CTU

CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

Faculty of Electrical Engineering
Department of Cybernetics

Bachelor's thesis

Self-Organizing Neural Gait Generator for Multi-Legged Walking Robot

Jan Feber

May 2021

Supervisor: Ing. Rudolf Jakub Szadkowski

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Feber Jan** Personal ID number: **482397**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Self-Organizing Neural Gait Generator for Multi-Legged Walking Robot

Bachelor's thesis title in Czech:

Neurální emergentní generátor vzorů chůze pro vícenohý kráčející robot

Guidelines:

1. Familiarize yourself with legged locomotion and biomimetic Central Pattern Generator (CPG) based controllers and gait pattern generation [1–4].
2. Propose CPG-based generator of gaits with self-organizing neural network dynamics [5].
3. Evaluate the proposed solutions with hexapod walking robot in the CoppeliaSim framework.
4. Possibly deploy the proposed solution on a real hexapod walking robot and experimentally verify the feasibility of generated gaits.

Bibliography / sources:

[1] Dürr V, Schmitz J, Cruse H., Behaviour-based modelling of hexapod locomotion: linking biology and technical application, Arthropod Struct Dev, 2004.
[2] A. Pikovsky, M. Rosenblum, and J. Kurths, Synchronization: A universal concept in nonlinear sciences, ser. Cambridge Nonlinear Science Series. Cambridge University Press, 2001.
[3] S. Aoi, P. Manoonpong, Y. Ambe, F. Matsuno, and F. Wörgötter, Adaptive control strategies for interlimb coordination in legged robots: a review, Frontiers in neurorobotics, vol. 11, p. 39, 2017.
[4] 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.
[5] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations. Cambridge, MA, USA: MIT Press, 1986.

Name and workplace of bachelor's thesis supervisor:

Ing. Rudolf Jakub Szadkowski, Artificial Intelligence Center, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **06.01.2021** Deadline for bachelor thesis submission: **21.05.2021**

Assignment valid until: **30.09.2022**

Ing. Rudolf Jakub Szadkowski	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Declaration

I declare that the presented work was developed independently ant that I have listed all sources of the information used within it in accordance with the methodical instructions for observing the ethiclal principles in the preparation of university theses.

Prague, May 21, 2021

. . . . . .

Jan Feber

Acknowledgement

Writing this thesis would not be possible without the help of the thesis leader Ing. Rudolf Jakub Szadkowski, who introduced me to the thematics of locomotion and bio-inspired controllers. He guided me through the whole process of model proposal and thesis writing and has provided me with a piece of advice no matter what time of day I have asked him.

I also wish to thank the entire team of the *Computational Robotics Laboratory* for creating a supportive environment and organizing meetings where we, the students, were advised through the writing process by research experts.

Thanks should also go to the teachers, who passed on their knowledge on me during the years of my education, without which I could have never written this thesis:

Mgr. Jiří Musil; Mgr. Jana Čermochová; RNDr. Zdeněk Papež and RNDr. Jan Thomas, who provided me cornerstones for mathematics and physics on which I could build during my university studies.

doc. RNDr. Jiří Velebil, Ph.D.; Ing. Matěj Dostál, Ph.D.; doc. Mgr. Petr Habala, Ph.D.; prof. RNDr. Josef Tkadlec, CSc.; prof. Petr Hájek, PhD, DrSc.; RNDr. Miroslav Korbelář, Ph.D.; prof. Ing. Mirko Navara, DrSc.; RNDr. Aleš Němeček; RNDr. Petr Olšák and doc. Ing. Tomáš Kroupa, Ph.D. for introducing me to the advanced mathematics and for showing me how powerful tool the mathematics is.

prof. Ing. Jan Faigl, Ph.D.; Ing. Jan Bayer; Bc. Jindřiška Deckerová; Bc. Jakub Marek; RNDr. Marko Genyk-Berezovskyj and doc. RNDr. Daniel Průša, Ph.D. for teaching me everything I know about programming.

I would like to thank the whole team responsible for the study program *Open Informatics*, provided by the *Faculty of Electrical Engineering* of the *Czech Technical University in Prague*. They do great work for improving the program and maintaining its quality on a high level.

I am deeply indebted to my student colleague Jakub Brož, who helped me so many times during my university studies.

I cannot begin to express my thanks to my parents Ing. Jiří Feber and Ing. Jitka Febrová, who nurtured me, always supported me in my life, and encouraged me to study. I would also like to extend my deepest gratitude to my brother Ing. Jakub Feber for being my role model since my childhood and that he always stood by me, and to his wife Ing. Nina Feber, who supports me, encourages me to make the needed step to the unknown, if it is necessary, and who recommended me to study the *Open Informatics*. I am also grateful to my uncle RNDr. Vladimír Hanák who has always been open to mathematics-physics-philosophy-related discussions, which the other family members have not found amusing nor worth discussing.

I would like to express my deepest appreciation to my best friends Šimon and Markéta Sajnerovi and Jiří Macek, who have never let me down, no matter how difficult their situation was, and who have always trusted in me, supported me, and who I could have always trusted.

I am incredibly grateful to prof. MUDr. Jan Janda, CSc., who cured me when I was little. He treaded me prudently, chose the right therapy, and rescued me from the danger of wrong body development.

Last but not least, I would like to express my deepest gratefulness to our Lord and Savior Jesus Christ for His affection and patience, for His sacrifice, and for standing by me, even though I have not always appreciated it.

Abstrakt

Vzory chůze popisují periodicky se opakující kráčivý pohyb vícenohého robotu určením fáze pohybu jednotlivých nohou. Aby mohl robot autonomně vykonávat úkoly ve špatně přístupném měnícím se prostředí, je nutné proces lokomoce automatizovat. Během lokomoce probíhá v neurálním systému mnoho komplexních procesů, jejichž některé principy jsou popsány díky probíhajícímu výzkumu lokomoce vícenohých organismů. Některé z těchto principů, jako například Centrální Generátory Vzorů (CGV) a pravidla určující vzájemnou koordinaci nohou, jsou v této práci využity. CGV je neurální oscilátor, který v živých organismech produkuje rytmus pro lokomoci. Koordinační pravidla určují, jak jsou pohyby nohou mezi sebou v rámci fáze koordinovány. Řídící systémy navržené pro řízení lokomoce často vyžadují proces manuálního zadávání velkého množství hyperparametrů určujících konkrétní vzor chůze, což je proces, který se tato práce snaží automatizovat. V této práci jsou představeny dvě metody, které se různým způsobem vypořádávají s neznámým vztahem mezi fází CGV a pohybovými akcemi nohou. První z metod využívá aproximace vztahu mezi vzdáleností stavů CGV ve stavovém prostoru a jejich vzájemným fázovým posunem. Druhá metoda odhaduje neznámou fázi CGV a hledá vztah mezi fází CGV a jeho stavy. Obě metody úspěšně generují všechny tři požadované vzory chůze, což je demonstrováno simulacemi šestinohého kráčejícího robotu v simulátoru *CoppeliaSim*.

Klíčová slova: šestinohý kráčející robot, lokomoce, centrální generátor vzorů, vzor chůze, strojové učení, biologicky inspirovaný

Abstract

The gait patterns describe periodically repeating motion of a legged robot by determining a phase of its legs' movement. If a robot on a long-term mission in an inaccessible unknown dynamic environment should function autonomously, it is crucial to automatize the locomotion process. The ongoing research of legged organisms' locomotion describes some principles of complex neural system processes, such as Central Pattern Generators (CPGs) and inter-leg coordination rules used in this thesis. The CPG is a neural oscillator producing rhythm for locomotion in living organisms. The coordination rules determine how legs' actions are coordinated within the CPG's phase. Many locomotion controllers require a process of hand-setting many gait-pattern-determining hyperparameters, which this thesis aims to automatize. Two different methods are proposed in this work, dealing with the unknown relation between the CPG's phase and the legs' actions. The first method uses an approximation of a relation between a distance of CPG's states in its state space and the phase offset of the CPG's states. The second method estimates CPG's unknown phase and finds the phase's relation to CPG's states. Both methods successfully generate all three desired gait patterns, which is demonstrated by running simulations on a hexapod walking robot in the *CoppeliaSim* simulator.

Keywords: hexapod walking robot, locomotion, central pattern generator, gait pattern, machine learning, bio-inspired

Contents

List of Figures

List of Tables

symbol	meaning
$a = b$	a equals b
$a \coloneqq b$	a is defined as b
$\mathbb{R}^{\text{dim(cpg)}}$	space in which the CPG exists ($\mathbb{R}^4$ in this work)
i, j	weight's index (for most of the work represents relevant leg's index; $i, j = 1, 2, \ldots, 6$)
t	time
T	CPG's period
$\boldsymbol{y}$	CPG's state ($\boldsymbol{y} \in \mathbb{R}^{\text{dim(cpg)}}$)
$\boldsymbol{y}(t)$	CPG's state in moment t ($\boldsymbol{y}(t) \in \mathbb{R}^{\text{dim(cpg)}}$)
l	the limit cycle trajectory (closed curve) consisting of states $\boldsymbol{y}(t)$ ($l \subset \mathbb{R}^{\text{dim(cpg)}}$)
$\boldsymbol{w}_i$	weight relevant to the i-th leg ($\boldsymbol{w}_i \in \mathbb{R}^{\text{dim(cpg)}}$)
$\boldsymbol{w}_{\text{sig}}$	weight attracted by the CPG's state to produce learning signal ($\boldsymbol{w}_{\text{sig}} \in \mathbb{R}^{\text{dim(cpg)}}$)
m_i	weight determining the phase of its relevant $\boldsymbol{w}_i$ weight ($m_i \in [0, 2\pi)$)
p_i^{rbf}	RBF neuron's signal activating the swing of the i-th leg invoked by the weight $\boldsymbol{w}_i$
$\phi^{\text{cpg}}(t)$	CPG's phase
$\hat{\phi}^{\text{cpg}}(t)$	estimated CPG's phase
ϕ_i	required phase value corresponding to the i-th leg (phase of the i-th leg's activation)
$\phi_{i,j}$	phase offset of the weights $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ (i.e., $\phi_{i,j} = \lvert\phi_i - \phi_j\rvert$)
$\Delta\phi$	consecutive legs' phase offset determining the gait pattern (the methods' input value)
$\phi(t)$	phase (used only in example)

Table 1: Table of used common symbols

symbol	meaning
$\lVert\boldsymbol{v}\rVert_2$	Euclidean norm of the vector $\boldsymbol{v}$
$\lVert\boldsymbol{v}\rVert_{\text{cpg}}$	special norm of the vector $\boldsymbol{v}$
$\boldsymbol{d}_i^j$	difference between weights $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ ($\boldsymbol{d}_i^j \in \mathbb{R}^{\text{dim(cpg)}}$)
$\boldsymbol{d}_i^{\text{cpg}}$	difference between weight $\boldsymbol{w}_i$ and the CPG's state $\boldsymbol{y}$ ($\boldsymbol{d}_i^{\text{cpg}} \in \mathbb{R}^{\text{dim(cpg)}}$)
$\boldsymbol{d}_{\text{sig}}$	difference between weight $\boldsymbol{w}_{\text{sig}}$ and the CPG's state $\boldsymbol{y}$ ($\boldsymbol{d}_i^{\text{cpg}} \in \mathbb{R}^{\text{dim(cpg)}}$)
$\boldsymbol{r}_i^j$	the "*force*" of influencing the weight $\boldsymbol{w}_i$ by the weight $\boldsymbol{w}_j$ ($\boldsymbol{r}_i^j \in \mathbb{R}^{\text{dim(cpg)}}$)
$\boldsymbol{r}_i^{\text{cpg}}$	the "*force*" of influencing the weight $\boldsymbol{w}_i$ by the CPG's state $\boldsymbol{y}$ ($\boldsymbol{w}_i^{\text{cpg}} \in \mathbb{R}^{\text{dim(cpg)}}$)
c_i^{ε}	coeficient regulating the ε_i dynamics
c_i	coeficient regulating the influences $\boldsymbol{d}_i^j$ and $\boldsymbol{d}_i^{\text{cpg}}$ on the $\boldsymbol{w}_i$ weight's dynamics
ε_i	the radius of the weight's $\boldsymbol{w}_i$ dynamic vicinity

Table 2: Table of used symbols specific for the SNM

symbol	meaning
q	index of the leg's side ($q \in \{0, 1\}$, where 0 means left and 1 means right)
k	index of the leg's anatomic position ($k \in \{\text{front}, \text{middle}, \text{hind}\}$)
$d^q_{(\text{front,middle})}$	signed distance between weights m^q_{front} and m^q_{middle}
$d^q_{(\text{hind,middle})}$	signed distance between weights m^q_{hind} and m^q_{middle}
d^q_k	signed distance from the m^q_k to the $m^{q-1)}_k$
$\boldsymbol{r}^{\text{cpg}}_{\text{sig}}$	the "*force*" of influencing the weight $\boldsymbol{w}_{\text{sig}}$ by the CPG's state $\boldsymbol{y}$ ($\boldsymbol{r}^{\text{cpg}}_{\text{sig}} \in \mathbb{R}^{\dim(\text{cpg})}$)
$c^q_{(\text{front,middle})}$	coeficient regulating the $d^q_{(\text{front,middle})}$ influence on m^q_{front} weight's dynamics
$c^q_{(\text{hind,middle})}$	coeficient regulating the $d^q_{(\text{hind,middle})}$ influence on m^q_{hind} weight's dynamics
c^q_k	coeficient regulating the d^q_k influence on m^q_k weight's dynamics
c_{sig}	coeficient regulating the ε dynamics
ε	the radius of the weight's $\boldsymbol{w}_{\text{sig}}$ dynamic vicinity
$\text{s}(t)$	signal generated by RBF neuron relevant to the weight $\boldsymbol{w}_{\text{sig}}$
$\hat{\text{s}}(t)$	modification of the signal $\text{s}(t)$ to meet the requirements of the phase learning process
$\text{p}(t)$	CPG's phase coeficient estimate
a	the value determining the slope of the $\text{p}(t)$ (learns the correct value)
p^m_i	RBF signal given by the phase estimation $\hat{\phi}^{\text{cpg}}(t)$ and the weight m_i

Table 3: Table of used symbols specific for the PLM

Chapter 1
Introduction

The legged robots' movement (i.e., locomotion) is a complex problem due to many degrees of freedom, increasing with the number of legs. It requieres many hyperparameters to be set correctly. Every time the conditions change (change of the terrain, malfunction of some joint, etc.), new hyperparamters' values has to be found to preserve the stable locomotion. As the robot can be on a long term mission in an inaccessible environment without any connection to the outer world, it is important for the robot to be able to self-learn the necessary parameters.

To propose the learning mechanisms, the researchers inspire in nature, where such problems are already solved and tested by millions of years of use. Many different biological models clarifying the locomotion were already proposed and compared (for example, the comparison of stick insect and cockroach models in [1]). Although the models are not complete, some principles are already being successfully used in robotics.

One of the easiest concepts to notice, while observing locomotion, is the presence of gaits. Gaits are periodically repetitive motion patterns. In other words, there can be observed a certain rhythm in the locomotion. Each leg performs two actions during locomotion, which are altering. It moves forward performing swing, or it lies on the ground performing stance. For each leg, the swing and stance repeat periodically.

To periodically repeating actions, a phase is assigned, determining the actions' course. As the phase progresses, the relevant actions repeat with the period. Therefore, each action corresponds to a phase. Consequently, there is a phase difference between each two locomotion actions.

For humans, the phase difference between the legs during walking is half of the period. The legs are altering each other. Therefore each leg performs stance for half of the period, and the other half of the period it performs swing. For four and more legged organisms, there are many more possible combinations. For instance, the four-legged organisms can move the front left leg and the right hind leg simultaneously, altering with the front right and left hind legs, or each leg can move individually, or the front legs can move together altering with the hind legs. The swing and stance length and the phase offset of the legs changes depending on the coordination within the legs.

Different ways of the legs' coordination are suitable for different situations. The organisms switch the way of coordinating the limbs (i.e., switch the gait pattern) to move efficiently at differing speeds or on various terrains, for instance. For quadrupeds, the three most known gait patterns are walk, trot and gallop. This thesis focusses primarily on hexapods (i.e., six-legged robots), for which the common gaits are tripod gait, transition gait, and wave gait.

The tripod gait can be seen very often in the insects' locomotion. It consists of altering the activity of two triplets of legs. One triplet includes the left hind leg, the right middle leg and the left front leg, and the other includes the right hind leg, the left middle leg and the right front leg. The legs in the triplet undergo the same actions (swing and stance) at the same time. If the legs of the first triplet are performing stance, than the legs from the second triplet are performing swing, and it works alike the other way around. All the gaits are described in detail in Chapter 2 Problem Statement.

One key aspect is used to determine which gait pattern is the robot using. During one gait-cycle, each leg performs only one swing, so the leg swings in specific phase. The difference between the phase of two legs is a phase offset. Two neighbouring legs, which undergo the swing phase one after the other, are consecutive legs (for instance, the left hind leg and the left middle leg). The phase offset of two consecutive legs is constant for each given gait pattern (Note that the time duration between

swings of consecutive legs, determined by their phase offset, also corresponds to the duration of the legs' swing for the particular gait pattern). Therefore, as explained in [2], the gait pattern is determined by the phase offset of action of two consecutive legs.

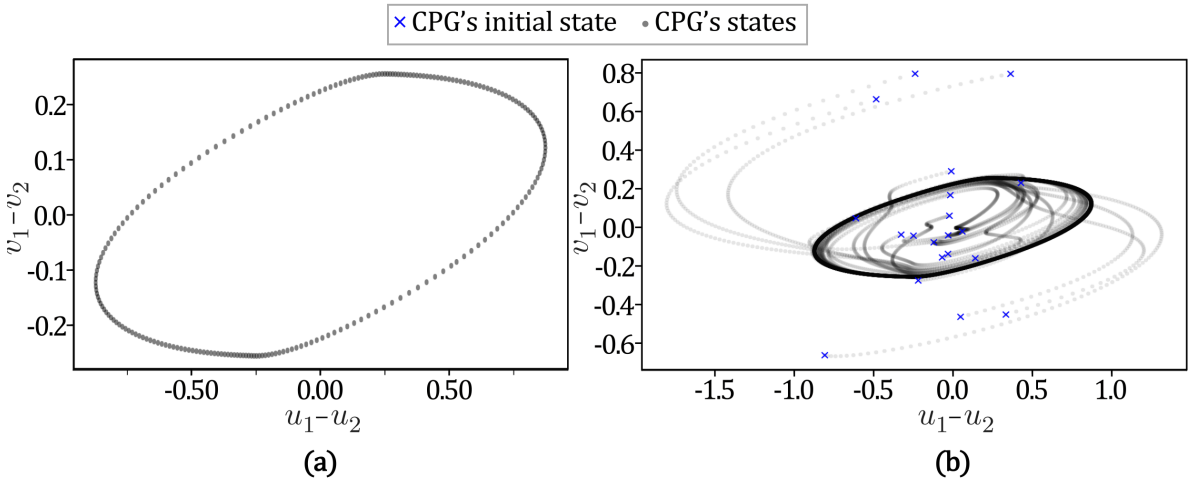

Figure 1: Matsuoka's neural oscillator shown in 2D projection. In (a), the course of one period can be seen. Each dot represents one iteration of Euler's method with step size $= 0.01$. The density of the dots signifies how much time is spent in each area. In (b), the limit cycle feature is shown. There are 20 random initialization points (blue crosses). Each of them ends on the limit cycle after a few iterations. The dotted path shows the trajectory of each of the initializations.

To determine the phase, another bio-inspired concept, called a Central Pattern Generator (CPG), is often used in combination with the gait pattern concept. CPGs were found in both vertebrates and invertebrates. It is a neural network behaving as an oscillator generating a rhythmical signal without any rhythmical input. It is believed the CPGs do function as a rhythmical underlie for locomotion in living organisms.

A review of open research topics, including CPGs, is presented in paper [3]. Many different models were created to describe CPGs. Some of the CPG models are described in review [4] together with the main reasons for using CPGs in robotics. Three of the reasons, the most important of them in this work's context, are described in following paragraphs.

The first one is rhythmicity. While observing any living organism during the walk, a rhythm can be seen in the movement. If no irregularities are in the way, then all movements repeat periodically, creating a gait pattern. Therefore, the CPG's function as a rhythmical underlie for the gait patterns in robotics, as they do for the living organisms.

The second reason is the stability and adaptability of the CPG. A well-known sinus function itself can generate rhythmical movement. However, rhythm itself is not enough. Robots are being constructed to work on uneven terrains, which means the robots must handle perturbations. The sinus function does not provide any mechanism to do so. As mentioned above, the CPGs do not need any outside input to generate a rhythmical signal. However, they can accept input from the sensors. The input influences the course of the cycle and, therefore, the movement which depends on it. Moreover, the CPGs dispose of the ability of stability due to the limit cycle feature. The limit cycle is a closed trajectory to which each state of the oscillator converges (see Fig. 1 (b)). It means that if the CPG is perturbed due to its input, it tends to restore its previous trajectory.

Finally, the third reason is the CPGs' ability to synchronize. The CPG's input can be not only from sensors but also from other CPGs. The mutual connection of the CPGs then leads to mutual binding and synchronization of the connected CPGs, which is used in many proposed locomotion controllers.

Many approaches and different architectures have been proposed on the topic of learning the hexa-

pod to walk with the use of CPGs. Few examples follow. [5] uses a different architecture of CPGs' interconnection for every gait pattern. In [6] authors propose architecture with one CPG for each leg (all the CPGs are connected using a one-directional ring hierarchy) and map the actions of the leg's joints directly on the CPG signal. Architecture in [2] uses CPG with independent frequency and amplitude control, to control the gait dynamic more easily, and focuses on the mapping of the action of particular joints on the CPG signal. Authors in [7] study also mostly the mapping of the leg actions to the CPG signal. However, in this study, they modify the signal to suit better different gaits (as for different gaits different swing and stance duration is suitable). Architecture proposed in [8] goes even further and proposes a method for avoiding obstacles or changing the gait. [9] learns gait using reinforcement learning and aims to fine-tune the parameters to gain efficient locomotion. In [10], the authors aim the parameter tunning more than the architecture design.

All the mentioned examples use some matrix of weights or matrix of connections or any other predefined tuple of numbers to determine a particular gait pattern. That means a change of many parameters for different gait. Therefore, if a new gait is required, then it is necessary to create a new matrix or tuple of parameters by hand. That is limiting for a robot, which is meant to be fully autonomous. However, [11] presents a set of rules defining the relations between hexapod's legs, which hold for any hexapod gait.

This work aims to use the defined rules and create a self-organized learning mechanism to coordinate the legs to produce a gait pattern based on a given phase offset of two consecutive legs. Two methods are proposed in this work. The first method takes advantage of the similarities between the phase offset and norm of the difference of two points on the limit cycle in the multidimensional space. The second method focuses on the problem decomposition and estimates the relation of the CPG's phase and the CPG's states.

To prove both methods working, simulations in *CoppeliaSim* simulator were run, where both methods successfully generated each of the three given gait patterns (transition, tripod and wave). For both methods, the plots, demonstrating the results of each of the proposed mechanisms (which the methods are composed of), are provided. Finally, the comparison of both proposed methods is presented.

The remainder of the thesis is organized as follows. The definition of the CPG, together with the mechanism of weights (determining the gait), coordination rules, and the gait patterns definition is presented in chapter 2. Both the methods are proposed in chapter 3, where the first method is introduced in section 3.1, and the second method is introduced in section 3.2. The resulting plots and results of the proposed mechanisms successfully generating the given gait patterns are shown in chapter 4. The methods are discussed and compared in chapter 5. Finally, the conclusion is presented in chapter 6.

Chapter 2
Problem Statement

This work aims to automatize a process of determining which gait pattern should be learned, which most of the other multi-legged gait controllers require to be done manually before the locomotion learning process begins. As the robot's locomotion in this work is implemented by producing a gait pattern, a CPG is used to determine the rhythm for the gait. This chapter firstly introduces the CPG used in this work and the role of the weights in the process of locomotion, and secondly, it describes the gait patterns which this work aims to learn using coordination rules observed from animal behaviour.

