Populations of Spiking Neurons for Reservoir Computing: Closed Loop Control of a Compliant Quadruped.

Alexander Vandesompele, Gabriel Urbain, Francis wyffels, Joni Dambre

IDLab-AIRO, Electronics and Information Systems Department, Ghent University - Imec, Ghent, Belgium

Abstract

Compliant robots can be more versatile than traditional robots, but their control is more complex. The dynamics of compliant bodies can however be turned into an advantage using the physical reservoir computing framework. By feeding sensor signals to the reservoir and extracting motor signals from the reservoir, closed loop robot control is possible. Here, we present a novel framework for implementing central pattern generators with spiking neural networks to obtain closed loop robot control. Using the FORCE learning paradigm, we train a reservoir of spiking neuron populations to act as a central pattern generator. We demonstrate the learning of predefined gait patterns, speed control and gait transition on a simulated model of a compliant quadrupedal robot.

Keywords: spiking neural networks, compliant robotics, quadruped control, reservoir computing

1. INTRODUCTION

Compliant robots can provide a greater robustness, flexibility and safety compared to traditional, stiff robots (Pfeifer et al., 2007). However, the control paradigms used in traditional robotics cannot be applied to compliant robots, due to the complexity of predicting the state of the compliant body. The complex dynamics can however be turned into an advantage with the concept of embodied computation (also referred to as morphological computation), where the physical body is treated as a computational resource

(Hauser et al., 2011; Pfeifer et al., 2007; Füchslin et al., 2013).

Physical reservoir computing provides a framework for harvesting the body as a computational resource (Caluwaerts et al., 2013). Monitoring the non-linear body dynamics of a compliant body can be a useful source of information. In some systems extremely little additional computation is required to accomplish a task, for instance locomotion control of a tensegrity robot (Caluwaerts et al., 2013) and control of a soft robotic octopus arm (Nakajima et al., 2013). By combining the body feedback with some additional computational power (e.g. a 'brain'), more complex locomotion tasks can be accomplished (Degrave et al., 2015). The computations that naturally occur in the body are then augmented with a small 'brain' to achieve partially embodied control. In Degrave et al. (2015) this was found to be necessary for gait generation with a quadruped robot. In Burms et al. (2015) and Urbain et al. (2017), more complex tasks were addressed using, respectively, a tensegrity robot and a mass-spring network.

An example of a low level brain function is the generation of rythmic activity by central pattern generators (CPG). CPGs are neural networks in the spinal cord of vertebrate animals, that have been observed to generate rythmic activity and are involved in rythmic movements such as locomotion and respiration (Delcomyn, 1980). Even though biological CPGs can be active without sensory input or descending input from other brain regions, both inputs can modulate the CPG. In decerebrated cat experiments, gait frequency and even gait transition can be controlled with a simple electrical stimulus to the spinal cord (Shik, 1966). Other decerebrated cat experiments revealed that also sensory inputs can modulate the ongoing rythmic activity (reviewed in Rossignol et al. (1993)).

CPGs can be implemented with a neural network by using the reservoir computing framework (Wyffels & Schrauwen, 2009). In reservoir computing, a reservoir is excited by inputs and provides a spatiotemporal expansion of this input. The reservoir is typically a randomly connected neural network, of which the weights are rescaled such that the network operates at 'the edge of chaos' (Legenstein & Maass, 2007). The output is then a linear mapping of the reservoir activity. In our setup, by feeding body sensors of a robot to a randomly connected reservoir, a spatiotemporally enriched interpretation of the body sensors is created. Thanks to this expansion, more complex

patterns can be extracted from the reservoir activity using only linear regression. Reservoir computing with spiking neurons is traditionally performed with a liquid state machine (Maass et al., 2002). Here, we propose using population coding, where the unit of the reservoir is a population of spiking neurons. Whilst this method allows to apply the same principles as in the well established rate based reservoir computing, it also allows to potentially profit from using a spike based implementation. The number of tunable parameters, both at neuron and population level allows for optimizing reservoir dynamics for closed loop dynamical systems. Additionally, efficient hardware implementations (e.g. SpiNNaker, Furber et al. (2014)) could allow to run the network with low power usage on mobile robots. Lastly, this framework allows interfacing with spike-based sensors (e.g. the DVI camera, Lichtsteiner et al. (2008)) that provide low latency and low redundancy sensor data.

