Technical University of Crete, Greece
School of Electronic and Computer Engineering

Diploma Thesis

Bio-inspired Motion Control of the Humanoid Robot NAO using Central Pattern Generators (CPGs)

Vasileios Kousanakis

Chania, June 2015

Πολυτεχνειο Κρητης

ΣΧΟΛΗ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Διπλωματικη Εργασια

Βιολογικά-εμπνευσμένος Ελεγχος Κίνησης του Ανθρωποειδούς Ρομπότ NAO με Χρήση Κεντρικών Γεννητριών Προτύπων

Βασίλειος Κουσανάκης

Χανιά, Ιούνιος 2015

Abstract

Recent developments in computational and hardware systems enabled the development of more complex artificial systems. In particular, this has brought to life new challenges in the field of robotics, such as the development of humanoid robots with the ability to interact with complex environments, like those where humans live and act. Research has addressed the locomotion of such robots and the way they interact with humans. However, the increasing complexity in the design of the body of humanoid robots, necessitates the development of more satisfactory control methodologies than the existing ones. For that purpose, research in robotics has attempted to get inspired by the study of living organisms, which demonstrate robust and adaptable locomotor behavior. Key components of their motion control are neural networks, called Central Pattern Generators (CPGs), such as those located on the spine of the vertebrate bodies, and are responsible for the production of rhythmic control signals during walking, running, swimming or flying, even in the absence of sensory feedback. Reproducing robustly such motion control schemes in humanoid robots represents a significant challenge, made even more important from the requirement of controlling fast and efficiently a large number of degrees of freedom in such robots.

In this thesis, we examined through robotic experiments on the NAO humanoid robot, the robustness of its built-in walking behavior on steps, inclines, as well as on rugged and granular substrates. The robot and its built-in walking behavior are developed by the company Aldebaran Robotics. The analysis of the experiments showed that the robot has some difficulty in dealing with such substrates. At this point, the idea of using a completely different approach to create a walking behavior for the robot, came up. An attempt is made to set up and exploit a CPGs-based motion control scheme, as an alternative way of achieving a stable walking locomotion for the simulated humanoid robot NAO, using the Webots simulator. The type of Central Pattern Generators selected is based on the Hopf nonlinear oscillator, which has properties useful for the control of the joints of the robot during walking, and is able to reproduce them in a satisfactory way. The procedure followed starts with the recording of the joint trajectories of the robot during walking, then these are learned by the CPGs via a Hebbian learning process, and, then, the CPGs provide desired joint trajectories to the robot, in order to reproduce the walking gait. Furthermore, an optimization process based on genetic algorithms is employed, to achieve the fine tuning of the CPGs via a well-defined objective function.

Περίληψη

Οι πρόσφατες εξελίξεις, τόσο σε υπολογιστικά συστήματα, όσο και στα μηχανικά μέρη των ρομποτικών συστημάτων, επέτρεψαν την ανάπτυξη πιο σύνθετων συστημάτων, επιτρέποντας τη διερεύνηση και νέων προκλήσεων. Μια τέτοια πρόκληση είναι η δημιουργία ανθρωποειδών ρομπότ, με τη δυνατότητα να αντιμετωπίζουν περίπλοκα περιβάλλοντα, όπως αυτά στα οποία κινείται ο άνθρωπος. Σημαντικές μελέτες επικεντρώνονται στην κίνησή τους, στον έλεγχό της, όπως και στον τρόπο που αλληλεπιδρούν με ανθρώπινα εργαλεία, αλλά και με τον ίδιο τον άνθρωπο. Η αυξανόμενη πολυπλοκότητα στο σχεδιασμό του σώματός τους, καθιστά αναγκαία την χρήση ακόμη πιο εξελιγμένων μεθόδων ελέγχου της κίνησης από τις ήδη υπάρχουσες. Στην αναζήτηση τέτοιων μεθόδων, οι επιστήμονες της ρομποτικής στράφηκαν προς την μελέτη έμβιων οργανισμών, οι οποίοι μπορούν να επιδείξουν αξιοσημείωτη ευρωστία και προσαρμοστικότητα στην κίνησή τους. Ένα βασικό συστατικό του ελέγχου αυτής της κίνησης είναι τα νευρωνικά δίκτυα, που ονομάζονται Κεντρικές Γεννήτριες Προτύπων (Central Pattern Generators – CPGs), όπως αυτά που βρίσκονται στην σπονδυλική στήλη των σπονδυλωτών οργανισμών, και τα οποία είναι σε μεγάλο βαθμό υπεύθυνα για την παραγωγή των σημάτων ελέγχου κατά την διάρκεια ρυθμικών κινήσεων των οργανισμών. Η αναπαραγωγή αντίστοιχων μηχανισμών ελέγχου κίνησης σε ανθρωποειδή ρομπότ συνιστά μία σημαντική πρόκληση. Ένα από τα βασικά χαρακτηριστικά αυτών των δικτύων είναι ότι μπορούν να αναπαράγουν εύρωστες ρυθμικές συμπεριφορές, ακόμη και χωρίς την παρουσία αισθητηριακής πληροφορίας.

Σε αυτή την διπλωματική εργασία, έγινε μελέτη με ρομποτικά πειράματα στο ρομπότ NAO της εταιρίας Aldebaran Robotics, πάνω στην ευστάθεια του περπατήματος που έχει υλοποιηθεί από την κατασκευάστρια εταιρία, σε περιβάλλοντα με σκαλιά, επικλινείς επιφάνειες, καθώς και επιφάνειες με εμπόδια και άμμο. Έπειτα, γίνεται απόπειρα, με τη χρήση κεντρικών γεννητριών προτύπων, να επιτευχθεί η αναπαραγωγή σταθερού περπατήματος στον εξομοιωτή Webots, για το ανθρωποειδές ρομπότ NAO. Το είδος των CPGs, που επιλέχθηκε να χρησιμοποιηθεί, έχει σαν βάση του τον μη-γραμμικό ταλαντωτή Hopf, ο οποίος έχει τις δυναμικές ιδιότητες που συσχετίζονται με τα σήματα των αρθρώσεων του ανθρωποειδούς που παράγονται κατά το περπάτημα, και μπορεί να τα αναπαράγει σε ένα ικανοποιητικό βαθμό. Η διαδικασία που ακολουθήθηκε ξεκινάει από την καταγραφή των τροχιών των αρθρώσεων με κατάλληλους αισθητήρες, την εκμάθηση των σημάτων αυτών μέσω του CPG, και, στη συνέχεια, του ελέγχου των αρθρώσεων του ρομπότ μέσω των CPGs, με σκοπό την αναπαραγωγή του περπατήματος. Η παραπέρα βελτιστοποίηση των CPGs έγινε μέσω κατάλληλης εξελικτικής διαδικασίας, με τη χρήση γενετικού αλγορίθμου. Για την επίτευξη της διαδικασίας αυτής έγινε αναλυτικός ορισμός του προβλήματος, με την διατύπωση αντικειμενικής συνάρτησης, που εξελίσσει τα CPGs, με σκοπό τη βελτίωση της απόκρισης του συστήματος ελέγχου της κίνησης του ανθρωποειδούς ρομπότ κατά το περπάτημα.

Acknowledgements

I would like to thank my advisor Dr. Dimitris P. Tsakiris, of the Institute of Computer Science (FORTH), first for the opportunity he gave me to join his laboratory two years ago for a summer internship and, then, for accepting to guide me through completing my studies by co-supervising my diploma thesis. It was inspiring to watch the work of his team and himself in the laboratory. I am grateful for that.

Moreover, I would like to thank Professor George Stavrakakis, my thesis co-supervisor from TUC, and Associate Professor Michail Lagoudakis of the TUC for the interest they showed in my work, and for their participation in my thesis committee.

I would also like to specially thank Dr. Manolis Hourdakis for all the interesting ideas that he shared with me, the patience he has shown and for the midnight and weekend messages he replied to in order to help me.

Furthermore, I would like to thank Thodoris Evdemon and Augousta Chatzidaki for those two wonderful years we shared in the lab. Thodori - your computer is free from now on, you can shut it down, and Augousta – I am sorry for all my pranks but I know you will miss me.

After that, I must specially thank Evina and all my friends from Chania for the wonderful years we had there.

I need to thank particularly my brother Manolis for the patience that he has shown me while we were living together. It was the best thing that has happened to me and I know he was always the one who believed in me the most.

Last but not least, I would like to thank my parents and my brother Yiannis - here no words are sufficient.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

During the last decade, one of the major research directions in robotics has been to develop robots that can assimilate seamlessly within human environments. A significant development in this direction is the transition from wheel-based robots to ones with anthropomorphic bodies [7], due to their ability to deal with non-planar and rugged environments, especially environments where humans live and work [8], [9].

In contrast to wheel-based robots, in bipedal locomotion the stability of the robot can be easily disturbed. The robot's feet are continuously impacting the ground in a unilateral and under-actuated way [10], therefore, even small disturbances may cause problems to the postural stability of the robot. To produce a stable gait, a controller must be able to respond accurately to these disturbances, in a timely manner, taking into account sensory information.

Moreover, anthropomorphic robots encompass a large number of degrees of freedom (DOFs) and are characterized by complex dynamics and non-linearities, which may be difficult to model accurately and control [11]. To reduce the complexity of the control problem, researchers in robotics have sought inspiration from other fields, including biology and neuroscience. One such idea towards reducing the complexity of controlling a large number of DOFs is to replicate the function of Central Pattern Generators (CPG), namely of neuronal circuits that reside in the spinal cord of various vertebrates, and have been shown to actively participate in the generation of rhythmic locomotion patterns. From a computational perspective, the CPGs can encode complex control signals in a distributed manner, have well-described dynamics and response properties that resemble, to a large extent, to the properties of animal locomotion. Therefore, they are an efficient tool, which can be used to study the issues that arise from locomotion characteristics like the unpredictability of the unstructured environments or the delays inherent in robotic sensors.

1.1 Thesis Contribution

In this thesis, we examine how a stable gait can be reproduced on the simulated NAO V4 humanoid robot, using biologically-inspired CPG neural networks. Models of the CPGs have well described dynamics, exhibiting oscillatory behavior. This oscillatory behavior is very similar to the gait patterns exhibited by several animals [8], and is exploited here to specify the desired joint trajectories which achieve walking on the NAO humanoid robot. In addition, we use an optimization methodology, based on Genetic Algorithms (GA), to optimize the CPG network's performance with respect to the properties of its dynamic interaction with the environment. In order to document the need for improved walking behavior of the NAO, we test experimentally the limits of its built-in walking behavior in environments with steps, inclines, sand and rugged substrates.

1.2 Thesis Outline

The structure of this thesis is as follows: In Chapter 2, information about humanoid robots, the Aldebaran NAO v4 robot and experiments implemented on the real NAO V4 robot, in order to test the robustness of its built-in walking behavior in non-planar environments, is presented. In Chapter 3, the essentials of Central Pattern Generators (CPG) and of Genetic Algorithms (GA) is reviewed. In Chapter 4, the design, learning process and optimization via GAs of a CPG network is presented. Chapter 5 shows results from the simulation studies performed. Chapter 6 presents conclusions and some ideas for future work.

Chapter 2

The Humanoid Robot NAO and its Walking Behavior

2.1 Humanoid Robots and the Aldebaran Robotics NAO robot

As robots gradually take the role of human assistants, they must be able to confront the complexities of environments where humans live. One way to make that happen is to create human-like robots, called humanoids. The humanoids are designed in such a way that they resemble the morphology of the human body (Figure 2.1.1), and posses behaviors like grasping with hands [12] and bipedal locomotion [13]. Every robot has a specific number of DOFs, depending on the features and complexity of the tasks one wishes it to perform [1]. However, higher DOFs lead to an increase in the complexity of the control problem, and this is not always beneficial or desired.

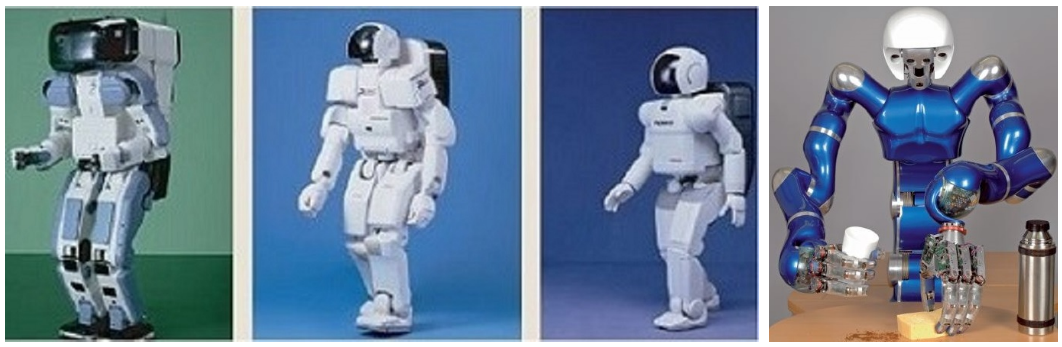


FIGURE 2.1.1: The left three figures show the Honda P2, P3 and Asimo humanoid robots. The right figure shows the robot Justin. [1]

After the middle of the 20th century, researcher groups started to give their systems multiple abilities for sensing, control and actuation. It was at this time that roboticists tried to developed behaviors inspired by human abilities. They started to develop systems with sensors, motion and manipulators. In 1973, the group of Ichiro Kato at Waseda University in Japan, created the first humanoid robot which integrated all those tasks, called WABOT-1 (Figure 2.1.2). After that, many companies and researchers started working in the field of humanoid robots. In 1996, the Honda company built the Honda humanoid P2 (Figure 2.1.1), which was the first autonomous humanoid capable of stable bipedal walking with on-board power and processing. Since that time, many bipedal humanoid robots came about, such as the P3 and ASIMO (Figure 2.1.1), the Justin robot which is an autonomous two-armed humanoid robot with 7-DOF-torque-controlled arms and 12-DOF hands (Figure 2.1.1), the Atlas robot of DARPA's projects (Figure 2.1.2) which is a 2 m humanoid robot designed for a variety of search and rescue tasks, and the industrial robot Baxter of Rethink Robotics (Figure 2.1.2), which is a two-armed robot with an animated face, that is used for simple industrial jobs such as loading, unloading, sorting and handling of materials. Some current challenges for humanoid robots include walking and running in rugged terrains, the whole body control problem, large disturbance handling, etc.

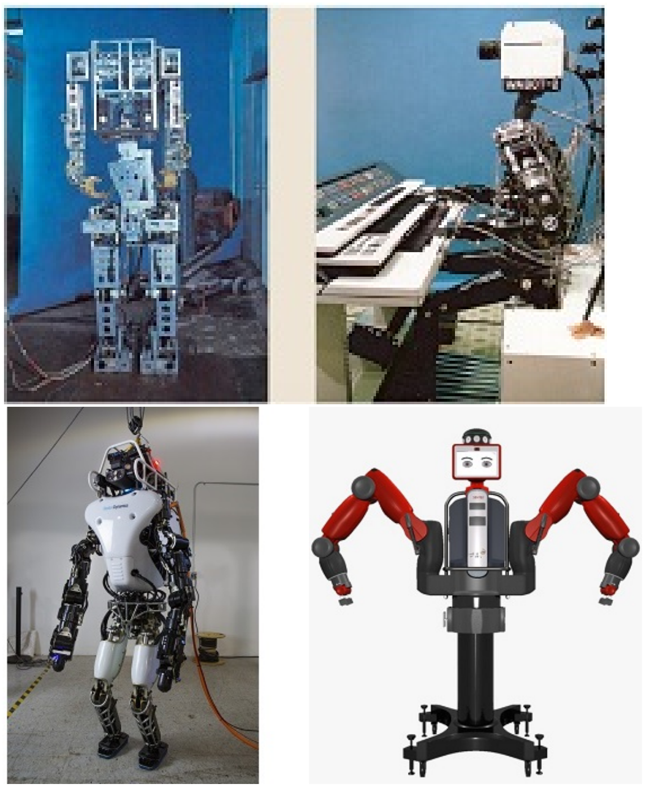


FIGURE 2.1.2: The top figures show the robots WABOT-1 (1973) and WABOT-2 (1984) [1]. The bottom left figure shows the robot Atlas of Boston Dynamics. The bottom right figure shows the robot Baxter of Rethink Robotics.

In 2006, the French company Aldebaran Robotics developed a humanoid robot called NAO. It is an autonomous, medium-size humanoid robot which has low cost and is easy to program [1]. Since its first public release in 2008, many new versions have been produced (V1,V2 ... and, recently, V5). This thesis focuses on the NAO version V4.

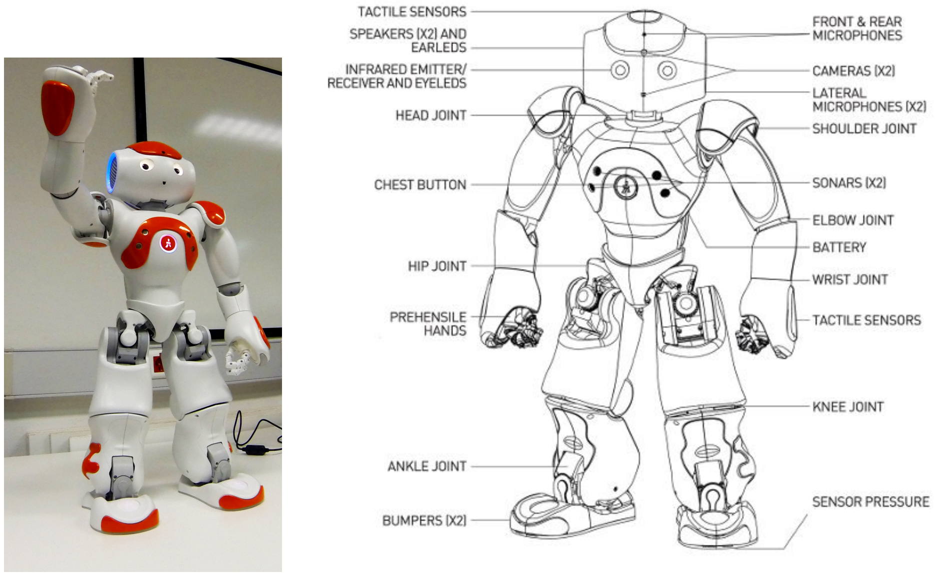


FIGURE 2.1.3: The left figure shows a NAO robot. The right figure shows a schematic of the NAO V4 sensor and joint placement. [2]

The NAO humanoid robot (Figure 2.1.3) has 25 degrees of freedom (DOFs), with 2 DOFs in the head, 6 DOFs in each arm, 5 DOFs in each leg and 1 DOF at the pelvis (it has a pair of pelvis joints, but they cannot work separately from each other). The configuration and names of these DOFs are shown in Figure 2.1.4. The NAO V4 is a robot of 57.3 cm in height, 27.5 cm in width and 31.1 cm in depth humanoid robot, weighting 5.18 kg. It is also equipped with a variety of sensors. It has four Force Sensitive Resistors (FSRs) in each sole (Figure 2.1.5), which provide force information with a delay of about 50 ms. The mean estimate of the FSR sensors gives the Center of Pressure (COP) in each sole. The FSRs evaluate the pressure applied to each foot (Table 2.1), and their working range is from 0 N to 25 N. Also, there is a 2-axis gyroscope and a 3-axis accelerometer. This inertial measurement unit is located in the torso of the robot, embedded with its own processor.

Embedded in the robot are an Intel ATOM z530@1.6 Ghz CPU, with 1 GB RAM and 2 GB Flash memory, with built-in Linux operating system. The robot is electrically powered by a lithium-ion battery located at its back and capable of keeping the NAO operational for about 60 minutes in active use. The NAO is capable of both wireless and wired communication with remote computer using the IEEE 802.11 protocol. It can also be connected with Kinect or Asus 3D sensor, or with an Arduino device via its

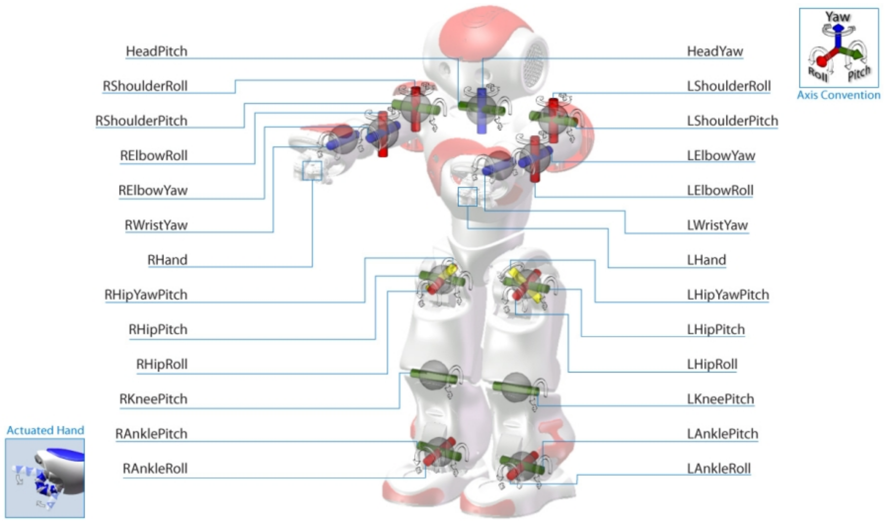


FIGURE 2.1.4: NAO V4 joints in the three axis convention and their names. [2].

