S. Krajči (ed.): ITAT 2018 Proceedings, pp. 116–123
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Rudolf J. Szadkowski, Petr Čížek, and Jan Faigl

# Learning Central Pattern Generator Network with Back-Propagation Algorithm

Rudolf J. Szadkowski[1], Petr Čížek[1], and Jan Faigl[1]

Czech Technical University in Prague, Technicka 2, 16627 Prague, Czech Republic,
szadkrud,petr.cizek,faiglj@fel.cvut.cz

An adaptable central pattern generator (CPG) that directly controls the rhythmic motion of multilegged robot must combine plasticity and sustainable periodicity. This combination requires an algorithm that searches the parametric space of the CPG and yields a non-stationary and non-divergent solution. We model the CPG with the pioneering Matsuoka's neural oscillator which is (mostly) non-divergent and provides constraints ensuring non-stationarity. We embed these constraints into the CPG formulation which we further implemented as a layer of an artificial neural network. This enables the CPG to be learnable by back-propagation algorithm while sustaining the desirable properties. Moreover, the proposed CPG can be integrated into more complex networks and trained under different optimization objectives. In addition to the theoretical properties of the developed system, its flexibility is demonstrated in successful learning of the tripod motion gait with its practical deployment on the real hexapod walking robot.

## 1 Introduction

The movement of legged robots relies on synchronized control of each its joint. Since these joints are part of the same body, the velocity of each joint is dependent on the position of all robot's joints. The problem of generating such synchronized control signals gets harder with increasing number of legs (or the number of joints per leg). A widely used generator of such signals is a system of interconnected Central Pattern Generators (CPGs). The system based on CPGs can be described as two or more coupled oscillators. CPGs appear in many vertebrates and insects where they are responsible for controlling rhythmic motions, such as swimming, walking or respiration [1, 2]. It also appears in biologically inspired robotics, where CPGs are used for locomotion control of legged robots [3].

A CPG network can be modeled as a non-linear dynamic system with coupled variables. Such a non-linear dynamic system is parameterized in the way that it contains a stable limit cycle, but finding such a parametrization is difficult because an analytical description of the high-dimensional non-linear dynamic system is hard or impossible. Moreover, even a small change in the parameters can result in a sudden change of the system's qualitative properties that can range from chaotic to stationary and somewhere between is the desired periodic behavior.

Parameters of the CPG networks can be found experimentally (i.e., tuned manually or automatically by evolutionary algorithms [4]) or they can be heuristically designed. Such design-dependent methods make CPG networks difficult to scale on other robotic bodies or adapt to the locomotion control in different environments. The scaling problem can be partially bypassed by precomputing a trajectory for each foot tip and employing inverse kinematics to determine the control signals for the particular leg's joints [5, 6]. However, the inverse kinematic depends on the robot's body, and identification of the parameters that have to be manually fine-tuned to ensure a proper behavior.

The motivation for the presented approach is to develop a fully automatic CPG learning and this paper explores the possibility of learning a CPG network modeled by Matsuoka's neural oscillators [7] with back-propagation algorithm (BP). To boost the BP algorithm that learns the desired locomotion control for our multi-legged walking robot, we propose two methods pruning the parameter space of the CPG network.

The particular contributions presented in the paper are considered as follows.

- A normalization layer that prunes the parameter space from parametrization with stable stationary solutions.
- An inductive learning method that exploits the structure of robot's body and further reduces the searched parametric space.
- Experimental evaluation of the proposed learning using real hexapod walking robot for which the proposed CPG network learned by the designed algorithm exhibits successful locomotion control following tripod gait, where the developed CPG network directly produces the control signal for each of 18 actuators of the robot.

## 2 Related Work

Different biomimetic approaches including CPGs [1], Recurrent Neural Networks [8] or Self-Adjusting Ring Modules [9] to produce rhythmic patterns have been studied and deployed for locomotion control of robots [3] in recent years. These approaches differ mainly in the complexity of

the underlying model and have different levels of abstraction ranging from biomechanical models [10] simulating membrane potentials and ion flows inside neurons, down to a model of two coupled neurons in a mutual inhibition [11]. Amongst them, the CPGs based on Matsuoka's neural oscillator [7] are being used as the prevalent model. Further details on the Matsuoka's model are in Section 3 as we built on its properties [7, 12, 13] in our work.

Deployment of the CPG oscillators on legged robots is also particularly difficult because of different kinematics and dynamics of each robot. A different amount of post-processing is used to translate the CPG outputs to joint coordinates. Namely, approaches using inverse kinematics [5, 6] suffer from necessary hand fine-tuning of both the parameters of CPG as-well-as kinematics. Besides, existing approaches are using the separate neural network as motor control unit [11] or use CPG outputs directly as joint angles [14]. Furthermore, CPGs can seamlessly switch between different output patterns, thus different gaits [15] which further supports the direct joint control. In our work, we use a dedicated output layer to shape the outputs of CPGs as we assume simple transformations of the output signal are easier to learn by changing parameters of the output layer while the gait change is in charge of the CPG.

