 5.5]	[0.0, 3.5]	[0.06, 0.53]
run2	0.21	[1.7, 5.6]	[0.0, 3.5]	[0.07, 0.53]
run3	0.22	[1.7, 5.5]	[0.0, 3.4]	[0.07, 0.53]
run4	0.21	[1.7, 5.6]	[0.2, 3.5]	[0.06, 0.53]
run5	0.22	[1.7, 5.5]	[0.0, 2.6]	[0.09, 0.52]

Table 5.2: Summary of results for the evolution of multi-segmental controllers with Ekeberg's segmental network.

segment varied between 0.0 and 3.5% of the oscillation period and reach a speed of 0.53 m/s. This range of positive phase lags corresponds approximately to the phase lags up to +3.0% obtained, by varying local concentration of excitatory bathes, in the spinal cord of the real lamprey [Matsushima & Grillner 92]. Similarly to Ekeberg's model, the frequency of oscillation depends mainly on the global level of excitation and the lag between segments is mainly controlled by the extra excitation on the five first segments. Finally, Ekeberg's model and these five solutions all have a very similar relation between the speed of swimming and the levels of global excitation and extra excitation.

These solutions have all similar intersegmental couplings and are significantly different from that of Ekeberg's model in which only the CIN have asymmetric projections (favouring the caudal direction, *cf.* Table 3.1, pp 60). Table 5.3 gives the interconnections of one of them (see Appendix E for the configurations of all evolved controllers). Although the exact extents of the interconnections vary from one solution to another, they present strong similarities in the rostral/caudal asymmetries of interconnections. For instance, all solutions have asymmetries favouring the caudal direction for the ipsilateral LIN to CIN and contralateral CIN to CIN connections. They also have asymmetries favouring the rostral direction for the ipsilateral EIN to EIN, ipsilateral EIN to CIN and contralateral CIN to EIN connections. The fact observed with the artificial controllers that connections between neurons of the same type extend more caudally if the connection is contralateral and more rostrally if the connection is ipsilateral (self-connection) is thus also true here. Similarly to Ekeberg's model, it is possible, for some of the controllers, to produce backwards propagating waves when the most caudal segments receive more excitation, but the phase difference between

from: / *to:*	EINl	CINl	LINl	EINr	CINr	LINr	BS
EINl	0.4 [5, 3]	-	-	-	-2.0 [1, 0]	-	2.0
CINl	3.0 [11, 2]	-	-1.0 [3, 9]	-	-2.0 [0, 5]	-	7.0
LINl	13.0 [4, 1]	-	-	-	-1.0 [9, 4]	-	5.0
MNl	1.0 [4, 1]	-	-	-	-2.0 [11, 11]	-	5.0
EINr	-	-2.0 [1, 0]	-	0.4 [5, 3]	-	-	2.0
CINr	-	-2.0 [0, 5]	-	3.0 [11, 2]	-	-1.0 [3, 9]	7.0
LINr	-	-1.0 [9, 4]	-	13.0 [4, 1]	-	-	5.0
MNr	-	-2.0 [11, 11]	-	1.0 [4, 1]	-	-	5.0

Table 5.3: Evolved interconnections between biological segmental network (best of run2). The table gives the weights of the segmental network (identical to Ekeberg's) and, between brackets, the extensions of the evolved interconnections in the rostral and caudal direction respectively (to be compared with Ekeberg's values Table 3.1 pp60).

segments is then very small, showing that these controllers preferentially propagate traveling waves from head to tail.

One solution, the controller evolved in run 4, propagates a traveling wave from head to tail even without extra excitation on the most rostral segments. This controller therefore presents an aspect of the real lamprey which Ekeberg's controller does not reproduce which is the fact that isolated spinal cords of the lamprey spontaneously propagate a traveling wave of neural activity even in a uniform excitatory bath.

It is interesting to compare the evolved interconnections with the couplings observed so far in the real lamprey. As summarized in [Wadden *et al.* 97], it has been observed that LINs have long caudal projections (up to 50 segments), CINs project mainly caudally (at least 14 segments) with some rostral extensions, and EINs have relatively short projections (between 5 and 7 segments) in both directions. The details of how different types of neurons in different segments are targeted by these projections are, however, not known. The evolved controllers have interconnections which respect these observations to some extent. For the solution given in Table 5.3, for instance, only the long rostral projections of the EIN to CIN, the CIN to LIN and CIN to MN strongly disagree with the biological observations. The others correspond to the observed biological interconnections if I hypothesize that the projections observed in the real lamprey from one type of neuron not necessarily target all neuron types in other segments in an identical way. For instance, although CIN neurons have globally long caudal projections and short rostral projections, it may be that their projections to EINs are mainly rostral and that the existing caudal projections do not target EINs.

5.4 Evolution of sensory feedback from stretch sensitive cells

5.4.1 Encoding, GA, and evaluation

Similarly to the artificial controllers, it is possible to evolve sensory feedback from stretch sensitive cells to the segmental oscillators. I realised a set of 5 runs (populations of 40 chromosomes, 100 generations) which evolved sensory feedback to the 6 interneurons and the 2 motoneurons of Ekeberg's segmental network. The speed of the speed barrier was fixed to be 40% higher than the speed of swimming without sensory feedback. The same fitness function and GA as for the corresponding stage in the previous chapter are used.

5.4.2 Results

All runs converged to controllers capable of crossing the speed barrier. The solutions vary in the exact value of the weights but are almost identical in the respective signs of the connections (of the 8 possible connections, 5 connections have the same sign in all runs and the 3 others have the same sign in all runs except one — see Appendix F for a description of the evolved configurations). Similarly to the corresponding runs with an artificial controller, they all developed strong excitatory connections from the stretch sensitive cells to the motoneurons. It is interesting to note that the evolved sensory feedback pathways correspond very closely to those observed in the real lamprey. It has been shown that there exist both inhibitory and excitatory edge cells in the lamprey with the inhibitory edge cells projecting contralaterally and the excitatory cells projecting ipsilaterally [Viana Di Prisco *et al.* 90]. Table 5.4 compares the evolved connections of the best run with the connections established sofar with paired intracellular recordings and morphological observations. For all the established biological connections, the evolved controllers have developed sensory feedback connections with the same sign. Note that signs of the evolved connections from the edge cells to the EINs and LINs are somewhat counterintuitive because these connections counteract the effect of the other connections which tend to switch the neural activity to the side which is excessively extended.

	Evolved		Biological	
	ECl	ECr	ECl	ECr
EINl	−1.0	5.0		
CINl	2.4	−0.9	+	−
LINl	0.8	−1.0	+	−
MNl	3.9	3.4	+	

Table 5.4: Comparison between the established biological sensory feedback [Viana Di Prisco *et al.* 90] and that evolved by the best run. EC stands for Edge Cell, and *l* and *r* indicate the left and right side of the spinal cord. The biological feedback comes from both inhibitory and excitatory edge cells. The signs of the empirically established connections are given by + and −.

As the relative strengths of the biological sensory feedback connections are not known for the moment, it would be interesting to test whether the biological circuitry has similar relative strengths to the weights of the connections evolved here; and, similarly, to test whether there exist unobserved biological connections similar to those evolved (those projecting to the EINs, for instance).

Ekeberg studied two types of connections separately, the excitatory connections to ipsilateral CIN and LIN, and found that each connection was sufficient to cross a speed barrier which would not have been crossed without sensory feedback [Ekeberg *et al.* 95]. The connection to LIN provided the best performance with the crossing of a barrier whose speed was 40% larger than the lamprey's swimming speed. He did unfortunately not provide the synaptic weights of the connections for a comparison with those evolved here.

5.5 Discussion of the results

The fact that the GA was able to find sets of synaptic weights and interconnections for CPGs under the constraints of the biological observations showed that it has the potential to be a useful tool for developing neurobiological models.

The GA can be used as a synthetic tool, but also indirectly as a kind of analytical tool. The synthetic aspect is illustrated in the evolution of segmental oscillators, in which case the circuitry and its behaviour are well known and the GA is simply used to automatically generate the synaptic weights such that the simulated circuitry produces the desired behaviour. Although it is possible, in this case, to define satisfactory values by hand (as Ekeberg did), the GA proved useful for doing that work automatically and for optimizing the variable values for better fitting biological observations (e.g. producing a larger range of frequencies). When the circuitry is less well known (e.g. the intersegmental coupling), the GA can be used for searching different possibilities and for generating potential neural configurations which optimize some criteria, therefore enabling an indirect analysis of how a behaviour arises from a neural configuration. The hope is then that the evolved potential solutions give some insights for new experiments for determining the configuration and function of the actual biological circuitry.

From a general point of view, note that, although the GA can be very useful for demonstrating that a model can produce a specific behaviour (by finding efficient sets of unknown variables), it is less useful for invalidating a hypothetical model as an inability to find successful variable instantiations may be due to failings of the model or to problems with the GA set up, or both. The use of evolutionary algorithms for designing neural networks will be further discussed in chapter 9.

The development of controllers which preserve the biological segmental connectivity has led to several observations which may interest neurobiologists. Firstly, the generation of segmental oscillators with the biological connectivity shows that Ekeberg's set of synaptic weights can be modified in order to optimize the frequency range. Frequency ranges much closer to those observed in the real lamprey (between 0.25 and 10 Hz) can be obtained. The main observation is that the increase of the frequency range is obtained through an increase of the strength of some inhibitory connections (the contralateral CIN to LIN and ispilateral LIN to CIN connections). I also find, in agreement with [Hellgren *et al.* 92, Jung *et al.* 96], that the biological connectivity can produce oscillations without the excitatory neurons (provided that the other synaptic weights are readjusted). Moreover, I demonstrate here that removing the excitatory neurons does not prevent the biological connectivity from being able to cover a large frequency range of oscillations.

Couplings between the biological segmental networks are also evolved which can produce phase lags which are constant over the spinal cord and independent of the frequency of oscillations, as observed in the real lamprey. As mentioned in chapter 2, there are opposing views on the origin on these phase lags in the real lamprey, in particular whether they are due to differences in the intrinsic frequencies of the oscillators or to the nature of the coupling, and, in the latter case, whether the dominant coupling is ascending or descending (see for instance [Matsushima & Grillner 92, Wadden *et al.* 97, Kopell 95, Williams & Sigvardt 95]). Here, at a connectionist level, I have developed CPGs which rely both on asymmetries of the couplings and on differences of intrinsic frequencies for creating travelling waves. The coupling is optimized for waves to travel from head to tail while the extra excitation of the most rostral segments determines the exact phase lag. Extra excitation of the first segments amounts to an increase of the intrinsic frequency of those segments which therefore tend to lead the others. As predicted theoretically for abstract oscillators [Rand *et al.* 88], the higher the intrinsic frequency difference, the larger the phase difference between all segments. The optimization of the intersegmental coupling for head to tail propagation is illustrated, firstly, by the difficulty of inducing backwards propagation when the most caudal segments receive extra excitation, and secondly, by the fact that some controllers spontaneously propagate waves from head to tail even without extra excitation. The evolved CPGs have no dominant coupling (for instance, no systematic asymmetry) and the interconnections between segments combine both caudal and rostral asymmetries of couplings. Finally, the evolved couplings are, to some extent, similar to those observed sofar in the real lamprey, if we hypothesize that the observed projections from one neuron type in one segment to neurons in other segments represent the total of all projections from that neuron and do not necessarily mean that the projections to different types of neurons must be identical.

Interestingly, the evolution of sensory feedback to Ekeberg's controller led to connections which correspond very closely to those observed in the real lamprey, with all connections from excitatory and inhibitory stretch sensitive cells established in the real lamprey being also present with the same sign in the evolved controllers. Crossing a speed barrier may therefore be a good example of the situation for which sensory feedback has been developed through natural evolution for the real lamprey.

5.6 Summary

Evolutionary algorithms have the potential to be powerful tools for developing neurobiological models because of their optimisation capacity and their flexibility. This chapter presented a simple example of how a GA can be used to automatically generate some parts of a biological model similar to Ekeberg's.

The GA was used to determine synaptic weights for segmental networks with the biological connectivity and generated sets of weights which lead to a significantly larger range of frequencies than Ekeberg's segmental network, much closer to that observed in the real lamprey.

The GA was also used to investigate the space of possible intersegmental couplings between the biological segmental networks, as the details of the coupling have not been decoded yet. Couplings have been evolved which can produce the anguiliform swimming of the lamprey with a similar performance in terms of frequency and speed ranges as Ekeberg's model, while being able to reach higher phase lags. The different evolved couplings present several common features which would be interesting to study in further detail and compare with future anatomical findings on the intersegmental coupling of the real lamprey.

Finally, sensory feedback connections from stretch sensitive cells have been evolved for allowing the lamprey to cross a local speed barrier. Interestingly, the evolved connections present several similarities with those observed sofar in the lamprey, suggesting that swimming through water with variations of flow speed may be a good example of the situation for which sensory feedback has been developed through natural evolution for the real lamprey.

Chapter 6

Evolving swimming CPGs using a developmental encoding scheme

In addition to evolution (i.e. the process of how replicating genetic structures mutate and succeed in reproducing themselves), animals (and plants) are the result of a further adaptive mechanism: *development*. Development is the process of how genetic structures are decoded for the "building" of the body. Biological genetic encoding is remarkably compact; rather than encoding all the details of a complete body plan, genes encode *rules* of how the body should be constructed. The genetic instructions then rely on the physics of the environment for the development of the complete body.

As mentioned in chapter 2, biological development has inspired researchers to develop indirect encodings for the artifical evolution of neural networks, which are more compact and modular than direct encodings of network parameters. This chapter presents the results of a collaboration with Jérôme Kodjabachian from the Animat Lab (Ecole Normale Supérieure, Paris[1]), in which we investigated the evolution of swimming controllers using the developmental encoding scheme he designed, SGOCE (Simple Geometry Oriented Cellular Encoding) [Kodjabachian & Meyer 98a, Kodjabachian & Meyer 98b]. In the scheme, *developmental programs* are evolved which determine how neurons located on a geometric 2D substrate produce new cells through cellular division and how they form efferent or afferent interconnections.

This chapter presents how, using this encoding scheme, swimming central pattern

[1] Now at OASIS-LIP6, Université Pierre et Marie Curie, Paris.

generators can be developed in only one evolutionary stage instead of the two used previously (the work presented here does not include sensory feedback). Using simpler neurons and fewer neural segments, the controllers are generated based on a fitness function which only considers mechanical aspects such as the control of speed and of direction of swimming.

The next sections present the developmental encoding, the geometric substrate for the development and the evolutionary algorithm used. The results of 10 evolutions with a fitness function rewarding the capacity to control the speed and the direction of swimming of the simulated lamprey are then presented. The results presented here can also be found in [Ijspeert & Kodjabachian 98a].

Note: The design of the developmental encoding and the evolutionary algorithm were realised by Jérôme Kodjabachian, while I developed the neural simulation and the mechanical simulation (which is a reproduction of Ekeberg's mechanical simulation). In particular, both the concept and the software of the SGOCE encoding scheme used in this chapter were developed by Jérôme Kodjabachian.

6.1 Methods

We use the SGOCE (Simple Geometry Oriented Cellular Encoding) evolutionary algorithm for designing the swimming controllers [Kodjabachian & Meyer 98a, Kodjabachian & Meyer 98b]. The algorithm uses an encoding scheme that relates the animat's genotype — the developmental program — and phenotype — a dynamical neural network; and an evolutionary algorithm that generates the developmental programs within some syntactic constraints.

The encoding scheme is a geometry-oriented variation of Gruau's cellular encoding [Gruau 95]. In the scheme, developmental programs are evolved which determine how neurons divide and become connected to each other on a two-dimensional metric substrate. The substrate contains *precursor cells* which will develop into sets of neurons, and nodes which corresponds to inputs (the control signals) and outputs (the signals sent to the muscles).

The task here is to generate neural controllers for swimming given a fixed body structure of the lamprey and given a high level characterisation of the desired behaviour defined by the fitness function. Through two input signals applied to the network we want to be able to initiate swimming and to modulate the speed and the direction of motion by simply varying the amplitude of these signals. Compared to the two previous chapters, the method used here presents the following similarities and differences. The similarities are that:

- the same mechanical simulation of the lamprey is used,
- the controllers receive tonic signals as input for modulating the speed and direction of swimming,
- a similar fitness function is used consisting of a product of factors between 0.05 and 1.00.
- an evaluation consists of several simulations with different commands settings.

The differences are that:

- controllers are composed of simpler neurons and fewer neural segments,
- controllers are encoded with the developmental scheme, and evolve with an evolutionary algorithm whose genetic operators are adapted to the tree-structure of the developmental programs,
- neuron parameters such as neuron biases and time constants are evolved as well as the connectivity (connections and synaptic weights) of the controllers,
- stretch sensitive cells are not included,
- controllers are evolved in one stage only, with a fitness function only based on mechanical aspects, namely the control of speed and direction, instead of both neural and mechanical aspects.

One of the main differences from the previous evolutions is that the general organisation of the evolved controllers is not limited to be composed of identical segmental oscillators which are interconnected to make the complete central pattern generator. Complete controllers are here evolved in only one stage and their structure is less strictly specified (although user-specified geometrical considerations limit the type of controllers which can be generated, see next sections). This aspect, and the fact that neuron parameters are evolved as well as the connectivity, means that a larger variety of neural configurations can be developed and that the search space is less restriced than in chapter 4.

6.1.1 Neuron model

We use here a simpler neuron model than that developed by Ekeberg. The model is also a leaky-integrator, but with only one state variable, the mean membrane potential. The motivations behind this choice of neuron model were primarily to have an easier adaptation to Jérôme Kodjabachian's earlier work, but also to test whether simpler neuron models than Ekeberg's are sufficient for producing swimming patterns.

The membrane potential m_i of a neuron i is calculated by the equation:

$$\tau \cdot dm_i/dt = -m_i + \sum w_{i,j} x_j$$

where $x_j = (1+e^{(m_j+b_j)})^{-1}$ represents the neuron's short-term average firing frequency, b_j is the neuron's bias, τ is a time constant associated with the passive properties of the neuron's membrane, and $w_{i,j}$ is the synaptic weight of a connection from neuron N_j to neuron N_i. Each neuron exhibits an internal dynamics and even small networks of these neurons have proven able to produce rich dynamics [Beer 95].

Conversely to the previous chapters, in which the neuron parameters were fixed, in the present chapter the neurons' biasses and time constants will be evolved together with the connectivity of the network.

6.1.2 Developmental encoding scheme

Neural controllers are developed from several precursor cells located on a 2D substrate (see next section). The development is determined by a developmental program which determines how neurons divide and get connected to each other. Each cell within the

DIVIDE α r	create a new cell
GROW α r w	create a connection to another cell
DRAW α r w	create a connection from another cell
SETBIAS b	modify the bias parameter
SETTAU τ	modify the time constant parameter
SETINPUTW w	modify the input weight
DIE	trigger cellular death

Table 6.1: Set of instructions used in the developmental programs

control architecture is assumed to hold a copy of the developmental program. The program has a tree-like structure similar to that of encodings used in genetic programming. The nodes in the tree belong to a small set of developmental instructions (see Table 6.1), or are structural instructions added to help define the grammar (see below). A local coordinate frame is attached to each precursor cell. A cell division instruction (DIVIDE) makes it possible for a given cell to generate a copy of itself. A direction parameter (α) and a distance parameter (r) associated with that instruction specify the position of the daughter cell to be created in the coordinates of the local frame attached to the mother cell. Then, the local frame associated to the daughter cell is centered on that cell's position and is oriented as the mother cell's frame. Each time a cell divides, its daughter cell receives a copy of its afferent and efferent connections. Two instructions (GROW and DRAW) respectively create one new efferent and one new afferent connection. The cell or the input-output node to be connected to the current one is the closest to a target position that is specified by the instruction parameters, provided that the target position lies on the substrate. No connection is created if the target is outside the substrate's limits. The synaptic weight of the connection is given by the parameter w. Two additional instructions (SETTAU and SETBIAS) specify the values of a neuron's time constant τ and bias b. Finally the instruction DIE causes a cell to die. In addition to the connections created by the developmental program, we cause each created neuron to automatically draw an afferent connection from the input node of their side. The synaptic weight of that connection is determined by the SETINPUTW instructions. An example of how a developmental program is decoded into the corresponding set of neurons is given in Figure 6.1.

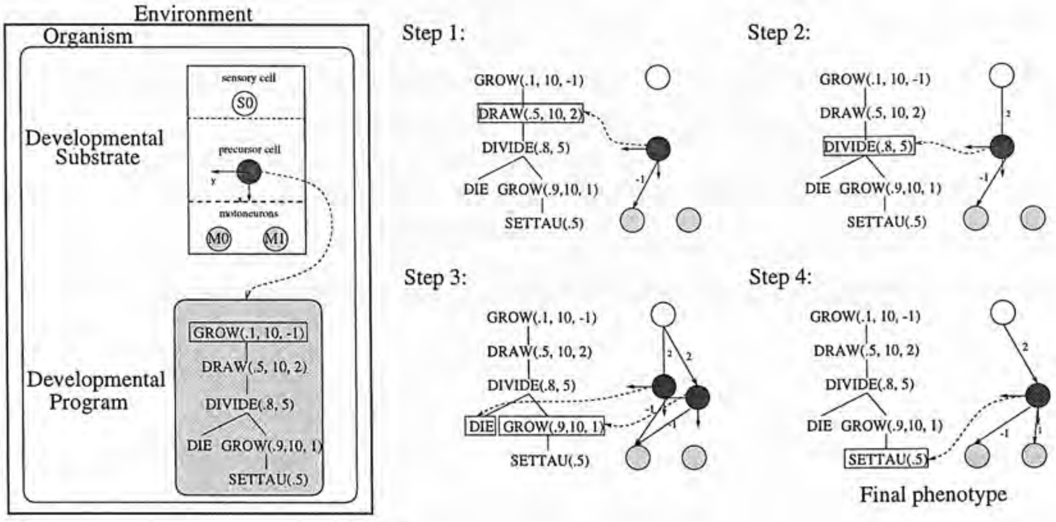


Figure 6.1: Example of the decoding of a developmental program into a set of neurons. Firstly, a connection is grown by the precursor cell to an output cell (Step 1). Secondly, another connection is drawn by this precursor cell from the input cell (Step 2). In both cases, the cell with which the connection is made is the closest one to a target point whose polar coordinates are given in the local frame of the cell by the instruction's parameters. At the next develomental step (Step 3), the precursor cell creates a daughter cell that reads the right sub-node of the DIVIDE instruction while the mother cell reads the left sub-node and is killed, because the corresponding instruction is a DIE instruction. Note that the DIVIDE instruction has duplicated the connections of the initial cell. Finally, the daughter cell grows a new connection to another output cell (Step 4) and modifies the value of its time constant parameter. After the development, this cell is considered as a neuron with its specific connections and parameters.

6.1.3 Substrate

We use a substrate which corresponds to the body of the lamprey (Figure 6.2). Nine output nodes are fixed on each side of the body. These nodes are connected to the muscles such that the signals sent to them determine the contraction of the muscles. We also fix the position of 18 precursor cells, located symmetrically on both sides of the body. The substrate can therefore be seen as made of 9 segments, with 2 output nodes and 2 precursor cells each, corresponding to the 9 joints of the mechanical simulation. Each of the 18 precursor cells holds a copy of the same developmental program. However the local frame attached to each cell need not have the same orientation, and in particular we make the precursor cells on each side (left and right) of the body have opposite orientations, which forces the developed controllers to be symmetrical. The two input nodes are also included in the substrate, but as each neuron automatically has an afferent connection from them, their exact position is not important. Note that

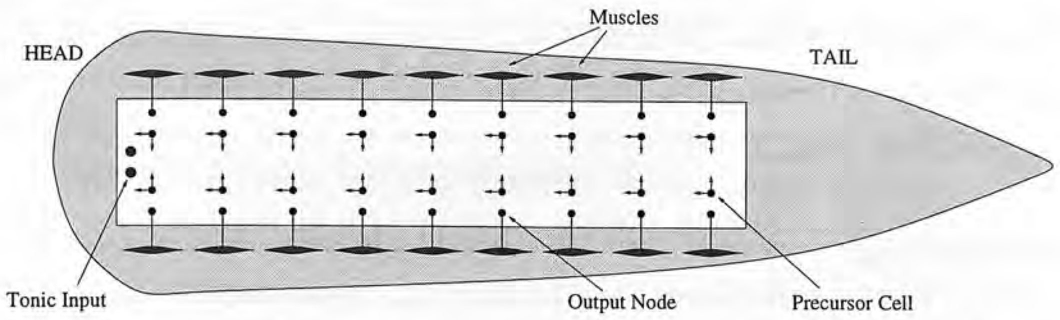


Figure 6.2: Substrate of the developmental process. The rectangular substrate extends over the length of the mechanical body. It contains 18 output nodes located symmetrically on both sides of the substrate which determine the contraction state of the muscles (Figure 3.6, chapter 3). It contains also two input nodes which provide the left and right tonic input of the CPG. 18 precursor cells are spread over the substrate. Left and right precursor cells have local frames which are oriented with a left-right symmetry. As all precursor cells develop using the same developmental code, this means that a left-right symmetry of the developed network is automatically created.

connections created by the GROW and DRAW instructions are not limited to neurons issued from the same precursor cell but these instructions can potentially connect any cell on the substrate within a fixed range that corresponds to the maximal value of the parameter r of the GROW and DRAW instructions. This means that networks can in principle be evolved with paths of connections between all neurons of the substrate. In the present paper, the range of connection is limited such that only connections between neurons of neighbouring segments can be developed.

6.1.4 Grammar

In order to reduce the size of the genetic search-space and the complexity of the generated networks, a context-free tree-grammar is used to force each evolvable program to have the structure of a well-formed tree (Table 6.2). The set of terminal symbols consists of the developmental instructions listed in Table 6.1 and of additional *structural instructions* that have no side-effect on the developmental substrate. The grammar of Table 6.2 permits the scheduling of the development to be organized into several marked stages. Firstly, cell divisions will occur and the connections from the command nodes to the precursor cells will be copied. Secondly, the different cells created will either die or modify their time constant, bias and input weight parameters. Finally, each surviving cell will create connections with its surrounding cells. As no more than two successive cell divisions can occur, this grammar limits the total number of neurons in a controller to 72 neurons.

Terminal symbols
DIVIDE, GROW, DRAW, SETBIAS, SETTAU, SETINPUTW, DIE, NOLINK, DEFBIAS, DEFTAU, DEFINPUTW, SIMULT4.
Variables
Start1, Level2, Neuron, Bias, Tau, InputW, Connex, Link.
Production rules
Start1⟶DIVIDE(Level2, Level2)
Level2⟶DIVIDE(Neuron, Neuron)
Neuron⟶SIMULT4(Bias, Tau, InputW, Connex) | DIE
Bias⟶SETBIAS | DEFBIAS
Tau⟶SETTAU | DEFTAU
InputW⟶SETINPUTW | DEFINPUTW
Connex⟶SIMULT4(Link, Link, Link, Link)
Link⟶GROW | DRAW | NOLINK
Starting symbol
Start1.

Table 6.2: Grammar used to restrict the trees which can be evolved. The production rules organize the development into several stages which schedule the development and which restrict the maximum number of neurons.

6.1.5 Evolutionary algorithm

The SGOCE evolutionary algorithm is a steady state genetic algorithm that involves a population of N randomly generated well-formed developmental programs distributed over a circle and whose functioning is sketched in Figure 6.3.

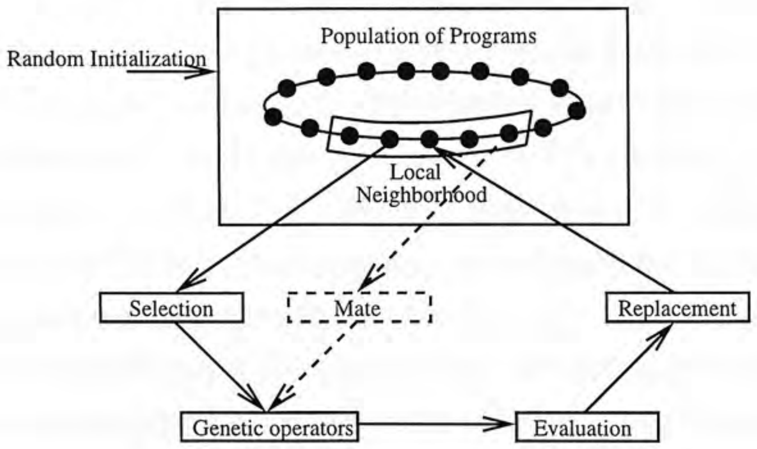


Figure 6.3: The evolutionary algorithm.

The following procedure is repeated until a given number of individuals have been generated and tested:

1. A position P is chosen on the circle.

2. A two-tournament selection scheme is applied, in which the best of two programs randomly selected from the neighbourhood of P is kept (more details can be found in [Kodjabachian & Meyer 98a]).

3. The selected program is allowed to reproduce and three genetic operators possibly modify it. The first operator, the recombination operator, is applied with probability p_c. It exchanges two *compatible* sub-trees, i.e. sub-trees which can be derived from the same grammatical rule, between the program to be modified and another program selected from the neighbourhood of P. Two types of mutation are used. The first mutation operator is applied with probability p_m. It changes a randomly selected sub-tree into another compatible, randomly generated one. The second mutation operator is applied with probability 1. It modifies the values of a random number of parameters, implementing a *constant perturbation strategy* [Spencer 94]. The number of parameters to be modified N_{mut} is drawn from a binomial distribution $B(n,p)$, and N_{mut} parameters among all the parameters are then modified by a small random quantity.

4. The fitness of the new program is assessed by collecting statistics while the swimming of the lamprey mechanical model, controlled by the corresponding artificial neural network, is simulated over a given period of time (see below).

5. A two-tournament anti-selection scheme, in which the worse of two randomly chosen programs is selected, is used to decide which individual (in the neighborhood of P) will be replaced by the modified program.

In all the experiments reported here $p_c = 0.6$, $p_m = 0.2$, $n = 6$ amd $p = 0.5$.

6.1.6 Fitness function

A developmental program is given a fitness value which depends on the capacity of its corresponding network to control swimming efficiently. Our aim is to develop controllers which can produce patterns of oscillations necessary for swimming when receiving tonic (i.e. non-oscillating) input, and which can modulate the speed and the direction of swimming when the amplitude of the left and right control signals are varied.

We therefore define a fitness function, based on the analysis of the mechanical simulation, rewarding controllers which:

1. produce contorsions, where a contorsion is defined as the passage of the centre of the body from one side to the other of the line between head and tail,
2. reach high swimming speeds,
3. can modulate the speed of swimming when the tonic input is varied, with the speed increasing monotonically with the level of input,
4. can change the direction of swimming when an asymmetrical tonic input (between the two sides of the body) is applied.

The evaluation of a developmental program consists of a series of simulations of the corresponding developed controller with different commands settings. Fixed-time simulations (6000 ms) are carried out with different levels of tonic input in order to determine the range of speeds the controller can produce. Asymmetrical initial states for the neurons are created by applying an asymmetrical input during the first 50 ms of the simulation. Starting from a fixed level of input (1.0), the input is varied by increasing and decreasing steps of 0.1 and the range of speed in which the speed increases monotonically with the tonic input is measured.[2] If the range of speeds includes a chosen[3] speed (0.15 m/s), a simulation is performed (at the tonic level corresponding to that speed) with a short period of asymmetric tonic input in order to evaluate the capacity of the controller to induce turning.[4] Turning is evaluated by measuring the deviation angle between the directions of swimming before and after the asymmetric input.

The mathematical definition of the fitness function is the following:

[2] The speed of swimming is measured as the average speed during the second half of the simulation.

[3] This value corresponds to approximately half the maximum speed reached by the best solutions.

[4] The sequence of control signals for the turning evaluation is: symmetric left and right signals for 3000 ms, asymmetric ($\pm 10\%$ between left and right) for 1000 ms, and symmetric for the last 2000 ms.

$$fitness = fit\_contorsion \cdot fit\_max\_speed \cdot fit\_speed\_range \cdot fit\_turning \in [(0.05)^4, 1.0]$$

where *fit_contorsion, fit_max_speed, fit_speed_range* and *fit_turning* are functions which are limited between 0.05 and 1.0 and which vary linearly between these values when their corresponding variables vary between two boundaries, a *bad* and a *good* boundary (same transformation function as that used in chapter 4). The variables for each functions and their corresponding boundaries are given in Table 6.3. If the speed range does not include the chosen speed of 0.15 m/s, the turning ability is not measured and *fit_turning*=0.05. The fact that the fitness function is a product rather than a sum ensures that controllers which perform equally in all four aspects will be more favoured compared to controllers performing well in some aspects but not in others.

Function	Variable	[*bad*,*good*] Boundaries
fit_contorsion	Number of contorsions	[0,10] contorsions
fit_max_speed	Maximum speed	[0.0,1.0] m/s
fit_speed_range	Relative speed range	[0.0,1.0]
fit_turning	Deviation angle	[0.0,0.75π]

Table 6.3: Variables and boundaries for the fitness function. A contorsion is measured as the passage of the center of the body from one side to the other of the line between head and tail. The speed range is measured relative to the maximum speed.

6.2 Results

6.2.1 Evolutions

We carried out ten evolutions with populations of 50 controllers. Each run started with a different random population. A run was allowed to evolve for 100 generations (5,000 tournaments), taking approximately 450 CPU hours on an Ultra 1 Model 140s SUN station. The evolutions were mainly stopped for time reasons, but 100 generations also approximately correspond to the time when the different chromosomes converge to an identical solution and cease to improve their fitness value significantly (e.g. see Figure 6.4). Table 6.4 gives the performances of the best evolved controllers of each run (controllers 1 to 10 respectively) and Figures 6.5 and 6.6 show the swimming behaviour they induce. The evolved controllers can be classified into three categories: those which do not produce oscillations (controllers 1,2,3, and 4), those which produce chaotic behaviour (controllers 5 and 6) and those which produce stable oscillations (controllers 7,8,9, and 10). We describe next the main characteristics of the evolved controllers, for a more detailed analysis see [Ijspeert & Kodjabachian 98b].

6.2.2 Non-oscillating networks

Four evolutions converged to non-oscillating solutions (controllers 1,2,3, and 4). These controllers do not have the circuitry to sustain oscillations, but still manage to propel the body forward by creating a few contorsions which give the body an initial thrust. These contorsions are due to the delayed transitions of the neurons from their initial state to their equilibrium state (see Appendix G). The delays in the transitions lead to delays in the contraction of the muscles, which happen to propel the body forward. Once the equilibrium states are reached, the contorsions of the body cease and the lamprey eventually comes to stand still. As the duration of the simulations was limited, these controllers were rewarded for their remaining speed at the end of the simulation.

6.2.3 Chaotic networks

Two evolutions converged to solutions producing chaotic oscillations with variable cycle duration and variable signal shapes (controllers 5 and 6). As the phase lag between segments is on average positive (i.e. the neural activity travels from head to tail), these solutions still manage to propel the lamprey forward but with a speed and a direction of swimming which are not constant. Because of these variations of speed, these solutions do not present a monotonic relation between the tonic excitation and the speed. However they received a high fitness value because the speeds measured in the evaluation function happened to increase monotonically for the excitation steps tested. The non-monotonicity is revealed when the control of speed is analysed with smaller excitation steps than those used for the evaluation.

Note that it is not surprising that chaotic oscillators have been evolved, as systems made of as few as three interconnected neurons can already present chaotic dynamics with this mathematical model of a neuron [Beer 95].

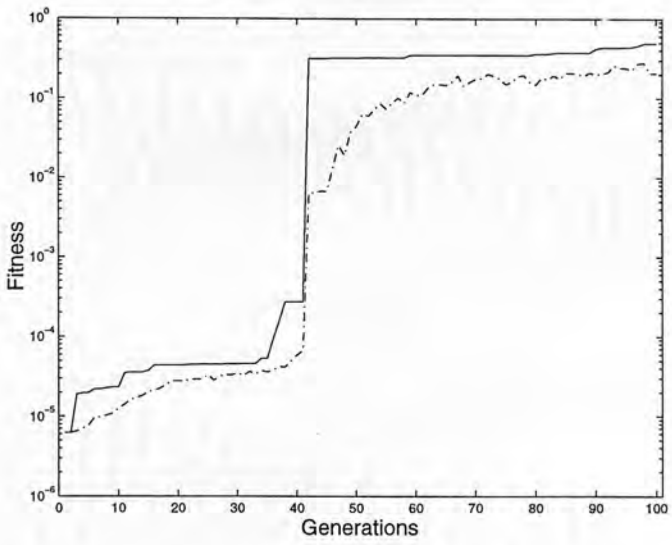
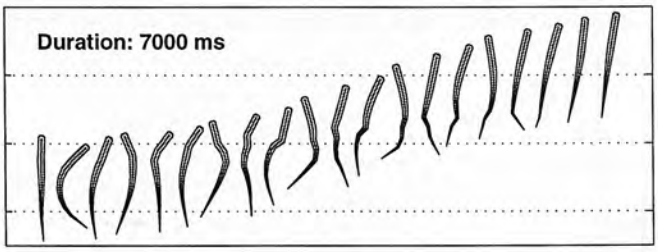


Figure 6.4: Evolution of the best run (run10). The best and average fitness of the population are represented by a continuous and a dotted line, respectively.

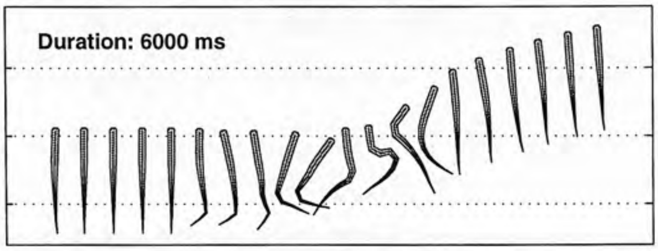
Run	Fitness	Speed [m/s]	Frequency [Hz]	Relative Lag [%]
1	0.00	[0.00, 0.08]	–	–
2	0.00	[0.00, 0.11]	–	–
3	0.01	[0.00, 0.11]	–	–
4	0.14	[0.00, 0.15]	–	–
5	0.24	[0.00, 0.16]	[1.16, 1.60]	–
6	0.27	[0.06, 0.27]	[0.46, 2.50]	–
7	0.01	[0.00, 0.12]	[0.69, 1.75]	[0.0, 0.0]
8	0.05	[0.02, 0.16]	[0.94, 0.95]	[0.0, 8.2]
9	0.24	[0.00, 0.27]	[0.57, 2.31]	[1.5, 11.2]
10	0.49	[0.09, 0.47]	[1.24, 5.38]	[16.0, 20.7]

Table 6.4: Results of the evolutions of central pattern generators. Controllers 1,2,3, and 4 do not produce oscillations. For these controllers, the given speeds indicate the range of speeds which can be reached through the initial thrust after 6000ms. Controllers 5 and 6 produce unstable oscillations, i.e. oscillations without a constant period. The given frequencies correspond therefore to the average (minimal and maximal) frequencies and the given speeds correspond to minimal and maximal speeds, although the speeds does not depend monotonically on the excitation. Also, because the shape of the signals is too variable, the relative lag is not given. The controllers 7,8,9, and 10 produce stable oscillations. For these controllers, the speed, frequency and lag values correspond to the minimal and maximal values obtained within the range of excitations for which the speed depends monotonically on the excitation. The relative phase lag is the average of the phase lags measured over the spinal cord (see text for the variations of the lag along the spinal cord). Note that, because there are only 9 neural segments, a relative lag of 11.1% per segment corresponds to a wavelength equal to the length of the lamprey.

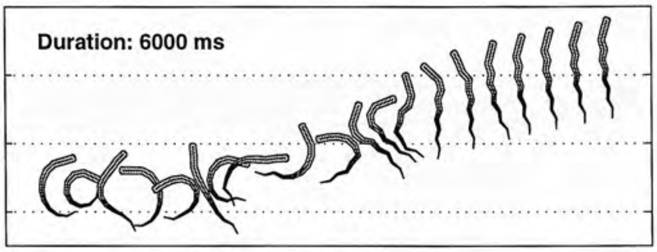
Controller 1:



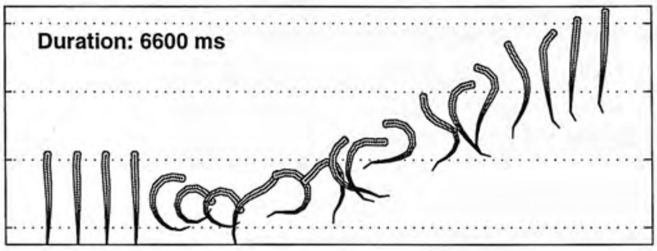
Controller 2:



Controller 3:



Controller 4:



Controller 5:

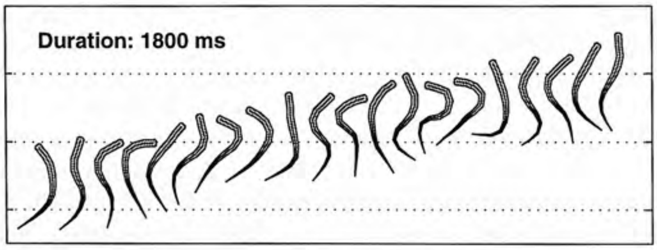
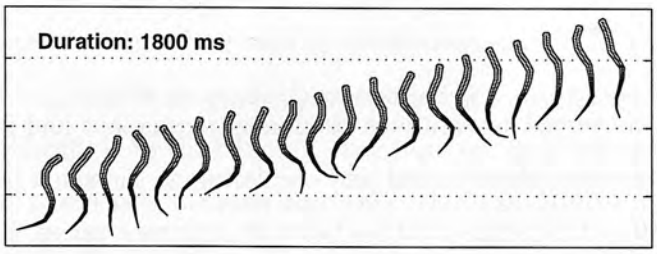
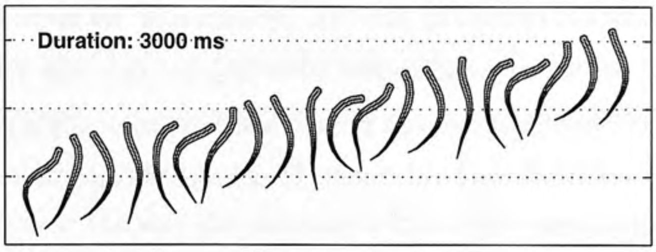


Figure 6.5: Swimming produced by controllers 1 to 5. The horizontal lines are separated by 200mm.

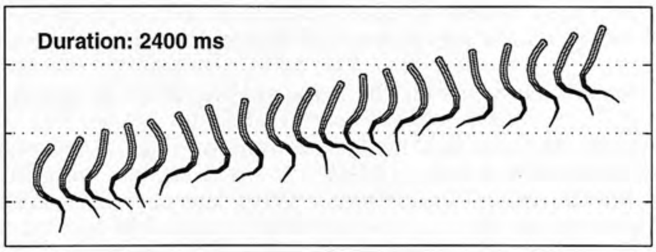
Controller 6:



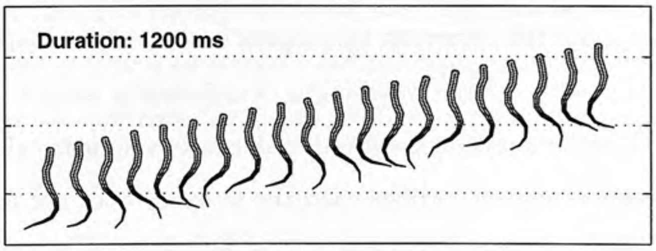
Controller 7:



Controller 8:



Controller 9:



Controller 10:

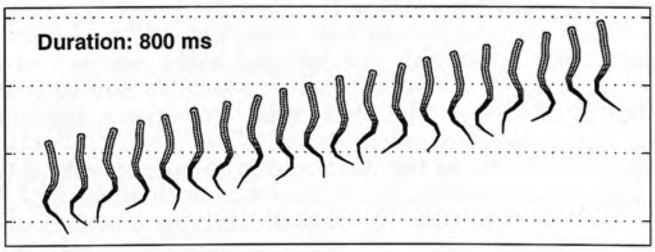


Figure 6.6: Swimming produced by controllers 6 to 10. The horizontal lines are separated by 200mm.

6.2.4 Oscillating networks

Four evolutions converged to solutions producing permanent and stable oscillations, with all segments being phase-locked and oscillating at the same frequency (controllers 7,8,9, and 10). The phase relation between segments varies, however, from one controller to the other, as do their configurations, which vary in the number of oscillators for producing the oscillatory activity. By oscillator, we mean the circuits which could potentially oscillate by themselves when isolated from the rest of the network (provided that the simulated neurons receive adequate tonic input). It can be shown that the smallest oscillator with this model of neurons is composed of 2 neurons with self connections [Beer 95].

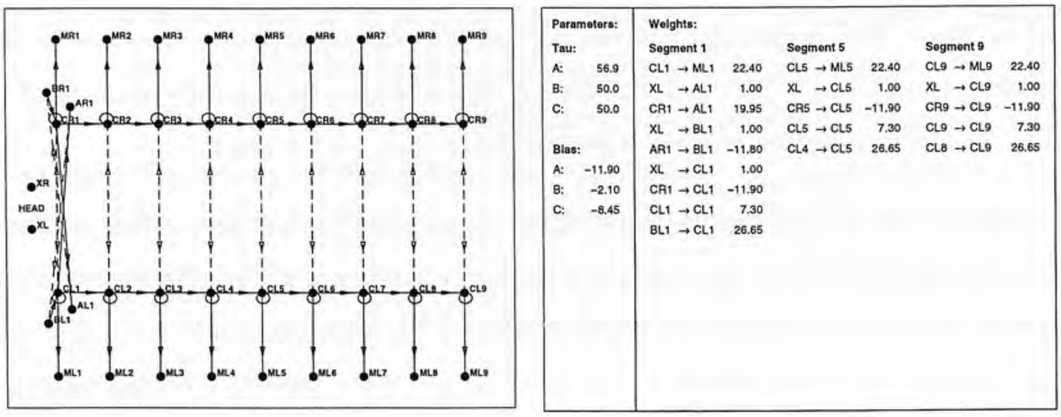
Controller 7 has the peculiarity of being made of two separate chains of oscillators on both sides of the body which are connected through excitatory contralateral connections (see Appendix G). Neurons on the same side oscillate in synchrony, and, because neurons on contralateral sides belong to different oscillators, the contralateral phase relation depends on the initial conditions. With the asymmetrical initial conditions of the evaluation function, the body produces a "C" bending, but with a contralateral phase relation which is not perfectly out of phase. Because of the different inertial forces between head and tail segments, this results in the body making an asymmetrical traveling undulation with a large wavelength, which slowly propels the body forward (see Figure 6.6). If the simulation is started with nearly symmetrical initial conditions, left and right muscles contract in phase and the lamprey does not progress.

Controller 8 is composed of one oscillator per segment interconnected over the spinal cord (see Appendix G). Each oscillator is distributed over both sides through inhibitory contralateral connections, and the left and right neurons of each segment are therefore perfectly out of phase. The phase relation between segments is slightly negative between segments closest to the head and gradually increases to become positive towards the tail. This means that the muscle activity travels forwards to the head and backwards to the tail from a segment in the middle of the trunk. As the negative phase difference between the first segments is small, the corresponding swimming is not unlike *carangiform* swimming in which most of the body is rigid except the tail which moves back and forth (Figure 6.6).

The two controllers which produce regular oscillations and reach the highest speeds (controllers 9 and 10), exhibit an *anguiliform* swimming which is very similar to that of the real lamprey. Similarly to the lamprey, they have left and right neurons oscillating out of phase, and phase lags between segments which are almost constant over the spinal cord (although, for controller 10, the signal shapes slightly vary from head to tail), leading to caudally directed traveling waves (Figures 6.8 and 6.10).

Controller 9 has the peculiarity of including only one oscillator. It is also the controller with the smallest number of neurons, with one oscillatory circuit in the first segment (closest to the head) and two neurons per segment in the other segments (Figure 6.7). That rostral oscillator entrains the neurons in the other segments through a chain of caudally directed excitatory connections. Because of the synaptic time constants, these neurons oscillate with a lag compared to their rostral neighbours, which leads to the observed anguiliform swimming. The fact that only the first segment contains an oscillator is due to the border effect of the substrate. The same DRAW instruction is responsible both for the creation of a connection completing an oscillator circuit in the first segment, and for the creation of intersegmental links connecting each of the other segments to the preceding one (Figure 6.11). Before evaluating this controller, the neurons labelled A and B in the figure are pruned in all but the first segment because they have no efferent connections towards the muscles or the C neurons, which implies that these cells cannot influence the behavior of the simulated lamprey and thus need not be simulated.

Controller 10 has a similar organisation to controller 8, with one oscillator per segment being distributed over both sides of the spinal cord through inhibitory contralateral connections (Figure 6.9). The segments are interconnected over the spinal cord by closest neightbour coupling (because of the limitation of the length of a connection, parameter r of the GROW and DRAW instructions, no longer coupling was developed). Note that of the two controllers producing anguiliform swimming, controller 10 is more robust against "lesions" than controller 9 in the sense that destroying neurons or connections in the head oscillator or in the coupling between segments of the controller 9 has a significantly larger impact on the swimming pattern than for controller 10.



Parameters:		Weights:							
Tau:		Segment 1		Segment 5		Segment 9	
A:	56.9	CL1 → ML1	22.40	CL5 → ML5	22.40	CL9 → ML9	22.40
B:	50.0	XL → AL1	1.00	XL → CL5	1.00	XL → CL9	1.00
C:	50.0	CR1 → AL1	19.95	CR5 → CL5	-11.90	CR9 → CL9	-11.90
		XL → BL1	1.00	CL5 → CL5	7.30	CL9 → CL9	7.30
Bias:		AR1 → BL1	-11.80	CL4 → CL5	26.65	CL8 → CL9	26.65
A:	-11.90	XL → CL1	1.00				
B:	-2.10	CR1 → CL1	-11.90				
C:	-8.45	CL1 → CL1	7.30				
		BL1 → CL1	26.65				

Figure 6.7: Architecture of controller 9, one of the two controllers producing anguiliform swimming. *Left:* Neurons on the substrate. For reasons of clarity, the lateral axis of the substrate has been mutliplied by ten. Ml_i and Mr_i represent the left and right muscle nodes of segment i, Xl and Xr are the input nodes. A,B and C are three types of neurons. Excitatory and inhibitory connections are represented through continuous and dashed lines respectively. Each neuron also receives a connection from the input nodes, which is not shown here. *Right:* Parameters and connection weights. Only the weights of the connections to the left neurons are given, the others are symmetric.

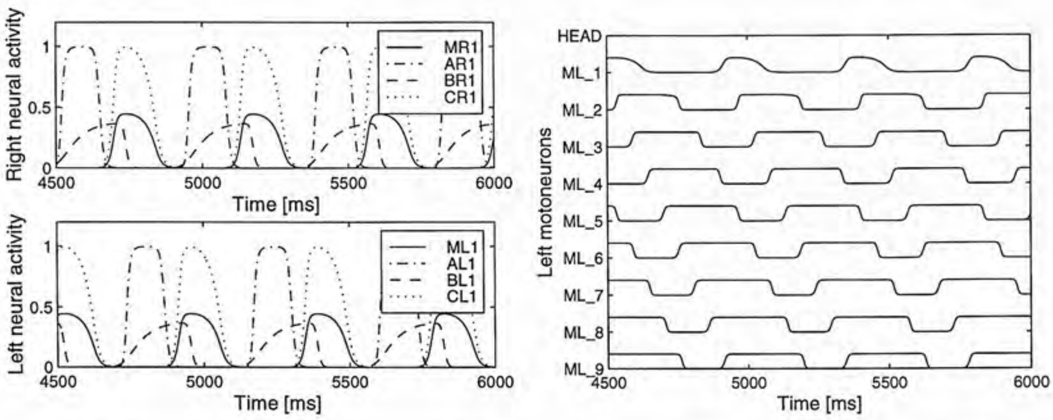
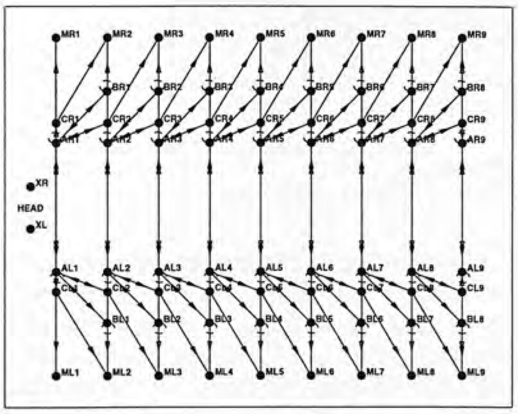
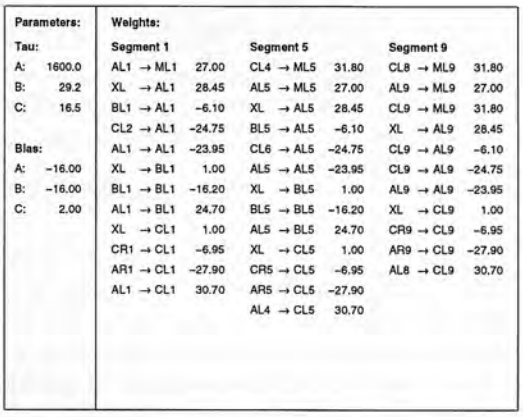


Figure 6.8: Neural activity in controller 9. *Left:* Neural activity of left and right neurons and the signals sent to the muscles in segment 1 (the oscillator segment, see text). *Right:* Signals sent to the left muscles along the nine segments.





Parameters:		Weights:					
Tau:		Segment 1		Segment 5		Segment 9	
A:	1600.0	AL1 → ML1	27.00	CL4 → ML5	31.80	CL8 → ML9	31.80
B:	29.2	XL → AL1	28.45	AL5 → ML5	27.00	AL9 → ML9	27.00
C:	16.5	BL1 → AL1	-6.10	XL → AL5	28.45	CL9 → ML9	31.80
		CL2 → AL1	-24.75	BL5 → AL5	-6.10	XL → AL9	28.45
Bias:		AL1 → AL1	-23.95	CL6 → AL5	-24.75	CL9 → AL9	-6.10
A:	-16.00	XL → BL1	1.00	AL5 → AL5	-23.95	CL9 → AL9	-24.75
B:	-16.00	BL1 → BL1	-16.20	XL → BL5	1.00	AL9 → AL9	-23.95
C:	2.00	AL1 → BL1	24.70	BL5 → BL5	-16.20	XL → CL9	1.00
		XL → CL1	1.00	AL5 → BL5	24.70	CR9 → CL9	-6.95
		CR1 → CL1	-6.95	XL → CL5	1.00	AR9 → CL9	-27.90
		AR1 → CL1	-27.90	CR5 → CL5	-6.95	AL8 → CL9	30.70
		AL1 → CL1	30.70	AR5 → CL5	-27.90		
				AL4 → CL5	30.70		

Figure 6.9: Architecture of controller 10, one of the two controllers producing anguiliform swimming. *Left:* Neurons on the substrate. For reasons of clarity, the lateral axis of the substrate has been mutliplied by ten. Ml_i and Mr_i represent the left and right muscle nodes of segment i, Xl and Xr are the input nodes. A,B and C are three types of neurons. Excitatory and inhibitory connections are represented through continuous and dashed lines respectively. Each neuron also receives a connection from the input nodes, which is not shown here. *Right:* Parameters and connection weights. Only the weights of the connections to the left neurons are given, the others are symmetric.

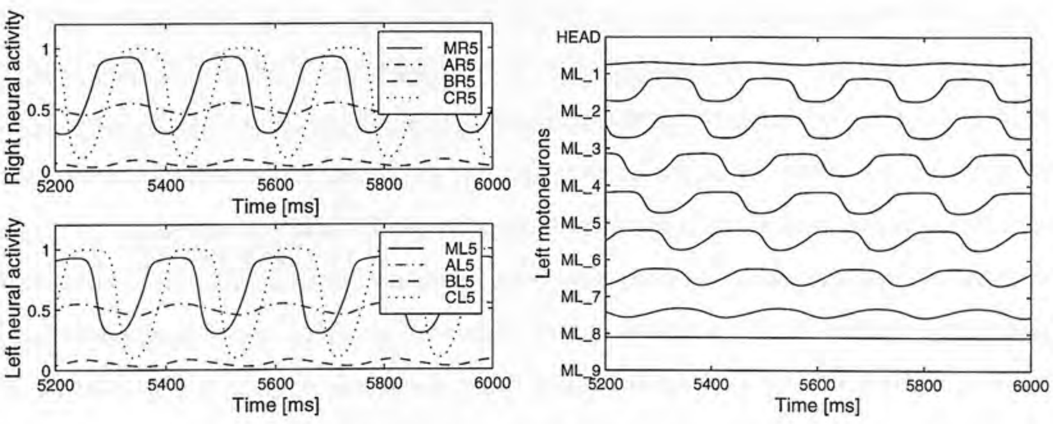


Figure 6.10: Neural activity in controller 10. *Left:* Neural activity of left and right neurons and the signals sent to the muscles in segment 5. *Right:* Signals sent to the left muscles along the nine segments.

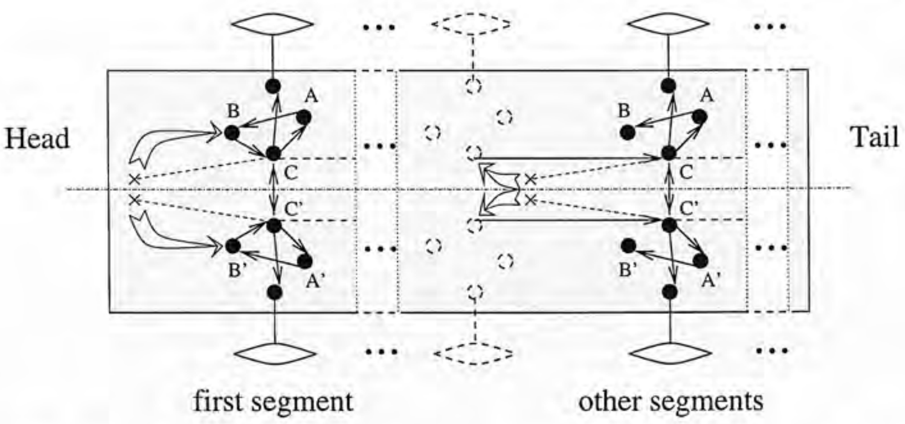


Figure 6.11: Side-effect during the development of Controller 9: During the development, a DRAW instruction was read by all cells C and looked for the closest cell to a target point (represented as a cross on the figure) whose position in the local frame of the corresponding cell C was given by the DRAW instruction's parameters. In all but the first segment, this led to the creation of an inter-segment connection with the homologous cell C in the preceding segment. In the first segment, however, as there was no preceding segment, the closest cell to the target point was cell B in the first segment itself. Therefore, a connection from cell B to cell C was created in the first segment. That connection complements a recurrent circuit implementing an oscillator whose periodic activity is propagated into the other segments of the animat by the inter-segment connections. Note that for reasons of clarity, we have reversed left and right cells A compared to Figure 6.7.

These four swimming controllers (controllers 7,8,9, and 10) modulate the speed of swimming when the level of tonic input is varied. They need the tonic input to produce oscillations; they converge to an equilibrium state, and hence a motionless body, when no input is given. Once sufficient input is applied, the speed of swimming increases with the level of input, and these controllers have a range of excitations in which the speed increase is monotonic with the excitation (even with smaller excitation steps than those used in the evaluation of the fitness value). Table 6.4 gives the speed ranges which can be obtained within that monotonic region. This monotonic relation is important, from a control point of view, in order to ensure that an increase of excitation amounts to an increase of speed.

Controllers 9 and 10 also offer good control of the direction of swimming. When, during swimming, an asymmetrical excitation is applied to these networks for a fixed duration, the lamprey turns proportionately to the asymmetry of excitation. The effect of the asymmetry is to change the duration of bursts between left and right muscles leading to a change of direction towards the side with the longest bursts. If the asymmetry

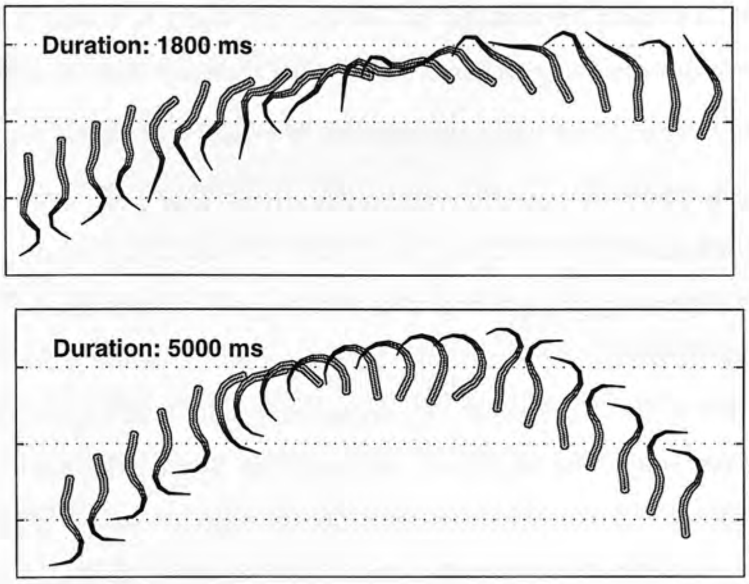


Figure 6.12: *Top:* slow turning induced by controller 10, *bottom:* Sharp turning induced by controller 9.

is small (for instance ±10% of the excitation level), the lamprey goes on propagating a traveling undulation and therefore swims in a circle until the asymmetry is stopped (Figure 6.12, top). If the asymmetry is large (for instance ±30% of the excitation level), the undulations almost stop as one side becomes completely contracted. This bending leads the lamprey to turn very sharply, and allows important changes of direction when the duration of the asymmetry is short (Figure 6.12, bottom).

6.3 Discussion

6.3.1 Method

We have presented how an evolutionary algorithm with a developmental encoding scheme could be used to develop interesting swimming controllers. This demonstrated that the method has the potential to develop, given a fixed body structure, efficient controllers which fulfill desired characteristics defined by the fitness function.

Evolution and search space The fact that the evolutionary process determines both the number of neurons, the number of connections and the parameters (synaptic weights, time constants and biases) of the neurons leads to a search space with, in principle, an infinite number of dimensions.[5] We therefore constrained the search in several ways to make the work of the evolutionary algorithm easier.

Firstly, we chose the initial positions and orientations of the precursor cells and output nodes in the substrate, which allowed us to force the developed controllers to be symmetrical and organized in segments, thus reducing the overall number of parameters to be evolved. Secondly, the choice of a specific grammar made it possible to further reduce the size of the search space by considering only a sub-space of the developmental program space. In particular, the grammar limited the numbers of neurons in the evolved controllers, therefore restricting the search space to a finite number of dimensions. The limited numbers of neurons also helped to keep the evaluation time of the corresponding neural networks within reasonable bounds.

As mentioned, the evolved controllers depend on several geometrical parameters fixed by the experimenter such as the number and the position of the precursor cells, the position of the muscle nodes, the maximum length of a connection, *etc.* which have partly determined the architecture of the evolved solutions. An interesting extension of this work would be to analyse the effect of these parameters, and potentially have them evolving as well.

Comparison with the previous evolutions of swimming controller In contrast to the staged evolution and direct encoding of the previous chapters, in the experiment reported here, we use a more compact encoding, the developmental encoding, and we evolve swimming controllers in one stage only. This new approach brings several interesting features.

Firstly, an interesting aspect of the developmental encoding is the fact that the number of neurons and of connections is not fixed *a priori* but is determined by the evolutionary process. In chapter 4, we evolved the number of neurons and connections indirectly by fixing a maximum number of neurons and evolving the weights of all possible connec-

[5] When no grammar is used, the size of the developmental program, and therefore the number of neurons, is, in principle, not limited, and the addition of a neuron means the addition of new variables (synaptic weights, time constants and biases) which each adds a new dimension to the search space.

tions while using a random pruning operator which little by little led to circuits with fewer connections and potentially fewer neurons.

Secondly, another interesting aspect of the developmental encoding is that the developmental instructions can participate in the construction of different neural mechanisms depending on the local context, as illustrated in the development of controller 9, when the same DRAW instruction was used to either complete an oscillatory circuit in the first segment, or create intersegmental connections in the other segments. Such a property is due to the side-effects that occur at the extremities of the segment chain and to the fact that the instructions for developing connections are context dependent. The evolutionary algorithm was opportunistic enough to take advantage of that property. One idea to make such opportunistic solutions appear more frequently would be to define more context dependent developmental mechanisms. This point will be further discussed in chapter 9.

Finally, swimming controllers have been developed in one stage only, with a fitness function which only specifies desired characteristics for the mechanical simulation, without considering neural aspects. Having a fitness evaluation only based on mechanical aspects is especially interesting for control problems where neural requirements are not known and where only the desired behaviour of the mechanical system can be characterized. An interesting application of the technique could be the generation of a locomotion controller for a swimming or walking robot, for instance.

The drawbacks compared to the previous evolutions are that the evolutions are significantly slower than with the staged approach. The performances in terms of frequency and speed range of the evolved solutions are also not as good, although this may be due to the different neuron type used: because Ekeberg's neurons have independent state variables for the excitatory and inhibitory inputs, they provide a better basis for oscillation as they have a more complex dynamics (see, for instance, Figure3.2 where constant inputs lead to an increase followed by a decrease of the neuron's output). Finally, because there were less constraints on the evolution, the evolutionary process took an opportunist advantage of small flaws of the fitness function which led to a lower success rate of the different evolutions to produce interesting swimming controllers. In the previous evolutions, first evolving segmental oscillators ensured that swimming controllers were able to produce regular oscillations and satisfactory swimming in all runs. The two methods will further be compared in chapter 9.

Comparison with previous applications of SGOCE In [Kodjabachian & Meyer 98a], walking controllers for a six-legged animat were evolved and several similarities can be noticed between the corresponding experiments and the ones described here.

Firstly, the same models of neurons and muscles were used in both cases. Moreover, a left-right symmetry and a segmental organization of the body and nervous system were hypothesized. Secondly, the set of developmental instructions and the syntactical constraints used were nearly identical, the only difference being the addition of the instruction SETINPUTW in the current work. Furthermore, the evolutionary algorithm was run with the same parameter values for solving both tasks. Only the population size was reduced in the current work, because of the longer time required by the evaluations. Finally, in both cases, locomotion controllers involving central pattern generators were successfully evolved.

The main differences between the experiments described here and those of [Kodjabachian & Meyer 98a] lay in the evaluation strategies and in the inclusion of command cells. While the experiments involving the six-legged animat used a global fitness criterion involving measures of body speed and leg movements, the current work uses a more elaborate fitness function that combines four criteria and calls for several evaluations of a given individual with different command settings.

Also, command cells were explicitly included in the current work, which allowed neural circuitries to be developed whose swimming patterns could be initiated and modulated by simple input signals. Note that, while no command input was used in the first stage of the evolution of six-legged locomotion controllers — which led to the evolution of constant-speed, straight-walking controllers — those controllers were however found to be able to stop [Kodjabachian & Meyer 98a] or to turn [Kodjabachian & Meyer 98b] when connected to a second, higher level neural network.

6.3.2 Evolved CPGs

A variety of different controllers have been evolved. The controllers differ both in their behaviours and their configurations. The evolved controllers present three kinds of neural activity: non-oscillating signals which propel the lamprey forward through an initial thrust, chaotic oscillations and stable oscillations. The corresponding configurations differ in the number of neurons, their parameters, and their connectivity. In particular the solutions can be classified depending on their numbers of oscillators: no oscillators, only one oscillator, and one or more oscillators per segment.

Evolved swimming patterns The controllers without oscillators, which propel the body forward by an initial thrust, are not very interesting from a control point of view as they can not sustain the speed of motion. These controllers have taken advantage of the limited time of simulation. To prevent the emergence of this kind of solution, further evolutions should be carried out with longer simulations which would significantly reduce their fitness value. The controllers producing unstable oscillations are also of limited interest because their speed and direction of swimming are not constant. These controllers would therefore be difficult to use by higher control centers. A simple way to prevent these solutions being generated could be to add a factor in the fitness function rewarding regularity of both the speed and the direction.

Four controllers have been developed which can be described as central pattern generators. They generate stable and permanent oscillations when receiving a tonic (i.e. non-oscillating) input. Of these controllers, the two which reach the highest speeds produce an anguiliform swimming very similar to that of the lamprey. Segments oscillate with left and right neurons out of phase, and there is a constant phase lag between segments which leads to a wave of neural activity traveling from head to tail. These two controllers offer good control of the speed and the direction. Increasing the amplitude of both tonic inputs amounts to increase the speed of swimming. One of our requirements was that this increase be monotonic, in order to facilitate control and to be certain that an increase of excitation amounts to an increase of speed. We can therefore define for both controllers a range of excitations in which that condition is respected. When asymmetrical levels of inputs are applied, turning is induced because of a difference of the duration of the muscular bursts between both sides. Although we have not required a monotonic relation between the turning capacity and the asymmetry of excitation (turning was tested at only one asymmetry), the turning angle increases with the amount of asymmetry. For both controllers, two kinds of turning can be obtained: a small asymmetry leads to slow turning in which the undulations continue to travel, and a large asymmetry leads to sharp turning with a strong bending of the body. Although this means that the body loses speed, it allows turning on the spot. For both types of turning, straight swimming resumes when the asymmetry ceases.