2.1 CPG and Weights

In this work, a Matsuoka's neural oscillator [12] is used as a CPG model. The Matsuoka's neural oscillator phase is hard to express analytically, and therefore it is unknown at the beginning of the learning process. This fact forces the method to be more generalized and not to rely only on oscillators with a known period. The Matsuoka's neural oscillator is given by the following differential equations:

$$\tau \dot{v}_1^{\text{cpg}} = \text{h}(u_1^{\text{cpg}}) - v_1^{\text{cpg}}, \tag{1}$$
$$\tau \dot{v}_2^{\text{cpg}} = \text{h}(u_2^{\text{cpg}}) - v_2^{\text{cpg}}, \tag{2}$$
$$\gamma \dot{u}_1^{\text{cpg}} = -u_1^{\text{cpg}} - \text{h}(u_2^{\text{cpg}})\alpha - v_1^{\text{cpg}}\beta + 1, \tag{3}$$
$$\gamma \dot{u}_2^{\text{cpg}} = -u_2^{\text{cpg}} - \text{h}(u_1^{\text{cpg}})\alpha - v_2^{\text{cpg}}\beta + 1, \tag{4}$$
$$\text{h}(x) \coloneqq \max(x, 0), \tag{5}$$

where the CPG hyperparameters are set to $\tau = 0.5; \gamma = 0.25; \alpha = \beta = 2.5$. The Matsuoka's oscillator is visualised in Fig. 1. As described by the equations (1) - (5), the CPG's state is determined by four variables:

$$\boldsymbol{y} \coloneqq (u_1^{\text{cpg}}, u_2^{\text{cpg}}, v_1^{\text{cpg}}, v_2^{\text{cpg}}) \in \mathbb{R}^4, \tag{6}$$

where $\boldsymbol{y}$ stands for the CPG's state. Therefore, the weights $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n)$ are described also by four variables:

$$\boldsymbol{w}_i \coloneqq (u_1^i, u_2^i, v_1^i, v_2^i) \in \mathbb{R}^4, \tag{7}$$

for $i = 1, 2, \ldots, n$, where n is the number of weights.

The weights are then used by Radial Basis Function (RBF) neurons to determine when to fire. RBF neurons' signals can be used as a trigger for a specific movement (as used in this work to run the simulation), or as a synchronization signal for other CPGs, for instance. The RBF neuron's signal is given by equations:

$$\delta_i \coloneqq \|\boldsymbol{y} - \boldsymbol{w}_i\|_2^2, \tag{8}$$

$$p_i^{\text{rbf}} \coloneqq \exp(-\epsilon \cdot \delta_i), \tag{9}$$

where p_i^{rbf} is the RBF neuron's current pulse value, and ϵ is a positive constant.

The gait learning process varies for different architectures of locomotion controllers. Approach used in this work uses the weights $\boldsymbol{w}_i$ corresponding to some important movement actions. In this work, the i-th weight signalizes the start of the i-th leg's swing followed by the stance.

As the CPG's state changes, it activates the weights' respective RBF neurons, which trigger the relevant movement actions. If the movement actions are triggered in the correct order, then the mechaism

produces the given gait pattern. Hence, based on a model in [13], to learn a gait pattern means to spread the weights, relevant to the important movement actions, around the CPG's limit cycle correctly.

Let the limit cycle (the closed trajectory) be called $l \subset \mathbb{R}^4$. Although there is no time t variable in equations (1) - (5), there is a time dependence because the equations are differential equations with respect to time t. Hence the notation $\boldsymbol{y}(t)$ is used to express the CPG's states dependence on the time ($\boldsymbol{y}$ is a specific state, $\boldsymbol{y}(t)$ denotes CPG's states in general by their dependency on time).

If no perturbations occur, then the CPG's states $\boldsymbol{y}(t)$ repeat periodically with an unknown period T (i.e., $\boldsymbol{y}(t) = \boldsymbol{y}(t+T)$). Despite the fact, that one of the reasons to use CPGs for locomotion is the preturbaition handeling, this work consideres the learning process with no perturbations. Hence the states $\boldsymbol{y}(t)$ are considered as periodically repeating.

Since $\boldsymbol{y}(t)$ is periodic, its phase $\phi^{\text{cpg}}(t) \in [0, 2\pi)$ can be defined. The phase repeats periodically (i.e., $\phi^{\text{cpg}}(t) = \phi^{\text{cpg}}(t+T)$). Nevertheless, the period T and the CPG's phase course $\phi^{\text{cpg}}(t)$ are both unknown.

If every i-th leg is related to a specific weight $\boldsymbol{w}_i$ then assigning the weight a phase value ϕ_i means assigning the respective leg a state of phase in which the leg movement should be triggered. In other words, if the i-th leg has been assigned phase ϕ_i, then the goal is to find the moment t_i of the i-th leg action triggering, for which holds the equality $\phi^{\text{cpg}}(t_i) = \phi_i$.

The concept of CPG and weights was introduced in this section. The following section presents the rules for organizing the weights within themselves and then the relation between the weights' organization within the phase and the gait patterns is described.

2.2 Gait Patterns and Coordination Rules

The $\boldsymbol{w}_i$ weight determines the beginning of the i-th leg's swing movement. To produce a stable gait pattern for hexapod walking robot, the weights have to be organized within the phase like shown in Figs. 2 and 3.

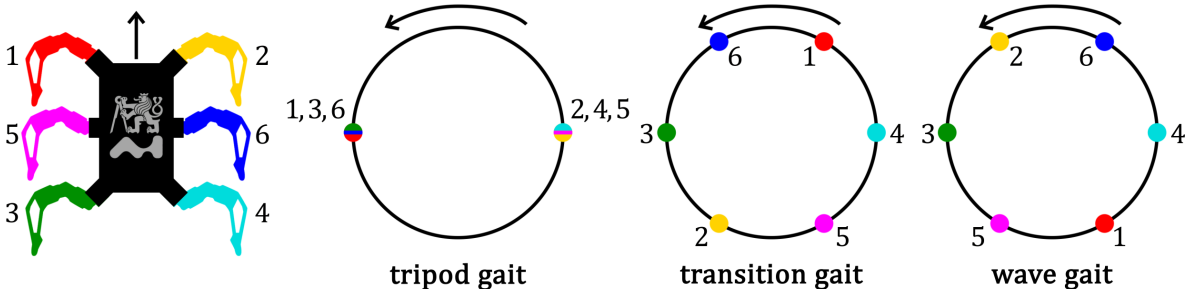

Figure 2: On the left, a scheme of a hexapod robot is visualized. The arrow indicates the direction of the forward walk. The colors of the legs correspond to the colors of the respective weights in all plots in this work. The three cycles with colorful dots on them represent one period of each gait pattern. The colors of the dots correspond to the colors of the legs of the robot on the left. The arrow indicates the direction of the phase growth (means that, for example, in the wave gait, the second leg is activated before third leg, and so on). The dot consists of more colors if the respective weights (visualized as colorful dots) are overlapping.

As mentioned in [2], the gait pattern is clearly defined by a phase offset $\Delta\phi$ between two consecutive legs. The phase offset values and corresponding gaits are given by Tab. 4. Therefore, the phase offset value is the input value determining the gait pattern, which the methods, introduced in this work, should produce.

To organize the weights to produce the given gait pattern, the rules introduced in [11], are used. The rules give insight into how the weights are related based on the relevant legs' anatomic position.

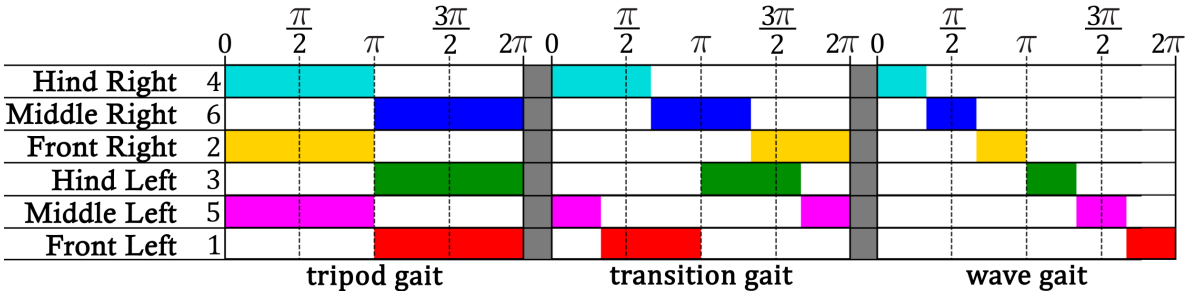

Figure 3: The color bars correspond to the swing phase of the corresponding leg movement. The colors themselves correspond to the colors of the respective legs from the robot scheme in Fig. 2. For each gait, the table visualizes when the leg's swing occurs during the period of the corresponding gait pattern. The leg performs a stance during the rest of the period.

phase offset	gait
$\pi/3$	wave gait
$2\pi/3$	transition gait
π	tripod gait

Table 4: Phase offset of consecutive legs for corresponding gait pattern

The rules describe the distribution loading, leg placing and interleg coordination. For purposes of this work, only the three following coordination rules are important:

1. while a leg is lifted-off, suppress the lift-off of the consecutive leg
2. if the leg touched the ground, initiate the lift-off of the consecutive leg
3. do not lift-off the contralateral legs (e.g., both front legs) at the same time

The first two rules are fulfilled by keeping the given phase offset of consecutive legs. The third rule implies additional interaction between contralateral legs. The interaction between weights corresponding to the legs can be of three types. The weight (i) is not influenced , or (ii) is repulsed by some other weight, or (iii) is attracted by some other weight.

The aim is to organize the weights to be near the limit cycle (near enough so their respective RBF neurons would produce signals with distinct peaks), and the organization of the weights within the limit cycle has to produce one of the given gait patterns. To do so, the mentioned coordination rules should be used to determine attractive and repulsive forces between the weights. The produced gait pattern should correspond to the given input value of the phase offset. In other words, the method searches for $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_6)$, that for each $i = 1, 2, \ldots 6$ the following inequality holds:

$$\text{dist}(\boldsymbol{w}_i, \boldsymbol{y}(t_i)) \leq \hat{\varepsilon}, \tag{10}$$

$$\phi^{\text{cpg}}(t_i) = \phi_i \tag{11}$$

where t_i is each time moment for which holds the equality Eq. 11; $\hat{\varepsilon}$ is a small constant which ensures, that the produced RBF signal pulses (Eq. 9) are distinct. The Eq. 10 represents that the weight is close enough to the CPG's state $\boldsymbol{y}(t_i)$ ($\boldsymbol{y}(t_i) = \boldsymbol{w}_i$ is the ideal state) to produce an RBF signal with distinct pulses. To produce a stable gait pattern a following equality has to be kept:

$$(\phi_i - \phi_j) \mod 2\pi \approx (\phi^{\text{cpg}}(t_i) - \phi^{\text{cpg}}(t_j)) \mod 2\pi, \tag{12}$$

i (leg number)	ϕ_i (tripod gait)	ϕ_i (transition gait)	ϕ_i (wave gait)
4	0	0	0
6	π	$2\pi/3$	$\pi/3$
2	0	$4\pi/3$	$2\pi/3$
3	π	π	π
5	0	$5\pi/3$	$4\pi/3$
1	π	$\pi/3$	$5\pi/3$

Table 5: The table shows the phase values of legs organized within the phase to produce the given gait pattern. The leg numbers are ordered top to down based on the order of the legs' activity during the wave gait. Note that the difference between the legs 2 and 3 does not correspond to the given phase offset in the transition gait as it does for the two other gaits. The reason is that it would conflict with rule 3. mentioned above.

where $i, j = 1, 2, \ldots 6$; ϕ_i is a constant phase value of the relevant leg action depending on the given gait pattern as shown in Tab. 5. It does not matter in which state of the phase the gait pattern starts. The most important is the legs' organization within the period (i.e., relative phase dependencies), not the phase values themselves.

This section introduced importnat two important concepts for this work. Firstly, the coordination rules, which determine the interactions within the weights in the methods proposed in this work. Secondly, the gaits description with use of phase was introduced (i.e., phase assigned to individual legs' actions and the phase offset between the legs).

2.3 Summary

In this chapter, the initial state of the problem and the goal were presented. At first, a particular CPG was introduced, then the weights $\boldsymbol{w}_i \in \mathbb{R}^{\dim(\text{cpg})}$ with their respective RBF neurons, and the relation between the CPG's states and its phase were presented. Then the weights' role in generating the stable locomotion was introduced, in the context of this work. The gait patterns are given by the phase offset of two consecutive legs, which is also used as an input for the proposed methods to determine which gait pattern should the methods learn. Important coordination rules were presented, determining the relation between the robot's legs, which imply the interaction rules for the weights corresponding to the legs.

In the following chapter, two different methods for solving the problem are proposed. The first considers only the weights $\boldsymbol{w}_i$, their interactions and their position towards the CPG's limit cycle. The second method decomposes the problem, focuses on the CPG's phase course and learns the phase's dependency on the time, which is then used to determine the weights' positions on the CPG's limit cycle.

Both methods build on the relation between the CPG's phase $\phi^{\text{cpg}}(t)$ and its corresponding states $\boldsymbol{y}(t)$ (points of the CPG's limit cycle). The relation of the CPG's states and the CPG's phase is used to organize the weights $\boldsymbol{w}_i$ around the CPG's limit cycle, while respecting the given phase offset $\Delta\phi$, to produce the given gait patterns. However, both methods have to solve the problem, that the CPG's limit cycle's shape l (and consequently the CPG's states $\boldsymbol{y}(t)$) and the CPG's phase course $\phi^{\text{cpg}}(t)$ are both unknown.

Chapter 3
Proposed Method

The method has to find the place on the CPG's limit cycle for all $\boldsymbol{w}_i$ weights, which have to maintain the given phase offsets $\Delta\phi$ between them. However, both, the limit cycle's shape l (given by states $\boldsymbol{y}(t)$) and the CPG's phase $\phi^{\text{cpg}}(t)$, are unknown. This section describes in detail two methods developed to solve the given problem.

The first method in this chapter is tackling the problem in one space $\mathbb{R}^{\dim(\text{cpg})}$. It uses the relation between the phase offset and the norm of the $\boldsymbol{w}_i$ weights' differences (i.e., the distance of the weights). Hence the section describing the first method is called 3.1 *Method Based on Norm Properties*. The method expects the limit cycle's shape being close to the surface of some ellipsoid.

The second introduced method decomposes the problem and approaches it with a solution consisting of more systems connected. It aims to estimate the CPG's phase and find the CPG's state corresponding to each phase value. Parallelly, it organizes the weights within the phase, to be then mapped to the CPG's states corresponding to the weights' phases. The section describing the second method is called 3.2 *Method Based on the Phase Learning*.

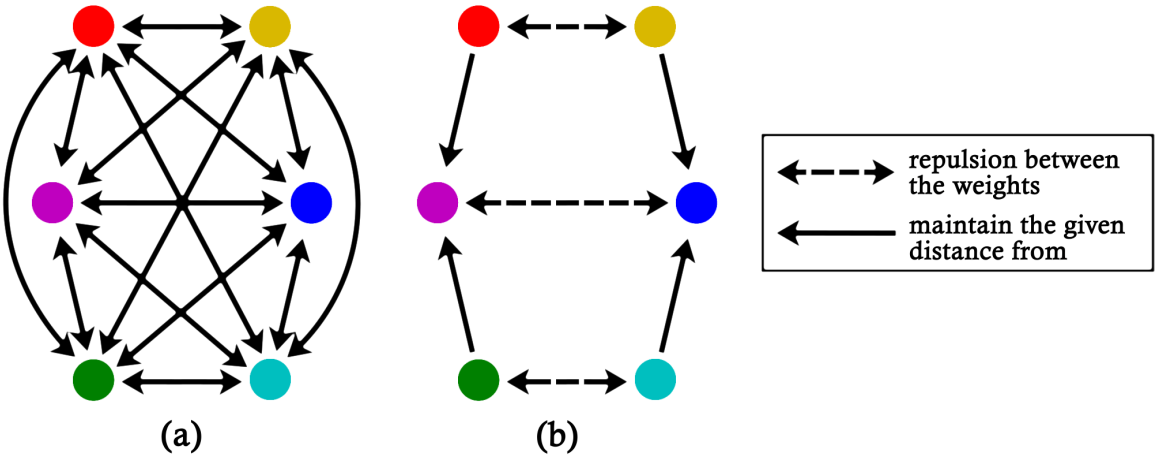

Figure 4: The scheme visualizes the dependencies within the weights for both proposed methods. In (a), the scheme for the method using a special norm is shown. Each weight maintains the given distance from every other weight. Hence, the number of dependencies for 6 weights is $6(6-1) = 30$. In (b), the scheme for the method using phase learning is shown. The number of dependencies is much lower (equal to 10), which resolves in lower computational complexity. Note that extending the model for more legs, where would be an equal count of legs on both sides of the robot, the complexity difference becomes even more significant. For n legs in total, the number of dependencies of the model from (a) is equal to $n(n-1) = n^2 - n$, and for (b) the number of dependencies is equal to $n + 2(\frac{n}{2} - 1) = 2n - 2$. Nevertheless, the mentioned extension is only theoretical and not a subject of this work. Note that the colors of the dots match the colors of their respective legs from Fig. 2.

3.1 Method Based on Norm Properties

The method proposes dynamics for weights $\boldsymbol{w}_i$ by using two kinds of forces influencing the weights. The first kind of force is determined by a repulsive and attractive interactions between the weights (introduced in section 3.1.2) based on the norm of their differences (relation between norm and phase is introduced in section 3.1.1). The second kind of force is the weights' $\boldsymbol{w}_i$ attraction by the CPG's state $\boldsymbol{y}$ to keep the weights close to the CPG's limit cycle (mechanism described in section 3.1.3). The weights interaction forces are then combined with the attraction by the CPG's state to determine the weights' dynamics.

The first kind of force is dependent on the similarity between the phase offset of two weights and their distance. The CPGs limit cycles can differ in shape, dimensionality. The method consideres the limit cycle trajectory lying on (or nearby) the surface of an ellipsoid. Hence, the special norm, which transforms the ellipsoid into a unit sphere and measures the norm of all states $\boldsymbol{y}(t)$ approximately as one (i.e., $\|\boldsymbol{y}(t)\|_{\text{cpg}} \approx 1$, where $\|-\|_{\text{cpg}}$ is a notation of the special norm) is introduced to enable to use the similarity between the weights' phase offset and their distance.

The second kind of force forces the weights to stick nearby the CPG's limit cycle. The learning process is considered without any outer perturbations influencing the CPG. Therefore, the CPG's state $\boldsymbol{y}$ is always considered to be on the CPG's limit cycle. Hence, the force attracts the weights to the CPG's state $\boldsymbol{y}$ (as described in section 3.1.3) as the CPG's state $\boldsymbol{y}$ is always representing a point of the CPG's limit cycle.

The two forces are then combined to produce dynamics for the weights $\boldsymbol{w}_i$ to be close enough to the limit cycle and to be correctly organized within the themselves (introduced in section 3.1.4) to produce the gait pattern given by the phase offset of consective legs $\Delta\phi$.

3.1.1 Phase Offset and Norm Relation

To understand the relation between the phase and norm, the section is introduced with a simple example. Without loss of generality, imagine having a simple system, which behaves as a single CPG without any perturbations, which is given by equations:

$$x \coloneqq \cos(\phi(t)), \tag{13}$$

$$y \coloneqq \sin(\phi(t)), \tag{14}$$

where $\phi(t) \in [0, 2\pi)$ is the phase dependent on time. Therefore values of x and y repeat periodically with period 2π. The CPG's limit cycle, given by equations Eqs. 13 and 14, is visualized as a unit circle in Fig. 5 (a), where the angle between two circle's points by the center of the circle corresponds to the phase offset between the points.

There is a relation between the distance of the points on the unit circle and their phase offset. Consider two weights $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^2$ on the unit circle and their phase offset $\phi_{1,2} \in [0, \pi]$. As the weights $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are on the unit circle, they are both equally distant from the center $\boldsymbol{c}$ of the unit circle. Hence, the center $\boldsymbol{c}$ of the unit circle and the weights $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ form an isosceles triangle. Moreover, the angle by the center $\boldsymbol{c}$ is known, as it is equal to the weights' phase offset $\phi_{1,2}$. Hence the distance between the weights can be computed not only using the of the norm of their difference:

$$\text{dist}(\boldsymbol{w}_1, \boldsymbol{w}_2) = \|\boldsymbol{w}_1 - \boldsymbol{w}_2\|_2 , \tag{15}$$

but also using the weights' phase offset:

$$\text{dist}(\boldsymbol{w}_1, \boldsymbol{w}_2) = \text{d}(\phi_{1,2}), \tag{16}$$

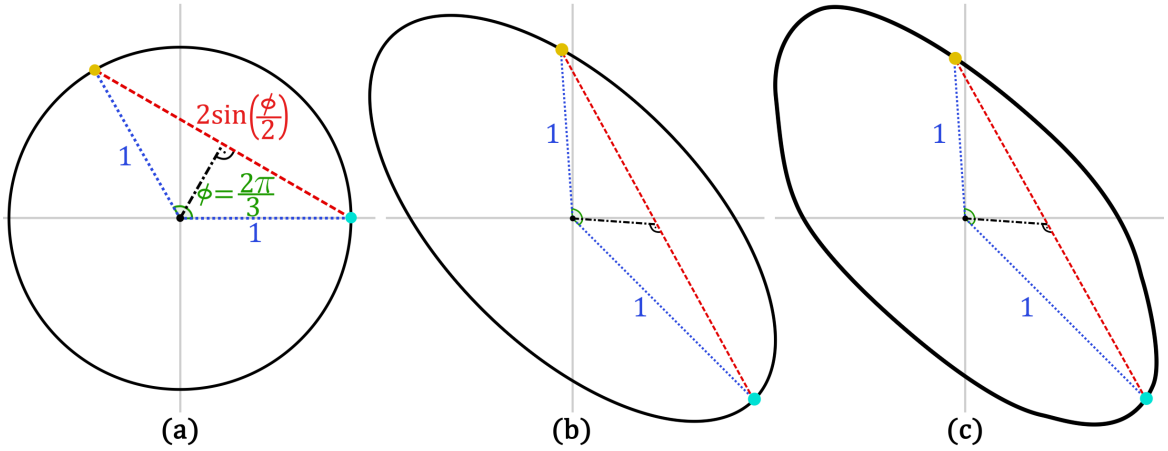

Figure 5: There are three different limit-cycles in the figure. (a) a circle, (b) an ellipse, (c) a complex shape similar to the ellipse. By using different metrics, a different shape is considered as a unit sphere. For the unit sphere in Euclidean metrics (using the norm $\|-\|_2$), relations between the weights are defined by the same rules as for the weights on a unit circle. The figure demonstrates how it would work for two weights which are on coordinates $\boldsymbol{w}_1 = (1, 0)$ (cyan) and $\boldsymbol{w}_2 = (-\frac{1}{2}, \frac{\sqrt{3}}{2})$ (yellow) in the (a). The phase offset between them corresponds to the angle $\phi = \frac{2\pi}{3}$. The distances between the weights and the center of the CPG (dotted blue) are equal to 1 and, together with the line joining the weights (dashed red), form an isosceles triangle. The distance between the weights can then be computed as the length of the base of the triangle. Because the length of the sides and the angle between them is known, the base length can be computed. We split the triangle by its height (the line from the center of the CPG to the middle of the red dashed line) and make a right triangle. Because the angle by the center of the CPG in the right triangle is half of the ϕ, and the side adjacent to the center of the CPG is equal to 1, the half of the base of the isosceles triangle can be computed as $\sin(\frac{\phi}{2})$. Therefore the distance between the weights $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ equals to $2\sin(\frac{\phi}{2}) = \|\boldsymbol{w}_1 - \boldsymbol{w}_2\|_2$. Using a special norm $\|-\|_{\text{cpg}}$in (b), the blue lines have both norm equal to one and the norm of the weights' distance is equal to $2\sin(\frac{\phi}{2}) = \|\boldsymbol{w}_1 - \boldsymbol{w}_2\|_{\text{cpg}}$, where ϕ is the phase offset of the weight. If the special norm from (b) is used in (c), then the relation between distance and phase offset in (c) is approximately the same as in (b). Therefore the special norm from (b) is used to approximate the relation in (c).

$$\mathrm{d}(b) := 2\sin\left(\frac{\min(b, 2\pi - b)}{2}\right). \tag{17}$$

The phase-distance relation (given by Eq. 16) is visualized in Fig. 5 (a). If the weight's phase offset $\phi_{1,2}$ is given, then the phase-distance relation implies the weights' distance, i.e., the weights should be placed on the unit circle in a way that satisfies the equality $\mathrm{dist}(\boldsymbol{w}_1, \boldsymbol{w}_2) = \mathrm{d}(\phi_{1,2})$, where $\phi_{1,2}$ is a fixed constant giving the desired phase offset of the weights $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$.

The example introduced the relation between a phase offset and norm of difference of two points on the unit circle.[1] As the method's input value is a phase offset $\Delta\phi$ of consecutive legs, the phase offset between each pair of the legs' corresponding weights can be computed. In this work the phase value for each particular leg is assigned at first, then the phase offsets of the assigned values are computed to represent the phase offsets between the weights corresponding to the legs.

The weights corresponding to the hind legs are assigned a value of zero (beginning of the phase), as the activity of the legs goes from hind legs to front legs. Each consecutive leg's respective phase value is increased by the given phase offset. For example lets assume the transition gait phase offset $\Delta\phi = 2\pi/3$, then the $\boldsymbol{w}_i$ weights are assigned values $\phi_4 = 0$, $\phi_6 = 2\pi/3$, $\phi_2 = 4\pi/3$, $\phi_3 = 0$, $\phi_5 = 2\pi/3$ and $\phi_1 = 4\pi/3$, where ϕ_i corresponds to the $\boldsymbol{w}_i$ weight.