Parametrization of the oscillator can be found experimentally, e.g., using evolutionary algorithms with fitness function minimizing energy consumption [11], maximizing the velocity [4], or using parameter optimization [16]. Besides, a modified back-propagation algorithm has been used on an adaptive neural oscillator in [17] to imitate an external periodic signal by its output signal, but it fails to sustain oscillations for complex waveforms. Further works on the parameter constraining of CPGs to maintain stable oscillations have been published [7,12,13,16]; however, to the best of our knowledge we are the first to teach a network of CPGs to perform a locomotion gait of a hexapod walking robot using back-propagation. Furthermore, we propose two methods to prune the space of possible CPG parameters.

## 3 Central Pattern Generator Network

The CPG network used in this paper is based on the Matsuoka's neural oscillator [7]. Matsuoka's neural oscillator is a pair of symmetrically connected adaptive neurons, extensor, and flexor, that imitate the behavior of biological neurons where after peaking, the neuron starts to repolarize until its activation drops to resting potential. Features of Matsuoka's neurons were extensively studied; hence, necessary conditions under which the neural network enters the stable stationary state [7], effects of time-variant tonic input [12], and approximation of oscillator's fundamental frequency and amplitude [13] are well documented in the literature. The description of the particular CPG model used in this work is as follows.

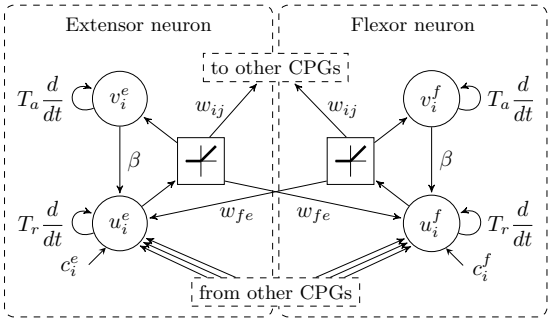



Figure 1: CPG unit connected to the CPG network.

### 3.1 CPG Model

The dynamics of the CPG network with $N$ units can be described by a set of equations

$$T_r \dot{u}_i^e = -u_i^e - w_{fe} g(u_i^f) - \beta v_i^e - \sum_{j=1}^{N} w_{ij} g(u_j^e) + c_i^e, \quad (1)$$

$$T_a \dot{v}_i^e = g(u_i^e) - v_i^e, \quad (2)$$

$$T_r \dot{u}_i^f = -u_i^f - w_{fe} g(u_i^e) - \beta v_i^f - \sum_{j=1}^{N} w_{ij} g(u_j^f) + c_i^f, \quad (3)$$

$$T_a \dot{v}_i^f = g(u_i^f) - v_i^f, \quad (4)$$

where the subscript $i \in N$ denotes the particular CPG and the superscript $\mu \in \{e, f\}$ distinguishes the extensor and flexor neurons, respectively. Each tuple of the variables $u_i^e, v_i^e$ describes the dynamics of the extensor neuron. The variable $u_i^e$ represents activation of the neuron and $v_i^e$ represents its self-inhibitory input, which makes this neuron adaptive. Similarly $u_i^f, v_i^f$ describe the dynamics of the flexor neuron. The function $g$ is a rectifier

$$g(x) = \max(0, x) \quad (5)$$

that is an activation function that adds non-linearity to the system. Each neuron $(i, \mu)$ inhibits itself through the variable $v_i^\mu$ scaled by the parameter $\beta > 0$. The extensor-flexor pair (i.e., the CPG unit) mutually inhibits itself through the symmetric connection with the weight $w_{fe} > 0$. Finally, the CPG units are inter-connected with the symmetric inhibiting connections $w_{ij} \in W$ for $w_{ij} \geq 0$ and $w_{ii} = 0$, where $W$ is a symmetric matrix. The only source of excitation for this CPG network is the tonic input $c_i^e, c_i^f$ $(\geq 0)$ which is given externally. In general, the tonic input may be time-dependent and can be used to regulate the output of the CPG network [12]. $T_r$ and $T_a$ (both $> 0$) are reaction times for their respective variables. The structure of the CPG unit is visualized in Fig. 1.

All the equations (1), (2), (3), and (4) are differentiable except the cases when $u_i^\mu = 0$, since the rectifier is used as the activation function. However, we assume this will not cause any problems because the rectifier is used inside the Rectified Linear Units (ReLU), which are widely used in deep neural networks.

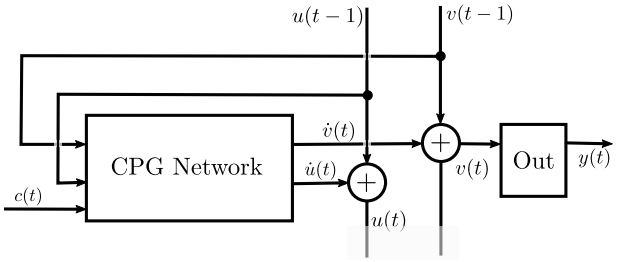
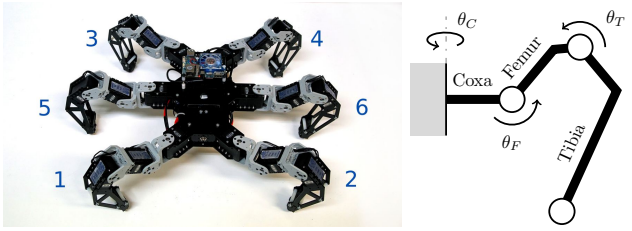



Figure 2: The CPG network connected to the output layer `Out`. Notice the output $y$ is not fed back to the network. Also notice that the self-inhibitory input $u$ is not connected to `Out`.




