Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Optimization of Stereotypical Trotting Gait on HyQ

Master Thesis

March 30, 2015

Supervised by

Farbod Farshidian
Prof. Dr. Jonas Buchli

Author

Brahayam David Pontón Junes

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

Verfasst von (in Druckschrift):
Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**

Ich bestätige mit meiner Unterschrift:
- Ich habe keine im Merkblatt „Zitier-Knigge“ beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.

Abstract

Over the last decades, locomotion of legged robots has become a very active field of research, because of the versatility that such robots would offer in many applications. With very few exceptions, in general, legged robot experiments are performed in controlled lab environments. One of the reasons of this limited use is that in real world environments, legged robots have to interact with an unknown environment, and in order to do it successfully and safely, they need to be compliant, such as humans and animals are. In the context of this Thesis, a framework to optimize a stereotypical trotting gait for the Hydraulic Quadruped robot HyQ using variable impedance is proposed. This is an important step towards closing the gap between robot capabilities and nature's approach for animal locomotion.

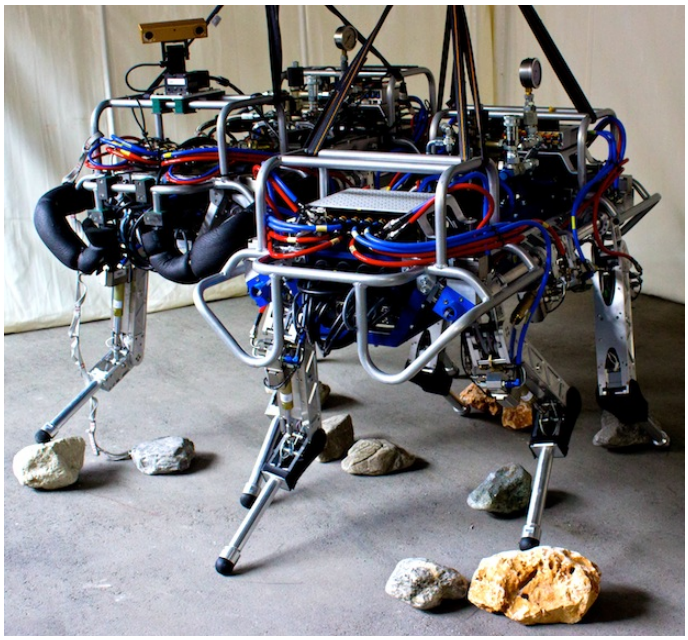


Figure 0.1: Hydraulically powered Quadruped robot *HyQ*. Picture from IIT.

The proposed framework makes use of the reinforcement learning algorithm PI^2 (Policy Improvement with Path Integrals) to optimize the parameters of a CPG-based gait generator and the robot impedance during locomotion.
The proposed learning method is evaluated in a series of experiments on a simulation of HyQ, where it achieves an energetically efficient and robust trotting gait at different speeds while handling joint and torque limits.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Prof. Jonas Buchli. His ideas and vision were inspiring, his trust encouraging and motivating, and his guidance and expert support invaluable. It has been an honour and a pleasure to work with you.

I would like to extend my gratitude to my co-supervisor Farbod Farshidian for his strong support, by sharing his deep knowledge, thoughtful comments and unconventional ideas.

I am greatly indebted and grateful to Christian Gehring for letting me use his Optimization Framework with Reinforcement Learning Algorithms.

Many thanks also to Thiago Boaventura, Michael Neunert, Peter Fankhauser, for sharing with me their great expertise and helping me when needed.

And finally, I thank my parents and my little sister for being always there for me, for their support and encouragement.

Brahayam Ponton

Zürich, March 30, 2015

Dedication

La presentación de la presente tesis es un paso más en mi búsqueda de crecimiento académico y en la creación paulatina de la persona que siempre anhele ser desde mi infancia. Por eso, quiero aprovechar su finalización para agradecer a las personas que me han ayudado a llegar aquí.

Quiero agradecer a mi padre y a mi madre que siempre han sido mi ejemplo a seguir y una de mis mayores fuentes de inspiración, ya que sus historias de éxito me han ensenado que siempre se puede salir adelante y que la realización de nuestras metas no es más que el producto de la dedicación, el esfuerzo y la pasión por lo que uno hace.

También quiero agradecer a mi hermana porque desde que llego a mi vida ha sido una fuente inagotable de alegría; sus ocurrencias y su sola presencia han iluminado mis días más tristes.

Quiero expresarle a mi familia mi más profundo agradecimiento y mi cariño porque tras vivir físicamente lejos por un largo periodo de tiempo, he confirmado que nuestro vínculo familiar va más allá de convivir los unos con los otros. Nuestras vidas están basadas en el apoyo mutuo, el amor incondicional y el sincero deseo de ver que el otro alcance el éxito. Mi conexión con mi familia es mi mayor fortaleza y es lo que me ha impulsado a perseguir mis sueños. Fue el apoyo de mi familia el que me trajo al Instituto Federal Suizo de Tecnología de Zurich en primer lugar, ya que ellos a fin de verme realizado como ser humano y profesional, me alentaron a volar lejos del nido en busca de una educación de excelencia, a pesar de que eso significaba dejar de compartir los almuerzos diarios y celebrar juntos los cumpleaños.

También me gustaría agradecer a la ETH, mis profesores y mis compañeros porque me recibieron con los brazos abiertos, me dieron un nuevo lugar al que llamar hogar y me ayudaron a seguir alimentando mi hambre de conocimiento.

Finalmente, quisiera agradecer a Dios por velar cada uno de mis pasos, por traerme aquí y por haberme bendecido con una familia tan maravillosa.

Dedication

The presentation of this work is one step more in my search for academic growth and in the gradual creation of the person I have yearned to be since my childhood. This is the reason for which I want to use this space for expressing my gratitude to the people that have helped me to get here.

I want to thank my father and my mother that have been my role model and one of my greatest sources of inspiration. My parent's stories of success have taught me that it is always possible to succeed and that the realization of our goals is not more than the result of dedication, effort and the passion professed to one's activities.

I also want to thank my little sister because, since her arrival to my life, she has been an endless source of happiness. Her witticism and presence have enlightened my worst days.

I would like to express my deepest thankfulness and love to my family because after living physically far from each other, I have confirmed that our family bond goes beyond the mere fact of living together. Our lives are based on mutual support, unconditional love and the sincere wish for seeing each other succeeding in life. My bond with my family is my most important strength and it is what has driven me to pursue my dreams. It was my family's support that brought me to the Swiss Federal Institute of Technology in Zurich in the first place. My parents and my sister encouraged me to fly away from the nest in the search for a first class education in order to see me fulfilling my professional aspirations despite of the implications that this decision entailed such as stop sharing the daily meals and celebrating each other's birthday.

I also would like to thank ETH, its faculty and its students because they received me with open arms, gave me a new place to call home and advance my knowledge.

Finally, I would like to thank God for watching over each one of my actions, for bringing me here and for having blessed me with my marvelous family.

Contents

List of Figures

List of Tables

1 Introduction

"My heart is on the work."
— Andrew Carnegie, *Scottish - American industrialist and philanthropist, 1835 - 1919*

This introductory chapter presents the motivation and goals behind this project and, in general legged locomotion research. It also gives a brief overview of previous work and state of the art in legged robotics and outlines the structure of the report.

1.1 Motivation and Objectives

Why is locomotion control an important and interesting problem? In addition to all the fun inherent in working with robots, locomotion control of legged robots presents an exciting problem, because of its challenges and still unresolved issues. Legged robots need good coordination skills for many degrees of freedom, have to deal with uncertainties in the model, the environment and instantaneous changes in the contact situation, need adaptation capabilities for different terrains and environment conditions, need to handle sensory input, redundancies, under-actuation, real-time control and conflicts among several tasks or priorities. Despite of that, progress is achieved everyday, because legged robots are a promising technology for many applications.

Some of them include its use in unstructured and unknown environments. For example in exploration, rescue missions, radioactive places, among others. Also important in locomotion research, is the understanding of biological principles, helpful for the design of devices for rehabilitation and active prostheses to compensate motor deficits. The capabilities of legged animals, in terms of dexterity and versatility outperform any robot, and become, therefore, a source of ideas and inspiration for the robotics community. Biological principles and ideas can be applied in the design of new and better control strategies for legged robots.

The goal of this project is to apply principles from nature like variable impedance control and the use of adaptive frequency oscillators for synchronization, in order to implement a learning and adaptation layer over a parametrized gait generator for trotting. The learning layer is expected to optimize the trotting gait, in terms of energy efficiency, robustness and speed tracking.

(a) Single leg hopper (b) BigDog (c) WildCat

Figure 1.1: Robot prototypes - Mark Raibert

1.2 Previous Work on Legged Locomotion

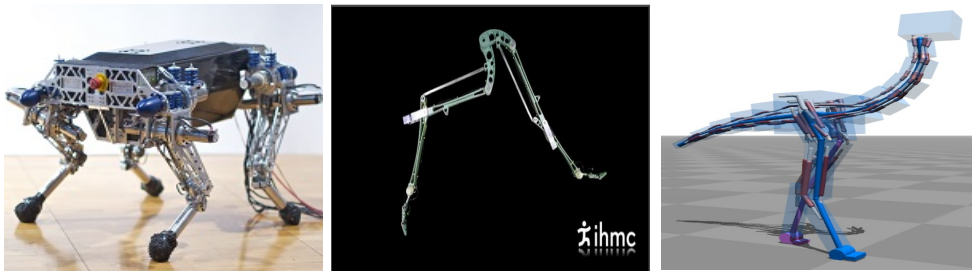
Although a vast amount of literature exists about learning and control in legged locomotion of static and dynamic gaits, this section will present only some of the most relevant and interesting examples of legged robots.

Mark Raibert and his collaborators have, without any doubt, strongly influenced the development and research in dynamic legged locomotion. He initiated his work in the 1980's with experiments on single leg hoppers [51], and then on biped and quadruped robots with pneumatic actuation. Since then, the prototypes have been improved, being able to achieve impressive performance with BigDog [49] and WildCat [52]. The drawback is that, apart from videos, no information about control strategies and designs have been published, so that the results cannot be validated by other groups.

Control strategies and hardware designs have evolved in the last years, from high gain position control and robots with stiff actuators that try to follow precisely a preplanned trajectory, to interaction / force / impedance control with compliant robots, able to perform robustly more dynamic manoeuvres. Such a very good example is StarlETH [24], which was built with inherent compliance by using series elastic actuators, uses model-based control and a hierarchical optimization framework to handle priorities and several tasks. StarlETH has shown several gaits like walking and trotting.

The design principles to embed intelligence in the robot have also evolved. The basic idea of homeostasis or equilibrium inspired the well known feedback control; the development of computational power and parallel processing allowed the use of search and planning algorithms and, hierarchical optimization and control schemes. Now, the use of mechanical intelligence, opens a new field of research, an example of it, can be seen in the design of "Fast Runner a robot Ostrich" [48], characterized by its innovative and self-stabilizing leg design.

Regarding gait optimization, there are two general ways to approach the problem: direct and shooting methods. In shooting or learning methods, the optimization is performed based on a finite parametrization of the control input variables. The cost of each policy parametrization is evaluated by forward simulating the dynamics of the system



(a) StarlETH [24] (b) FastRunner Ostrich [46] (c) Bipedal creature [18]

Figure 1.2: State of the art legged robots

from an initial condition using the current policy parameters. The parameters update is done based on the performance of the different rollouts. A state of the art example of gait optimization through shooting methods can be seen in [18]. In this work, the authors define the basic structure of bipedal creatures, they use muscle models including neural delays to generate locomotion torques and forces, and optimize simultaneously the muscle properties and control parameters and, also the muscle routing geometry in these muscle-based bipedal creatures by using the Covariance Matrix Adaptation - CMA Algorithm. The results they achieve are very impressive, because the synthesized controllers generate natural looking motion, are able to withstand external perturbations and uneven terrain up to a certain extent, and allow speed and steering control.

The other possible approach for gait optimization is the use of direct methods. In direct methods, the optimization algorithm searches simultaneously control and state trajectories, and imposes the dynamics as a set of optimization constraints. This approach does not require simulation. A very nice and principled example of this approach is presented in [38, 46]. This work introduces several interesting features like fully autonomous optimization of contact transitions, because it does not restrict the search to fixed orderings of the hybrid transition modes; it makes use of time discretization, that takes into account only the integrals of contact forces over a period; and defines the problem as a general optimization of a cost function over the control and state trajectories, the contact forces and the length of the timesteps, subject to constraints imposed by the dynamics of the rigid bodies, the inelastic impacts and friction forces. This work finds a locally optimal solution for the problem using a sparse sequential quadratic programming solver. Impressive results are shown for several tasks, including gait optimization of the FastRunner ostrich robot.

Learning methods have been applied for learning locomotion. They have shown successful results, where they outperform any previous hand-tuned controller. Several examples of this can be found in the literature, such as the following ones. In [31, 32], the authors present a policy gradient reinforcement learning algorithm that optimizes a quadrupedal trotting gait on the Sony Aibo robot. It optimizes the parameters of a parametrized gait for achieving maximum possible forward speed. The result of this machine learn-

ing optimization approach is that the robot achieves a locomotion gait faster than any previously hand-coded or learned gait on Aibo.
In [65], a learning algorithm for bipedal walking is presented. It makes use of actor-critic methods and temporal difference learning TD(0) to online optimize the control policy for bipedal walking on the real robot. It represents the closed loop dynamics of the robot integrated from one footstep to the next by means of a return map. With this learning structure, the robot is able to perform learning and execution simultaneously. It learns a walking gait in less than 20 minutes and can continuously adapt its control policy to different terrains at every step by minimizing the eigenvalues of the return map.
In [12], reinforcement learning for locomotion of a single legged robot is presented. The goal in this project was to improve the performance in highly dynamic tasks, such as jumping and hopping , in terms of maximizing jump height, jump distance and energy efficiency in periodic motion. It makes use of the reinforcement learning algorithm Policy Improvements with Path Integrals (PI^2), in a model-free approach, to optimize a parametrized control policy of the joint velocities and the parameters of a virtual model controller for periodic hopping. Learning is performed in a combination of simulation and hardware experiments, being able to push, in this way, the robot capabilities to its limits.

1.3 Thesis outline

This introduction is followed by chapter 2, that presents the theoretical background underlying the system modelling, control and learning techniques used in this Thesis.
Chapter 3 presents the details of the implementation, how the concepts seen in chapter 2 fit together, from adaptive frequency oscillators and variable impedance control to cost function design and reinforcement learning for gait optimization.
Chapter 4 presents the experiments performed and the results obtained in simulation.
Finally, Chapter 5 presents final conclusions and future work possibilities.

2 Background Theory

"An approximate solution to the right problem is far better than an exact answer to an approximate problem."
— John Wilder Tukey, *American statistician, 1915 -2000*

2.1 Description of HyQ

HyQ is a hydraulically-powered quadruped robot, which has been developed at the Istituto Italiano di Tecnologia (IIT), in Genoa, as a platform to study legged locomotion in highly dynamic motions such as running and jumping, as well as careful navigation over rough terrain. This section summarizes the more important characteristics of this quadruped robot and is based on the work presented in [60] and [61].

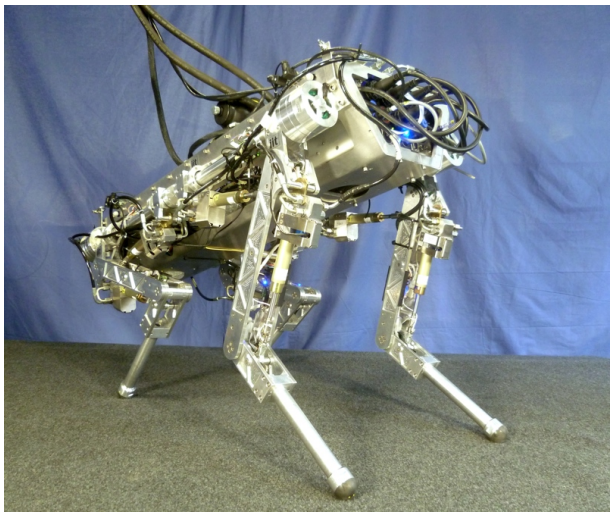


Figure 2.1: Hydraulically powered Quadruped robot *HyQ*. Taken from [19]

HyQ is a 1 meter tall robot, composed of 12 torque-controlled joints that use a hydraulic actuation system. It weights approximately 70 kg when externally powered and 90 kg with on-board hydraulic power supply.
HyQ has been designed keeping in mind the following goals:

- To design a quadruped robot, mechanically able to perform highly dynamic motions, keep balance and be able to navigate autonomously, especially in difficult terrains. To test different actuation mechanisms to improve energy autonomy and efficiency.

- To serve as an open platform to study control in legged locomotion with special focus on dynamic gaits, to test and analyse different control algorithms for gait generation and transition.

The designers started by taking inspiration from nature and using knowledge from existing robots. Some important examples of existing quadruped robots are SCOUT [47], KOLT [43], StarlETH [24], and the impressive BigDog [50] and WildCat, from which there is very few information available, but motivates research and proved that technologically such a project is feasible. On the other hand, animals have evolved and reached a point of exceptional agility, from where some ideas can be drawn.
In nature, a wide variety of quadrupedal locomotion gaits can be seen, such as trotting, bounding, galloping, among others. A study of locomotion in horses, conducted by Hoyt and Taylor [23], has shown that these animals choose the gait and speed that is energetically optimal and minimize the risk of injuries due to excessive musculoskeletal forces at foot touch-down. Among these gaits, trot is energetically efficient over a wide range of velocities [44]. For this reason a trotting gait can be chosen as a good starting point for dynamic locomotion in HyQ.

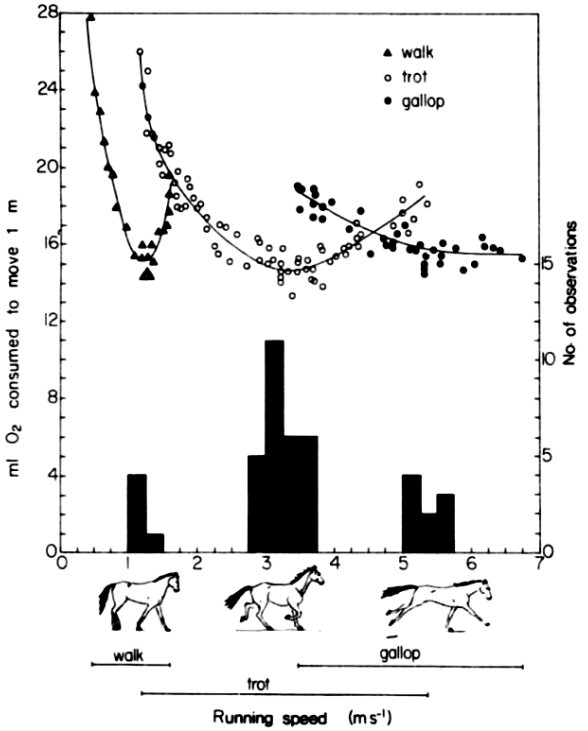


Figure 2.2: Oxygen consum per unit distance vs. walking or running speed at the given gait. Taken from [23].

In [21], a biology experimental study over a large range of quadruped animals, the authors concluded that there is a relation between trotting speeds and stride frequencies with animal's body mass. The results of this study are reproduced in Table 2.1.

Quadruped's Mass	50kg	60kg	70kg	80kg	90kg
Minimum trotting gait	1.57m/s	1.64m/s	1.71m/s	1.77m/s	1.82m/s
	1.70Hz	1.67Hz	1.64Hz	1.62Hz	1.61Hz
Preferred trotting gait	2.6m/s	2.7m/s	2.8m/s	2.88m/s	2.96m/s
	2.01Hz	1.97Hz	1.93Hz	1.9Hz	1.87Hz
Maximum trotting gait	3.56m/s	3.73m/s	3.86m/s	3.97m/s	4.07m/s
	2.33Hz	2.27Hz	2.26Hz	2.17Hz	2.13Hz

Table 2.1: Relation between trotting speeds and stride frequencies with animal's body mass.

