Online optimization of swimming and crawling in an amphibious snake robot

Alessandro Crespi and Auke Jan Ijspeert

***Abstract*—An important problem in the control of locomotion of robots with multiple degrees of freedom (e.g., biomimetic robots) is to adapt the locomotor patterns to the properties of the environment. This article addresses this problem for the locomotion of an amphibious snake robot, and aims at identifying fast swimming and crawling gaits for a variety of environments. Our approach uses a locomotion controller based on the biological concept of central pattern generators (CPGs) together with a gradient-free optimization method, Powell's method. A key aspect of our approach is that the gaits are optimized online, i.e. while moving, rather than as an off-line optimization process.**

We present various experiments with the real robot and in simulation: swimming, crawling on horizontal ground, and crawling on slopes. For each of these different situations, the optimized gaits are compared with the results of systematic explorations of the parameter space. The main outcomes of the experiments are (1) optimal gaits are significantly different from one medium to the other, (2) the optimums are usually peaked, i.e. speed rapidly becomes suboptimal when the parameters are moved away from the optimal values, (3) our approach finds optimal gaits in much fewer iterations than the systematic search, and (4) the CPG has no problem dealing with the abrupt parameter changes during the optimization process. The relevance for robotic locomotion control is discussed.

***Index Terms*—Amphibious snake robot, locomotion control, online optimization, swimming, serpentine crawling**

I. Introduction

BOTH animals and biomimetic robots (or, more generally, robots with multiple degrees of freedom) face a complex problem when adapting their locomotion to their environment. Indeed, animals and robots must generally carefully adapt their gaits depending on multiple criteria: whether the ground is soft or hard, slippery or sticky, flat or uneven, horizontal or with a slope. In some cases the environment may even dramatically change between ground, water, and/or air, and locomotion must be adapted to the corresponding physics. In this article, we address the problem of adaptive locomotion with an amphibious snake robot. This problem is very relevant for such a robot because of the tight interaction with the environment: indeed it has multiple contact points with the ground when crawling, and complex interaction with the water when swimming.

We propose a framework for learning locomotion controllers based on two components: a central pattern generator and a gradient-free optimization algorithm: Powell's method. Our approach is inspired by a control mechanism that nature has found to deal with the redundancies in animal bodies and the requirement to easily modulate locomotion: central pattern generators. Central pattern generators (CPGs) are neural networks capable of producing coordinated patterns of rhythmic activity without any rhythmic inputs from sensory feedback or from higher control centers [1]. Even completely isolated CPGs in a Petri dish can produce patterns of activity, called fictive locomotion, that are very similar to intact locomotion when activated by simple electrical or chemical stimulation [2]. Typically, varying simple stimulation allows modulation of both the speed and direction of locomotion. From a control point of view, CPGs therefore implement some kind of internal model, i.e. a controller that "knows" which torques need to be rhythmically applied to obtain a given speed of locomotion. Interestingly, CPGs combine notions of stereotypy (steady state locomotion tends to show little variability) and of flexibility (speed, direction and types of gait can continuously be adjusted).

In this article, we implement a CPG model as a system of coupled amplitude-controlled phase oscillators inspired from the lamprey's swimming CPG. The CPG can produce and modulate the travelling waves necessary for swimming and serpentine locomotion of the amphibious robot Amphibot II (figure 1). The CPG has several explicit parameters, which can be continuously modified, controlling the shape of the generated gaits. Interesting properties of the CPG include: (1) it is computationally cheap, (2) it exhibits limit cycle behavior (temporary perturbations are rapidly forgotten), (3) the limit cycle behavior has an analytical solution with explicit frequency, amplitude, and phase lag parameters that can be used as control parameters, and (4) it produces smooth trajectories even when the control parameters are abruptly changed.

These properties allow us to run an optimization algorithm in parallel of the locomotion controller and to regularly update the CPG parameters online, i.e. during locomotion. The criterion optimized is the forward speed. To maximize the speed (and also to obtain any locomotion at all), the parameters of the CPG have to be adapted depending on the environmental conditions. We demonstrate how a gradient-free optimization algorithm (the Powell's method) can be used to search for the CPG parameters (phase lag, oscillation amplitude) that produce the maximum speed of the robot for a given environment. Our goal is to demonstrate that the CPG implemented as a system of coupled nonlinear oscillators is an ideal building block for doing online optimization in a redundant robotic system. Indeed the optimization algorithm can run in parallel to the CPG and regularly update its parameters. Despite abrupt parameter changes, the produced

A. Crespi and A.J. Ijspeert are with the Biologically Inspired Robotics Group, Faculty of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland. (e-mails: {alessandro.crespi,auke.ijspeert}@epfl.ch)

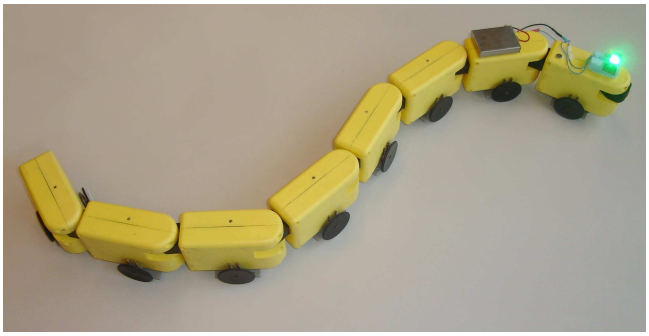


Fig. 1. The robot with passive wheels and the tracking LED mounted on it.

trajectories will smoothly converge towards the new limit cycle after a short transient period. This means that the robot does not need to be stopped or reset between iterations.

This article extends a previous conference article [3] that presents the CPG model and its interactive control with a human-in-the-loop. This work is also related to CPG models used to control a fish [4] and a salamander-like robot [5] that we constructed using the same hardware modules. The novelty here is the online learning and the adaptation to different terrains (different grounds, different slopes) and environments (ground and water). This possibility to do *online* optimization, i.e. learning while moving, is one of the main contributions of this article. Being able to learn gaits online, as opposed to do *offline* optimization with a model or a simulator for instance, is of great importance for biomimetic robotics. Indeed it might be one of the only solutions to tackle the problem of adapting gaits to complex, possibly unknown, environments. Keeping a realistic and up-to-date model of the interaction forces with such environments might be impossible or not accurate enough to allow alternative (e.g., model-based), approaches (see below).

In the rest of the article, we will first review related work (section II). We will then describe the snake robot Amphibot II (section III), the central pattern generator model (section IV), and the simulated robot model (section V). We then present the results obtained with the optimization experiments with the real and simulated robots (section VII). Finally, a discussion of the results is presented in section VIII.

II. Related work

A considerable number of snake-like robots has been constructed. Most of them were designed for use on ground [6], [7], [8], [9], [10], a few were designed for swimming [11], [12]; only a small minority of them has the capability to be amphibious [13], [14]. Their control architecture can roughly be divided into three categories: sine-based, model-based, and CPG-based.

Sine-based approaches use simple time-indexed sine-based functions for generating travelling waves (see for instance [8], [10]). The advantages of such an approach are its simplicity and the fact that important quantities such as frequency, amplitude and wavelength are explicitly defined. A disadvantage is that online modifications of the parameters of the sine function (e.g., the amplitude or the frequency) will lead to discontinuous jumps of setpoints, which will generate jerky movements, risking damages of the motors and gearboxes. This problem can to some extent be overcome by filtering the parameter and/or the outputs but the approach then loses its simplicity. Another disadvantage is that sine-based functions do not offer simple ways of integrating sensory feedback signals. In this article, we will argue that a simple CPG model can combine, without much computational costs, the simplicity of time-indexed sine-based functions with additional interesting properties related to numerical integration of a system of differential equations.

Model-based approaches use kinematic ([15], [16]) or dynamic ([17], [18], [19], [20]) models of the robot to design control laws for gait generation. The control laws are sometimes based on sine-based functions as above (e.g. [15], [20]), but the model-based approaches offer a way to identify fastest gaits for a given robot by using kinematic constraints or approximations of the equations of motion, for instance. Model-based approaches are therefore very useful for helping to design controllers but have two limitations. First, the performance of controllers will deteriorate when models become inaccurate, which is rapidly the case for interaction forces with a complex environment (e.g., friction with uneven ground). Second, the resulting controllers are not always suited for interactive modulation by a human operator.

CPG-based approaches use dynamical systems, e.g. systems of coupled nonlinear oscillators or recurrent neural networks, for generating the travelling waves necessary for locomotion (see for instance [21], [22], [12], [23]). These approaches are implemented as differential equations integrated over time, and the goal is to produce the travelling wave as a limit cycle. If this is the case, the oscillatory patterns are robust against transient perturbations (i.e., they asymptotically return to the limit cycle). Furthermore, the limit cycle can usually be modulated by some parameters which offer the possibility to smoothly modulate the type of gaits produced. Finally, CPGs can readily integrate sensory feedback signals in the differential equations, and show interesting properties such as entrainment by the mechanical body [24].