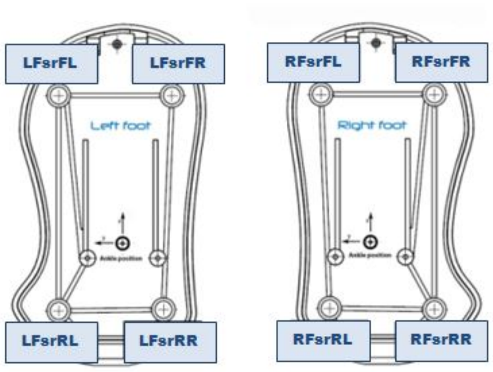


FIGURE 2.1.5: The left and right sole FSRs.

USB port. The NAO also has a variety of interaction devices and sensors: the head has 2 loudspeakers as left and right ears, 2 microphones in the front upper head and 2 in the back upper head, 2 identical video cameras located in the forehead and in the chin, which provide video with up to 1280x960 pixel resolution at 30 frames per second, 2 infrared (I/R) sensors, one in each eye, with wavelength 940 nm and power consumption 8mW/sr, and some LEDs in the upper head and in the eyes. Other sensors are a sonar with 2 emitters and 2 receivers each one in the left and right side of the chest. The robot has 3 tactile capacitive sensors, 1 chest button and 2 feet bumpers in each feet; it also has 36 Magnetic Rotary Encoders (MRE) at each joint using Hall-effect sensor technology, to provide information on each joint angle [2].

FSR Name	Position X(m) [Ankle Frame]	Position Y(m) [Ankle Frame]
LFsrFL	0.07025[LEFT]	0.0299[LEFT]
LFsrFR	0.07025[LEFT]	-0.0231[LEFT]
LFsrRL	-0.03025[LEFT]	0.0299[LEFT]
LFsrRR	-0.02965[LEFT]	-0.0191[LEFT]
RFsrFL	0.07025[RIGHT]	0.0231[RIGHT]
RFsrFR	0.07025[RIGHT]	-0.0299[RIGHT]
RFsrRL	-0.03025[RIGHT]	0.0191[RIGHT]
RFsrRR	-0.02965[RIGHT]	-0.0299[RIGHT]

Table 2.1: Force Sensitive Resistor (FSR) positions in the Ankle Frame. [2]

2.2 Robotic Experiments regarding the Stability of the Built-in NAO Walking Behavior

This was a first attempt to acquire knowledge from the bipedal NAO robot. So it is considered as preliminary work, aiming to test the built-in functionalities of the robot. The Aldebaran Robotics company have implemented a NAOqi function, which commands the robot NAO to walk, designed primarily for walk on flat terrain. This section is an evaluation of this gait behavior on non-planar terrains, and examines the ability of the robot to adjust its gait and balance despite the terrain anomalies. A series of tests were performed in different terrains, like sand, grass, inclines and rough terrain. The NAO succeeded to descend a 1 cm-high step, many other tests were performed, but this was the test with the most successful results. When the steps were lower, the NAO descended them very easily, but as the height of the steps was increased, the NAO started to fall (Figure 2.2.1).

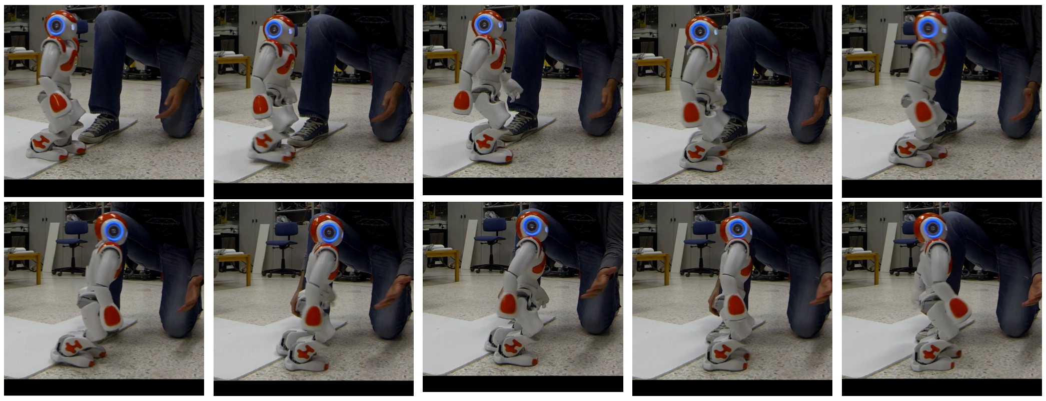


Figure 2.2.1: NAO descending from a 1 cm-high step.

The NAO succeed to walk on 4-degree (Figure 2.2.2) and 6-degree (Figure 2.2.3) inclines. In the first, the NAO had no problem, and all tests were successful. In the higher slope, the NAO started having difficulties, and had a 50% success ratio.

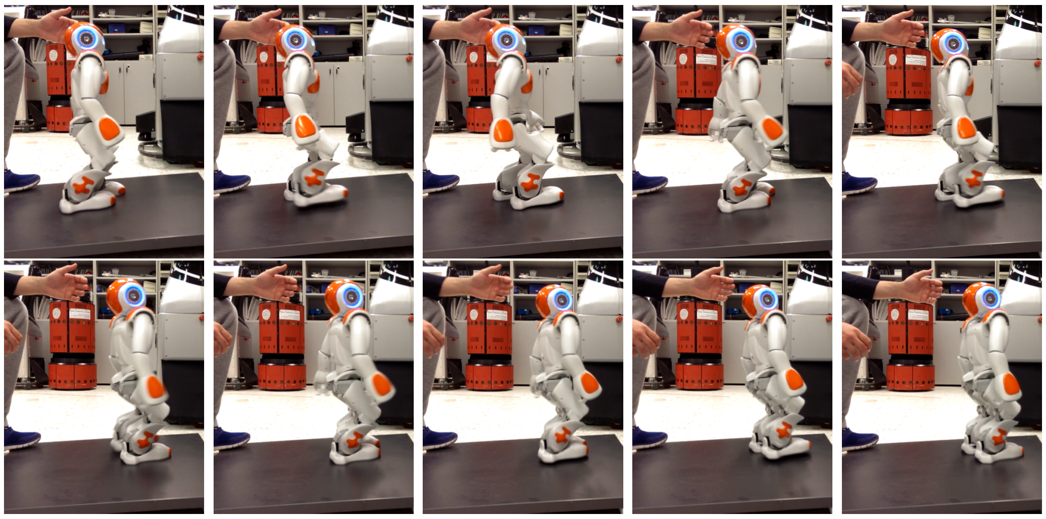


FIGURE 2.2.2: The NAO walking on a 4-degree incline.

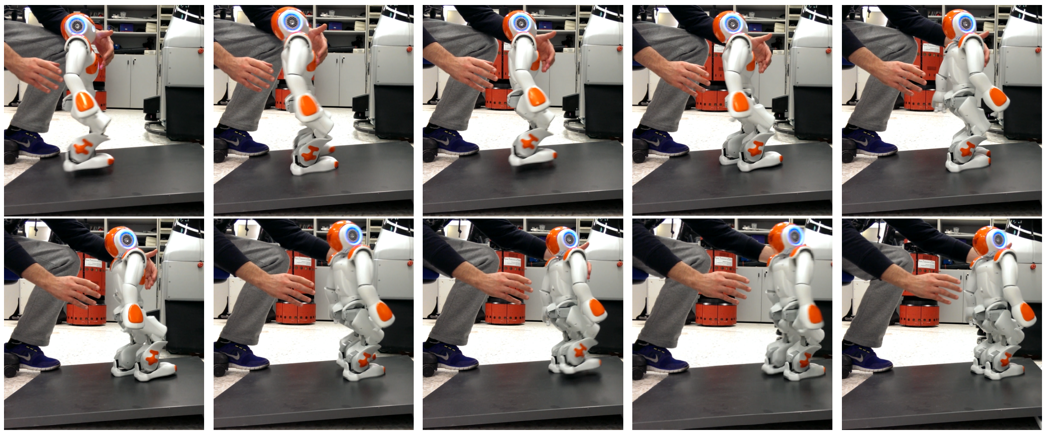


FIGURE 2.2.3: The NAO walking on a 6-degree incline.

In a planar terrain full of thin wires, the NAO had no problem in most of the tests (Figure 2.2.4). It was observed however that, when the wires became thicker, the NAO started to fall down even from the beginning of the attempt.

In compact sand, the NAO succeeded to walk quite well (Figure 2.2.6). Also, when the sand was softer, even if the NAO was not as steady as before, it still succeeded to walk relatively well (Figure 2.2.7). However, in sandy terrain, the NAO fell when it encountered holes in the sand.

In outdoor environments, as in an inclined driveway with bricks, the NAO managed to walk successfully in almost all tests (Figure 2.2.8). In grass, the robot found it difficult to make progress and fell most of the times, unless the grass was very dry and hard (Figure 2.2.9).

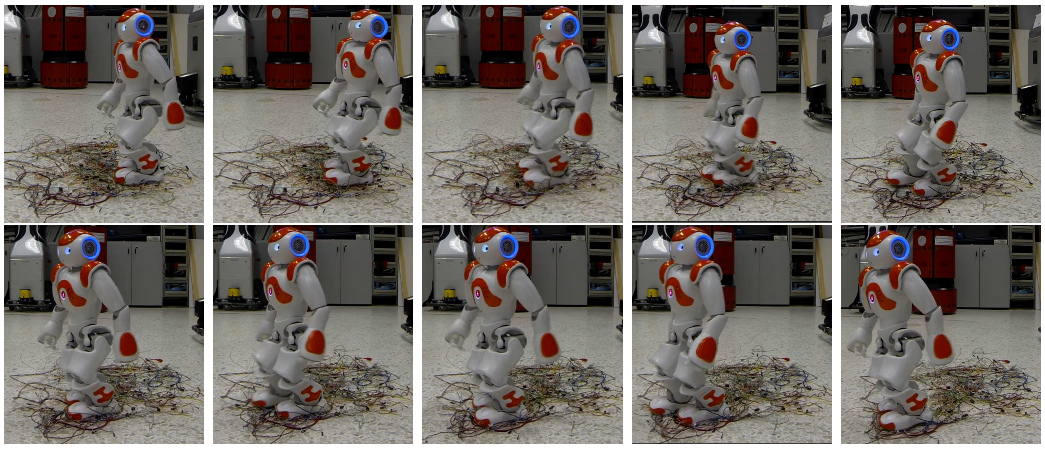


FIGURE 2.2.4: Planar terrain full of wires.

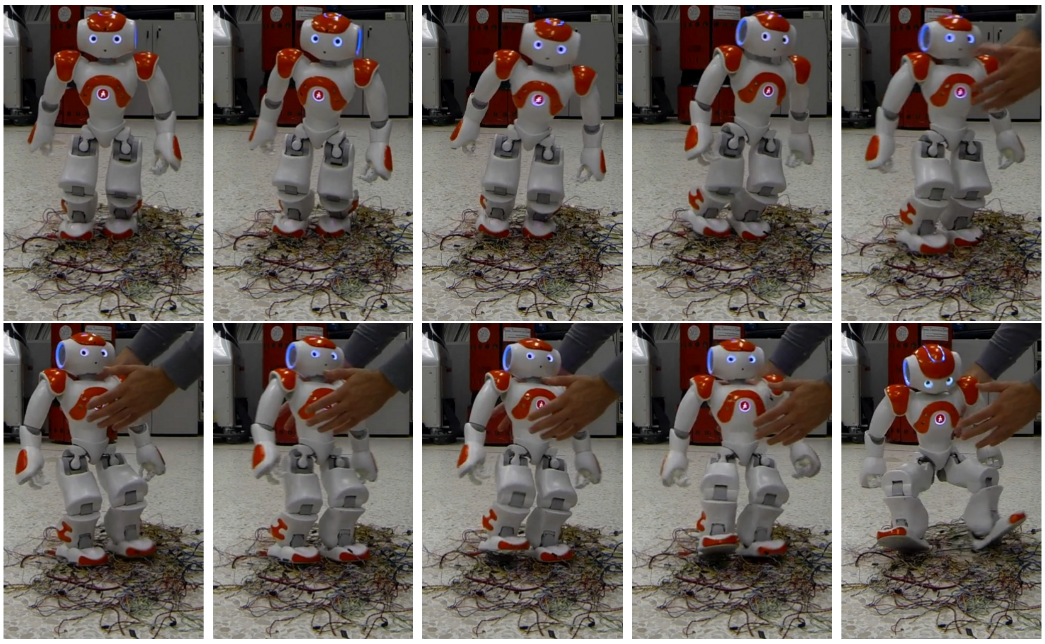


FIGURE 2.2.5: Failed attempt on planar terrain of wires.

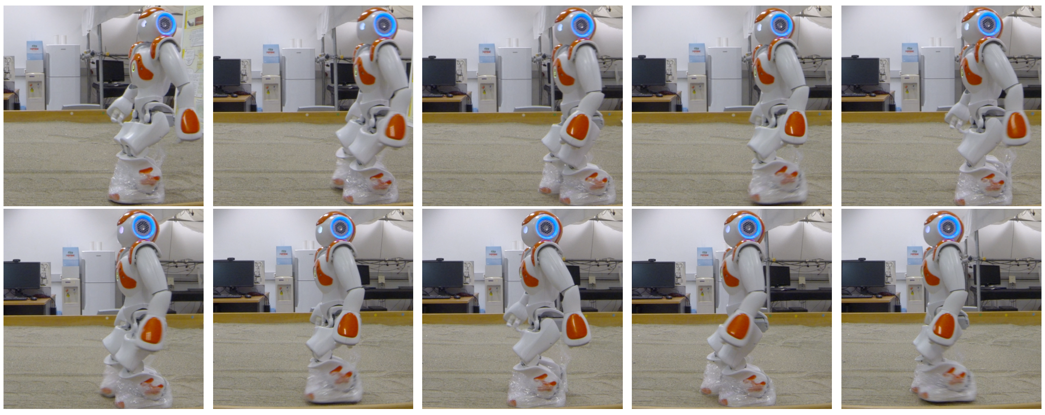


FIGURE 2.2.6: Compacted sand terrain.

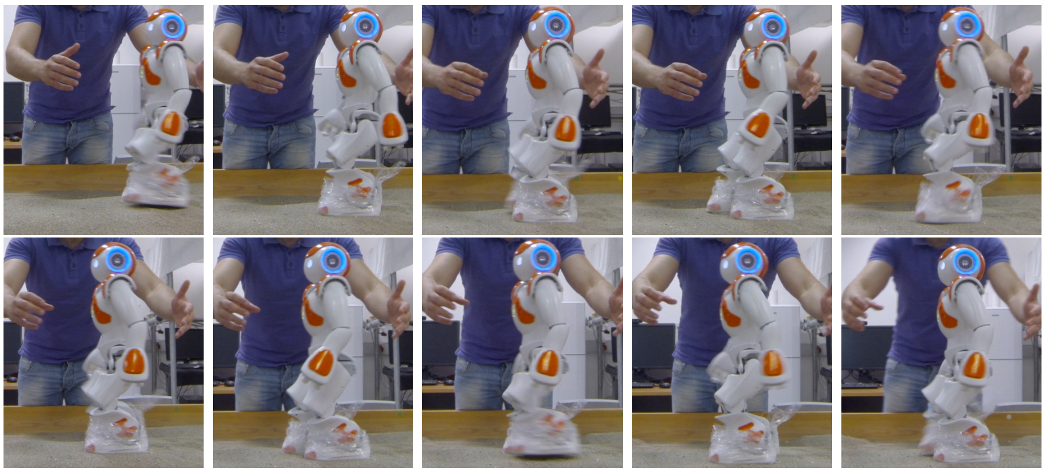


FIGURE 2.2.7: Soft sand terrain.

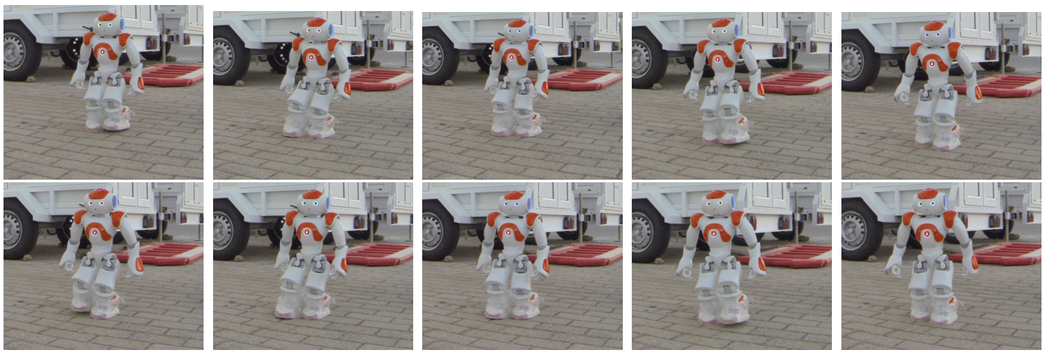


FIGURE 2.2.8: Rough slope terrain outdoor.

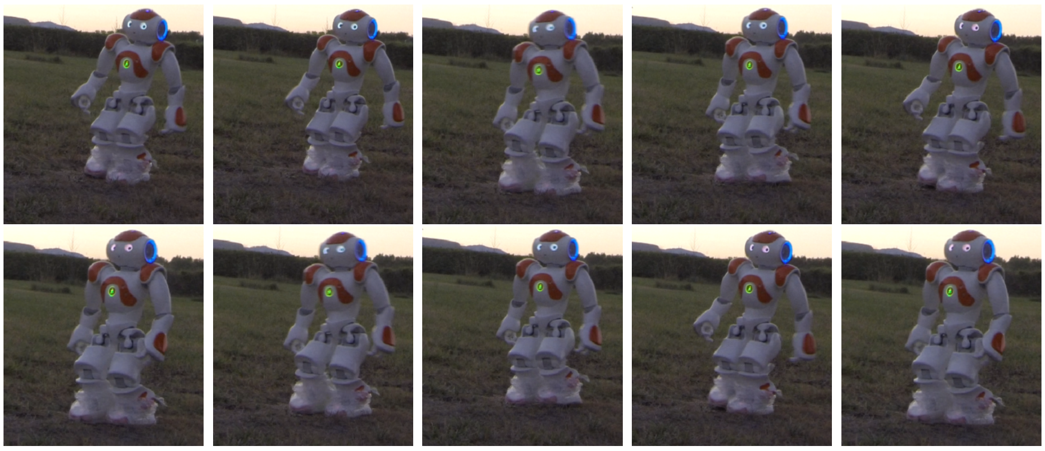


FIGURE 2.2.9: Grass terrain.

Chapter 3

Background

3.1 Central Pattern Generators (CPGs) and Nonlinear Oscillators

3.1.1 Central Pattern Generators

Central Pattern Generators or CPGs are neuronal circuits found in both vertebrate and invertebrate animals [14], [15], which can produce rhythmic motor patterns, even in the absence of sensory feedback. Some of the most common such motor patterns are walking, breathing, flying and digesting. Some of their intrinsic properties, which make them appealing for engineering applications, are: distributed control, the ability to deal with redundancy and disturbance, fast control loops, and allowing modulation of locomotion by simple control signals. Also, because the CPGs have the ability to adjust their behavior to account for variations of the environment, based on sensory information, they may be used to produce robust locomotion control in robots.

Each type of locomotion has its own properties, which can differ according to the species and their environment. This has led researchers to explore different types of CPGs, in order to be used in different types of problems ([16], [17], [18], [8], [19], [20], [21], [22], [14], [15], [23]). There are some properties that must be noted for the CPGs. Firstly, the limit cycle behavior, which provides to the system an embedded mechanism for auto-recovery from environment perturbations. In addition, because there are only few control parameters, they can easily be used in order to produce different desired behaviors in the implemented robot, such as forward locomotion, backward gait or speed control.

CPGs are distributed networks and, as a result, they can be used in modular robots [24], snake robots ([16], [17], [18]) or reconfigurable robots [25]. CPG models are able to create smooth transitions to the produced trajectories. Thus, if a perturbation causes a big change in the trajectory of the system, the system will be made to return to its periodic behavior by the CPG. Also, sensory feedback can be integrated to the model, so this can improve the locomotion of the system. Most of the CPG models have the potential to be integrated in learning and optimization algorithms, in order to adapt to different environments or system conditions. In hexapod and octopod robots, CPG models that have been used were inspired by insect locomotion [22]. CPG models inspired from the lamprey anguilliform swimming circuit are mostly used for controlling swimming robots. They reproduce the undulation of the elongated body from head to tail [15]. Researchers have discovered that, integrating sensory feedback in the locomotion control, in cooperation with the CPG model, can lead to a more robust gait. This approach leads to more stable locomotion in non-planar terrain. The sensory feedback that is selected for reinforcement of the CPG model is a criterion for the robustness of the robot [23].

CPG models can also be used for the control of biped locomotion in humanoid robots [4], as an alternative way for controlling the locomotion of robots, which previously was based exclusively on finite-state machines, sine-generators, prerecorded reference trajectories (e.g. for ZMP-based control [11]) or heuristic control laws (e.g. Virtual Model Control [26]).

3.1.2 Nonlinear Oscillators

The presentation of the mathematical background on CPG models employed in this work may start by considering the simple planar linear system :

$$\dot{x} = -y \tag{3.1}$$

$$\dot{y} = x \tag{3.2}$$

which corresponds to the harmonic oscillator. Representing the system in polar coordinates (r, θ), where $x = r\cos\theta$, $y = r\sin\theta$, we get :

$$\dot{r} = 0 \tag{3.3}$$

$$\dot{\theta} = 1 \tag{3.4}$$

Thus, the trajectories of this system form closed circular orbits, but these orbits are not isolated (Figure 3.1.1). If a perturbation occurs, while the system moves on a specific closed orbit, it will be deflected from its current orbit and will move along another orbit, where it will remain until another perturbation occurs. This linear system does not have the structural stability property, i.e. its qualitative properties are not maintained in the presence of perturbations. Therefore, stable oscillations can not be produced by this linear system.

