Evolution and Development of a Central Pattern Generator for the Swimming of a Lamprey

Auke Jan Ijspeert* and Jérôme Kodjabachian**

Research Paper No 926,*
Department of Artificial Intelligence, University of Edinburgh

*Department of Artificial Intelligence, University of Edinburgh,
5 Forrest Hill, Edinburgh EH1 2QL, Email: aukei@dai.ed.ac.uk
**AnimatLab, OASIS-LIP6, Université Pierre et Marie Curie,
4 place Jussieu, F-75005 Paris, Email: Jerome.Kodjabachian@poleia.lip6.fr

Abstract

This paper describes the design of neural control architectures for locomotion using an evolutionary approach. Inspired by the central pattern generators found in animals, we develop neural controllers which can produce the patterns of oscillations necessary for the swimming of a mechanically simulated lamprey.

This works follows experiments in which swimming controllers were evolved using a simple encoding scheme [1, 2]. Here, controllers are developed using the SGOCE evolutionary algorithm [3, 4] in which a genetic programming approach is used to evolve developmental programs which encode the growing of a dynamical neural network. The developmental programs determine how neurons located on a 2D substrate produce new cells through cellular division and how they form efferent or afferent interconnections. Swimming controllers are generated when the growing networks eventually create connections to the muscles located on both sides of the rectangular substrate. These muscles are part of a 2D mechanical simulation of the body of the lamprey in interaction with water.

Controllers are evolved with a fitness function which rewards the capacity of modulating the speed and the direction of swimming when tonic input signals are varied. The results show that efficient controllers capable of producing the neural activity necessary for swimming can be developed. The evolved controllers are central pattern generators which can produce relatively complex patterns of oscillations when receiving simple tonic inputs. The best solutions generate travelling waves of neural activity, and propagate, similarly to the swimming of a real lamprey, undulations of the body from head to tail projecting the lamprey forward through water. By simply varying the amplitude of two input signals, the speed and the direction of swimming can be modulated.

Keywords: Neural control; genetic programming; developmental encoding; SGOCE; simulation; central pattern generator; swimming; lamprey.

*A shorter version of this paper has been submitted to Artificial Life.

1 Introduction

This paper presents an experiment in which dynamical neural networks are developed for controlling the locomotion of a simulated fish. The experiment takes inspiration from biology at three levels, the type of locomotion, the type of control mechanisms and the type of design method. It can be considered as an experiment in computational neuroethology [5, 6], in the sense that we study how a network of neurons can generate a (simple) behaviour, locomotion, in a (simple) environment, water.

Most animal locomotion is fascinating by its agility and energy efficiency. Using muscles as basic actuators, animals can fly, run, swim, *etc.* with great dexterity. This allows them to move easily in environments in which man-made robots have significant difficulties. As wheeled robots, for instance, are strongly limited in the kind of environments in which they can move, some engineers have turned to biological systems for inspiration and developed robots which move using more animal-like types of gaits [7, 8]. The means of locomotion are then more complex than powered wheels and involve a greater number of actuators generally used in a rhythmic way. Biological types of locomotion require therefore complex control mechanisms as motion is obtained only when the different actuators are well coordinated.

The control mechanisms in animals are provided by networks of neurons. For many animals, both vertebrate and invertebrate, the control of locomotion is distributed and the patterns of oscillations are produced by circuitries called *central pattern generators* (CPGs)[9, 10, 11]. These circuits can produce the different patterns of oscillation necessary for locomotion without oscillating input from the brain or from sensory feedback. In the cat for instance, the walking gait can be induced by sending a simple signal to the brainstem. Different gaits can then be obtained by increasing the amplitude of the signal, with the gait passing from walking to trotting to running [12]. A CPG therefore provides the template of neural activity necessary for locomotion which can be modulated, when necessary, by higher control and by sensory feedback depending on the external conditions.

The bodies and nervous systems of animals are the result of two adaptation phenomena: natural evolution and development. In this work, we will design neural controllers for locomotion using a method which takes inspiration from these phenomena in the form of two abstract ideas: 1) an evolutionary algorithm which creates potential solutions by breeding and mutating other solutions and which selects the best solutions according to a fitness criterion, 2) a representation which encodes solutions through a develomental program containing the rules of how a solution is grown. Evolutionary algorithms are now commonly used for designing neural networks [13, 14], and several developmental encoding schemes have recently be designed (see [15] for a review).

We study in particular the swimming of the lamprey. A lamprey is a fish which projects itself in water by undulation of its body, without the use of fins. The undulation is a traveling wave propagating from head to tail. The neural circuitry for locomotion of the lamprey has been studied extensively by neurobiologists (see [16] for a review), who developed models of the CPG whose capacity to reproduce most of physiological observations has been demonstrated through simulation [17, 18]. In [1, 2], we showed that a genetic algorithm with a simple direct encoding scheme could be used to design a connectionist model of the CPG of the lamprey with several similarities with the biological circuitry. That work was inspired by Ekeberg's connectionist model of the biological

CPG and his mechanical simulation of the lamprey [19]. We showed in particular that a variety of architectures could produce patterns of oscillation for swimming, several of them showing a better performance than the biological configuration (in terms of the ranges of frequencies of oscillation and speeds of swimming that they can produce). We also demonstrated that the genetic algorithm could be used to improve Ekeberg's handcrafted model in order to better fit physiological observations.

In the work presented here, we evolve connectionist CPGs using a genetic programming approach with a developmental encoding, SGOCE (Simple Geometry Oriented Cellular Encoding)[3, 4]. Developmental programs are evolved which determine how neurons located on a 2D substrate produce new cells through cellular division and how they form efferent or afferent interconnections. That method has been successfully used to develop neural controllers for the walking of a virtual six-legged insect and for behaviours, such as gradient following and obstacle avoidance [3, 4]. 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. The evolved controllers could therefore easily be used by higher control centers for sensori-motor coordination experiments of a lamprey-animat. The main differences with our previous experiments on the lamprey are that swimming controllers are evolved in one stage instead of two, and that the fitness function is only based on the behaviour of the mechanical simulation without taking in consideration the neural activity. This study has a double motivation; firstly we would like to gain a better understanding of what are the necessary characteristics for a network of neurons to produce swimming patterns, and secondly we are interested in developing a method for automatically generating neural controllers for a specific task. Such a method could be interesting for the design of control mechanisms for a swimming robot, for instance.

Swimming controllers have been evolved by other researchers. In [20, 21], for instance, the parameters of algorithmic controllers are evolved for the swimming of simulated swimming creatures. Another example is Sims' evolution of swimming controllers composed of "neural nodes" which represent simple mathematical functions such as sum,product, sine, cosine, *etc.* [22]. The research presented here differs from these works in that we develop controllers implemented as dynamical neural networks.

In the next sections we describe the neuronal and mechanical model of the lamprey (Section 2), the developmental encoding scheme (Section 3), the evolutionary algorithm (Section 4), the fitness function (Section 5) and the results of ten evolutions (Section 6). The method and the results are further discussed in Section 7.

2 Neuronal and mechanical model of the lamprey

The CPG of the lamprey has been modeled at several levels of abstraction. Models based on biophysical simulations of neurons have proved that the current knowledge of the CPG circuitry and the type of interneurons involved is sufficient to produce simulations which agree with the physiological observations [17, 18]. At a more abstract level, connectionist models have been made which demonstrate that the connectivity in itself can produce most of the observed patterns of oscillations without the need for complicated neuronal mechanisms [23, 24, 19]. Finally, at the most abstract level, the CPG has been modeled as a chain of abstract oscillators [25, 26].

In this work, we develop connectionist models, i.e. models composed of neurons of intermediate complexity between abstract binary neurons used traditionally in artificial neural networks and detailed compartmental models used in computational neuroscience. Instead of simulating each activity spike of a real neuron, a neuron unit is modeled as a *leaky-integrator* which computes the average firing frequency [27]. According to this model, the mean membrane potential m_i of a neuron N_i is governed 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 [28].

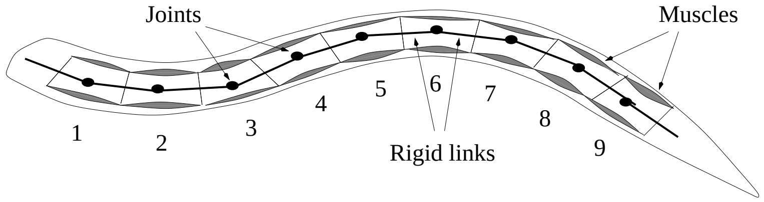


Figure 1: Schema of the mechanical simulation, as developed in [19]. The body is composed of ten rigid links connected trough nine one-degree of freedom joints. The torques of each joint is determined by the contraction of two muscles fixed in parallel.

The network of neurons determine the muscular activity of a simple mechanical simulation of the body of the lamprey in interaction with water (Figure 1). The two-dimensional simulation used in this work is based on that developed by Ekeberg [19]. The body of the lamprey is simulated as 10 rigid links connected by one degree of freedom joints. Two muscles are connected in parallel to each joint and are modeled as a combination of springs and dampers. Muscles can be contracted when they receive an input; their spring constant is then increased proportionally to the input, thus reducing the resting length. The motion of the body is calculated from the accelerations of the links. These accelerations are due to three forces: the torques due to the muscles, the inner forces linked with the mechanical constraints keeping the links together, and the forces due to the water (for more details, see [19, 29]).

3 Developmental encoding scheme

We use the SGOCE (Simple Geometry Oriented Cellular Encoding) evolutionary algorithm for designing the swimming controllers [3, 4]. The algorithm uses an encoding scheme that relates the animat's genotype —a developmental program— and phenotype —a dynamical neural network; and an evolutionary algorithm that breeds and selects the developmental programs within a space defined by some syntactic constraints.

The encoding scheme is a geometry-oriented variation of Gruau's cellular encoding [30]. In the scheme, developmental programs are evolved which determine how neurons divide and get 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 correspond to inputs (the control signals) and outputs (the signals sent to the muscles).

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 1: Set of instructions used in the developmental programs

Each cell within the control architecture is assumed to hold a copy of the developmental program. The developmental program has a tree-like structure like those of genetic programming. The nodes in the tree belong to a small set of developmental instructions (see Table 1), or are structural instructions added to help define the grammar (see below). A local coordinate frame is attached to each precursor cell (Figure 2). 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 the side of the precursor cell it stems from. The synaptic weight of that connection can be modified by the SETINPUTW instructions. An example of how a developmental program is decoded into the corresponding set of neurons is given in Figure 2.

We use a substrate which corresponds to the body of the lamprey (Figure 3). 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

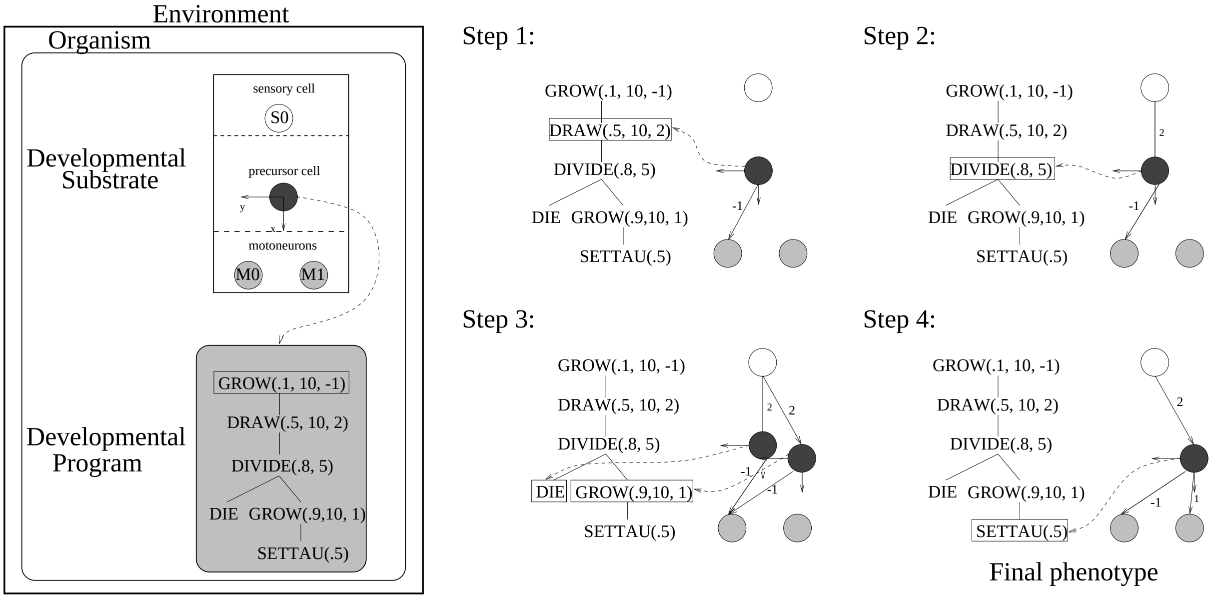


Figure 2: Example of the decoding of a developmental program into a set of neurons. Firstly, a connection is grown by the precursor cell to an output cell (Step 1). Secondly, another connection is drawn by this precursor cell from the input cell (Step 2). In both cases, the cell with which the connection 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.

