POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

# Walking motion generation in bio-inspired hexapod robot using Reinforcement Learning

Tesi di Laurea Magistrale in
Space Engineering - Ingegneria Spaziale

Author: **Stefano Trotta**

Student ID: 945253
Advisor: Prof. Mauro Massari
Academic Year: 2021-2022

# Abstract

Over the last decades, space exploration has become a scientific research milestone and it holds a prominent position in many space agencies' road-maps. Hitherto, space exploration missions has involved only wheeled locomotion systems. However, due to wheeled locomotion limits, this trend could be replaced soon by legged robots, the gait generation of which has become an important research topic in recent years. This aspect is much more important in space, where direct control is very limited by large delays or even impossible. Therefore, this thesis aims to lay the foundations for a space exploration hexapod robot that is capable of autonomous navigation and, in order to enable its adaptability behaviour to environment change, it is equipped with a deep reinforcement learning agent. In the following chapters, a bio-inspired hexapod robot, called Boogie, and its adaptive locomotion controller are presented. Initially, the robot architecture is derived from biomimetic considerations with the aim to grant omnidirectional walking and augmented movement flexibility. Then, the artificial Central Pattern Generator (CPG) used to generate the robot locomotion is introduced. It is based on real neurobiological control systems and it has two layers: the first layer produces the three more common locomotion patterns, tuning the hexapod interlimb coordination. The second layer is the one that directly guarantees the robot adaptability by controlling each limb behaviour. The adaptability is enabled by a reinforcement learning (RL) algorithm that tunes the CPG parameters. Finally, in order to validate the proposed controller and verify its effectiveness, a walking simulation has been performed in Simulink Simscape Multibody™.

**Keywords:** hexapod robot, gait generation, Central Pattern Generator, bio-inspired, Reinforcement Learning, Deep Deterministic Policy Gradient

# Abstract in lingua italiana

Negli ultimi decenni, l'esplorazione spaziale è diventata una pietra miliare della ricerca scientifica e attualmente ricopre una posizione rilevante nelle tabelle di marcia di molte agenzie spaziali. Fino ad ora, nelle missioni di esplorazione spaziale sono stati adoperati solamente sistemi a ruote. Questa tendenza, però, a causa dei limiti del movimento su ruote, potrebbe presto essere sostituita dai robot a zampe, per i quali negli ultimi anni la ricerca scientifica si è concentrata sulla generazione della camminata. Questo aspetto è ancora più importante nello spazio, dove un controllo diretto è reso arduo, se non addirittura impossibile, dai ritardi di comunicazione. A tal proposito, questa tesi si propone di porre le basi per un robot per l'esplorazione spaziale in grado di attuare una navigazione in maniera autonoma e che è controllato da un algoritmo di deep reinforcement learning che possa permettergli di adattarsi alle variazioni dell'ambiente. Nei prossimi capitoli verranno presentati un robot esapode bio-ispirato, chiamato Boogie, e il controllo adattivo utilizzato per la generazione del suo movimento. Per prima cosa, l'architettura del robot è stata definita attraverso considerazioni biomimetiche volte a garantirgli un'aumentata flessibilità di movimento e la possibilità di camminare in ogni direzione. Successivamente viene introdotto il Central Pattern Generator (CPG) artificiale utilizzato per generare la camminata del robot. Questo CPG trae spunto dai veri sistemi di controllo neurobiologici ed è composto da due strutture: la prima è in grado di regolare la coordinazione delle gambe in modo da produrre i tre tipi di camminata esapode più comune; la seconda, invece, si occupa di garantire l'adattabilità del robot modificando il comportamento delle singole zampe. Questa adattabilità è ottenuta tramite un algoritmo di reinforcement learning (RL) che modifica i parametri del CPG. Infine, per validare il sistema di controllo proposto e verificare la sua efficacia, una camminata è stata simulata in Simulink Simscape Multibody™.

**Parole chiave:** robot esapode, generazione dell'andatura, Central Pattern Generator, bio-ispirazione, apprendimento per rinforzo, Deep Deterministic Policy Gradient

# Contents

# List of Figures

# List of Tables

# List of Symbols

| Variable | Description | SI unit |
|---|---|---|
| $\boldsymbol{a}$ | oscillator frequency tuning parameter | - |
| $\boldsymbol{a}_t$ | action taken at timestep t | - |
| $\mathcal{A}$ | action space | - |
| $\mathcal{C}_l$ | logical value for ground contact, l-th leg | - |
| $\boldsymbol{L}_c$ | coxa length | m |
| $\boldsymbol{L}_f$ | femur length | m |
| $\boldsymbol{L}_c$ | tibia length | m |
| $\mathcal{N}$ | DDPG noise sample | - |
| $\boldsymbol{o}_t$ | observation vector at timestep t | - |
| $\boldsymbol{O}$ | observation space | - |
| $\mathcal{O}$ | hexapod trunk orientation | rad |
| $\dot{\mathcal{O}}$ | hexapod trunk orientation derivative | $\frac{rad}{s}$ |
| $\boldsymbol{p}_{CG}$ | hexapod trunk position vector | m |
| $\mathscr{P}$ | state-transition distribution | - |
| $\boldsymbol{Q}(s\|a)$ | Q-value function | - |
| $\mathscr{R}$ | RL reward function | - |
| $\boldsymbol{R}_{cont}$ | contact radius | m |
| $\boldsymbol{R}_{cont}$ | leg radius | m |
| $\boldsymbol{r}_t$ | reward function evaluated at timestep t | - |
| $\boldsymbol{R}_\varsigma$ | cumulative reward function along trajectory $\varsigma$ | - |
| $\boldsymbol{s}_t$ | state vector at timestep t | - |
| $\mathcal{S}$ | state space | - |
| $\boldsymbol{t}$ | timestep | s |
| $\boldsymbol{T}$ | cycle time | s |
| $\boldsymbol{T}_s$ | simulation duration time | s |

| Variable | Description | SI unit |
|---|---|---|
| $\boldsymbol{T}_{sample}$ | DDPG sampling time | s |
| $\boldsymbol{T}_{st}$ | stance phase period | s |
| $\boldsymbol{T}_{sw}$ | swing phase period | s |
| $\boldsymbol{v}_{CG}$ | hexapod trunk velocity vector | $\frac{m}{s}$ |
| $\boldsymbol{w}_{pitch}$ | orientation weight | - |
| $\boldsymbol{w}_{v_x}$ | forward velocity weight | - |
| $\boldsymbol{w}_{y}$ | lateral deviation weight | - |
| $\boldsymbol{w}_{z}$ | vertical deviation weight | - |
| $\boldsymbol{w}_{\Delta t}$ | duration reward weight | - |
| $\boldsymbol{Y}_{t}$ | target function | - |
| $\boldsymbol{\alpha}$ | oscillator velocity convergence rate parameter | - |
| $\boldsymbol{\alpha}_{pitch}$ | pitch angle | rad |
| $\boldsymbol{\beta}$ | duty factor | - |
| $\boldsymbol{\gamma}$ | discount factor | - |
| $\boldsymbol{\delta}_{TC}$ | angle between coxa axis and trunk surface | rad |
| $\boldsymbol{\Theta}_{i,l}$ | i-th joint angle position, l-th leg | rad |
| $\dot{\boldsymbol{\Theta}}_{i,l}$ | i-th joint angle velocity, l-th leg | $\frac{rad}{s}$ |
| $\boldsymbol{\vartheta}^{i}_{j}$ | phase lag between leg i and leg j | rad |
| $\boldsymbol{\Theta}^{Q}$ | parameterized critic function | - |
| $\boldsymbol{\Theta}^{\pi}$ | parameterized actor function | - |
| $\boldsymbol{\kappa}$ | *soft update* hyperparameter | - |
| $\boldsymbol{\mu}$ | oscillator amplitude | - |
| $\boldsymbol{\pi}$ | stationary policy | - |
| $\boldsymbol{\pi}^{*}$ | optimal policy | - |
| $\boldsymbol{\varsigma}$ | RL trajectory | - |
| $\boldsymbol{\tau}_{i,l}$ | torque for l-th leg, i-th joint motion | N m |
| $\boldsymbol{\varphi}_{i,l}$ | i-th joint angle of l-th leg | rad |
| $\boldsymbol{\omega}$ | oscillator frequency | $\frac{rad}{s}$ |
| $\boldsymbol{\omega}_{st}$ | stance frequency | $\frac{rad}{s}$ |
| $\boldsymbol{\omega}_{sw}$ | swing frequency | $\frac{rad}{s}$ |

# List of Acronyms

| Acronym | Description |
|---|---|
| **CG** | Center of Gravity |
| **CPG** | Central Pattern Generator |
| **DDPG** | Deep Deterministic Policy Gradient |
| **DPG** | Deep Policy Gradient |
| **DRL** | Deep Reinforcement Learning |
| **DOF** | Degree Of Freedom |
| **IC** | Initial Condition |
| **MDP** | Markov Decision Process |
| **MIMO** | Multiple-Input and Multiple-Output |
| **POMDP** | Partially Observable Markov Decision Process |
| **RL** | Reinforcement Learning |

# 1 | Introduction

## 1.1. Background

One of the greatest challenges of the $21^{st}$ century is space exploration and for this reason several nations are currently putting it in their political agenda. As of 2022, 16 different government space agencies (out of the existing 77) have launch capabilities, 6 of which have full launch capabilities and extraterrestrial landing capabilities. At present, many countries are participating in, or planning for, space programs that aspire to reach the Moon, Mars and near-Earth asteroids, study them directly in-situ and even collect rocks and sand samples to send back to Earth ([5, 12, 13]).

Robotic systems have often been employed to achieve many of these scientific objectives that would otherwise be impractical, too expensive or even impossible. Until now, planetary explorations that didn't involve human crew have been done solely by rovers, that are wheeled ground vehicles. Some of the most successful robotic exploration missions have been NASA's MER (Mars Exploration Rover, [32]), which involved the two rovers Spirit and Opportunity ([26]), the more recent MSL (Mars Science Laboratory, [22]) with the rover Curiosity ([40]), and the ongoing Mars 2020 ([33]) that launched the Mars rover Perseverance ([11]). Over the years, rovers have been constantly improved, but, due to their reliability and low complexity, they have never been replaced by new, different systems. However, wheeled robots have low movement performance on rough and steep terrains because wheels require continuous contact with ground; moreover, the environment could induce on the wheel an effect called *wheel slip* ([49]), in which wheels do not roll on the surface they are on, but slip. These problems prevent rovers from exploring scientific interesting areas like volcanoes, mountains and dark craters.

Since versatile locomotion is essential for space exploration and celestial bodies involve a unique set of unknowns, legged walking robots have been developed in order to improve the movement performance. These systems can move along unstructured terrains thanks to their multiple legs, which grant them good mobility in natural ground types. Among the existent solutions, hexapod robots have the highest walking static stability and this is the reason why they have become an important research topic in recent years, especially

the studies on gait and locomotion planning. In space exploration, autonomous locomotion controllers are very important for a correct walking, even because, due to the planet distance, direct control is very limited by large delay, if not impossible. Fortunately, recent studies ([7]) in robotics have focused on implementing neural networks for gait generations in order to allow the robots to adapt to environment changes and they have obtained promising results. Therefore, training reliable neural networks capable of drive robots on every terrain could become the keystone for the success of the next generation of space exploration missions.
This thesis focuses exactly on that need presenting a preliminary design of a hexapod robot, called Boogie, and the implemented RL-based locomotion controller. The robot architecture has been designed starting from insect physiognomy with the aim to grant omnidirectional walking and augmented movement flexibility. Also the adaptive locomotion controller draws inspiration from the real world: an artificial, two-layer Central Pattern Generator (CPG), based on neurobiological hierarchical control systems is proposed. This two-layer structure characterizes the adopted bio-inspired learning approach: the first layer is used to generate the basic hexapod locomotion patterns, while the second layer adapts the limb motion to the environment change by means of a Deep Deterministic Policy Gradient (DDPG) algorithm.

## 1.2. Structure of the Thesis

The present thesis is structured as follows: after discussing the architecture design and configuration of the proposed Boogie robot (chapter 2), gait generation and interlimb coordination through Hopf oscillators are presented (chapter 3). Chapter 4 contains some notions about deep reinforcement learning, particularly about the adopted Deep Deterministic Policy Gradient (DDPG) algorithm, and illustrates the implemented RL network architecture. In chapter 5, the modelling of the simulation environment and Boogie dynamics in Simulink™ are reported; simulation results and performance are shown and commented in chapter 6. Finally, chapter 7 draws the conclusions and presents ideas for future developments of the model.

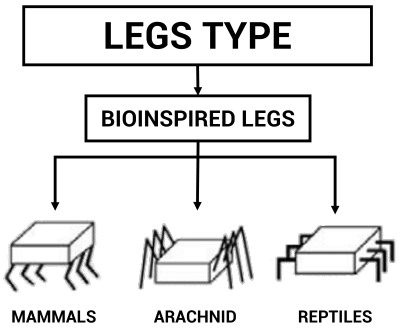
# 2 | Robot design and configuration

## 2.1. Biomimetic robot design

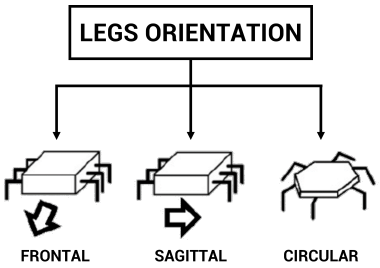
Due to their impressive rapidity and coordination, in robotics insects are considered as absolute models for legged robot design, either for architecture or for locomotion pattern. In this section, the physiognomy of the proposed hexapod will be described, highlighting the biomimetic criteria followed in the design phase.

### 2.1.1. Hexapod body configuration

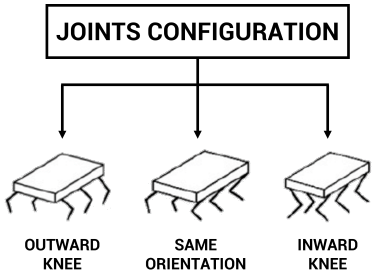
The basic architectures commonly used for hexapod robots are only two: rectangular and hexagonal shapes ([47]); both have advantages and disadvantages compared to the other one. Owing to the planetary exploration aim of the robot, the hexagonal configuration has been preferred in this work since its radial symmetry requires all the legs to be equal; in this way, it cannot be identified neither a body front or rear, implying no preferential direction for the walking. Moreover, many studies ([6, 38, 45]) have proven that an omnidirectional chassis grants a greater stability margin, it has a way better turning capability than rectangular architecture and this lets the robot move at once in any direction, an important feature for space exploration. The hexagonal chassis used in this research has an outer radius equal to 15 $cm$ and an height of 1 $cm$.
The natural outgrowth of the architecture choice is the legs orientation adopted in the design: as shown in fig. 2.1b the options available are three, but frontal and sagittal configuration embed an unidirectional movement direction. In order to let the mechanism move in all directions, the circular solution has been selected, thus the six legs are placed over the vertexes of the hexagon in a radial disposition. Furthermore, the bioinspired leg type (fig. 2.1a) adopted is the arachnid one due to its higher stability, while an outward knee orientation has been preferred for a more realistic mimic capability. In this respect, the design leg followed is described in detail in the next section.




(a) Bioinspired leg types.




(b) Legs orientation for hexapods design leg.




(c) Joint configuration options.

Figure 2.1: Type setting of hexapod legs design[47].

### 2.1.2. Hexapod legs design

The hexapod legs design is one of the most tough parts of walking robots, because it determines which locomotion patterns can be used by the robot itself; for this reason, stereotypical insect legs have been used as biological inspiration for limbs modelling. Typically, an insect leg is made up of six basic appendages connected by five different joints. As shown in fig. 2.2, from proximal to distal they are: coxa, trochanter, femur, tibia, tarsus and pretarsus ([21]).

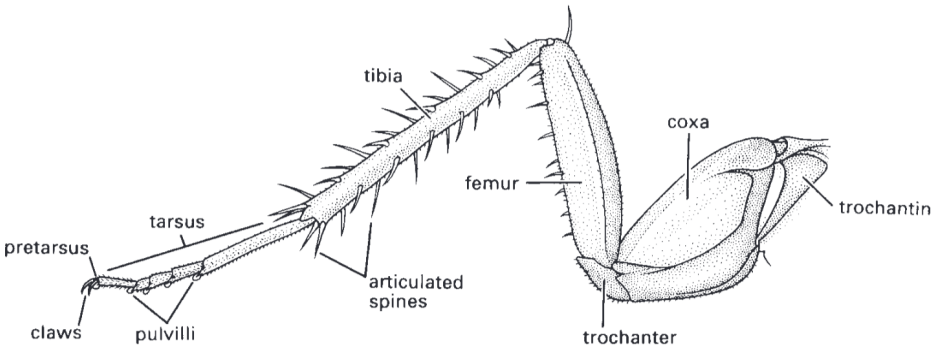



Figure 2.2: Typical anatomical structure of an insect leg[21].

Due to its biological complexity, it would be difficult to directly reproduce the anatomical arrangement of insect limbs, thus the number of degrees of freedom (DOFs) must be decreased in order to reduce the complexity of both mechanisms and control.
Standard hexapods require at least two 1-DOF joints in order to walk forward: one is needed to move the limb forward and backward; the other is used to lift the leg up and down. A correlation between these two actions and two specific anatomical segments, namely coxa and tibia, can be created. Nevertheless, implementing only 2 DOFs could result in feet slippage on the ground, since only the knee (see fig. 2.3) could compensate the hip rotation; moreover, using this expedient will also entail a change in the body height during walking ([52]). In order to avoid this problem, a third joint, the ankle, and the corresponding tibia segment have been added: with these 3 DOFs the combination of knee and ankle rotations can ensure a constant body height level while the hexapod moves forward ([10, 34]).
Trochanter, tarsus and pretarsus have been neglected in this design owing to their small size and features. This choice has been made with the purpose of reducing the overall system complexity and because three is the minimum number of DOFs required for an omnidirectional walking robot ([23]).
The final configuration of the developed leg is shown in the following figure.

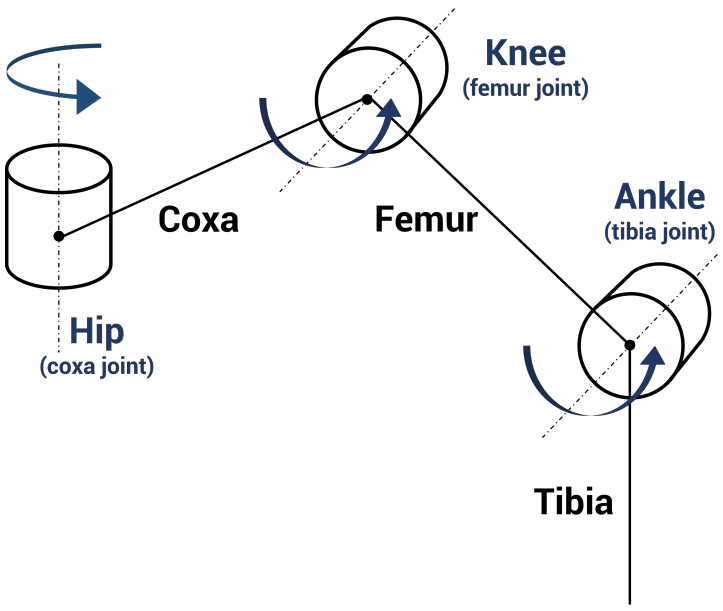



Figure 2.3: Implemented configuration of hexapod legs. A 3 DOFs mechanism has been selected to grant omnidirectional walking, avoiding slipping of the feet. Hip allows the leg to move forward and backward; femur and tibia joints are used to move the limb up and down.

Based on the optimization design study performed by Zhao et al., ([51]), the ratio of the limbs (i.e. $coxa : femur : tibia$) has been chosen as $1 : 4 : 3$. This ratio is coherent with the data proposed by Fichters in their survey on insects' legs ([14]), in which they state that tibia and femur lengths are highly correlated by a coefficient that ranges from 0.78 and 0.97, while coxa has no explicit correlation with neither segments.
Another result recovered from the work of Zhao is the value of the angle between the coxa axis and the trunk surface (the red angle in fig. 2.4) which has been set to $\delta_{tc} = \frac{\pi}{6}$ in order to grant a better movement flexibility. Also those data have been proven to be consistent with insects' anatomical study.

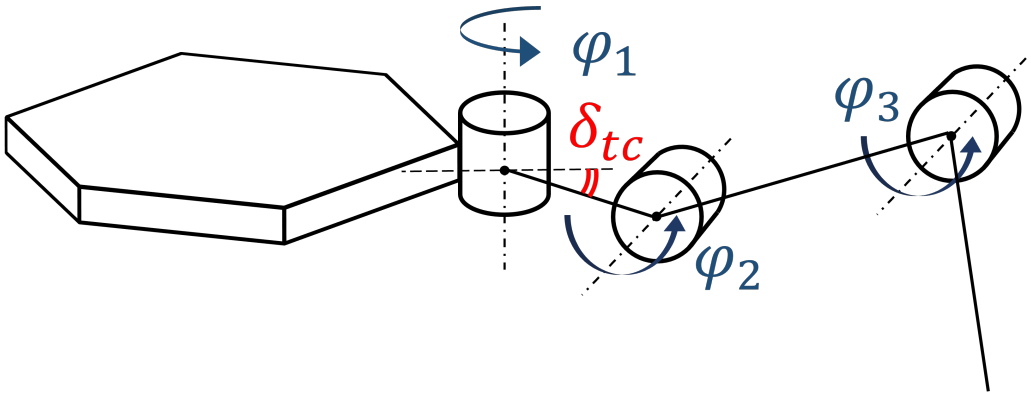



Figure 2.4: Scheme of the anatomically inspired leg configuration.