However, this assignment of the values conflicts with the third coordination rule (i.e., do not lift-off two contralateral legs at the same time) because the weights corresponding to contralateral legs have the same phase value assigned. Therefore, the value of π is added (with mod 2π operation applied) to all values of respective weights corresponding to the robot's left side legs. In case of the transition gait the final values are $\phi_4 = 0$, $\phi_6 = 2\pi/3$, $\phi_2 = 4\pi/3$, $\phi_3 = \pi$, $\phi_5 = 5\pi/3$ and $\phi_1 = \pi/3$. The phase values determine the phase offset of each pair of the legs. Therefore, using the phase-distance relation (introduced in Eq. 16), a matrix $\boldsymbol{M}$ of distances between the weights is given, as:

$$\begin{aligned}
\boldsymbol{M} &= \begin{pmatrix} \mathrm{dist}(\boldsymbol{w}_1, \boldsymbol{w}_1) & \mathrm{dist}(\boldsymbol{w}_1, \boldsymbol{w}_2) & \cdots & \mathrm{dist}(\boldsymbol{w}_1, \boldsymbol{w}_6) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{dist}(\boldsymbol{w}_6, \boldsymbol{w}_1) & \mathrm{dist}(\boldsymbol{w}_6, \boldsymbol{w}_2) & \cdots & \mathrm{dist}(\boldsymbol{w}_6, \boldsymbol{w}_6) \end{pmatrix} = \\
&= \begin{pmatrix} \|\boldsymbol{w}_1 - \boldsymbol{w}_1\|_2 & \|\boldsymbol{w}_1 - \boldsymbol{w}_2\|_2 & \cdots & \|\boldsymbol{w}_1 - \boldsymbol{w}_6\|_2 \\ \vdots & \vdots & \ddots & \vdots \\ \|\boldsymbol{w}_6 - \boldsymbol{w}_1\|_2 & \|\boldsymbol{w}_6 - \boldsymbol{w}_2\|_2 & \cdots & \|\boldsymbol{w}_6 - \boldsymbol{w}_6\|_2 \end{pmatrix} = \\
&= \begin{pmatrix} \mathrm{d}(\phi_{1,1}) & \mathrm{d}(\phi_{1,2}) & \cdots & \mathrm{d}(\phi_{1,6}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{d}(\phi_{6,1}) & \mathrm{d}(\phi_{6,2}) & \cdots & \mathrm{d}(\phi_{6,6}) \end{pmatrix},
\end{aligned}$$

where $\phi_{i,j} = |\phi_i - \phi_j|$ is a phase offset (i.e., phase difference) of the weights $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$.

The unit circle, which acts as the CPG's limit cycle given by Eq. 13 and 14 in the given example, is called a unit circle because each point of it given as (x, y) has an Euclidean norm equal to 1 ($\|(x, y)\|_2 = 1$). Nevertheless, if a different norm is used, a different shape is considered a unit circle (or unit sphere for more dimensions).

If the CPG's limit cycle has the shape of an ellipse, a different than the Euclidean norm can be used to measure the norm of the limit cycle points to preserve the phase-distance relation (the distance is invoked by the used norm). The ellipse, depicted in Fig. 5 (b), is an example of what can be considered as a unit circle if a different norm is used. As the choice of norm depends on the CPG's limit cycle's shape, let the special norm[2] of vector $\boldsymbol{v}$ for a specific CPG be denoted as $\|\boldsymbol{v}\|_{\mathrm{cpg}}$.

[1] The unit circle represents the limit cycle in the given example.

[2] The norm can be computed using various approaches. Approach used in this work is described in the Appendix A Special Norm Finder

If the CPG's limit cycle has the shape of the ellipse, a special norm, which measures all the ellipse's points as equal to one, have to be used to preserve the phase-distance relation. Therefore, the distances between the weights are still determined by the phase offsets between the weights. Only the norm measure changes:

$$\boldsymbol{M} = \begin{pmatrix} \|\boldsymbol{w}_1 - \boldsymbol{w}_1\|_{\text{cpg}} & \|\boldsymbol{w}_1 - \boldsymbol{w}_2\|_{\text{cpg}} & \cdots & \|\boldsymbol{w}_1 - \boldsymbol{w}_6\|_{\text{cpg}} \\ \vdots & \vdots & \ddots & \vdots \\ \|\boldsymbol{w}_6 - \boldsymbol{w}_1\|_{\text{cpg}} & \|\boldsymbol{w}_6 - \boldsymbol{w}_2\|_{\text{cpg}} & \cdots & \|\boldsymbol{w}_6 - \boldsymbol{w}_6\|_{\text{cpg}} \end{pmatrix} = \\ = \begin{pmatrix} \text{d}(\phi_{1,1}) & \text{d}(\phi_{1,2}) & \cdots & \text{d}(\phi_{1,6}) \\ \vdots & \vdots & \ddots & \vdots \\ \text{d}(\phi_{6,1}) & \text{d}(\phi_{6,2}) & \cdots & \text{d}(\phi_{6,6}) \end{pmatrix}.$$

Unfortunately, the limit cycles do not have a shape of a perfect ellipsoid nor the phases are proportional to the angles. However, the shape of the limit cycle lies approximately on the ellipsoid's surface.[3]

The limit cycle shown in Fig. 5 (c) has a shape similar to the ellipse. Therefore the ellipse from 5 (b) can be used as an approximation of the limit cycle from 5 (c). Hence, the relation between the norm (which consideres the ellipse in 5 (b) as a unit circle) and phase offset is also used as an approximation for the CPG's limit cycle's shape, which is close to the surface of an ellipsoid (i.e., for each $\boldsymbol{l} \in l$: $\|\boldsymbol{l}\|_{\text{cpg}} \approx 1$, where $l \in \mathbb{R}^{\dim(\text{cpg})}$ is the CPG's limit cycle trajectory).

The special norm $\|-\|_{\text{cpg}}$, which is used to approximate the relation between the distance and the phase offset of the CPG's limit cycle's points, was introduced in this section. The following section uses the special norm to determine the attractive and repulsive forces between the weights.

3.1.2 Repulsion and Attraction Within the Weights

Based on the previous section, to ensure the correct organization of the weights, the following equality has to be adhered to:

$$\text{d}(\phi_{i,j}) = \|\boldsymbol{w}_i - \boldsymbol{w}_j\|_{\text{cpg}}, \tag{18}$$

for all $i, j = 1, 2, \ldots, 6$. The $\phi_{i,j}$ is a constant for each pair of i and j, what makes the equality's left hand side value constant. Therefore, the only way to gain equal values on both sides of the equation Eq. 16 is to move the weights to adjust the equality's right hand side value. The equality implies three interaction rules for the weights:

- if $\text{d}(\phi_{i,j}) > \|\boldsymbol{w}_i - \boldsymbol{w}_j\|_{\text{cpg}}$, then the weights are too close to each other and should be mutually repulsive,
- if $\text{d}(\phi_{i,j}) < \|\boldsymbol{w}_i - \boldsymbol{w}_j\|_{\text{cpg}}$, then the weights are too far from each other and should be mutually attractive,
- if $\text{d}(\phi_{i,j}) = \|\boldsymbol{w}_i - \boldsymbol{w}_j\|_{\text{cpg}}$, then the weights are at a required distance from each other and should not interact.

The given interaction rules for a pair of weights are visualized in Fig. 6. A scheme, visualizing the relations within the weights, is shown in Fig. 4 (a). To fulfill the interaction rules the following equations are proposed:

$$\boldsymbol{d}_i^j \coloneqq \boldsymbol{w}_j - \boldsymbol{w}_i, \tag{19}$$

[3]The method expects the shape of the limit cycle lying approximately on the ellipsoid's surface, as mentioned on the introduction of this chapter.

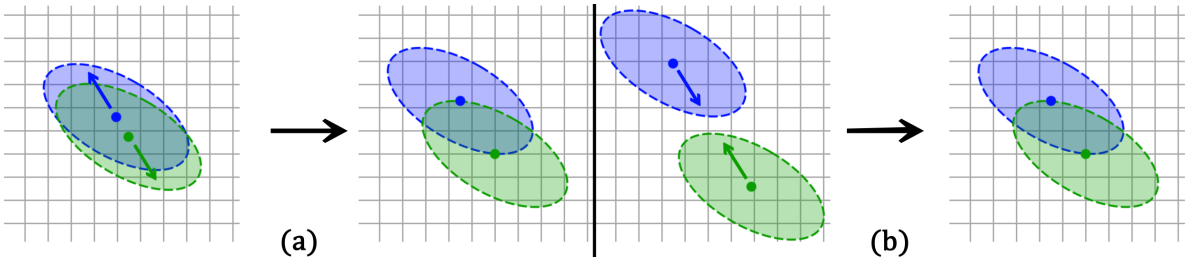

Figure 6: Illustration of maintaining the given distance of two weights. The blue and green dots represent the weights. The shaded vicinity, bordered by the dashed ellipse, represents the area in which the norm value of the weights' difference is lower than required. The dashed ellipse (the curve itself) represents the points, which have the required distance from the respective weight. (a) The weights are too close to each other. Therefore the repulsive forces take them away. Each weight is forced to the direction opposite to the direction where the other weight is. (b) The weights are too far from each other. Hence the attraction forces push them against each other.

$$\boldsymbol{r}_i^j := \frac{\text{clip}_{-2}^{2}\left(\left\|\boldsymbol{d}_i^j\right\|_{\text{cpg}} - \text{d}(\phi_{i,j})\right)}{n}\boldsymbol{d}_i^j, \tag{20}$$

$$\text{clip}_y^z(x) := \begin{cases} y & x \leq y \\ z & x \geq z \\ x & \text{otherwise,} \end{cases} \tag{21}$$

$$\text{where } y < z,$$

where $\boldsymbol{r}_j^i$ is a force vector of the weight $\boldsymbol{w}_j$ forcing the weight $\boldsymbol{w}_i$ and n is the number of weights ($n = 6$).

This section defined dynamic rules determining the forces within the weights $\boldsymbol{w}_i$. Those rules were proposed to hold the coordination rules from Chapter 2 Problem Statement. Next section focuses on placing the weights $\boldsymbol{w}_i$ close enough to the CPG's limit cycle.

3.1.3 Attraction of the Weights to the Limit Cycle

In the previous section, the attraction and repulsion within the weights were defined. However, for the RBF neurons to produce the disinct signals, the weights must be placed close enough to the limit cycle. Therefore, in this section the mechanism of the weights' attraction to the CPG's state $\boldsymbol{y}$ is described.

The CPG's limit cycle shape, denoted in this work as a closed trajectory $l \subset \mathbb{R}^{\dim(\text{cpg})}$, is not known. Nevertheless, the CPG's state $\boldsymbol{y}$ from Eq. 6 is a point on the CPG's limit cycle, which changes with time. Therefore, if the weights are attracted by the $\boldsymbol{y}$, then the weights get closer to the CPG's limit cycle. The l is unknown, but the $\boldsymbol{y}$ is an element of the l. Hence, to attract the weights $\boldsymbol{w}_i$ to the CPG's limit cycle l, the attraction by the CPG's state $\boldsymbol{y}$ is used.

However, the $\boldsymbol{y}$ is moving around the the CPG's limit cycle, as it changes with time, and the weights would be moving around the CPG's limit cycle too, if the weights would be attracted by the $\boldsymbol{y}$ all the time. As each weight is meant to stay on one place of the CPG's limit cycle, the $\boldsymbol{w}_i$ weight's attraction by the CPG's state $\boldsymbol{y}$ is defined only within the boundaries of the weight's dynamic vicinity, as defined by the following equations:

$$\boldsymbol{d}_i^{\text{cpg}} := \boldsymbol{y} - \boldsymbol{w}_i, \tag{22}$$

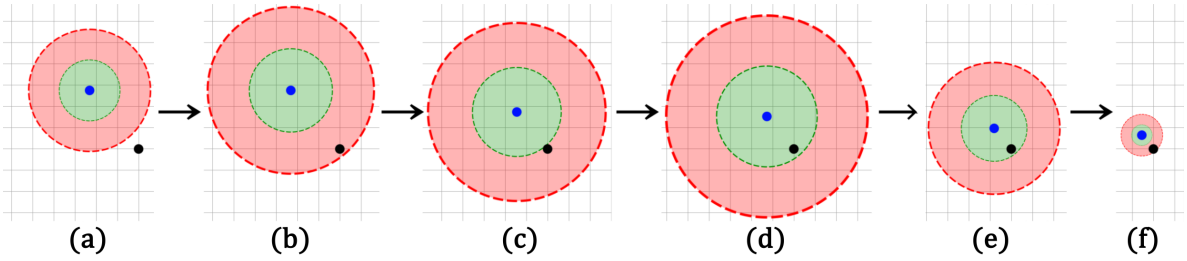

Figure 7: The figure visualizes the functionality of the weight $\boldsymbol{w}_i$ (blue dot) being attracted by the CPG's current state $\boldsymbol{y}$ (the $\boldsymbol{y}$ is moving in the dynamic system, but it is a static black dot for simplicity of the visualization) based on the weight's dynamic vicinity (dashed filled red circle) determined by the weight's radius ε_i. If the $\boldsymbol{y}$ is outside of the weight's vicinity like in (a), the vicinity grows as shown in (b). The weight $\boldsymbol{w}_i$ is then attracted by the $\boldsymbol{y}$ and the vicinity still grows, as depicted in (c). If the vicinity is relatively large (i.e., the weight is close to the $\boldsymbol{y}$), then the $\boldsymbol{y}$ reaches the distance shorter than half of the ε_i (the dashed filled green circle), like in (d). The vicinity then starts to shrink, while the weight $\boldsymbol{w}_i$ is still being attracted by the $\boldsymbol{y}$, as shown in (e). If the $\boldsymbol{y}$ is still in the weight's vicinity, it attracts the weight $\boldsymbol{w}_i$, as depicted in (f), and the vicinity changes as described in previous steps. However, in the dynamic system, the $\boldsymbol{y}$ moves, so it leaves the vicinity entirely, if the ε_i is small enough. The weight $\boldsymbol{w}_i$ then stops moving.

$$\boldsymbol{r}_i^{\text{cpg}} := \text{f}\left(\frac{\left\|\boldsymbol{d}_i^{\text{cpg}}\right\|_2}{\varepsilon_i}\right)\boldsymbol{d}_i^{\text{cpg}}, \tag{23}$$

$$\text{f}(x) := \begin{cases} -x^2+1 & x \leq 1 \\ 0 & \text{otherwise,} \end{cases} \tag{24}$$

where $\boldsymbol{r}_i^{\text{cpg}}$ is a force vector of the $\boldsymbol{w}_i$ weight's attraction by the CPG's state $\boldsymbol{y}$; function $\text{f}(x)$ ensures, that the weight $\boldsymbol{w}_i$ is not attracted by the $\boldsymbol{y}$ if the $\boldsymbol{y}$ is outside of the vicinity of $\boldsymbol{w}_i$ determined by ε_i (ε_i is a radius of the vicinity). If the $\boldsymbol{y}$ is inside the vicinity of the $\boldsymbol{w}_i$, then the closer they are, the stronger is the attraction. The ε_i is defined by equations:

$$c_i^{\varepsilon} := \psi \cdot \left(\text{clip}_0^1\left(\frac{\left\|\boldsymbol{d}_i^{\text{cpg}}\right\|_2}{\varepsilon_i}\right) - \frac{1}{2}\right)^3, \tag{25}$$

$$\dot{\varepsilon}_i := \begin{cases} \sigma \cdot c_i^{\varepsilon} & c_i^{\varepsilon} < 0 \\ \rho \cdot \varepsilon^2 \cdot c_i^{\varepsilon} & \text{otherwise,} \end{cases} \tag{26}$$

where the hyperparameters are set to $\psi = 8$, $\sigma = 10$ and $\rho = 0.25$. The equations ensure the following behaviour of the weight's vicinity's dynamics:

- an increase of the ε_i (and therefore the growth of the dynamic vicinity of the weight $\boldsymbol{w}_i$) if $\left\|\boldsymbol{d}_i^{\text{cpg}}\right\|_2 > \frac{\varepsilon_i}{2}$,
- a decrease of the ε_i (and therefore the shrinking of the dynamic vicinity of the weight $\boldsymbol{w}_i$) if $\left\|\boldsymbol{d}_i^{\text{cpg}}\right\|_2 < \frac{\varepsilon_i}{2}$.

The growth of the weight's vicinity is essential because it can not be assumed that the weight $\boldsymbol{w}_i$ will be close enough to the $\boldsymbol{y}$ after the initialization. The shrinking is required when the weight reaches close enough to the limit cycle. If the weight is close to the $\boldsymbol{y}$, then the vicinity starts to shrink. The shrinking vicinity ensures that the moving $\boldsymbol{y}$ leaves the vicinity and the weight stops being attracted

by the $\boldsymbol{y}$. Hence, the weight does not follow the $\boldsymbol{y}$ moving around the limit cycle. The relation of the weight $\boldsymbol{w}_i$, CPG's state $\boldsymbol{y}$ and the dynamic vicinity given by ε_i are illustrated in Fig. 7.

In this section, the attractive force mechanism, forcing the $\boldsymbol{w}_i$ weights to be close to the CPG's limit cycle with an unknown shape, was proposed. The mechanism of the altering influence of the proposed forces producing the self-organizing dynamics for the $\boldsymbol{w}_i$ weights is proposed in the following section.

3.1.4 Combination of the Attractions and Repulsions and the Convergence

Given the attraction of the weight $\boldsymbol{w}_i$ to the CPG's state $\boldsymbol{y}$ and the attractions and repulsions with other weights $\boldsymbol{w}_{j\neq i}$ from the two previous sections, a dynamics for the weight $\boldsymbol{w}_i$ is defined as follows:

$$\dot{\boldsymbol{w}}_i = \eta \cdot \left(c_i \cdot \boldsymbol{d}_i^{\text{cpg}} + (1 - c_i) \cdot \sum_{j=1}^{n} \boldsymbol{r}_i^j \right), \tag{27}$$

$$c_i \coloneqq \text{f}\left(\frac{\left\| \boldsymbol{d}_i^{\text{cpg}} \right\|_2}{\varepsilon_i} \right), \tag{28}$$

where $c_i \cdot \boldsymbol{d}_i^{\text{cpg}}$ can be rewritten as $\boldsymbol{r}_i^{\text{cpg}}$ using the Eq. 23. The combination factor c_i is changing the influence of dynamics between attracting by the state $\boldsymbol{y}$ and the influence from the other weights. It ensures that if the weight is far from the CPG's limit cycle, then the attractive force caused by the CPG's state $\boldsymbol{y}$ has greater influence, than the force caused by the weights interactions (i.e., the weight tries to get close to the CPG's limit cycle at first). If it is close enough to the CPG's limit cycle, it is being influenced by the other weights and starts to *"search for its position"* within the CPG's limit cycle.

To ensure the convergence, a decreasing learning-rate η is introduced. The learning-rate η can be represented by any function of time, which has a decreasing trend, and its limit of time approaches infinity equals zero.

In the proposed method, the attraction and repulsion within the weights $\boldsymbol{w}_i$ and the attraction by the CPG's state $\boldsymbol{y}$ are used to produce the dynamics to self-organize the weights and place them nearby the CPG's limit cycle of unknown shape l to produce a gait pattern given by the input value $\Delta\phi$. The forces within the weights are proposed based on the phase-distance relation, observed on the unit circle, which approximates the phase-distance relation on the complex shape of the CPG's limit cycle using the CPG-specific special norm $\|-\|_{\text{cpg}}$. The forces attracting the weights $\boldsymbol{w}_i$ to the CPG's state $\boldsymbol{y}$ are used to keep the weights close to the CPG's limit cycle of unknown shape with use of the weights' dynamic vicinities. Based on the given input value $\Delta\phi$, the final weights' organization around the CPG's limit cycle successfully generates each of the three gait patterns given by the input value $\Delta\phi$, as shown in the results in Chapter 4 Results.

3.2 Method Based on the Phase Learning

This method decomposes the problem, organizing the weights along the CPG's limit cycle, into two different tasks. The first one is to organize the weights within the phase to produce a stable gait pattern (i.e., respecting the coordination rules introduced in Chapter 2 Problem Statement). The second task is to place the weights on the limit cycle.

The previous method solves the task with the use of the special norm approximatting the phase-distance relation of the weights $\boldsymbol{w}_i$. The method, proposed in this section, estimates the CPG's phase course and maps the weights $\boldsymbol{w}_i$ on the CPG's state using the $\boldsymbol{w}_i$ weight's learned phase m_i.

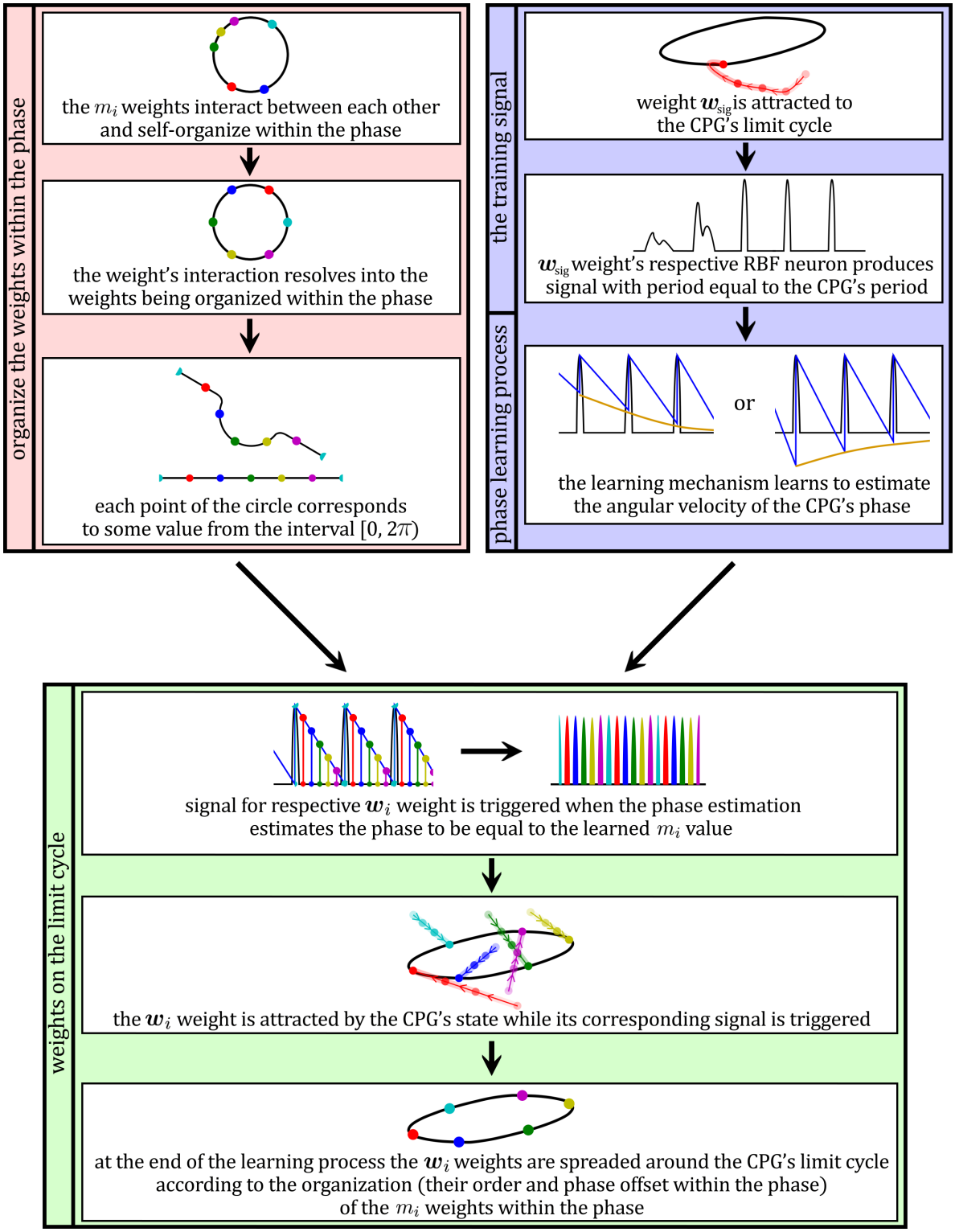

Figure 8: The figure visualizes the whole method based on phase learning. Processes regarding the m_i weights organizing within the phase are presented in the cell with red background. Cell with blue background includes the process of generating the training signal for the phase learning (phase estimation) and the phase learning process itself. The mapping of the m_i weights on the estimated phase and its use for organizing the $\boldsymbol{w}_i$ weights is shown in the cell with green background. The weights correspond to the legs by their colors, as shown in Fig. 2. The plots in this figure are not results of any experiment. They serve only as a visualization of the gait learning pipeline.