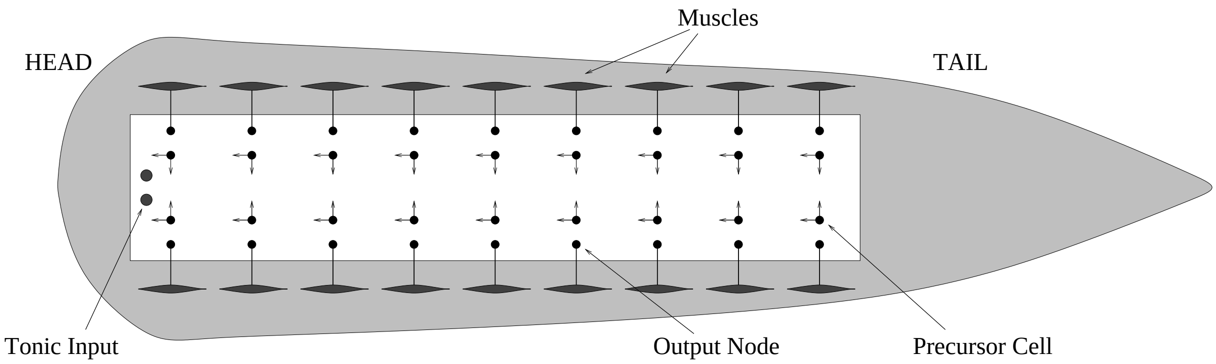


Figure 3: Substrate of the developmental substrate. The rectangular substrate extends over the length of the mechanical body. It contains 18 output nodes located symmetrically on both sides of the substrate which determine the contraction state of the muscles (Figure 1). It contains also two input nodes 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.

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 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.

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 2). The set of terminal symbols consists of the developmental instructions listed in Table 1 and of additional *structural instructions* that have no side-effect on the developmental process. The grammar of Table 2 permits the scheduling of the development to be organised 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 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.

4 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 4.

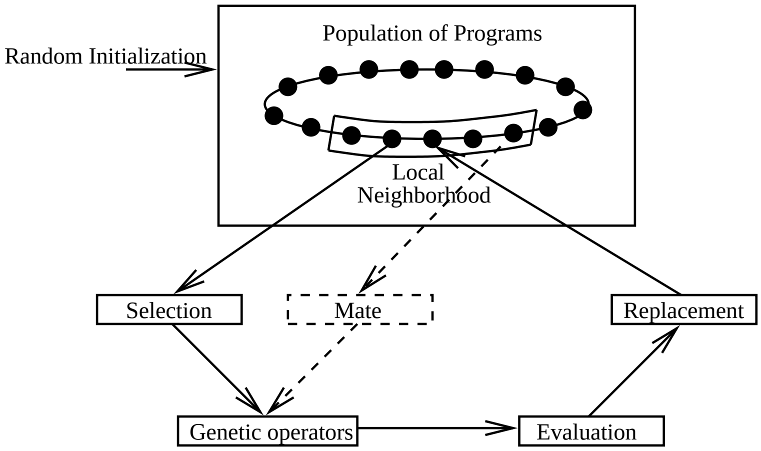


Figure 4: The evolutionary algorithm.

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 [31]).

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* [32]. The number of parameters to be modified is drawn from a binomial distribution $B(n, p)$.

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 $p = 0.5$.

5 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 center 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.[1] If the range of speeds includes a chosen[2] 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.[3] 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:

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

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. The variables for each functions and their corresponding boundaries are given in Table 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.

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

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

[3] 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.

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 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 Results

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 typically start by increasing the contorsion factor, followed by the maximum speed factor. After each increase of the maximal speed, the speed range factor rapidly increases towards one (i.e. most of the speeds from 0.0ms to the maximal speed can be obtained). By definition, the *fit_turning* factor only starts to increase when the speed level for the turning capacities is reached (which is not the case for all evolutions). At the end of the evolutions, most runs have developed solutions which score approximately one (i.e. the maximum) for the contorsion and speed range factors, and the differences in the final fitness values (Table 4) therefore mainly represent the different performances concerning the maximum speed of swimming and the capacity of changing direction. Note that the evolutions were stopped at an arbitrary limit (100 generations) which does not necessarily mean that they had converged to a maximum. Some evolutions were for instance still improving in the five last generations. We stopped the evolutions mainly for time reasons. However, as we will see next, the evolutions were long enough to generate a variety of interesting controllers.

Table 4 gives the performances of the best controllers of each run (controllers 1 to 10 respectively) and Figure 5 shows the swimming behaviour they induce (more details of the controllers can be found in the Appendix). 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).

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. 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.

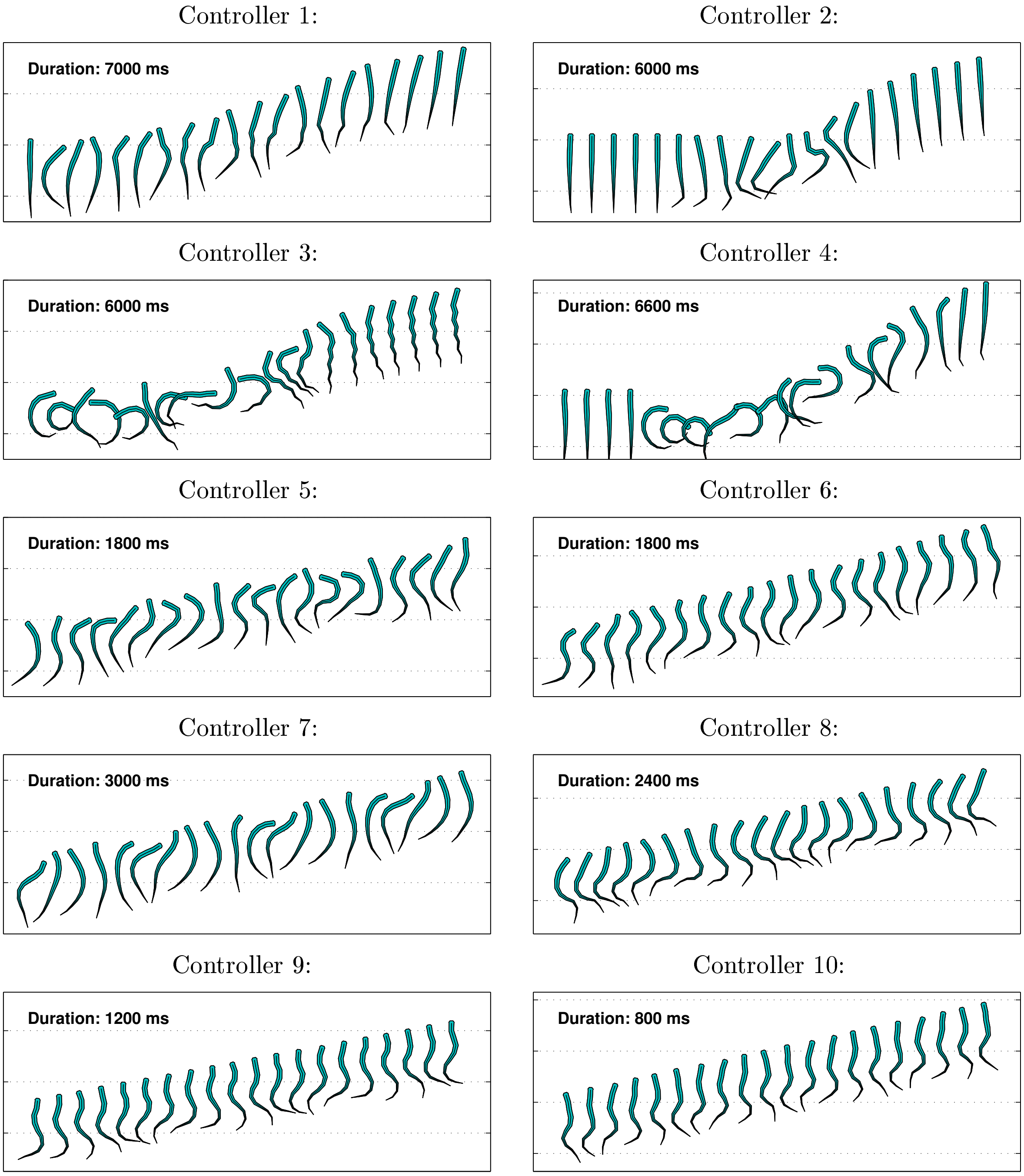


Figure 5: Swimming produced by controllers 1 to 10. The horizontal lines are separated by 200mm.

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 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.

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 (see Appendix, Figures 27 and 31).

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 [28].

Oscillating networks Four evolutions converged to solutions producing permanent and stable oscillations (controllers 7,8,9, and 10). All the segments oscillate at the same frequency, i.e. they are phase locked. The phase between neurons on the opposite side of a segment, and between neurons of neighbouring segments vary from one controller to the other.

Except controller 7, all controllers oscillate with the left and right neurons of each segment perfectly out of phase. In controller 7, the phase relation between opposite neurons depends on the initial conditions. This controller has also the particularity to have all its segments oscillating in phase. 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 5).

In controller 8, the left and right neurons of each segment are perfectly out of phase. The phase relation between segments is however more complex, with a phase lag which is slightly negative between the segments closest to the head and which 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.

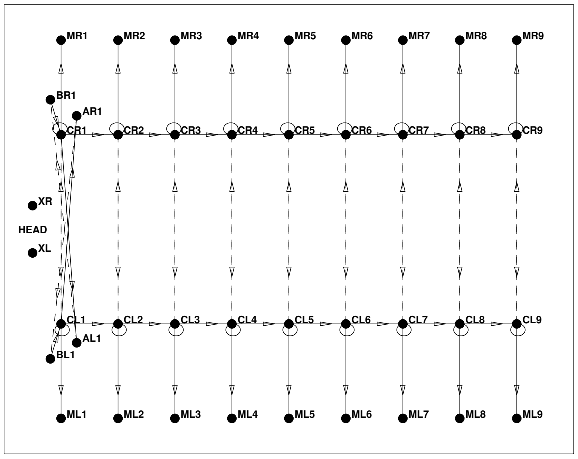
The most interesting solutions are controllers 9 and 10. These two solutions have the left and right neurons of each segment perfectly out of phase, and phase lags between neighbouring segments which lead to a traveling neural activity from head to tail (Figures 7 and 9). The phase lags between segment are almost constant over the spinal cord, although, for controller 10, the signal shapes slightly vary from head to tail. Because of the traveling wave and the constancy of the phase along the spinal cord, these two solutions produce an *anguiliform* swimming which is very similar to that of the real lamprey (Figure 5).

The configurations of these four controllers differ in the number of oscillators they use for producing the oscillatory activity, i.e. the circuits which could potentially oscillate by themselves when isolated from the rest of the network (provided that the simulated neurons receive the adequate tonic input). It can be shown that the smallest oscillator with this model of neurons is composed of 2 neurons with self connections [28].

Controller 7, for instance, has developed separate chains of oscillators on both sides of the body (see Appendix, Figure 33), and it is possible to have only the neurons of one side oscillating. Because there are no inhibitory connections between contralateral sides, the phase relation is not necessary out of phase (as observed earlier), but depends on the initial conditions. For instance, if the simulation is started with symmetrical initial conditions, left and right muscles contract in phase and the lamprey does not progress.

Controllers 8 and 10 (see Appendix Figure 37, and Figure 8) have oscillators which are distributed over both sides with inhibitory contralateral connections. They have one oscillator per segment which are interconnected over the spinal cord. Because of the limitation of the length of a connection (parameter r of the GROW and DRAW instructions), only closest neighbour coupling has been developed between the oscillators.

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). 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



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: 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.