In conclusion, from a mechatronic viewpoint, each leg can be considered as a manipulator made up of three segments connected through two hinge joints. The segment lengths are $L_{coxa}$, $L_{femur}$ and $L_{tibia}$, while the joint angles are $\varphi_1$ (coxa), $\varphi_2$ (femur) and $\varphi_3$ (tibia). In result, the hexapod robot can be treated as a walking vehicle, propelled by six independent limbs. Finally, it should be noticed that physical constraints on joint angles have been introduced. In fact, the hinges described so far could ideally rotate indefinitely and may result in leg collisions. Table 2.1 collects the leg dimensions and the joint angle ranges used in this work for the simulation frame.

Table 2.1: Leg dimensions and joint angle ranges.

| **Segment** | **Length** $[cm]$ | **Joint** | **Angles** | **Angle ranges** $[°]$ |
|---|---|---|---|---|
| **Coxa** | $L_{coxa} = 3$ | **Hip** | $\varphi_1$ | $[-45; 45]$ |
| **Femur** | $L_{femur} = 12$ | **Knee** | $\varphi_2$ | $[0; 30]$ |
| **Tibia** | $L_{tibia} = 9$ | **Ankle** | $\varphi_3$ | $[-120; -60]$ |

## 2.2. Sensors

An important key role in deep reinforcement learning training is played by the observation vector: it contains the sensor measurements of the surrounding environment.
The Boogie robot proposed in this thesis is equipped with an IMU placed in correspondence of the hexagon center of gravity (CG) to measure body attitude, velocity and acceleration. Moreover, every joint is provided with a servomotor capable of generating a max torque of 10 $Nm$ and each motor has absolute encoder to measure relative joint angle, joint angle velocity and the generated torque. The robot can also detect ground using force sensors placed on the tip of each leg, represented as hemisphere. The overall hexapod architecture and the placement of the sensors can be appreciated in fig. 2.5.

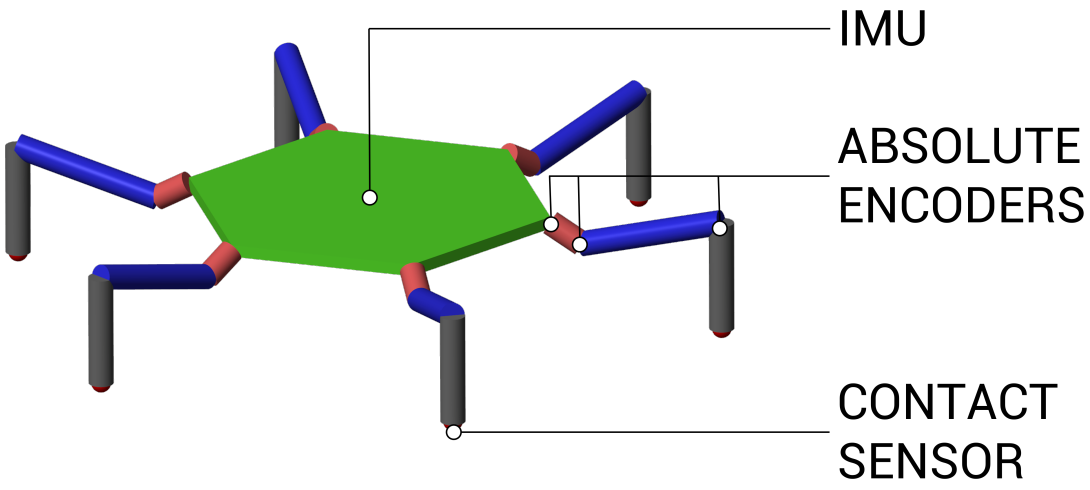



Figure 2.5: Representation of the Boogie hexapod in Simulink ®. It is equipped with an IMU placed at the hexagon CG, 18 absolute encoders on the leg joints and 6 force sensors for ground detection.

# 3 | Hexapod gait generation

In this chapter, the locomotion of the Boogie robot is shown and described in detail. The control system used to generate the walking pattern has biological inspiration and it is capable of generating and reproducing different types of gait: in order to reach this goal, joint movements are guided by coupled Central Pattern Generators (CPGs), formulated as non-linear Hopf oscillators.

## 3.1. Gait description: a biological inspired approach

During animal locomotion, gait is one of the most important action. A definition of *gait* has been proposed by Mahajan and Figueroa in 1997 ([27]): "The gait of an articulated living creature, or a walking machine, is the corporate motion of the legs, which can be defined as the time and location of the placing and lifting of each foot, coordinated with the motion of the body, in order to move the body from one place to another". A locomotion gait can be characterised using three parameters ([44]): the *cycle time* ($T$), the *duty factor* ($\beta$) and the *relative phase lag* ($\theta_j^i$).

Typically, a gait is a periodic relation between the motion of all limbs during walking and it is normally cyclic because the same sequence of lifting and placing the legs is repeated. A step cycle is the full path of leg movements during which all limbs lift and place exactly one time each.

**Definition 1 (Cycle time[44]).** In a gait, the cycle time (or stride) is the time interval in which one step cycle is performed.

In a cycle time, two different phases can be studied: the stance (or support) phase, that is the timespan in which the foot is in contact with ground propelling the body forward, and the swing (or transfer) phase, when the leg is lifted and it is moved in the following support position.

**Definition 2 (Duty factor[44]).** In a gait, the duty factor is the time fraction of a cycle time in which the leg is in stance phase.

From the previous statement, it is clear that $\beta \in (0, 1)$. Moreover, eq. (3.1) can be easily derived and eq. (3.2) is obtained knowing that swing and stance phases are the two parts that constitute the stride, so $T = T_{st} + T_{sw}$.

$$T_{st} = \beta \cdot T \tag{3.1}$$

$$T_{sw} = (1 - \beta) \cdot T \tag{3.2}$$

Finally, from definition 2, an algebraic expression for the duty factor can be obtained:

$$\beta = \frac{T_{st}}{T} = \frac{T_{st}}{T_{st} + T_{sw}} \tag{3.3}$$

Physiologists have discovered that in nature animals tend to increase their locomotion velocity by reducing $T$, increasing step number per second ([18]). Their observations have led to the deduction that the technique adopted to decrease the stride mainly consists in a reduction of $T_{st}$, without changing $T_{sw}$.

**Definition 3 (Phase lag[4]).** In a gait, the phase lag $\theta_j^i$ of leg $i$ is the fraction of a cycle period elapsed from the setting down of a chosen reference foot (of leg $j$) until the foot of leg $i$ is set down.

$$\theta_j^i = \frac{\Delta t_i}{T} \tag{3.4}$$

$\Delta t_i \leq T$ is the time delay between the placing events of reference leg and foot $i$; thus, $\theta_i \in [0; 1]$. In this thesis, the left front leg (*1* in fig. 3.1) has been taken as reference, so eq. (3.4) can be rewritten in a simplified form (eq. (3.5)).

$$\theta_1^i = \theta_i = \frac{\Delta t_i}{T} \tag{3.5}$$

In many biological systems of multi-legged locomotion, including hexapods, the number of degrees of freedom (DOFs) is larger than is required to correctly execute a walking path: as a result, the coordination of all the legs requires the control system to choose one out of various possible alternative movements. These multiple possibilities generate different types of gait. Ramdya et al., ([39]) highlighted that through biological investigation three basic locomotion patterns of *Drosophila melanogaster*, a popular model for insect locomotion studies, can be extracted: *tripod* locomotion, *quadruped* locomotion or ripple gait and *metachronal* locomotion or wave gait.

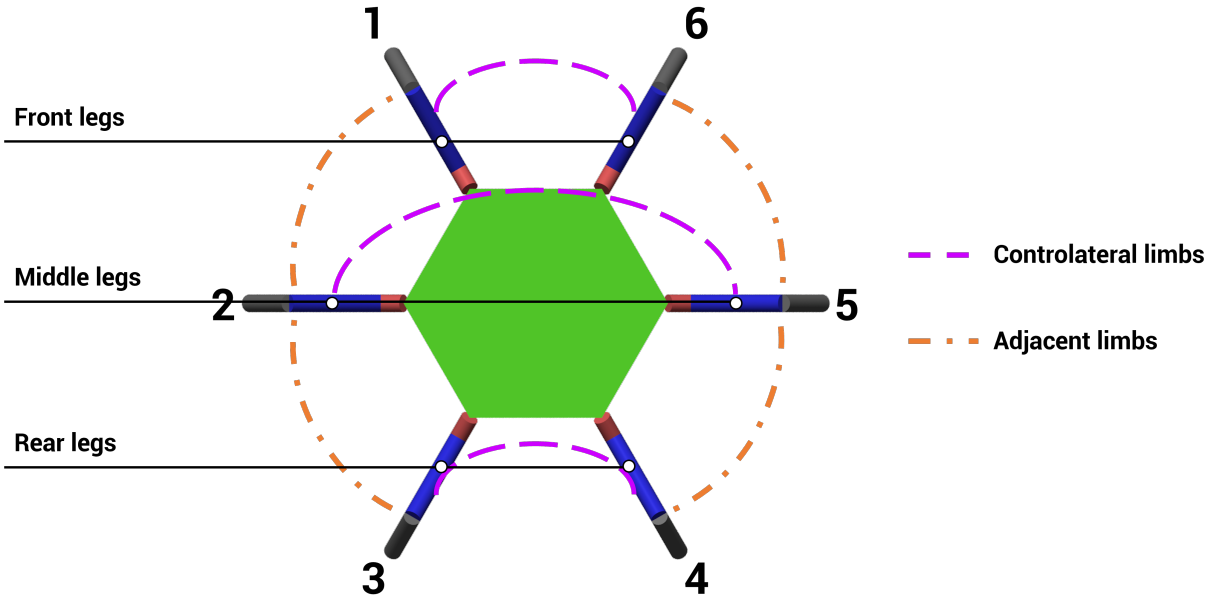
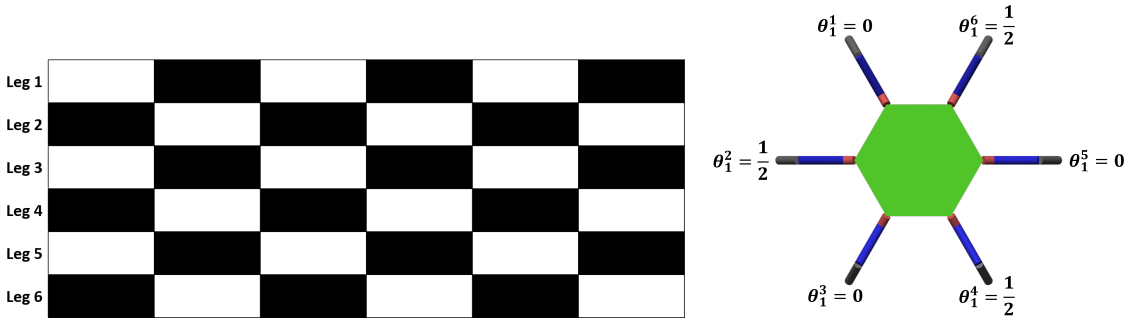



Figure 3.1: Legs' convention scheme.

Based on this evidence, the present thesis has been focused on these three straightforward-walking gaits, which obey the assumptions proposed by Wilson ([50]) and recently taken up by Campos et al., ([4]):

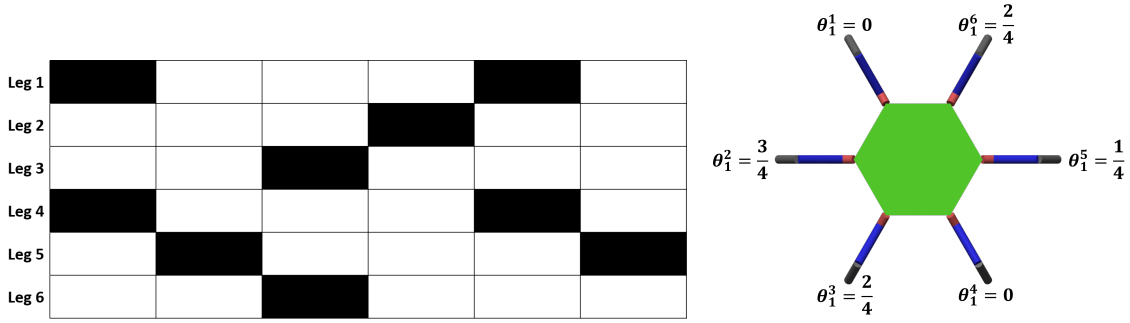
1. any leg moves forward only when the one behind it is in stance position;
2. any controlateral pair performs a strict alternation, namely $\theta^i_{i+3} = \pm 0.5$.

Figures 3.2, 3.3 and 3.4 show the feet placing pattern and the relative phase lag for the three gaits considered. In the diagrams on the left, a black cell indicates that the corresponding leg is swigging, while a black cell means stance condition.



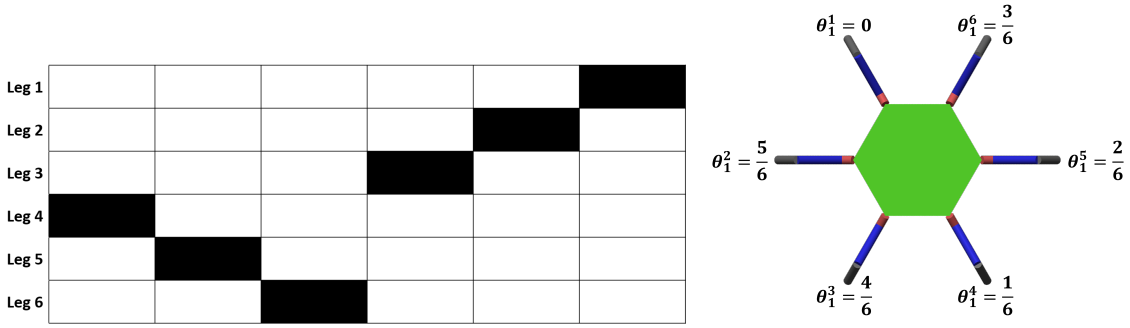

(a) Footfall pattern diagram. (b) Legs' phase lag.

Figure 3.2: Tripod gait schemes.




(a) Footfall pattern diagram. (b) Legs' phase lag.

Figure 3.3: Ripple gait schemes.




(a) Footfall pattern diagram. (b) Legs' phase lag.

Figure 3.4: Wave gait schemes.

In order to better understand the difference between the three most common hexapodal gaits implemented, they are examined in depth in the following sections.

### 3.1.1. Wave gait

Metachronal locomotion (fig. 3.4) is the slowest among the three considered because only one leg is required to swing at a time. It is used by hexapod insects in unhurried walking and very often it is associated to a duty factor of $\beta = \frac{5}{6}$, implying that during one step cycle, every foot touches the ground for $\frac{5}{6}T$.
A cyclic pattern of six steps can be highlighted in one stride, as shown in fig. 3.4a: it is like a wave that propagates from the rear leg to the front leg of one side, and then from back to front on the other side (e.g. $4 \rightarrow 5 \rightarrow 6 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 4$). In each step just one leg swings, while the other five are in contact with the ground to provide the forward locomotion. This configuration corresponds to a phase lag between two adjacent limbs of $\theta_j^i = 60°$ and half a period for controlateral limbs, as already stated previously.

### 3.1.2. Ripple gait

The quadrupedal locomotion (fig. 3.3) is the mid-velocity hexapodal gait, used as middle way while changing from tripod to wave gait and vice versa. Its typical duty factor is $\beta = \frac{3}{4}$, thus it consists of four steps. The peculiarity of this walking is that the front leg of one side moves in phase with rear leg of the other side, as depicted in fig. 3.3a. The legs swinging together are on opposite sides in order to grant robot stability.
The phase lag between adjacent limbs is $\theta_j^i = 90°$, meaning that every quarter of a period one or two legs move forward (e.g. 1 & 4 $\rightarrow$ 5 $\rightarrow$ 3 & 6 $\rightarrow$ 2 $\rightarrow$ 1 & 4 ).

### 3.1.3. Tripod gait

The fastest and still statically stable gait employed by hexapod insects is the tripod one (fig. 3.2). This is the reason why it is also the most common implemented gait in robotics, typically with a duty factor of $\beta = \frac{1}{2}$, meaning that each foot touches the ground for half the step cycle. If $\beta < \frac{1}{2}$, it is said that hexapod "is running".
Tripod gait comprises two steps in which ipsilateral front and rear legs move together in phase with the controlateral middle leg, as shown in fig. 3.2a. On each side, middle limb is half a period out of phase with respect to its adjacent ones; thus, at each time, three legs move together in phase (e.g. 1, 3 & 5 $\rightarrow$ 2, 4 & 6 $\rightarrow$ 1, 3 & 5).

## 3.2. Locomotion control through CPG

### 3.2.1. Biological CPG

The control of robot locomotion is a complex and widely studied problem that engineers and neurobiologists have tried to resolve in many different ways. In this field, robotics and neuroscience join forces to increasingly improve robots capability to mimic natural animal coordinated walking behaviours.
Neuroethology and neurophysiology researchers have studied for long the control system in walking animals to elucidate its operating principles. They found that, in vertebrates, brain consciousness is not involved in rhythmic coordinated behaviours, like walking or breathing. Despite that, vertebrates are still capable of adapting to the changing terrain. This ability derives from the fact that their rhythmic limb activities are controlled by a spinal network that are referred to as Central Pattern Generators (CPGs) ([9, 19, 36]) which do not require regulation command from the brain-stem level ([20]).
Therefore, CPGs are essentially neural circuits capable of generating basic coordinated patterns of periodic output signals, integrating commands from various sources, to meet

the requirements of the surrounding environment ([16]). Hence, the limb motions are modified adaptively to achieve robustness against environmental variations.

Based on the similarity between biological legged animals and legged robots, many bio-inspired robots have been proposed with a gait control mechanism that mimics the one of vertebrates. Bionic CPGs are typically used to control the joint of multi-legged robots and they are implemented through the paradigm of neural networks or systems of coupled oscillators. In this paper the latter is proposed.
Biological CPGs can be mathematically reproduced by a cluster of neural oscillators formulated as coupled non-linear differential equations, whose amplitude, frequency and relative phase lag can be accurately tuned by adjusting CPGs' parameters. Over the years, different models of non-linear oscillators working as artificial CPGs have been developed and the most popular are Hopf, van der Pol and Rayleigh oscillators. In the present thesis, motivated by the works of Campos et al., and Ouyang et al., ([4, 37]), a set of 18 Hopf oscillators has been employed to move all the hexapod joints.

### 3.2.2. Hopf oscillator

Hopf oscillator is a satisfactory model for reproducing biological CPGs in robotic locomotion applications owing to its eminent features ([3, 42]):

- robustness for disturbances, due to its stable limit cycle;
- fast rate of convergence;
- parameters with explicit physical meaning, namely frequency and amplitude, that can be adjusted independently.

Hopf oscillator is formulated as two coupled non-linear differential equations, reported in eq. (3.6)

$$\begin{cases} \dot{x} = \alpha \left(\mu^2 - x^2 - y^2\right) x - \omega y, \\ \dot{y} = \alpha \left(\mu^2 - x^2 - y^2\right) y + \omega x, \end{cases} \tag{3.6}$$

where $x$ and $y$ are the state variables, $\mu$ is the amplitude of the steady state oscillation, $\omega$ is the oscillator's frequency and $\alpha$ is a positive constant which influences the convergence speed.

As all the other ones, Hopf oscillator can be represented as a couple of neurons, the first is excitatory and the second inhibitory. A depiction of this description is proposed in fig. 3.5.

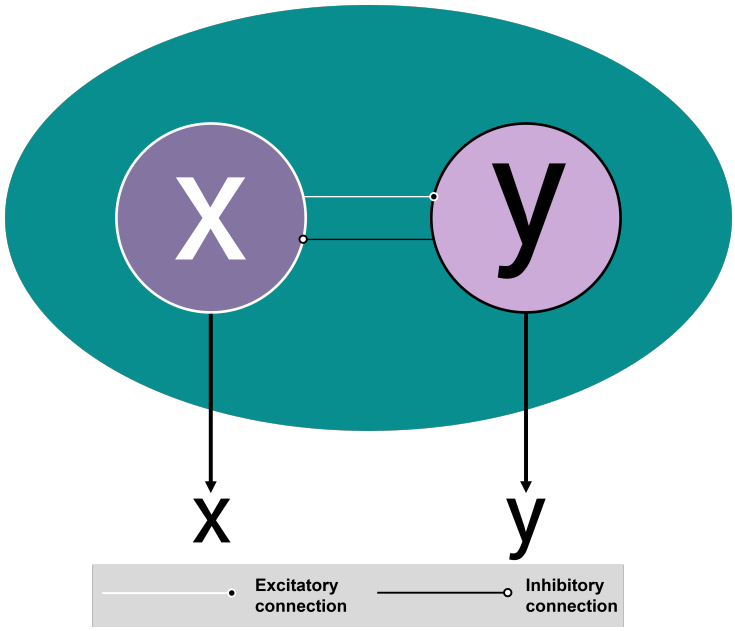



Figure 3.5: Neuron model of Hopf oscillator.