Two types of weights, $\boldsymbol{w}_i$ and m_i (corresponding to the i-th leg), are introduced. The weights $\boldsymbol{w}_1, \dots \boldsymbol{w}_6 \in \mathbb{R}^{\dim(\text{cpg})}$ are weights in the space of the CPG (the same weights as in the previous section 3.1) defined by the Eq. 7, which are used to produce the RBF neurons signals, as described by Eqs. 8 and 9. The weights $m_1, \dots, m_6 \in [0, 2\pi)$ are used to represent the weights organization within the phase, where m_i represents the phase in which the i-th leg's swing should be triggered. The m_i weights are being organized within the interval $[0, 2\pi)$ to fulfill the coordination rules. The phase values, given as the m_i weights, are used to determine the points on the CPG's limit cycle to which the $\boldsymbol{w}_i$ weights should be attracted.

The first task is solved by organizing the weights m_i within the phase while respecting the coordination rules introduced in Chapter 2 Problem Statement, as presented in section 3.2.1.

The second task is the $\boldsymbol{w}_i$ weights placing nearby the CPG's limit cycle on the positions corresponding to the m_i weights. The relation between $\boldsymbol{w}_i$ and m_i weights is learnd by estimating the CPG's phase course (dependency of the phase's growth on time). The phase estimation $\hat{\phi}^{\text{cpg}}(t)$, estimating the CPG's phase $\phi^{\text{cpg}}(t)$, is introduced in section 3.2.2.2. To estimate the phase course, a training signal generated by the CPG, proposed in section 3.2.2.1, is used. The weights m_i (representing the phase values) determine which point of the CPG's limit cycle $\boldsymbol{y}(t)$ corresponds to the CPG's phase estimation $\hat{\phi}^{\text{cpg}}(t)$ equal to the weight m_i. The corresponding CPG's limit cycle point is used to attract the final weight $\boldsymbol{w}_i$, as proposed in section 3.2.2.3. In other words, the CPG's phase estimation is used to select the place on the CPG's limit cycle to which the $\boldsymbol{w}_i$ weights are attracted according to their corresponding m_i weights.

3.2.1 Weights Within the Phase

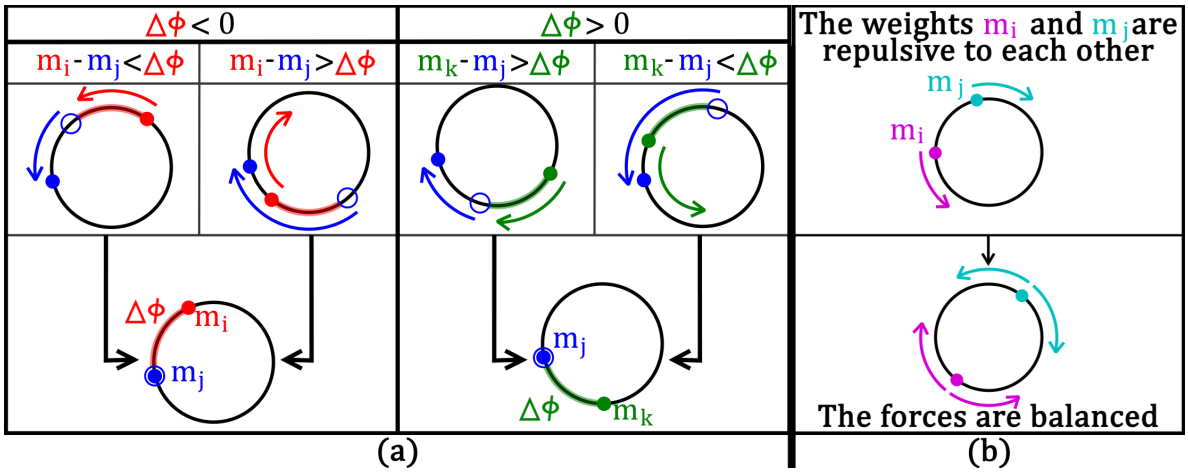

Figure 9: (a) The figure visualizes maintaining the given phase offset of the hind leg (red) from the middle leg (blue) on the left and the front leg (green) from the middle leg (blue) on the right. The red, green, and blue dots represent the weights (the blue weight is static in this illustration) within the phase (black circle). The red (green) arc represents the signed distance, which the hind (front) leg should maintain from the middle leg. The blank blue circle represents where the blue dot should be, in relation to the red (green) dot, to fulfill the condition of the given signed distance. The red, green, and blue arrows represent the movement of the respective parts (red and green weights and the blank blue circle) (b) The figure depicts the two weights (cyan and magenta dots) which are repulsive to each other (i.e., they correspond to the contralateral legs) within the phase (the black circle). The cyan and magenta arrows represent the direction of the forces influencing the corresponding weights. The mechanism, visualized in (a) corresponds to the bold arrows from the Fig. 4 (b), and the mechanism presented in (b) is represented by the dashed arrows in Fig. 4 (b).

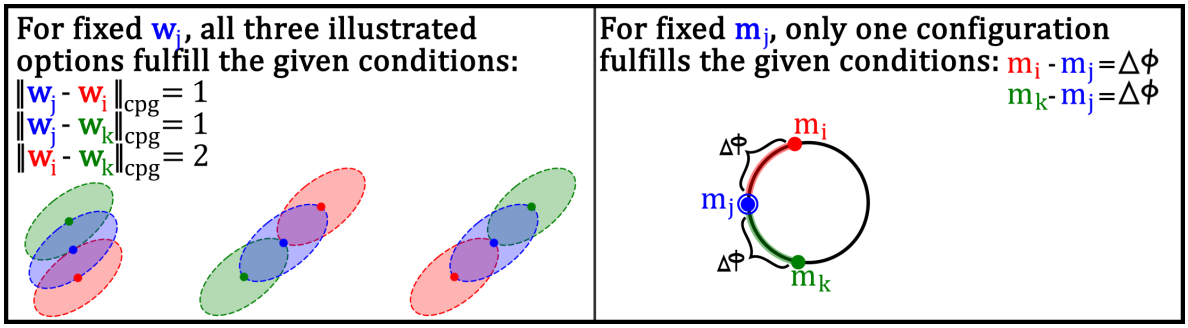

Figure 10: The figure visualizes the advantage of using only one-dimensional weights, where signed distance is used. On the left hand side, the example of using two-dimensional weights $\boldsymbol{w}_i$, $\boldsymbol{w}_j$ and $\boldsymbol{w}_k$ is shown. Three of infinitely many combinations, respecting the given conditions, are depicted. The weights are visualized as colorful dots. The dahsed ellipses represent the unit circle of the respective weights (i.e., the special norm of each ellipse's point distance from its respective weight is equal to one). On the right hand side is shown the only valid configuration fulfilling the given conditions for one-dimensional weights m_i, m_j and m_k using signed distance.

In this section, the method is proposed, which organizes (systematically places) the m_i weights within the phase (i.e., $m_i \in [0, 2\pi)$) according to the coordination rules by using the repulsive and attractive forces within the weights.

Using only one-dimensional weights, limited by interval $[0, 2\pi)$, gives a significant advantage in comparison to the organization of weights in more dimensional unlimited. While moving the weights in two and more dimensions, two criteria have to be considered, which are that the weights has to interact within themselves and at the same time the weights has to be kept close to the limit cycle. Using one-dimensional weights within the interval, only the weights interactions have to be considered as the limit cycle is represented by the interval itself.

There are only two directions for the movement of the weights within the phase, which have useful meaning. Consider the interval as a closed circle (i.e., the unit circle in x and y coordinates, where $x = \cos(\phi)$ and $y = \sin(\phi)$ for every $\phi \in [0, 2\pi)$). Given any weight $m \in [0, 2\pi)$, there are two possible directions for the weight movement on the circle. The weight is moving in the positive direction around the circle, or in the negative direction around the circle. As the phase progresses from zero in the positive direction (its value grows), the order within the phase is easily definable (see Fig. 10). The weights of lower values precedes the weights of higher values as the phase proceeds (i.e., as the robot's walking cycle proceeds).

The positive and negative direction is used in the weights interaction in the form of signed distance of the weights. The distance of one-dimensional values is usually denoted as an absolute value of their difference. However, by removing the absolute value, the signed distance is obtained (i.e., simply the difference of the values). The distance tells, that $\bar{b}$ is $|\bar{a} - \bar{b}|$ far from the $\bar{a}$, in positive or negative direction, but the signed distance specifies the direction (positive or negative) of the distance (i.e., $\bar{b}$ is $\bar{a} - \bar{b}$ far from $\bar{a}$).

The m_i weights' relative position within the phase is clearly defined by the given phase offset of the weights, corresponding to the difference of the weights' values. For instance lets assume the signed distance between first and second weight is given as $m_1 - m_2 = \phi_{2,1} = -\phi_{2,3}$ and the signed distance between first and third is given as $m_3 - m_2 = \phi_{2,3} = -\phi_{2,1}$. Then the weights m_1, m_2 and m_3 have clearly defined both, order with respect to the phase progression ((i) m_1, (ii) m_2, (iii) m_3) and differences (phase offsets) between the weights, as shown in the visualization in Figs. 9 and 10.

The m_i weights, corresponding to the phase of the i-th leg's action, are defined within the given interval $[0, 2\pi)$. The weights should interact as they are all binded to the circle on which they interact. The signed distance of the weights is computed as the difference of the weights' values. To behave as

on the circle, the function $\mathrm{g}(x)$, defined in Eq. 29, is applied to the weights' difference.

$$\mathrm{g}\left(x\right) \coloneqq \begin{cases} \mathrm{sign}(x) \cdot (2\pi - |x|) & |x| > \pi \\ x & \text{otherwise,} \end{cases} \tag{29}$$

The input phase offset $\Delta\phi$ together with the coordination rules determine the relation between the weights. The front legs' weights m_{front} should each maintain the phase offset from the relevant middle legs' weights $\mathrm{g}(m_{\text{front}} - m_{\text{middle}}) = \Delta\phi$, (where middle leg's weight is m_{middle}) and also the hind legs' weights m_{hind} should each maintain the phase offset from the relevant middle leg's weight $\mathrm{g}(m_{\text{hind}} - m_{\text{middle}}) = -\Delta\phi$. Those equalities imply three interaction rules, similar to the interaction rules given in section 3.1.2. This method's interaction rules are:

- if $\mathrm{g}(m_{\text{front}} - m_{\text{middle}}) < \Delta\phi$ (or $\mathrm{g}(m_{\text{hind}} - m_{\text{middle}}) > -\Delta\phi$), then the weight m_{front} (or m_{hind}) should repulse from m_{middle}, respectively,
- if $\mathrm{g}(m_{\text{front}} - m_{\text{middle}}) > \Delta\phi$ (or $\mathrm{g}(m_{\text{hind}} - m_{\text{middle}}) < -\Delta\phi$), then the weight m_{front} (or m_{hind}) should atract to m_{middle}, respectively,
- if $\mathrm{g}(m_{\text{front}} - m_{\text{middle}}) = \Delta\phi$ (or $\mathrm{g}(m_{\text{hind}} - m_{\text{middle}}) = -\Delta\phi$), then the weight m_{front} (or m_{hind}) should not be influenced by m_{middle}, respectively.

Given the third coordination rule, the contralateral legs can not undergo the same action simultaneously. Therefore, their respective weights must repulse each other.

The mechanism of weights interaction within the phase and its advantages compared to the multidimensional organization are illustrated in Figs. 9 and 10. The scheme in Fig. 4 (b) provides an overview of the relations within the weights. The rules form the following equations:

$$d^q_{(\text{front,middle})} \coloneqq \mathrm{g}\left(m^q_{\text{front}} - m^q_{\text{middle}}\right) - \Delta\phi, \tag{30}$$

$$d^q_{(\text{hind,middle})} \coloneqq \mathrm{g}\left(m^q_{\text{hind}} - m^q_{\text{middle}}\right) + \Delta\phi, \tag{31}$$

$$d^q_k \coloneqq \mathrm{g}\left(m^q_k - m^{1-q}_k\right), \tag{32}$$

where $\Delta\phi$ is the given phase offset; $k \in \{\text{front}, \text{middle}, \text{hind}\}$ represents the leg's anatomic position; $q \in \{0, 1\}$ representing left and right side of the robot, respectively (e.g., m^0_{front} is a weight corresponding to the left front leg). The Eqs. 30 and 31 were created based on the interaction rules. The Eq. 32 is given by the third coordination rule.

The Eqs. 30 - 31 define the forces, which force the weights based on their mutual interaction. The influence of the forces $d^q_{(\text{front,middle})}$, $d^q_{(\text{hind,middle})}$ and d^q_k on their respective weights' dynamics is given by their coeficients $c^q_{(\text{front,middle})}$, $c^q_{(\text{hind,middle})}$ and c^q_k, respectively. The coeficients are normalized by π (i.e., the half of the phase, which is the absolute value of maximum possible difference of two weights from interval $[0, 2\pi)$). Therefore their values are from interval $[0, 1]$. They are all computed by the same formula, but the c^q_k result is subtracted from one. The reason is that different behavior of the forces is required.

The forces maintaining the given distance ($d^q_{(\text{front,middle})}$ and $d^q_{(\text{hind,middle})}$) should be strongest when the weights' distance is very different from the given value. The closer the difference is to the given value, the less intense the force should be. Therefore, the corresponding coefficients ($c^q_{(\text{front,middle})}$ and $c^q_{(\text{hind,middle})}$) approach the value one if the absolute value of difference of the given value and current difference is close to the π (what is the maximum possible value of the difference's absolute value). If the given value and current difference are approximately the same, the coefficients approach zero and the force does not influence the weights.

On the contrary, the repulsive forces d_k^q should be strongest when the weights' difference is small because it is a wrong state. Therefore the influence of the repulsive forces grows with smaller weights' difference. If the difference approaches π, then the coefficient c_k^q approaches zero and the repulsive forces do not affect the weights.

The coeficients are given by the following equations:

$$c^q_{(\text{front,middle})} := \text{clip}_0^1\left(\left|\frac{d^q_{(\text{front,middle})}}{\pi}\right|\right), \tag{33}$$

$$c^q_{(\text{hind,middle})} := \text{clip}_0^1\left(\left|\frac{d^q_{(\text{hind,middle})}}{\pi}\right|\right), \tag{34}$$

$$c^q_k := 1 - \text{clip}_0^1\left(\left|\frac{d^q_k}{\pi}\right|\right), \tag{35}$$

where the function $\text{clip}_y^z(x)$ is given by Eq. 21. The variables from Eqs. 30 - 35 are then used to form the dynamic rules for the weights:

$$\dot{m}^q_{\text{front}} = \left(d^q_{\text{front}} \cdot c^q_{\text{front}} - d^q_{(\text{front,middle})} \cdot c^q_{(\text{front,middle})}\right) \cdot \mu, \tag{36}$$

$$\dot{m}^q_{\text{hind}} = \left(d^q_{\text{hind}} \cdot c^q_{\text{hind}} - d^q_{(\text{hind,middle})} \cdot c^q_{(\text{hind,middle})}\right) \cdot \mu, \tag{37}$$

$$\dot{m}^q_{\text{middle}} = \left(d^q_{\text{middle}} \cdot c^q_{\text{middle}}\right) \cdot \mu, \tag{38}$$

where $\mu = 5$ is a constant hyperparameter chosen empirically and the weights m_k^q correspond to the weights m_i as follows: $m_1 = m^0_{\text{front}}$, $m_2 = m^1_{\text{front}}$, $m_3 = m^0_{\text{hind}}$, $m_4 = m^1_{\text{hind}}$, $m_5 = m^0_{\text{middle}}$ and $m_6 = m^1_{\text{middle}}$. Note that all the weights m_i are bounded by interval $[0, 2\pi)$. Hence, if any of the weights declines below zero, the value of 2π is then added to the weight. Similarly, if the weight rises above the value of 2π, the value of 2π is then subtracted from the weight. It can be expressed more easily by defining that the $(\text{mod } 2\pi)$ operation is applied to the weights.

Process, described in this section, corresponds to the illustration in Fig. 8 in the cell with the red background.

This section introduces the mechanism of organizing the weights m_i within the phase to correspond to the given gait pattern and the coordination rules. The following section focuses on using the learned m_i weights, organized within the phase, to map the $\boldsymbol{w}_i$ weights on the CPG's limit cycle using the CPG's phase estimation.

3.2.2 Attaching the Weights to the CPG

Each point $\boldsymbol{y}(t)$ on the CPG's limit cycle corresponds to some phase $\phi^{\text{cpg}}(t)$. Therefore, each weight $\boldsymbol{w}_i$ can be assigned a point on the CPG's limit cycle, which corresponds to the weight's phase given by the value m_i. This subsecion proposes a mapping between the unknown CPG's phase $\phi^{\text{cpg}}(t)$ and its corresponding CPG's states $\boldsymbol{y}(t)$ by estimating the CPG's phase course to map the weights $\boldsymbol{w}_i$ on the CPG's limit cycle based on their corresponding phase values m_i.

It is known, that the CPG's limit cycle $l \subset \mathbb{R}^{\dim(\text{cpg})}$ consits of the CPG's states $\boldsymbol{y}(t)$ which repeat periodically (i.e., $\boldsymbol{y}(t) = \boldsymbol{y}(t+T)$, where T is the period). Hence, a phase can be assigned to each CPG's state on the CPG's limit cycle. The phase grows proportionally to time t, and same as the states $\boldsymbol{y}(t)$, the phase values also repeat periodically (i.e., $\phi^{\text{cpg}}(t) = \phi^{\text{cpg}}(t+T)$). Therefore, each state $\boldsymbol{y}(t)$ can be assigned a phase value $\phi^{\text{cpg}}(t) \in [0, 2\pi)$.

As both, the states $\boldsymbol{y}(t)$ and phase $\phi^{\text{cpg}}(t)$, are dependent on time and repeat periodically with the same time period T, the phase and the state are considered as mutually determining each other. In other words, there exists a bijection between the states $\boldsymbol{y}(t)$ and the phase $\phi^{\text{cpg}}(t)$. Informally speaking, in each moment in time t we can look on the phase value $\phi^{\text{cpg}}(t)$ and assign it the current CPG's state $\boldsymbol{y}(t)$ as its corresponding state.

As the $\boldsymbol{w}_i$ weights' phase is known (each weight $\boldsymbol{w}_i$ is related to its phase value m_i), the corresponding state $\boldsymbol{y}_i$ on the CPG's limit cycle can be assigned. Both, the phase $\phi^{\text{cpg}}(t_i) = m_i$ and the state $\boldsymbol{y}(t_i) = \boldsymbol{y}_i$, are given by a moment in time t_i. The phase value is known via the weight $m_i = \phi^{\text{cpg}}(t_i)$.

As the time t_i is known in the moment when the phase equals the weight (i.e., $\phi^{\text{cpg}}(t_i) = m_i$), the t_i can be used to determine the state $\boldsymbol{y}(t_i)$ which correspond to the phase $\phi^{\text{cpg}}(t_i)$. The assigned $\boldsymbol{y}(t_i)$ is the point, which the weight $\boldsymbol{w}_i$ tries to achieve.

However, the CPG's shape (i.e., the states $\boldsymbol{y}(t)$), as well as the phase dependency on time $\phi^{\text{cpg}}(t)$ are both unknown. In each moment t only the CPG's state $\boldsymbol{y}(t)$ is known. To place correctly the weights $\boldsymbol{w}_i$ close to their corresponding $\boldsymbol{y}(t_i)$, the phase $\phi^{\text{cpg}}(t_i)$ must be known.

This section proposes a solution how to learn the phase course in section 3.2.2.2 using the signal which is generated by the CPG (hence, it has the same period T and the same phase) as proposed in section 3.2.2.1, and then, based on the learned (estimated) phase course, the solution how to place the weights $\boldsymbol{w}_i$ near to its corresponding CPG's states $\boldsymbol{y}(t_i)$ is described in section 3.2.2.3.

3.2.2.1 The Training Signal

The mechanism, which estimates the unknown CPG's phase, requieres the input signal of the same period T, as the CPG's phase has. In this section, a mechanism to gain the specific input signal, based on the CPG, is proposed.

The phase can be measured from an arbitrary moment in time (i.e., it is relative). Each moment in time t determines the CPG's state $\boldsymbol{y}(t)$, which always corresponds to the same phase $\phi^{\text{cpg}}(t)$. Hence, everytime the CPG reaches the state $\boldsymbol{y}(t)$, the phase $\phi^{\text{cpg}}(t)$ has the same value. Therefore, to estimate the CPG's phase is the same as to estimate the occurrence of some specific CPG's state in time (the estimated phase is considered equal to zero in the moment of the state's occurrence, i.e., for state $\boldsymbol{y}(t_0)$ in moment t_0 the estimated phase value is equal to zero $\phi^{\text{cpg}}(t_0) = 0$).

Hence, to determine the phase course of the CPG, a signal $\text{s}(t)$, signalizing the occurrence of the selected CPG's state, is used, as it has the same phase course as the CPG.

The signal $\text{s}(t)$ is generated by one weight $\boldsymbol{w}_{\text{sig}}$ with its corresponding RBF neuron. The RBF neuron (generating the signal $\text{s}(t)$) fires when the CPG's state $\boldsymbol{y}$ is close enough to the weight $\boldsymbol{w}_{\text{sig}}$. The weight is placed (its position is learned) nearby the CPG's limit cycle, which selects its closest point of the CPG's limit cycle as the state $\boldsymbol{y}(t_0)$, which indicates the zero value for the estimated phase. The signal $\text{s}(t)$ consists of pulses, which occure as the CPG's state approaches the state $\boldsymbol{y}(t_0)$ selected by the learned weight $\boldsymbol{w}_{\text{sig}}$.

The phase estimation process expects the input signal fulfilling few requirements. The signal's maximum within each period has to be a strict maximum within the period, and its value has to approach the value one. Therefore, the signal $\hat{\text{s}}(t)$ is proposed, which modifies the signal $\text{s}(t)$ to meet the requirements.

The weight $\boldsymbol{w}_{\text{sig}}$ has to be close enough to the CPG's limit cycle to produce the correct RBF signal. To do so, the weight $\boldsymbol{w}_{\text{sig}}$ is randomly initialized, and then the same mechanism as introduced in section 3.1.3 is used.

$$\boldsymbol{d}_{\text{sig}} \coloneqq \boldsymbol{y} - \boldsymbol{w}_{\text{sig}}, \tag{39}$$

$$\boldsymbol{r}_{\text{sig}}^{\text{cpg}} \coloneqq \mathrm{f}\left(\frac{\left\|\boldsymbol{d}_{\text{sig}}\right\|_2}{\varepsilon}\right)\boldsymbol{d}_{\text{sig}}, \tag{40}$$

$$c_{\text{sig}} \coloneqq \psi \cdot \left(\text{clip}_0^1\left(\frac{\left\|\boldsymbol{d}_{\text{sig}}\right\|_2}{\varepsilon}\right) - \frac{1}{2}\right)^3, \tag{41}$$

$$\dot{\varepsilon} = \begin{cases} \sigma \cdot c_{\text{sig}} & c_{\text{sig}} < 0 \\ \rho \cdot \varepsilon^2 \cdot c_{\text{sig}} & \text{otherwise,} \end{cases} \tag{42}$$

$$\dot{\boldsymbol{w}}_{\text{sig}} \coloneqq \eta \cdot \boldsymbol{r}_{\text{sig}}^{\text{cpg}}, \tag{43}$$

where function $\mathrm{f}(x)$ is defined in Eq. 24; $\text{clip}_y^z(x)$ is defined in Eq. 21; the constant hyperparameters $\psi = 8$, $\sigma = 10$ and $\rho = 0.25$ were used; for the decreasing learning rate η, ensuring the convergency, a function dependent on time, with limit approaching infinity equal to zero, was used.

The signal $\mathrm{s}(t)$ is produced by the RBF neuron's signal (the same mechanism as presentedin 9, with different variables):

$$\mathrm{s}(t) \coloneqq \exp\left(-\epsilon \cdot \left\|\boldsymbol{y} - \boldsymbol{w}_{\text{sig}}\right\|_2^2\right), \tag{44}$$

where constant hyperparameter $\epsilon = 8$ was used. The signal $\mathrm{s}(t)$ is then modulated to produce the needed training signal $\hat{\mathrm{s}}(t)$:

$$\hat{\mathrm{s}}(t) \coloneqq \text{clip}_0^1\left(\left(\mathrm{s}(t) - 0.9\right) \cdot 10\right). \tag{45}$$

The dependence on the time t is not explicit, but the time influences the components (they are defined by their time derivatives) of which the signal $\mathrm{s}(t)$ is composed. Process, described in this section, corresponds to the first two steps in the cell with blue background in the illustration showed in Fig. 8.

The training signal $\hat{\mathrm{s}}(t)$, which has the same phase course as the CPG, is proposed with the use of the weight $\boldsymbol{w}_{\text{sig}}$ and its respective RBF neuron. The next step is to estimate the CPG's phase course with the use of the learned signal $\hat{\mathrm{s}}(t)$.