However, one difficulty with CPG-based approaches is to determine how to design the CPG to produce a particular pattern. Many CPG models do not have explicit parameters defining quantities such as frequency, amplitude, and wavelength (for instance, a van der Pol oscillator does not have explicit frequency and amplitude parameters). This does not need to be the case. In this article, we use a CPG model based on amplitude-controlled phase oscillators. An interesting aspect of this approach is that the limit cycle of the CPG has a closed form solution, with explicit frequency, amplitude and wavelength parameters. The approach therefore combines the elegance and robustness of the CPG approaches with the simplicity of sine-based approaches. Furthermore, our CPG model is computationally very light which makes it well suited to be programmed on a microcontroller on board of the robot. The implementation of the CPG is inspired from lamprey models [25]. It is close to the CPG model presented in [23], but differs in the following aspects: (1) it is made of a double chain of oscillators, (2) it has differential equations controlling the amplitudes of each oscillator (not only the phase), and (3)

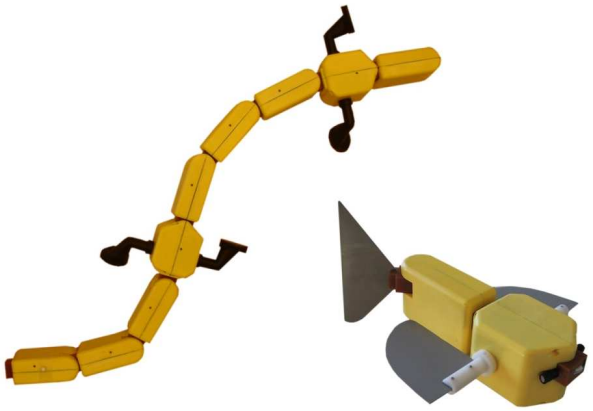


Fig. 2. The salamander (left) and fish (right) robots that have been constructed using the same elements used for the snake robot described in this article.

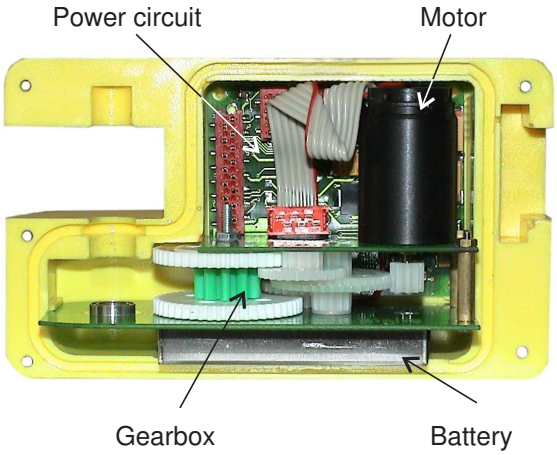


Fig. 3. Internal view of a part of an actuated element. The output axis, normally inserted at the output of the gearbox, is not shown.

the CPG is used to control not only serpentine crawling but also swimming.

III. Hardware description

Our online optimization method is tested with AmphiBot II, an amphibious snake/lamprey robot capable of swimming and serpentine crawling. The robot is an improved version of AmphiBot I [14], a previous amphibious prototype. The main improvements are the following: we added wireless communication, onboard trajectory generation, better electronics, stronger motors, and compliant connection elements. The modules used for Amphibot II were also used to construct Salamandra robotica, a salamander-like robot [5] and Boxybot, a fish robot [4] (figure 2).

The AmphiBot II robot has a modular design and is constructed out of 7 actuated elements and a head element (which is externally identical to the others). The external casing of each element consists of two symmetrical parts molded using lightened polyurethane resin. The elements are connected using a compliant connection piece fixed to the output axis. All the output axes are aligned, therefore producing planar locomotion. To ensure the waterproofing of the robot, custom O-rings are used.[1] The total length of the robot is 77 cm. The asymmetric friction with the ground, required to correctly crawl on the ground, is obtained by fixing a couple of passive wheels to each element. The wheels are removed for swimming, except for experiments implying transitions between water and ground. The density of the robot is slightly lower than 1 kg/m^3, so that it floats under the surface when in water. The battery is placed at the bottom of the elements to have the center of mass below the vertical center, therefore ensuring the vertical stability of the robot during both swimming and crawling.

A. Actuated elements

Each element contains three printed circuits connected with a flat cable, a DC motor with integrated incremental encoder, a set of gears and a rechargeable Li-Ion battery. The elements are thus completely independent from each other.

The motor controller is based on a PIC16F876A microcontroller, which runs a PID motor controller developed at the Autonomous Systems Laboratory of the EPFL and is connected to the I^{2}C bus of the robot. The controller receives feedback from the motor encoder through a quadrature detector, and drives the H-bridges powering the motor with a PWM signal.

The electronics (with the exception of the motor) are powered using 5 V. This voltage is generated from the battery voltage by the power circuit, using a step-up converter. The power circuit also features a battery charger (when empty, the battery can be recharged in approximately one hour) and a battery protection system which avoids damaging the battery by discharging it excessively.

A water detector circuit, used internally to detect and localize any leakage, is placed at the bottom of the element, and is connected to a LED fixed through the top of the element, therefore allowing the user to immediately detect the leakage.

The 2.83 W DC motor (Faulhaber 1724 T 003 SR) has a maximum torque of 4.2 mN·m and drives a gearbox with a reduction factor of 125. The output axis of the gears is fixed to the connection piece, which is inserted into the next element. Six wires are inserted into the axis, and connected to the power boards of two adjacent elements: two are used for the external power, two for the I^{2}C bus, one for the power switch and the last one is reserved for future usage and currently unconnected.

B. Head element

The head element, like the body elements, has three printed circuits (a power board without all the motor-related circuits, a controller board, and a water detector). The controller circuit is based on a PIC18F2580 microcontroller, which is master on the I^{2}C bus of the robot. It implements the CPG (described in section IV), and sends out the setpoints to the motor controllers of each element in real time. The main microcontroller communicates, using a serial line, with a PIC16LF876A microcontroller, which controls a nRF905 radio

[1]During extensive swimming tests, air is insufflated inside the robot by a small pump through a highly flexible silicone tube for maintaining a little overpressure inside the elements and avoiding leakage.

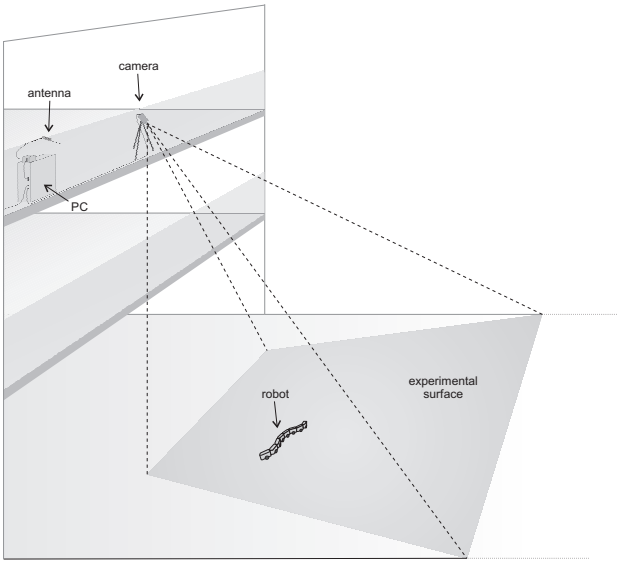


Fig. 4. The video tracking system used for crawling. The setup for swimming is very similar.

transceiver. The antenna is internal to the element and consists of a simple $\lambda/4$ wire (where λ is the wavelength of the used frequency). The radio system uses the 868 MHz ISM band: preliminary experiments showed that a 10 mW signal (the power transmitted by the nRF905) on this frequency can penetrate in water up to at least 30 cm (the maximum tested depth). The more common 2.4 GHz band has not been used because it is heavily absorbed by the water. The maximal bandwidth is approximately 50 kbps, largely enough to send control commands and parameters to the online trajectory generator.

C. Video tracking

To run an optimization algorithm, we need an estimation of the performance of the robot (the speed in this article) for a given set of locomotion parameters. Several solutions to this problem exist. For simplicity, we chose video tracking in this article (in future work, we are planning to provide the robot means to estimate its speed on its own). The tracking system that has been developed for these experiments is relatively simple: a bright 48 lm green led having an irradiation angle of 130° and powered by an independent Li-Ion battery is fixed on the head of robot. The experimental setup is filmed using a Basler a622f camera connected through a IEEE 1394 interface to a PC on which a simple tracking program is running. The whole system is depicted in figure 4.

The tracking program acquires the data from the camera at 15 fps, with a resolution of 800x600 pixels and a depth of 8 bits per pixel (grayscale). The used image processing algorithm is trivial: the coordinates (S_x, S_y) of the LED spot (in pixels) are calculated as the average coordinates of all the pixels having a lightness higher than a given threshold (currently 192). The coordinates are then converted to the real (homogeneous) coordinates of the robot on the plane $(R_x/R_w, R_y/R_w)$ using a 2D transformation matrix:

$$\begin{pmatrix} R_x \\ R_y \\ R_w \end{pmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{pmatrix} S_x \\ S_y \\ 1 \end{pmatrix} \quad (1)$$

The coefficients $a, b, ..., i$ are obtained (for a given placement and orientation of the camera) by solving a linear system:

$$\begin{gathered} aP_{n,x} + bP_{n,y} + c - gD_{n,x}P_{n,x} - hD_{n,x}P_{n,y} - iD_{n,x} = 0 \\ dP_{n,x} + eP_{n,y} + f - gD_{n,y}P_{n,x} - hD_{n,y}P_{n,y} - iD_{n,y} = 0 \\ (\forall n \in [1...4]) \end{gathered} \quad (2)$$

where $D_1...D_4$ are the real coordinates of four reference points (aligned on two parallel lines), and $P_1...P_4$ their coordinates in pixels. The system is currently solved numerically by writing it into matrix form and using SVD decomposition.

The tracking system includes a TCP/IP server, allowing the coordinates of the robot (and its visibility status) to be remotely retrieved in real time.