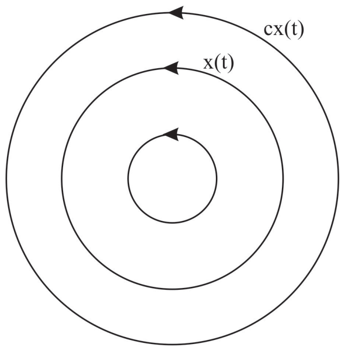


FIGURE 3.1.1: Phase portrait of the harmonic oscillator. [3]

Nonlinear systems may display oscillatory behavior of fixed amplitude and frequency, irrespective of their initial state and despite the occurrence of perturbation. This type of oscillation is known as a limit cycle, and corresponds to an isolated closed trajectory of the nonlinear system, which is called, the nonlinear oscillator. If all neighboring trajectories approach the limit cycles, we say that the limit cycle is stable or attracting (Figure 3.1.3). Stable limit cycles can be used to model systems that exhibit sustainable oscillations, i.e. which have the property to oscillate even in the absence of external forces, like the CPGs. A characteristic example is the beating of the heart, which can be described by an oscillation with a specific waveform and amplitude. If an external perturbation occurs, the system may momentarily move away from its limit cycle, but it will return eventually to it.

Consider first the planar non linear system :

$$\dot{x} = -y + x(1 - x^2 - y^2) \tag{3.5}$$

$$\dot{y} = x + y(1 - x^2 - y^2) \tag{3.6}$$

derived from the harmonic oscillator, by the addition of the nonlinear terms on the right. Representing this system in polar coordinates, we get the equations:

$$\dot{r} = r(1 - r^2), \qquad r >= 0 \tag{3.7}$$

$$\dot{\theta} = 1 \tag{3.8}$$

Here, the radial and angular dynamics of the system are uncoupled. This provides the possibility to analyze them separately. The motion in the θ-direction is simply a rotation with constant angular velocity. From the radial dynamics, we see that $r* = 0$ is an unstable fixed point and $r* = 1$ is a stable one (Figure 3.1.2). In the phase plane, for any trajectory other than $r* = 0$, we can see that all trajectories of the system spiral asymptotically towards the limit cycle at $r = 1$ (Figure 3.1.3). Perturbations of the system, while it evolves on the limit cycle, will return the system to the limit cycle. Therefore, the system exhibits stable oscillatory behavior, and is an example of a non linear oscillator.

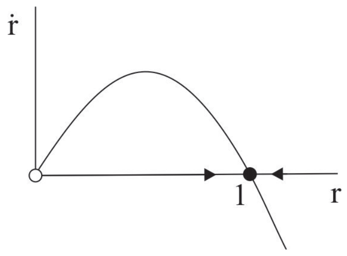


FIGURE 3.1.2: Radial dynamics of the non linear system. [3]

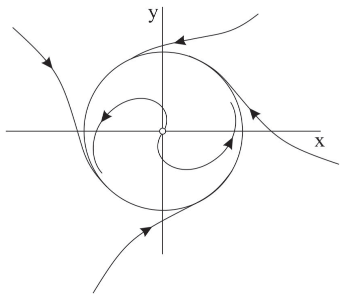


FIGURE 3.1.3: Phase portrait of the non linear system. Trajectories converge to the limit cycle. [3]

A generalization of the above nonlinear oscillator is the Hopf oscillator defined as follows:

$$\dot{x} = -\omega y + \gamma(\mu - (x^2 + y^2))x \tag{3.9}$$

$$\dot{y} = \omega x + \gamma(\mu - (x^2 + y^2))y \tag{3.10}$$

where x and y are the state variables, γ controls the speed of recovery after a perturbation, ω controls the intrinsic frequency and μ determines the steady state amplitude of the oscillation. In polar coordinates, we get the equations :

$$\dot{r} = \gamma r(\mu - r^2), \qquad r \geq 0 \tag{3.11}$$

$$\dot{\theta} = \omega \tag{3.12}$$

For $\mu > 0$, the oscillator possesses a unique stable periodic solution ($r_* = \sqrt{\mu}$) namely a limit cycle, described by the following equations:

$$x(t) = \sqrt{\mu}\sin(\omega t + \theta_0) \tag{3.13}$$

$$y(t) = \sqrt{\mu}\cos(\omega t + \theta_0), \tag{3.14}$$

where θ_0 is an initial condition.

The Adaptive Hopf oscillator [19] is a generalization of the Hopf oscillator, to which a Hebbian-type learning rule is added, as described in the following equations:

$$\dot{x} = -\omega y + \gamma(\mu - (x^2 + y^2))x + \epsilon F(t) \tag{3.15}$$

$$\dot{y} = \omega x + \gamma(\mu - (x^2 + y^2))y \tag{3.16}$$

$$\dot{\omega} = -\epsilon F(t)\frac{y}{\sqrt{x^2 + y^2}} \tag{3.17}$$

The two state equations are modified to receive a periodic input signal $F(t)$, which adds a perturbation to the x-state dynamics. Also, the intrinsic frequency ω of the system appears, in the dynamics of equation 3.17, which is the tool for frequency adaptation: when the Hopf oscillator's intrinsic frequency is close to one frequency component of the periodic input F, the oscillations will synchronize to the frequency of the periodic input (entrainment). The learning rule (equation 3.17) ensures this synchronization. In the equations above, $r = \sqrt{(x^2 + y^2)}$, γ controls the speed of recovery after a perturbation, μ controls the amplitude of the oscillator, $F(t)$ is a periodic input to which the oscillator will adapt its frequency and $\epsilon > 0$ is a learning constant. The signal $F(t)$ may have more than one frequency component, but the oscillator has the ability to adapt to only one frequency (state ω), which will depend on the initialization of the frequency of the oscillator. In polar coordinates, this system takes the form :

$$\dot{r} = \gamma r(\mu - r^2) + \epsilon F\cos\theta \tag{3.18}$$

$$\dot{\theta} = \omega - \frac{\epsilon}{r}F\sin\theta \tag{3.19}$$

$$\dot{\omega} = -\epsilon F\sin\theta \tag{3.20}$$

3.1.3 Network of CPGs Controlling the Joints of a Humanoid Robot

We consider here a CPG model composed of a chain of N Adaptive Hopf oscillators presented in [4] and shown in Figure 3.1.4. Each oscillator learns one frequency component of a periodic input signal, in our case data from one joint encoder of the NAO humanoid robot during walking. Therefore, the whole CPG learns to reproduce the periodic behavior of this particular joint during walking. Several such CPGs could be used, one for each joint of a humanoid robot, to learn a particular walking behavior.

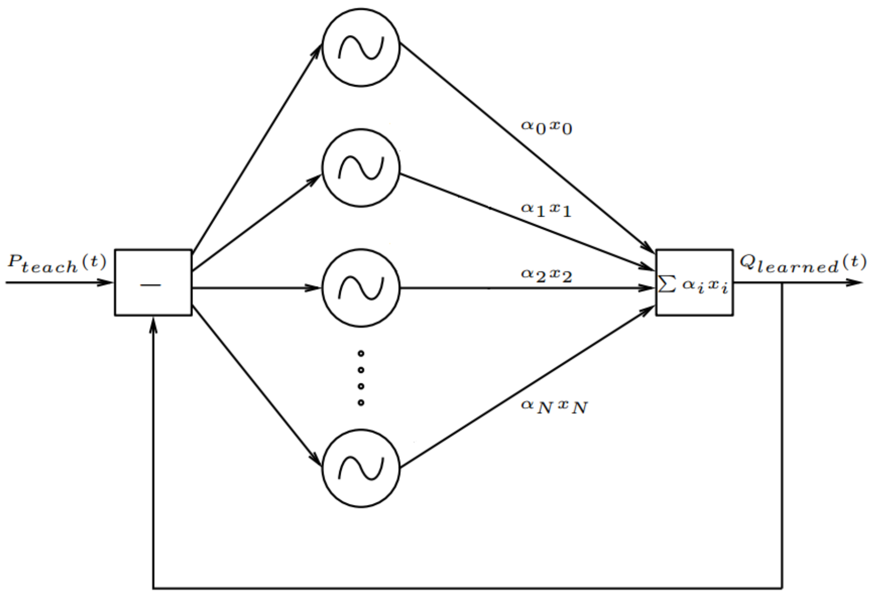


FIGURE 3.1.4: Structure of a CPG composed of a chain of N Adaptive Hopf oscillators and associated to one joint of the humanoid robot. [4]

The equations describing each oscillator of the CPG are $(i = 1, \ldots, N)$:

$$\dot{x}_i = -\omega_i y_i + \gamma(\mu - r_i^2)x_i + \epsilon F(t) + \tau \sin(\theta_i - \phi_i) \tag{3.21}$$

$$\dot{y}_i = \omega_i x_i + \gamma(\mu - r_i^2)y_i \tag{3.22}$$

$$\dot{\omega}_i = -\epsilon F(t)\frac{y_i}{r_i} \tag{3.23}$$

$$\dot{\alpha}_i = \eta x_i F(t) \tag{3.24}$$

$$\dot{\phi}_i = sin(\frac{\omega_i}{\omega_0}\theta_0 - \theta_i - \phi_i) \tag{3.25}$$

When the system has converged, the states x_i and y_i evolve on the limit cycle of the system $r_i = \sqrt{x_i^2 + y_i^2}$. After convergence, each oscillator has converged to one of the dominant frequencies contained in the input signal. Each oscillator converges to one frequency and the total output Qlearned of the CPG is the weighted sum of the output x_i of its oscillators, weighted by the variables α_i: $Qlearned = \sum_{i=1}^{N} \alpha_i x_i$. Also, there is knowledge for the phase difference between the oscillators, and this gives the possibility of reproducing any phase relationship between them. The parameters $\epsilon > 0$ and τ are coupling constants, where ϵ affects the frequency of the i_{th} oscillator ω_i. The variable $Pteach$ is the input signal that the CPG has to adapt to and $Qlearned$ it is the CPG's output. The difference $F(t) = Pteach–Qlearned$, is the remaining periodic input signal, to which the CPG has to adapt its frequency components. Each oscillator is coupled to the first oscillator in the chain. The parameter τ determines the strength of this connection between the 1st oscillator and the ith oscillator, in order to keep correct phase relations between oscillators, using the Kuramoto coupling scheme for achieving phase synchronization [27]. The parameter η is the coupling constant, which affects α_i and, as a result, the time of the full adaptation of the teaching signal. The parameter α_i is the amplitude associated with the frequency of the ith oscillator. In the learning process, the amplitude of each oscillator will be maximized only if the frequency components ω_i converge to a frequency component of $F(t)$. The parameter γ controls the speed of the system's recovery after a perturbation occurs. The parameter μ controls the amplitude of each oscillator. The parameter θ_i is the instantaneous phase of the i oscillator. The parameter ϕ_i is the phase difference between oscillators i and 1. It should be mentioned that the oscillators can acquire any arbitrary phase relationship, as in our case.

This CPG model of coupled nonlinear oscillators has the ability to learn a large class of periodic signals provided. Once the teaching signal is removed from the system, when the learning process is completed, its trajectories stay in the vicinity of the limit cycle of the dynamical system. A humanoid controlled by a network of such CPGs (Figure 3.1.5) could learn a prerecorded walking behavior and execute it in a relatively robust manner.

After the learning process ends, the CPG has fully adapted to the $Pteach$ learning signal. The CPG is now capable of reproducing the adapted signal, having all its dynamic properties.

The equations describing each oscillator of CPG, after the leaning process ($F = 0$), are ($i = 1, \dots, N$):

$$\dot{x}_i = -\omega_i y_i + \gamma(\mu - r_i^2)x_i + \tau \sin(\theta_i - \phi_i) \tag{3.26}$$

$$\dot{y}_i = \omega_i x_i + \gamma(\mu - r_i^2)y_i \tag{3.27}$$

$$\dot{\phi}_i = sin(\frac{\omega_i}{\omega_0}\theta_0 - \theta_i - \phi_i) \tag{3.28}$$

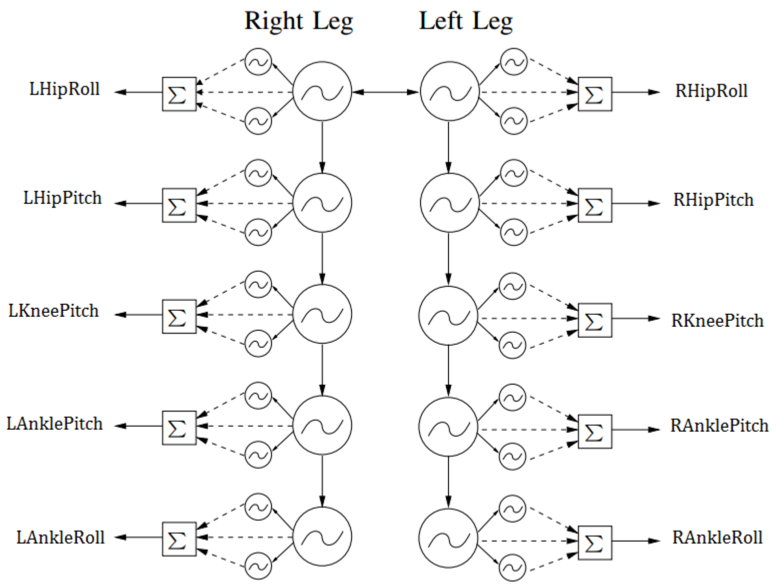


FIGURE 3.1.5: Structure of the network of CPGs for the NAO humanoid robot legs. [4]

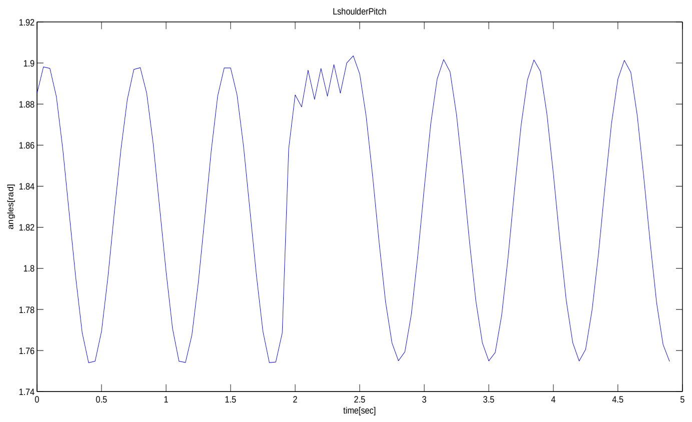


FIGURE 3.1.6: The signal is the output of the CPG, and the figure shows the CPG recovery after perturbation applied from time $t = 2$ sec until time $t = 2.5$ sec. x - axis time in seconds and y-axis x angles in radians.

The behavior of the CPG is shown in Figure 3.1.6 and 3.1.7, where we demonstrate the ability of its attractor to recover efficiently after a perturbation by simulating the dynamical system (3.21) - (3.25) in Matlab. As Figure 3.1.7 shows, the CPG corresponding to the LShoulderPitch joint, after learning, has a stable limit cycle. At time $t = 2sec$, a perturbation is applied, which causes the CPG to deviate from the limit cycle. However, as the figure demonstrates, as soon as this perturbation is removed, at $t = 2.5sec$, the oscillator trajectory is capable of approaching again rapidly the limit cycle. This property clearly becomes important when designing a robotic controller using a CPG neural network, in which case, the recovery of a predefined rhythmic behavior is important, in order to attenuate possible interference from the environment.

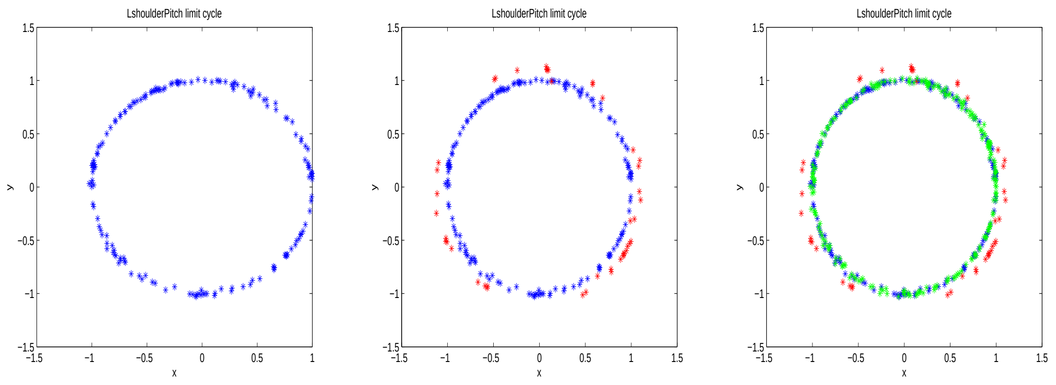


FIGURE 3.1.7: Phase portrait of the CPG response (the x-axis is the angle and the y-axis is the angular velocity of the joint): (Left) before the perturbation (CPG trajectory in blue for time = 0 - 2 sec), (Center) during the perturbation (CPG trajectory in red for time = 2 - 2.5 sec), (Right) after the perturbation is removed (CPG trajectory in green for time = 2.5 - 5 sec).

3.2 Genetic Algorithms

A Genetic algorithm (GA) is a stochastic optimization technique, able to solve optimization problems residing in both convex and non-convex manifolds [28], [29]. It is a search heuristic, that mimics the process of natural selection, and belongs to a larger class of evolutionary algorithms, which apply different forms of operations, in order to generate possible candidate solutions to a problem. The GA repeatedly changes a population of individual solutions, each one called a single chromosome. In every step of the evolution, the GA applies various operations on a given set of chromosomes, such as the mutation and crossover operations. In that way, the algorithm carries information from all of the evolution through the evolution of the generations. As the evolution goes through the population, the GA moves be closer to finding a good solution to the problem. This method can be used for solving a variety of optimization problems, in which the standard optimization methods are not well suited, including problems with discontinuous, non-differentiable, stochastic, or highly nonlinear objective functions. Due to their ability to traverse non-convex manifolds without falling into local minima, genetic algorithms have been widely used in the literature of robotics, in order to optimize parameters in various problems [30], [31], [32].

In its most typical form, a genetic algorithm generates a single point at each iteration, which evaluates given an objective function that describes the problem, and then reshuffles the chromosomes in search for solutions that minimize or maximize this objective function. The sequence of points approaches a good, possibly optimal solution. The GA, generates a number of solutions depending on the population at each iteration. Solutions with a higher score are selected by the algorithm for the next iterations.

There are various selection schemes, the most popular of which is the roulette wheel scheme, where chromosomes are ranked against their evaluation, and are selected by a rule. The GA uses three main types of rules at each step of its routine for creating the next generation from its current population. Those rules are inspired by the natural evolution like the inheritance, mutation, selection and crossover.

- The Selection rule chooses some individuals, called parents, from the current population, in order to create the population of the next generation.
- The Crossover rule combines two parents, in order to create the children for the next generation.
- The Mutation rule applies random changes to specific parents to create the children.

Besides these operators, there are various other ones available, that address different features of the parametric space of the optimization problem.

At the beginning of the process, a population of individuals is created randomly. As the process continues, the individuals are evaluated and a score is assigned to each one. The score is based on the evaluation of the optimization function that the programmer selected. After that, based on their score, two of the best individuals are selected. The individuals with higher fitness value have higher probability to be selected. Those individuals are combined to create one or more offspring, which are then mutated randomly. The procedure continues until a certain number of generations pass or the optimal solution is found, depending on the GA mechanism.

There is an extended nomenclature associated with genetic algorithms. In the following we provide the basic terminology, and a brief description of their meaning.

Fitness function: The fitness function is a function to be optimized. It may have certain constraints and certain variables that need to be minimized or maximized by GA. It is also known as the 'objective function'.

Individuals: Besides the selection of the objective function, candidate solutions, which arise from the evolution of the GA and are called individuals, are also vitally important. The solution of the objective function for every individual is its score. The individuals are referred to as 'genomes' or atoms.

Populations and Generations: The collection sum of the individuals creates a population. The individuals that appear in a population are not necessarily unique. While the populations evolve via new iterations of the GA, a series of computations are performed in order to produce a new population called the 'new generation'.