These results define a base for stating the basic design specifications of HyQ. These performance targets are the ability to walk in flat and rough terrain, locomote with walking and flying trot up to a speed of 3 m/s, maintain stability, execute a vertical jump with a safe landing, and power autonomy for several hours.

2.1.1 HyQ Leg Design

Each leg in HyQ is composed of three active revolute degrees of freedom (DOF), as shown in Figure 2.3. There are two joints in the sagital plane, named Hip Flexion-Extension (HFE) and Knee Flexion-Extension (KFE). A third joint, named Hip Abduction-Adduction (HAA) is responsible for lateral leg motion. They allow foot positioning in the 3D workspace. A limitation of this configuration is that it does not allow to simultaneously choose the contact angle and the foot location. The contact angle is important, because it determines the direction of the experimented force.

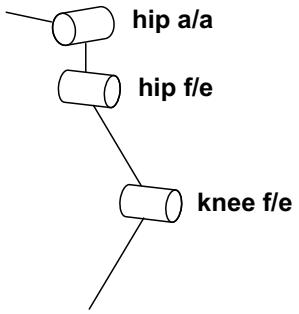


Figure 2.3: Kinematic structure of the active joints in HyQ's leg. Taken from [60].

There is also an additional passive prismatic joint, located along the axis of the lower leg segment, that connects it to the foot with a spring, this is called the ankle joint. This joint adds passive compliance at the foot, which is important to cope with initial impacts due to force peaks at foot touch-down during locomotion. A 5mm layer of visco-elastic rubber covers the foot. Its purpose it to provide additional compliance and to improve traction by increasing friction.

The desired total leg compliance is obtained as a combination of active and passive components, and can be controlled by varying the leg stiffness by using the active torque-controlled joints. HyQ belongs to the family of robots with articulated legs, which is inpired in the kinematic structure of cursorial mammal types (like a horse), which show in nature impressive stability during dynamic locomotion.

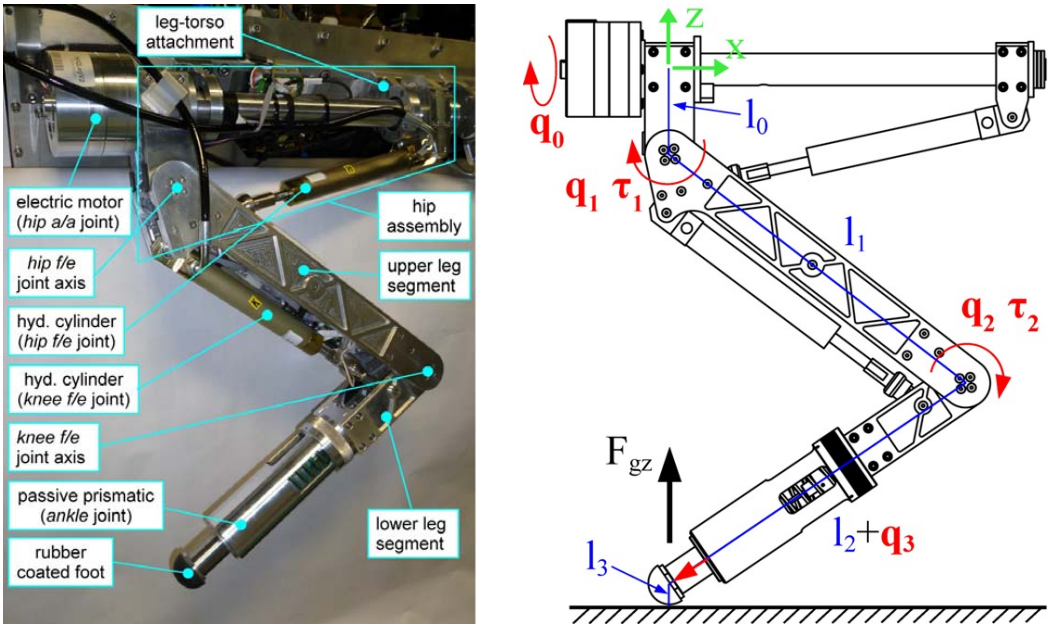


Figure 2.4: Description of HyQ's leg main components. Pictures taken from [61]. Left: Leg of HyQ and the names of its different components. Right: Schematic of HyQ's leg. The Hip Abduction-Adduction (HAA) joint is represented by q_0, the Hip Flexion-Extension (HFE) joint by q_1, and the Knee Flexion-Extension (KFE) joint by q_2. The prismatic passive Ankle joint is represented by q_3.

All of the joints, HAA, HFE and KFE, are actuated with hydraulic cylinders, which feature a good dynamic range, high bandwidth, excellent power-to-weight ratio, and robustness to torque peaks due to the intrinsic compliance. The hydraulic cylinders act directly between two leg segments. In order to select the actuator specifications, torque estimations for a vertical jump from a crouched position were performed. Details on this can be found in [60, 61]. Table 2.2 summarizes the specifications for the actuators.

Specification	Value
Cylinder bore and rod diameter	16mm, 10 mm
Cylinder piston and annulus area (A_p, A_{pr})	2.01 cm^2, 1.23 cm^2
Cylinder stroke	80mm
Max. operating pressure	16MPa

Table 2.2: Specifications of hydraulic actuators

Each active joint counts with three different sensors:

- Relative optical encoder: This high resolution sensor has 80000 counts per revolution, which allows direct low noise estimation of joint position and velocity.
- Absolute magnetic encoder: Used for easy and automated joint initialization and as a sensor redundancy safety measure.
- Force/Torque sensor: Used for force/torque feedback control of the joints. For the hydraulic joints a strain-gauge based load cell mounted between the cylinder rod and the rod end allows to measure the cylinder output force. The load cell range goes up to 5 kN. The passive ankle joint counts with a linear potentiometer that measures spring compression and is used to determine ground reaction forces, based on Hook's law. Its maximum range is 0.035m.

2.1.2 HyQ Leg - Mechanical Considerations

The main criteria for the mechanical design are to keep the leg robust, with low inertia and modular, so that it can easily be installed in the torso. For achieving these goals, strength but light materials were used. For example, Ergal (a strong aluminium alloy) is used for the torso. Stainless steel is used for heavily stressed parts, such as joint end-stops, connection between motor and leg, cylinder attachments, among others. A detailed summary of the leg's mass and inertia properties can be found in [60].
In [30], Jaegger studied a group of Labrador Retriever dogs and estimated statistics of different joint angles, like minimum and maximum joint ranges. Although animals have a complex kinematic structure, like additional degrees of freedom, the maximum and minimum joint ranges offer a rough idea of what would be a good criteria to select the joint ranges in HyQ, where all the ranges of revolute degrees of freedom were chosen to be 120 degrees.

Location	Parameter	value
Leg	l_0	0.08m
	l_1	0.35m
	l_2	0.35m
	l_3	0.02m
Hip a/a	q_0	range [-90° to +30°]
Hip f/e	q_1	range [-70° to +50°]
Knee f/e	q_2	range [20° to 140°]
ankle (passive)	q_3	range [-0.035m to 0m]

Table 2.3: Geometric parameters of HyQ robot leg

The segment lengths were chosen based mainly on two criteria: The commercial hydraulic actuators should provide the required force and fit into the leg, and the di-

mensions in HyQ, ratio between front and hind Hip joints to fully stretched leg should approximate 1, as suggested in [34], according to which it is a sign of a fast racing dog. Table 2.3 summarizes lengths and joint ranges of the different leg segments in HyQ.

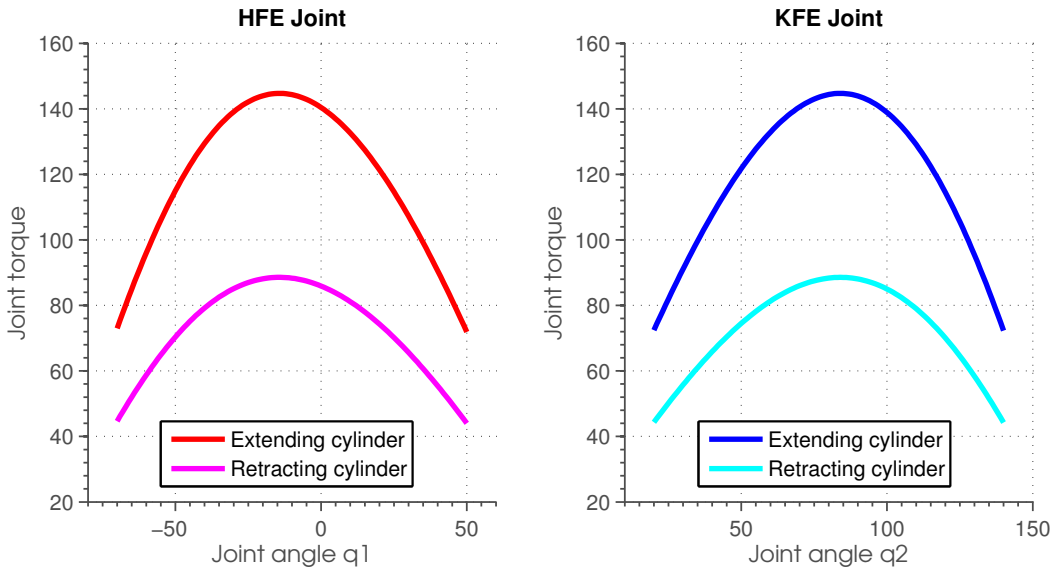


Figure 2.5: Maximum torque profiles in HyQ for the hydraulically powered revolute joints. Based on [61].

Figure 2.5 shows the nonlinear relation between torque profile and joint angle for the HFE and KFE joints with hydraulic actuators. A simple derivation of the analytical relation between torques and joint angles for the kinematic structure of HyQ can be found in [61].

2.1.3 Hydraulically powered Quadruped Robot HyQ

HyQ is composed of a robot torso and four identical legs, attached to the torso in the forward/backward configuration, in which front and hind knees point to each other, as forming an "x".

Several groups have conducted research to determine which configuration works better for quadruped locomotion between the different combinations of forward and backward leg positioning. In [70] for example, it was found that the forward/backward configuration improves performance by decreasing slippage of the feet.

The torso has a trapezoidal-shaped cross section and was built of an Ergal sheet of 3mm (determined based on finite element model analysis). The total weight of HyQ's legs is 24kg, which corresponds to roughly 26% of the total mass. In nature, animals of similar weight have a ratio of leg mass to body weight between 19 and 26%, which sets HyQ in the upper limit. Table 2.4 summarizes the main characteristic of HyQ.

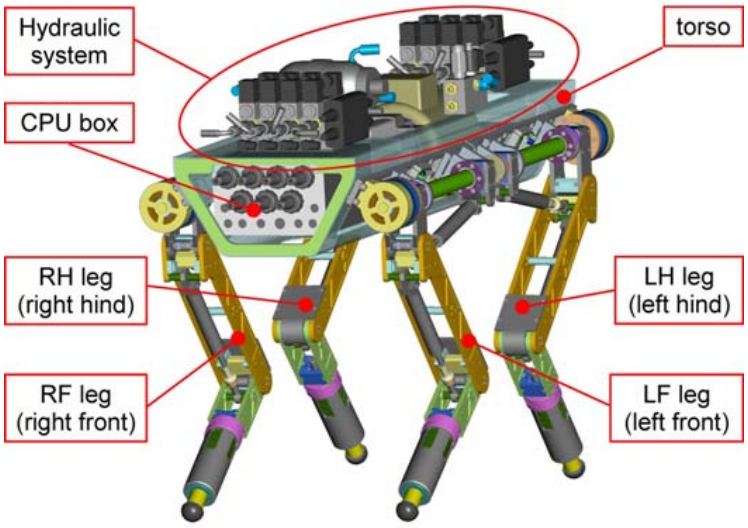


Figure 2.6: CAD model of HyQ explaining its main components. Picture taken from [61]

Description	Value
Dimensions	1m x 0.5m x 0.98m (Length x Width x Height)
Leg length (hip a/a axis to ground)	from 0.339m to 0.789m
Distance of left to right hip a/a axis	0.414m
Distance of front to hind hip f/e axis	0.747m
Weight	70kg (external hydraulic system) 91kg (onboard hydraulic system)
Number of active DOF	12 hydraulic
Joint range motion	120° (for each joint)
Hydraulic actuator type	double-acting cylinders (80mm stroke and 16mm bore)
Maximum torque (hydraulic)	145Nm (peak torque at Pmax = 16MPa)
Onboard sensors	Joint position (relative and absolute), joint torque, cylinder pressure, foot spring compression, IMU.
Onboard computer	PC104 Pentium, real-time Linux
Control frequency	800Hz

Table 2.4: Some general specifications of HyQ

2.2 System model

HyQ and in general legged robots are not rigidly attached to an environment like a robotic arm, but they can move freely in space and have to deal with changing contact conditions. Indeed, they use ground as support for locomotion.

System Modelling

$$\mathbf{q} = \begin{pmatrix} q_b \\ q_r \end{pmatrix} \tag{2.1a}$$

$$M(q)\ddot{q} + h(q, \dot{q}) + J_s^T F_s = S^T \tau \tag{2.1b}$$

$$\dot{r}_s = J_s \dot{q} = 0 \tag{2.1c}$$

$$\ddot{r}_s = J_s \ddot{q} + \dot{J}_s \dot{q} = 0 \tag{2.1d}$$

$$F_s = \left(J_s M^{-1} J_s^T\right)^{-1} \left(J_s M^{-1} \left(S^T \tau - h\right) + \dot{J}_s \dot{q}\right) \tag{2.1e}$$

They can be modelled using the concept of generalized coordinates. It describes the kinematics and dynamics of a floating base system as a n_q dimensional vector q composed of n_b free floating base coordinates q_b and $n_r = n_q - n_b$ actuated joints coordinates q_r, as shown in equation 2.1a. The fixed body frame B represents the floating base and can move arbitrarily with respect to the inertial frame I, as shown in Figure 2.7.

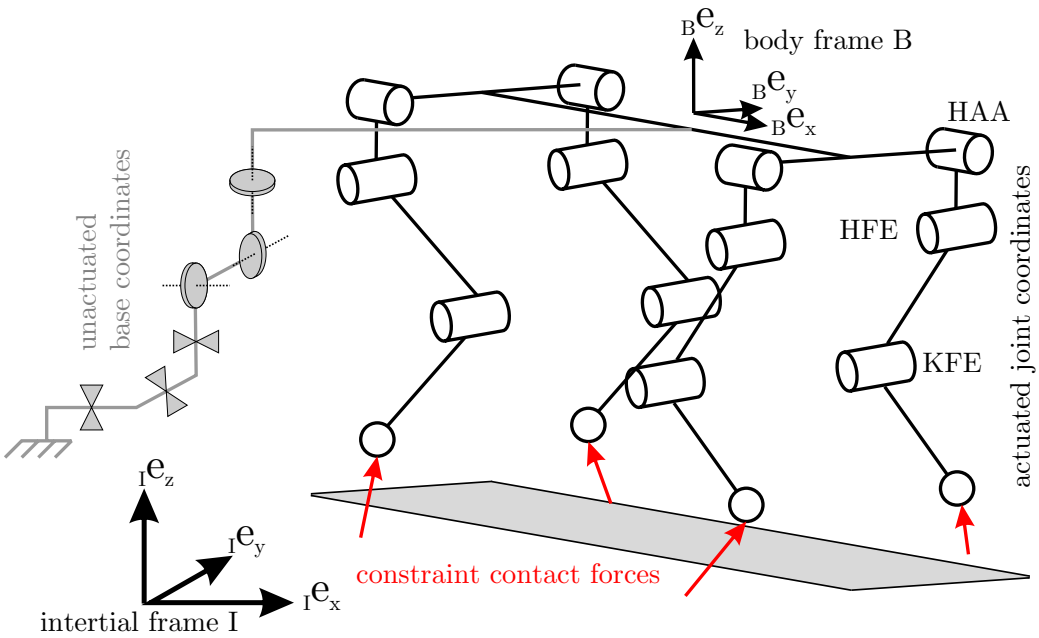


Figure 2.7: Kinematic Structure of the Quadruped robot HyQ. It does not show the passive ankle joints. Taken from [24].

The system dynamics are given by equation 2.1b as a function of the generalized coordinates. $M(q)$ represents the mass matrix, the vector $h(q, \dot{q})$ represents the sum of the Coriolis, centrifugal and gravitational forces. The matrix $S = [0_{n_r \times n_b}\ I_{n_r}]$ is called the selection matrix and separates actuated from not actuated coordinates. τ is the vector of generalized forces.

In order to define F_s and J_s, we first define n_s as the number of active contact points. *"The definition of active contact is that the corresponding contact is closed, which means that the relative normal (N) distance between the contact point and the environment is and remains closed ($r^N_{s_i} = \dot{r}^N_{s_i} = \ddot{r}^N_{s_i} = 0$) with a pressure force exerted between them ($F^N_{s_i} \geq 0$)."* [24].

Then, r_s and F_s are the vectors of stacked vectors of position $r_{s_i} \in \mathbb{R}^{3\times1}$ and force $F_{s_i} \in \mathbb{R}^{3\times1}$ at the n_s active contact points. Therefore, $F_s \in \mathbb{R}^{3n_s\times1}$ and $J_s = \frac{\partial r_s}{\partial q} \in \mathbb{R}^{3n_s\times n_q}$.

Equations 2.1c and 2.1d describe the model of a hard contact, which neglects contact slippage. By using this contact model and the equation of motion, a closed formed solution of the contact forces can be derived, as shown in equation 2.1e.

2.3 Locomotion Gaits

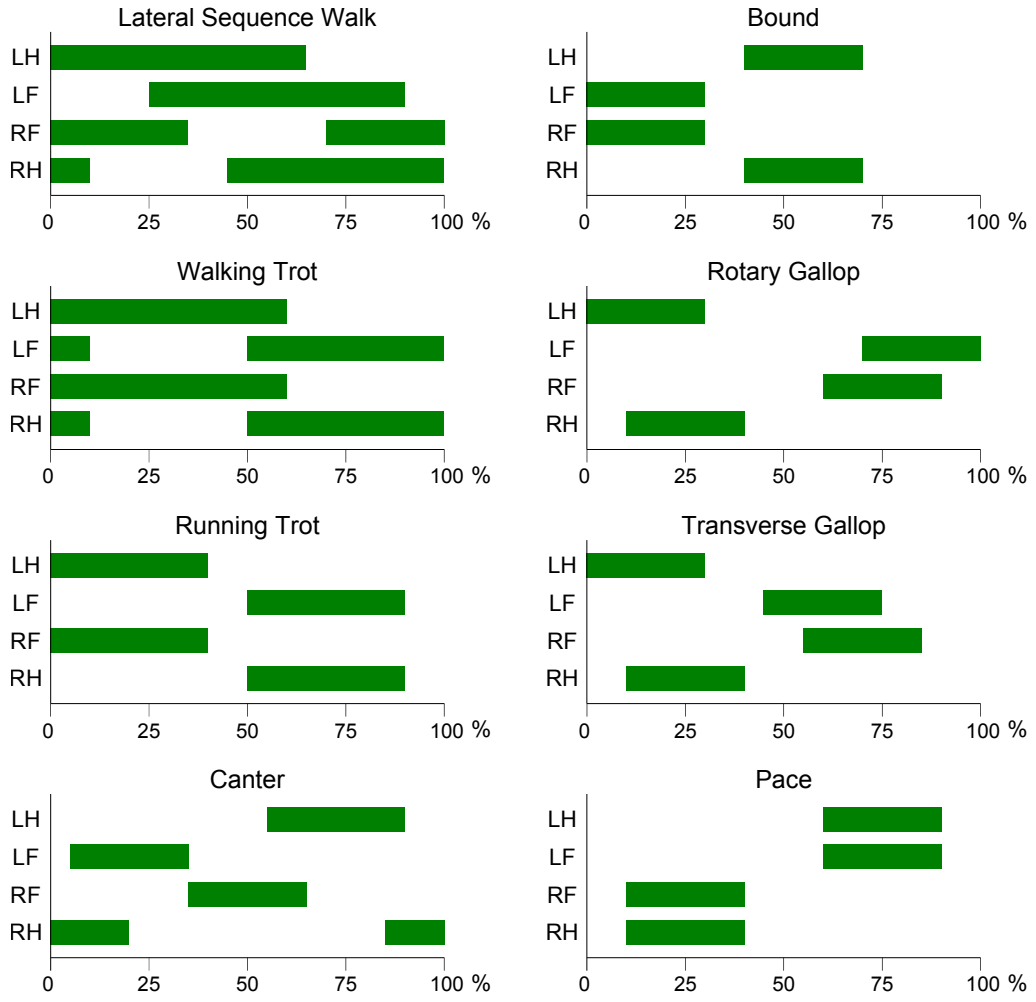


Figure 2.8: Gait Graphs

Legged locomotion patterns have been analysed since long time ago. In the 19th century, the English photographer Eadweard Muybridge pioneered the study of dynamic gaits in animal locomotion. He used multiple cameras to capture motion of quick gaits and analysed the phases these gaits undergo. In this way, he realized for example that quick gaits like gallop, running trot, among others, do have a flight phase, during which all legs left the ground [66].