(a) (b)

Figure 3: (a) Hexapod robot with the numbered legs. (b) Schema of the leg. Each leg consists of three parts – Coxa, Femur, and Tibia.

Note that except tonic inputs $c_i^e, c_i^f$, there are used only inhibiting connections, because such a system is less prone to become chaotic or divergent [13].

### 3.2 Output layer

In this work, we consider the self-inhibitory inputs $v^e, v^f$ as hidden variables, we do not work with them outside of the CPG network. The output layer combines the activation variables $u^e, u^f$ with the affine transformation

$$\mathbf{y} = W_{out}\mathbf{u} + \mathbf{b}_{out}, \tag{6}$$

where $\mathbf{u} = (\mathbf{u}^e, \mathbf{u}^f)$ and $W_{out} \in \mathbb{R}^{N\times 2N}, \mathbf{b}_{out} \in \mathbb{R}^{N\times 1}$ are the learnable parameters. The connection of the CPG network and the output layer is illustrated in Fig. 2.

The main advantage of having $W_{out}$ and $\mathbf{b}_{out}$ as learnable parameters are that the BP algorithm can scale and translate the limit cycle formed by the CPG network. Here, we assume that these transformations are easier to learn by changing the parameters of the output layer than by changing parameters of the CPG network. It is because a change of any parameter of the CPG network can generally cause a non-linear change in the amplitude, frequency, and shift of the generated signals [6]. Another advantage of the proposed output layer is that it can develop complex signals as it can combine outputs from different CPGs.

## 4 Proposed Locomotion Control Learning

In this section, we propose the normalization layer and inductive learning method adapted to learning a CPG network for a hexapod walking robot, see Fig. 3a. Each leg of the robot has three joints called coxa, femur, and tibia (see Fig. 3b) for which an appropriate control signal has to be generated to control the locomotion of the robot. In the total, the robot has 18 controllable joints and depending on the control signals; the robot can move with various motion gaits [18], e.g., tripod, quadruped, wave, and pentapod. During the locomotion, each leg is either in a swing phase to reach a new foothold or in the stance phase in which it supports the body. The motion gait prescribes the order in which the swing and support phases alternate for individual legs; hence, all the legs must work in coordination to simultaneously achieve the desired behavior. The hexapod walking robot is thus used for benchmarking the proposed learning method, where the CPG network has to learn to generate control signals that realize the locomotion control of the robot with the tripod motion gait.

### 4.1 Normalization layer

The proposed normalization layer is based on early experiments with randomly parametrized CPG networks which in most cases ends up oscillating or converges to a static behavior. The static behavior is caused by the stable fixed points that may appear in the corresponding dynamic system. Therefore, we propose to employ a sufficient condition for the CPG network to be free of stable fixed points.

**Condition.** *For a CPG network of $N$ units, if all the values of the tonic input $c_i^\mu$, where $i \in N$ and $\mu \in \{e, f\}$, are from the range $[c_{min}, c_{max}]$ and*

$$w_{fe} < \frac{c_{min}}{c_{max}}(1+\beta) - \max_{i\in N}\left(\sum_j^N w_{ij}\right), \tag{7}$$

$$w_{fe} > 1 + T_r/T_a \tag{8}$$

*then the CPG network has no stable fixed point.*

*Proof.* First, we state adapted theorem from [7].

**Theorem.** *Assume that for some $i$ and $k$ ($i \neq k$)*

$$c_i(1+\beta) - \sum_j^{2N} a_{ij}c_j > 0, \tag{9}$$

$$c_k(1+\beta) - \sum_j^{2N} a_{kj}c_j > 0, \tag{10}$$

$$a_{ik} > 1 + T_r/T_a, \tag{11}$$

*then the CPG network has no stable fixed point. The term $\{a_{ij}\} = A^{(2N,2N)}$ is a matrix of the form*

$$A = \begin{bmatrix} W & w_{fe}I \\ w_{fe}I & W \end{bmatrix} \tag{12}$$

*and* $\mathbf{c} = (c^e, c^f)$*, where I is the identity matrix of the same dimensions as* $W$.

Since the CPGs should act as independent units, it is intuitive that each extensor-flexor neuron pair (a CPG) is able to oscillate on its own. Thus, a weaker form of the theorem is used, where the following conditions must hold for each $i$-th CPG:

$$\frac{c_i^e}{c_i^f}(1+\beta) - \frac{1}{c_i^f}\sum_j^N w_{ij}c_j^e > w_{fe} \tag{13}$$

$$\frac{c_i^f}{c_i^e}(1+\beta) - \frac{1}{c_i^e}\sum_j^N w_{ij}c_j^f > w_{fe} \tag{14}$$