The state values $x$ and $y$ have harmonic, sine-like solution waves and both could be selected to control joint movements. Since for every joint only one signal is required as output, $x$ has been chosen in this work. Considering the 6 coxae, the control of the hips has been set as follow: the descending phase of $x$ corresponds to the stance phase, in which the leg moves backward propelling the body forward; while the swing phase is set corresponding to the ascending phase of the state variable, so that the foot can be placed in an advanced position. The convention adopted is represented in the following figure.

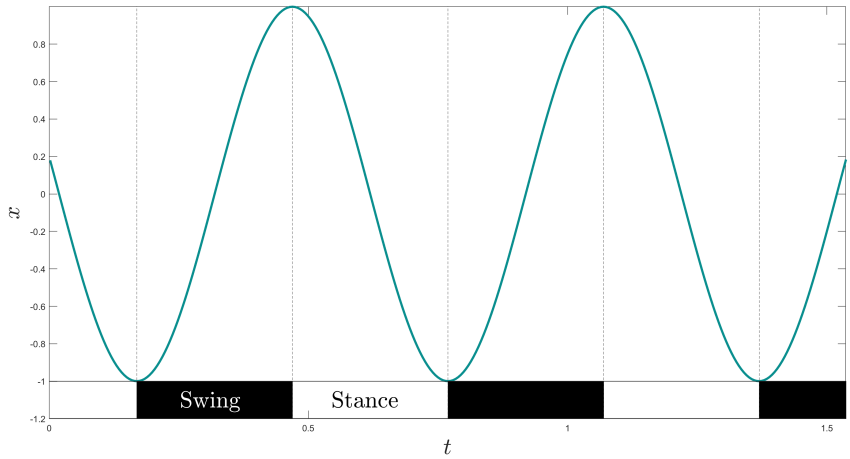


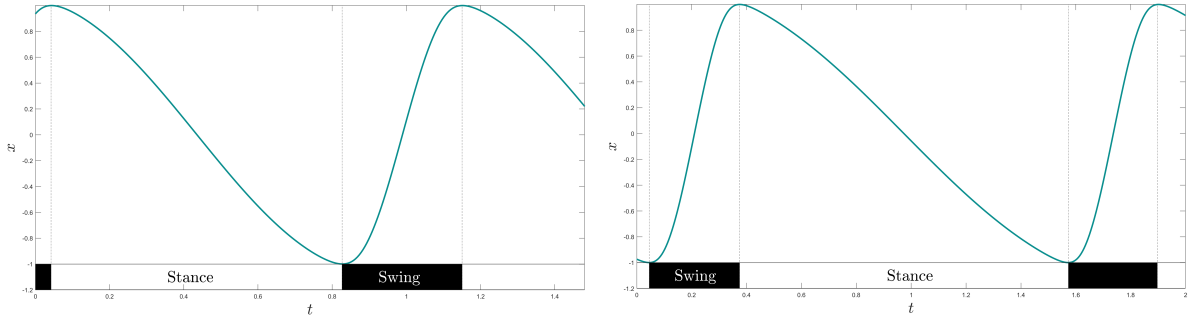

Figure 3.6: Correspondence between oscillator state value $x$ and hexapod gait phase: during signal ascension, the robot performs swing phase; descending output corresponds to stance phase.

In this configuration the oscillator generates an output in which the swing phase will last as the stance phase because of the symmetry of its waveform, as shown in fig. 3.6. Even if this model is appropriate for the tripod gate ($\beta = 0.5$, cf. section 3.1.3), it is not suitable for gaits whose duty factor is not $\beta = 0.5$. For this reason, in order to correctly implement the other two gaits analyzed before and based on the work proposed by Righetti et al., ([41]), the following equation has been employed for an independent control of swing and stance duration.

$$\omega = \frac{\omega_{st}}{e^{-ay} + 1} + \frac{\omega_{sw}}{e^{ay} + 1} \tag{3.7}$$

The frequency presented in eq. (3.7) alternates between $\omega_{sw}$ and $\omega_{st}$ depending on the phase of the inhibitory neuron $y$, thus it is possible to control $T_{sw} = \frac{\pi}{\omega_{sw}}$ and $T_{st} = \frac{\pi}{\omega_{st}}$, that are swing and stance step phases period respectively. The speed of the switch between the two frequencies is regulated by the parameter $a$, the time ratio between the two phases. Therefore, it is possible to implement a gait with a specific duty factor $\beta$ just assigning a fixed $\omega_{sw}$ and deriving the stance frequency (eq. (3.8)) from eq. (3.3). In fig. 3.7 it can be appreciated how step phases change while $\beta$ increases.

$$\omega_{st} = \frac{1 - \beta}{\beta} \omega_{sw}. \tag{3.8}$$




(a) Correspondence with $\beta = \frac{3}{4}$.

(b) Correspondence with $\beta = \frac{5}{6}$.

Figure 3.7: Comparison of $T_{sw}$ and $T_{st}$ for different duty factor. They can also be compared to fig. 3.6 to notice that the greater $\beta$ the higher the difference between the durations. All the three graphs have been generated using $\alpha = 300$, $\omega_{sw} = \frac{\pi}{0.3}\left[\frac{rad}{s}\right]$, $\mu = 1$ and $a = 5$.

### 3.2.3. Interlimb coordination

In nature, biological neurons can set up a neural mesh via synaptic pathways to control interlimb movements. In this work, the rhythmic interlimb coordination has been imple-

mented by dynamically coupling the six hip joints, controlled by individual oscillators, to ensure synchronization during the overall locomotion. The diffusive coupling equations ([4, 37]) are reported below.

$$\dot{\boldsymbol{X}}_{\boldsymbol{i}} = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = \begin{bmatrix} \alpha\left(\mu^2 - x_i^2 - y_i^2\right) & -\omega_i \\ \omega_i & \alpha\left(\mu^2 - x_i^2 - y_{ij}^2\right) \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + k \cdot \sum_{j \neq i} \overline{\boldsymbol{R}}\left(\theta_j^i\right) \begin{bmatrix} 0 \\ \frac{x_j + y_j}{\sqrt{x_j^2 + y_j^2}} \end{bmatrix} \tag{3.9}$$

where $i, j = 1, 2, \ldots, 6$ denotes the legs' number, according to the convention proposed in fig. 3.1. $\alpha$ has been set equal to 100 following the results obtained by Ouyang et al., ([37]), while $k = 0.5$ is the constant coupling strength. $\theta_j^i$ is the phase lag between two joints (definition 3). Finally, $\overline{\boldsymbol{R}}$ is a rotation matrix that rotates the linear terms onto each other to perform the gait properly. It is defined as:

$$\overline{\boldsymbol{R}}\left(\theta_j^i\right) = \begin{bmatrix} \cos\left(\theta_j^i\right) & -\sin\left(\theta_j^i\right) \\ \sin\left(\theta_j^i\right) & \cos\left(\theta_j^i\right) \end{bmatrix}.$$

Equation (3.9) can be simplified in the following equation:

$$\dot{\boldsymbol{X}}_{\boldsymbol{i}} = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = \begin{bmatrix} \alpha\left(\mu^2 - x_i^2 - y_i^2\right) & -\omega_i \\ \omega_i & \alpha\left(\mu^2 - x_i^2 - y_{ij}^2\right) \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} f_{x_i} \\ f_{y_i} \end{bmatrix}$$

where $f_x$ and $f_y$ are the coupling terms coming from the other oscillators in the network. Consequently, fig. 3.5 can be updated as follows.

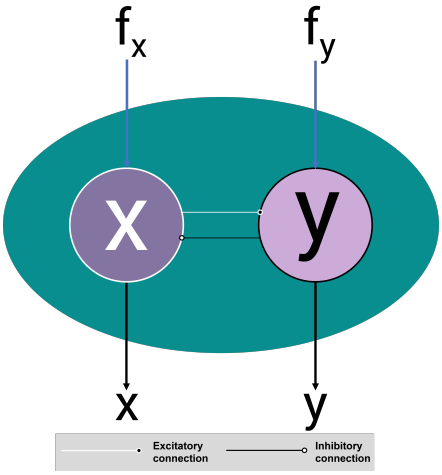



Figure 3.8: Neuron model of Hopf oscillator in neural network.

Figure 3.9 depicts the whole diagram of the neural network proposed for coxas' coordination. The arrows represent the dynamical coupling that allows legs coordination and it must be noticed that the network model chosen is not a fully connected one, in order to simplify the high-level optimization. This means that every CPG is not linked to all the other five, but only to the adjacent ones. A detailed representation of these 7 bidirectional coupling is shown in fig. 3.10, in which the terms like $f_x^i$ represent the coupling terms from the other oscillators.

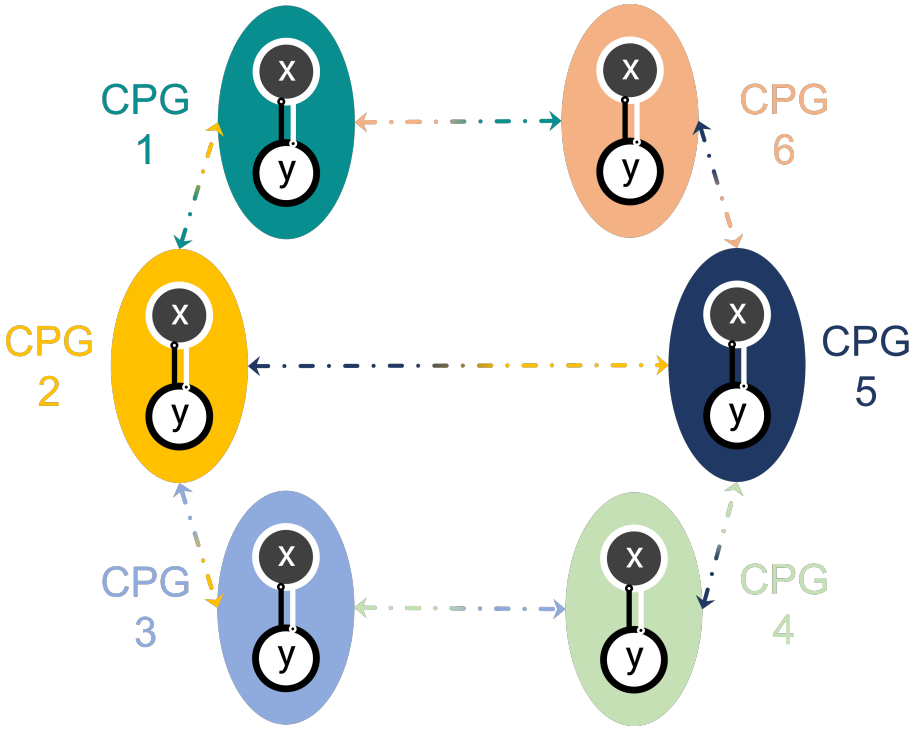



Figure 3.9: Hip CPGs network made of six Hopf oscillators.

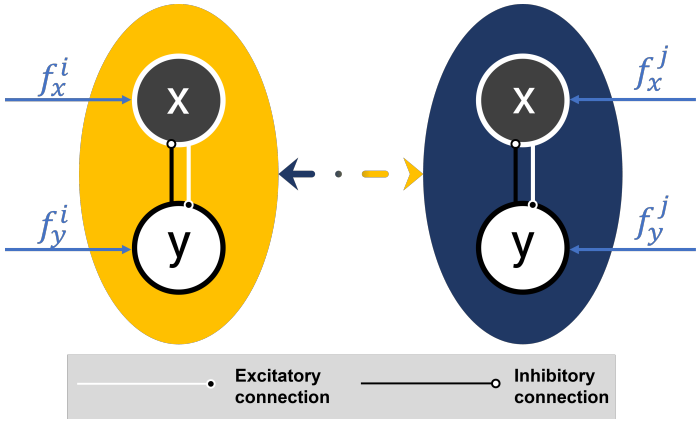



Figure 3.10: Bidirectional coupling between two oscillators.

### 3.2.4. Coxae gait generation simulations

Several simulations have been conducted in Simulink® to validate the implementation of the mathematical model and to verify the effectiveness of the interlimb coordination method adopted. Metachronal, ripple and tripod hexapodal gaits have been generated and will be showed in this section. All simulations presented in this work have been done using MATLAB® and Simulink®. All the three gaits have been simulated for 10 $s$ using $\mu = 1$, $a = 5$, $\alpha = 300$, $\omega_{sw} = 4\pi$ $[\frac{rad}{s}]$ and $k = 0.5$. Duty factor and relative phase lags $\theta_j^i$ used in $\overline{\boldsymbol{R}}$ are collected in table 3.1, sorted by gait. The initial conditions are set to 0 for all the x-states, while y-states have random values $\in [-1; 1]$.

Table 3.1: Duty factor and relative phase between coxas for the gaits implemented.

| Gait | $\beta$ | $\theta_1^6$ | $\theta_1^2$ | $\theta_6^5$ | $\theta_2^5$ | $\theta_2^3$ | $\theta_4^5$ | $\theta_3^4$ |
|---|---|---|---|---|---|---|---|---|
| **Tripod** | $\frac{1}{2}$ | $\pi$ | $\pi$ | $\pi$ | $\pi$ | $\pi$ | $\pi$ | $\pi$ |
| **Ripple** | $\frac{3}{4}$ | $\pi$ | $\frac{\pi}{2}$ | $\frac{\pi}{2}$ | $\pi$ | $\frac{\pi}{2}$ | $\frac{\pi}{2}$ | $\pi$ |
| **Metachronal** | $\frac{5}{6}$ | $\pi$ | $\frac{\pi}{3}$ | $\frac{\pi}{3}$ | $\pi$ | $\frac{\pi}{3}$ | $\frac{\pi}{3}$ | $\pi$ |

**Tripod gait** In fig. 3.11 the results of the tripod gait simulation are shown. As planned, three coxa trajectories are in phase at each time: legs 1, 3 and 5 moves together in phase as well as legs 2, 4 and 6. About 3 $s$ are necessary to allow oscillators reach the desired configuration. If figs. 3.11, 3.12 and 3.13 are set side by side it is possible to note how fast or slow one gait is. For example, considering the last five seconds and taking the coxa 1 line, it has 11 peaks in tripod gait, 7 in ripple and only 5 in wave.

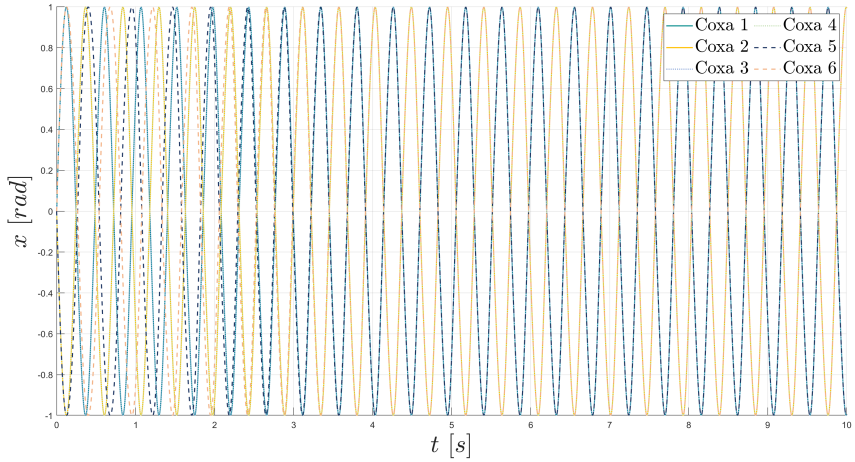



Figure 3.11: Simulation of a tripod gait.

**Ripple gait** Figure 3.12 depicts the planned hip trajectories for a quadrupedal walking, the mid-velocity gait. A transient of 5 $s$ is required to achieve the correct phase lags. After that interval, the ripple rhythmic alternation begins. Coxae 1 and 6 are in phase, as well as the trajectories of coxa 3 and coxa 6.

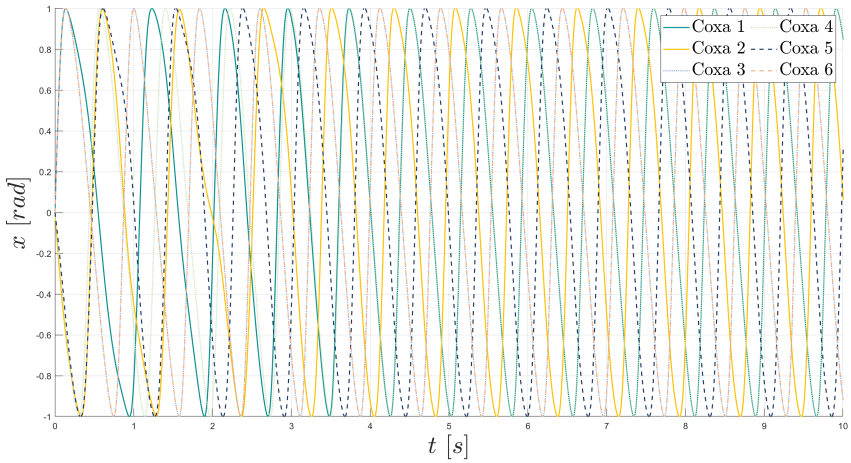



Figure 3.12: Simulation of a quadrupedal gait.

**Metachronal gait** Figure 3.13 shows the coxa joint trajectories $x_i$ for the slowest gait. It can be easily verified that they are coordinated as wanted: every oscillator output has a lag of a sixth of the period $T$ with respect to the adjacent ones, while the lag between contralateral limbs is half a period as before. Around 5 $s$ are needed to reach the desired coordination.
Comparing this image with fig. 3.4a, it can be stated that the footfall described in section 3.1.1 has been correctly implemented: in fact just before five seconds, leg 1 begins the stance phase, followed by 6, 5, 4, 3 and 2 in that order every $\frac{T}{6}$.

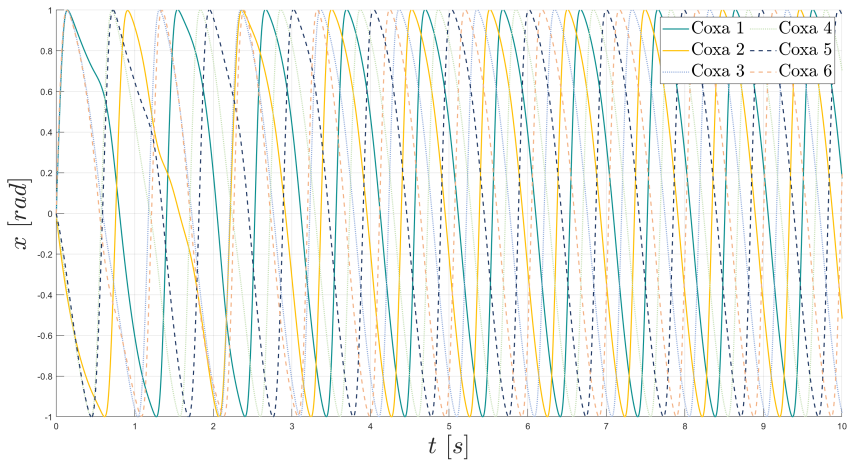



Figure 3.13: Simulation of wave gait.

## 3.3. Two-layer CPG locomotion model

In section 3.2.3 the controller for the movement of the six hip joints has been presented. In this section the two-layer CPG model for knees and ankles motion is presented.

### 3.3.1. Two-layer CPG network

The proposed Boogie hexapod model has 18 DoFs, but just 12 of them are regulated by the neural network described previously. Based on the paper of Ouyang et al., ([37]) the remaining joint controllers are again formulated as Hopf oscillators, but they interact in pairs (knee-ankle) only with the CPG of the coxas network that belongs to their limb. The resulting control scheme is a 3D two-layer CPG model in which two layers can be highlighted:

- the upper layer, called[37] *body layer*, is tasked with generating basic locomotion patterns, namely one of the typical gaits discussed in section 3.1;
- the lower layer, the *limb layer*[37] (one per each leg), receives the locomotion pattern information from the upper one and controls knee and ankle joints to tune the walking, in order to adapt to the changing environment.