These ideas can be captured in what are called gait graphs. Gait graphs are graphs that depict the timing and relative phases of flight and stance phase of all legs. Figure 2.8 is a gait Graph showing eight different quadruped gaits. They will be useful, because the gait phase $\phi \in [0, 2\pi]$ will be used to know the progress made so far within each stride and in this way, it will be possible to synchronise and learn variable gain stiffness and damping for control of each joint. These gait graphs are based on the works presented in [1, 2, 8].

2.4 Reactive Controller Framework

In this Thesis, the base controller over which the parameter optimization will be performed is the Reactive Controller Framework [3]. In this section, the parts of the control framework relevant for this Thesis will be briefly presented, a detailed explanation of the framework can be found in [3, 4].

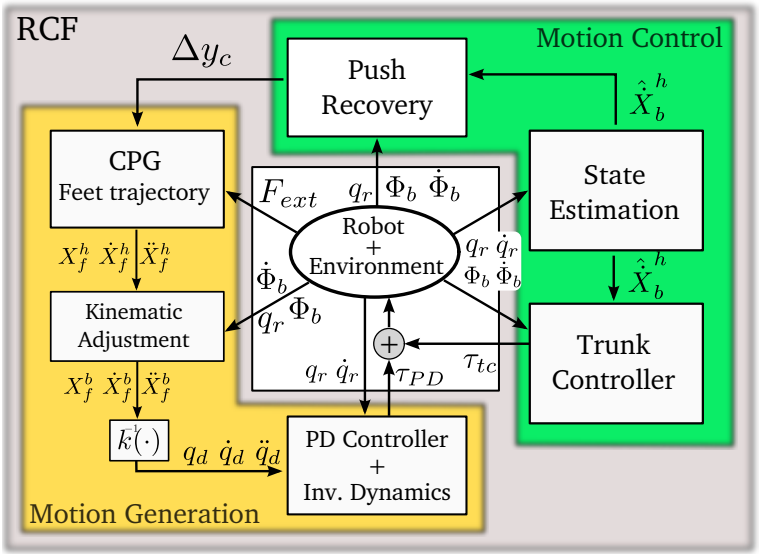


Figure 2.9: Reactive Controller Framework. Taken from [3]

The Reactive Controller Framework has been designed for robust quadrupedal locomotion. It is composed of two modules, as shown in Figure 2.9. The first one is dedicated to the generation of elliptic trajectories for the feet, whereas the purpose of the second one is the control of stability of the robot.

The first module is composed of three sub-modules: an elliptic trajectory generator for the feet (CPG based on task space intentions), a kinematic adjustment scheme and a trajectory tracking controller. The second module is also composed of three sub-modules: a State estimation, a Trunk controller and a Push recovery sub-modules. From these sub-modules, the kinematic adjustment scheme and the push recovery sub-module will not be used.

2.4.1 Workspace Central Pattern Generator - WCPG

The WCPG is composed by a network of nonlinear oscillators, one for each foot, that generate elliptical trajectories in cartesian coordinates. The outputs of these oscillators are filtered by a nonlinear filter, whose output during the swing phase is the normal elliptic trajectory, and during the stance phase, once the foot has made contact with the ground, cut the ellipses. This feature allows to adapt the trajectories to the terrain profile for robust locomotion.

Figure 2.10 shows a reshaped elliptical trajectory. The most important parameters that define the generation of the elliptical trajectories are: the height F_{c_i}, the length L_s, the stride frequency ω_s, and the duty cycle D. For example, in Figure 2.10, the upper half represents the normal elliptical trajectory during the swing phase with the defined height and length, and the lower half shows the reshaped part of the trajectory, where z_{td_i}, is the point at which the foot touch down occurs (start of stance phase) and the ellipse is reshaped.

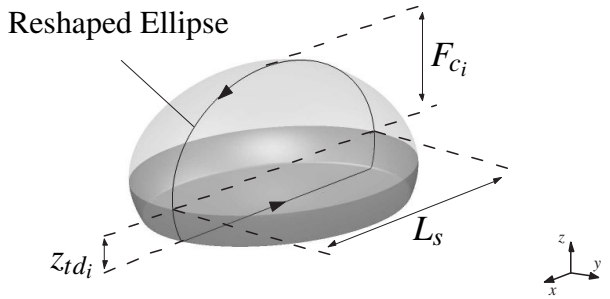


Figure 2.10: Workspace Central Pattern Generator - WCPG. Taken from [4]

The duty cycle D is the percentage of the gait phase $\phi \in [0, 2\pi]$, during which the foot is in stance phase, as shown in Figure 2.8. An important relation between the WCPG parameters shown so far, and the desired forward velocity V_f is:

$$\omega_s = \frac{V_f}{L_s} D \tag{2.2}$$

2.4.2 Trajectory Tracking Controller

The elliptical trajectories generated in Cartesian coordinates are transformed in desired joint space trajectories by an inverse kinematics transformation.

The trajectory tracking controller receives the desired joint space trajectories q_d $\dot{q}_d$ $\ddot{q}_d$ and uses an inverse dynamics algorithm to provide feed-forward commands, and a PD position and torque controller to provide feedback commands. This approach is advantageous because, as a model-based control method enables movement dexterity and accuracy, and allows to reduce the PD feedback gains, improving motion compliance and robustness [36].

$$\tau = \mathit{InvDyn}(q, \dot{q}, \ddot{q}_d) + K_P S(q_d - q) + K_d S(\dot{q}_D - \dot{q}) \tag{2.3}$$

where τ is the vector of generalized forces, S is the selection matrix, K_P and K_D are the position and velocity feedback gains and the inverse dynamics are computed using QR decomposition, as presented in [36].

2.4.3 State Estimation

This sub-module is in charge of the estimation of translational velocities. Angular velocities and accelerations can be directly measured by the gyroscopes and inertial measurement unit (IMU).

The estimation of body velocities is done by mapping joint velocities of the stance legs, assuming there is no slip or that the friction force constraints the forward movement of the feet in stance phase. A detailed explanation and equations can be found in [3].

2.4.4 Trunk Controller

This sub-module has as objective the correction of the robot's attitude. It accomplishes that by providing joint feed-forward commands, that result in a force applied to the trunk of the robot to correct the attitude. This sub-module also performs Gravity compensation.

The forces to be applied to the trunk for attitude correction are calculated based on a PD law for the deviations of the roll and pitch angles from their desired values. These forces are mapped to joint torques without affecting the feet positions. Decoupling the effect of these forces from the forces that come from the Trajectory Tracking Controller can be done by mapping trunk correction forces into the nullspace of the Jacobian associated with the stance legs. A detailed explanation and equations can be found in [3].

2.5 PI^2 Policy Improvement with Path Integrals

PI^2 is one of the state-of-the-art algorithms for reinforcement learning. It is based on the combination of classical optimal control and dynamic programming with modern methods from statistical learning theory.
More precisely, PI^2 states the problem as a stochastic optimal control problem (nonlinear-second order PDE), and finds the exact solution of the transformed Stochastic HJB equation (linear-second order PDE) by using the Feynman-Kac formula. Finally, the path integral is evaluated by generating rollouts with Monte Carlo sampling of the control system, which allows to iteratively improve the control policies. A complete derivation can be found in [68].
Some of the advantages offered by PI^2 are:

- Depending on how the problem is formulated, PI^2 can be used as model based, semi-model based or even model free learning algorithm.
- Control variables to be optimized, are not restricted to motor commands, but they can be something else, like for example a parametrized policy for desired state trajectories (to be used as input for a tracking controller), desired control gains (gain scheduling), among others. The algorithm does not learn the value function, but the controls directly via iterative update.
- The algorithm is numerically robust, it does not involve matrix inversions, tuning of learning rates, computation of gradients (sensitive to noise). It can easily work with discontinuities in the cost function.
- It has only one open parameter, namely, the exploration noise.
- PI^2 is very efficient, which means that it has fast convergence rate and works very well even when using few rollouts.
- It is formulated for continuous state-action spaces, which makes it suitable for learning in real high-dimensional robotic systems.

Algorithms 1 and 2 present the pseudocode for PI^2 main algorithm and the pseudocode for reusing rollouts if desired. They are based on [11, 68].

2.5.1 Basic steps in the Derivation of PI^2

This subsection summarizes the basic steps involved in the derivation of the PI^2 algorithm, and is based on the work presented in [68]. Some extra details are provided in the Appendix A.
As stated at the beginning of this section PI^2 relies in the principles of stochastic optimal control and statistical learning theory. Therefore, the first step is to define the stochastic

Algorithm 1 Policy Improvement with Path Integrals for a 1D parametrized policy

Define

Immediate cost function	$r_t = \overbrace{q_t}^{state\ cost} + \overbrace{\frac{1}{2}\theta^T \mathrm{R}\theta}^{control\ cost}$	**Cost terms**
Terminal cost function	ϕ_{t_N}	
Control cost matrix	$\mathrm{R} \in \mathbb{R}^{M\times M}$	
Parametrized policy	$a_t = {g_t}^T(\theta+\varepsilon), a \in \mathbb{R}^{1\times N}$	**Policy terms**
Basis functions	$g_t \in \mathbb{R}^{M\times 1},\ g \in \mathbb{R}^{M\times N}$	
Initial parameter vector	$\theta_0 \in \mathbb{R}^{M\times 1}$	
Number of param. / basis functions	M	
Exploration zero-mean noise	$\varepsilon \sim \mathcal{N}(0, \gamma^r \Sigma_\varepsilon), \varepsilon \in \mathbb{R}^{M\times 1}$	**PI^2 terms**
Variance of the exploration noise	$\Sigma_\varepsilon \in \mathbb{R}^{M\times M}$	
Decay rate of the exploration noise	γ	
Parameter update weighting function	$\omega = f\left(\delta\theta_t \in \mathbb{R}^{1\times N}\right)$	
Maximum number of param. updates	L	**General terms**
Current number of param. updates	r	
Number of rollouts per update	K	
Number of time steps per rollout	N	

Precompute

for $j = 1$ to N **do**

Projection Matrix $$M_{t_j} = \frac{R^{-1} g_{t_j} g_{t_j}^T}{g_{t_j}^T R^{-1} g_{t_j}}$$

end for

Main loop

while $r \leq L$ or until convergence of the trajectory cost J **do**

Generate k=1...K stochastic parameters $\theta_k = \theta + \varepsilon_k$.

Execute the rollouts with the parametrized policy $a_t = {g_t}^T \theta_k$.

Collect state costs $q_{t_j,k}$ and terminal costs $\phi_{t_N,k}$ for all rollouts.

for all rollouts k = 1...K **do**

Parameters with projected noise $$\hat{\theta}_{t_j,k} = \theta + M_{t_j}\varepsilon_k$$

Cumulative costs $$S(\tau_{i,k}) = \phi_{t_N,k} + \sum_{j=i}^{N-1} q_{t_j,k} + \frac{1}{2}\sum_{j=i+1}^{N-1} \hat{\theta}_{t_j,k}^T \mathrm{R}\hat{\theta}_{t_j,k}$$

Probability $$P(\tau_{i,k}) = \frac{\exp\left(-\frac{1}{\lambda}S(\tau_{i,k})\right)}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}S(\tau_{i,k})\right)}$$

$$\exp\left(-\tfrac{1}{\lambda}S(\tau_{i,k})\right) = \exp\left(-h\frac{S(\tau_{i,k}) - \min\left[S(\tau_i)\right]}{\max\left[S(\tau_i)\right] - \min\left[S(\tau_i)\right]}\right), h = 10$$

end for

for all timesteps i = 1...N **do**

Parameter update in time $$\delta\theta_{t_i} = \sum_{k=1}^{K}\left[P(\tau_{i,k}) M_{t_i}\varepsilon_k\right]$$

end for

for all parameters θ_m m = 1..M **do**

Weighting time updates $$[\delta\theta]_m = \omega([\delta\theta_{t_i=1...N}]_m)$$

end for

Parameter update $$\theta_{r+1} \leftarrow \theta_r + \delta\theta$$

Execute one noiseless rollout to evaluate the trajectory cost $J_{r+1} = \phi_{t_N} + \sum_{i=1}^{N-1} r_{t_i}$ of the updated parameters θ_{r+1}

end while

return Optimized parameters $\theta \in \mathbb{R}^{M\times 1}$

Algorithm 2 Reusing rollouts in PI^2

Given

Number of rollouts to reuse	P
Current iteration	r
Current updated parametes	θ_r
Noisy parametes from last iteration	θ_k for $k = 1...K$
Costs of rollouts from last iteration	$S(\tau_{1,k})$ for $k = 1...K$

Function

```
if P > 0 then
  Indices = Sort the cheapest rollouts [S(τ_{1,k=1...K})]
  if Reevaluate Rollouts = TRUE then
    Append θ_k for k = Indices_{1...P} into rollouts to be executed
    Update noises ε_k = θ_k − θ_r for k = Indices_{1...P}
  else
    Update noises ε_k = θ_k − θ_r for k = Indices_{1...P}
    Reevaluate parameters with projected noise as shown in Algorithm 1.
    Reevaluate cost of the trajectories as shown in Algorithm 1.
  end if
end if
```

optimal control problem. It consists of a dynamical system represented by a stochastic differential equation 2.4a with state vector $x_t \in \mathbb{R}^n$, control vector $u_t \in \mathbb{R}^m$. The functions $f_t : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ and $G_t : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ are the passive dynamics function and the state dependent control transition matrix. W_t is an m-dimensional standard Brownian motion which is given on the probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t \geq 0}, \mathbf{P})$.

We want to minimize the expected cost, as given in equation 2.4c, composed of a final cost ϕ_{t_N} and an immediate cost r_t, as given in equation 2.4b. The immediate cost can be designed with any arbitrary state cost, but it requires a quadratic cost for the controls.

Problem Statement

$$dX_t = (f_t + G_t u_t)dt + (G_t \Sigma_\varepsilon^{\frac{1}{2}})dW \tag{2.4a}$$

$$r_t = q_t + \frac{1}{2} u_t^T R u_t \tag{2.4b}$$

$$V(x_{t_i}) = \min_{u_{t_i \ldots t_N}} \mathbb{E}_{\tau_i} \left[\phi_{t_N} + \int_{t_i}^{t_N} r_t dt \right] \tag{2.4c}$$

The *Hamilton-Jacobi-Bellman* equation for a stochastic process 2.4a and cost functional 2.4c is as given in equation 2.5a. From this equation, we can derive the optimal controls, by taking the derivative with respect to the control variables an setting it to zero, as in equation 2.5b. If we put back this result into the HJB equation, we obtain a second order and nonlinear partial differential equation in terms of the cost functional.

Stochastic HJB Equation

$$-\frac{\partial V_t}{\partial t} = \min_u \{r_t + (\nabla_x V_t)^T (f_t + G_t u_t) + \frac{1}{2} \mathit{Trace}(G_t \Sigma_\varepsilon G_t^T \nabla_{xx} V_t)\} \quad (2.5a)$$

$$u(x_t) = u_t = -R^{-1} G_t^T (\nabla_x V_t) \quad (2.5b)$$

$$-\frac{\partial V_t}{\partial t} = q_t - \frac{1}{2}(\nabla_x V_t)^T G_t R^{-1} G_t^T (\nabla_x V_t) + (\nabla_x V_t)^T f_t + \frac{1}{2} \mathit{Trace}(G_t \Sigma_\varepsilon G_t^T \nabla_{xx} V_t)$$

Using the transformation 2.6a and the assumption 2.6b, the second order nonlinear PDE 2.5 can be transformed into a second order linear PDE 2.6c with boundary condition $\Psi_{t_N} = \exp\left(-\frac{1}{\lambda}\phi_{t_N}\right)$. This equation is known as the Chapman-Kolmogorov backward PDE.

Transformation of the Stochastic HJB Equation

$$V_t = -\lambda \log \Psi_t \quad (2.6a)$$

$$\Sigma_t = \lambda G_t R^{-1} G_t^T = G_t \Sigma_\varepsilon G_t^T \quad (2.6b)$$

$$-\frac{\partial \Psi_t}{\partial t} = -\frac{1}{\lambda} q_t \Psi_t + f_t^T (\nabla_x \Psi_t) + \frac{1}{2} \mathit{Trace}\{(\nabla_{xx} \Psi_t) G_t \Sigma_\varepsilon G_t^T)\} \quad (2.6c)$$

Now we make use of the Feynman–Kac formula, that establishes a connection between parabolic partial differential equations and stochastic processes. In this case, it offers the possibility to solve the PDE by simulating random paths of a stochastic process, as given in equations 2.7.

Using Feynman-Kac Formula

$$\Psi_{t_i} = \mathbb{E}_{\tau_i}\left(\Psi_{t_N} \exp\left(-\int_{t_i}^{t_N} \frac{1}{\lambda} q_t dt\right)\right) \quad (2.7a)$$

$$\Psi_{t_i} = \mathbb{E}_{\tau_i}\left(\exp\left(-\frac{1}{\lambda}\phi_{t_N} - \frac{1}{\lambda}\int_{t_i}^{t_N} q_t dt\right)\right) \quad (2.7b)$$

$$\Psi_{t_i} = \lim_{dt \to 0} \int p(\tau_i | x_i) \exp\left[-\frac{1}{\lambda}\left(\phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt\right)\right] d\tau_i \quad (2.7c)$$

τ_i is a trajectory composed by all the states along the trajectory $(x_{t_i}, ..., x_{t_N})$. In order to compute the solution of 2.7a, we have to evaluate an expectation over the trajectories, which means that we need to approximate the probability $p(\tau_i | x_i)$, here is where the statistical learning theory plays an important roll.

For doing that, we first discretize the dynamical system, as shown in equation 2.8a, and approximate the standard Brownian motion dW with $\sqrt{dt}\,\varepsilon_t$, where ε_t is a normally

distributed random variable with zero mean and unit variance $\varepsilon_t \sim \mathcal{N}(0,1)$. This equation can also be decomposed into the controlled c and uncontrolled states m, which helps us to realize that only the controlled states are affected by the Brownian motion. Therefore, when we approximate the probability $p(\tau_i|x_i)$, only the controlled states contribute to it, given that the uncontrolled states are considered to have deterministic dynamics and therefore, its probability is a Dirac delta.