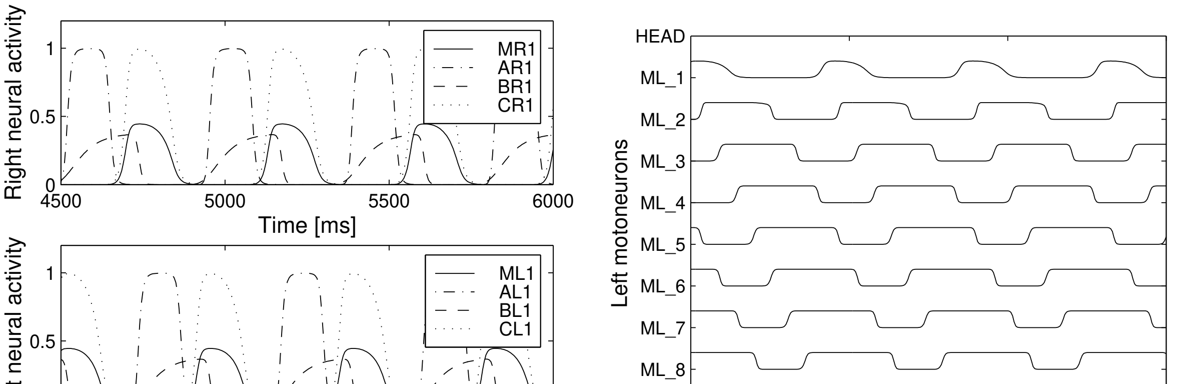
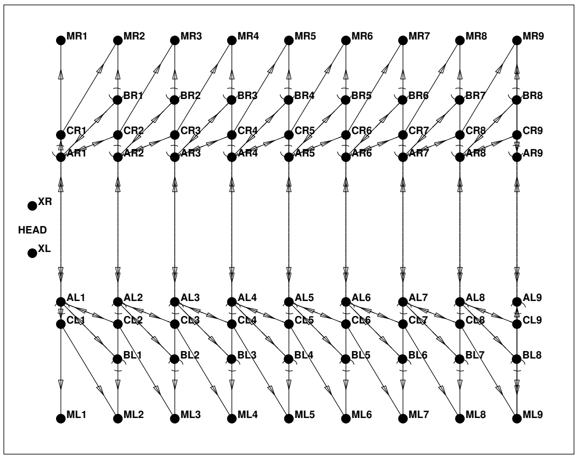


Figure 7: Neural activity in controller 9. *Left:* Neural activity of left and right neurons and the signals sent to the muscles in segment 1 (the oscillator segment, see text). *Right:* Signals sent to the left muscles along the nine segments.



Parameters:

Tau:		
A:		1600.0
B:		29.2
C:		16.5
Bias:		
A:		−16.00
B:		−16.00
C:		2.00

Weights:

Segment 1		**Segment 5**		**Segment 9**	
AL1 → ML1	27.00	CL4 → ML5	31.80	CL8 → ML9	31.80
XL → AL1	28.45	AL5 → ML5	27.00	AL9 → ML9	27.00
BL1 → AL1	−6.10	XL → AL5	28.45	CL9 → ML9	31.80
CL2 → AL1	−24.75	BL5 → AL5	−6.10	XL → AL9	28.45
AL1 → AL1	−23.95	CL6 → AL5	−24.75	CL9 → AL9	−6.10
XL → BL1	1.00	AL5 → AL5	−23.95	CL9 → AL9	−24.75
BL1 → BL1	−16.20	XL → BL5	1.00	AL9 → AL9	−23.95
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 8: 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.

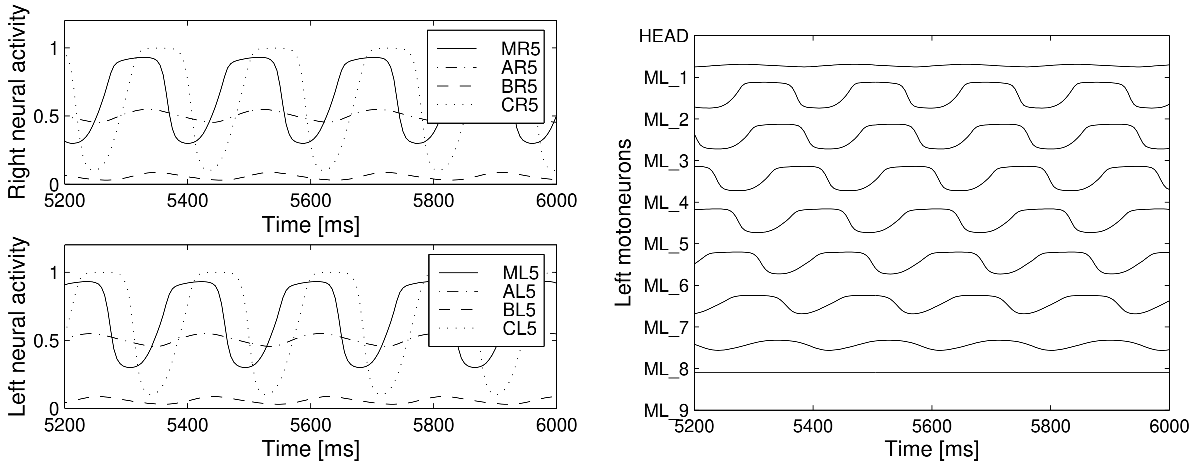


Figure 9: Neural activity in controller 10. *Left:* Neural activity of left and right neurons and the signals sent to the muscles in segment 5. *Right:* Signals sent to the left muscles along the nine segments.

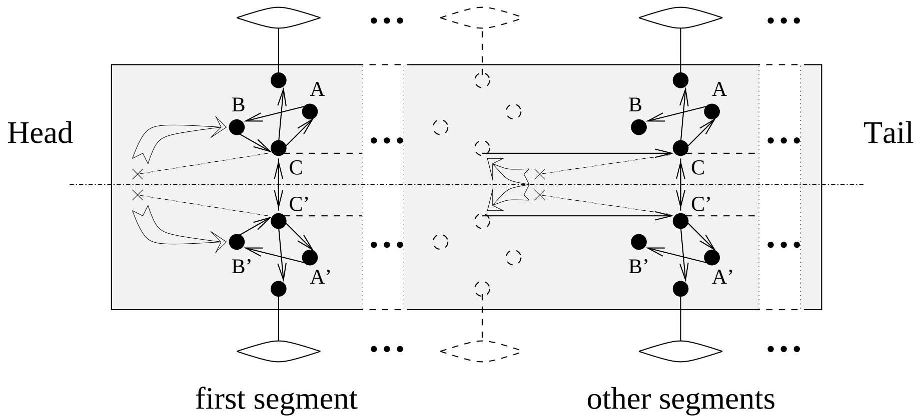


Figure 10: Side-effect during the development of Controller 9: During the development, a DRAW instruction was read by all cells C and looked for the closest cell to a target point (represented as a cross on the figure) whose position in the local frame of the corresponding cell C was given by the DRAW 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.

one (Figure 10). 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.

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.

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 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.

The change of speed is obtained through variations of the frequency of oscillation and the phase lag between segments (see Appendix, Figures 35, 39, 43, and 47), although in controllers 7, 8 and 10, there also some variations of the amplitude of the signals sent to the muscles. In controller 7, increasing the excitation level has the effect of increasing the frequency while the phase lag between segments remains at zero. In controller 8, the frequency is almost constant for any excitation level and it is the phase lag between segments which increases with the excitation level. In controller 9, increasing the excitation level leads to an increase of both the frequency and the phase lag between segments. Finally controller 10 has a frequency wich increases with the excitation level while the lag between segment slightly decreases. Of the controllers producing stable oscillations those which produce anguiliform swimming like the lamprey (controllers 9 and 10) reach the highest speeds.

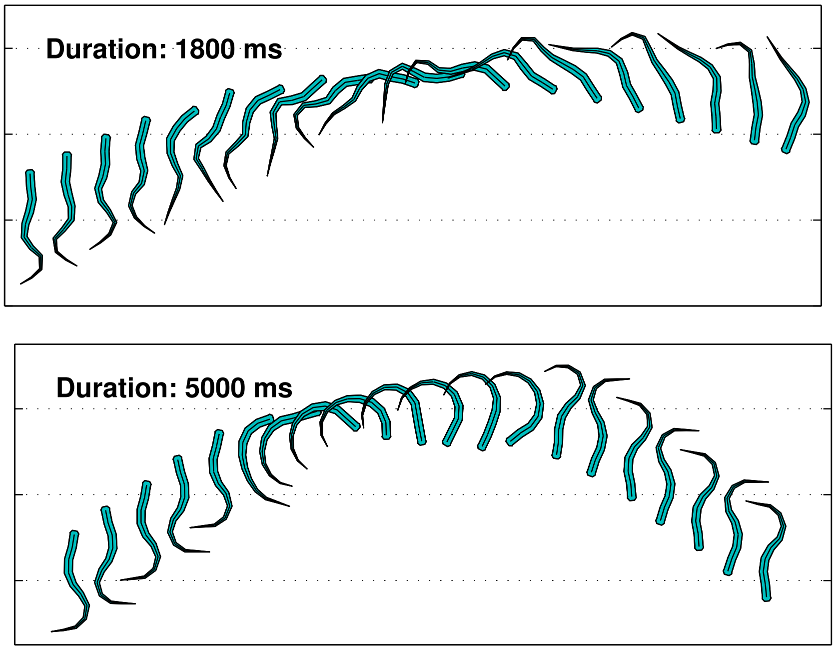


Figure 11: *Top:* slow turning induced by controller 10, *bottom:* Sharp turning induced by controller 9.

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 is small (for instance $\pm 10\%$ of the excitation level), the lamprey goes on propagating a traveling undulation and therefore swims in a circle until the asymmetry is stopped (Figure 11, top). If the asymmetry is large (for instance $\pm 30\%$ 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 11, bottom).

7 Discussion

7.1 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 evolutions were carried out with relatively small populations (50 individuals) compared to most genetic programming experiments. The reduced size of the population is one of the reasons why the results of the evolution differ significantly depending on the initial random populations. Some evolutions converged to local maxima with very low fitness value, and had difficulties to get out of these because of the reduced genetic diversity of the small population. For finding optimal solutions, further experiments should be run with larger populations to prevent those premature convergences.

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, and we therefore canalized the search in several ways to make the task 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. While such symmetry hypotheses are commonly used when evolving locomotion controllers with fixed topologies [33, 34, 35], SGOCE allows the evolution of *variable* symmetric topologies.

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. Moreover, the grammar indirectly 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.

Evolutionary algorithms for neural networks From a general point of view, using evolutionary algorithms is an interesting technique for designing neural networks compared to traditional learning algorithms (such as variations of the backpropagation algorithm for dynamical networks, see [36] for instance) because of their flexibility. The fitness function of an evolutionary algorithm does, for instance, not need to be differentiable or even continuous like error functions minimized by most learning algorithms. There is also no need to provide a specific oscillation (limit cycle) that the network should learn. This has allowed us to characterize the desired behaviour of the controller at a higher level, by determining desired characteristics of the swimming control in the mechanical simulation. These features make evolutionary algorithms a popular design technique in the field of neural networks applied to control animats [37, 38].

Comparison with previous evolutions of swimming controllers for the lamprey In [1], we used a staged evolution approach to generate swimming controllers, with a first stage in which we evolved segmental oscillators and a second stage in which we evolved the coupling between the oscillators. We used the same neuron models as Ekeberg [19] which have three state variables instead of one. For the first stage, the fitness function was based on neural aspects, such as the range of frequencies of oscillation, and in the second stage, the fitness function was based on both neural aspects and mechanical aspects, such as the speed of swimming. We used a direct encoding of the synaptic weights in the first stage, and of the extent of the interconnections in the second stage.

In contrast, 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 [1], we evolved the number of neurons and connections indirectly by fixing a maximum number of neurons and evolving the weights of all possible connections 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.

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. Although, the staged approach of the previous work allowed swimming controllers to be generated in less computational time, 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.

Comparison with previous applications of SGOCE In [3] 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 same parameter values were used by the evolutionary algorithm for solving both tasks, the only difference to [3] being that in the work reported here the population size was reduced 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 [3] lay in the evaluation strategies and in the inclusion of command cells. While the six-legged animat experiments used a global fitness criterion involving measures of body speed and leg movements, the current work uses a more elaborated 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 [3] or to turn [4] when connected to a second, higher level neural network. More experiments are needed to tell whether the use of command inputs can help solve tasks that involve controlling the speed and direction of locomotion. In [39], where SGOCE is applied to the control of a real hexapod robot, an incremental strategy involving such command inputs is proposed.

7.2 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 turn, swimming straight 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 closer to the biological connectivity found in the lamprey [40, 10, 41, 16]. The spinal cord of the lamprey is indeed considered to be an interconnection of segmental oscillators, as small parts of the spinal cord up to 2 segments can be made to oscillate in isolation (the whole spinal cord is made of approximately 100 segments). 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. Controller 10 therefore shares the most properties with the biological CPG of the lamprey.

By contrast with our previous work in which we 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.

Further directions The evolved controllers work in an open loop manner, in the sense that the mechanical simulation does not send feedback to the controller. In the lamprey, sensory feedback is provided by stretch sensitive cells, the edge cells, which are located on both sides of the spinal cord [42]. These cells are useful for coordinating the neural activity with the actual body state when swimming in unstationary water. Ekeberg has shown in particular that these cells are necessary for crossing a local speed barrier [43]. Similarly to [1], where we evolved such a feedback, further experiments using the developmental encoding should include these cells in order to close the control loop.