Comparison of the best controllers with the lamprey's CPG Controller 9 has a very simple neural configuration for creating the traveling wave of neural activity. It is made of an oscillator in the first segment (the segment closest to the head) which entrains, through closest neighbour excitatory connections, a chain of single neurons on both sides of the spinal cord. Controller 10 has a more complex configuration with oscillators in each segment. This controller is therefore closer to the biological connectivity found in the lamprey, with an interconnection of segmental oscillators which can be made to oscillate independently when isolated [Buchanan & Grillner 87, Grillner *et al.* 88, Grillner *et al.* 91, Grillner *et al.* 95] (see also section 2.3). Both evolved controllers share the property of the lamprey that the frequency of oscillation increases with the level of excitation. In the lamprey, this has been shown *in vitro* by measuring the frequency of oscillations in excitatory baths with different concentrations. Another interesting property of the lamprey is that the phase lag between segments relative to the cycle duration remains constant for different frequencies of oscillations. This means that the lamprey maintains a constant wavelength of the undulation (corresponding to approximately the length of the body) for any frequency of oscillations. In controller 9 this is not the case, as the relative lag between segments changes significantly with the frequency; in controller 10, however, the change of relative phase lag changes only slightly for important changes of the frequency (see Appendix G). Controller 10 therefore shares the most properties with the biological CPG of the lamprey.

By contrast with the previous chapters in which I developed swimming controllers with some predefined similarities with the biological configuration and with some requirements on the neural activity, we generated here controllers with only a characterisation of the desired behaviour in terms of the control of the speed and the direction of swimming. It is therefore interesting to see that the best evolutions converged to controllers sharing several similarities with the lamprey, the main one being the anguiliform swimming.

6.4 Summary

This chapter presented how swimming controllers for a mechanical simulation of a lamprey can be designed using an evolutionary algorithm with a developmental encoding scheme. Developmental programs determine how neurons on a 2D substrate divide and get connected to each other. The swimming controllers were evolved in one stage only, and with a fitness function based on the performance of the swimming, namely the control of speed and direction by two input tonic signals. The desired behaviour of a controller is therefore characterized at a higher level than in the previous chapters where neural aspects were also taken in account. Such a design method can potentially be applied to the evolution of controllers for a real swimming robot without major changes in the design technique.

A variety of different controllers were evolved. Some of the controllers have taken advantages of "flaws" of the fitness function and produce swimming which is either not permanent but based on an initial thrust, or not regular because of chaotic neural activity. The best controllers exhibit stable neural oscillations which can be varied by two input signals to modulate the speed and direction of swimming. The two best controllers produce swimming gaits which are very similar to that of the real lamprey with an undulation of almost constant wavelength traveling from head to tail. Because of the context-dependency of the developmental program, one of these two controllers has a suprisingly compact neural configuration.

some predefined connectivity with the bilateral configuration and with some constraints on the neural activity, we generated locomotion controllers with only a characterisation of the desired behaviour in terms of the control of the speed and the direction of swimming. It is therefore interesting to see that the best evolutions produced locomotion sharing several similarities with the lamprey, the most notable being the distributed swimming.

6.4 Summary

This chapter presented how swimming controllers for a mechanical simulation of a lamprey can be designed using an evolutionary algorithm with a developmental encoding scheme. Developmental programs determine how neurons on a 2D substrate divide and get connected to each other. The swimming controllers were evolved in one stage only and with a fitness function based on the performance of the swimming, namely the control of speed and direction by two input tonic signals. The desired behaviour of a controller is therefore characterized at a higher level than in the previous chapters where neural aspects were also taken into account. Such a design method can potentially be applied to the evolution of controllers for a real lamprey robot without major changes in the design technique.

A variety of different controllers were evolved. Some of the controllers have taken advantage of the fact that the simulation is frictionless and produce swimming which is not purely based on undulation but on other means of producing thrust, or that the mechanical simulation can produce thrust in other ways. Some controllers exhibit the same neural organisation as the lamprey with a chain of coupled oscillators, and the best controllers exhibit locomotion similar to the lamprey with two input signals which can modulate the speed and direction of swimming. The two best controllers produce swimming that is very similar to that of the real lamprey, and are more efficient than the lamprey-like controller. One of these two controllers has a distributed swimming. Because of the developmental nature of the encoding, one of these two controllers has a more compact neural organisation.

Chapter 7

From lamprey to salamanders: evolving CPGs for swimming and walking

I present next how swimming controllers can be extended to control the walking of a salamander-like animat. Keeping the idea of staged evolution in which increasingly complex control systems are incrementally built using elements of the previous stage, this can be seen as a fourth evolutionary stage following the evolution of the swimming controllers.

Salamanders propel themselves in water by undulation of the body while holding their limbs against their body. Similarly to lampreys, the undulation is a traveling wave propagating from head to tail. When the salamander switches from swimming to trotting, its body ceases to propagate an undulation and, instead, performs an S-shaped standing wave with the nodes at the level of the girdles [Frolich & Biewener 92]. The bending of the body helps the salamander to increase the reach of its limbs which are attached laterally to the body. Electromyographic recordings along the salamander's trunk and tail have shown that these two gaits are produced by two distinct axial motor programs, with a traveling neural wave for swimming and a standing wave for trotting [Frolich & Biewener 92, Delvolvé *et al.* 97].

As discussed in chapter 2, little is known about the neural circuitry controlling the locomotion of the salamander, but it has been hypothesised that it is based on a similar organisation to that of the lamprey [Cohen 88, Delvolvé *et al.* 97]. Studying the evolution of central pattern generators among vertebrates, Cohen, for instance, suggests that CPGs of higher vertebrates may have evolved from chains of coupled segmental oscillators, as observed in the lamprey, with some segmental oscillators having specialised to control fins and limbs. Furthermore, the possibility of a chain of oscillators producing the S-wave observed in the trotting salamander has been demonstrated using chains of mathematical oscillators with a coupling between the extremities and the middle of the chain [Ermentrout & Kopell 94a].

Following these ideas, I develop CPGs which can exhibit both the swimming and trotting gaits of the salamander and which are based on limb oscillators projecting to lamprey-like swimming circuits for the body (i.e. the trunk and the tail). The body circuitry is based on Ekeberg's model of the lamprey's CPG and on the controllers developed in chapter 4. The research presented here has been published in [Ijspeert *et al.* 98c] and [Ijspeert *et al.* 98a].

7.1 Methods

For investigating salamander-like locomotion, I developed a salamander-like animat by transforming the mechanical simulation of the lamprey. Taking the same approach as in chapter 4, neural controllers able to produce the swimming and trotting gaits are developed with a direct encoding. The neural controllers are based on the swimming controllers for the lamprey (both Ekeberg's model and evolved controllers), and are composed of two interconnected oscillators which control the limb muscles and which project to a lamprey-like body CPG.[1] The limb oscillators are copies of the body segmental oscillators (only the interneurons). The evolutionary process is used to define the synaptic weights of the connections between limb oscillators and the connections from the limb oscillators to the body CPG. A fitness function is defined which rewards the capacity of the controllers to modulate the speed and the direction of the mechanical simulation.

Two different experiments are reported, which correspond to evolutions with the same fitness function but with different constraints on the neural configuration. In experi-

[1] The experiments presented in this chapter are realised without sensory feedback, and the neural simulations therefore do not include edge cells or other proprioceptive neurons.

ment A, the limb oscillators project to the body motoneurons, while in experiment B, they project to the body interneurons (see below). In the next sections, I present the mechanical simulation of the salamander, the details of the neural configurations and their encoding for each experiment, and the fitness function used for the experiments.

7.1.1 Mechanical model of the salamander

The 2D mechanical simulation of the salamander is an extension of Ekeberg's simulation of the body of the lamprey. Rigid links representing the limbs are attached to the first and the fifth joints (Figure 7.1), therefore spliting the lamprey-like body into a five-segment trunk part and a five-segment tail part.[2] This kind of anatomy corresponds approximately to that of a metamorphosed[3] *Ambystoma Tigrinum* salamander [Frolich & Biewener 92]. In order to also approximate the variations of the body diameter from head to tail, the geometry of the body is changed compared to that used for the lamprey simulation, and the width of the links of the middle of the trunk are increased compared to those of the rest of the trunk. The new geometrical parameters are given in Table 7.1. Similarly to the lamprey, each link of the body is assumed to be 30 mm long, with an elliptical cross section of constant height (30 mm) and variable width. Limbs are simulated as 50 mm long cylinders, with a diameter of 12 mm. The masses and moments of inertia of the links are calculated by assuming the density of the salamander to be constant and equal to that of water.

Similarly to the lamprey, the accelerations of the links are due to three forces: the torques due to the muscles, inner forces linked with the mechanical constraints and the forces due to the environment.

Muscles are simulated as springs and dampers as for the lamprey's simulation (see Equations 3.8, chapter 3), but the parameters of the muscles of the tail and the trunk are changed in order to increase the torques they can create and to compensate for the

[2] In [Ijspeert *et al.* 98c], I used, similarly to [Delvolvé *et al.* 97], the term *trunk* for the whole body, with the *anterior trunk* corresponding to the intergirdle part of the body and *posterior trunk* corresponding to the tail. However, in other papers such as [Frolich & Biewener 92], the term *trunk* is only used to refer to the intergirdle part of the body, with the *anterior* and *posterior trunk* being different areas of that body part. To avoid any confusion, the terms *trunk* and *tail* are used in this chapter.

[3] This species has an alternative life cycle, the *neotenic* cycle, in which individuals reach adulthood in a fully aquatic larvaform [Frolich & Biewener 92].

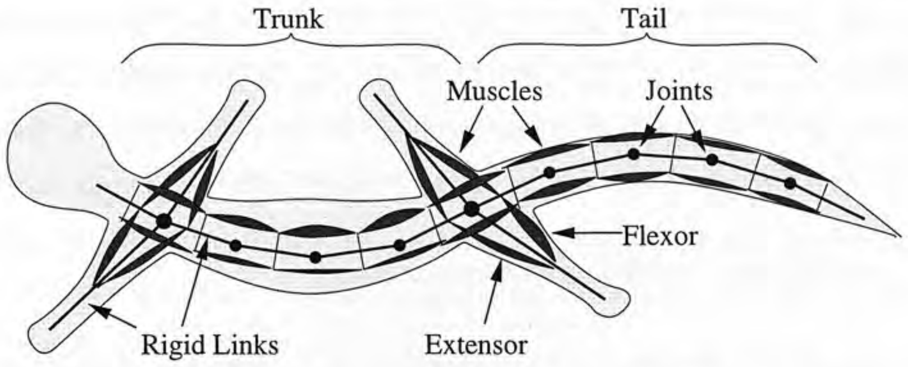


Figure 7.1: Mechanical configuration of the salamander-like animat.

link	w_i [mm]	m_i [g]	I_i [g mm^2]	$\lambda_\perp$ [Ns^2/m^2]	$\lambda_\parallel$ [Ns^2/m^2]
1	22.0	15.6	1637	0.45	0.3
2	25.0	17.7	2016	0.45	0.2
3	28.0	19.8	2454	0.45	0.1
4	23.0	15.6	1637	0.45	0.0
5	18.6	13.2	1272	0.45	0.0
6	15.2	10.8	964	0.45	0.0
7	11.8	8.4	701	0.45	0.0
8	8.5	6.0	475	0.45	0.0
9	5.1	3.6	275	0.45	0.0
10	1.7	1.2	90	0.45	0.0
limbs	12.0	5.7	1229	0.30	0.1

Table 7.1: Parameters for the mechanical simulation of the salamander-like animat.

larger masses and moments of inertia of the body. In the simulations presented here the parameters for the trunk and tail muscles are: $\alpha = 20.0$ [N mm], $\beta = 2.0$ [N mm], $\gamma = 20.0$ and $\delta = 200.0$ [N mm ms]; and those of the limb muscles: $\alpha = 20.0$ [N mm], $\beta = 2.0$ [N mm], $\gamma = 4.0$ and $\delta = 200.0$ [N mm ms]. To compensate for the differences in torques required for the swing and stance phases, the flexor motoneuron signals are amplified by 30% while the extensor motoneuron signals are reduced by 40% when the torques are calculated.

The inner forces ensure that all links stay connected, and correspond to those of the lamprey's simulation with new constraints for the attachment of the limbs. When the limbs enter into contact with the trunk or the tail links, elastic and damping forces are also temporary added to prevent the limbs penetrating those links: $F_{contact} = Ax + B\dot{x}$, where $A = 0.1$ [N/mm], $B = 2.0$ [N ms/mm] and x represents the distance that the end of the limb penetrates the body link.

The forces due to the environment depend on whether the salamander is in water or on land. In water, it is assumed that each link (limbs included) is subjected to inertial forces due to the water, and these forces are calculated as in the lamprey's simulation. On land, all trunk and tail links are subjected to a friction force, representing the fact that the body of the salamander slides on the ground when the salamander is trotting. The friction force is opposite to the motion of the links and proportional to their weight (gm): $F_{env} = \mu gm$. In the simulations presented here, μ has been fixed at 0.4.

As only the accelerations in the horizontal plane are calculated, vertical movements of the limbs are not calculated, and instead, the position of the extremity of the limb (the foot), in the air or on the ground, is assumed to be determined by the signals sent to the limb muscles. The limb is assumed to be in the air when the signal of the extensor is larger than that of the flexor, and on the ground otherwise. The contact of a limb with the ground is then represented as a constant friction force applied to the extremity of the limb link ($F_{foot\_ground} = 1.0[\mathrm{N}]$). The motoneurons for the flexor and extensor therefore not only determine the torque of the limb, but also its stance and swing phase.

The same integration method as for the lamprey's simulation is used, namely the fourth order Runge-Kutta method, with integration steps of 5 ms for the neural simulation and integration steps of 0.5 ms for the mechanical simulation.

Note that this simulation is only a first approximation of a salamander, and does not claim to be a realistic biomechanical simulation of any existing real salamander. In particular, the different parameters of the simulation, such as the geometrical parameters and the muscle parameters, have not been tuned to correspond to biological data.

7.1.2 Neural configuration in experiment A

The neural configurations in experiment A are made of a limb CPG and a body (trunk and tail) CPG (Figure 7.2). The body CPG corresponds to the swimming controller for the lamprey. Two sets of evolutions are carried out: in the first one the body CPG corresponds to Ekeberg's biological model; and, in the second one, the controller

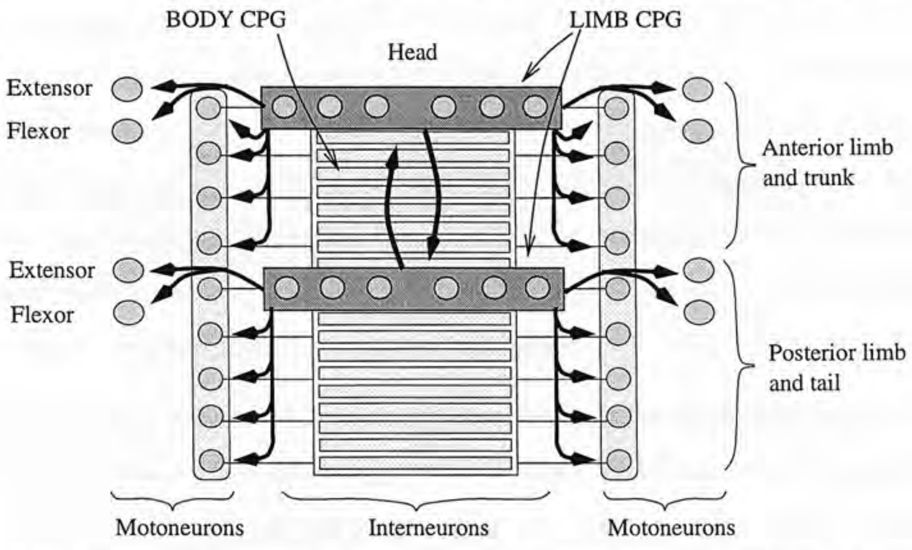
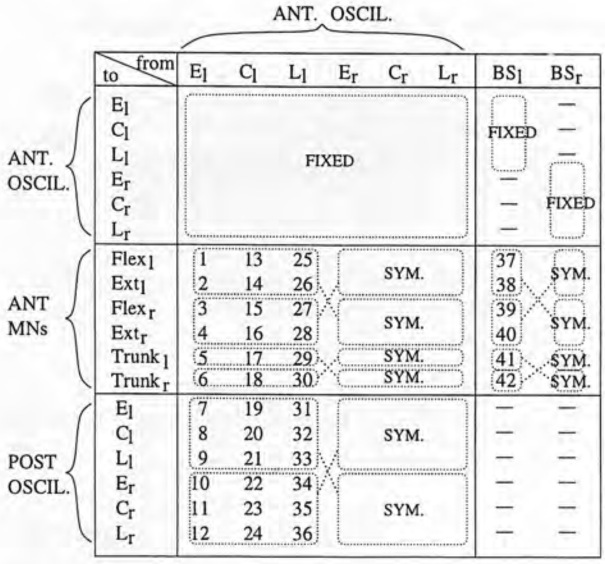


Figure 7.2: Neural configuration in experiment A. Arrows represent bundles of connections whose weights are evolved.

corresponds to one of the evolved swimming controllers of chapter 4 (controller 1 which, similarly to Ekeberg's model, has 6 interneurons per segmental network). The limb CPG is made of two oscillators which are copies of the 6-interneuron segmental oscillators of the body CPG.

In this experiment, I choose the limb oscillators to be interconnected, and to project to the motoneurons of the corresponding limbs and to the motoneurons of the body muscles, with the anterior limb oscillator projecting to the trunk motoneurons (joints 1 to 4), and the posterior limb oscillator projecting to the tail motoneurons (joints 5 to 9, see Figure 7.2). The evolutionary process is used to determine the synaptic weights of all the possible connections. In order to take into account the symmetry of the body and to reduce the search space, a left-right symmetry is imposed. The limb oscillators are also assumed to project identically to the corresponding trunk and tail motoneurons, i.e. joints 1 to 4 receive the same connections from the anterior limb oscillator and similarly for joints 5 to 9 and the posterior limb oscillator.

A direct encoding similar to that used in chapters 4 and 5 is used. A chromosome consists of 84 genes (real numbers $\in [0,1]$) which encode the synaptic weights of the connections described above (Figure 7.3). For the connections to the muscle motoneurons, the genes are transformed into synaptic weights in the range $[-5.0, 0.0]$ or $[0.0, 15.0]$



	to \ from	E_l	C_l	L_l	E_r	C_r	L_r	BS_l	BS_r
		ANT. OSCIL.							
ANT. OSCIL.	E_l	FIXED						FIXED	—
	C_l								—
	L_l								—
	E_r							—	FIXED
	C_r							—	
	L_r							—	
ANT MNs	$Flex_l$	1	13	25	SYM.			37	SYM.
	Ext_l	2	14	26				38	
	$Flex_r$	3	15	27	SYM.			39	SYM.
	Ext_r	4	16	28				40	
	$Trunk_l$	5	17	29	SYM.			41	SYM.
	$Trunk_r$	6	18	30	SYM.			42	SYM.
POST OSCIL.	E_l	7	19	31	SYM.			—	—
	C_l	8	20	32				—	—
	L_l	9	21	33				—	—
	E_r	10	22	34	SYM.			—	—
	C_r	11	23	35				—	—
	L_r	12	24	36				—	—

Figure 7.3: Encoding of genes 1 to 42 in experiment A. Genes 43 to 84 similarly encode the projections from the posterior oscillator.

if the presynaptic neuron is, respectively, inhibitory or excitatory, and for the connections between oscillators, the synaptic weights are in the range $[-2.0, 0.0]$ or $[0.0, 6.0]$. These connections between oscillators are chosen to have smaller synaptic weights in order to reduce the effect of inter-oscillator coupling on the oscillator activity.

The aim is to develop controllers which produce either the swimming gait or the trotting gait depending on which CPG receives most excitation. The whole CPG can be seen to be made of two CPGs in parallel which "compete" for the same motoneurons.

7.1.3 Neural configuration in experiment B

In experiment B, neural configurations have the same organisation as for experiment A, except that the limb oscillators project to the interneurons of the body CPG rather than the body motoneurons (Figure 7.4). Conversely to experiment A, in which the limb and body CPGs are not coupled, there is here a one-directional coupling between the two CPGs, which is similar to the control circuitry hypothesised by neurobiologists in [Delvolvé *et al.* 97].

Only one set of evolutions is carried out, with body circuitry corresponding to Ekeberg's model of the lamprey CPG. Similarly to the experiment A, the two oscillators of the

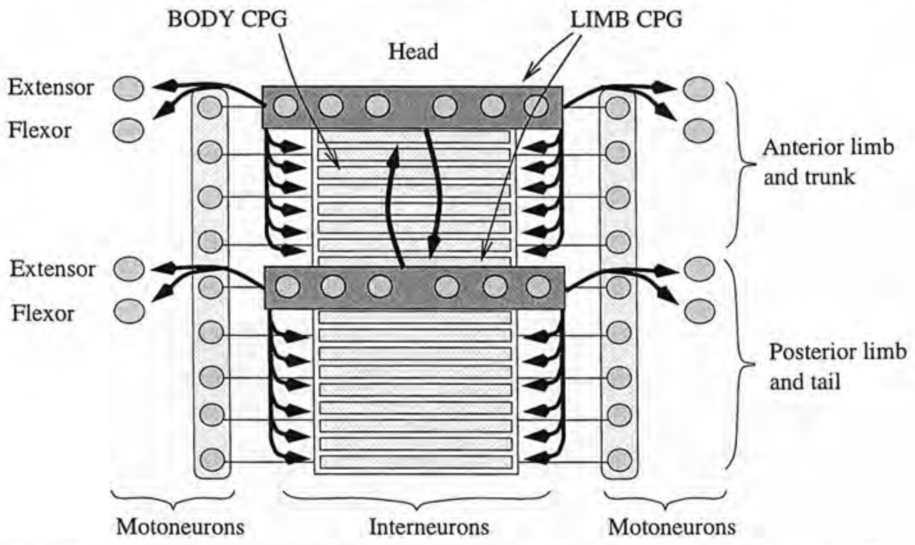


Figure 7.4: Neural configuration in experiment B. Arrows represent bundles of connections whose weights are evolved.

limb CPG are copies of the 6-interneuron segmental networks of the body circuitry. The oscillators project to the interneurons of the body segmental oscillators, with the anterior oscillator projecting to the anterior (trunk) segments, i.e. the first 50 segments between the head and the hindlimbs, and the posterior oscillator projecting to the last 50 segments of the tail. In order to reduce the search space, I fix a left-right symmetry and a symmetry of the projections to the body segments, i.e. all segments from the trunk receive the same connections from the anterior limb oscillator and similarly for the tail segments and the posterior limb oscillator.

The synaptic weights of the connections are encoded into chromosomes consisting of 104 genes (Figure 7.5). The weights of the connections between the limb oscillators are in the range $[-2.0, 0.0]$ or $[0.0, 6.0]$ depending on the sign of the presynaptic neuron, and the weights of the connections to the limb motoneurons and the body interneurons are in the range $[-5.0, 0.0]$ or $[0.0, 15.0]$.

In this experiment, the aim is to develop controllers which can produce the swimming gait when only the body CPG receives tonic excitation, and the trotting gait when *both* the body and the limb CPGs receive excitation. As the limb oscillators and the segmental oscillators of the body are identical networks with the same connections from the brain stem, they have the same intrinsic frequency when they receive the same

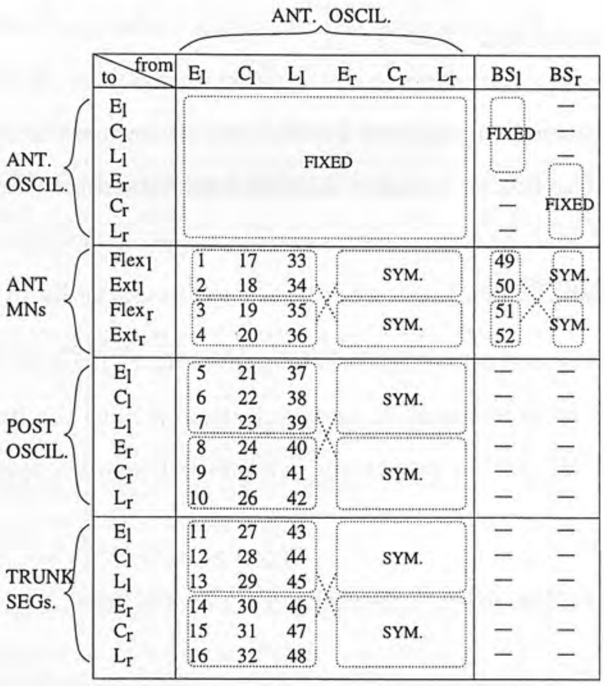


Figure 7.5: Encoding of genes 1 to 52 in experiment B. Genes 53 to 104 similarly encode the projections from the posterior oscillator.

excitation. The connections between limb oscillators and from the limb oscillators to the body oscillators will then determine the phase relation between all the oscillators.

7.1.4 Genetic Algorithm

For these experiments, a similar genetic algorithm to the one described in chapters 4 and 5 is used. The 2-point crossover and mutation operators are used, together with a pruning operator. The pruning operator is only applied to the connections between the limb oscillators. The GA parameters used in both experiments are given in Table 7.2.

Population size	200 (exp. A) 100 (exp. B)
Number of children	60 (exp. A) 30 (exp. B)
Crossover probability	0.5
Mutation probability	0.4
Mutation range	0.2
Pruning probability	0.05

Table 7.2: GA parameters for experiments A and B

7.1.5 Fitness function

The *fitness* of a controller depends on its capacity to control the motion of the mechanical simulation. The fitness function is defined to reward solutions which:

1. trot as fast as possible,
2. can change the speed of the trot when the excitatory drive is varied (excitation of the limb CPG in experiment A, and excitation of both the limb and body CPGs in experiment B), with a monotonic relation between the level of excitation and the speed,
3. can change direction when left-right asymmetrical drive is applied,
4. maintain, most of the time, one foot on the ground on each side of the body (this factor was added to prevent the emergence of gaits which would tend to roll in a 3D simulation).

The mathematical definition of the fitness function is the following:

$$\mathit{fitness} = \mathit{fit\_max\_speed} \cdot \mathit{fit\_speed\_range} \cdot \mathit{fit\_turning\_slow} \cdot \mathit{fit\_turning\_fast} \cdot \mathit{fit\_feet\_ground} \in [(0.05)^5, 1.0]$$

where *fit_max_speed, fit_speed_range, fit_turning_slow, fit_turning_fast* and *fit_feet_ground* are functions which are limited between 0.05 and 1.0 and which vary linearly between these values when their corresponding variables vary between two boundaries, a *bad* and a *good* boundary (same transformation function as in chapter 4). The variables for each function and their corresponding boundaries are given in Table 7.3. An evaluation consists of several simulations (1000ms) with different levels of excitatory drive for determining the range of speeds which can be obtained. If the speed range includes the chosen speeds of 0.15 and/or 0.30 m/s, the capacity to induce turning is measured at the corresponding level of excitation and *fit_turning_slow* and/or *fit_turning_fast* is measured. Turning is induced by applying a symmetrical excitation for 1200ms (straight motion), followed by a left-right asymmetrical excitation ($\pm 20\%$) for 600ms, followed by 1200ms of symmetrical excitation. The angle of deviation corresponds to the difference of headings before and after the asymmetrical excitation. The factor *fit_feet_ground* depends on the minimal time left or right feet have spent on the ground in all the simulations of the evaluations.[4] The lower this variable is, the more likely it is that the corresponding simulation would tend to roll in a 3D simulation, as it indicates that the body is insufficiently supported on one side during a part of the cycle. A value of 1.0 means that, in all simulations, a leg (either the fore- or the hindlimb) was kept on the ground on each side of the body at all times.

Function	Variable	[*bad,good*] Boundaries
fit_max_speed	Maximum speed	[0.0, 0.6] m/s
fit_speed_range	Relative speed range	[0.0, 1.0]
fit_turning_slow	Deviation angle	[0.0, *pi*]
fit_turning_fast	Deviation angle	[0.0, *pi*]
fit_feet_ground	Minimum time on ground	[0.5, 0.9]

Table 7.3: Variables and boundaries for the fitness function. The speed range is measured relative to the maximum speed.

7.2 Results of experiment A

I carried out two sets of 10 evolutions of 40 generations starting with different random populations of 200 chromosomes each. The first set of evolutions develops controllers based on Ekeberg's model as body CPG, while the second set is based on one of the evolved swimming controllers of chapter 4 (controller 1). The fitness and maximum speed of the best evolved controllers of each run are given in Table 7.4. The corresponding controllers and graphs showing their performance can be found in Appendixes H and I. Note that some animated gifs illustrating the results can also be found at http://www.dai.ed.ac.uk/students/aukei .

Performance of the controllers All evolutions converged to controllers which use the four legs for locomotion (see Appendixes H and I). The controllers all exhibit a gait which is very close to the trot with contralateral limbs oscillating out of phase and

[4] For each simulation i, the time t_{i_left} (and t_{i_right}) an anterior or posterior foot has been in contact with the ground on the left (and right) side of the body is calculated (as a ratio of the duration of the simulation). The variable *minimum time on ground* is then the minimum of all t_{i_left} and t_{i_right}.

Table 7.4: Fitness and maximum speed of the trotting controllers based on Ekeberg's swimming controller for the body CPG in runs A1 to A10, and on one of the evolved swimming controllers in runs A11 to A20.

Run	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
Fit	0.09	0.11	0.06	0.06	0.08	0.12	0.19	0.09	0.15	0.10
Spd m/s	0.52	0.60	0.52	0.45	0.48	0.70	0.55	0.52	0.78	0.70
Run	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20
Fit	0.25	0.02	0.05	0.07	0.07	0.05	0.03	0.04	0.04	0.05
Spd m/s	0.59	0.46	0.33	0.52	0.73	0.40	0.38	0.37	0.40	0.47

diagonally opposed limbs almost in synchrony. This type of gait was favoured by the *fit_feet_ground* factor of the fitness function and by the constraints of the mechanical simulation. The *fit_feet_ground* factor has penalized solutions which use only two limbs, and the mechanical constraints due to the joint-less rigid limbs have penalized gaits in which contralateral limbs are simultaneously on the ground during part of the cycle, as, when two opposite feet are on the ground, the single degree of freedom between the two limbs is fixed and the motion of the salamander is "braked" by the friction forces representing the contact of the two feet on the ground. Because of the penalization of the coactivation of contralateral limbs, the trot is the only gait which satisfies *fit_feet_ground*'s equilibrium requirement of having one foot on the ground on each side most of the time.

The fastest gaits are obtained when body movements are used and when these movements are appropriately coordinated with the limb movements. This is the case, for instance, in all controllers reaching speeds of 0.55 m/s or higher (controllers A2, A6, A7, A9, A10, A11 and A15). Similarly to the trotting gait used by the real salamander, these fastest solutions synchronously bend one side of the trunk with the swing phases of the contralateral forelimb and the ipsilateral hindlimb (see Figure 7.6), which therefore leads to an increased reach of these limbs and allows higher speeds to be obtained. As there is some time delay between the contraction of the body muscles and the maximal bending of the body, the best coordination is obtained when the beginning of the contraction of one side of the trunk slightly precedes (and covers most of) the ispilateral contractions of the forelimb flexor and the hindlimb extensor (Figure 7.6).[5]

[5] Because of the elasticity of the trunk, there is always a delay between the burst of motoneuron activity of one side of the trunk and the maximal bending of that side. This is similar to what is

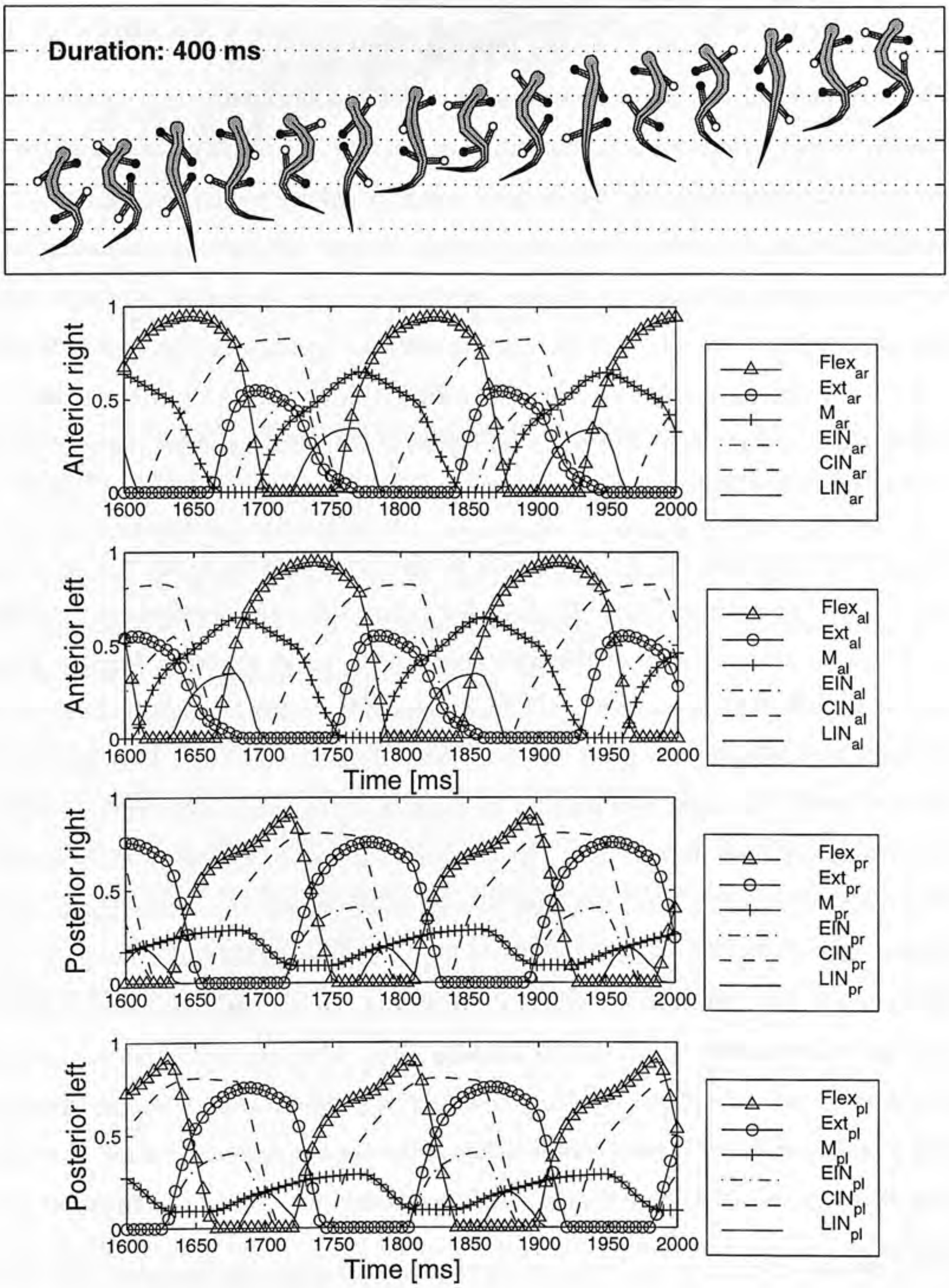


Figure 7.6: Trotting controlled by controller A15. M_a and M_p correspond to the motoneuron activity in the trunk and in the tail, respectively.

Interestingly, the same pattern of activity of trunk muscles, compared to the ispilateral limb muscles of the forelimb active during the stance phase and those of the hindlimb active during the swing phase, is shown in the data presented in [Delvolvé *et al.* 97].

For all the controllers presenting a trunk-limb coordination similar to that of the salamander, except controller A10, the tail is contracted out of phase compared to the trunk, which means that the whole body makes, similarly to the real salamander, a standing S-wave with the nodes at the girdles. Because the wave is a standing wave, the body becomes straight during some part of the cycle. Note that, when the tail is contracted without an out of phase relation with the trunk, as is the case in controller A10, the tail makes big sweeping movements which prevents the salamander from trotting in a straight line (Appendix H, Figure H.10). The sweeping movements are worsened by the pendulum effect induced by the swinging of the rigid hindlimbs.

The speed of trotting can be modulated by the level of tonic input applied to the limb CPG (Appendixes H and I). Increasing the tonic input amounts to increasing the frequency of oscillation of the limb oscillators,[6] which therefore leads to higher trotting speeds. This is similar to the real salamander which increases trotting speed primarily by increasing the cycle frequency [Frolich & Biewener 92]. In some controllers, the increase of input also leads to an increase of the motoneuron amplitudes and therefore larger limb movements. The controllers based on Ekeberg's CPG as body CPG (controllers A1 to A10) reach on average higher speeds than those based on the evolved body controller (controllers A11 to A20). The reason for this is not very clear, but I suspect it is linked to an intrinsic property of the two limb oscillators based on the evolved body CPG which has led in many cases, when the oscillators are coupled, to a coupled network whose frequency does not vary significantly with the excitation level (see Appendix I). It was therefore difficult for the evolutionary process to develop controllers which could cover a large range of speeds.

The direction of trotting can be varied by applying left-right asymmetrical tonic input to the limb CPG (Figure 7.7). Turning is then induced because of a variation of the

observed in the real salamander where the cessation of EMG activity in one side of the trunk occurs prior to maximal lateral displacement to the contralateral side [Frolich & Biewener 92].

[6] This a characteristic of the segmental oscillators of the lamprey's CPG and of the evolved swimming CPGs.

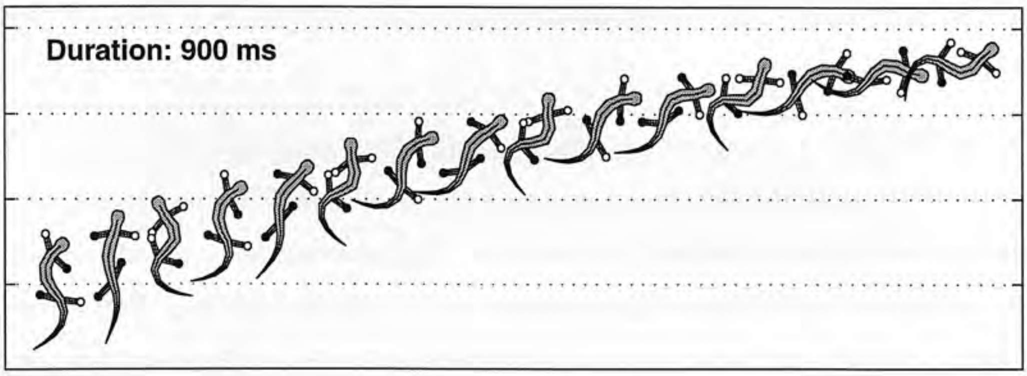


Figure 7.7: Turning induced by controller A9.

amplitude of the limb motoneurons which leads the limbs on one side of the body to make larger movements than the other. The controllers which have the best turning capacity also use trunk movements for turning, by making larger contractions of one side of the trunk in the direction of the turn. Note that two controllers, A7 and A11, had their turning capacity, and therefore their fitness value, overestimated because of inaccuracies in the numerical integration.[7]

Finally, the swimming gait can be obtained when tonic input is applied only to the body CPG, with some extra excitation to the five most rostral segmental networks. Tonic input is then also applied to the flexor muscles in order to hold the limbs against the body. The same traveling waves as for lamprey-like swimming are then produced, leading to typical anguiliform swimming. Because of the differences in geometry, the speeds of swimming are approximately 20% lower than with the lamprey simulation.

Configurations of the controllers Although the controllers of the different runs exhibit a similar trotting gait, a variety of different limb CPG configurations have been developed (both for the controllers based on Ekeberg's body CPG and those based on the evolved body CPG), with the couplings between the two limb oscillators

[7] Because of inaccuracies in the numerical integrations, these two controllers do not trot straight at some levels of excitation and therefore received a high fitness value because ability to turn was measured around those levels. For these controllers, an integration for the neural simulation with steps of 5 ms leads to oscillations with a "double" period: a short period which corresponds to the trotting steps and long period (for instance, 7 times the short period for controller A11) in which the amplitude of the neuron bursts vary, leading to a zigzag progression. The double period disappears with only the short period remaining when smaller integration steps are taken (1 ms steps).

and the projections to the motoneurons varying significantly from one run to another (Appendix H and I).

The coupling of the two limb oscillators has the effect of significantly changing their neural activity, such as the burst durations or the amplitude of the bursts, compared to when the oscillators are isolated. In many cases, for instance, some types of interneuron do not fire anymore in one of the segments because of the coupling. The variety of couplings has also led the anterior and posterior segments to oscillate with a variety of phase relations, from oscillators which are approximately in phase, such as in controller A8, to oscillators which are approximately out of phase, such as in controller A3. Note that, because the burst durations between neurons of the two oscillators are not identical due to the coupling, the phase relation between the two oscillators can not be measured accurately (one should measure the phase separately for the onset and the offset of the burst, and this for each type of interneuron).

The fact that the phase relation between the oscillators varies significantly from one run to another means that the connections from the oscillators to the limb and body motoneurons also vary significantly in order to produce the trotting gait observed in all solutions. In order words, the projections from the oscillators to the motoneurons compensate the different phases between the two oscillators observed in the different controllers in order to produce a similar motoneuron output for all controllers.

For the controllers exhibiting a coordination between body and limb movements similar to that of the salamander, the coordination is sometimes reflected by strong similarities of the projections to the motoneurons in the neural configurations. In controller A9, for instance, the projections from the anterior oscillator to the trunk motoneurons and the anterior flexor motoneuron on the same side are very similar, which leads to an almost identical neural activity of the ispilateral trunk and anterior flexor motoneurons.

7.3 Results of experiment B

Experiment B is carried out with 10 evolutions of 50 generations starting with different random populations of 100 chromosomes each. The body CPG corresponds to Ekeberg's CPG model. The fitness and maximum speed of the best evolved controllers of each run are given in Table 7.5. These best controllers are analysed below.

Table 7.5: Fitness and max. speed of the walking controllers

Run	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
Fit	0.05	0.01	0.07	0.02	0.02	0.07	0.01	0.03	0.11	0.01
Spd m/s	0.64	0.40	0.52	0.38	0.54	0.65	0.25	0.45	0.75	0.34

Performance of the controllers Similarly to the evolutions of experiment A, all runs converged to solutions exhibiting a trotting gait. The trotting gait is here obtained by applying excitatory drive not only to the limb CPG but also to the body CPG, which means that the interneurons of the segments of the body are activated. As the same excitatory drive is applied to the two segments of the limb CPG and to those of the body CPG, all segments oscillate at the same frequency and are therefore phase-locked.

All evolved controllers, except three (runs B7, B8 and B10), exhibit a body-limb coordination very similar to that of the salamander (Appendix J). The three exceptions have the trunk and tail segments of the body oscillating in phase which leads to the body making C-like undulations. The resulting movements are not well coordinated with the limbs, and the maximal speed of these solutions is therefore relatively low. For the other controllers, the trunk and the tail are approximately out of phase leading to the typical standing S-wave undulation (see Figure 7.8, top). Similarly to the real salamander and to the previous experiment, the body movements are coordinated with the limb movements such as to increase their reach when they are in the swing phase. The controller B9 presents, similarly to the best controllers of the previous experiment, a body-limb motoneuron coordination very similar to data of the real salamander in [Delvolvé *et al.* 97].

Unlike experiment A, in which the activity of the body motoneurons is directly determined by the limb oscillators, the limb oscillators project here to the interneurons of body segments and therefore influence the body motoneurons only indirectly. The S-wave undulation is obtained because the limb oscillators force the trunk and tail segments to oscillate out of phase. The neural activity within the body segments is therefore the result of the interaction between the lamprey-like coupling between segments which tends to make neighbouring segments oscillate with a similar phase, and

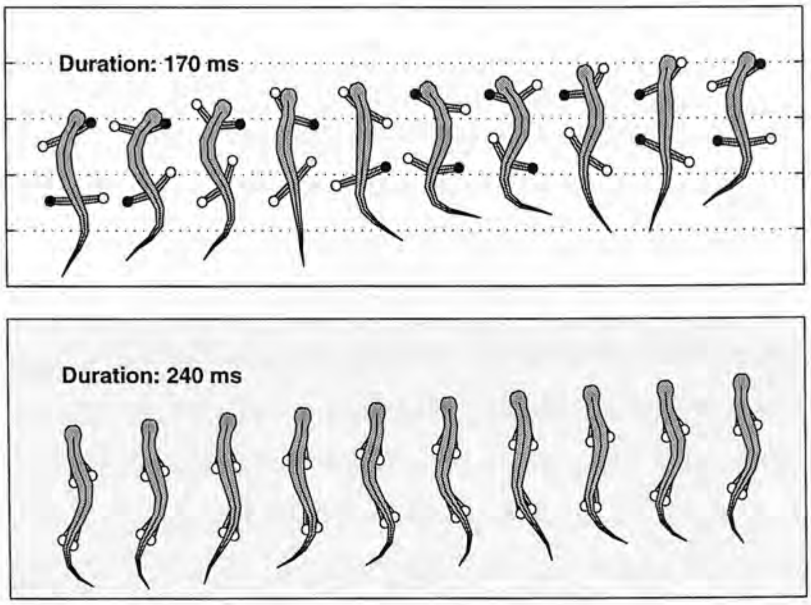


Figure 7.8: Trotting (*top*) and swimming (*bottom*) salamander with the fittest controller (run B9). Notice the typical standing wave of the body during trotting compared to the traveling wave during swimming. See also animated gifs at http://www.dai.ed.ac.uk/students/aukei/ , in particular the gifs illustrating the transition from swimming to trotting.

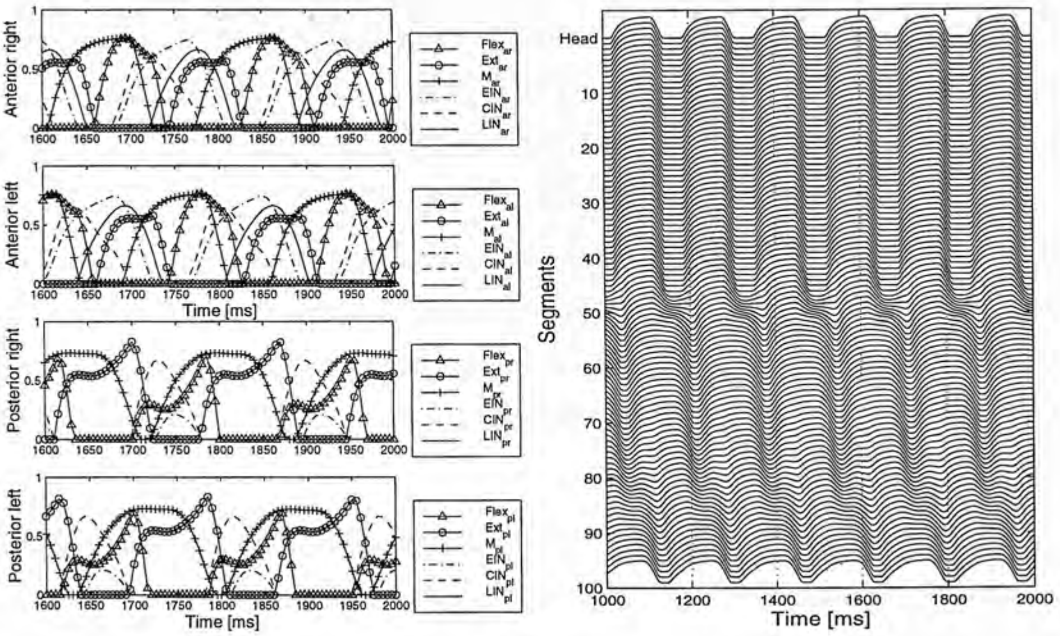


Figure 7.9: Neural activity during trotting (run B9). *Left:* Neural activity in the limb oscillators (M_a and M_p represent the motoneuron activity of body segments 5 and 95, respectively). *Right:* Motoneuron activity along the left side of the body. Remember that interneurons of segments 1 to 50, i.e. the trunk segments, receive projections from the anterior limb oscillator, while segments 51 to 100, the tail segments, receive projections from the posterior limb oscillator.

the projections from the limb oscillators which leads to a steep change of phase at the level of segment 50. The effect of the coupling is, for instance, illustrated in Figure 7.9, right.

A general observation about the activity of the body circuitry is that there is less variability in the signal shapes and phases in the trunk segments than in the tail segments. In all controllers, the first 40 segments (in the trunk) oscillate with identical phases and signal shapes (see Appendix J).[8] There is then an area between segment 40 and segment 60 (i.e. around the transition from anterior to posterior limb oscillator influences) in which signal shapes and phases change significantly. The rest of the tail segments then tend to oscillate with identical signal shapes, either in synchrony (controllers B1, B2, B3, B6, B7) or with caudally-directed phase lags (controllers B4, B5, B8, B10). In some controllers such as B4 and B9 (Figure 7.9), the signal shapes also change along the tail segments. The fact that, for all controllers, trunk rather than tail segments oscillate in synchrony is probably due to the caudal asymmetry of Ekeberg's intersegmental coupling.

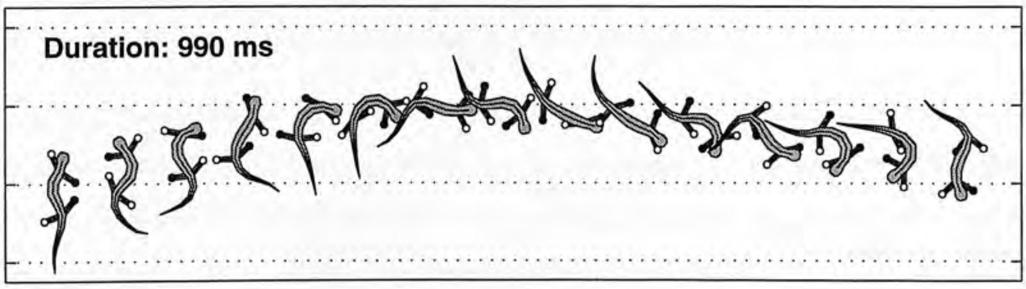


Figure 7.10: Turning induced by controller B9.

Similarly to the experiment A, the gaits produced by the evolved controllers can be modulated by the excitatory drive, and the speed of trotting increases with the level of excitation applied to the body and limb CPGs. Turning can also be induced by a left-right asymmetry of input (to both the limb and the body CPGs), and controller B9, in particular, can exhibit sharp turning movements because of large differences in the

[8] Note that with Ekeberg's intersegmental coupling, there is no lag between segments as long as no extra excitation is given to the first segments, and therefore both the intersegmental coupling and the projections from the anterior oscillator tend to make segmental networks in the trunk oscillate in phase.

contraction between the left and right sides of the trunk, in addition to differences in the amplitude of the limb motoneurons (Figure 7.10). Finally, when external excitatory drive is applied only to the body CPG with some extra excitation of the most rostral segments, the lamprey-like swimming gait can be produced (see Figure 7.8, bottom). Tonic drive is then also applied to the flexor muscles in order to hold the limbs against the body.

Configurations of the controllers There is, similarly to the previous experiment, no identical neural structure between the different controllers, but rather a variety of different neural configurations (Appendix J), which however produce relatively similar gaits. The coupling between the two limb oscillators leads to several phase relations between them, depending on the controllers. In controller B4, for instance, the two oscillators are almost out of phase while, in controller B9, they are close to synchrony. Similarly to experiment A, these different phase relations between oscillators means that the projections from the oscillators to the limb motoneurons and to the body segments also vary significantly from one controller to another for producing the trotting gaits.

7.4 Discussion

7.4.1 Gaits

The best evolved CPGs produce a trotting gait similar to that of real salamanders with the fore- and hindlimbs out of phase and the body making S-shaped standing undulations which increase the reach of the limbs. Although the fitness function was defined to reward trotting gaits and therefore the emergence of trotting gaits is not surprising, it is interesting to see that optimizing the speed and the control of direction has led to a limb-body coordination very similar to that of the salamander.

The gaits do not produce, however, an exact reproduction of a salamander gait and, for instance, for many controllers, the body movements are significantly more accentuated than during trotting in the real salamander. The tail, in particular, has a tendency to make large sweeping movements while in the real salamander the tail stays much more rigid with only slight movements to compensate for the trunk movements. The differences are mainly due to the fact that the mechanical simulation is only a first approximation of a salamander's body (see below), and also probably because energy efficiency was not taken into consideration by the fitness function. The evolved gaits may therefore be near optimal for the speed of locomotion (and the control of direction), but suboptimal for the speed versus mechanical energy consumption ratio, with a large amount of mechanical energy being used for the body movements compared to the forward progression. An interesting extension of this work would be to include a factor in the fitness function rewarding such energy efficiency.

Another difference with the gait of the real salamander is that, for all controllers making body undulations, the head makes big lateral movements during trotting. In the real salamanders, muscles in the neck oscillate out of phase compared to the trunk muscles and therefore compensate for the body movements by keeping the head oriented towards the direction of progression [Frolich & Biewener 92, Delvolvé *et al.* 97]. A similar feature could be obtained in these simulations if the mechanical simulation was extended to have more links for representing the whole body and if special projections from the anterior oscillator were evolved for the neck muscles.

The speed of trotting of the evolved controllers can be modulated with the level of excitation applied to the trotting CPG (i.e. to the limb CPG in experiment A and to both the limb and the body CPG in experiment B). All these controllers have a range of levels of excitation in which the speed increases monotonically with the excitation. The increase of speed is mainly due to an increase of the frequency of oscillation, and in some cases also to an increase in the step size. Similarly, in the *Ambystoma Tigrinum* salamander, the speed of trotting is mainly due to an increase of the stepping frequency [Frolich & Biewener 92]. In other salamander species which also use walking gaits, the increase of speed from walking to trotting is accompanied not only by an increase of frequency, but also an increase in step size and a decrease of contact interval [Ashley-Ross 94b].

Turning can be induced by the controllers when asymmetrical excitation is applied to the trotting CPG. The asymmetrical excitation leads to larger limb movements on one side of the body as long as the asymmetry lasts. The controllers which can induce the sharpest turning also use larger trunk contractions in the direction of turning, as controller B9, for instance. Note that, unlike the experiment on the evolution of swimming controllers for the lamprey where the turning capacity more or less emerged without being rewarded by the fitness function, factors explicitly rewarding turning were necessary for the evolution of locomotion controllers for the salamander. Initial experiments showed that, without such factors, most evolved controllers could only move in a straight direction even with asymmetries of input [Ijspeert *et al.* 98c]. I have not found kinematic or EMG studies of turning in the real salamander with which the evolved controllers could be compared.

7.4.2 Neural controllers

In both experiments, the locomotion controllers consist of two parts, a limb CPG and a body CPG. The difference between the two experiments lies in the organisation of the controllers, with experiment A having a "parallel" organisation in which the limb and body CPGs do not interact directly but they project to the same motoneurons, while in experiment B, there is a more "hierarchical" organisation, with the limb CPG projecting to the body CPG, creating a unilateral coupling between them.

The limb CPGs are made of two oscillators which are copies of the segmental networks of the body circuitry. This could be seen as segmental oscillators of the body CPG having specialised to control the limbs. It seems reasonable to think that a similar specialisation has occurred in vertebrates through natural evolution. It is well established that evolution from swimming gaits to legged gaits has seen morphological changes of the bones and the musculature of some segments of the body to become fins and then limbs. It is therefore quite probable that the oscillators for the limbs have followed a corresponding specialisation from the body segmental networks (see [Cohen 88] for a discussion).

It is less clear, however, what kind of interconnections between the segments of the body circuitry have evolved to allow both travelling waves for swimming and standing waves for trotting in the salamander. Here, similarly to what has been hypothesized in [Delvolvé *et al.* 97], controllers are developed which are based on a lamprey organisation, with limb oscillators providing phasic input to the body CPG for producing the standing wave. Of the two experiments, experiment B is the most biological plausible in the sense that it is unlikely that the real salamander has two separate control circuitrs for swimming and trotting which only interact at the level of the motoneurons as in experiment A. That experiment was, however, useful for investigating whether interconnected oscillators which are copies of the swimming segmental networks could be used for producing trotting gaits whose speed and direction could be modulated by simple input signals.

The neural configurations of experiment B have the following similarities with the organisation proposed in [Delvolvé *et al.* 97] (see Figure 2.5, chapter 2): they are made of two coupled limb oscillators which project to a lamprey-like body CPG, and which, during trotting, provide phasic inhibition and excitation to different parts of the body segments, forcing the anterior and posterior parts of the lamprey-like body CPG to oscillate in antiphase. The differences are that the neural configurations of experiment B do not have body segments representing the neck of the salamander, and that the posterior limb oscillator projects to all segments of the tail, while in [Delvolvé *et al.* 97] they project only to the most caudal segment.

In [Ermentrout & Kopell 94b], the production of S-shaped standing waves is investigated in a chain of coupled mathematical oscillators with both short range and long range couplings. The oscillators are coupled with closest neighbour couplings which tend to make oscillators oscillate in synchrony, and with long range couplings between oscillators 1 and $m + 1$ and between oscillators m and $2m$ in a $2m$-long chain which tend to make these coupled oscillators oscillate in antiphase. It is found that for a range of strengths of the long range inhibitory coupling, a S-shaped standing wave (i.e. all segments in each half oscillating in synchrony and both halves oscillating out of phase) is a stable solution. In particular, the S-wave is stabilized by large values of inward (i.e. from the extremity to the middle of the chain) coupling. That type of configuration is different from the neural configurations of experiment B in the sense that it does not have distinct limb oscillators which project unilaterally to a chain of oscillators and that only single oscillators of the chain receive long couplings instead of all oscillators of the anterior and posterior parts in the controllers I evolve. The chain of oscillators of [Ermentrout & Kopell 94b] and the evolved controllers are, however, similar in that, in both cases, a chain of oscillators, which would oscillate in synchrony if undisturbed, is forced to oscillate in antiphase above and below its middle.

Note that one may ask why I have used two limb oscillators for the trotting gait while one oscillator with antisymmetric projections to the hind- and forelimbs would in principle be enough for producing a trotting gait. There are three reasons for this: firstly, I am interested in a controller which is biologically plausible (in nature, different limbs are usually controlled by different oscillators); secondly, the two coupled oscillators provide a richer dynamics than a single one; and, finally, having two oscillators makes the control system more robust against lesions.

7.4.3 Neural activity

Producing the trotting gaits with an S-shaped undulation is easier with the controllers in experiment A than B, because the limb oscillators of experiment A project directly to the trunk motoneurons and the trotting pattern can be obtained without activating the body CPG. In experiment B, both the body and the limb CPGs are active during trotting and the limb CPGs have to force the trunk CPG to oscillate out of phase.

The neural activity produced by the evolved controllers of experiment B presents several similarities with the EMG recordings reported in [Frolich & Biewener 92, Delvolvé *et al.* 97]. During trotting, most segments of the trunk (all segments from 1 to 40 in all controllers) oscillate in synchrony as observed in [Frolich & Biewener 92, Delvolvé *et al.* 97]. There is also a coordination between trunk motoneurons and limb motoneurons very similar to that measured in [Delvolvé *et al.* 97], with the trunk motoneurons being active just prior to and during the activation of the ipsilateral forelimb flexor and the ipsilateral hindimb extensor. The evolved controllers do not, however, produce the double burst pattern observed in the tail of the salamander, with the first burst traveling in the rostral direction and the second burst traveling in the caudal direction.[9] Also, although the evolved controllers and the real salamander both produce a traveling neural wave along the body during swimming, the evolved controllers pro-

[9] Note, however, that in the area around segment 80 in controller B9 (Figure 7.9, right), the shape of the motoneuron signals are more complex than signals with a clear single burst per cycle, with the signals exhibiting two maxima at each cycle.

duce a lamprey-like wave with a constant phase lag between segments while in the real salamander three distinct waves with slightly different phase lags between segments have been observed [Delvolvé *et al.* 97].

7.4.4 Mechanical simulation

The mechanical simulation is only a crude description of the body of the salamander. Firstly, limbs are simulated as rigid links attached to the body links by one-degree of freedom joints, instead of the three-jointed limbs of the real salamander. This simplification restricts the possible motions of the body — for instance, having two opposite limbs on the ground means that the angle, and therefore the degree of freedom, between the limbs is fixed and that any rotation of the limbs has to be executed "against" the friction forces representing the contact on the ground. The rigid limbs also lead to important tail and trunk movements as a limb in stance phase forces the body links to which it is attached to move following a circular arc. Finally, having rigid limbs also means that the control of the limb is greatly simplified as only two muscles have to be controlled, compared to all the muscles actuating the three joints of the real salamander's limb. A second important simplification of the simulation is that only accelerations in the horizontal plane are calculated, which means that the vertical motion of the limbs is not calculated and that equilibrium problems are not considered. This simplification implied, in particular, that the contact of a limb on the ground and the forces which result from that contact had to be abstracted in the present simulation.

Despite these simplifications, the present mechanical simulation provided an interesting model for evaluating the neural controllers. The main interest was to study the control of body movements and the production of traveling and standing waves for the two locomotion gaits of the salamander. Having a model of the body, in particular, forced the limb oscillators and motoneurons to be well coordinated with the body segmental networks and motoneurons, for the production of an efficient trotting gait. The inherent dynamics of the body had to be taken into account in the coordination, and this, for instance, led the evolved controllers to have a very similar timing of the motoneuron signals of the trunk and the limbs to that observed in the real salamander. The mechanical simulation also allowed the study of how the speed and the direction of the mechanical body could be modulated by the neural controllers, which would be difficult to assess by only analysing neural activity. Finally, although the limbs are simplified in the simulation and would need a much more complex control in a more realistic 3D model, I could imagine that extending the controller to a more realistic model could be done by coordinating oscillators for the different muscles of a limb with the signals of the corresponding principal limb oscillators that I have evolved (see for instance [Ashley-Ross 94a], for a detailed analysis of hindlimb kinematics).

Note that with this mechanical simulation which includes several joints to represent the body of the salamander, I did not encounter the problem Lewis had with his robot GEO with a one-joint spine that forward progression could only be obtained with a traveling activity in the spine controller [Lewis 96].

7.5 Summary

This chapter presented how a CPG for the swimming and the trotting of a simulated salamander could be developed from the swimming circuitry of the lamprey. The evolution of the locomotion controllers for the salamander-like animat can be seen as a fourth evolutionary stage following the evolution of swimming controllers in chapter 4.

Two experiments are carried out in which locomotion controllers are evolved which are composed of two coupled limb oscillators and of a lamprey-like body CPG. In both cases, the limb oscillators are copies of the segmental networks of the lamprey-like body CPG. The experiments differ in the organisation of the controllers, with experiment A in which the limb and body CPGs have a "parallel" organisation and experiment B in which they have a "hierarchical" organisation with a unilateral coupling from the limb CPG to the body CPG.

In both experiments, the best evolved controllers exhibit a trotting gait very similar to that observed in the real salamander, with the body making an S-shaped standing wave which is well coordinated with the limb movements. The controllers can switch from the traveling swimming gait to the trotting gait depending on how the excitation is applied to the different parts of the controllers. The speed of trotting can also be modulated with the level of excitation and turning can be induced when asymmetrical inputs are applied.

The controllers and their neural activity are compared with EMG recordings of real salamanders and with the neural organisation proposed by neurobiologists. It is found that, although there are several features of the EMG recordings which are not displayed by the evolved controllers, there are several similarities which suggest that the evolved controllers, especially in experiment B, may not be too far from the neural organisation of the real salamander.

Chapter 8

Towards a complete animat

This chapter presents a preliminary experiment in which the capacity to sense their environment is given to the simulated lamprey and salamander. This can be seen as a first step towards the development of a complete animat capable of sensing and behaving in an environment.

A very simple visual system is added on top of the locomotion CPGs of the simulated lamprey and salamander enabling them to localise and to track a randomly moving target. The visual system is composed of two retinae which compute the bearing of the target and which determine the command signals sent to the locomotion CPGs. The simple control mechanism is hand-coded to perform tracking behaviour, and no evolution is therefore performed in this chapter.

Rather than aiming to simulate a realistic visual system, the main motivation of this experiment is to investigate how the evolved CPGs cope with continuously changing commands. The experiments presented so far were realised with very simple commands signals sent to the CPGs: commands were either constant, or piecewise constant for the turning sequences. Here, the tracking of a randomly moving object leads to continously changing inputs to the CPGs. The experiment will therefore investigate how "usable" the CPGs are by higher control centres for behaviours which require constant variations of the motor output.

8.1 Methods

8.1.1 Simple visual system

A very simple visual system is provided for the simulated lamprey and salamander, enabling them to localise a target in their 2D environment (Figure 8.1). The visual system computes the angle α between the position of the target and the direction of heading of the simulated animat. Based on this bearing, the outputs of two "retinae", R_l and R_r, are calculated as follows:

$$\left.\begin{array}{lcl} R_l & = & \alpha/max\_angle \\ R_r & = & 0.0 \end{array}\right\} \text{ if } \alpha \in [0, max\_angle]$$
$$\left.\begin{array}{lcl} R_l & = & 0.0 \\ R_r & = & |\alpha|/max\_angle \end{array}\right\} \text{ if } \alpha \in [-max\_angle, 0] \qquad (8.1)$$
$$\left.\begin{array}{lcl} R_l & = & 0.0 \\ R_r & = & 0.0 \end{array}\right\} \text{ otherwise}$$

The output of a retina therefore varies linearly between 0.0 and 1.0 when the target is on the corresponding side of the body and when the angle $|\alpha|$ varies between 0.0 and *max_angle*, a positive number determining the opening angle of the visual field. If $|\alpha|$ is larger than *max_angle*, the output of the retina is nil, which means that the simulated animats have a dead angle in which they can not see the target from either retina. In the experiments presented here, *max_angle* was set to 150^o. Note that, in this very simple model, the output of the retinae does not depend on the distance of the target.

The outputs of the retinae are used to determine the tonic excitation applied to the locomotion CPG. Each retina projects to the input nodes of the CPG through a ispilateral excitatory connection and through a contralateral inhibitory connection (Figure 8.1). The excitatory and inhibitory connections have the same synaptic weights, which are fixed by hand depending on the locomotor CPG. This leads to a simple tracking behaviour similar to that exhibited by one of the vehicles Braitenberg imagined [Braitenberg 84], except that commands are here sent to CPGs for animal-like locomotion, instead of the powered wheels of a robot.

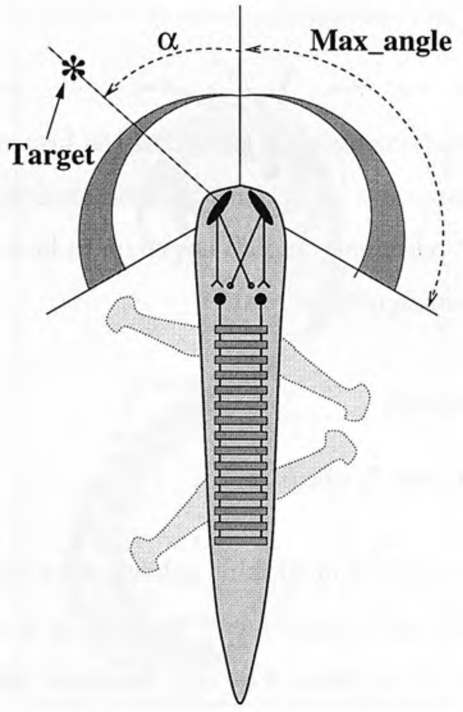


Figure 8.1: Simple visual system for the simulated lamprey and salamander. The simulated retinae have crossed inhibitory connections (filled circles) and ipsilateral excitatory connections ("V"s) to the locomotion CPG.

8.1.2 Target

The target is made to move randomly. At each integration step of the neural simulation, the target makes a step of random length, with a random variation of direction compared to the previous step:

$$\begin{aligned} step\_length &= min\_length + var\_length \cdot rand \qquad (8.2) \\ new\_angle &= old\_angle + var\_angle \cdot (rand - 0.5) \qquad (8.3) \end{aligned}$$

where *rand* is a uniformly distributed random number $\in [0.0, 1.0]$.

The average step size is set in each experiment so that the average instantaneous speed of the target corresponds approximately to that of the simulated animat.

8.2 Results

I carried out tracking experiments with several of the best controllers evolved in this thesis, both with the lamprey's and with the salamander's simulation. The results presented here are only preliminary and do not claim to be a quantitative study of the effect of varying command signals.

8.2.1 Tracking lamprey

Controller evolved in run 7, chapter 4

Figure 8.2 shows a short sequence of tracking behaviour by the simulated lamprey, with the controller evolved in run 7, chapter 4. The lamprey is able to follow the target, keeping close to it most of the time. The only problems which some times arise are when the target is allowed large variations of direction, which leads the lamprey to be temporarily distanced from the target when the target makes a sharp turn. The lamprey is then disadvantaged by the fact that it can not turn on the spot and because of its inertia in the water.[1]

The commands sent by the simulated retinae are continuously changing over time (Figure 8.3). A first cause of variation is the random movement of the target compared to the lamprey which leads to important variations of the amplitude of the signals sent to both sides of the CPG. Changes are especially important when the lamprey crosses the path of the target and the target passes from one visual field to the other behind the head of the lamprey. A second cause of variation is the periodic movements of the head which lead to constant small periodic variations of the amplitudes of the signals.