Approximating the Path Integral

$$x_{t+1} = x_t + (f_t + G_t u_t)dt + (G_t \Sigma_\varepsilon^{\frac{1}{2}})\sqrt{dt}\,\varepsilon_t \tag{2.8a}$$

$$\begin{bmatrix} x_{t+1}^m \\ x_{t+1}^c \end{bmatrix} = \begin{bmatrix} x_t^m \\ x_t^c \end{bmatrix} + \begin{bmatrix} f_t^m \\ f_t^c + G_t^c u_t \end{bmatrix} dt + \begin{bmatrix} 0 \\ G_t^c \end{bmatrix} \left(\sqrt{dt}\,\Sigma_\varepsilon^{\frac{1}{2}} \varepsilon_t \right) \tag{2.8b}$$

$$p(\tau_i|x_i) = \prod_{j=i}^{N-1} p(x_{j+1}|x_j) \propto \prod_{j=i}^{N-1} p(x_{j+1}^c|x_j) \tag{2.8c}$$

$$= \frac{1}{\alpha} \exp\left(-\frac{1}{2} \sum_{j=i}^{N-1} \|x_{t+1} - x_t - (f_t + G_t u_t)dt\|^2_{\Sigma_t{}^{-1}} \right) \tag{2.8d}$$

$$\textbf{where: } \alpha = \prod_{j=i}^{N-1} \left((2\pi)^l \det \Sigma_\varepsilon \right)^{\frac{1}{2}} \tag{2.8e}$$

$$\Psi_{t_i} = \lim_{dt \to 0} \int \frac{1}{\alpha} \exp\left(-\frac{1}{\lambda} S(\tau_i) \right) d\tau_i^c \tag{2.8f}$$

$$S(\tau_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2} \sum_{j=i}^{N-1} \left\| \frac{x_{t_{j+1}}^c - x_{t_j}^c}{dt} - f_{t_j}^c \right\|^2_{H_{t_j}^{-1}} dt \tag{2.8g}$$

$$\textbf{where: } H_{t_j} = G_t^c R^{-1} G_t^{cT} \tag{2.8h}$$

Results and control variables

$$u_{t_i} = R^{-1} G_{t_i} (\nabla_x V_t) = \lambda R^{-1} G_{t_i} \frac{\nabla_{x_{t_i}} \Psi_{t_i}}{\Psi_{t_i}} \tag{2.9a}$$

$$u_{t_i} = \int P(\tau_i) u_L(\tau_i) d\tau_i^c \tag{2.9b}$$

$$P(\tau_i) = \frac{\exp -\frac{1}{\lambda} \left(S(\tau_i) + \frac{\lambda}{2} \sum_{j=i}^{N-1} \log |H_{t_j}| \right)}{\int \exp -\frac{1}{\lambda} \left(S(\tau_i) + \frac{\lambda}{2} \sum_{j=i}^{N-1} \log |H_{t_j}| \right) d\tau_i^c} \tag{2.9c}$$

$$u_L(\tau_i) = R^{-1} G_{t_i}^{c\,T} \left(G_{t_i}^c R^{-1} G_{t_i}^{c\,T} \right)^{-1} \left(G_{t_i}^c \varepsilon_{t_i} - b_{t_i} \right) \tag{2.9d}$$

$$b_{t_i} = \lambda H_{t_i} \Phi_{t_i} \tag{2.9e}$$

$$[\Phi_{t_i}]_j = \mathit{Trace} \left(H_{t_i}^{-1} \left(\partial_{|x_{t_i}^c|_j} H_{t_i} \right) \right) \tag{2.9f}$$

When we put back this probability into 2.7c, we can rewrite the transformed functional cost Ψ_{t_i} as presented in 2.8f. In equation 2.8g, the norm $\|x\|_\Sigma$ equals $x^T \Sigma x$, which is known as Mahalanobis distance.

By reintroducing this result (eq. 2.8f) into the equation for the control variables (eq. 2.5b), the optimal control variables per timestep can be computed, as shown in equations 2.9.

2.6 Parametrized Policies for Function Approximation

Function approximation with parametrized policies is useful when we need a simpler representation of another function. For example, instead of representing a trajectory $f(t)$ with its values indexed by time, it would be simpler to approximate this trajectory by a linear combination of a parameter vector ω and a set of basis functions Ψ (possible nonlinear) as $f(t) = \sum_i \omega_i \Psi_i(t)$. For learning, this means that it is possible to learn in applications with high number of degrees of freedom, such as learning in the context of quadruped locomotion.

The main idea of function approximation is to push the nonlinearity into the basis functions and allow easy learning of the linear parameters. Besides, good basis functions allow to perform a good approximation of any arbitrary function with the minimum number of parameters.

In the following, several basis functions will be presented, as well as their advantages.

2.6.1 Gaussian Basis Functions

Gaussian basis functions are a very popular choice in many fields such as system identification and machine learning. They are local in the time domain. They are characterized by its center c_i and its standard deviation σ. The number of basis functions M to be used in an approximation is usually chosen by the user.

Figure 2.11 shows an example of 5 Gaussian basis functions and its weights approximating a function.

Gaussian basis functions

$$\Psi_i(t) = \exp\left(-\frac{1}{2\sigma_i^2}(t - c_i)^2\right) \tag{2.10a}$$

$$f(x) = \sum_{i=1}^{M} \omega_i \Psi_i(x) \tag{2.10b}$$

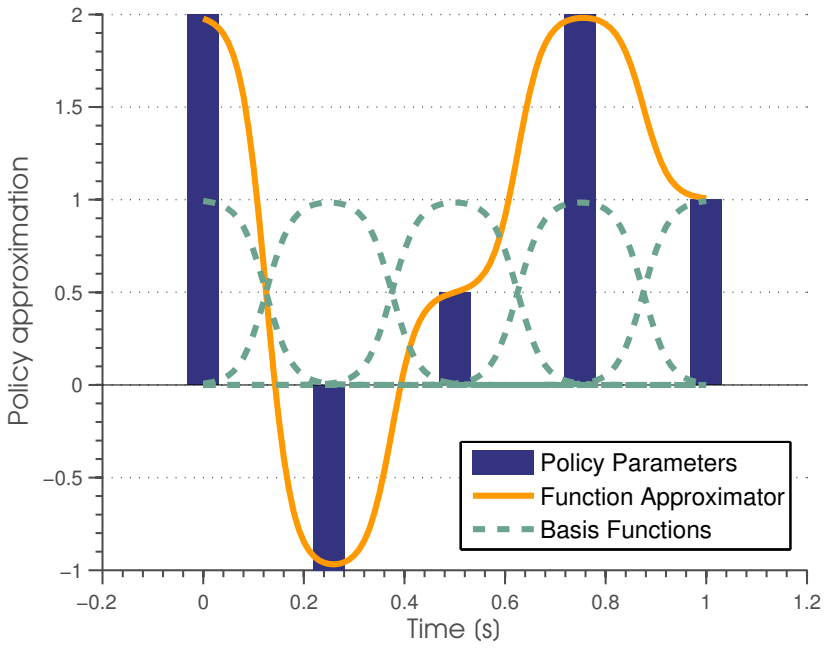


Figure 2.11: Function approximation using Gaussian basis functions

2.6.2 Fourier Basis Functions

Fourier basis functions are suitable for approximation of periodic functions. They have many in applications electrical engineering, signal and image processing, among others. They are not local in time, but they are local in the frequency domain.

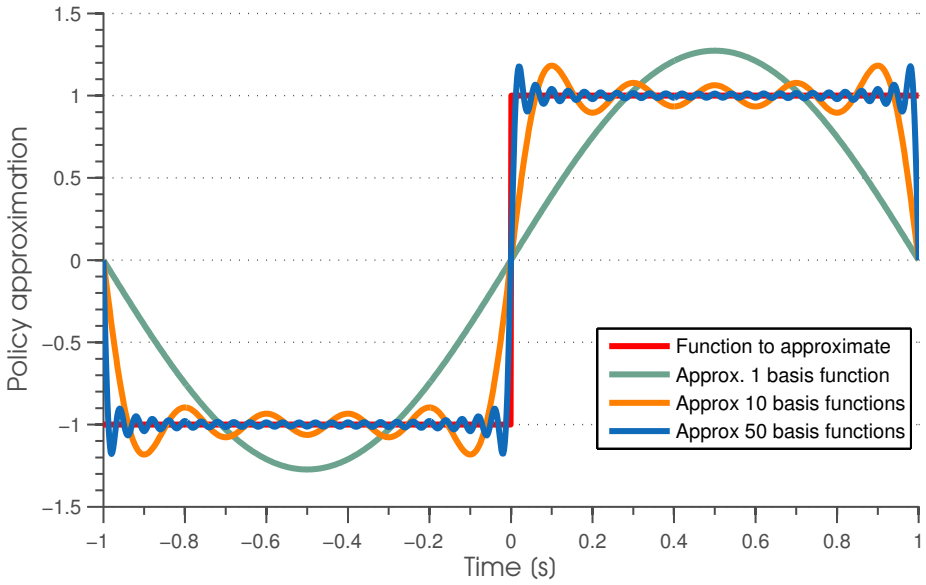


Figure 2.12: Function approximation using Fourier Series as basis functions

They approximate functions using a baseline $\frac{a_0}{2}$ and an infinite sequence of sines and cosines.

Figure 2.12 show the approximation of a pulse train by using different number M of fourier basis functions. It is clear, that by increasing the number of basis functions the approximation is each time better, but an approximation is not perfect.

Fourier basis functions

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{M} a_k \cos(k\Omega t) + b_k \sin(k\Omega t) \tag{2.11a}$$

2.6.3 Von Mises Basis Functions

Von Mises basis functions are used to approximate periodic signals. They are characterized by its center $c_i \in [0, 2\pi]$ and its width h_i.

Figure 2.13 shows a policy with 5 basis functions. As can be seen, an advantage of these Basis functions is that the frequency of the signal can be changed, while keeping the shape of the policy.

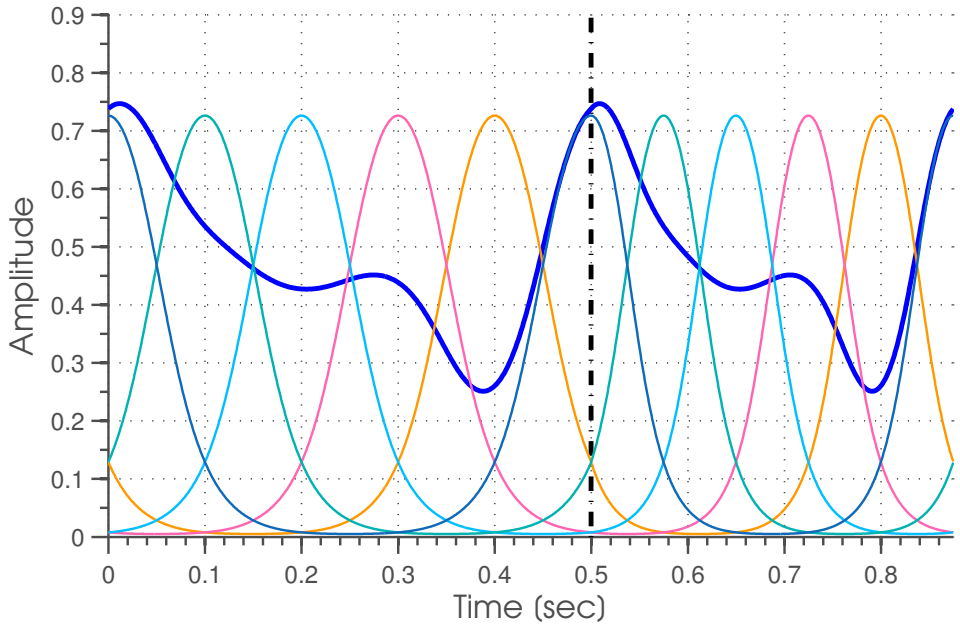


Figure 2.13: Function approximation using Von Mises Basis Functions

Von Mises basis functions

$$\Psi_i(\phi) = \exp\left(h_i\left(\cos\left(\phi - c_i\right) - 1\right)\right) \tag{2.12a}$$

2.6.4 Gaussian Process Learning

Another way to do function approximation is to learn a Gaussian Process that approximates the probability distribution of our samples, or said differently, performs a stochastic approximation of the function.

For example, for a group of N samples, with feature vectors $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathbb{R}^{N_d \times 1}$ and output signals $y_1, ..., y_N \in \mathbb{R}$, the probability of our outputs given the input feature vectors can be approximated with a zero-mean normal distribution with covariance matrix $\mathbf{K}$, as given in equation 2.13a. The elements of the covariance matrix can be calculated using equation 2.13b, where a squared exponential covariance function has been used [39, 69].

Learning a Gaussian Process

$$p(y_1, ..., y_N | \mathbf{x}_1, ..., \mathbf{x}_N) = \mathcal{N}(y_1, ..., y_N | 0, \mathbf{K}) \tag{2.13a}$$

$$\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j) = v_0 \exp\left(-\frac{1}{2}\sum_{k=1}^{N_d} v_k^d (x_{ik} - x_{jk})^2\right) + v_1 \delta_{ij} \tag{2.13b}$$

$$p(y_{N+1} | \mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{x}_{N+1}, y_1, ..., y_N) = \mathcal{N}(y_{N+1} | \mu, \sigma^2) \tag{2.13c}$$

$$\mu = \mathbf{k}(\mathbf{x}_{N+1})^T \mathbf{K}^{-1} \mathbf{y} \tag{2.13d}$$

$$\sigma^2 = \kappa(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) - \mathbf{k}(\mathbf{x}_{N+1})^T \mathbf{K}^{-1} \mathbf{k}(\mathbf{x}_{N+1}) \tag{2.13e}$$

$$\textbf{where: } \mathbf{y} = [y_1, ..., y_N] \tag{2.13f}$$

$$\textbf{where: } \mathbf{k}(\mathbf{x}_{N+1}) = [\kappa(\mathbf{x}_1, \mathbf{x}_{N+1}), ..., \kappa(\mathbf{x}_N, \mathbf{x}_{N+1})]^T \tag{2.13g}$$

δ_{ij} is the Kronecker delta, that equals 1 when $i = j$, and 0 otherwise. x_{ik} and x_{jk} are the $k-th$ element of the $i-th$ and $j-th$ input vectors. The covariance matrix K does approximation and prediction of a function based on similarity between input feature vectors.

Prediction of the output value for a new input vector can be done using the equation 2.13c, where the prediction is characterized by a mean μ and a variance σ^2, as given by equations 2.13d and 2.13e respectively.

An example can be seen in Figure 2.14, where we want to approximate a sinusoidal signal. For that purpose, we have collected five noisy sample points and by using the Gaussian Process Learning algorithm, we can fit a Gaussian Process. As result, we get an approximation characterized by the mean and the variance. As can be seen, using the measure of similarity between data points works well in practice, even for few sample points. Indeed, the measure of the variance can be used to select new points, where we should collect data, in order to improve the approximation and increase our certainty.

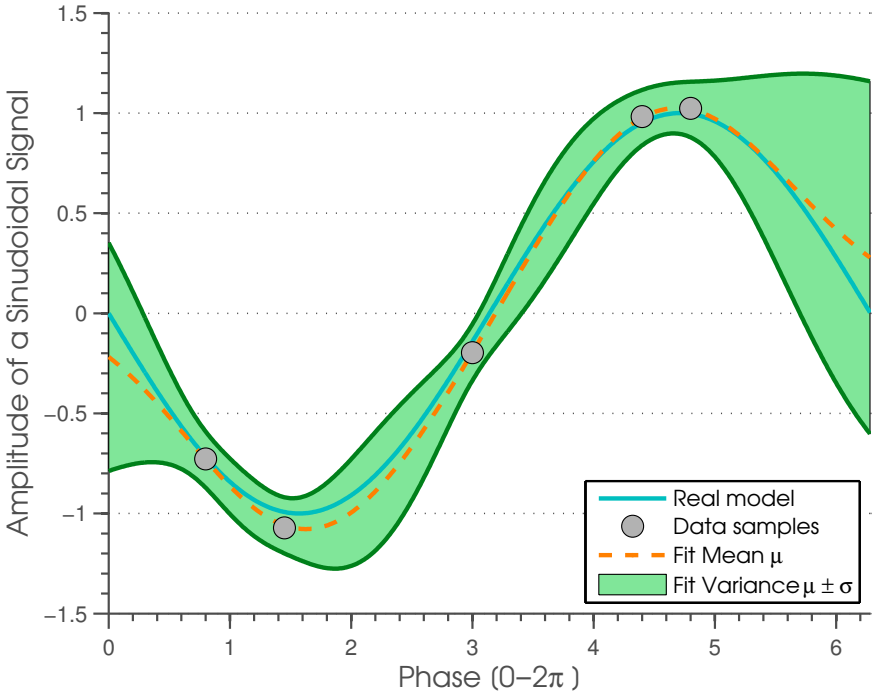


Figure 2.14: Learning a sinusoidal signal with a Gaussian Process

2.6.5 Rhythmic Control Policy - RCP

The representation of trajectories is an important issue in robotics. Some of the desirable properties for such a representation include: easy to represent, learnability and reusability of a trajectory, and robustness against perturbations or modifications of the environment conditions. In this context, dynamic movement primitives and rhythmic movement primitives were developed to easily encode single go-to-goal motions and periodic movements respectively, as presented in [26, 27, 28, 29].

A rhythmic control policy is a movement primitive for rhythmic or periodic motions based on nonlinear oscillators. The RCP is composed of two main systems: a canonical and a transformation system.

The canonical system is composed by equations 2.14. The main equation of this set of differential equations is the equation 2.14a, which represents the evolution of the phase $\phi \in [0, 2\pi]$ of the rhythmic motion with time constant $\tau = \frac{period}{2\pi}$. The canonical system is unique within the RCP and synchronizes the transformation systems that will be explained later. The phase variable can be parametrized differently, for example, instead of using a time dependent phase, the evolution of the phase could be coupled to another state of the system.

Equations 2.14b and 2.14c are used to smoothly change the amplitude r and baseline g of the trajectory, up to a desired value r_0 and g_0 respectively.

Rhythmic canonical system

$$\tau\dot{\phi} = 1 \tag{2.14a}$$
$$\tau\dot{r} = \alpha_r\left(r_0 - r\right) \tag{2.14b}$$
$$\tau\dot{g} = \alpha_g\left(g_0 - g\right) \tag{2.14c}$$

The transformation system (equations 2.15) is designed based on a globally stable second-order linear system, which is modulated with a nonlinear term $f(\phi, r)$. The term $f(\phi, r)$ connects the canonical and transformation systems, it is designed to be periodic, so that the resulting trajectory oscillates with the desired shape.
As explained before, the Von Mises basis functions are characterized by its centers $c_i \in [0, 2\pi]$ and its bandwidths h_i, which are generally chosen equally spaced in the phase range ϕ. The constant parameters for the differential equations of the transformation system are chosen such that the second order differential equation is critically damped when the nonlinear term equals zero. Therefore, $\alpha_z = 25$ and $\beta_z = \alpha_z/4$ are a good choice. The terms α_r and α_g can be chosen to be $\alpha_z/2$.

Rhythmic transformation system

$$\tau\dot{z} = \alpha_z\left(\beta_z\left(g - y\right) - z\right) + f \tag{2.15a}$$
$$\tau\dot{y} = z \tag{2.15b}$$
$$f(\phi, r) = \frac{\sum_{i=1}^{N}\Psi_i\omega_i}{\sum_{i=1}^{N}\Psi_i}r \tag{2.15c}$$
$$\textbf{where: } \Psi_i = \exp\left(h_i\left(\cos\left(\phi - c_i\right) - 1\right)\right) \tag{2.15d}$$

For learning the weights ω_i of the rhythmic control policy, regression can be applied. The technique suggested in [26] is locally weighted regression (LWR), as presented in [59], because it can automatically learn the number of necessary basis functions (some thresholds have to be set by the user though). For simplicity, we present here LWR in batch mode.
The learning process goes as follows: First of all, the frequency of the rhythmic motion has to be learnt (this will be shown in the next section), then locally weighted regression can be applied to learn the weights ω_i for the demonstrated trajectory. By measuring the demonstrated trajectory (y_{demo}, $\dot{y}_{demo}$ and $\ddot{y}_{demo}$), the nonlinear term f_{target} can be computed for each timestep, as shown in equation 2.16a. Then, the criteria used for finding the weights, is to minimize the squared difference between f_{target} and the weight of each receptive field ω_i scaled by the trajectory amplitude (usually set to 1 for

learning), as shown in equation 2.16b. As result of the optimization, the weights can be computed as shown in equation 2.16c.