Hence, the whole control scheme for the Boogie robot is shown in fig. 3.14 and the mathematical model of the limb layer is reported in eq. (3.10).

$$\begin{aligned}\dot{\boldsymbol{X}}_{m,l} = \begin{bmatrix}\dot{x}_{m,l}\\ \dot{y}_{m,l}\end{bmatrix} = &\begin{bmatrix}\alpha\left(\mu^2 - x_{m,l}^2 - y_{m,l}^2\right) & -\omega_l\\ \omega_l & \alpha\left(\mu^2 - x_{m,l}^2 - y_{m,l}^2\right)\end{bmatrix}\begin{bmatrix}x_{m,l}\\ y_{m,l}\end{bmatrix}\\ &+ k\cdot\sum_{n\neq m}\overline{\boldsymbol{R}}\left(\theta_n^m\right)\begin{bmatrix}0\\ \frac{x_n+y_n}{\sqrt{x_n^2+y_n^2}}\end{bmatrix}\end{aligned} \tag{3.10}$$

where $\mu$ is set equal for all the oscillators. $\theta_m^n$, with $n = 1, 2, 3$ and $m = 2, 3$, is the phase difference between the m-th and the n-th joints of the same leg, represented by $l = 1, 2, \ldots, 6$. It should be noticed that with this control scheme, in a limb layer, information just start from the hip and never reach it, while knee and ankle are bidirectionally coupled, as represented by the arrows in fig. 3.14. This means that only the second and the third joints of a leg work to adapt the locomotion to the environment, knowing the gait imposed by the upper layer. Through this locomotion controller, the coordination can be adjusted with just few parameters, namely $\mu$ and $\theta_j^i$, reducing the complexity of the control problem. Together with frequencies, these two parameters can be directly

tuned thanks to Hopf oscillator features. This enhances the adaptability of the Boogie locomotion, but the manual modulation is not convenient or optimal and it appears like a challenge. Based on the paper of Ouyang et al., [37] a reinforcement learning-based controller has been employed to optimize the robot locomotion and it will be explored in chapter 4. Figure 3.16 represents the overall implemented bio-inspired locomotion controller architecture.

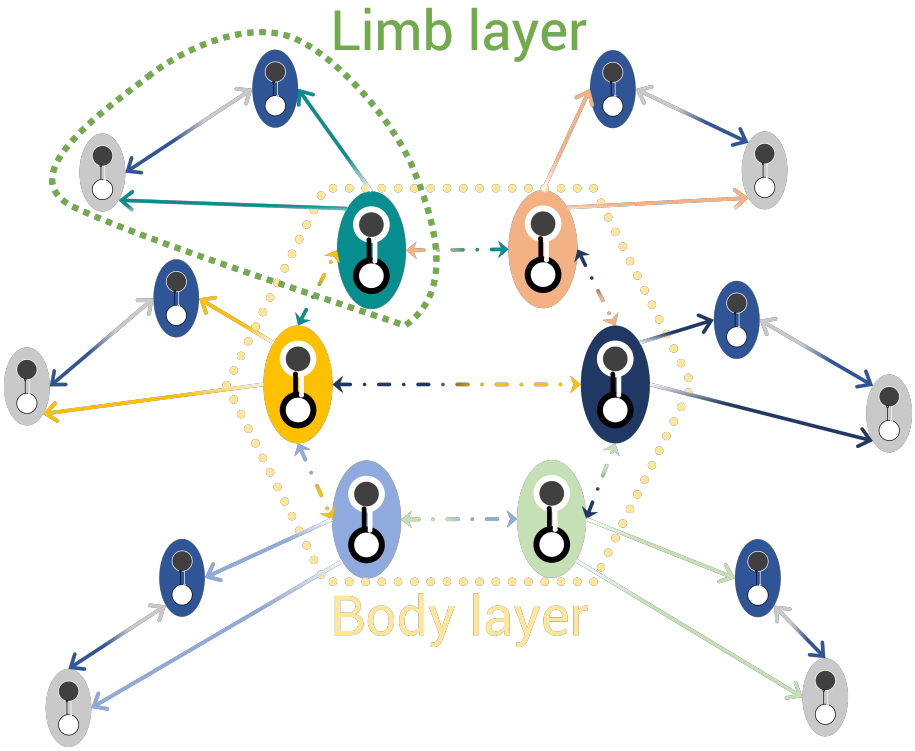



Figure 3.14: Complete two-layer CPG network. The blue and grey oscillators represent femur and tibia movement controllers, respectively, per each limb layer. As represented by the arrows, the body layer is not modified by the six limb layers behaviour; in fact, coxa oscillators interact only with their counterparts.

### 3.3.2. Simulation of limb layer

Repeating what done in section 3.2.4, various simulations have been done to verify the performance of the two-layer CPG-based locomotion controller. This time, the focus has been set on the limb layer of leg 1 in order to show better its working principle. The results presented in fig. 3.15 have been obtained reproducing a tripod gait ($\beta = \frac{1}{2}$) for 10 $s$ and setting $\mu = 1$, $a = 5$, $\alpha = 300$, $\omega_{sw} = 4\pi[\frac{rad}{s}]$, $k = 0.5$. The oscillators' initial conditions are reported in table 3.2 and they have been chosen after some trial and error attempts; it resulted that setting $x_i = 0$ and $y_i = \pm\mu$ is the best choice because it allows the oscillators

to reach the convergence faster. The wanted phase difference between femur and tibia is of 180°, thus $y_{tibia} = -1$. This consideration has been employed also in the next dynamic simulations, but putting $x_i$ equal to the desired initial joint angle displacement, so as not to wait until coordination is reached.

Table 3.2: Limb layer simulation initial conditions.

| | **Coxa** | **Femur** | **Tibia** |
|---|---|---|---|
| $x_{IC}$ | 0 | 0 | 0 |
| $y_{IC}$ | 1 | 1 | -1 |

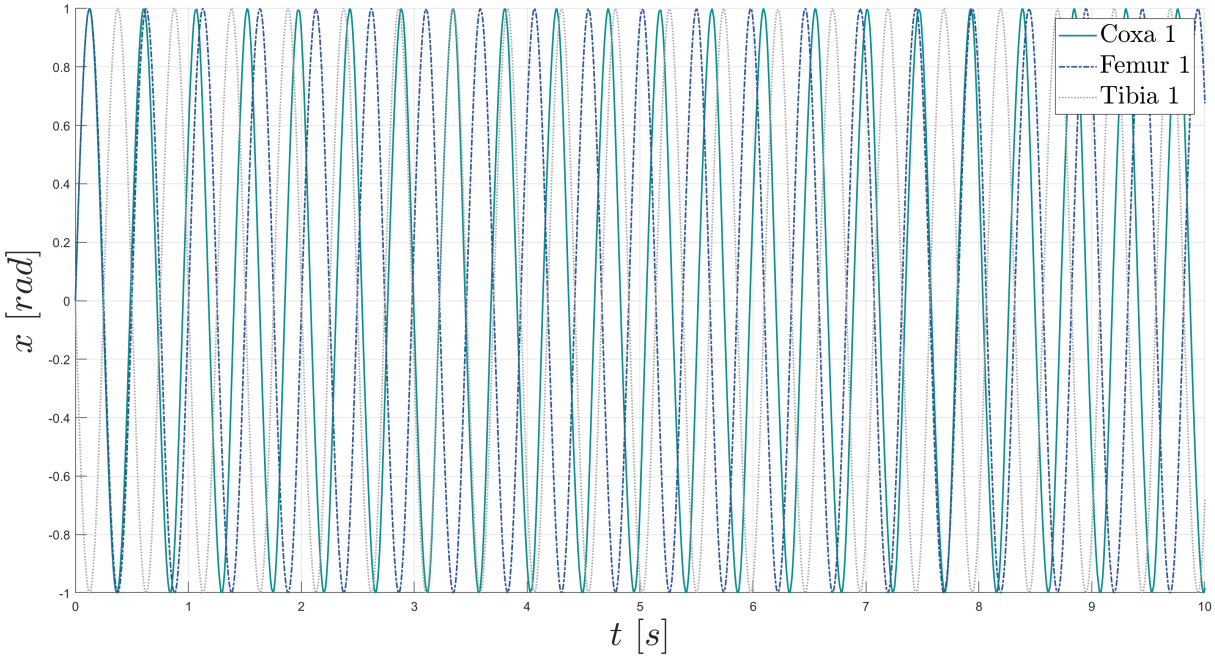



Figure 3.15: Limb layer simulation.

In fig. 3.15, the limb layer receives from the body layer the signal from coxa joint 1; then, it provides two output signals for femur and tibia joints for the interaction with the environment. As desired, setting $\theta_{knee}^{ankle} = \theta_{hip}^{knee} = \theta_{hip}^{ankle} = \pi$, the phase difference between knee joint and ankle joint is locked to 180° in the limb layer.

After having validated the convergence and the correctness of the implemented bio-inspired CPG controller, the RL-based architecture of the control scheme able to generate an optimal locomotion directly tuning the CPG parameters is depicted in fig. 3.16.

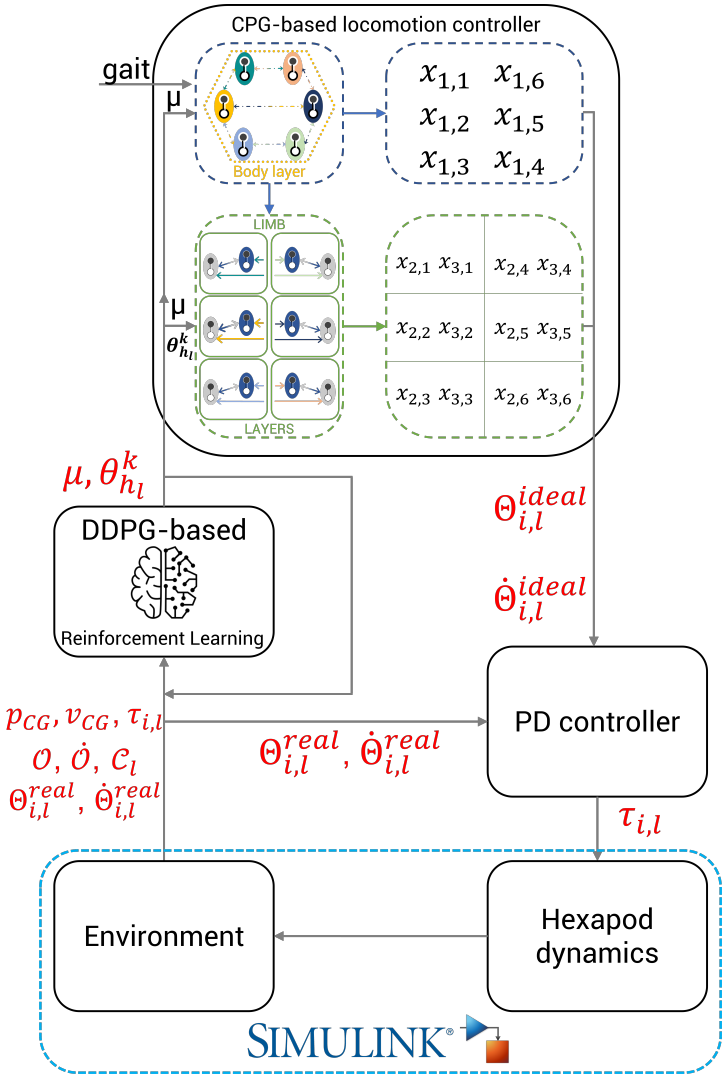



Figure 3.16: Overall bio-inspired locomotion controller architecture. The scheme has a cascaded structure with a feedback loop. Three major parts can be highlighted: [a] The two-layer CPG controller, which generates rhythmic signals to implement the gait selected, provided as input. The CPG parameters are $\mu$ and $\theta^k_{h,l}$, which represent the amplitude and the phase difference between the hip joint and the knee joint of leg l, respectively. The CPG output are $\Theta^{ideal}_{i,l}$, the ideal movement of the i-th limb belonging to the l-th leg, and $\dot{\Theta}^{ideal}_{i,l}$, its derivative. [b] The dynamic model implemented through Simscape ®. A PD controller is employed for determining the 18 torques, namely $\tau_{i,l}$. The Simscape ® simulation outputs the observation state at every time step: $p_{CG} = \left[p_x, p_y, p_z\right]^\top$ and $v_{CG} = \left[v_x, v_y, v_z\right]^\top$ are the hexapod trunk position and velocity, respectively; $\mathcal{O}$ and $\dot{\mathcal{O}}$ are the body orientation and its rate of change; $\Theta^{real}_{i,l}$ and $\dot{\Theta}^{real}_{i,l}$ are the real joint angle and angle velocity, while $\mathcal{C}_l$ is a boolean value for ground contact. The observation vector includes also the action vector at the previous timestep. [c] A neural network trained via DDPG RL, which outputs the parameters of the Hopf oscillators.

# 4 | Reinforcement learning implementation for locomotion optimization

## 4.1. Problem statement

### 4.1.1. Markov Decision Process

Hexapod robot locomotion can be formulated as a Markov Decision Process (MDP) because, at each timestep, its state depends only on the previous state at the antecedent timestep. Indeed, an MDP may be considered as an interaction between an *agent* and the *environment* in discrete timesteps: the environment is the world the agent lives in and interacts with; the agent is the one which takes the action at every step of interaction.
At each timestep $t$, the state vector $s_t \in \mathcal{S}$, describes at the same time the state of the agent and the environment. This vector, rather than a partial observation ($o_t \in \boldsymbol{O}$) of it, is used by the agent to choose the action for the current timestep: the agent takes the real-valued action $a_t \in \mathcal{A} \subset \mathbb{R}^N$ and leads the environment to the very next time step, reaching the new state $s_{t+1}$. In this thesis, $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. The advancement to the next time step happens through an environment state-transition distribution $\mathscr{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0; 1]$. $\mathscr{P}(s'|s, a)$ is the probability that the environment passes from state $s$ to state $s'$ by means of the action $a$. Moreover, when a simulation episode begins, the initial state is random $s_0 \sim \rho(s)$ and each transition process is evaluated by a scalar reward function $\mathscr{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ ([1, 35, 37]).
At this point, the MDP may be defined as a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathscr{R}, \mathscr{P}, \rho)$ ([46]).

### 4.1.2. The RL problem

The sequence of states and actions that happen in the environment is called *trajectory*, *episode* or *rollout* $\varsigma = (s_0, a_0, s_1, a_1, \dots)$. In a MDP, the actor decides the action $a$ to make under the state $s$ following a rule, the stationary policy, which maps states into probability

distribution over actions $\pi : \mathcal{S} \to \mathscr{P}(\mathcal{A})$. $\Pi$ is the set of all stationary policies.
A critical role in RL is played by the reward function, which in this work depends only on the state-action pair:

$$r_t = \mathscr{R}\left(s_t, a_t\right).$$

The *cumulative reward* over a trajectory with a finite horizon h is:

$$R_\varsigma = \sum_{t=0}^{T} \gamma^t r_t = \sum_{(s,a)\in\varsigma} \gamma^t \mathscr{R}\left(s_t, a_t\right)$$

with a discount factor $\gamma \in [0; 1]$ ([24, 35]). The probability of a $T$-step trajectory $\varsigma$ can now be evaluated as:

$$P(\varsigma|\pi) = \rho(s_0) \prod_{t=0}^{T-1} \mathscr{P}(s_{t+1}|s_t, a_t)\pi(a_t|s_t).$$

The objective of the RL is to maximize the *expected return*, $J(\pi)$, defined as:

$$J(\pi) = \int_\varsigma P(\varsigma|\pi)R_\varsigma = \mathop{\mathbb{E}}_{\varsigma\sim\pi}\left[R_\varsigma\right].$$

Thus, the RL optimization problem can be summarised as:

$$\pi^* = \underset{\pi\in\Pi}{argmax} J(\pi)$$

where $\pi^*$ is the *optimal policy* that the agent should select.

## 4.2. Deep deterministic policy gradient

Complex systems like hexapod robots are defined as multiple-input and multiple-output (MIMO) systems due to their high-dimensional state space and action space, which are generally continuous. Dealing with MIMO, stochastic and Deterministic Policy Gradients (DPG) have a crucial difference. The stochastic ones integrates over both state and action spaces, thus requiring immense, time-consuming search in those vast spaces. Quite the opposite, the deterministic policy gradients only integrate over the state space without sampling in $\mathcal{A}$. As a consequence of these behaviours, computing the stochastic policy

gradient requires more samples than the deterministic one, and the difference between the two increases the more dimensions the action has. ([43]).
The Deep Deterministic Policy Gradient (DDPG) is a model-free, online and off-policy RL algorithm formulated for the first time in 2015 ([25]). Using deep function approximators it can address unprocessed, high-dimensional sensory inputs and acquire policies in a high-dimensional continuous action space. The DDPG formulation combines an actor-critic approach with DPG algorithm and it merges their merit too, giving place to a more robust and efficient in learning algorithm. Moreover, its off-policy and deterministic features guarantee a more sample-efficient learning through the ability of generating a deterministic action. Thanks to this features, in the recent years DDPG has been widely adopted in robot control.
In this thesis, a DDPG-based RL approach has been implemented to optimize the locomotion control presented in section 3.3.1 and it is applied on learning the adaptive control policy $\pi$. Since this problem is a deterministic case, $\pi$ is a deterministic policy which returns a single action in a deterministic way $\pi(s) = a$ ([2]) and it is supposed to be parameterized by $\theta^\pi$. Thus, the actor function $\pi(s|\theta^\pi)$ specifies the policy by deterministically mapping states to a specific position action in continuous space. In this way, the RL problem described in section 4.1.2 is translated into finding the optimal value for the parameter $\theta^\pi$. The direction for the update of $\theta^\pi$ to maximise $J(\pi)$ is the gradient of $J(\pi)$ with respect to $\theta^\pi$:

$$\nabla_{\theta^\pi} J(\pi) = \mathbb{E}\left[\nabla_a Q(s,a)\nabla_{\theta^\pi}\pi(s)\right].$$

In addition to the *actor*, the DDPG consists of other 3 neural networks (NNs) in total: the *critic*, the *target actor* and the *target critic*. The critic is also called *Q-value function*, $Q(s,a)$ because it evaluates the Q-value, which is a numerical value that represents the discounted future reward for a state-action pair. The critic is parameterized by $\theta^Q$.
The DDPG adopts two crucial ideas to avoid the instability problem caused by the use of neural network to approximate the actor network and the critic network ([37]):

1. The loss function $L$ can be derived knowing the target function and th NNs as function approximators:

$$L(\theta^Q) = \mathbb{E}\left[(Q(s_t,a_t|\theta^Q) - Y_t)^2\right]$$

where $Y_t$ is the target in a supervised learning sens; it is computed by using the Bellman equation as an intermediate optimum ([2]):

$$Y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}|\theta^Q)).$$

Now, it must be said that another assumption is formulated when NNs are used for RL: *the samples are independently and identically distributed.* Obviously, this presumption is violated when dealing with an environment for the robot locomotion in which samples are generated from sequential exploration. To overcome this, the DDPG makes use of an experience *replay buffer* to store trajectories of interaction experience between policy and environment. At each timestep h, both $\theta^\pi$ and $\theta^Q$ are updated by sampling a mini batch uniformly from the buffer and using the value function, the DPG and the Bellman equation. Specifically, at each timestep h the critic network $\theta^Q$ is updated by minimizing the loss:

$$L = \frac{1}{H}\sum_{h=1}\left[(Q(s_t, a_t|\theta^Q) - Y_h)\right]^2$$

where

$$Y_h = r_h + \gamma Q'\left(s_{h+1}, \pi'(s_{h+1}|\theta^{\pi'})\middle|\theta^{Q'}\right)$$

and $H$ is the mini batch sample's time. At the same time, $\theta^\pi$ is updates with the sampled policy gradient:

$$\nabla_{\theta^\pi} J = \frac{1}{H}\sum_{h=1} \nabla_a Q(s, a|\theta^Q)|_{s=s_h,\ a=\pi(s_h)} \nabla_{\theta^\pi}\pi(s|\theta^\pi)|_{s_h}.$$

2. The DDPG evaluates $Y_t$ by using frozen copies of policy and value functions. They are parameterized by vectors $\theta^{\pi'}$ and $\theta^{Q'}$, respectively. These two copies are implemented to evaluate the target values, while $\pi'$ and $Q'$ slowly track $Q$ and $\pi$ in the original networks as follows:

$$\theta^{Q'} \leftarrow \kappa\theta^Q + (1-\kappa)\theta^{Q'}$$

$$\theta^{\pi'} \leftarrow \kappa\theta^\pi + (1-\kappa)\theta^{\pi'}.$$

$\kappa \ll 1$ is a *hyperparameter* which produces the so called *soft update* by scaling down the update step, in order to enhance the stability avoiding non-stationary target values.

Finally, it must be mentioned the *exploration noise.* In order to guarantee exploration in

the continuous action space, the DDPG employs an exploration policy $\pi'$ in which noise is added to the policy network' actions $\pi$:

$$\pi'(s_t) = \pi(s_t) + \mathcal{N}$$

where $\mathcal{N}$ represents a noise sampled from a noise process and can be chosen to suit the environment. Following the suggestion of Lillicrap et al., ([25]), in the present study the Ornstein-Uhlenbeck process ([48]) was used since it allows temporally correlated exploration for exploration efficiency in physical control problems with inertia.
The structure of the implemented DDPG-based reinforcement learning is represented in fig. 4.1.