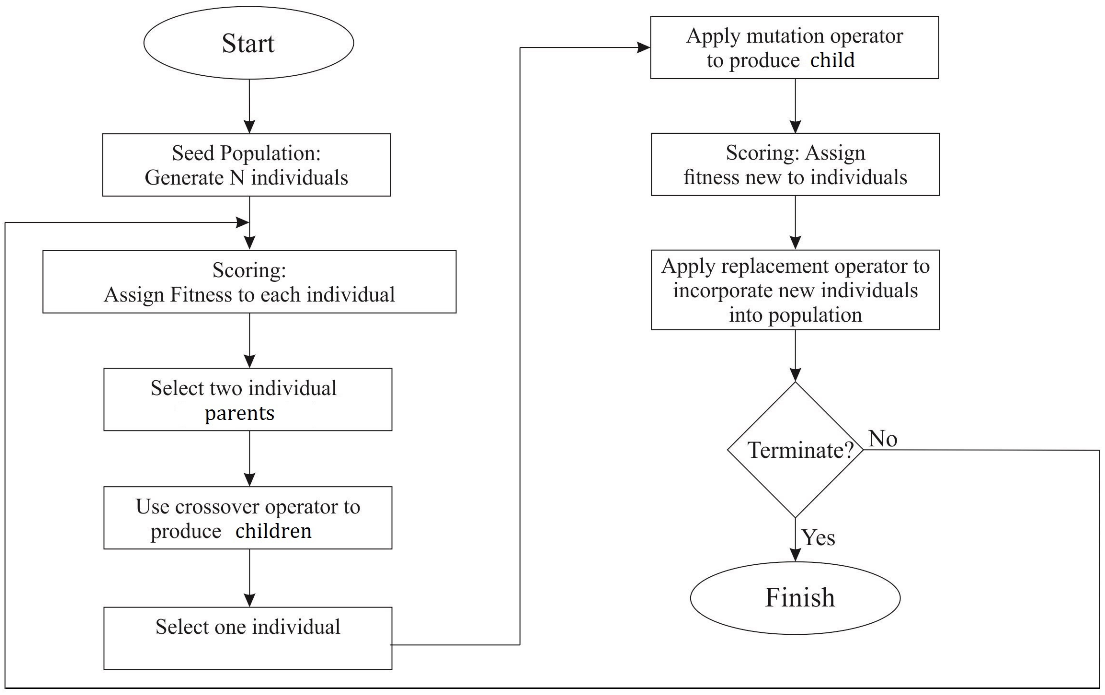

Start
Seed Population: Generate N individuals
Scoring: Assign Fitness to each individual
Select two individual parents
Use crossover operator to produce children
Select one individual
Apply mutation operator to produce child
Scoring: Assign fitness new to individuals
Apply replacement operator to incorporate new individuals into population
Terminate?
No
Yes
Finish

FIGURE 3.2.1: Genetic Algorithm flowchart [5].

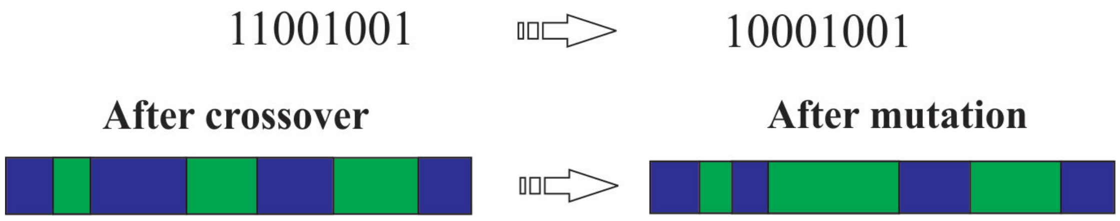


FIGURE 3.2.2: Mutation [6].

Encoding: Encoding is usually used in connection with the chromosomes. Types of encoding are binary, permutation value encoding or tree encoding.

Selection: The selection of the individuals in order to create offspring is crucial for the evolution of the algorithm. There are some types of selections that can be chosen, but it all depends on the problem, the most common of which is the proportional roulette. The main point is that, as the generations pass, the members of the population must be closer and closer to the solution of the problem, so the algorithm only keeps the individuals that best fit the solution. The selection mechanisms is shown in Figure 3.2.1.

Cross over: The step after selecting the propriety individuals is called 'crossover'. It is a genetic operator that mates two individuals called parents to create a new offspring, which has the best characteristics from its parents (Figure 3.2.2).

Mutation: In order to create more unique individuals, that have most of their characteristics from their parents, but also have something new, which will make them fit the solution of the problem better, some small changes are made to the chromosome called

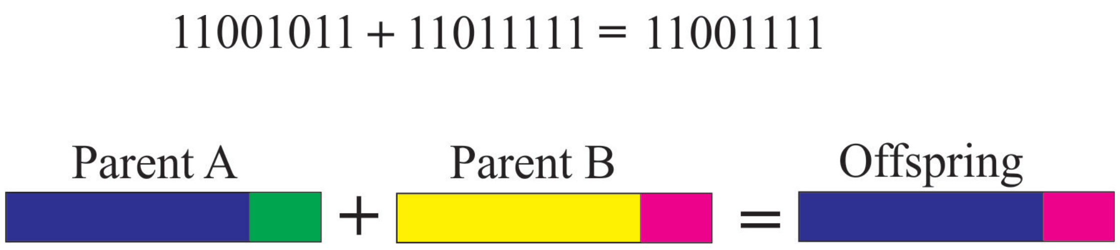


Figure 3.2.3: Single Point Crossover [6].

'mutations'. The mutation helps the algorithm to search over a larger range for the best solution, avoiding convergence to local minima (Figure 3.2.3).

Chapter 4

Implementation

In this chapter we outline the developments carried out, in order to design and optimize a walking gait controller for the simulated Nao humanoid. The interrelationship of the implemented modules is depicted in Figure 4.0.1, and can be summarized as follows: The first step consists of the analysis of the sensory information from the joint encoders of the robot, in order to derive the qualitative characteristics of the signals in module "Prerecorded Joint Trajectories" (Section 4.1). Results from this analysis were used to design the architecture and derive the learning parameters for the CPG associated with each joint, which is composed of a chain of coupled Adaptive Hopf oscillators in module "CPG Learning Process" (Section 4.2). To control the robot, a network of CPGs was designed, trained and evaluated in the Webots simulation environment, in order to derive quantitative metrics regarding its stability (Section 4.3.1). These metrics were employed by a genetic algorithm (module "GA"), which evaluated a population of different solutions, in an attempt to optimize the stability of the controller (Section 4.3.2). The various software packages employed throughout the Thesis are described in Section 4.4.

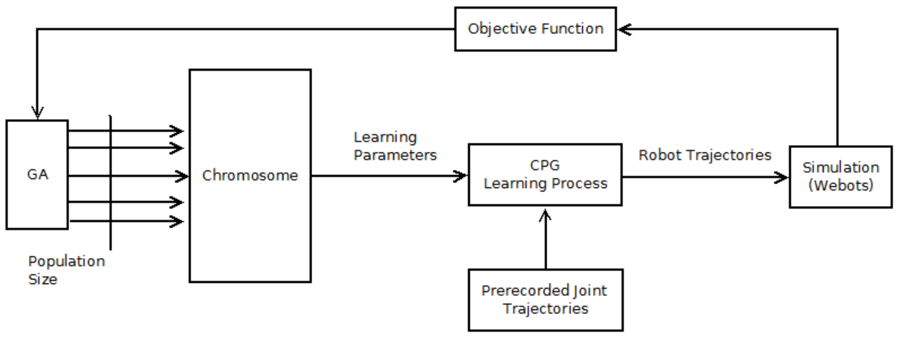


FIGURE 4.0.1: Components of the implemented system.

4.1 Recordings and Characteristics of Joint Trajectories

The robot which was used in our simulation studies is the NAO v4 H25 robot, a humanoid consisting of 25 DOFs. From a preliminary investigation of the trajectories of the NAO, produced by the built-in NAOqi walking gait function (Section D, Equation D.1) as a sequence of joint angles, we observed that a large number of joints had minimal involvement in this gait . To reduce the computational requirements from processing the joint trajectories, we have excluded these joints from the learning process, and locked them in a standard angle. The following analysis involves the eighteen remaining joints, which contribute significantly in the walking behavior. Those joints are (L:Left - R:Right) L-RShoulderPitch, L-RShoulderRoll, L-RElbowYaw, L-RElbowRoll, L-RHipRoll, L-RHipPitch, L-RKneePitch, L-RAnklePitch, L-RAnkleRoll.

For recording the joint angles of the simulated robot, the corresponding Naoqi function was used (Section D, Equation D.2). The robot walked on a planar surface of WEBOTS and the period of sampling was between $15\mu s$ and $20\mu s$. From the joint angle trajectory recordings obtained, it was observed that the angles of the eighteen joints exhibited large variations during various sampling periods. These variations are mostly evident in the peaks of the oscillatory signals (Figure 4.1.1), and are possibly due to external disturbances being introduced into the system by the environment e.g. gravity. As the figure indicates, there seems to be some inherent variability between different gait cycles, which is also reflected in the simulated output, where in many cases the gait appears unstable.

To gain a better insight into the characteristics of the trajectories produced by these joints, our first action was to reduce the gaps in the sampling values caused by the large sampling intervals. For this reason, we employed a lowpass interpolation method, the MATLAB function *interp* [33], in order to generate intermediate points between those sampling intervals. Interpolation is a well known mathematical process, that uses an iterative method, in which a series of discrete points is approximated through some analytical function (e.g. polynomial). This function is then used to generate intermediate points within the given discrete set. The result of the interpolation can be seen by comparing (Figure 4.1.2) with (Figure 4.1.3). On the former figure, there are plots with no interpolation, while, on the latter, the plot shows the interpolated trajectory.

The Fourier properties of the target signal have an important role in the CPG learning process, as they are the frequencies to which the CPG oscillators will converge. Consequently, prior determination of these frequencies will enable us to evaluate the quality of the learning process. To determine these values, we have carried out a Fourier analysis

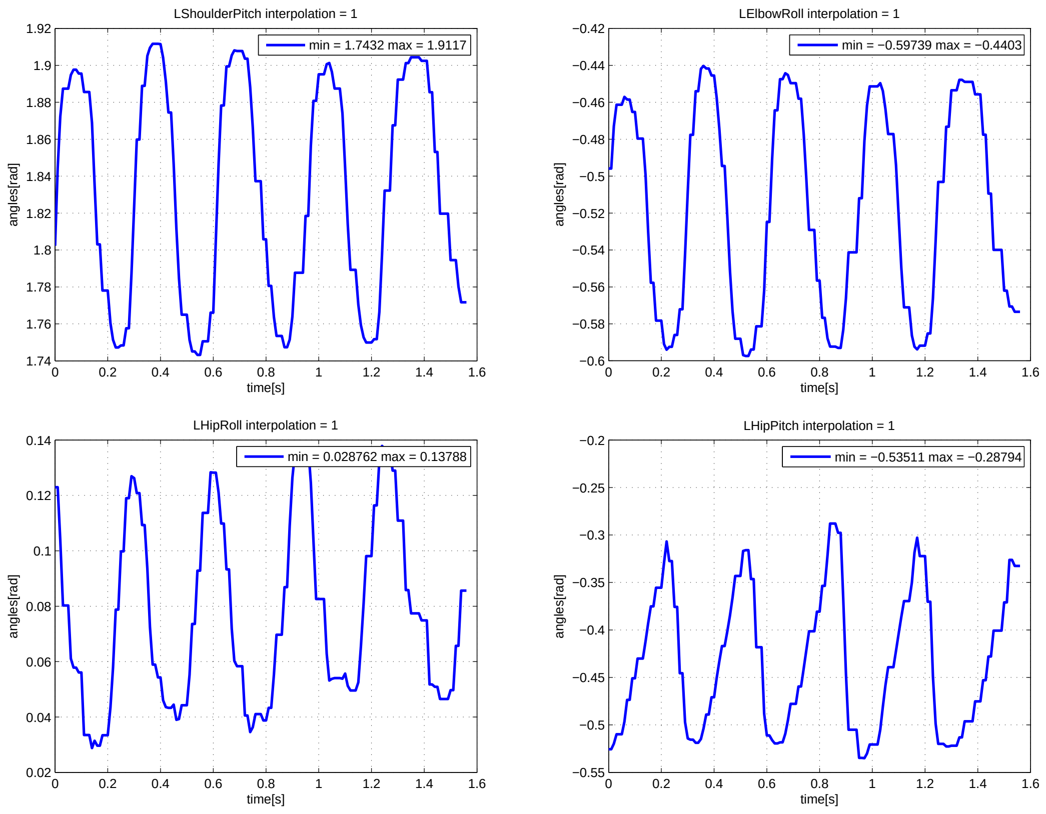


FIGURE 4.1.1: Joint angle trajectories of the Left Shoulder Pitch, Left Elbow Roll, Left Hip Roll and Left Hip Pitch joints. The x-axis is time (sec) and y-axis is the joint angle (radians). The legend indicates the minimum and maximum angle in all 4 periods of the joint trajectories presented.

of the recorded joint trajectories, which was used to empirically determine the number of neurons needed and initialize the oscillators. As it can be be seen in (Figure 4.1.4), about 8-10 dominant frequencies appear for each joint trajectory, which fall in the range of 0-40 Hz. To allow our network to adapt to these frequencies, we have used 9 neurons, and distributed their initial frequencies uniformly in the range of 0 - 90 Hz.

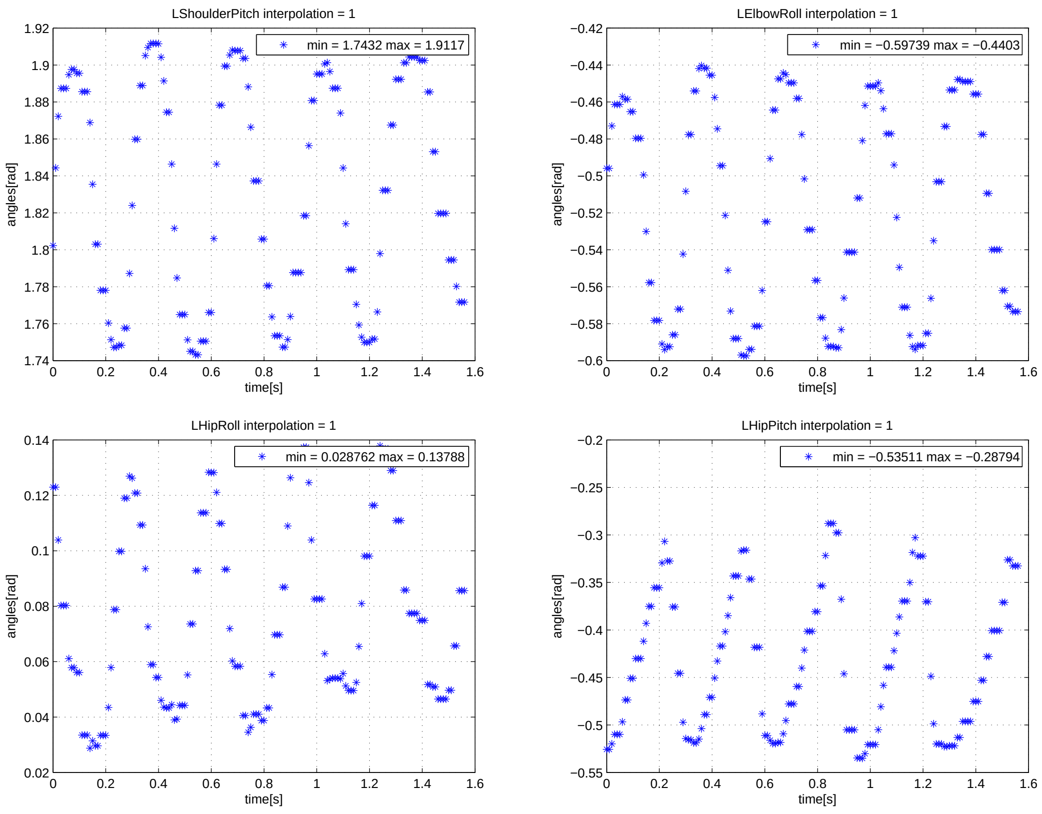


FIGURE 4.1.2: Joint angle trajectories of the Left Shoulder Pitch, Left Elbow Roll, Left Hip Roll and Left Hip Pitch joints. The x-axis is time (sec) and the y-axis is the joint angle (radians). The legend indicates the minimum and maximum angle in all 4 periods of the joints, with no use of interpolation method.

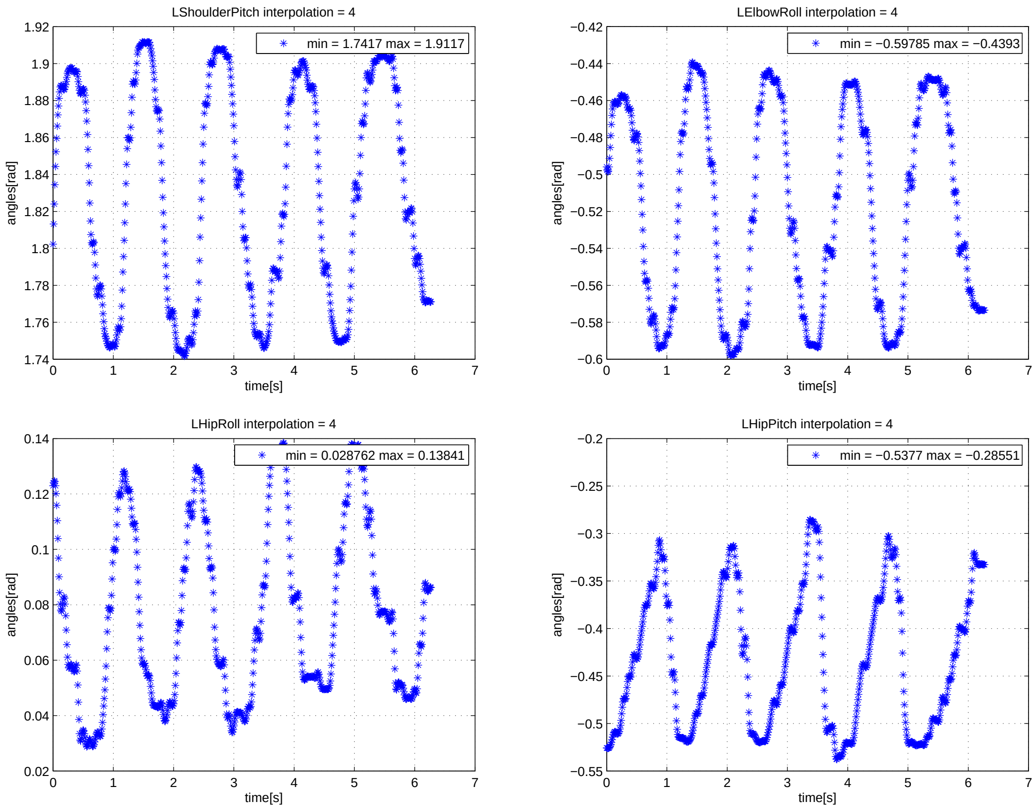


FIGURE 4.1.3: Joint angle trajectories of the Left Shoulder Pitch, Left Elbow Roll, Left Hip Roll and Left Hip Pitch joints. The x-axis is time (sec) and the y-axis is the joint angle (radians). The legend indicates the minimum and maximum angle in all 4 periods of the joints with use of interpolation method, which adds 4 intermediate points.

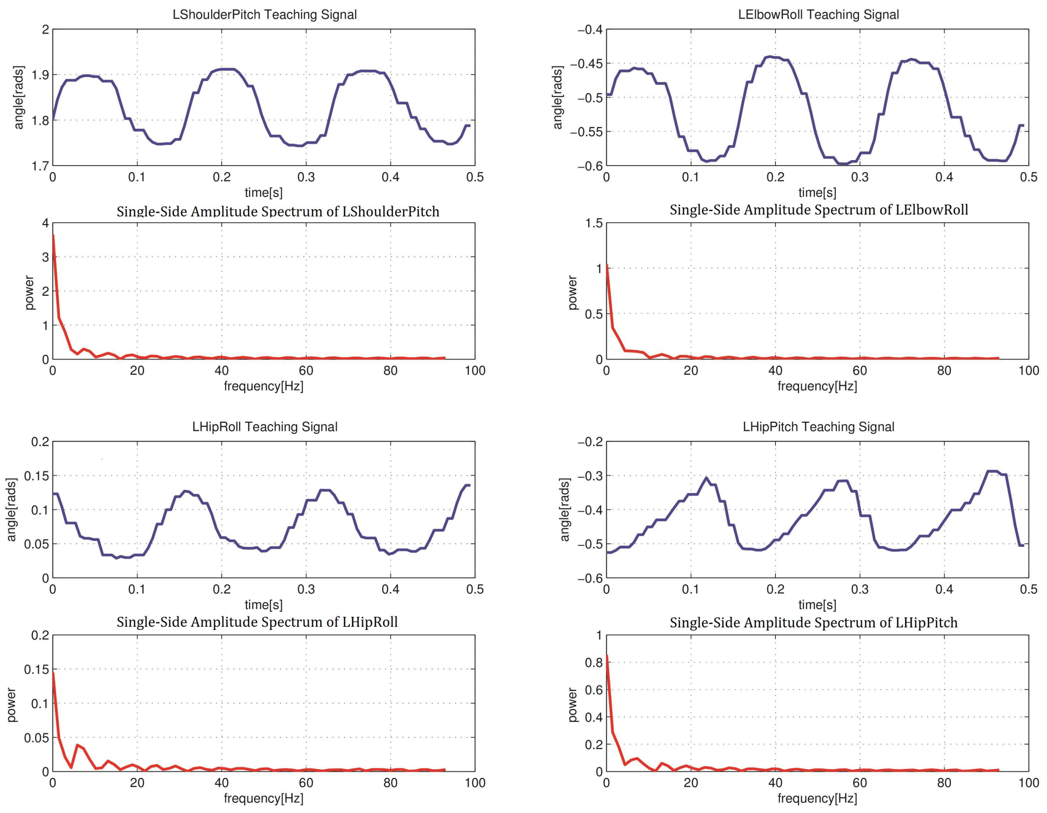


FIGURE 4.1.4: The fast Fourier transform of the Left Shoulder Pitch, Left Elbow Roll, Left Hip Roll and Left Hip Pitch joint angles. Under each signal, the corresponding Fast Fourier Transform (FFT) is shown. The x-axis shows the corresponding time (sec) and the y-axis shows the joint angles (radians). For the FFT plots, the x-axis shows the frequency (Hz) and the y-axis shows the power.