As the speed and the direction of swimming can be modulated through two input signals, the movements of the simulated lamprey (in a plane) can be determined. The evolved controllers can therefore be used by higher neural centers for the generation of simple behaviours, with the two input nodes of the CPG providing the interface between the CPG and the higher control centers. It would therefore be interesting to extend the simulation by adding a simple simulated visual system to it and to evolve higher neural centers for simple sensory-motor coordination experiments such as avoiding obstacles or approaching and attaching to a host (the lamprey is a parasite).

Another interesting extension would be to evolve CPGs for controlling the swimming of a real robot. This would require to develop controllers for a 3D simulation (see [43] for instance) which, depending on the degree of freedom of the spinal joints, implies more muscles for controling the pitch and roll movements and some feedback from vestibular systems to achieve roll stability.

8 Conclusion

This paper has 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. Swimming controllers are developed given the fixed body structure of the lamprey and given a high level characterisation of the desired behaviour in terms of the control of the speed and the direction of swimming.

Similarly to central pattern generators found in animals, neural circuitries are evolved which can produce relatively complex patterns of oscillations when receiving simple tonic input. By simply varying the amplitude of two input signals, the speed and the direction of swimming can be varied. The two best controllers produce an anguiliform swimming very similar to that of the lamprey, with a traveling wave of neural activity propagating from head to tail.

Acknowledgments: We would like to thank John Hallam and Jean-Arcady Meyer for their comments on earlier versions of this paper. The facilities were provided by the University of Edinburgh and by the AnimatLab. Auke Jan Ijspeert is supported by grants from the Swiss National Research Fund and from the European Commission (Marie Curie Fellowship).

Appendix:

Controller 1

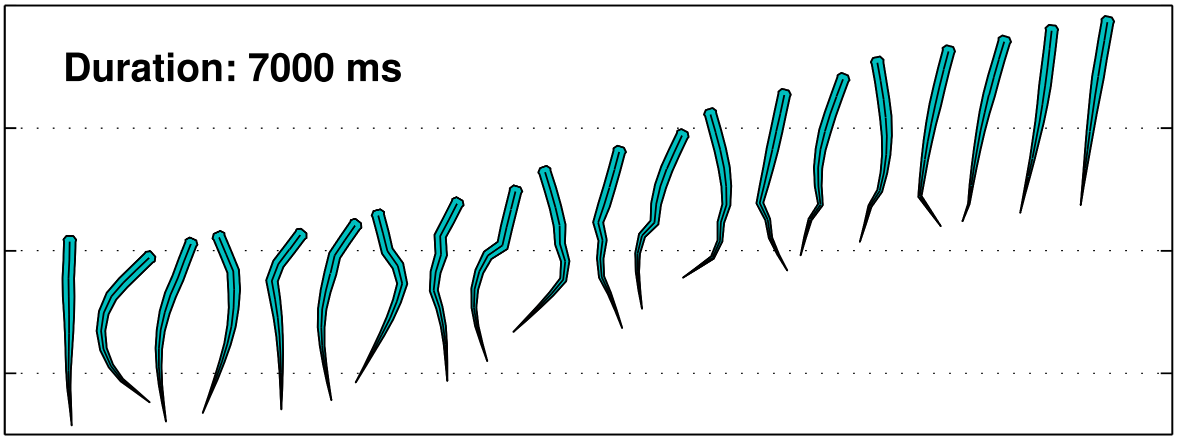
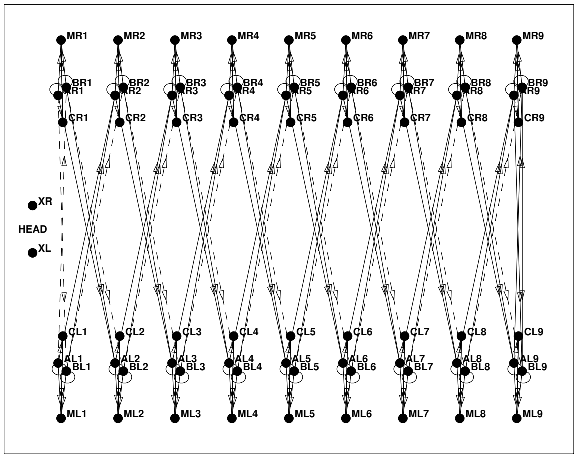


Figure 12: Swimming.



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 13: Neural configuration.

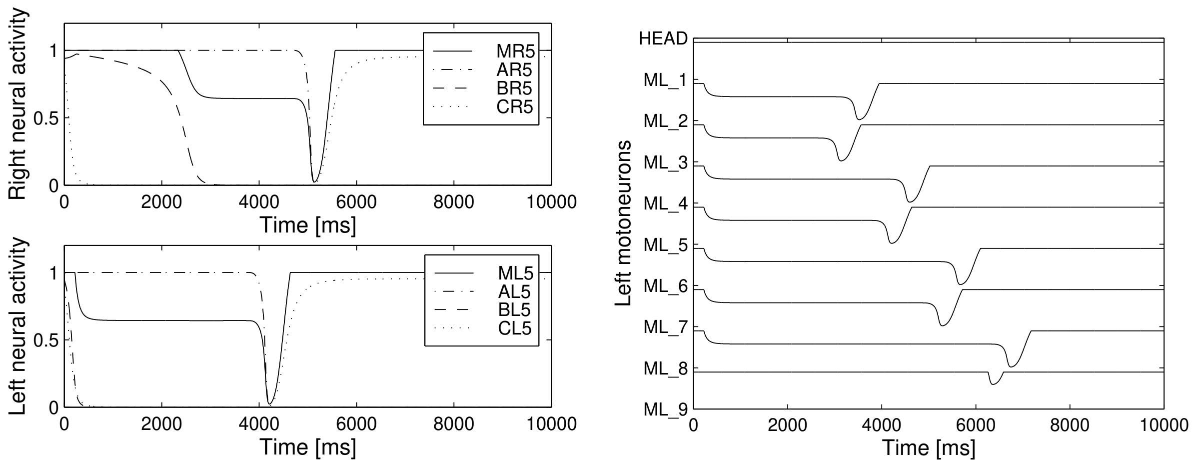


Figure 14: Neural activity.

Controller 2

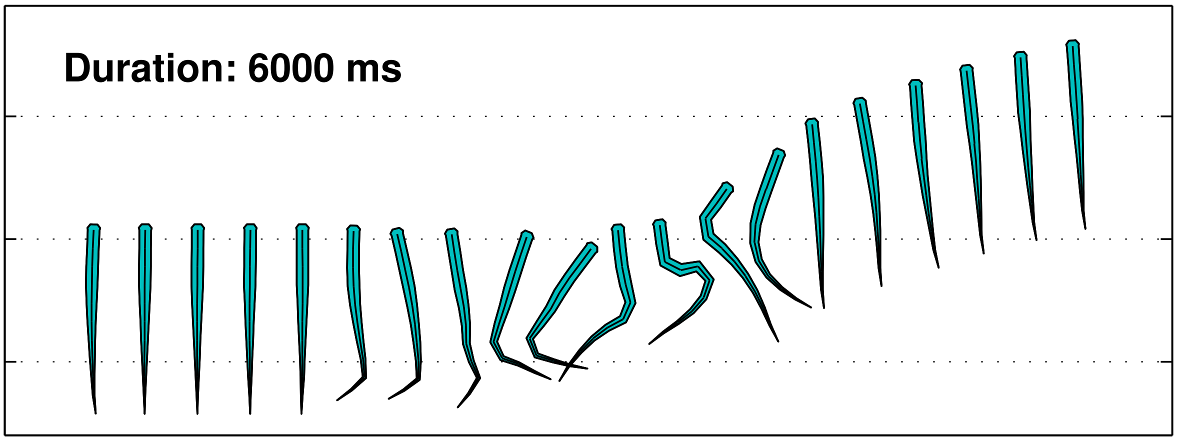
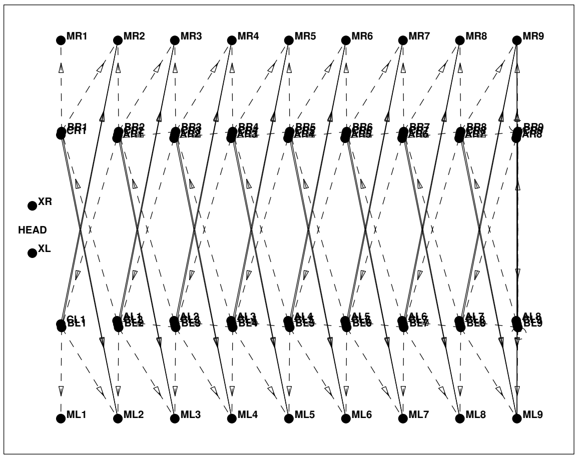


Figure 15: Swimming.



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 16: Neural configuration.

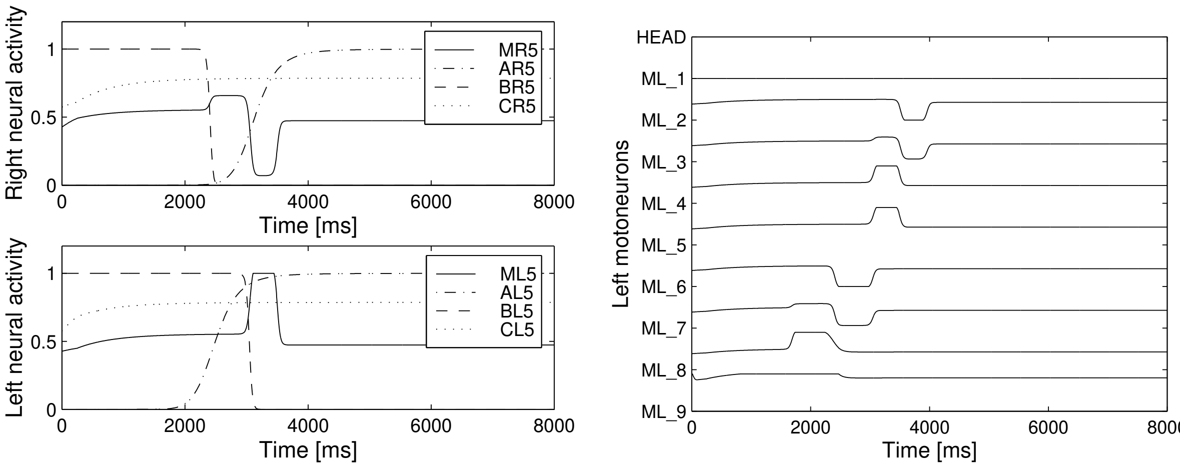


Figure 17: Neural activity.

Controller 3

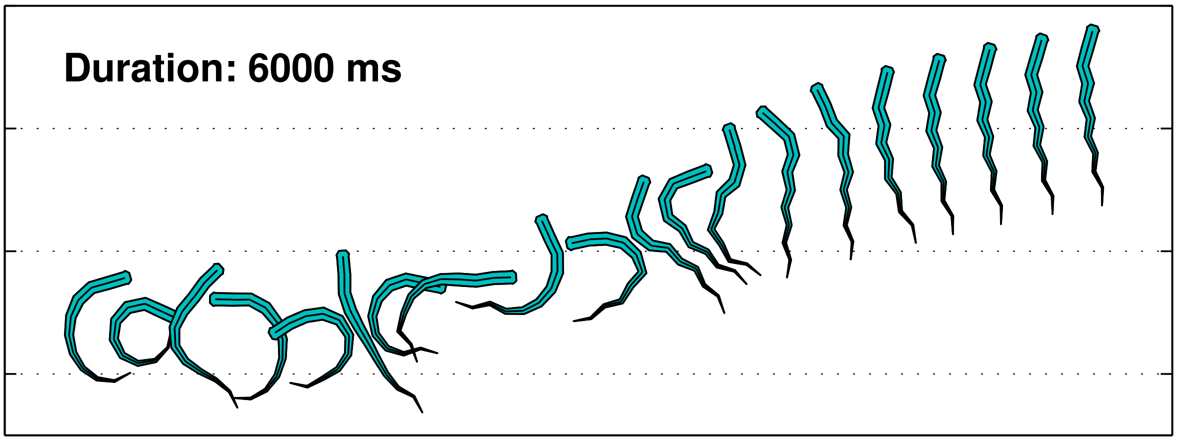
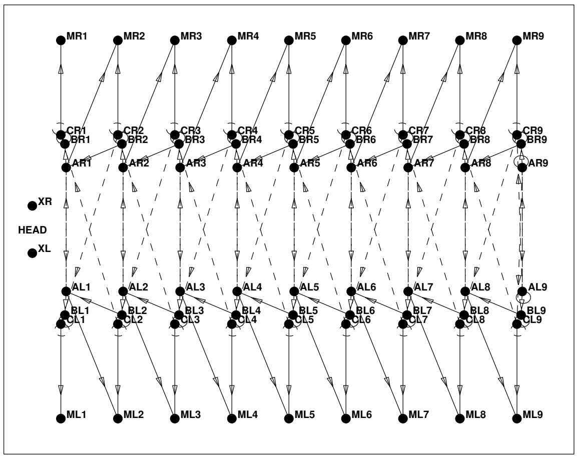


Figure 18: Swimming.