3.2.2.2 Phase Learning Process

Given the training signal $\hat{\mathrm{s}}(t)$, the mechanism for learning the phase course is proposed in this section. To gain the CPG's phase value $\phi^{\text{cpg}}(t)$, the relation between the phase and time has to be known. The phase grows proportionally to time. However, growth of the phase by 2π does not have to correspond to the growth of the time by 2π for all oscillators. The zero point of phase (i.e., moment, when $\phi^{\text{cpg}}(t) = 0$) can be assigned to an arbitrary time moment. To determine the phase's course, the phase differential (i.e., how fast the phase grows with respect to time) has to be known.

The signal $\hat{\mathrm{s}}(t)$ has distinct peaks repeating periodically, with the value of the peaks being equal to one. The peaks serve as a marker of the moment when phase value $\phi^{\text{cpg}}(t)$ should be equal to zero. In other words, each peak resets the phase, so the phase starts to grow from zero again.

To learn the phase growth (the phase's dependency on time), the parameter $\mathrm{p}(t)$ is introduced. If the $\mathrm{p}(t)$ is learned properly, its domain corresponds to the interval $[0, 1]$, where the signal's peak value equal to one resets the value of $\mathrm{p}(t)$ to one. The value then decreases, and if learned correctly, it approaches zero at the moment when another peak occures. The peak then resets the value of $\mathrm{p}(t)$ to one again and so on. The $\mathrm{p}(t)$ is learned correctly if its functional value declines with the correct speed.

The decline of $\mathrm{p}(t)$ functional value is given by the value of a, which is being learned. The a represents $\mathrm{p}(t)$ function's differential with respect to time, during most of the period's duration. In

other words, the a is a parameter indirectly estimating the phase's angular velocity.[4] As the $\mathrm{p}(t)$ is meant to decrease, the differential a has to be always negative.

If the absolute value of a is too large, the value of $\mathrm{p}(t)$ decreases too fast (i.e., the estimation estimates the phase growth faster than the phase really grows). Then the value of $\mathrm{p}(t)$ is lower than zero in the moment of the signal's peak (the signal's peak resets the value of $\mathrm{p}(t)$ to one).

In the other case, if the absolute value of a is too small, then the value of $\mathrm{p}(t)$ decreases too slow (i.e., the estimation estimates the phase growth slower than the phase really grows). Hence, its value is greater than zero in the moment of the signal's peak (the signal's peak resets the value of $\mathrm{p}(t)$ to one).

The function $\mathrm{p}(t)$ is given by following equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{p}(t) = a, \tag{46}$$

and if $\hat{\mathrm{s}}(t) = 1$, then the value of $\mathrm{p}(t)$ is reseted to be equal to one. In other words, if there is a signal's peak, then the $\mathrm{p}(t)$ is set to one, else the value of $\mathrm{p}(t)$ decreases based on the value of a.

The a is given by following equation:

$$a \coloneqq \begin{cases} a - \kappa \cdot \mathrm{p}(t_-) & \hat{\mathrm{s}}(t) = 1 \\ a & \text{otherwise,} \end{cases} \tag{47}$$

where t_- is the moment right before the moment t (i.e., the $\mathrm{p}(t_-)$ denotes the value of $\mathrm{p}(t)$ right before it is reset to one by the signal's peak); hyperparameter κ is a small positive constant (in this work $\kappa = 0.02$); $\hat{\mathrm{s}}(t)$ is given by Eq. 45.

The Eq. 47 implies that the differential a is constant until the signal's peak occures. If the value $\mathrm{p}(t)$ is not equal to zero in the moment of the signal's peak, then the value of a changes. The change is dependent on how far from zero the value $\mathrm{p}(t)$ is in the moment of the signal's peak:

- If $\mathrm{p}(t_-) < 0$, then the $\mathrm{p}(t)$ decreases too fast. Therefore the a is increased by $-\kappa \cdot \mathrm{p}(t_-)$.
- If $\mathrm{p}(t_-) > 0$, then the $\mathrm{p}(t)$ decreases too slow. Therefore the a is decreased by $-\kappa \cdot \mathrm{p}(t_-)$.

If learned properly, the value $\mathrm{p}(t)$ decreases periodically from one to zero in exactly T long time period, given by the input signal $\hat{\mathrm{s}}(t)$. Hence, the estimated CPG's phase value can be expressed as:

$$\hat{\phi}^{\mathrm{cpg}}(t) \coloneqq 2\pi(1 - \mathrm{p}(t)) \tag{48}$$

The example of learning the phase estimation with differently initialized a are shown in Fig. 11 and 12, where as the training signal was used the RBF neuron signal of weight on the Matsuoka's neural oscillator's limit cycle and the sinus function $\sin(t)$, respectively. The progress of the differential a is shown in Fig. 13.

Process, described in this section, corresponds to the third step in the cell with blue background in the illustration shown in Fig. 8 (the phase learning) and to the first step in the cell with green background in the illustration shown in Fig. 8 (where the phase is learned and the mapping of the weights m_i on the learned phase is shown).

The CPG's phase is estimated and the phase values for weights $\boldsymbol{w}_i$ are given by their corresponding weights m_i. The following mechanism uses the estimated CPG's phase $\hat{\phi}^{\mathrm{cpg}}(t)$ to assign the $\boldsymbol{w}_i$ weights a CPG's state $\boldsymbol{y}(t)$ on the CPG's limit cycle based on the weights learned phase m_i.

[4] The phase's differetial with respect to time is equal to $-2\pi a$

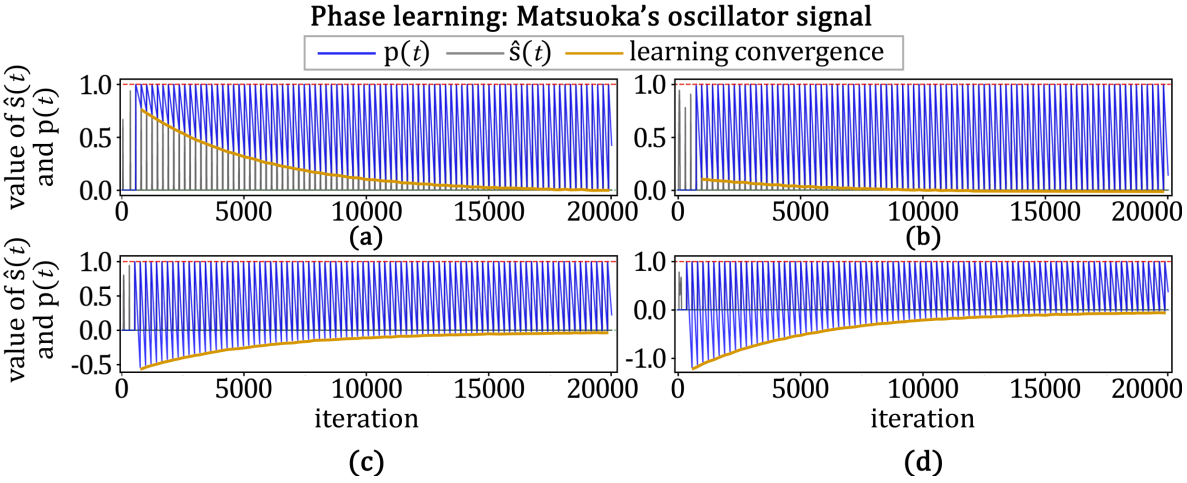

Figure 11: Plots of the phase learning process with four different initializations of a, learning the phase course of the signal with a period of the CPG (Matsuoka's neural oscillator). The vertical axis represents both, the value of $\mathrm{p}(t)$ and the value of the training signal $\hat{\mathrm{s}}(t)$. The horizontal axis represents the count of iterations ($t = \texttt{iteration\_number}/100$). The $\hat{\mathrm{s}}(t)$ signal is modulated signal $\mathrm{s}(t)$ generated by the RBF neuron coupled with the weight $\boldsymbol{w}_{\mathrm{sig}}$. The weight $\boldsymbol{w}_{\mathrm{sig}}$ is close to the CPG's limit cycle. The learned phase course $\mathrm{p}(t)$ (blue) converges for all the cases to the optimal value, which is shown by the converging orange line, representing the value of $\mathrm{p}(t)$ in the moment of the training signal's peaks ($\hat{\mathrm{s}}(t) = 1$). The initial values of a are -0.1, -0.4, -0.7 and -1 in the (a), (b), (c) and (d), respectively. The progression of the differential a for all the cases is shown in Fig. 22 in the left-hand side plot. Note that the $\mathrm{p}(t)$ remains zero until the modified input signal approaches value one.

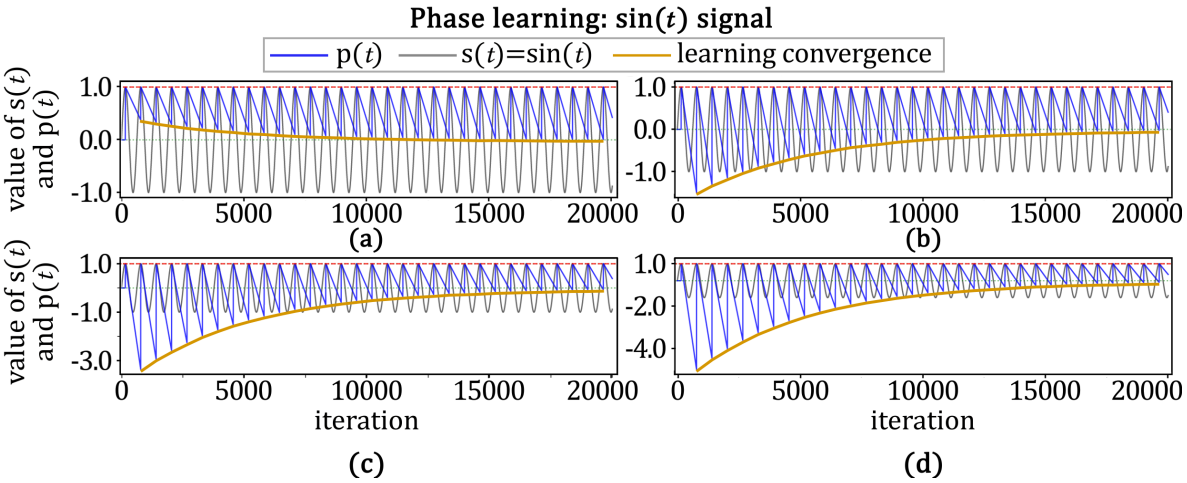

Figure 12: Plots of the phase learning process with four different initializations of a, learning the phase course of the sinus function signal. The vertical axis represents both, the value of $\mathrm{p}(t)$ and the value of the unmodified input signal $\mathrm{s}(t) = \sin(t)$. The horizontal axis represents the count of iterations ($t = \texttt{iteration\_number}/100$). The input signal $\hat{\mathrm{s}}(t)$ is the modified signal of the sinus function $\hat{\mathrm{s}}(t) = \mathrm{clip}_0^1\left((\mathrm{s}(t) - 0.98) \cdot 50\right)$. The learned phase course $\mathrm{p}(t)$ (blue) converges for all the cases to the optimal value, which is shown by the converging orange line, representing the value of $\mathrm{p}(t)$ in the moment of the training signal's peaks ($\hat{\mathrm{s}}(t) = 1$). The initial values of a are -0.1, -0.4, -0.7 and -1 in the (a), (b), (c) and (d), respectively. The progression of the differential a for all the cases is shown in Fig. 22 in the right hand side plot. Note that the $\mathrm{p}(t)$ remains zero until the modified input signal approaches value one.

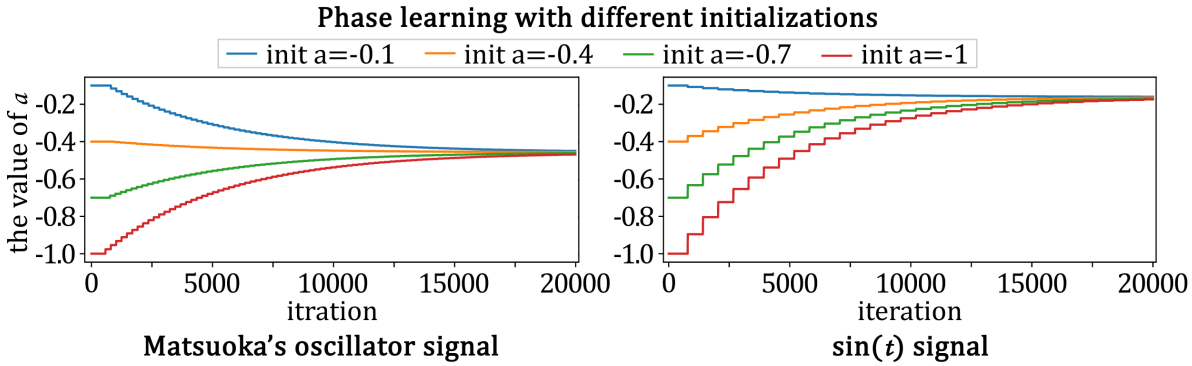

Figure 13: The plots represent the progression of the differential a for phase learning shown in Figs. 11 on the left hand side and 12 on the right hand side. The vertical axis represents the value of a, and the horizontal axis represents the count of iterations ($t = \texttt{iteration\_number/100}$).

3.2.2.3 Weights on the Limit Cycle

This section introduces a method for organizing the weights $\boldsymbol{w}_i$ within the CPG's limit cycle with the use of methods introduced in previous sections. In section 3.2.1 the weights m_i (for $i = 1, \ldots, 6$) were organized within the phase (i.e., placed within the interval $[0, 2\pi)$). The sections 3.2.2.2 and 3.2.2.1 proposed the metod for estimating the phase based on a signal generated by the RBF neuron respective to the weight placed on the CPG's limit cycle. With estimated CPG's phase $\hat{\phi}^{\text{cpg}}(t)$, the weights m_i are used to determine when the corresponding weights $\boldsymbol{w}_i$ are being attracted by the CPG's state $\boldsymbol{y}$.

The m_i weights combined with the phase estimation $\hat{\phi}^{\text{cpg}}(t)$ produce a signal using the RBF neurons, generated by following equation:

$$p_i^m := \exp\left(-\epsilon \cdot \mathrm{g}\left(m_i - \hat{\phi}^{\text{cpg}}(t)\right)^2\right), \tag{49}$$

where the hyperparameter $\epsilon = 24$; function $\mathrm{g}(x)$ is defined in Eq. 29. The signal sends pulse to the respective weight $\boldsymbol{w}_i$ as the estimated CPG's phase $\hat{\phi}^{\text{cpg}}(t)$ approaches the value m_i.

The signal p_i^m has the period of the used CPG and it could be used as a rhythmical input for the i-th leg to invoke locomotion. However, it does not have the properties of the CPG, like synchronization ability and perturbation resistance.

The reason is, that the signal-producing mechanism produces the signal based on the estimated phase $\hat{\phi}^{\text{cpg}}(t)$, but not based on the CPG's phase $\phi^{\text{cpg}}(t)$ itself. Once the CPG's limit cycle phase's course estimate is learned, it behaves almost independently on the CPG (if the phase learning process is still active, the estimated course is being influenced with a delay, else the phase estimation is not influenced by the CPG's state).

To actually use the CPG to produce a signal for individual legs, the weights $\boldsymbol{w}_i$ have to be placed on (or nearby) the CPG's limit cycle. To do so, the p_i^m RBF neuron signal is used. Each weight m_i, which is placed within the phase, has its corresponding weight $\boldsymbol{w}_i$ (introduced in Eq. 7). The p_i^m signal determines the moment in time when the weight $\boldsymbol{w}_i$ should trigger the action via its corresponding RBF neuron and in that moment the weight $\boldsymbol{w}_i$ should be close enough to the CPG's current state $\boldsymbol{y}$ (introduced in Eq. 6) to produce the RBF signal as defined in Eq. 9. Hence the signal p_i^m is used to define when the weight $\boldsymbol{w}_i$ is attracted by the CPG's current state $\boldsymbol{y}$, as defined in the following equation:

$$\dot{\boldsymbol{w}}_i = (\boldsymbol{y} - \boldsymbol{w}_i) \cdot p_i^m. \tag{50}$$

The Eq. 50 represents the dynamics for the weights $\boldsymbol{w}_i$, which places them around the CPG's limit cycle based on the m_i weights organization within the phase.

Process, described in this section 3.2.2.3, corresponds to the cell with green background shown in illustration in Fig. 8. The final result of the whole section 3.2.2 is shown in the last step of the cell with green background in Fig. 8.

This section presented a method solving the given problem of organizing the weights around the CPG's limit cycle to produce a gait pattern given by the phase offset $\Delta\phi$ of the consecutive legs by decomposing the problem on two separated tasks, the m_i weights organization within the phase and the estimation of unknown CPG's phase. The weights m_i, organized within a phase (based on the rules from Chapter 2 Problem Statement), determine the moment when the signal p_i^m should send a pulse. The p_i^m signal determines when the weight $\boldsymbol{w}_i$ is attracted by the CPG's state $\boldsymbol{y}$, which results in weights $\boldsymbol{w}_i$ being organized along the CPG's limit cycle. The m_i weights are organized within the phase according to the coordination rules from Chapter 2 Problem Statement. A corresponding CPG's state is assigned to each of the weights $\boldsymbol{w}_i$ via the estimated CPG's phase $\hat{\phi}^{\text{cpg}}(t)$ (as mentioned in the introduction of this section 3.2.2, there is a bijection between the CPG's phase and its states). Hence the organization within the limit cycle also corresponds to the given coordination rules introduced in Chapter 2 Problem Statement.

Based on the given input value $\Delta\phi$, the final weights' organization around the CPG's limit cycle successfully generates each of the three gait patterns given by the input value $\Delta\phi$, as shown in the results in Chapter 4 Results. Moreover, the method, proposed in this section 3.2, eliminates most of the drawbacks of the method proposed in the previous section 3.1, as discussed in Chapter 5 Discussion.

Chapter 4
Results

Two proposed methods, the Special Norm based Method (SNM) and the Phase Learning based Method (PLM), were tested to prove their ability to self-organize the robot's legs to produce desired gait patterns. Both methods proved functional for all three desired gait patterns, as shown by simulation results from *CoppeliaSim* simulator.

In the following section 4.1 the experimental setup is described, then the results for SNM are presented in section 4.2, and for PLM the results are shown in section 4.3. The last section 4.4 introduces experiments, which were proposed to compare the computational and learning time of the methods.

4.1 Experiments Setup

The dynamic system, consisting of introduced equations, was run for three different inputs of phase offset of consecutive legs to demonstrate the solutions.

An Euler's method with step size $s = 0.01$ was used to obtain values from differential equations. The simulation runs for 20000 iterations.

To produce a signal, based on the weights organization (distribution) around the CPG's limit cycle to invoke the robot's movement in the *CoppeliaSim* simulator, the RBF neurons, introduced in Eq. 9, were used. The signals obtained from RBF neurons were then modified and their peaks were used to trigger the predefined swing movement, followed by a predfined stance movement for the robot's legs.

The methods were both implemented using *Python 3* programming language. In the *CoppeliaSim* simulator, a hexapod model *PhantomX-v3.1* was used to run the simulations. To convert the generated movement instructions to *CoppeliaSim* simulator, a *Python* API was used.

4.2 Method Using Special Norm

The SNM method takes one $\boldsymbol{w}_i \in \mathbb{R}^{\dim(\text{cpg})}$ weight for each of the six legs and organizes them around the CPG's limit cycle of unknown shape. The weights are organized with use of attractive and repulsive forces within the weights and by attractive force invoked by the CPG's state $\boldsymbol{y}$. The results of the experiments proved that the method successfully produces all three desired gait patterns.

Each of the desired gait patterns is determined by the phase offset of two consecutive legs. For all three given phase offsets ($2\pi/3$, π, and $\pi/3$) the method organized the weights nearby the limit cycle in a valid order (see Fig. 14) producing a signal to generate the given gait pattern using RBF neurons (see Fig. 15). The RBF neuron signals were then used to trigger the swing of the robot's legs in the *CoppeliaSim* simulator (see Fig. 16).

Observing the RBF signals in Fig. 15, the various amplitude of the peaks is visible, as well as the various width of the pulses (for the transition gait and wave gait in (a) and (c), respectively). The inconsistency of the pulses could cause problems for some locomotion controllers in mapping the pulses to the legs' actions.

The different amplitudes are caused by the weights being differently far from the CPG's limit cycle. The weights are influenced by two kinds of forces. The repulsion and attraction to the other weights and the attraction to the CPG's limit cycle. Only the repulsive forces within the weights are forcing the weights away from the CPG's limit cycle.

The behaviour of the forces within the weights is strongly influenced by the implementation of the special norm. The special norm's definition determines the vicinity of the weights mutual influence.

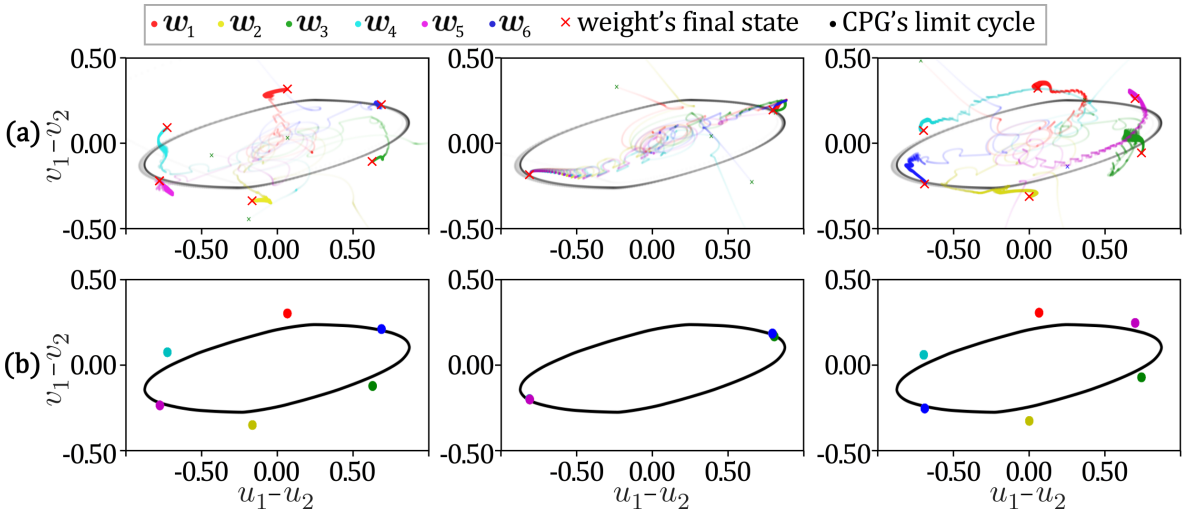

Figure 14: The figure shows the CPG and weights $\boldsymbol{w}_i$ of the SNM. The progression of the weights and the CPG is plotted in the row (a) (the upper plots). In the row (b) (the lower plots), a final state of the weights is shown. The input values of phase offset of consecutive legs were $\Delta\phi = \frac{2\pi}{3}$, $\Delta\phi = \pi$ and $\Delta\phi = \frac{\pi}{3}$, respectively, from left to right. The values correspond to the transition, tripod, and wave gait patterns, respectively, from left to right. The red crosses mark the end states of the weights in (a). Some weights are overlapping, which means that the legs corresponding to the overlapping weights move simultaneously. The colors of the weights match the colors of their respective leg from Fig. 2.