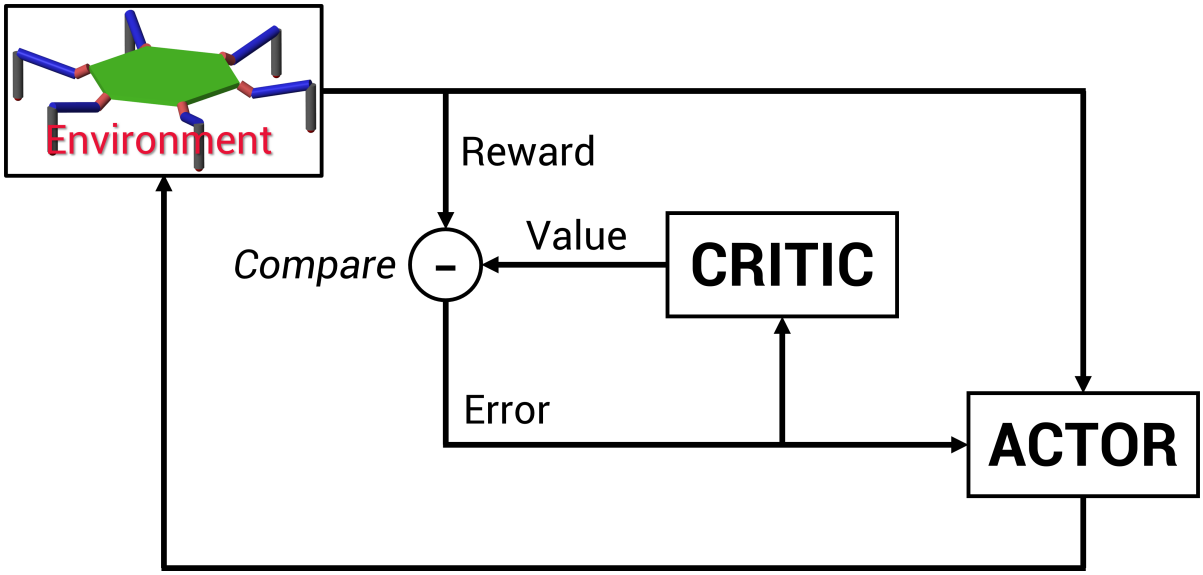
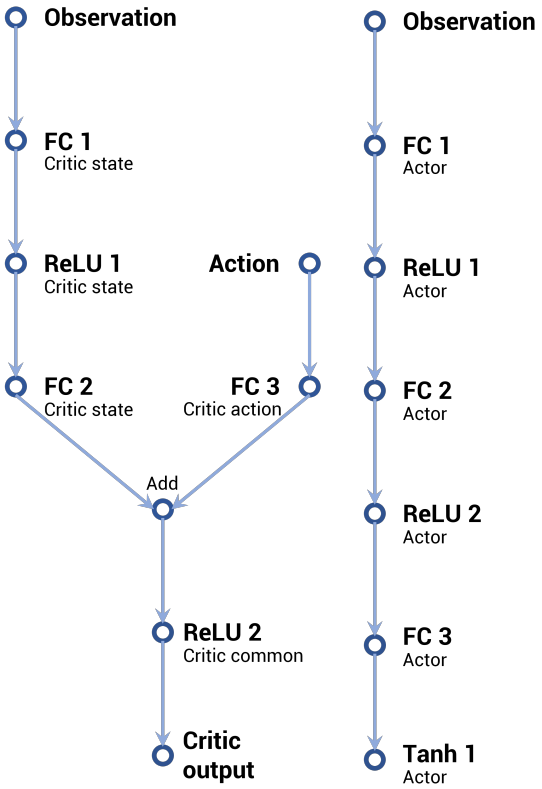



Figure 4.1: DDPG-based reinforcement learning structure.

## 4.3. RL Network architecture

As previously said, in DDPG algorithm the critic and the actor are parameterized as deep neural network, namely $\theta^Q$ and $\theta^\pi$. In this work, their architectures were inspired by the recent study by Fujimoto et al., ([17]) that achieved very good results. The critic network (fig. 4.2a) consists of five hidden layers: three fully-connected (FC) layers and two Rectified Linear Unit (ReLU) layers. Whereas, the actor network (fig. 4.2b) is made of six hidden layers: three FC layers alternated by two ReLU layers, and a final Tanh layer. The latter generates an output in the range $[-1; 1]$, which modulates the two-layer CPG parameters.




(a) Critic network. (b) Actor network.

Figure 4.2: Networks' architecture of the implemented DDPG-based RL.

## 4.4. Action and observation vectors

### 4.4.1. Observation vector

The interaction between Boogie and the surrounding environment happens by means of observations and actions. The actions are taken by the actor, that learns the task receiving information on the real robot via observation vector. In this MDP, the observation vector at time t is defined as:

$$o_t =< p_{CG}, v_{CG}, \tau_{i,l}, \mathcal{O}, \dot{\mathcal{O}}, \mathcal{C}_l, \Theta_{i,l}^{real}, \Theta_{i,l}^{real}, \mathcal{A} >$$

where each element is normalised between $[-1; 1]$.
It is important to specify that the observation vector is only a portion of the state vector. This means that the MDP analysed is not fully observable, e.g. the robot is not equipped with exteroceptive sensors that identify the terrain type or sense obstacles presence. When the MDP is not fully described, it is referred as Partially Observable Markov Decision

Process (POMDP). Since in the present work the agent generates a continuous trajectory and not a discrete action, $o_t$ is sufficient enough for learning the locomotion tasks.

### 4.4.2. Action vector

The action vector $a_t$ outputted by the control policy contains the oscillators' amplitude and the limb layers' coupling parameters for interlimb coordination and terrain adaptation:

$$a_t =< \mu, \theta^k_{h,1}, \theta^k_{h,2}, \theta^k_{h,3}, \theta^k_{h,4}, \theta^k_{h,5}, \theta^k_{h,6} > .$$

$a_t$ is the input of the two-layer CPG architecture which generates the movement for all the 18 joints. As described in section 4.3, the last actor network layer generates real outputs in $[-1; 1]$ and these unitary values must be converted to be feasible with the CPG's inputs. By reason of this, since the output element corresponding to the amplitude, $a_{\mu,t}$, must be positive, it has been translated from $[-1; 1]$ to $[0; 1]$:

$$a_{\mu,t'} = \frac{a_{\mu,t} + 1}{2}.$$

Finally, all the elements in $a_t$ are multiplied by the relative max value. Thus, the CPG inputs are $a_\mu \in [0; \mu_{max}]$ and $a_\theta \in [-\theta_{max}; \theta_{max}]$.

## 4.5. Reward function

The reward function is the incentive mechanism through which the actor's action are evaluated. In an episode, the reward function is used to tell the agent what is correct and what is wrong by means of positive rewards or negative penalties.
The aim of this work is to find a policy that allows the locomotion of the robot. To encourage this process, the adopted reward function $r_t$ motivates the robot moving forward as fast as possible, and penalises the hexapod when it rotates or deviates too much from the desired path. Equation (4.1) formalizes the implemented reward function.

$$r_t = w_{v_x} \cdot v_x + w_{\Delta t} \frac{T_{samp}}{T_s} - w_y \cdot (p_y - p_{y,in})^2 - w_z \cdot (p_z - p_{z,in})^2 - w_{pitch} \cdot \alpha^2_{pitch} \qquad (4.1)$$

where $w_i$ indicates the positive weight for the i-th term, $p_{y,in}$ and $p_{z,in}$ are the initial positions along y and z axes, and $\alpha_{pitch}$ is the pitch angle. The first term provides a positive

reward when the robot has a positive forward velocity. The second term encourages the agent to avoid early episode termination (e.g. the robot falls forward at the beginning of the episode): at each time step a small, constant and positive reward is provided. The remaining terms are the ones that discourage the actor from searching in unwanted states, such as large deviations from the desired height and orientation.

## 4.6. Episode stopping criteria

When training the control policy, the algorithm has complete freedom in exploring the action space. Hence, in some episodes the agent could select some actions that are not physically feasible in reality or that could lead to unstable and unrecoverable states. A flag, called *isdone flag*, was introduced as a stopping criteria for these cases.
The flag is a logical value that is used to instantly terminate the episode when at least one of the following is true:

- The robot moves too far on the lateral direction;
- the height of the trunk CG from the ground is below a certain $h_{min}$, which means the robot has fallen;
- the height of the trunk CG from the ground is above a certain $h_{max}$, which means the robot has jumped too high;
- roll, pitch or yaw angles are outside bounds and the robot could have fallen or rolled over;
- any of the 18 joints touches the ground, invalidating the episode.

# 5 | Dynamic modelling of Boogie hexapod in SimMechanics™

Simulink™ is a block diagram programming (MATLAB™-based) environment.
It is adopted to model multidomain dynamical systems using object-oriented method and to simulate and analyze them before moving to hardware. Within the range of Simulink™ applications, Simscape Multibody™ supplies a simulation environment for 3D mechanical systems, such as robots ([28]). This software has been chosen as the simulation environment due to its ease of use and large-scale integration with MATLAB™ without any need of other specific program.
In this chapter, the main dynamical subsystems modeled in the project will be investigated and described in detail, showing the choice made during the design phase and the trade-off adopted to improve the convergence and the velocity of the training.
In fig. 5.1 the top layer of the Simulink project is depicted. Three major sections can be distinguished: from right to left they are the *physical model*, the *reinforcement learning agent* and the *neural network inputs*, namely the observation vector, the reward function value and the "isdone" flag. Since in chapter 4 the last two have been already largely explored, the following sections will be mostly focused on the physical subsystem.

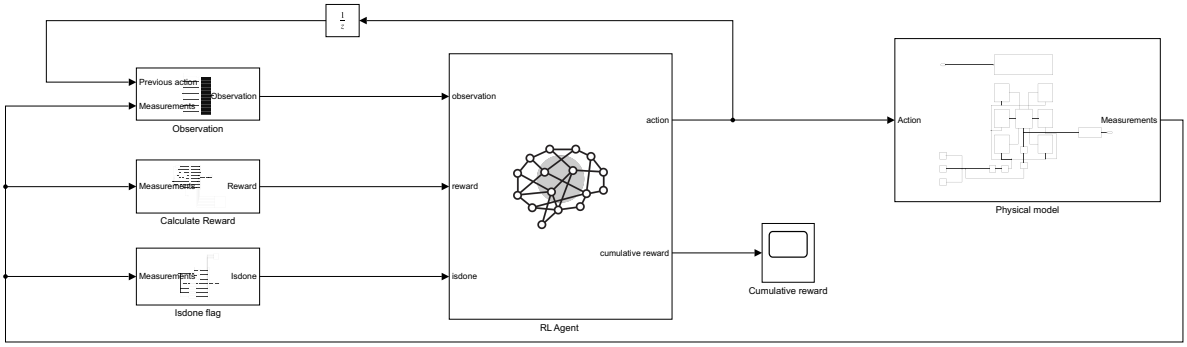
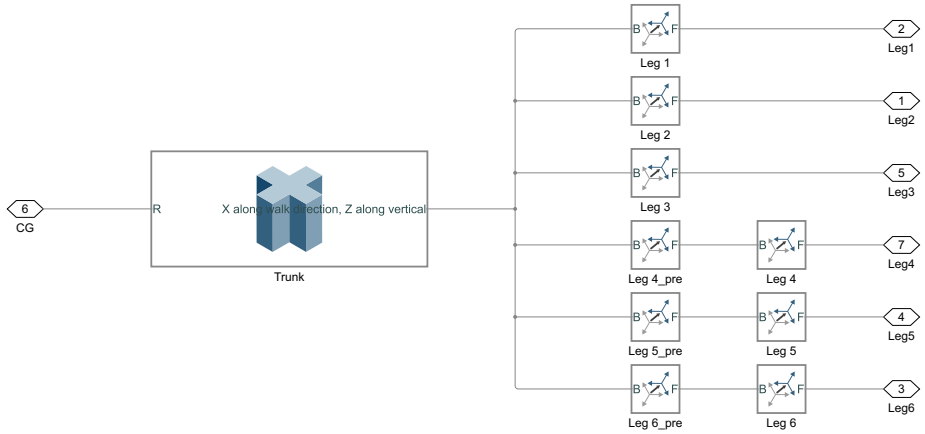


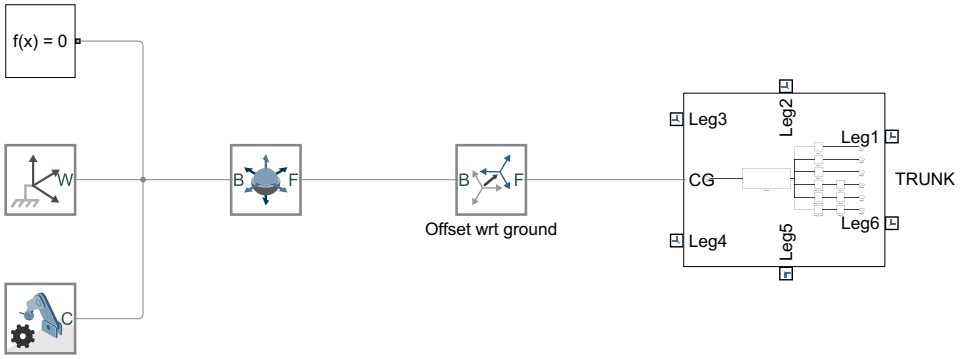

Figure 5.1: Top layer of the proposed Simulink project. The RL agent is the core of the model: based on the measurements coming from the sensors, through the observation vector and the value of the reward function it generates the *action* output that will change the state of the robot.

## 5.1. Body design

For the sake of simplicity, the chassis has been modeled as a mere hexagon using the dimensions shown in section 2.1.1 and the weight is derived assuming the robot made in aluminum, thus considering a density of 2700 $\frac{kg}{m^3}$.
The trunk is the central part of the model: the six legs will be hinged to it and their local reference frames are related to the one of body's center of gravity. Figure 5.2a shows how the reference frames of the six limbs have been positioned starting from the trunk's CG: they have been placed on the six hexagon's vertices with the x-axis pointing radially outward from the center, the z-axis normal to the hexagon's plane and the y-axis orthogonal to the other two completing an orthonormal basis. It must be noticed that the first three legs' reference frames have a z-axis pointing downward, unlike the others: during forward locomotion, a clockwise rotation direction for left coxae (i.e. 1, 2 and 3) and a counterclockwise one for right limbs (i.e. 4, 5 and 6) are required.




(a) Scheme of the legs' reference frame with respect to the trunk CG.




(b) Connection between trunk's CG and world reference frame through a *6-DOFs Joint* block. On the left, starting from the top, there are the *Solver Configuration*, the *World frame* and the *Mechanism Configuration* blocks.

Figure 5.2: Boogie's body block diagram in Simulink™.

The free movement of the robot in the simulation environment is granted by the *6-DOFs Joint* block, represented as a spherical joint in fig. 5.2b. It is not actuated and it is also used to link the hexapod to the environment characteristics, like the gravity vector. This joint plays a fundamental role for the walking experiment, because it allows the robot to rotate and move in any direction with respect to the world reference frame. In this block, the initial conditions (ICs) for the velocity of body's CG have been set equal to 0, since the hexapod starts the simulation from standstill. Instead, ICs for the position have been set using a *Rigid Transform* block (in the image, *offset wrt ground*): knowing knee and ankle initial angles, in order to place the CG at the correct height with respect to the ground, its value has been evaluated using a direct kinematics approach; so, when the simulation begins the feet touch the ground without going through it.
On the left of fig. 5.2b, two more blocks which deserve attention are present. On the top, there is the *Solver configuration* one, used for defining solver settings to employ for the simulation; in this specific case, standard settings have been unchanged. Instead, on the bottom of the figure, the *Mechanism Configuration* block can be found: it sets mechanical and simulation parameters that apply to the robot and it has been used for specifying the uniform gravity vector which the hexapod is subject to. In the simulated scenario, the robot is intended to walk on a flat terrain under the effect of Earth's gravity, thus the vector has been set equal to $\vec{\mathbf{g}} = [0\ 0\ -9.80665]^\top \frac{m}{s^2}$. Obviously, this value can be changed according to the eventual planetary walking to simulate.

## 5.2. Legs design

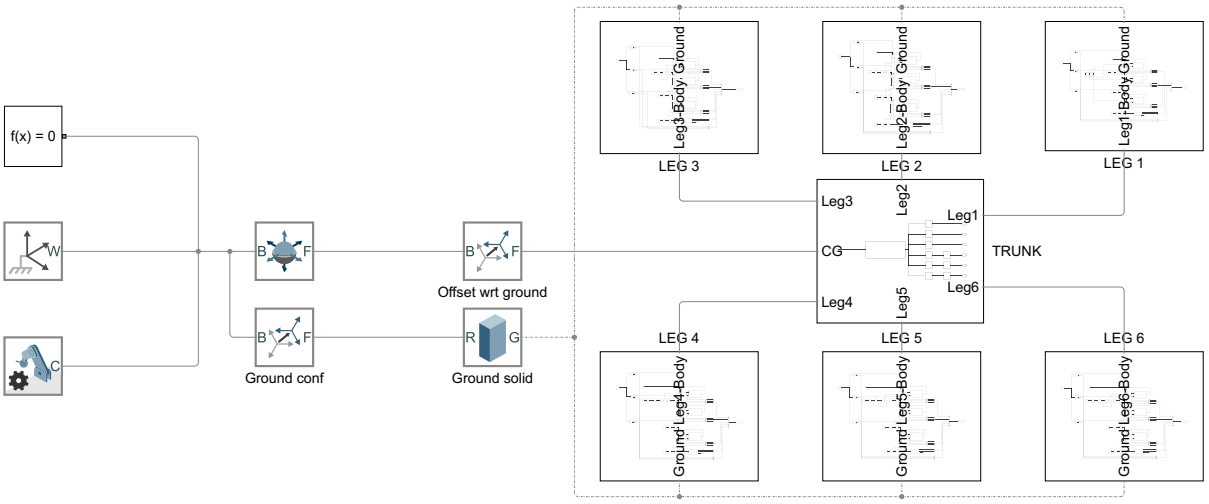



Figure 5.3: Complete Boogie's physical block diagram.

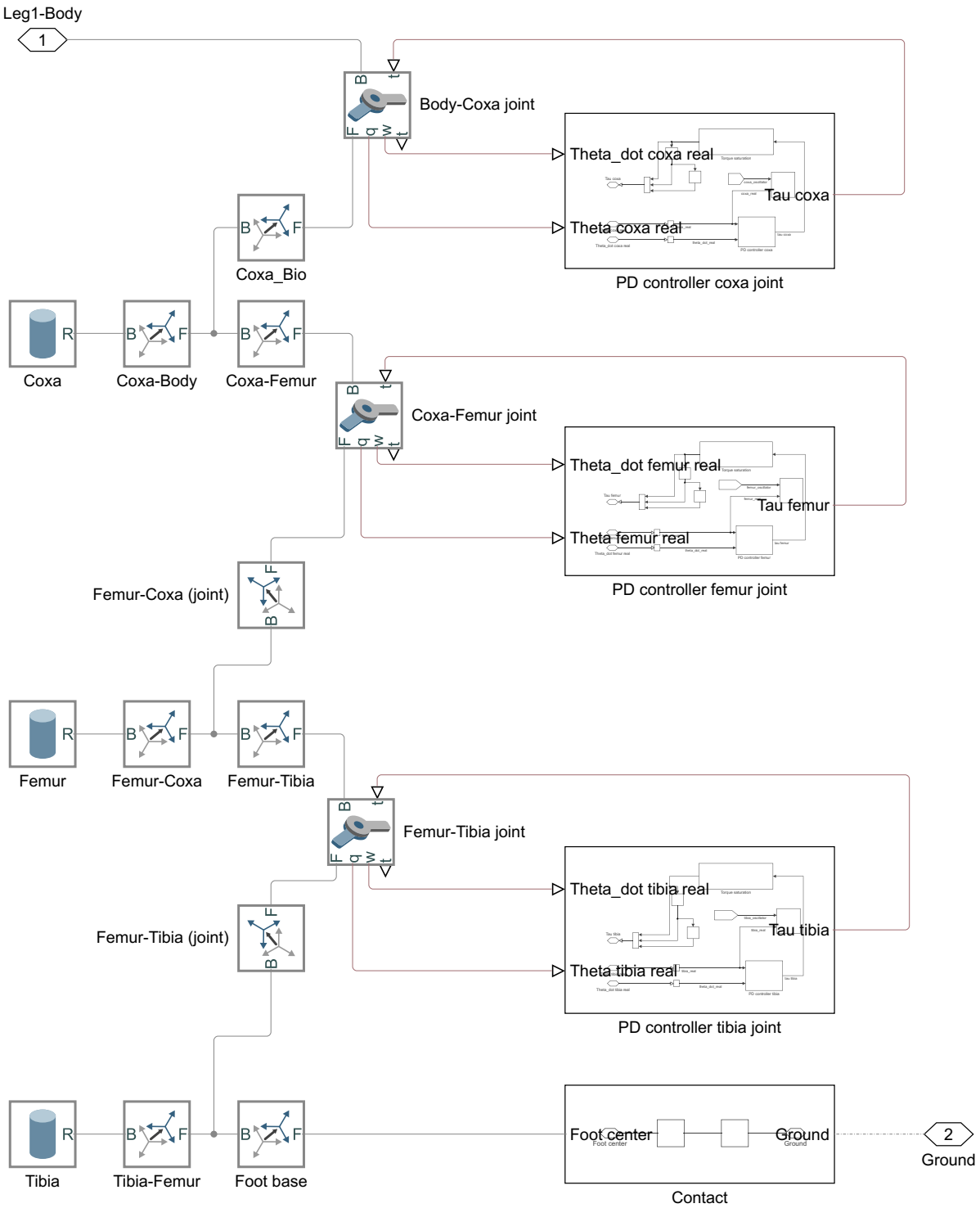



Figure 5.4: Block diagram of a 3 DOFs biologically inspired leg in Simulink™. On the left the *solid* blocks used for modelling the appendages are shown. In the middle, the three *revolute joint* employed to hinge the solids together. On the right, the PD controller subsystems used for torque generation and the ground contact block.