$$w_{fe} > 1 + T_r/T_a. \tag{15}$$

Now, we can focus on the effect of the tonic input $c$. For any parametrization $W, \beta, T_r, T_a, w_{fe}$ we can find a vector $c$ that would break these conditions. Let's relax the problem by clipping the values of $\mathbf{c}$ into the range $[c_{min}, c_{max}]$ where $c_{min} > 0$. Then, it must become independent on the mutable $\mathbf{c}$ vector to simplify the system of conditions. This can be done by substituting $\mathbf{c}$ with such $\mathbf{c}_i^-$ that minimizes the left side expression of (13) or (14) for the $i$-th CPG. W.l.o.g. we consider finding $\mathbf{c}_i^-$ just for (13) as

$$\mathbf{c}_i^- = \underset{\mathbf{c}\in[c_{min},c_{max}]^{2N}}{\operatorname{argmin}} \frac{c_i^e}{c_i^f}(1+\beta) - \frac{1}{c_i^f}\sum_j^N w_{ij}c_j^e. \tag{16}$$

Since all the parameters are positive and $w_{ii} = 0$, the min argument in (16) decreases monotonically with decreasing $c_i^e$ and increasing $c_j^f$ values. Thus, we can substitute these variables with their respective extremes

$$\mathbf{c}_i^- = \begin{cases} c_j^e \in \mathbb{R}^+, j \neq i \\ c_i^e = c_{min} \\ c_j^f = c_{max}, j \neq i \\ c_i^f = c_i' \end{cases} \tag{17}$$

that leaves just $c_i'$ as the variable to minimize

$$F(c) = \frac{c_{min}}{c}(1+\beta) - \frac{c_{max}}{c}\sum_j^N w_{ij}, \tag{18}$$

$$c_i' = \underset{c_i^f \in [c_{min},c_{max}]}{\operatorname{argmin}} F(c_i^f). \tag{19}$$

Notice that now, we are searching a scalar value $c_i'$ that minimizes the given expression.

The equation $\frac{dF(c)}{dc} = 0$ has a solution only if $F$ has such parameters $\beta, W, c_{min}$, and $c_{max}$ that make the function $F$ constant. Since it is unlikely that such a parametrization will emerge during the learning, we consider $F$ does not have any local extremes in the range $[c_{min}, c_{max}]$. Therefore, the minimization (19) can be simplified to

$$c_i' = \operatorname{argmin}\{F(c_{min}), F(c_{max})\}. \tag{20}$$

The condition (13) implies $F > 0$, because $w_{fe}$ must be greater than zero and the following condition must hold too

$$1 + \beta > \frac{c_{max}}{c_{min}}\sum_j^N w_{ij}. \tag{21}$$

Now, we define variable $\varepsilon > 0$ that

$$1 + \beta = \frac{c_{max}}{c_{min}}\sum_j^N w_{ij} + \varepsilon \tag{22}$$

and substitute the right side of (22) into $F(c_{min})$ and $F(c_{max})$

$$F(c_{max}) = \frac{c_{min}}{c_{max}}\varepsilon, \tag{23}$$

$$F(c_{min}) = \varepsilon. \tag{24}$$

Since $\frac{c_{min}}{c_{max}} \in (0, 1]$ and $\varepsilon > 0$, the expression $F(c_{max})$ always minimizes (20). Therefore

$$c_i' = c_{max}. \tag{25}$$

After substituting $c_i'$ into (17) and then $\mathbf{c}_i^-$ into (13) we get

$$\frac{c_{min}}{c_{max}}(1+\beta) - \sum_j^N w_{ij} > w_{fe}. \tag{26}$$

Finally, to make this condition independent on the $i$-th CPG, we can choose such an inequality (26) that has the largest value of the $\sum_j^N w_{ij}$ expression

$$w_{fe} < \frac{c_{min}}{c_{max}}(1+\beta) - \max_{i\in N}\left(\sum_j^N w_{ij}\right). \tag{27}$$

Combining (15) and (27) we get the desired (8) and (7). ■

We integrate the conditions (7) and (8) into the BP framework by redefining the variables $w_{fe}$ and $\beta$ as functions

$$w_{fe}(\hat{w}_{fe}, T_r, T_a) = 1 + T_r/T_a + \exp(\hat{w}_{fe}), \tag{28}$$

$$\beta(\hat{\beta}, w_{fe}, w^*) = (w_{fe} + w^*)\frac{c_{max}}{c_{min}} + \exp(\hat{\beta}) - 1, \tag{29}$$

where $\hat{w}_{fe}, \hat{\beta} \in \mathbb{R}$ are new independent parameters and $w^*$ is defined as

$$w^* = \max_{i\in N}\left(\sum_j^N w_{ij}\right). \tag{30}$$

Then, the max operator is approximated by the differentiable smoothmax defined as

$$\operatorname{softmax}(x) = \frac{\exp(x)}{\sum\exp(x)}, \tag{31}$$

$$\operatorname{smoothmax}(x) = \operatorname{softmax}(x)x. \tag{32}$$

Since all the parameters must be positive, other parameters are defined as exponent of the underlying parameter as

$$\begin{aligned} T_a &= \exp(\hat{T}_a), \\ T_r &= \exp(\hat{T}_r), \\ w_{ij} &= \exp(\hat{w}_{ij}), i \neq j, \end{aligned} \tag{33}$$

where $\hat{T}_a, \hat{T}_r, \hat{w}_{ij} \in \mathbb{R}$. The weights $w_{ij}, i \neq j$ cannot reach zero during learning, but they can approach it.