IV. Central pattern generator model

Our CPG model is based on a system of amplitude-controlled phase oscillators. The design of the CPG is loosely inspired from the neural circuit controlling swimming in the lamprey [26]: it spontaneously produces travelling waves with constant phase lags between neighboring segments along the body, and it is made of multiple oscillators connected as a double chain. An oscillator in the model corresponds to an oscillatory center in the lamprey, i.e. a subnetwork of several thousands of neurons located in one segment of the spinal cord that is capable of producing oscillations independently of other centers.

The CPG model is a double chain of oscillators with nearest neighbor coupling (figure 5). The chain is designed to generate a travelling wave, from the head to the tail of the robot. This wave is used to achieve anguilliform swimming in water and serpentine locomotion on ground. The total number of oscillators is $2N$ where $N = 7$ is the number of actuated joints in the robot. Actuated joints are numbered 1 to N from head to tail. Oscillators in the left chain of the CPG are numbered 1 to N and those on the right side are numbered $N+1$ to $2N$ from head to tail.

The CPG is implemented as the following system of $2N$ coupled oscillators:

$$\begin{cases} \dot{\theta}_i &= 2\pi\nu_i + \sum_{j \in T(i)} w_{ij} \sin\Big(\theta_j - \theta_i - \phi_{ij}\Big) \\ \ddot{r}_i &= a_i\Big(\frac{a_i}{4}(R_i - r_i) - \dot{r}_i\Big) \\ x_i &= r_i\Big(1 + \cos(\theta_i)\Big) \end{cases} \quad (3)$$

where the state variables θ_i and r_i represent, respectively, the phase and the amplitude of the i^{th} oscillator, the parameters ν_i and R_i determine the intrinsic frequency and amplitude, and a_i is a positive constant. The coupling between the oscillators is defined by the weights w_{ij} and the phase biases ϕ_{ij}. An oscillator i receives inbound couplings from the oscillators in the discrete set $T(i)$ according to the topology shown in

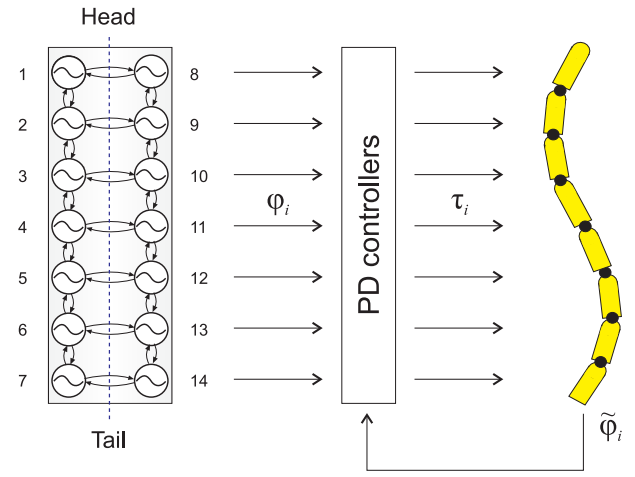


Fig. 5. Structure of the CPG used in the robot.

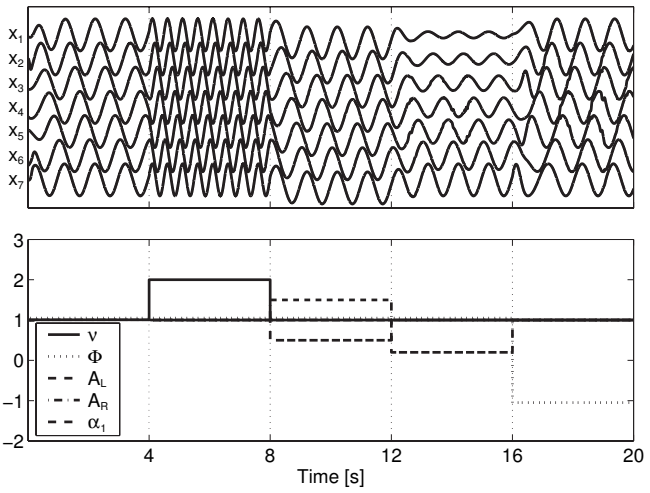


Fig. 6. Effect of changing the parameters of the CPG. Top: setpoint signals, Bottom: control parameters. Initial parameters are $A_L = A_R = 1$, $\nu = 1$ Hz, $\Phi \equiv N \cdot \Delta\phi = 1$ and $\alpha_1 = 1.0$. At $t = 4$ s, ν is temporarily changed to 2.0 Hz, at $t = 8$ s, A_L and A_R are temporarily changed to 0.5 and 1.5 respectively which leads to a negative offset of the setpoint oscillations. At $t = 12$ s, α_1 is changed to 0.2 which leads to oscillations of increasing amplitude from head to tail. At $t = 16$ s, Φ is set to -1.0 which leads to a reversal of the direction of the travelling wave.

Figure 5. For instance, oscillator number 2 receives coupling from oscillators number 1, 3, and 9 ($T(2) = [1, 3, 9]$). The variable x_i is the rhythmic and positive output signal extracted out of oscillator i. The first differential equation determines the time evolution of the phase θ_i. It can easily be shown that two (or more) coupled oscillators will synchronize (i.e., oscillate at the same frequency and with a constant phase lag) if the coupling weights w_{ij} are sufficiently large compared to the differences of intrinsic frequencies (see Appendix). The phase lag between the oscillators will then depend on ϕ_{ij}, w_{ij} and ν_i. The second differential equation is a second order linear differential equation that ensures that the amplitude r_i smoothly converges to R_i in a critically dampened fashion.

The setpoints φ_i, i.e. the desired angles for the N actuated joints, are obtained by taking the difference between signals from the left and right oscillators. A standard PD motor controller is then used to compute τ_i the voltage (i.e., torque) applied to the motor:

$$\begin{aligned} \varphi_i &= x_i - x_{N+1} \\ \tau_i &= K_p e_i + K_d \dot{e}_i \end{aligned} \tag{4}$$

where $e_i = \varphi_i - \tilde{\varphi}_i$ is the tracking error between the desired angles φ_i and the actual angles $\tilde{\varphi}_i$ measured by the motor incremental encoders, and K_p and K_d are the proportional and derivative gains.

In order to reflect the symmetries of the robot and to reduce the number of parameters to optimize, we set several parameters to the same values. The frequency parameters are equal for all oscillators, i.e. $\nu_i = \nu$. We also chose the amplitude parameters on one side of the CPG to be an affine function of the maximal amplitude on that side: $R_i = \alpha_i \cdot A_L$ for the left side ($i = [1, ..., N]$) and $R_i = \alpha_{i-N} \cdot A_R$ for the right side ($i = [N + 1, ..., 2N]$). The α_i parameters are linearly interpolated between the open parameter α_1 (head) and $\alpha_N = 1.0$ (tail). The parameter α_1 therefore acts as an amplitude gain, and allows the CPG to make undulations of increasing amplitude from head to tail, as it is often seen during anguilliform swimming. The phase biases ϕ_{ij} are equal to π between left and right oscillators (i.e., these will oscillate in anti-phase). The phase biases between neighbor oscillators are set to $\Delta\phi$ for the descending connections and to $-\Delta\phi$ for the ascending connections. The parameter $\Delta\phi$ will determine the phase lag between modules, see below. We used $w_{ij} = 4$ for all connections and $a_i = 100$ for all oscillators. The PD coefficients K_p and K_d are tuned manually for each element (e.g., elements in middle of the chain require larger gains than those at the extremities for good trajectory tracking).

With these settings, the CPG asymptotically converges to a limit cycle that is defined by the following closed form solution for the i^{th} actuated joint (a skeleton of the proof is given in Appendix):

$$\varphi_i^{\infty}(t) = \alpha_i \big(A_L - A_R + (A_L + A_R) \cdot \cos(2\pi\nu \cdot t + i\Delta\phi + \phi_0) \big) \tag{5}$$

where ϕ_0 depends on the initial conditions of the system. This means that the system always stabilizes into a travelling wave which depends on the five control parameters ν, $\Delta\phi$, A_L, A_R, and α_1. Indeed the frequency, phase lag, amplitude and offset are directly determined by ν, $\Delta\phi$, $\alpha_i(A_L + A_R)$, and $\alpha_i(A_L - A_R)$, respectively. It is here useful to introduce $\Phi \equiv N\Delta\phi$, the total phase lag between head and tail. The control parameters can be modified online by the optimization algorithm (or by a human operator) from a control PC using the wireless connection. The CPG will rapidly adapt to any parameter change and converge to the modified travelling wave after a short transient period. An example of how the CPG reacts to parameter changes can be observed in figure 6: when the parameters are changed, the oscillator smoothly converges to the new limit cycle, without any discontinuities in the outputs.

The differential equations are integrated by the microcontroller of the head (see section III-B) using the Euler method, with a time step of 10 ms and using fixed point arithmetics.

Note that similar lamprey CPG models based on phase oscillators have been extensively studied by Kopell and Ermentrout and their colleagues [25], [27], [28], [29], [30]. More generally, the behavior of networks of phase oscillators is a large field of study since the pioneering work of Winfree and Kuramoto. See for instance [31], [32], [33], [34], [35], [36]. Unlike our model, most of these models do not have the amplitude as a state variable (it is typically a constant). As discussed in Section II, the closest model used to control a robot is the one developed by Conradt and Varshavskaya [23]. Compared to previous neural network models that we developed of the lamprey CPG [37], [38], the model in this article is simpler (much fewer state variables) and therefore better suited for being programmed on a microcontroller on board of the robot, while keeping the essential features of lamprey travelling wave generation.