4.2 Training of the Central Pattern Generators

From the analysis carried out in the previous section, we have derived various important characteristics of the joint angle signals produced by the robot during walking, the most important being the oscillatory trajectory followed by each joint, and the form of this trajectory (Figure 4.1.1). To approximate and reproduce these signals on the robot, we have used appropriately trained CPGs. In the following, we outline some design choices made for the architecture of the CPGs, and details regarding their learning process.

4.2.1 Architecture and Initialization of the CPG

The selected CPG model controlling the movement of each robot joint (see chapter 3.1.3) consists of a chain of Adaptive Hopf oscillators (Equations : 3.21 - 3.25), each one of which is responsible for learning a specific frequency component of a given input signal. Learning is accomplished through an iterative gradient descent method, in which the cumulative output of all neurons of the CPG is adjusted, by iteratively reducing some error norm on the teaching signal. For the selected CPG network, these adjustments are implemented using Hebbian associative learning.

Due to the gradient descent nature of learning, the initialization of the parameters of the network will have a strong effect on the learning process. One important part of this initialization is to assign correctly the frequencies of each neuron in the network. There are various ways to distribute these values; however, from our investigation, it was concluded that a uniform initialization $(10, 20, ..., 90Hz)$, within a range that contains all dominant Fourier frequency components of the signal, is the best way to ensure convergence.

Another important part of the CPG design was the selection of the number of neurons. This number is positively correlated to the time required to learn a given signal, and has a strong effect on the qualitative characteristics of the signal that will be produced by the network. To determine this number, we carried out an empirical investigation, from which we concluded that a large number of neurons, e.g. around 20, will lead to smaller errors in the learning process; however, it will also require significantly larger processing times. As an acceptable trade-off, it was chosen to use 9 neurons per CPG.

4.2.2 Learning for the CPGs

The error of the learning process is calculated as the absolute value of the teaching signal minus the output of the CPG: $|Pteach - Qlearned|$. The error rate is high at the beginning of the learning process (indicating a large deviation between the two), and decreases progressively with time, as gradient descent performs changes to the neurons. This decrease is dependent on the parameters $\epsilon, \eta, \mu, \gamma$ of the network and the error of the learning. However, we note that the value of this error is not the only indicator of the quality of learning. If the error falls rapidly to a very low level, the dynamic properties of the signal will not be adequately adapted by the CPG. To prevent this, one can reduce the values of the learning rate constants to a lower level.

Furthermore, the effect of the constants in equations 3.21 - 3.25 on the behavior of the CPG needs to be understood. These affect, not only the quality of the learning process, but also the time required for convergence. If the learning rate ϵ and coupling constant η are high, the CPG is going to converge to the learning signal faster. In Figures 5.1.5 - 5.1.8 (Chapter 5.1), we demonstrate the effect of these parameters on the learning process of the CPG. When high values, like $\epsilon = 10.9$ and $\eta = 10.4$, are used, then the CPG network adjusts rapidly to the properties of the given teaching signal, but fails to generalize. To prevent this, we have experimented with a large number of values for these parameters. Lower values like $\epsilon = 0.08$ and $\eta = 0.02$, where found to be more appropriate. Results using these values are shown in Figures 5.1.1 - 5.1.4.

Apart from the above parameters, a significant role in the learning process of the model is attributed to the parameter *tspan*, which specifies the interval of integration used by the ode45 solver of Mathworks to solve the differential equations [34]. The value of *tspan* needs to be larger than two, because of the structure of the ode45 solver. The effect of this parameter on a specific joint angle is shown in Figure 4.2.1. In our implementation, $tspan = 3$ was selected. If the value of *tspan* increases too much, differentiation will occur over a larger dt, leading to worse approximation of the oscillations, which in turn can cause vibrations in the joints.

During the learning process, the Hebbian learning reduces the value of the error signal, and allows the output of the CPG to approximate the teaching signal. When the learning process comes to an end, the learning signal is set to zero, and the CPG can reproduce the initial trajectories in the absence of any teaching feedback. To determine whether the learning process has come to an end, we have used three empirical rules: small rate of change of the final error, small rate of change of the frequency of each oscillator, and visual inspection of the teaching and output signals of the network. After learning, this teaching signal can be reproduced sufficiently well by storing the number

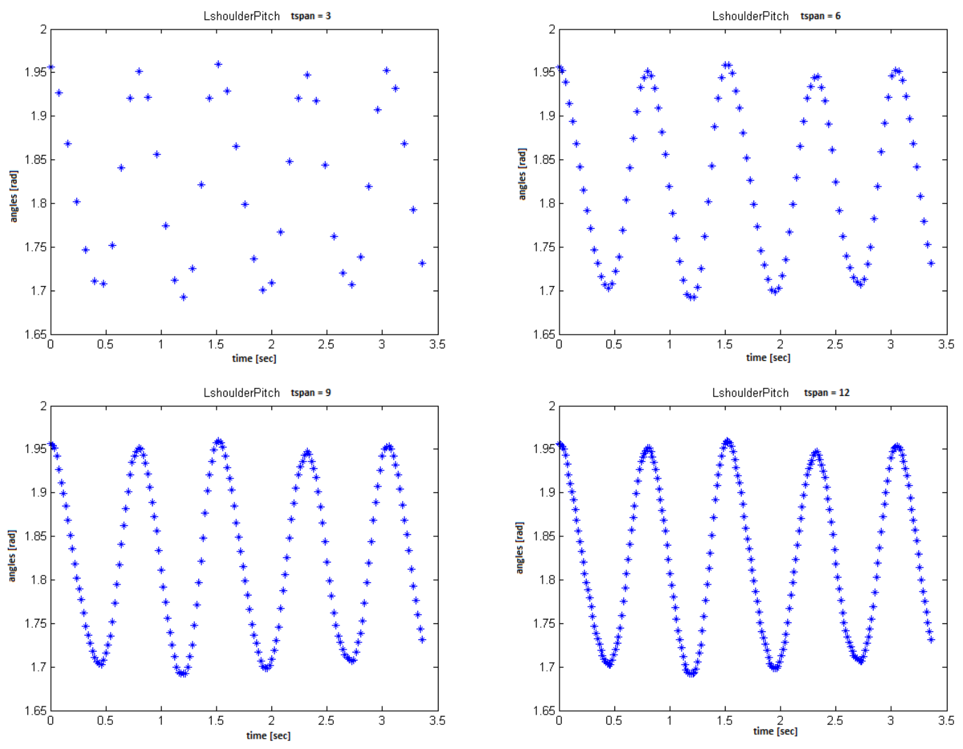


FIGURE 4.2.1: Plots of the Left Shoulder Pitch joint trajectory, shows the effect of the *tspan* parameter, for four different time values, namely for *tspan* = 3, 6, 9, 12. In each plot, the x-axis shows the time in seconds and the y-axis shows the angle of the joint in radians.

N of oscillators, the coupling constants τ and η, the parameter μ, and the state of all oscillators $x_i, y_i, \omega_i, \alpha_i, \phi_i$, for $i = 1 \dots N$, at the final time step of the procedure.

4.3 Optimization of the CPGs with Genetic Algorithm

To optimize the CPG gait behavior, we have employed a Genetic Algorithm (GA), from the GAUL open library [5], specifically the Darwinian GA [35], an iterative stochastic optimization method. The implementation of the GA involved two aspects: (i) formulation of the objective function, (ii) design of the evolutionary procedure. These are outlined below.

4.3.1 Objective Function

To optimize the CPG, we evaluate an objective function which is related to the distance that the NAO robot travels for each CPG iteration. This metric is required to be maximized. This objective function is:

$$\frac{1}{walkDistance + 0.01} \tag{4.1}$$

4.3.2 Evolutionary Procedure

In accordance to general guidelines of genetic algorithm design, a population of 30 chromosomes and 20 generations was considered sufficient. This is also evident from the optimization process, which required a small amount of generations to converge. To initialize the chromosomes, for the ϵ and η parameters we used values 0.001 *to* 1 and for the γ and μ parameters values from 0.001 *to* 2. A general guideline when creating a chromosome is to represent it as a binary number, but the GAUL library gives the possibility to use a floating number representation of the chromosome.

To design our evolutionary procedure, we have used 3 genetic operators: mutation and crossover, as well as a migration scheme. Crossover is useful for exploitation, i.e. traversing locations of the parametric space that are closed to the region defined by the two parent chromosomes. To implement this, we have used a single point crossover rule, in which a point is selected in the two parent chromosomes, and all data beyond that point are swapped, in order to create two children (Figure 3.2.3). Mutation is a mechanism that reinforces the randomness of the chromosomes (Figure 3.2.2), by adding a small variation to each value. It is useful for traversing unknown locations of the parametric space, and can prevent the optimization procedure from getting stuck in local optima. For the given experiments, a mutation value of 0.2, migration 0.1 and crossover value 0.9 was used. In addition, to ensure that some of the higher score solutions will be carried onwards across generations, we have used a migration scheme. For this reason, the best chromosomes from each population were transfered to the next generation without applying any changes to them.

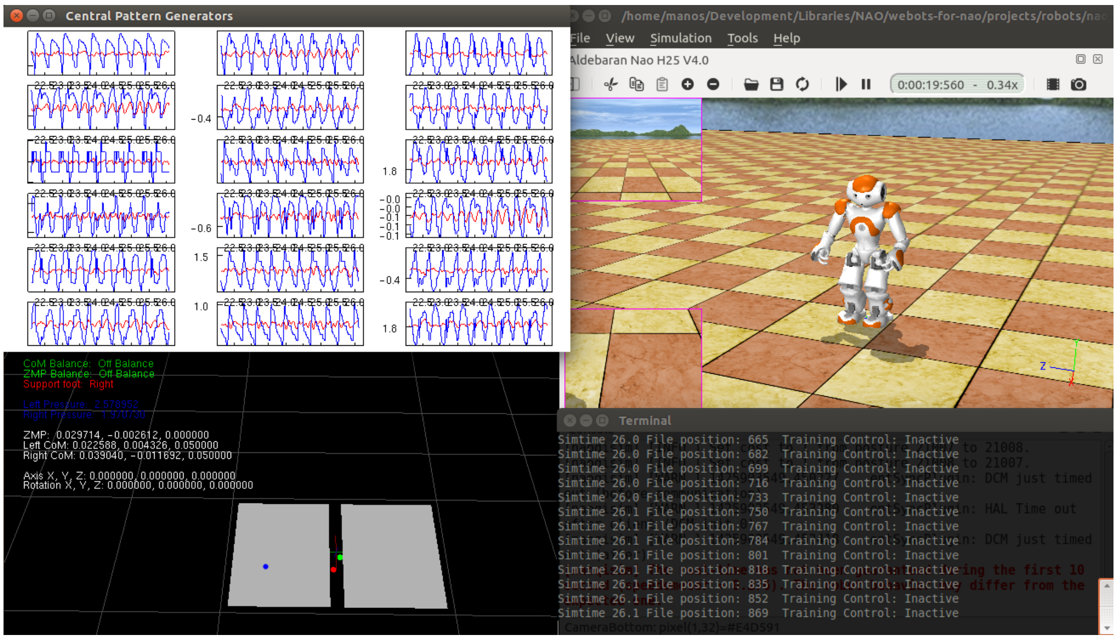
4.4 Software Tools

For the implementation of this Thesis, a variety of different software tools used. Starting from the Webots, which is a mobile robotic simulation software, and provides an emulator of the NAO V4 robot. This robot is fully customized with the real robot's sensors, dimensions, links, masses and joints. The problem faced up with the use of this tool was the delay, which was at a range of 60%-70% of the real robot speed (see Appendix A).

The C++ language was used for the communication with the simulated robot NAO, via the NAOqi OS (Appendix C), which is the operating system of NAO. All the joint trajectories were recorded and the sensory values of the robot obtained (COM, CoP, WalkDistance). For the calculation of the gait distance, a capture of position was occurred, before the start of the gait and at the end of the gait. The norm of those two points gave the distance of the gait. For this operation the $getPosition()$ function was used [2]. This function is a built-in NAOqi operation that gives the 3D coordinates of the robot's position (x, y, z); only the x, y data were used. The multi-paradigm computing environment Matlab was the main tool used for processing. All the plots and the coding for the CPG model was made here in Matlab.

The first attempt with the optimization process was made with the Matlab genetic algorithm functions. A communication was established between the Matlab environment and the Microsoft Visual Studio. Matlab had the control of the learning process, at the time that the CPG network fully adapt the learning signal the process stops and Matlab passes the joint angles from the CPG network to the Visual Studio(VS). VS works like a middle layer tool, it communicates with the Webots and passes the new joint trajectories to the simulated NAO robot. NAO executes the gait locomotion, parallel sensory feedback is captured and evaluated from the VS. When the locomotion stops, a new instance of the objective function is created for the Genetic Algorithm. This process, proved inefficient, because the time needed for specifying optimal solutions was large (approximately 5 days for 20 generations of 30 populations). The solution adopted was to implement both the Genetic Algorithm and the CPG learning process in a C++ environment.

A compatible with C++, Genetic Algorithm library used called GAUL, the CPG was implemented in C++ and the whole system gained significant in speed. A run with the same population and generation is now taking approximately 30 hours. A graphical display of the system while the CPG is in learning mode, and the emulator is in standInit position waiting for the CPG reproduce mode, is shown here Figure 4.4.1.


Central Pattern Generators
/home/manos/Development/Libraries/NAO/webots-for-nao/projects/robots/nao
File View Simulation Tools Help
Aldebaran Nao H25 V4.0
0:00:19:560 - 0.34x
Terminal

```
CoM Balance:  Off Balance
ZMP Balance:  Off Balance
Support foot:  Right

Left Pressure:  2.578952
Right Pressure:  1.970730

ZMP:  0.029714, -0.002612, 0.000000
Left CoM: 0.022588, 0.004326, 0.050000
Right CoM: 0.039040, -0.011692, 0.050000

Axis X, Y, Z: 0.000000, 0.000000, 0.000000
Rotation X, Y, Z: 0.000000, 0.000000, 0.000000
```

```
Simtime 26.0 File position: 665  Training Control: Inactive
Simtime 26.0 File position: 682  Training Control: Inactive
Simtime 26.0 File position: 699  Training Control: Inactive
Simtime 26.0 File position: 716  Training Control: Inactive
Simtime 26.0 File position: 733  Training Control: Inactive
Simtime 26.1 File position: 750  Training Control: Inactive
Simtime 26.1 File position: 767  Training Control: Inactive
Simtime 26.1 File position: 784  Training Control: Inactive
Simtime 26.1 File position: 801  Training Control: Inactive
Simtime 26.1 File position: 818  Training Control: Inactive
Simtime 26.1 File position: 835  Training Control: Inactive
Simtime 26.1 File position: 852  Training Control: Inactive
Simtime 26.1 File position: 869  Training Control: Inactive
```

FIGURE 4.4.1: Learning process for the CPG model. The top left window plots the group of oscillators used to control NAO's joints. The bottom left window shows the robot's center of mass and pressure points for each foot, plotted against its support polygons. The right window is the simulation of the robot, in the Webots simulator.

Chapter 5

Results

In this chapter, we outline the results from the computational experiments carried out throughout this Thesis. Results obtained during the training of the CPG controller are shown in Section 5.1 and results from its optimization using GAs in Section 5.2.

5.1 Learning in the CPG

As discussed before, the learning process in the CPG model is dependent on the ϵ, η, μ and γ parameters. To determine their values, we have carried out a series of simulation trials. In the implementation of the CPG learning process, 18 joints participate. This was proved a time consuming process, and it was chosen to present only 4 characteristic results of those simulations, i.e the ones for the LShoulderPitch, LElbowRoll, LHipRoll and LHipPitch joints. During the early analysis of the simulation results, it was observed that from the four parameters, only the two have the most significant impact in the adaptation of the learning process. For this reason, it was decided to keep stable values in the μ and γ parameters, and vary the ϵ and η parameters. Further analysis showed that, as the ϵ and η values became high, the learning process may converge faster, but the qualitative characteristics of the output signal was not good enough. In contrast, when the values of these parameters were kept relatively low, the learning process lasted longer, in order for the CPG model to fully adapt to the learning signal, but the qualitative characteristics of the output signal was better.

As detailed below, we analyzed further the learning process by reviewing some key points in the simulation. Plots in Figure 5.1.1 - 5.1.4 shows the training in four different phases of the process for the LShoulderPitch, LElbowRoll, LHipRoll and LHipPitch joints, using learning rate $\epsilon = 0.08$ and coupling constant $\eta = 0.02$. Each row of these figures contains two sub-figures, the one on the left shows a piece from the timeline of the adaptation process, in green color shows the output of the CPG signal and with blue color shows the Teaching signal at the same period of time. The one the right sub-figure shows the error rate of the learning process between the two signals that appears on the left sub-figure. The error is calculated as the absolute value of the Teaching signal minus the CPG output. The error at the beginning of the learning process is high, as it can be seen in the first 4 sub-figures; at this time, the Learning signal is not yet adapted to the frequency of the Teaching signal. In sub-figures five and six, the CPG model starts to converge slowly to the Learning signal and the error of the process starts to fall. In sub-figures seven and eight, at $time > 4000sec$, the CPG model is fully adapted to the Learning signal. As a result, the learning process is stopped and the Teaching signal is set to zero. In the last two sub-figures, the CPG network is capable of reproducing the adapted signal even if the error rate of the system may not be continuously low.

In Figures 5.1.5 - 5.1.8, we set the learning rate to $\epsilon = 10.5$ and the coupling constant to $\eta = 10.9$, which are higher values than above. As the learning process begins, it is observed that in the first six sub-figures, the CPG signal converges rapidly to the Teaching signal. The same observation is made for the error rate, which falls in low values. At this time, the CPG model is fully adapted to the Learning signal. At $time = 80sec$, the learning process is stopped and the Teaching signal is set to zero. However, in the last 4 sub-figures it is observed that, the CPG model is not able to reproduce the teaching signal in the absence of the learning process. There is an obvious problem in the reproduction for both the amplitude and the frequency of the Learning signal.

Therefore, relatively low values of the parameters ϵ and η are necessary for the learning process to be successful.

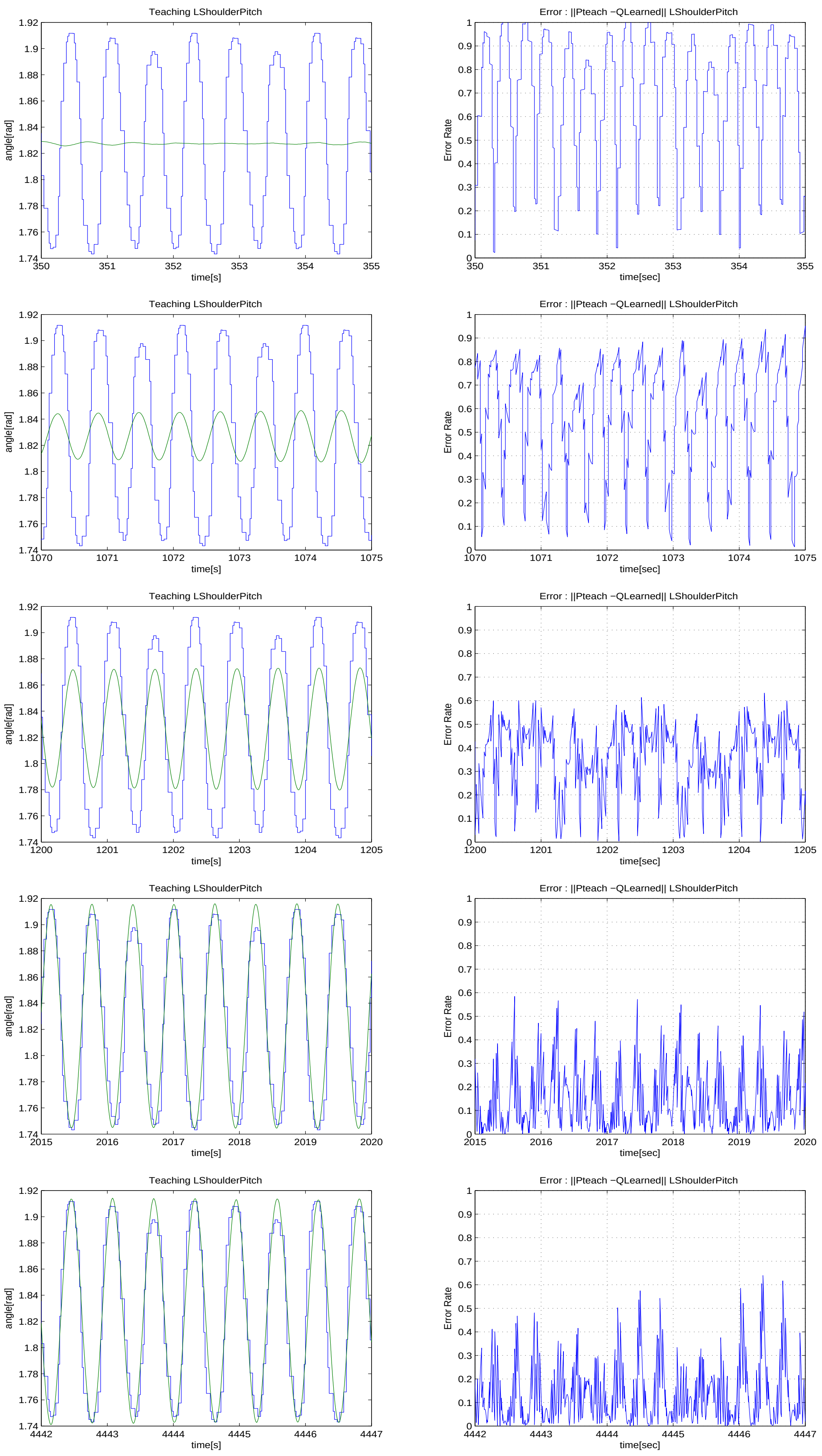


FIGURE 5.1.1: Left plots show the Left Shoulder Pitch signal, the blue color signal is the prerecorded joint trajectory (Pteach signal) and the green color signal is the CPG output (Qlearned). The right plots show the evolution of the Error Rate in the specific times corresponding to the left plot. The learning occurred with low learning rate and coupling constant value.