Parameters:

Tau:	
A:	50.0
B:	348.2
C:	50.0

Bias:	
A:	16.05
B:	16.05
C:	11.90

Weights:

Segment 1		Segment 5		Segment 9	
CL1 → ML1	11.05	AL4 → ML5	32.05	AL8 → ML9	32.05
XL → AL1	–31.40	CL5 → ML5	11.05	CL9 → ML9	11.05
AR1 → AL1	–22.75	XL → AL5	–31.40	XL → AL9	–31.40
BL2 → AL1	25.70	AR5 → AL5	–22.75	AR9 → AL9	–22.75
XL → BL1	–22.75	BL6 → AL5	25.70	AL9 → AL9	32.05
AL1 → BL1	–26.50	XL → BL5	–22.75	XL → BL9	–22.75
BL1 → BL1	32.05	AL5 → BL5	–26.50	AL9 → BL9	–26.50
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 19: Neural configuration.

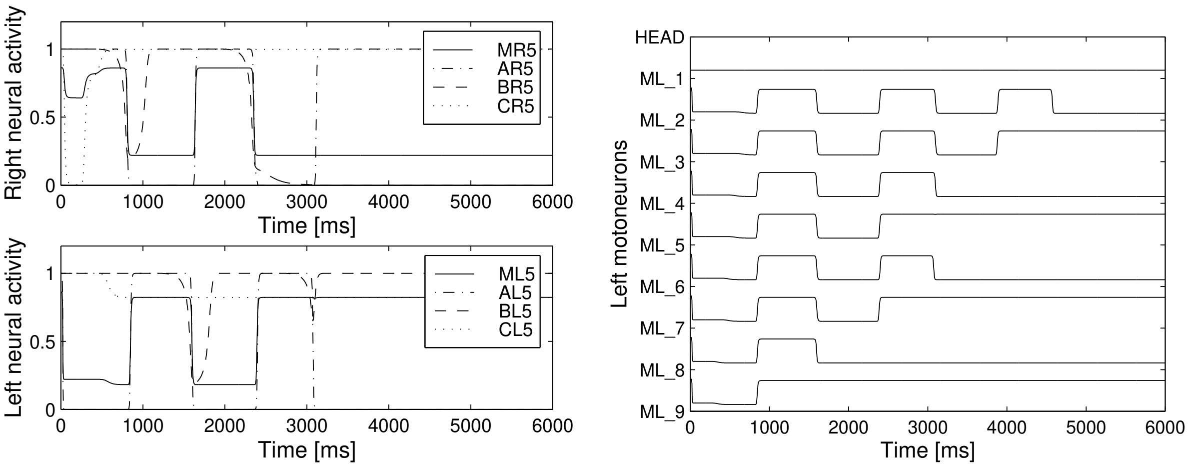


Figure 20: Neural activity.

Controller 4

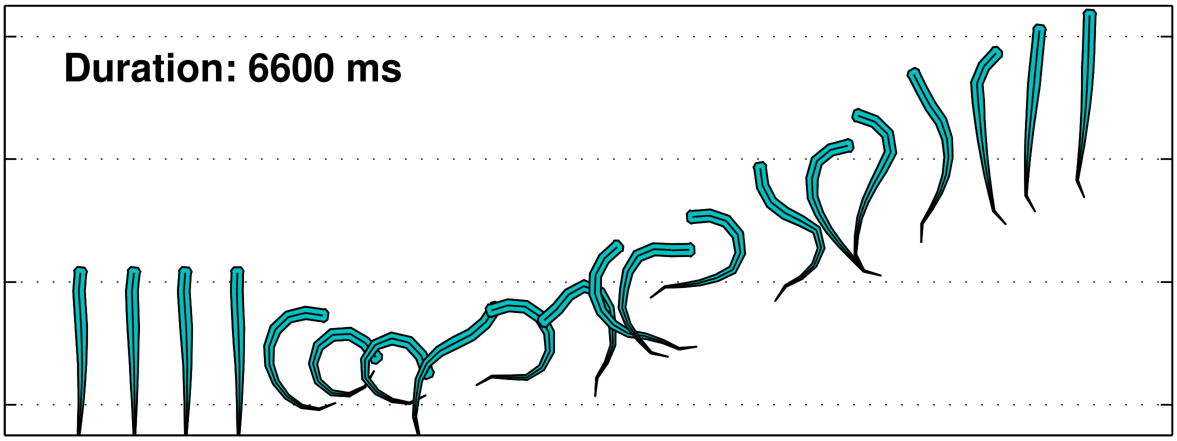
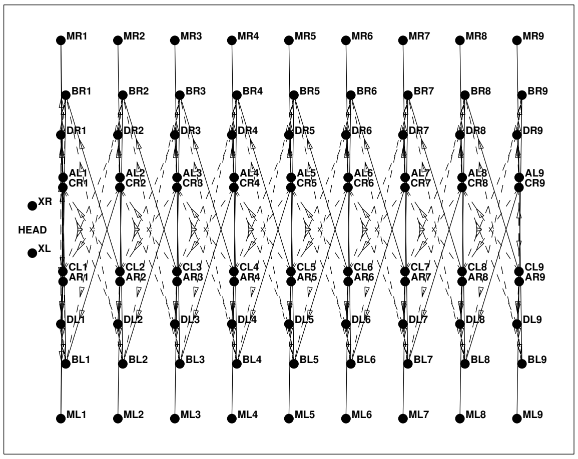


Figure 21: Swimming.



Parameters:

	Tau:	Bias:
A:	421.3	−16.00
B:	50.0	2.00
C:	50.0	2.00
D:	746.4	−16.00

Weights:

Segment 1		Segment 5		Segment 9	
CR1 $\rightarrow$ ML1	10.75	BR5 $\rightarrow$ ML5	30.90	BR9 $\rightarrow$ ML9	30.90
BR1 $\rightarrow$ ML1	30.90	XL $\rightarrow$ AL5	22.20	XL $\rightarrow$ AL9	22.20
XL $\rightarrow$ AL1	22.20	XL $\rightarrow$ BL5	1.00	XL $\rightarrow$ BL9	1.00
XL $\rightarrow$ BL1	1.00	DL5 $\rightarrow$ BL5	−16.05	DL9 $\rightarrow$ BL9	−16.05
DL1 $\rightarrow$ BL1	−16.05	DL5 $\rightarrow$ BL5	−18.55	DL9 $\rightarrow$ BL9	−18.55
DL1 $\rightarrow$ BL1	−18.55	DL5 $\rightarrow$ BL5	0.00	DL9 $\rightarrow$ BL9	0.00
DL1 $\rightarrow$ BL1	0.00	CR6 $\rightarrow$ BL5	10.75	XL $\rightarrow$ CL9	1.00
CR2 $\rightarrow$ BL1	10.75	BR6 $\rightarrow$ BL5	−23.95	CR9 $\rightarrow$ CL9	−26.80
BR2 $\rightarrow$ BL1	−23.95	XL $\rightarrow$ CL5	1.00	XL $\rightarrow$ DL9	1.00
XL $\rightarrow$ CL1	1.00	CR6 $\rightarrow$ CL5	−26.80	AR9 $\rightarrow$ DL9	22.20
CR2 $\rightarrow$ CL1	−26.80	DR6 $\rightarrow$ CL5	−18.00	CL9 $\rightarrow$ DL9	−32.00
DR2 $\rightarrow$ CL1	−18.00	XL $\rightarrow$ DL5	1.00		
XL $\rightarrow$ DL1	1.00	AR5 $\rightarrow$ DL5	22.20		
AR1 $\rightarrow$ DL1	22.20	CL5 $\rightarrow$ DL5	−32.00		
BR1 $\rightarrow$ DL1	−23.95				
DR1 $\rightarrow$ DL1	−18.00				
CL1 $\rightarrow$ DL1	−32.00				

Figure 22: Neural configuration.

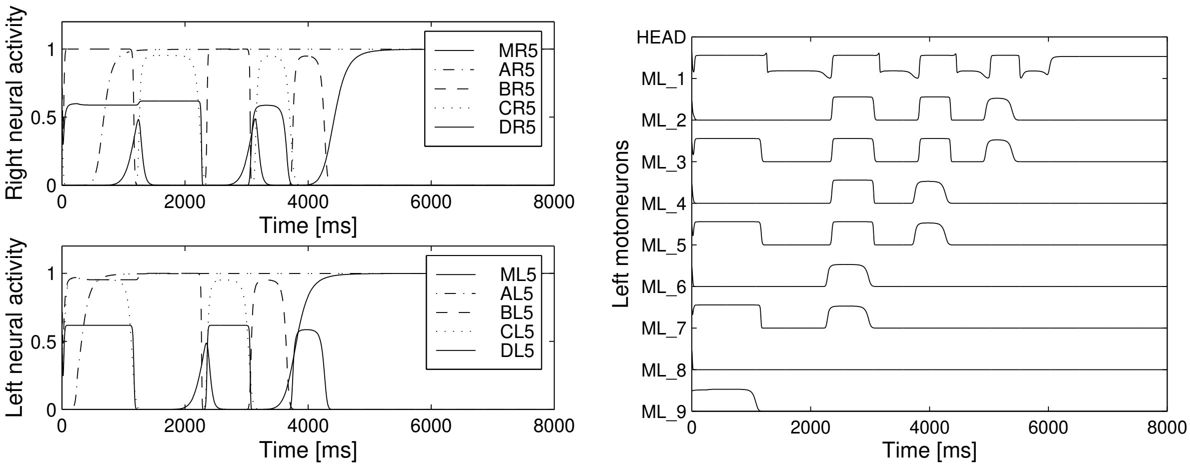


Figure 23: Neural activity.

Controller 5

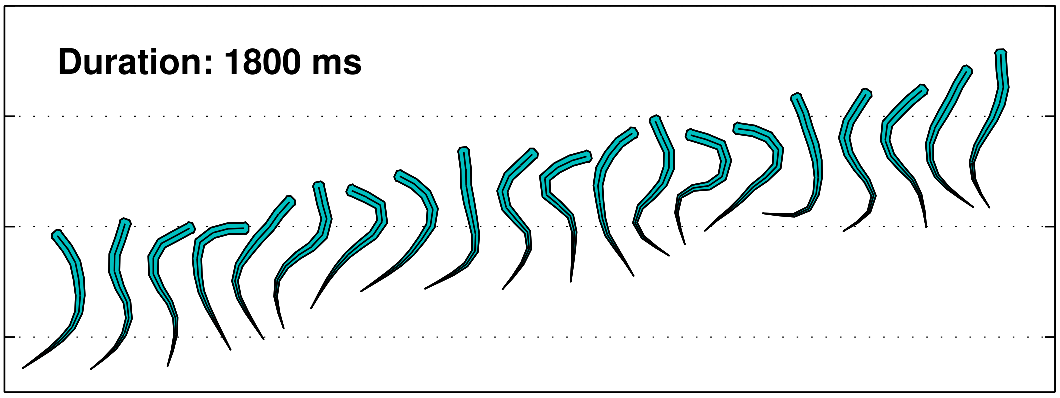
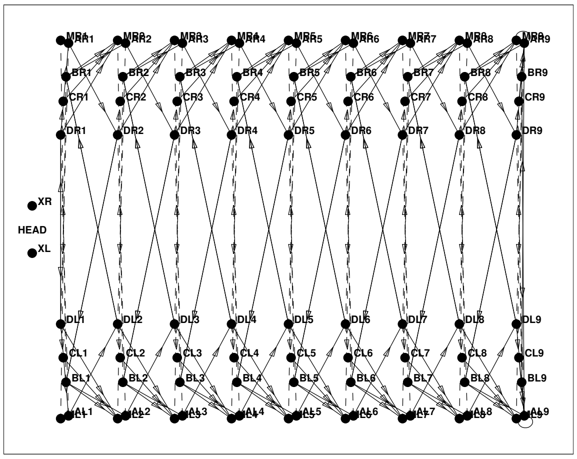


Figure 24: Swimming.



Parameters:		Weights:					
Tau:		**Segment 1**		**Segment 5**		**Segment 9**	
A:	91.7	BR1 → ML1	–26.25	CL4 → ML5	31.30	CL8 → ML9	31.30
B:	122.0	XL → AL1	–17.90	BL4 → ML5	21.55	BL8 → ML9	21.55
C:	91.7	DL1 → AL1	14.80	DL4 → ML5	0.05	DL8 → ML9	0.05
D:	1600.0	XL → BL1	–18.75	BR5 → ML5	–26.25	BR9 → ML9	–26.25
		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 25: Neural configuration.