V. Simulation

In order to test our approach more systematically and to allow easier adjustments of the environment (e.g., variations of the slope) a simulated model of the robot has been created with Webots [39]. It is controlled by the same CPG of the real robot (with the exception that it is implemented on a PC using standard floating point arithmetics) and has the same mechanical and physical properties of the real robot. The wheels are modelled with asymmetric friction (simulated with a simplification of the Coulomb friction model):

$$\begin{aligned} F_\perp &= -\mu_\perp \cdot F_N \cdot \frac{v_\perp}{|\vec{v}|} \\ F_\parallel &= -\mu_\parallel \cdot F_N \cdot \frac{v_\parallel}{|\vec{v}|} \end{aligned} \quad (6)$$

where $F_\perp$ and $F_\parallel$ are the friction forces perpendicular and parallel to the main axis of each element, F_N is the normal force due to gravity, $\mu_\perp$ and $\mu_\parallel$ are (dynamic) friction coefficients, and $v_\perp$ and $v_\parallel$ are the perpendicular and parallel components of the velocity $\vec{v}$ of the center of mass of the element. The used friction coefficients are $\mu_\perp = 1.0$ and $\mu_\parallel = 0.05$. This friction model is only a first approximation of the dynamics of the passive wheels, and although the simulation is giving maximal speeds similar to the ones obtained with the real robot, the underlying parameters are often quite different (see figures 8 and 11).

VI. Optimization algorithm

The function we want to optimize is the locomotion speed $v(\vec{x})$ of the robot, where $\vec{x}$ is the parameter vector containing the parameters to be optimized (oscillation amplitude A, total phase lag $\Phi \equiv N\Delta\phi$ and amplitude gain α_1). The value of the function for a given set of parameters can be automatically estimated using the video tracking system (the parameters $\vec{x}$ can be sent to the robot using a TCP/IP gateway, see below).

As the convergence time is critical in this context (online optimization of locomotion parameters), methods requiring a large number of function evaluations (e.g., genetic algorithms) have to be avoided. Moreover, we do not have any gradient information for $v(\vec{x})$, and are therefore limited to gradient-free methods. The algorithm we chose is Powell's method [40], which is an heuristic optimization algorithm that rapidly converges for smooth functions. The main risk associated with this kind of algorithm is the possibility to converge to a local optimum of the function, rather than to the global one; however, systematical tests with the snake robot show that the speed function $v(\vec{x})$ is rather smooth with typically a single global optima for a given frequency. A brief description of the algorithm, inspired from the one found in [40], follows.

a) One dimensional optimization: The goal of function optimization is to find x such that $f(x)$ is the highest or lowest value in a finite neighborhood. From now on we just consider the problem of function minimization. Note that function maximization is trivially related because it is equivalent to a minimization $-f(x)$. The main idea of one-dimensional function optimization is to bracket the minimum with three points $a < b < c$ such that $f(b)$ is less than both $f(a)$ and $f(c)$. In this case and if f is nonsingular, f must have a minimum between a and c. Now suppose that a new point x is chosen between b and c. If $f(b) < f(x)$, the minimum is bracketed by the triplet (a, b, x). In the other case if $f(x) < f(b)$, the new bracketing points are (b, x, c). In both cases, the bracketing interval decreases and the function value of the middle point is the minimum found so far. Bracketing continues until the distance between the two outer points is tolerably small [40]. The challenge is finding the best strategy for choosing the new point x in the bracketing interval at each iteration. The Powell's algorithm is based on Brent's method, which is a combination of golden section search and parabolic interpolation [41], [40].

b) Multi-dimensional optimization: Consider a line defined by a starting point P and a direction $\vec{n}$ in N-dimensional space. It is possible to find the minimum of a multidimensional function f on this line using a one-dimensional optimization algorithm [40] (e.g., Brent's method, see above). Direction-set methods for multidimensional function minimization consist of sequences of such line minimizations. The methods differ by the strategies in choosing a new direction for the next line minimization at each stage. Powell's method starts with the unit vectors e_1, e_2, ..., e_N of the N-dimensional search space as a set of directions. One iteration of the algorithm does N line minimizations along the N directions in the set. After each iteration, Powell's method checks if it is beneficial to replace one of the directions in the set by $v_i = P_0 - P_N$ where P_0 was the starting point at the current iteration and P_N the new point after the N line minimizations. For most problems, this significantly increases the speed of convergence compared to using the original unit vectors. The mechanisms for deciding whether or not to include the new direction v_i after each iteration and which direction in the set should be replaced are described in [41], [40]. Note that there is no learning rate; the algorithm simply always goes to the optimum in the next direction. The Powell's method has two open parameters, i.e., the stopping thresholds of the one dimensional and of the multi-dimensional optimizations.

VII. Results

Several optimization experiments, both with the real snake robot and in simulation, have been done, using two fixed frequencies, $\nu = 0.4$ Hz and $\nu = 1.0$ Hz. The frequency has not

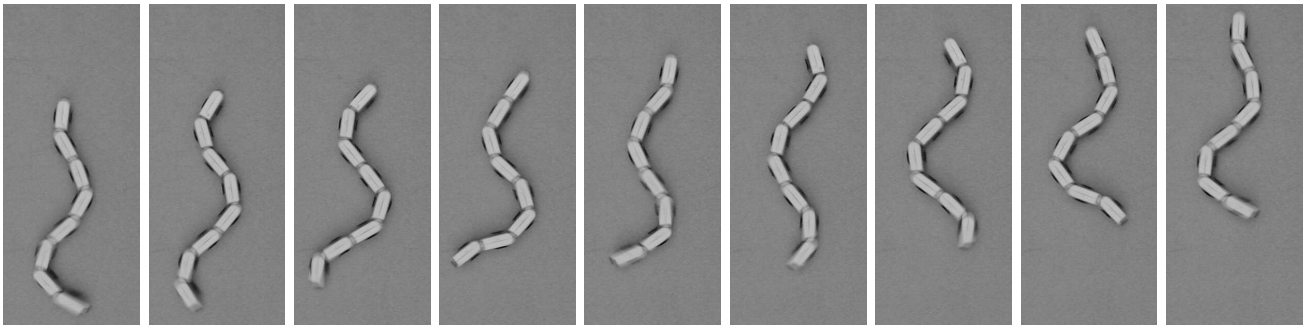
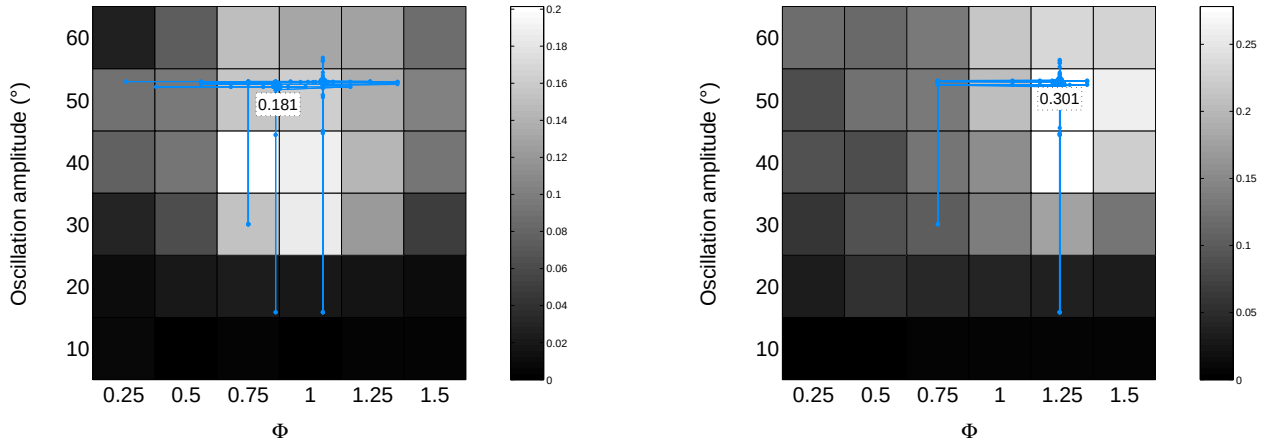


Fig. 7. The robot crawling at $A = 53°$, $\Phi = 1.24$, $\nu = 1.0$ Hz and $\alpha_1 = 0.90$. The time step between the snapshots is 0.12 s. Videos of the robot are available at `http://birg.epfl.ch/amphibot`.



(a) 0.4 Hz (b) 1.0 Hz

Fig. 8. Results of the optimization for crawling. The data from the systematic tests with $\alpha_1 = 1.0$ are plotted in the background, and the evaluations done by five runs of the optimization algorithm are represented by the small dots (the α_1 dimension is not visible in the plot). All the speeds are in m/s. The speed indicated in the caption inside the plot is the highest speed obtained with optimization.

been included in the optimized parameters as the systematic tests showed a direct dependence of the speed on the frequency [42], [3]. The optimization has been done with the real robot for crawling on a horizontal plane and for swimming, and with the simulator for crawling on a horizontal plane and on a slope (ascending and descending). All the experiments with the real robot have been repeated five times. Note that to keep them tractable, all the systematic tests have been done with a fixed value of $\alpha_1 = 1.0$ (and thus in a 2D space, for a given frequency), while the optimization experiments are carried in a 3D space (A, Φ, and α_1). The comparison of the optimization with the systematic tests permits a validation of the optimization algorithm.