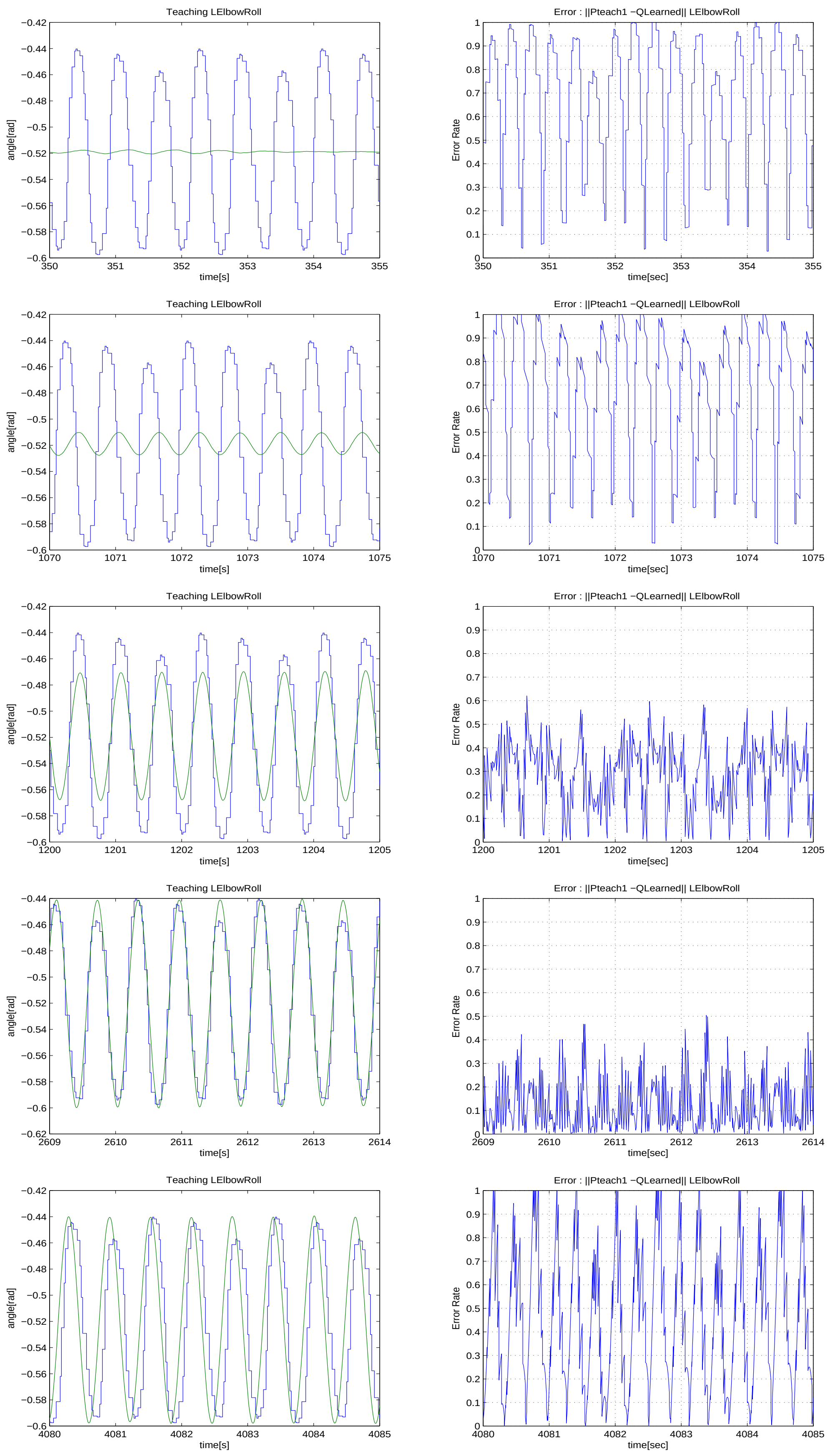


FIGURE 5.1.2: Left plots show the Left Elbow Roll signal, the blue color signal is the prerecorded joint trajectory (Pteach signal) and the green color signal is the CPG output (Qlearned). The right plots show the evolution of the Error Rate in the specific times corresponding to the left plot. The learning occurred with low learning rate and coupling constant value.

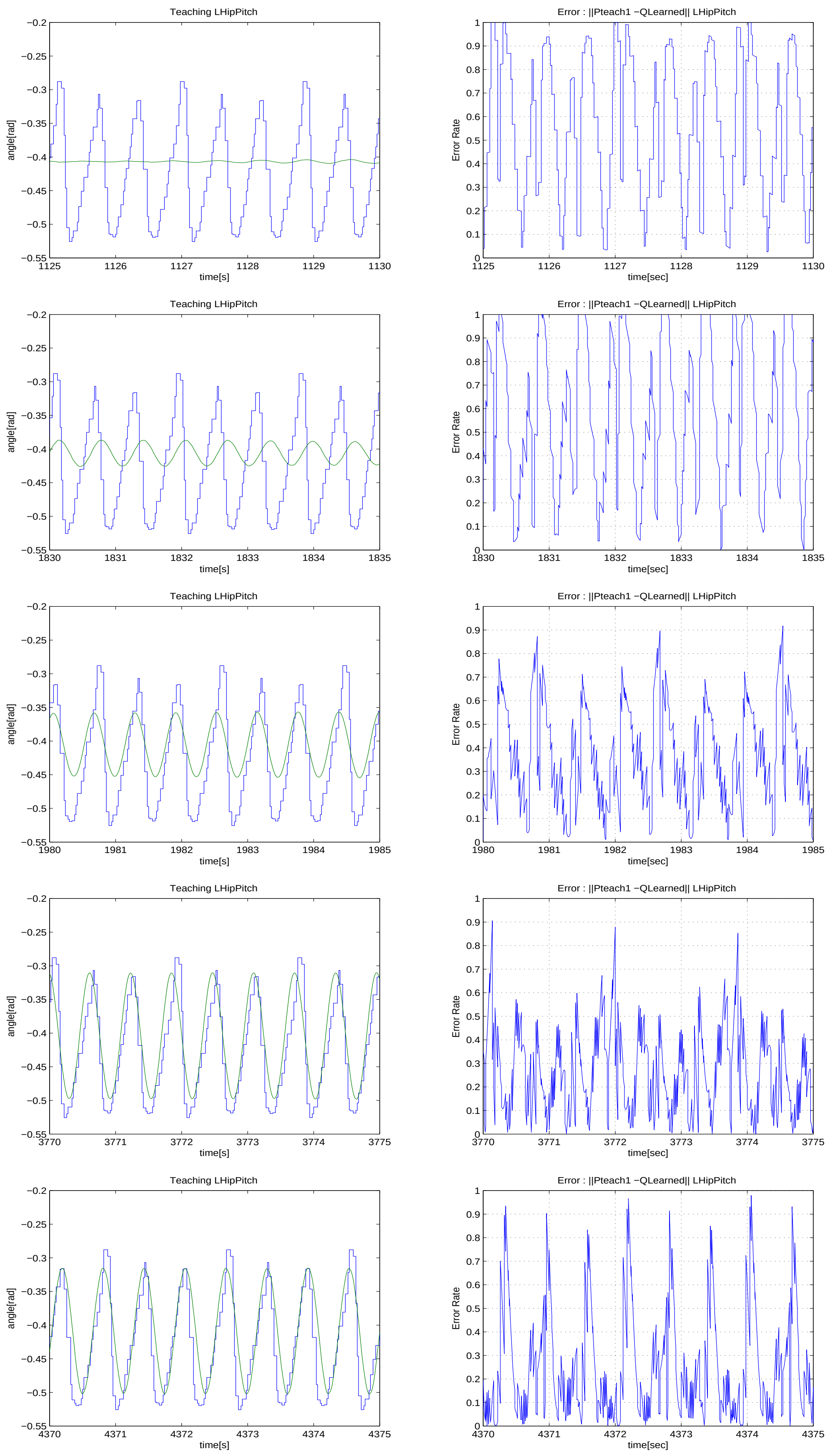


FIGURE 5.1.3: Left plots show the Left Hip Pitch signal, the blue color signal is the prerecorded joint trajectory (Pteach signal) and the green color signal is the CPG output (Qlearned). The right plots show the evolution of the Error Rate in the specific times corresponding to the left plot. The learning occurred with low learning rate and coupling constant value.

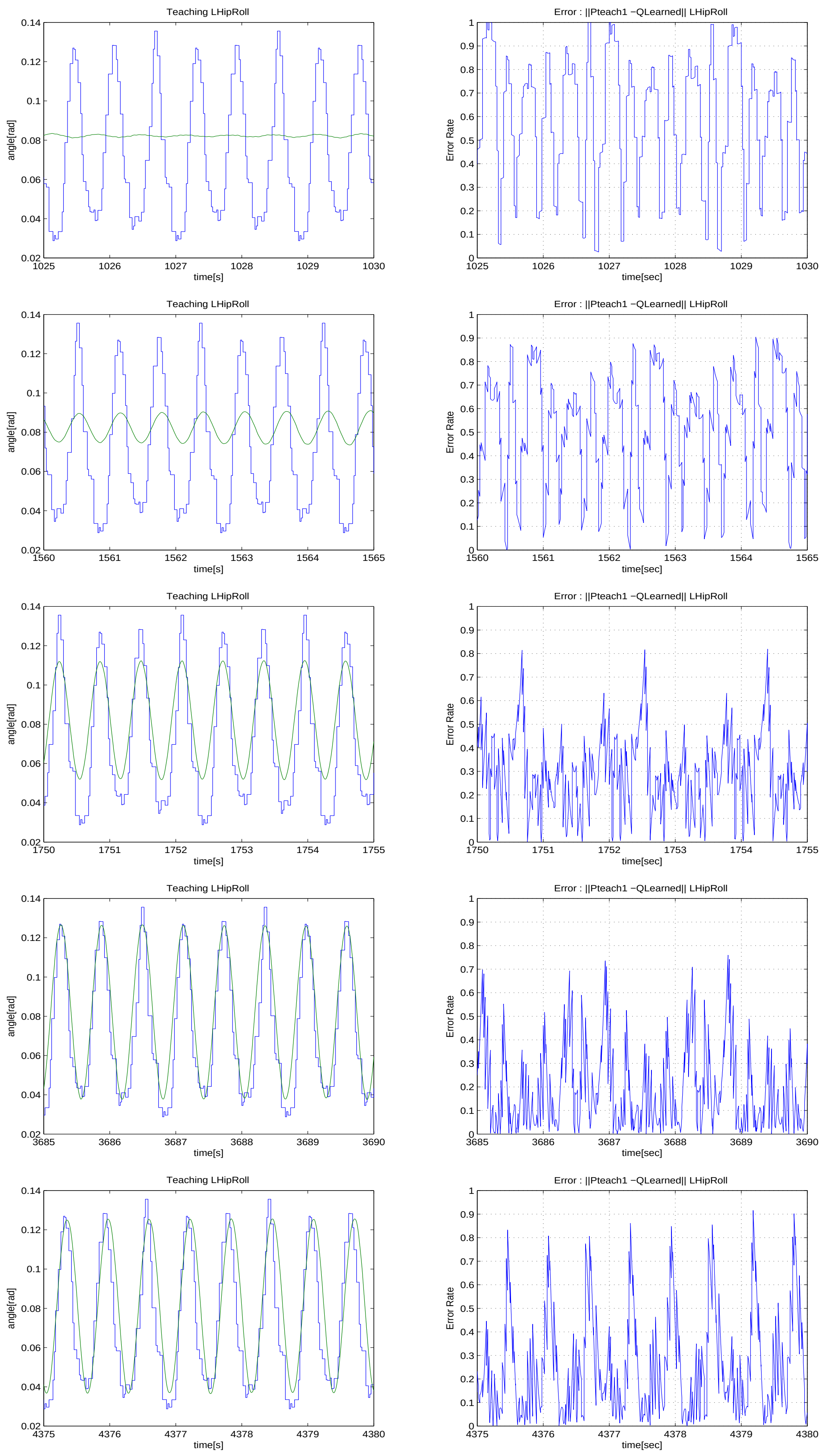


FIGURE 5.1.4: Left plots show the Left Hip Roll signal, the blue color signal is the prerecorded joint trajectory (Pteach signal) and the green color signal is the CPG output (Qlearned). The right plots show the evolution of the Error Rate in the specific times corresponding to the left plot. The learning occurred with low learning rate and coupling constant value.

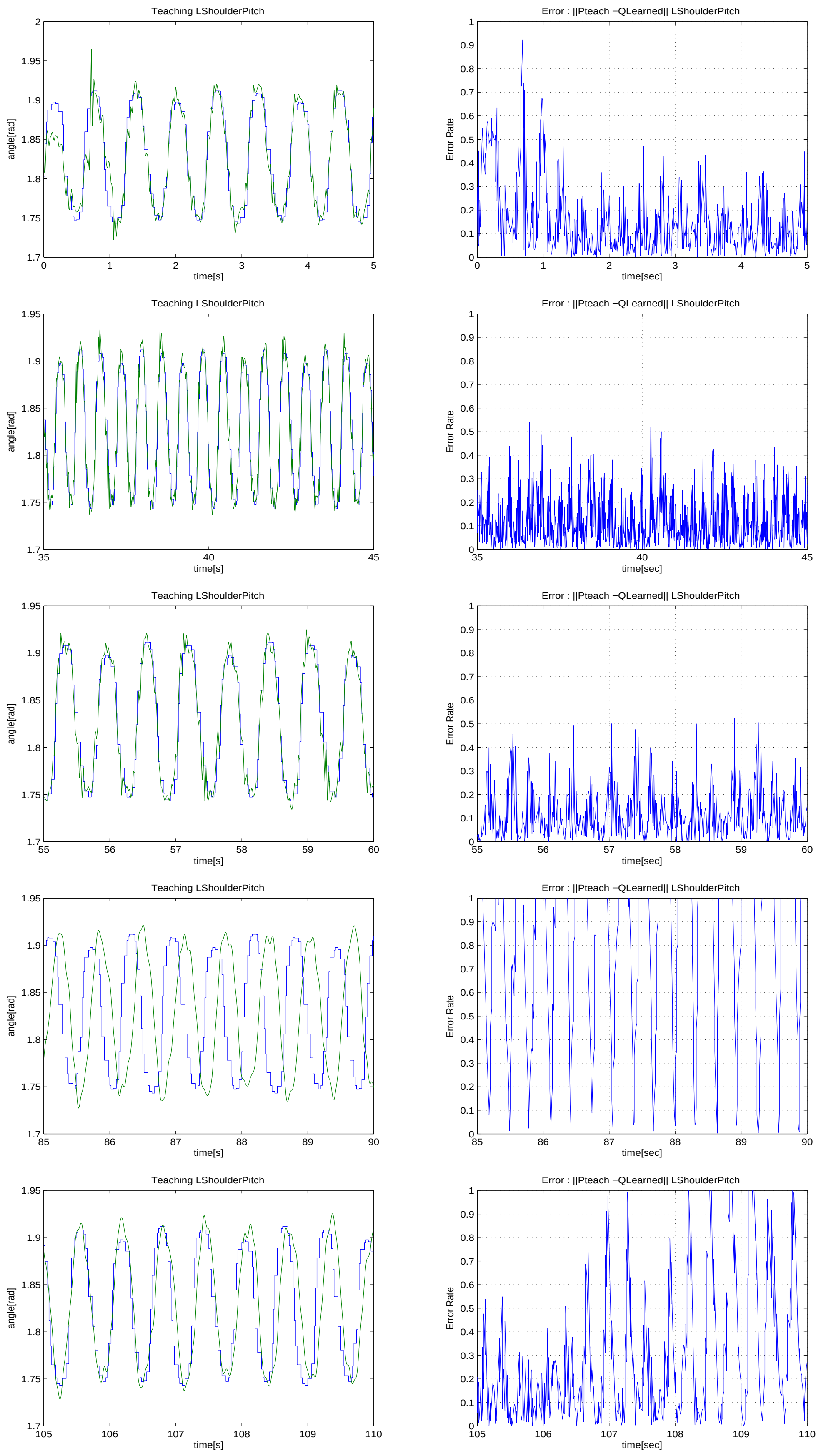


FIGURE 5.1.5: Left plots show the Left Shoulder Pitch signal, the blue color signal is the prerecorded joint trajectory (Pteach signal) and the green color signal is the CPG output (Qlearned). The right plots show the evolution of the Error Rate in the specific times corresponding to the left plot. The learning occurred with high learning rate and coupling constant value.

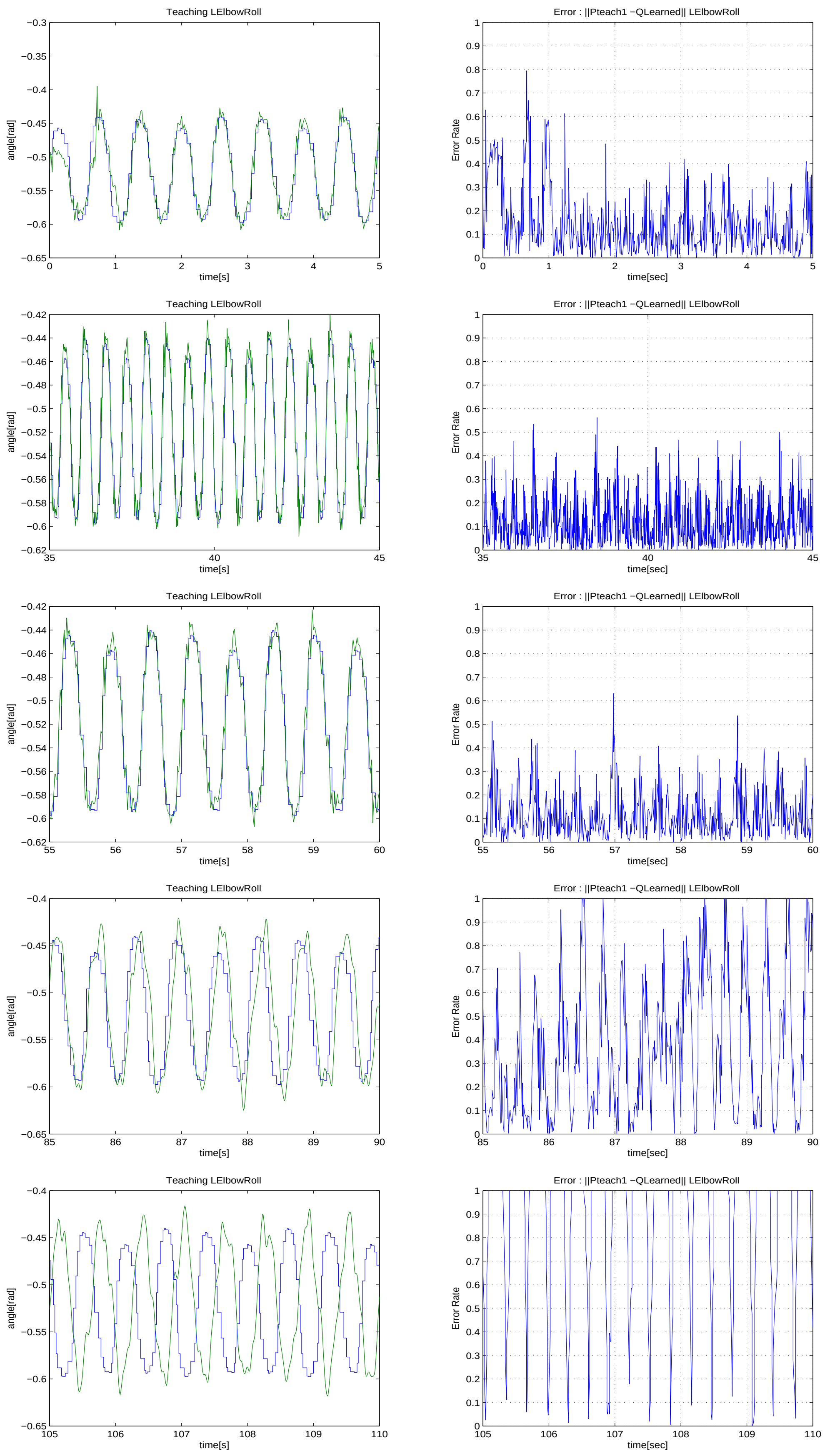


FIGURE 5.1.6: Left plots show the Left Left Elbow Roll, the blue color signal is the prerecorded joint trajectory (Pteach signal) and the green color signal is the CPG output (Qlearned). The right plots show the evolution of the Error Rate in the specific times corresponding to the left plot. The learning occurred with high learning rate and coupling constant value.