In this paper, we demonstrate the feasability of using populations of spiking neurons in embodied computation by creating stable closed loop locomotion control for a compliant robot. To achieve this, we applied the physical reservoir computing framework to a simulated model of the Tigrillo robot (Willems et al., 2017), a compliant quadrupedal platform. We add a 'brain' to the robot which is also a reservoir, consisting of spiking neurons. This neural network is trained to function as a CPG and, similar to biological CPGs, can be modulated by both body sensors and simple control inputs. To create a stable dynamical system, capable of generating robust periodic movements, online linear regression can be applied (FORCE learning, Sussillo & Abbott (2009)) in a gradual fashion (Caluwaerts et al., 2013). In Nicola & Clopath (2017), FORCE learning was applied to spiking neural networks for the extraction of complex, dynamical signals. Here, we apply FORCE learning for closed-loop locomotion of a robot model, introducing the robot body in the loop. Figure 1 presents an overview of the implemented system. Four readout neurons are trained to produce motor signals for the actuated joints of the Tigrillo model. Four body sensors, sensing the angle of the passive joints, are fed as input to the neural network.

In the next section the components of the closed loop system are presented and the learning algorithms involved are detailed, including a method for monitoring the reservoir state. The results section presents different gait patterns that have been learned, as well as gait frequency control and gait

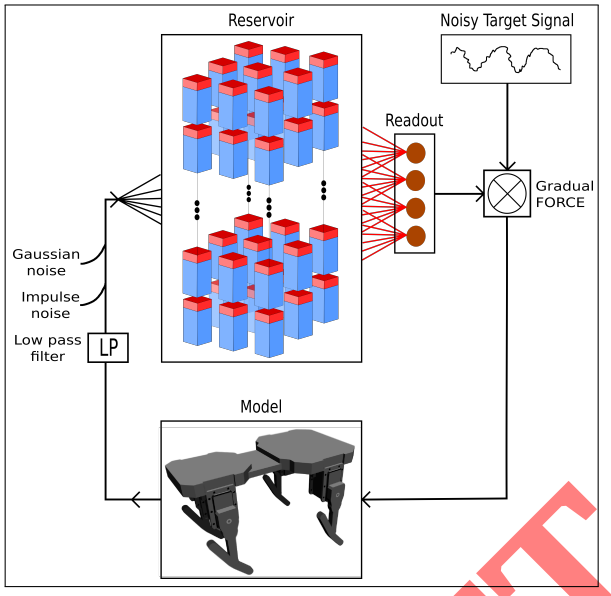

Figure 1: Overview of the closed loop control system. Learned connection weights in red.

switching control.

2. MATERIALS & METHODS

2.1. Simulation

The model used in this work is based on the physical Tigrillo robot (Willems et al., 2017), see Figure 2. Tigrillo is a low-cost platform developed for researching compliance in quadrupeds. The robot has four legs, consisting of two joints: one actuated with a servo motor (hips and shoulders) and one passive joint (knees and elbows). The passive joints are loaded with a spring, providing compliance. The angle of the passive joints (on the physical robot measured with Hall effect sensors) reflects the state of the robot body and its interaction with its environment, and is therefore useful as a sensor input in the closed loop control system. The Tigrillo model is a parametrized stick and box model that mimicks the weight distribution and physics of the physical robot.

All simulations have been performed on the Neurorobotics Platform (NRP, Falotico et al. (2017)). The NRP provides an interface between an environment simu-

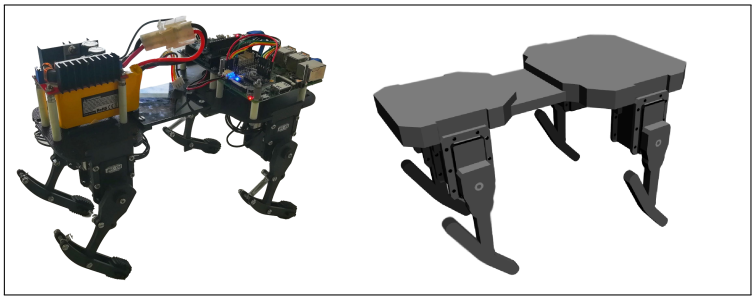

Figure 2: Tigrillo physical robot (left) and model (right).

lator (Gazebo) and a spiking neural network simulator (e.g, NEST, Gewaltig & Diesmann (2007)). In this work, ODE (Drumwright et al., 2010) was used as physics engine.