A. Optimization of crawling (real robot)

The parameters A, $\Phi \equiv N\Delta\phi$ and α_1 have been optimized at fixed frequencies of $\nu = 0.4$ Hz and $\nu = 1.0$ Hz, on a horizontal linoleum experimental surface. The speed function was evaluated automatically, using the video tracking system, by running the robot for a fixed period of 10 s with the parameters to be evaluated and then measuring its distance from the initial position. Whenever the robot left the experimental surface (i.e., when it was not anymore visible by the tracking camera), the measure was automatically stopped and then restarted from the beginning after a manual repositioning of the robot. The optimization has been run five times, starting from a point at the center of the parameter space ($A = 30°$, $\Phi = 0.75$ and $\alpha_1 = 0.5$). For comparison, systematic tests have been done for the same frequencies, with a fixed value of $\alpha_1 = 1.0$, amplitudes between $10°$ and $60°$ (with a step of $10°$) and a total phase lag between 0.25 and 1.50 (with a step of 0.25).

The results are plotted in figure 8. For $\nu = 0.4$ Hz, the five different optimization runs converged to two slightly different optima: the average parameter values are $A = 52.3°$, $\Phi = 0.87$ and $\alpha_1 = 0.70$, with an average speed of 0.178 m/s (0.231 BL/s) for the first optimum, and $A = 52.3°$, $\Phi = 1.05$ and $\alpha_1 = 0.50$, with an average speed of 0.169 m/s (0.220 BL/s) for the second optimum. The maximal speed obtained during systematic tests was 0.201 m/s (0.261 BL/s). The reason for the slightly lower quality of the optima found by Powell's algorithm (and for the fact that different optima were found) is likely due to variations in the estimation of the speed and to our relatively large tolerance parameters of Powell's algorithm which stops it a bit prematurely in this case. In practice, this is not a problem since the obtained speeds are reasonable. For $\nu = 1.0$ Hz, the algorithm converged to a single optimal result having average parameter values of $A = 52.8°$, $\Phi = 1.24$ and $\alpha_1 = 0.81$, and an average obtained speed of 0.296 m/s (0.384

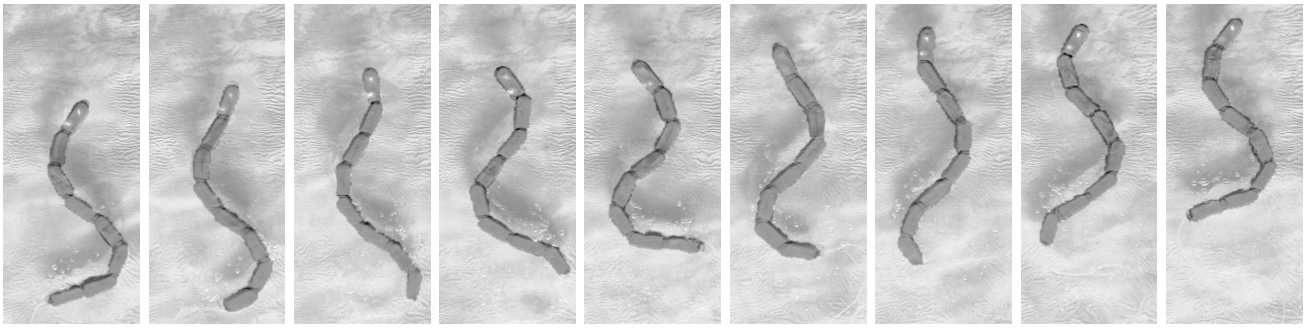
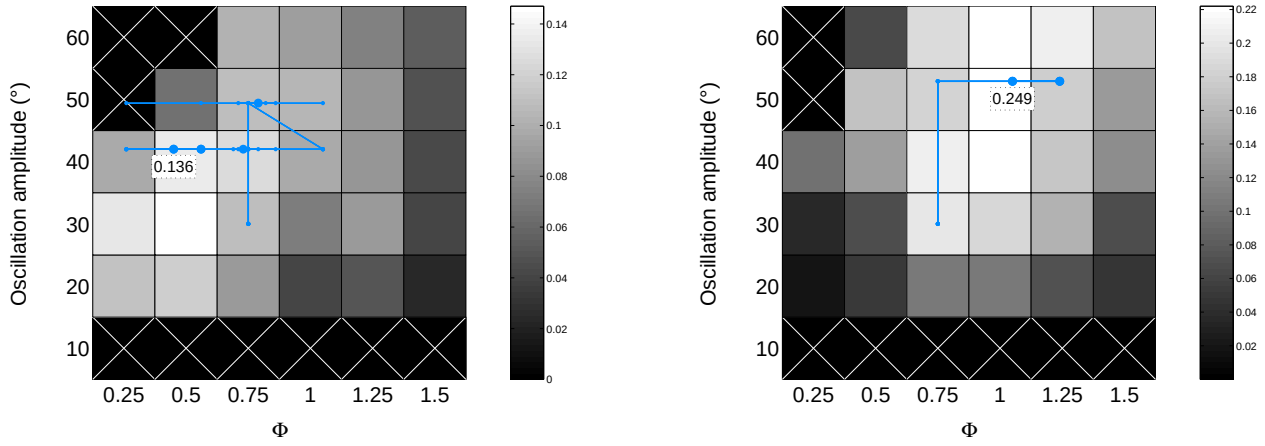


Fig. 9. The robot swimming at $A = 53°$, $\Phi = 1.05$, $\nu = 1.0$ Hz and $\alpha_1 = 0.82$. The time step between the snapshots is 0.12 s. Videos of the robot are available at `http://birg.epfl.ch/amphibot`.



(a) 0.4 Hz (b) 1.0 Hz

Fig. 10. Results of the optimization for swimming at $\nu = 0.4$ Hz and $\nu = 1.0$ Hz. The speed indicated in the caption is the highest speed obtained during optimization.

BL/s). This is better than the maximal speed obtained during systematic tests, which was 0.278 m/s (0.361 BL/s). There were always two iterations of the algorithm, with an average of 32.2 evaluations at $\nu = 0.4$ Hz and of 29.2 evaluations at $\nu = 1.0$ Hz. Figure 7 shows snapshots of the optimal crawling gait at $\nu = 1.0$ Hz.

The obtained gaits have a high amplitude for both frequencies. The other parameters depend on the frequency, which is in agreement with the results obtained during systematic tests; the wavelength is shorter (i.e., $\Phi \equiv N\Delta\phi$ is larger) for $\nu = 1.0$ Hz than for $\nu = 0.4$ Hz, and α_1 increases with the frequency.

B. Optimization of swimming (real robot)

The parameters A, Φ and α_1 have been optimized at fixed frequencies of $\nu = 0.4$ Hz and $\nu = 1.0$ Hz, in an aquarium measuring 2.5 x 0.8 m. The speed function was evaluated automatically, using the video tracking system, by running the robot with the parameters to be evaluated and then measuring its distance from the point reached after an acceleration phase of 1.50 s. For each measure, the robot was placed at the beginning of the aquarium, and stopped when it reached the position threshold (15 cm before the end of the aquarium), or a maximum run time of 10 s (whichever came first). As for crawling, the optimization has been run five times, with the same starting point. Systematic tests have been done for the same frequencies [3], with the same parameter range than for crawling.

The results are plotted in figure 10. For $\nu = 0.4$ Hz, the algorithm converged to several distinct optima (having similar amplitudes but different values of Φ and α_1) with similar resulting speeds and an average of 0.132 m/s (0.171 BL/s). The parameter values for the best result are $A = 42.0°$, $\Phi = 0.45$ and $\alpha_1 = 0.50$, with a resulting speed of 0.136 m/s (0.177 BL/s). The maximal speed obtained during systematic tests was 0.147 m/s (0.191 BL/s). For $\nu = 1.0$ Hz, the algorithm found four optimal results, all having the same amplitude ($A = 52.9°$), but with different values of Φ and α_1. The resulting average speed is 0.220 m/s (0.286 BL/s). The best result has $\Phi = 1.05$ and $\alpha_1 = 0.82$, and the obtained speed is 0.249 m/s (0.323 BL/s). The average speed obtained with the optimization is therefore very similar to the one found during systematic tests, which was 0.222 m/s (0.288 BL/s). There was always only one iteration of the algorithm, with an average of 14.4 evaluations at $\nu = 0.4$ Hz and of 10.4 evaluations at $\nu = 1.0$ Hz. Figure 9 shows snapshots of the optimal swimming gait.

The parameters of the obtained optimal gaits clearly depend on the frequency. As for crawling, the wavelength is shorter when the frequency is higher; the amplitude increases with

the frequency, and there is only a slight change of α_1. The fact that all optimal values of α_1 are smaller than 1.0 is in accordance with the anguilliform swimming with increasing amplitude observed in animals [43].

C. Optimization of simulated crawling

We reproduce here in simulation the optimization of crawling done with the real robot. The use of a simulation allows us to test the optimization in environments that are difficult to realize (see next subsection). The parameters A, Φ and α_1 have been optimized at fixed frequencies of $\nu = 0.4$ Hz and $\nu = 1.0$ Hz, in an environment with friction coefficients $\mu_\perp = 1.0$ and $\mu_\parallel = 0.05$. For each evaluation, the simulator was started with the robot in the initial position, and its average speed measured over 20 s after a stabilization time of 10 s. The starting from was at the center of the parameter space ($A = 30°$, $\Phi = 0.75$ and $\alpha_1 = 0.5$). Systematic tests have been done for the same frequencies, with a fixed value of $\alpha_1 = 1.0$, amplitudes between $10°$ and $60°$ (with a step of $10°$) and a total phase lag between 0.25 and 1.50 (with a step of 0.25).