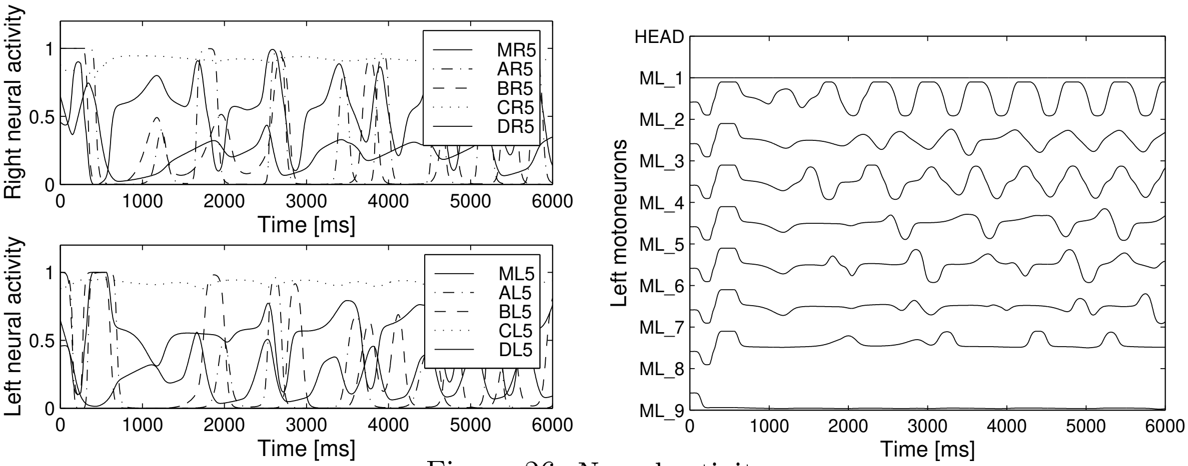


Figure 26: Neural activity.

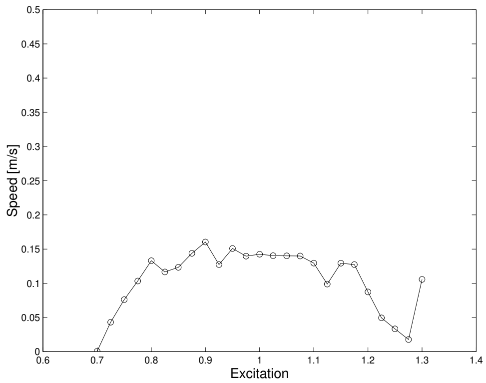


Figure 27: Influence of the tonic input on the speed of swimming.

Controller 6

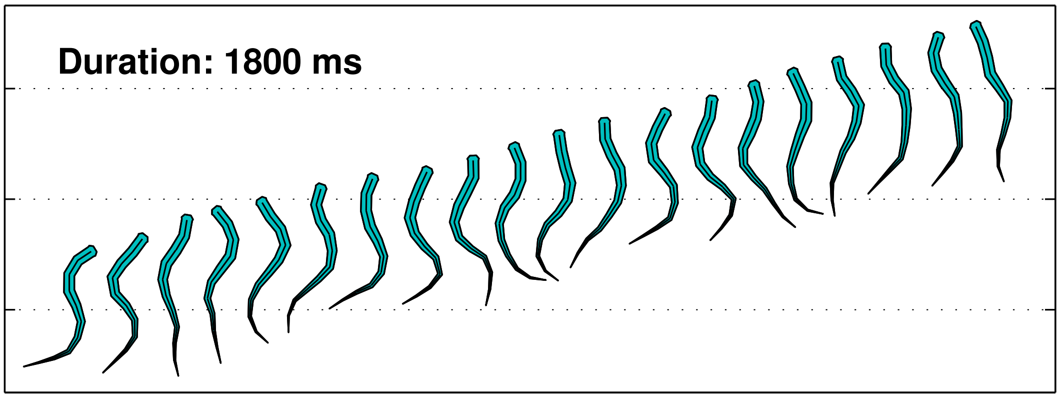
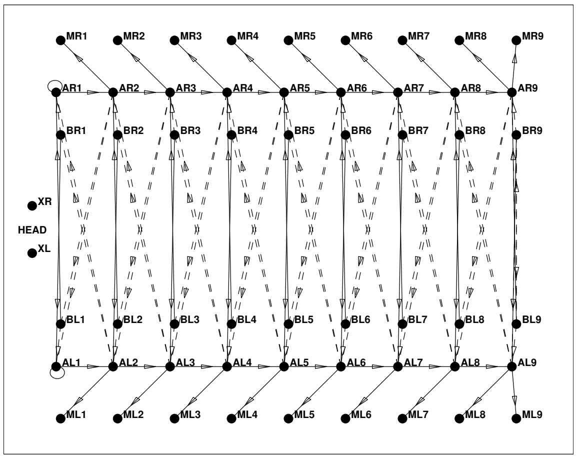


Figure 28: Swimming.



Parameters:

Tau:	
A:	178.7
B:	50.0
Bias:	
A:	9.40
B:	4.20

Weights:

Segment 1		Segment 5		Segment 9	
AL2 → ML1	25.55	AL6 → ML5	25.55	AL9 → ML9	25.40
XL → AL1	−19.85	XL → AL5	−19.85	XL → AL9	−19.85
BR1 → AL1	29.05	AL4 → AL5	25.40	AL8 → AL9	25.40
AR2 → AL1	−27.95	BR5 → AL5	29.05	BR9 → AL9	29.05
AL1 → AL1	25.55	AR6 → AL5	−27.95	BR9 → AL9	−27.95
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 29: Neural configuration.

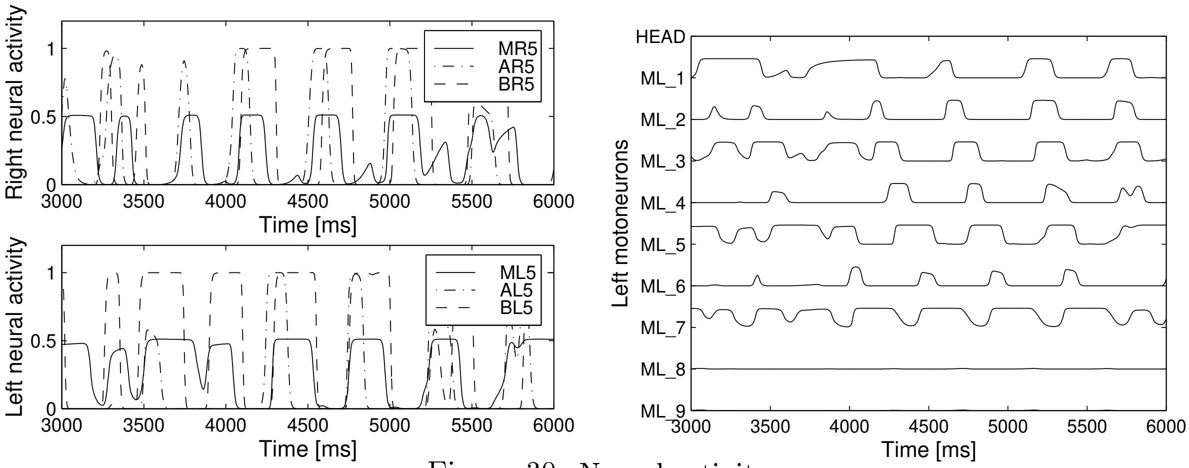


Figure 30: Neural activity.

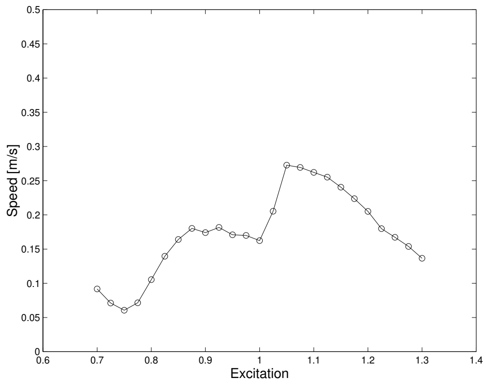


Figure 31: Influence of the tonic input on the speed of swimming.

Controller 7

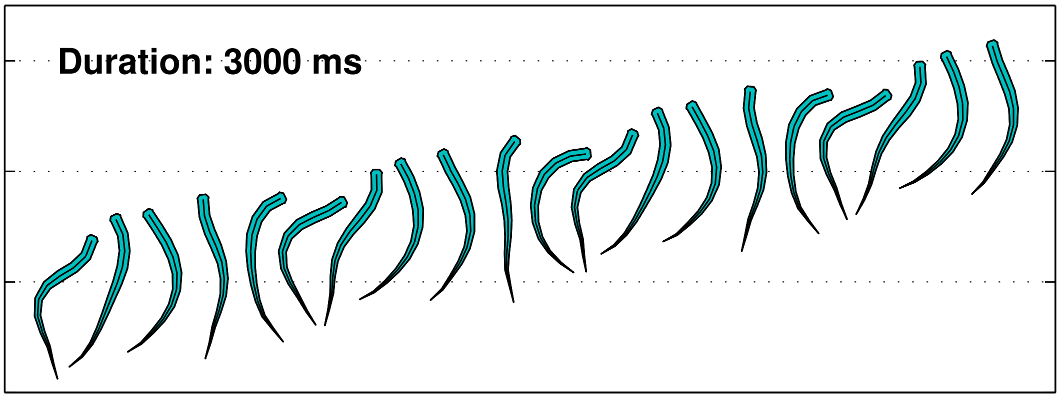
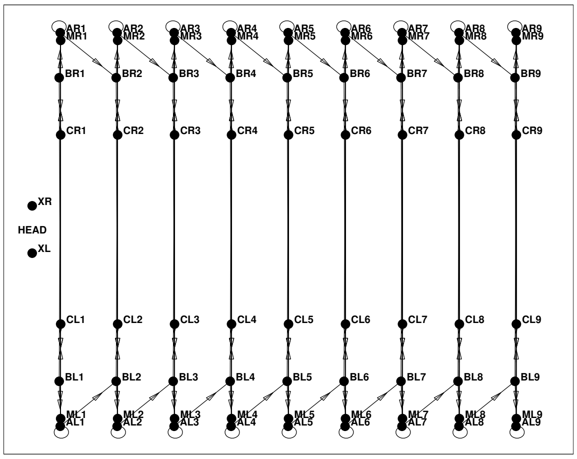


Figure 32: Swimming.



Parameters:		
Tau:	A:	421.3
	B:	50.0
	C:	147.7
Bias:	A:	−3.95
	B:	2.00
	C:	2.00

Weights:

Segment 1		Segment 5		Segment 9	
BR1 → ML1	12.35	BR5 → ML5	12.35	BR9 → ML9	12.35
CR1 → ML1	26.40	CR5 → ML5	26.40	CR9 → ML9	26.40
CR1 → ML1	32.05	CR5 → ML5	32.05	CR9 → ML9	32.05
BL1 → ML1	7.15	BL5 → ML5	7.15	AL9 → ML9	26.40
XL → AL1	1.00	XL → AL5	1.00	BL9 → ML9	7.15
AL1 → AL1	19.90	AL5 → AL5	19.90	XL → AL9	1.00
CL1 → AL1	−32.00	CL5 → AL5	−32.00	AL9 → AL9	19.90
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 33: Neural configuration.

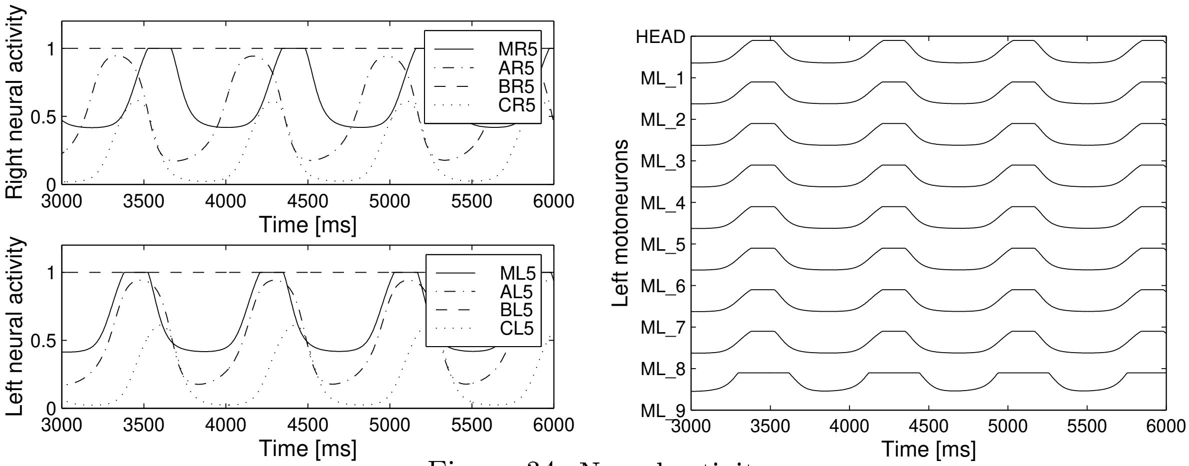


Figure 34: Neural activity.