2.2. CMA-ES

To find hip joint motor signals for gaits that suit the body dynamics, the covariance matrix adaptation evolutionary strategy (CMA-ES) algorithm (Hansen & Ostermeier, 2001) is used. CMA-ES is an evolutionary algorithm that samples solutions from a multi-variate normal distribution. Every iteration, the mean and the covariance matrix of the distribution are updated. The mean is updated to increase the likelihood of previously successful solutions. The covariance matrix is updated to increase the likelihood of a previously successful search step. CMA-ES can handle non-convex fitness landscapes with many local maxima well. It requires few initial parameters and doesnt require derivation of the search space.

The CMA-ES is used to optimize parameters of a parametrized CPG (described in Gay et al. (2013)). This CPG is implemented by a set of three equations:

$$\begin{aligned} \dot{r} &= \gamma(\mu - r^2)r \\ \dot{\phi} &= \omega \\ \lambda &= r\cos(\phi_L) + o \end{aligned} \tag{1}$$

Where ϕ and r describe the current phase and the radius of the oscillator, respectively. Both are used to calculate the actual control value λ (in

degrees). μ is the target amplitude of the oscillator and γ is a positive gain that defines the speed of convergence of the radius to the target amplitude. o is the offset and ω the radial frequency of the oscillator. ϕ_L is a filter applied on the phase of the oscillator and is different for the stance and swing phase of the control as determined by the duty factor (d):

$$\phi_L = \begin{cases} \frac{\phi_{2\pi}}{2d} & \text{if} \quad \phi_{2\pi} < 2\pi d \\ \frac{\phi_{2\pi}+2\pi(1-2d)}{2(1-d)} & \text{otherwise} \end{cases}$$

$$\text{and} \quad \phi_{2\pi} = \phi \pmod{2\pi} \tag{2}$$

The Tigrillo platform has four actuated hip joints controlled by four phase-coupled CPGs. The front left leg is chosen as reference leg and the phase difference of the remaining 3 legs is described by three phase offset (po) parameters. This is implemented by adding a term to the formula for the phase (ϕ) in equation 1). For instance, for the coupling between the front left and front right oscillators:

$$\dot{\phi}_{fr} = \omega + w_{fr} sin(\phi_{fl} - \phi_{fr} - po_{fr}) \tag{3}$$

with w_{fr} the coupling strength.

Initial parameters had a Gaussian distribution with 0.5 mean and 0.2 SD. Each generation consisted of 25 individuals, other parameters were kept at default as described in Hansen (2006). Different gaits were found by optimizing different subsets of parameters. The parameters optimized in the search for the walking gait are listed in Table 1. The CPG frequency was kept constant at 1.44 Hz. The distance travelled from the origin was used as fitness function.

2.3. The Neural Network

The neural network is a reservoir consisting of 300 populations of spiking neurons (unless specified otherwise), arranged in a three dimensional structure of 3x3 layers (Figure 1). Each excitatory neuron of a population connects to a neuron of another population with a probability proportional

Parameter	Symbol	Range	Unit
Front amplitude	μ_f	[20, 140]	degrees
Hind amplitude	μ_h	[20, 140]	degrees
Front duty cycle	d_f	[0.15, 0.85]	*NA*
Hind duty cycle	d_h	[0.15, 0.85]	*NA*
Front offset	o_f	[−60, 60]	degrees
Hind offset	o_h	[−60, 60]	degrees
Front right phase offset	po_{fr}	[150, 210]	degrees
Hind left phase offset	po_{hl}	[240, 300]	degrees
Hind right phase offset	po_{hr}	[60, 120]	degrees

Table 1: Parameters and their ranges included in the *CMA-ES* optimization for the walking gait.