The results are plotted in figure 11. For $\nu = 0.4$ Hz, the algorithm converged to an optimum with $A = 37.4°$, $\Phi = 0.45$ and $\alpha_1 = 0.82$, with a speed of 0.207 m/s (0.269 BL/s). The maximal speed obtained during systematic tests was slightly higher, 0.222 m/s (0.288 BL/s). For $\nu = 1.0$ Hz, the algorithm found optimal parameter values of $A = 41.5°$, $\Phi = 0.56$ and $\alpha_1 = 0.45$, and a resulting speed of 0.401 m/s (0.521 BL/s), slightly lower than the maximal one found with systematic tests, which was 0.415 m/s (0.539 BL/s). There was always only one iteration of the algorithm, with 9 evaluations at $\nu = 0.4$ Hz and 14 evaluations at $\nu = 1.0$ Hz.

The amplitude and wavelength of the optimum are similar for the two frequencies, and only the α_1 parameter decreases with the frequency. This is a clear difference compared to the results obtained with the real robot: the obtained maximal speeds are higher than the real ones, and the position of the optima in the systematical tests is clearly different. This mostly owes to the used friction model, which is too simplified and needs to be improved in the future to better model the passive wheels' dynamics.

D. Optimization of simulated crawling on a slope

The movement of a snake on a slope has different parameters than on a flat ground [44], and it is clearly expected that this will be also the case for a snake robot.

The parameters A, Φ and α_1 have been optimized in the same way of the simulated crawling, using an environment in which the ground was rotated of a given angle θ. Systematic tests have also been done with the same parameter range.

The results for $\theta = 15°$ (where a positive angle means that the robot climbs on the slope) are plotted in figure 13. For $\nu = 0.4$ Hz, the algorithm converged to an optimum with $A = 57.4°$, $\Phi = 0.69$ and $\alpha_1 = 0.70$, with a speed of 0.054 m/s (0.070 BL/s), which is very similar to the maximal speed obtained with systematic tests, 0.055 m/s (0.071 BL/s). For $\nu = 1.0$ Hz, the algorithm found optimal parameter values of $A = 51.5°$, $\Phi = 0.68$ and $\alpha_1 = 0.78$, and a resulting speed of 0.117 m/s (0.152 BL/s), slightly higher than the maximal one found during systematic tests, 0.109 m/s (0.142 BL/s). There always were two iterations of the algorithm, with a total of 17 evaluations at $\nu = 0.4$ Hz and 22 evaluations at $\nu = 1.0$ Hz. Figure 12 shows snapshots of the resulting optimal gait.

The parameters of the found optima are similar for both frequencies (with a slightly higher amplitude at $\nu = 0.4$ Hz). The algorithm clearly found waves with higher amplitudes than for crawling on a plane, as it is the case for real snakes [44]. The explanation is simple: having higher oscillation amplitudes means an increase of the number of elements perpendicular to the slope, which therefore reduces the slipping downward the slope.

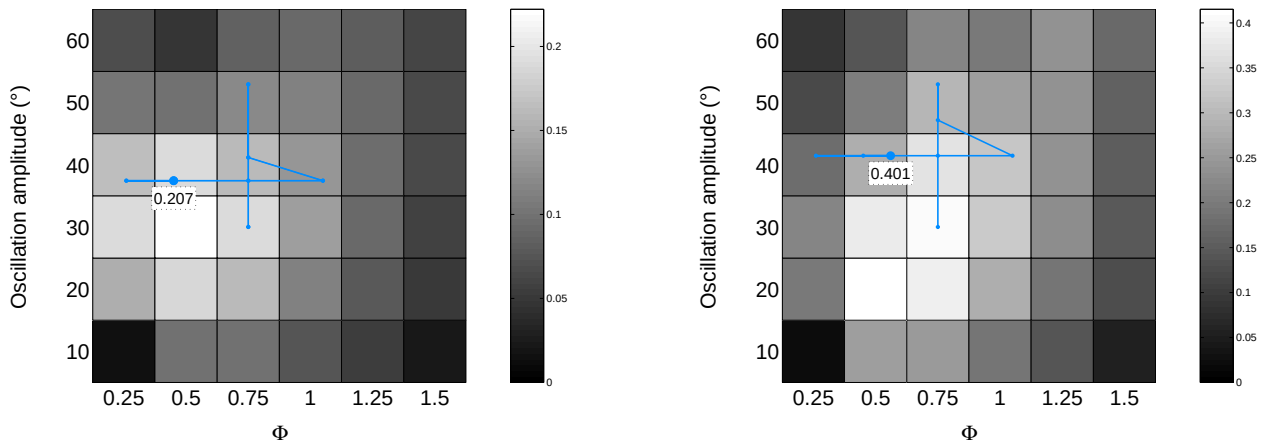
The results for a downward slope ($\theta = -15°$) are plotted in figure 14. For $\nu = 0.4$ Hz, the algorithm converged to an optimum with $A = 0.2°$, $\Phi = 1.31$ and $\alpha_1 = 0.18$, with a speed of 118.7 m/s (154.2 BL/s). The maximal speed obtained with systematic tests is 95.3 m/s (123.8 BL/s). For $\nu = 1.0$ Hz, the algorithm found optimal parameter values of $A = 0.3°$, $\Phi = 1.41$ and $\alpha_1 = 0.11$, and a resulting speed of 118.7 m/s (154.2 BL/s), whereas the maximal one found during systematic tests is 100.6 m/s (130.6 BL/s). There always were two iterations of the algorithm, with a total of 17 evaluations at $\nu = 0.4$ Hz and 22 evaluations at $\nu = 1.0$ Hz.

The obtained speeds are clearly physically unrealistic and are caused by the simplicity of the physical model that does not include velocity-dependent friction terms. The optimal gaits are very similar for both frequencies, and can be summarized as having the robot as straight as possible and letting it freely roll down the slope. This strategy is very different from the one found for climbing.

VIII. Discussion and conclusion

This work has shown that the fastest gaits are considerably different from one medium to the other. For instance, crawling up a slope requires undulations with large amplitudes, while crawling down a slope requires very small amplitudes. And slow swimming requires shorter phase lags than slow crawling. This dependence on the environment is in agreement with observations made by others [44]. In agreement with our previous studies [42], [3], frequency is the parameter whose influence on the speed of locomotion is the simplest: with all other parameters fixed (i.e., amplitude, phase lag, and amplitude gain), increasing the frequency generally leads to an increase of speed (in the range tested). This makes the frequency a useful control parameter. But one should notice that the optimal gaits change with the frequency. It is therefore important to adapt all parameters when the frequency is changed. Another important observation is that the optima are peaked. For a given medium and a given frequency, the speed of locomotion drops rapidly when the parameters are changed compared to their optimal values. In other words, two seemingly very similar gaits might result in dramatically different speeds of locomotion.

All these observations confirm the importance to finely adapt gaits to the environment. There is not a single gait



(a) 0.4 Hz (b) 1.0 Hz

Fig. 11. Results of the optimization for simulated crawling at $\nu = 0.4$ Hz and $\nu = 1.0$ Hz. The speed indicated in the caption is the highest speed obtained during optimization.

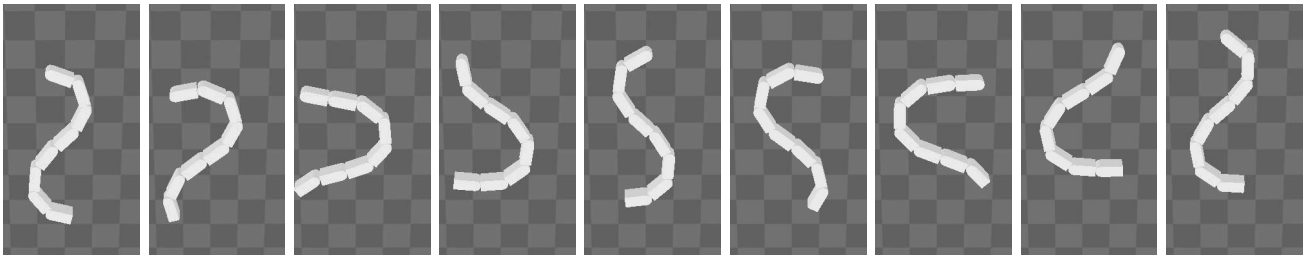
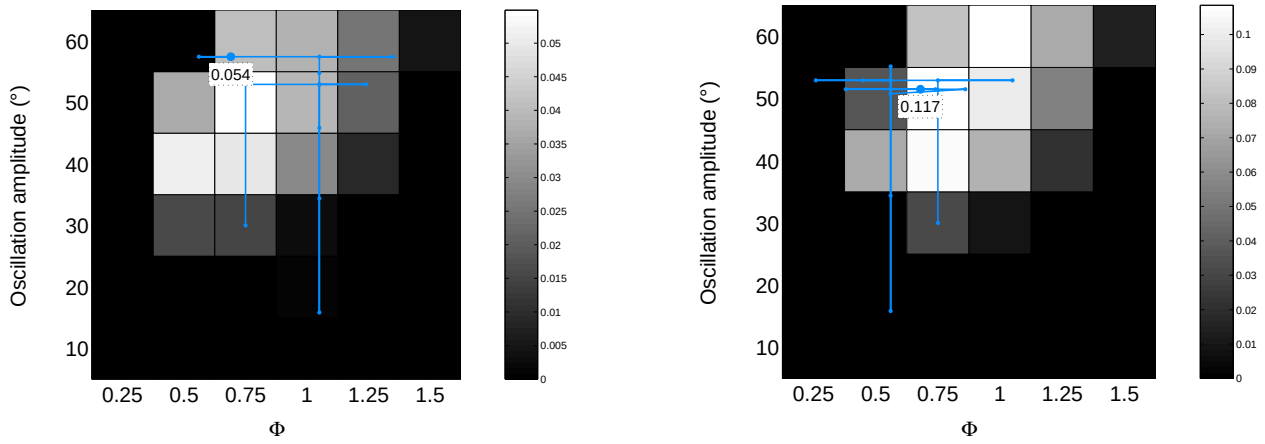


Fig. 12. The simulated robot crawling on a slope ($\theta = 15°$) at $A = 51°$, $\Phi = 0.68$, $\nu = 1.0$ Hz and $\alpha_1 = 0.78$. The time step between the snapshots is 0.12 s.

which performs satisfactorily in all conditions, and a robot that would rely on a single gait for various environments would be strongly suboptimal for most conditions. These empirical tests with our robot have therefore confirmed the necessity of designing online optimization methods for snake robots. Note that in this article, we essentially explored variations of a single type of gait (serpentine crawling). Real snakes exhibit a larger variety of gaits such as side-winding, concertina, and others [44], which our simple planar robot cannot perform.