Despite these important variations of command signals, the CPG maintains a quasi-periodic activity in all segments, with phase lags between each segment leading to the traveling wave necessary for progression (Figure 8.4). The main effect of the varying commands is on the amplitude and burst duration of the interneurons, which leads to variations of the amplitude and burst durations of the motoneurons (Figure 8.4, right). When commands are sent to turn to one side, the amplitude and burst duration of the

[1] The random movements of the target are not physically based, and therefore not subjected to any forces by the water (unlike the simulated lamprey).

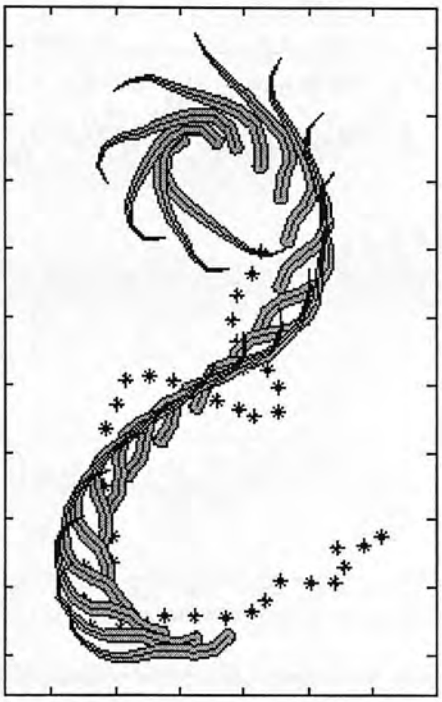


Figure 8.2: Lamprey tracking the randomly moving target (controller evolved in run 7, chapter 4).

motoneurons on that side increase, while they decrease on the other side.

Crossing the speed barrier with controller of run 7, chapter 4

In this section, I make a simple experiment to test whether vision and active correction of commands can help crossing a speed barrier. In chapter 4, sensory feeback from stretch sensitive cells was evolved which enabled the lamprey to cross a speed barrier which could not be crossed without sensory feedback. That experiment was carried out with fixed commands for swimming straight, and I mentioned in the discussion of that chapter that further experiments should be carried out for determining the respective importance of sensory feedback and active corrections from higher control centres for crossing the barrier.

Here, I make a first step in that direction by having the lamprey tracking a fixed target situated behind the speed barrier. The visual system provides commands for keeping the direction of swimming towards the target, therefore correcting the effect of the speed barrier which tends to force the lamprey to change direction.

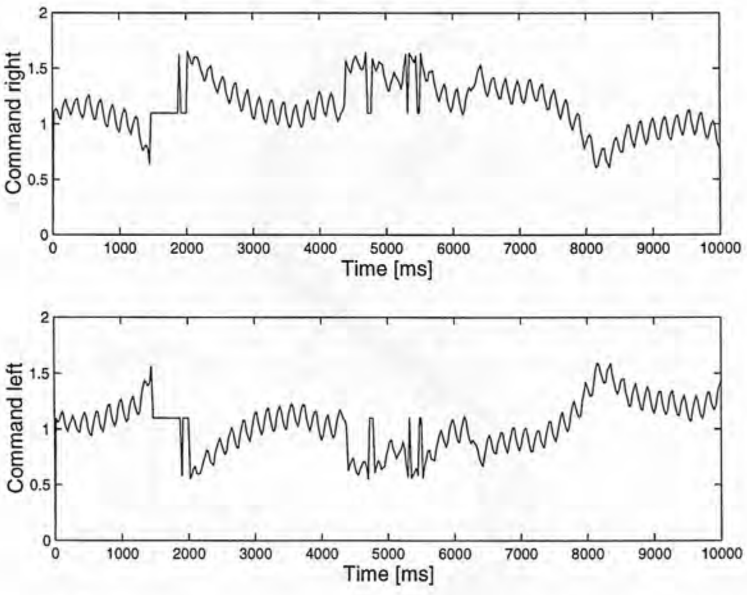


Figure 8.3: Varying commands (controller evolved in run 7, chapter 4).

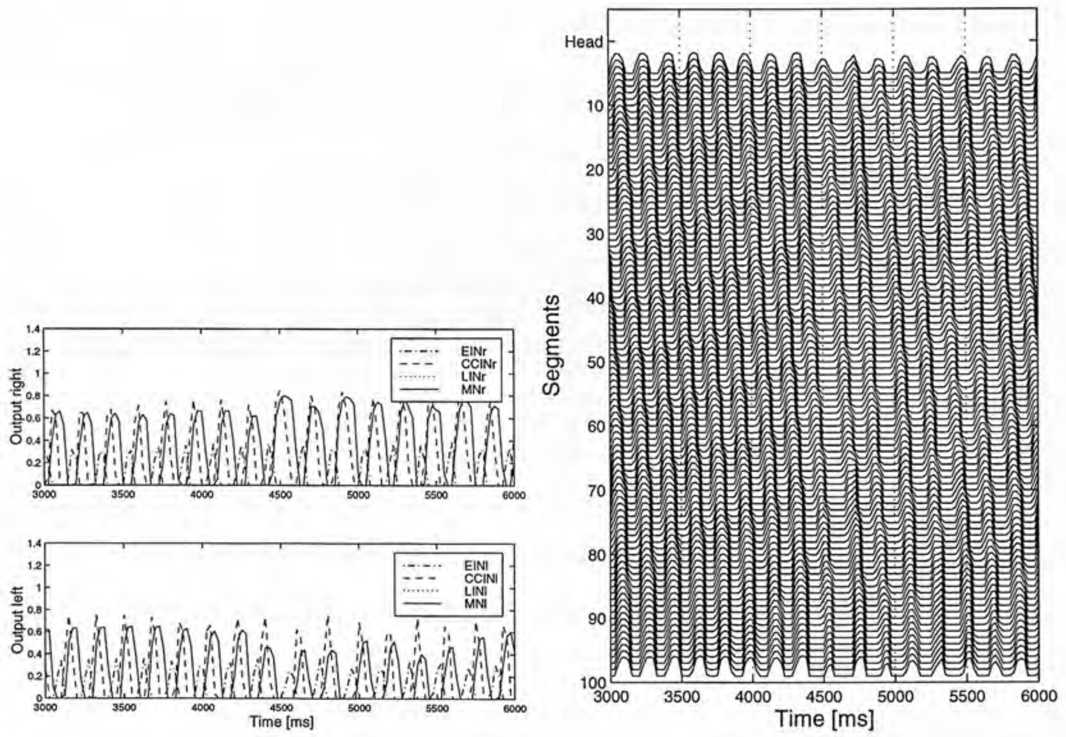


Figure 8.4: Effect of the varying commands on the neural activity (controller evolved in run 7, chapter 4). *Left:* Neural activity in segment 50, *right:* neural activity of the left motoneurons over the spinal cord.

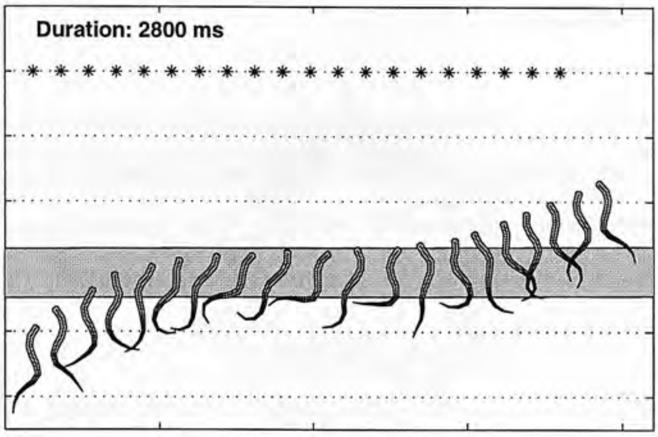


Figure 8.5: Crossing the speed barrier while tracking a fixed target.

The experiment is realised in the same way as that for the evolution of sensory feedback: same controller, same average excitation level (there are now variations due to the retinal signals) and same speed barrier. The speed barrier is half the length of the lamprey and I carried out several tests with different speeds of water in the barrier. Without sensory feedback and without corrections in the commands (i.e. straight swimming commands), the lamprey can cross a speed barrier whose speed is up to approximately 90% of the lamprey's swimming speed. With the tracking behaviour, that limit is increased to 95%, and tracking can therefore help the lamprey to cross speed barrier it would not have crossed without. The effect of the visual system is to correct the direction of swimming at the critical stage when it starts to change significantly because of the speed barrier. This is illustrated in Figure 8.5, where the visual system leads to a change of direction in the middle of the sequence shown. Without that correction, the lamprey would have been pushed to the right and finally repelled by the barrier.

Note that with the evolved sensory feedback, it was found that barriers with speeds up to 140% could be crossed (chapter 4). Sensory feedback from the stretch sensitive cells therefore enables crossing speed barriers with significantly larger speeds, and this simple experiment thus tends to show that sensory feedback plays a crucial role in the capacity to swim in non-stationary water (see discussion). I also tested the combination of sensory feedback and the visual system but did not manage to have the lamprey cross barriers with speeds larger than 140% of the lamprey's swimming speed.

Controller 10, chapter 6

I repeated the experiment of tracking a randomly moving target with the best controller evolved with the developmental encoding (controller 10, chapter 6), and found a similar result to the corresponding experiment with the other evolved controller. The visual system emits very similar varying commands which do not affect the CPG's capacity to produce swimming patterns.

This controller can perform sharp turns when it receives strong asymmetrical input (see chapter 6). It then completely bends the body of the lamprey in the direction of the turn, which leads to an important change of direction when the lamprey has enough initial speed. However this can lead to problems when the strong asymmetry lasts too long, as this leads the lamprey to come to a standstill. In some of the tests carried out, the lamprey then stayed in "lethargy" for a short period until the target moved sufficiently to reduce the strong asymmetry.

This problem is easily solved by reducing the maximum asymmetry that the commands coming from the visual system can produce (i.e. by reducing the weights between the retinae and the CPG input nodes). This means, however, that the capacity to turn is reduced as well. Another possibility would be to use a more sophisticated control mechanism which ensures that strong asymmetries of input applied to the CPG are limited in time.

8.2.2 Tracking salamander

Controller B9, chapter 7

The CPGs evolved for the salamander are also able to sustain a tracking behaviour when connected to the visual system. I tested the tracking behaviour during trotting with the controller B9, the best controller of experiment B. In this case, the command signals from the retinae are sent to both the limb and the body CPGs (as was the case for the evaluation of the turning capacity).

The trotting gait produced by the controller B9 exhibits large head movements from left to right due to the strong bending of the trunk. During tracking, this leads to

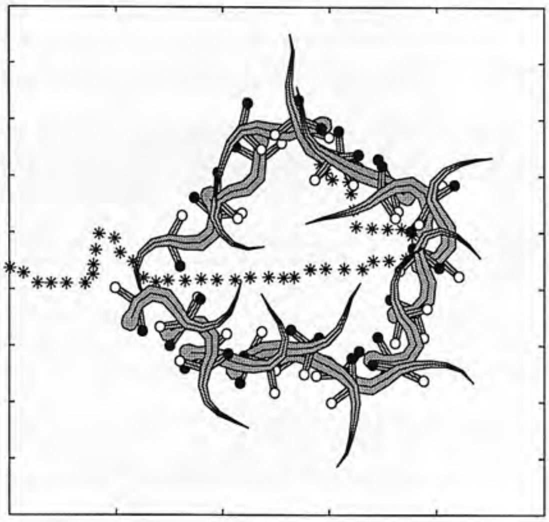


Figure 8.6: Salamander tracking the randomly moving target (controller B9).

large periodic variations of the command signals in addition to the variations due to the movements of the target (Figure 8.7). In comparison to the experiments with the controllers for the lamprey, the amplitude of the variations due to the head movements are significantly larger, therefore making the signal indicating the position of the target less distinct. Note that for this controller, only small asymmetries of commands are necessary for inducing turning, and the maximum amplitudes of variation are therefore less important than those of Figure 8.3 for the lamprey experiment.

Despite these strongly variable commands, the salamander is able to track the target relatively well (Figure 8.6). As movements by the target are partially masked by movements of the head, the salamander changes direction only when the target makes significant changes of direction. The tracking is therefore a little bit less accurate than with the lamprey's controllers. Movements of the tail also lead to a progression which is less smooth than for the lamprey. Important changes of direction are usually accompanied by important tail movements which tend to continue for a while after the turn, preventing the salamander from making a straight progression. This effect is mainly due to the fact mentioned in chapter 7 that the limbs are rigid links which therefore force the body to make circular movements.

Similarly to the previous experiments, varying commands do not significantly affect the oscillations in the limb oscillators and in the body CPG (Figure8.8).

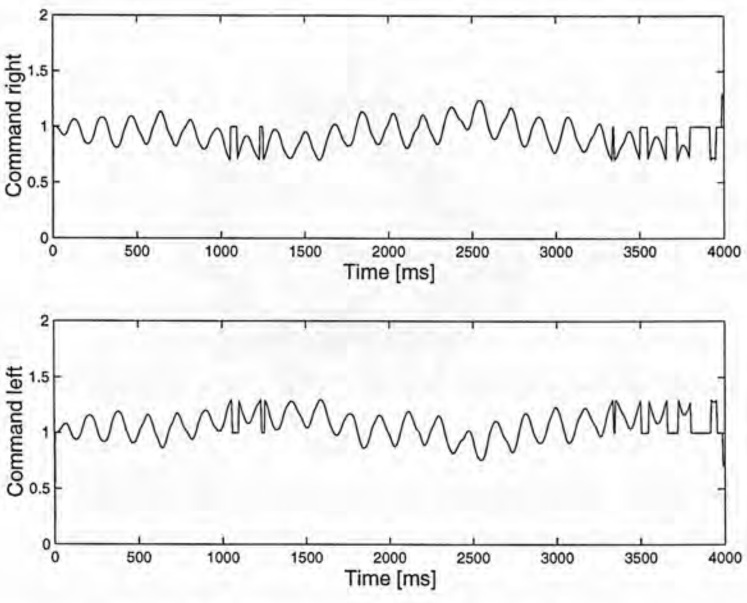


Figure 8.7: Varying commands (controller B9).

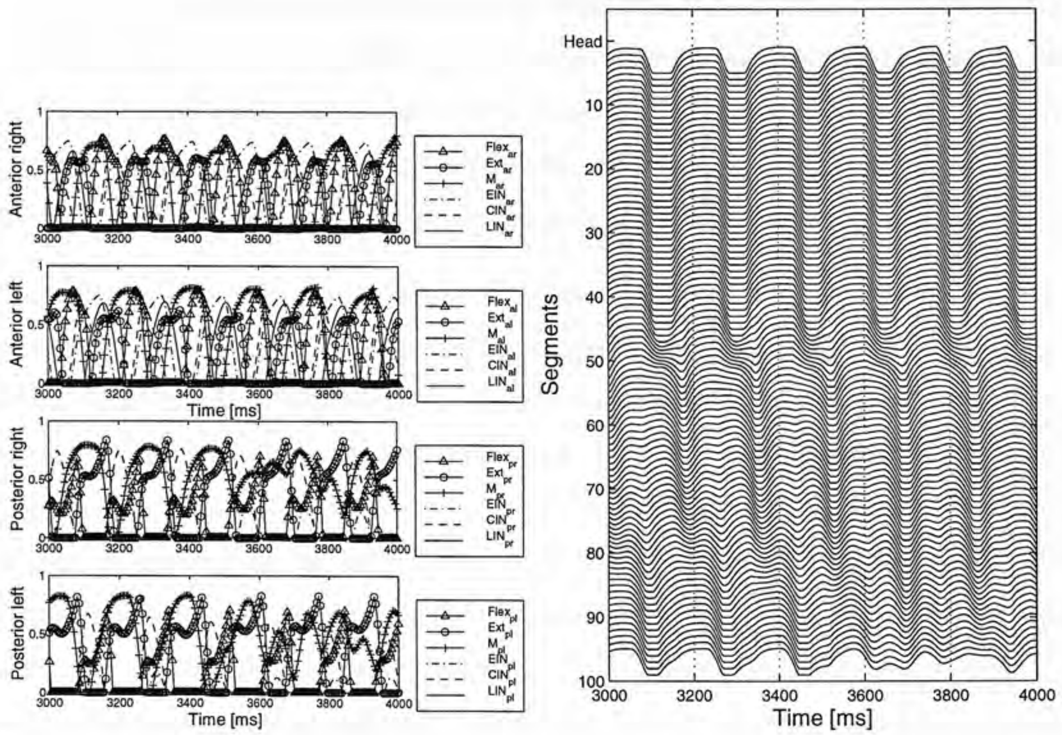


Figure 8.8: Effect of the varying commands on the neural activity (controller B9). *Left:* Neural activity in the limb oscillators, *right:* neural activity of the left motoneurons over the spinal cord.

8.3 Discussion

The results showed that all controllers tested were able to perform tracking behaviour when provided with a very simple visual system. The visual system is made of two retinae which compute the angle of a randomly moving object and which determine the signals sent to the locomotion CPGs through four hand-coded connections.

No parameters are evolved in this preliminary experiment, as a satisfactory control mechanism could easily be developed by hand. In future experiments the tracking behaviour could probably be improved by optimising both the function which computes the output of the retinae given the position of the target, and the connection parameters from the retinae to the CPGs.

The main motivation of the experiment was to investigate the effect of continously changing commands on the pattern generation of the CPGs evolved in the previous chapters. Analysis of the turning capacities of the CPGs had already demonstrated that they were able to cope with abrupt changes of commands followed by long periods of constant commands. Here it is shown that the CPGs can perfectly well cope with the constant variations of input due to the tracking task. These variations have a periodic component which corresponds to the movements of the head and a random component which corresponds to the random motion of the target. The effect of the variable commands on the CPGs is mainly to change the amplitude and the burst duration of the neurons, and in particular of the motoneurons. In all experiments, the simulated lamprey and salamander never stopped completely — except when a too large asymmetry of input is applied to the CPG, in which case the movements of the simulated animat stop until the the asymmetry is reduced. These cases can easily be avoided by either limiting the maximal asymmetry of input which can be applied to the CPG or having a mechanism for ensuring that the strong asymmetry is short.

The visual system also allowed testing whether active commands could help the lamprey to cross a speed barrier as discussed in chapter 4, when tracking a fixed target. It is found that, while active corrections in the commands are not as powerful compared to sensory feedback from stretch sensitive cells for enabling the crossing of a speed barrier, they enable the crossing of barriers which would not be crossed with constant straight swimming commands. Command corrections primarily solve the problem of change of direction induced by the barrier, while sensory feedback has a more local effect (i.e. at the level of each segmental oscillator) which prevents excessive bending and ensures good coordination between the neural activity and the actual movements of the body. Active commands and sensory feedback from stretch-sensitive cells can therefore be seen to have a complementary effect (although, in this simple experiment, the addition of active commands to sensory feedback did not improve the crossing capacity compared to sensory feedback alone).

The simulated salamander with controller B9 performed a less accurate tracking of the randomly moving object than the lamprey because of important head movements. As observed in chapter 7, the real salamander makes head movements of significantly smaller amplitude, firstly because it has neck muscles which contract in antiphase compared to the trunk muscles, therefore counteracting the movements of the trunk, and secondly because its limbs are not rigid like those of the simulation and do not force the body to follow circular arcs. Improvements of the mechanical simulation such as simulating more links at the level of the neck and simulating the limbs as being articulated by several joints should reduce the head movements in the simulation, especially if the controllers are extended such that the neck muscles are contracted out of phase compared to the trunk muscles. Improvements of the mechanical simulation of the salamander will be further discussed in the next chapter.

The visual system in this experiment was very simple. In particular, it did not compute the distance of the target and did not influence the speed of locomotion.[2] Further experiments should both improve the visual system and the control centre transforming the sensory information into motor signals. Of particular interest for the first point are the models of the visual system of the salamander developed by Eurich. In the first model, *Simulander I*, a feedforward neural network is developed representing tectal neurons, brainstem interneurons and neck motoneurons which is able to simulate the orientation movements of the head toward a prey [Eurich *et al.* 95]. The second model, *Simulander II*, simulates the control of the projectile tongue of some salamanders and performs accurate depth perception using binocular neurons in the optic tectum (i.e.

[2] While the excitation of the left and right sides of the CPGs varied, the total excitation level, i.e. the sum of left and right excitations, was constant, leading therefore to a more or less constant frequency of oscillations and speed of motion.

neurons which receive direct input from both retinae) [Eurich *et al.* 97]. It would be most interesting to combine these biological models with the locomotion CPGs developed in this thesis.

8.4 Summary

This chapter presented a simple, preliminary, experiment in which several of the evolved controllers for the lamprey and the salamander were connected to a simple visual system for tracking a randomly moving target. The visual system is made of two retinae which compute the angle between the target and the heading of the simulated animat, and which determine the commands sent to the locomotion CPGs. The simple system is hand-coded rather than evolved.

A successful tracking behaviour was obtained for the controllers tested, both for the lamprey and for the (trotting) salamander. The tracking of the salamander was a little bit less accurate because of the substantial head movements during trotting induced by the controller tested.

The simple visual system also allowed a preliminary test to be made on the respective importance of active commands and sensory feedback from stretch sensitive cells for crossing a speed barrier. The first results tend to show that, while sensory feedback plays a crucial role for crossing the barrier in keeping a good coordination between neural activity and the movements of the body, active commands can also help, but to a lesser extent, by preventing important changes of direction due to the barrier.

The main point demonstrated by this experiment is that the evolved CPGs can cope with continuously changing command inputs. The commands vary both because of the random movements of the target and because of the periodic movements of the head. These variations do not affect significantly the pattern generation of the evolved CPGs, their main effect being to change the amplitude and the burst durations of neurons on opposite sides of the CPG.

This simple experiment illustrates how the evolved CPGs can be used by a higher control centre for a simple behaviour. The fact that no problem was encountered demonstrates that the locomotion CPGs are sufficiently robust to be integrated in more complex control systems, for the development of animats with more complex behaviours than simple tracking.

Chapter 9

Discussion

The following questions initiated this thesis:

- How can neural networks be used to control locomotion, and what can we learn from the neural circuitry found in vertebrates?
- How efficient are evolutionary algorithms for the design of neural controllers?
- Can evolutionary algorithms be used as design tools in neurobiology?
- What kinds of neural circuitry can produce the undulatory swimming of lampreys? In particular, are there alternative neural configurations to those found in Nature which can control locomotion with at least the same efficiency?
- How can controllers for undulatory swimming be extended to control both the swimming and the trotting of a salamander-like animat?

The next sections summarize the main findings of this thesis and how the thesis contributes to answering these questions. The results are also placed in perspective compared to related work.

9.1 Neural networks for the control of locomotion

Animal-like locomotion is characterised by a large number of actuators, a rhythmic activity and the fact that locomotion is only obtained when the different actuators are appropriately coordinated. The problem of controlling such locomotion is therefore to be able to transform simple commands concerning the direction and the speed of motion into the multiple signals sent to all the actuators. In chapter 2, I reviewed findings about biological control, with a special emphasis on the control of undulatory locomotion.

9.1.1 Biological control

The review of neurobiological findings shows that animal locomotion is controlled by networks of neurons organised in a distributed way. In most animals, the patterns of neural activity underlying the motion are produced centrally, rather than peripherally; in other words, sensory feedback is not required for the production of the motor sequences.

In vertebrates and in many invertebrates, the oscillatory signals are generated in circuits called central pattern generators. These circuits can produce patterns of oscillations without oscillatory input, either from higher control centres or from sensory feedback. The control of locomotion is distributed and signals sent to the different muscles are not generated in the brain, but in lower centres of the central nervous system; locomotion CPGs in vertebrates are, for instance, located in the spinal cord. Signals traveling from the brain to the central pattern generators encode higher level information (about the speed of motion, for example) rather than the multiple muscle signals. In the lamprey, simple tonic drive applied to spinal cord is sufficient to initiate swimming, and variation of the amplitude of the drive leads to variation of the speed of motion.

The central pattern generators, themselves, are distributed and are systems of coupled neural oscillators. In the lamprey, different body segments have neural oscillators which can be made to oscillate independently of the others. Similarly, the control of limbed locomotion is organised into different neural oscillators controlling the muscles of different joints [Grillner *et al.* 88].

Finally, central pattern generators can produce significantly different gaits depending on their input signals. The same neurons can then be involved in the generation of the different neural patterns, which means that the patterns are a consequence of a distributed computation over the whole CPG.

From a control point of view, CPGs can be seen as providing templates for the rhythmic movements necessary for locomotion which can be modulated, when necessary, by signals from higher control centres or from sensory feedback.

9.1.2 Artificial neural control

In this thesis, the controllers evolved for the locomotion of two mechanically simulated animats are strongly inspired by biological neural controllers: they are composed of abstract neurons simulated at a connectionist level and they are organised similarly to central pattern generators.

Other representations than neural networks could have been used to implement controllers, and locomotion controllers have for instance been implemented as explicit algorithmic controllers [Raibert & Hodgins 93, Terzopoulos *et al.* 94], finite state machines [Brooks 89] or classifiers [Bull *et al.* 95]. In particular, the anguiliform swimming of the lamprey could be generated by algorithmic controllers producing sinusoidal waves, with a few parameters determining the frequency, the wavelength and the amplitudes of the signals sent to the muscles, as illustrated for testing the mechanical simulation of the lamprey (section 3.2.1). Algorithmic controllers for swimming have, for instance, been evolved in [Ventrella 98, Usami *et al.* 98]. Compared to neural-based controllers, algorithmic controllers present the advantages of being easier to design (at least for simple control dynamics, see below), of providing an explicit control mechanism (parameters such as frequencies and wavelengths are explicitly represented), and, depending on how the connectionist model is implemented, of being computationnaly faster (there is no integration of a system of coupled differential equations, for instance).

The use, in this thesis, of connectionist models rather than algorithmic (or other) controllers, was motivated by several reasons. The main motivation was to gain better insight into the functioning of biological controllers, by designing biologically plausible controllers and therefore visiting the space of possible neural configurations for animal-like locomotion. Other motivations are linked to properties of biological neural networks that connectionist models reproduce and which may be useful in locomotion control. These properties include distributed control, rich dynamics, and robustness against noise.

Distribution over a network of neurons organised into CPGs which are themselves organised as systems of coupled oscillators is an interesting way to organise control. This distributed control is characterized by fast parallel computation, input and outputs at different levels with, for instance, integration of sensory feedback for fast reactions, modularity (with limb CPGs, joint oscillators,...), and a good robustness against lesions. Although my simulations were run on a single processor, and, therefore, there is no physical parallel computation nor robustness against (physical) lesions, these properties could potentially be reproduced in a hardware implementation. Central pattern generators similar to that of the lamprey have, for instance, been implemented into analog VLSI [DeWeerth *et al.* 97, Patel *et al.* 98].

Networks of neurons, both biological or simulated at a connectionist level, exhibit very rich dynamics (continous-time recurrent networks are universal dynamics approximators [Funahashi & Nakamura 93]). When the controller is built to take advantage of this dynamics, it could prove to be well adapted for controlling a mechanical body and producing the changes in the multiple signals sent to the muscles necessary for changing speed, direction or even gait (similarly to biological CPGs). In this thesis, for instance, neural controllers capable of such a modulation are successfully evolved. Very rich dynamics may be difficult to produce with algorithmic controllers without their becoming too complex.[1]

Similar motivations have led other researchers to develop neural controllers for locomotion [Beer 90, Quinn & Espenschied 93, Cruse *et al.* 95, Gruau 95, Kodjabachian & Meyer 98a]. Most work has been done on the control of legged locomotion. To the best of my knowledge, Ekeberg's model was the first example of swimming controlled by neural circuitry, and this thesis is the first example of the artifical evolution of neural controllers for swimming.[2] It is also the first example of

[1] Note that a genetic programming approach may be a potential method for evolving complex dynamics at an algorithmic level. Spencer, for instance, used genetic programming for evolving locomotion controllers for a simulated hexapod insect [Spencer 94] (see also section 2.6.2).

[2] Sims also evolved controllers for swimming, with controllers made of so-called neurons [Sims 94b], but his neurons have little in common with biological neurons and compute functions such as sum, product, min, max, sinus, cosinus,...

development of neural controllers for both an aquatic and a terrestrial gait.

Note that this thesis does not claim that artificial locomotion control should be neural based, as such a claim would require a detailed quantitative comparison between neural-based and other types of controllers. My primary aim is to investigate the neural control of locomotion and to illustrate how neural controllers can be used for controlling the locomotion of two simple animats. The results demonstrate that effective neural-based control can be achieved.

9.2 Evolutionary algorithms for designing neural controllers

Evolutionary algorithms present several interesting features for the design of connectionist models compared to traditional learning algorithms. Learning algorithms for dynamical neural networks such as variations of the backpropagation algorithm, (see [Pearlmutter 95], for instance), iteratively update synaptic weights given an error function which calculates the difference between the current and the desired state trajectories of the network. These algorithms compute the gradient of the error function in terms of the synaptic weights and update the weights accordingly (gradient search — see section 2.5.1).

Evolutionary algorithms have in comparison several advantages, at the only cost of being significantly slower — being a stochastic population-based search method, they require many evaluations. Firstly, the fitness function does not need to be differentiable or even continuous. Secondly, there is no need to provide a specific oscillation (limit cycle) that the network should learn. Learning algorithms are restricted to problems in which the desired state trajectory is known in advance, which is not the case for many control problems, especially in controlling the locomotion of a mechanical system. As the mechanical system — the body — typically has its own complex and non-linear dynamics, there is no (simple) way to determine in advance which command signals leads to the desired mechanical behaviour. In our case, using evolutionary algorithms has allowed us to characterise the desired behaviour of the network at a "higher" level than a specific limit cycle, for instance, in terms of the range of frequencies which can be produced or in terms of the speed of swimming of the mechanical simulation. Also, the parallel and stochastic nature of evolutionary algorithms makes them less likely to be trapped in a local optimum than the gradient search of learning algorithms. Finally, evolutionary algorithms are particularly flexible and the type of evolved controllers can be modified at three levels: the encoding (e.g. by adding or removing constraints on the connectivity), the genetic operators (e.g. adding a pruning operator) or the fitness function (e.g. adding or removing factors). These properties have made evolutionary algorithms a popular method for designing neural controllers for animats [deGaris 90, Beer & Gallagher 92, Gruau 95, Kodjabachian & Meyer 98b, Floreano 98].

9.2.1 Considerations on the evolutionary process

When using evolutionary algorithms for designing neural controllers, several aspects have to be considered:

- whether to use a staged approach,
- the type of encoding (for instance, direct or indirect),
- the definition of the fitness function.
- the type of evolutionary algorithm (GA, ES or GP), and the setting of its search parameters,

Staged or non-staged approach

In this thesis, both staged and non-staged approaches were taken for the evolution of locomotion controllers. Swimming controllers for the lamprey simulation have, for instance, been evolved with a staged approach in chapter 4 and with a non-staged technique (although without sensory feedback) with a developmental encoding in chapter 6.

In the experiment with the staged approach, three stages were used in order to define a neural organisation similar to that of the biological controller. In this case, the fitness functions of the different stages were relatively easy to determine as they rewarded several properties observed in the real lamprey which I wanted the artificial solutions to display as well. The staged approach enabled the relative easy and rapid evolution of efficient swimming controllers composed of between 600 and 800 neurons.

In the experiment in which swimming controllers were evolved in one stage only, the evolutions were in comparison significantly slower, for the generation of controllers with 10 times fewer neurons (although in this case the time constants and the biasses were evolved as well). The task for the evolutionary process was more difficult here as it had to develop neural oscillators and their couplings at the same time. This and small flaws in the fitness function also lead to a lower success rate of the evolutions of controllers capable of producing stable oscillations. Another difference with the previous evolution is that the neurons used here have a less rich inherent dynamics than those developed by Ekeberg (which have properties such as frequency adaptation), which may have made the creation of oscillators with variable frequency more difficult. On the other hand, because there were fewer constraints on the neural organisation, a larger diversity of controllers were evolved (see next sections). An interesting aspect of these evolutions is that there was a single fitness function, based only on mechanical aspects of behaviour. The evolutionary search was therefore not biased by prerequisites on the neural activity of preliminary stages.

From these experiments, the following observations can be made. The advantages of a staged approach are that: 1) it enables incremental design, 2) it reduces the search space when elements of the previous stage are frozen[3] (therefore potentially reducing the evolution time, with the risk, however, of producing sub-optimal solutions, see below), 3) it allows specification of the modular structures of the control systems. The disadvantages are that: 1) elements of one stage are not rewarded for their capacity to be used by the next stage, which may therefore lead to sub-optimal control systems, 2) it requires fitness functions for each evolutionary stage and 3) it requires knowledge of how to divide the evolutionary process into the different stages and how to divide the control system into submodules.

The choice of a staged or non-staged evolution therefore significantly depends on the approach of the experimenter and on the task. Staged evolutions correspond more to an engineering approach, while non-staged evolutions are closer to the Artificial

[3] Note that elements of the previous stage need not necessarily be frozen. For instance, in the evolution of swimming controller with the direct encoding (chapter 4), it would have been possible to continue to evolve the segmental connectivity, rather than freezing it, in the second evolutionary stage where the intersegmental couplings are evolved. This would have increased the search space of the second stage, but with an initial population which is not random.

Life philosophy which studies life-as-it-could-be and which tries to keep evolution as open as possible, typically using "minimal" (or high level) fitness functions such as the capacity of the controlled agent to survive in an environment.

The task is also influential for this choice. Firstly, some problems may be too hard to be tackled monolithically with current computational power. The controllers evolved for the salamander, for instance, which can exhibit both swimming and trotting gaits and can modulate the speed and direction, would be very hard to evolve from scratch in one go. Other aspects of the task which have to be considered are: how much one would like the controllers to satisfy particular constraints, how much intuition one has on the organisation of a good controller, how easy it would be to design fitness functions for intermediate stages.

The general trend in the evolution of neural controllers for locomotion is to use a staged-approach. Controllers for legged walking are often developed by first evolving controllers for a single leg, and then evolving the coupling between different leg controllers [Beer & Gallagher 92, Lewis *et al.* 93, Gruau & Quatramaran 97]. Alternatively, controllers for biped stick legs have been evolved in stages with a different fitness function for each stage, while keeping the same encoding for the whole evolution [deGaris 90]. Gruau developed controllers for six-legged walking in one stage only using his cellular encoding [Gruau 95], but without considering the control of speed and of direction. In a similar experiment with SGOCE, Kodjabachian used an incremental approach with first the evolution of a controller layer capable of producing a walking gait, and then the evolution of a control layer for controlling the speed of motion [Kodjabachian & Meyer 98a], or for controlling the direction such as to be able to perform gradient following [Kodjabachian & Meyer 98b].

Encoding

The type of encoding is an important feature of evolutionary algorithms, as it determines the type of solutions which can be generated and the size of the search space which is visited. Two types of encodings have been used in this thesis, a direct encoding for the evolutions presented in chapters 4, 5, and 7, and the indirect developmental encoding SGOCE, developed by Jérôme Kodjabachian, in chapter 6.

The interesting aspects of a direct encoding are that it is simple to implement and that it requires simple genetic operators whose effects on the chromosomes can be easily assessed. The disadvantage is that the encoding is not compact and that it grows large in size when the number of neurons is increased. In my case, the direct encoding was used in association with a staged approach in order to reduce the size of the search space for each stage.

Indirect encodings such as the developmental encoding of SGOCE encode rules of how the network is grown. This usually leads to more compact encodings as the same rules can be used to develop more than one set of neurons. In the case of SGOCE, compactness is obtained by having all initial cells read the same developmental program. Although this compactness is then not directly due to the encoding but to external symmetries fixed by the experimenter, one could imagine extending the encoding to evolve these symmetries as well (as discussed in chapter 6). An interesting aspect of SGOCE is that the developmental rules are context-dependent — the effect of a rule depends on the geometrical susbtrate. When the evolutionary process takes advantage of this feature, this can lead the single developmental program to generate asymmetrical controllers which are surprisingly well adapted to their task, such as controller 9 in chapter 6. A second interesting aspect of SGOCE, and indirect encodings in general, is that they allow variable topologies of networks, which is harder to do in a direct encoding, except when pruning operators are used. Compared to direct encodings, indirect encodings have usually the disadvantage of needing more complex genetic operators, whose effect on chromosomes (and therefore on the evolutionary process) is more difficult to assess.

Fitness function

The definition of the fitness function is a crucial part of design with evolutionary algorithms. In this thesis, the fitness functions are products of factors varying between 0.05 and 1.0 and have usually been defined incrementally. Each factor corresponds to a desired aspect of the controllers, and I have defined the fitness functions by incrementally adding new factors when the controllers evolved in preliminary testing did not satisfy some of my requirements.[4] The incremental design of the fitness function for initial evolutions of swimming controllers is described in [Ijspeert 96]. The fact that the fitness functions are products rather than sums ensures that solutions which perform equally in all aspects will be preferred over controllers performing well in some aspects but not in others.

In order to obtain locomotion controllers with good control capacities, each evaluation was a set of simulations with different command settings so as to determine how the controllers react to different inputs. The fitness functions are therefore somewhat more complex than those used in most other examples of evolutions of locomotion controllers. Many works indeed concentrate on pattern generation without considering control aspects, such as the control of speed or of the direction [deGaris 90, Beer & Gallagher 92, Lewis *et al.* 93, Gruau 95].

Evolutionary algorithms and search parameters

Once an encoding and a fitness function have been defined, the design problem is transformed into an optimisation problem of the fitness function in the search space determined by the encoding. In this thesis, real number genetic algorithms and genetic programming were used as search algorithms. The search processes themselves have not been optimized and I have not tried to perform the best possible search (fastest convergence, for instance). As long as there was no premature convergence to local maxima with low fitness values and that interesting controllers were generated in a reasonable time, the different search parameters were not changed. More work is therefore required to assess quantitatively the effect of differing search parameters on the search performance. My feeling is that, while faster convergence to good solutions can probably be obtained and that potentially better solutions than those presented here can be generated, such an improvement of the search process would not significantly change the overall results of this thesis. Other genetic operators could also be used, and in particular it could be interesting to use, with the real number genetic algorithm, a more sophisticated mutation operator similar to the one used with evolution

[4] For instance, the factor rewarding solutions keeping one foot on the ground in the evolutions of controllers for the salamander was added after initial tests showed that several evolutions converged to solutions which would not be stable in a 3D simulation.

strategies (see section 2.5.2).

9.2.2 Evolutionary algorithms as a tool for neurobiological modelling

I believe that evolutionary algorithms can prove very useful tools for neurobiological modeling, when used, as shown in a simple example in chapter 5, to instantiate variables which are difficult to measure. Biophysical and connectionist models of a biological circuitry typically require the setting of many variables, few of which are easily measurable in the real circuit. Given a fitness function which characterizes the observed behaviour of the network and an encoding of the network which represents the unknown variables while including available anatomical and functional knowledge, the evolutionary process can be used to instantiate the unknown variables such as the neural activity to fit the observed behaviour of the circuit.

I demonstrated, in particular, that the GA is able to find sets of synaptic weights and interconnections for CPGs when the search space was restricted to solutions presenting the biological segmental connectivity. Ekeberg's segmental oscillator could, for instance, be optimized to exhibit a larger range of frequencies, much closer to that observed in the real lamprey.

Although it is possible, in this case, to define satisfactory values by hand, the GA proved useful both as a synthetic tool for generating and optimizing the synaptic weights of the segmental network, and as an indirect analytic tool for searching through the different possibilities of intersegmental coupling and sensory feedback connections. The fact that the GA can automatically instantiate these variables is especially interesting as setting all these parameters by hand is a hard and time-consuming task — because of strong non-linearities and interdependency between variables — which may become intractable for large circuits.

Several researchers in neurobiology are currently using evolutionary algorithms in a similar way, either for setting synaptic weights of neural network models [Eurich *et al.* 95, Eurich *et al.* 97], or for determining conductance parameters of biophysical models of single neurons [West & Wilcox 97]. From a general point of view, note that, although the GA can be very useful for demonstrating that a model can produce a specific behaviour (by finding efficient sets of unknown variables), it is less useful for invalidating a hypothetical model as an inability to find successful variable instantiations may be due to failings of the model or to problems with the GA set up, or both.

9.3 Neural control of swimming

This thesis considered the evolution of swimming controllers for a simulated lamprey. Three types of evolution were carried out: staged evolution of artificial swimming controllers with a direct encoding, staged evolution of biologically plausible swimming controllers and evolution of swimming controllers with a developmental encoding scheme.

9.3.1 Staged evolution of artificial swimming controllers

The staged evolution of artificial swimming controllers with a direct encoding demonstrated how an evolutionary method could be used to generate efficient swimming controllers. The swimming controllers are composed of neurons similar to those of Ekeberg's model and the evolutionary process is used to determine the sign (inhibitory or excitatory) of the neurons and the connectivity between neurons (which neurons are connected and the synaptic weight of the connections). The controllers were chosen (by the definition of the three evolutionary stages) to share several similarities with the biological controller: to be composed of segmental networks which can be made to oscillate independently from each other; to be able to produce traveling waves of neural activity without sensory feedback for coordination; and to integrate sensory feedback for being able to cross a speed barrier.

The evolutions showed that, within these similarities of organisation, there exists a variety of different neural configurations which can exhibit the same swimming gaits and the same control capacities as the biological controller modelled by Ekeberg. The solutions differ in terms of which neurons are interconnected, of the excitatory or inhibitory influence of the interneurons and even of the number of neurons involved in the rhythmogenesis. Simple signals — the excitation applied to both sides of the spinal cord and the extra excitation given to the most rostral segments — modulate the frequency, the wavelength and the amplitude of the motoneuron activity and can therefore be used to control the speed and direction of swimming. The evolved controllers can be said to control swimming with at least the same efficiency as Ekeberg's biological model. Most evolved controllers cover larger ranges of frequency, phase between segments and speed than Ekeberg's model, but with, in general, less independence in the control of the frequency and the phase between segments.

9.3.2 Staged evolution of biologically plausible controllers

Biologically plausible controllers were evolved with a very similar approach to the staged evolution of artificial controllers, the main difference being that the search was restricted for the evolved controllers to have the segmental connectivity observed in the real lamprey. As discussed in section 9.2.2, the primary point of these evolutions was to demonstrate how evolutionary algorithms could potentially be used for neurobiological modeling.

These evolutions led to several interesting findings. Firstly they showed that the frequency range of Ekeberg's segmental network could be optimized to better correspond to that observed in the real lamprey. The increase of the frequency range is due to stronger weights of some inhibitory connections. It may be interesting to test with paired cellular recordings whether these inhibitory connections are also stronger in the real segmental circuit compared to other inhibitory connections.

The investigation of intersegmental coupling showed that couplings could be evolved which can produce the anguiliform swimming of the lamprey with a similar performance in terms of frequency and speed ranges as Ekeberg's model, while being able to reach higher phase lags. One of these controllers propagates, unlike Ekeberg's model and similarly to the real lamprey, a traveling wave even without extra excitation. The different evolved couplings present several common features which would be interesting to study in further detail and compare with future anatomical findings on the intersegmental coupling of the real lamprey.

Finally, the evolution of sensory feedback connections for crossing a speed barrier led to connections with similar signs to those observed sofar in the real lamprey, suggesting that the crossing of a speed barrier is a good example of the kinds of situation for which sensory feedback has been developed through natural evolution for the real lamprey. In chapter 8, the effect of active corrections in the commands is also tested, and it is found that, although active commands can help the lamprey to cross a speed barrier it would not have crossed without, sensory feedback seems to play a crucial role for crossing barrier with higher water speeds.

9.3.3 Evolution with a developmental encoding

The evolution with the developmental encoding scheme SGOCE investigated the generation of swimming controllers with fewer constraints on the organisation of the swimming controllers. The controllers were, for instance, not forced to be composed of segmental oscillators. Neuron parameters — the time constant and the bias — were evolved rather than preset. To reduce the search space, simpler and fewer neurons (nine neural segments, instead of 100) were used. Another difference with the previous evolutions is that the fitness function was only based on control aspects of the mechanical simulation rather than also depending on the neural activity.

These differences led to the generation of controllers with a larger variety of neural organisations and with a larger variety of swimming gaits produced. The type of swimming gaits included projection forward with an initial thrust, swimming with chaotic neural activity, caranguiform-like swimming (with most of the body rigid and a traveling wave of increasing amplitude in the tail) and anguiliform swimming similar to the real lamprey. The different evolved controllers varied in the number of oscillators (i.e. neural sub-circuits which could be made to oscillate when isolated from the complete circuitry) they were made of, with controllers made of only one oscillator, one oscillator per neural segment or more oscillators. The two controllers which produce stable neural activity and reach the highest speed exhibit an anguiliform swimming gait very similar to that of the real lamprey. Compared to the best controllers of the previous evolutions, the maximum speed these controllers can reach is lower but on the other hand they present a good control of direction, with the capacity to make sharp turns.

Because the fitness function of the second experiment is only based on mechanical aspect, that experiment comes closest to the type of evolution which should be carried out for evolving neural controllers for a real robot. That fitness function depends only on mechanical aspects which could easily be measured from the swimming of a real robot (speed of swimming, changes of direction,...).

As mentioned in chapter 6, one interesting aspect of this developmental encoding is that the developmental rules are context-dependent — the effect of a rule depends on the position of the cell to which that rule is applied. Similarly to biological genetic encoding which relies on the environment and the laws on physics for the developmental process, this context-dependency can lead to the generation of relatively complex asymmetrical neural controllers while staying remarkably compact (as for the controller 9 developed in chapter 6). It would be interesting to extend this work by introducing more context-dependent instructions such as weight- bias- or time-parameter-setting instructions which set the parameter of a cell through a function depending on the cell's position. This could, for instance, introduce variable synaptic weights over the spinal cord (e.g. stronger weights towards the tail for the connections to the muscles) which may lead to even better swimming capacities. In the longer term, it would also be interesting to work towards an encoding which can reproduce the typical stages observed in vertebrate prenatal development. As summarized in [Lewis 96], the following prenatal movements can be distinguished: 1) *uncoordinated head flexions*, 2) *C-bendings*, 3) *S-Bending*, and finally 4) *Traveling S-waves.* This suggests that the locomotor circuitry is progressively constructed, with, for instance, a gradual construction of intrasegmental followed by intersegmental coordination. It has been suggested that an activity-dependent mechanism may be involved in this development for relating sensory feedback and muscular activity. One pattern of movements could, for instance, be the trigger for the next stage in the network's construction. Similarly to Lewis' research but at a connectionist level, it would be most interesting to address similar questions by looking in more detail to the development dynamics which can be obtained with the SGOCE encoding (i.e. analyse the muscular activity at different stages during the decoding of the developmental program) and potentially introducing some kind of activity depending mechanisms for structuring the development into different stages with typical patterns of movements. Ideally, such research should be done with the evolution and development of not only the neural controller but also of the body.

9.4 From swimming to terrestrial locomotion

The evolution of locomotion controllers for the salamander-like animat demonstrated how lamprey-like swimming controllers could be extended to exhibit both the swimming and the trotting gaits observed in the real salamander.

The circuitry underlying locomotion in the real salamander is not decoded yet, and the evolution of potential locomotion controllers can be seen to be one step ahead of neurobiology, compared to the evolution of controllers for the lamprey. The evolved controllers have, however, a neural organisation which is close to the one hypothesised by neurobiologists. The controllers are composed of a body CPG which corresponds to a lamprey-like swimming controller and of a limb CPG made of two further oscillators, one for the forelimbs and one for the hindlimbs. These limb oscillators are copies of the segmental oscillators of the body CPG, and can therefore be seen as segmental oscillators specialised to control the movements of the limbs.

Two experiments are carried out with slightly different neural organisations. In experiment A, a "parallel" organisation is chosen with limb and body CPGs projecting to the same motoneurons but without interaction at the level of the interneurons, while in experiment B, the organisation is more "hierarchical" with the limb oscillators projecting to the interneurons of the body CPG, therefore creating a unilateral coupling between the limb and the body CPGs. This second configuration is very similar to the organisation proposed by neurobiologists [Delvolvé *et al.* 97].

In both experiments, controllers were evolved which could exhibit the swimming or the trotting gait depending on the tonic drive applied to the controllers. The best controllers all present a body-limb coordination very similar to the one observed during trotting in the real salamander, with also similarities between the simulated motoneuron activity and EMG recordings of muscle activation in the real animal. This suggests that the evolved controllers may not be too different from the actual circuitry underlying locomotion in the salamander.

The controllers are developed such that simple input signals — the tonic drive applied to the different subcircuits — modulate the speed, the direction and the type of gait. To the best of my knowledge this is the first example of simulated neural controllers capable of producing both an aquatic and a terrestrial gait.

9.5 Considerations on the mechanical simulations

The mechanical simulations allowed the neural controllers to be tested on their capacity to generate the motion of a body with its own dynamics.

Although the simulations may not be completely realistic (see next sections), they still provided a critical test for the efficiency of the controllers to produce locomotion. In particular, they required an appropriate timing (i.e. appropriate burst durations and phase relations) and amplitudes of the multiple motoneuron signals. Analysing only neural activity does not provide a good evaluation of how good a controller is, as it gives no means of evaluating whether the burst durations, the details of the phase relations, the amplitudes of the motoneurons, ..., are suitable for gait generation.

Having mechanical bodies meant that the controllers had to be adapted to the complex dynamics of the mechanical simulation determined by the elasticity and damping of the muscles, friction forces due to the ground, inertial forces due to water forces, *etc.* In other words, as expressed by Raibert, "rather than issuing commands, the nervous system can only make "suggestions" which are reconciled with the physics of the system and the task" [Raibert & Hodgins 93]. For the salamander-like animat, this meant that a good interlimb coordination was necessary for locomotion and that the coordination of limb and body motoneurons had to take into account the elasticity of the body, which, interestingly, led to a motoneuron coordination very similar to EMG recordings in the real salamander.

Other important features of mechanical simulations include the possibilities: of testing the effects of sensory feedback, of investigating control aspects such as changes of direction or changes of speeds, of making sensori-motor coordination experiments. Finally, because mechanical simulations offer a first approximation of a physical robot, the design method for these simulations may be transferrable to design controllers for a real robot (see next sections).

Note that, as the simulations are simple and by no means realistic biomechanical simulations of animals (or physical robots), the mechanical simulations may have placed wrong constraints on the evolution of neural configurations. The similarities between the neural activity of the simulated swimming controllers and those of the real lamprey as well as those between the motoneuron activity in the simulated salamander-like animat and the EMG recordings in the real animal suggest, however, that the mechanical simulations include some realistic features of a real body. Furthermore, as the design technique has not encountered important problems for generating controllers for these two simulations, it can probably be applied without major problem to the design of controllers for similar but different simulations, for instance, a realistic simulation of swimming robot (see next sections).

Finally, many experiments in the control of locomotion of animats have used kinematic rather than mechanical (or dynamical) simulations of the body. Kinematic simulations, i.e. simulations which do not compute forces but only velocities, usually fail to render the complex dynamics of physically plausible environments and bodies. With the exception of simple gaits, for instance statically stable walking gaits, in which the dynamics of the environment and of the body are not important, there is little chance that these controllers could be transferred to control physical robots without major redesign.

9.6 Further work and perspectives

9.6.1 More realistic mechanical simulations

An important extension of this work is to develop more realistic mechanical simulations. This is necessary both for gaining better understanding of the control systems of real animals, and for coming closer to robotics.

Ekeberg has already extended his 2D lamprey simulation to a 3D simulation [Ekeberg *et al.* 95], with links being attached by spherical (3 degrees of freedom) joints. Torques on the joints are then determined by four muscles arranged symmetrically around the joint. When controlled by a crossed-oscillator network[5] extended from the 2D controller of [Ekeberg 93], coordinated yaw, pitch and roll movements can be produced. As Ekeberg mentions, these simulations are based on important simplifications concerning the calculations of the drag forces due to the water. In particular, the water is considered to be stationary and only local interaction between each link and the water is taken into account. More realistic simulations of the water flow should be developed for obtaining a simulation which comes closer quantitatively to the swimming of the real lamprey.

The simulation of the salamander-like animat requires several improvements to make it closer to a realistic biomechanical simulation of the real animal. Firstly, the simulation should be extended to the third dimension, such as to calculate the accelerations of the links in three dimensions rather than only in the horizontal plane. More joints and links should be used to simulate the limbs. The body should also be made of more than 9 links, in particular, neck links and muscles should be simulated.

A more complex mechanical simulation would require more complex control mechanisms than those evolved in this thesis. The different limb joints, for instance, need to be coordinated within a limb. Balance may also need to be controled actively, although the fact that the body of the salamander slides on the ground means that balance is less crucial in salamander locomotion than in tetrapods which are only supported by the limbs. Body muscles should probably be activated for preventing cyclic torsions of the body, and keeping enough rigidity during trotting for an effective transmission of the forces created by the limbs in the stance phase, *etc.*.

Note that the neural organisation proposed for the salamander in this thesis could probably accommodate these more complex control mechanisms, without having to be completely changed. The different limb joint oscillators could, for instance, be entrained by the limb oscillators I developed. The passage to the control of a 3D body, could also be obtained by the development of a crossed-oscillator body CPG similarly to that proposed in [Ekeberg *et al.* 95].

[5] The segmental crossed-oscillator network is composed of two separate segmental oscillators which are crossed in the section of the spinal cord such that ventral neurons of one side are connected to dorsal neurons on the contralateral side. The crossed oscillators are also synaptically weakly coupled to make them run in parallel (see [Ekeberg *et al.* 95]).

9.6.2 Implementation in a real robot

Complementing a more realistic salamander (or lamprey)[6] biomechanical simulation, it would be most interesting to build a salamander-like robot: firstly, because a physical implementation can only function when all elements necessary for locomotion are correctly implemented (compared to a simulation, in which some physical aspects may be wrongly represented), and secondly, because a robot which could both swim and walk would be a first, and would be very useful in many outdoor applications.

Such a amphibian robot would require solution of two main problems: the physical implementation and the design of a locomotion controller. The physical implementation would require addressing several technological challenges concerning the linear actuators, the joints, sensors, making the system waterproof, batteries, creating the correct density, *etc.*

The evolutionary design of a neural controller as proposed in this thesis would also need to be adapted in order to be applicable for the design of controllers for a physical robot. Two approaches can be taken: *online* evolution in which the controllers are evaluated directly in the robot, or *offline* evolution by evolving solutions in a simulator of the robot.

Online evolution requires an automatic external setup for evaluating a controller in the real robot, and a communication system for transmitting that information to the evolved controller (the genetic algorithm software can either be in the robot in which case the fitness value is communicated to the robot, or in the external setup, in which case a controller configuration has to be transmitted to the robot). Online evolution has successfully been used for the evolution of 6- or 8-legged walking controllers [Lewis *et al.* 93, Gruau & Quatramaran 97],[7] and there exist several examples of online evolution of behaviour controllers [Floreano 97, Floreano 98]. The drawbacks of this approach are that it requires a substantial setup if one wants the evolution to be unsupervised by the experimenter (the evaluation and communication system, the

[6] In this discussion, I mainly concentrate on the salamander, as a "lamprey system" can be seen as being included in a "salamander system".

[7] Note that for these two examples, the fitness value of a solution was given by visual inspection of the experimenter instead of having a complex external setup as in the other examples.

robot must be continuously powered, ...), and that care must be taken not to damage the robot (either by the result of a bad controller or simply by wear). But the main drawback is the time needed for the evolution, as evolutionary algorithms require numerous evaluations.

An alternative is therefore to perform offline evolution and having some preliminar evolutions in simulation. Oscillators could, for instance, first be evolved in a neural simulation, like the staged approach taken to evolve swimming controllers, followed by the evolution of the couplings between these oscillators (and possibly still further evolution of the oscillators) using the robotics setup. Alternatively, neural controllers could be evolved for a mechanical simulation for some generations and then transferred and evolved within the robot. This requires the simulation to be sufficiently realistic and close to the real robot for the transfer to be as smooth as possible. Examples of such an approach include [Jakobi *et al.* 95, Miglino *et al.* 95, O.Michel 96]. These examples however represent situations in which the robot and its environment had simple dynamics which could easily be simulated. Evolving locomotion controllers which take advantage of complex dynamics of the body and the environment may be significantly more difficult.

9.6.3 Central pattern generators for higher vertebrates

I believe that the investigation of the control of salamander locomotion and the design of potential controllers, could help us better understand and design controllers for the locomotion of higher vertebrates.

The next step would be, for instance, to study control of locomotion of quadrupeds such as cats. Because cats use limbs not only to move forward but also for supporting the body, they need locomotion controllers which are more complex than those for the salamander. The control of balance is then crucial. Another complexity of cat locomotion is that cats use significantly different gaits for different speeds.

I also hope that research in CPGs may significantly help medecine in a not too distant future, in order to help people with locomotor difficulties. I can see two directions in which research and development of CPGs may help handicaped persons. Firstly, CPG inspired controllers could be used for controlling robotic devices such as "legged-chairs" or biped exoskeletons. Secondly, gaining a better understanding of biped CPGs and how to modulate them may lead to devices capable of stimulating intact CPGs situated below the damaged part of the spinal cord of paraplegics, or, alternatively, capable of directly producing the rhythmic stimulation of muscles necessary for locomotion (and therefore replacing the damaged CPG). For all these applications, difficult problems such as how to control these devices and, in the case of the stimulation devices, how to insert, couple and maintain the device in a hostile environment (the body), need, of course, to be solved. Initial work towards legged-chairs and neuromuscular stimulation can be found in [Wellman *et al.* 95, Yamaguchi & Zajac 90], respectively, as reported in [Reeve 98].

9.6.4 Towards a better understanding of lower vertebrates' functioning

Salamanders and lampreys may be, among vertebrates, at the right level of complexity for gaining a better understanding of the functioning of the complex systems which are vertebrates. They may be simple enough to permit a more or less complete understanding of their global organisation and functioning, while having enough similarities with more complex vertebrates for their study to be useful for gaining a better understanding of vertebrates as a whole.

In the case of the salamander, for instance, a significant amount of work has been carried out on the analysis of its visual system. As mentioned at the end of chapter 9, two nice models simulating depth perception and prey localisation have been made [Eurich *et al.* 95, Eurich *et al.* 97]. It would be very interesting to work towards a complete model of the salamander (similarly to Arbib's *computational frog* [Cobas & Arbib 92, Arbib & Lee 94] but with a mechanical simulation of the body of the animal) which would integrate a central pattern generator for locomotion with models of higher control centres and models of the visual system. Such a model would, for example, enable computational sensori-motor coordination experiments which could be compared with actual observations of visually-guided behaviour in real salamanders.

Having complete models (i.e. which include the biomechanics of the body and a model of the central nervous system) of simple animals is important for gaining a better understanding of how behaviours arise from the central nervous system, and how sensory information is transformed into motor actions. Analysing only neural activity (either through recordings in the real animal, or through numerical simulations), without analysing the motor actions and behaviours it elicits, may never bring a full understanding of an animal's functioning.

9.6.5 Towards a complete animat

In parallel to gaining a better understanding of lower vertebrates, it would also be very interesting to work towards the development of a complete animat (either physical or simulated) displaying salamander-like behaviour, without necessary waiting for neurobiological data to become available. Rather than trying to replicate the functioning and the organisation of the real salamander, the emphasis would then rather be on the design method for obtaining a complete system.

This would mean integrating the motor controllers developed in this thesis within a system including sensory systems (a visual system, a vestibular system, touch sensors, *etc.*) and control centres for different behaviours such as tracking a prey, escaping predators, avoiding obstacles, finding mates, *etc.* A first small step in that direction was realised with a simple visual system in chapter 8.

This raises the typical questions of animat research of incremental design, modularity, hierarchy,... If this synthetic approach is carried out with neural based controllers, it may, similarly to the research presented in this thesis on the control of locomotion, bring a better understanding of the functioning of real animals, and/or, develop design methods which could be used by neurobiologists for making more complete models of central nervous systems.

The ambition is then, of course, to incrementally build more complex and adaptive systems, and little by little move towards the level of primate intelligence.

understanding of new behaviours arise from the central nervous system, and how sensory information is transformed into motor actions. Analyzing only neural activity (either through recording in the real animal, or through numerical simulations), and not analyzing the motor actions and behaviours it elicits, may never bring a full understanding of neuronal functioning.

9.6.5 Towards a complete animat

In parallel to gaining a better understanding of lower vertebrates, it would be be very interesting to work towards the development of a complete animat (either physical or simulated) displaying salamander-like behaviour, without necessary waiting for neurobiological data to become available. Rather than trying to replicate the functioning and the organisation of the real salamander, the emphasis would then rather be on the design method for obtaining a complete system.

This would mean integrating the motor controllers developed in this thesis within a system including sensory systems (a visual system, a vestibular system, touch sensors, etc.), and control centres for different behaviours such as tracking a prey, escaping predators, avoiding obstacles, finding mates, etc. A first small step in that direction was realised with a simple visual system in chapter 6.

This raises the typical questions of animat research of flow control, modularity, hierarchy, ... If the synthetic approach is carried out with neural networks, such a work may, similarly to the research presented in this thesis in the context of locomotion, bring a better understanding of the functioning of neural networks and develop design methods which could be used by neurobiologists for a better understanding of real central nervous systems.

The animat salamander based on a simulation or a robot would in addition be a suitable and interesting model for investigating the evolution of sensory and adaptive

Chapter 10

Conclusion

This thesis investigated the evolutionary design of neural networks for the control of animal-like locomotion. It was inspired by the neural organisation of locomotor circuits in vertebrates and studied in particular the undulatory locomotion of lampreys and salamanders.

The motivations behind the thesis were 1) to address the problems of control of animal-like locomotion, 2) to explore the space of possible neural configurations for the control of undulatory locomotion, and 3) to investigate how biologically plausible neural controllers might automatically be generated using evolutionary algorithms.

The thesis has therefore links with several fields and research areas such as neurobiology, robotics, animat research, evolutionary algorithms, and neural networks. It probably fits best in computational neuroethology, the field which investigates how behaviour arises from neural circuits and which designs neural-based control mechanisms for animats. The two ambitions of this field are to develop methods for designing neural controllers for increasingly complex animats, and, through this synthetic approach, to gain a better understanding of central nervous systems of real animals. This thesis provides two small steps in each direction on the subject of neural control of locomotion.

My work was inspired by neurobiological findings on the swimming circuitry of the lamprey and by observed analogies between the locomotion of salamanders and that of lampreys. This research was especially inspired by Ekeberg's neuronal and mechanical simulations of the lamprey which constituted the starting elements of my experiments.

Another source of inspiration was research carried out in the fields of neural networks and evolutionary algorithms, and their application to animat control.

The first part of this thesis investigated the neural control of lamprey-like swimming. Evolutionary algorithms were used to design connectionist models for controlling the swimming of a 2D mechanical simulation of a lamprey which is a reproduction of the simulation developed by Ekeberg. The evolved controllers could be compared with models of the circuitry found in the lamprey. The second part studied salamander-like locomotion. A salamander-like mechanical simulation was developed capable of both aquatic and terrestrial locomotion. Based on lamprey-like swimming controllers, controllers were evolved which can produce both the swimming and the trotting gaits observed in salamanders. As the locomotion circuitry of the salamander has not been decoded yet, these controllers could not be compared with actual biological models, but they provided a demonstration that a neural organisation of the locomotor circuitry similar to that suggested by neurobiologists could successfully be implemented in a connectionist model for exhibiting the gaits of the real salamander.

The main outcomes of this thesis can be summarized as follows:

- By visiting the space of possible neural configurations for the swimming of the lamprey using a staged evolution approach, I demonstrated that the circuitry found in the lamprey is not unique for producing the neural activity for anguiliform swimming. The genetic algorithm can be used to design alternative neural configurations, which share some similarities with the biological organisation (such as being made of coupled segmental oscillators), but which differ in how the neurons are connected and in the number of neurons. The evolved controllers produce swimming gaits very similar to that of the real lamprey and of Ekeberg's model, whose frequency, wavelength and therefore speed can be modulated by simple signals. To the best of my knowledge, this is the first example of evolution of neural controllers for swimming.

- This thesis also illustrated how a genetic algorithm can be used as a neurobiological modelling tool for automatically generating a part of a model which has the observed biological connectivity. The GA successfully generated synaptic weights for the biological model Ekeberg designed by hand, and even optimised Ekeberg's model for better fitting biological observations.

- In a collaboration with Jérôme Kodjabachian, we demonstrated that the developmental encoding he developed can be used to evolve swimming controllers for the mechanical simulation of the lamprey within a single stage. These evolutions led to a even larger variety of controller organisations compared to the evolutions with a staged approach. The interesting aspect of these evolutions is that their fitness function was only based on mechanical aspects and the approach could therefore be potentially applied to the evolution of a swimming robot without major changes in the design method.

- This thesis presented what, I believe, is the first example of a mechanical simulation of a salamander-like animat. Based on the simulation of the lamprey, a simple 2D simulation of a salamander-like was developed. Following a hypothesis on the neural organisation of the salamander's locomotion circuitry proposed by neurobiologists, connectionist models were developed based on a lamprey-like swimming CPG. The evolved controllers were able to produce both the swimming and the trotting gaits of the real salamander, and exhibited a neural activity which has several similarities with that measured in the real animal. Future neurobiological measurements will tell how close the evolved potential controllers are to the actual circuitry used by the salamander.

- A simple preliminary experiment with a very simple visual system was also carried out to illustrate how the evolved controllers can be integrated in higher control centres. With all tested controllers a successful tracking behaviour was implemented demonstrating that the controllers can cope with continously changing inputs.

Following these findings, the main conclusions of this thesis are that:

- Evolutionary algorithms are interesting tools for designing neural controllers which can be useful both for the animat field for designing controllers given a high-level description of the desired behaviour, and for neurobiology for instantiating unknown variables given a description of the organisation of the biological circuitry and of its general behaviour.

- Connectionist models constitute an interesting implementation of an animal-like locomotion control mechanism. Animal-like locomotion requires a control mechanism which can transform simple commands concerning the speed and direction of motion into the multiple rhythmic signals sent to the different actuators. Such a control mechanism can be provided by oscillatory networks of neurons, and this thesis demonstrated how simulated neural networks could be developed to generate, similarly to central pattern generators found in animals, the patterns of neural activity necessary for lamprey- and salamander-like locomotion. The controllers were designed such that very simple input signals can be used to modulate the speed, the direction and even the type of gait of the motion.