to the Euclidean distance between both populations (see Table 4). This distance-based connectivity is not only biologically plausible but also makes the simulation and the potential hardware implementation feasible as it reduces the overall number of connections. The delay of spike transmission between populations is fixed at 100ms. Each population consists of 40 neurons of the leaky-integrate-and-fire (LIF) type with exponentially decaying post-synaptic current (*iaf_psc_exp*, as described in Tsodyks et al. (2000)). Neuron parameters are close to the default, bioplausible values, or hand tuned for desired population response properties (Table 2). The ratio of inhibitory/excitatory neurons is 1/4. Within a population, excitatory neurons connect to inhibitory neurons and vice versa (see Figure 3 and Table 4). All neurons in a population receive a white noise current of mean 0 and SD 2, this is important in maintaining a responsive population.

2.4. The interface between neural network and body

Interfacing the spiking neural network with the robot body requires translating spiking activity to analog values and vice versa. The motors of the actuated joints expect an analog value, the desired joint angle. Each motor has a readout neuron that provides this value. The parameters of the readout neurons have been adapted such that its membrane potential can be used directly as motor signal (see Table 3). Most importantly, the spiking

threshold is set to infinity, preventing the neuron from firing which would reset the membrane potential. As a result the readout neuron is simply a leaky integrator of its incoming spikes. In the other direction, body sensor data is fed to the neural network. Therefore a DC current proportional to the values of a sensor is injected into a sensor population, whose activity then closely reflects the sensor stream. In this fashion the interface between the spiking network and the body is accomplished.

The readout neurons are connected to all reservoir populations. The weights of these connections are learned with FORCE learning (Sussillo & Abbott, 2009). Therefore, the reservoir states (i.e. the population activities) need to be known at all times. To observe the reservoir states, each population is monitored by a monitor neuron. Monitor neurons are identical to readout neurons, but are connected to a single reservoir population with unit weight. The membrane potential of the monitor neuron represents the population activity (Figure 4 shows an example of the membrane potential of a few monitor neurons) and is used by the FORCE learning algorithm.

Parameter	Value
Membrane resting potential [mV]	-65
Spiking threshold [mV]	-50
Post spike reset membrane potential [mV]	-75
Membrane capacitance [nF]	0.2
Membrane time constant [ms]	30
Duration refractory period [ms]	2
Post-synaptic time constant [ms]	0.5

Table 2: Parameters of the LIF neuron model.

2.5. Gradual FORCE learning

The aim of the learning is to find connection weights that make the membrane potential of the four readout neurons (see Figure 1) produce the predefined target signals, i.e. the four motor signals as found by the CMA-ES optimization.

Parameter	Value
Membrane resting potential [mV]	0
Spiking threshold [mV]	∞
Membrane capacitance [nF]	0.2
Membrane time constant [ms]	30
Post-synaptic time constant [ms]	5.5

Table 3: Parameters of the LIF readout and monitor neurons.

Connection	Variable	Pconnect
Intra-population: Excitatory to inhibitory	C_{ei}	0.1
Intra-population: Inhibitory to excitatory	C_{ie}	0.1
Inter-population: Excitatory to excitatory	C_{ee}	$0.3e^{-D^2}$
Excitatory to monitor neuron	C_{em}	1.0

Table 4: Connectivity of the population model per connection type. Pconnect = connection probability between any two neurons. D = Euclidean distance between two populations.

Reservoirs with feedback are challenging to train due to the feedback introducing delayed effects. FORCE learning allows to impose behaviour of a reservoir with feedback using the recursive least-squares algorithm. It suppresses unstable behaviour by reducing the error magnitude from the onset of the learning procedure.

By using monitor neurons, we effectively isolate the unweighted contribution of each reservoir population to a readout neuron. Since readout neurons and monitor neurons have identical parameters, the membrane potential of a readout neuron will be a linear combination of all monitor neuron membrane potentials. Therefore it is possible to use the monitor neurons membrane potentials as reservoir states, and FORCE learning can be applied in an identical fashion as with rate-based neural networks.