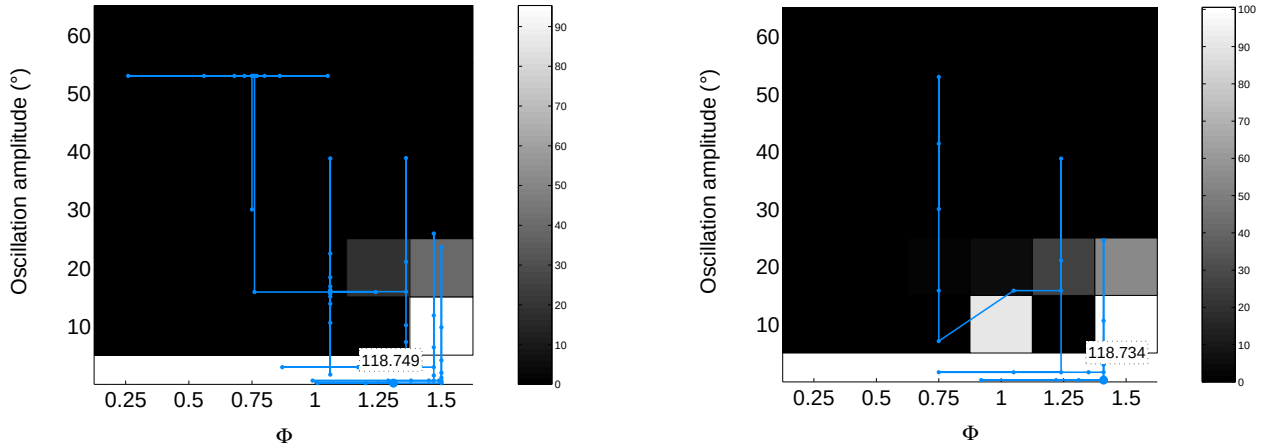
Any method for doing online optimization requires to fulfill at least two characteristics: (1) to allow parameters to be changed online —i.e. to have a control mechanism which smoothly adapts to parameter changes and does not need to be reset between evaluations— and (2) to be fast — in order to avoid excessive wear and tear, and prohibitive testing durations. The results presented here show that our control mechanism satisfactorily fulfills the two requirements. The CPG is a useful building block that is well-suited for optimizing the locomotion and modulating it (e.g., adapting the speed and the direction, see [3]), and for optimizing it. The Powell's method proved to be a useful algorithm for rapidly finding the optimal parameters of the CPG in a given environment. It is significantly faster than doing extensive systematic evaluations of the robot speed on the parameter space. For instance, a systematic exploration of the three-dimensional parameter space considered during our experiments, with 6 steps for each parameter, would require 216 evaluations of the function, whereas the Powell's method can obtain similar results with an order of magnitude less evaluations (between 6 and 37 during the described experiments). In preliminary studies, it has also be found to be one or two orders of magnitude faster than alternative methods such as genetic algorithms [45]. An analysis of the results shows that there is clearly space for improvements: particulary, the stopping conditions of the one dimensional optimization and of the Powell's algorithm itself have to be carefully calibrated, in order to minimize the number of evaluations needed and to avoid stopping the algorithm too early (or too late), as it seems to have been the case during some of the experiments. Compared to related work on learning (e.g. [46], [47]), our approach is very empirical and is fast enough to learn gaits directly on the robot without requiring a simulator or a model. As mentioned in the introduction, we believe this empirical approach is the only viable for many situations, for instance, for complex terrains that cannot be modelled or simulated accurately enough.

In previous work [3], we used results from systematic searches to design interface functions to maintain an optimal gait for a given frequency. Frequency is used as the control parameter that monotonously adjusts speed, and the interface functions adjust the other parameters (amplitude and phase lag) by linearly interpolating between the optimal values found with the systematic search. Two interface functions, one for locomotion on a wooden floor and one for locomotion in



(a) 0.4 Hz (b) 1.0 Hz

Fig. 13. Results of the optimization for simulated crawling on a slope ($\theta = 15°$).



(a) 0.4 Hz (b) 1.0 Hz

Fig. 14. Results of the optimization for simulated crawling on a slope ($\theta = -15°$).

water, were designed. A human operator could thus easily control the speed (and direction) of locomotion by adjusting the frequency (and the asymmetry of amplitudes), without having to worry about the other control parameters. This is done transparently for the human operator, except for the switch between functions for different environments.

This work extends the previous results by allowing to find optimal interface functions for a given environment much faster. There are two interesting outcomes: (1) It is much less tedious to create a database of interface functions for a variety of environments. This database can be used by the human operator to rapidly switch between different locomotion modes (ideally this decision should be made by the robot itself, see below). (2) The optimization is fast enough to be run during operation time for novel environments. For instance, if the robot is brought to a new terrain for a specific mission (e.g., search and rescue), and one notices that locomotion is slow, the operator could rapidly run the optimization process. The optimization takes in average 20 evaluations (i.e., less than 4 minutes), which seems acceptable for finding a good gait.

This work will be extended in several directions in the future. First the robot will be modified such as to be capable of estimating its speed independently of any external device (e.g., by doing odometry with a passive wheel, using an optical mouse sensor, integrating signals from accelerometers, and/or measuring the optical flow of a camera), and of running the optimization algorithm on board instead of on the off-board PC. The current setup with an external camera and PC was used because it was readily available. Second, it would be useful to find ways to allow the robot to discover its environment by itself, for instance to distinguish between water and ground, to estimate the slope, the friction of the surface, etc. This could be done with the addition of sensors and/or by measuring the response to predefined movements. With that information available, we could build some simple heuristics to re-use previously learned gaits: e.g. to query a database with previously optimized gaits for parameters that match the estimated terrain, and to only run the optimization algorithm if the current environment seems too different from previous ones or if the speed seems suboptimal compared to the previous situation. Finally, it would be interesting to apply the online optimization method to deal with real (or simulated)

mechanical damage, e.g. to find new gaits that allow forward motion despite some mechanical failures (e.g., one or several modules that are blocked). The optimization algorithm could be started after an unexpected drop in forward speed, and should in principle lead to modified gaits adapted to the new situation.

ACKNOWLEDGMENT

We acknowledge the technical support of André Guignard and André Badertscher in the design and the construction of the robot. We thank Francesco Mondada and the Autonomous Systems Laboratory (ASL) at the EPFL, for their PD motor controller. This work was made possible thanks to the financial support from the Swiss National Science Foundation.

APPENDIX

The limit cycle of the CPG is determined by the time evolution of the amplitude and phase variables. We here show the particular case of two oscillators coupled bi-directionally with coupling weights $w_{12} = w_{21} = w$ and phase biases $\phi_{12} = -\phi_{21} = \Delta\phi$:

$$\begin{cases} \dot{\theta}_1 &= 2\pi\nu_1 + w\sin(\theta_2 - \theta_1 - \Delta\phi) \\ \ddot{r}_1 &= a(\frac{a}{4}(R_1 - r_1) - \dot{r}_1) \\ \dot{\theta}_2 &= 2\pi\nu_2 + w\sin(\theta_1 - \theta_2 + \Delta\phi) \\ \ddot{r}_2 &= a(\frac{a}{4}(R_2 - r_2) - \dot{r}_2) \end{cases} \quad (7)$$

It is easy to demonstrate that the state variables r_1 and r_2 asymptotically converge to R_1 and R_2, respectively, from any initial condition. Indeed, the variables $[r_1, \dot{r}_1]$ have $[R_1, 0]$ as single stable fixed point. Since we are interested in determining whether these two oscillators will synchronize (i.e., evolve with a constant phase difference), and, if yes, with which phase difference, it is useful to introduce the phase difference $\psi = \theta_2 - \theta_1$. The time evolution of the phase difference is determined by

$$\dot{\psi} = f(\psi) = \dot{\theta}_2 - \dot{\theta}_1 = 2\pi(\nu_2 - \nu_1) - 2w\sin(\psi - \Delta\phi) \quad (8)$$

If the oscillators synchronize, they will do so at the fixed points ψ_∞ (i.e., points where $f(\psi_\infty) = 0$):

$$\psi_\infty = \arcsin(\frac{\pi(\nu_2 - \nu_1)}{w}) + \Delta\phi \quad (9)$$

In our case we have $\nu_1 = \nu_2 = \nu$, and this equation has a single solution $\psi_\infty = \Delta\phi$. This solution is asymptotically stable because $\partial f(\psi_\infty)/\partial\psi < 0$. The outputs of the oscillators therefore asymptotically converge to oscillations that are phase-locked with a phase difference of $\Delta\phi$: $x_1^\infty(t) = R_1(1 + \cos(2\pi\nu t + \phi_0))$ and $x_2^\infty(t) = R_2(1 + \cos(2\pi\nu t + \Delta\phi + \phi_0))$ where ϕ_0 is a constant that depends on initial conditions. Since the complete CPG is made of multiple bi-directionally coupled oscillators and that all parameters ϕ_{ij} are consistent (i.e., the sums of the parameters ϕ_{ij} are equal to a multiple of 2π on any closed path between oscillators), the same reasoning can be recursively applied to demonstrate convergence of the complete CPG. Note that more in-depth analysis of networks of phase oscillators can be found in [32], [25], [27], [28], [29], [30].