Appendix A

Results of chapter 4: segmental oscillators

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-	8.6	-3.8	-4.3	-	-0.1	-0.1
CCINl	-	7.8	-2.9	-0.5	-	-	2.9
LINl	-	10.3	-	-4.9	6.8	-1.5	-
MNl	-0.7	-	-	-4.6	-	-0.8	10.2
EINr	-4.3	-	-0.1	-	8.6	-3.8	-0.1
CCINr	-0.5	-	-	-	7.8	-2.9	2.9
LINr	-4.9	6.8	-1.5	-	10.3	-	-
MNr	-4.6	-	-0.8	-0.7	-	-	10.2

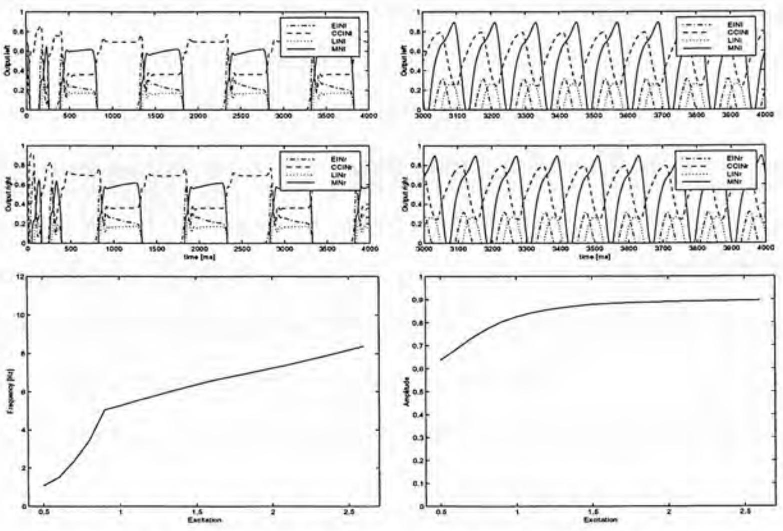


Figure A.1: **Run1:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-	-	-	-2.2	-	-4.6	4.3
CCINl	-	-	-	-	-	-	-
LINl	-3.3	-	-0.5	-0.4	-	-1.7	15.0
MNl	-0.2	-	-0.3	-4.4	-	-	11.3
EINr	-2.2	-	-4.6	-	-	-	4.3
CCINr	-	-	-	-	-	-	-
LINr	-0.4	-	-1.7	-3.3	-	-0.5	15.0
MNr	-4.4	-	-	-0.2	-	-0.3	11.3

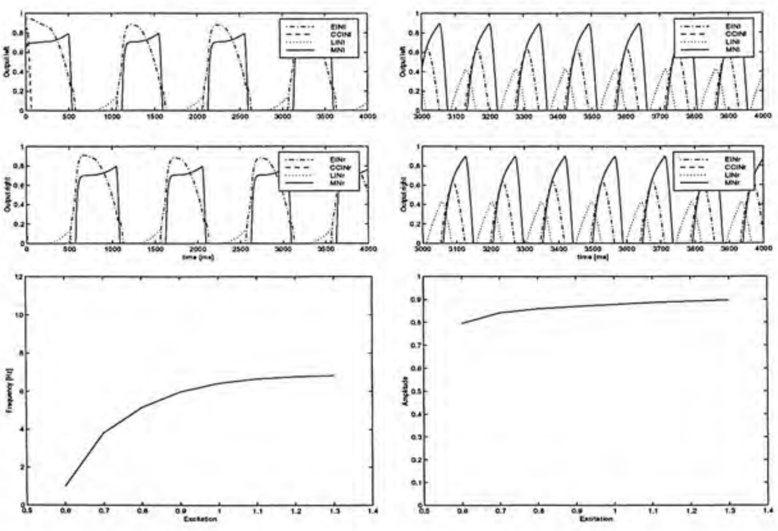


Figure A.2: **Run2:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-1.4	-4.0	-	-4.5	-	-	2.4
CCINl	-1.4	-1.0	-	-4.1	-4.0	-	3.7
LINl	-	-	-	-	-	-	-
MNl	-	-	-	-4.9	-3.5	-	7.6
EINr	-4.5	-	-	-1.4	-4.0	-	2.4
CCINr	-4.1	-4.0	-	-1.4	-1.0	-	3.7
LINr	-	-	-	-	-	-	-
MNr	-4.9	-3.5	-	-	-	-	7.6

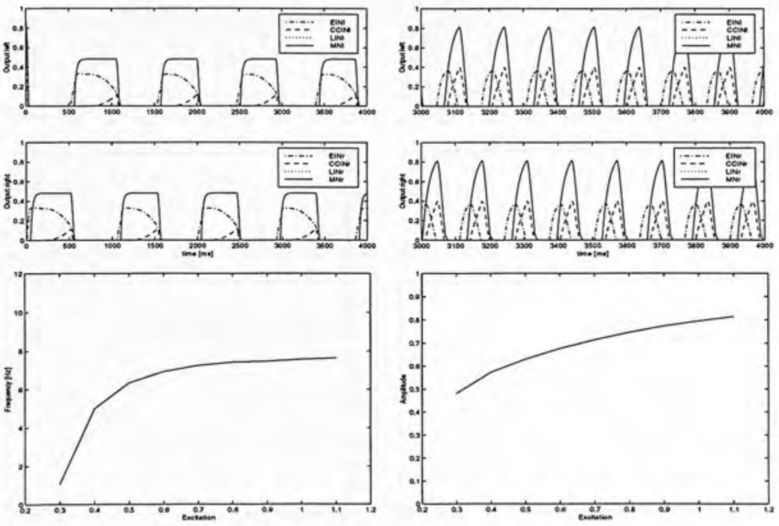


Figure A.3: **Run3:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-	-0.1	-0.2	9.6	-2.2	-	-
CCINl	4.4	-1.1	-5.0	-	-2.9	-	-
LINl	11.5	-1.3	-2.2	-	-3.2	-3.4	7.2
MNl	-	-0.2	-0.1	-	-5.0	-1.3	6.4
EINr	9.6	-2.2	-	-	-0.1	-0.2	-
CCINr	-	-2.9	-	4.4	-1.1	-5.0	-
LINr	-	-3.2	-3.4	11.5	-1.3	-2.2	7.2
MNr	-	-5.0	-1.3	-	-0.2	-0.1	6.4

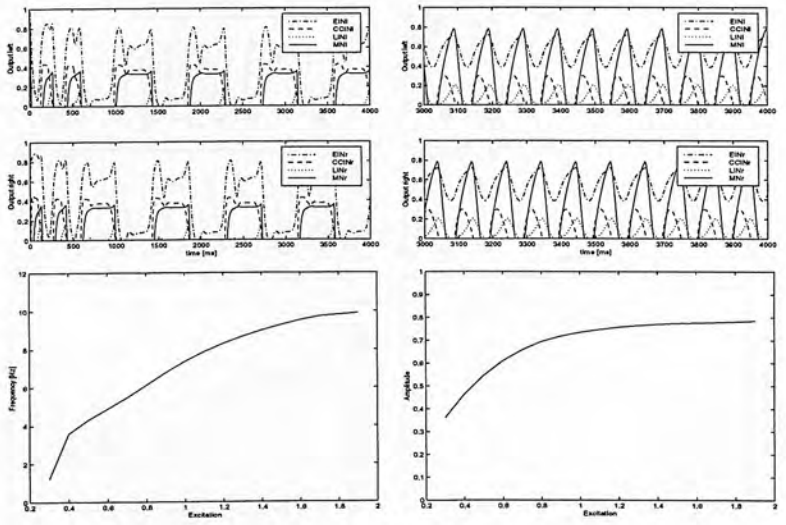


Figure A.4: **Run4:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-0.7	-2.2	-	-	-	-3.0	-
CCINl	-	-1.8	-	-	-4.4	-3.9	7.2
LINl	-	-3.5	-1.7	-0.4	-2.1	-2.1	15.0
MNl	-4.9	-	-	-0.3	-4.9	-	9.1
EINr	-	-	-3.0	-0.7	-2.2	-	-
CCINr	-	-4.4	-3.9	-	-1.8	-	7.2
LINr	-0.4	-2.1	-2.1	-	-3.5	-1.7	15.0
MNr	-0.3	-4.9	-	-4.9	-	-	9.1

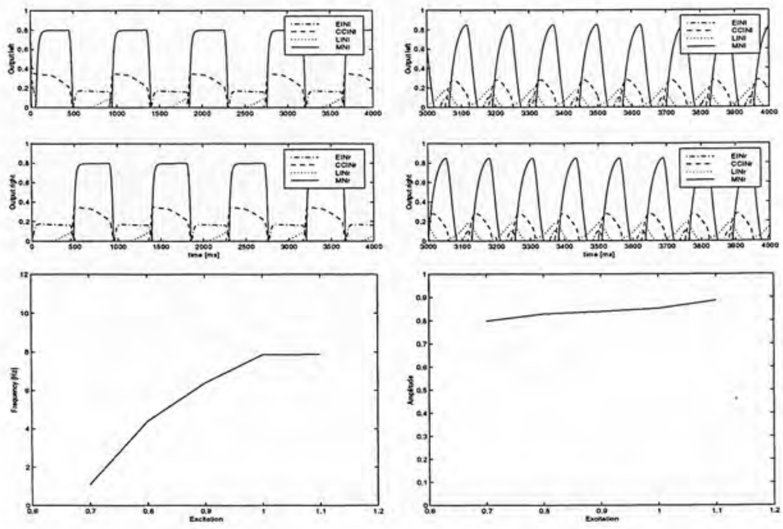


Figure A.5: **Run5:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-1.9	-5.0	-	-4.2	-0.1	-	12.1
CCINl	-2.8	-1.7	-	-5.0	-2.1	-	4.7
LINl	-	-	-	-	-	-	-
MNl	-5.0	-	-	-	-	-	4.8
EINr	-4.2	-0.1	-	-1.9	-5.0	-	12.1
CCINr	-5.0	-2.1	-	-2.8	-1.7	-	4.7
LINr	-	-	-	-	-	-	-
MNr	-	-	-	-5.0	-	-	4.8

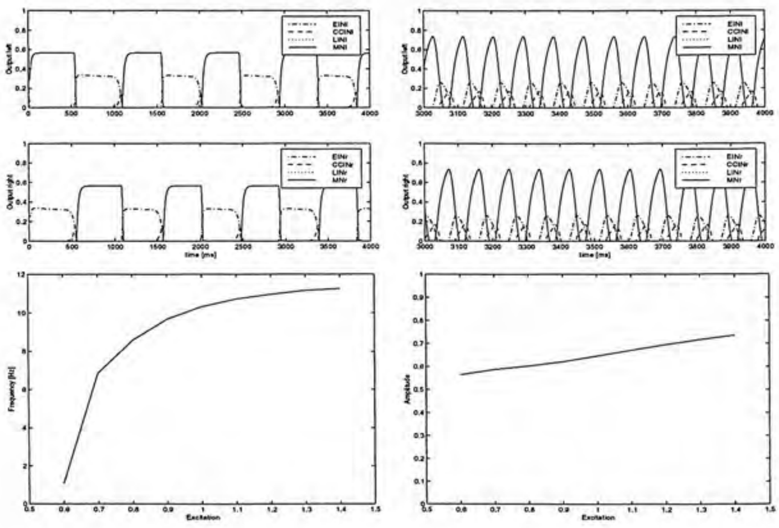


Figure A.6: **Run6:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-0.8	-3.8	-	-0.9	-0.7	-	0.8
CCINl	-	-	-	-3.5	-3.7	-	13.6
LINl	-	-	-	-	-	-	-
MNl	-0.4	-3.2	-	-	-	-	3.8
EINr	-0.9	-0.7	-	-0.8	-3.8	-	0.8
CCINr	-3.5	-3.7	-	-	-	-	13.6
LINr	-	-	-	-	-	-	-
MNr	-	-	-	-0.4	-3.2	-	3.8

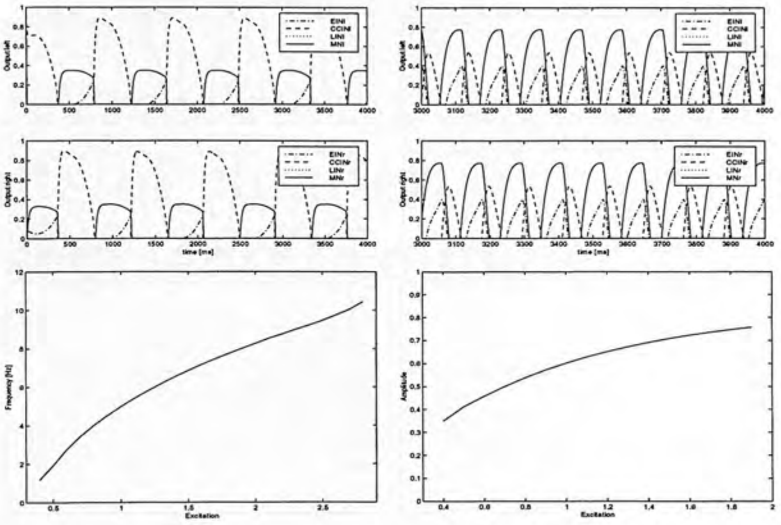


Figure A.7: **Run7:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-0.3	-3.6	-	-3.0	-	-	4.5
CCINl	-0.7	-0.2	-	-2.7	-3.8	-	2.7
LINl	-	-	-	-	-	-	-
MNl	-	-	-	-4.2	-	-	8.6
EINr	-3.0	-	-	-0.3	-3.6	-	4.5
CCINr	-2.7	-3.8	-	-0.7	-0.2	-	2.7
LINr	-	-	-	-	-	-	-
MNr	-4.2	-	-	-	-	-	8.6

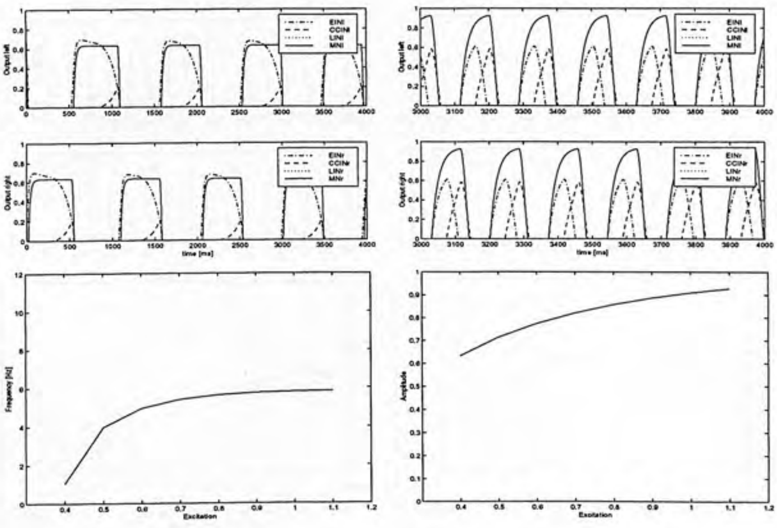


Figure A.8: **Run8:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-	-	-5.0	-2.7	9.5	-	-
CCINl	-	-	-3.6	-0.5	11.4	-2.3	5.5
LINl	-0.4	-	-2.9	-2.5	10.1	-2.2	11.0
MNl	-	-	-	-4.3	-	-	4.0
EINr	-2.7	9.5	-	-	-	-5.0	-
CCINr	-0.5	11.4	-2.3	-	-	-3.6	5.5
LINr	-2.5	10.1	-2.2	-0.4	-	-2.9	11.0
MNr	-4.3	-	-	-	-	-	4.0

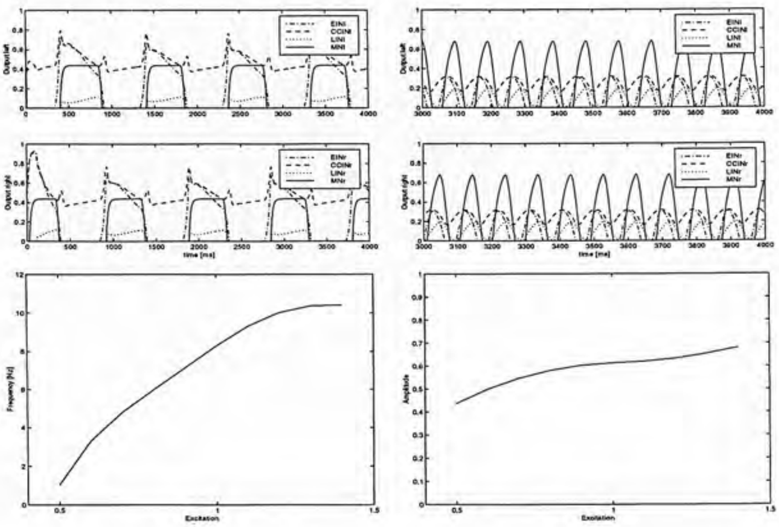


Figure A.9: **Run9:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-	-0.2	-	-3.2	-4.6	-	6.7
CCINl	-3.7	-2.3	-	-0.9	-2.6	-	2.4
LINl	-	-	-	-	-	-	-
MNl	-4.5	-	-	-	-	-	3.7
EINr	-3.2	-4.6	-	-	-0.2	-	6.7
CCINr	-0.9	-2.6	-	-3.7	-2.3	-	2.4
LINr	-	-	-	-	-	-	-
MNr	-	-	-	-4.5	-	-	3.7

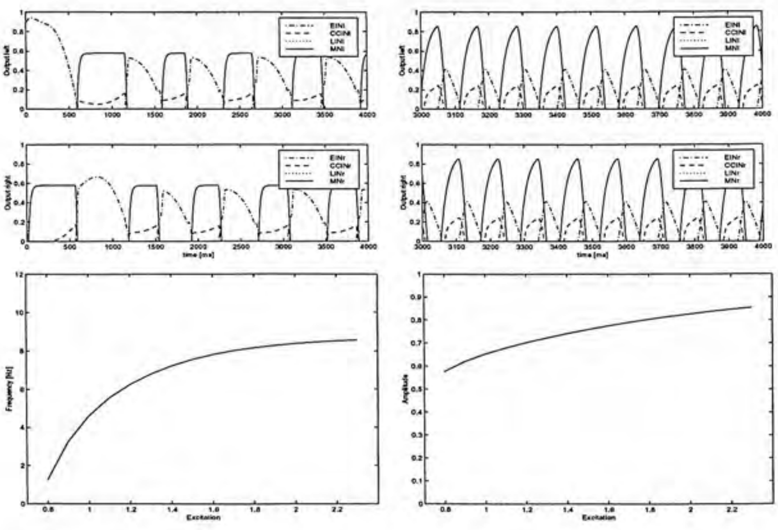


Figure A.10: **Run10:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

Appendix B

Results of chapter 4: intersegmental coupling

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-	8.6 [2, 5]	-3.8 [0, 2]	-4.3 [2, 12]	-	-0.1 [9, 4]	-0.1
CCINl	-	7.8 [4, 5]	-2.9 [3, 12]	-0.5 [9, 1]	-	-	2.9
LINl	-	10.3 [8, 11]	-	-4.9 [6, 8]	6.8 [2, 7]	-1.5 [2, 9]	-
MNl	-0.7 [4, 2]	-	-	-4.6 [11, 5]	-	-0.8 [3, 12]	10.2
EINr	-4.3 [2, 12]	-	-0.1 [9, 4]	-	8.6 [2, 5]	-3.8 [0, 2]	-0.1
CCINr	-0.5 [9, 1]	-	-	-	7.8 [4, 5]	-2.9 [3, 12]	2.9
LINr	-4.9 [6, 8]	6.8 [2, 7]	-1.5 [2, 9]	-	10.3 [8, 11]	-	-
MNr	-4.6 [11, 5]	-	-0.8 [3, 12]	-0.7 [4, 2]	-	-	10.2

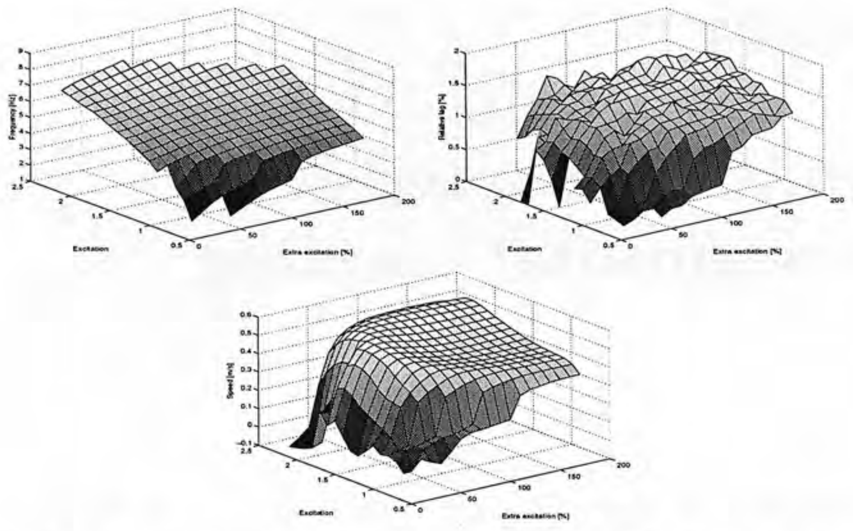


Figure B.1: **Run1:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom:* *idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-	-	-	-2.2 [4, 6]	-	-4.6 [3, 6]	4.3
CCINl	-	-	-	-	-	-	-
LINl	-3.3 [3, 1]	-	-0.5 [6, 12]	-0.4 [10, 10]	-	-1.7 [6, 3]	15.0
MNl	-0.2 [9, 2]	-	-0.3 [12, 6]	-4.4 [8, 4]	-	-	11.3
EINr	-2.2 [4, 6]	-	-4.6 [3, 6]	-	-	-	4.3
CCINr	-	-	-	-	-	-	-
LINr	-0.4 [10, 10]	-	-1.7 [6, 3]	-3.3 [3, 1]	-	-0.5 [6, 12]	15.0
MNr	-4.4 [8, 4]	-	-	-0.2 [9, 2]	-	-0.3 [12, 6]	11.3

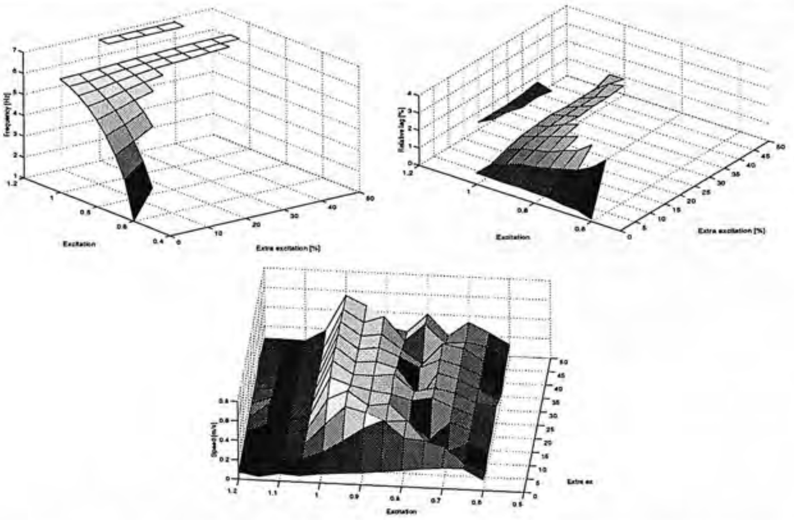


Figure B.2: **Run2:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom:* *idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-1.4 [10, 0]	-4.0 [2, 4]	-	-4.5 [4, 5]	-	-	2.4
CCINl	-1.4 [2, 12]	-1.0 [4, 10]	-	-4.1 [5, 6]	-4.0 [4, 1]	-	3.7
LINl	-	-	-	-	-	-	-
MNl	-0.0 [5, 9]	-	-	-4.9 [7, 6]	-3.5 [8, 9]	-	7.6
EINr	-4.5 [4, 5]	-	-	-1.4 [10, 0]	-4.0 [2, 4]	-	2.4
CCINr	-4.1 [5, 6]	-4.0 [4, 1]	-	-1.4 [2, 12]	-1.0 [4, 10]	-	3.7
LINr	-	-	-	-	-	-	-
MNr	-4.9 [7, 6]	-3.5 [8, 9]	-	-0.0 [5, 9]	-	-	7.6

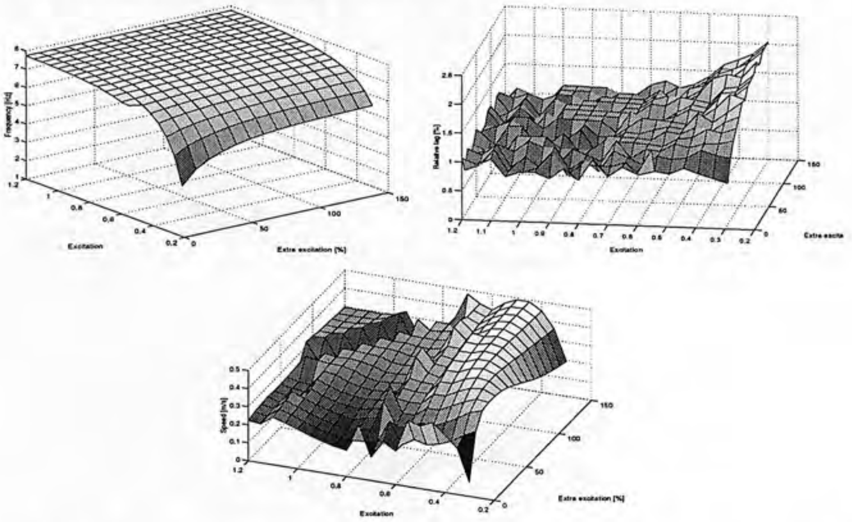


Figure B.3: **Run3:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-	-0.1 [11, 8]	-0.2 [1, 11]	9.6 [9, 10]	-2.2 [4, 8]	-	-
CCINl	4.4 [3, 10]	-1.1 [10, 0]	-5.0 [8, 8]	-	-2.9 [2, 3]	-	-
LINl	11.5 [2, 5]	-1.3 [9, 12]	-2.2 [4, 4]	-	-3.2 [3, 1]	-3.4 [6, 8]	7.2
MNl	-	-0.2 [10, 3]	-0.1 [4, 9]	-	-5.0 [3, 6]	-1.3 [6, 3]	6.4
EINr	9.6 [9, 10]	-2.2 [4, 8]	-	-	-0.1 [11, 8]	-0.2 [1, 11]	-
CCINr	-	-2.9 [2, 3]	-	4.4 [3, 10]	-1.1 [10, 0]	-5.0 [8, 8]	-
LINr	-	-3.2 [3, 1]	-3.4 [6, 8]	11.5 [2, 5]	-1.3 [9, 12]	-2.2 [4, 4]	7.2
MNr	-	-5.0 [3, 6]	-1.3 [6, 3]	-	-0.2 [10, 3]	-0.1 [4, 9]	6.4

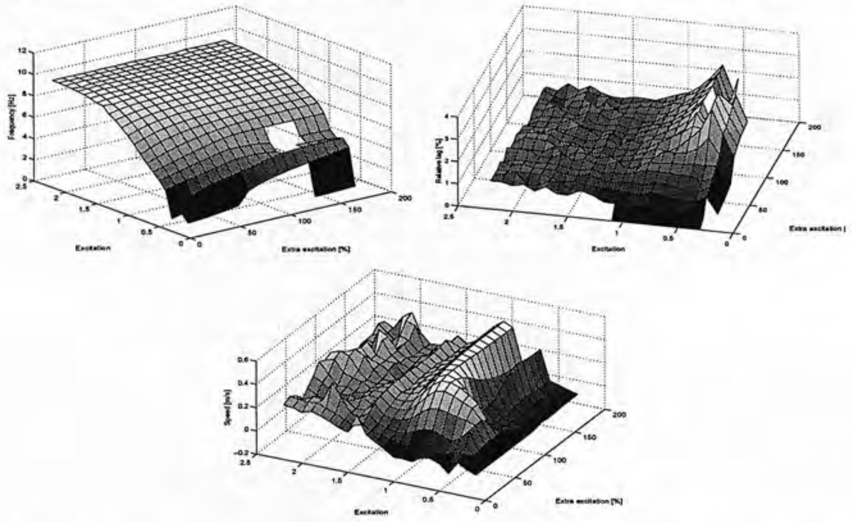


Figure B.4: **Run4:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-0.7 [5, 6]	-2.2 [9, 8]	-	-	-	-3.0 [2, 6]	-
CCINl	-	-1.8 [9, 0]	-	-	-4.4 [6, 6]	-3.9 [4, 12]	7.2
LINl	-	-3.5 [7, 1]	-1.7 [6, 2]	-0.4 [5, 9]	-2.1 [3, 3]	-2.1 [7, 11]	15.0
MNl	-4.9 [12, 2]	-	-	-0.3 [7, 10]	-4.9 [0, 3]	-	9.1
EINr	-	-	-3.0 [2, 6]	-0.7 [5, 6]	-2.2 [9, 8]	-	-
CCINr	-	-4.4 [6, 6]	-3.9 [4, 12]	-	-1.8 [9, 0]	-	7.2
LINr	-0.4 [5, 9]	-2.1 [3, 3]	-2.1 [7, 11]	-	-3.5 [7, 1]	-1.7 [6, 2]	15.0
MNr	-0.3 [7, 10]	-4.9 [0, 3]	-	-4.9 [12, 2]	-	-	9.1

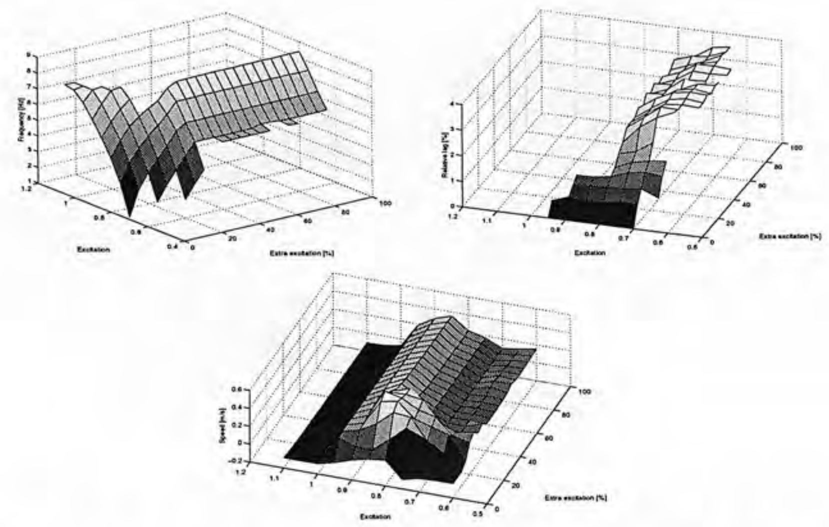


Figure B.5: **Run5:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-1.9 [11, 4]	-5.0 [6, 8]	-	-4.2 [4, 8]	-0.1 [5, 8]	-	12.1
CCINl	-2.8 [5, 5]	-1.7 [7, 0]	-	-5.0 [8, 2]	-2.1 [3, 2]	-	4.7
LINl	-	-	-	-	-	-	-
MNl	-5.0 [4, 0]	-	-	-	-	-	4.8
EINr	-4.2 [4, 8]	-0.1 [5, 8]	-	-1.9 [11, 4]	-5.0 [6, 8]	-	12.1
CCINr	-5.0 [8, 2]	-2.1 [3, 2]	-	-2.8 [5, 5]	-1.7 [7, 0]	-	4.7
LINr	-	-	-	-	-	-	-
MNr	-	-	-	-5.0 [4, 0]	-	-	4.8

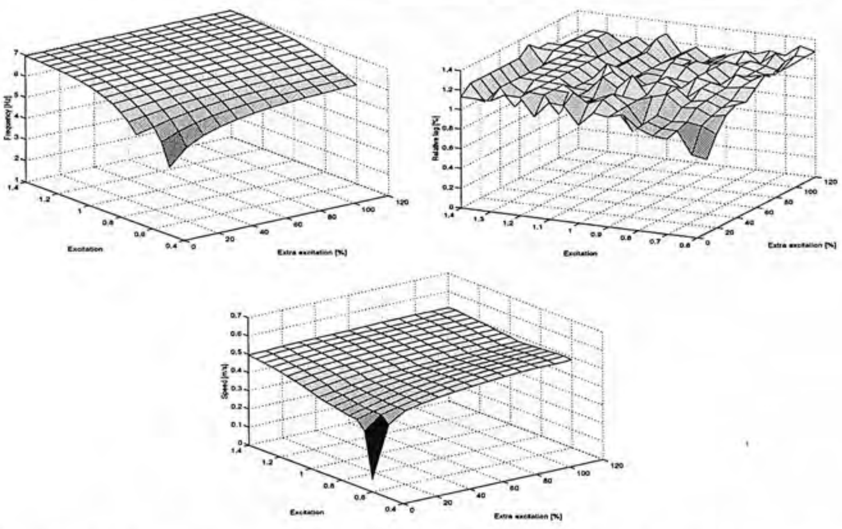


Figure B.6: **Run6:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-0.8 [12, 4]	-3.8 [12, 10]	-	-0.9 [5, 10]	-0.7 [1, 10]	-	0.8
CCINl	-0.0 [12, 8]	-	-	-3.5 [2, 2]	-3.7 [9, 9]	-	13.6
LINl	-	-	-	-	-	-	-
MNl	-0.4 [9, 2]	-3.2 [8, 1]	-	-	-	-	3.8
EINr	-0.9 [5, 10]	-0.7 [1, 10]	-	-0.8 [12, 4]	-3.8 [12, 10]	-	0.8
CCINr	-3.5 [2, 2]	-3.7 [9, 9]	-	-0.0 [12, 8]	-	-	13.6
LINr	-	-	-	-	-	-	-
MNr	-	-	-	-0.4 [9, 2]	-3.2 [8, 1]	-	3.8

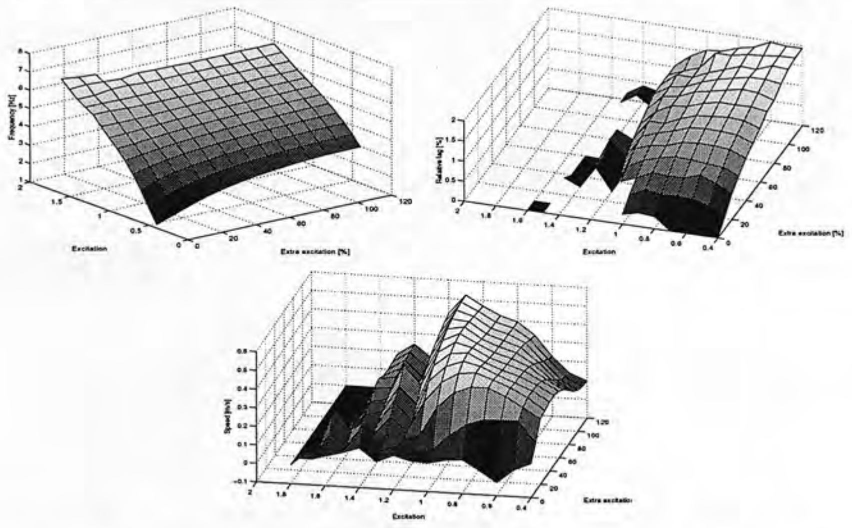


Figure B.7: **Run7:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-0.3 [11, 1]	-3.6 [0, 8]	-	-3.0 [0, 5]	-	-	4.5
CCINl	-0.7 [0, 5]	-0.2 [3, 11]	-	-2.7 [11, 1]	-3.8 [6, 2]	-	2.7
LINl	-	-	-	-	-	-	-
MNl	-	-	-	-4.2 [12, 4]	-	-	8.6
EINr	-3.0 [0, 5]	-	-	-0.3 [11, 1]	-3.6 [0, 8]	-	4.5
CCINr	-2.7 [11, 1]	-3.8 [6, 2]	-	-0.7 [0, 5]	-0.2 [3, 11]	-	2.7
LINr	-	-	-	-	-	-	-
MNr	-4.2 [12, 4]	-	-	-	-	-	8.6

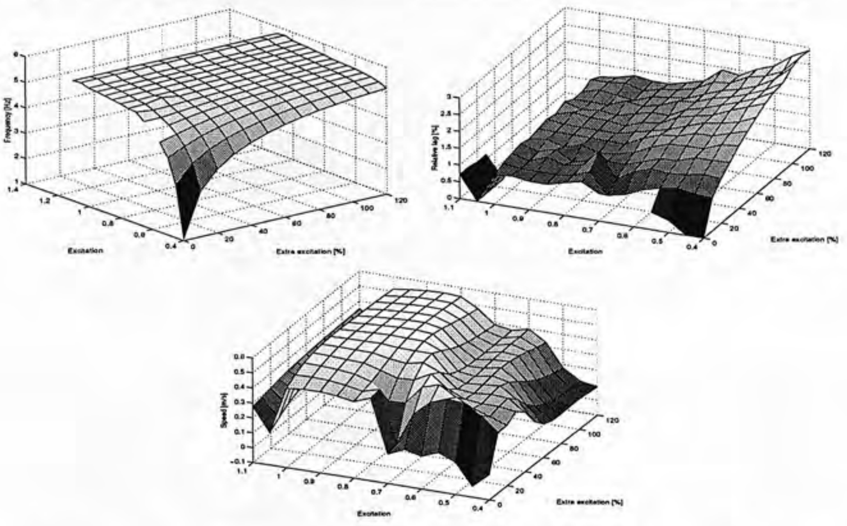


Figure B.8: **Run8:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-	-	-5.0 [5, 2]	-2.7 [1, 4]	9.5 [10, 9]	-	-
CCINl	-	-	-3.6 [5, 6]	-0.5 [2, 1]	11.4 [2, 12]	-2.3 [4, 1]	5.5
LINl	-0.4 [8, 3]	-	-2.9 [11, 11]	-2.5 [5, 7]	10.1 [11, 12]	-2.2 [2, 8]	11.0
MNl	-	-	-	-4.3 [1, 5]	-	-	4.0
EINr	-2.7 [1, 4]	9.5 [10, 9]	-	-	-	-5.0 [5, 2]	-
CCINr	-0.5 [2, 1]	11.4 [2, 12]	-2.3 [4, 1]	-	-	-3.6 [5, 6]	5.5
LINr	-2.5 [5, 7]	10.1 [11, 12]	-2.2 [2, 8]	-0.4 [8, 3]	-	-2.9 [11, 11]	11.0
MNr	-4.3 [1, 5]	-	-	-	-	-	4.0

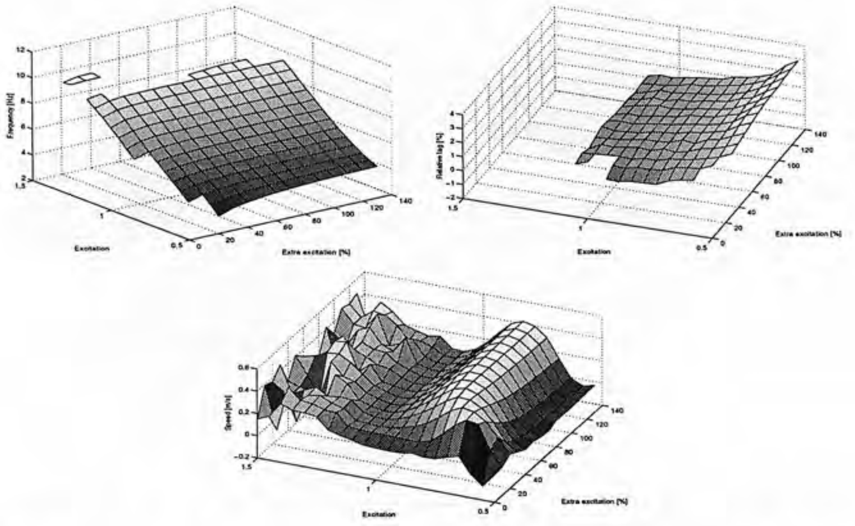


Figure B.9: **Run9:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-	-0.2 [1, 4]	-	-3.2 [6, 5]	-4.6 [0, 9]	-	6.7
CCINl	-3.7 [9, 1]	-2.3 [6, 11]	-	-0.9 [7, 0]	-2.6 [2, 7]	-	2.4
LINl	-	-	-	-	-	-	-
MNl	-4.5 [4, 3]	-	-	-	-	-	3.7
EINr	-3.2 [6, 5]	-4.6 [0, 9]	-	-	-0.2 [1, 4]	-	6.7
CCINr	-0.9 [7, 0]	-2.6 [2, 7]	-	-3.7 [9, 1]	-2.3 [6, 11]	-	2.4
LINr	-	-	-	-	-	-	-
MNr	-	-	-	-4.5 [4, 3]	-	-	3.7

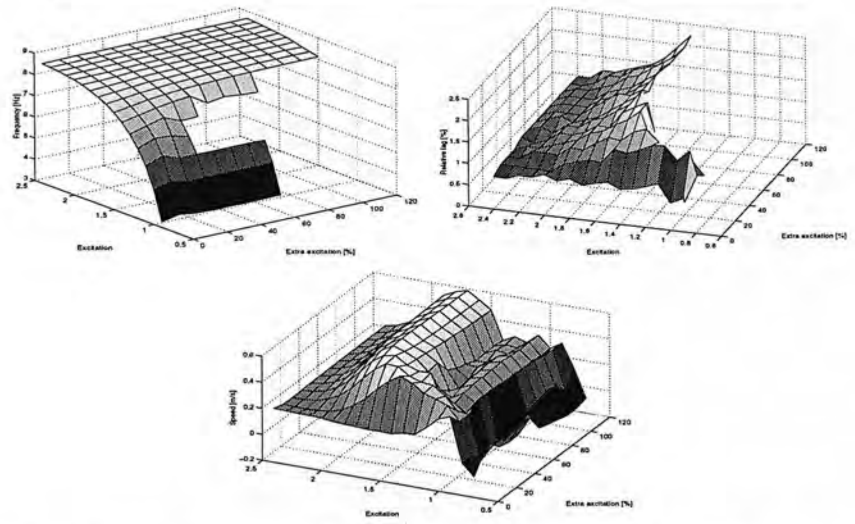


Figure B.10: **Run10:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

Appendix C

Results of chapter 4: sensory feedback

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	-0.8	-3.8	-	-0.9	-0.7	-	0.0	0.8	0.8
CCINl	-	-	-	-3.5	-3.7	-	1.6	3.0	13.6
LINl	-	-	-	-	-	-	-	-	-
MNl	-0.4	-3.2	-	-	-	-	4.2	-0.3	3.8
EINr	-0.9	-0.7	-	-0.8	-3.8	-	0.8	0.0	0.8
CCINr	-3.5	-3.7	-	-	-	-	3.0	1.6	13.6
LINr	-	-	-	-	-	-	-	-	-
MNr	-	-	-	-0.4	-3.2	-	-0.3	4.2	3.8

Table C.1: **Evolved sensory feedback for artificial controller No 7**, best of run1

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	-0.8	-3.8	-	-0.9	-0.7	-	0.1	3.4	0.8
CCINl	-	-	-	-3.5	-3.7	-	2.2	2.3	13.6
LINl	-	-	-	-	-	-	-	-	-
MNl	-0.4	-3.2	-	-	-	-	4.2	1.9	3.8
EINr	-0.9	-0.7	-	-0.8	-3.8	-	3.4	0.1	0.8
CCINr	-3.5	-3.7	-	-	-	-	2.3	2.2	13.6
LINr	-	-	-	-	-	-	-	-	-
MNr	-	-	-	-0.4	-3.2	-	1.9	4.2	3.8

Table C.2: **Evolved sensory feedback for artificial controller No 7**, best of run2

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	-0.8	-3.8	-	-0.9	-0.7	-	0.5	4.1	0.8
CCINl	-	-	-	-3.5	-3.7	-	2.0	4.1	13.6
LINl	-	-	-	-	-	-	-	-	-
MNl	-0.4	-3.2	-	-	-	-	2.8	2.9	3.8
EINr	-0.9	-0.7	-	-0.8	-3.8	-	4.1	0.5	0.8
CCINr	-3.5	-3.7	-	-	-	-	4.1	2.0	13.6
LINr	-	-	-	-	-	-	-	-	-
MNr	-	-	-	-0.4	-3.2	-	2.9	2.8	3.8

Table C.3: **Evolved sensory feedback for artificial controller No 7**, best of run3

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	-0.8	-3.8	-	-0.9	-0.7	-	0.1	0.7	0.8
CCINl	-	-	-	-3.5	-3.7	-	4.2	0.5	13.6
LINl	-	-	-	-	-	-	-	-	-
MNl	-0.4	-3.2	-	-	-	-	1.4	2.4	3.8
EINr	-0.9	-0.7	-	-0.8	-3.8	-	0.7	0.1	0.8
CCINr	-3.5	-3.7	-	-	-	-	0.5	4.2	13.6
LINr	-	-	-	-	-	-	-	-	-
MNr	-	-	-	-0.4	-3.2	-	2.4	1.4	3.8

Table C.4: **Evolved sensory feedback for artificial controller No 7**, best of run4

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	-0.8	-3.8	-	-0.9	-0.7	-	0.1	5.0	0.8
CCINl	-	-	-	-3.5	-3.7	-	-0.6	0.0	13.6
LINl	-	-	-	-	-	-	-	-	-
MNl	-0.4	-3.2	-	-	-	-	3.3	3.6	3.8
EINr	-0.9	-0.7	-	-0.8	-3.8	-	5.0	0.1	0.8
CCINr	-3.5	-3.7	-	-	-	-	0.0	-0.6	13.6
LINr	-	-	-	-	-	-	-	-	-
MNr	-	-	-	-0.4	-3.2	-	3.6	3.3	3.8

Table C.5: **Evolved sensory feedback for artificial controller No 7**, best of run5

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	-0.8	-3.8	-	-0.9	-0.7	-	0.2	2.7	0.8
CCINl	-	-	-	-3.5	-3.7	-	0.0	1.0	13.6
LINl	-	-	-	-	-	-	-	-	-
MNl	-0.4	-3.2	-	-	-	-	-	-	3.8
EINr	-0.9	-0.7	-	-0.8	-3.8	-	2.7	0.2	0.8
CCINr	-3.5	-3.7	-	-	-	-	1.0	0.0	13.6
LINr	-	-	-	-	-	-	-	-	-
MNr	-	-	-	-0.4	-3.2	-	-	-	3.8

Table C.6: **Evolved sensory feedback for artificial controller No 7**, best of run6

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	-0.8	-3.8	-	-0.9	-0.7	-	0.1	0.4	0.8
CCINl	-	-	-	-3.5	-3.7	-	-0.3	3.2	13.6
LINl	-	-	-	-	-	-	-	-	-
MNl	-0.4	-3.2	-	-	-	-	-	-	3.8
EINr	-0.9	-0.7	-	-0.8	-3.8	-	0.4	0.1	0.8
CCINr	-3.5	-3.7	-	-	-	-	3.2	-0.3	13.6
LINr	-	-	-	-	-	-	-	-	-
MNr	-	-	-	-0.4	-3.2	-	-	-	3.8

Table C.7: **Evolved sensory feedback for artificial controller No 7**, best of run7

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	-0.8	-3.8	-	-0.9	-0.7	-	0.3	0.3	0.8
CCINl	-	-	-	-3.5	-3.7	-	-1.1	1.9	13.6
LINl	-	-	-	-	-	-	-	-	-
MNl	-0.4	-3.2	-	-	-	-	-	-	3.8
EINr	-0.9	-0.7	-	-0.8	-3.8	-	0.3	0.3	0.8
CCINr	-3.5	-3.7	-	-	-	-	1.9	-1.1	13.6
LINr	-	-	-	-	-	-	-	-	-
MNr	-	-	-	-0.4	-3.2	-	-	-	3.8

Table C.8: **Evolved sensory feedback for artificial controller No 7**, best of run8

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	-0.8	-3.8	-	-0.9	-0.7	-	-0.0	0.0	0.8
CCINl	-	-	-	-3.5	-3.7	-	-0.2	0.1	13.6
LINl	-	-	-	-	-	-	-	-	-
MNl	-0.4	-3.2	-	-	-	-	-	-	3.8
EINr	-0.9	-0.7	-	-0.8	-3.8	-	0.0	-0.0	0.8
CCINr	-3.5	-3.7	-	-	-	-	0.1	-0.2	13.6
LINr	-	-	-	-	-	-	-	-	-
MNr	-	-	-	-0.4	-3.2	-	-	-	3.8

Table C.9: **Evolved sensory feedback for artificial controller No 7**, best of run9

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	-0.8	-3.8	-	-0.9	-0.7	-	0.2	2.7	0.8
CCINl	-	-	-	-3.5	-3.7	-	3.8	1.2	13.6
LINl	-	-	-	-	-	-	-	-	-
MNl	-0.4	-3.2	-	-	-	-	-	-	3.8
EINr	-0.9	-0.7	-	-0.8	-3.8	-	2.7	0.2	0.8
CCINr	-3.5	-3.7	-	-	-	-	1.2	3.8	13.6
LINr	-	-	-	-	-	-	-	-	-
MNr	-	-	-	-0.4	-3.2	-	-	-	3.8

Table C.10: **Evolved sensory feedback for artificial controller No 7**, best of run10

Appendix D

Results of chapter 5: segmental oscillators

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	7.6	-	-	-	-2.6	-	9.5
CCINl	-	-	-4.4	-	-1.2	-	8.8
LINl	6.3	-	-	-	-2.3	-	4.4
MNl	-	-	-	-	-1.4	-	3.2
EINr	-	-2.6	-	7.6	-	-	9.5
CCINr	-	-1.2	-	-	-	-4.4	8.8
LINr	-	-2.3	-	6.3	-	-	4.4
MNr	-	-1.4	-	-	-	-	3.2

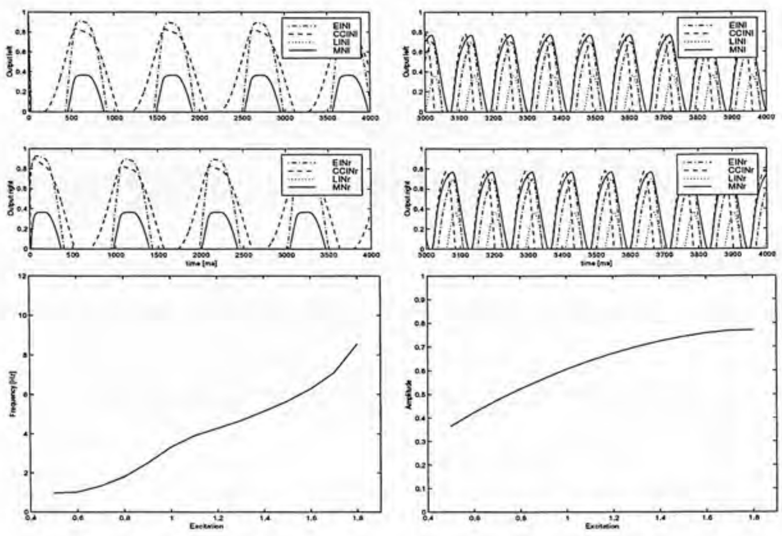


Figure D.1: **Run1:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	3.9	-	-	-	-3.4	-	10.3
CCINl	11.3	-	-4.8	-	-1.3	-	13.0
LINl	4.9	-	-	-	-4.3	-	5.0
MNl	1.9	-	-	-	-1.9	-	2.0
EINr	-	-3.4	-	3.9	-	-	10.3
CCINr	-	-1.3	-	11.3	-	-4.8	13.0
LINr	-	-4.3	-	4.9	-	-	5.0
MNr	-	-1.9	-	1.9	-	-	2.0

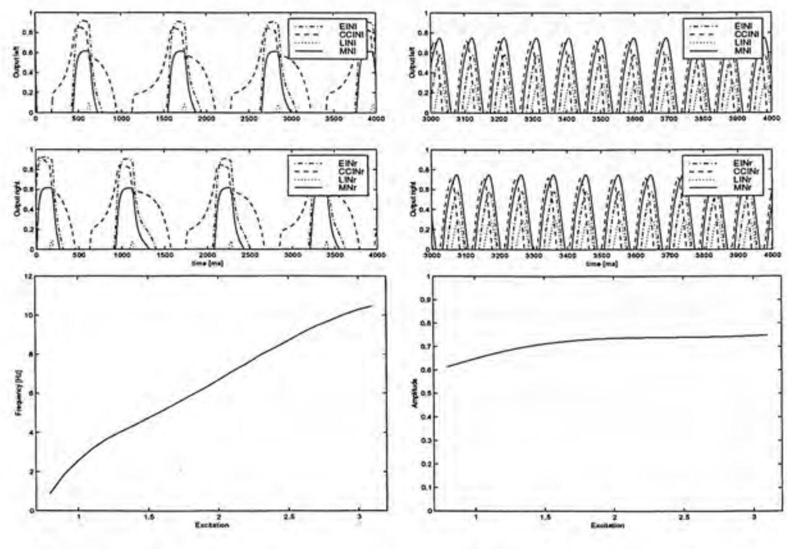


Figure D.2: **Run2:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	10.1	-	-	-	-3.8	-	3.8
CCINl	12.8	-	-4.0	-	-1.3	-	11.1
LINl	6.0	-	-	-	-4.2	-	4.3
MNl	-	-	-	-	-1.8	-	3.0
EINr	-	-3.8	-	10.1	-	-	3.8
CCINr	-	-1.3	-	12.8	-	-4.0	11.1
LINr	-	-4.2	-	6.0	-	-	4.3
MNr	-	-1.8	-	-	-	-	3.0

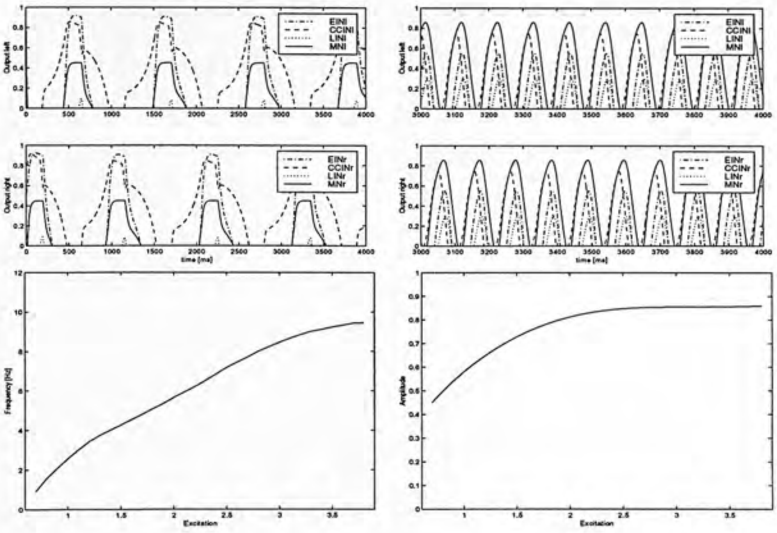


Figure D.3: **Run3:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	-	-	-	-	-3.6	-	7.4
CCINl	10.4	-	-3.8	-	-1.2	-	13.1
LINl	5.3	-	-	-	-1.4	-	5.4
MNl	-	-	-	-	-1.5	-	4.1
EINr	-	-3.6	-	-	-	-	7.4
CCINr	-	-1.2	-	10.4	-	-3.8	13.1
LINr	-	-1.4	-	5.3	-	-	5.4
MNr	-	-1.5	-	-	-	-	4.1

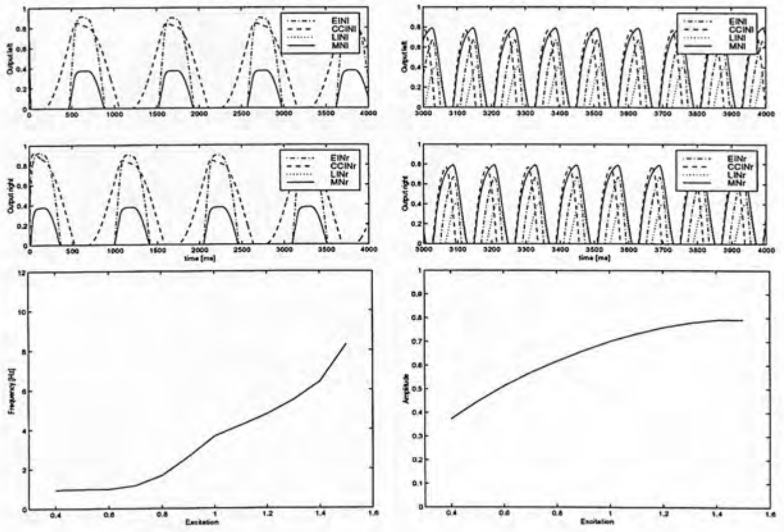


Figure D.4: **Run4:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	6.8	-	-	-	-0.0	-	7.1
CCINl	3.4	-	-4.5	-	-1.3	-	7.6
LINl	3.7	-	-	-	-4.2	-	6.9
MNl	-	-	-	-	-2.0	-	4.5
EINr	-	-0.0	-	6.8	-	-	7.1
CCINr	-	-1.3	-	3.4	-	-4.5	7.6
LINr	-	-4.2	-	3.7	-	-	6.9
MNr	-	-2.0	-	-	-	-	4.5

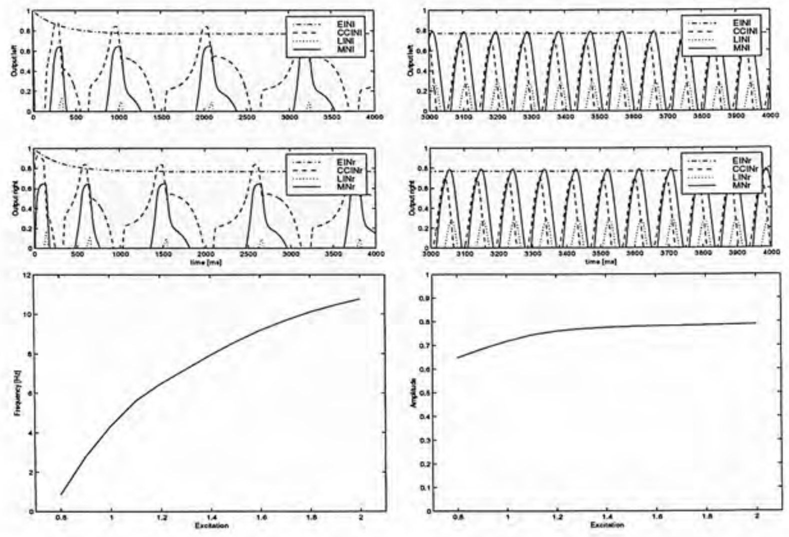


Figure D.5: **Run5:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	14.3	-	-	-	-1.0	-	-
CCINl	5.2	-	-4.3	-	-1.3	-	14.8
LINl	7.2	-	-	-	-4.2	-	3.4
MNl	2.0	-	-	-	-1.8	-	1.5
EINr	-	-1.0	-	14.3	-	-	-
CCINr	-	-1.3	-	5.2	-	-4.3	14.8
LINr	-	-4.2	-	7.2	-	-	3.4
MNr	-	-1.8	-	2.0	-	-	1.5

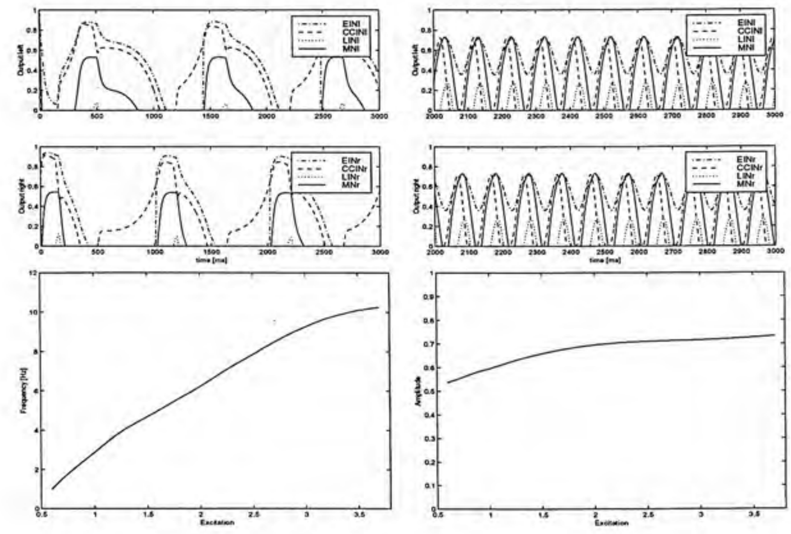


Figure D.6: **Run6:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	4.5	-	-	-	-1.7	-	6.5
CCINl	6.4	-	-3.3	-	-1.6	-	-
LINl	9.4	-	-	-	-5.0	-	5.4
MNl	1.9	-	-	-	-2.6	-	2.7
EINr	-	-1.7	-	4.5	-	-	6.5
CCINr	-	-1.6	-	6.4	-	-3.3	-
LINr	-	-5.0	-	9.4	-	-	5.4
MNr	-	-2.6	-	1.9	-	-	2.7

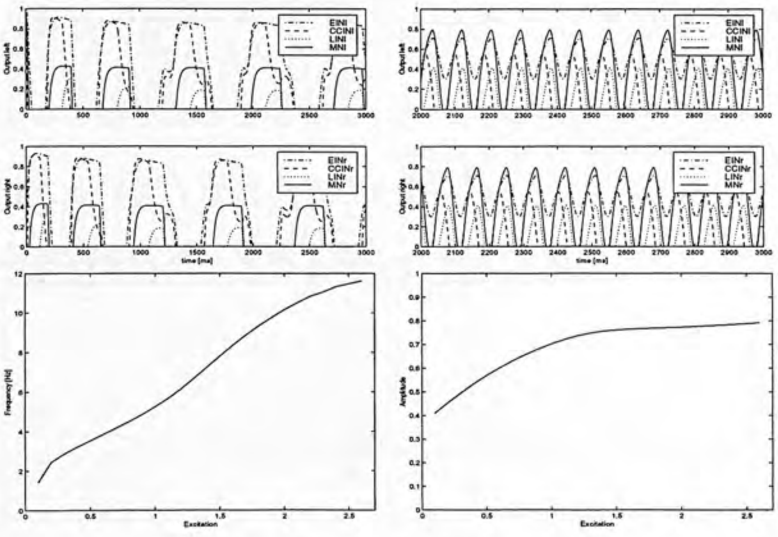


Figure D.7: **Run7:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	2.4	-	-	-	-2.3	-	9.6
CCINl	10.0	-	-4.1	-	-1.3	-	10.3
LINl	4.2	-	-	-	-4.1	-	6.6
MNl	-	-	-	-	-1.8	-	3.4
EINr	-	-2.3	-	2.4	-	-	9.6
CCINr	-	-1.3	-	10.0	-	-4.1	10.3
LINr	-	-4.1	-	4.2	-	-	6.6
MNr	-	-1.8	-	-	-	-	3.4

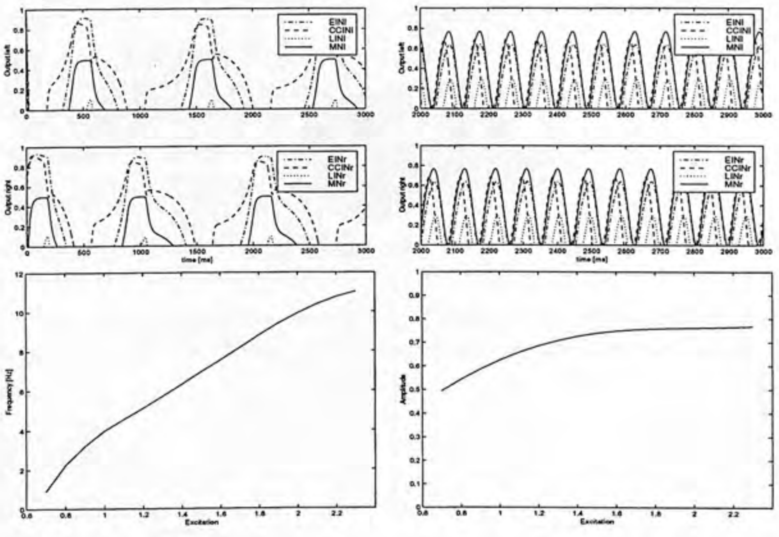


Figure D.8: **Run8:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	6.6	-	-	-	-3.4	-	1.1
CCINl	-	-	-4.5	-	-1.3	-	10.0
LINl	4.1	-	-	-	-4.3	-	5.8
MNl	-	-	-	-	-2.0	-	3.3
EINr	-	-3.4	-	6.6	-	-	1.1
CCINr	-	-1.3	-	-	-	-4.5	10.0
LINr	-	-4.3	-	4.1	-	-	5.8
MNr	-	-2.0	-	-	-	-	3.3

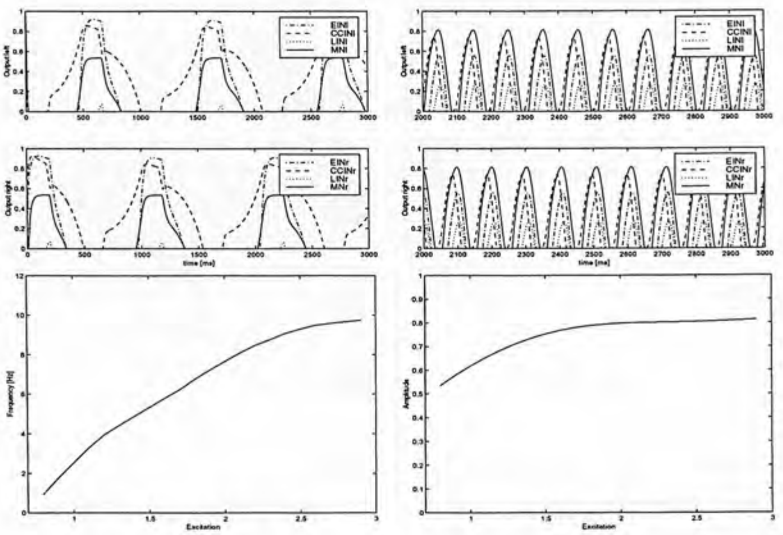


Figure D.9: **Run9:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	14.3	-	-	-	-1.5	-	-
CCINl	12.0	-	-4.0	-	-1.3	-	13.6
LINl	5.2	-	-	-	-4.1	-	5.3
MNl	-	-	-	-	-1.9	-	3.3
EINr	-	-1.5	-	14.3	-	-	-
CCINr	-	-1.3	-	12.0	-	-4.0	13.6
LINr	-	-4.1	-	5.2	-	-	5.3
MNr	-	-1.9	-	-	-	-	3.3

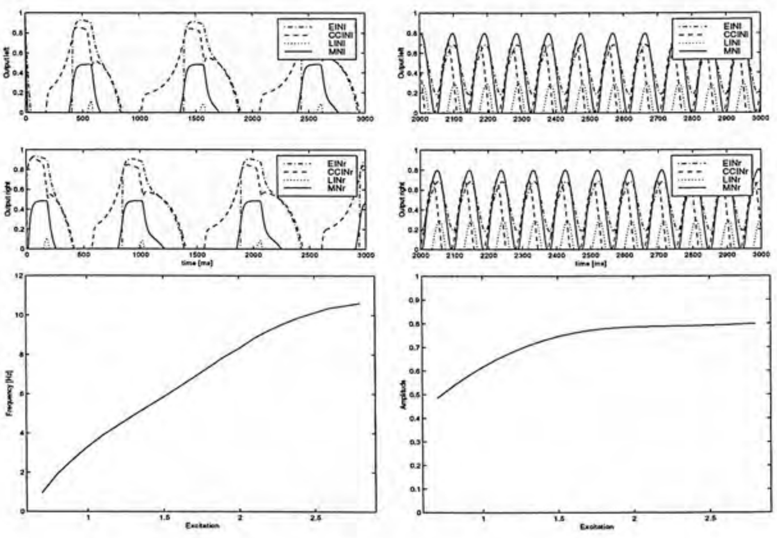
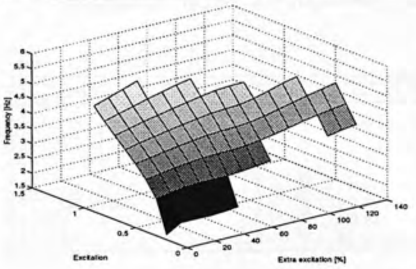


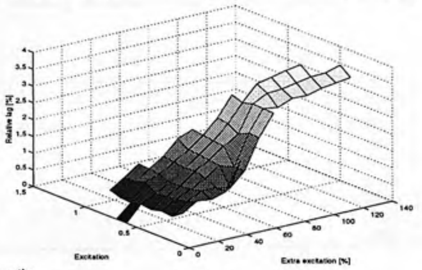
Figure D.10: **Run10:** *Top:* Neural configuration, *Middle:* Simulation at lowest and highest frequency, *Bottom:* Effect of the excitation on the frequency and the amplitude of MN signals.