Based on the design choices discussed in chapter 2, the six 3-DOFs legs have been modeled. Coxa, femur and tibia have been shaped as solid cylinders with the *solid* block using the lengths reported in table 2.1 and a radius of $R_{leg} = 1\ cm$; again, the selected material is aluminum. These solids have been linked together by means of *revolute joint* blocks that work like mechanical hinges, allowing only the rotation around the desired axis. These constraints have been treated as ideal hinges fixing the internal mechanism's spring stiffness and damping coefficient to 0. Instead, in order to limit the angle ranges according to table 2.1, the coefficients shown in table 5.1 have been adopted. These values have been selected starting from the ones implemented in two MathWorks projects ([29, 31]), which have aims similar to the one of this work. Then, they were reduced by 30% because of the different order of magnitude in length and to speed up the simulation, due to the higher complexity of the model and the huge number of simulation episodes to perform.

Table 5.1: Coefficients' value to limit angle ranges.

| **Parameter** | **Value** | **Unit** |
|---|---|---|
| Spring stiffness | 350 | $\frac{N \cdot m}{deg}$ |
| Damping coefficient | 35 | $\frac{N \cdot m \cdot s}{deg}$ |
| Transition region width | 2 | $deg$ |

At this point, the whole robot design could be easily concluded by simply connecting the six limb subsystems to the main body block by means of six more *joint*s, obtaining the architecture depicted in fig. 5.3. However, as in every simulation environment, one of the most difficult challenges is to properly model the limits and the restrictions that characterize the physical environment. For this reason, in the following sections the PD controllers and the implemented ground contact methodology will be analysed.

### 5.2.1. PD controller

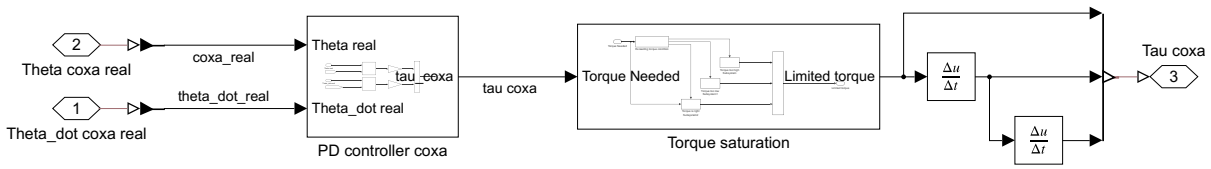



Figure 5.5: Block diagram of the implemented PD controller for torque generation. In figure, the coxa's one is represented, but the scheme is the same also for femur and tibia.

The implementation of the 18 PD controller blocks was necessary to generate the torques needed to move the limbs' appendages complying with the joints' angular position path coming from the oscillators. Indeed, the *joint* blocks receive the torque as input and return the measurements of the joints' angle position and its derivative. The parameters' tuning has been done through trial-and-error simulation and their values are shown in table 5.2. In fig. 5.6 the proper functioning of the PD controllers is shown and described.

Table 5.2: PD controller coefficients.

| **Joints** | $\mathbf{K_p}$ [-] | $\mathbf{K_p}$ [-] |
|---|---|---|
| Hip | -5 | -3 |
| Knee | -5 | -4 |
| Ankle | -4 | -2 |

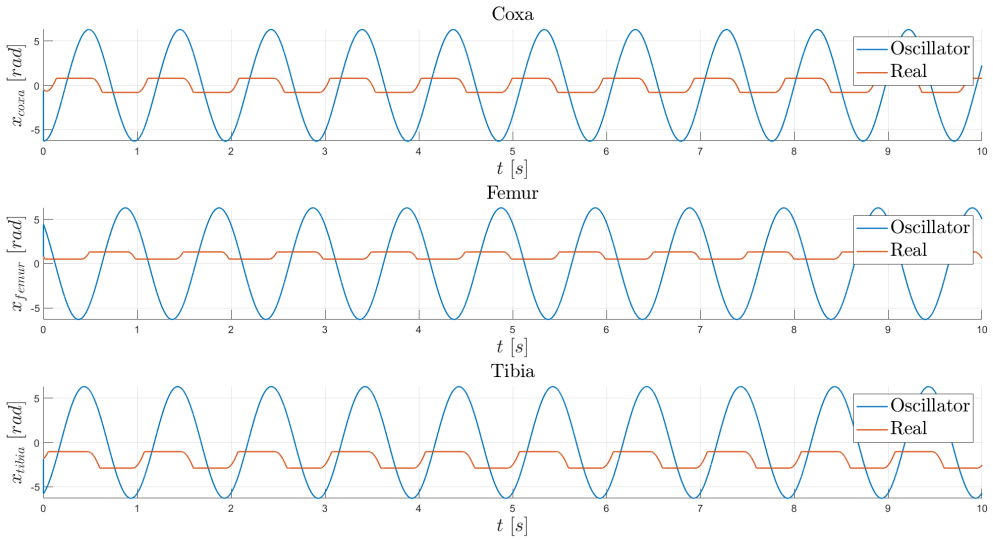



Figure 5.6: Comparison between oscillators' output and real joints' position. It is evident that the two waveforms have the same frequency, but the *real* one anticipates the *oscillator* output at the beginning of the ascending and descending phase. This happens because of the limit on joints' position: the real position is capped, but it starts to change just after the oscillator's minimum or maximum due to the derivative part of the controller.

In order to limit the torques in a physically feasible range, a *torque saturation* subsystem is added in the block diagram, as depicted in fig. 5.5. It checks if the torque outputted by the controller is in the span $[-10; 10]\,Nm$ and, if not, it cap the momentum to the value

$\pm 10\ Nm$ until the output returns in the admissible range. Even if Simulink™ is equipped with a built-in block, called exactly *saturation block*, that performs the same job, it is preferred to directly implement a specific subsystem because the proprietary block slowed down the simulation due to its complexity.

### 5.2.2. Contact force simulation

Different studies, like [8] and [15], have demonstrated that ground contact can be modeled as a mechanical spring-damper system leading to good and reliable simulation results. In Simulink™ this method is implemented with the *Spatial contact force* block, which models the contact between geometries associated with a pair of body ([30]). Setting the *method* parameter to *Smooth Spring-Damper*, the normal contact force is evaluated, according to Newton's Third Law, as

$$f_n = s\,(d) \cdot (k \cdot d + b \cdot d')$$

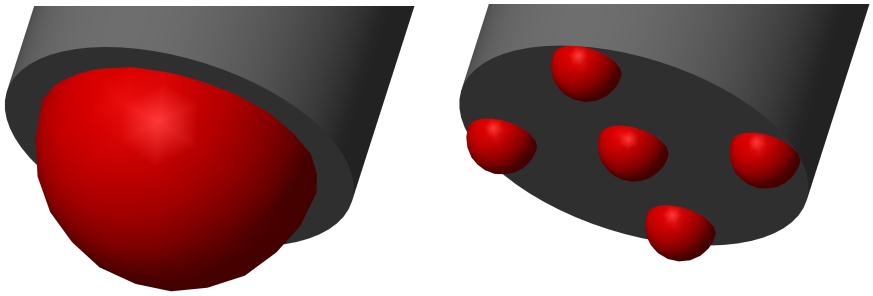
where $d$ is the penetration depth between the two geometries and $d'$ is its first time derivative; $k$ is the normal-force stiffness, or rather, the resistance of the contact spring to geometric penetration; $b$ is the normal-force damping, that is the resistance of the contact damper to motion while the geometries are penetrating; $s(d)$ is the smoothing function. Table 5.3 collects the coefficients' value adopted in the simulation.

Table 5.3: *Spatial contact force* coefficients' value.

| **Parameter** | **Value** | **Unit** |
|---|---|---|
| Stiffness | $1 \times 10^6$ | $\frac{N}{m}$ |
| Damping | $1 \times 10^6$ | $\frac{N \cdot s}{m}$ |
| Transition region width | $1 \times 10^{-3}$ | $m$ |
| Static friction coefficient | 1 | $[-]$ |
| Dynamic friction coefficient | 0.9 | $[-]$ |
| Critical velocity | $1 \times 10^{-3}$ | $\frac{m}{s}$ |

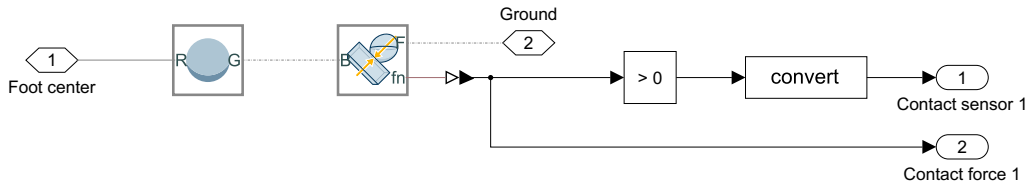
Also in this case, they have been obtained from the MathWorks™ projects cited before ([29, 31]), but can be changed depending on the features and the lay of the ground to simulate. In this work, the floor is modeled as a simple $10^3 \times 10^3 \times 1\ cm$ solid parallelepiped and the hexapod robot is intended to walk on a flat terrain.

Regarding the feet geometry for contact simulation, two options were considered, both based on hemispheres and illustrated in fig. 5.7: the first possibility is to use only one hemisphere of radius $R_{cont} = 0.8R_{leg} = 0.8\ cm$ placed in the foot's center (fig. 5.7a); alternatively, five hemispheres of radius $R_{cont} = 0.2R_{leg} = 0.2\ cm$ organized in a cross-shaped arrangement (fig. 5.7b) could be taken into account. The block diagrams of both configurations are represented in fig. 5.8.

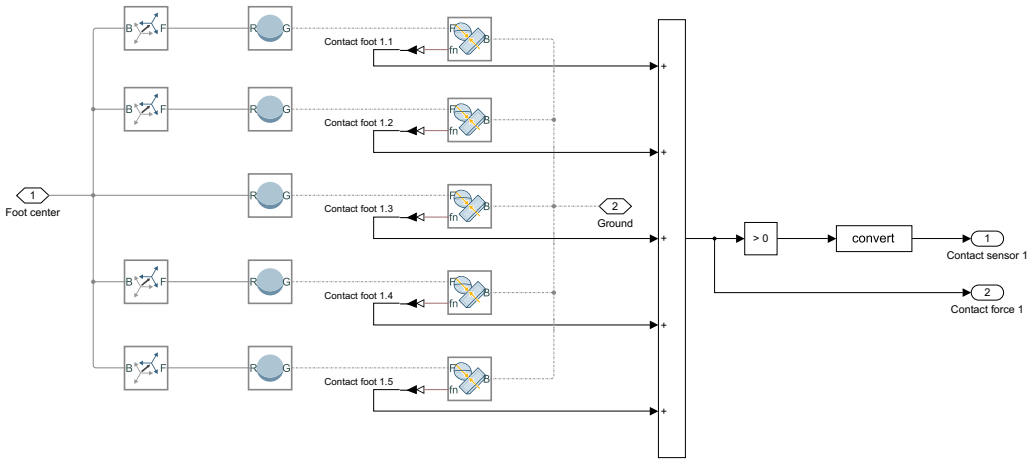


(a) Single, central hemisphere. (b) Five cross-shaped hemispheres.

Figure 5.7: Simulink™ representation of the two considered feet geometry.




(a) Single hemisphere block diagram.




(b) Five, smaller hemispheres block diagram.

Figure 5.8: Block diagram of the two considered feet geometry. The *spatial contact force* blocks output the normal force acting on the hemispheres, then these information are converted into a boolean value, $\mathcal{C}_l$, that indicates when ground contact happens.

Although using multiple geometries leads to a better accuracy in mapping the contact point and a higher precision in contact simulation, it also corresponds to a way bigger computational cost: in the presented situation, the number of geometries varies from 6 to 30, thus it is evident why the chosen solution is the single hemisphere case for this preliminary study.

## 5.3. Physical model validation

A simplified walking path was implemented on a Boogie prototype to validate the physical model before proceeding with the RL agent implementation. This tripod locomotion was generated using trigonometric functions for coxa and femur's movement, while the tibia's angle was derived from an inverse kinematics approach with the aim to keep constant the CG's height to its initial value $h_{in}$. The adopted functions are reported below.

$$\begin{cases} \underline{\text{Legs 1, 3 and 5}} \\ \varphi_1 = \frac{\pi}{4}\sin(\pi t) \\ \begin{cases} \varphi_2 = \frac{\pi}{3}\cos(\pi t), & \cos(\pi t) \geq 0 \\ \varphi_2 = 0, & \cos(\pi t) < 0 \end{cases} \\ \varphi_3 = \arcsin\left(\frac{h_{in} - L_f \cdot \sin\varphi_2}{L_t + R_{cont}}\right) + \pi - \varphi_2 \end{cases} \qquad \begin{cases} \underline{\text{Legs 2, 4 and 6}} \\ \varphi_1 = \frac{\pi}{4}\sin(\pi(t+1)) \\ \begin{cases} \varphi_2 = \frac{\pi}{3}\cos(\pi(t+1)), \cos(\pi(t+1)) \geq 0 \\ \varphi_2 = 0, & \cos(\pi(t+1)) < 0 \end{cases} \\ \varphi_3 = \arcsin\left(\frac{h_{in} - L_f \cdot \sin\varphi_2}{L_t + R_{cont}}\right) + \pi - \varphi_2 \end{cases}$$

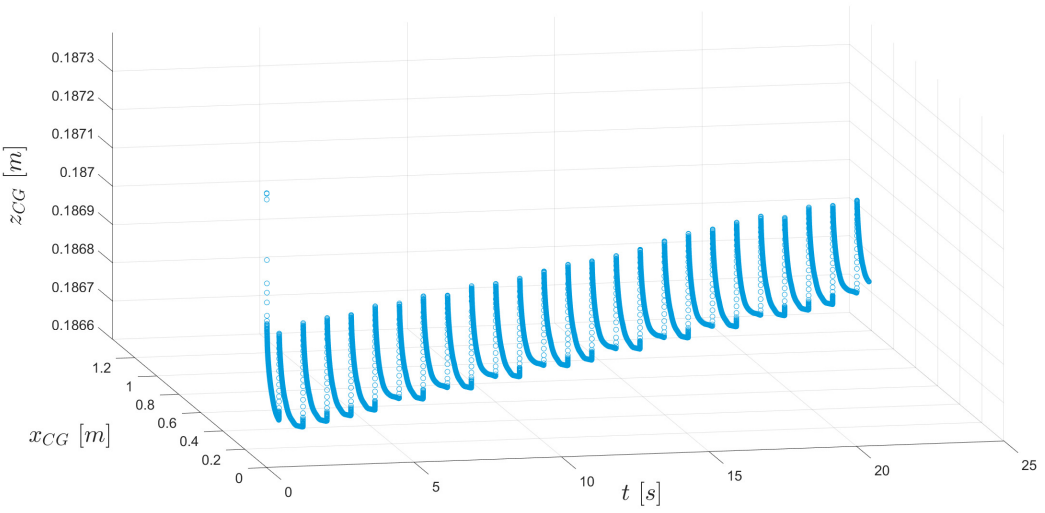
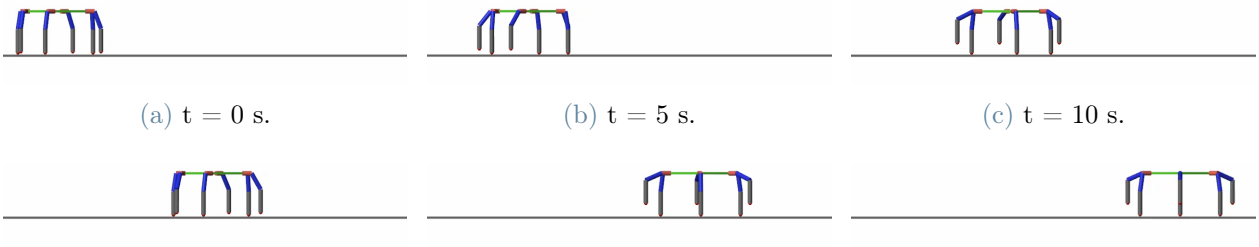



Figure 5.9: Trunk's CG displacement. As desired, the height is almost constant while the robot is moving forward.



(a) t = 0 s. (b) t = 5 s. (c) t = 10 s.

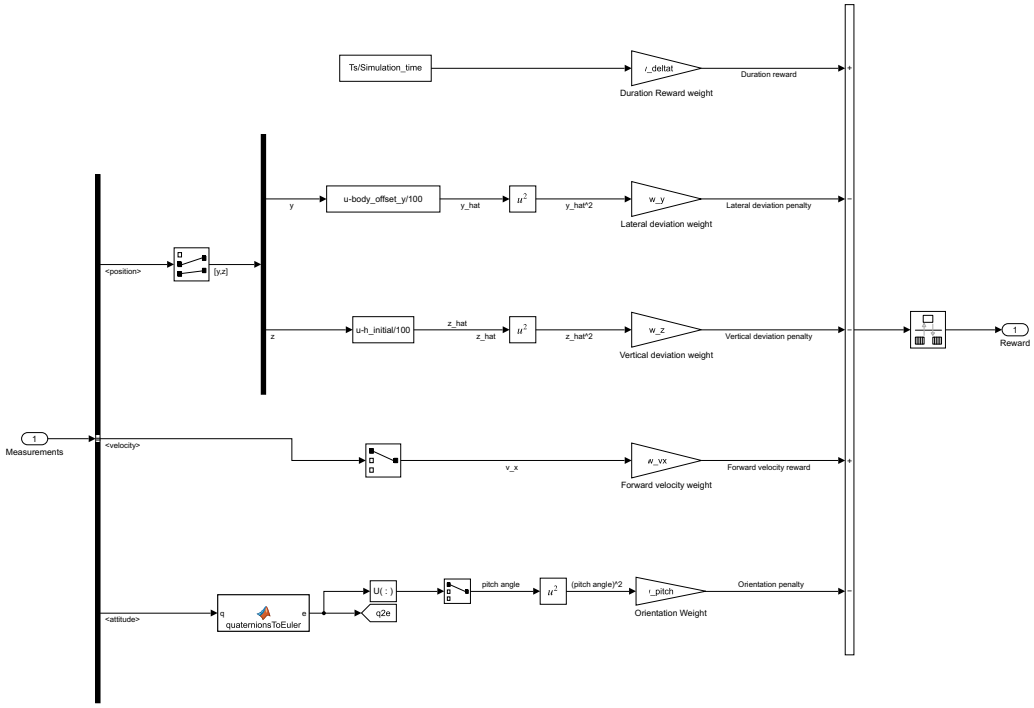
(d) t = 15 s. (e) t = 20 s. (f) t = 25 s.

Figure 5.10: Hexapod movement over simulation time. The six pictures represent the $xz$-plane with $x$-axis pointing right and $z$-axis pointing upward.

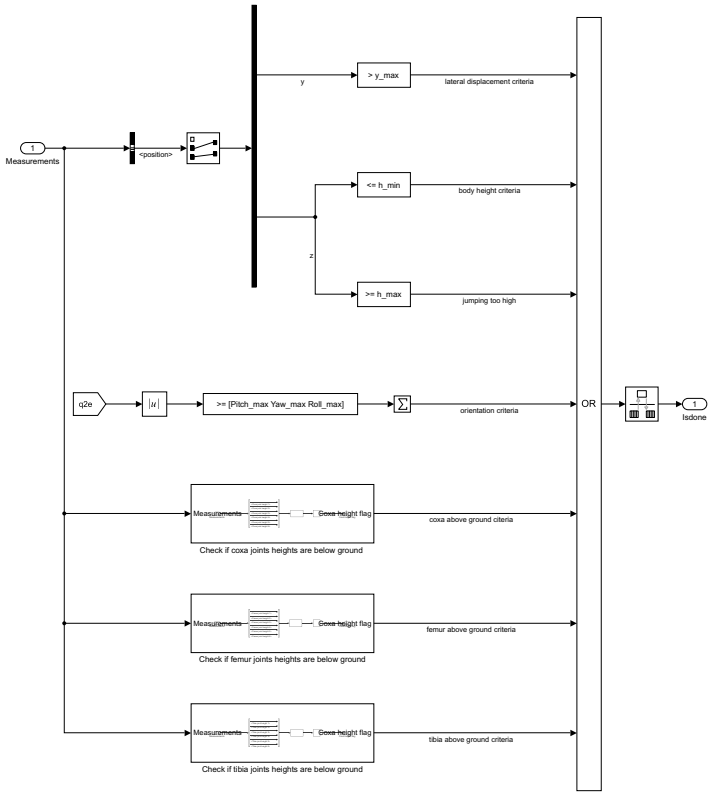
Figures 5.9 and 5.10 shows the result obtained during validation test. As clearly depicted in fig. 5.9, the trunk's CG remains at constant height while moving forward. Figure 5.10 depicts six frames of the simulation, taken every $5s$: the robot moves correctly along $x$-axis (going right) and it cyclically alternates odd-numbered and even-numbered legs.
It may therefore be concluded that the modelled physical environment is appropriate for the simulation since the PD controllers and the *revolute joint*s work as expected, allowing the hexapod movement. Also the contact force model is satisfactory because it allows the robot to propel itself through the force applied on the ground by the limbs.