To make a stable closed loop system, the control signals are first learned in open loop with the control signals as input to the actuators. Subsequently, to

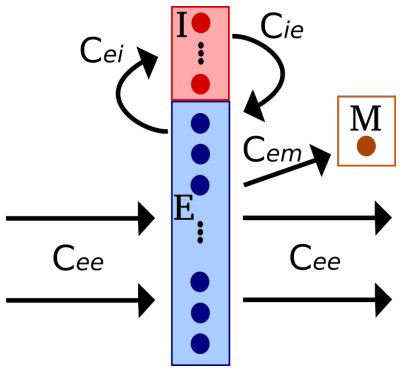

Figure 3: Connectivity of the population model. E = excitatory, I = inhibitory, M = monitor neuron. For C_{ei}, C_{ie}, C_{ee} and C_{em} see Table 4.

ensure a smooth transition to closed loop control, the target signal and readout signal are gradually mixed (as described in Caluwaerts et al. (2013)). The contribution of the readout neuron is gradually increased during this transition. Finally, the system is capable of autonomously producing the target signals in a stable closed loop fashion (as detailed in the results section).

The regularization variable α of the FORCE learning algorithm must be selected large enough to prevent overfitting, but not too large as it could fail to approximate the target function sufficiently fast (Sussillo & Abbott, 2009). After a parameter sweep, a value of 50 for α was observed to be effective. Furthermore, to ensure robustness of the closed loop system, it is necessary to insert sources of noise during learning. Here, impulse noise and gaussian noise were added to the sensor signals (see also Figure 1). Similarly, noise is added to the target signals. A low pass filter is added to smoothen the sensor signals before injecting them to the reservoir. The actuators also posses low-pass properties, which filters out some of the noise due to the implementation with spiking neurons.

3. RESULTS

3.1. Gait Generation

The system is capable of learning and sustaining different gaits that have been found using CMA-ES. Figures 5 and 6 display the target motor signals and the generated motor signals during a closed loop walking and bounding

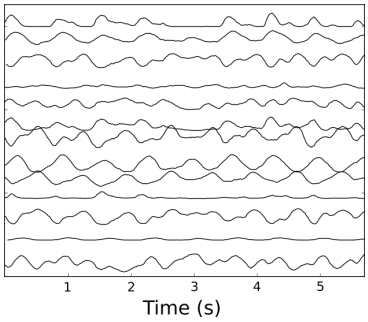

Figure 4: Membrane potentials of a few monitor neurons. Each monitor neuron represents the activity of one population of spiking neurons.

gait, respectively. In these experiments learning took 40 seconds simulated time for open loop training followed by 40 seconds for closed loop training. After learning, the gait generation is sufficiently robust to continue after disturbances such as moving the robot or stopping movement by turning the robot on its back for a while. The learned motor signal are a bit noisy. This is mainly due to the rather small population sizes.

3.2. Speed Control

In order to obtain gaits with tunable speed, we added an extra control input to the reservoir that serves as a control signal to control gait frequency. Similarly to sensor inputs, the control input is implemented as a DC current to the reservoir. During training, incremental frequencies of the same gait are paired with an incremental control signal. After learning, the control signal can be used to alter the frequency (Figure 7). The total learning time for this experiment was 200 seconds.

3.3. Gait Transition

Here, the network is trained to produce both gaits presented previously (walking and bounding). Again, a simple high level control input is used during and after training to control the gait transition (Figure 8). For this experiment, the reservoir was made more powerful by increasing the number of populations to 600 and the number of neurons per population to 100. The total learning time for this experiment was 200 seconds.

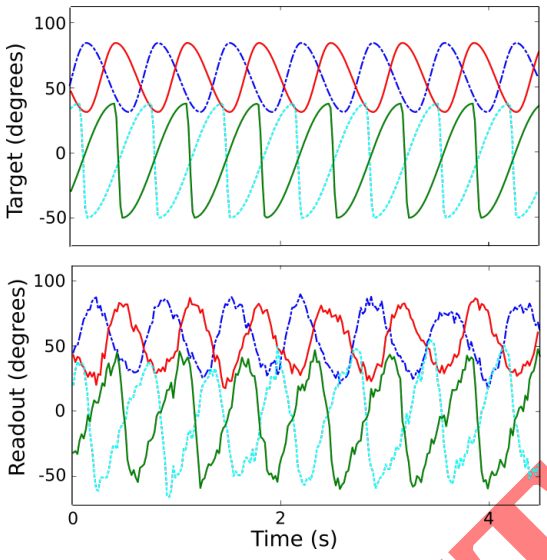

Figure 5: Walking Gait. Top pane: target motor signals (red, blue, green and cyan for front left, front right, hind left and hind right legs respectively). Bottom pane: readout signals during closed-loop control, after FORCE learning.