Appendix E

Results of chapter 5: intersegmental coupling

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	0.4 [10, 2]	-	-	-	-2.0 [10, 5]	-	2.0
CCINl	3.0 [3, 1]	-	-1.0 [5, 7]	-	-2.0 [1, 6]	-	7.0
LINl	13.0 [1, 6]	-	-	-	-1.0 [2, 4]	-	5.0
MNl	1.0 [3, 9]	-	-	-	-2.0 [11, 8]	-	5.0
EINr	-	-2.0 [10, 5]	-	0.4 [10, 2]	-	-	2.0
CCINr	-	-2.0 [1, 6]	-	3.0 [3, 1]	-	-1.0 [5, 7]	7.0
LINr	-	-1.0 [2, 4]	-	13.0 [1, 6]	-	-	5.0
MNr	-	-2.0 [11, 8]	-	1.0 [3, 9]	-	-	5.0





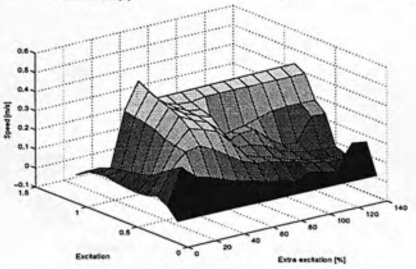


Figure E.1: **Run1:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	0.4 [5, 3]	-	-	-	-2.0 [1, 0]	-	2.0
CCINl	3.0 [11, 2]	-	-1.0 [3, 9]	-	-2.0 [0, 5]	-	7.0
LINl	13.0 [4, 1]	-	-	-	-1.0 [9, 4]	-	5.0
MNl	1.0 [4, 1]	-	-	-	-2.0 [11, 11]	-	5.0
EINr	-	-2.0 [1, 0]	-	0.4 [5, 3]	-	-	2.0
CCINr	-	-2.0 [0, 5]	-	3.0 [11, 2]	-	-1.0 [3, 9]	7.0
LINr	-	-1.0 [9, 4]	-	13.0 [4, 1]	-	-	5.0
MNr	-	-2.0 [11, 11]	-	1.0 [4, 1]	-	-	5.0

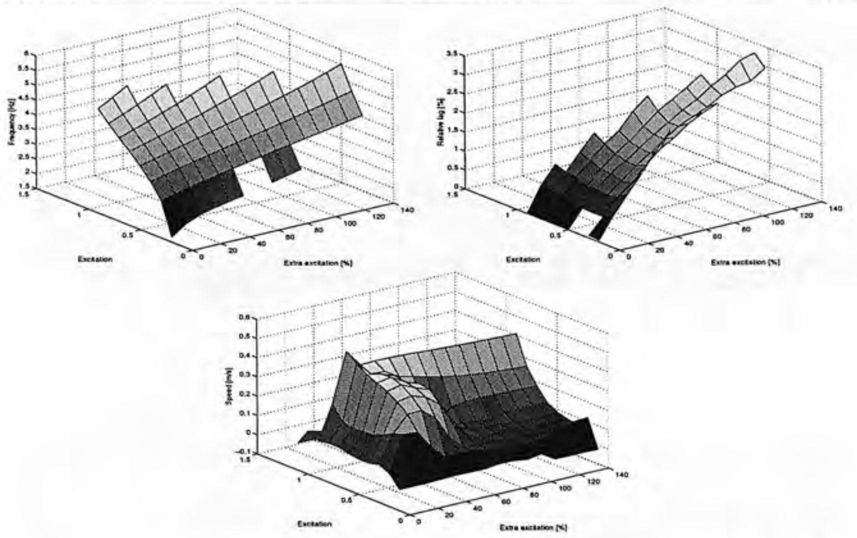


Figure E.2: **Run2:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	0.4 [1, 0]	-	-	-	-2.0 [9, 5]	-	2.0
CCINl	3.0 [2, 2]	-	-1.0 [5, 8]	-	-2.0 [2, 7]	-	7.0
LINl	13.0 [3, 6]	-	-	-	-1.0 [11, 11]	-	5.0
MNl	1.0 [5, 10]	-	-	-	-2.0 [4, 4]	-	5.0
EINr	-	-2.0 [9, 5]	-	0.4 [1, 0]	-	-	2.0
CCINr	-	-2.0 [2, 7]	-	3.0 [2, 2]	-	-1.0 [5, 8]	7.0
LINr	-	-1.0 [11, 11]	-	13.0 [3, 6]	-	-	5.0
MNr	-	-2.0 [4, 4]	-	1.0 [5, 10]	-	-	5.0

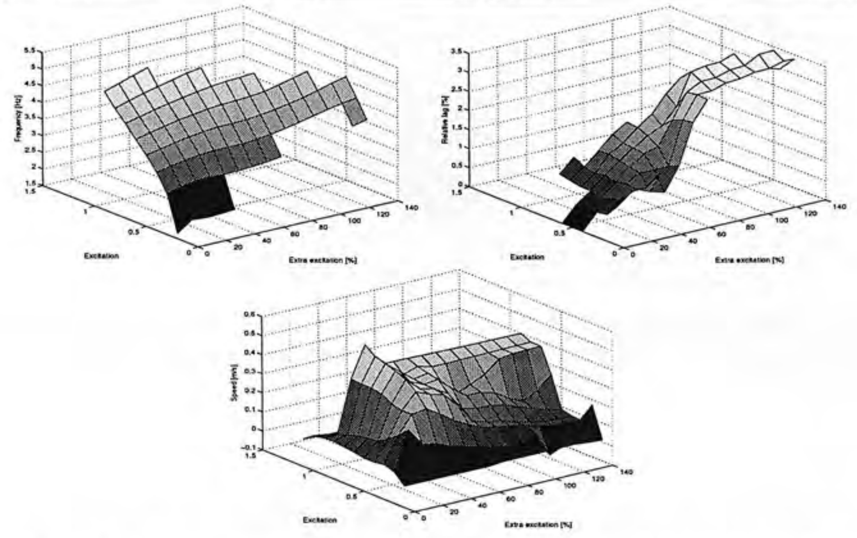


Figure E.3: **Run3:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	0.4 [11, 5]	-	-	-	-2.0 [8, 3]	-	2.0
CCINl	3.0 [12, 8]	-	-1.0 [1, 3]	-	-2.0 [0, 5]	-	7.0
LINl	13.0 [4, 7]	-	-	-	-1.0 [5, 8]	-	5.0
MNl	1.0 [2, 10]	-	-	-	-2.0 [8, 1]	-	5.0
EINr	-	-2.0 [8, 3]	-	0.4 [11, 5]	-	-	2.0
CCINr	-	-2.0 [0, 5]	-	3.0 [12, 8]	-	-1.0 [1, 3]	7.0
LINr	-	-1.0 [5, 8]	-	13.0 [4, 7]	-	-	5.0
MNr	-	-2.0 [8, 1]	-	1.0 [2, 10]	-	-	5.0

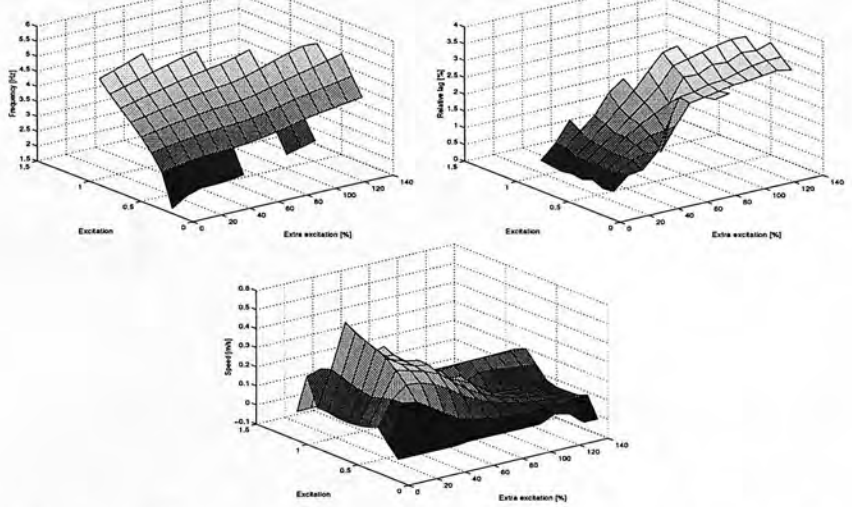


Figure E.4: **Run4:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

	EINl	CCINl	LINl	EINr	CCINr	LINr	BS
EINl	0.4 [5, 2]	-	-	-	-2.0 [4, 2]	-	2.0
CCINl	3.0 [11, 6]	-	-1.0 [7, 7]	-	-2.0 [5, 8]	-	7.0
LINl	13.0 [4, 8]	-	-	-	-1.0 [4, 2]	-	5.0
MNl	1.0 [11, 1]	-	-	-	-2.0 [5, 6]	-	5.0
EINr	-	-2.0 [4, 2]	-	0.4 [5, 2]	-	-	2.0
CCINr	-	-2.0 [5, 8]	-	3.0 [11, 6]	-	-1.0 [7, 7]	7.0
LINr	-	-1.0 [4, 2]	-	13.0 [4, 8]	-	-	5.0
MNr	-	-2.0 [5, 6]	-	1.0 [11, 1]	-	-	5.0

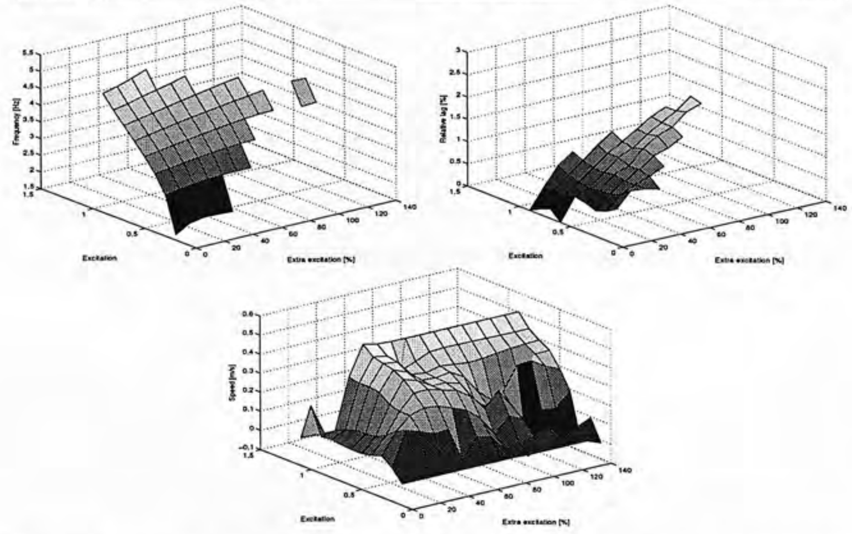


Figure E.5: **Run5:** *Top:* Neural configuration, *Middle:* Effect of the global and extra excitations on the frequency and the relative phase lag, *Bottom: idem* on the speed.

Appendix F

Results of chapter 5: sensory feedback

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	0.4	-	-	-	-2.0	-	-0.4	1.2	2.0
CCINl	3.0	-	-1.0	-	-2.0	-	0.1	-0.2	7.0
LINl	13.0	-	-	-	-1.0	-	4.7	4.7	5.0
MNl	1.0	-	-	-	-2.0	-	4.1	4.0	5.0
EINr	-	-2.0	-	0.4	-	-	1.2	-0.4	2.0
CCINr	-	-2.0	-	3.0	-	-1.0	-0.2	0.1	7.0
LINr	-	-1.0	-	13.0	-	-	4.7	4.7	5.0
MNr	-	-2.0	-	1.0	-	-	4.0	4.1	5.0

Table F.1: **Biological connectivity, sensory feedback**, best of run1

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	0.4	-	-	-	-2.0	-	-0.4	-1.2	2.0
CCINl	3.0	-	-1.0	-	-2.0	-	3.3	-2.0	7.0
LINl	13.0	-	-	-	-1.0	-	1.5	3.6	5.0
MNl	1.0	-	-	-	-2.0	-	0.3	4.3	5.0
EINr	-	-2.0	-	0.4	-	-	-1.2	-0.4	2.0
CCINr	-	-2.0	-	3.0	-	-1.0	-2.0	3.3	7.0
LINr	-	-1.0	-	13.0	-	-	3.6	1.5	5.0
MNr	-	-2.0	-	1.0	-	-	4.3	0.3	5.0

Table F.2: **Biological connectivity, sensory feedback**, best of run2

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	0.4	-	-	-	-2.0	-	-0.8	1.6	2.0
CCINl	3.0	-	-1.0	-	-2.0	-	0.9	-0.9	7.0
LINl	13.0	-	-	-	-1.0	-	0.1	3.2	5.0
MNl	1.0	-	-	-	-2.0	-	4.5	3.5	5.0
EINr	-	-2.0	-	0.4	-	-	1.6	-0.8	2.0
CCINr	-	-2.0	-	3.0	-	-1.0	-0.9	0.9	7.0
LINr	-	-1.0	-	13.0	-	-	3.2	0.1	5.0
MNr	-	-2.0	-	1.0	-	-	3.5	4.5	5.0

Table F.3: **Biological connectivity, sensory feedback**, best of run3

	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	0.4	-	-	-	-2.0	-	-0.4	2.5	2.0
CCINl	3.0	-	-1.0	-	-2.0	-	3.1	4.0	7.0
LINl	13.0	-	-	-	-1.0	-	1.5	1.9	5.0
MNl	1.0	-	-	-	-2.0	-	3.2	1.7	5.0
EINr	-	-2.0	-	0.4	-	-	2.5	-0.4	2.0
CCINr	-	-2.0	-	3.0	-	-1.0	4.0	3.1	7.0
LINr	-	-1.0	-	13.0	-	-	1.9	1.5	5.0
MNr	-	-2.0	-	1.0	-	-	1.7	3.2	5.0

Table F.4: **Biological connectivity, sensory feedback**, best of run4

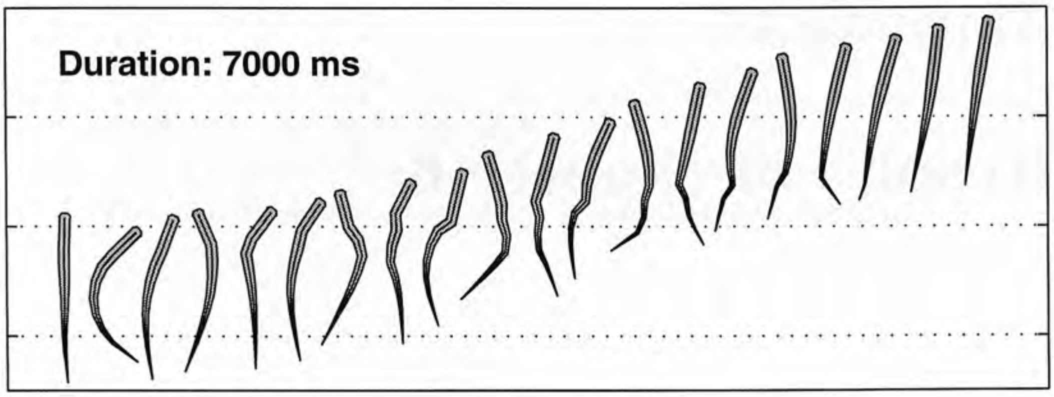
	EINl	CCINl	LINl	EINr	CCINr	LINr	ECl	ECr	BS
EINl	0.4	-	-	-	-2.0	-	-1.0	5.0	2.0
CCINl	3.0	-	-1.0	-	-2.0	-	2.4	-0.9	7.0
LINl	13.0	-	-	-	-1.0	-	0.8	-1.0	5.0
MNl	1.0	-	-	-	-2.0	-	3.9	3.4	5.0
EINr	-	-2.0	-	0.4	-	-	5.0	-1.0	2.0
CCINr	-	-2.0	-	3.0	-	-1.0	-0.9	2.4	7.0
LINr	-	-1.0	-	13.0	-	-	-1.0	0.8	5.0
MNr	-	-2.0	-	1.0	-	-	3.4	3.9	5.0

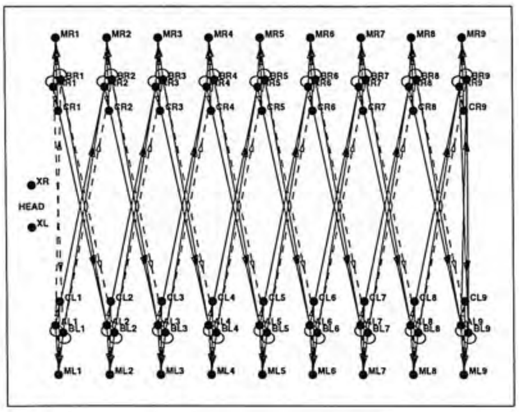
Table F.5: **Biological connectivity, sensory feedback**, best of run5

Appendix G

Results of chapter 6

Controller 1





Parameters:		Weights:					
Tau:		Segment 1		Segment 5		Segment 9	
A:	162.4	AL1 → ML1	32.05	AL5 → ML5	32.05	AL9 → ML9	32.05
B:	678.6	CL1 → ML1	32.05	CL5 → ML5	32.05	CL9 → ML9	32.05
C:	383.0	CL1 → ML1	32.05	CL5 → ML5	32.05	CL9 → ML9	32.05
		CL1 → ML1	15.95	CL5 → ML5	15.95	CL9 → ML9	15.95
Bias:		BL1 → ML1	32.05	BL5 → ML5	32.05	BL9 → ML9	32.05
A:	1.35	XL → AL1	-12.75	XL → AL5	-12.75	XL → AL9	-12.75
B:	2.75	AL1 → AL1	12.05	AR4 → AL5	27.70	AR8 → AL9	27.70
C:	2.00	XL → BL1	-12.15	BR4 → AL5	21.30	BR8 → AL9	21.30
		AR1 → BL1	-5.00	AL5 → AL5	12.05	AL9 → AL9	12.05
		BL1 → BL1	12.05	XL → BL5	-12.15	XL → BL9	-12.15
		XL → CL1	1.00	BR4 → BL5	-5.00	AR9 → BL9	27.70
		AL1 → CL1	-10.05	BL5 → BL5	12.05	BR9 → BL9	21.30
				XL → CL5	1.00	BR8 → BL9	-5.00
				AL5 → CL5	-10.05	BL9 → BL9	12.05
						XL → CL9	1.00
						AL9 → CL9	-10.05

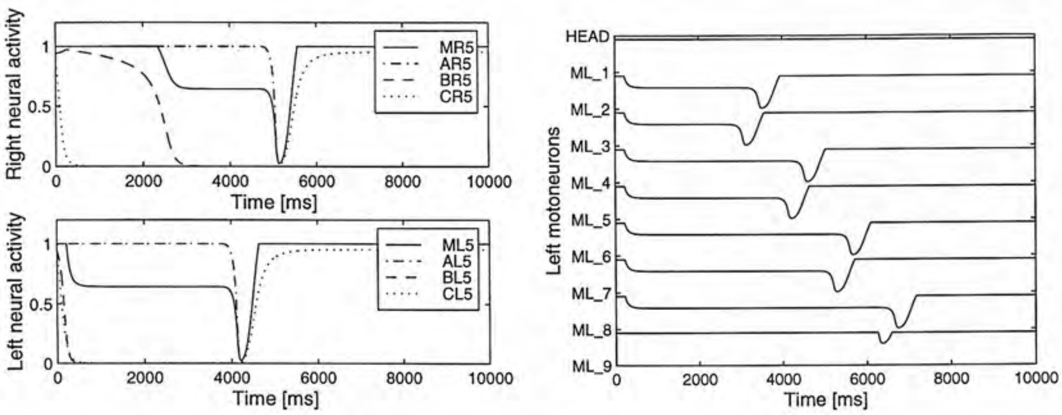
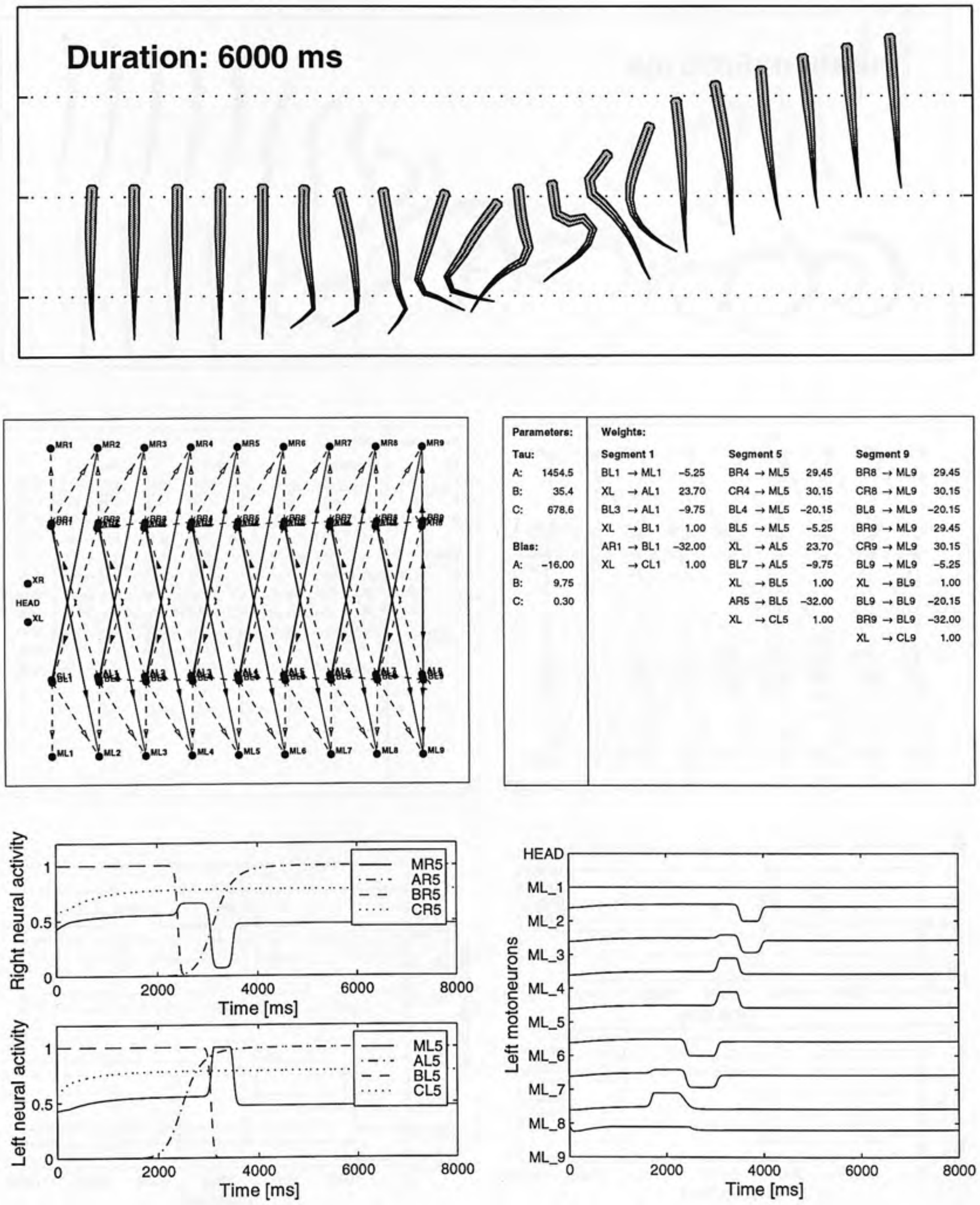


Figure G.1: **Run1.** *Top:* Swimming, *Middle:* Neural configuration, *Bottom:* Neural activity.

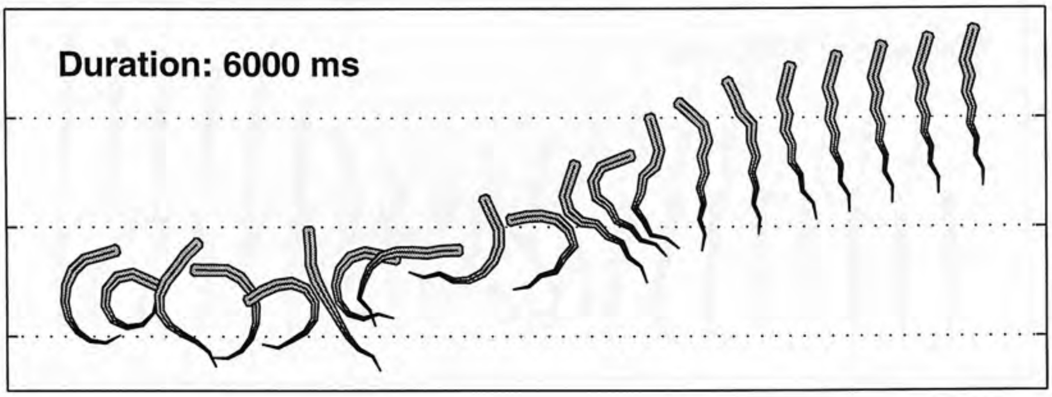
Controller 2

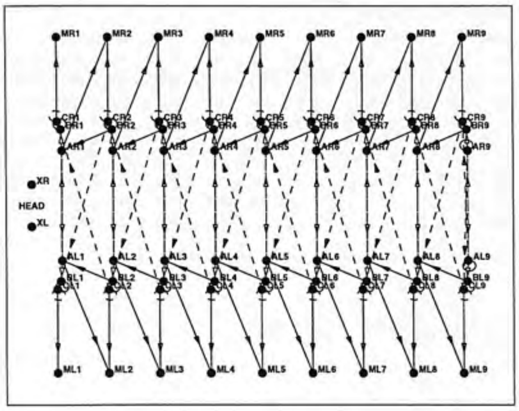


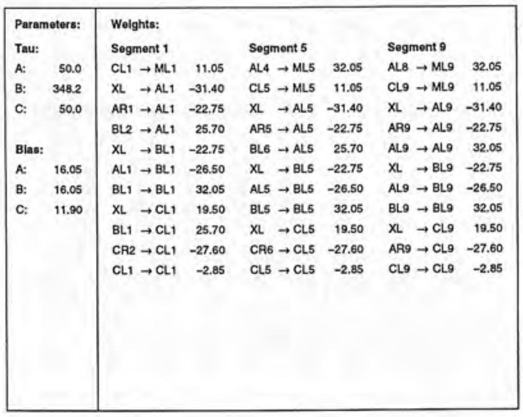
Parameters:		Weights:								
Tau:		Segment 1			Segment 5			Segment 9		
A:	1454.5	BL1	→ ML1	-5.25	BR4	→ ML5	29.45	BR8	→ ML9	29.45
B:	35.4	XL	→ AL1	23.70	CR4	→ ML5	30.15	CR8	→ ML9	30.15
C:	678.6	BL3	→ AL1	-9.75	BL4	→ ML5	-20.15	BL8	→ ML9	-20.15
		XL	→ BL1	1.00	BL5	→ ML5	-5.25	BR9	→ ML9	29.45
Bias:		AR1	→ BL1	-32.00	XL	→ AL5	23.70	CR9	→ ML9	30.15
A:	-16.00	XL	→ CL1	1.00	BL7	→ AL5	-9.75	BL9	→ ML9	-5.25
B:	9.75				XL	→ BL5	1.00	XL	→ BL9	1.00
C:	0.30				AR5	→ BL5	-32.00	BL9	→ BL9	-20.15
					XL	→ CL5	1.00	BR9	→ BL9	-32.00
								XL	→ CL9	1.00

Figure G.2: **Run2:** *Top:* Swimming, *Middle:* Neural configuration, *Bottom:* Neural activity.

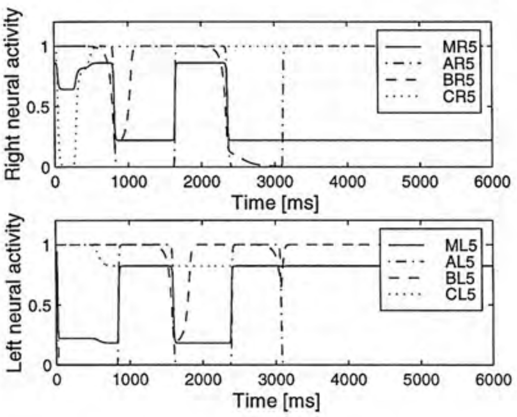
Controller 3







Parameters:		Weights:					
Tau:		Segment 1		Segment 5		Segment 9	
A:	50.0	CL1 → ML1	11.05	AL4 → ML5	32.05	AL8 → ML9	32.05
B:	348.2	XL → AL1	-31.40	CL5 → ML5	11.05	CL9 → ML9	11.05
C:	50.0	AR1 → AL1	-22.75	XL → AL5	-31.40	XL → AL9	-31.40
		BL2 → AL1	25.70	AR5 → AL5	-22.75	AR9 → AL9	-22.75
Bias:		XL → BL1	-22.75	BL6 → AL5	25.70	AL9 → AL9	32.05
A:	16.05	AL1 → BL1	-26.50	XL → BL5	-22.75	XL → BL9	-22.75
B:	16.05	BL1 → BL1	32.05	AL5 → BL5	-26.50	AL9 → BL9	-26.50
C:	11.90	XL → CL1	19.50	BL5 → BL5	32.05	BL9 → BL9	32.05
		BL1 → CL1	25.70	XL → CL5	19.50	XL → CL9	19.50
		CR2 → CL1	-27.60	CR6 → CL5	-27.60	AR9 → CL9	-27.60
		CL1 → CL1	-2.85	CL5 → CL5	-2.85	CL9 → CL9	-2.85



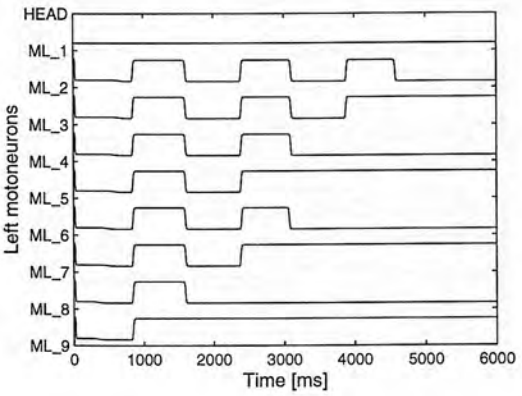
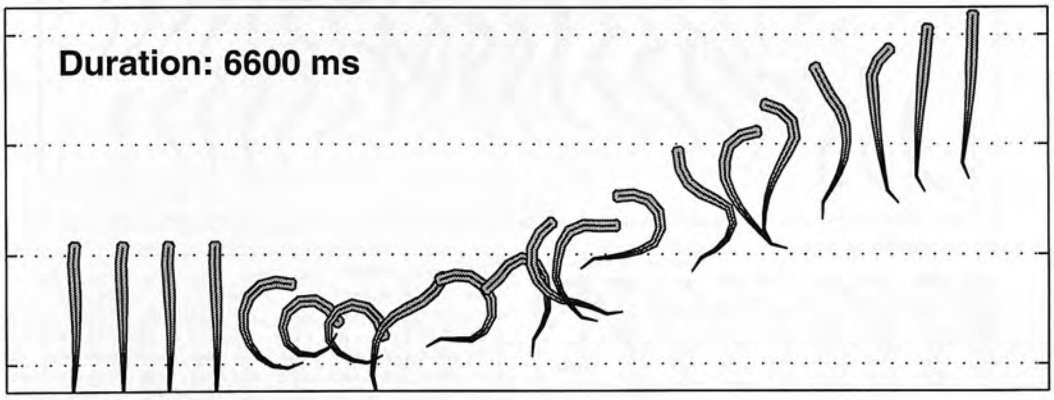
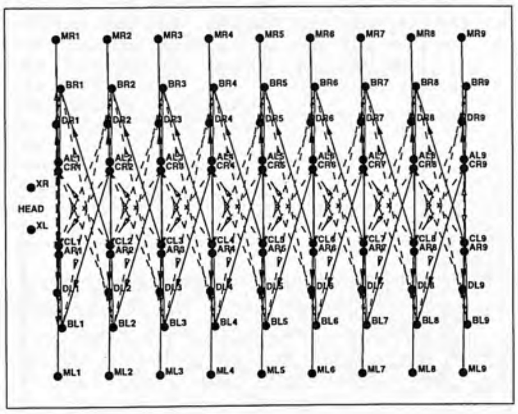


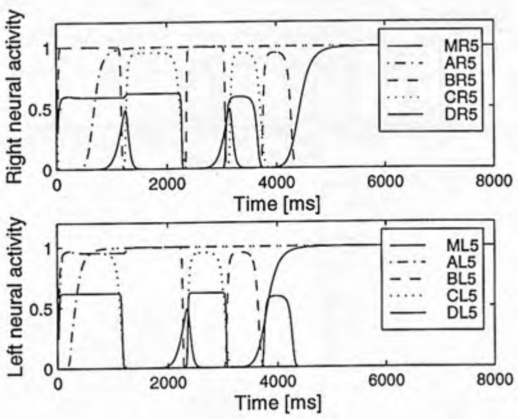
Figure G.3: **Run3:** *Top:* Swimming, *Middle:* Neural configuration, *Bottom:* Neural activity.

Controller 4




MR1 MR2 MR3 MR4 MR5 MR6 MR7 MR8 MR9
BR1 BR2 BR3 BR4 BR5 BR6 BR7 BR8 BR9
XR
HEAD
XL
BL1 BL2 BL3 BL4 BL5 BL6 BL7 BL8 BL9
ML1 ML2 ML3 ML4 ML5 ML6 ML7 ML8 ML9

Parameters:		Weights: Segment 1		Segment 5		Segment 9	
Tau:							
A:	421.3	CR1 → ML1	10.75	BR5 → ML5	30.90	BR9 → ML9	30.90
B:	50.0	BR1 → ML1	30.90	XL → AL5	22.20	XL → AL9	22.20
C:	50.0	XL → AL1	22.20	XL → BL5	1.00	XL → BL9	1.00
D:	746.4	XL → BL1	1.00	DL5 → BL5	-16.05	DL9 → BL9	-16.05
		DL1 → BL1	-16.05	DL5 → BL5	-18.55	DL9 → BL9	-18.55
Bias:		DL1 → BL1	-18.55	DL5 → BL5	0.00	DL9 → BL9	0.00
A:	-16.00	DL1 → BL1	0.00	CR6 → BL5	10.75	XL → CL9	1.00
B:	2.00	CR2 → BL1	10.75	BR6 → BL5	-23.95	CR9 → CL9	-26.80
C:	2.00	BR2 → BL1	-23.95	XL → CL5	1.00	XL → DL9	1.00
D:	-16.00	XL → CL1	1.00	CR6 → CL5	-26.80	AR9 → DL9	22.20
		CR2 → CL1	-26.80	DR6 → CL5	-18.00	CL9 → DL9	-32.00
		DR2 → CL1	-18.00	XL → DL5	1.00		
		XL → DL1	1.00	AR5 → DL5	22.20		
		AR1 → DL1	22.20	CL5 → DL5	-32.00		
		BR1 → DL1	-23.95				
		DR1 → DL1	-18.00				
		CL1 → DL1	-32.00				



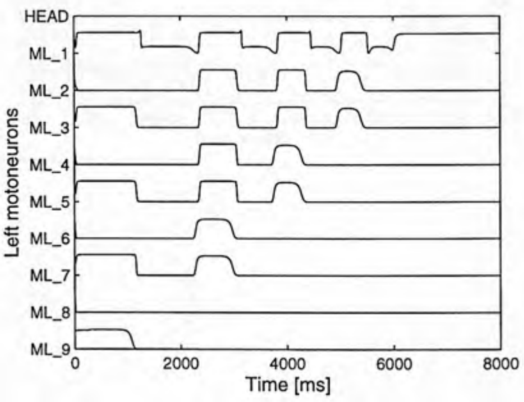
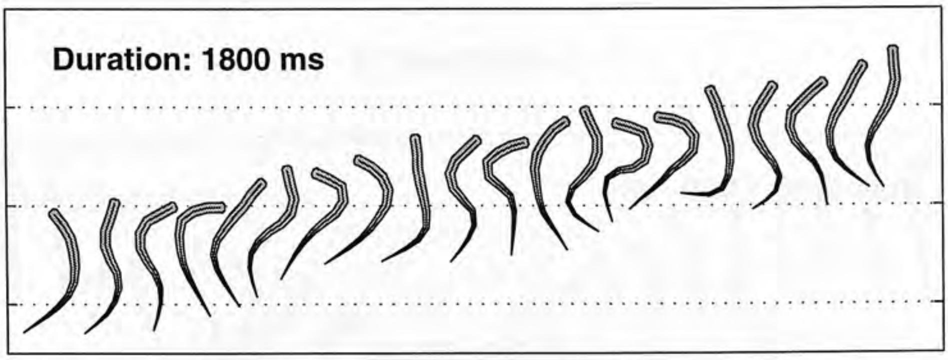
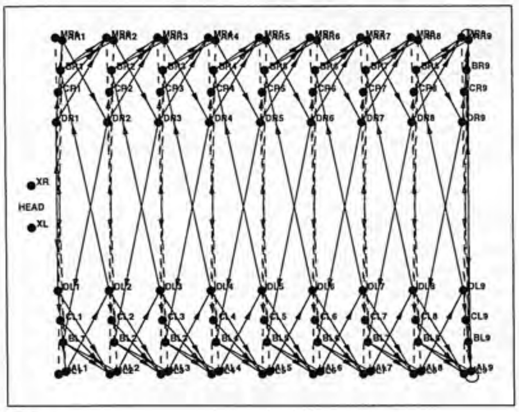


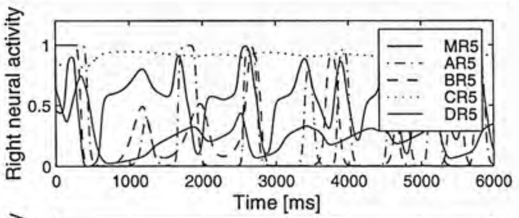
Figure G.4: **Run4:** *Top:* Swimming, *Middle:* Neural configuration, *Bottom:* Neural activity.

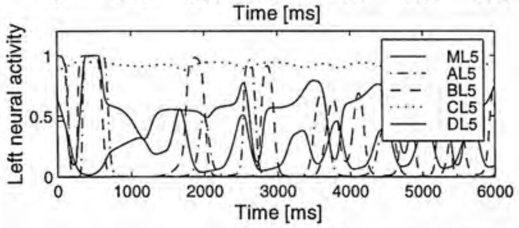
Controller 5

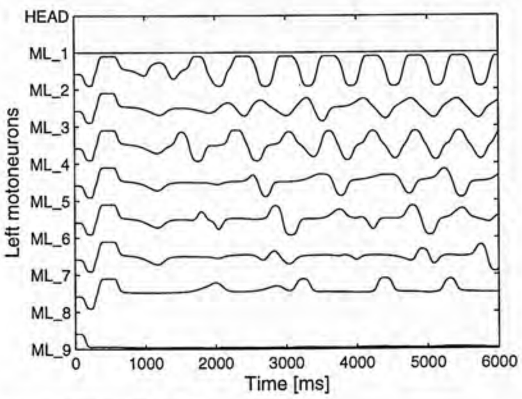




Parameters:		Weights: Segment 1		Segment 5		Segment 9	
Tau:		BR1 → ML1	-26.25	CL4 → ML5	31.30	CL8 → ML9	31.30
A:	91.7	XL → AL1	-17.90	BL4 → ML5	21.55	BL8 → ML9	21.55
B:	122.0	DL1 → AL1	14.80	DL4 → ML5	0.05	DL8 → ML9	0.05
C:	91.7	XL → BL1	-18.75	BR5 → ML5	-26.25	BR9 → ML9	-26.25
D:	1600.0	DR2 → BL1	25.65	XL → AL5	-17.90	XL → AL9	-17.90
Bias:		DR2 → BL1	4.15	BL4 → AL5	14.80	AL9 → AL9	14.85
A:	10.05	XL → CL1	1.00	XL → BL5	-18.75	BL8 → AL9	14.80
B:	8.40	DL1 → CL1	-2.05	DR6 → BL5	25.65	CL9 → AL9	31.30
C:	2.00	XL → DL1	1.00	DR6 → BL5	4.15	BL9 → AL9	21.55
D:	0.60	AR1 → DL1	-30.15	XL → CL5	1.00	DL9 → AL9	0.05
		DR1 → DL1	4.15	DL5 → CL5	-2.05	XL → BL9	-18.75
				XL → DL5	1.00	AR9 → BL9	25.65
				AL4 → DL5	14.85	XL → CL9	1.00
				AR5 → DL5	-30.15	DL9 → CL9	-2.05
						XL → DL9	1.00
						AL8 → DL9	14.85
						AR9 → DL9	-30.15







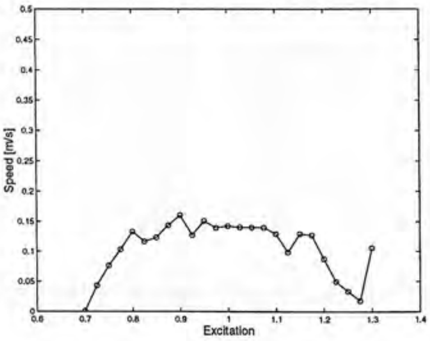
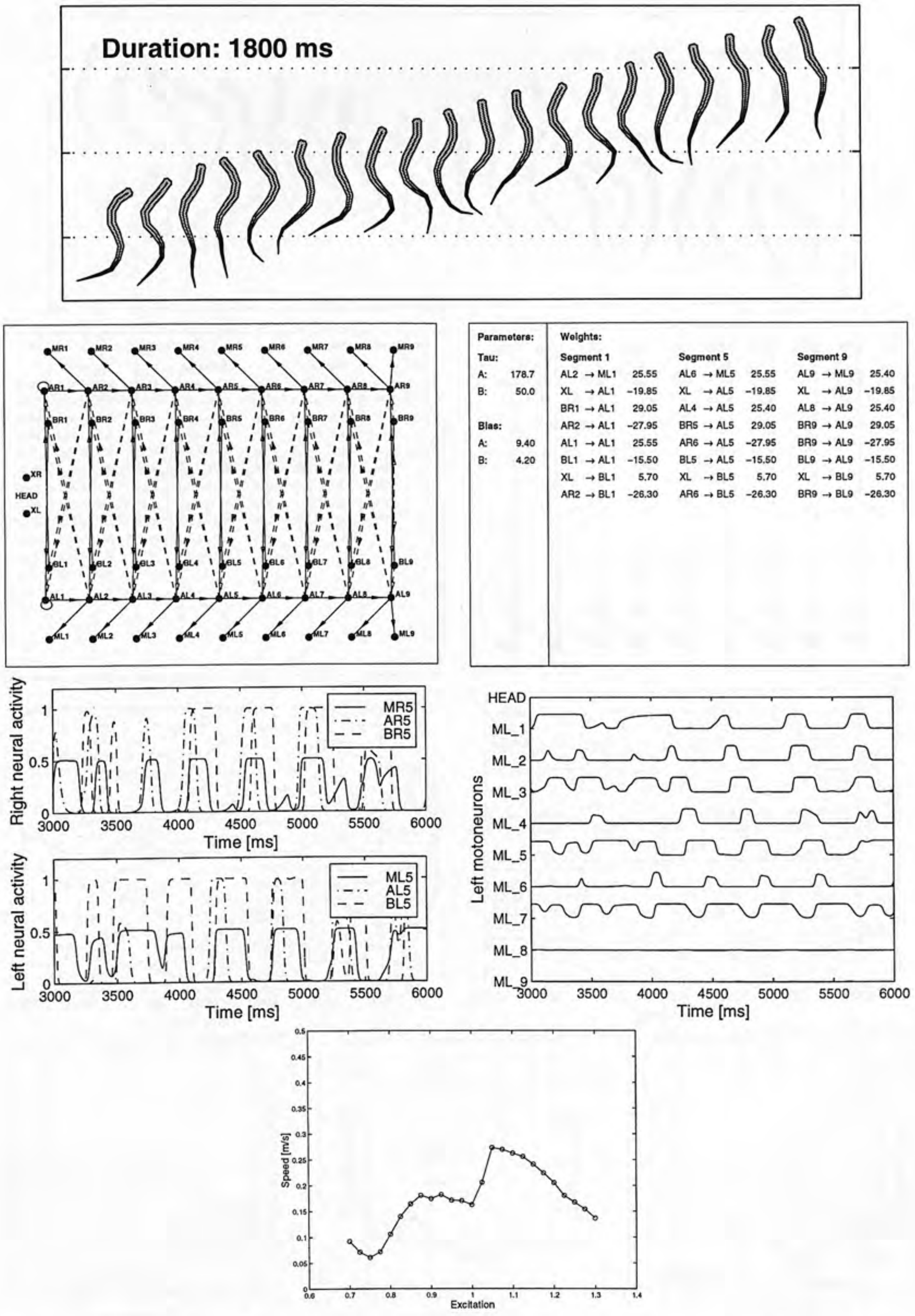


Figure G.5: **Run5:** *Top:* Swimming, *Middle top:* Neural configuration, *Middle bottom:* Neural activity, *Bottom:* Influence of the tonic input on the speed of swimming.

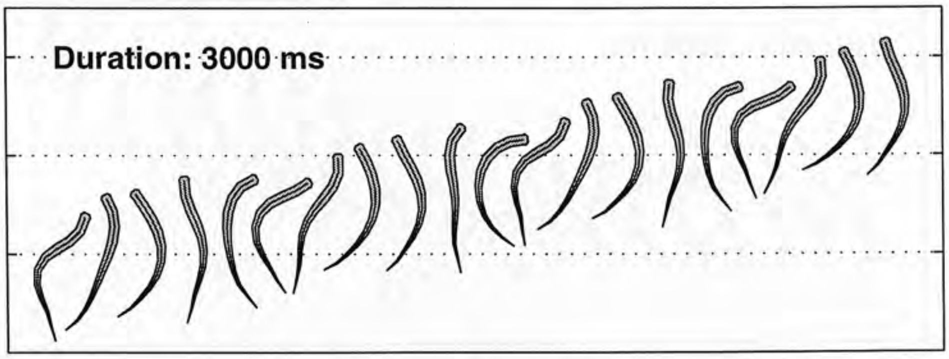
Controller 6

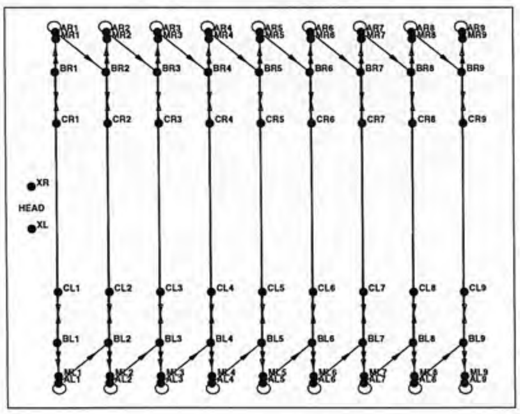


Parameters:		Weights:								
Tau:		Segment 1			Segment 5			Segment 9		
A:	178.7	AL2	→ ML1	25.55	AL6	→ ML5	25.55	AL9	→ ML9	25.40
B:	50.0	XL	→ AL1	-19.85	XL	→ AL5	-19.85	XL	→ AL9	-19.85
		BR1	→ AL1	29.05	AL4	→ AL5	25.40	AL8	→ AL9	25.40
Bias:		AR2	→ AL1	-27.95	BR5	→ AL5	29.05	BR9	→ AL9	29.05
A:	9.40	AL1	→ AL1	25.55	AR6	→ AL5	-27.95	BR9	→ AL9	-27.95
B:	4.20	BL1	→ AL1	-15.50	BL5	→ AL5	-15.50	BL9	→ AL9	-15.50
		XL	→ BL1	5.70	XL	→ BL5	5.70	XL	→ BL9	5.70
		AR2	→ BL1	-26.30	AR6	→ BL5	-26.30	BR9	→ BL9	-26.30

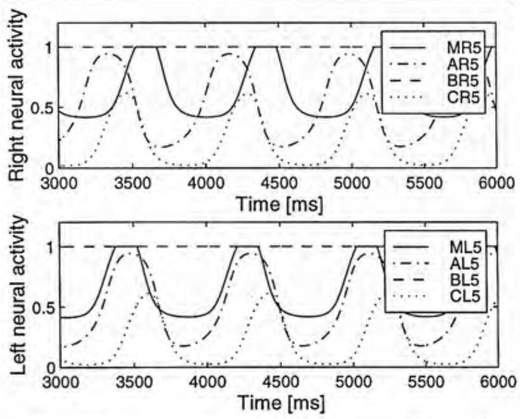
Figure G.6: **Run6:** *Top:* Swimming, *Middle top:* Neural configuration, *Middle bottom:* Neural activity, *Bottom:* Influence of the tonic input on the speed of swimming.

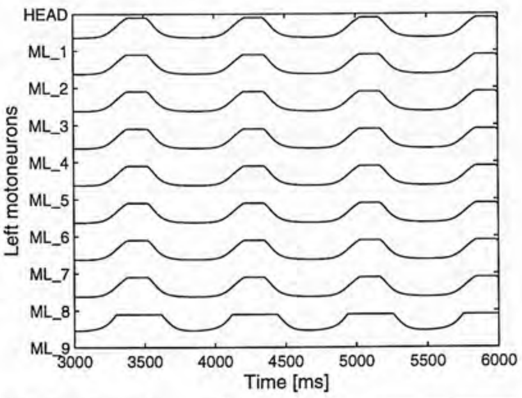
Controller 7




AR1 MR1 AR2 MR2 AR3 MR3 AR4 MR4 AR5 MR5 AR6 MR6 AR7 MR7 AR8 MR8 AR9 MR9
BR1 BR2 BR3 BR4 BR5 BR6 BR7 BR8 BR9
CR1 CR2 CR3 CR4 CR5 CR6 CR7 CR8 CR9
XR
HEAD
XL
CL1 CL2 CL3 CL4 CL5 CL6 CL7 CL8 CL9
BL1 BL2 BL3 BL4 BL5 BL6 BL7 BL8 BL9
ML1 AL1 ML2 AL2 ML3 AL3 ML4 AL4 ML5 AL5 ML6 AL6 ML7 AL7 ML8 AL8 ML9 AL9

Parameters:		Weights:					
Tau:		Segment 1		Segment 5		Segment 9	
A:	421.3	BR1 → ML1	12.35	BR5 → ML5	12.35	BR9 → ML9	12.35
B:	50.0	CR1 → ML1	26.40	CR5 → ML5	26.40	CR9 → ML9	26.40
C:	147.7	CR1 → ML1	32.05	CR5 → ML5	32.05	CR9 → ML9	32.05
		BL1 → ML1	7.15	BL5 → ML5	7.15	AL9 → ML9	26.40
Bias:		XL → AL1	1.00	XL → AL5	1.00	BL9 → ML9	7.15
A:	-3.95	AL1 → AL1	19.90	AL5 → AL5	19.90	XL → AL9	1.00
B:	2.00	CL1 → AL1	-32.00	CL5 → AL5	-32.00	AL9 → AL9	19.90
C:	2.00	XL → BL1	1.00	XL → BL5	1.00	CL9 → AL9	-32.00
		XL → CL1	-7.85	AL4 → BL5	26.40	XL → BL9	1.00
		AL1 → CL1	7.75	XL → CL5	-7.85	AL8 → BL9	26.40
				AL5 → CL5	7.75	XL → CL9	-7.85
						AL9 → CL9	7.75





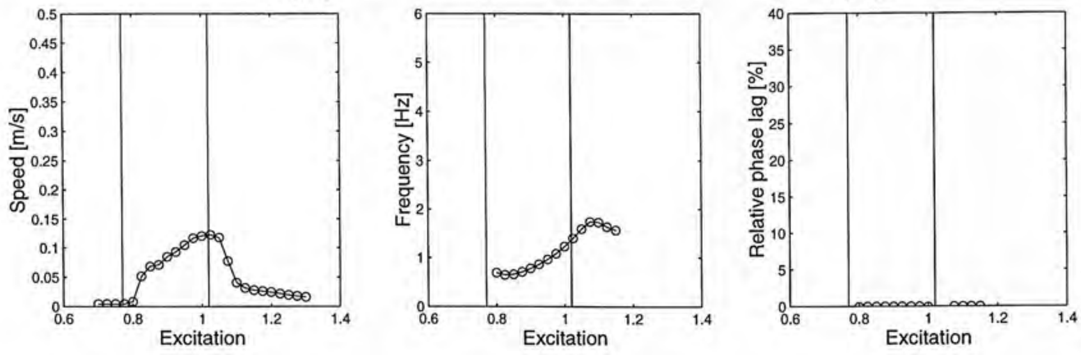
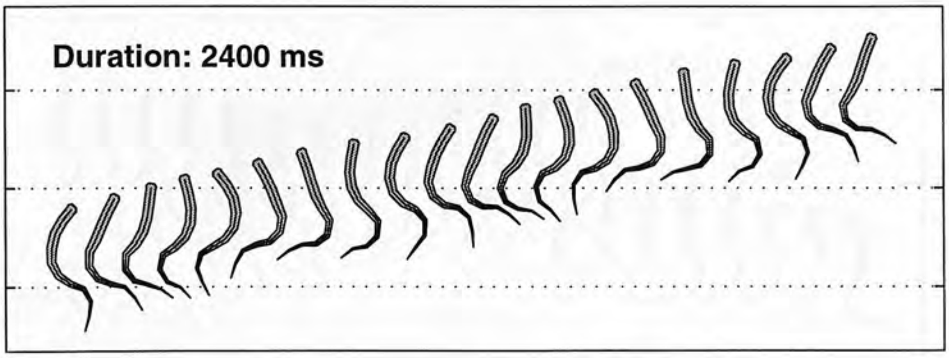
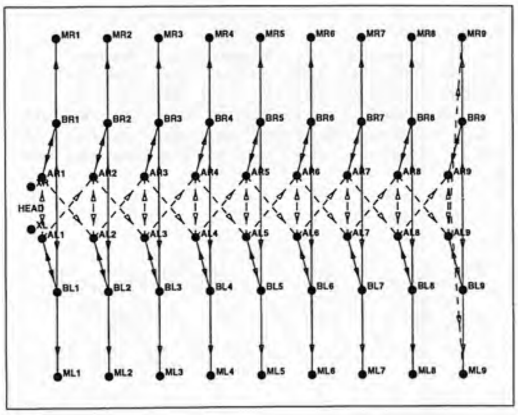


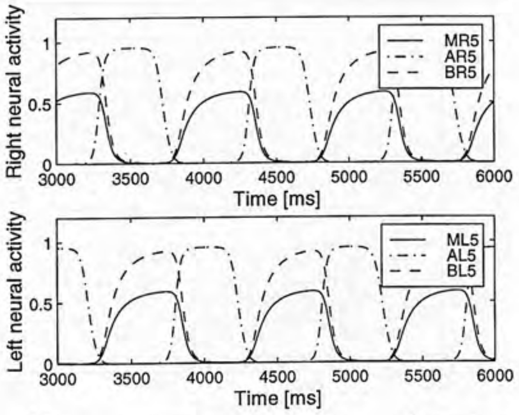
Figure G.7: **Run7:** *Top:* Swimming, *Middle top:* Neural configuration, *Middle bottom:* Neural activity, *Bottom:* Influence of the tonic input on the speed of swimming, the frequency of oscillation and the relative phase lag between segments.

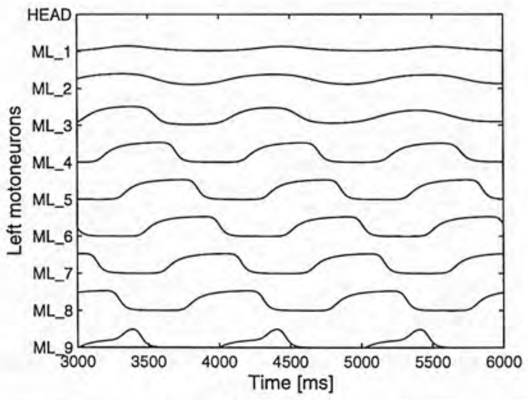
Controller 8





Parameters:		Weights: Segment 1		Segment 5		Segment 9	
Tau:		BL1 → ML1	32.05	BL5 → ML5	32.05	AR9 → ML9	-18.65
A:	383.0	XL → AL1	1.00	XL → AL5	1.00	BL9 → ML9	32.05
B:	216.2	AR1 → AL1	-15.35	AR4 → AL5	-18.65	XL → AL9	1.00
Bias:		BL1 → AL1	19.65	AR5 → AL5	-15.35	AR8 → AL9	-18.65
A:	2.00	XL → BL1	1.00	BL5 → AL5	19.65	AR9 → AL9	-15.35
B:	2.00	BR1 → BL1	4.55	XL → BL5	1.00	BL9 → AL9	19.65
		AL1 → BL1	-15.30	BR5 → BL5	4.55	XL → BL9	1.00
				AL5 → BL5	-15.30	BR9 → BL9	4.55
						AL9 → BL9	-15.30





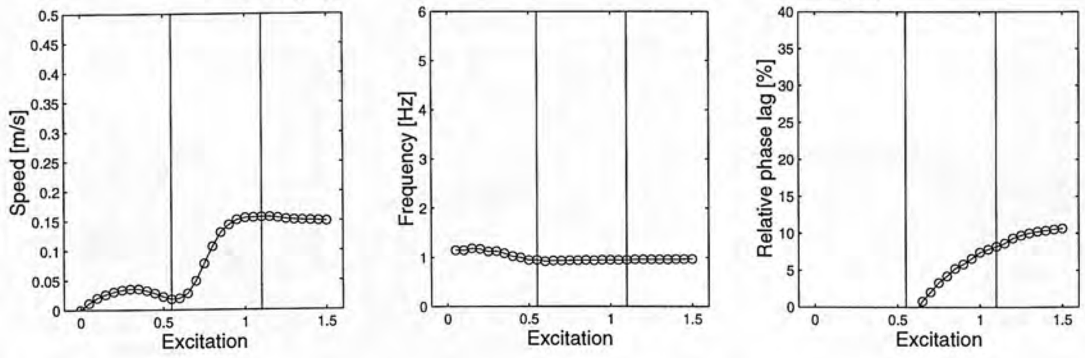
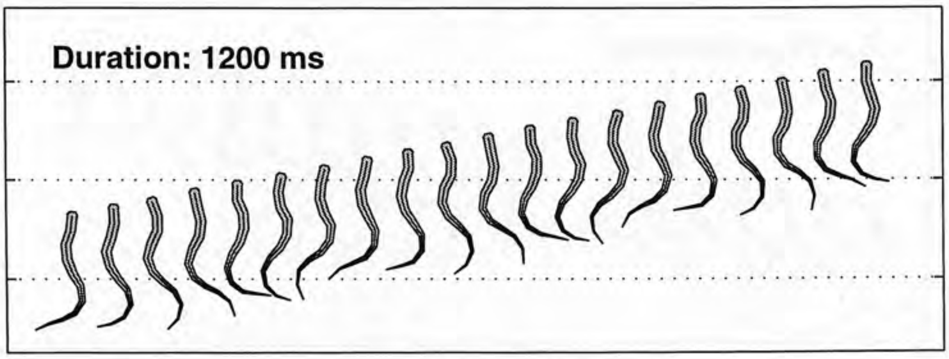
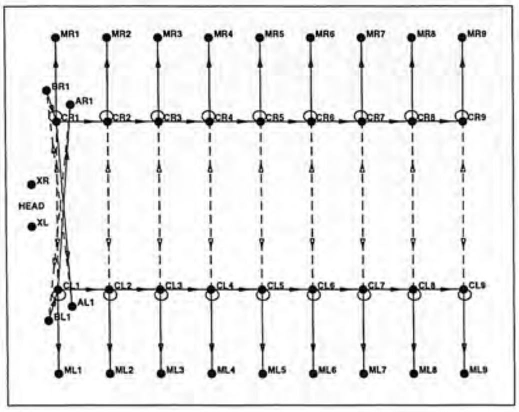
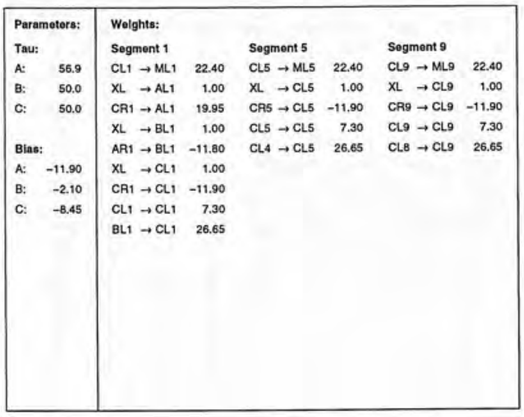


Figure G.8: **Run8:** *Top:* Swimming, *Middle top:* Neural configuration, *Middle bottom:* Neural activity, *Bottom:* Influence of the tonic input on the speed of swimming, the frequency of oscillation and the relative phase lag between segments.

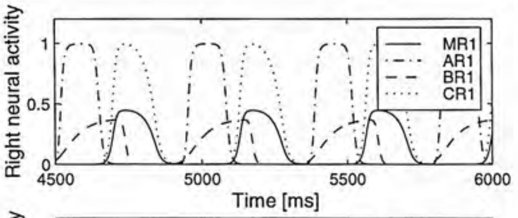
Controller 9

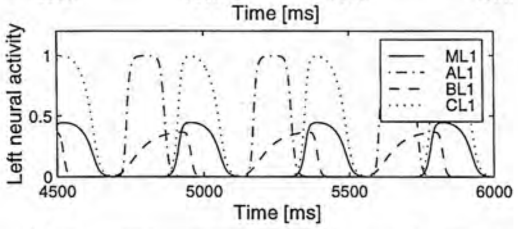


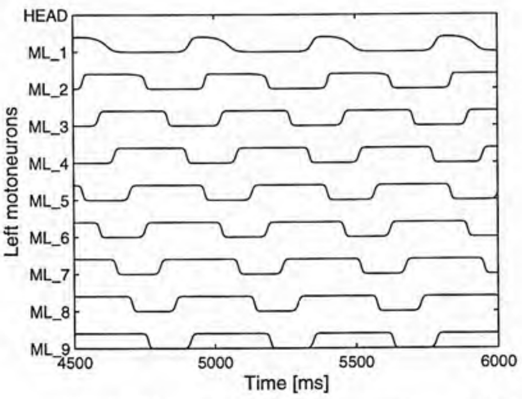


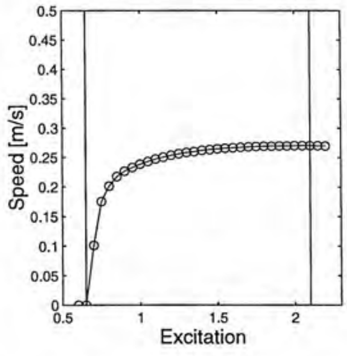


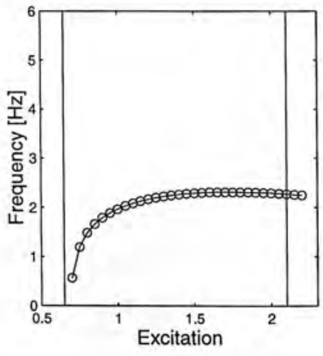
Parameters:		Weights: Segment 1		Segment 5		Segment 9	
Tau:							
A:	56.9	CL1 → ML1	22.40	CL5 → ML5	22.40	CL9 → ML9	22.40
B:	50.0	XL → AL1	1.00	XL → CL5	1.00	XL → CL9	1.00
C:	50.0	CR1 → AL1	19.95	CR5 → CL5	-11.90	CR9 → CL9	-11.90
		XL → BL1	1.00	CL5 → CL5	7.30	CL9 → CL9	7.30
Bias:		AR1 → BL1	-11.80	CL4 → CL5	26.65	CL8 → CL9	26.65
A:	-11.90	XL → CL1	1.00				
B:	-2.10	CR1 → CL1	-11.90				
C:	-8.45	CL1 → CL1	7.30				
		BL1 → CL1	26.65				











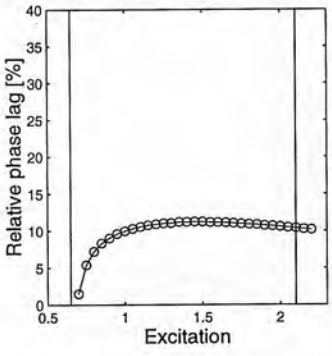
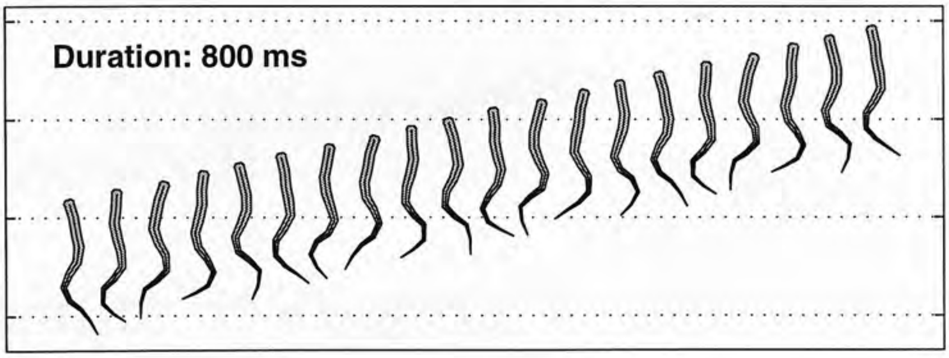
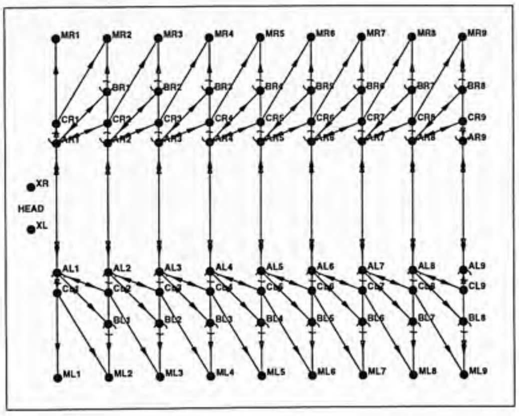
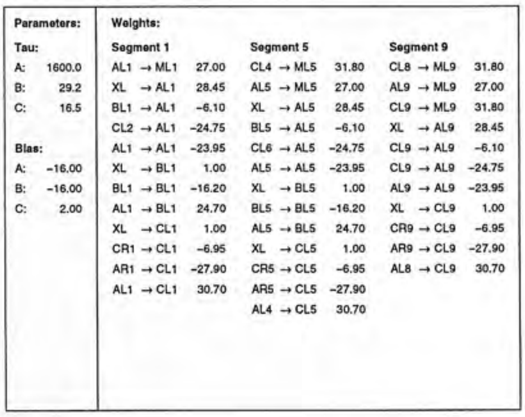


Figure G.9: **Run9:** *Top:* Swimming, *Middle top:* Neural configuration, *Middle bottom:* Neural activity, *Bottom:* Influence of the tonic input on the speed of swimming, the frequency of oscillation and the relative phase lag between segments.

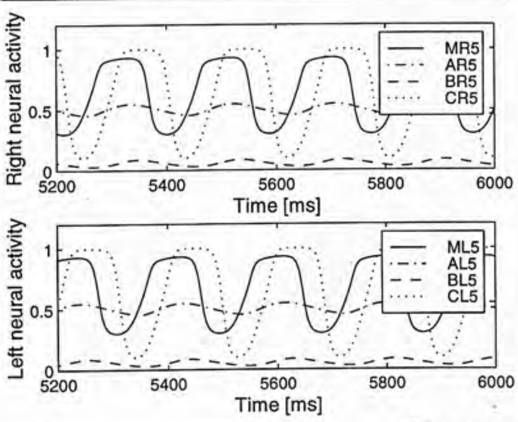
Controller 10

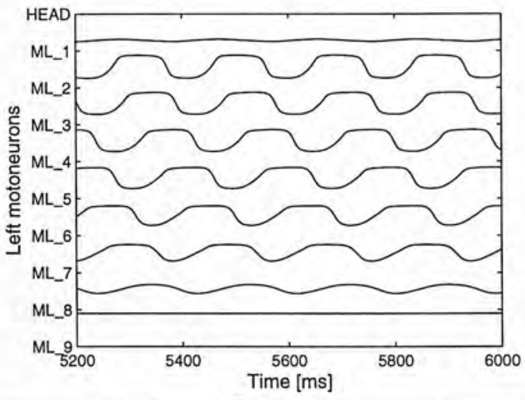


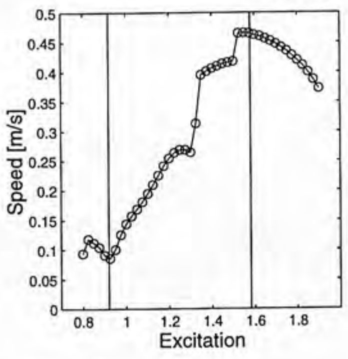


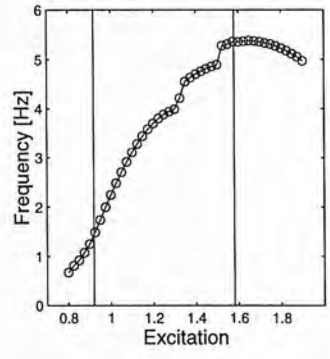


Parameters:		Weights:					
Tau:		Segment 1		Segment 5		Segment 9	
A:	1600.0	AL1 → ML1	27.00	CL4 → ML5	31.80	CL8 → ML9	31.80
B:	29.2	XL → AL1	28.45	AL5 → ML5	27.00	AL9 → ML9	27.00
C:	16.5	BL1 → AL1	-6.10	XL → AL5	28.45	CL9 → ML9	31.80
		CL2 → AL1	-24.75	BL5 → AL5	-6.10	XL → AL9	28.45
Bias:		AL1 → AL1	-23.95	CL6 → AL5	-24.75	CL9 → AL9	-6.10
A:	-16.00	XL → BL1	1.00	AL5 → AL5	-23.95	CL9 → AL9	-24.75
B:	-16.00	BL1 → BL1	-16.20	XL → BL5	1.00	AL9 → AL9	-23.95
C:	2.00	AL1 → BL1	24.70	BL5 → BL5	-16.20	XL → CL9	1.00
		XL → CL1	1.00	AL5 → BL5	24.70	CR9 → CL9	-6.95
		CR1 → CL1	-6.95	XL → CL5	1.00	AR9 → CL9	-27.90
		AR1 → CL1	-27.90	CR5 → CL5	-6.95	AL8 → CL9	30.70
		AL1 → CL1	30.70	AR5 → CL5	-27.90		
				AL4 → CL5	30.70		









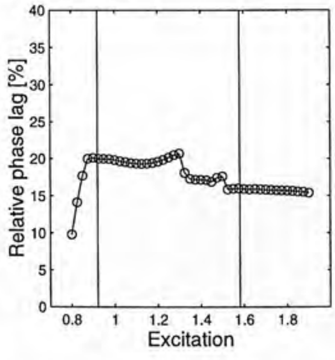


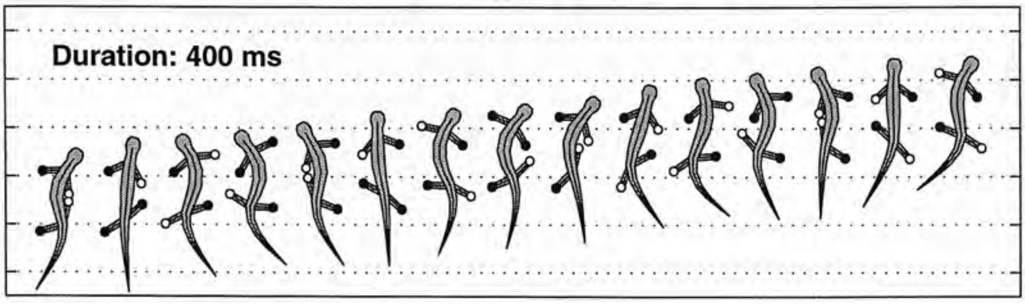
Figure G.10: **Run10:** *Top:* Swimming, *Middle top:* Neural configuration, *Middle bottom:* Neural activity, *Bottom:* Influence of the tonic input on the speed of swimming, the frequency of oscillation and the relative phase lag between segments.