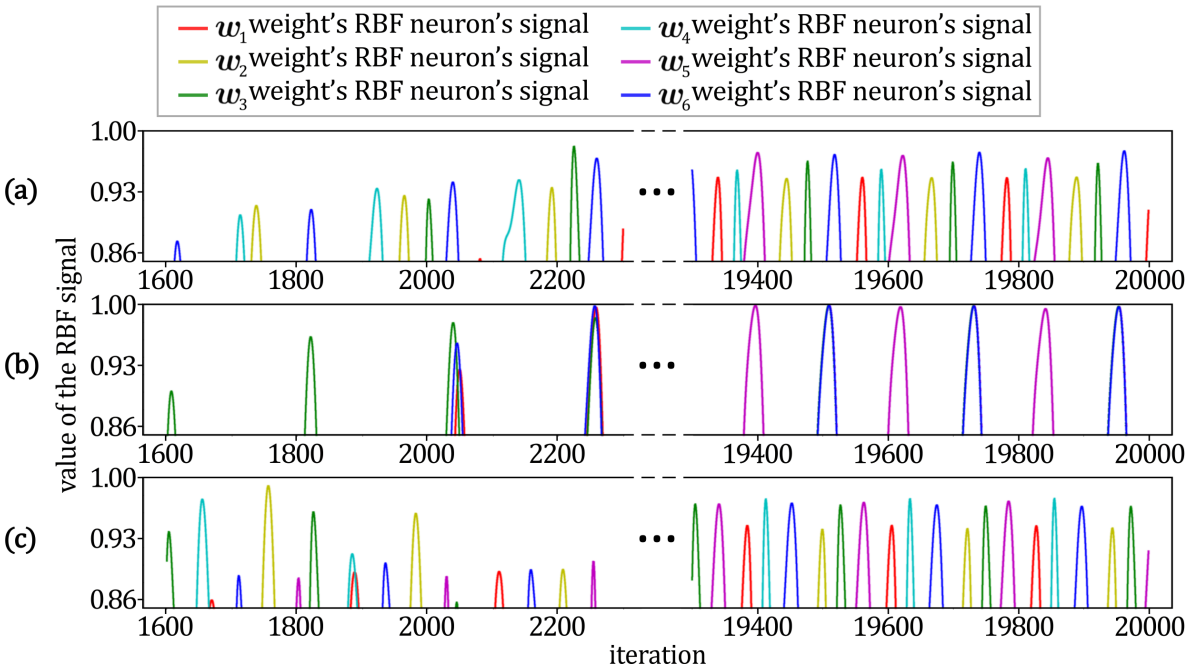

Figure 15: Plots of the generated transition (a), tripod (b), and wave (c) gait patterns' signals by the SNM. The plots show the signals generated by the RBF neurons based on the difference of the corresponding weight and the CPG's current state. The left plots show the generating of the signals initially, and the right plots show the generated signals at the end of the simulation. The horizontal axis represents the iteration number (t = `iteration_number/100`). The vertical axis represents the signals' value. The signals for simultaneously moving legs are overlapping in (b). Note that the colors of the signals match the colors of their respective leg from Fig. 2.

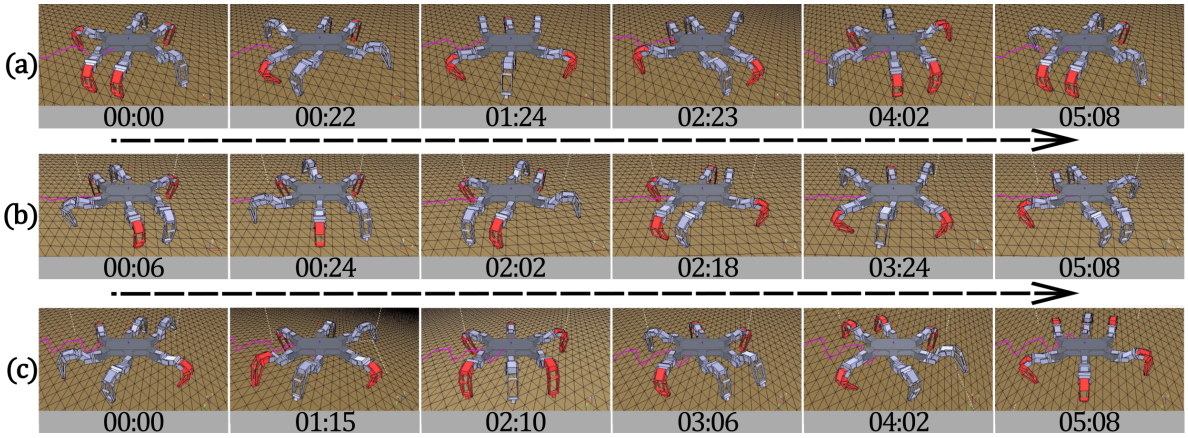

Figure 16: A showcase of applying the proposed method using special norm in the *CoppeliaSim* simulator. Each sequence of images shows six chosen frames from one cycle of each of the gait patterns. The images contain a timestamp. The timestamp measures the time since the beginning of the showed cycle in format *seconds:frames*, where each second consists of 25 frames. The gates are transition gait in (a), tripod gait in (b), and wave gait in (c). The arrows between the image sequences indicate both the direction of the robot's locomotion and the individual frames' time passage.

There are many approaches how to find a suitable ellipsoid to generate the special norm, which considers all the points of the ellipsoid's surface having their norm equal to one. Different approaches would affect the results differently. The approach used for the shown experiments is presented in the Appendix A Special Norm Finder.

The various width of the pulses (see Fig. 15) is caused by the specifics of the used CPG and the RBF neuron functions. The RBF neuron fires the pulse based on the Euclidean distance of its respective weight $\boldsymbol{w}_i$ to the CPG's state $\boldsymbol{y}$. However, the CPG's state $\boldsymbol{y}$ moves with different speeds through different space regions. Therefore, it is in the RBF neuron's radius for a longer time period for some weights than for the other weights. Hence, the pulse of the weights, in whose vicinity the $\boldsymbol{y}$ moves slower, have a longer duration (i.e., are wider in the resulting plot). The diverse pulses' width could cause problems for some locomotion controllers while mapping the legs' actions on the RBF signals.

Despite the irregularities of the pulses, the method proved itself by successfully generating all three given gait patterns.

4.3 Method Using Phase Learning

To produce stable gait patterns, the PLM decomposes the problem of organizing the weights within themselves and keeping them close to the CPG's limit cycle.

The method organizes the $m_i \in [0, 2\pi)$ weights within the phase (the weights are learned the values which satisfy the coordination rules) and parallelly it estimates the unknown CPG's phase $\phi^{\text{cpg}}(t)$.

Based on the m_i weights and the learned phase estimation, the method organizes the $\boldsymbol{w}_i \in \mathbb{R}^{\dim(\text{cpg})}$ weights around the CPG's limit cycle of an unknown shape, by using the relation of the CPG's states $\boldsymbol{y}(t)$ and their respective estimated phase $\hat{\phi}^{\text{cpg}}(t)$.

The weights $\boldsymbol{w}_i$, organized around the CPG's limit cycle, produce signal via RBF neurons, which are used to trigger the swing of the robot's legs in the *CoppeliaSim* simulator to produce the given gait patterns. The results of the experiments proved that the method successfully produces all three given gait patterns.

The results of the particular tasks are shown in the following sections (organizing m_i weights within the phase in section 4.3.1, generating the learning signal and estimating the CPG's phase in section 4.3.2, placing the $\boldsymbol{w}_i$ weights around the CPG's limit cycle in section 4.3.3).

4.3.1 Weights Within the Phase

The weights m_i were organized within the interval $[0, 2\pi)$, representing the phase of i-th leg swing triggering. The interval is easily visualized as a unit circle, which represents the limit cycle. The learning process results consist of the four randomly initialized sets of weights for each given input value (i.e., for each given gait pattern). Each gait pattern was learned successfully for each of the random initializations, as shown in Fig. 17 for the transition gait, in Fig. 18 for the tripod gait, and in Fig. 19 for the wave gait.

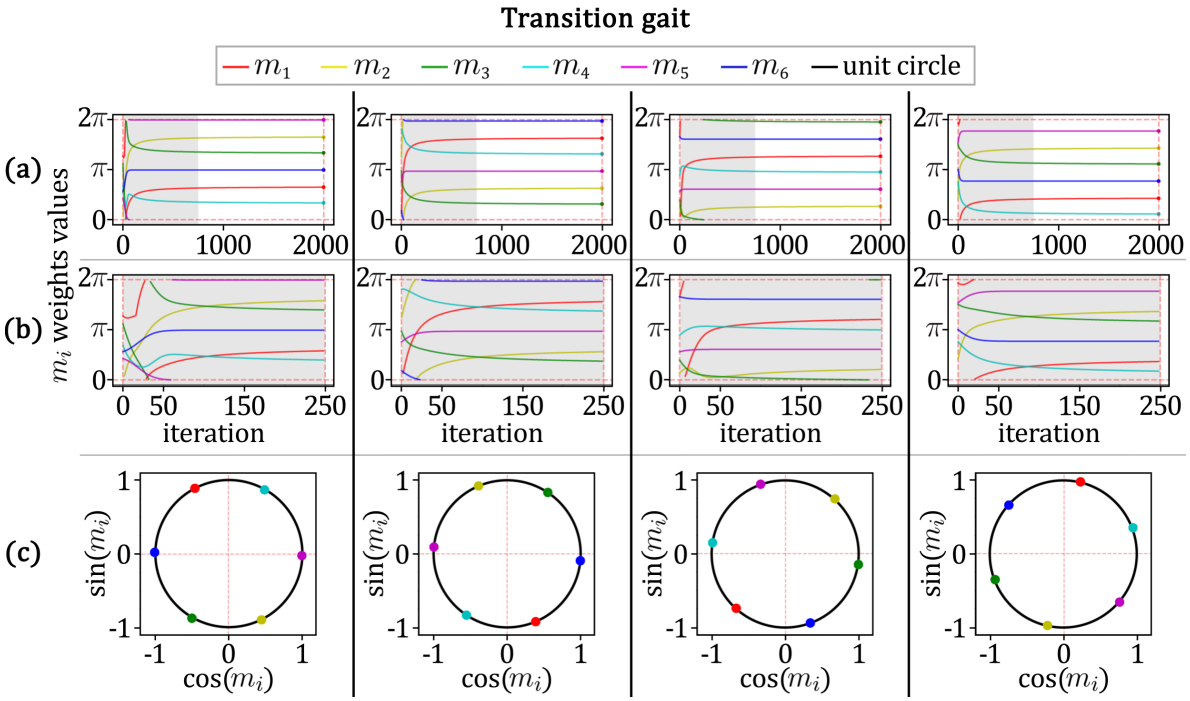

Figure 17: The weights m_i for the transition gait. (a) Four different random initializations of the weights and their progression within the first 2000 iterations. (b) Close up of the first 250 iterations of the weights progression (the area with grey background in (a)). In both, (a) and (b), the vertical axis represents the value of the weights and the horizontal axis represents the iteration number ($t =$ `iteration_number/100`). (c) The final weights' states (colorful dots) of the four different random initializations mapped on the cycle (black)(horizontal axis is given by $\cos(m_i)$, and the vertical axis by $y = \sin(m_i)$).

4.3.2 Phase Learning Process and the Training Signal

The method of learning the CPG's phase dependency on time learns the phase course based on the signal $\mathrm{s}(t)$, which has the same period T as the CPG. The input signal for the method must be periodic and it must have distinct peaks approaching the value one in its maximum (globally local maximum within each period). Hence the input signal $\mathrm{s}(t)$ is modified in signal-specific function $\hat{\mathrm{s}}(t)$ (the function $\hat{\mathrm{s}}(t)$ differs for various signals $\mathrm{s}(t)$). The phase's dependency on time is given by function $\mathrm{p}(t)$,

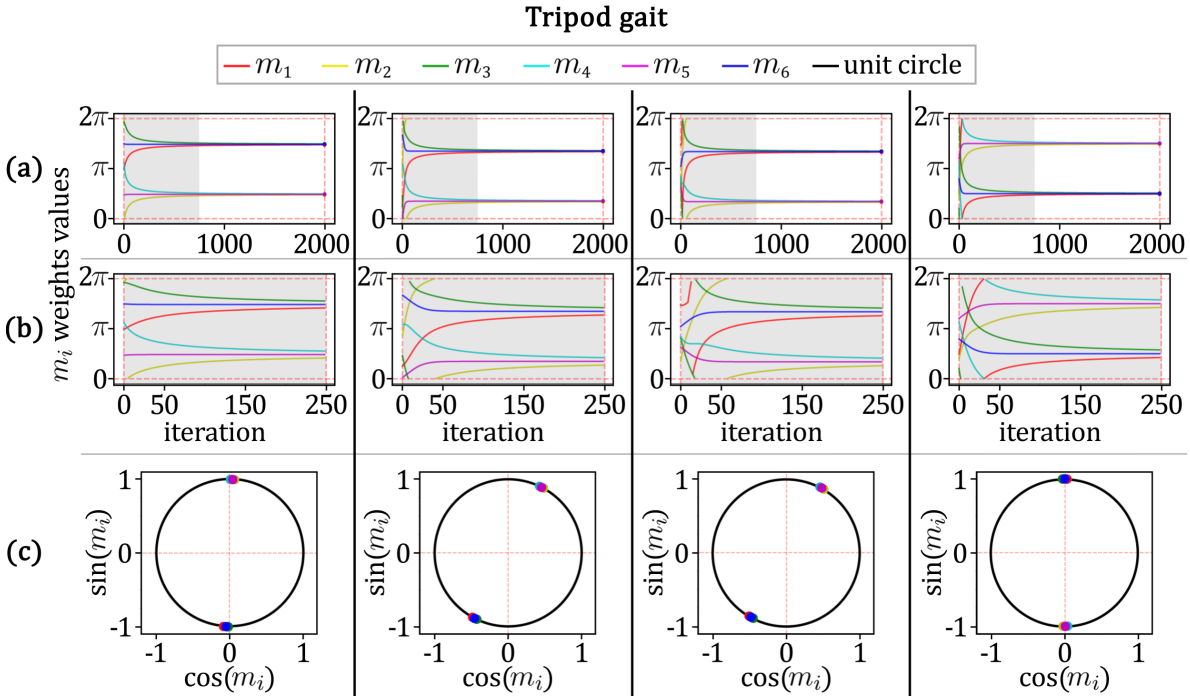

Figure 18: The weights m_i for the tripod gait. (a) Four different random initializations of the weights and their progression within the first 2000 iterations. (b) Close up of the first 250 iterations of the weights progression (the area with grey background in (a)). In both, (a) and (b), the vertical axis represents the value of the weights and the horizontal axis represents the iteration number (t = `iteration_number/100`). (c) The final weights' states (colorful dots) of the four different random initializations mapped on the cycle (black)(horizontal axis is given by $\cos(m_i)$, and the vertical axis by $y = \sin(m_i)$).

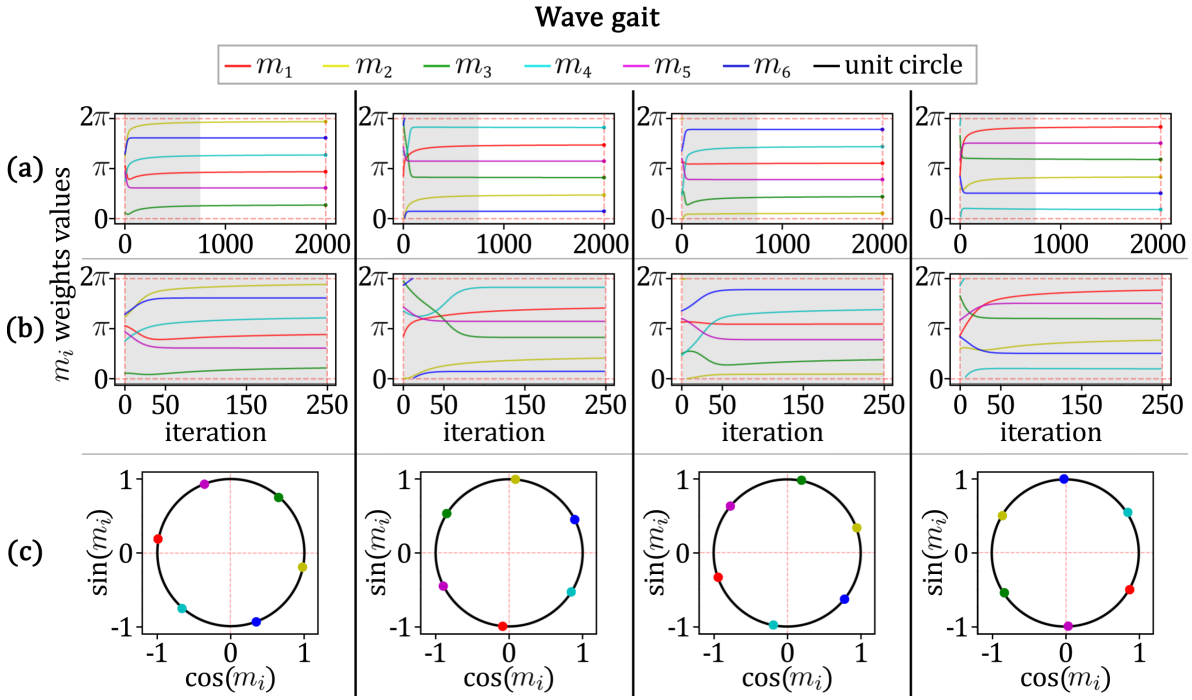

Figure 19: The weights m_i for the wave gait. (a) Four different random initializations of the weights and their progression within the first 2000 iterations. (b) Close up of the first 250 iterations of the weights progression (the area with grey background in (a)). In both, (a) and (b), the vertical axis represents the value of the weights and the horizontal axis represents the iteration number ($t = \texttt{iteration\_number}/100$). (c) The final weights' states (colorful dots) of the four different random initializations mapped on the cycle (black)(horizontal axis is given by $\cos(m_i)$, and the vertical axis by $y = \sin(m_i)$).

whose decrease is given by its differential a.[5]

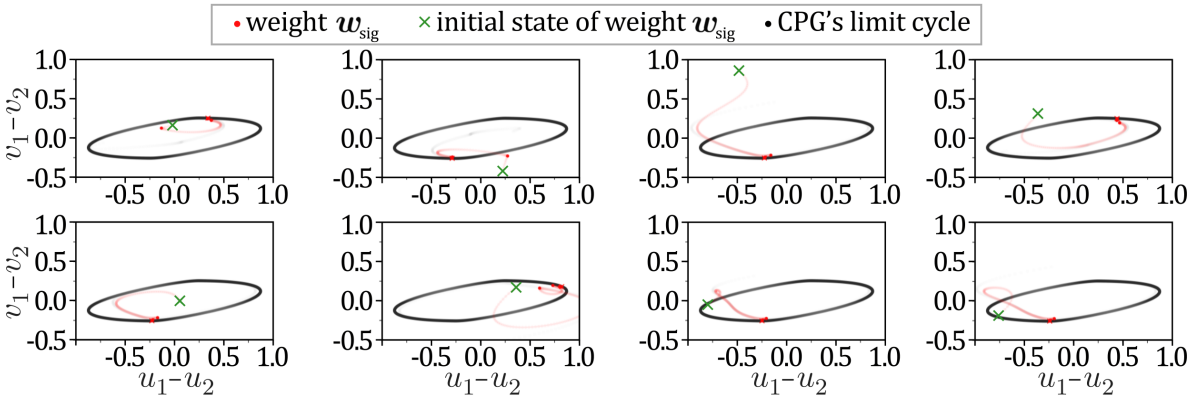

Figure 20: The plots show the progression of eight CPGs (black), where for each CPG, one weight $\boldsymbol{w}_{\text{sig}}$ (red) was randomly initialized. The place of the weight's initialization is marked by the green cross.

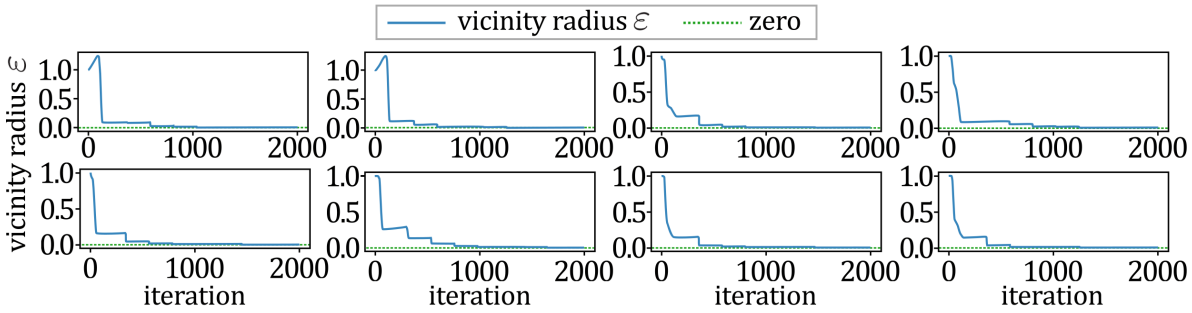

Figure 21: The plots show the progression of the ε determining the radius of the weight's dynamic vicinity for each of the respective weight $\boldsymbol{w}_{\text{sig}}$ from Fig. 20. The vertical axis represents the value of the ε, and the horizontal axis represents the iteration number ($t = \texttt{iteration\_number/100}$).

To produce the signal $\mathrm{s}(t)$ which has the same period T as the CPG, an RBF neuron depending on the weight $\boldsymbol{w}_{\text{sig}}$ is used, as described in Chapter 3 Proposed Method in section 3.2.2.1.

The CPG's state $\boldsymbol{y}$ attracts the weight $\boldsymbol{w}_{\text{sig}}$ (see Fig. 20) based on the weight's dynamic vicinity given by the vicinity's radius ε (see Eq. 43 and Fig. 21). If the weight is close enough to the CPG's limit cycle, it starts to produce the signal $\mathrm{s}(t)$ via the weight's respective RBF neuron (see Fig. 22 (a)).

The phase learning process starts, as the input signal $\hat{\mathrm{s}}(t)$ (the modified signal $\mathrm{s}(t)$, see the grey signal in Fig. 22 (b)) reaches the required value one. Until that, the differential a is set to zero, so the value of $\mathrm{p}(t)$ is constant and equal to zero. After the value one is reached by the input signal $\hat{\mathrm{s}}(t)$, the differential a initializes to a negative value a_0.

The choice of a_0 influences the learning process speed significantly. Obviously, the closer the initialization is to the optimal value, the shorter the learning process. In general, values closer to zero (e.g., -0.1) are safer to use as an initialization value. The reason is that the differential a changes if in the moment of the input signal $\hat{\mathrm{s}}(t)$ reaching value one the value of $\mathrm{p}(t)$ is not equal to zero, and the change depends on how far from the value zero the value $\mathrm{p}(t)$ is.

The value of a lowers, if the $\mathrm{p}(t)$ is greater than zero in the input signal's peak moment, and grows, if the $\mathrm{p}(t)$ is lower than zero in the input signal's peak moment.

[5]The a indirectly represents the phase's angular velocity, which is estimated as $-2\pi a$.

If the value of a is way too low (e.g., -10), the value $\mathrm{p}(t)$ is then too far from the zero at the time of input signal $\hat{\mathrm{s}}(t)$ approaching one, and the differential a can become positive, what results in unwanted behavior (this issue will be the subject of future research). Therefore, it is recommended to use a_0 closer to zero. If the a_0 value is too close to zero, it slows down the learning process, but it converges to the correct value.

The initialization of the differential a should always depend on the given input signal $\hat{\mathrm{s}}(t)$. It is a parameter that has to be tuned to achieve fast learning for the given input signal $\hat{\mathrm{s}}(t)$.

In this work, the initial value for a was set to $a_0 = -0.4$. The example of using different initial values for two different input signals is shown in Chapter 3 Proposed Method in section 3.2 in section 3.2.2.

The learning process is shown using two different signals (to show that the method works for signals of different periods), RBF neuron of the weight on the Matsuoka's neural oscillator in Fig. 11, and sinus function signal in Fig. 12. The progression of a for both training signals of all four initializations is shown in Fig. 13.

All the phase learning results from simulations in *CoppeliaSim* are similar, as they are not dependent on the particular gait pattern. Therefore, only one of the phase learning processes (with $a_0 = -0.4$) generated during the simulations is presented in this section (see Fig. 22).

In all cases, the value of a converges to the same value, which estimates the optimal slope for the function $\mathrm{p}(t)$ to estimate the given signal's phase.

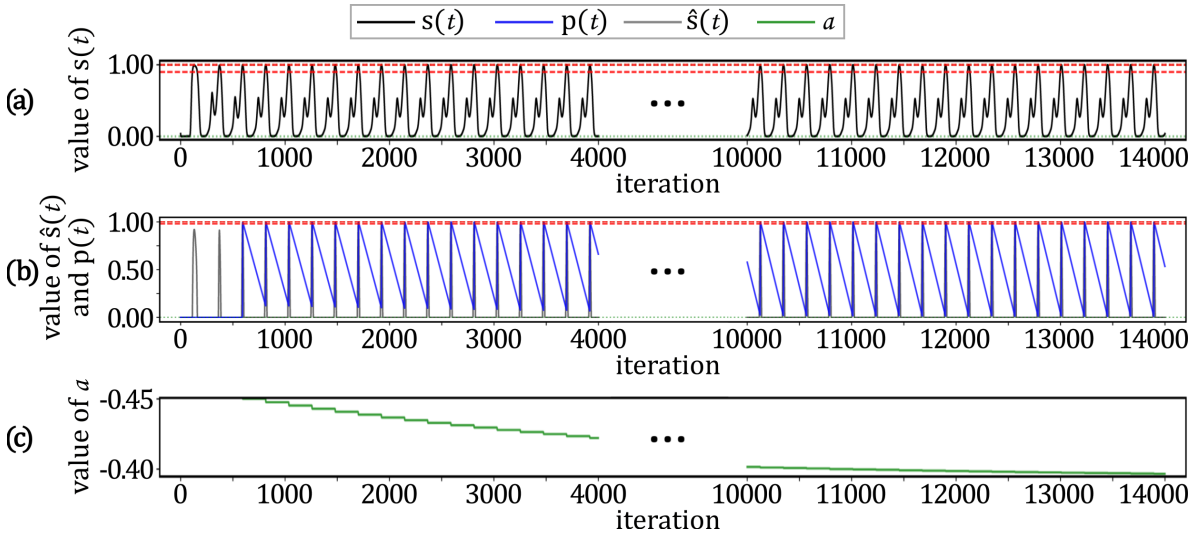

Figure 22: (a) The unmodified input signal $\mathrm{s}(t)$ (see Eq. 44) from RBF neuron. The area between the red dashed lines is the part of the signal used as an input signal $\hat{\mathrm{s}}(t)$ for the phase learning in (b), where the phase learning is shown. The blue line represents the estimation of the learned phase course $\mathrm{p}(t)$, and the grey signal $\hat{\mathrm{s}}(t)$ is the modified input signal $\mathrm{s}(t)$ from (a). In (c) the progression of the differential a (green) is shown, which determines the decline of the $\mathrm{p}(t)$ in (b). The vertical axis represents the value of the corresponding variable ($\mathrm{s}(t)$ in (a), $\hat{\mathrm{s}}(t)$ and $\mathrm{p}(t)$ in (b), and a in (c)) and the horizontal axis represents the iteration number ($t = \texttt{iteration\_number/100}$).