REFERENCES

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

[2] S. Grillner. Neural control of vertebrate locomotion – central mechanisms and reflex interaction with special reference to the cat. In W.J.P. Barnes and M.H. Gladden, editors, *Feedback and motor control in invertebrates and vertebrates*, pages 35–56. Croom Helm, 1985.

[3] A. J. Ijspeert and A. Crespi. Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2007)*, pages 262–268, 2007.

[4] D. Lachat, A. Crespi, and A.J. Ijspeert. BoxyBot: a swimming and crawling fish robot controlled by a central pattern generator. In *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob 2006)*, pages 643–648, 2006.

[5] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420, 2007.

[6] G.S. Chirikjian and J.W. Burdick. Design, implementation, and experiments with a thirty-degree-of-freedom 'hyper-redundant' robot. In *4th International Symposium on Robotics and Manufacturing (ISRAM 1992)*, 1992.

[7] B. Klaassen and K.L. Paap. GMD-SNAKE2: A snake-like robot driven by wheels and a method for motion control. In *Proceedings of 1999 IEEE International Conference on Robotics and Automation (ICRA 1999)*, pages 3014–3019, 1999.

[8] G.S.P. Miller. Snake robots for search and rescue. In J. Ayers, J.L. Davis, and A. Rudolph, editors, *Neurotechnology for biomimetic robots*. Bradford/MIT Press, Cambridge London, 2002.

[9] H.R. Choi and S.M. Ryew. Robotic system with active steering capability for internal inspection of urban gas pipelines. *Mechatronics*, 12:713–736, 2002.

[10] D.P. Tsakiris, M. Sfakiotakis, A. Menciassi, G. La Spina, and P. Dario. Polychaete-like undulatory robotic locomotion. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, pages 3029–3034, 2005.

[11] K.A. McIsaac and J.P. Ostrowski. A geometric approach to anguilliform locomotion: Simulation and experiments with an underwater eel-robot. In *Proceedings of 1999 IEEE International Conference on Robotics and Automation (ICRA 1999)*, pages 2843–2848, 1999.

[12] C. Wilbur, W. Vorus, Y. Cao, and S.N. Currie. A lamprey-based undulatory vehicle. In J. Ayers, J.L. Davis, and A. Rudolph, editors, *Neurotechnology for biomimetic robots*. Bradford/MIT Press, Cambridge London, 2002.

[13] H. Yamada, S. Chigisaki, M. Mori, K. Takita, K. Ogami, and Hirose S. Development of amphibious snake-like robot ACM-R5. In *Proceedings of the 36th International Symposium on Robotics (ISR 2005)*, 2005.

[14] A. Crespi, A. Badertscher, A. Guignard, and A.J. Ijspeert. AmphiBot I: an amphibious snake-like robot. *Robotics and Autonomous Systems*, 50–4:163–175, 2005.

[15] J. Ostrowski and J. Burdick. Gait kinematics for a serpentine robot. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation (ICRA 1996)*, pages 1294–1299, 1996.

[16] F. Matsuno and K. Suenaga. Control of redundant 3D snake robot based on kinematic model. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA 2003)*, pages 2061–2066, 2003.

[17] P. Prautsch and T. Mita. Control and analysis of the gait of snake robots. In *Proceedings of the 1999 IEEE International Conference on Control Applications*, pages 502–507, August 1999.

[18] H. Date, Y. Hoshi, M. Sampei, and N. Shigeki. Locomotion control of a snake robot with constraint force attenuation. In *Proceedings of the American Control Conference*, pages 113–118, June 2001.

[19] J. Ute and K. Ono. Fast and efficient locomotion of a snake robot based on self-excitation principle. In *Proceedings of the 7th International Workshop on Advanced Motion Control*, pages 532–539, July 2002.

[20] K. McIsaac and J. Ostrowski. Motion planning for anguilliform locomotion. *IEEE Transactions on Robotics and Automation*, 19(4):637–652, 2003.

[21] Z. Lu, B. Ma, S. Li, and Y. Wang. Serpentine locomotion of a snake-like robot controlled by cyclic inhibitory CPG model. In *The Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, pages 96–101, 2005.

[22] D.P. Tsakiris, M. Sfakiotakis, and A. Vlakidis. Biomimetic centering for undulatory robots. In *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob 2006)*, pages 744–749, 2006.
[23] J. Conradt and P. Varshavskaya. Distributed central pattern generator control for a serpentine robot. In *Proceedings of the 2003 International Conference on Artificial Neural Networks (ICANN 2003)*, 2003.
[24] G. Taga. A model of the neuro-musculo-skeletal system for anticipatory adjustment of human locomotion during obstacle avoidance. *Biological Cybernetics*, 78(1):9–17, 1998.
[25] A.H. Cohen, P.J. Holmes, and R. Rand. The nature of coupling between segmented oscillations and the lamprey spinal generator for locomotion: a mathematical model. *Journal of Mathematical Biology*, 13:345–369, 1982.
[26] 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.
[27] 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. *Journal of Neurophysiology*, 64:862–871, 1990.
[28] 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.
[29] 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.
[30] K.A. Sigvardt and T.L. Williams. Effects of local oscillator frequency on intersegmental coordination in the lamprey locomotor CPG: theory and experiment. *Journal of Neurophysiology*, 76(6):4094–4103, 1996.
[31] A.T. Winfree. Biological rhythms and the behavior of populations of coupled oscillators. *Journal of Theoretical Biology*, 16:15–42, 1967.
[32] Y. Kuramoto. *Chemical oscillations, Waves, and Turbulence*. Springer Verlag, 1984.
[33] A.T. Winfree. *The geometry of biological time*. Springer Verlag, 1990.
[34] Y. Kuramoto. Collective behavior of coupled phase oscillators. In M.A. Arbib, editor, *The handbook of brain theory and neural networks*, pages 223–226. MIT Press, 2003.
[35] B. I. Triplett, D. J. Klein, and K. A. Morgansen. Discrete time kuramoto models with delays. In *Lecture Notes in Control and Information Sciences*, volume 331, pages 9–23. Springer Verlag, 2006.
[36] R. Sepulchre, D. Paley, and N.E. Leonard. Stabilization of planar collective motion with limited communication. *IEEE Transactions on Automatic Control*, In press.
[37] A.J. Ijspeert, J. Hallam, and D. Willshaw. Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology. *Adaptive Behavior*, 7(2):151–172, 1999.
[38] A.J. Ijspeert and J. Kodjabachian. Evolution and development of a central pattern generator for the swimming of a lamprey. *Artificial Life*, 5(3):247–269, 1999.
[39] O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004.
[40] 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.
[41] R. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, 1973.
[42] A. Crespi and A.J. Ijspeert. AmphiBot II: An amphibious snake robot that crawls and swims using a central pattern generator. In *Proceedings of the 9th International Conference on Climbing and Walking Robots (CLAWAR 2006)*, 2006.
[43] G.B. Gillis. Undulatory locomotion in elongate aquatic vertebrates: Anguilliform swimming since Sir James Gray. *American Zoologist*, 36:656–665, 1996.
[44] S. Hirose. *Biologically Inspired Robots. Snake-like locomotors and manipulators*. Oxford University Press, 1993.
[45] D. Marbach and A.J. Ijspeert. Online optimization of modular robot locomotion. In *Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA 2005)*, pages 248–253, 2005.
[46] K. Dowling. *Limbless Locomotion: Learning to Crawl with a Snake Robot*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1997.
[47] G.M. Kulali, M. Gevher, A.M. Erkmen, and I. Erkmen. Intelligent gait synthesizer for serpentine robots. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA 2002)*, pages 1513–1518, 2002.

Alessandro Crespi is a postdoctoral researcher at the Biologically Inspired Robotics Group (BIRG) at EPFL. He has a BSc/MSc and PhD in computer science from the EPFL. His research interests are in the field of biologically inspired amphibious robots. He is mainly working on the development of the electronics of the robots, and on the experiments to characterize their locomotion.

Auke Jan Ijspeert is an assistant professor at the EPFL (the Swiss Federal Institute of Technology at Lausanne), and head of the Biologically Inspired Robotics Group (BIRG). He has a BSc/MSc in physics from the EPFL, and a PhD in artificial intelligence from the University of Edinburgh. His research interests are at the intersection between robotics, computational neuroscience, nonlinear dynamical systems, and applied machine learning. He is interested in using numerical simulations and robots to get a better understanding of sensorimotor coordination in animals, and in using inspiration from biology to design novel types of robots and adaptive controllers. With his colleagues, he has received the Best Paper Award at ICRA2002, the Industrial Robot Highly Commended Award at CLAWAR2005, and the Best Paper Award at the IEEE-RAS Humanoids 2007 conference. He was the Technical Program Chair of 5 international conferences (BioADIT2004, SAB2004, AMAM2005, BioADIT2006, LATSIS2006), and has been a program committee member of over 30 conferences.