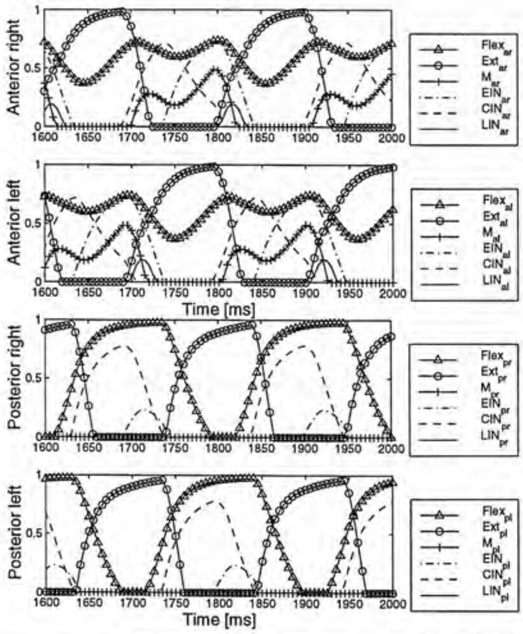
Appendix H

Results of chapter 7: Runs A1 to A10

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	–	-0.5	-0.2	6.8	-0.9	-0.5	12.4	4.1
Ext_al	3.6	-3.0	-1.4	9.6	–	-0.4	18.0	6.2
Flex_ar	6.8	-0.9	-0.5	–	-0.5	-0.2	4.1	12.4
Ext_ar	9.6	–	-0.4	3.6	-3.0	-1.4	6.2	18.0
Trunk_l	0.4	-1.0	–	8.1	-3.1	–	11.0	10.5
Trunk_r	8.1	-3.1	–	0.4	-1.0	–	10.5	11.0
E_pl	3.0	-1.4	-0.3	–	-2.0	–	–	–
C_pl	2.8	-1.7	–	0.5	-0.2	-0.4	–	–
L_pl	1.7	–	–	4.9	–	-1.7	–	–
E_pr	–	-2.0	–	3.0	-1.4	-0.3	–	–
C_pr	0.5	-0.2	-0.4	2.8	-1.7	–	–	–
L_pr	4.9	–	-1.7	1.7	–	–	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	1.3	–	–	2.5	-1.5	-3.1	7.9	16.7
Ext_pl	1.2	-3.3	–	13.2	-0.1	–	6.7	16.1
Flex_pr	2.5	-1.5	-3.1	1.3	–	–	16.7	7.9
Ext_pr	13.2	-0.1	–	1.2	-3.3	–	16.1	6.7
Tail_l	2.1	–	-0.1	0.3	-0.6	-0.4	-0.3	-0.8
Tail_r	0.3	-0.6	-0.4	2.1	–	-0.1	-0.8	-0.3
E_al	5.0	-1.8	-0.3	1.0	-0.0	-1.1	–	–
C_al	0.0	-0.2	-0.9	1.1	-1.1	-1.2	–	–
L_al	5.0	-1.1	–	1.4	–	-1.2	–	–
E_ar	1.0	-0.0	-1.1	5.0	-1.8	-0.3	–	–
C_ar	1.1	-1.1	-1.2	0.0	-0.2	-0.9	–	–
L_ar	1.4	–	-1.2	5.0	-1.1	–	–	–





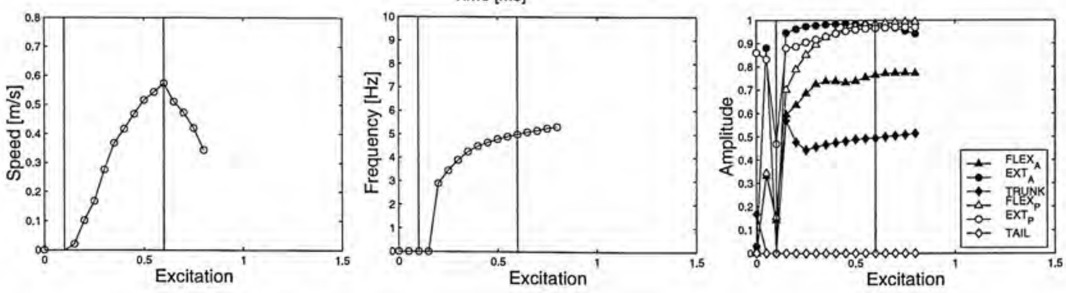
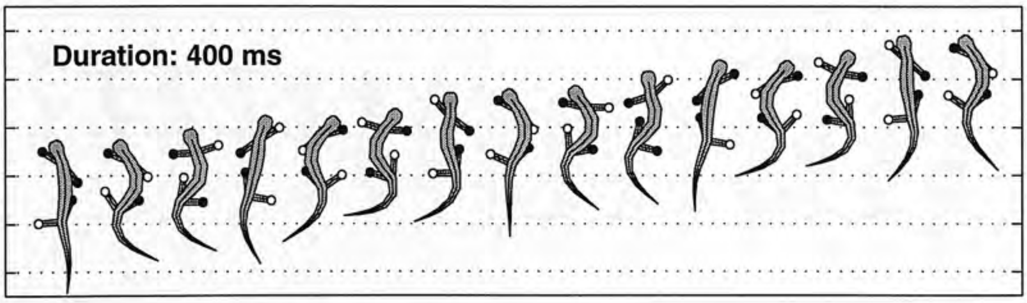
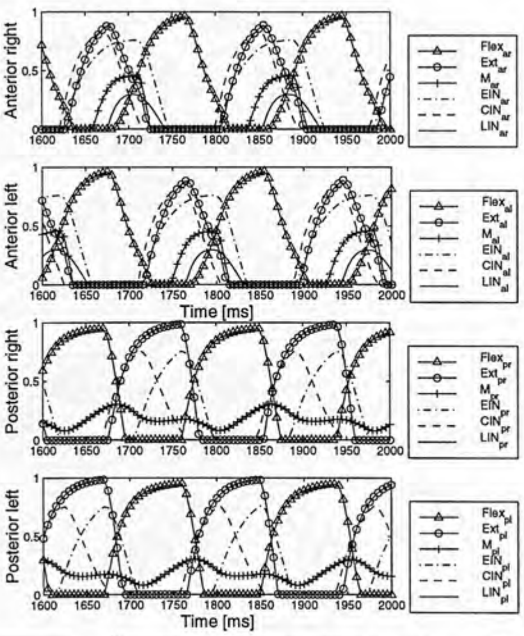


Figure H.1: **Run_A1:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	11.5	-1.2	-0.0	0.7	–	-3.4	18.8	7.5
Ext_al	–	–	-2.8	6.6	-2.8	-0.7	13.7	7.1
Flex_ar	0.7	–	-3.4	11.5	-1.2	-0.0	7.5	18.8
Ext_ar	6.6	-2.8	-0.7	–	–	-2.8	7.1	13.7
Trunk_l	2.5	–	-0.4	–	-3.2	-2.6	1.2	1.7
Trunk_r	–	-3.2	-2.6	2.5	–	-0.4	1.7	1.2
E_pl	–	-1.2	-0.1	1.0	–	-0.6	–	–
C_pl	–	-0.2	–	1.0	-1.3	-1.6	–	–
L_pl	0.2	-0.8	-1.6	1.0	-2.0	–	–	–
E_pr	1.0	–	-0.6	–	-1.2	-0.1	–	–
C_pr	1.0	-1.3	-1.6	–	-0.2	–	–	–
L_pr	1.0	-2.0	–	0.2	-0.8	-1.6	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	8.5	-3.2	-1.2	1.5	–	-4.5	7.5	6.4
Ext_pl	12.6	–	-1.0	7.9	-4.1	-3.8	17.4	8.1
Flex_pr	1.5	–	-4.5	8.5	-3.2	-1.2	6.4	7.5
Ext_pr	7.9	-4.1	-3.8	12.6	–	-1.0	8.1	17.4
Tail_l	0.7	–	-0.8	1.9	-0.1	-0.3	0.4	0.4
Tail_r	1.9	-0.1	-0.3	0.7	–	-0.8	0.4	0.4
E_al	–	-0.1	-0.2	–	–	-1.6	–	–
C_al	0.7	-1.1	-0.1	–	-0.0	-1.0	–	–
L_al	4.3	-1.1	-0.2	–	-0.4	-1.7	–	–
E_ar	–	–	-1.6	–	-0.1	-0.2	–	–
C_ar	–	-0.0	-1.0	0.7	-1.1	-0.1	–	–
L_ar	–	-0.4	-1.7	4.3	-1.1	-0.2	–	–





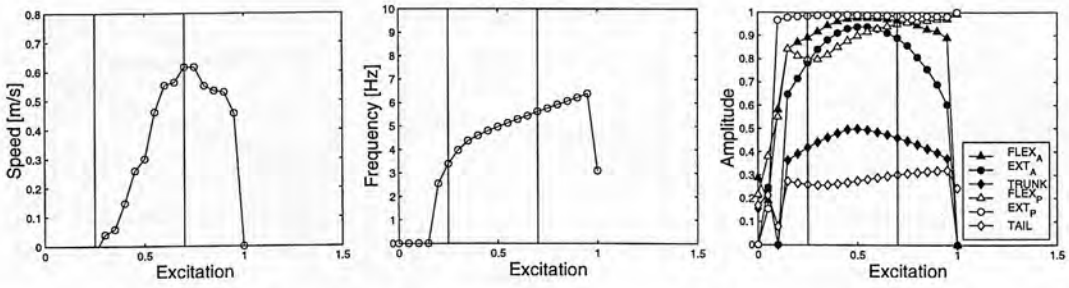
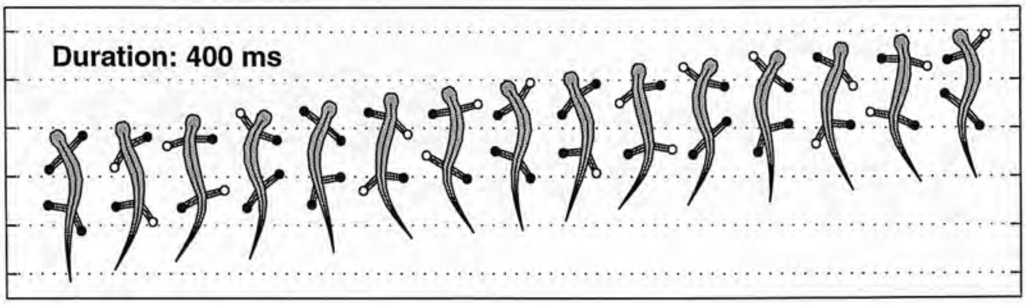
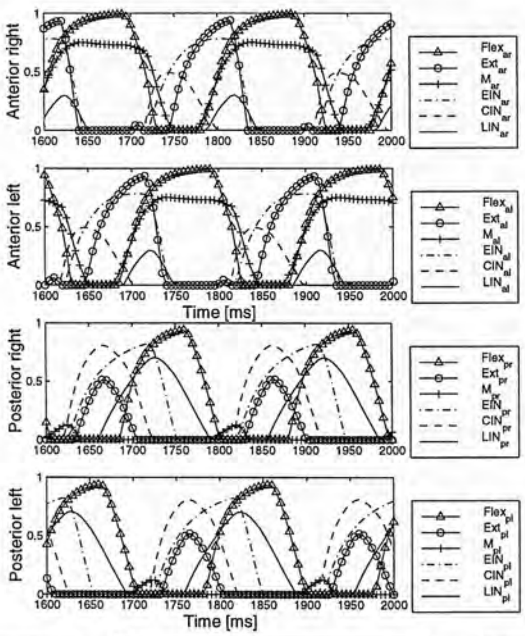


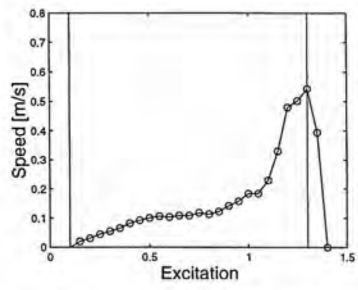
Figure H.2: **Run_A2:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

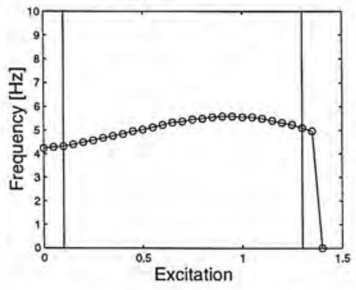
	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	–	-2.6	–	–	–	-3.5	9.5	4.9
Ext_al	11.2	-0.3	-0.1	4.3	-4.1	-4.5	2.4	6.1
Flex_ar	–	–	-3.5	–	-2.6	–	4.9	9.5
Ext_ar	4.3	-4.1	-4.5	11.2	-0.3	-0.1	6.1	2.4
Trunk_l	4.8	-2.1	–	–	-0.0	-0.4	1.3	3.4
Trunk_r	–	-0.0	-0.4	4.8	-2.1	–	3.4	1.3
E_pl	–	-0.6	-0.2	4.6	-0.2	-0.0	–	–
C_pl	2.7	–	–	5.9	–	-2.0	–	–
L_pl	0.6	-0.1	-1.4	0.3	-1.0	-0.0	–	–
E_pr	4.6	-0.2	-0.0	–	-0.6	-0.2	–	–
C_pr	5.9	–	-2.0	2.7	–	–	–	–
L_pr	0.3	-1.0	-0.0	0.6	-0.1	-1.4	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	2.4	–	-0.0	5.8	–	-2.8	4.5	2.2
Ext_pl	–	-0.0	-2.5	3.3	-2.5	-0.6	5.7	17.2
Flex_pr	5.8	–	-2.8	2.4	–	-0.0	2.2	4.5
Ext_pr	3.3	-2.5	-0.6	–	-0.0	-2.5	17.2	5.7
Tail_l	0.4	-0.7	-0.5	1.4	-0.1	-0.1	0.1	0.2
Tail_r	1.4	-0.1	-0.1	0.4	-0.7	-0.5	0.2	0.1
E_al	5.6	-1.4	-0.0	4.7	–	-0.2	–	–
C_al	1.5	-1.4	–	0.1	-0.7	-1.4	–	–
L_al	0.0	-1.0	-1.9	3.0	–	-0.6	–	–
E_ar	4.7	–	-0.2	5.6	-1.4	-0.0	–	–
C_ar	0.1	-0.7	-1.4	1.5	-1.4	–	–	–
L_ar	3.0	–	-0.6	0.0	-1.0	-1.9	–	–









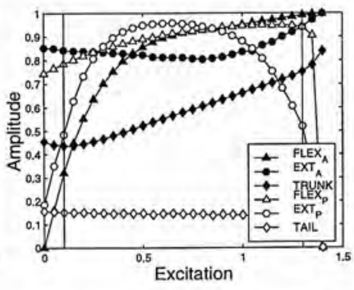
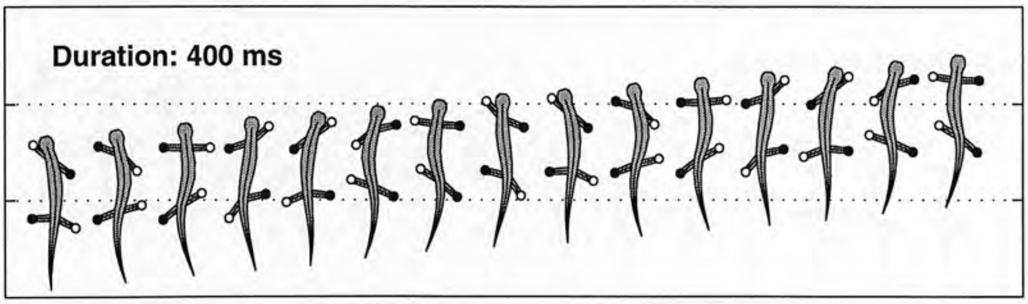
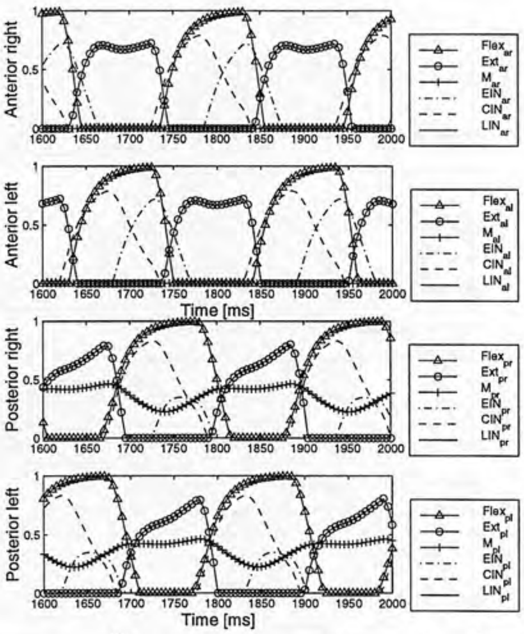


Figure H.3: **Run_A3:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	1.0	–	-1.4	4.8	-3.9	-0.8	5.7	15.5
Ext_al	14.1	-4.3	-0.1	2.6	–	-0.5	1.8	1.9
Flex_ar	4.8	-3.9	-0.8	1.0	–	-1.4	15.5	5.7
Ext_ar	2.6	–	-0.5	14.1	-4.3	-0.1	1.9	1.8
Trunk_l	4.5	-3.1	-4.7	15.0	–	–	-6.4	-3.6
Trunk_r	15.0	–	–	4.5	-3.1	-4.7	-3.6	-6.4
E_pl	–	-0.9	-0.0	2.2	-1.8	-2.0	–	–
C_pl	2.7	-1.1	-0.1	0.1	–	-1.4	–	–
L_pl	3.7	–	-2.0	0.4	–	–	–	–
E_pr	2.2	-1.8	-2.0	–	-0.9	-0.0	–	–
C_pr	0.1	–	-1.4	2.7	-1.1	-0.1	–	–
L_pr	0.4	–	–	3.7	–	-2.0	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	0.1	–	-3.0	2.7	-2.0	-0.3	21.0	7.0
Ext_pl	13.1	-4.0	-3.4	8.3	-0.4	–	16.9	8.1
Flex_pr	2.7	-2.0	-0.3	0.1	–	-3.0	7.0	21.0
Ext_pr	8.3	-0.4	–	13.1	-4.0	-3.4	8.1	16.9
Tail_l	–	-0.4	-0.4	0.1	-0.1	-0.4	2.1	1.8
Tail_r	0.1	-0.1	-0.4	–	-0.4	-0.4	1.8	2.1
E_al	0.2	-2.0	-0.5	0.3	–	-1.2	–	–
C_al	4.7	-0.2	-1.3	0.5	-1.1	-1.8	–	–
L_al	5.1	-0.8	-0.5	0.2	-1.9	–	–	–
E_ar	0.3	–	-1.2	0.2	-2.0	-0.5	–	–
C_ar	0.5	-1.1	-1.8	4.7	-0.2	-1.3	–	–
L_ar	0.2	-1.9	–	5.1	-0.8	-0.5	–	–





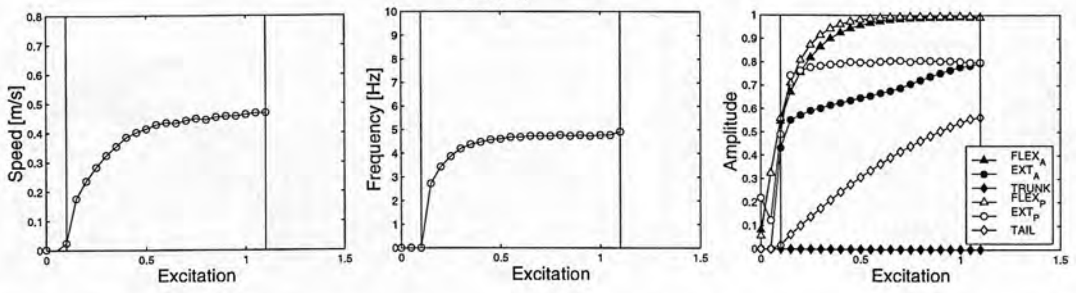
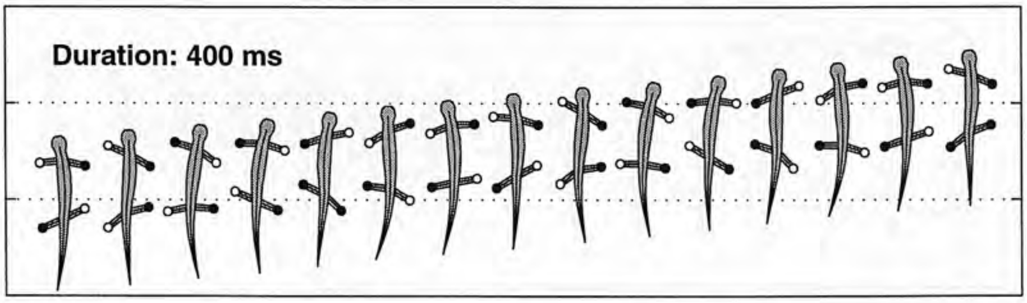
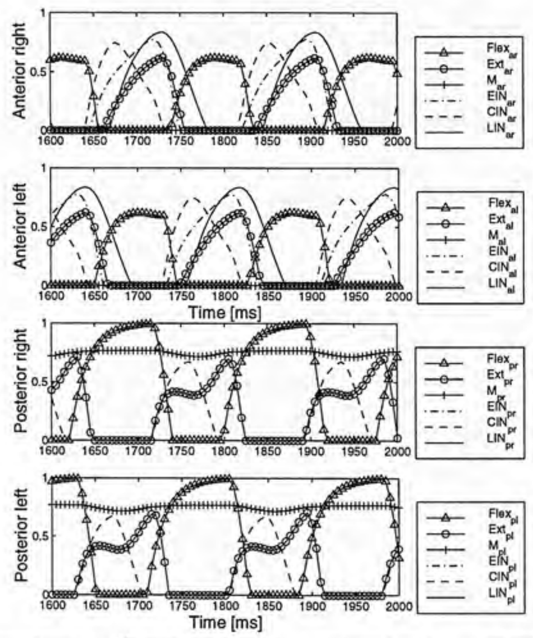


Figure H.4: **Run_A4:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	13.5	-3.5	-0.3	11.8	-0.1	-0.5	2.2	4.5
Ext_al	3.3	-0.1	–	–	-1.5	-0.3	0.4	0.7
Flex_ar	11.8	-0.1	-0.5	13.5	-3.5	-0.3	4.5	2.2
Ext_ar	–	-1.5	-0.3	3.3	-0.1	–	0.7	0.4
Trunk_l	–	-2.6	-2.6	–	-1.1	-1.8	1.9	1.4
Trunk_r	–	-1.1	-1.8	–	-2.6	-2.6	1.4	1.9
E_pl	–	–	-1.7	1.5	-1.4	-0.8	–	–
C_pl	–	-0.7	-0.1	5.1	-0.2	-1.9	–	–
L_pl	1.3	-1.2	–	3.3	–	-1.7	–	–
E_pr	1.5	-1.4	-0.8	–	–	-1.7	–	–
C_pr	5.1	-0.2	-1.9	–	-0.7	-0.1	–	–
L_pr	3.3	–	-1.7	1.3	-1.2	–	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	–	-4.1	-0.2	2.9	–	-1.4	17.2	6.9
Ext_pl	5.6	-0.7	-2.6	3.2	-4.6	–	3.2	1.4
Flex_pr	2.9	–	-1.4	–	-4.1	-0.2	6.9	17.2
Ext_pr	3.2	-4.6	–	5.6	-0.7	-2.6	1.4	3.2
Tail_l	–	-0.1	-0.4	1.4	–	-0.2	1.3	3.9
Tail_r	1.4	–	-0.2	–	-0.1	-0.4	3.9	1.3
E_al	4.8	-1.8	-0.4	0.4	-0.2	-1.3	–	–
C_al	–	-1.8	-0.1	2.1	–	-0.4	–	–
L_al	4.0	-0.4	–	–	–	-0.3	–	–
E_ar	0.4	-0.2	-1.3	4.8	-1.8	-0.4	–	–
C_ar	2.1	–	-0.4	–	-1.8	-0.1	–	–
L_ar	–	–	-0.3	4.0	-0.4	–	–	–





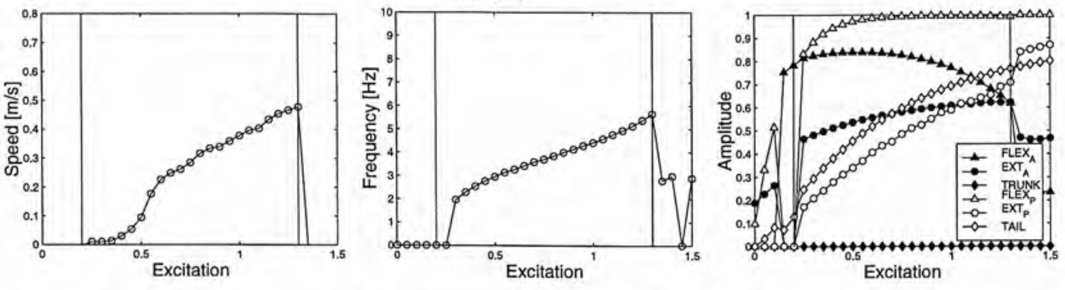
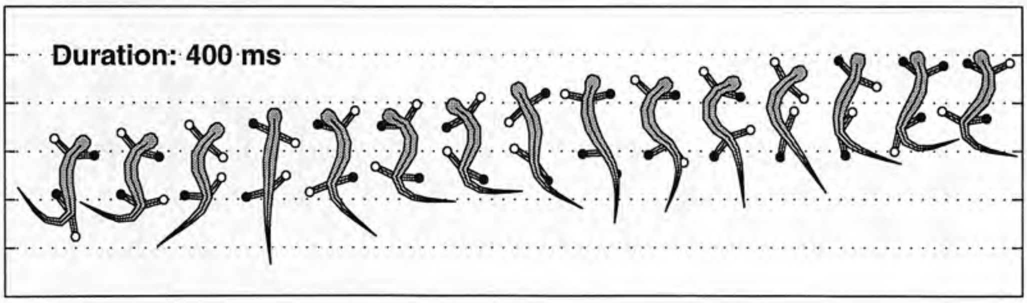
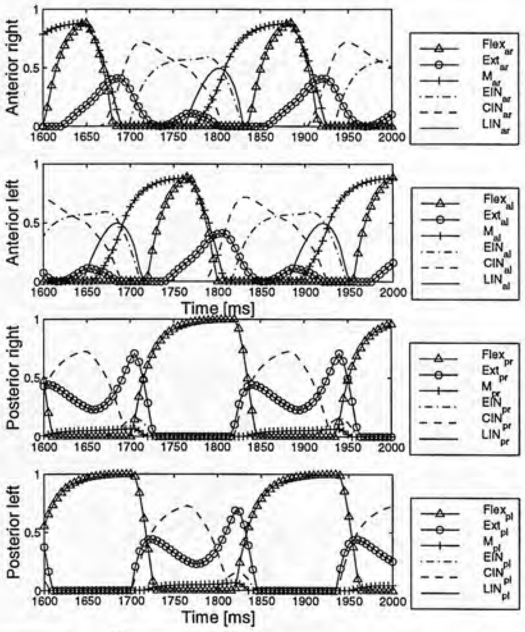


Figure H.5: **Run_A5:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	0.1	-5.0	-2.5	14.4	–	-2.5	4.0	1.3
Ext_al	–	-1.6	-1.1	1.0	-1.4	–	16.5	7.1
Flex_ar	14.4	–	-2.5	0.1	-5.0	-2.5	1.3	4.0
Ext_ar	1.0	-1.4	–	–	-1.6	-1.1	7.1	16.5
Trunk_l	1.1	-1.6	–	–	–	-1.8	6.7	20.0
Trunk_r	–	–	-1.8	1.1	-1.6	–	20.0	6.7
E_pl	0.7	-0.6	-1.4	2.4	-1.7	-0.9	–	–
C_pl	0.7	-1.6	-0.3	0.4	-0.3	-1.3	–	–
L_pl	1.4	-0.1	-0.2	2.4	-0.3	–	–	–
E_pr	2.4	-1.7	-0.9	0.7	-0.6	-1.4	–	–
C_pr	0.4	-0.3	-1.3	0.7	-1.6	-0.3	–	–
L_pr	2.4	-0.3	–	1.4	-0.1	-0.2	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	–	-4.0	–	14.1	–	–	21.7	8.3
Ext_pl	0.9	-1.1	-2.4	4.0	-3.5	–	6.9	6.0
Flex_pr	14.1	–	–	–	-4.0	–	8.3	21.7
Ext_pr	4.0	-3.5	–	0.9	-1.1	-2.4	6.0	6.9
Tail_l	0.0	-0.0	–	–	-0.6	-0.2	0.3	0.4
Tail_r	–	-0.6	-0.2	0.0	-0.0	–	0.4	0.3
E_al	0.6	-1.2	–	–	-0.4	-0.5	–	–
C_al	2.0	–	-0.8	3.2	-0.6	-1.7	–	–
L_al	0.4	-2.0	-1.6	–	–	-0.8	–	–
E_ar	–	-0.4	-0.5	0.6	-1.2	–	–	–
C_ar	3.2	-0.6	-1.7	2.0	–	-0.8	–	–
L_ar	–	–	-0.8	0.4	-2.0	-1.6	–	–





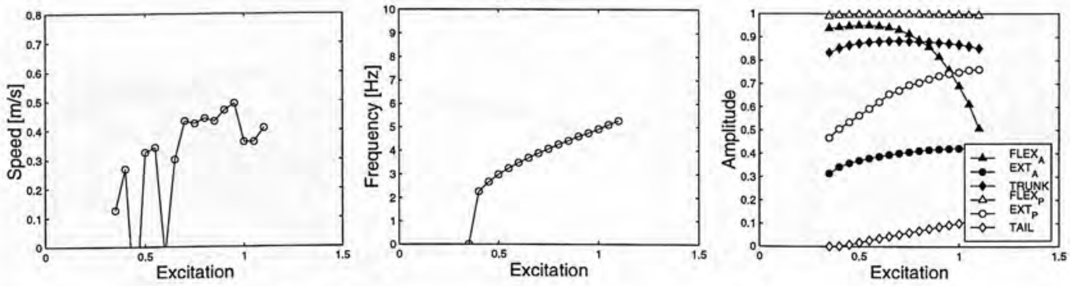
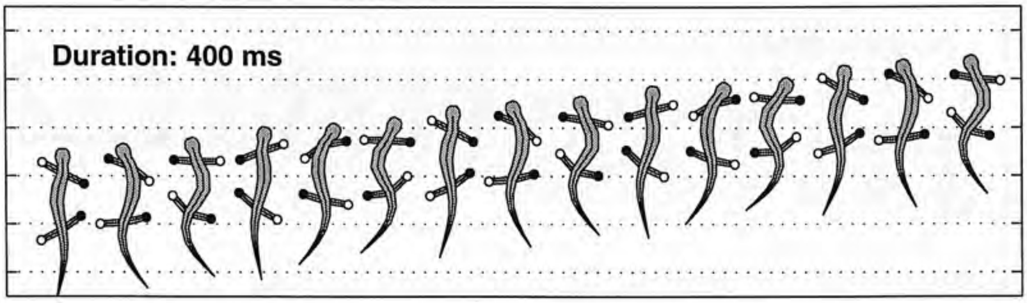
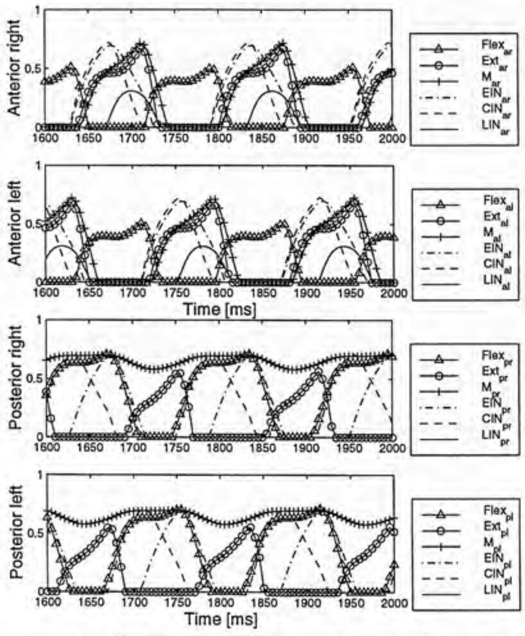


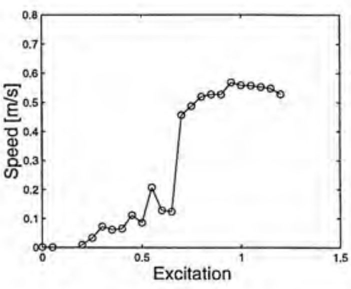
Figure H.6: **Run_A6:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

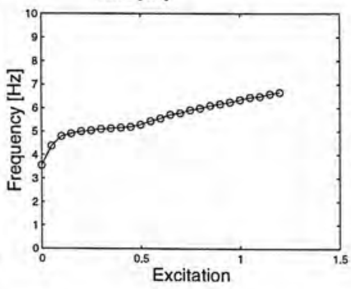
	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	10.1	-3.4	–	–	-0.4	–	2.4	1.5
Ext_al	15.0	-0.7	-0.3	0.9	-4.0	-0.4	7.9	20.5
Flex_ar	–	-0.4	–	10.1	-3.4	–	1.5	2.4
Ext_ar	0.9	-4.0	-0.4	15.0	-0.7	-0.3	20.5	7.9
Trunk_l	8.3	-0.4	–	1.1	-2.3	-1.7	3.9	9.7
Trunk_r	1.1	-2.3	-1.7	8.3	-0.4	–	9.7	3.9
E_pl	0.3	–	–	1.0	-1.4	-0.5	–	–
C_pl	–	-1.3	-0.1	3.4	–	-0.3	–	–
L_pl	–	-1.7	-0.2	–	-0.3	-1.1	–	–
E_pr	1.0	-1.4	-0.5	0.3	–	–	–	–
C_pr	3.4	–	-0.3	–	-1.3	-0.1	–	–
L_pr	–	-0.3	-1.1	–	-1.7	-0.2	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	–	-0.4	-2.5	7.5	-1.9	-0.5	5.0	2.8
Ext_pl	–	-4.4	-0.3	–	-0.8	-4.1	5.2	4.6
Flex_pr	7.5	-1.9	-0.5	–	-0.4	-2.5	2.8	5.0
Ext_pr	–	-0.8	-4.1	–	-4.4	-0.3	4.6	5.2
Tail_l	0.5	-0.0	–	0.9	-0.2	–	1.7	4.1
Tail_r	0.9	-0.2	–	0.5	-0.0	–	4.1	1.7
E_al	–	–	–	0.8	-1.3	-1.4	–	–
C_al	1.7	-0.3	–	–	-1.3	-1.7	–	–
L_al	4.7	-1.3	-0.5	5.1	-0.9	-1.2	–	–
E_ar	0.8	-1.3	-1.4	–	–	–	–	–
C_ar	–	-1.3	-1.7	1.7	-0.3	–	–	–
L_ar	5.1	-0.9	-1.2	4.7	-1.3	-0.5	–	–









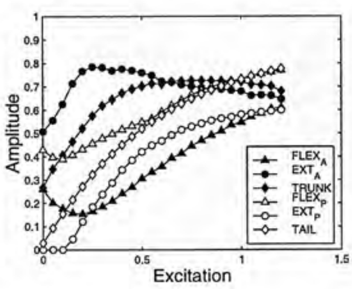
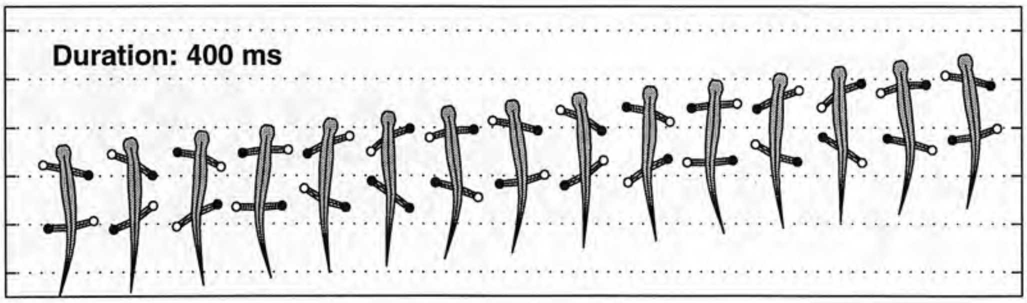
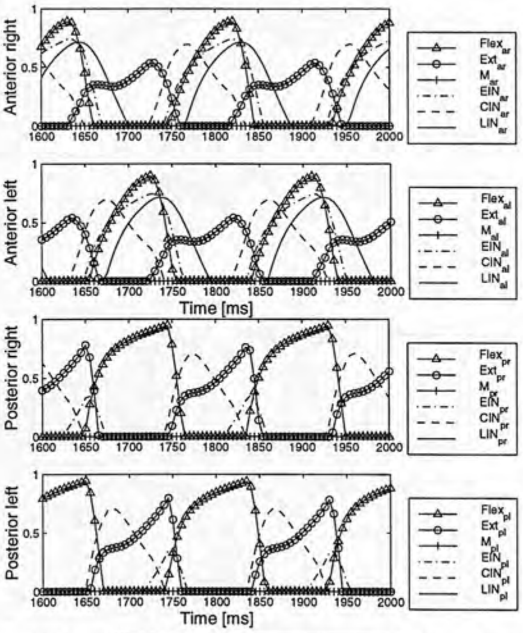


Figure H.7: **Run_A7:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	–	–	–	4.1	-3.4	-0.5	3.9	4.2
Ext_al	–	-2.1	–	–	-0.8	–	3.6	2.1
Flex_ar	4.1	-3.4	-0.5	–	–	–	4.2	3.9
Ext_ar	–	-0.8	–	–	-2.1	–	2.1	3.6
Trunk_l	1.4	-4.0	–	12.0	-3.2	–	8.1	21.9
Trunk_r	12.0	-3.2	–	1.4	-4.0	–	21.9	8.1
E_pl	2.2	-1.9	–	3.3	-0.6	–	–	–
C_pl	–	–	-1.2	4.4	-1.6	–	–	–
L_pl	–	-1.5	–	0.1	–	–	–	–
E_pr	3.3	-0.6	–	2.2	-1.9	–	–	–
C_pr	4.4	-1.6	–	–	–	-1.2	–	–
L_pr	0.1	–	–	–	-1.5	–	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	11.0	-3.8	-3.9	9.6	-0.2	-2.1	12.8	17.2
Ext_pl	9.1	-0.9	-2.1	1.0	-5.0	-0.4	10.0	18.2
Flex_pr	9.6	-0.2	-2.1	11.0	-3.8	-3.9	17.2	12.8
Ext_pr	1.0	-5.0	-0.4	9.1	-0.9	-2.1	18.2	10.0
Tail_l	3.0	-1.0	-0.7	2.5	-0.2	-0.4	-0.1	-0.1
Tail_r	2.5	-0.2	-0.4	3.0	-1.0	-0.7	-0.1	-0.1
E_al	6.0	-0.2	-0.0	0.5	-0.9	-0.3	–	–
C_al	0.3	-0.2	-0.2	–	–	–	–	–
L_al	0.1	-0.3	-0.0	–	-0.3	-0.5	–	–
E_ar	0.5	-0.9	-0.3	6.0	-0.2	-0.0	–	–
C_ar	–	–	–	0.3	-0.2	-0.2	–	–
L_ar	–	-0.3	-0.5	0.1	-0.3	-0.0	–	–





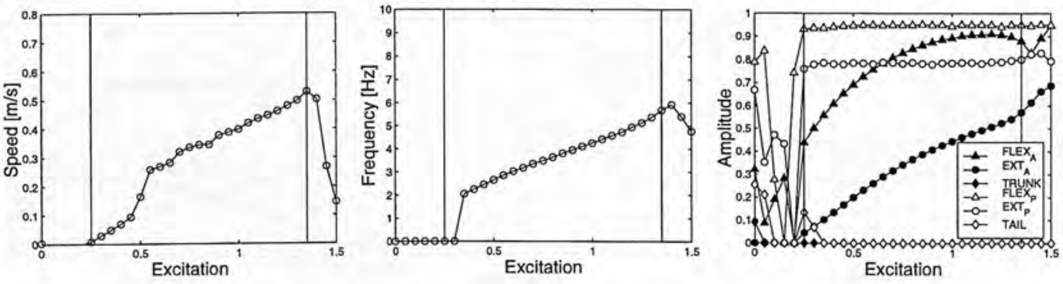
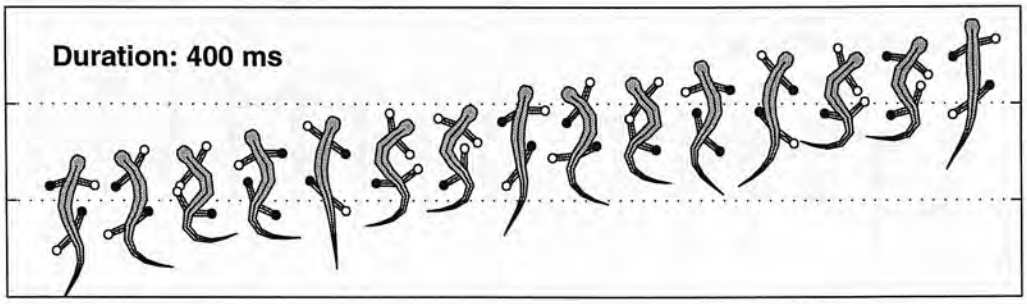
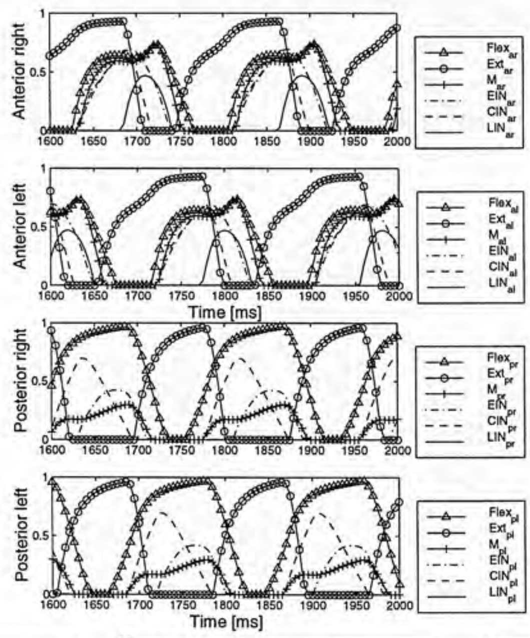


Figure H.8: **Run_A8:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	9.2	-0.5	-0.2	11.5	-2.3	-0.2	1.7	5.1
Ext_al	–	–	-3.9	5.4	-0.4	–	5.3	7.7
Flex_ar	11.5	-2.3	-0.2	9.2	-0.5	-0.2	5.1	1.7
Ext_ar	5.4	-0.4	–	–	–	-3.9	7.7	5.3
Trunk_l	13.6	-0.5	–	–	-3.2	–	8.9	5.4
Trunk_r	–	-3.2	–	13.6	-0.5	–	5.4	8.9
E_pl	2.1	-0.9	-0.7	4.2	-1.4	-0.7	–	–
C_pl	1.0	-1.6	-0.1	0.3	-0.5	–	–	–
L_pl	3.2	-1.9	-1.8	0.5	-1.6	-0.0	–	–
E_pr	4.2	-1.4	-0.7	2.1	-0.9	-0.7	–	–
C_pr	0.3	-0.5	–	1.0	-1.6	-0.1	–	–
L_pr	0.5	-1.6	-0.0	3.2	-1.9	-1.8	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	6.4	-0.1	-3.5	3.0	-1.9	-2.1	8.3	14.3
Ext_pl	8.5	-4.5	-5.0	–	-0.1	-0.5	17.3	8.7
Flex_pr	3.0	-1.9	-2.1	6.4	-0.1	-3.5	14.3	8.3
Ext_pr	–	-0.1	-0.5	8.5	-4.5	-5.0	8.7	17.3
Tail_l	1.7	-0.1	-0.4	1.6	-0.9	-0.2	0.9	0.7
Tail_r	1.6	-0.9	-0.2	1.7	-0.1	-0.4	0.7	0.9
E_al	1.2	-0.3	-0.9	1.2	-1.4	-1.3	–	–
C_al	2.6	-0.6	-2.0	5.7	-1.8	-0.0	–	–
L_al	2.9	-0.2	-0.4	3.4	-0.7	-2.0	–	–
E_ar	1.2	-1.4	-1.3	1.2	-0.3	-0.9	–	–
C_ar	5.7	-1.8	-0.0	2.6	-0.6	-2.0	–	–
L_ar	3.4	-0.7	-2.0	2.9	-0.2	-0.4	–	–





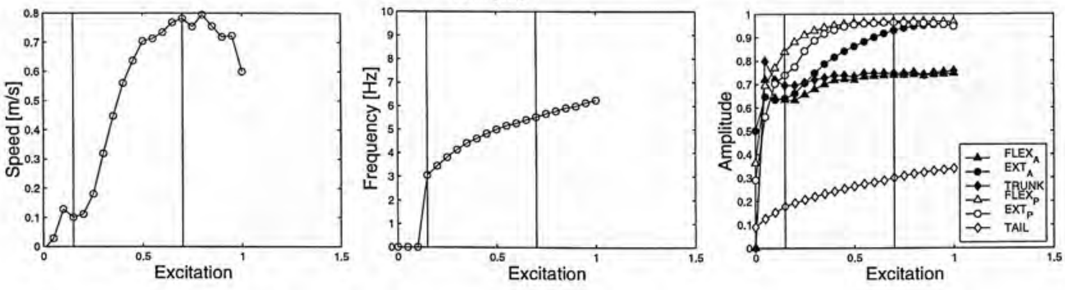
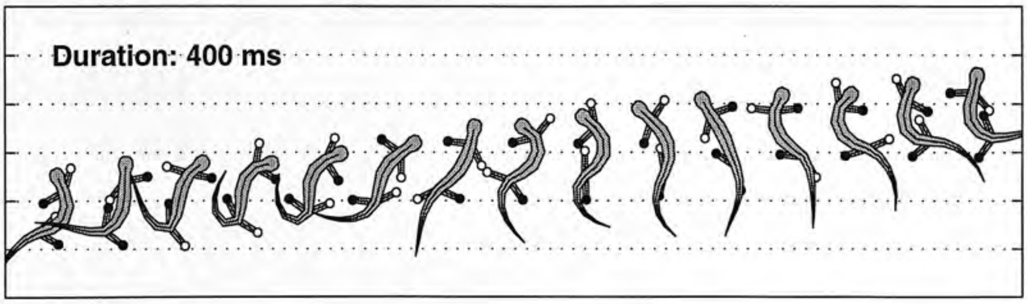
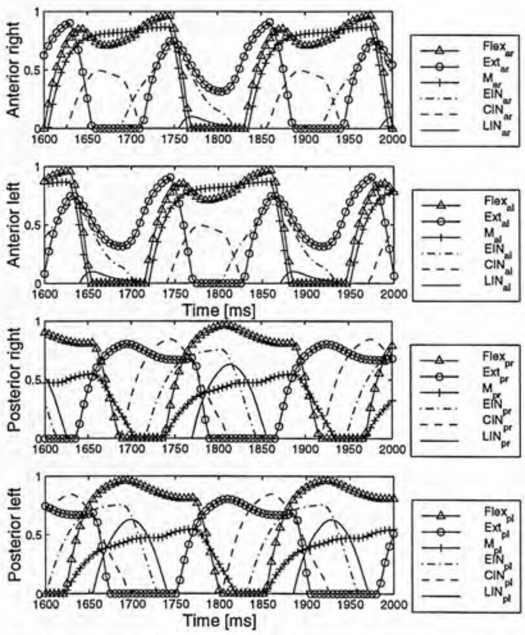


Figure H.9: **Run_A9:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	–	-0.3	-3.8	6.7	-4.0	-2.5	8.7	17.9
Ext_al	–	-3.5	–	9.3	-1.4	-2.5	10.1	5.1
Flex_ar	6.7	-4.0	-2.5	–	-0.3	-3.8	17.9	8.7
Ext_ar	9.3	-1.4	-2.5	–	-3.5	–	5.1	10.1
Trunk_l	–	-0.1	-0.4	3.8	-4.8	–	5.9	16.4
Trunk_r	3.8	-4.8	–	–	-0.1	-0.4	16.4	5.9
E_pl	0.6	-2.0	–	2.1	-0.2	-0.0	–	–
C_pl	3.6	-0.1	-0.6	–	-0.8	-0.2	–	–
L_pl	0.3	-0.0	-0.9	5.4	-0.1	-0.2	–	–
E_pr	2.1	-0.2	-0.0	0.6	-2.0	–	–	–
C_pr	–	-0.8	-0.2	3.6	-0.1	-0.6	–	–
L_pr	5.4	-0.1	-0.2	0.3	-0.0	-0.9	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	0.1	–	–	–	-0.2	-2.7	11.6	17.4
Ext_pl	1.9	-0.3	-4.7	11.9	-0.2	-0.2	6.4	5.9
Flex_pr	–	-0.2	-2.7	0.1	–	–	17.4	11.6
Ext_pr	11.9	-0.2	-0.2	1.9	-0.3	-4.7	5.9	6.4
Tail_l	3.0	-0.4	-0.3	–	–	-1.0	4.0	2.0
Tail_r	–	–	-1.0	3.0	-0.4	-0.3	2.0	4.0
E_al	5.2	–	–	2.2	-1.7	-0.5	–	–
C_al	0.3	-1.6	-1.6	3.7	-0.5	-0.8	–	–
L_al	5.8	-0.8	–	0.4	-0.0	-0.3	–	–
E_ar	2.2	-1.7	-0.5	5.2	–	–	–	–
C_ar	3.7	-0.5	-0.8	0.3	-1.6	-1.6	–	–
L_ar	0.4	-0.0	-0.3	5.8	-0.8	–	–	–





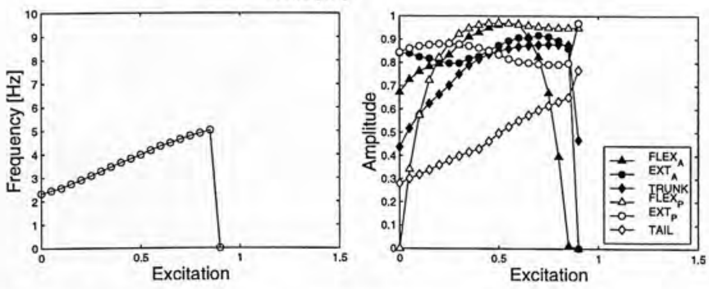


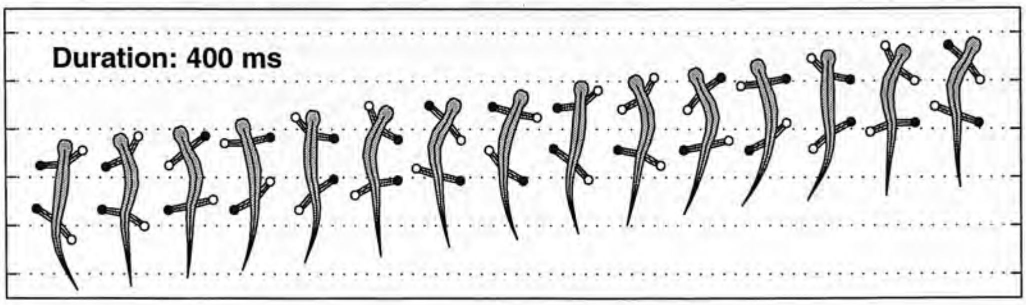
Figure H.10: **Run_A10:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

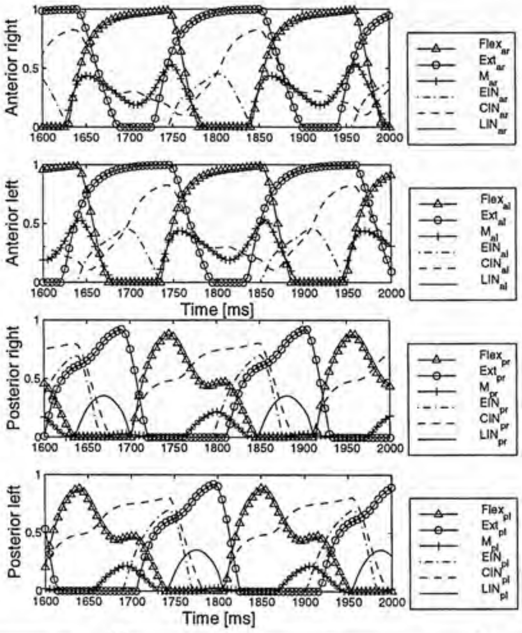
Appendix I

Results of chapter 7: Runs A11 to A20

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_al	–	7.8	-2.9	-0.5	–	–	2.9	–
L_al	–	10.3	–	-4.9	6.8	-1.5	–	–
E_ar	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_ar	-0.5	–	–	–	7.8	-2.9	–	2.9
L_ar	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_al	-4.5	10.7	-1.9	-0.1	1.3	–	15.1	8.5
Ext_al	–	11.1	–	-3.3	–	-2.7	5.7	17.0
Flex_ar	-0.1	1.3	–	-4.5	10.7	-1.9	8.5	15.1
Ext_ar	-3.3	–	-2.7	–	11.1	–	17.0	5.7
Trunk_l	-2.5	2.3	–	-1.2	0.3	-3.0	10.7	6.9
Trunk_r	-1.2	0.3	-3.0	-2.5	2.3	–	6.9	10.7
E_pl	-0.0	–	–	-1.8	–	-0.9	–	–
C_pl	-0.2	0.7	-1.3	-0.2	0.1	-0.2	–	–
L_pl	-1.6	4.4	-0.1	-0.5	–	-1.6	–	–
E_pr	-1.8	–	-0.9	-0.0	–	–	–	–
C_pr	-0.2	0.1	-0.2	-0.2	0.7	-1.3	–	–
L_pr	-0.5	–	-1.6	-1.6	4.4	-0.1	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_pl	–	7.8	-2.9	-0.5	–	–	2.9	–
L_pl	–	10.3	–	-4.9	6.8	-1.5	–	–
E_pr	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_pr	-0.5	–	–	–	7.8	-2.9	–	2.9
L_pr	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_pl	-1.9	14.2	-4.5	–	0.5	-2.0	18.1	9.1
Ext_pl	-0.4	–	–	-3.5	5.3	-4.6	6.9	6.5
Flex_pr	–	0.5	-2.0	-1.9	14.2	-4.5	9.1	18.1
Ext_pr	-3.5	5.3	-4.6	-0.4	–	–	6.5	6.9
Tail_l	-0.7	2.0	-0.1	-0.4	0.2	–	0.9	1.2
Tail_r	-0.4	0.2	–	-0.7	2.0	-0.1	1.2	0.9
E_al	-2.0	–	-0.6	-0.0	0.3	-1.0	–	–
C_al	–	3.2	-1.4	-1.1	2.7	-1.3	–	–
L_al	-0.6	0.0	-1.6	-1.2	2.3	-1.8	–	–
E_ar	-0.0	0.3	-1.0	-2.0	–	-0.6	–	–
C_ar	-1.1	2.7	-1.3	–	3.2	-1.4	–	–
L_ar	-1.2	2.3	-1.8	-0.6	0.0	-1.6	–	–





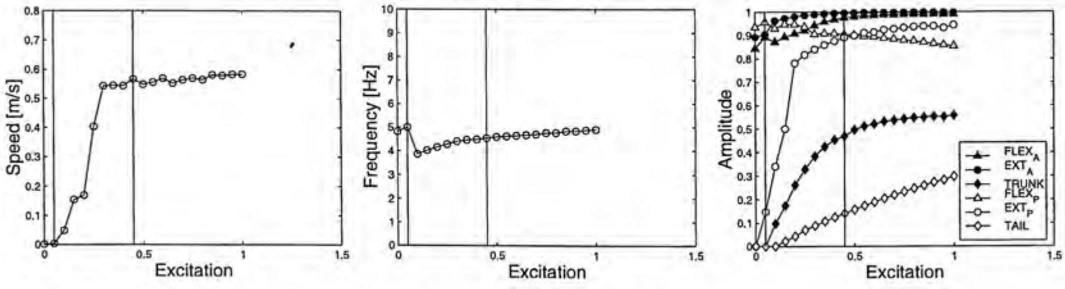
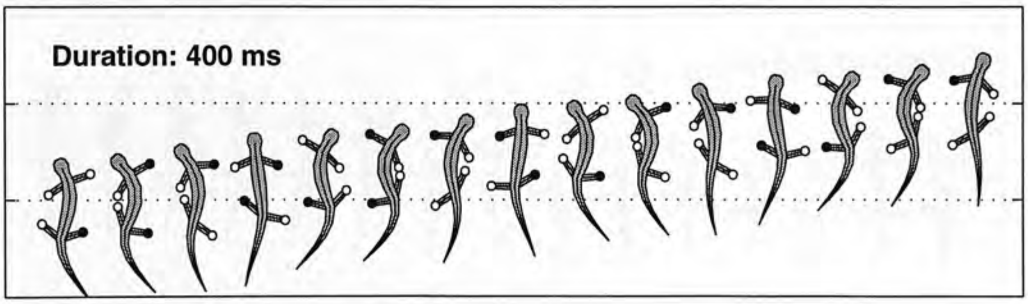
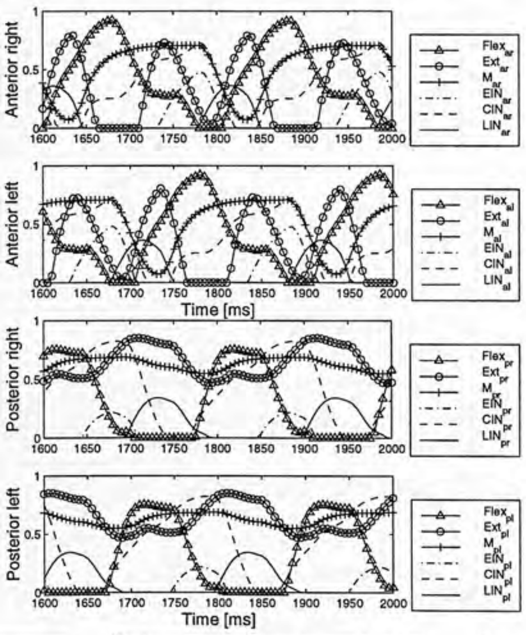


Figure I.1: **Run_A10:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_al	–	7.8	-2.9	-0.5	–	–	2.9	–
L_al	–	10.3	–	-4.9	6.8	-1.5	–	–
E_ar	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_ar	-0.5	–	–	–	7.8	-2.9	–	2.9
L_ar	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_al	-2.8	6.9	-1.0	-0.1	–	-2.4	6.3	3.9
Ext_al	-2.4	4.9	-0.1	-4.6	13.2	–	10.0	7.3
Flex_ar	-0.1	–	-2.4	-2.8	6.9	-1.0	3.9	6.3
Ext_ar	-4.6	13.2	–	-2.4	4.9	-0.1	7.3	10.0
Trunk_l	–	0.3	-2.2	-0.0	1.4	–	4.2	12.5
Trunk_r	-0.0	1.4	–	–	0.3	-2.2	12.5	4.2
E_pl	-0.9	3.5	-1.3	-0.8	1.7	-1.3	–	–
C_pl	-0.4	4.1	-0.7	–	4.6	-0.2	–	–
L_pl	-0.0	6.0	-1.8	-0.6	0.1	-0.3	–	–
E_pr	-0.8	1.7	-1.3	-0.9	3.5	-1.3	–	–
C_pr	–	4.6	-0.2	-0.4	4.1	-0.7	–	–
L_pr	-0.6	0.1	-0.3	-0.0	6.0	-1.8	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_pl	–	7.8	-2.9	-0.5	–	–	2.9	–
L_pl	–	10.3	–	-4.9	6.8	-1.5	–	–
E_pr	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_pr	-0.5	–	–	–	7.8	-2.9	–	2.9
L_pr	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_pl	-5.0	–	-4.8	–	–	-0.8	3.3	2.9
Ext_pl	-0.3	13.5	-0.7	-2.7	1.5	-1.5	5.0	6.6
Flex_pr	–	–	-0.8	-5.0	–	-4.8	2.9	3.3
Ext_pr	-2.7	1.5	-1.5	-0.3	13.5	-0.7	6.6	5.0
Tail_l	–	–	-0.3	-0.6	1.7	–	3.9	1.8
Tail_r	-0.6	1.7	–	–	–	-0.3	1.8	3.9
E_al	-0.0	5.4	-0.2	–	2.4	-1.5	–	–
C_al	-1.8	3.8	-0.8	–	4.2	-0.6	–	–
L_al	–	4.9	-1.8	-0.8	6.0	-1.4	–	–
E_ar	–	2.4	-1.5	-0.0	5.4	-0.2	–	–
C_ar	–	4.2	-0.6	-1.8	3.8	-0.8	–	–
L_ar	-0.8	6.0	-1.4	–	4.9	-1.8	–	–





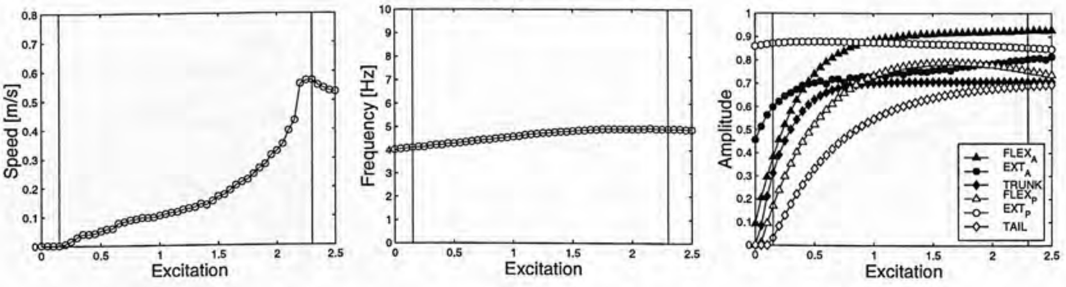
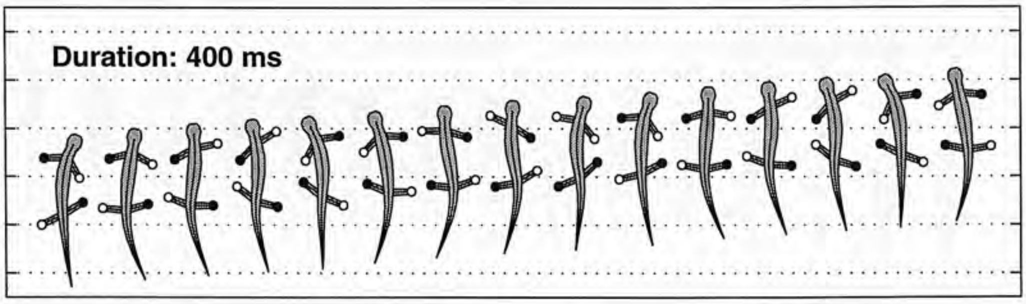
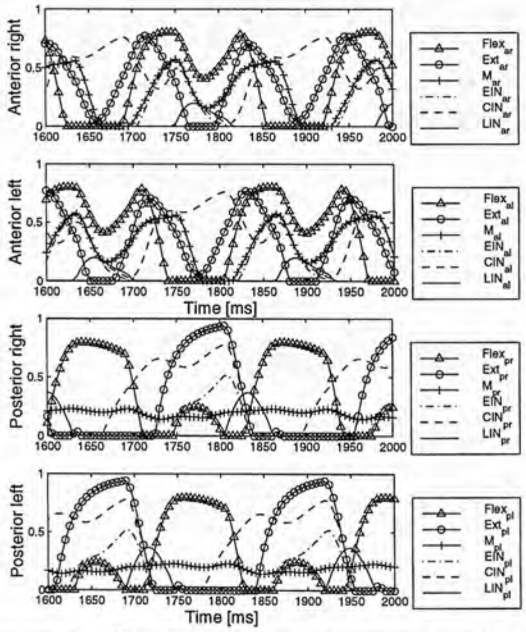


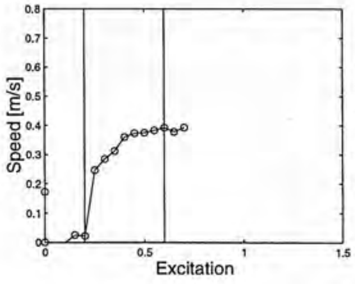
Figure I.2: **Run_A12:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

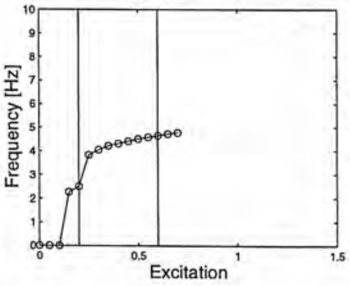
	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_al	–	7.8	-2.9	-0.5	–	–	2.9	–
L_al	–	10.3	–	-4.9	6.8	-1.5	–	–
E_ar	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_ar	-0.5	–	–	–	7.8	-2.9	–	2.9
L_ar	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_al	-0.4	0.2	-3.3	-3.5	9.1	-4.0	10.6	19.4
Ext_al	-1.5	8.5	-5.0	-1.4	5.7	-3.2	19.7	10.2
Flex_ar	-3.5	9.1	-4.0	-0.4	0.2	-3.3	19.4	10.6
Ext_ar	-1.4	5.7	-3.2	-1.5	8.5	-5.0	10.2	19.7
Trunk_l	–	1.8	-3.3	–	10.9	-4.3	1.9	1.9
Trunk_r	–	10.9	-4.3	–	1.8	-3.3	1.9	1.9
E_pl	-0.5	3.2	-1.9	-1.2	–	-1.4	–	–
C_pl	-0.6	3.1	–	-1.5	3.0	-0.9	–	–
L_pl	-1.7	–	–	-0.5	5.0	-1.8	–	–
E_pr	-1.2	–	-1.4	-0.5	3.2	-1.9	–	–
C_pr	-1.5	3.0	-0.9	-0.6	3.1	–	–	–
L_pr	-0.5	5.0	-1.8	-1.7	–	–	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_pl	–	7.8	-2.9	-0.5	–	–	2.9	–
L_pl	–	10.3	–	-4.9	6.8	-1.5	–	–
E_pr	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_pr	-0.5	–	–	–	7.8	-2.9	–	2.9
L_pr	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_pl	-2.6	7.8	–	-0.7	5.1	-4.5	12.5	11.8
Ext_pl	–	8.0	-3.7	-4.0	–	–	5.9	6.0
Flex_pr	-0.7	5.1	-4.5	-2.6	7.8	–	11.8	12.5
Ext_pr	-4.0	–	–	–	8.0	-3.7	6.0	5.9
Tail_l	-0.3	1.8	-0.1	-0.2	2.1	-0.4	1.7	2.0
Tail_r	-0.2	2.1	-0.4	-0.3	1.8	-0.1	2.0	1.7
E_al	-1.3	1.7	-1.3	-1.0	4.7	-0.4	–	–
C_al	-0.9	–	–	-0.1	0.3	-1.9	–	–
L_al	–	2.0	-1.0	–	1.6	–	–	–
E_ar	-1.0	4.7	-0.4	-1.3	1.7	-1.3	–	–
C_ar	-0.1	0.3	-1.9	-0.9	–	–	–	–
L_ar	–	1.6	–	–	2.0	-1.0	–	–









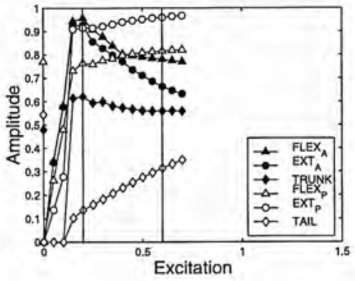
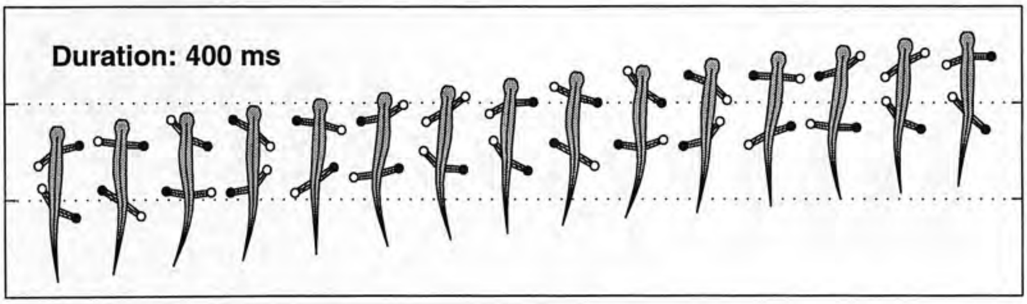
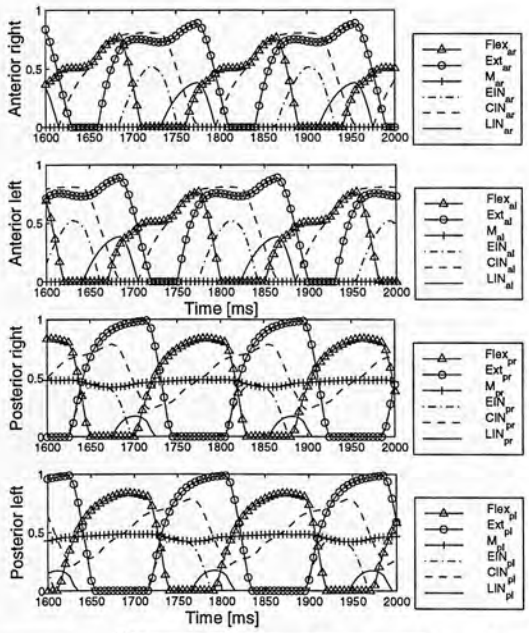