4.3.3 Weights on the Limit Cycle

The function $\mathrm{p}(t)$ is used to determine the CPG's phase estimation $\hat{\phi}^{\mathrm{cpg}}(t)$, as shown in Eq. 48. To organize the weights $\boldsymbol{w}_i$ (for $i = 1, \ldots, 6$) within the CPG's limit cycle the RBF neurons' signals, corresponding to the weights m_i, were used. The i-th RBF neuron fires when the estimated phase

$\hat{\phi}^{\text{cpg}}(t)$ is approaching the learned weight m_i. The i-th RBF neuron's pulse determine the moment when the CPG's state $\boldsymbol{y}$ attracts the corresponding weight $\boldsymbol{w}_i$.

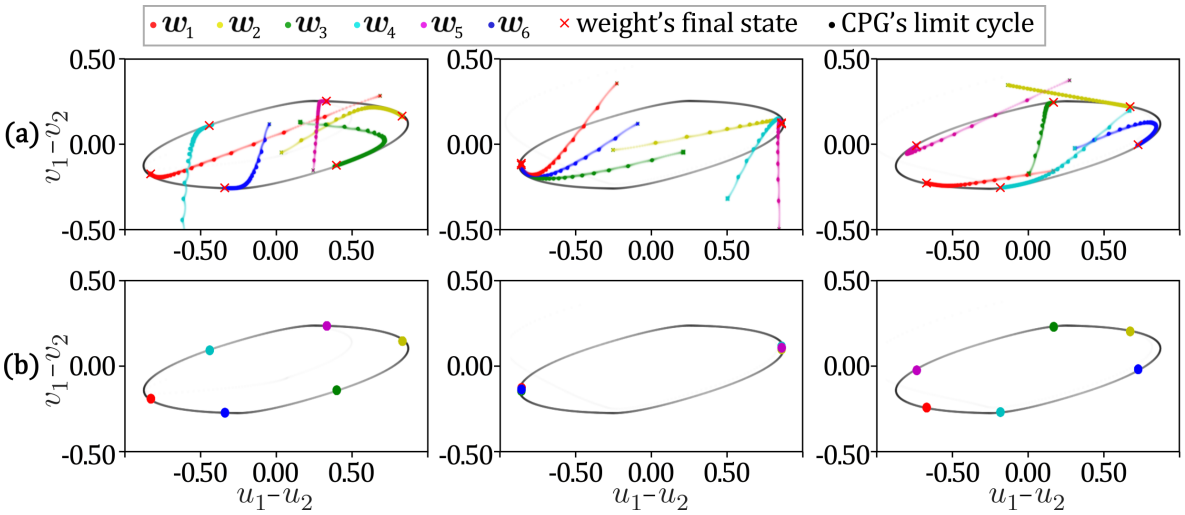

Figure 23: The figure shows the CPG and weights $\boldsymbol{w}_i$ of the PLM. The progression of the weights and the CPG is plotted in the row (a) (the upper plots). In the row (b) (the lower plots), a final state of the weights is shown. The input values of phase offset of consecutive legs were $\Delta\phi = \frac{2\pi}{3}$, $\Delta\phi = \pi$ and $\Delta\phi = \frac{\pi}{3}$, respectively, from left to right. The values correspond to the transition, tripod, and wave gait patterns, respectively, from left to right. The red crosses mark the end states of the weights in (a). Some weights are overlapping, which means that the legs corresponding to the overlapping weights move simultaneously. The colors of the weights match the colors of their respective leg from Fig. 2.

No other forces influence the weights' positions. As the CPG's state $\boldsymbol{y}$ only attracts the weights during periodically repeating time segment (determined by the RBF neurons' pulses, see Eqs. 49 and 50), the weights are attracted to one segment of the CPG's limit cycle, as shown in Fig. 23.

The produced RBF neurons' signal started as unorganized, but the correctly learned weights produced correct rhythm, as shown in Fig. 24. The signals were then used to trigger the swing of the robot's legs in the simulation (see Fig. 25, where a showcase of one gait pattern's cycle is provided for each of the given gait patterns).

The pulses for different legs have different widths. Its cause is the same, as for the method SNM (see section 4.2). RBF neurons' signal depends on the movement of the CPG's state $\boldsymbol{y}$ through the respective weight's $\boldsymbol{w}_i$ vicinity. As the CPG's state moves with various speeds through different space regions, it also spends a different amount of time in the radius of different weights. Hence, if the $\boldsymbol{y}$ moves slower around some weight, then the weight's RBF neuron's pulse is wider, then the pulse of the RBF neuron corresponding to weight, around which the $\boldsymbol{y}$ moves faster.

However, unlike the pulses generated by the SNM, all the pulses have the same amplitude (approaching value one), which signifies improvement of the signal's quality in comparison with the SNM.

4.4 Computational Time

The last two experiments were run to compare the proposed methods in terms of computational time and learning speed. The locomotion controllers for robots usually consist of many CPGs (six: one for each leg, eighteen: one for each joint, etc.). Hence, it is assumed that the learning mechanism runs on many CPGs parallelly, which increases computational time.

The methods have different computational complexity (see scheme in Fig. 4). Therefore, the

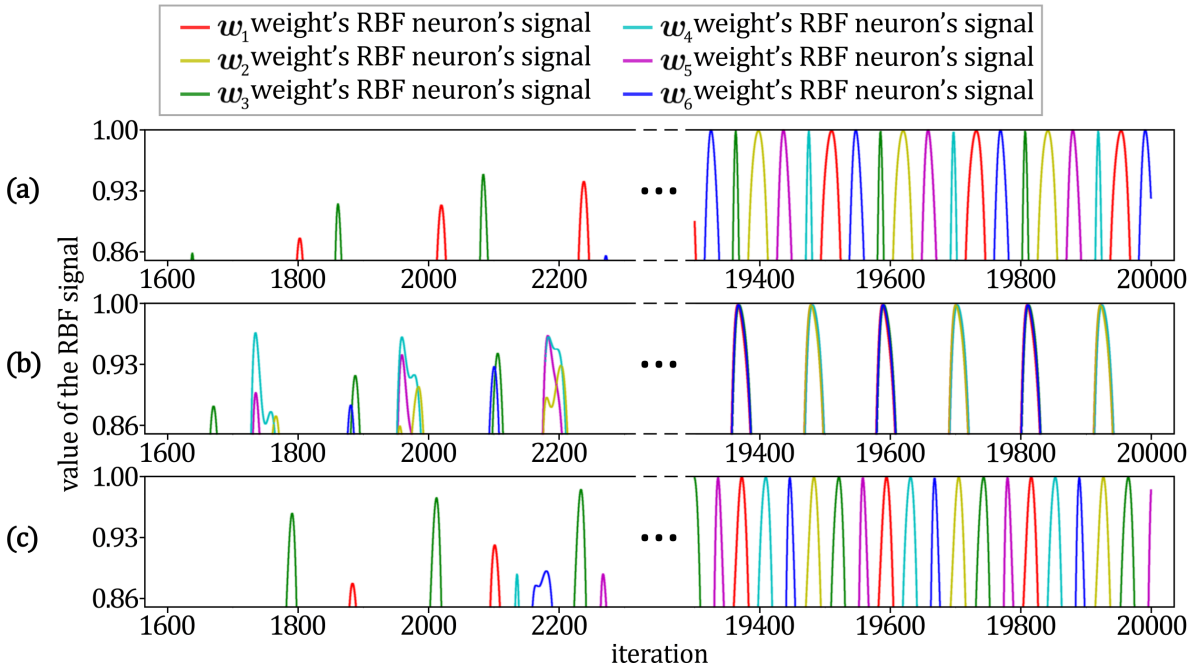

Figure 24: Plots of the generated transition (a), tripod (b), and wave (c) gait patterns' signals by the PLM. The plots show the signals generated by the RBF neurons based on the difference of the respective weight and the CPG's current state. The left plots show the generating of the signals initially, and the right plots show the generated signals at the end of the simulation. The horizontal axis represents the iteration number (t = `iteration_number/100`). The vertical axis represents the signals' value. The signals for simultaneously moving legs are overlapping in (b). Note that the colors of the signals match the colors of their respective leg from Fig. 2.

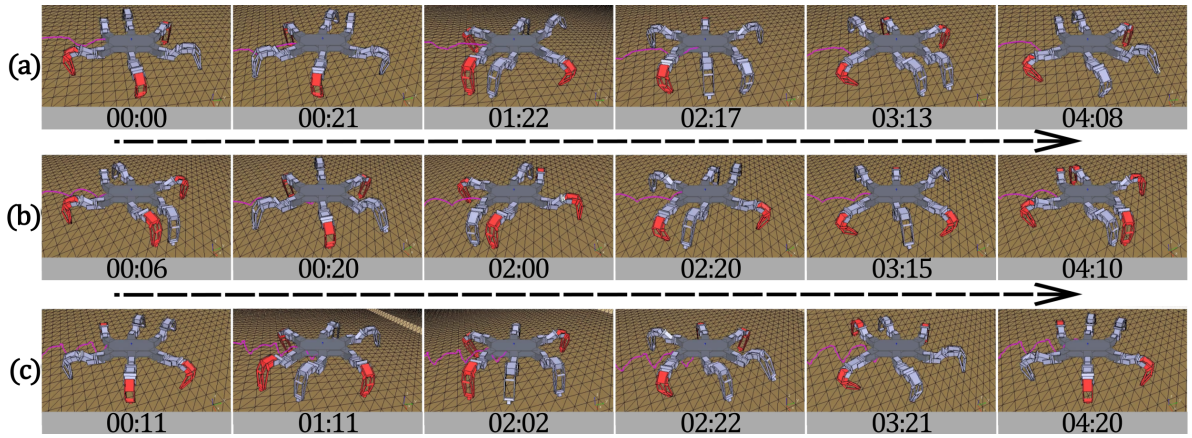

Figure 25: A showcase of applying the proposed method using the phase learning in the *CoppeliaSim* simulator. Each sequence of images shows six chosen frames from one cycle of each of the gait patterns. The images contain a timestamp. The timestamp measures the time since the beginning of the showed cycle in format *seconds:frames*, where each second consists of 25 frames. The gaits are transition gait in (a), tripod gait in (b), and wave gait in (c). The arrows between the image sequences indicate both the direction of the robot's locomotion and the individual frames' time passage.

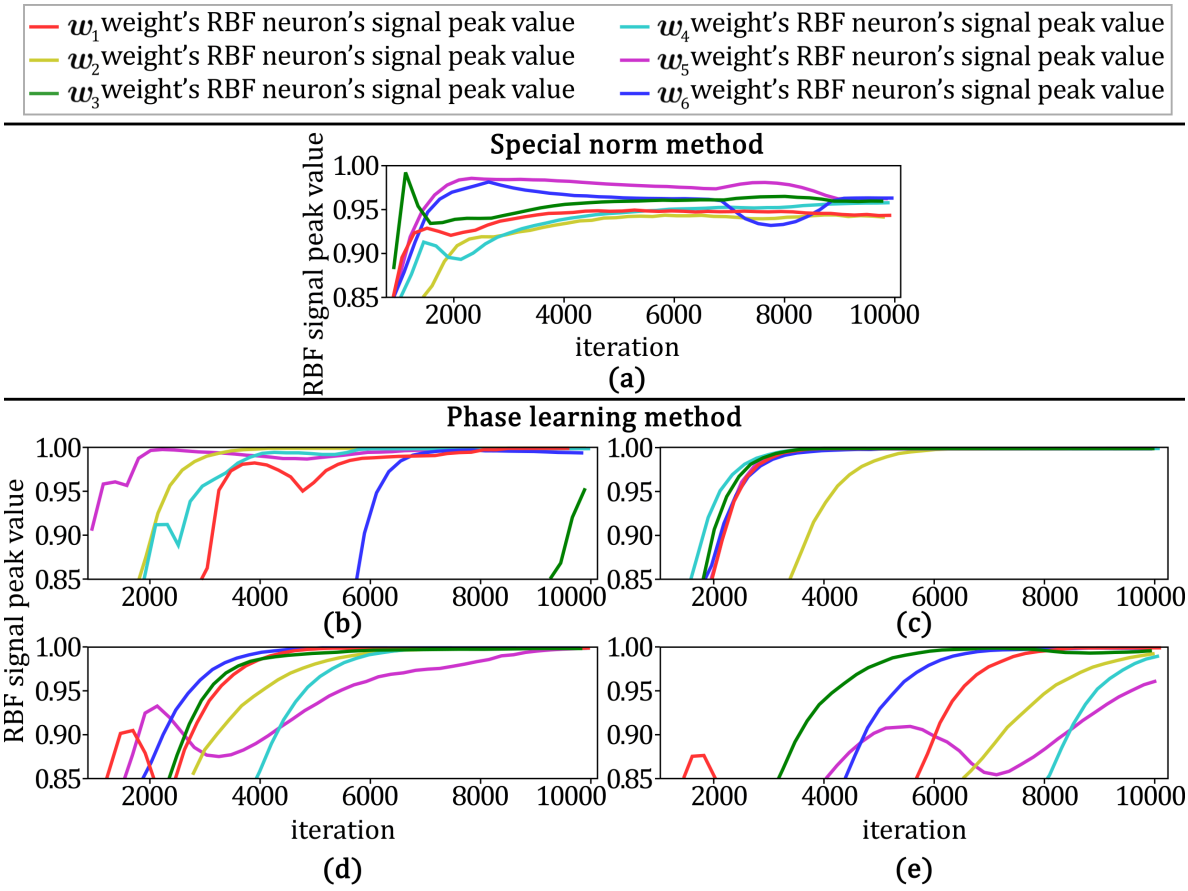

Figure 26: The figure shows the comparison of progression of the signals generated by the $\boldsymbol{w}_i$ weights' corresponding RBF neurons by both methods for the transition gait. Each line represents the value of the corresponding signal's peak. The learning process is finished when none of the lines changes its value (if the lines have converged). In the upper part (a), the method using special norm is plotted. In the lower part (b), (c), (d), and (e), the results of the method using the phase learning with four different a initializations are shown. For all the plots, the same randomly initialized values for the weights $\boldsymbol{w}_i$ were used, and for the plots (b), (c), (d), and (e), the same randomly initialized values for weights m_i were used. The plot is meant to visualize the methods' comparison with the identical initialization of the weights. The initial values of a for plots (b), (c), (d) and (e) are equal to -0.1, -0.4, -0.7 and -1, respectively. The vertical axis represents the value of the signals and the horizontal axis represents the number of iterations ($t = \texttt{iteration\_number}/100$). The colors of the lines correspond to the weights of their respective legs as shown in Fig. 2. Note that the signals are dependent on the random initialization. Therefore, the plots are just an example of how the progress can look like.

method		4 CPGs	8 CPGs	16 CPGs	32 CPGs	64 CPGs
SNM	mean [s]	27.434	49.792	92.849	179.184	351.325
	variance [s^2]	0.172	0.205	0.035	0.292	2.506
PLM	mean [s]	9.425	9.702	9.886	10.582	11.329
	variance [s^2]	0.115	0.001	0.001	0.002	0.009

Table 6: Computational time of simulating $n \in \{4, 8, 16, 32, 64\}$ CPGs for 20000 iterations comparison. The experiment was run ten times for each n. The results, shown in the table, for the given n are obtained by computing the mean and variance of the ten runs.

experiment, comparing the required computational time for running the learning process on more CPGs simultaneously, was proposed.

The learning process of both methods ran for 20000 iterations for various counts of simultaneously running CPGs (4, 8, 16, 32 and 64), where each test instance was run ten times. The results are presented in Tab. 6 as the means of the ten runs for each instance (in seconds).

The second comparing experiment was meant to show that the learning speed is dependent on the random initialization. Therefore not much can be said about the comparison of the methods' learning speed.

Both methods were given the same initial values for the weights $\boldsymbol{w}_i$. For the SNM, only one test instance was run, as it depends only on the initialization of the weights $\boldsymbol{w}_i$. For the PLM, four instances were run, where the weights m_i were initialized identically for all of them, but the test instances differed in the initialization of the differential a. The initialization value for a was $a_0 \in \{-0.1, -0.4, -0.7, -1\}$. The results, shown in Fig. 26, are further discussed in the Chapter 5 Discussion.

Chapter 5
Discussion

The discussion consists of multiple sections. The goal and both proposed methods are briefly reminded in section 5.1. Although both methods accomplished the given task, the Special Norm based Method (SNM) has many drawbacks, discussed in section 5.2. Therefore, the Phase Learning based Method (PLM) was designed to eliminate the drawbacks of the SNM, as discussed in section 5.3. The discussion about the initialization effect on the methods is provided in section 5.4. The absence of the ripple gait[6] among the generated gait patterns is discussed in section 5.5. Finally, future work possibilities are proposed in section 5.6, which concludes this chapter.

5.1 Goal and Methods' Mechanism

The gait pattern is generated by moving the robot's legs in the correct order. The CPG is used to provide a periodic rhythm for the robot's locomotion. To represent the legs' activity, the weights $\boldsymbol{w}_i \in \mathbb{R}^{\dim(\text{cpg})}$ are introduced to be spread around the CPG's limit cycle to produce signals via their respective RBF neurons if the CPG's state is close enough to the corresponding weight. The aim of this work was to organize the weights around the CPG's limit cycle to produce a stable gait pattern using the coordination rules, introduced in Chapter 2 Problem Statement, to determine the relations between the weights, which enables them to interact and self-organize.

To accomplish the goal, two methods were proposed. Both methods are meant to produce the same result. However, each of them solves the task with a different approach.

The SNM handles both problems at once. It takes the weights, organizes them within themselves, and at the same time keeps them close to the CPG's limit cycle.

The PLM separates the two tasks into independent parallelly running processes. It uses additional weights m_i (each m_i weight is related to the $\boldsymbol{w}_i$ weight), which are organized within the phase. To determine the $\boldsymbol{w}_i$ weight's position on the CPG's limit cycle, the weight is assigned its corresponding limit cycle point, using the estimation of the unknown CPG's phase as each point on the CPG's limit cycle corresponds to some phase.

For the SNM, the experiments demonstrated the forces of attraction to the CPG's limit cycle and the forces initiated by the weights' interaction are able to place the weights $\boldsymbol{w}_i$ near enough to the CPG's limit cycle to produce a signal with distinct peaks for triggering the legs' actions.

The experiments results for PLM showed a working mechanism of phase estimation and its use to map the estimated phase values to the CPG's states $\boldsymbol{y}(t)$, which can be used for any task of assigning a phase-related action to a specific CPG's state. The experiments with PLM also proved that the proposed mechanism for self-organizing the weights within the phase (assigning the weights values from interval $[0, 2\pi)$ while fulfilling the coordination rules) is relatively fast, deterministic, and providing wanted configurations.

The experiments showed how the proposed mechanisms, organizing the weights around the CPG's limit cycle, work. For all the given phase offsets ($\Delta\phi \in \{\pi/3, 2\pi/3, \pi\}$), both methods generated correct corresponding gait patterns (wave gait, transition gait and tripod gait). The following sections discuss the drawbacks of the SNM and how the drawbacks are eliminated by using the PLM. Then the methods' initialization and the absence of ripple gait are discussed, followed by the proposed improvements for future work.

[6] One of the common gait patterns for hexapod walking robots.

5.2 SNM: Consequences of Using the Special Norm

The SNM uses the special norm approximating the relation between the phase offset of the CPG's states and their distance. The weights' final positions are dependent on the used special norm, as the weights' movement is given by the special norm of the weights' differences. However, the method has many drawbacks.

Firstly, the computational complexity of the learning process is high while using the special norm. Computing the norm itself requires costly operations, [7] which are frequently used during the learning process.

In the SNM, all the weights $\boldsymbol{w}_i$ are restricted by all the other $\boldsymbol{w}_i$ weights (see Fig. 4 (a)) to reduce the number of possible configurations in the multidimensional space (as illustrated in Fig. 10). Every two weights maintain the given distance between each other, given by their phase offset and costly computed by the special norm of their difference.

Hence, the time, needed for the SNM computation, grows significantly with the greater number of simultaneously simulated CPGs. The Tab. 6, shows the computational time of the two methods while simulating 4, 8, 16, 32 and 64 CPG's simultaneously. Usually, 6, 18, or more CPGs are used at the same time to produce a gait pattern for a hexapod walking robot.

The computational units deployed on robots are limited by performing many other tasks, not only the locomotion computation. Therefore, the growth of computational complexity is not negligible.

Secondly, even though the amount of restrictions is large, the method can result into different legs' coordination. Unfortunately, not all of the results do fully correspond to the given coordination rules.

For instance, consider three weights being all on the limit cycle and should maintain the given order (first, second, third) according to the limit cycle's direction. Even after being restricted to move only within the unknown limit cycle, there are still two possible configurations, which are both valid in terms of the given restrictions of the given distance maintaining. Both orders, (first, second, third) and (third, second, first), on the CPG's limit cycle are valid in terms of the given distance maintaining restrictions.

The problem of multiple configurations, fulfilling the given restrictions, is shown in the results of the SNM. Specifically, the generated transition gait has the weights in reversed order (see Fig. 14) in comparison to the illustrated example in Fig. 2. The produced gait is stable. Nevertheless, it does not fully correspond to the coordination rules. Rules 1. and 2. are kept reversely. Usually, the middle leg is the consecutive leg of the hind leg. However, the weights' reversed order makes the hind leg the consecutive leg of the middle leg. Therefore, the legs' movement does not travel from hind legs to front legs, but in reverse from front legs to hind legs.

Thirdly, the approximation defined by the special norm may not be precise enough. The reason why the weights in Fig. 14 are so far from the limit cycle (for the transition and wave gaits) is, that the unit sphere, given by special norm, only approximates the limit cycle. Therefore, the force, determined by the special norm, pushes the weights farther from each other in some direction than the dimension of the CPG's limit cycle in that direction is.

Moreover, the weights' distance from the limit cycle determines the signal, which the weights' corresponding RBF neurons generate. As can be seen in Fig. 15, the peaks' values vary for individual weights' signals. The peaks' amplitudes variance could cause issues for some locomotion controllers while mapping the signals to the legs' movements.

Although, the SNM results could be improved by fine-tuning the parameters or finding better special norm formula, I decided to create different approach, which would eliminate the drawbacks by its design. The result is the PLM.

[7] like the dot products of matrix and two vectors, see Appendix A Special Norm Finder, where the implementation of the special norm $\|-\|_{\text{cpg}}$ is described

5.3 PLM: Eliminating the SNM's Drawbacks

The PLM was designed with SNM's drawbacks in mind. Most of the SNM's disadvantages were caused by usage of the special norm to approximate the distance and phase offset relation, which defined a connection between CPG's phase and CPG's states. By using the special norm, the SNM simultaneously tackled both problems of organizing the weights within the period and keeping the weights close to the limit cycle. The PLM decomposes the two problems into two separate tasks and eliminates the three most significant SNM's drawbacks.[8]

Firstly, the computational complexity was lowered. Instead of interactions between all weights, only a few interactions are necessary while organizing the weights within the phase, as shown in Fig. 4 (b). As the weights m_i, used for being organized within the phase, are only one-dimensional, it reduces the computational complexity of computing their distances[9].

Secondly, the PLM brings the advantage of easily definable order of the weights on the limit cycle (as shown in Fig. 9 (a)) by using only one-dimensional weights to be organized within the phase. The definable order solves the problem of the weights' final state being configurated reversely within the limit cycle, as mentioned in the section 5.2 as a drawback of the SNM.

Thirdly, because the weights $\boldsymbol{w}_i$ (for $i = 1, \ldots, 6$) are influenced only by the attractive force of the CPG's current state in the PLM, the weights are close to the limit cycle. Therefore, all the peaks in the RBF neurons' signal are similar and approaching value one. See the difference between the Figs. 15 and 24 of the RBF neuron's signal of transition and wave gaits.