Locally weighted regression (LWR) - Batch mode

$$f_{target} = \tau^2 \ddot{y}_{demo} - \alpha_z(\beta_z(g - y_{demo}) - \tau \dot{y}_{demo}) \tag{2.16a}$$

$$J_i = \sum_{t=1}^{N} \Psi_i(t)(f_{target}(t) - \omega_i r)^2 \tag{2.16b}$$

$$\omega_i = \frac{b^T \Gamma_i f_{target}}{b^T \Gamma_i b} \tag{2.16c}$$

$$b = \begin{pmatrix} r(1) & \cdots & r(N) \end{pmatrix}^T \tag{2.16d}$$

$$f_{target} = \begin{pmatrix} f_{target}(1) & \cdots & f_{target}(N) \end{pmatrix}^T \tag{2.16e}$$

$$\Gamma_i = \begin{pmatrix} \Psi_i(1) & & & 0 \\ & \Psi_i(2) & & \\ & & \ddots & \\ 0 & & & \Psi_i(N) \end{pmatrix} \tag{2.16f}$$

Figure 2.15 shows an example of a rhythmic control policy. In the last row, first and second plots, the canonical system is shown. Here, it is possible to see the periodic evolution of the phase ϕ and its derivative $\dot{\phi}$. In the first row, the demonstrated trajectory y_{demo}, $\dot{y}_{demo}$ and $\ddot{y}_{demo}$ (black) and the learned trajectories (blue) are presented. Finally, in the last column, the basis functions for one period and the corresponding learned weights are depicted.

2.7 Adaptive Frequency Oscillators

This section describes a way to learn the frequency of a periodic signal using nonlinear oscillators. As seen so far, nonlinear oscillators are widely used in engineering because of their interesting properties. In this section, we will make use of its synchronization ability when coupled to other dynamical systems to learn the frequency of a periodic signal.

The learning method presented in this section is based on [15, 54, 55]. The main idea is to identify the parameter that most influences the evolution of the frequency of the dynamical system. Then, the learning rule is constructed by making the parameter a dynamical system. The advantage of this, is that the learning is not done as an offline optimization process, but instead is it part of the dynamics and can be done online very efficiently.

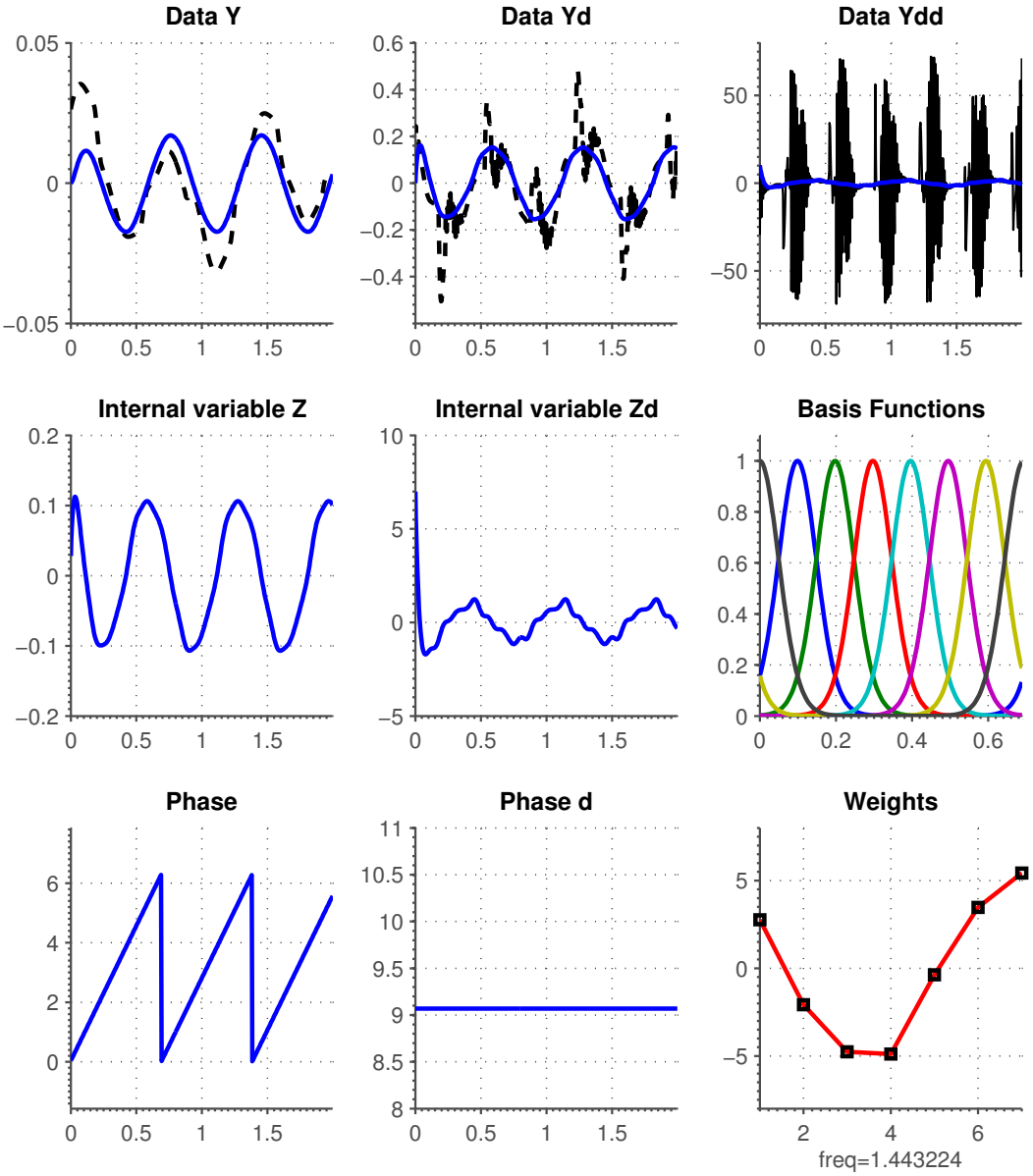


Figure 2.15: Example of a Rhythmic Control Policy

Single frequency learning

$$\dot{\phi} = \frac{\omega}{\tau} - KF\sin(\phi) \tag{2.17a}$$

$$\dot{\omega} = -KF\sin(\phi) \tag{2.17b}$$

First, we examine the method for identification of a single frequency, as presented in equations 2.17. A periodic signal can be represented as the evolution of its magnitude and phase; as we are interested in learning the frequency and do not want this to be

influenced by the amplitude of its oscillations, the learning rule can be constructed based on the phase equation of the dynamical system, as shown in 2.17a. ϕ is the phase, ω the frequency, F represents the input signal, whose frequency we are interested in, τ is the relaxation time and K is a term that determines the coupling strength.

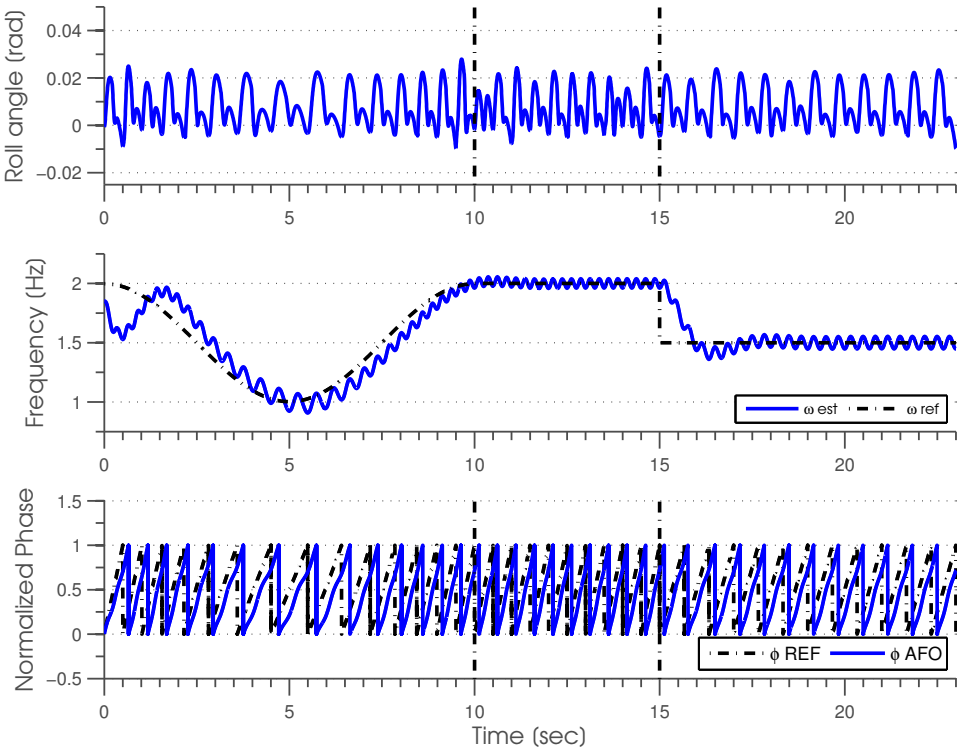


Figure 2.16: Single frequency learning

Equation 2.17b is our learning rule for the frequency. This equation provides the adaptation ability and plasticity to the oscillator, by allowing the change of its frequency parameter ω. When the frequency of the oscillator is close to the frequency of the periodic input, it will synchronize, phase-lock (this effect is known as entrainment); in this way, we can learn the frequency of a signal. More details about frequency learning, entrainment basin and proof of convergence can be found in [55].

Figure 2.16 shows an example of frequency learning, where we want to learn the frequency of the locomotion gait of the robot by estimating the frequency of one of its periodic variables, namely, the roll angle. The first plot shows the input signal, whose frequency, we are willing to learn. In the second plot, the reference frequency ω_{ref} and the estimated frequency ω_{est}, using equations 2.17, are shown. The estimated frequency is integrated to obtain an estimated phase ϕ_{AFO}. In the last plot, this estimated phase is compared with the reference phase, and we can see that they are close, but they have a phase difference. This can be improved as will be seen later.

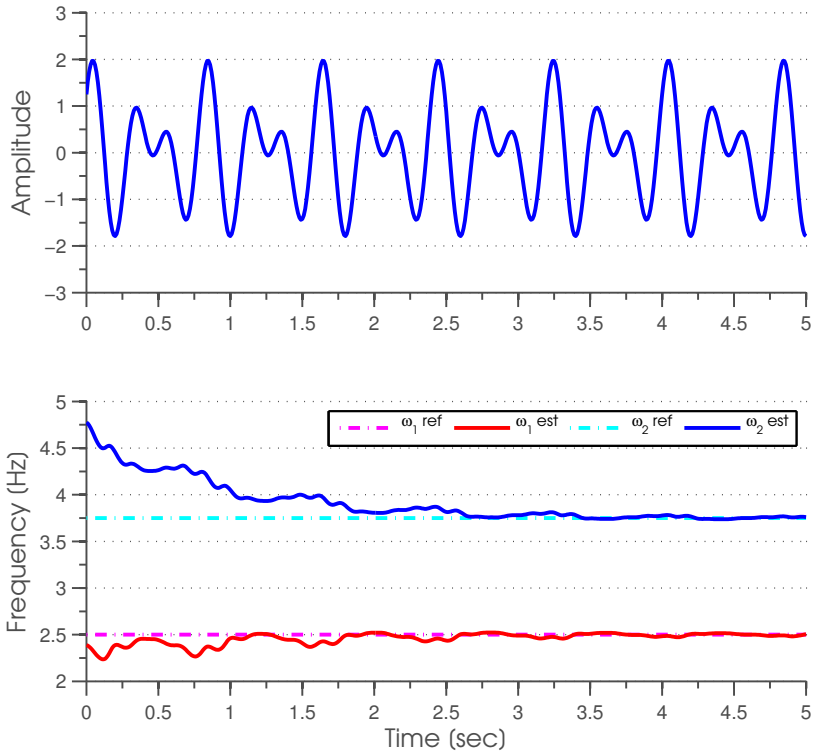


Figure 2.17: Multiple frequency learning

When we are interested in learning multiple frequencies of a multiple frequency signal, a similar approach can be used. The main idea is to let each oscillator learn a frequency component of the periodic signal [15].. Equations 2.18 show the process. The difference between this framework and the one presented before for single frequency learning, is the introduction of a feedback loop. This feedback loop helps to stabilize the learned frequencies as the difference $e(t)$ between the input signal $y_{measured}(t)$ and the approximated signal $\hat{y}(t)$ approaches zero. The approximated signal $\hat{y}(t)$ is computed as a linear combination of M adaptive frequency-phase oscillators, as shown in equation 2.18d, and also allows amplitude adaptation by using equation 2.18e.

Multiple frequency learning

$$\dot{\phi}_i = \frac{\omega_i}{\tau} - Ke\,(t)\sin\,(\phi_i) \tag{2.18a}$$

$$\dot{\omega}_i = -Ke\,(t)\sin\,(\phi_i) \tag{2.18b}$$

$$e\,(t) = y_{measured}(t) - \hat{y}(t) \tag{2.18c}$$

$$\hat{y}(t) = \sum_{i=1}^{M} \alpha_i \cos\,(\phi_i) \tag{2.18d}$$

$$\dot{\alpha}_i = \eta \cos\,(\phi_i) e\,(t) \tag{2.18e}$$

Figure 2.17 shows an example of learning a periodic signal with $M = 2$ adaptive frequency oscillators. The first plot shows the periodic signal, and the second one shows the evolution of the learned frequencies as they approach the two main frequency components of the periodic signal.

The improvement that can be done for phase estimation, comes from the [40, 54], where a phase resetting mechanism is suggested to compensate the slightly difference between the frequency at which the legs touch the ground and the frequency of the control trajectories (in our case, variable impedance synchronization). This can be done by using feet contact information to adjust the phase estimate, coupling the true body locomotion phase and the controller.

Phase resetting mechanism using feet contact information

$$\dot{\phi}_{\text{pr}} = \hat{\omega}^n_{\text{pr}} + \delta(t - t_{\text{contact}})(\phi^{HyQ}_{\text{contact}} - \phi_{\text{pr}}) \tag{2.19a}$$

$$\hat{\omega}^{n+1}_{\text{pr}} = \hat{\omega}^n_{\text{pr}} + K(\omega^n_{\text{est}} - \hat{\omega}^n_{\text{pr}}) \tag{2.19b}$$

The equations used for the phase resetting mechanism are equations 2.19. $\hat{\omega}^n_{\text{pr}}$ is a filtered version of the frequency estimated with AFO ω^n_{est}. The effective phase velocity $\dot{\phi}_{\text{pr}}$ comes from the filtered frequency $\hat{\omega}^n_{\text{pr}}$ and a term that compensates for the difference between the phase estimate ϕ^n_{pr} and the real phase of HyQ $\phi^{HyQ}_{\text{contact}}$ at touch down. Figure 2.18 shows in pink the improvement in the phase estimate by using this phase resetting mechanism.

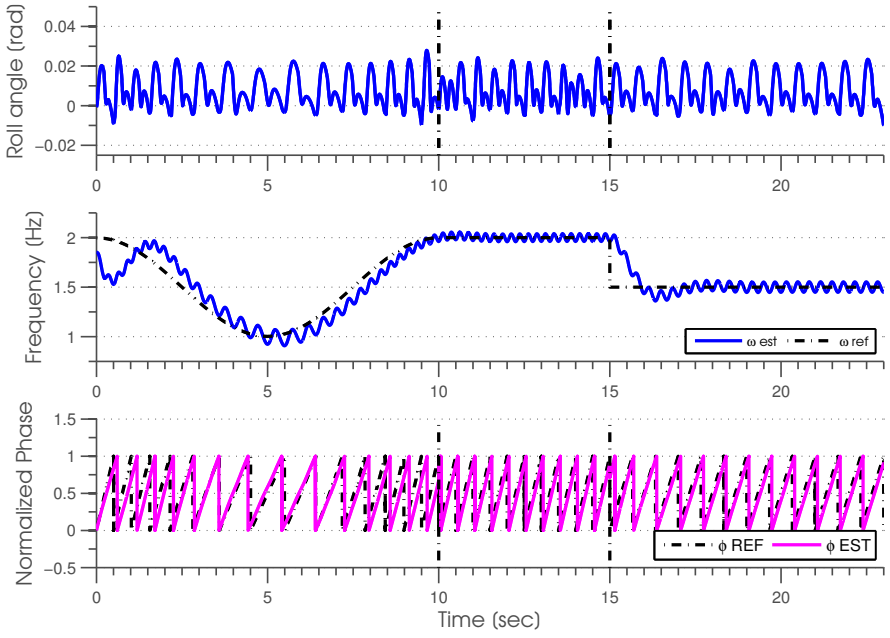


Figure 2.18: Phase resetting mechanism by using feet contact information.

2.8 Impedance control

For tasks involving interaction with the environment, such as force control and locomotion, high gain control approaches are not suitable, instead the tasks require the robot to behave compliant. Impedance control is one the frameworks that offer the possibility to introduce compliance in the robot control design.

The main idea in Impedance Control is to design the disturbance response when a perturbation causes the system to deviate from its desired trajectory [13] . In other words, it designs a specific impedance behaviour for the robot interaction with the environment.

Impedance Control can define a target impedance for any point of the robot, provided that a kinematic transformation can be defined between the actuated joints and the point. Usually the end-effectors are selected as the points, where we want to define an impedance. Equation 2.20a defines a common impedance behavior by using a second order dynamical system. This equation relates the interaction forces F_s and position x_{ee} of the end effector. $K_{P,x}$, $K_{D,x}$ and $K_{M,x}$ are the end effector desired stiffness, damping and inertia matrices.

Impedance Control

$$F_s = K_{P,x}\left(x_{ee_d} - x_{ee}\right) + K_{D,x}\left(\dot{x}_{ee_d} - \dot{x}_{ee}\right) + K_{M,x}\ddot{x}_{ee} \tag{2.20a}$$

$$\ddot{q}_d = J_s^{\dagger}\left(K_{M,x}^{-1}\left[K_{P,x}\left(x_{ee_d} - x_{ee}\right) + K_{D,x}\left(\dot{x}_{ee_d} - \dot{x}_{ee}\right) + F_s\right] - \dot{J}_s\dot{q}\right) \tag{2.20b}$$

$$F_s = K_{P,x}\left(x_{ee_d} - x_{ee}\right) + K_{D,x}\left(\dot{x}_{ee_d} - \dot{x}_{ee}\right) \tag{2.20c}$$

$$\tau_d = J_s^T F_s = J_s^T\left[K_{P,x}\left(\Delta x\right) + K_{D,x}\left(\Delta\dot{x}\right)\right] \tag{2.20d}$$

$$\tau_d = J_s^T\left[K_{P,x}\left(J_s\Delta q\right) + K_{D,x}\left(\dot{J}_s\Delta q + J_s\Delta\dot{q}\right)\right] \tag{2.20e}$$

$$\tau_d = (J_s^T K_{P,x} J_s)\Delta q + (J_s^T K_{D,x} J_s)\Delta\dot{q} \tag{2.20f}$$

In order to map this impedance from task space to joint space, we use the contact model equation 2.1d, that relates acceleration in joint and task space. By plugging equation 2.20a in 2.1d, we obtain equation 2.20b, that specifies the joint accelerations required to realize the desired impedance in Cartesian space ($J_s^{\dagger}$ is the Moore-Penrose pseudoinverse of the Jacobian). By using these joint space accelerations and the equation of motion of the robot (equation 2.1b), the torques that satisfy the desired dynamics at the end effector can be computed.

In our Framework for HyQ, we do not specify a desired inertia at the end-effector and, Coriolis and gravity forces are compensated by feed-forward commands. For this reason, Impedance Control law simplifies to a spring-damper system between the end-effector

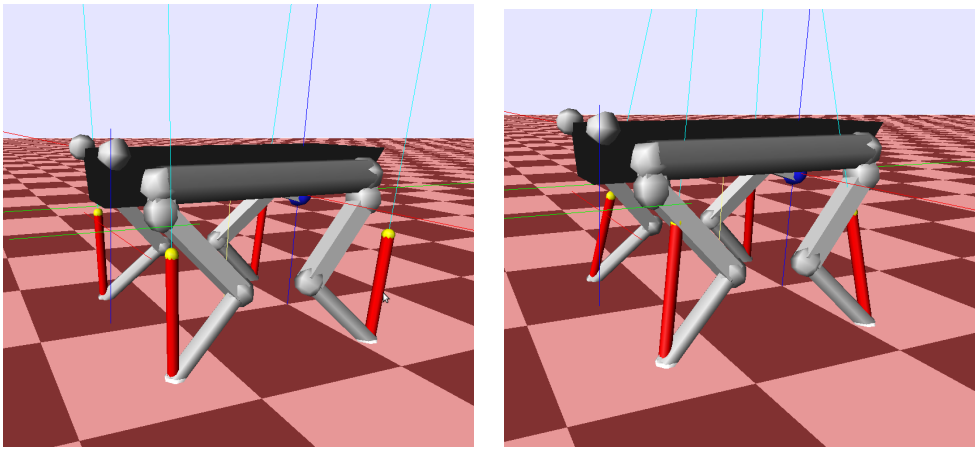


Figure 2.19: Impedance Control in Task Space (left) and Joint Space (right). Taken from [14]

and the desired trajectory, as shown in equation 2.20c. By using the Jacobian J_s, we can map changes in Cartesian coordinates to changes in Joint space coordinates, which is expressed in equation 2.20e. Equation 2.20f is obtained by assuming that the geometric stiffness due to changes in the Jacobian $\dot{J}_s$ is negligible in comparison to the other terms [5].