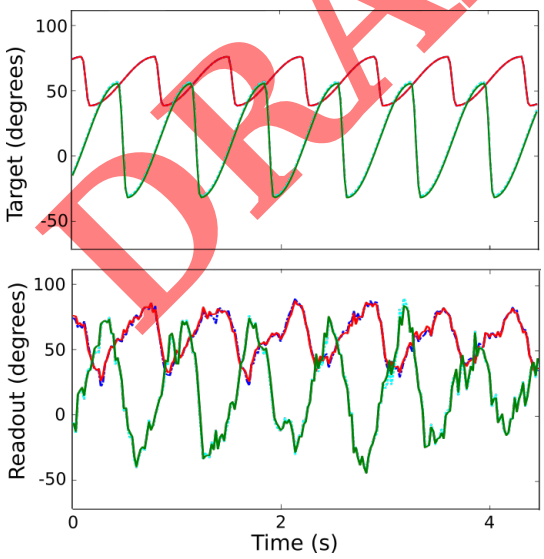

Figure 6: Bounding gait. Top pane: target motor signals (red, green for front, hind legs respectively) Bottom pane: readout signals during closed-loop control, after FORCE learning.

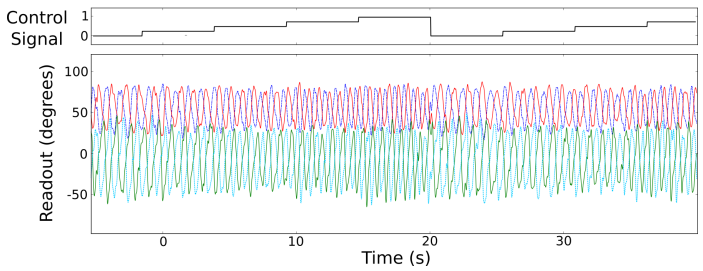

Figure 7: Tunable frequency post learning.

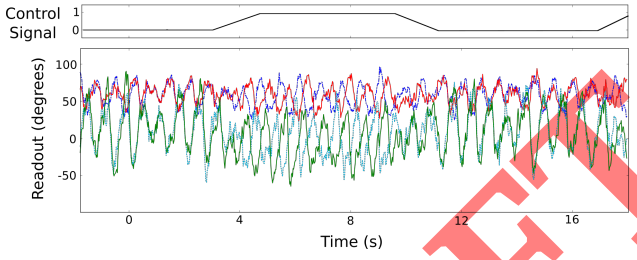

Figure 8: Gait transitioning controlled by external input.

4. CONCLUSIONS

Spiking neural networks could be advantageous for robotics. Potential benefits are energy efficient hardware implementations, efficient sensors and the possibility to apply learning principles observed in biological networks. This work proposes a novel architecture that enables the use of populations of spiking neurons as reservoir units that complement and exploit the physical reservoir that the robot body is. Using only simple learning rules, stable closed loop locomotion control is achieved, even if only minimal sensor data is provided. As in biological spinal networks, the CPG output can be modulated by both simple high level inputs and body sensor input.

In this work, a population of spiking neurons is treated at the same conceptual level as a single artificial neuron, in order to use the same learning paradigm. However, a population of spiking neurons is potentially much more powerful due to the larger number of parameters, both the neuron and population parameters. In future work, the potential of the populations as unit for reservoir computing will be further investigated. Aditionally, an im-

plementation on neuromorphic hardware (SpiNNaker) will allow to run the network in real time on the physical Tigrillo robot.

ACKNOWLEDGMENT

This research was supported by the HBP Neurorobotics Platform funded from the European Unions Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 785907 (Human Brain Project SGA2).

References

Burms, J., Caluwaerts, K., & Dambre, J. (2015). Reward-modulated hebbian plasticity as leverage for partially embodied control in compliant robotics. *Frontiers in neurorobotics*, *9*, 9.