Figure I.3: **Run_A13:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_al	–	7.8	-2.9	-0.5	–	–	2.9	–
L_al	–	10.3	–	-4.9	6.8	-1.5	–	–
E_ar	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_ar	-0.5	–	–	–	7.8	-2.9	–	2.9
L_ar	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_al	-4.1	8.9	-1.1	-1.0	0.3	-0.5	13.1	9.3
Ext_al	-0.4	–	–	-2.9	1.2	–	1.9	4.4
Flex_ar	-1.0	0.3	-0.5	-4.1	8.9	-1.1	9.3	13.1
Ext_ar	-2.9	1.2	–	-0.4	–	–	4.4	1.9
Trunk_l	-2.0	6.4	–	–	–	-0.4	-1.0	-3.0
Trunk_r	–	–	-0.4	-2.0	6.4	–	-3.0	-1.0
E_pl	-1.4	5.4	-0.2	–	–	-1.4	–	–
C_pl	-0.8	1.3	-0.3	-0.2	5.8	–	–	–
L_pl	-0.1	–	-0.7	–	–	-0.7	–	–
E_pr	–	–	-1.4	-1.4	5.4	-0.2	–	–
C_pr	-0.2	5.8	–	-0.8	1.3	-0.3	–	–
L_pr	–	–	-0.7	-0.1	–	-0.7	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_pl	–	7.8	-2.9	-0.5	–	–	2.9	–
L_pl	–	10.3	–	-4.9	6.8	-1.5	–	–
E_pr	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_pr	-0.5	–	–	–	7.8	-2.9	–	2.9
L_pr	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_pl	-4.0	2.5	-0.5	–	4.9	–	0.9	0.5
Ext_pl	–	0.5	–	-4.1	–	-4.6	17.6	9.6
Flex_pr	–	4.9	–	-4.0	2.5	-0.5	0.5	0.9
Ext_pr	-4.1	–	-4.6	–	0.5	–	9.6	17.6
Tail_l	-0.2	1.2	-0.1	–	0.1	-0.1	0.9	2.6
Tail_r	–	0.1	-0.1	-0.2	1.2	-0.1	2.6	0.9
E_al	–	3.1	-0.0	-1.6	–	-2.0	–	–
C_al	–	–	-0.1	-0.2	0.6	-0.7	–	–
L_al	-1.6	0.4	-1.9	-0.3	3.9	-0.0	–	–
E_ar	-1.6	–	-2.0	–	3.1	-0.0	–	–
C_ar	-0.2	0.6	-0.7	–	–	-0.1	–	–
L_ar	-0.3	3.9	-0.0	-1.6	0.4	-1.9	–	–





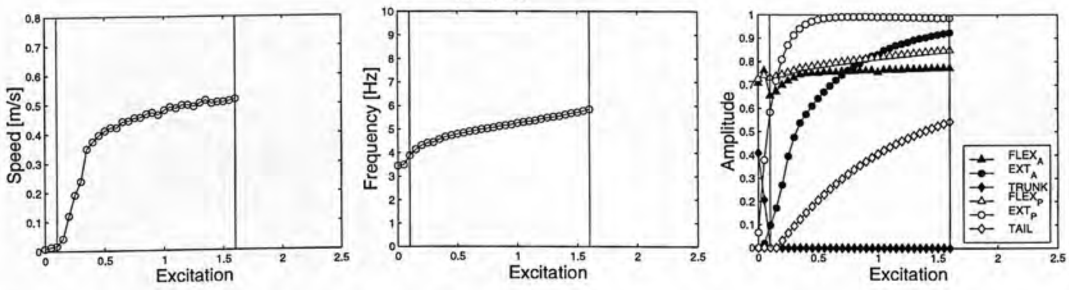
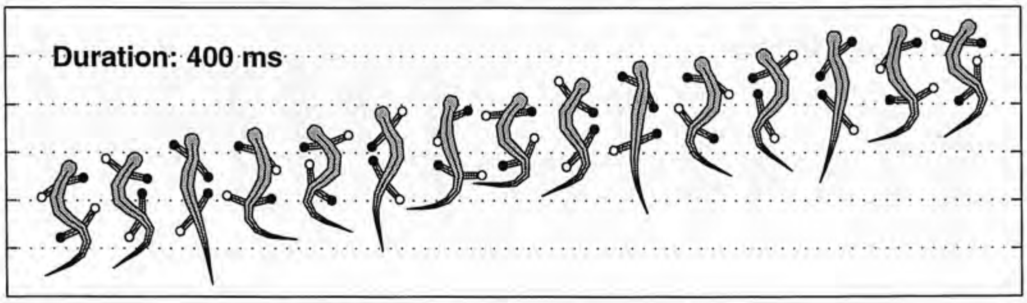
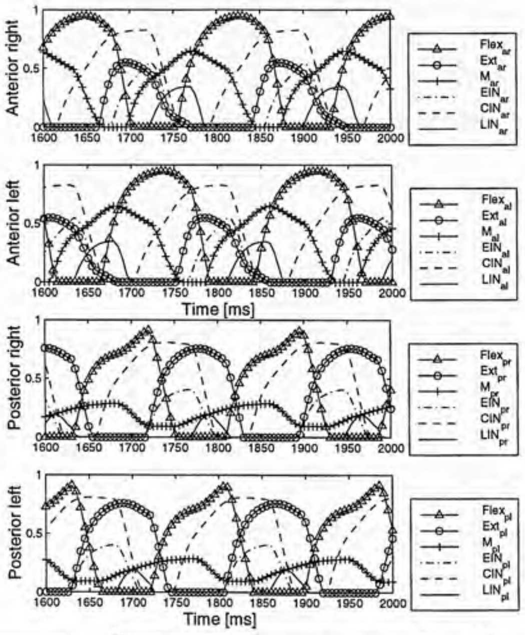


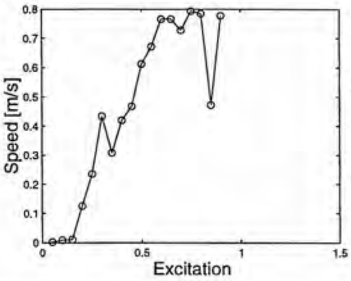
Figure I.4: **Run_A14:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

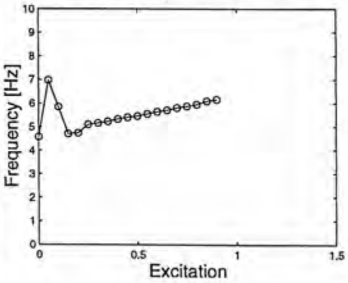
	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_al	–	7.8	-2.9	-0.5	–	–	2.9	–
L_al	–	10.3	–	-4.9	6.8	-1.5	–	–
E_ar	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_ar	-0.5	–	–	–	7.8	-2.9	–	2.9
L_ar	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_al	-3.8	–	–	–	12.0	–	3.1	1.8
Ext_al	-1.1	–	-2.9	-3.2	3.0	-0.3	8.9	4.7
Flex_ar	–	12.0	–	-3.8	–	–	1.8	3.1
Ext_ar	-3.2	3.0	-0.3	-1.1	–	-2.9	4.7	8.9
Trunk_l	-0.4	7.0	–	-0.5	0.6	-2.7	9.3	20.7
Trunk_r	-0.5	0.6	-2.7	-0.4	7.0	–	20.7	9.3
E_pl	-0.2	4.7	-1.3	-1.4	1.4	-0.1	–	–
C_pl	–	2.8	-0.2	-1.8	–	–	–	–
L_pl	-2.0	5.6	–	–	–	-1.5	–	–
E_pr	-1.4	1.4	-0.1	-0.2	4.7	-1.3	–	–
C_pr	-1.8	–	–	–	2.8	-0.2	–	–
L_pr	–	–	-1.5	-2.0	5.6	–	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_pl	–	7.8	-2.9	-0.5	–	–	2.9	–
L_pl	–	10.3	–	-4.9	6.8	-1.5	–	–
E_pr	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_pr	-0.5	–	–	–	7.8	-2.9	–	2.9
L_pr	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_pl	-4.0	7.9	-5.0	-0.6	2.5	–	15.9	11.9
Ext_pl	-0.2	5.1	-0.2	-3.6	–	-2.8	3.2	1.7
Flex_pr	-0.6	2.5	–	-4.0	7.9	-5.0	11.9	15.9
Ext_pr	-3.6	–	-2.8	-0.2	5.1	-0.2	1.7	3.2
Tail_l	-0.0	0.2	-0.1	–	1.7	-0.7	1.2	1.5
Tail_r	–	1.7	-0.7	-0.0	0.2	-0.1	1.5	1.2
E_al	-1.3	4.4	–	–	1.8	-1.2	–	–
C_al	-0.8	2.9	-0.9	–	4.6	-0.1	–	–
L_al	-0.7	5.8	-1.5	-0.9	0.4	-1.0	–	–
E_ar	–	1.8	-1.2	-1.3	4.4	–	–	–
C_ar	–	4.6	-0.1	-0.8	2.9	-0.9	–	–
L_ar	-0.9	0.4	-1.0	-0.7	5.8	-1.5	–	–









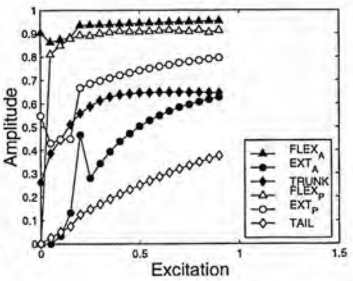
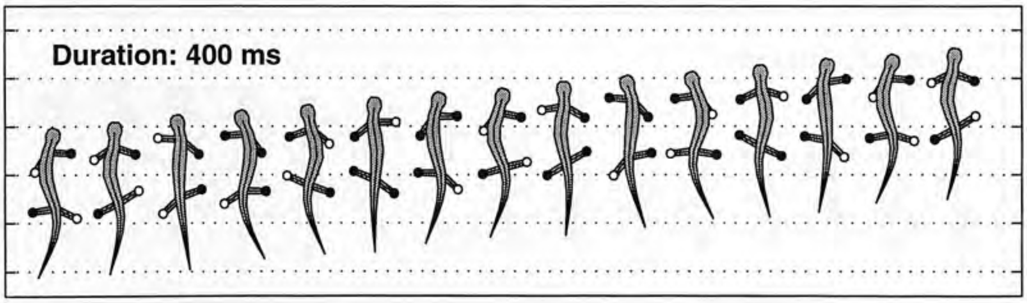
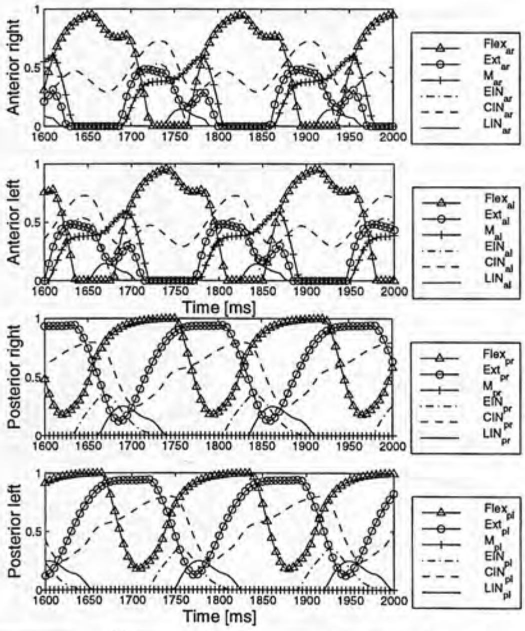


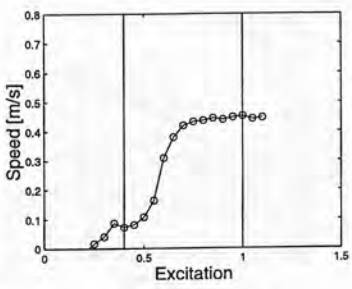
Figure I.5: **Run_A15:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

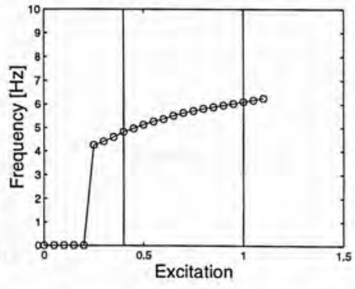
	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_al	–	7.8	-2.9	-0.5	–	–	2.9	–
L_al	–	10.3	–	-4.9	6.8	-1.5	–	–
E_ar	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_ar	-0.5	–	–	–	7.8	-2.9	–	2.9
L_ar	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_al	-3.6	12.5	-0.0	–	8.5	-2.5	12.9	16.8
Ext_al	-0.8	–	-4.8	-4.1	0.2	-0.7	2.2	5.1
Flex_ar	–	8.5	-2.5	-3.6	12.5	-0.0	16.8	12.9
Ext_ar	-4.1	0.2	-0.7	-0.8	–	-4.8	5.1	2.2
Trunk_l	-0.6	6.5	-0.3	-3.0	0.8	-3.1	12.6	13.7
Trunk_r	-3.0	0.8	-3.1	-0.6	6.5	-0.3	13.7	12.6
E_pl	-0.8	4.0	-1.0	-0.7	–	–	–	–
C_pl	–	–	-0.4	–	–	-0.5	–	–
L_pl	-0.1	–	-1.2	-0.3	0.0	-0.1	–	–
E_pr	-0.7	–	–	-0.8	4.0	-1.0	–	–
C_pr	–	–	-0.5	–	–	-0.4	–	–
L_pr	-0.3	0.0	-0.1	-0.1	–	-1.2	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_pl	–	7.8	-2.9	-0.5	–	–	2.9	–
L_pl	–	10.3	–	-4.9	6.8	-1.5	–	–
E_pr	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_pr	-0.5	–	–	–	7.8	-2.9	–	2.9
L_pr	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_pl	–	10.3	–	–	13.5	-4.4	5.3	6.8
Ext_pl	-2.9	–	-0.6	–	7.5	-0.2	4.6	8.5
Flex_pr	–	13.5	-4.4	–	10.3	–	6.8	5.3
Ext_pr	–	7.5	-0.2	-2.9	–	-0.6	8.5	4.6
Tail_l	-0.2	0.1	-0.4	-0.6	0.1	-0.0	-1.4	-0.5
Tail_r	-0.6	0.1	-0.0	-0.2	0.1	-0.4	-0.5	-1.4
E_al	–	0.3	-0.1	-2.0	6.0	-1.1	–	–
C_al	-1.5	0.5	–	-0.8	4.3	-0.6	–	–
L_al	-0.1	4.8	-1.9	-1.0	0.2	-1.1	–	–
E_ar	-2.0	6.0	-1.1	–	0.3	-0.1	–	–
C_ar	-0.8	4.3	-0.6	-1.5	0.5	–	–	–
L_ar	-1.0	0.2	-1.1	-0.1	4.8	-1.9	–	–









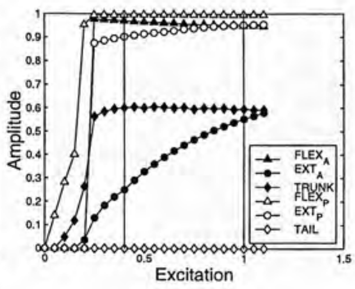
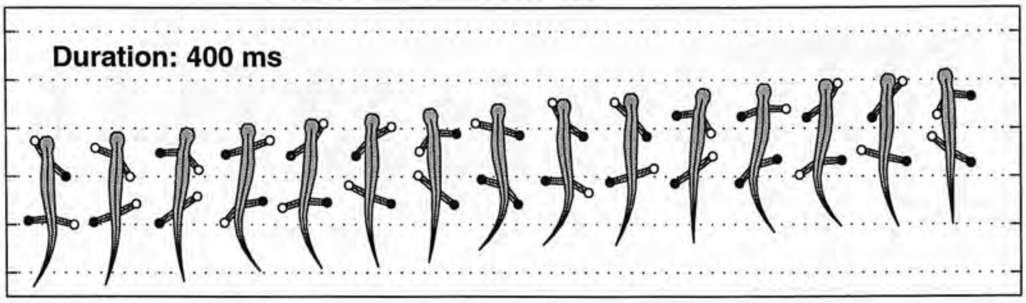
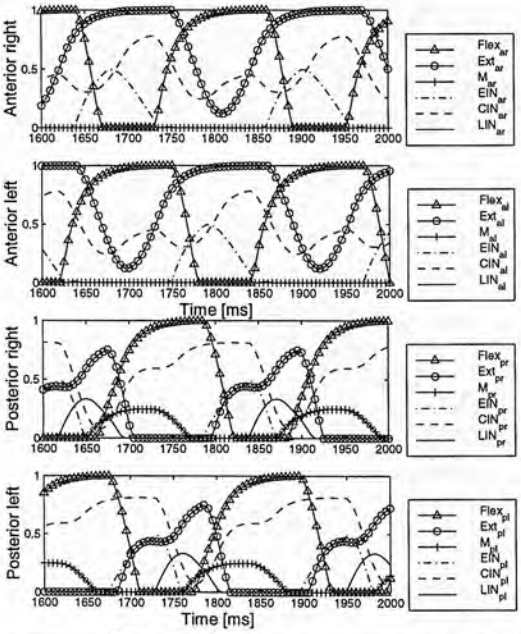


Figure I.6: **Run_A16:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_al	–	7.8	-2.9	-0.5	–	–	2.9	–
L_al	–	10.3	–	-4.9	6.8	-1.5	–	–
E_ar	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_ar	-0.5	–	–	–	7.8	-2.9	–	2.9
L_ar	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_al	-4.3	0.8	-5.0	–	3.1	-0.8	8.2	8.5
Ext_al	–	12.6	-0.1	-2.1	10.1	-0.1	19.0	6.3
Flex_ar	–	3.1	-0.8	-4.3	0.8	-5.0	8.5	8.2
Ext_ar	-2.1	10.1	-0.1	–	12.6	-0.1	6.3	19.0
Trunk_l	-0.8	6.5	-1.3	-0.1	–	-1.3	-2.7	-5.0
Trunk_r	-0.1	–	-1.3	-0.8	6.5	-1.3	-5.0	-2.7
E_pl	-1.4	0.4	-1.1	-0.4	0.5	–	–	–
C_pl	–	0.6	-0.2	–	2.4	-2.0	–	–
L_pl	-0.3	2.0	-0.1	-1.7	–	-0.2	–	–
E_pr	-0.4	0.5	–	-1.4	0.4	-1.1	–	–
C_pr	–	2.4	-2.0	–	0.6	-0.2	–	–
L_pr	-1.7	–	-0.2	-0.3	2.0	-0.1	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_pl	–	7.8	-2.9	-0.5	–	–	2.9	–
L_pl	–	10.3	–	-4.9	6.8	-1.5	–	–
E_pr	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_pr	-0.5	–	–	–	7.8	-2.9	–	2.9
L_pr	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_pl	-4.6	0.1	-1.7	–	0.4	–	4.8	12.5
Ext_pl	-0.8	–	-0.0	-4.9	–	-2.8	2.0	1.0
Flex_pr	–	0.4	–	-4.6	0.1	-1.7	12.5	4.8
Ext_pr	-4.9	–	-2.8	-0.8	–	-0.0	1.0	2.0
Tail_l	-0.8	0.2	-0.1	-0.1	3.0	-0.3	0.1	0.2
Tail_r	-0.1	3.0	-0.3	-0.8	0.2	-0.1	0.2	0.1
E_al	-0.0	1.9	-0.1	-1.9	1.8	-1.1	–	–
C_al	-0.7	0.2	-1.7	-0.1	1.4	-1.8	–	–
L_al	-1.3	–	-2.0	-0.9	0.6	-0.4	–	–
E_ar	-1.9	1.8	-1.1	-0.0	1.9	-0.1	–	–
C_ar	-0.1	1.4	-1.8	-0.7	0.2	-1.7	–	–
L_ar	-0.9	0.6	-0.4	-1.3	–	-2.0	–	–





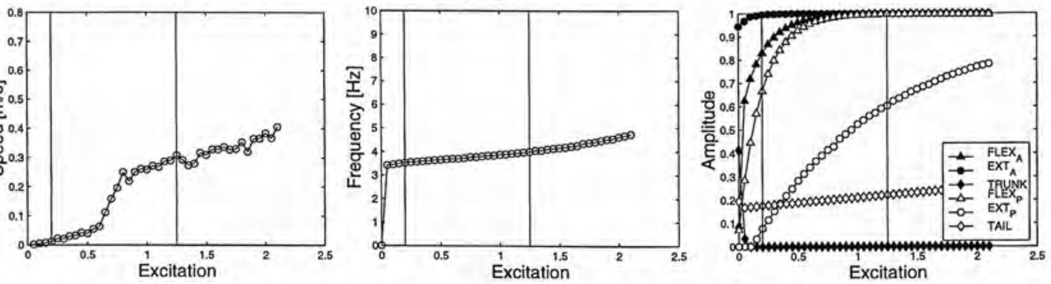
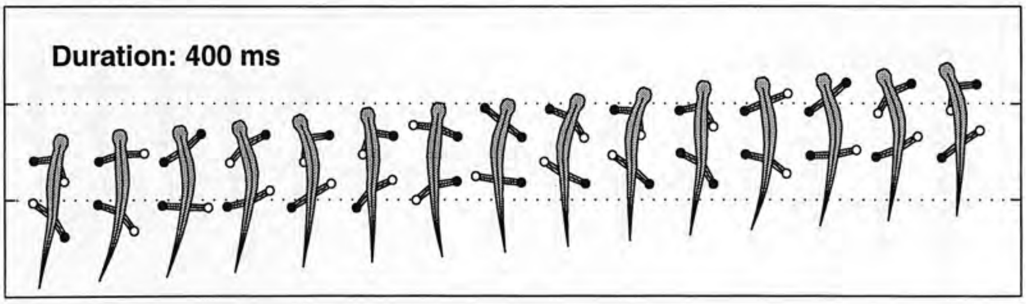
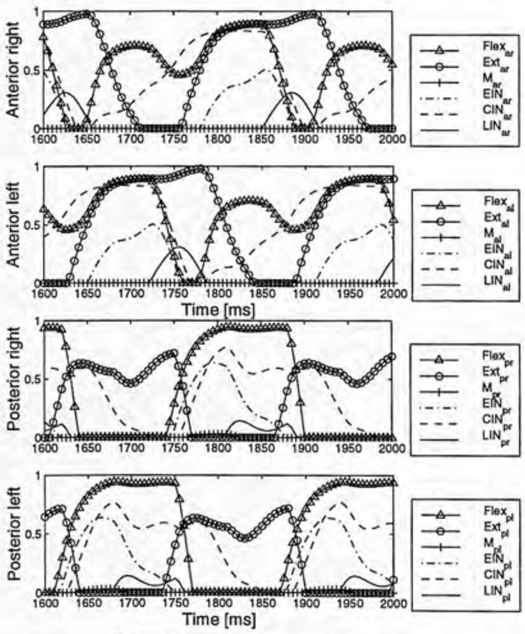


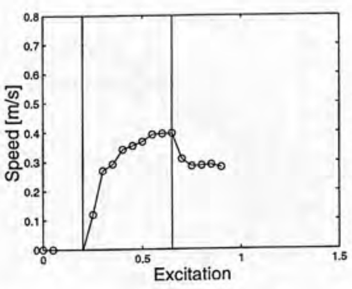
Figure I.7: **Run_A17:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

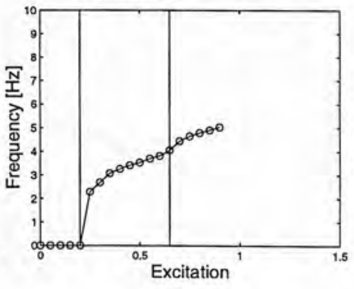
	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_al	–	7.8	-2.9	-0.5	–	–	2.9	–
L_al	–	10.3	–	-4.9	6.8	-1.5	–	–
E_ar	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_ar	-0.5	–	–	–	7.8	-2.9	–	2.9
L_ar	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_al	-0.2	–	-4.4	-0.7	1.5	-1.7	11.2	16.7
Ext_al	-0.2	–	–	-2.9	10.9	-1.2	17.4	8.2
Flex_ar	-0.7	1.5	-1.7	-0.2	–	-4.4	16.7	11.2
Ext_ar	-2.9	10.9	-1.2	-0.2	–	–	8.2	17.4
Trunk_l	–	3.2	–	-0.5	–	–	-2.4	-6.1
Trunk_r	-0.5	–	–	–	3.2	–	-6.1	-2.4
E_pl	-1.0	5.3	-1.4	–	4.4	-0.0	–	–
C_pl	-0.1	–	-0.1	-1.7	1.5	-1.1	–	–
L_pl	-0.1	–	-0.3	-0.1	6.0	–	–	–
E_pr	–	4.4	-0.0	-1.0	5.3	-1.4	–	–
C_pr	-1.7	1.5	-1.1	-0.1	–	-0.1	–	–
L_pr	-0.1	6.0	–	-0.1	–	-0.3	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_pl	–	7.8	-2.9	-0.5	–	–	2.9	–
L_pl	–	10.3	–	-4.9	6.8	-1.5	–	–
E_pr	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_pr	-0.5	–	–	–	7.8	-2.9	–	2.9
L_pr	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_pl	–	1.2	-0.5	-5.0	–	-0.5	9.8	9.7
Ext_pl	-4.5	1.2	–	-0.7	9.1	-2.1	14.2	6.2
Flex_pr	-5.0	–	-0.5	–	1.2	-0.5	9.7	9.8
Ext_pr	-0.7	9.1	-2.1	-4.5	1.2	–	6.2	14.2
Tail_l	–	–	-0.5	-0.3	0.1	-0.2	1.6	0.7
Tail_r	-0.3	0.1	-0.2	–	–	-0.5	0.7	1.6
E_al	-0.5	1.2	-1.7	-1.5	5.9	-0.9	–	–
C_al	–	1.7	–	-0.8	0.7	-0.3	–	–
L_al	-1.4	3.8	-0.6	-1.3	–	-0.1	–	–
E_ar	-1.5	5.9	-0.9	-0.5	1.2	-1.7	–	–
C_ar	-0.8	0.7	-0.3	–	1.7	–	–	–
L_ar	-1.3	–	-0.1	-1.4	3.8	-0.6	–	–









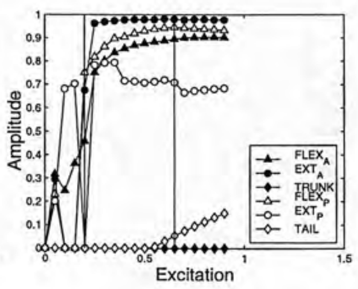
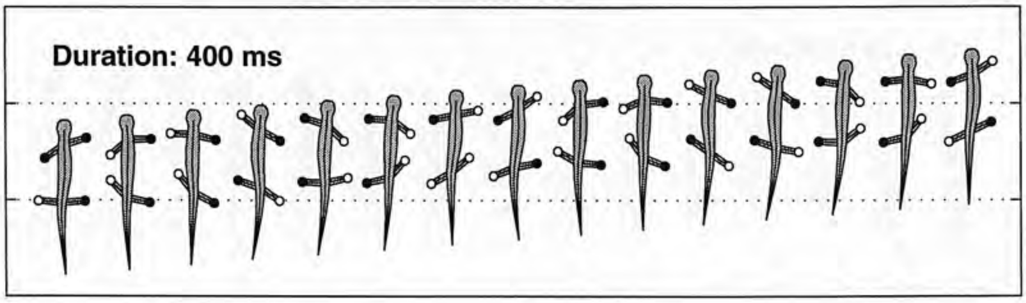
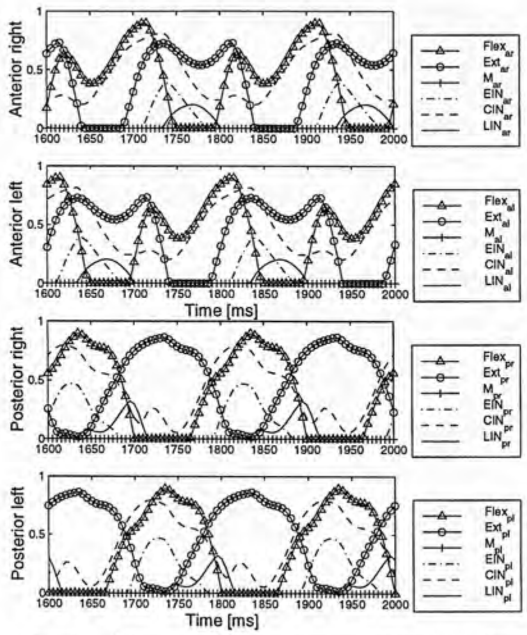


Figure I.8: **Run_A18:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_al	–	7.8	-2.9	-0.5	–	–	2.9	–
L_al	–	10.3	–	-4.9	6.8	-1.5	–	–
E_ar	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_ar	-0.5	–	–	–	7.8	-2.9	–	2.9
L_ar	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_al	-3.0	–	-5.0	-2.0	–	–	7.3	4.2
Ext_al	-0.3	2.6	-1.5	-3.6	1.1	-2.6	1.1	3.3
Flex_ar	-2.0	–	–	-3.0	–	-5.0	4.2	7.3
Ext_ar	-3.6	1.1	-2.6	-0.3	2.6	-1.5	3.3	1.1
Trunk_l	-4.2	0.7	–	-0.1	6.7	-2.1	-3.4	-5.1
Trunk_r	-0.1	6.7	-2.1	-4.2	0.7	–	-5.1	-3.4
E_pl	-1.5	–	–	-1.2	0.6	-1.8	–	–
C_pl	-2.0	–	-0.0	-0.5	1.0	-0.3	–	–
L_pl	-1.1	4.1	-0.4	-0.3	0.2	–	–	–
E_pr	-1.2	0.6	-1.8	-1.5	–	–	–	–
C_pr	-0.5	1.0	-0.3	-2.0	–	-0.0	–	–
L_pr	-0.3	0.2	–	-1.1	4.1	-0.4	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_pl	–	7.8	-2.9	-0.5	–	–	2.9	–
L_pl	–	10.3	–	-4.9	6.8	-1.5	–	–
E_pr	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_pr	-0.5	–	–	–	7.8	-2.9	–	2.9
L_pr	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_pl	–	–	-4.4	-3.4	1.3	-1.1	20.0	8.1
Ext_pl	-1.8	–	–	–	1.7	-2.7	2.0	3.5
Flex_pr	-3.4	1.3	-1.1	–	–	-4.4	8.1	20.0
Ext_pr	–	1.7	-2.7	-1.8	–	–	3.5	2.0
Tail_l	-0.5	0.1	-0.3	-0.1	0.2	–	-0.5	-0.2
Tail_r	-0.1	0.2	–	-0.5	0.1	-0.3	-0.2	-0.5
E_al	-1.8	–	-2.0	-0.3	–	-1.8	–	–
C_al	-1.2	5.4	-0.2	-0.0	–	-1.1	–	–
L_al	-1.8	0.6	-1.5	-0.3	2.7	-1.8	–	–
E_ar	-0.3	–	-1.8	-1.8	–	-2.0	–	–
C_ar	-0.0	–	-1.1	-1.2	5.4	-0.2	–	–
L_ar	-0.3	2.7	-1.8	-1.8	0.6	-1.5	–	–





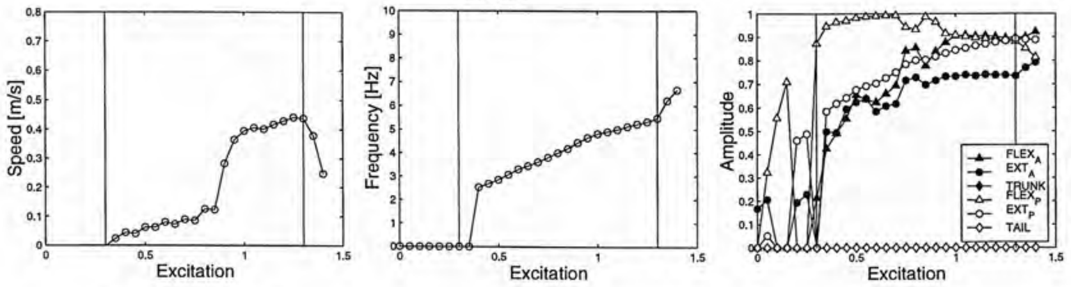
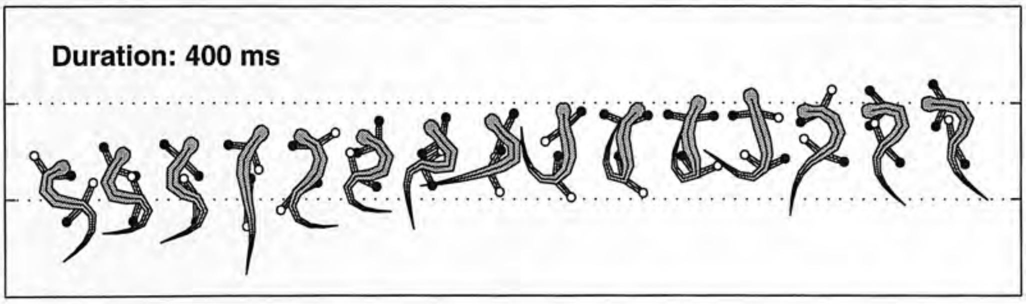
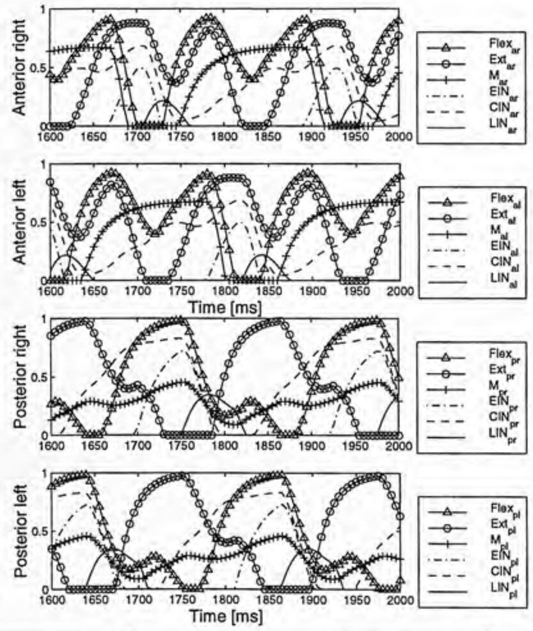


Figure I.9: **Run_A19:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_al	–	7.8	-2.9	-0.5	–	–	2.9	–
L_al	–	10.3	–	-4.9	6.8	-1.5	–	–
E_ar	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_ar	-0.5	–	–	–	7.8	-2.9	–	2.9
L_ar	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_al	-3.9	–	–	-1.5	15.0	-1.1	6.2	18.7
Ext_al	-0.3	11.2	-3.8	-2.8	12.2	-3.2	3.6	5.8
Flex_ar	-1.5	15.0	-1.1	-3.9	–	–	18.7	6.2
Ext_ar	-2.8	12.2	-3.2	-0.3	11.2	-3.8	5.8	3.6
Trunk_l	-5.0	9.1	-0.9	–	0.8	–	6.1	7.5
Trunk_r	–	0.8	–	-5.0	9.1	-0.9	7.5	6.1
E_pl	-0.0	2.1	–	-2.0	–	–	–	–
C_pl	-0.1	–	–	-0.4	–	-0.2	–	–
L_pl	-0.7	–	–	-0.2	4.0	-2.0	–	–
E_pr	-2.0	–	–	-0.0	2.1	–	–	–
C_pr	-0.4	–	-0.2	-0.1	–	–	–	–
L_pr	-0.2	4.0	-2.0	-0.7	–	–	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	–	8.6	-3.8	-4.3	–	-0.1	-0.1	–
C_pl	–	7.8	-2.9	-0.5	–	–	2.9	–
L_pl	–	10.3	–	-4.9	6.8	-1.5	–	–
E_pr	-4.3	–	-0.1	–	8.6	-3.8	–	-0.1
C_pr	-0.5	–	–	–	7.8	-2.9	–	2.9
L_pr	-4.9	6.8	-1.5	–	10.3	–	–	–
Flex_pl	–	10.4	-2.9	-1.6	2.4	-0.1	4.7	12.8
Ext_pl	-2.5	–	–	–	7.2	-2.1	7.4	16.6
Flex_pr	-1.6	2.4	-0.1	–	10.4	-2.9	12.8	4.7
Ext_pr	–	7.2	-2.1	-2.5	–	–	16.6	7.4
Tail_l	–	3.0	-0.9	-0.1	0.8	-0.5	2.5	2.1
Tail_r	-0.1	0.8	-0.5	–	3.0	-0.9	2.1	2.5
E_al	-1.9	6.0	-0.9	-1.2	5.3	-0.6	–	–
C_al	-0.7	0.8	-1.4	-0.6	3.5	-0.5	–	–
L_al	-0.6	4.2	-0.4	-1.1	0.9	-0.5	–	–
E_ar	-1.2	5.3	-0.6	-1.9	6.0	-0.9	–	–
C_ar	-0.6	3.5	-0.5	-0.7	0.8	-1.4	–	–
L_ar	-1.1	0.9	-0.5	-0.6	4.2	-0.4	–	–





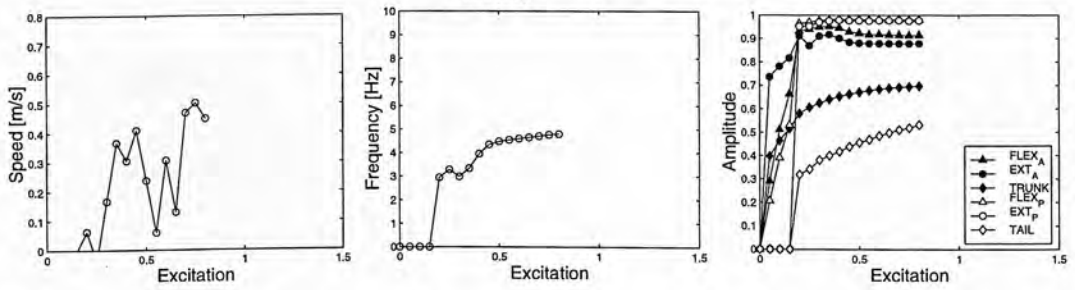


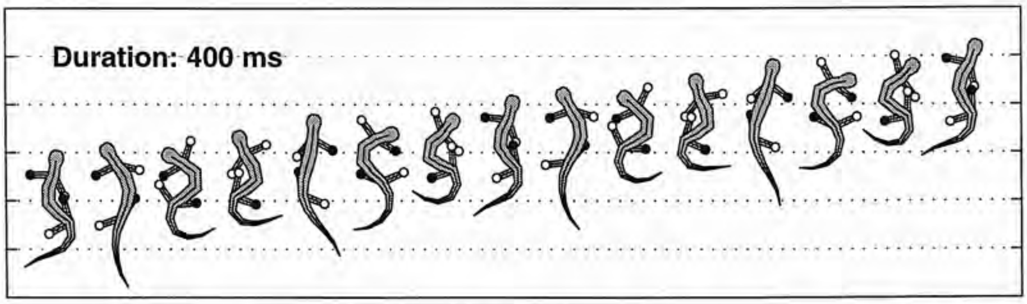
Figure I.10: **Run_A20:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

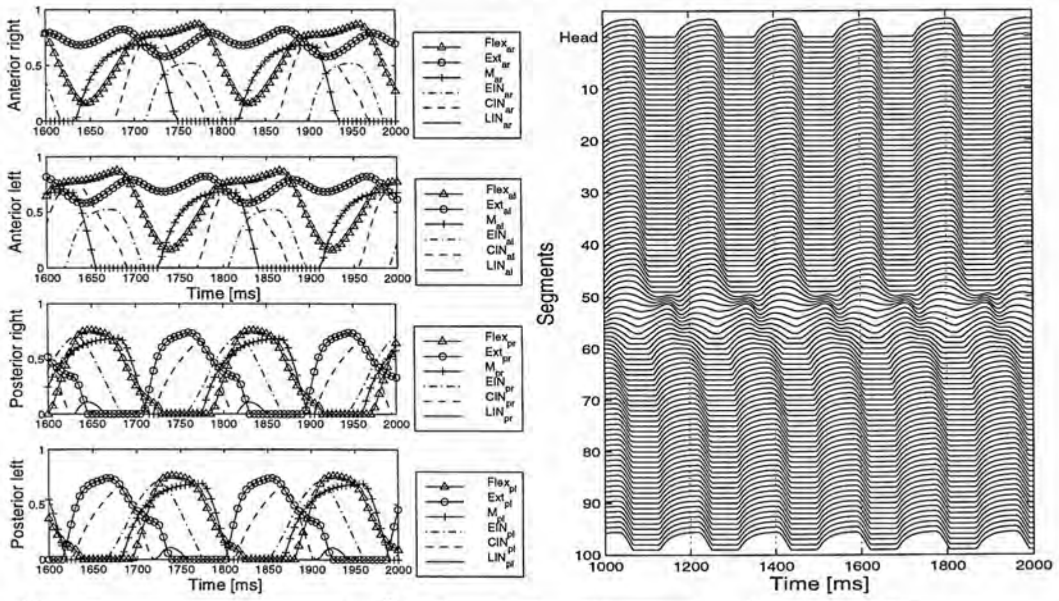
Appendix J

Results of chapter 7: Runs B1 to B10

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	0.9	-2.8	-4.4	–	–	-4.5	8.9	16.7
Ext_al	1.1	-1.2	-3.6	2.3	–	–	2.6	1.6
Flex_ar	–	–	-4.5	0.9	-2.8	-4.4	16.7	8.9
Ext_ar	2.3	–	–	1.1	-1.2	-3.6	1.6	2.6
E_pl	3.0	-1.9	-1.6	5.0	-1.1	-0.8	–	–
C_pl	1.5	–	–	5.2	–	–	–	–
L_pl	–	–	–	–	-0.4	-1.2	–	–
E_pr	5.0	-1.1	-0.8	3.0	-1.9	-1.6	–	–
C_pr	5.2	–	–	1.5	–	–	–	–
L_pr	–	-0.4	-1.2	–	–	–	–	–
E_trunk_l	0.3	–	-2.0	1.6	–	-0.3	–	–
C_trunk_l	1.6	-0.8	-1.4	1.2	-1.4	-0.9	–	–
L_trunk_l	–	-1.8	-1.6	–	-1.3	-1.5	–	–
E_trunk_r	1.6	–	-0.3	0.3	–	-2.0	–	–
C_trunk_r	1.2	-1.4	-0.9	1.6	-0.8	-1.4	–	–
L_trunk_r	–	-1.3	-1.5	–	-1.8	-1.6	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	–	-3.0	-1.2	–	-0.4	–	-4.1	-2.1
Ext_pl	7.6	-3.9	-1.7	15.0	-3.4	-4.7	11.7	5.5
Flex_pr	–	-0.4	–	–	-3.0	-1.2	-2.1	-4.1
Ext_pr	15.0	-3.4	-4.7	7.6	-3.9	-1.7	5.5	11.7
E_al	6.0	-0.2	-0.6	3.4	-0.7	-0.2	–	–
C_al	–	-1.9	-0.6	0.2	-1.8	-1.2	–	–
L_al	0.1	-0.6	-1.5	2.0	-0.7	-1.2	–	–
E_ar	3.4	-0.7	-0.2	6.0	-0.2	-0.6	–	–
C_ar	0.2	-1.8	-1.2	–	-1.9	-0.6	–	–
L_ar	2.0	-0.7	-1.2	0.1	-0.6	-1.5	–	–
E_tail_l	0.4	-1.5	-1.4	0.8	-1.4	-1.1	–	–
C_tail_l	4.5	-1.2	-0.2	4.7	-2.0	-0.7	–	–
L_tail_l	0.0	-0.1	-0.6	4.4	-0.7	-0.8	–	–
E_tail_r	0.8	-1.4	-1.1	0.4	-1.5	-1.4	–	–
C_tail_r	4.7	-2.0	-0.7	4.5	-1.2	-0.2	–	–
L_tail_r	4.4	-0.7	-0.8	0.0	-0.1	-0.6	–	–





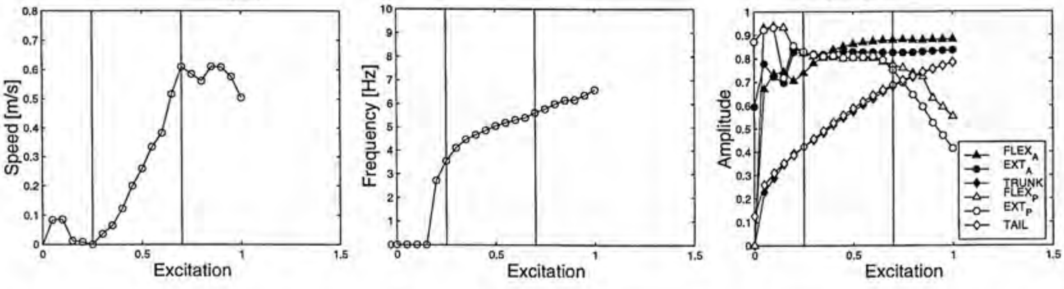
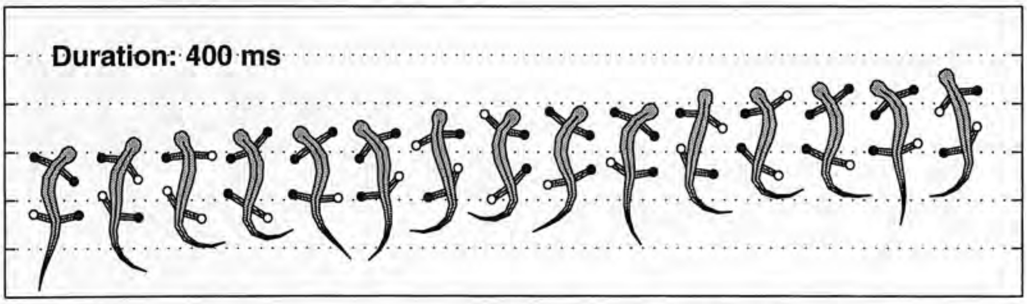
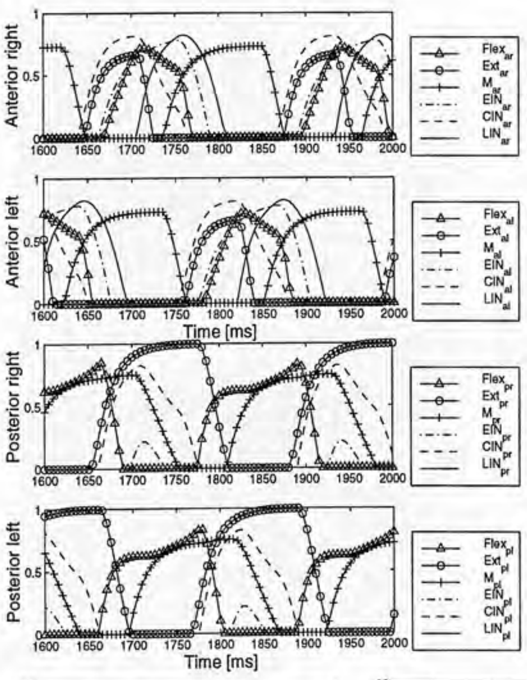


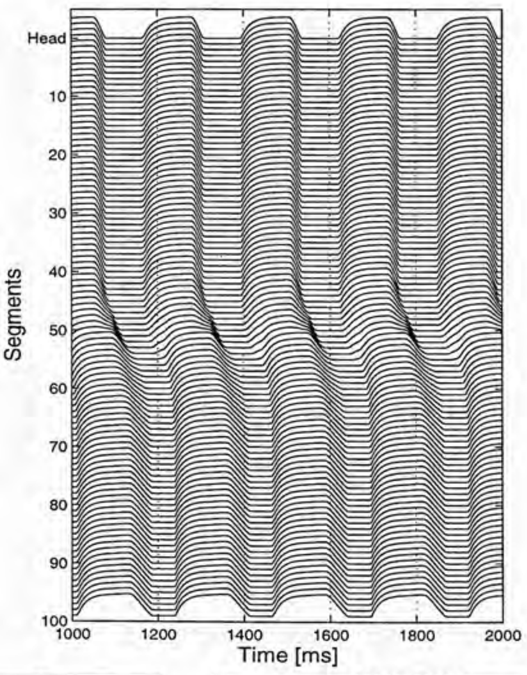
Figure J.1: **Run_B1:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

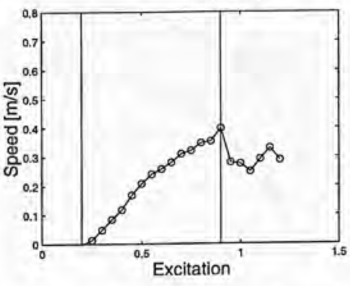
	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	4.9	-4.8	-2.9	0.9	-0.7	-3.3	16.2	5.4
Ext_al	2.7	-4.8	-2.0	–	–	–	-3.4	-6.6
Flex_ar	0.9	-0.7	-3.3	4.9	-4.8	-2.9	5.4	16.2
Ext_ar	–	–	–	2.7	-4.8	-2.0	-6.6	-3.4
E_pl	5.5	-1.3	-0.1	5.5	-1.5	-0.4	–	–
C_pl	–	–	-1.9	1.9	–	-0.4	–	–
L_pl	0.0	-1.7	-0.3	1.7	-0.9	–	–	–
E_pr	5.5	-1.5	-0.4	5.5	-1.3	-0.1	–	–
C_pr	1.9	–	-0.4	–	–	-1.9	–	–
L_pr	1.7	-0.9	–	0.0	-1.7	-0.3	–	–
E_trunk_l	0.8	–	-1.7	–	-1.4	-0.4	–	–
C_trunk_l	–	-0.7	-2.0	–	-0.4	-1.1	–	–
L_trunk_l	–	–	-1.9	0.6	–	–	–	–
E_trunk_r	–	-1.4	-0.4	0.8	–	-1.7	–	–
C_trunk_r	–	-0.4	-1.1	–	-0.7	-2.0	–	–
L_trunk_r	0.6	–	–	–	–	-1.9	–	–

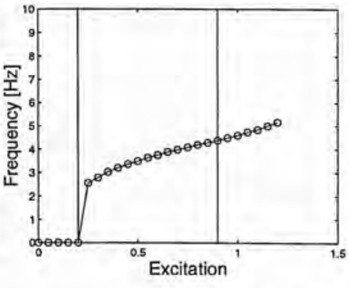
	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	–	-3.1	-1.1	10.0	-1.1	-3.7	-2.9	-7.1
Ext_pl	6.4	-0.2	-0.9	9.2	-2.1	-1.7	-2.5	-7.5
Flex_pr	10.0	-1.1	-3.7	–	-3.1	-1.1	-7.1	-2.9
Ext_pr	9.2	-2.1	-1.7	6.4	-0.2	-0.9	-7.5	-2.5
E_al	0.8	-1.0	-1.2	–	-0.3	-1.2	–	–
C_al	1.5	-0.1	-1.8	–	-1.0	-0.5	–	–
L_al	5.8	-0.7	-0.5	2.6	-0.8	-1.2	–	–
E_ar	–	-0.3	-1.2	0.8	-1.0	-1.2	–	–
C_ar	–	-1.0	-0.5	1.5	-0.1	-1.8	–	–
L_ar	2.6	-0.8	-1.2	5.8	-0.7	-0.5	–	–
E_tail_l	5.5	-2.0	-1.4	2.2	-1.2	-1.8	–	–
C_tail_l	4.3	-0.8	-0.7	5.4	-0.0	-1.1	–	–
L_tail_l	5.3	-1.1	-0.5	–	-0.6	-0.5	–	–
E_tail_r	2.2	-1.2	-1.8	5.5	-2.0	-1.4	–	–
C_tail_r	5.4	-0.0	-1.1	4.3	-0.8	-0.7	–	–
L_tail_r	–	-0.6	-0.5	5.3	-1.1	-0.5	–	–











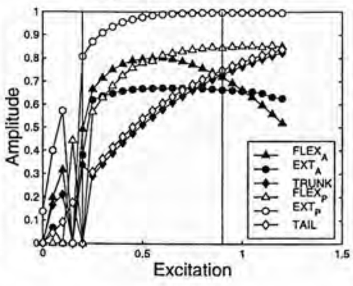
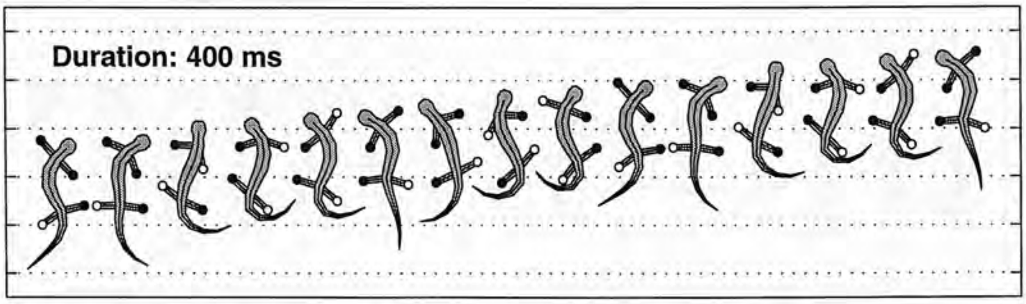
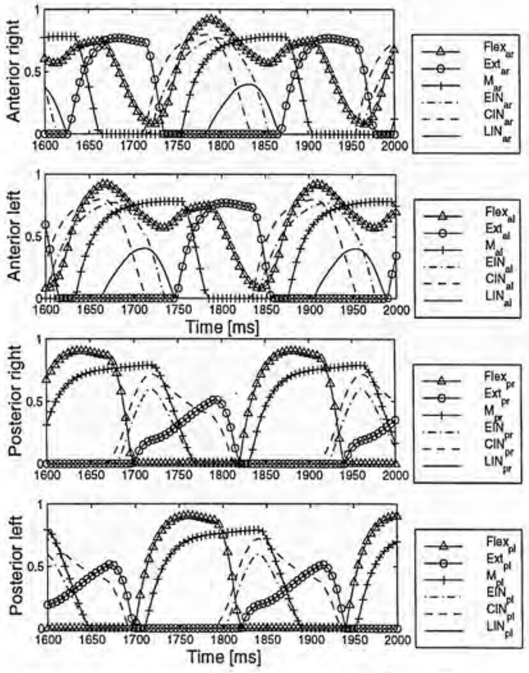


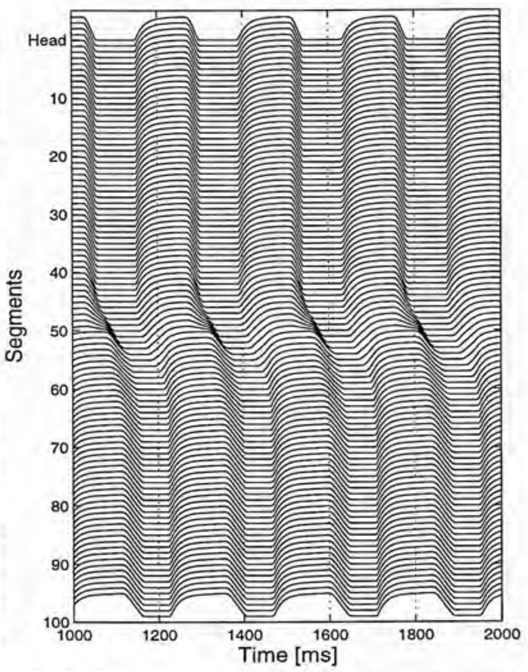
Figure J.2: **Run_B2:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	1.2	-4.1	-4.6	1.2	-1.1	-3.2	-2.5	-7.5
Ext_al	9.0	-1.6	-3.3	13.3	–	-0.3	4.2	12.6
Flex_ar	1.2	-1.1	-3.2	1.2	-4.1	-4.6	-7.5	-2.5
Ext_ar	13.3	–	-0.3	9.0	-1.6	-3.3	12.6	4.2
E_pl	0.2	-0.1	-0.1	3.0	–	–	–	–
C_pl	0.4	-1.9	-1.2	0.7	-0.8	-0.0	–	–
L_pl	–	-0.8	–	2.8	-1.5	-1.3	–	–
E_pr	3.0	–	–	0.2	-0.1	-0.1	–	–
C_pr	0.7	-0.8	-0.0	0.4	-1.9	-1.2	–	–
L_pr	2.8	-1.5	-1.3	–	-0.8	–	–	–
E_trunk_l	4.3	-1.8	-0.3	3.8	-1.1	-1.8	–	–
C_trunk_l	3.7	–	-1.7	–	-1.1	-0.9	–	–
L_trunk_l	0.2	-0.6	-1.4	–	–	–	–	–
E_trunk_r	3.8	-1.1	-1.8	4.3	-1.8	-0.3	–	–
C_trunk_r	–	-1.1	-0.9	3.7	–	-1.7	–	–
L_trunk_r	–	–	–	0.2	-0.6	-1.4	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	1.9	-0.3	-3.8	3.6	-2.4	-1.6	-6.4	-3.6
Ext_pl	15.0	-0.3	-2.0	0.1	-3.7	-4.4	2.3	6.8
Flex_pr	3.6	-2.4	-1.6	1.9	-0.3	-3.8	-3.6	-6.4
Ext_pr	0.1	-3.7	-4.4	15.0	-0.3	-2.0	6.8	2.3
E_al	2.7	-0.9	-0.6	2.6	-1.4	-0.4	–	–
C_al	1.5	-1.0	-0.5	–	-0.7	-0.9	–	–
L_al	5.0	-0.5	-0.6	1.0	-1.6	-0.0	–	–
E_ar	2.6	-1.4	-0.4	2.7	-0.9	-0.6	–	–
C_ar	–	-0.7	-0.9	1.5	-1.0	-0.5	–	–
L_ar	1.0	-1.6	-0.0	5.0	-0.5	-0.6	–	–
E_tail_l	–	-0.8	-1.8	0.1	-0.8	-1.6	–	–
C_tail_l	2.9	-0.5	-0.3	0.0	-1.3	-1.6	–	–
L_tail_l	4.2	-1.3	-0.8	4.7	-0.8	-1.4	–	–
E_tail_r	0.1	-0.8	-1.6	–	-0.8	-1.8	–	–
C_tail_r	0.0	-1.3	-1.6	2.9	-0.5	-0.3	–	–
L_tail_r	4.7	-0.8	-1.4	4.2	-1.3	-0.8	–	–







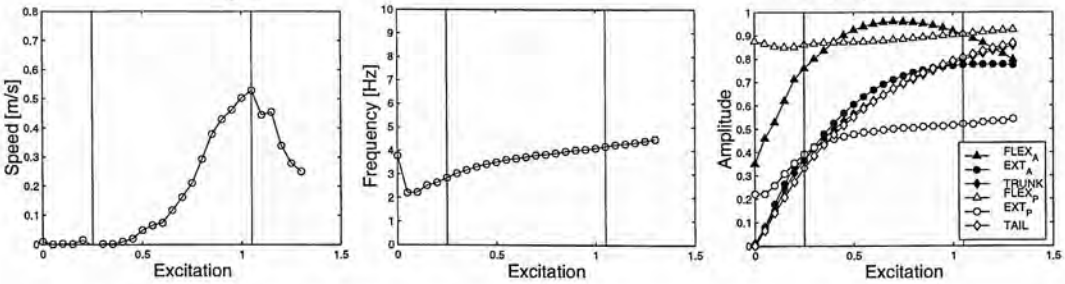
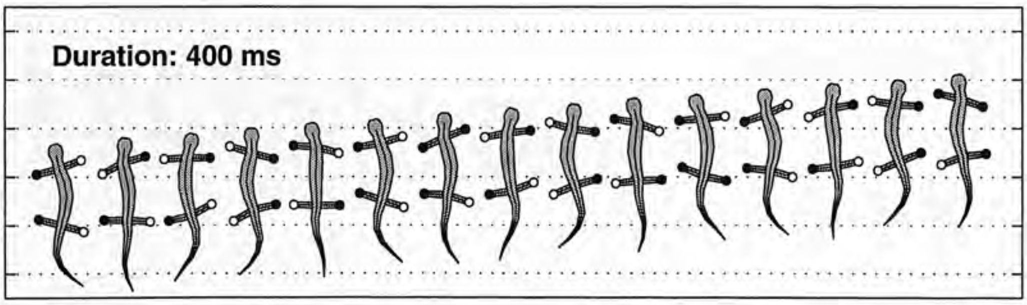
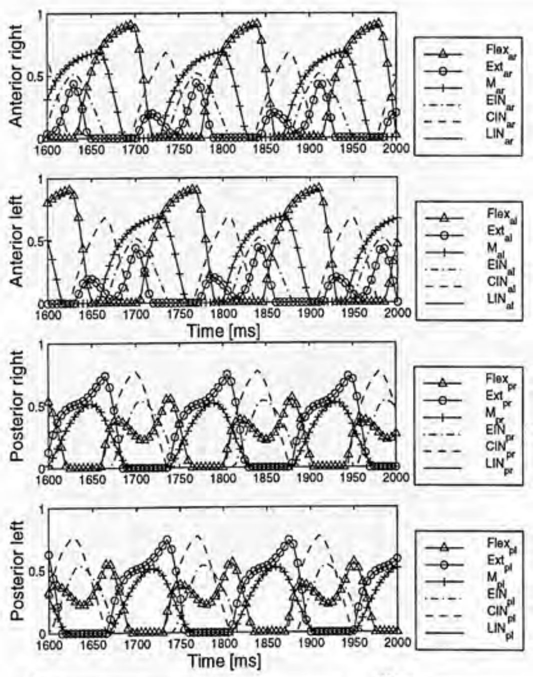


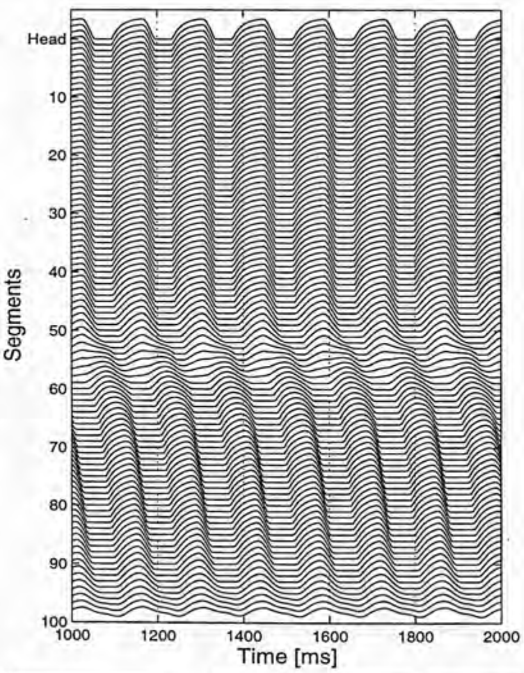
Figure J.3: **Run_B3:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	–	-3.7	-3.0	–	-2.3	-0.2	0.7	2.0
Ext_al	–	-5.0	-1.9	9.8	-1.4	-1.5	10.6	10.1
Flex_ar	–	-2.3	-0.2	–	-3.7	-3.0	2.0	0.7
Ext_ar	9.8	-1.4	-1.5	–	-5.0	-1.9	10.1	10.6
E_pl	–	-0.3	-1.6	5.8	-0.1	-1.6	–	–
C_pl	–	–	-0.1	–	-1.1	–	–	–
L_pl	0.4	-0.0	-0.2	2.7	–	-1.3	–	–
E_pr	5.8	-0.1	-1.6	–	-0.3	-1.6	–	–
C_pr	–	-1.1	–	–	–	-0.1	–	–
L_pr	2.7	–	-1.3	0.4	-0.0	-0.2	–	–
E_trunk_l	0.5	–	-1.2	1.6	-0.3	-0.3	–	–
C_trunk_l	0.1	-0.2	-1.8	–	–	-1.9	–	–
L_trunk_l	6.0	-1.3	–	4.8	-1.2	-1.4	–	–
E_trunk_r	1.6	-0.3	-0.3	0.5	–	-1.2	–	–
C_trunk_r	–	–	-1.9	0.1	-0.2	-1.8	–	–
L_trunk_r	4.8	-1.2	-1.4	6.0	-1.3	–	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	2.4	–	-3.1	8.0	-5.0	-3.3	1.3	2.1
Ext_pl	3.9	–	-3.0	–	-1.2	-2.3	2.9	7.0
Flex_pr	8.0	-5.0	-3.3	2.4	–	-3.1	2.1	1.3
Ext_pr	–	-1.2	-2.3	3.9	–	-3.0	7.0	2.9
E_al	2.9	-1.9	-1.0	1.2	-0.6	-1.9	–	–
C_al	2.8	-2.0	-1.2	1.5	-1.3	-0.2	–	–
L_al	–	-2.0	-1.9	4.2	-1.8	-1.0	–	–
E_ar	1.2	-0.6	-1.9	2.9	-1.9	-1.0	–	–
C_ar	1.5	-1.3	-0.2	2.8	-2.0	-1.2	–	–
L_ar	4.2	-1.8	-1.0	–	-2.0	-1.9	–	–
E_tail_l	0.9	-0.7	–	3.2	-1.5	-1.6	–	–
C_tail_l	0.8	-0.1	-1.1	3.6	-1.0	-1.1	–	–
L_tail_l	–	-1.6	-2.0	0.1	-0.1	-0.6	–	–
E_tail_r	3.2	-1.5	-1.6	0.9	-0.7	–	–	–
C_tail_r	3.6	-1.0	-1.1	0.8	-0.1	-1.1	–	–
L_tail_r	0.1	-0.1	-0.6	–	-1.6	-2.0	–	–







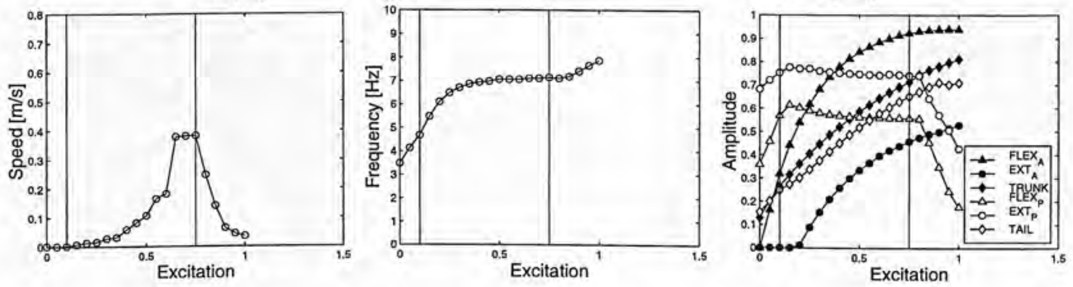
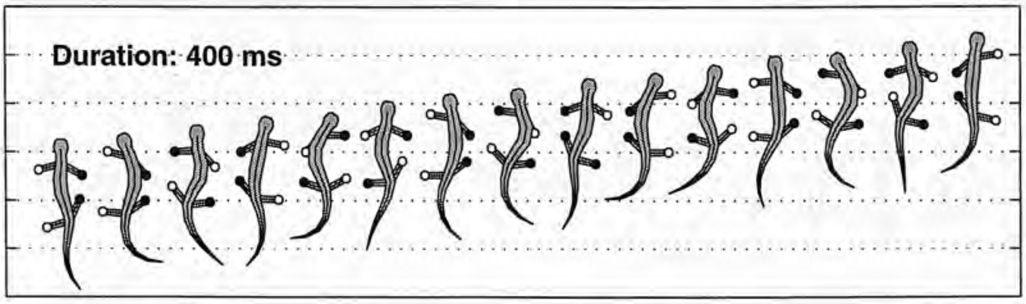
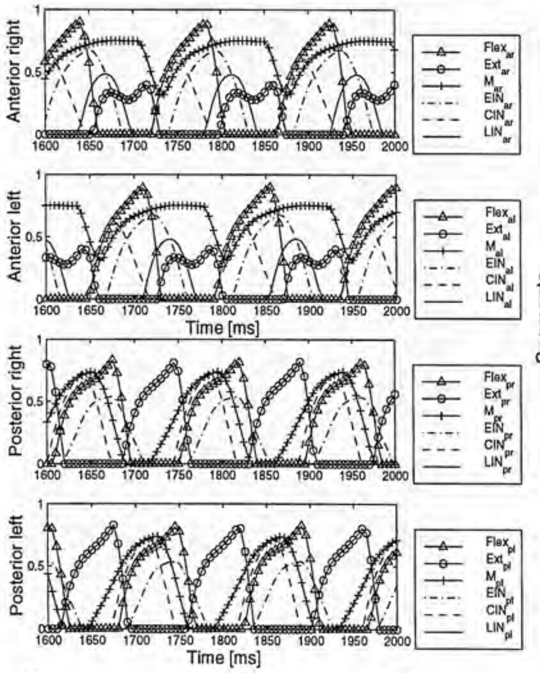