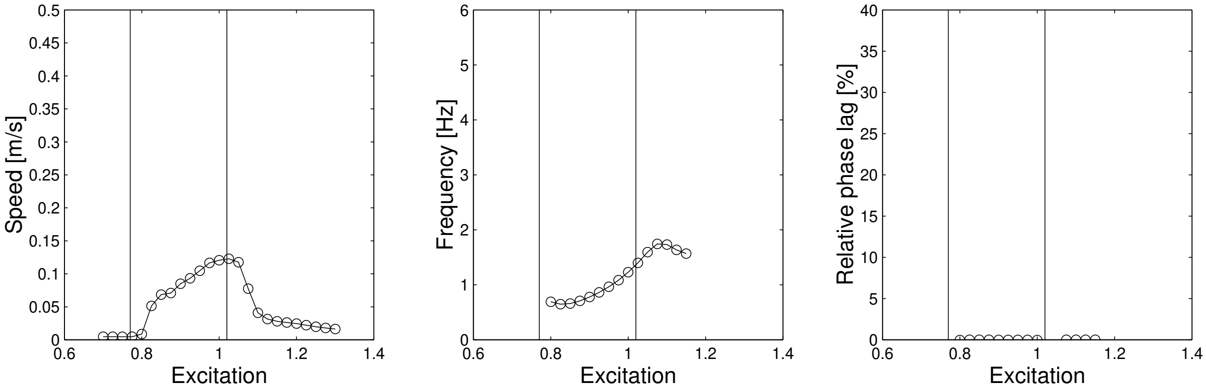


Figure 35: Influence of the tonic input on the speed of swimming, the frequency of oscillation and the relative phase lag between segments.

Controller 8

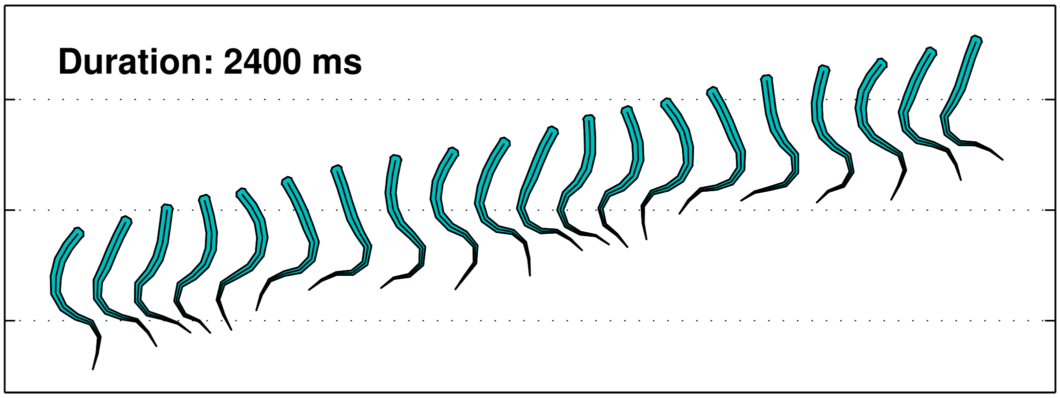
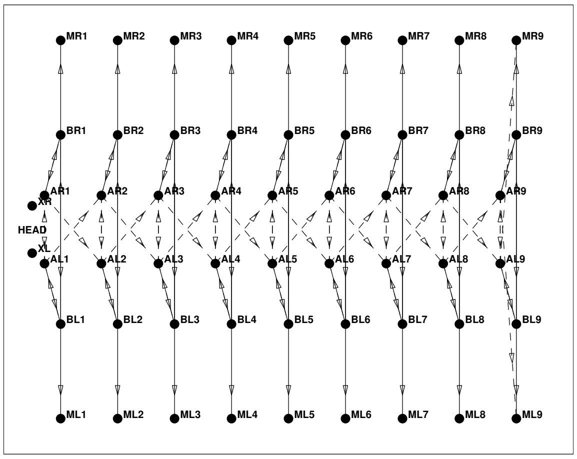


Figure 36: Swimming.



Parameters:

Tau:	
A:	383.0
B:	216.2

Bias:	
A:	2.00
B:	2.00

Weights:

Segment 1		Segment 5		Segment 9	
BL1 → ML1	32.05	BL5 → ML5	32.05	AR9 → ML9	−18.65
XL → AL1	1.00	XL → AL5	1.00	BL9 → ML9	32.05
AR1 → AL1	−15.35	AR4 → AL5	−18.65	XL → AL9	1.00
BL1 → AL1	19.65	AR5 → AL5	−15.35	AR8 → AL9	−18.65
XL → BL1	1.00	BL5 → AL5	19.65	AR9 → AL9	−15.35
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 37: Neural configuration.

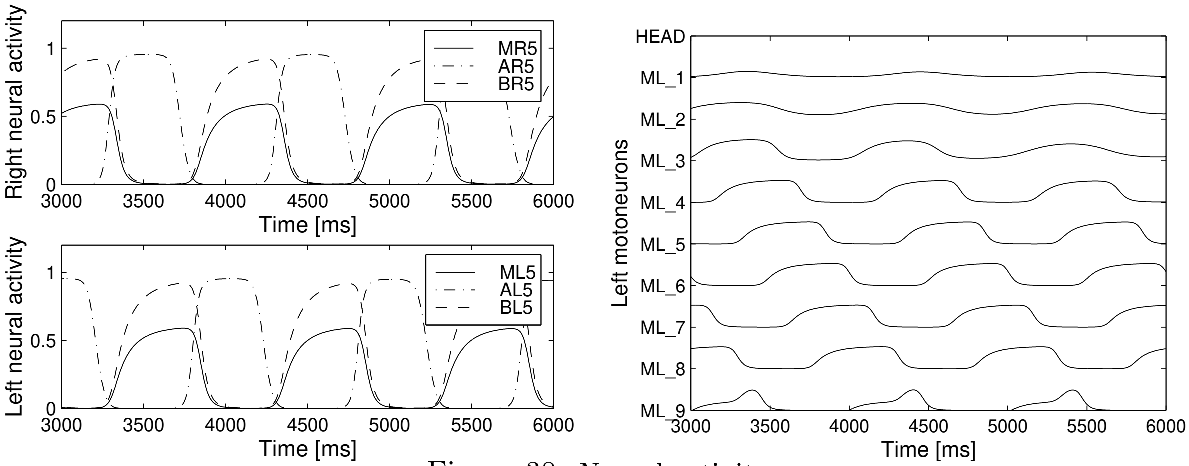


Figure 38: Neural activity.

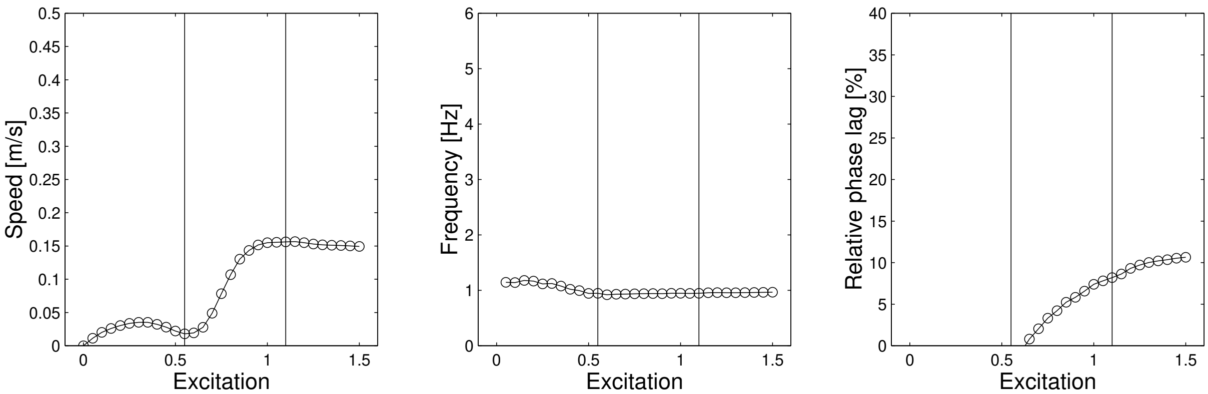


Figure 39: Influence of the tonic input on the speed of swimming, the frequency of oscillation and the relative phase lag between segments.

Controller 9

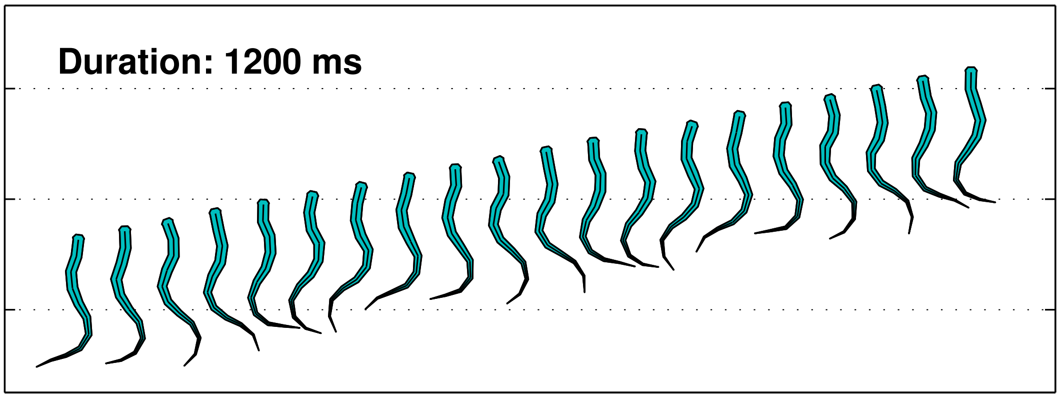
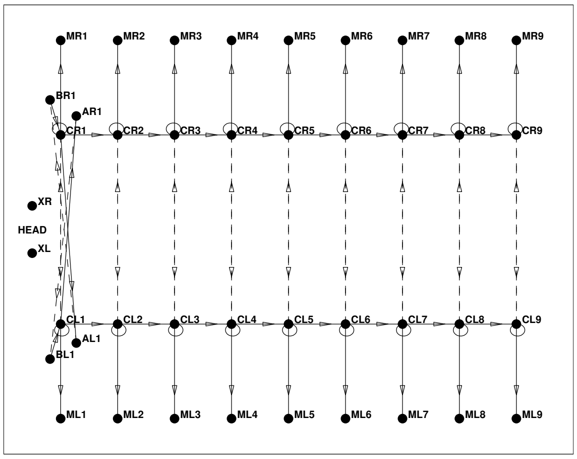


Figure 40: Swimming.



Parameters:

Tau:	
A:	56.9
B:	50.0
C:	50.0
Bias:	
A:	−11.90
B:	−2.10
C:	−8.45

Weights:

Segment 1		Segment 5		Segment 9	
CL1 → ML1	22.40	CL5 → ML5	22.40	CL9 → ML9	22.40
XL → AL1	1.00	XL → CL5	1.00	XL → CL9	1.00
CR1 → AL1	19.95	CR5 → CL5	−11.90	CR9 → CL9	−11.90
XL → BL1	1.00	CL5 → CL5	7.30	CL9 → CL9	7.30
AR1 → BL1	−11.80	CL4 → CL5	26.65	CL8 → CL9	26.65
XL → CL1	1.00				
CR1 → CL1	−11.90				
CL1 → CL1	7.30				
BL1 → CL1	26.65				

Figure 41: Neural configuration.

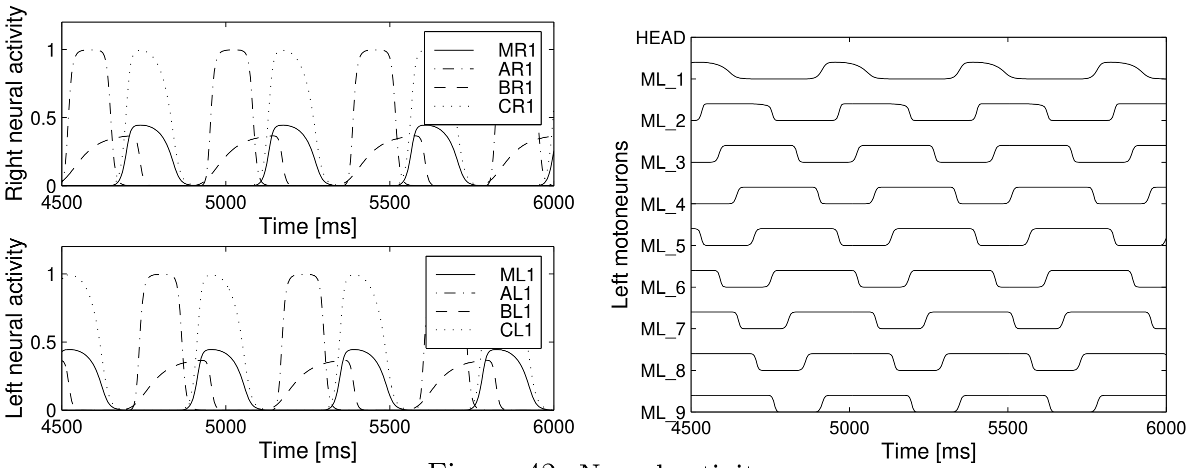


Figure 42: Neural activity.