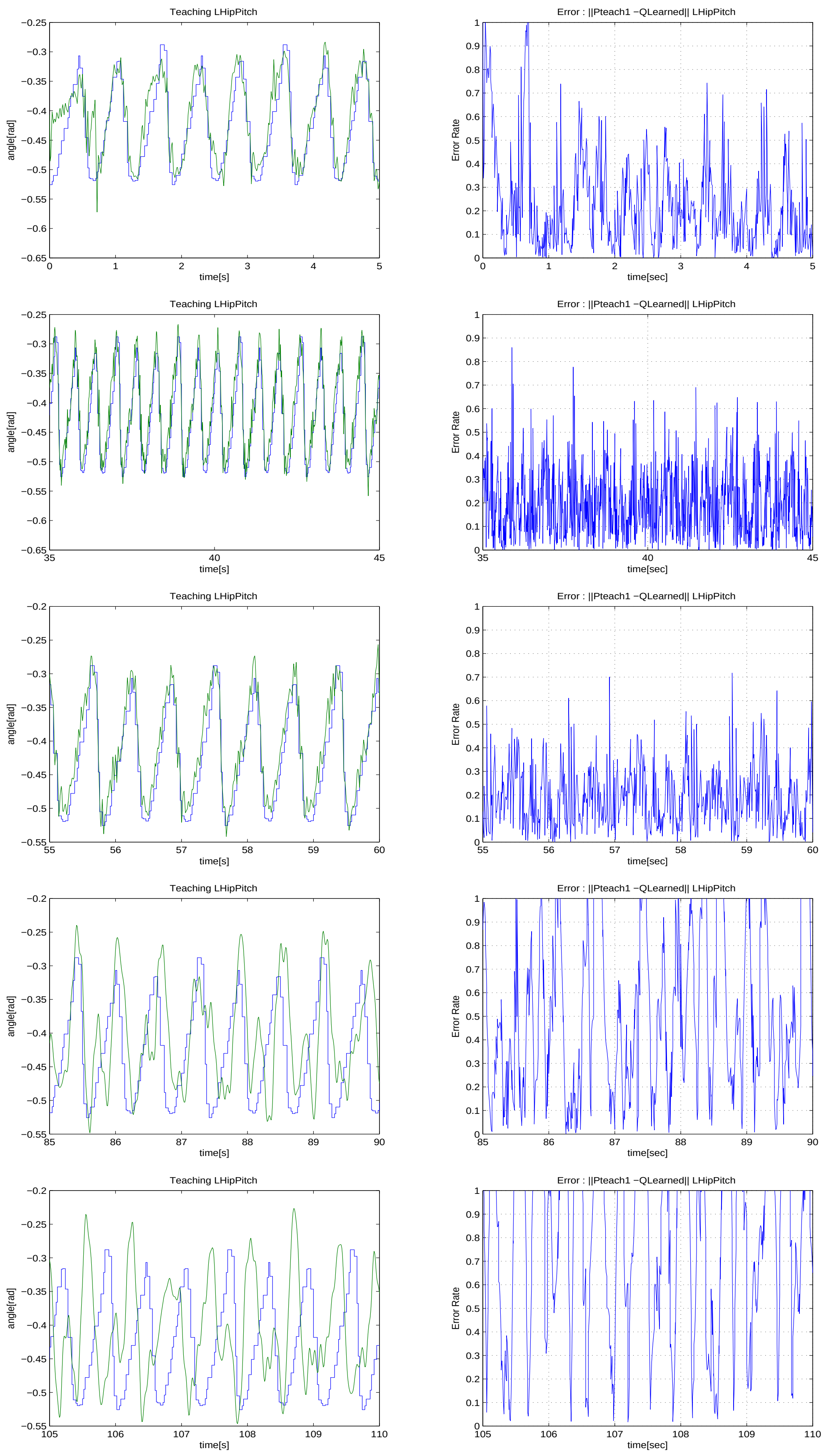


FIGURE 5.1.7: Left plots show the Left Hip Pitch signal, the blue color signal is the prerecorded joint trajectory (Pteach signal) and the green color signal is the CPG output (Qlearned). The right plots show the evolution of the Error Rate in the specific times corresponding to the left plot. The learning occurred with high learning rate and coupling constant value.

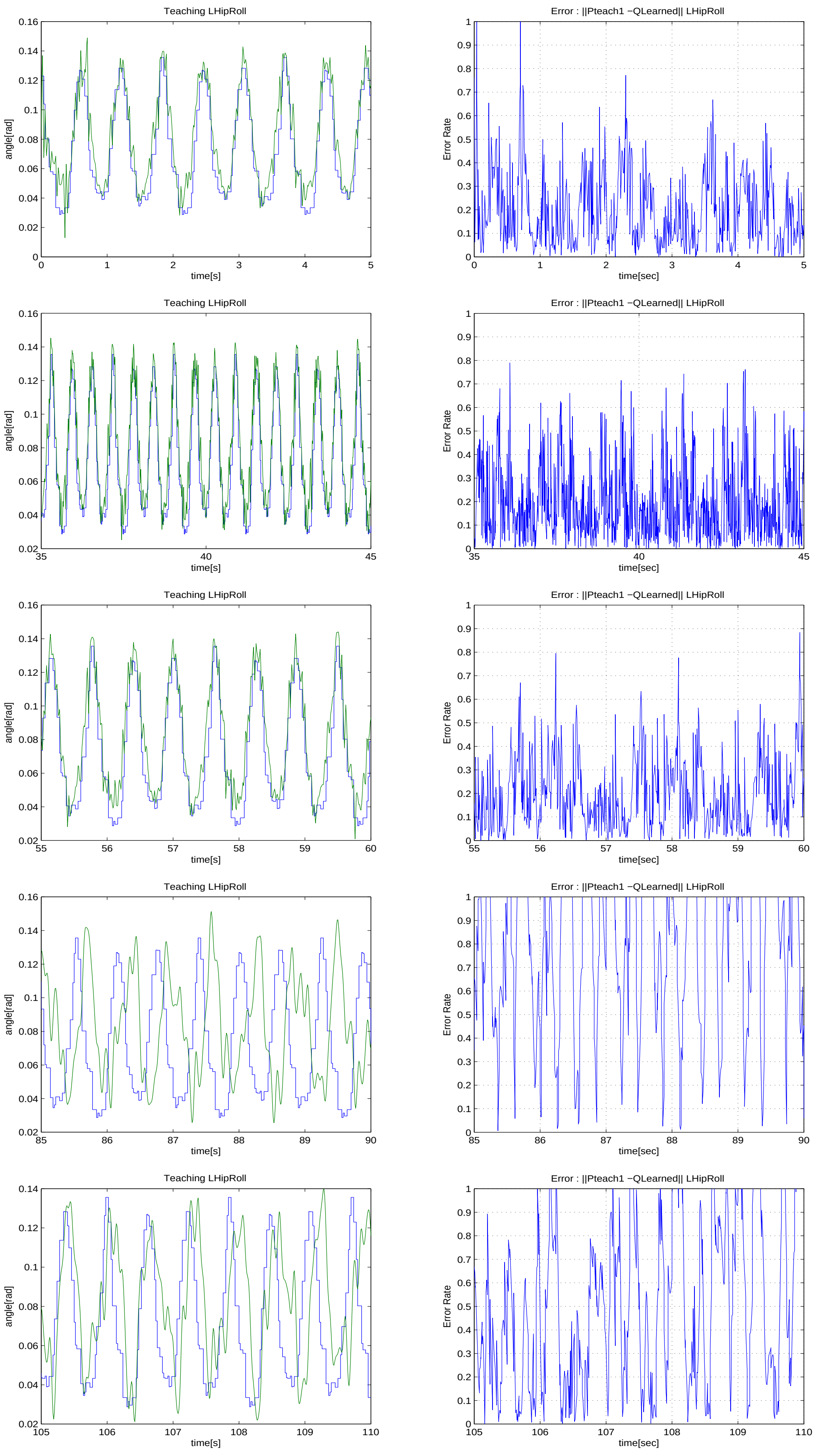


FIGURE 5.1.8: Left plots show the Left Hip Roll signal, the blue color signal is the prerecorded joint trajectory (Pteach signal) and the green color signal is the CPG output (Qlearned). The right plots show the evolution of the Error Rate in the specific times corresponding to the left plot. The learning occurred with high learning rate and coupling constant value.

5.2 Optimization of the System by Genetic Algorithms

As discussed earlier (Section 4.3), to optimize the CPG controller we use a genetic algorithm. The Genetic Algorithm used is part of the Gaul library, which is an open source programming library released under the GNU general public license [5].

During the evolutionary process, chromosomes encoded the parameters $\epsilon, \eta, \gamma, \mu$ from equations 3.21 to 3.25. To evaluate the chromosomes, we used an objective function that is proportional to the distance traveled by the robot. However, we note that this metric alone cannot guarantee the production of a stable gait. Evolution was run for 20 generations, while the population consisted of 30 chromosomes. As discussed previously, 18 joints of the robot were used in order to reproduce the gait. The upper body joints have no contact with the ground; as a result no forces occur to the encoders. Because such a large number of joints slows the optimization process, it was chosen to leave the upper body joints out of the optimization process. To reduce computational costs, we included only the lower body of the robot, locking the upper part joints in the configuration of the standInit pose. Doing that, the simulation runs 50% faster. The joint angle trajectories that were evaluated are from the (R:Right - L:Left) R-LHipRoll, R-LHipPitch, R-LKneePitch, R-LAnklePitch, R-LAnkleRoll joints.

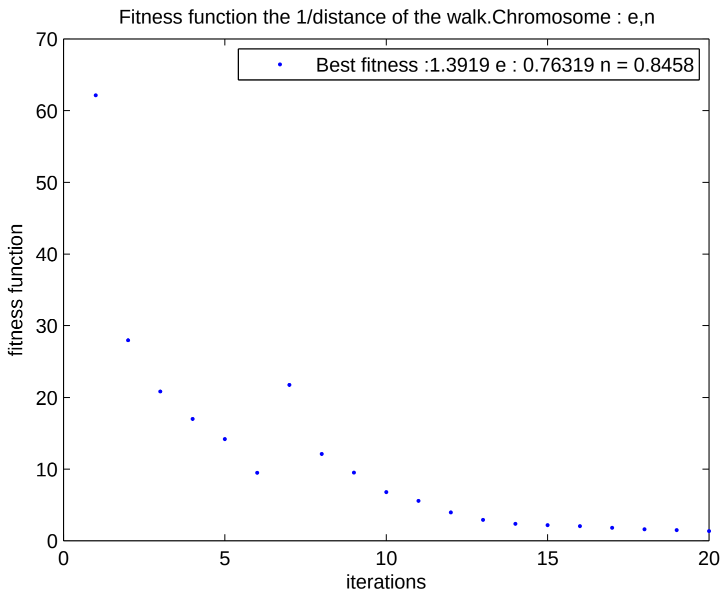


FIGURE 5.2.1: First optimization experiment with the Genetic Algorithm: fitness function is the walking distance, parameters evolved ϵ, η. The x-axis shows the iterations of the genetic algorithm, and the y-axis shows the fitness function.

Figure 5.2.1 presents the results from the first experiment using the GA. The objective function chosen is the $\frac{1}{walkDistance+0.01}$. As the graph demonstrates, chromosomes in the initial generations fluctuated strongly around large values, however, as evolution progressed, the genetic algorithm was able to converge to lower values of the objective function. We also notice the jump in the objective function at generation 7, caused by a built-in mechanism in GAUL, which introduces additional noise in the chromosomes when the rate of change of the objective function is small. This helps evolution avoid getting stuck into local minima. For the given experiment, after the evolutionary process, the best values found are $\epsilon = 0.76319$ and $\eta = 0.8458$. This corresponds to a walking distance of 15 steps.

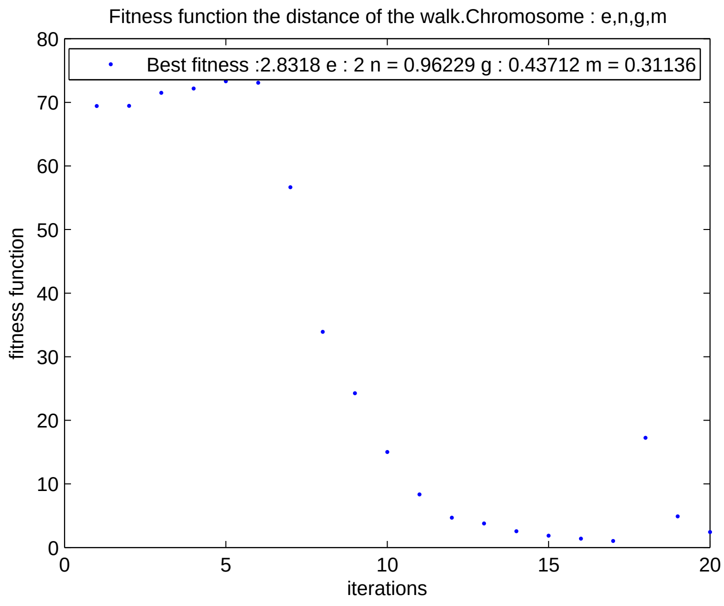


FIGURE 5.2.2: Second optimization experiment with the Genetic Algorithm: fitness function is the walking distance, parameters evolved ϵ, η, μ, γ. The x-axis shows the iterations of the genetic algorithm. The y-axis shows the fitness function.

Figure 5.2.2 shows a second experiment, where the evolution is using four parameters in the encoding of the chromosome. It is evident that the genetic algorithm requires more iterations to converge. However, the inclusion of additional parameters apparently precluded the GA from finding a solution better than before. The best chromosome had fitness function value 2.8318 and chromosome values $\epsilon = 2$, $\eta = 0.96229$, $\gamma = 0.43712$ and $\mu = 0.31136$. This corresponds to a walking distance of 10 steps. Again we notice the jump in the objective function at generations 7 and 18.

As can be seen, there is little correlation between the results. In the first experiment, the GA converges to a minimum that corresponds to the parameter values $\epsilon = 0.76319$ and $\eta = 0.8458$. In the second experiment, the GA converges in different values :$\epsilon = 2$, $\eta = 0.96229$. This can be explained from two facts: first, the system in both experiments is pre-programmed to be executed for specific times, 20 generations of 30 population. Moreover, in the second experiment, the GA had to investigate the values for a chromosome of 4 parameter, unlike the first experiment where the chromosome has

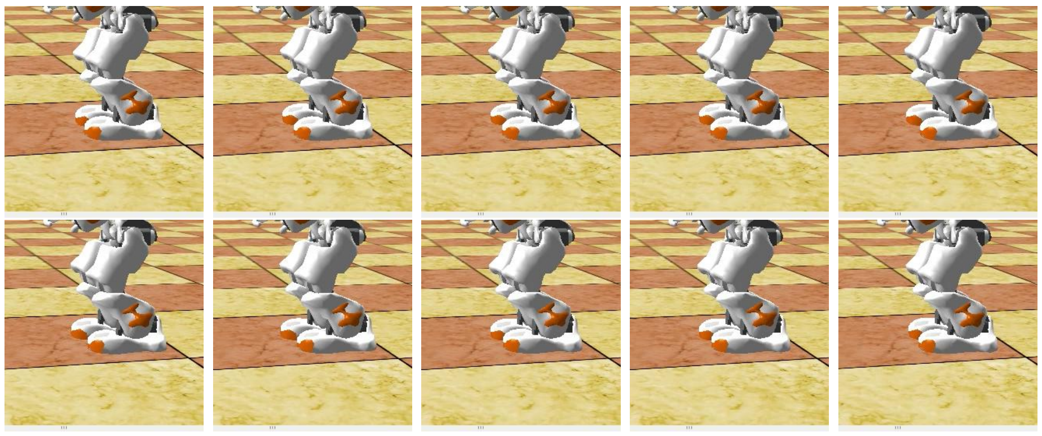


FIGURE 5.2.3: One step of the walking behavior with random parameters $\epsilon = 0.59874$, $\eta = 0.36548$ and $fitnessfunction = 62.5768$, at the begin of the first optimization experiment.

only 2 parameters. So, the GA may need more generations and bigger populations in order to converge to a better result; this means more simulated time. The second fact is that the GA in the beginning of the procedure is creating a random population, which affects the evolution of the simulation. So for this two experiments, the GA starts the simulation from two different manifolds to investigate for the best solution.

Comparing those 2 GA experiments, the system has converged to a better solution in the first experiment, since the robot was able to walk 5 steps longer.

In figures 5.2.3 - 5.2.6 presents the walking behavior of the NAO robot at the beginning of the optimization process, and, after that, at the end of the process with the best parameter values of the GA. In Figure 5.2.7 a closer view of the resulting gait can be seen; each sub-figure shows the left foot of the robot, and the plot above shows the evolution of the angle value of the left knee pitch joint. Also, Figure 5.2.8, shows that the NAO robot is able to walk multiple steps with the use of the CPG network. In this attempt, the best parameter values were used, based on the optimization process.

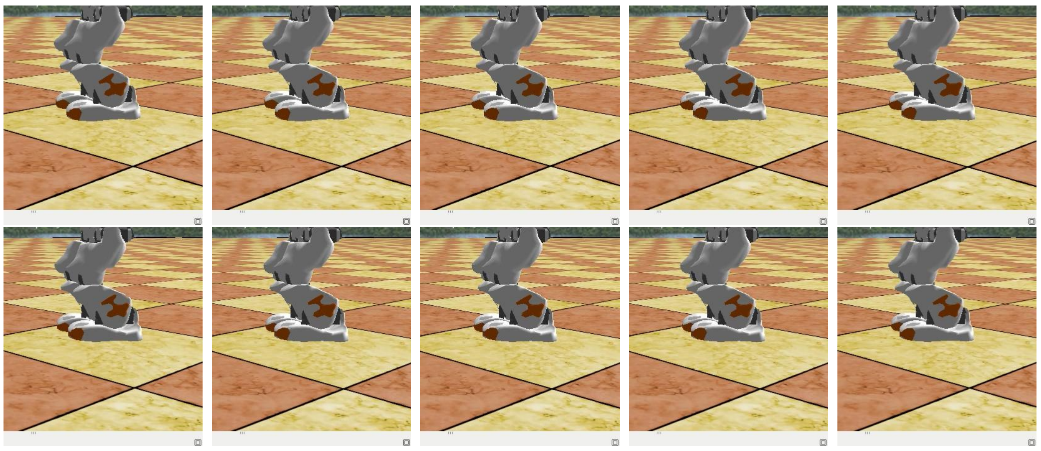


FIGURE 5.2.4: One step of the walking behavior with random parameters $\epsilon = 0.55465$, $\eta = 0.65465$, $\gamma = 0.65749$, $\mu = 0.24194$ and $fitnessfunction = 69.4963$ at the begin of the second optimization experiment.

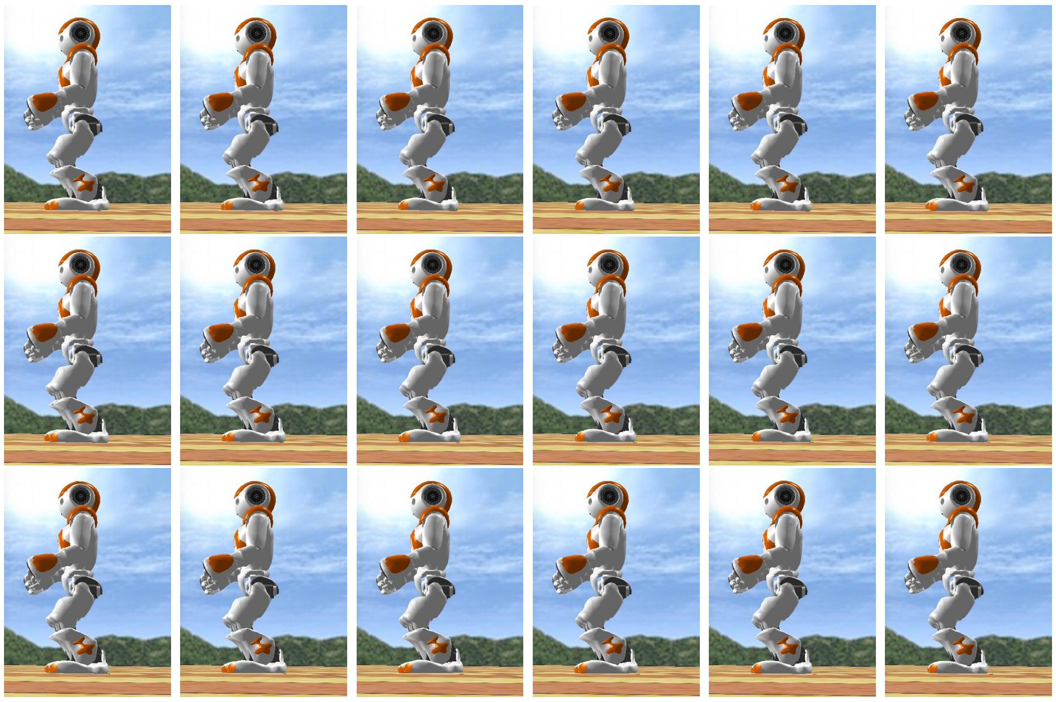


FIGURE 5.2.5: Two steps of the walking behavior with known parameters from the first optimization experiment $\epsilon = 0.76319$, $\eta = 0.8458$, $\gamma = 8.0$, $\mu = 1.0$ and $fitnessfunction = 1.3919$.