The BP algorithm learns the proposed new parameters $\hat{T}_a, \hat{T}_r, \hat{w}_{ij}, \hat{w}_{fe}$, and $\hat{\beta}$ that are later normalized by (28), (29), and (33).

### 4.2 Proposed Architecture and Inductive Learning

We propose to divide the CPG network into smaller sub-networks to reduce the search parameter space. These sub-networks are independently learned and then merged into larger sub-networks until a single final network remains. The proposed learning of the CPG network is performed in three phases. First, we learn a single CPG to generate a signal for one joint which gives us the shared parameters $(w_{fe}, T_a, T_r, \beta)$. Then, six triplets of CPGs are learned to generate a control signal for the particular leg. Therefore, for each leg $k \in [1, \ldots, 6]$, we get parameters $W^k$ and $W^k_{out}, \mathbf{b}^k_{out}$. In the final phase, we connect all six CPG sub-networks into one. We choose to connect CPG sub-networks only by coxa-CPGs as it is assumed this is enough for each CPG sub-network to synchronize. Therefore, for the subspace $\mathbf{u}^e = (u^e_{coxa,1}, \ldots, u^e_{coxa,6}, u^e_{femur,1}, \ldots, u^e_{tibia,1})$ (and similarly for $\mathbf{u}^f$), $W \in \mathbb{R}^{18\times 18}$ is organized as follows

$$W = \begin{bmatrix} W_{coxa,coxa} & W_{coxa,femur} & W_{coxa,tibia} \\ W_{femur,coxa} & 0 & W_{femur,tibia} \\ W_{tibia,coxa} & W_{tibia,femur} & 0 \end{bmatrix},$$

where $W_{ij}, i \neq j$ is the matrix of the connections between the $i$-th and $j$-th joints that can be expressed as

$$W_{ij} = \begin{bmatrix} w^1_{ij} & 0 & 0 \\ 0 & \cdots & 0 \\ 0 & 0 & w^6_{ij} \end{bmatrix},$$

where the weights $\{w^k_{ij}\} = W^k$ are taken from the matrices parametrizing the previously learned CPG sub-networks.

For the rearranged vector $\mathbf{u} = (u^e_1, u^f_1, \ldots, u^e_6, u^f_6)$, the term $W_{out} \in \mathbb{R}^{18\times 36}$ is composed of the matrices $W^k_{out}$ of the previously learned CPG network that controls the $k$-th leg

$$W_{out} = \begin{bmatrix} W^1_{out} & 0 & 0 \\ 0 & \cdots & 0 \\ 0 & 0 & W^6_{out} \end{bmatrix}.$$

All the zeroes in the $W$ and $W_{out}$ matrices are unlearnable constants imposing a structure onto the CPG network.

### 4.3 Objective Function

The utilized loss function of the CPG network is defined as a positive distance of the output vector from the desired one

$$\mathscr{L}(\mathbf{y}(t), \mathbf{d}(t)) = \|\mathbf{y}(t) - \mathbf{d}(t)\|, \tag{34}$$

where $\mathbf{d}(t) \in [0,1]^{18}$ is the target signal for each of 18 robot's actuators at the time $t$.

During early evaluation of the proposed learning, we observed that in many cases, the output signal has undesired lower frequency harmonics. This caused the output signal to fit the target signal only for a couple of the first periods. We propose to address this issue by an additional term to the objective function (34)

$$+ \|r - \omega\|, \tag{35}$$

where $r \in \mathbb{R}^+$ is a new hyperparameter and $\omega$ is an approximation of the fundamental frequency of the CPG oscillations that can be expressed as [13]

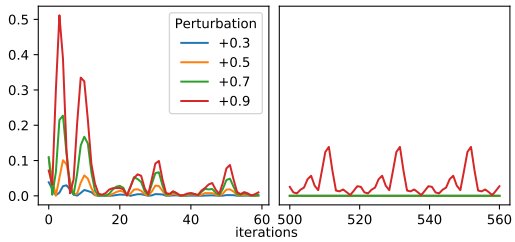
$$\omega = \frac{1}{T_a}\sqrt{\frac{(T_r + T_a)\beta - T_r w_{fe}}{T_r w_{fe}}}. \tag{36}$$

The hyperparameter $r$ should be equal to the fundamental frequency of the desired signal. However, since (36) is just an approximation; it might lead to undesired local minima. Therefore, we propose to switch off the regularization once the term (35) is lesser than a predefined threshold.