## 5.4. Reinforcement learning agent

The RL agent was implemented using the homonymous Simulink™ block, represented in the center of fig. 5.1. The *RL agent* uses the physical model described so far as a training and simulation environment and interacts with it to generate the *action* vector. The interaction with the environment is possible by virtue of the observation vectors, in which all the measurements are collected.
In addition to that, the block receives in input also the reward and the *isdone flag* (section 4.6). The first block evaluates the cumulative episode reward based on the reward function presented in section 4.5 starting from the sensors' measurements (fig. 5.11a. Instead, the second subsystem compute the logical *isdone flag*, which defines when an episode must be interrupted because the robot has explored some choices that may damage it or that are physically unacceptable (fig. 5.11b); this block too depends on the observation vector.




(a) *Calculate reward* subsystem block.




(b) *Isdone flag* subsystem block.

Figure 5.11: *RL agent* inputs' subsystem block.

# 6 | Simulation and Results

## 6.1. Simulation setup

The simulations have been set up in MATLAB™ and performed using Simulink SimMechanics. They were performed on a cluster equipped with an AMD Opteron 6376 with 64 cores and a frequency of 2.3 GHz. The cluster allows to train the agent performing 11 episodes in parallel using the *Parallel Computing Toolbox™*, thus reducing the time required to complete one entire simulation.
Due to the high computational cost and the consequent computational time required by the complex simulation, it has been decided to investigate only the tripod gait on a flat terrain in order to simplify the problem and produce a preliminary result. Moreover, to further decrease the complexity of the policy research and the simulation time, the action vector to find was reduced to 3 elements: the amplitude and the phase differences between the hip joint and the knee joint of even legs and odd legs.

$$a_t =< \mu, \theta^k_{h,even}, \theta^k_{h,odd} > .$$

Obviously, this has been possible because the selected gait was the tripod one.
In the following tables, the values of the main adopted parameters are recollected.

Table 6.1: Oscillators initial conditions.

| | $\mathbf{Coxa_{1,3,5}}$ | $\mathbf{Femur_{1,3,5}}$ | $\mathbf{Tibia_{1,3,5}}$ | $\mathbf{Coxa_{2,4,6}}$ | $\mathbf{Femur_{2,4,6}}$ | $\mathbf{Tibia_{2,4,6}}$ |
|---|---|---|---|---|---|---|
| $\mathbf{x_{IC}}$ [rad] | $\frac{\pi}{4}$ | $\frac{\pi}{4}$ | $-\frac{7\pi}{12}$ | $-\frac{\pi}{4}$ | $\frac{\pi}{4}$ | $-\frac{7\pi}{12}$ |
| $\mathbf{y_{IC}}$ [rad] | 0 | $-\frac{\pi}{4}$ | $\frac{\pi}{4}$ | 0 | $\frac{\pi}{4}$ | $-\frac{\pi}{4}$ |

Table 6.2: Oscillators parameters value.

| Parameter | Value | Unit |
|---|---|---|
| $T_s$ | 10 | $s$ |
| $\alpha$ | 300 | $[-]$ |
| $\omega_{sw}$ | $\pi$ | $\frac{rad}{s}$ |
| $k$ | 0.5 | $[-]$ |
| $a$ | 5 | $[-]$ |

Table 6.3: RL parameters value.

| Parameter | Value | Unit |
|---|---|---|
| $w_{v_x}$ | 2 | $[-]$ |
| $w_{\Delta t}$ | 0.6 | $[-]$ |
| $w_y$ | 0.9 | $[-]$ |
| $w_z$ | 0.4 | $[-]$ |
| $\mu_{max}$ | $\pi$ | $[-]$ |
| $\theta^k_{h,l}\|_{max}$ | $\frac{2}{3}\pi$ | $rad$ |
| $T_{sample}$ | 0.05 | $s$ |

Table 6.4: Hyper-parameter choices for the implemented simulation.

| Hyper-parameter | Value |
|---|---|
| Critic Learning Rate | $10^{-3}$ |
| Actor Learning Rate | $10^{-4}$ |
| Optimizer | Adam |
| Target Update Rate | $10^{-3}$ |
| Batch Size | 128 |
| Iterations per time step | 1 |
| Discount Factor | 0.99 |
| Normalized Observations | True |
| Exploration Policy | OU, $\theta = 0.15$, $\mu = 0$, $\sigma = 0.3$ |
| Max episodes number | 2500 |
| Max steps number per episode | 200 |

## 6.2. Simulation results

The simulation took almost 50 hours to perform a little more than 1600 episodes. Since the adopted stopping criteria was an average reward equal or higher than 30 for over 500 episodes, as shown in fig. 6.1 the simulation stopped before the max number threshold reported in table 6.4. It can be seen that for the first 100 episodes the reward increases and then stabilizes around 10; then it collapses to 0 just after a local peak of 30. Until episode 580, the reward is practically 0 because the agent explores a portion of the action space in which the hexapod robot jumps over the imposed limit, thus causing episodes to end at the very beginning. From the $580^{th}$ episode, the agent starts to explore different action configurations and the rewards become more significant. From this episode the average reward firstly decreases, where some episode are evaluated even under -100, then it increases and stabilizes around 30, with a slight negative slope towards the end.

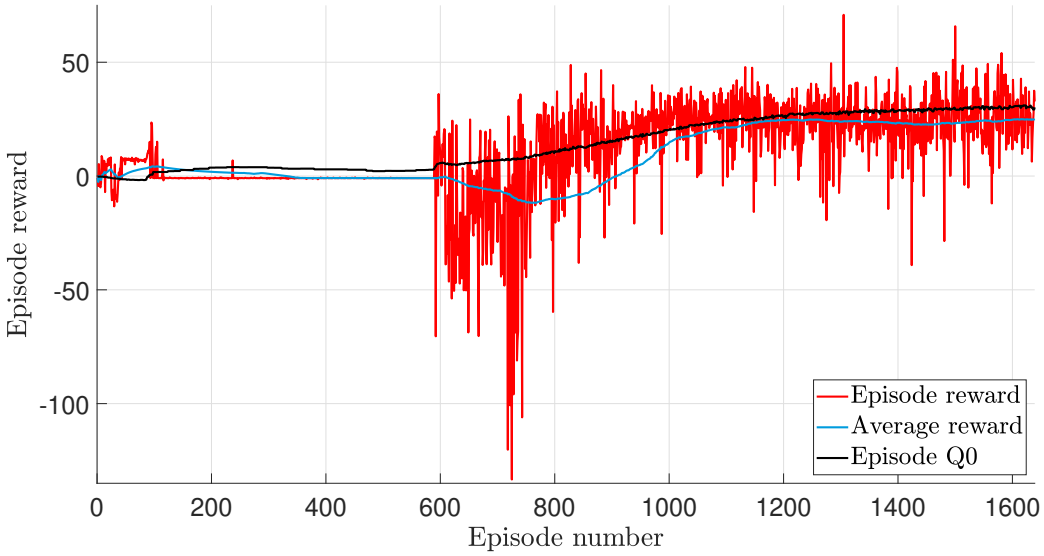
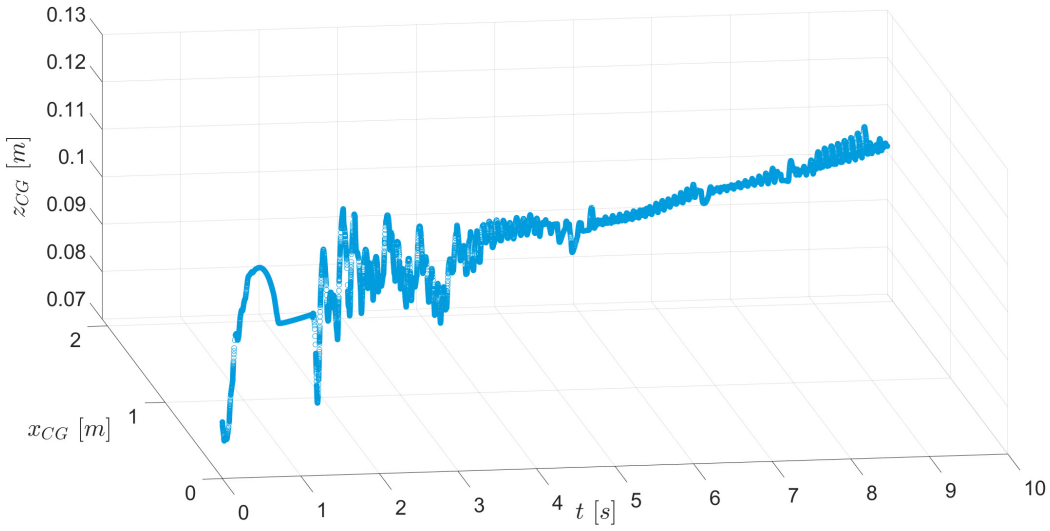


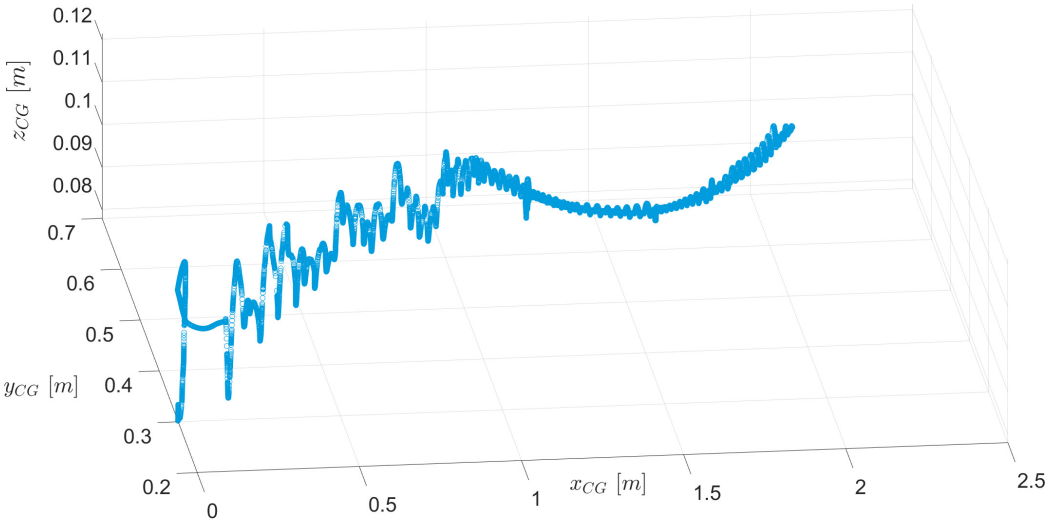

Figure 6.1: RL training episode reward.

From the $800^{th}$ episode, some episode rewards are equal or even higher than 50, for example episode 1305 has a reward of 70.84 and 65.82 is assigned to number 1500. Since 70.84 is the greatest reward obtained in this simulation, the relative episode will be analyzed in the detail.

Figure 6.2 depicts the displacement of the body CG in the three-dimensional space over time: the agent makes the robot move continuously forward along the $x$-axis, with small lateral and vertical variations. Initially, the hexapod robot walks almost in a straight line, but from the fourth second it starts to move along a circular path.




(a) Trunk CG displacement in time.




(b) Trunk CG displacement in space.

Figure 6.2: Body CG displacement.

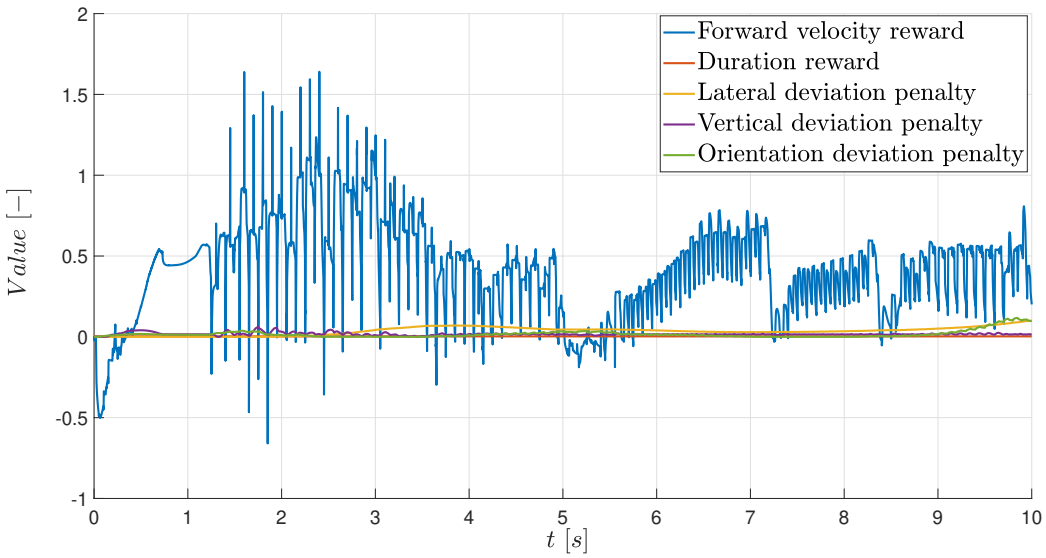


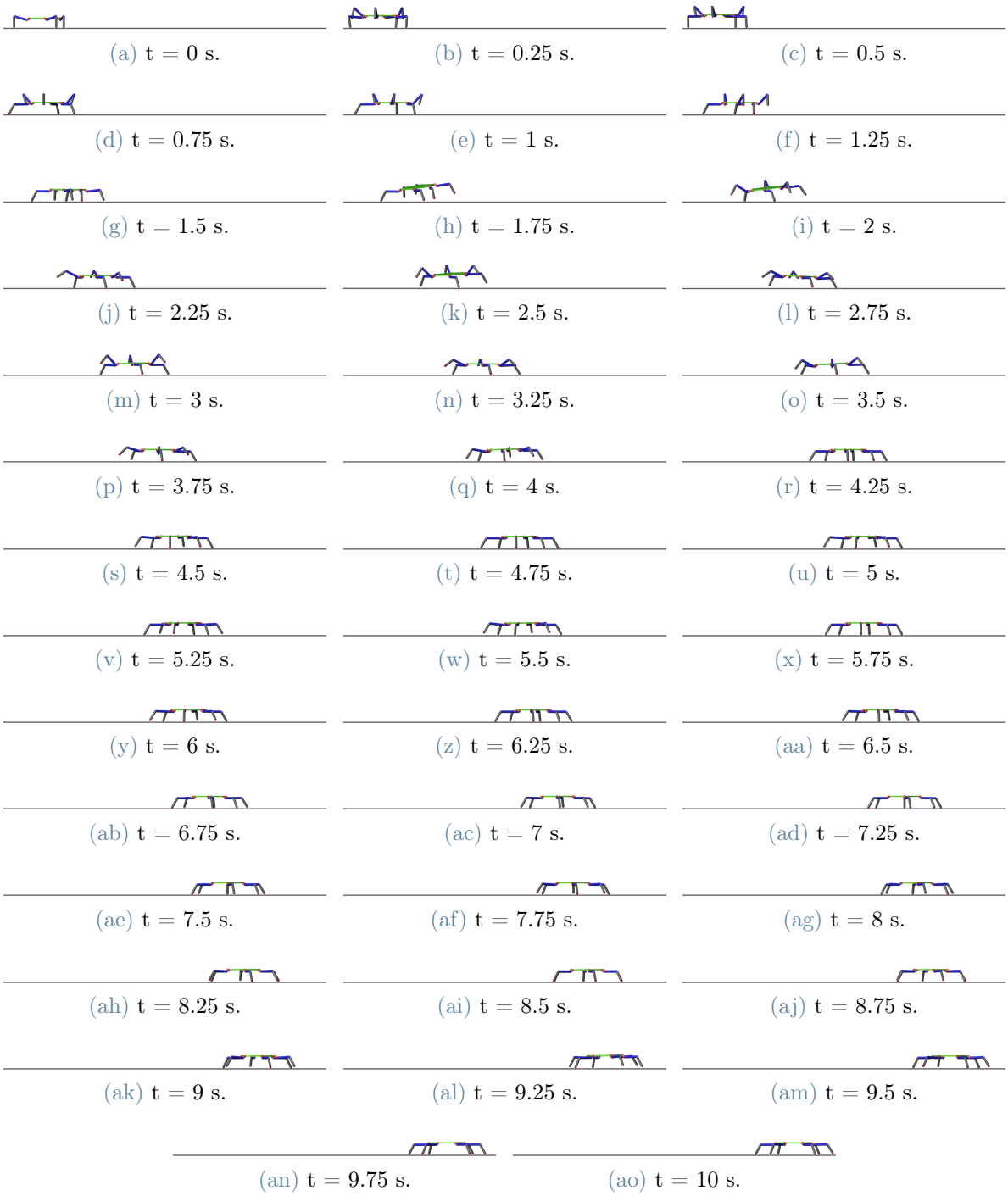

Figure 6.3: RL training reward function.

Looking at the evolution of the reward function's terms, represented in fig. 6.3, the forward velocity reward stands out from the others for its magnitude. It is negative only at the beginning and in some sporadic instants, while in the rest of time it has a positive value. Knowing that $w_{v_x} = 2$, it can be deduced that in the first half the velocity along $x$ reaches peaks of $v_x = 0.75\ \frac{m}{s}$, while in the second half it is limited between $0\ \frac{m}{s}$ and $0.45\ \frac{m}{s}$.

These results could appear consistent with the desired ones, but they must be contextualised by the effective locomotion of the robot, represented in fig. 6.4. The initial movement looks like a correct tripod gait step, in fact three legs are in stance position and the other three limbs are swigging. However, after that the robot starts to move forward using an irregular locomotion pattern, which is not the expected one: from 1.5 s it starts to move using only three limbs to jump forward, while the other three legs are kept in the air. In some instants, like figs. 6.4h, 6.4i and 6.4k, only one or two feet touch the ground in order to propel the body forward.

An unconventional behaviour could also be found in the coxa oscillators' output (fig. 6.5): unlike the plots shown in section 3.2.4, they intersect the zero line just one time and oscillate around small non-zero values. Despite this, it must be highlighted that the implemented interlimb coordination method has proven to be robust enough to always grant the desired phase lag between the six coxae.

All these unexpected and undesired events can be explained looking at the neural network outputs. The amplitude $\mu$ is depicted in fig. 6.6, while the phase lags are represented in fig. 6.7. After the first second, the amplitude plot resembles a square wave that changes



(a) t = 0 s. (b) t = 0.25 s. (c) t = 0.5 s.

(d) t = 0.75 s. (e) t = 1 s. (f) t = 1.25 s.

(g) t = 1.5 s. (h) t = 1.75 s. (i) t = 2 s.

(j) t = 2.25 s. (k) t = 2.5 s. (l) t = 2.75 s.

(m) t = 3 s. (n) t = 3.25 s. (o) t = 3.5 s.

(p) t = 3.75 s. (q) t = 4 s. (r) t = 4.25 s.

(s) t = 4.5 s. (t) t = 4.75 s. (u) t = 5 s.

(v) t = 5.25 s. (w) t = 5.5 s. (x) t = 5.75 s.

(y) t = 6 s. (z) t = 6.25 s. (aa) t = 6.5 s.

(ab) t = 6.75 s. (ac) t = 7 s. (ad) t = 7.25 s.

(ae) t = 7.5 s. (af) t = 7.75 s. (ag) t = 8 s.

(ah) t = 8.25 s. (ai) t = 8.5 s. (aj) t = 8.75 s.

(ak) t = 9 s. (al) t = 9.25 s. (am) t = 9.5 s.

(an) t = 9.75 s. (ao) t = 10 s.

Figure 6.4: Hexapod movement over simulation time.