As final result, the stiffness $K_{P,x}$ and damping $K_{D,x}$ gain matrices in task space can be mapped to stiffness $K_{P,q}$ and damping $K_{D,q}$ gain matrices in joint space by using the following relations:

$$K_{P,q} = J_s^T K_{P,x} J_s$$
$$K_{D,q} = J_s^T K_{D,x} J_s$$

An important observation done in [14], states that impedance control implemented in task space differs from impedance control implemented in joint space for HyQ standing still in that, the ground reaction forces that result from the first one are vertical, while the ground reaction forces for the second one are not. Vertical forces are desirable, because they help to reduce the probability of slippage during locomotion. This suggest that an implementation of impedance control in task space is worthy.

Another important detail, is that the damping matrix for impedance control, should be selected such that the dynamics of the system result in a critically damped behaviour. This is important in order not to introduce high frequency signals that could possibly excite unwanted dynamics in HyQ. For such purpose, the damping gains can be selected as:

$$K_{D,x} = 2\sqrt{K_{P,x}}$$

3 Learning and Control

> *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."*
>
> — Definition of Machine Learning, *from [37].*

This chapter provides a description of the design and control architecture implemented for gait optimization using the reinforcement learning algorithm PI^2. The learning algorithm is built on top of the physics and control environment SL (Simulation Laboratory) [58], and using the Optimization engine using reinforcement learning algorithms [17].
SL is a simulation and real-time control software, that allows testing and debugging of robot programs in simulation, and on the real robot without further modifications. It is built in a modular way, which decouples low and high level control. Low level control is done by the *Motor Servo*, which is in charge of handling input/output with the robot or simulation, basic feedback loops and computes motor commands based on desired quantities. The *Task Servo* executes high level control algorithms to obtain the desired motor behaviour for a given task. It creates, for example, desired position and velocity, and feed-forward commands that accomplish some goal.
The Optimization engine is a relatively new tool, that offers a general framework for optimization using reinforcement learning algorithms. This Thesis makes use of this framework and also collaborates by supplying the PI^2 implementation.

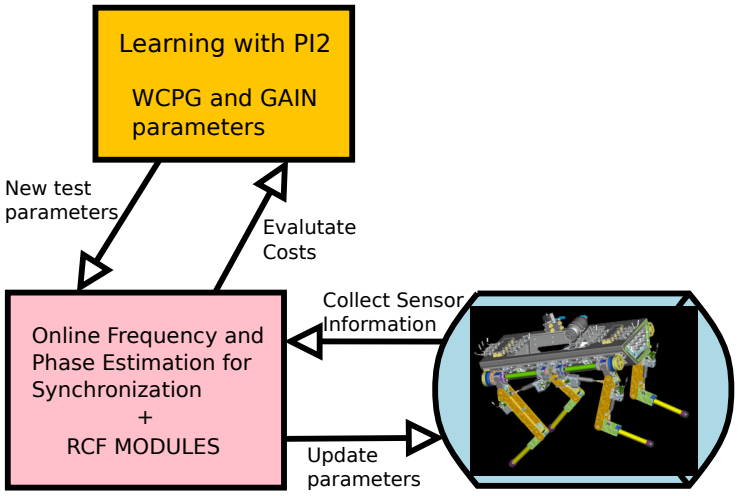


Figure 3.1: Brief picture of the Learning Process with PI^2

A brief picture of the learning and control setup consists in the following key ideas. First, online learning of frequency and phase is performed to synchronize the feedback control policies (Control and Adaptation Layer). Then, by means of executing and evaluating roll-outs, the feedback control policies for variable impedance control and trunk stabilization are tested and improved (Learning Layer). A roll-out is a single execution of the policy parameters.

3.1 Control and Adaptation setup

This section describes the main ideas involved in the control and synchronization of feedback policies of a trotting gait for the hydraulic quadruped robot HyQ.
As stated earlier, the base control algorithm for this optimization is the Reactive Controller Framework - RCF [3]. The variables to be optimized and controlled can be separated in two groups:

- The first group comprises the high level variables that define the workspace CPG, namely, the stride frequency ω_s, the step length L_s, the step height F_{c_i}, the duty cycle D and the desired forward speed of the robot V_f. This group will be referred to as WCPG-parameters group.

- The second group is composed of the feedback gains for the PD control of pitch and roll dynamics, and the feedback gains for the PD torque control of the joints. This group is called GAIN-parameters group.

On the one hand, the variables of the WCPG-parameters group are kept constant during each roll-out. The variables for the pitch and roll dynamics of the GAIN-parameters group are also constant during each roll-out. The variables mentioned above do not require any synchronization with the gait.
On the other hand, the variables for stiffness $K_{P,x}$ and damping $K_{D,x}$ for the torque control of the joints do require synchronization with the gait frequency. Learning is only done for the stiffness variables of the feedback policies, the damping variables are set appropriately to obtain a critically damped behaviour.
Impedance control in Task space, as presented in section 2.8, is used to parametrize the feedback gains ($K_{P,x}$, $K_{D,x}$). Variable impedance control is used, therefore, the parametrization is periodic and uses Von Mises basis functions (section 2.6.3). The selection of the optimal number of basis functions used to parametrize a policy is still an open research question. In this Thesis, to select the number of basis functions used to parametrize the variable impedance, we take an approach based on [6, 7]. In these works, the authors suggest a very intuitive approach. They represent the trajectories from demonstrations with GMM. This allows to reduce the dimensionality of the trajectory by using only a certain number of classifiers to encode it. Additionally, they use the learned

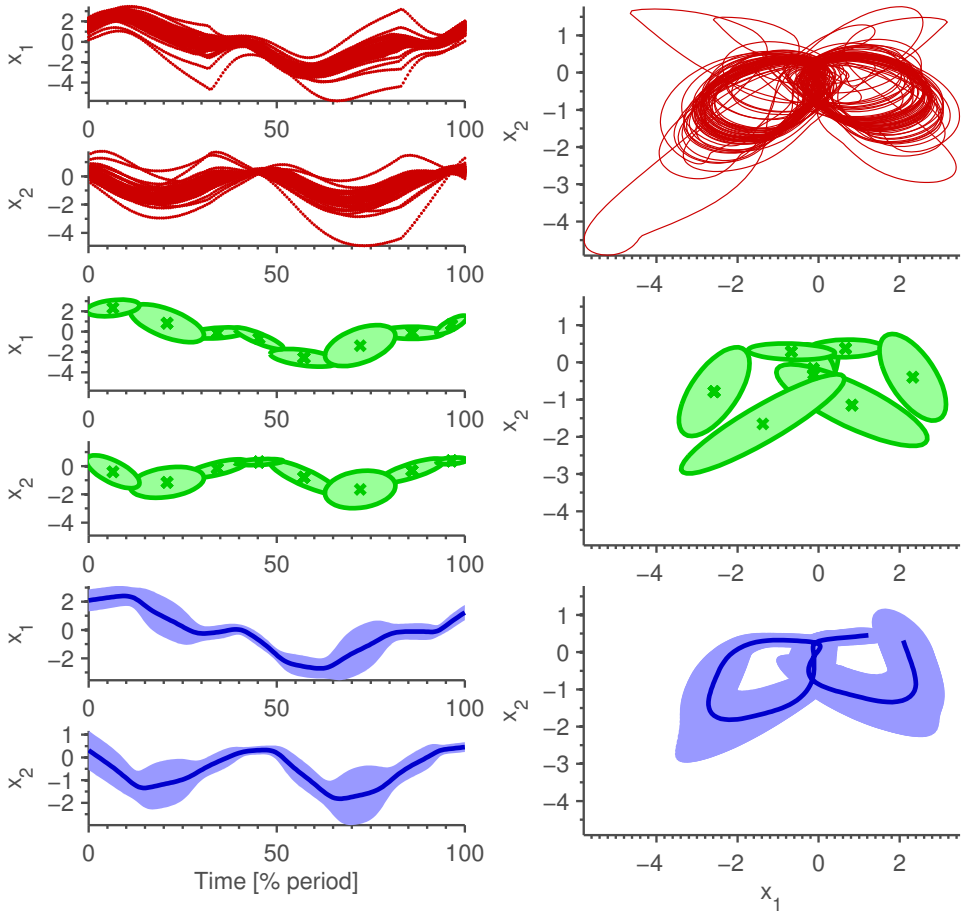


Figure 3.2: Gaussian Mixture Model - Gaussian Mixture Regression for Roll and Pitch trajectories. In the first row, plots of the roll (x_1) and pitch (x_2) trajectories are shown. The second row shows the fit of the Gaussian Mixture Model to the roll and pitch trajectories, with 8 centroids. Finally, the last row shows the generalization through the use of Gaussian Mixture Regression. [6, 7]

variance along the trajectory to specify the impedance. For example, in regions of high variance, the robot can behave compliant, and in regions with low variance, the robot increases the stiffness to closely follow the learned trajectory and reject disturbances. Figure 3.2 shows the representation of the pitch-roll angle trajectories for HyQ using 8 basis functions in GMM. This number was optimized by using the BIC score (equation 3.4d) as shown in Figure 3.3. BIC score will be explained later in detail.

Based on this optimized number, the number of basis functions used to represent the variable impedance in HyQ is chosen to be 10, in order to give enough expressiveness to the policy and to keep the complexity at a reasonable level. This means, for example, that each parameter of the stiffness matrix $K_{P,x}$, namely, $P_{gain_{\mathrm{X}}(\phi)}$, $P_{gain_{\mathrm{Y}}(\phi)}$ and $P_{gain_{\mathrm{Z}}(\phi)}$ is parametrized as a periodic policy of 10 parameters using Von Mises basis functions.

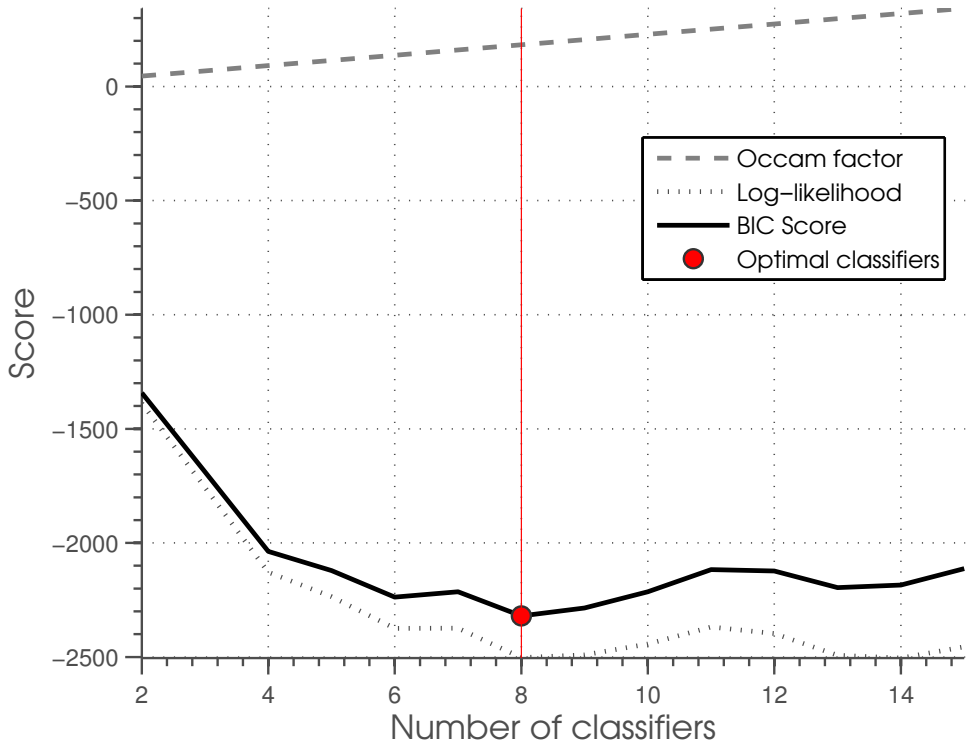


Figure 3.3: BIC score for roll and pitch trajectories estimated using GMM-GMR

These parameters are parametrized as a function of the phase variable ϕ, which represents the gait phase, as explained in section 2.3. This is what makes the policy periodic. In order to apply the policy in the simulation or the robot, the policy needs to be synchronized with the locomotion frequency ω_s. One could attempt to apply the policy by using as reference frequency, the one calculated with equation 2.2. This would indeed be a perfect frequency guess, if the system where deterministic. As it is not, it is not the best that can be done. Here comes into play the theory on adaptive frequency oscillators (AFO).

As explained in section 2.7, AFO can be used to learn the frequency of a signal, in this case, we are interested in using the frequency of the roll-angle to synchronize the feedback policies. A phase resetting mechanism can be used for improving the phase estimate using feet contact information as external input signals. By using the equations on AFO and by using equation 2.2 to initialize our frequency, we can learn the frequency of the roll-angle and use the phase variable obtained from the phase resetting mechanism to synchronize the feedback gains. Figure 3.4 shows the approach.

The feedback policies $K_{P,x}$ and $K_{D,x}$ can now be mapped to joint space by using equations 3.1, where $J_{\text{HIP-EE}}$ stands for the Jacobian between hip and end-effector. This is defined as given in equation 3.1g, where $r_{\text{HIP-EE}}$ is the relative position between hip and end-effector and q represents the active joints of the leg.

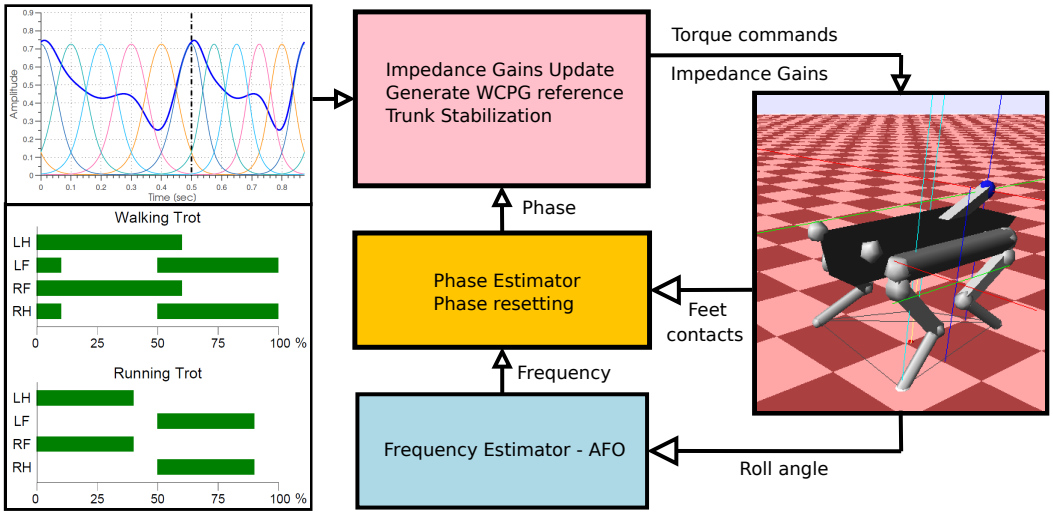


Figure 3.4: Frequency and phase estimation method.

As we optimize a trotting gait, which is a symmetrical gait, we can apply our policy to the two pairs of diagonal legs by just phase-shifting the policy. Feedback policies for the front and hind legs are the same, but only phase shifted.

Feedback gains for torque control of the joints of one leg

$$K_{P,x} = \begin{pmatrix} P_{gain_{\mathsf{X}}(\phi)} & 0 & 0 \\ 0 & P_{gain_{\mathsf{Y}}(\phi)} & 0 \\ 0 & 0 & P_{gain_{\mathsf{Z}}(\phi)} \end{pmatrix} \tag{3.1a}$$

$$K_{D,x} = \begin{pmatrix} D_{gain_{\mathsf{X}}(\phi)} & 0 & 0 \\ 0 & D_{gain_{\mathsf{Y}}(\phi)} & 0 \\ 0 & 0 & D_{gain_{\mathsf{Z}}(\phi)} \end{pmatrix} \tag{3.1b}$$

$$K_{P,q} = \begin{pmatrix} P_{gain_{\mathsf{HAA}}(\phi)} & & \\ & P_{gain_{\mathsf{HFE}}(\phi)} & \\ & & P_{gain_{\mathsf{KFE}}(\phi)} \end{pmatrix} \tag{3.1c}$$

$$K_{D,q} = \begin{pmatrix} D_{gain_{\mathsf{HAA}}(\phi)} & & \\ & D_{gain_{\mathsf{HFE}}(\phi)} & \\ & & D_{gain_{\mathsf{KFE}}(\phi)} \end{pmatrix} \tag{3.1d}$$

$$K_{D,q} = J^T_{\text{HIP-EE}} K_{D,x} J_{\text{HIP-EE}} \tag{3.1e}$$

$$K_{D,q} = J^T_{\text{HIP-EE}} K_{D,x} J_{\text{HIP-EE}} \tag{3.1f}$$

$$J_{\text{HIP-EE}} = \frac{\partial r_{\text{HIP-EE}}}{\partial q} \tag{3.1g}$$

3.2 Learning setup

This section describes the main ideas involved in the cost function design for optimization of a trotting gait for the hydraulic quadruped robot HyQ.
As described in the last section, the parameters to be optimized are divided into two groups. This fact is not arbitrary, but is done to explicitly highlight the difference between the parameters in each group and therefore, to design an appropriately cost function.
The first group contains the WCPG-parameters. These parameters define the elliptical trajectories used for locomotion, as explained in section 2.4.1. They are optimized by letting the algorithm explore different combinations of WCPG-parameters. This is done in the following way:

- The duty cycle D is kept constant at 0.55 for a walking trot and at 0.3 for a running trot. The other parameters: forward velocity V_f, step frequency ω_s and step height F_{c_i} are explored, while the step length L_s is implicitly defined by equation 2.2.

- An immediate cost function $\phi_{wcpg}(t)$ and a final-time cost function $\Phi_{wcpg}(t_N)$ are defined. The immediate cost function $\phi_{wcpg}(t)$ takes care of penalizing errors for tracking a desired speed $J_{\text{speed tracking}}(t)$ (equation 3.2c) and favours parameter configurations with high energy efficiency, which means a low value of $J_{\text{energy efficiency}}(t)$ (equation 3.2d), while keeping the robot at a locomotion gait within its joint limits. The term $J_{\text{speed tracking}}(t)$ optimizes the forward speed of the robot, so that it closely follows a desired forward speed. The term $J_{\text{energy efficiency}}(t)$ plays an important roll by helping the optimization find an energy efficient gait by optimizing the locomotion step frequency ω_s. Finally, the term $J_{\text{closeness to joint limits}}(t)$ helps to generate exploration and optimized gaits, whose desired cartesian feet trajectories lie within the robot workspace. This is important in order to avoid hitting the joint limits, where smaller torques can be generated.

- The final-time cost $\Phi_{wcpg}(t_N)$ highly penalizes, if the robot falls J_{fall}. This final-time cost is used in simulation, when learning is performed over a wide range of parameters. On the real-robot, learning is performed using the parameters optimized in simulation and a lower exploration noise, so that learning is only local and safe.

- Using the cost functions explained before, the algorithm seeks for the configuration of parameters that locally minimize the cost. In simulation, the exploration noise is highly enough to allow exploration over a wide range of parameters, which allows to find a good configuration.