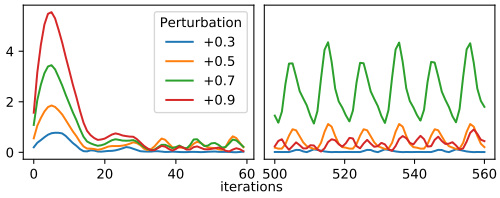
## 5 Experimental evaluation

The proposed learning method has been experimentally verified using `rmsprop` [19] algorithm, which is commonly used to learn recurrent neural networks. Since the following experiments are meant to benchmark and map problems of the CPG network learning, we use a constant tonic input $\mathbf{c} = \mathbf{1}$. Therefore, $c_{min} = c_{max} = 1$. The initial state $(\mathbf{u}^e_{init}, \mathbf{v}^e_{init}, \mathbf{u}^f_{init}, \mathbf{v}^f_{init})$ is set to $\mathbf{u}^e_{init} = 0.1$, $\mathbf{u}^f_{init} = -0.1$, and $\mathbf{v}^f_{init} = \mathbf{v}^f_{init} = 0$. The target signal is formed of eighteen sequences of joint angles that were recorded for a course of five tripod gait cycles. The hexapod robot was driven by a default regular gait based on [20], which is suitable for traversing flat terrains, and it uses the inverse kinematics for following the prescribed triangular leg foot-tip trajectory. This 4.7 seconds long record of all joint signals is sampled to 2350 equidistant data points, and each signal is further normalized in the range $[0,1]$, smoothed using Gaussian convolution to filter out signal peaks, and finally downsampled by the factor of 3.

Preliminary experiments have shown that the process of learning profoundly depends on initial parameters and in some runs, the BP algorithm seems to stuck in local minima from which the learning becomes very slow. This observation is consistent with [17]. The performance of the




(a)




(b)

Figure 4: Squared signal errors of the first leg (a) and body (b) caused by perturbations. The perturbations have been introduced only at the start by adding a constant value to all the trajectory components. For the leg CPG network (a) after 500 iterations, the errors vanished except for +0.9 perturbation. The body CPG network (b) does not recover even after 500 iterations except for +0.3 perturbation.

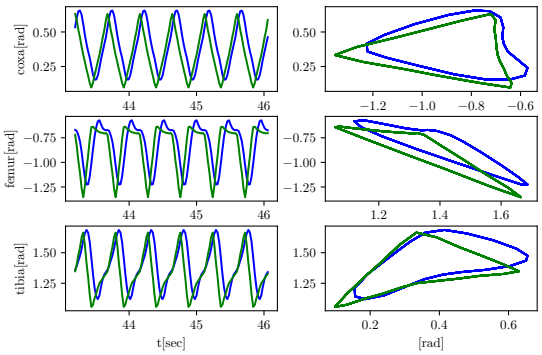



Figure 5: Signals controlling the joints of the first leg. Green ones are desired, and blue ones are generated by the CPG network. The time evolution is on the left, while projected phase-space trajectories are on the right. Each row corresponds to one joint, i.e., coxa, femur, and tibia. In the phase-space column, the variable pairs from the top are coxa-femur, femur-tibia, and tibia-coxa.

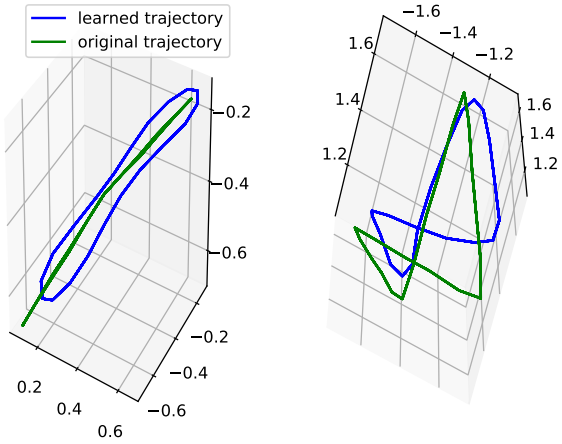



Figure 6: Synchronization of multiple legs. On the left, the coxa-control trajectory for legs 1, 2, and 3 depicted in Fig. 3a. The original trajectory moves almost diagonally in the pictured cube and is "wrapped" by the learned trajectory. On the right, the tibia-control trajectory for the legs 1, 2, and 3.

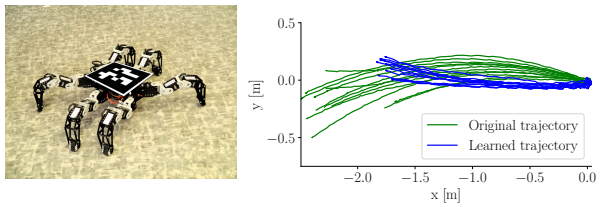
BP algorithm has been improved by adding the regularization term (35). After that, the learning is performed in the three following consecutive steps.

First, each single CPG unit is learned to generate the sinusoid $\sin(t/2)$ that has the same frequency as the fundamental frequency of the desired control signal, which is deterministically set to 3 Hz. The CPG is learned in 2000 epochs, each back-propagating a batch of size 50 data-points. Note that the number of the needed epochs depends on the initial random parametrization.

Next, the parameters of the sinusoid generator is retrained to generate the desired joint control signals. The generator of each joint control is learned with 2000 epochs. We experimented with the stability of the learned limit cycle of the first leg by perturbing it, see Fig. 4a. Finally, the joints CPGs are connected as described in Sec. 4 with non-diagonal values of $W_{coxa,coxa}$ initialized to 0.5, and learned with 4000 epochs. We experimented with the stability of this final CPG network and results are depicted in Fig. 4b.