Overall, the PLM eliminates all the SNM's drawbacks introduced in section 5.2. However, both methods are dependent on their initial state.

5.4 Effect of Initialization

The common issue of both proposed methods is their dependency on the initialization, as the initial state strongly influences the methods' learning speed.

The SNM is dependent only on the initialization of the weights $\boldsymbol{w}_i$. If they are initialized far from the limit cycle and in a very different order than is required, the learning process can last for about 20000 iterations or more. On the other hand, if the initialization is close to the final result, the learning process can be swift. As the initialization is random, no estimation can be done.

For instance, see the Fig. 27 the *Special norm method* column. For transition gait, the gait is relatively stable around 4000-th iteration, and after that, the weights are only being fine-tuned. Nevertheless, for the wave gait, the gait is becoming stable around 10000-th iteration.

The PLM depends mainly on two initialization factors. The differential a, influencing the phase learning, and the initialization of the weights $\boldsymbol{w}_i$.

The differential a, indirectly representing the phase's angular velocity, and its initialization value's influence is already described in section 4.3.2. The initialization of the weights $\boldsymbol{w}_i$ influences the learning speed by how far from the final positions the weights are initialized. The weights are attracted only for a short time segment during each period. Therefore, if the weights are initialized far from their final positions, it can take many cycles for them to get to the final positions.

The third initialization factor of the PLM is the initialization of the weights m_i. However, as can be seen in Figs. 17, 18 and 19, organizing the weights m_i within the interval $[0, 2\pi)$ stabilizes around 2000-th iteration. That is faster than the phase learning process. Therefore, it does not slow down the process of organizing the weights $\boldsymbol{w}_i$ as the weights m_i are already organized correctly at the beginning of the weights $\boldsymbol{w}_i$ being organized around the limit cycle.

[8] The three drawbacks, discussed in section 5.2, are computational complexity, various possible outputs and the weights' RBF neurons producing signals of different amplitudes.

[9] The SNM computes the special norm (i.e., the dot products of matrix and two vectors) of the difference of two vectors, and the PLM only applies the function from Eq. 29 to a difference of two real numbers.

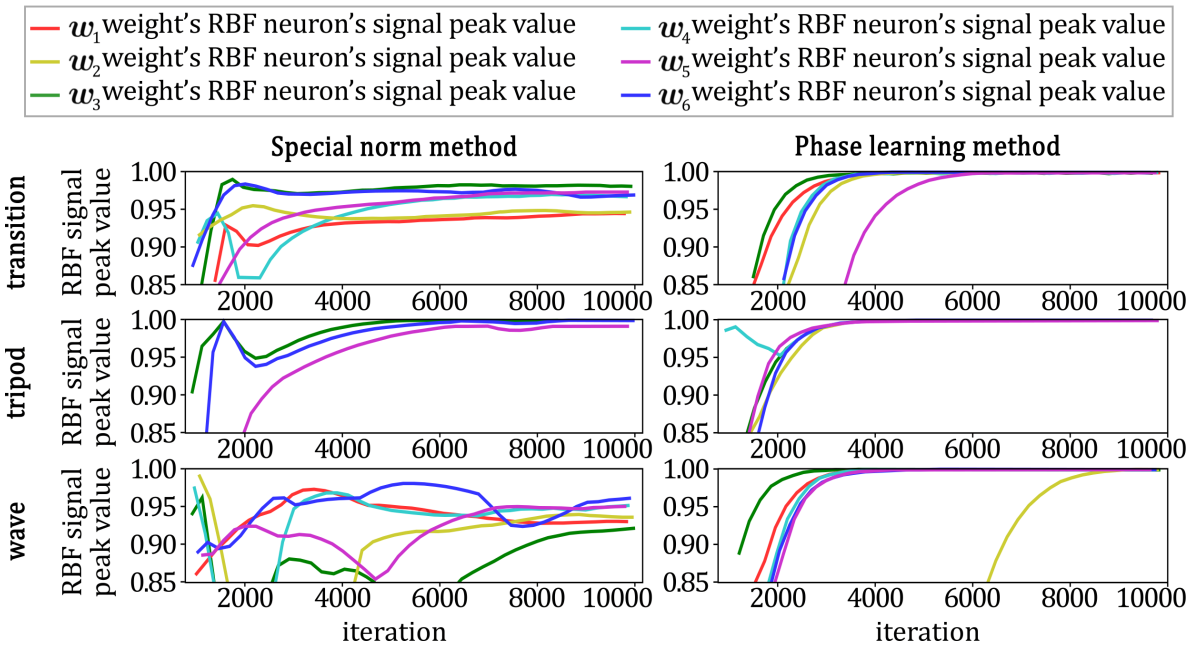

Figure 27: The figure shows the comparison of progression of the signals generated by the w_i weights' corresponding RBF neurons by both methods for each gait. Each line represents the value of the respective signal's peak. The learning process is finished when none of the lines changes its value (if the lines have converged). On the left-hand side are shown the peaks of the signals generated by the method using special norm. The results of the method using phase learning are shown on the right-hand side. The plots visualize the transition gait, tripod gait, and wave gait from top to bottom, respectively. The vertical axis represents the value of the signals and the horizontal axis represents the number of iterations ($t = \texttt{iteration\_number}/100$). The colors of the lines correspond to the w_i weights' respective legs as shown in Fig. 2. Note that the signals are dependent on the random initialization. Therefore, the plots are just an example of how the progress can look like.

To compare the learning speed of both methods, additional experiments were run, where the same randomly initialized weights $\boldsymbol{w}_i$ were used for each CPG (see Fig. 26). Observing Figs. 27 and 26, the observation is following: iteration numbers of gait becoming stable for the SNM are approximately (4000, 8000, 10000, 6000) and for the PLM the iteration numbers are approximately (6000, 4000, 10000, 6000) for a initialized to value $a_0 = -0.4$. The variances of the values are significant and the values are similar for both methods. Therefore, the learning speed comparison does not provide much information.

The duration of the learning process is given by the initialization of the methods' parameters, and the given phase offset determines the resulting gait pattern. Nevertheless, two common gait patterns are given by the same phase offset, which resolves in the methods generating only one of them, as discussed in the following section.

5.5 The Ripple Gait

One of the common hexapod gaits is called ripple gait, which is similar to the transition gait, as the phase offset $\Delta\phi$ of consecutive legs is for both the gaits equal to $2\pi/3$. However, the coordination of the legs for the ripple gait is a bit different than the transition gait. In comparison with the transition gait, all legs' activity from one of the sides of the robot's body is shifted by $\pi/3$, resolving in the simultaneous action of two legs, one from each side of the robot's body, at the same time.

The following values are only an example of a valid ripple gait configuration, as more configurations are valid: $\phi_4 = 0$, $\phi_6 = 2\pi/3$, $\phi_2 = 4\pi/3$, $\phi_3 = 4\pi/3$, $\phi_5 = 0$ and $\phi_1 = 2\pi/3$ (see Tab. 5 for comparison with the transition gait values). The pairs of legs, undergoing the swing simultaneously, are hind right and middle left, middle right and front left, and front right and hind left. To generate the ripple gait by the methods, the value $\Delta\phi = 2\pi/3$ has to be the input.

However, the proposed methods generate the transition gait for the input value of $2\pi/3$. Therefore, I conclude that the coordination rules, presented in Chapter 2 Problem Statement, tend to prefer the transition gait to the ripple gait.

The difference between the transition gait and the ripple gait is in the phase offset of the contralateral legs, which is equal to π for the transition gait and $2\pi/3$ for the ripple gait. Hence, if the repulsion forces between the contralateral legs would be weaker, then the methods could possibly generate the ripple gait.

Despite that both solutions, ripple gait and transition gait, are valid solutions for the given input value $\Delta\phi = 2\pi/3$, both the proposed methods behave deterministically and always generate the transition gait.

The reason why this work's implementations of the coordination rules tend to prefer the transition gait over the ripple gait is a subject of future research, together with other model's extensions, discussed in the following section.

5.6 Future Work

There are many options for future work, which are of both kinds, directly extending the proposed model and building on the proposed model.

The possible model extensions include tuning the hyperparameters, debugging of the proposed mechanisms (for instance, the unwanted behavior caused by $a > 0$ or adjusting the RBF neurons to produce pulses of identical width for all weights), finding heuristics for the initialization of randomly initialized parameters, or extending the model to be able to generate gait patterns for robots with higher (or lower) number of legs.

The robot's body on a long-term mission can eventually decay. Hence, practical future work would be to automatize the locomotion process even more, to be able to generate an efficient gait pattern, based on the coordination rules, for the robot with malfunctioning joints or limbs.

Chapter 6
Conclusion

In this work, two methods for self-organizing the legs' actions within the CPG's limit cycle to produce the given gait patterns for the hexapod walking robot were proposed, tested, simulated and compared.

Each leg was assigned a weight to be represented in the CPG's space. To organize the weights around and nearby the CPG's limit cycle, a set of dynamic rules was proposed based on the biologically inspired coordination rules introduced in [11].

Gait pattern is determined by a phase offset of consecutive legs, as described in [2]. Hence, the phase offset of consecutive legs was used as the input value to determine the desired output gait pattern.

Both methods successfully produced all given gait patterns. The method based on the special norm is straightforward, but it has many drawbacks, including high computational complexity.

The drawbacks were successfully eliminated by proposing and using the second method based on learning the CPG phase's dependency on time, which enabled to decompose the problem into two simpler tasks. Moreover, the second method provides a tool for mapping any action, given by the phase value, on the CPG's limit cycle.

In future work, I would like to improve the phase learning mechanism to be resistant against positive values of the differential a, which causes unwanted behavior, and extend the model to generate a gait patterns for differing number of legs. Also, if reasonably changed from random to deterministic, the initialization process could improve the learning speed of the whole process. Another improvement could be made to the RBF neurons to make the width of its pulses equally wide for weights on different places of the CPG's limit cycle.

References

[1] A. Ayali, A. Borgmann, A. Büschges, E. Couzin-Fuchs, S. Daun-Gruhn, and P. Holmes, "The comparative investigation of the stick insect and cockroach models in the study of insect locomotion," *Current Opinion in Insect Science*, vol. 12, pp. 1–10, 2015, neuroscience * Special Section: Insect conservation. [Online]. Available: https://doi.org/10.1016/j.cois.2015.07.004

[2] 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. [Online]. Available: https://doi.org/10.1007/s10846-012-9661-1

[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, robotics and Neuroscience. [Online]. Available: https://doi.org/10.1016/j.neunet.2008.03.014

[4] J. Yu, M. Tan, J. Chen, and J. Zhang, "A survey on cpg-inspired control models and system implementation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 441–456, 2014. [Online]. Available: https://doi.org/10.1109/TNNLS.2013.2280596

[5] L. Xu, W. Liu, Z. Wang, and W. Xu, "Gait planning method of a hexapod robot based on the central pattern generators: Simulation and experiment," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2013, pp. 698–703. [Online]. Available: https://doi.org/10.1109/ROBIO.2013.6739542

[6] H. Yu, W. Guo, J. Deng, M. Li, and H. Cai, "A cpg-based locomotion control architecture for hexapod robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 5615–5621. [Online]. Available: https://doi.org/10.1109/IROS.2013.6697170

[7] L. Bai, H. Hu, X. Chen, Y. Sun, C. Ma, and Y. Zhong, "Cpg-based gait generation of the curved-leg hexapod robot with smooth gait transition," *Sensors*, vol. 19, no. 17, 2019. [Online]. Available: https://doi.org/10.3390/s19173705

[8] 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. [Online]. Available: https://doi.org/10.1109/TCST.2017.2692727

[9] W. Ouyang, H. Chi, J. Pang, W. Liang, and Q. Ren, "Adaptive locomotion control of a hexapod robot via bio-inspired learning," *Frontiers in Neurorobotics*, vol. 15, p. 1, 2021. [Online]. Available: https://doi.org/10.3389/fnbot.2021.627157

[10] H. Chung, C. Hou, and S. Hsu, "A cpg-inspired controller for a hexapod robot with adaptive walking," in *CACS International Automatic Control Conference (CACS)*, 2014, pp. 117–121. [Online]. Available: https://doi.org/10.1109/CACS.2014.7097173

[11] V. Dürr, J. Schmitz, and H. Cruse, "Behaviour-based modelling of hexapod locomotion: linking biology and technical application," *Arthropod Structure & Development*, vol. 33, no. 3, pp. 237–250, 2004, arthropod Locomotion Systems: from Biological Materials and Systems to Robotics. [Online]. Available: https://doi.org/10.1016/j.asd.2004.05.004

[12] K. Matsuoka, “Mechanisms of frequency and pattern control in the neural rhythm generators,” *Biological Cybernetics*, vol. 56, no. 5, pp. 345–353, Jul 1987. [Online]. Available: https://doi.org/10.1007/BF00319514

[13] R. Szadkowski and J. Faigl, “Neurodynamic sensory-motor phase binding for multi-legged walking robots,” in *International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8. [Online]. Available: https://doi.org/10.1109/IJCNN48605.2020.9207507

Appendix A
Special Norm Finder

The work consists of two methods for self-organizing the weights around the CPG's limit cycle. The first proposed method, proposed in Chapter 3 Method in section 3.1 is based on the relation between phase offset of two consecutive legs and the distance of their relevant weights. However, to use the relation, a special norm $\|-\|_{\text{cpg}}$ has to be used to approximate the distances of the CPG's limit cycle's points in relation to the CPG's phase. The CPG's limit cycle is a closed trajectory, which, for some CPGs, lies nearby the surface of some ellipsoid. Therefore, it was proposed that the norm, which transforms the ellipsoid into the unit sphere, can be used to approximate the relation between the CPG's phase and the distance of the CPG's limit cycle's points.

In this appendix, the ellipsoid norm, which transforms the ellipsoid into a unit sphere, is introduced in section A.1. Then one method for finding a suitable ellipsoid, to approximate the relation between the CPG's phase and the distance of the CPG's limit cycle's points, is proposed in section A.2. The proposed method for the ellipsoid finding is the method, which was used to run the simulations presented in this work.

A.1 Norm Given by the Ellipsoid's Semi-Axes

The norm is defined with the use of a scalar product in this section. Then the relation between the ellipsoid's semi-axis and the ellipsoid norm is introduced, where the ellipsoid norm is the norm, which considers the ellipsoid as a unit sphere.

For space $\mathbb{R}^n$, the scalar product is defined as a function:

$$\langle \boldsymbol{x}|\boldsymbol{y}\rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \tag{51}$$

where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, which meets the following requirements for all vectors $\boldsymbol{x}$ and $\boldsymbol{y}$:

- $\langle \boldsymbol{x}|\boldsymbol{y}\rangle = \langle \boldsymbol{y}|\boldsymbol{x}\rangle$
- $\langle \boldsymbol{x}|-\rangle : \mathbb{R}^n \to \mathbb{R}$ is a linear mapping
- $\langle \boldsymbol{x}|\boldsymbol{x}\rangle \geq 0, \langle \boldsymbol{x}|\boldsymbol{x}\rangle = 0 \Leftrightarrow x_i = 0$ for $i = 1, 2, \ldots n$,

where each x_i is a element of the vector $\boldsymbol{x}$ (i.e., $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$). The scalar product is usually given as:

$$\langle \boldsymbol{x}|\boldsymbol{y}\rangle \coloneqq \boldsymbol{x}\boldsymbol{G}\boldsymbol{y}, \tag{52}$$

where $\boldsymbol{G}$ is a positive definite matrix. The scalar product invokes the norm:

$$\|\boldsymbol{x}\| \coloneqq \sqrt{\langle \boldsymbol{x}|\boldsymbol{x}\rangle}, \tag{53}$$

which needs to meet three requirements:

- $\|\boldsymbol{x}\| \geq 0, \|\boldsymbol{x}\| = 0 \Leftrightarrow x_i = 0$ for $i = 1, 2, \ldots n$
- $\|a\boldsymbol{x}\| = |a| \, \|\boldsymbol{x}\|$, where a is scalar
- $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ (i.e., the norm preserves the triangle inequality)

The unit sphere is a shape in n dimensions, for which each of its surface points has the norm equal to one. Therefore the unit sphere does not have to be necessarily shaped like a sphere. The norms $\|-\|_1$ and $\|-\|_\infty$ generate unit spheres, which in two-dimensional space are squares, for instance.

To consider the ellipsoid as a unit sphere, the corresponding norm, which considers every norm of a point on the ellipsoid's surface equal to one, has to be found. Without loss of generality, consider only the ellipsoids with their center placed in the origin of the space for now (i.e., $\boldsymbol{c} = (0, 0, \ldots, 0)$, all other ellipsoids are only shifted in space, but the properties are the same). Ellipsoid is defined by its semi-axes, which define the ellipsoid's size in the directions of the axes. As all the points of the ellipsoid's surface are required to have their norm equal to one (i.e., the distance of the points from the center of the ellipsoid is equal to one), the semi-axes vectors should also have their norm equal to one. For each ellipsoid's dimension, one semi-axis is required (i.e., n-dimensional ellipsoid has n semi-axes). Hence, the semi-axes are considered as the basis of the space where the ellipsoid exists. Let the semi-axes be denoted as $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n \in \mathbb{R}_n$. Then the matrix $\boldsymbol{B}$ is defined as:

$$\boldsymbol{B} \coloneqq (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n)\,, \tag{54}$$

where the $\boldsymbol{b}_i$ for $i = 1, 2, \ldots n$ are column vectors representing the ellipsoid's semi-axes. As the matrix $\boldsymbol{B}$ is a matrix of the ellipsoid's basis (i.e., the ellipsoid's semi-axes), its inverse matrix $\boldsymbol{B}^{-1}$ is a matrix, which transforms the ellipsoid's semi-axes into a standard basis vectors (i.e., vectors in format $\boldsymbol{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$, where the value one is on the i-th position) whose Euclidean norm is equal to one (i.e., $\|\boldsymbol{e}_i\|_2 = 1$). That is the searched transformation, which "*sees*" the semi-axes as vectors with norm equal to one. Therefore, to compute the norm of vector $\boldsymbol{v}$, while considering the vector being in the ellipsoid's space (defined by the ellipsoid's basis), we can simply transform the vector $\boldsymbol{v}$ from the ellipsoid's basis to standard basis to gain its transformed version (i.e., vector $\hat{\boldsymbol{v}}$), and then compute its Euclidean norm:

$$\hat{\boldsymbol{v}} \coloneqq \boldsymbol{B}^{-1}\boldsymbol{v}, \tag{55}$$

$$\|\hat{\boldsymbol{v}}\|_2 = \sqrt{\hat{\boldsymbol{v}}^T\hat{\boldsymbol{v}}} = \sqrt{\left(\boldsymbol{B}^{-1}\boldsymbol{v}\right)^T\boldsymbol{B}^{-1}\boldsymbol{v}} = \sqrt{\boldsymbol{v}^T\left(\boldsymbol{B}^{-1}\right)^T\boldsymbol{B}^{-1}\boldsymbol{v}} = \sqrt{\boldsymbol{v}^T\boldsymbol{G}_\mathrm{e}\boldsymbol{v}}, \tag{56}$$

where matrix $\boldsymbol{G}_\mathrm{e} \coloneqq \left(\boldsymbol{B}^{-1}\right)^T\boldsymbol{B}^{-1}$. Therefore, the norm of the vectors, given in the ellipsoid's basis, is given as:

$$\sqrt{\boldsymbol{v}^T\boldsymbol{G}_\mathrm{e}\boldsymbol{v}} \coloneqq \|\boldsymbol{v}\|_\mathrm{e}\,, \tag{57}$$

where the $\|-\|_\mathrm{e}$ denotes the norm invoked by the ellipsoid's basis. Note that the norm also provides the definition of the scalar product for the ellipsoid's space:

$$\langle \boldsymbol{x}|\boldsymbol{y}\rangle \coloneqq \boldsymbol{x}^T\boldsymbol{G}_\mathrm{e}\boldsymbol{y}. \tag{58}$$

In this section, the norm, which for all the ellipsoid's surface points gives value one, was presented, which is used in the next section to determine the special norm invoked by the shape of the CPG's limit cycle.

A.2 Ellipsoid Finder and Special Norm

The previous section introduced the norm, which measures all of the ellipsoid's surface points' norm as one. In this section the method from the previous section is used to gain the special norm for the given CPG. The CPG's limit cycle is a closed trajectory $l \subset \mathbb{R}^{\dim(\mathrm{cpg})}$ consisting of points $\boldsymbol{l} \in l$, which for some CPGs could be approximated by a closed curve on the surface of some ellipsoid. The limit cycle consists of the CPG's states $\boldsymbol{y}(t)$. In this section, a method for finding an ellipsoid, whose surface is close to all states of the CPG's limit cycle, is proposed.

The states of the CPG's limit cycle should be as close as possible to the searched ellipsoid's surface. Therefore, one possible solution can be an ellipsoid, which contains all the states inside its volume and has the minimal possible size (i.e., the minimal ellipsoid wrapper of the limit cycle). The following solution approximates the idea of the minimal wrapper.

The limit cycle's point, which has the greatest Euclidean norm, has to be on the ellipsoid's surface. Hence, the point with the maximum Euclidean norm gives the first ellipsoid's semi-axis.

Because the point with the second greatest norm would be a point very close to the first (as the points are the points of a curve), it can not be chosen as the second semi-axis. The semi-axis has to be perpendicular to each other. Therefore, I propose to reject all the points by the first semi-axis, which is the projection of all the points to the linear subspace, which is perpendicular to the first semi-axis. Hence, all the points are now perpendicular to the first semi-axis. I propose to repeat the process for the new set of points (the rejected original points), find the one with the greatest Euclidean norm, and make it the second semi-axis. Then again reject all the remaining points by it.

The process repeats, until there is $n = \dim(\text{cpg})$ semi-axes chosen. The algorithm is shown as pseudocode in Algorithm 1, where the matrix $\boldsymbol{P}$ is a matrix of rejection by the vector $\boldsymbol{b}_i$ (i.e., the

Algorithm 1 Find the ellipsoid's basis: $\boldsymbol{B} = (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n)$

Require: $\boldsymbol{l}_j \in l$; $l \subset \mathbb{R}^{\dim(\text{cpg})}$; $m \geq n$; if $k \neq j$ then $\boldsymbol{l}_j \neq \boldsymbol{l}_k$ for $k, j = 1, 2, \ldots, m$
  $\boldsymbol{B} \coloneqq ()$
  $L \coloneqq (\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_m)$
  for i in $(1, 2, \ldots, n)$ **do**
    $\boldsymbol{b}_i = \arg\max_{\boldsymbol{l} \in L} \|\boldsymbol{l}\|_2$
    append $\boldsymbol{b}_i$ to $\boldsymbol{B}$
    $\boldsymbol{P} \coloneqq \boldsymbol{b}_i \cdot \left(\frac{1}{\boldsymbol{b}_i^T \boldsymbol{b}_i} \boldsymbol{b}_i \right)^T$
    $\boldsymbol{l}_j \coloneqq \boldsymbol{l}_j - \boldsymbol{P} \cdot \boldsymbol{l}_j$ for $j = 1, 2, \ldots, m$
    $L \coloneqq (\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_m)$
  end for
  return $\boldsymbol{B} = (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n)$

matrix of projection on the linear subspace perpendicular to vector $\boldsymbol{b}_i$); L is a finite subset of limit cycle's points (the $\boldsymbol{l}_j$ are uniformly distributed within l).

In this work, the single CPG was simulated for few thousands of iterations, and in each iteration, its state was stored and then used as the $\boldsymbol{l}_j$ point. The $\boldsymbol{l}_j$ points were used as an input for the algorithm and the output matrix $\boldsymbol{B}$ was then used to generate the matrix $\boldsymbol{G}_\text{e}$, as described in previous section A.1. With the matrix $\boldsymbol{G}_\text{e}$, the ellipsoid norm $\|-\|_\text{e}$ can be defined. The ellipsoid's basis (its semi-axis) are generated based on the CPG's states. Hence, I propose to use the ellipsoid norm $\|-\|_\text{e}$ as the searched special norm:

$$\|-\|_\text{cpg} \coloneqq \|-\|_\text{e} \,. \tag{59}$$

The proposed approach is not optimal, but it serves its purpose well, as was proved by successfully run simulations presented in this work.

Appendix B
Content of the Enclosed CD

```
CD
├── source_code
│   ├── coppeliasim
│   ├── dynsys_framework
│   ├── helpstring.txt
│   ├── loco_learn_self_org_plm
│   ├── loco_learn_self_org_snm
│   ├── main.py
│   ├── README.md
│   ├── requirements.txt
│   ├── results_plm.txt
│   ├── results_snm.txt
│   ├── robot
│   └── utils
├── transition_gait_plm.mp4
├── transition_gait_snm.mp4
├── tripod_gait_plm.mp4
├── tripod_gait_snm.mp4
├── wave_gait_plm.mp4
└── wave_gait_snm.mp4
```