Cost Function for Learning with PI^2

$$\phi_{wcpg}(t) = \begin{bmatrix} c_s & c_e & c_j \end{bmatrix} \begin{bmatrix} J_{\text{speed tracking}}(t) \\ J_{\text{energy efficiency}}(t) \\ J_{\text{closeness to joint limits}}(t) \end{bmatrix} \tag{3.2a}$$

$$\phi_{gain}(t) = c_t J_{\text{torques}}(t) + \phi_{wcpg}(t) \tag{3.2b}$$

$$J_{\text{speed tracking}}(t) = \left\| 1 - \frac{v_{\text{HyQ}}}{v_{\text{desired}}} \right\| \tag{3.2c}$$

$$J_{\text{energy efficiency}}(t) = \frac{\sum_{i \in \text{joints}} |\omega_i \tau_i| \, dt}{v_{\text{HyQ}} \, dt} \tag{3.2d}$$

$$J_{\text{closeness to joint limits}}(t) = \sum\nolimits_{i \in \text{joints}} f_{joint}(i) \tag{3.2e}$$

$$J_{\text{torques}}(t) = \sum\nolimits_{i \in \text{joints}} \tau_i^2 \tag{3.2f}$$

$$\Phi_{wcpg}(t_N) = c_{cpg} 0 + J_{\text{fall}} \tag{3.2g}$$

$$\Phi_{gain}(t_N) = c_{pr} \text{Var}_{\text{pitch-roll}} + \Phi_{wcpg}(t_N) \tag{3.2h}$$

The second group contains the GAIN-parameters. These parameters define the force interaction of the robot with the environment, as explained in section 2.8. The pitch and roll stiffness and damping parameters are learned as constant parameters. The stiffness parameters for torque control of the leg's joints are learned as variable gains, to allow leg compliance while in contact with the environment, and stiffness to track trajectories while in flight phase.
The optimization of these parameters is done in the following way:

- An immediate cost function $\phi_{gain}(t)$ and a final-time cost function $\Phi_{gain}(t_N)$ are defined. They are built on top of the cost functions for the WCPG-parameters. This is done to take into account the cost of WCPG-parameters, but also to allow flexibility in the selection of feedback gains based on new terms for the costs. This is useful, because the feedback gains also help at reducing the costs coming from $\phi_{wcpg}(t)$.

- The immediate cost function $\phi_{gain}(t)$ penalizes high torques J_{torques} (equation 3.2f); this includes all feed-forward torques the ones due to trunk stabilization, and the ones due to trajectory tracking given desired joint accelerations.

- The final-time cost function $\Phi_{gain}(t_N)$ is the most important term for the optimization of the gains. It penalizes the variance along the trajectory formed by the pitch and roll angles $\text{Var}_{\text{pitch-roll}}$. This term guides the learning towards discovering trajectories that form a stable limit cycle. This is important because in a stable limit cycle, trajectories in the neighbourhood of the nominal trajectory

approach the nominal trajectory, making the locomotion gait robust. The nominal trajectory is not specified but is discovered and can take any shape.

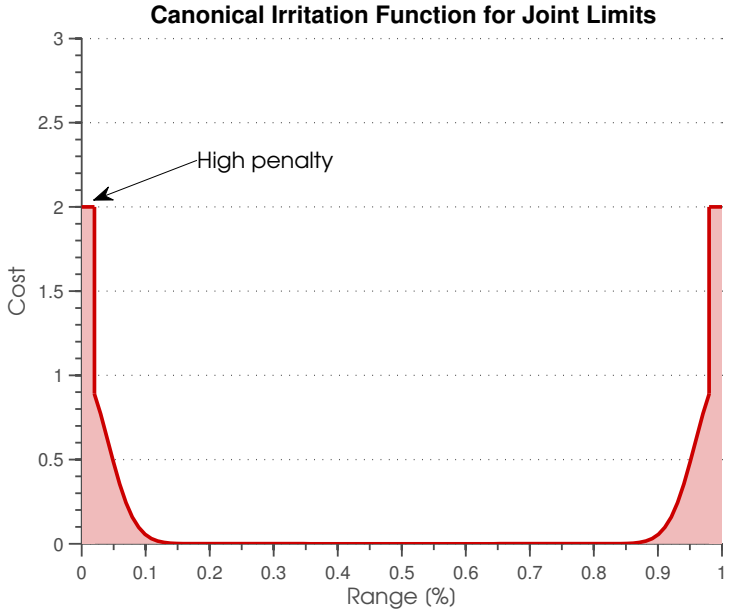


Figure 3.5: Function to penalize closeness to joint limits f_{joint}. The range of each joint is 120 degrees. In this plot the range is shown between 0 and 1, being 0 the minimum limit and 1 the maximum limit. The penalization function has almost no cost in the middle range, which increases when approaching the extremes.

The cost function was designed to be as simple as possible, but expressive enough to be able to efficiently perform a multi-criterion optimization. As it is a multi-criterion optimization, a trade-off between the different objectives is achieved as result of the optimization (Pareto optimal value). For this reason, the weights were selected so that the different objectives contribute to the total cost with the same order of magnitude. In this way all objectives are optimized.

The cost function design aims also at improving the stability and robustness of the locomotion gait. This is achieved by including the minimization of the variance of the roll-pitch trajectories in the cost function. To evaluate this stability improvement, the stability of the robot will be approximated by estimating the roll angle dynamics using a rhythmic control policy (RCP).

The first step is to estimate the split frequency for splitting the roll-pitch trajectories. It is important to mention that, to split the pitch and roll angle trajectories, in order to compute the variance of the limit cycle $\mathrm{Var}_{\text{pitch-roll}}$, an optimization problem to find the optimal frequency to split has to be solved. Neither the final frequency obtained from the AFO, nor a window average of it can be used, because experimentally they do not always perform well at splitting the trajectory. But AFOs perform well for synchronization.

Therefore, for finding the optimal frequency to split the trajectories, the problem, shown in equations 3.3, has to be solved.

Optimal frequency for splitting pitch-roll trajectories

$$f^*_{\text{split}} = \arg\min_{\text{freq}} \sum_{i=1}^{N} \left(T_{\text{pitch-roll}}(i) - T_{\text{fit}}(i)\right)^2 \tag{3.3a}$$

$$T_{\text{fit}} = \sum_{i=1}^{M} \Psi_i(\phi)\omega_i \tag{3.3b}$$

$T_{\text{pitch-roll}}$ represents the pitch and/or roll trajectory composed of N data points and T_{fit} is a periodic function of N data points, fit at a specified frequency $_{\text{freq}}$. Any regression method can be used to fit the function T_{fit}, as for example, least squares. As the goal of solving this problem is to extract the main frequency component, the periodic function T_{fit} used is a simple periodic function of Von Mises basis functions with M components, as shown in equation 3.3b. The Nelder-Mead simplex algorithm was used for finding the optimal frequency [33]. As initial guess for the optimization, the step frequency as given by equation 2.2, is used.

Once the optimal frequency for splitting has been found, the roll-dynamics can be estimated and the robot stability can be evaluated based on that estimation. In order to estimate the roll-dynamics, a rhythmic control policy (RCP) is fit. This is done in the following way.

The RCP stability coefficients (α_z and β_z) and the nonlinear term $f(\phi, r)$ as a parametrized periodic function need to be estimated. Therefore, in order to simultaneously find the best model fit for the data and to penalize model complexity of the nonlinear term, the Bayesian Information Criterion (BIC) is used as the score for the optimization.

The nonlinear term $f(\phi, r)$ is defined as a parametrized policy using M Von Mises basis functions, as given in equation (3.4a). Then, an stacked vector and matrix, as shown in equation 3.4b, is formed based on the measurements of the roll angle, roll angular velocity and roll angular acceleration trajectories. Ψ represents the basis functions for the nonlinear term and $\tau = \frac{1}{2\pi f^*_{\text{split}}}$ is a normalized period. For ease, equation 3.4b is renamed as shown in equation 3.4c, where $\theta \in \mathbb{R}^{M_T}$ represents the vector of $M_T = M+2$ optimization parameters.

As said before, the BIC score is used for the optimization. This score penalizes the fit error of the model to the data and also the model complexity, given in this case, by the number of basis functions M to represent the nonlinear term $f(\phi, r)$. Equation 3.4d shows the full BIC score, and equation 3.4e shows a simplification under certain assumptions, as shown in [9]. In this case, the optimization problem can be stated as shown in equation 3.4f. Figure 3.6 shows an example of an optimization problem, where BIC score is used to find the optimal number of parameters for the approximation of the roll-dynamics for HyQ.

Roll-dynamics and robot stability estimation using BIC score

$$f_{target} = \sum_{i=1}^{M} \Psi_i(\phi)\omega_i \tag{3.4a}$$

$$\begin{bmatrix} \vdots \\ \tau^2\ddot{Y}_{roll} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots \\ -\tau\dot{Y}_{roll} & g - Y_{roll} & \Psi^T \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \alpha_z \\ \alpha_z\beta_z \\ \omega \end{bmatrix} \tag{3.4b}$$

$$b = A\theta \tag{3.4c}$$

$$\ln p(\text{DATA}) \simeq \ln p(\text{DATA}|\theta) + \overbrace{\ln p(\theta) + \frac{M_T}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_{\text{DATA}}|}^{\text{Occam factor}} \tag{3.4d}$$

$$S_{\text{BIC}} = -\ln p(\text{DATA}) \approx -\ln p(\text{DATA}|\theta) + \frac{M_T}{2}\ln N \tag{3.4e}$$

$$\theta^* = \min_{M,\theta} S_{\text{BIC}} = \min_{M,\theta}\left(-\sum_{i=1}^{N}\ln p(A_{\text{i-row}}\theta - b_i|\theta, M) + \frac{M_T}{2}\ln N\right) \tag{3.4f}$$

$$\begin{bmatrix} \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{\tau} \\ -\frac{\alpha_z\beta_z}{\tau} & -\frac{\alpha_z}{\tau} \end{bmatrix} \begin{bmatrix} y - g \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{f_{target}}{\tau} \end{bmatrix} \tag{3.4g}$$

$$s_{1,2} = \frac{1}{2\tau}\left(-\alpha_z \pm \sqrt{\alpha_z^2 - 4\alpha_z\beta_z}\right) \tag{3.4h}$$

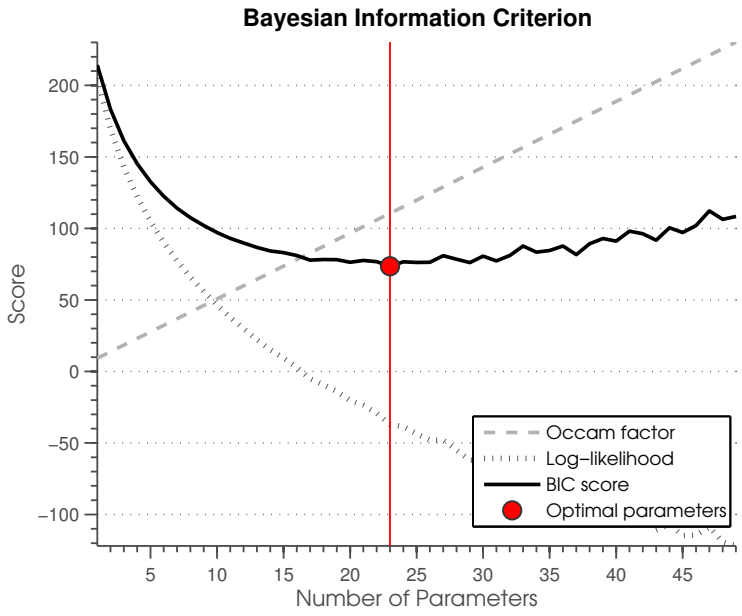


Figure 3.6: Plot of the Bayesian Information Criterion Score (black solid line), including Occam factor for penalization of model complexity (dashed gray line) and negative of the log likelihood of the conditional probability of the data given the parameters (gray dotted line).

Now that the unknown parameters have been estimated, equations 2.15a and 2.15b can be rewritten in matrix form, as in 3.4g and the poles of the state transition matrix can be computed as shown in equation 3.4h. The estimated poles give an approximation of the stability of HyQ. In the next section, the performance of this estimation at evaluating the robot stability will be presented.

4 Experiments and Results

"All that matters on the chessboard is good moves."

— Bobby Fischer, *World Chess Champion , 1943-2008*

In this chapter, some of the experiments and results, obtained in simulation will be presented.

4.1 An Optimization Example

In this section, an example of an optimization process for a walking trot at 1 [m/s] is presented. Figure 4.1 shows the algorithm convergence and variance for the optimization process, obtained from 6 learning curves.

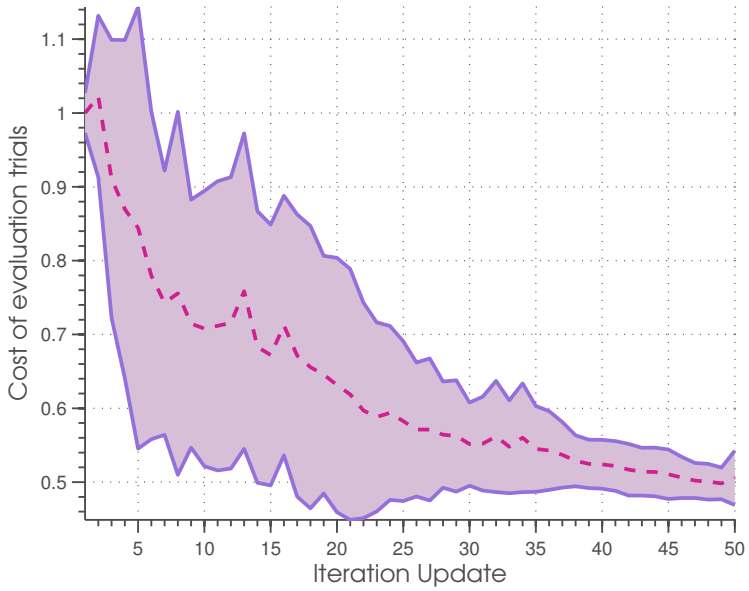


Figure 4.1: Algorithm convergence and variance, obtained from 6 learning curves. Each learning curve is composed of 50 iterations, where each iteration is composed of 8 rollouts and the best 6 rollouts are reused for the next iteration. The total of rollouts in the optimization process is 106 rollouts.

In the following, we will examine one of the learning curves (Figure 4.2). The initial parameters for the WCPG parameters and GAIN parameters can be seen in the following figures (GAIN parameters: Figure 4.3, 4.5; WCPG parameters: Figure 4.4). The cost function described in the last section (equations 3.2) is used in the following way. The

parameters of the WCPG are explored during the first 25 iteration updates, after this they are only very slightly changed.

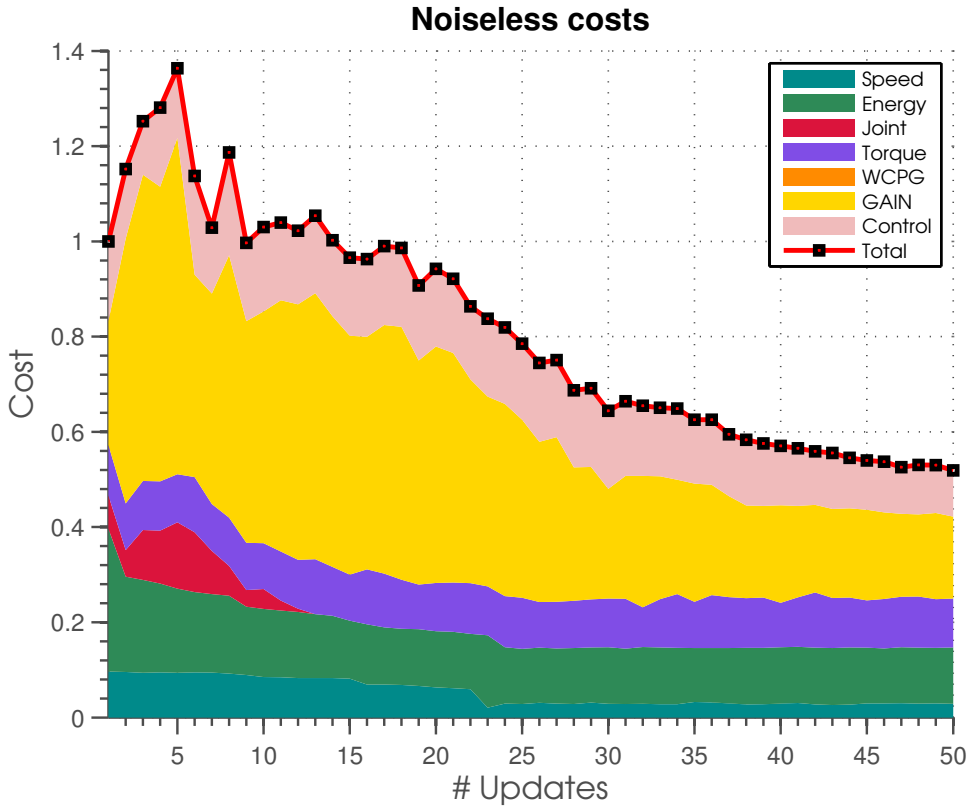


Figure 4.2: Example of Learning curve of Trotting Gait. This graph shows the contributions of the different objectives to the total cost. The first three areas from the bottom, blue, green and red represent the costs due to speed tracking error, energy efficiency and closeness to joint limits respectively. These costs, including penalization costs for high impedance gains (pink area), are used for the updates of the WCPG parameters. The purple area represents the cost due to feedforward torques, and the yellow area shows the cost due the the variance of the pitch-roll angles trajectory. The GAIN parameters are updated based on the sum of all the costs shown, as explained in (equations 3.2).

This time window has experimentally shown to be enough time to explore and find a set of WCPG parameters that produces a gait that accomplishes the main goals for this phase, namely, energy efficiency, good speed tracking and minimizes the cost due to closeness to joint limits. Figure 4.4 shows the evolution of the WCPG parameters during 20 updates, and it can be seen in Figure 4.2, that they have converged to a set of parameters where the energy efficiency cost (green area) and closeness to joint limits cost (red area) have been reduced. It can also be seen, that the speed tracking error cost has reduced (blue area), but the magnitude of the oscillations of the current from the desired speed make the cost to still remain high, this is improved by finely tuning

the impedance gains. Once the WCPG parameters have converged the impedance gains can be finely optimized as shown in Figure 4.3. Figure 4.3 shows the evolution of the impedance parameters at different number of iteration updates. For example, the purple line shows the gains after 20 updates, where the WCPG parameters are close to converge. It can be seen, that the impedance gains have a similar shape to the final results, but giving the algorithm the chance to finely tune these gains helps improving the speed tracking error cost (blue area), reduces the variance of the pitch-roll angles trajectory (yellow area) and reduces the cost due to high impedance gains (pink area), as can be seen in Figure 4.2.