A comparison of the desired control signal of the first leg and the learned signal is depicted in Fig. 5. The learned signal has a similar shape and the same frequency as the original signal. Binding between different triplets of the legs, the most difficult part is shown in Fig. 6. We can see that the learned trajectory has a similar structure to the desired limit cycles. The trajectory also stays within its limit cycle; the trajectory was generated by six gait-cycles, therefore, traveled the limit cycle multiple times.

We deployed the resultant CPG locomotion controller on the real hexapod (see Fig. 3a) and compared with the original controller [20] in 10 trials. The robot was requested to crawl on flat surface for 10 s and then stop. The velocity of the robot was estimated using an external visual localization system based on tracking of visual marker [21] running with 25 Hz. Moreover, the robot's




(a) (b)

Figure 7: (a) Used experimental setup of the robot with the tracking marker. (b) Visualization of the performed trajectories using the proposed CPG locomotion control and the reference controller.

stability was measured as smoothness of the locomotion using an XSens MTi-30 inertial measurement unit (IMU) attached to the robot trunk. The variances in vertical acceleration ($Acc_z$) and the orientation (pitch and roll angles) of the robot's body are the selected indicators of the locomotion stability.

The recorded robot trajectories visualized in Fig. 7 show that there is a transition effect for our CPG locomotion controller at the beginning of the trajectory where the CPG network starts to oscillate which makes the robot initial acceleration lower; however, the overall locomotion is smoother, as the velocity deviation is smaller.

The quantitative results are listed in Table 1 as average values of the indicators. The results indicate that the performance of the CPG locomotion controller is similar to the implementation [20] based on inverse kinematics (IKT).

Table 1: Experimental results

| | **Unit** | **IKT** [20] | **CPG** (Ours) |
|---|---|---|---|
| Velocity | $[\mathrm{m{\cdot}s^{-1}}]$ | 0.18±0.03 | 0.15±0.01 |
| $Acc_z$ var. | $[\mathrm{m{\cdot}s^{-2}}]$ | 18.31 | 23.03 |
| Pitch var. | $\times 10^{-3}$[rad] | 0.14 | 0.23 |
| Roll var. | $\times 10^{-3}$[rad] | 0.25 | 0.28 |

### 5.1 Lessons Learned and Discussion

During the experimental evaluation of the proposed learning of the CPG network, a couple of good practices how to learn the sinusoid generator came up as follows.

1. It is better to learn the network in batches containing at most two periods.
2. If the CPG network is restarted to the initial state, it is good to ignore the transient states.
3. Since it is not important at which place the system enters the limit cycle, it is suitable to phase-shift the target signal; so, to minimize the distance from the output signal.

Combination of sub-networks into one network has two difficulties. The parameters $(w_{fe}, T_a, T_r, \beta)$ must be the same for the whole CPG network, but the sub-networks are trained independently; so, they can end up with different parameters. In our case, the parameters are similar because all the CPG sub-networks are based on one CPG sub-network. Thus, the BP algorithm is able to adjust them during the learning of the complete network. Another difficulty is the choice of the initial $W_{coxa,coxa}$ weights. The higher the weights are, the stronger is the coupling between the legs. However, if the weight values are too high, the constraint (7) would be violated. Therefore, we used (7) to choose the initial $W_{coxa,coxa}$ weights.

Even though that the robustness is not the objective of the learning algorithm, it is a property of single Matsuoka's oscillator [22]. This property translated well into our 3-unit CPG network (see Fig. 4a) where the network can recover from perturbations. In the real world, robustness helps quickly react to simple temporal events, e.g., servo errors, or feedback from the environment.

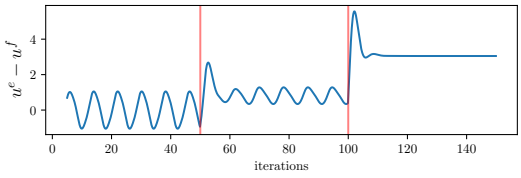
In this work, we chose a simple model with $c_{min} = c_{max} = 1$, i.e., we have a constant tonic input. The time-variant tonic input; however, introduces dynamic changes as we can see in Fig. 8. In the future work, we would like to use the tonic input to control the output of the CPG network dynamically.

## 6 Conclusion

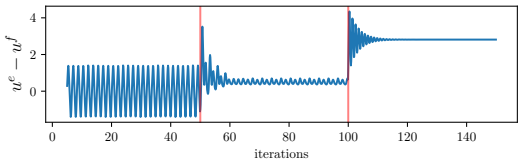
In this paper, we propose a new methodology for learning a CPG network modeled by symmetrically connected neural oscillators. The method is based on a combination of the back-propagation learning algorithm, normalization layer, and regularization term, where the normalization layer prunes the parameters spaces of the CPG network from the undesired non-periodic results, and thus help to speed up the learning process. The advantage of the proposed solution over the previous work on the CPG-based locomotion control is in the scalability of the method that enables to create such a CPG network that can directly control each actuator without the need to employ the inverse kinematics. The proposed method has been successfully deployed in the locomotion control of the real hexapod walking robot.