Caluwaerts, K., D'Haene, M., Verstraeten, D., & Schrauwen, B. (2013). Locomotion without a brain: physical reservoir computing in tensegrity structures. *Artificial life*, *19*, 35–66.

Degrave, J., Caluwaerts, K., Dambre, J., & Wyffels, F. (2015). Developing an embodied gait on a compliant quadrupedal robot. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4486–4491). IEEE.

Delcomyn, F. (1980). Neural basis of rhythmic behavior in animals. *Science*, *210*, 492–498.

Drumwright, E., Hsu, J., Koenig, N., & Shell, D. (2010). Extending open dynamics engine for robotics simulation. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots* (pp. 38–50). Springer.

Falotico, E., Vannucci, L., Ambrosano, A., Albanese, U., Ulbrich, S., Vasquez Tieck, J. C., Hinkel, G., Kaiser, J., Peric, I., Denninger, O. et al. (2017). Connecting artificial brains to robots in a comprehensive simulation framework: the neurorobotics platform. *Frontiers in neurorobotics*, *11*, 2.

Füchslin, R. M., Dzyakanchuk, A., Flumini, D., Hauser, H., Hunt, K. J., Luchsinger, R. H., Reller, B., Scheidegger, S., & Walker, R. (2013). Morphological computation and morphological control: steps toward a formal theory and applications. *Artificial Life*, *19*, 9–34.

Furber, S. B., Galluppi, F., Temple, S., & Plana, L. A. (2014). The spinnaker project. *Proceedings of the IEEE*, *102*, 652–665.

Gay, S., Santos-Victor, J., & Ijspeert, A. (2013). Learning robot gait stability using neural networks as sensory feedback function for central pattern generators. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 194–201). Ieee.

Gewaltig, M.-O., & Diesmann, M. (2007). Nest (neural simulation tool). *Scholarpedia*, *2*, 1430.

Hansen, N. (2006). The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation* (pp. 75–102). Springer.

Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, *9*, 159–195.

Hauser, H., Ijspeert, A. J., Füchslin, R. M., Pfeifer, R., & Maass, W. (2011). Towards a theoretical foundation for morphological computation with compliant bodies. *Biological cybernetics*, *105*, 355–370.

Legenstein, R., & Maass, W. (2007). Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, *20*, 323–334.

Lichtsteiner, P., Posch, C., & Delbruck, T. (2008). A 128x128 120 db 15μs latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, *43*, 566–576.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, *14*, 2531–2560.

Nakajima, K., Hauser, H., Kang, R., Guglielmino, E., Caldwell, D. G., & Pfeifer, R. (2013). A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm. *Frontiers in computational neuroscience*, *7*, 91.

Nicola, W., & Clopath, C. (2017). Supervised learning in spiking neural networks with force training. *Nature communications*, *8*, 2208.

Pfeifer, R., Lungarella, M., & Iida, F. (2007). Self-organization, embodiment, and biologically inspired robotics. *science*, *318*, 1088–1093.

Rossignol, S., Saltiel, P., Perreault, M.-C., Drew, T., Pearson, K., & Belanger, M. (1993). Intralimb and interlimb coordination in the cat during real and fictive rhythmic motor programs. In *Seminars in Neuroscience* (pp. 67–75). Elsevier volume 5.

Shik, S. (1966). Orlovskii. control of walking by means of electrical stimulation of the mid-brain. *Biophysics*, *11*, 756–765.

Sussillo, D., & Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron*, *63*, 544–557.

Tsodyks, M., Uziel, A., Markram, H. et al. (2000). Synchrony generation in recurrent networks with frequency-dependent synapses. *J Neurosci*, *20*, 825–835.

Urbain, G., Degrave, J., Carette, B., Dambre, J., & Wyffels, F. (2017). Morphological properties of mass–spring networks for optimal locomotion learning. *Frontiers in neurorobotics*, *11*, 16.

Willems, B., Degrave, J., Dambre, J. et al. (2017). Quadruped robots benefit from compliant leg designs. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017)*. IEEE.

Wyffels, F., & Schrauwen, B. (2009). Design of a central pattern generator using reservoir computing for learning human motion. In *2009 Advanced Technologies for Enhanced Quality of Life* (pp. 118–122). IEEE.