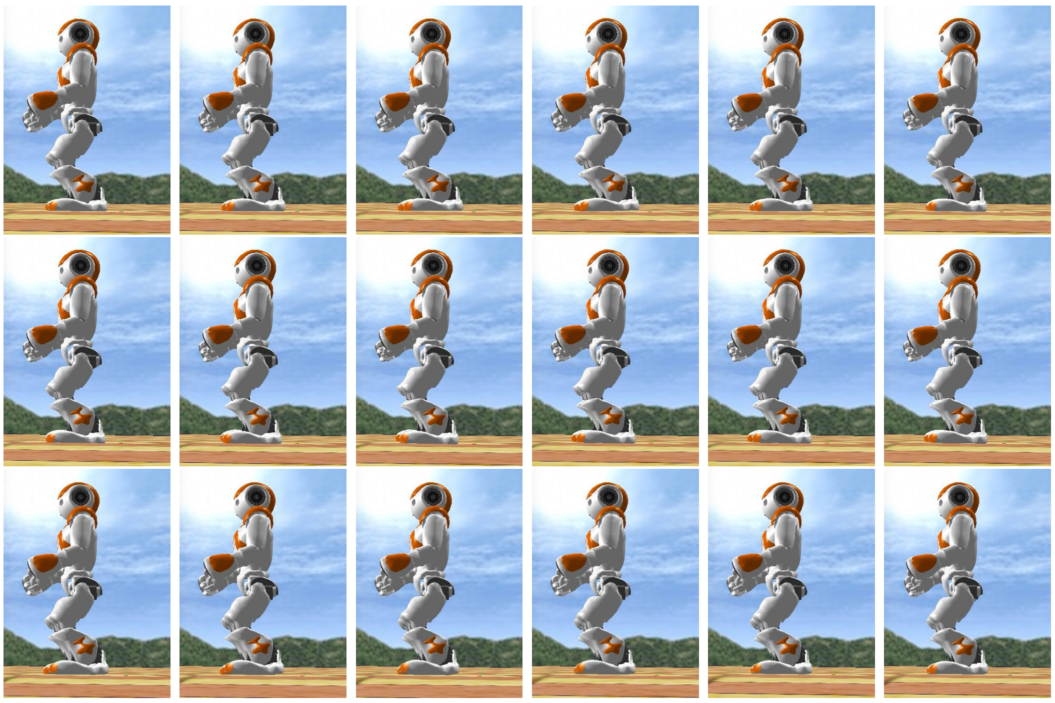


FIGURE 5.2.6: Two steps of the walking behavior with the best fitness function's Ga result with parameters $\epsilon = 2$, $\eta = 0.96229$, $\gamma = 0.43712$, $\mu = 0.31136$ and $fitness function = 2.8318$.

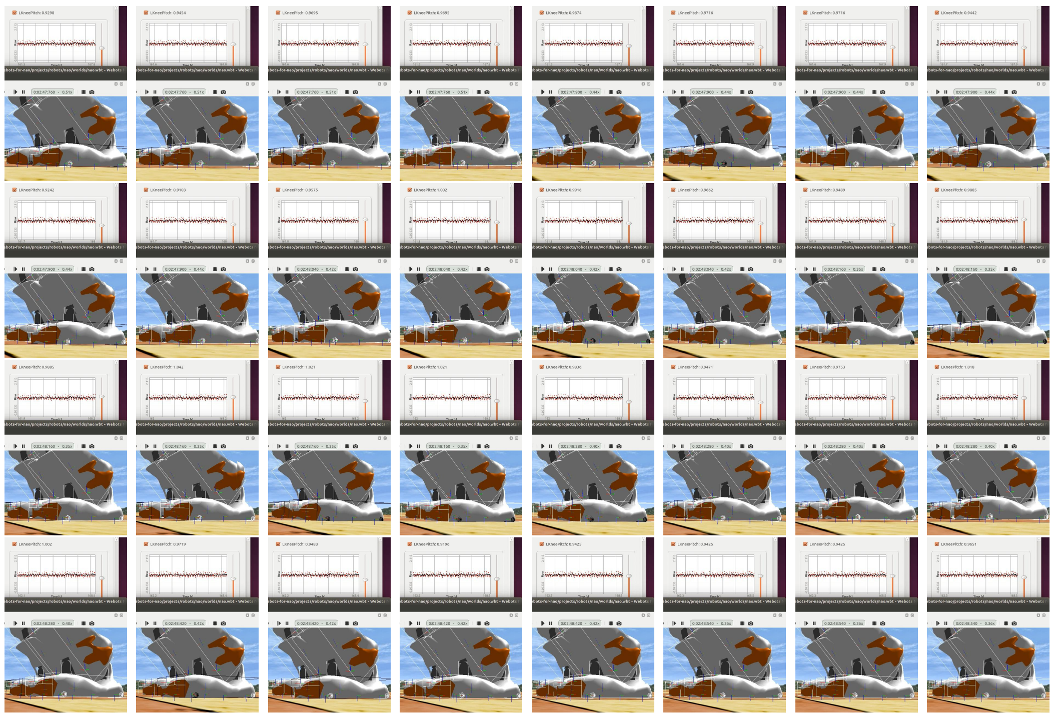


FIGURE 5.2.7: Closer observation of the walking. The evolution of the value of the Left Knee Pitch joint, can be observed in the plots.

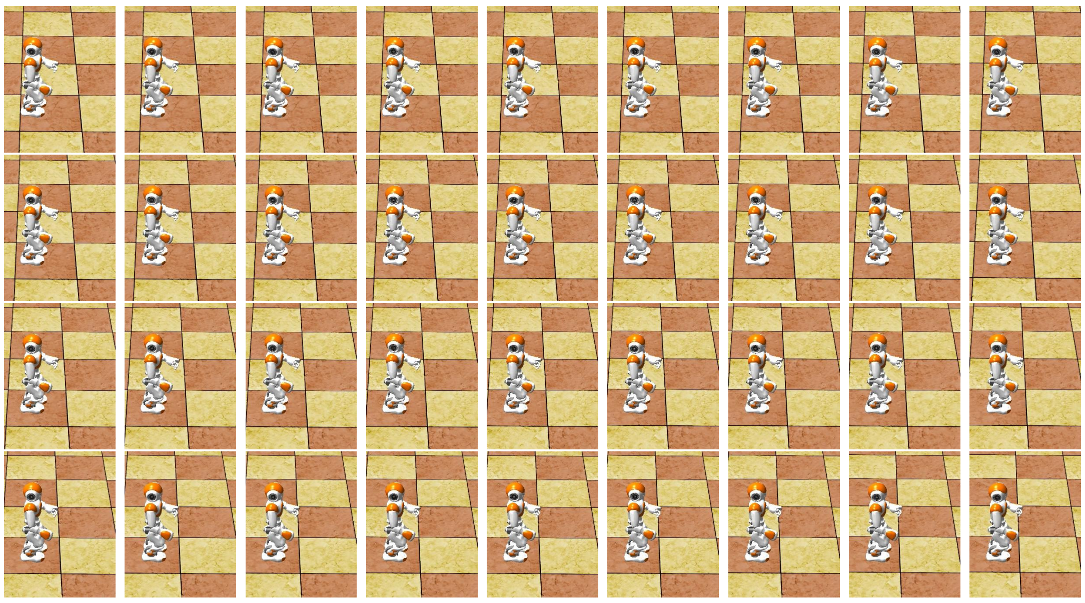


FIGURE 5.2.8: 12 steps of the walking, using the best parameter values based on the optimization process.

Chapter 6

Conclusions and Future Work

In the present thesis, we have investigated how CPGs can be used as parts of the control system for the NAO humanoid robot. The CPGs are neuronal circuits, which can produce rhythmic output patterns, compatible with those found in animal locomotion. As we have demonstrated in Figure 3.1.6, they exhibit significant additional properties, such as the ability to recover from perturbations, adapt quickly to a given signal and integrate sensory information. All these properties render them useful controllers for bipedal locomotion.

The robot control scheme was designed, based on a network of CPGs, each one of which was assigned to a specific joint in the humanoid robot. Each CPG is composed of a chain of adaptive Hopf oscillators, which have the ability to adjust to a specific sub frequency of a learning signal. After learning, the controller is capable of reproducing the learned signal, despite the possible presence of perturbations. However, the CPG model alone cannot compensate for all types of disturbances from the environment. To increase further the adaptability of the control, we have used a genetic algorithm, to optimize the CPG performance using information from the execution of the simulated gait. Genetic algorithms are a search heuristic that mimics the process of natural selections, in order to specify possible optimal solutions to a problem. Our main contribution in this context is the creation and assessment of an objective function capable of evaluating the controller, based on information from the robot's posture. As the results indicate, the genetic algorithm was able to increase the distance traversed by the humanoid.

For the future, we plan to investigate additional objective functions for the genetic algorithm, giving the controller the ability to compensate for disturbances in the postural balance of the robot during gait execution. The distance that the robot travels without falling is not the only rule able to support significant improvement of the gait. A combination of several objective functions, which take advantage of additional sensory information of the robot, e.g. from the Center of Mass, the Center of Pressure or the FSRs, is believed to be a promising direction. For example, we believe that the measurement of the distance of the COP from the edges of the support polygon can be easily combined with the distance rule, yielding better results. Moreover, the CPG network should be tested further regarding its robustness, and this can be done by using forces that will cause disturbances to the robot, to test whether the CPG network will absorb these perturbations.

Appendix A

Webots

Webots is a mobile robot simulation software developed by Cyberbotics a spin-off of EPEL. It provides the user with a rapid prototyping environment for simulating a variety of mobile robots. Many universities and industrial research centers worldwide are using this software for research and educational purposes.

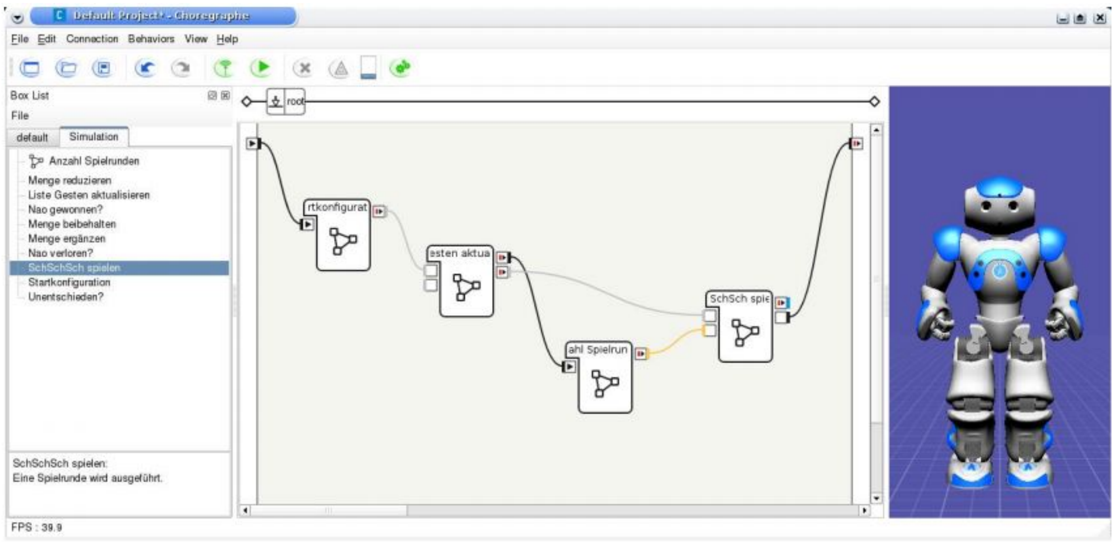
The simulation system uses virtual time, so it gives the opportunity of much faster experiments than when employing the real robot. This also depends on the complexity of the step and the power of the computer. Webots can be connected with integrated development environments (IDEs), like Microsoft Visual Studio and Choregraphe. The IDEs connect with the robot as third party software via the NAOqi operating system. A library of sensors is provided, so the programmer can plug into any of the sensor devices of his robot and tune it individually (range, noise, response, field of view, etc.). A variety of sensor devices are provided, such as distance sensors (infra-red and ultrasonic), range finders, light sensors, touch sensors, global positioning sensor (GPS), inclinometers, compass, cameras (1D, 2D, color, black and white), receivers (radio and infra-red), position sensors for servos and incremental encoders for wheels. Also, an actuator library is provided, including wheeled motor units, independent wheel motors, servos (for legs, arms, etc.), LEDs, emitters (radio and infra-red) and grippers.

A version of Webots called Webots for NAO has been released for the NAO developers, in order to help them develop new ideas and behaviors for the robotic platform without risking to damage their robot. This version of Webots provides the user with predefined NAO simulation with their ready-to-use controllers and sensors. It is a light version of Webots, so it does not allow the user to create either a robot or new robot controllers [36].

Appendix B

Choregraphe

Choregraphe is a visual programming interface implemented by Aldebaran Robotics. It is a multi-platform desktop application, which allows the user to create simulations, animations, behaviors and dialogs in a straightforward way, based on drag-and-drop operations on library without having to write code, as it is already provided with libraries of implemented behaviors. The user can program via a drag-and-drop mechanism, which allows to place modules from the library to the Choregraphe desktop. Also, it allows testing behaviors on a simulated robot, as it can be connected to the Webots simulator. This gives the programmer the freedom to create various scenarios fast and with no fear of damaging the real robot. It provides monitoring and control of the robot, and even the possibility of creating unique behaviors by programming in Python code [2] [9].



SchSchSch spielen:
Eine Spielrunde wird ausgeführt.

Appendix C

NAOqi OS

This is the operating system of the robot. In general, it is a virtual machine governed by the open standards of OVF/OVA. It is a GNU/Linux distribution, based on Gentoo and is embedded in GNU/Linux distribution, which has been specifically developed to fit the Aldebaran Robotics needs. It provides and runs a number of programs and libraries, among these are all the ones required by NAOqi. NAOqi comes with a list of modules, each of which comes with a list of default parameters. The only way to manage the robot is through NAOqi OS, so it is very important for the connection to be stable. The NAOqi OS provides a virtualized NAO environment with no graphical front end, just a console [2].

Appendix D

NAOqi Functions

The implemented walk function for NAO is :

$$\begin{aligned} void\ ALMotionProxy :: moveTo(constfloat\&x, constfloat\&y,\\ constfloat\&theta) \end{aligned} \tag{D.1}$$

The x and y parameters define the distance in meters on the x axis and y axis, while theta states the rotation in radians to the z axis. In the trajectory record of the gait the robot moved only in the x axis, with y = 0, z = 0. [2].

To record joint trajectories the following function was used [2] :

$$void\ AL :: ALValueALMemoryProxy :: getData(conststd :: string\&key) \tag{D.2}$$

The following implemented function is used for reentering the recorded angles to the joints:

$$\begin{aligned} void\ ALMotionProxy :: setAngles(constAL :: ALValue\&names, constAL ::\\ ALValue\&angles, constfloat\&fractionMaxSpeed) \end{aligned} \tag{D.3}$$

The parameter 'names' refers to the joint of the NAO robot, and 'angles' to the new angle of the joint. The 'fractionMaxSpeed' is set at 0.3 for two reasons: firstly, it was a number which was chosen after several tests, as it was found to be the speed of the joint, and also because, at this speed, the joint motion is smooth [2].

References

[1] C. Kemp, P. Fitzpatrick, H. Hirukawa, K. Yokoi, K. Harada, and Y. Matsumoto, "Humanoids," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008, pp. 1307–1333. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30301-5_57

[2] Aldebaran Robotics Nao documentation 2013 NAOqi 1.14.5. [Online]. Available: http://doc.aldebaran.com/1-14/index.html

[3] S. H. Strogatz, "Nonlinear dynamics and chaos," *Perseus Books Publishing L.L.C.*, 1994.

[4] L. Righetti and A. Ijspeert, "Programmable central pattern generators: an application to biped locomotion control," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 1585–1590.

[5] S. Adcock, "Gaul open source programming library designed to aid in the development of applications that use genetic, or evolutionary, algorithms," 2000. [Online]. Available: http://gaul.sourceforge.net/

[6] I. Abuiziah and N. Shakarneh, "A review of genetic algorithm optimization: Operations and applications to water pipeline systems," *International Journal of Mathematical, Computational, Statistical, Natural and Physical Engineering*, vol. 7, no. 12, pp. 1264 – 1270, 2013. [Online]. Available: http://waset.org/Publications?p=84

[7] S. Kajita and B. Espiau, "Legged robots," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008, pp. 361–389. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30301-5_17

[8] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642 – 653, 2008, robotics and Neuroscience. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608008000804

[9] A. Weirich, C. Haumann, S. Schüler, and J. J. Steil, "Learning lab - programming interaction with humanoid robots for pupils," in *Int. Conf. Robotics in Education*, 09/2011 2011, pp. 21–28.

[10] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, "Feedback control of dynamic bipedal robot locomotion." CRC PRESS, 2007.

[11] M. Vukobratovic and B. Borovac, "Zero-moment point — thirty five years of its life," *International Journal of Humanoid Robotics*, vol. 01, no. 01, pp. 157–173, 2004. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0219843604000083

[12] C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schaffer, B. Brunner, H. Hirschmuller, S. Kielhofer, R. Konietschke, M. Suppa, T. Wimbock, F. Zacharias, and G. Hirzinger, "A humanoid two-arm system for dexterous manipulation," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, Dec 2006, pp. 276–283.

[13] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The development of Honda humanoid robot," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2, May 1998, pp. 1321–1326 vol.2.

[14] E. Marder and D. Bucher, "Central pattern generators and the control of rhythmic movements," *Current Biology*, vol. 11, no. 23, pp. R986 – R996, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960982201005814

[15] P. Stein, S. Grillner, A. I. Selverston, and D. G. Stuart, *Neurons, networks and motor behavior*, 1997.

[16] M. Sfakiotakis and D. P. Tsakiris, "Neuromuscular control of reactive behaviors for undulatory robots," *Neurocomputing*, vol. 70, no. 10–12, pp. 1907 – 1913, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092523120600436X

[17] M. Sfakiotakis, D. P. Tsakiris, and A. Vlaikidis, "Biomimetic centering for undulatory robots," in *The First IEEE RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.*, Feb 2006, pp. 744–749.

[18] M. Sfakiotakis and D. P. Tsakiris, "Biomimetic centering for undulatory robots," in *International Journal of Robotics Research.* Sage Publications Ltd., Nov/Dec 2007, pp. 1267–1282.

[19] L. Righetti, J. Buchli, and A. J. Ijspeert, "Dynamic hebbian learning in adaptive frequency oscillators," *Physica D: Nonlinear Phenomena*, vol. 216, no. 2, pp. 269 –

281, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167278906000819

[20] P. Arena, "The central pattern generator: a paradigm for artificial locomotion," *Soft Computing*, vol. 4, no. 4, pp. 251–266, 2000. [Online]. Available: http://dx.doi.org/10.1007/s005000000051

[21] M. Okada, K. Tatani, and Y. Nakamura, "Polynomial design of the nonlinear dynamics for the brain-like information processing of whole body motion," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 2, 2002, pp. 1410–1415 vol.2.

[22] M. H. Dickinson, C. T. Farley, R. J. Full, M. A. R. Koehl, R. Kram, and S. Lehman, "How animals move: An integrative view," *Science*, vol. 288, no. 5463, pp. 100–106, 2000. [Online]. Available: http://www.sciencemag.org/content/288/5463/100.abstract

[23] Y. Fukuoka, H. Kimura, Y. Hada, and K. Takase, "Adaptive dynamic walking of a quadruped robot 'tekken' on irregular terrain using a neural system model," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 2, Sept 2003, pp. 2037–2042 vol.2.

[24] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen, "From swimming to walking with a salamander robot driven by a spinal cord model," *Science*, vol. 315, no. 5817, pp. 1416–1420, 2007. [Online]. Available: http://www.sciencemag.org/content/315/5817/1416.abstract

[25] A. Kamimura, H. Kurokawa, E. Toshida, K. Tomita, S. Murata, and S. Kokaji, "Automatic locomotion pattern generation for modular robots," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 1, Sept 2003, pp. 714–720 vol.1.

[26] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *The International Journal of Robotics Research*, vol. 20, no. 2, pp. 129–143, 2001. [Online]. Available: http://ijr.sagepub.com/content/20/2/129.abstract

[27] J. A. Acebrón, L. L. Bonilla, C. J. Pérez Vicente, F. Ritort, and R. Spigler, "The kuramoto model: A simple paradigm for synchronization phenomena," *Rev. Mod. Phys.*, vol. 77, pp. 137–185, Apr 2005. [Online]. Available: http://link.aps.org/doi/10.1103/RevModPhys.77.137

[28] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[29] J. Holland, *Adaptation in Natural and Artificial Systems*, 1975.

[30] M.-Y. Cheng and C.-S. Lin, "Genetic algorithm for control design of biped locomotion," in *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on*, vol. 2, Oct 1995, pp. 1315–1320 vol.2.

[31] E. Torres and L. Garrido, "Automated generation of cpg-based locomotion for robot nao," in *RoboCup 2011: Robot Soccer World Cup XV*, ser. Lecture Notes in Computer Science, T. Röfer, N. Mayer, J. Savage, and U. Saranlı, Eds. Springer Berlin Heidelberg, 2012, vol. 7416, pp. 461–471. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32060-6_39

[32] M. Oliveira, L. Costa, A. Rocha, C. Santos, and M. Ferreira, "Multiobjective optimization of a quadruped robot locomotion using a genetic algorithm," in *Soft Computing in Industrial Applications*, ser. Advances in Intelligent and Soft Computing, A. Gaspar-Cunha, R. Takahashi, G. Schaefer, and L. Costa, Eds. Springer Berlin Heidelberg, 2011, vol. 96, pp. 427–436. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20505-7_38

[33] "Mathworks documentation, interpolation function interp." [Online]. Available: http://www.mathworks.com/help/signal/ref/interp.html

[34] "Mathworks documentation, genetic algorithm." [Online]. Available: http://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

[35] C. Darwin, "On the origin of species," p. 470, 1859. [Online]. Available: http://www.biodiversitylibrary.org/item/71804

[36] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004.