The main properties of the proposed methodology arise from the idea that the proposed CPG network for the hexapod locomotion control is based on the architecture of the CPG connections that imitates the structure of the robot. The CPG is inductively learned by learning its parts and merging them. Therefore, the proposed method is promising to be easily extendable to other multi-legged robot bodies. Furthermore, since the proposed CPG network is learnable by the back-propagation algorithm, it can be integrated into more complex neural networks supporting back-propagation, which is a subject of our future work.

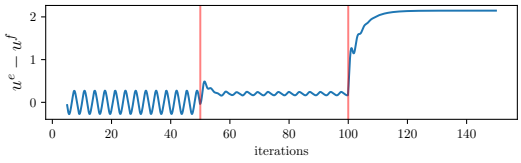
**Acknowledgments** – This work was supported by the




(a)




(b)




(c)

Figure 8: Output of three randomly generated CPG units with $c_{min} = 2; c_{max} = 4$. After each 50 iterations of total 150 iterations, the tonic input is set to $c^e = c^f = 2$; $c^e = 2, c^f = 4$; $c^e = 2, c^f = 8$, respectively. Note that the last setup violates the $c_{max}$ constraint.

Czech Science Foundation (GAČR) under research project No. 18-18858S. The support of the Grant Agency of the CTU in Prague under grant No. SGS16/235/OHK3/3T/13 to Rudolf Szadkowski is also gratefully acknowledged.

## References

[1] E. Marder and D. Bucher, "Central pattern generators and the control of rhythmic movements," *Current Biology*, vol. 11, no. 23, pp. R986–R996, 2001.

[2] E. Marder, D. Bucher, D. J. Schulz, and A. L. Taylor, "Invertebrate central pattern generation moves along," *Current Biology*, vol. 15, no. 17, pp. 685–699, 2005.

[3] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[4] R. D. Beer, H. J. Chiel, and J. C. Gallagher, "Evolution and analysis of model CPGs for walking: II. General principles and individual variability," *Journal of Computational Neuroscience*, vol. 7, no. 2, pp. 119–147, 1999.

[5] H. Yu, H. Gao, L. Ding, M. Li, Z. Deng, and G. Liu, "Gait Generation With Smooth Transition Using CPG-Based Locomotion Control for Hexapod Walking Robot," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 9, pp. 5488–5500, 2016.

[6] G. Zhong, L. Chen, Z. Jiao, J. Li, and H. Deng, "Locomotion control and gait planning of a novel hexapod robot using biomimetic neurons," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 2, pp. 624–636, 2018.

[7] K. Matsuoka, "Sustained oscillations generated by mutually inhibiting neurons with adaptation," *Biological Cybernetics*, vol. 52, no. 6, pp. 367–376, 1985.

[8] B. A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Transactions on Neural networks*, vol. 6, no. 5, pp. 1212–1228, 1995.

[9] M. Hild and F. Pasemann, "Self-Adjusting Ring Modules (SARMs) for Flexible Gait Pattern Generation." in *IEEE International Joint Conference on Artificial Intelligence (IJCAI)*, 2007, pp. 848–852.

[10] J. Hellgren, S. Grillner, and A. Lansner, "Computer simulation of the segmental neural network generating locomotion in lamprey by using populations of network interneurons," *Biological Cybernetics*, vol. 68, no. 1, pp. 1–13, Nov 1992.

[11] S. Steingrube, M. Timme, F. Wörgötter, and P. Manoonpong, "Self-organized adaptation of a simple neural circuit enables complex robot behaviour," *Nature physics*, vol. 6, no. 3, pp. 224–230, 2010.

[12] K. Matsuoka, "Mechanisms of frequency and pattern control in the neural rhythm generators," *Biological Cybernetics*, vol. 56, no. 5, pp. 345–353, 1987.

[13] ——, "Analysis of a neural oscillator," *Biological Cybernetics*, vol. 104, no. 4, pp. 297–304, 2011.

[14] A. J. Ijspeert, A. Crespi, and J.-M. Cabelguen, "Simulation and robotics studies of salamander locomotion," *Neuroinformatics*, vol. 3, no. 3, pp. 171–195, 2005.

[15] W. Chen, G. Ren, J. Zhang, and J. Wang, "Smooth transition between different gaits of a hexapod robot via a central pattern generators algorithm," *Journal of Intelligent & Robotic Systems*, vol. 67, no. 3, pp. 255–270, Sep 2012.

[16] L. Righetti and A. J. Ijspeert, "Design methodologies for central pattern generators: an application to crawling humanoids," in *Robotics: Science and Systems*, 2006, pp. 191–198.

[17] K. Doya and S. Yoshizawa, "Adaptive neural oscillator using continuous-time back-propagation learning," *Neural Networks*, vol. 2, pp. 375–385, 12 1989.

[18] N. Porcino, "Hexapod gait control by a neural network," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 1990, pp. 189–194.

[19] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[20] J. Mrva and J. Faigl, "Tactile sensing with servo drives feedback only for blind hexapod walking robot," in *10th International Workshop on Robot Motion and Control (RoMoCo)*, 2015, pp. 240–245.

[21] E. Olson, "AprilTag: A Robust and Flexible Visual Fiducial System," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3400–3407.

[22] M. M. Williamson, "Robot arm control exploiting natural dynamics," Ph.D. dissertation, Massachusetts Institute of Technology, 1999.