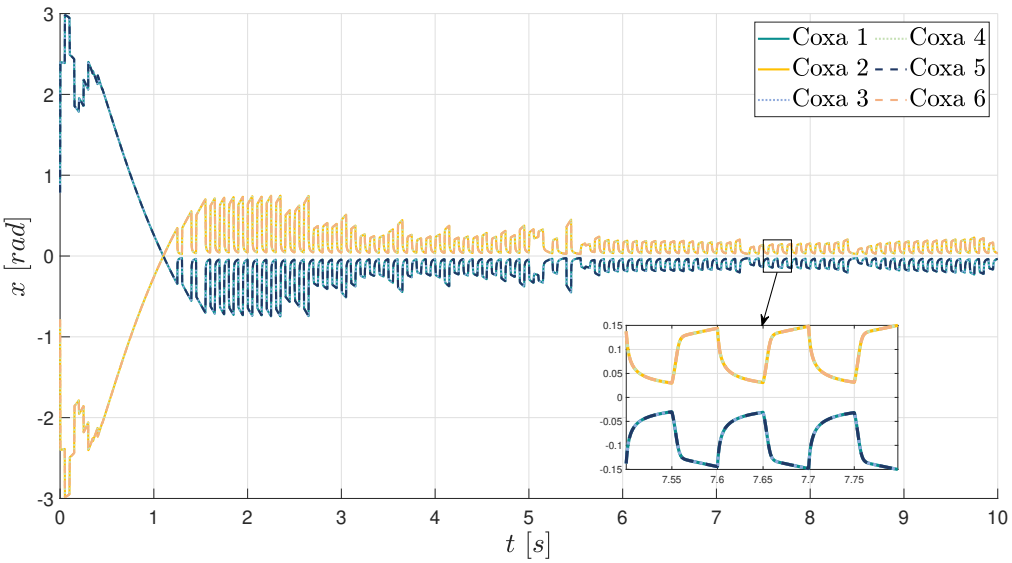



Figure 6.5: Coxa oscillators output.

every 0.05 $s$, which is the sampling time. This is the reason why the Hopf oscillators are not able to reproduce the desired movement: in order to maximize the forward velocity, the actor chooses a policy that generates some "micro vibrations" and to achieve this behaviour it continuously alternates $\mu = 0$ and $\mu = \pi$. Consequently, after the first two seconds, the phase lags are kept almost constant and equal to $\theta^k_{h,l}|_{max} = \frac{2}{3}\pi\ rad$.

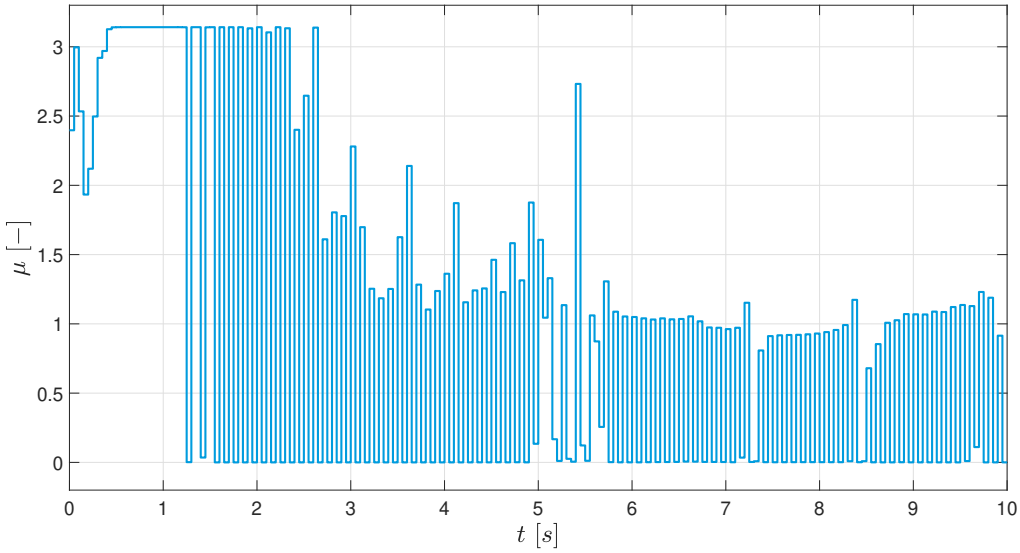
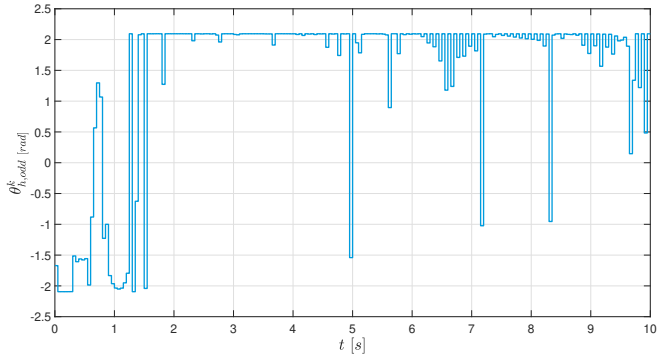
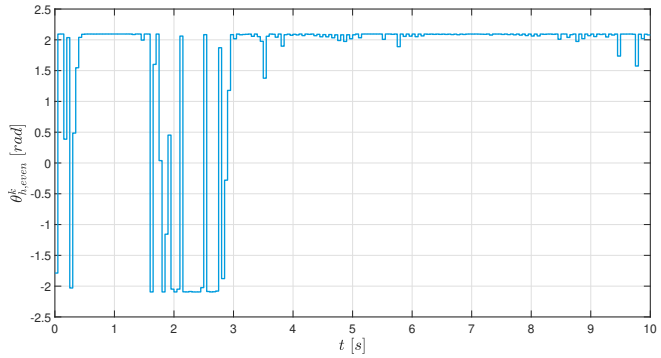



Figure 6.6: Neural network amplitude output.




(a) Odd legs.




(b) Even legs.

Figure 6.7: Neural network phase lag outputs.

# 7 | Conclusions and future developments

This thesis aims to lay the foundations for a space exploration hexapod robot. It investigates both a feasible architecture and an adaptive locomotion controller required to achieve this goal.

Concerning the configuration a radial-symmetric body has been chosen in order to have no preferential direction for walking, while legs' architecture is the result of a biomimetic approach, since it draws inspiration from stereotypical insect legs for augmented movement flexibility. Instead, the proposed locomotion approach is inspired by neurobiological control systems and it consists of an artificial set of coupled Hopf oscillators. The CPG controller contains two hierarchical layers. The first one controls the coxae joints to reproduces the three most common locomotion patterns for hexapod robots: tripod gait, ripple gait and wave gait. Instead, the second layer is used to regulate the limbs behaviour by means of a RL-based learning algorithm (DDPG) which tunes its amplitude and phase lag.

Several numerical simulations have been conducted to validate the proposed controller and verify its effectiveness. In this thesis the best one has been considered. Even though the first layer interlimb coordination has proved to be robust enough to unexpected conditions, the simulation has shown that the RL agent could not be implemented on a real hexapod robot. The obtained preliminary results demonstrate that the two-layer CPG works as desired, but the reinforcement learning agent must be changed. Therefore, future studies could try to train the neural network by:

- changing the hyper-parameter used in the simulation, trying to find out the best ones;
- removing the amplitude output in order to further simplify the model and focusing on the phase lag outputs, which are the one that have not been explored by the agent;

- modifying the reward function introducing penalties for undesired behaviours, e.g. limiting the joints angular velocity.

# Bibliography

[1] T. Azayev and K. Zimmerman. Blind hexapod locomotion in complex terrain with gait adaptation using deep reinforcement learning and classification. *Journal of Intelligent & Robotic Systems*, 99:659–671, 09 2020. doi: 10.1007/s10846-020-01162-8.

[2] N. Bach, A. Melnik, M. Schilling, T. Korthals, and H. Ritter. Learn to move through a combination of policy gradient algorithms: Ddpg, d4pg, and td3. In G. Nicosia, V. Ojha, E. La Malfa, G. Jansen, V. Sciacca, P. Pardalos, G. Giuffrida, and R. Umeton, editors, *Machine Learning, Optimization, and Data Science*, pages 631–644, Cham, 2020. Springer International Publishing. ISBN 978-3-030-64580-9. doi: 10.1007/978-3-030-64580-9-52.

[3] 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*, 19(17), 2019. ISSN 1424-8220. doi: 10.3390/s19173705.

[4] R. Campos, V. Matos, and C. Santos. Hexapod locomotion: A nonlinear dynamical systems approach. Master's thesis, Universidade do Minho, Escola de Engenharia, 12 2010. doi: 10.1109/IECON.2010.5675454.

[5] R. Chen and B. Dunbar. Viper mission overview, 2022. `https://www.nasa.gov/viper/overview`. [Online; accessed 10-September-2022].

[6] S.-K. Chu and G.-H. Pang. Comparison between different model of hexapod robot in fault-tolerant gait. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 32(6):752–756, 2002. doi: 10.1109/TSMCA.2002.807066.

[7] J. Coelho, F. Ribeiro, B. Dias, G. Lopes, and P. Flores. Trends in the control of hexapod robots: A survey. *Robotics*, 10(3), 2021. ISSN 2218-6581. doi: 10.3390/robotics10030100.

[8] E. Corral, R. G. Moreno, M. J. G. García, and C. Castejón. Nonlinear phenomena of contact in multibody systems dynamics: a review. *Nonlinear Dynamics*, 104:1269 – 1295, 2021. ISSN 1573-269X. doi: 10.1007/s11071-021-06344-z.

[9] F. Delcomyn. Neural basis of rhythmic behavior in animals. *Science*, 210(4469): 492–498, 1980. doi: 10.1126/science.7423199.

[10] X. Ding, Z. Wang, A. Rovetta, and J. Zhu. Locomotion analysis of hexapod robot. In B. Miripour, editor, *Climbing and Walking Robots*, Rijeka, 2010. IntechOpen. doi: 10.5772/8822.

[11] R. Dodge, D. Parsons, M. Abid, K. Chrystal, and B. Kartolov. Dynamics associated with the corer on m2020 perseverance rover. In *2021 IEEE Aerospace Conference (50100)*, pages 1–13, 2021. doi: 10.1109/AERO50100.2021.9438361.

[12] ESA. Landing on the Moon and returning home: Heracles, 2021. `https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/Exploration/Landing_on_the_Moon_and_returning_home_Heracles`. [Online; accessed 8-September-2022].

[13] ESA. Exomars - has life ever existed on Mars?, 2022. `https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/Exploration/ExoMars`. [Online; accessed 10-September-2022].

[14] E. Fitcher and B. Fichter. A survey of legs of insects and spiders from a kinematic perspective. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 984–986 vol.2, 1988. doi: 10.1109/ROBOT.1988.12188.

[15] P. Flores and H. M. Lankarani. *Contact Force Models for Multibody Dynamics*. Springer Cham, 2016. ISBN 978-3-319-80911-3. doi:10.1007/978-3-319-30897-5.

[16] A. Frigon and S. Rossignol. Experiments and models of sensorimotor interactions during locomotion. *Biological cybernetics*, 95:607–27, 01 2007. doi: 10.1007/s00422-006-0129-x.

[17] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods, 2018. URL `https://arxiv.org/abs/1802.09477`. doi: 10.48550/ARXIV.1802.09477.

[18] S. Grillner. Locomotion in vertebrates: central mechanisms and reflex interaction. *Physiological Review*, 55(2):247–304, 1975. doi: 10.1152/physrev.1975.55.2.247.

[19] S. Grillner. *Control of Locomotion in Bipeds, Tetrapods, and Fish*, volume 2, pages 1179–1236. John Wiley & Sons, Ltd, 1981. doi: 10.1002/cphy.cp010226.

[20] S. Grillner, T. Deliagina, A. El Manira, R. Hill, G. Orlovsky, P. Wallén, Ö. Ekeberg, and A. Lansner. Neural networks that co-ordinate locomotion and body orientation

in lamprey. *Trends in Neurosciences*, 18(6):270–279, 1995. ISSN 0166-2236. doi: 10.1016/0166-2236(95)80008-P.

[21] P. Gullan and P. Cranston. *The Insects: An Outline of Entomology*. Blackwell Pub, 5 edition, 10 2014. ISBN 978-1-118.84615.5.

[22] S. Hayati, G. Udomkesmalee, and R. T. Caffrey. Initiating the 2002 mars science laboratory (msl) focused technology program. *2004 IEEE Aerospace Conference Proceedings*, 1:638–652 Vol.1, 2004.

[23] B. Klaassen, R. Linnemann, D. Spenneberg, and F. Kirchner. Biomimetic walking robot scorpion: Control and modeling. *Robotics and Autonomous Systems*, 41:69–76, 11 2002. doi: 10.1016/S0921-8890(02)00258-0.

[24] A. Kume, E. Matsumoto, K. Takahashi, W. Ko, and J. Tan. Map-based multi-policy reinforcement learning: Enhancing adaptability of robots by deep reinforcement learning, 2017. doi: 10.48550/ARXIV.1710.06117.

[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning, 2015. doi: 110.48550/arxiv.1509.02971.

[26] R. A. Lindemann and C. J. Voorhees. Mars exploration rover mobility assembly design, test and performance. *2005 IEEE International Conference on Systems, Man and Cybernetics*, 1:450–455 Vol. 1, 2005.

[27] A. Mahajan and F. Figueroa. Four-legged intelligent mobile autonomous robot. *Robotics & Computer-lntegrated Manufacturing*, 13(1):51–61, 1997. ISSN 0736-5845. doi: 10.1016/S0736-5845(96)00028-2.

[28] MathWorks™. Simscape Multibody, model and simulate multibody mechanical systems, 2022. `https://ch.mathworks.com/products/simscape-multibody.html`. [Online; accessed 24-August-2022].

[29] MathWorks™. Train biped robot to walk using reinforcement learning agents, 2022. `https://it.mathworks.com/help/reinforcement-learning/ug/train-biped-robot-to-walk-using-reinforcement-learning-agents.html`. [Online; accessed 31-August-2022].

[30] MathWorks™. Spatial contact force, 2022. `https://it.mathworks.com/help/physmod/sm/ref/spatialcontactforce.html`. [Online; accessed 31-August-2022].

[31] MathWorks™. Quadruped robot locomotion using ddpg agent,

2022. https://it.mathworks.com/help/reinforcement-learning/ug/quadruped-robot-locomotion-using-ddpg-gent.html. [Online; accessed 31-August-2022].

[32] NASA. MARS exploration rovers, 2018. https://mars.nasa.gov/mer/index.cfm. [Online; accessed 10-September-2022].

[33] NASA. MARS 2020 mission - Perseverance rover, 2022. https://mars.nasa.gov/mars2020/. [Online; accessed 9-September-2022].

[34] M. Olaru Sorin, N. Mircea, and V. Stoian. Hexapod robot. Mathematical support for modeling and control. In *15th International Conference on System Theory, Control and Computing*, pages 1–6, 2011.

[35] OpenAI. Introduction to RL - Part 1: Key Concepts in RL, 2018. https://spinningup.openai.com/en/latest/spinningup/rl_intro.html#the-optimal-q-function-and-the-optimal-action. [Online; accessed 3-September-2022].

[36] G. Orlovsky, T. G. Deliagina, and S. Grillner. *Neuronal Control of Locomotion: From Mollusc to Man.* OUP Oxford, 1999. ISBN 9780198524052. doi: 10.1093/acprof:oso/9780198524052.001.0001.

[37] 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*, 15, 2021. ISSN 1662-5218. doi: 10.3389/fnbot.2021.627157.

[38] A. Preumont, P. Alexandre, and D. Ghuys. Gait analysis and implementation of a six leg walking machine. In *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*, pages 941–945 vol.2, 1991. doi: 10.1109/ICAR.1991.240551.

[39] P. Ramdya, R. Thandiackal, R. Cherney, T. Asselborn, R. Benton, A. J. Ijspeert, and D. Floreano. Climbing favours the tripod gait over alternative faster insect gaits. *Nature Communications*, 8, 2017. doi: 10.1038/ncomms14494.

[40] A. Rankin, M. Maimone, J. Biesiadecki, N. Patel, D. Levine, and O. Toupet. Driving Curiosity: Mars Rover mobility trends during the first seven years. In *2020 IEEE Aerospace Conference*, pages 1–19, 2020. doi: 10.1109/AERO47225.2020.9172469.

[41] L. Righetti and A. J. Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *2008 IEEE International Conference on Robotics and Automation*, pages 819–824, 2008. doi: 10.1109/ROBOT.2008.4543306.

[42] K. Seo, S.-J. Chung, and J.-J. E. Slotine. Cpg-based control of a turtle-like underwater vehicle. *Autonomous Robots*, 28:247 – 269, 2010. ISSN 1573-7527. doi: 10.1007/s10514-009-9169-0.

[43] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page I–387–I–395. JMLR.org, 2014. doi: 10.5555/3044805.3044850.

[44] S. M. Song and K. J. Waldron. Machines that walk: The adaptive suspension vehicle. *Robotica*, 7(4):368–368, 1989. doi: 10.1017/S0263574700006937.

[45] Y. Takahashi, T. Arai, Y. Mae, K. Inoue, and N. Koyachi. Development of multi-limb robot with omnidirectional manipulability and mobility. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, volume 3, pages 2012–2017 vol.3, 2000. doi: 10.1109/IROS.2000.895266.

[46] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots, 2018. doi: 10.48550/ARXIV.1804.10332.

[47] F. Tedeschi and G. Carbone. Design issues for hexapod walking robots. *Robotics*, 3 (2):181–206, 2014. ISSN 2218-6581. doi: 10.3390/robotics3020181.

[48] G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Phys. Rev.*, 36:823–841, Sep 1930. doi: 10.1103/PhysRev.36.823.

[49] D. Vidhyaprakash and A. Elango. Studies on Affecting Factors of Wheel Slip and Odometry Error on Real-Time of Wheeled Mobile Robots: A Review. *International Journal of Mechanical, Industrial and Aerospace Sciences*, 9.0(7), Jan. 2017. doi: 10.5281/zenodo.1339726.

[50] D. M. Wilson. Insect walking. *Annual Review of Entomology*, 11(1):103–122, 1966. doi: 10.1146/annurev.en.11.010166.000535.

[51] J. Zhao, H. Zhang, Y. Liu, J. Yan, X. Zang, and Z. Zhou. Development of the hexapod robot hitcr-ii for walking on unstructured terrain. In *2012 IEEE International Conference on Mechatronics and Automation*, pages 64–69, 2012. doi: 10.1109/ICMA.2012.6282808.

[52] M. Žák, J. Rozman, and F. V. Zbořil. Design and control of 7-dof omni-directional

hexapod robot. *Open Computer Science*, 11(1):80–89, 2021. doi: 10.1515/comp-2020-0189.

# Acknowledgements

Alla fine di questi sei, lunghi anni è difficile riuscire a sintetizzare tutta la mia gratitudine verso chi mi è stato accanto, per lungo o poco tempo, e mi ha permesso di poter arrivare a questo enorme traguardo.

Devo ringraziare, anzitutto, i miei genitori per tutti i loro sacrifici di questi anni, per avermi supportato nelle mie scelte quando volevo arrendermi e sopportato quando qualcosa andava male ed era difficile starmi attorno. Grazie anche a te, Ale, anche se non parliamo molto spero tu sappia che ti voglio un bene fraterno. Ora tocca a te, in bocca al lupo!
Un grazie in generale alla mia famiglia, che, nonostante la lontananza, non è mai veramente stata distante. Un grazie anche a chi ora non c'è più, ma è stato importante per la mia crescita e maturazione.

Grazie Teo per essere stato il fratello maggiore che non ho mai avuto, per aver dato ascolto a ogni mia singola preoccupazione e per avermi assecondato in ogni mio desiderio.

Grazie Andre per la nostra amicizia, per tutti questi anni, per tutto ciò che hai fatto senza che ti chiedessi niente. Grazie per la tua musica, per quella che mi hai fatto conoscere, per le playlist che fai solo per me. Grazie per tutti gli instore, tutti i concerti e quanto di bello abbiamo passato assieme. Traccia 8 per noi ha un significato diverso.

Grazie Eli, Bi, Marco, Marta, Fabio e Ale del gruppo *Misto?* per avermi accolto nel vostro gruppo e per l'amicizia sincera di questi anni.

Grazie Tony ed Ema per il supporto nella stesura di questa tesi e per il legame che si è creato nonostante la distanza. Grazie più in generale a tutto il gruppo dei lupi, che ora è diventato un continuo invito a nozze: grazie Bea, Giusy, Ale, Vale, Lois e Fabri per la compagnia quotidiana e soprattutto per quella della prima quarantena.

Grazie a tutti i miei amici del *Noumeno*: Add, Linda, Mariagiulia, Nico, Fede, Dave, Luca, Pano, Pisa e Ida. Grazie per tutti i bei momenti trascorsi assieme e scusate per tutte le volte che noi ingegneri abbiamo intasato i vostri discorsi lamentandoci del Politecnico.

Grazie Giù, Ale, Jaco, Oscar, Ste e tutto il gruppo degli aerospaziali. Solo voi sapete realmente quanto sono stati duri questi anni e vi ringrazio per averli condivisi insieme. Grazie soprattuto a Richi per essere stato il miglior compagno di viaggi, per tutti i passaggi, per tutte le difficoltà, per tutti gli esami che (purtroppo solo in triennale) abbiamo condiviso.

Grazie ai miei colleghi della magistrale per tutti i progetti fatti assieme e per aver reso gli ultimi tre anni un'esperienza indimenticabile. Grazie in particolar modo a Davide e Marco per il confronto costante nella nostra avventura comune.
Grazie soprattutto a Simo per esserci sempre stato nei miei momenti di difficoltà ed aver sempre creduto, ben più di me, nelle mie capacità. Grazie per avermi insegnato tanto e per tutta la pazienza che hai avuto con me, sei un santo.

Grazie infine a chi, da ben più di due anni, mi sta accanto e mi sostiene ogni giorno. Grazie, Ele, per il bene e l'affetto che solo tu sai dimostrarmi e per tutto il resto che già sai.