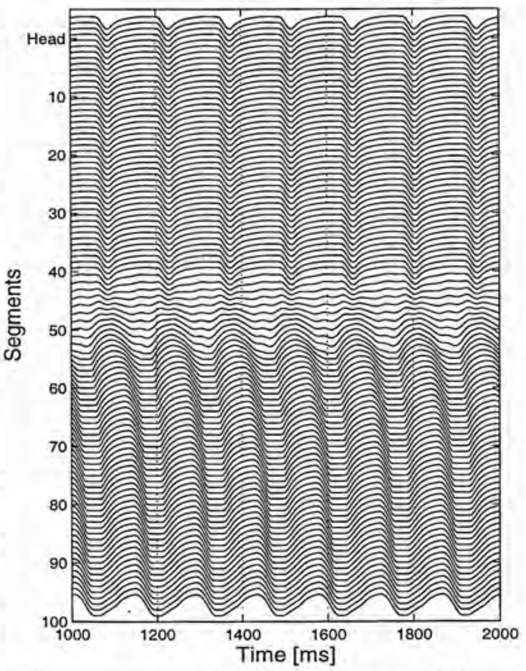
Figure J.4: **Run_B4:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	5.8	-3.0	-2.0	6.9	-4.4	-3.0	-2.9	-5.9
Ext_al	5.5	-0.5	-3.9	–	-0.1	-4.7	-3.1	-4.7
Flex_ar	6.9	-4.4	-3.0	5.8	-3.0	-2.0	-5.9	-2.9
Ext_ar	–	-0.1	-4.7	5.5	-0.5	-3.9	-4.7	-3.1
E_pl	1.5	-1.6	-0.2	0.6	-1.8	-0.4	–	–
C_pl	0.3	-0.2	–	–	-0.1	-1.4	–	–
L_pl	6.0	-1.5	-1.2	4.1	–	–	–	–
E_pr	0.6	-1.8	-0.4	1.5	-1.6	-0.2	–	–
C_pr	–	-0.1	-1.4	0.3	-0.2	–	–	–
L_pr	4.1	–	–	6.0	-1.5	-1.2	–	–
E_trunk_l	3.6	-2.0	-0.0	5.8	–	–	–	–
C_trunk_l	–	-0.8	-0.0	3.1	-0.5	-1.8	–	–
L_trunk_l	3.8	-1.7	-0.8	0.7	–	-1.5	–	–
E_trunk_r	5.8	–	–	3.6	-2.0	-0.0	–	–
C_trunk_r	3.1	-0.5	-1.8	–	-0.8	-0.0	–	–
L_trunk_r	0.7	–	-1.5	3.8	-1.7	-0.8	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	–	-2.8	-3.5	1.2	–	-2.5	21.3	7.8
Ext_pl	4.9	-3.9	-0.3	0.6	-3.2	-1.8	9.9	13.4
Flex_pr	1.2	–	-2.5	–	-2.8	-3.5	7.8	21.3
Ext_pr	0.6	-3.2	-1.8	4.9	-3.9	-0.3	13.4	9.9
E_al	5.3	-0.2	–	–	-1.0	-1.1	–	–
C_al	0.8	-1.8	-1.5	4.1	-1.5	-1.5	–	–
L_al	–	-0.6	-1.0	1.2	-1.1	-1.2	–	–
E_ar	–	-1.0	-1.1	5.3	-0.2	–	–	–
C_ar	4.1	-1.5	-1.5	0.8	-1.8	-1.5	–	–
L_ar	1.2	-1.1	-1.2	–	-0.6	-1.0	–	–
E_tail_l	3.3	–	-0.1	6.0	-0.3	-1.8	–	–
C_tail_l	–	-0.2	-2.0	2.7	-0.3	-1.4	–	–
L_tail_l	0.9	-1.6	-1.9	0.1	-1.5	-0.6	–	–
E_tail_r	6.0	-0.3	-1.8	3.3	–	-0.1	–	–
C_tail_r	2.7	-0.3	-1.4	–	-0.2	-2.0	–	–
L_tail_r	0.1	-1.5	-0.6	0.9	-1.6	-1.9	–	–







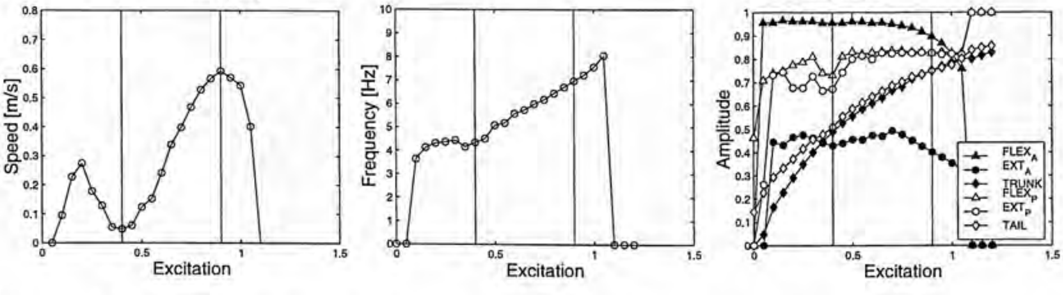
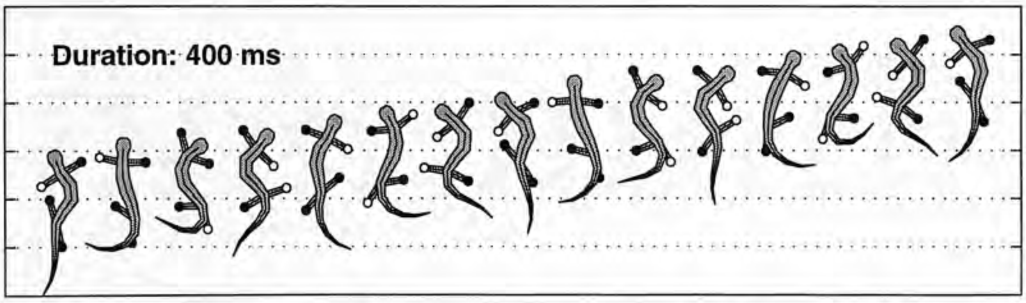
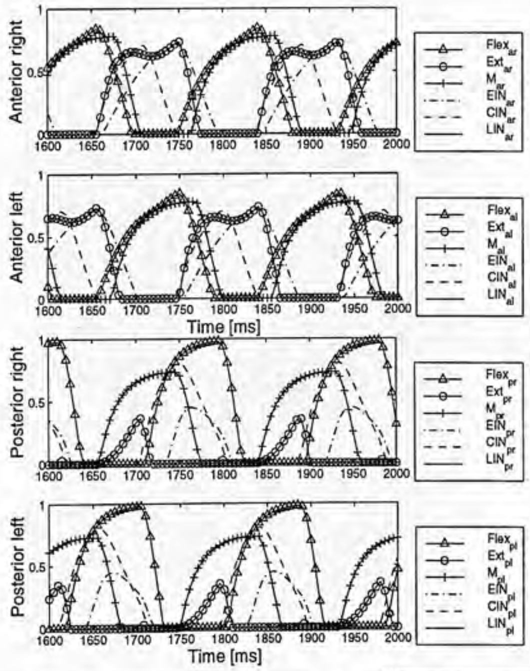


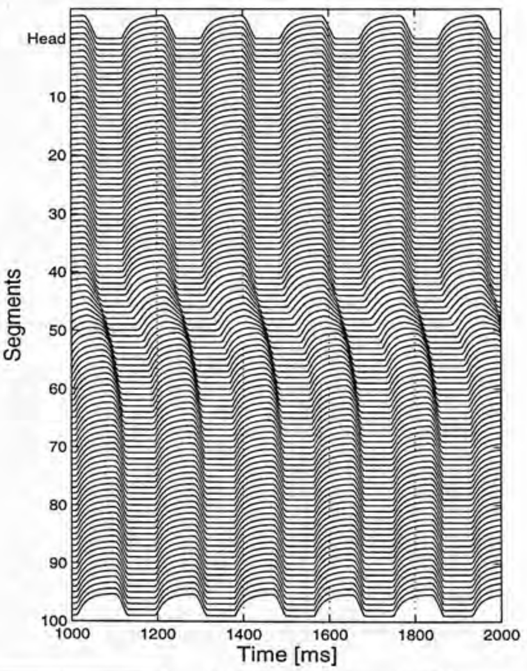
Figure J.5: **Run_B5:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

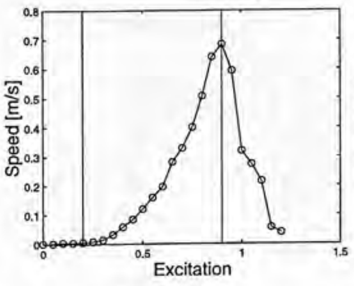
	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	0.7	-2.7	-0.0	–	-0.5	-0.8	8.2	4.8
Ext_al	4.3	–	-1.0	9.7	-4.0	-2.7	2.2	5.4
Flex_ar	–	-0.5	-0.8	0.7	-2.7	-0.0	4.8	8.2
Ext_ar	9.7	-4.0	-2.7	4.3	–	-1.0	5.4	2.2
E_pl	0.6	-1.8	-0.8	6.0	-1.8	-1.4	–	–
C_pl	0.4	-0.0	-0.2	–	-1.6	-0.3	–	–
L_pl	–	-0.5	-0.9	3.4	-0.0	-1.8	–	–
E_pr	6.0	-1.8	-1.4	0.6	-1.8	-0.8	–	–
C_pr	–	-1.6	-0.3	0.4	-0.0	-0.2	–	–
L_pr	3.4	-0.0	-1.8	–	-0.5	-0.9	–	–
E_trunk_l	1.9	-0.3	–	1.5	-1.8	–	–	–
C_trunk_l	3.1	-0.6	-0.3	–	-1.8	-0.7	–	–
L_trunk_l	2.6	–	–	2.8	-0.2	–	–	–
E_trunk_r	1.5	-1.8	–	1.9	-0.3	–	–	–
C_trunk_r	–	-1.8	-0.7	3.1	-0.6	-0.3	–	–
L_trunk_r	2.8	-0.2	–	2.6	–	–	–	–

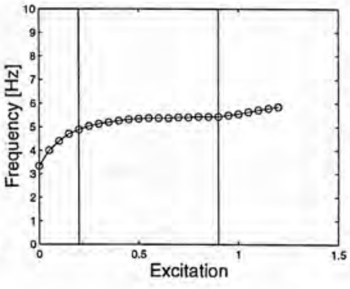
	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	14.3	–	-2.3	14.8	-5.0	-2.7	-2.5	-7.5
Ext_pl	11.6	-3.9	-1.4	4.9	-1.9	-3.0	4.4	10.3
Flex_pr	14.8	-5.0	-2.7	14.3	–	-2.3	-7.5	-2.5
Ext_pr	4.9	-1.9	-3.0	11.6	-3.9	-1.4	10.3	4.4
E_al	4.4	-0.3	–	–	-1.6	-0.6	–	–
C_al	3.1	-0.9	-0.5	4.7	-1.3	-0.7	–	–
L_al	0.5	-1.7	-1.3	4.3	-1.1	-0.9	–	–
E_ar	–	-1.6	-0.6	4.4	-0.3	–	–	–
C_ar	4.7	-1.3	-0.7	3.1	-0.9	-0.5	–	–
L_ar	4.3	-1.1	-0.9	0.5	-1.7	-1.3	–	–
E_tail_l	0.5	-0.1	-1.1	5.4	-1.0	-0.8	–	–
C_tail_l	5.6	-1.0	-1.9	3.2	–	-1.0	–	–
L_tail_l	–	–	-1.4	0.9	-1.8	-1.6	–	–
E_tail_r	5.4	-1.0	-0.8	0.5	-0.1	-1.1	–	–
C_tail_r	3.2	–	-1.0	5.6	-1.0	-1.9	–	–
L_tail_r	0.9	-1.8	-1.6	–	–	-1.4	–	–











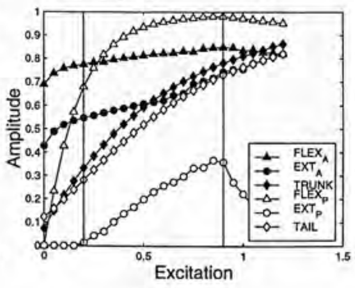
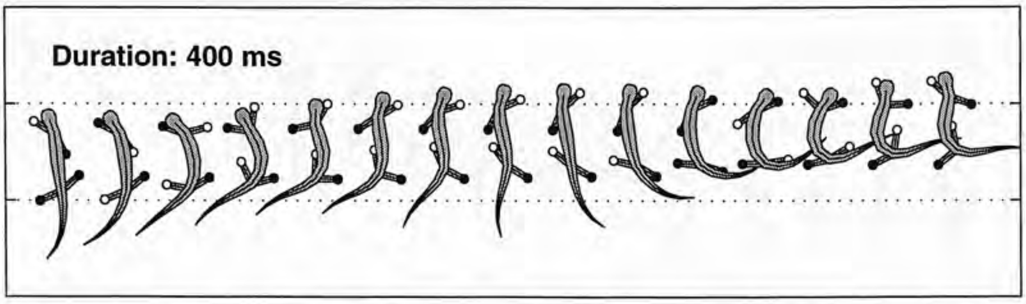
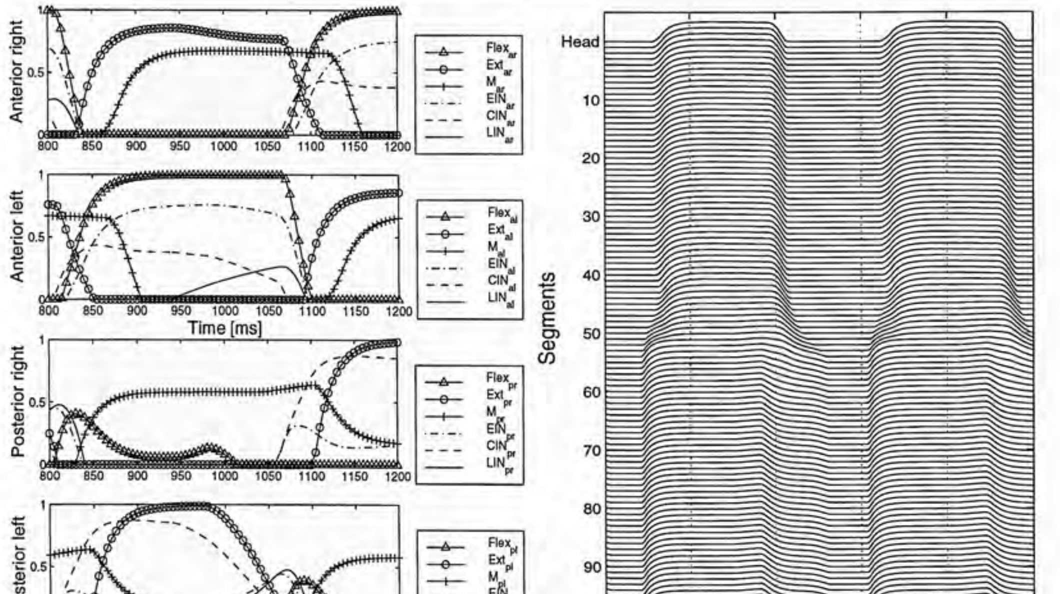


Figure J.6: **Run_B6:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	–	-2.9	-1.9	–	-4.2	-1.5	3.5	5.8
Ext_al	11.6	-2.3	-4.4	10.4	-4.3	-4.5	-3.0	-3.0
Flex_ar	–	-4.2	-1.5	–	-2.9	-1.9	5.8	3.5
Ext_ar	10.4	-4.3	-4.5	11.6	-2.3	-4.4	-3.0	-3.0
E_pl	–	–	-0.4	–	-1.6	-1.0	–	–
C_pl	2.7	-1.7	-1.9	5.1	-1.1	-0.7	–	–
L_pl	–	-0.8	-0.8	2.4	-0.6	-1.5	–	–
E_pr	–	-1.6	-1.0	–	–	-0.4	–	–
C_pr	5.1	-1.1	-0.7	2.7	-1.7	-1.9	–	–
L_pr	2.4	-0.6	-1.5	–	-0.8	-0.8	–	–
E_trunk_l	4.0	-0.2	-0.2	0.3	–	-1.0	–	–
C_trunk_l	5.4	–	-1.1	–	–	-1.0	–	–
L_trunk_l	–	-0.3	-0.4	3.3	-0.7	-0.6	–	–
E_trunk_r	0.3	–	-1.0	4.0	-0.2	-0.2	–	–
C_trunk_r	–	–	-1.0	5.4	–	-1.1	–	–
L_trunk_r	3.3	-0.7	-0.6	–	-0.3	-0.4	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	0.8	–	-1.0	13.0	-1.9	-2.6	-6.2	-3.8
Ext_pl	11.4	-1.7	-2.3	12.6	-2.2	-2.4	-2.6	-1.4
Flex_pr	13.0	-1.9	-2.6	0.8	–	-1.0	-3.8	-6.2
Ext_pr	12.6	-2.2	-2.4	11.4	-1.7	-2.3	-1.4	-2.6
E_al	0.0	-1.8	-0.6	–	-2.0	-0.9	–	–
C_al	3.2	-0.5	–	–	-1.8	-0.3	–	–
L_al	2.3	-0.3	-1.8	1.4	-1.7	-0.0	–	–
E_ar	–	-2.0	-0.9	0.0	-1.8	-0.6	–	–
C_ar	–	-1.8	-0.3	3.2	-0.5	–	–	–
L_ar	1.4	-1.7	-0.0	2.3	-0.3	-1.8	–	–
E_tail_l	0.6	-0.8	-0.9	–	-1.1	-1.6	–	–
C_tail_l	0.8	-1.1	-0.6	0.2	-1.3	-1.8	–	–
L_tail_l	3.6	–	-1.4	3.6	-1.5	-0.8	–	–
E_tail_r	–	-1.1	-1.6	0.6	-0.8	-0.9	–	–
C_tail_r	0.2	-1.3	-1.8	0.8	-1.1	-0.6	–	–
L_tail_r	3.6	-1.5	-0.8	3.6	–	-1.4	–	–





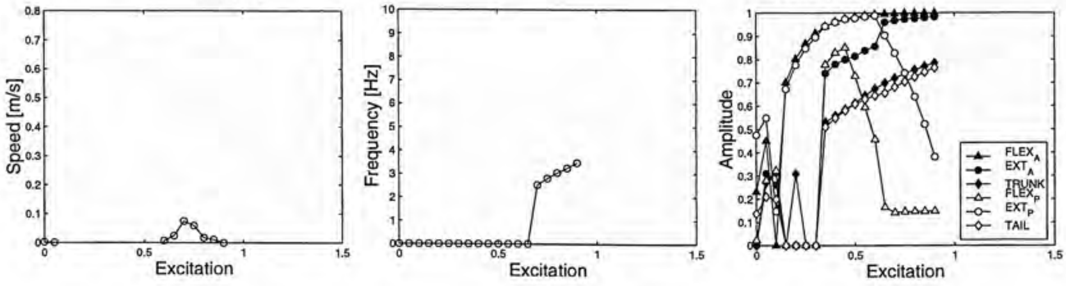
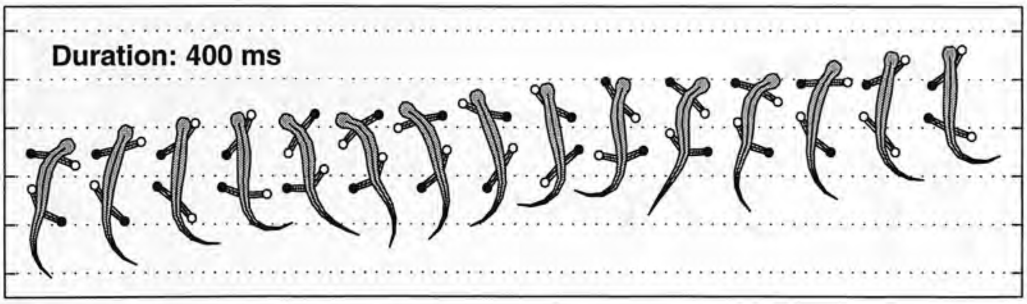
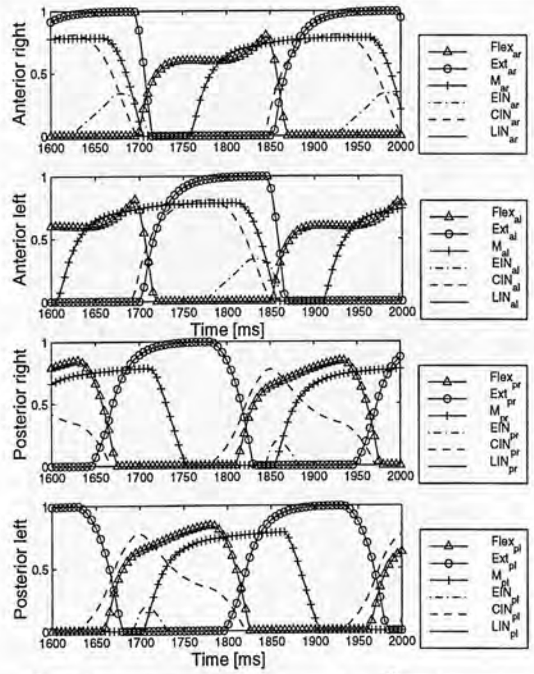


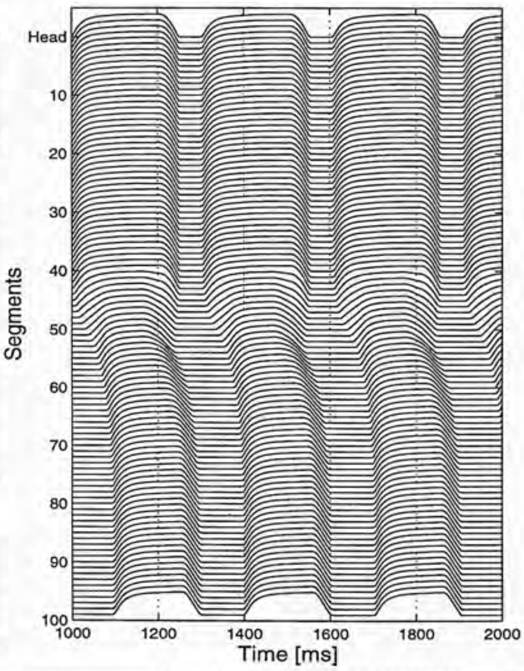
Figure J.7: **Run_B7:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	4.2	-4.4	-2.0	3.4	-2.7	-3.6	7.9	5.3
Ext_al	–	-2.8	-2.9	13.2	-5.0	-0.4	-3.7	-3.9
Flex_ar	3.4	-2.7	-3.6	4.2	-4.4	-2.0	5.3	7.9
Ext_ar	13.2	-5.0	-0.4	–	-2.8	-2.9	-3.9	-3.7
E_pl	–	–	–	0.1	-1.4	-0.2	–	–
C_pl	0.2	-1.0	-0.0	0.3	-0.9	–	–	–
L_pl	4.4	–	-0.4	–	-0.1	-0.2	–	–
E_pr	0.1	-1.4	-0.2	–	–	–	–	–
C_pr	0.3	-0.9	–	0.2	-1.0	-0.0	–	–
L_pr	–	-0.1	-0.2	4.4	–	-0.4	–	–
E_trunk_l	0.1	-1.9	-0.1	0.6	–	-0.6	–	–
C_trunk_l	0.3	–	-0.2	0.5	–	-0.8	–	–
L_trunk_l	3.2	–	-0.9	5.6	-1.9	–	–	–
E_trunk_r	0.6	–	-0.6	0.1	-1.9	-0.1	–	–
C_trunk_r	0.5	–	-0.8	0.3	–	-0.2	–	–
L_trunk_r	5.6	-1.9	–	3.2	–	-0.9	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	6.7	–	-5.0	0.9	-3.5	-1.2	-2.5	-7.5
Ext_pl	0.5	-0.1	-2.0	11.3	-0.6	-4.9	-5.8	-4.2
Flex_pr	0.9	-3.5	-1.2	6.7	–	-5.0	-7.5	-2.5
Ext_pr	11.3	-0.6	-4.9	0.5	-0.1	-2.0	-4.2	-5.8
E_al	5.4	-1.0	-1.0	–	-0.6	-0.6	–	–
C_al	0.4	-0.8	-0.6	–	–	-0.3	–	–
L_al	0.8	-1.2	-0.8	–	–	-0.1	–	–
E_ar	–	-0.6	-0.6	5.4	-1.0	-1.0	–	–
C_ar	–	–	-0.3	0.4	-0.8	-0.6	–	–
L_ar	–	–	-0.1	0.8	-1.2	-0.8	–	–
E_tail_l	0.7	-0.7	-0.2	–	-2.0	-0.9	–	–
C_tail_l	–	-0.4	-0.2	0.5	-0.5	-2.0	–	–
L_tail_l	2.9	-0.8	-1.3	1.4	-1.2	-0.4	–	–
E_tail_r	–	-2.0	-0.9	0.7	-0.7	-0.2	–	–
C_tail_r	0.5	-0.5	-2.0	–	-0.4	-0.2	–	–
L_tail_r	1.4	-1.2	-0.4	2.9	-0.8	-1.3	–	–







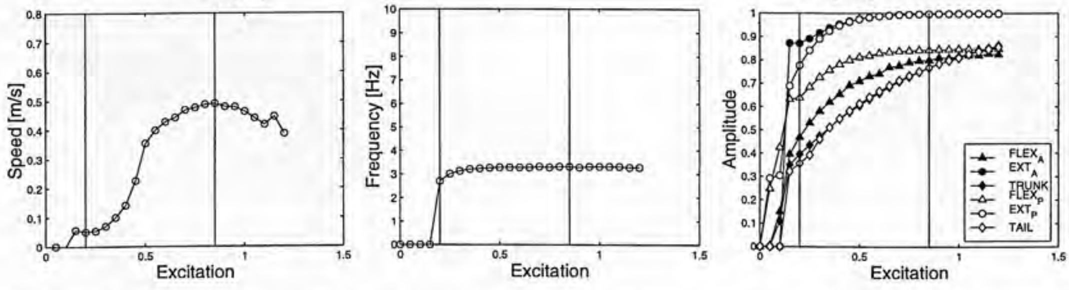
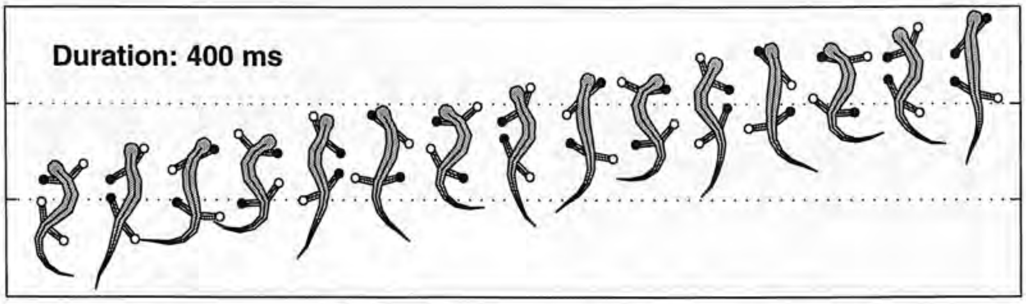
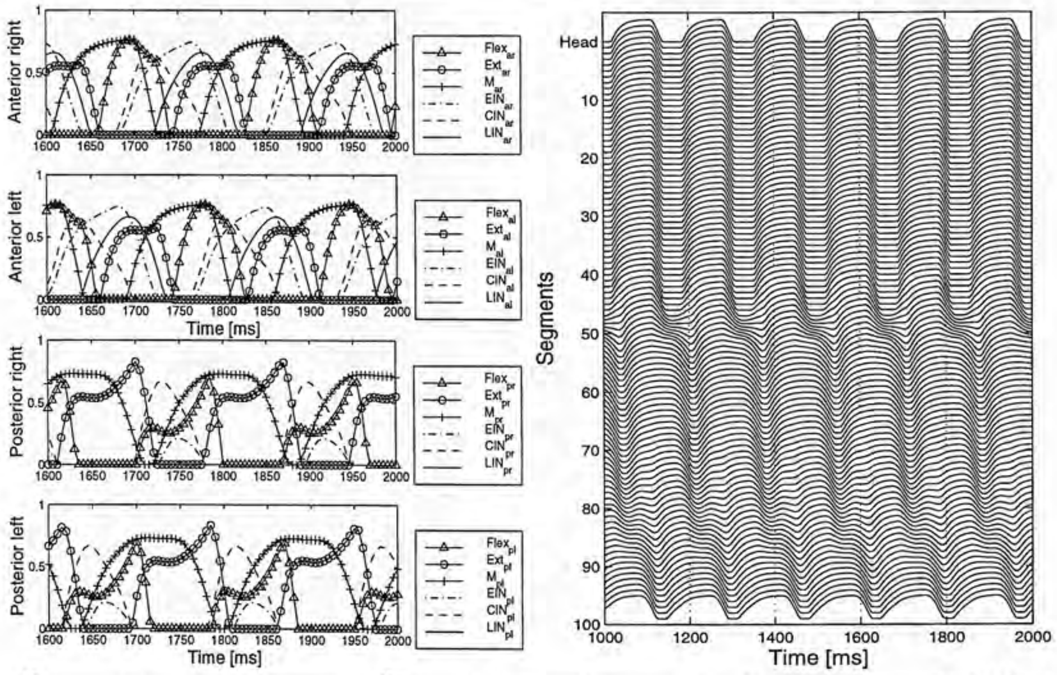


Figure J.8: **Run_B8:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	–	-4.3	-4.1	2.8	–	-5.0	-1.8	-5.4
Ext_al	3.8	-0.0	-3.5	1.0	-1.6	–	-6.1	-3.4
Flex_ar	2.8	–	-5.0	–	-4.3	-4.1	-5.4	-1.8
Ext_ar	1.0	-1.6	–	3.8	-0.0	-3.5	-3.4	-6.1
E_pl	5.2	-2.0	-0.4	3.7	-0.9	–	–	–
C_pl	–	–	-0.9	5.6	-1.5	-0.5	–	–
L_pl	5.8	-0.7	–	3.7	-0.0	-1.6	–	–
E_pr	3.7	-0.9	–	5.2	-2.0	-0.4	–	–
C_pr	5.6	-1.5	-0.5	–	–	-0.9	–	–
L_pr	3.7	-0.0	-1.6	5.8	-0.7	–	–	–
E_trunk_l	1.3	-0.3	-1.2	2.3	-0.4	-0.4	–	–
C_trunk_l	2.8	-1.8	-0.2	–	–	-2.0	–	–
L_trunk_l	0.5	-2.0	-1.2	–	-0.1	-1.2	–	–
E_trunk_r	2.3	-0.4	-0.4	1.3	-0.3	-1.2	–	–
C_trunk_r	–	–	-2.0	2.8	-1.8	-0.2	–	–
L_trunk_r	–	-0.1	-1.2	0.5	-2.0	-1.2	–	–

	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	8.3	-0.2	-2.1	6.5	–	-3.5	8.3	3.2
Ext_pl	2.0	-3.2	-3.2	0.2	-1.1	-1.1	1.9	1.5
Flex_pr	6.5	–	-3.5	8.3	-0.2	-2.1	3.2	8.3
Ext_pr	0.2	-1.1	-1.1	2.0	-3.2	-3.2	1.5	1.9
E_al	3.6	-0.4	-2.0	0.1	-1.5	–	–	–
C_al	–	-1.0	-1.2	–	-1.2	-0.5	–	–
L_al	5.6	-1.2	-1.6	6.0	-0.6	-2.0	–	–
E_ar	0.1	-1.5	–	3.6	-0.4	-2.0	–	–
C_ar	–	-1.2	-0.5	–	-1.0	-1.2	–	–
L_ar	6.0	-0.6	-2.0	5.6	-1.2	-1.6	–	–
E_tail_l	2.0	-1.2	-1.2	2.5	-1.6	-0.9	–	–
C_tail_l	3.8	-1.1	-1.4	3.9	-1.9	-0.7	–	–
L_tail_l	–	-0.7	-1.5	0.2	-1.3	-0.6	–	–
E_tail_r	2.5	-1.6	-0.9	2.0	-1.2	-1.2	–	–
C_tail_r	3.9	-1.9	-0.7	3.8	-1.1	-1.4	–	–
L_tail_r	0.2	-1.3	-0.6	–	-0.7	-1.5	–	–





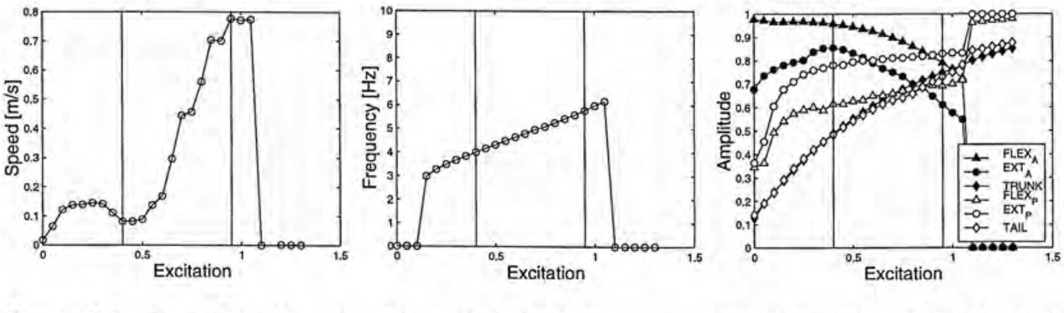
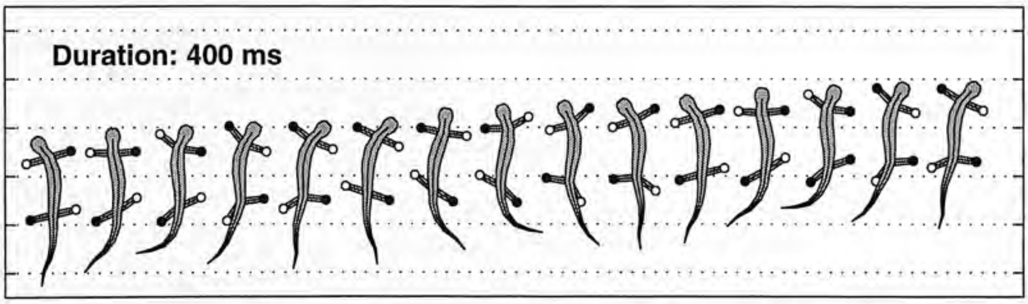
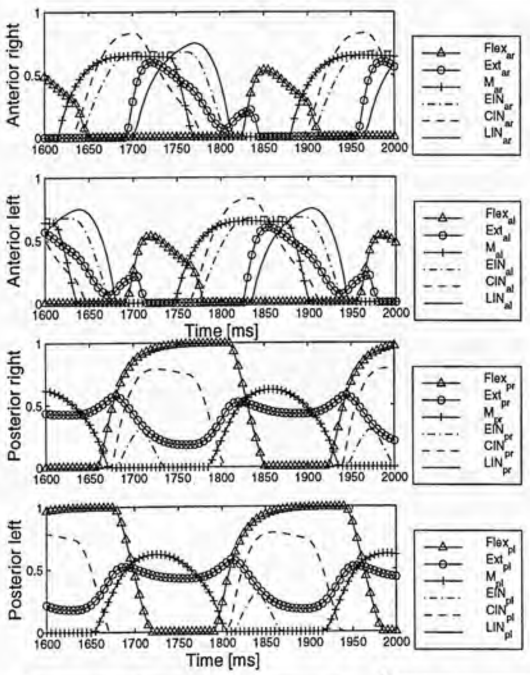


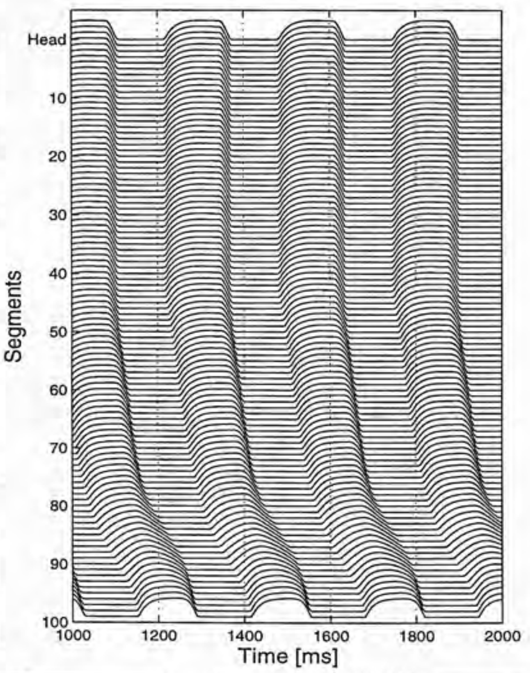
Figure J.9: **Run_B9:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

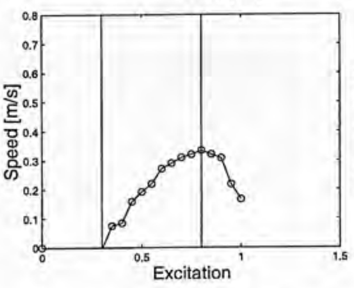
	E_al	C_al	L_al	E_ar	C_ar	L_ar	BS_al	BS_ar
E_al	0.4	–	–	–	-2.0	–	2.0	–
C_al	3.0	–	-1.0	–	-2.0	–	7.0	–
L_al	13.0	–	–	–	-1.0	–	5.0	–
E_ar	–	-2.0	–	0.4	–	–	–	2.0
C_ar	–	-2.0	–	3.0	–	-1.0	–	7.0
L_ar	–	-1.0	–	13.0	–	–	–	5.0
Flex_al	–	-1.3	-4.7	4.9	-1.1	-3.1	-0.8	-2.4
Ext_al	0.8	–	-0.7	2.3	-3.0	-0.6	-2.5	-7.5
Flex_ar	4.9	-1.1	-3.1	–	-1.3	-4.7	-2.4	-0.8
Ext_ar	2.3	-3.0	-0.6	0.8	–	-0.7	-7.5	-2.5
E_pl	3.8	-1.9	-0.5	–	–	-1.8	–	–
C_pl	–	-0.7	-1.0	5.0	-0.4	-0.8	–	–
L_pl	–	-1.2	-0.7	–	-0.6	–	–	–
E_pr	–	–	-1.8	3.8	-1.9	-0.5	–	–
C_pr	5.0	-0.4	-0.8	–	-0.7	-1.0	–	–
L_pr	–	-0.6	–	–	-1.2	-0.7	–	–
E_trunk_l	4.4	-0.0	–	–	-0.2	–	–	–
C_trunk_l	1.2	-1.4	–	–	-0.4	-1.7	–	–
L_trunk_l	4.1	-1.6	-0.7	0.6	-1.5	-1.7	–	–
E_trunk_r	–	-0.2	–	4.4	-0.0	–	–	–
C_trunk_r	–	-0.4	-1.7	1.2	-1.4	–	–	–
L_trunk_r	0.6	-1.5	-1.7	4.1	-1.6	-0.7	–	–

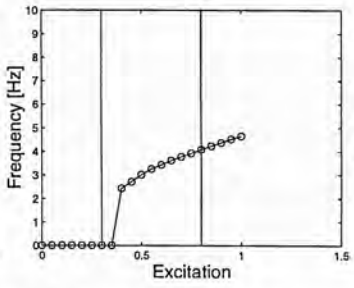
	E_pl	C_pl	L_pl	E_pr	C_pr	L_pr	BS_pl	BS_pr
E_pl	0.4	–	–	–	-2.0	–	2.0	–
C_pl	3.0	–	-1.0	–	-2.0	–	7.0	–
L_pl	13.0	–	–	–	-1.0	–	5.0	–
E_pr	–	-2.0	–	0.4	–	–	–	2.0
C_pr	–	-2.0	–	3.0	–	-1.0	–	7.0
L_pr	–	-1.0	–	13.0	–	–	–	5.0
Flex_pl	0.4	-0.7	-2.8	15.0	–	-1.9	2.8	2.5
Ext_pl	0.6	–	-0.5	10.4	-0.9	-1.2	-0.1	-0.2
Flex_pr	15.0	–	-1.9	0.4	-0.7	-2.8	2.5	2.8
Ext_pr	10.4	-0.9	-1.2	0.6	–	-0.5	-0.2	-0.1
E_al	5.4	-0.3	-0.0	–	-0.5	-1.0	–	–
C_al	6.0	-0.5	–	–	-1.3	-0.0	–	–
L_al	–	-1.4	-0.1	–	-1.7	-1.9	–	–
E_ar	–	-0.5	-1.0	5.4	-0.3	-0.0	–	–
C_ar	–	-1.3	-0.0	6.0	-0.5	–	–	–
L_ar	–	-1.7	-1.9	–	-1.4	-0.1	–	–
E_tail_l	–	-0.2	-1.1	–	-0.9	-1.3	–	–
C_tail_l	–	-1.2	-0.4	1.9	-0.6	-0.8	–	–
L_tail_l	0.6	-1.6	-1.6	0.2	-1.1	-0.1	–	–
E_tail_r	–	-0.9	-1.3	–	-0.2	-1.1	–	–
C_tail_r	1.9	-0.6	-0.8	–	-1.2	-0.4	–	–
L_tail_r	0.2	-1.1	-0.1	0.6	-1.6	-1.6	–	–











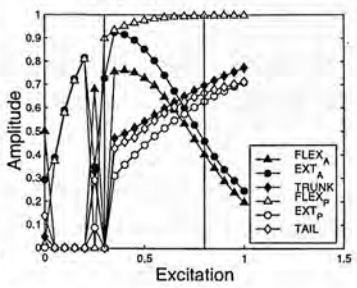


Figure J.10: **Run_B10:** Neural configuration (*top*), gait (*middle-top*), neural activity (*middle-bottom*), and effect of the excitation level (*bottom*).

Bibliography

[Altman & Kien 89] J.S. Altman and J. Kien. New models for motor control. *Neural Computation*, 1:173–183, 1989.

[Arbib & Lee 94] M.A. Arbib and H. B. Lee. Anuran visuomotor coordination for detour behavior: from retina to motor schemas. In *Proceedings, From Animals to Animats III.* MIT Press, 1994.

[Arbib 95] M.A. Arbib, editor. *The handbook of brain theory and neural networks.* MIT Press, 1995.

[Ashley-Ross & Lauder 97] M.A. Ashley-Ross and G.V. Lauder. Motor patterns and kinematics during backward walking in the pacific giant salamander: evidence for novel motor output. *Journal of Neurophysiology*, 78:3047–3060, 1997.

[Ashley-Ross 94a] M.A. Ashley-Ross. Hindlimb kinematics during terrestrial locomotion in a salamander (dicampton tenebrosus). *Journal of Experimental Biology*, 193:255–283, 1994.

[Ashley-Ross 94b] M.A. Ashley-Ross. Metamorphic and speed effects on hindlimb kinematics during terrestrial locomotion in the salamander (dicampton tenebrosus). *Journal of Experimental Biology*, 193:285–305, 1994.

[Bäck 96] T. Bäck. *Evolutionary algorithms in theory and practice.* Oxford University Press, 1996.

[Balakrishnan & Honavar 95] K. Balakrishnan and V. Honavar. Evolutionary design of neural architectures: Preliminary taxonomy and guide to literature. Tr #95-01, Dept. of Computer Science, Iowa State U., 1995.

[Beer & Gallagher 92] R.D. Beer and J.C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122, 1992.

[Beer 90] R.D. Beer. *Intelligence as Adaptive Behavior, an Experiment in Computational Neuroethology.* Academic Press, 1990.

[Beer 95]	R.D. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–510, 1995.
[Belew *et al.* 91]	R.K. Belew, J. McInerney, and N.N. Schraudolph. Evolving networks: using genetic algorithm with connectionist learning. Tr #cs90-174, Computer Science and Engineering Dept., University of California at San Diego, 1991.
[Braitenberg 84]	V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. MIT Press, Cambridge, 1984.
[Brooks 89]	R. A. Brooks. A robot that walk: Emergent behavior form a carefully evolved network. *Neural Computation*, 1(2):253–262, 1989.
[Buchanan & Grillner 87]	J.T. Buchanan and S. Grillner. Newly identified 'glutamate interneurons' and their role in locomotion in the lamprey spinal cord. *Science*, 236:312–314, 1987.
[Buchanan 92]	J.T. Buchanan. Neural network simulations of coupled locomotor oscillators in the lamprey spinal cord. *Biological Cybernetics*, 66:367–374, 1992.
[Bull *et al.* 95]	L. Bull, T.C. Fogarty, and M. Snaith. Evolution in multi-agent systems: Evolving communicating classifier systems for gait in a quadrupedal robot. In L.J. Eshelman, editor, *Proceedings of the sixth international conference on genetic algorithms*, pages 382–388. Morgan Kaufmann, 1995.
[Carrier 93]	D.R. Carrier. Action of the hypaxial muscles during walking and swimming in the salamander dicamptodon ensatus. *Journal of Experimental Biology*, 180:75–63, 1993.
[Cliff 95]	D. Cliff. Computational neuroethology. In M.A. Arbib, editor, *The handbook of brain theory and neural networks*, pages 626–630. MIT Press, 1995.
[Cliff *et al.* 93]	D. Cliff, I. Harvey, and P. Husbands. Explorations in evolutionary robotics. *Adaptive Behavior*, 2(1):73–110, 1993.
[Cobas & Arbib 92]	A. Cobas and M. Arbib. Prey-catching and predator-avoidance in frog and toad: defining the schemas. *Journal of Theoretical Biology*, 157:271–304, 1992.
[Cohen 88]	A.H. Cohen. Evolution of the vertebrate central pattern generator for locomotion. In A. H. Cohen, S. Rossignol, and S. Grillner, editors, *Neural control of*

rhythmic movements in vertebrates. Jon Wiley & Sons, 1988.

[Cruse *et al.* 95] H. Cruse, D.E. Brunn, Ch. Bartling, J. Dean, M. Dreifert, T. Kindermann, and J. Schmitz. Walking: A complex behavior controlled by simple networks. *Adaptive Behavior*, 3(4):385–418, 1995.

[deGaris 90] H. de Garis. Genetic programming: Building artificial nervous systems using genetically programmed neural network modules. In B.W. Porter and R.J. Mooney, editors, *Proceedings of the seventh international conference on machine learning*, pages 132–139. Morgan Kaufmann, 1990.

[Delcomyn 80] F. Delcomyn. Neural basis for rhytmic behaviour in animals. *Science*, 210:492–498, 1980.

[Delvolvé *et al.* 97] I. Delvolvé, T. Bem, and J.-M. Cabelguen. Epaxial and limb muscle activity during swimming and terrestrial stepping in the adult newt, *pleurodeles waltl. Journal of Neurophysiology*, 78:638–650, 1997.

[DeWeerth *et al.* 97] S.P. DeWeerth, G.N. Patel, M.F. Simoni, D.E. Schimmel, and R.L. Calabrese. A VLSI architecture for modeling intersegmental coordination. In R. Brown and A. Ishii, editors, *17th Conference on Advanced Research in VLSI*, pages 182–200. IEEE Computer Society, 1997.

[Eggenberger 97] P. Eggenberger. Evolving morphologies of simulated 3d organisms based on differential gene expression. In P. Husbands and I. Harvey, editors, *Proceedings of the Fourth European Conference on Artificial Life, ECAL97*, pages 205–213. MIT Press, 1997.

[Ekeberg 93] Ö. Ekeberg. A combined neuronal and mechanical model of fish swimming. *Biological Cybernetics*, 69:363–374, 1993.

[Ekeberg *et al.* 91] Ö. Ekeberg, P. Wallén, L. Brodin, and A. Lansner. A computer-based model for realistic simulations of neural networks i: The single neuron and synaptic interaction. *Biological Cybernetics*, 65:81–90, 1991.

[Ekeberg *et al.* 95] Ö. Ekeberg, A. Lansner, and S. Grillner. The neural control of fish swimming studied through numerical simulations. *Adaptive Behavior*, 3(4):363–384, 1995.

[Elman 90] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[Ermentrout & Kopell 91]	B. Ermentrout and N. Kopell. Multiple pulse interactions and averaging in systems of coupled neural oscillators. *J. Math.Biol.*, 29:195–217, 1991.
[Ermentrout & Kopell 94a]	B. Ermentrout and N. Kopell. Inhibition-produced patterning in chains of coupled nonlinear oscillators. *SIAM Journal of Applied Mathematics*, 54(2):478–507, 1994.
[Ermentrout & Kopell 94b]	B. Ermentrout and N. Kopell. Learning of phase lags in coupled neural oscillators. *Neural Computation*, 6:225–241, 1994.
[Eurich *et al.* 95]	W.C. Eurich, G. Roth, H. Schwegler, and W. Wiggers. Simulander: a neural network model for the orientation movement of salamanders. *Journal of comparative physiology*, 176:379–389, 1995.
[Eurich *et al.* 97]	W.C. Eurich, H. Schwegler, and R. Woesler. Coarse coding: applications to the visual system of salamanders. *Biological Cybernetics*, 77:41–47, 1997.
[Ferrar *et al.* 93]	C.H. Ferrar, T.L. Williams, and G. Bowtell. The effect of cell duplication and noise in a pattern generating network. *Neural Computation*, 5:587–596, 1993.
[Floreano 97]	D. Floreano. Evolutionary mobile robotics. In D. Quagliarelli, J. Periaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms in Engineering and Computer Science.* John Wiley and Sons, Ltd., 1997.
[Floreano 98]	D. Floreano. Evolutionary robotics in artificial life and behavior engineering. In T. Gomi, editor, *Evolutionary Robotics.* AAI Books, 1998.
[Frolich & Biewener 92]	L.M. Frolich and A.A. Biewener. Kinematic and electromyographic analysis of the functional role of the body axis during terrestrial and aquatic locomotion in the salamander *ambystoma tigrinum*. *Journal of Experimental Biology*, 62:107–130, 1992.
[Funahashi & Nakamura 93]	K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6:801–806, 1993.
[Getting 88]	P.A. Getting. Comparative analysis of invertebrate central pattern generators. In A. H. Cohen, S. Rossignol, and S. Grillner, editors, *Neural control of rhythmic movements in vertebrates.* Jon Wiley & Sons, 1988.
[Getting 89]	P.A. Getting. Reconstruction of small neural networks. In C. Koch and I. Segev, editors, *Methods in neural modeling*, pages 171–196. MIT Press, 1989.

[Gillis 97] G.B. Gillis. Anguiliform locomotion in an elongate salamander (siren intermedia): effects of speed on axial undulatory movements. *Journal of Experimental Biology*, 200:767–784, 1997.

[Goldberg 89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[Grillner 85] S. Grillner. Neural control of vetebrate locomotion – central mechanisms and reflex interaction with special reference to the cat. In W.J.P. Barnes and Gladden M.H., editors, *Feedback and motor control in invertebrates and vertebrates*, pages 35–56. Croom Helm, 1985.

[Grillner 96] S. Grillner. Neural networks for vertebrate locomotion. *Scientific American*, pages 48–53, January 1996.

[Grillner *et al.* 88] S. Grillner, Buchanan J.T., P. Wallén, and L. Brodin. Neural control of locomotion in lower vertebrates. In A. H. Cohen, S. Rossignol, and S. Grillner, editors, *Neural control of rhythmic movements in vertebrates*, pages 1–40. Jon Wiley & Sons, 1988.

[Grillner *et al.* 91] S. Grillner, P. Wallén, and L. Brodin. Neuronal network generating locomotor behavior in lamprey: Circuitry, transmitters, membrane properties, and simulation. *Annu. Rev. Neurosci.*, 14:169–199, 1991.

[Grillner *et al.* 95] S. Grillner, T. Degliana, Ö. Ekeberg, A. El Marina, A. Lansner, G.N. Orlovsky, and P. Wallén. Neural networks that co-ordinate locomotion and body orientation in lamprey. *Trends in Neuroscience*, 18(6):270–279, 1995.

[Gruau & Quatramaran 97] F. Gruau and K. Quatramaran. Cellular encoding for interactive evolutionary robotics. In P. Husbands and I. Harvey, editors, *Proceedings of the Fourth European Conference on Artificial Life, ECAL97*, pages 368–377. MIT Press, 1997.

[Gruau 95] F. Gruau. Automatic definition of modular neural networks. *Adaptive Behavior*, 3(2):151–184, 1995.

[Haykin 94] S. Haykin, editor. *Neural Networks, a comprehensive foundation.* Prentice Hall International Editions, 1994.

[Hellgren *et al.* 92] J. Hellgren, S. Grillner, and A. Lansner. Computer simulation of the segmental neural network generating locomotion in lamprey by using populations of network interneurons. *Biological Cybernetics*, 68:1–13, 1992.

[Hertz *et al.* 91] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the theory of neural computation.* Santa Fe Institute studies in the sciences of complexity, Lecture notes. Addison-Wesley, 1991.

[Holland 75] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[Hopfield 82] J.J Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, volume 79, pages 2554–2558. Washington : The Academy, 1982.

[Hopfield 84] J.J Hopfield. Neurons with graded response properties have collective computational properties like those of two-state neurons. In *Proceedings of the National Academy of Sciences*, volume 81, pages 3088–3092. Washington : The Academy, 1984.

[Ijspeert & Kodjabachian 98a] A.J. Ijspeert and J. Kodjabachian. Evolution and development of a central pattern generator for the swimming of a lamprey. *Submitted to Artificial Life*, 1998.

[Ijspeert & Kodjabachian 98b] A.J. Ijspeert and J. Kodjabachian. Evolution and development of a central pattern generator for the swimming of a lamprey. *Research Paper No 926, Department of Artifical Intelligence, University of Edinburgh*, 1998.

[Ijspeert 96] A.J. Ijspeert. Modelling the neural controller of a swimming lamprey: a comparison of naturally and artificially evolved networks. No 42, Department of Artificial Intelligence, U. of Edinburgh, 1996.

[Ijspeert *et al.* 97] A.J. Ijspeert, J. Hallam, and D. Willshaw. Artificial lampreys: Comparing naturally and artificially evolved swimming controllers. In P. Husbands and I. Harvey, editors, *Proceedings of the Fourth European Conference on Artificial Life, ECAL97*, pages 256–265. MIT Press, 1997.

[Ijspeert *et al.* 98a] A.J. Ijspeert, J. Hallam, and D. Willshaw. Evolution of a central pattern generator for the swimming and trotting gaits of the salamander. In *Accepted for publication in the proceedings of the Third International Conference on Computational Intelligence & Neurosciences, (ICCIN98)*, 1998.

[Ijspeert *et al.* 98b] A.J. Ijspeert, J. Hallam, and D. Willshaw. Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology. *Accepted for publication in Adaptive Behavior*, 1998.

[Ijspeert *et al.* 98c] A.J. Ijspeert, J. Hallam, and D. Willshaw. From lampreys to salamanders: evolving neural controllers for swimming and walking. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S.W. Wilson, editors, *From Animals to Animats, Proceedings of to the Fifth International Conference of The Society for Adaptive Behavior (SAB98)*, pages 390–399. MIT Press, 1998.

[Jakobi *et al.* 95] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: the use of simulation in evolutionary robotics. In F.Moran, A.Moreno, J.J. Merelo, and P.Chacon, editors, *Proceedings of the third European Conference on Artificial Life, ECAL95*. Springer-Verlag, 1995.

[Jalbert *et al.* 95] J. Jalbert, S. Kashin, and J. Ayers. A biologically-based undulatory lamprey-like auv. In *Proceedings of the Autonomous Vehicles in Mine Countermeasures Symposium. Naval Postgraduate School*, pages 39–52, 1995.

[Jordan 86] M.I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 531–546. Lawrence Erlbaum Associates, Amherst 1986.

[Jung *et al.* 96] R. Jung, T. Kiemel, and A.H. Cohen. Dynamical behavior of a neural network model of locomotor control in the lamprey. *J. of Neurophysiology*, 75:1074–1086, 1996.

[Kitano 90] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476, 1990.

[Kitano 95] H. Kitano. A simple model of neurogenesis and cell differentiation based on evolutionary large-scale chaos. *Artifical Life*, 2:79–99, 1995.

[Kleinfeld & Sompolinsky 89] D. Kleinfeld and H. Sompolinsky. Associative network models for central pattern generators. In C. Koch and I. Segev, editors, *Methods in neural modeling*, pages 195–246. MIT Press, 1989.

[Kodjabachian & Meyer 95] J. Kodjabachian and J.-A. Meyer. Evolution and development of control architectures in animats. *Robotics and Autonomous Systems*, 16:161–182, 1995.

[Kodjabachian & Meyer 98a] J. Kodjabachian and J.-A. Meyer. Evolution and development of modular control architectures for 1-d locomotion in six-legged animats. *to appear in Connection Science*, 1998.

[Kodjabachian & Meyer 98b]	J. Kodjabachian and J.-A. Meyer. Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle-avoidance in artificial insects. *IEEE Trans. on Neural Networks*, 9(5), 1998.
[Kopell 95]	N. Kopell. Chains of coupled oscillators. In M.A. Arbib, editor, *The handbook of brain theory and neural networks*, pages 178–183. MIT Press, 1995.
[Kopell *et al.* 91]	N. Kopell, G.B. Ermentrout, and T.L. Williams. On chains of oscillators forced at one end. *SIAM, Journal of Applied Mathematics*, 51(5):1397–1417, 1991.
[Koza 92]	J. Koza. *Genetic Programming.* MIT Press, 1992.
[Lang & Hinton 88]	K.j. Lang and G.E. Hinton. The development of time-delay neural networks architecture for speech recognition. Cmu-cs-88-153, Carnegie Mellon University, 1988.
[Lewis 96]	M.A. Lewis. *Self-organization of locomotory controllers in robots and animals.* Unpublished PhD thesis, Faculty of the Graduate School, University of Southern California, August 1996.
[Lewis *et al.* 93]	M.A. Lewis, A.H. Fagg, and G.A. Bekey. Genetic algorithms for gait synthesis in a hexapod robot. In Y.F. Zheng, editor, *Recent trends in mobile robots.* World Scientific, 1993.
[Lockery & Sejnowski 93a]	S.R. Lockery and T.J. Sejnowski. The computational leech. *Trends in Neuroscience*, 16(7):283–290, 1993.
[Lockery & Sejnowski 93b]	S.R. Lockery and T.J. Sejnowski. Voyages through weight space: network models of an escape reflex in the leech. In R. Beer, R.E. Ritzman, and T. McKenna, editors, *Biological neural networks in invertebrated neuroethology and robotics.* Academic Press, 1993.
[Matsushima & Grillner 92]	T. Matsushima and S. Grillner. Neural mechanisms of intersegmental coordination in lamprey: local excitability changes modify the phase coupling along the spinal cord. *J. of Neurophysiology*, 67:373–388, 1992.
[Meyer & Guillot 94]	J.-A. Meyer and A. Guillot. From sab90 to sab84: Four years of animat research. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB94).* MIT Press, 1994.
[Michalewicz 96]	Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution programs.* Springer Verlag, 1996.

[Miglino *et al.* 95]	O. Miglino, H.H. Lund, and S. Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4), 1995.
[Miller *et al.* 89]	G.F. Miller, P.M. Todd, and S.U. Hedge. Designing neural networks using genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann, 1989.
[Montana & Davis 89]	D. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann, 1989.
[Nishii *et al.* 94a]	J. Nishii, Y. Uno, and R. Suzuki. Mathematical models for the swimming pattern of a lamprey, i. analysis of collective oscillators with time-delayed interaction and multiple coupling. *Biological Cybernetics*, 72:1–9, 1994.
[Nishii *et al.* 94b]	J. Nishii, Y. Uno, and R. Suzuki. Mathematical models for the swimming pattern of a lamprey, ii. control of the central pattern generator by the brainstem. *Biological Cybernetics*, 72:11–18, 1994.
[Nolfi & Parisi 92]	S. Nolfi and D. Parisi. Growing neural networks. In *Proceedings, Artificial Life III.* Addison-Wesley, 1992.
[O.Michel 96]	O.Michel. An artificial life approach for the synthesis of autonomous agents. In J.-M. Alliot, E. Lutton, E. Ronald, and M. Schoenauer, editors, *Artificial Evolution.* Springer Verlag, 1996.
[Paap *et al.* 96]	K.L. Paap, M. Dehlwisch, and B. Klaassen. Gmd-snake: a semi-autonomous snake-like robot. In *Distributed Autonomous Robotic Systems 2.* Springer-Verlag, 1996.
[Patel *et al.* 98]	G.N. Patel, J.H. Holleman, and S.P DeWeerth. Analog VLSI model of intersegmental coordination with nearest-neighbor coupling. *Advances in Neural Information Processing*, 10, 1998.
[Pearlmutter 95]	B.A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: a survey. *IEEE Transactions on Neural networks*, 6(5):1212–1228, 1995.
[Press *et al.* 94]	W. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical recipes in C : the art of scientific computing, 2nd edition.* Cambridge University Press, 1994.

[Quinn & Espenschied 93] R.D. Quinn and K.S. Espenschied. Control of a hexapod robot using a biologically inspired neural network. In R.D. Beer, R.E. Ritzmann, and T.M. McKenna, editors, *Biological neural networks in invertebrate neuroethology and robotics.* Academic Press, 1993.

[Raibert & Hodgins 93] M.H. Raibert and J.K. Hodgins. Legged robots. In R.D. Beer, R.E. Ritzmann, and T.M. McKenna, editors, *Biological neural networks in invertebrate neuroethology and robotics*, pages 319–354. Academic Press, 1993.

[Rand *et al.* 88] R.H. Rand, A.H. Cohen, and P.J. Holmes. Systems of coupled oscillators as models of central pattern generators. In A. H. Cohen, S. Rossignol, and S. Grillner, editors, *Neural control of rhythmic movements in vertebrates.* Jon Wiley & Sons, 1988.

[Reeve & Hallam 95] R. Reeve and J. Hallam. Control of walking by central pattern generator. In *Proceedings of the 4th International Conference on Intelligent Autonomous Systems, Karlsruhe, Germany*, 1995.

[Reeve 98] R. Reeve. Dynamic walking: a step forward? In S.G. Tzafestas, editor, *Advances in Intelligent Autonomous Agents.* Kluwer Academic Publishers, 1998. In press.

[Saito & Fukuda 96] F. Saito and T. Fukuda. A first result of the Brachiator III: a new brachiation robot modeled on a Siamang. In *Proceedings, Artificial Life V*, pages 354–361. MIT Press, 1996.

[Schaffer *et al.* 92] J. Schaffer, D. Whitley, and L.J. Eskelman. Combinations of genetic algorithms and neural networks: a survey of the state of the art. In J. Schaffer and D. Whitley, editors, *Combinations of genetic algorithms and neural networks.* IEEE Computer Society Press, 1992.

[Schwefel 95] H.-P. Schwefel. *Evolution and Optimum Seeking.* John Wiley, 1995.

[Sigvardt & Williams 96] K.A. Sigvardt and T.L Williams. Effects of local oscillator frequency on intersegmental coordination in the lamprey locomotor cpg: theory and experiment. *J. of Neurophysiology*, 76(6):4094–4103, December 1996.

[Sims 94a] K. Sims. Evolving 3d morphology and behavior by competition. In *Proceedings, Artificial Life IV*, pages 28–39. MIT Press, 1994.

[Sims 94b] K. Sims. Evolving virtual creatures. In *Computer Graphics Proceedings, Annual Conferences Series,* pages 15–24. New York : Association for Computing Machinery, 1994.

[Somers & Kopell 93] D. Somers and N. Kopell. Rapid synchronization through fast threshold modulation. *Biological Cybernetics*, 68:393–407, 1993.

[Spencer 94] G. Spencer. Automatic generation of programs for crawling and walking. In K.E. Kinnear, editor, *Advances in Genetic Programming*, pages 335–353. MIT Press, 1994.

[Taga *et al.* 91] G. Taga, Y. Yamaguchi, and H. Shimizu. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological Cybernetics*, 65:147–159, 1991.

[Terzopoulos *et al.* 94] D. Terzopoulos, X. Tu, and R. Grzeszczuk. Artificial fishes: autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life*, 1(4):327–351, 1994.

[Terzopoulos *et al.* 96] D. Terzopoulos, T. Rabie, and R. Grzeszczuk. Perception and learning in artificial animals. In *Proceedings, Artificial Life V*, pages 354–361. MIT Press, 1996.

[Traven *et al.* 93] H. Traven, L. Brodin, A. Lansner, Ö. Ekeberg, P. Wallén, and S. Grillner. Computer simulations of nmda and non-nmda receptors mediated synaptic drive: sensory and supraspinal modulation of neurons and small networks. *J. of Neurophysiology*, 70(2):695–709, August 1993.

[Usami *et al.* 98] Y. Usami, H. Saburo, S. Inaba, and M. Kitaoka. Reconstruction of extinct animals in the computer. In C. Adami, R. K. Belew, H. Kitano, and C. E. Taylor, editors, *Proceedings, Artificial Life VI*, pages 173–177. MIT Press, 1998.

[Vanier & Bower 98] M.C. Vanier and J.M. Bower. A comparative survey of automated parameter-search methods for compartmental neural models. *to appear in Journal of Computational Neuroscience*, 1998.

[Ventrella 98] J. Ventrella. Attractiveness vs. efficiency. In C. Adami, R. K. Belew, H. Kitano, and C. E. Taylor, editors, *Proceedings, Artificial Life VI*, pages 178–186. MIT Press, 1998.

[Viana Di Prisco *et al.* 90] G. Viana Di Prisco, P. Wallén, and S. Grillner. Synaptic effects of intraspinal stretch receptor neurons mediating movement-related feedback during locomotion. *Brain Research*, 530:161–166, 1990.

[Wadden *et al.* 97]	T. Wadden, J. Hellgren, A. Lansner, and S. Grillner. Intersegmental coordination in the lamprey: simulations using a network model without segmental boundaries. *Biological Cybernetics*, 76:1–9, 1997.
[Wallén *et al.* 92]	P. Wallén, Ö. Ekeberg, A. Lansner, L. Brodin, H. Traven, and S. Grillner. A computer-based model for realistic simulations of neural networks ii: The segmental network generating locomotor rhythmicity in the lamprey. *J. of Neurophysiology*, 68:1939–1950, December 1992.
[Wan 90]	E.A. Wan. Temporal backpropagation for fir neural networks. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 1, pages 575–580, 1990.
[Wan 94]	E.A. Wan. Time series prediction by using a connectionist network with internal delay lines. In A.S. Weigend and N.A. Gerschenfeld, editors, *Time series prediction: forecasting the future and understanding the past*, pages 195–217. Reading, MA: Addison-Wesley, 1994.
[Wellman *et al.* 95]	P. Wellman, V. Krovi, V. Kumar, and W. Harwin. Design of a wheelchair with legs for people with motor disabilities. *IEEE Transactions on Rehabilitation Engineering*, 3:343–353, 1995.
[West & Wilcox 97]	R.M.E. West and G.L. Wilcox. Robust parameter selection for compartmental models of neurons using evolutionary algorithms. In J.M. Bower, editor, *Computational Neuroscience: Trends in Research 1997*. Plenum Publishing, New York, 1997.
[West *et al.* 98]	R.M.E. West, E. De Schutter, and G.L. Wilcox. Using evolutionary algorithms to search for control parameters in a nonlinear partial differential equation. In *Institute for Mathematics and its Applications, Volume on Evolutionary Algorithms and High Performance Computing*. Springer-Verlag, New York, 1998.
[Whitley & Hanson 89]	D. Whitley and T. Hanson. Optimizing neural networks using faster, more accurate genetic search. In *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, 1989.
[Whitley 95]	D. Whitley. Genetic algorithms and neural networks. In G. Winter J. Periaux, editor, *Genetic Algorithms in Engineering and Computer Science*. John Wiley and Sons, 1995.

[Williams & Peng 90]	R.J. Williams and J. Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2:490–501, 1990.
[Williams & Sigvardt 94]	T.L Williams and K.A. Sigvardt. Intersegmental phase lags in the lamprey spinal cord: experimental confirmation of the existence of a boundary region. *Journal of Computational Neuroscience*, 1:61–67, 1994.
[Williams & Sigvardt 95]	T.L. Williams and K.A. Sigvardt. Spinal cord of lamprey: generation of locomotor patterns. In M.A. Arbib, editor, *The handbook of brain theory and neural networks*, pages 918–921. MIT Press, 1995.
[Williams 92a]	T.L Williams. Phase coupling by synaptic spread in chains of coupled neuronal oscillators. *Science*, 258:662–665, 1992.
[Williams 92b]	T.L Williams. Phase coupling in simulated chains of coupled neuronal oscillators representing the lamprey spinal cord. *Neural Computation*, 4:546–558, 1992.
[Williams *et al.* 90]	T.L Williams, K.A. Sigvardt, N. Kopell, G.B. Ermentrout, and M.P. Rempler. Forcing of coupled nonlinear oscillators: studies of intersegmental coordination in the lamprey locomotor central pattern generator. *J. of Neurophysiology*, 64:862–871, September 1990.
[Winfree 80]	A.T. Winfree. *The geometry of biological time*. Springer Verlag, 1980.
[Yamaguchi & Zajac 90]	G.T. Yamaguchi and F.E. Zajac. Restoring unassisted natural gait to paraplegics via functional neuromuscular stimulation: a computer simulation study. *IEEE Transactions in Biomedical Engineering*, 37:886–902, 1990.
[Yao 93]	Xin Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 8(4):539–567, 1993.