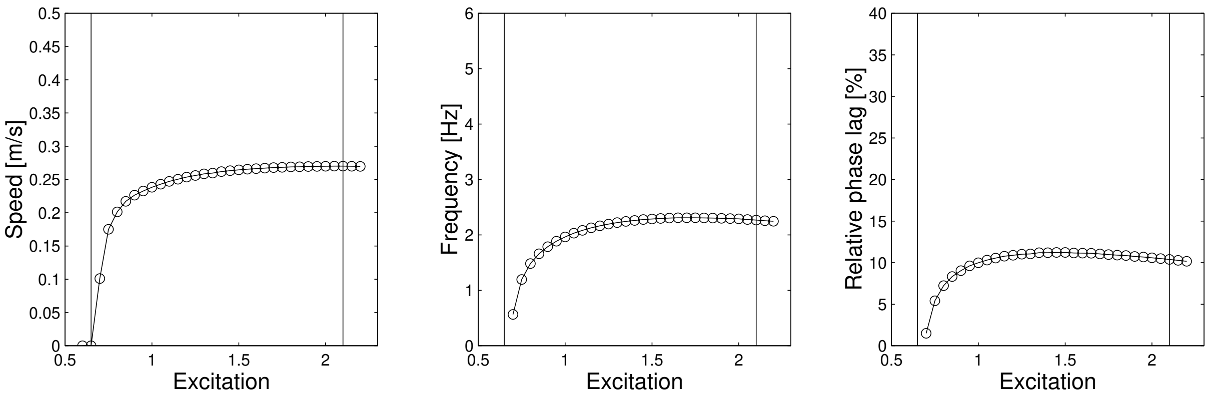


Figure 43: Influence of the tonic input on the speed of swimming, the frequency of oscillation and the relative phase lag between segments.

Controller 10

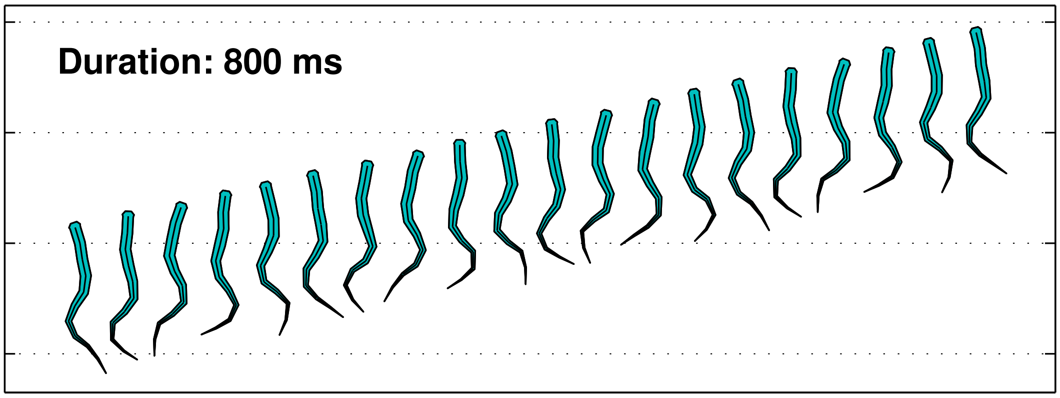
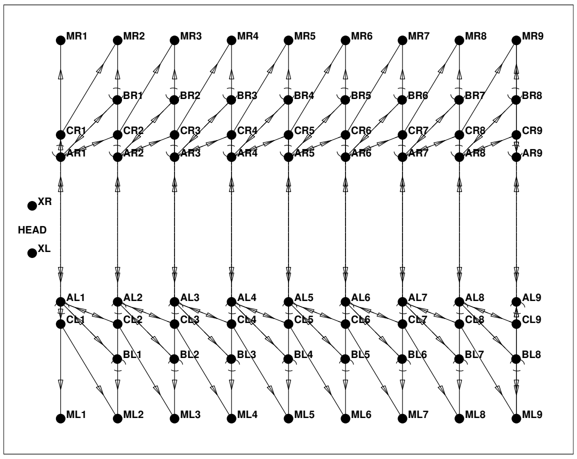


Figure 44: Swimming.



Parameters:

Tau:	
A:	1600.0
B:	29.2
C:	16.5

Bias:	
A:	−16.00
B:	−16.00
C:	2.00

Weights:

Segment 1		Segment 5		Segment 9	
AL1 → ML1	27.00	CL4 → ML5	31.80	CL8 → ML9	31.80
XL → AL1	28.45	AL5 → ML5	27.00	AL9 → ML9	27.00
BL1 → AL1	−6.10	XL → AL5	28.45	CL9 → ML9	31.80
CL2 → AL1	−24.75	BL5 → AL5	−6.10	XL → AL9	28.45
AL1 → AL1	−23.95	CL6 → AL5	−24.75	CL9 → AL9	−6.10
XL → BL1	1.00	AL5 → AL5	−23.95	CL9 → AL9	−24.75
BL1 → BL1	−16.20	XL → BL5	1.00	AL9 → AL9	−23.95
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 45: Neural configuration.

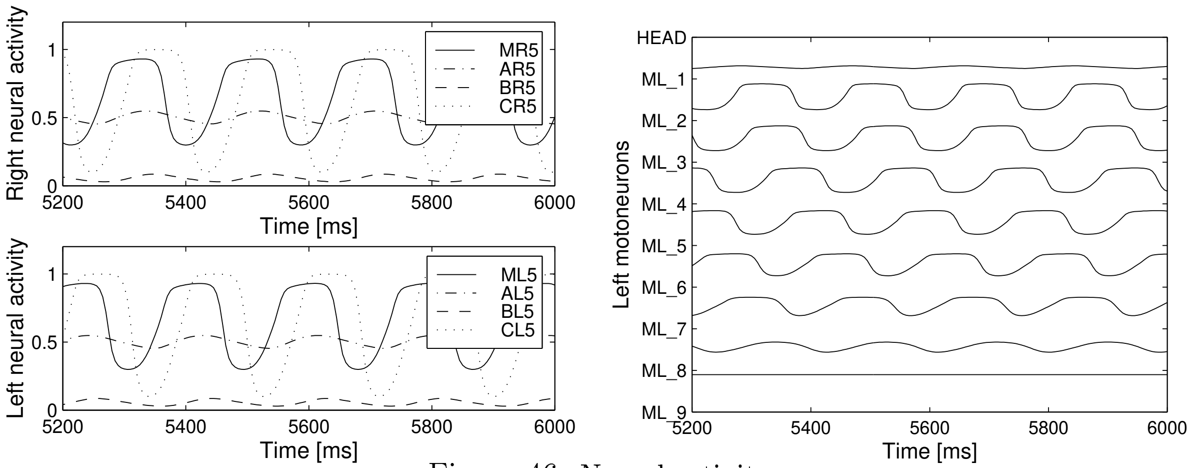


Figure 46: Neural activity.

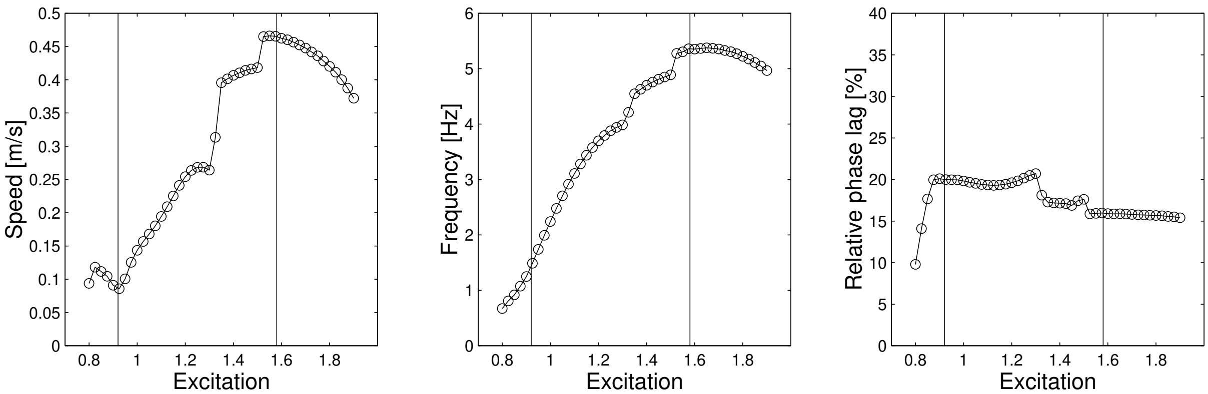


Figure 47: Influence of the tonic input on the speed of swimming, the frequency of oscillation and the relative phase lag between segments.

References

[1] 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.

[2] A.J. Ijspeert, J. Hallam, and D. Willshaw. From lampreys to salamanders: evolving neural controllers for swimming and walking. In *From Animals to Animats, Proceedings of to the Fifth International Conference of The Society for Adaptive Behavior (SAB98)*. MIT Press, 1998.

[3] 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.

[4] J. Kodjabachian and J.-A. Meyer. Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle-avoidance in artificial insects. *to appear in IEEE Trans. on Neural Networks*, 1998.

[5] R.D. Beer. *Intelligence as Adaptive Behavior, an Experiment in Computational Neuroethology.* Academic Press, 1990.

[6] D. Cliff. Computational neuroethology. In M.A. Arbib, editor, *The handbook of brain theory and neural networks*, pages 626–630. MIT Press, 1995.

[7] R. A. Brooks. A robot that walk: Emergent behavior form a carefully evolved network. *Neural Computation*, 1(2):253–262, 1989.

[8] 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.

[9] F. Delcomyn. Neural basis for rhytmic behaviour in animals. *Science*, 210:492–498, 1980.

[10] 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.

[11] 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.

[12] S. Grillner. Neural networks for vertebrate locomotion. *Scientific American*, pages 48–53, January 1996.

[13] 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.

[14] Xin Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 8(4):539–567, 1993.

[15] J. Kodjabachian and J.-A. Meyer. Evolution and development of control architectures in animats. *Robotics and Autonomous Systems*, 16:161–182, 1995.

[16] 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.

[17] 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.

[18] 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.

[19] Ö. Ekeberg. A combined neuronal and mechanical model of fish swimming. *Biological Cybernetics*, 69:363–374, 1993.

[20] 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.

[21] 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.

[22] K. Sims. Evolving virtual creatures. In *Computer Graphics Proceedings, Annual Conferences Series*, pages 15–24. New York : Association for Computing Machinery, 1994.

[23] J.T. Buchanan. Neural network simulations of coupled locomotor oscillators in the lamprey spinal cord. *Biological Cybernetics*, 66:367–374, 1992.

[24] T.L Williams. Phase coupling by synaptic spread in chains of coupled neuronal oscillators. *Science*, 258:662–665, 1992.

[25] 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.

[26] 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.

[27] 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.

[28] R.D. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–510, 1995.

[29] A.J. Ijspeert, J. Hallam, and D. Willshaw. Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology. *Dept. of Artificial Intelligence Research Paper 876, University of Edinburgh*, January 1998.

[30] F. Gruau. Automatic definition of modular neural networks. *Adaptive Behavior*, 3(2):151–184, 1995.

[31] A.J. Ijspeert and J. Kodjabachian. Evolution and development of a central pattern generator for the swimming of a lamprey. *Research Paper No ..., Department of Artifical Intelligence, University of Edinburgh*, 1998.

[32] 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.

[33] R.D. Beer and J.C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122, 1992.

[34] 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.

[35] N. Jakobi. Running across the reality gap: Octopod locomotion evolved in a Minimal Simulation. In Phil Husbands and Jean-Arcady Meyer, editors, *EvoRobot'98: Proceedings of the First European Workshop on Evolutionary Robotics*, Paris, 1998.

[36] B.A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: a survey. *IEEE Transactions on Neural networks*, 6(5):1212–1228, 1995.

[37] *Proceedings of the Artificial Life conferences*. MIT Press.

[38] *From Animals to Animats: Proceedings of the international conferences on Simulation of Adaptive Behavior*. MIT Press.

[39] David Filliat. Evolution de réseaux de neurones pour le contrôle d'un robot hexapode. Technical report, AnimatLab, ENS, Paris, 1998.

[40] 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.

[41] 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.

[42] 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.

[43] Ö. Ekeberg, A. Lansner, and S. Grillner. The neural control of fish swimming studied through numerical simulations. *Adaptive Behavior*, 3(4):363–384, 1995.