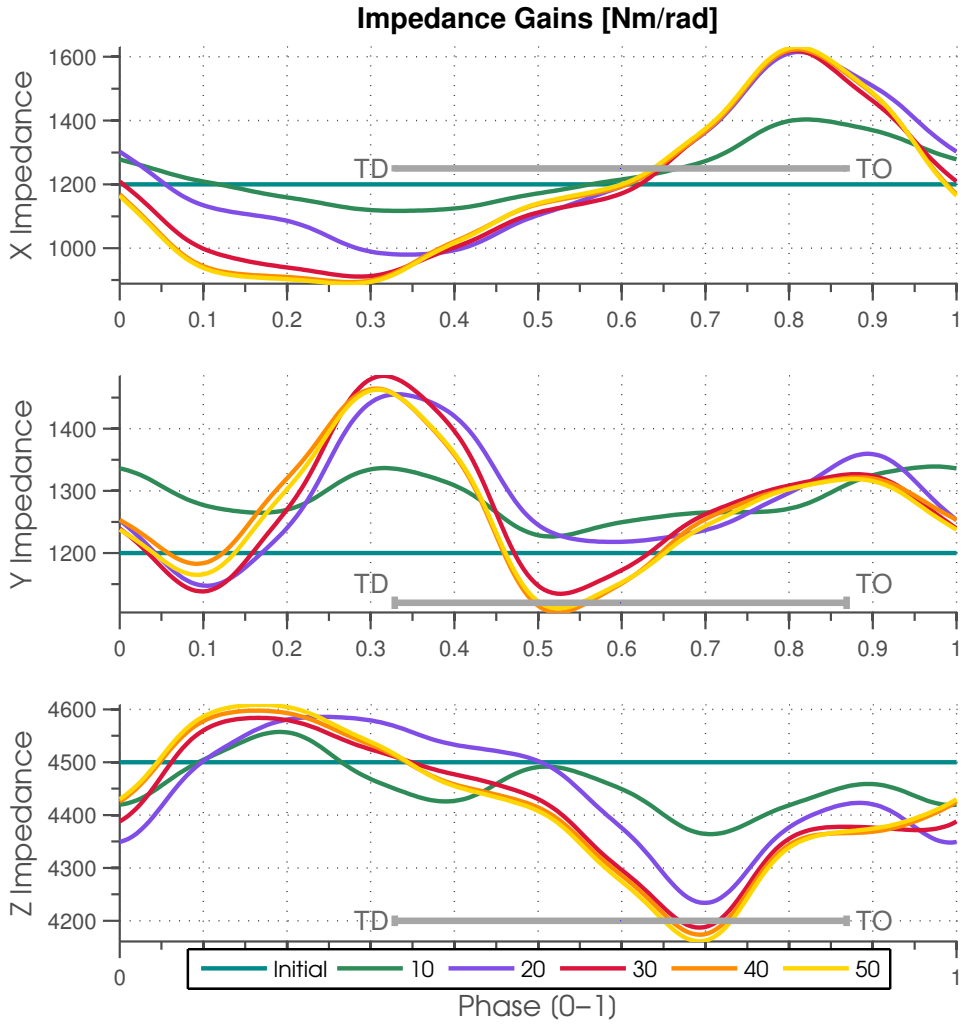


Figure 4.3: Impedance Gains and its evolution along the optimization

Impedance Gains are initialized at a constant value, the algorithm increases the gains until they achieve a low variance limit cycle for the roll and pitch trajectories and then, they are decreased, where possible, to reduce the cost due to the GAIN parameters.

The evolution of the Trunk stabilization parameters along the optimization are shown in Figure 4.5.

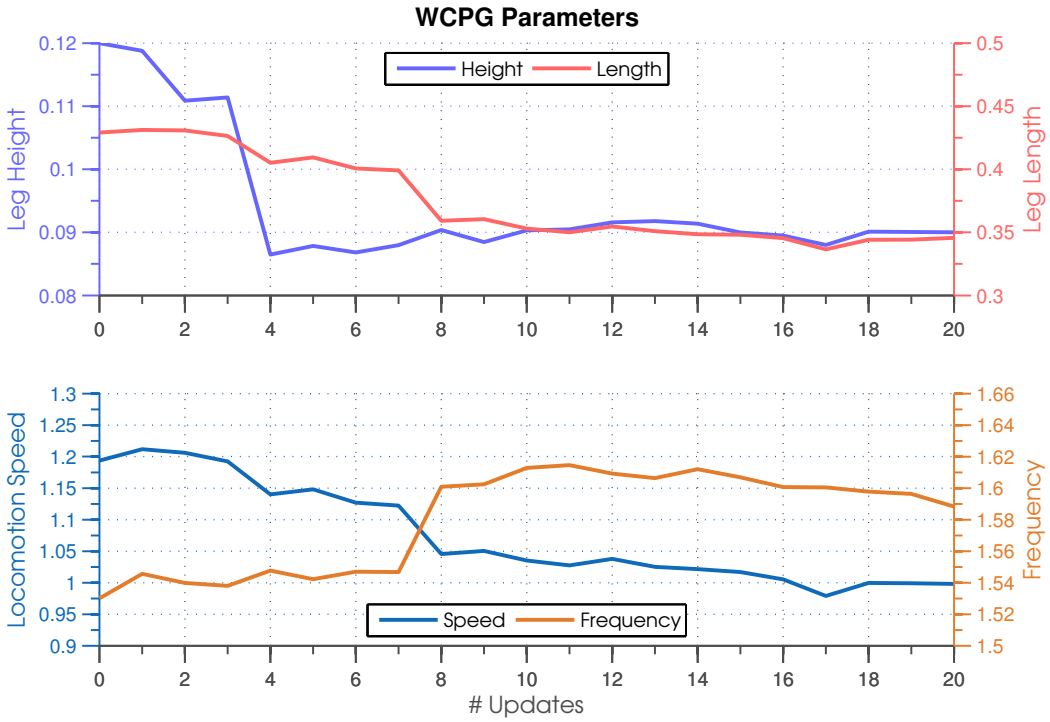


Figure 4.4: Evolution of WCPG parameters along the optimization

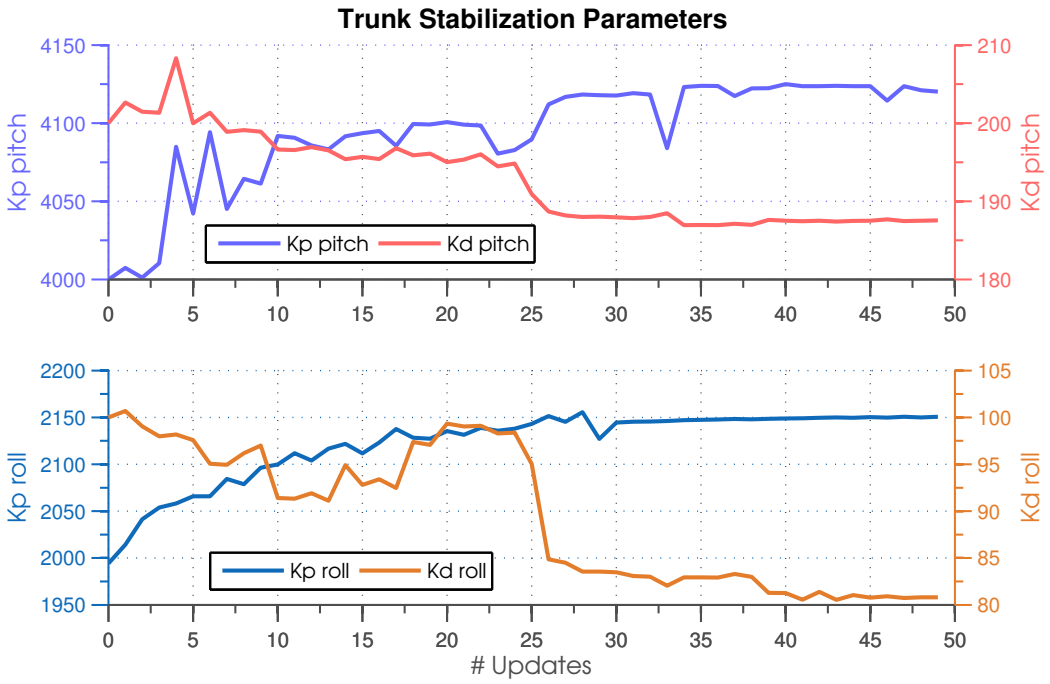


Figure 4.5: Evolution of Trunk Stabilization parameters along the optimization

Figure 4.6 shows how the robot learns a compliant policy. It reduces the stiffness during stance phase, so that when the leg makes a touch down, it can interact compliant enough with the environment. The effect of the compliance given by the policy can be seen in the trajectory tracking performance of one of the joints, as shown in Figure 4.6 (second plot).

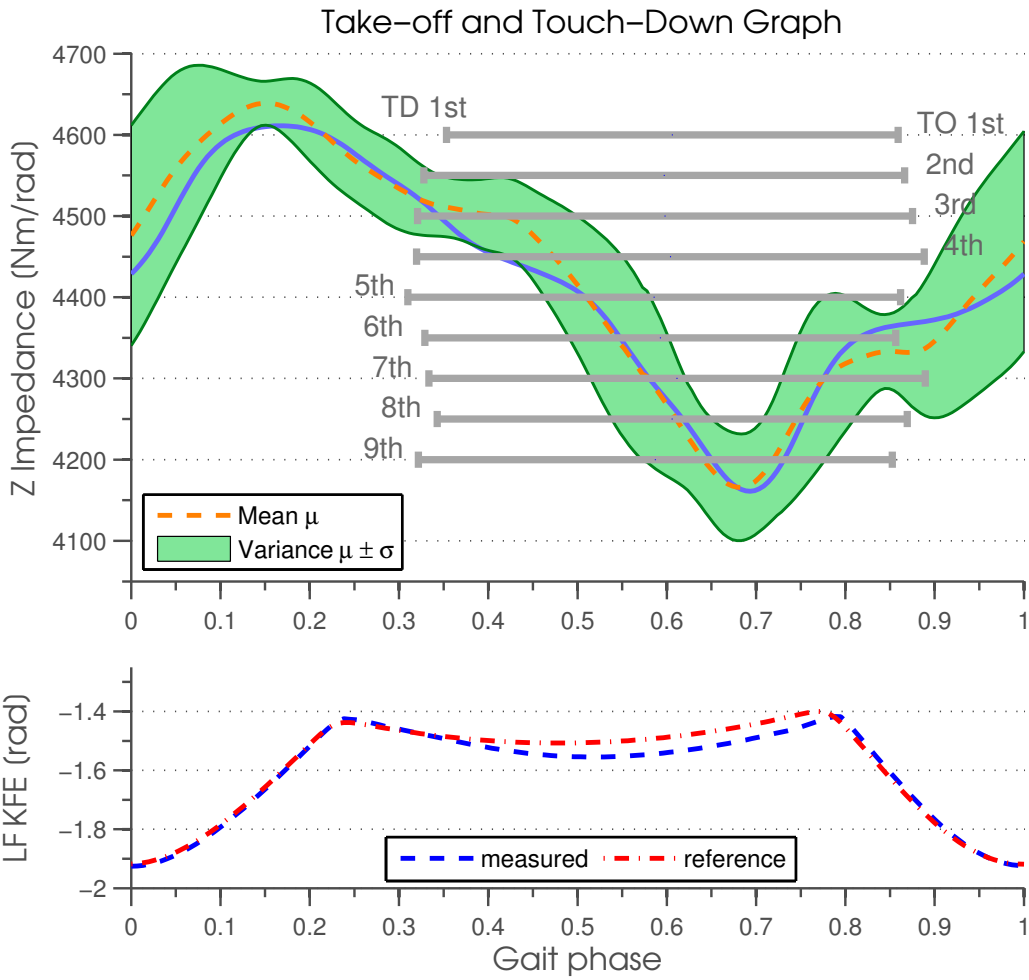


Figure 4.6: Graph of variable impedance at Take-off and Touch-down during an optimization experiment. The compliance of the robot can be seen in the reduced tracking performance of one of the joints.

4.2 Impedance results and Cost of Transport

In the last section, results for a single locomotion speed were shown. This section introduces more general results, for a wider range of locomotion speeds.

Figure 4.8 shows the learned variable impedance in the Z direction for different speeds. The impedance profiles were learned in simulation for several speeds (for example: 0.1, 0.3, 0.5, ..., 1.9 [m/s]) and then, they were generalized to a continuous surface by using

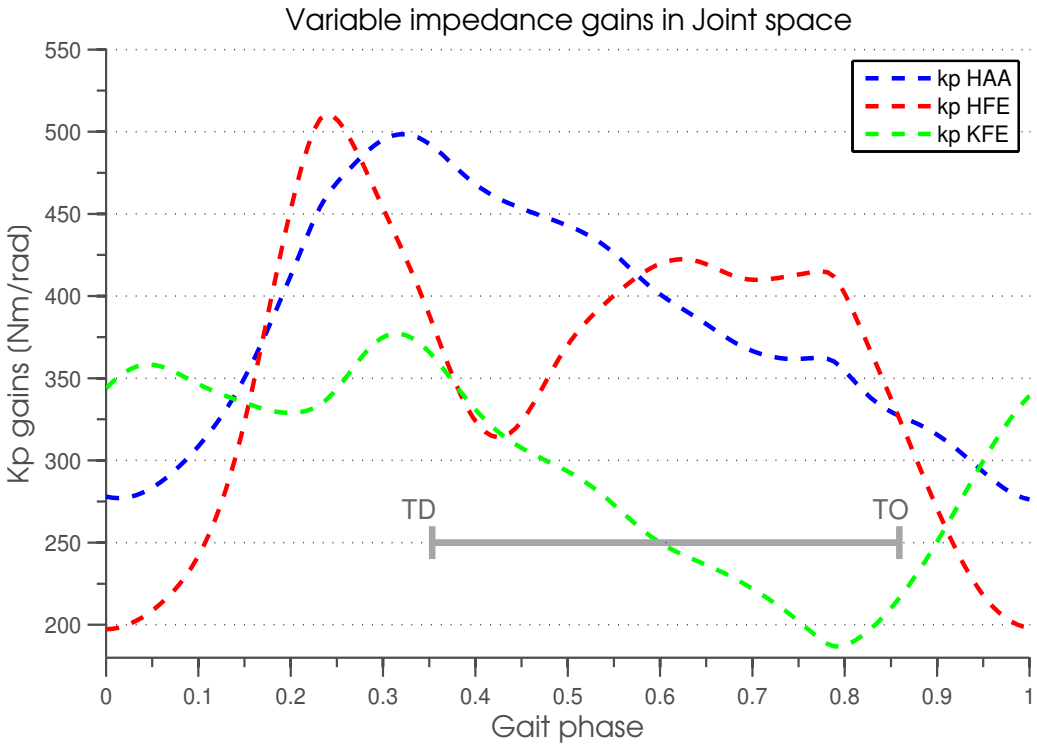


Figure 4.7: Variable impedance gains in Joint space. TD (touch down) and TO (take-off) allow to differentiate between stance and swing phases

function approximation using a Gaussian process. By observing the resulting graph, one can realize that the impedance profile's shape is similar at different speeds. This makes sense with the intuition that the robot softens when the leg is in contact with the ground, and stiffens up during swing phase for good trajectory tracking. One thing to notice is that the mean of the impedance profiles increases with the speed for speeds up to 1.5 m/s, and decreases after that for speeds up to 2 m/s.

An important concept to consider in the optimization of a locomotion gait, is the cost of transport or specific resistance, which is a dimensionless quantity that gives information about the energy efficiency of a mobile robot. The cost of transport allows to compare between different animals and locomotion modes. It is calculated as:

$$COT \triangleq \frac{W}{mgd} = \frac{P}{mgv}$$

where W is the energy that a system of mass m, under standard gravity g, needs to move a distance d; it can also be expressed in terms of power input P to move at constant speed v.

In our experiments, the cost of transport was determined for several speeds and using a time period of 10 seconds. Figure 4.9 shows the results. Two things to notice are that,

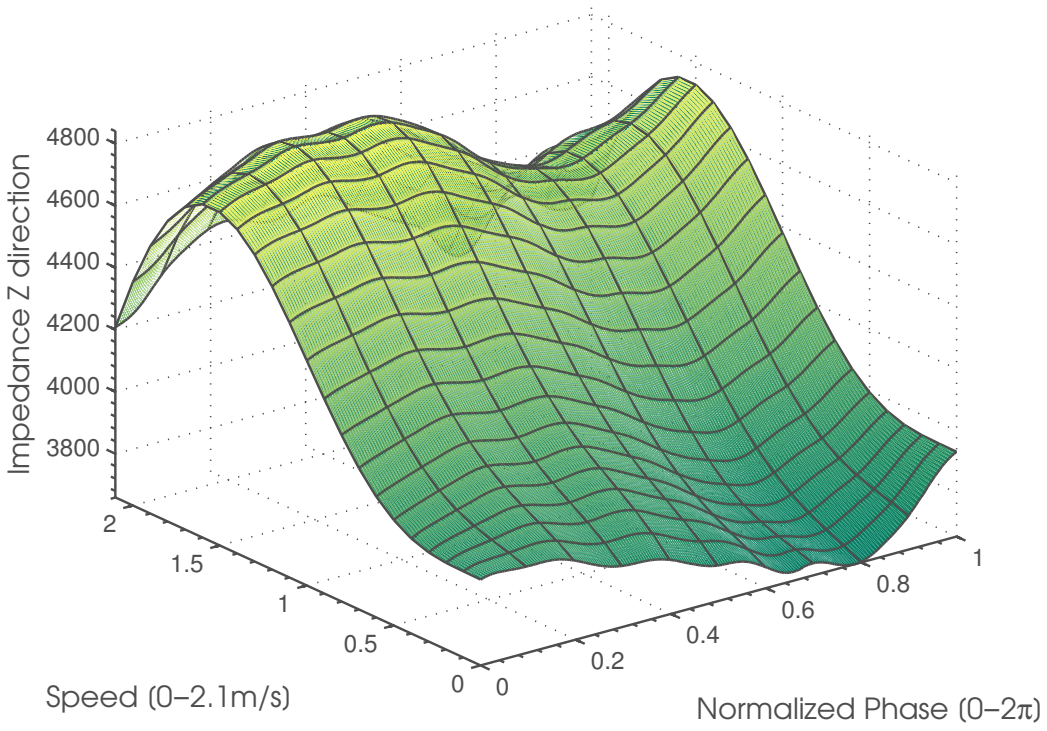


Figure 4.8: Impedance variation at different speeds.

first, for speeds between 0.5 and 1.5 m/s HyQ achieves its best performance in terms of energy efficiency; and second, around 1.8 m/s it is better to use a running trot than a walking trot in HyQ.

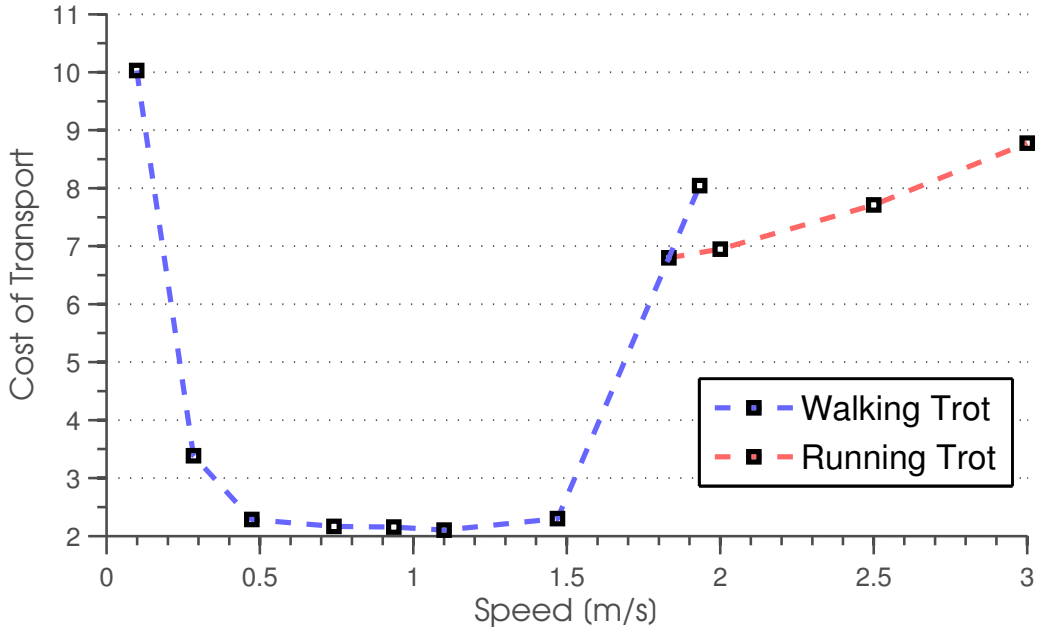


Figure 4.9: Cost of Transport for a Trotting gait.

4.3 Stability of Trotting Gait

The stability and robustness properties of a system quantify the ability of the system to cope with disturbances or perturbations. In simple systems, such as in linear systems, these properties can be easily determined using linear control theory tools. In more complex systems, such as in legged robots, estimating these properties is harder, because it involves a nonlinear dynamical system with high number of parameters.
In this project, a rhythmic control policy is used to approximate the dynamics of the roll angle of the robot (a periodic variable of the trotting gait), and in this way obtain an idea of the stability and robustness of the robot based on the poles of the second order system of the RCP. A RCP is the similar of a DMP, but used for cyclic systems. It encodes a demonstrated trajectory in terms of differential equations with well-defined attractor properties. By using the RCP to approximate the dynamics of the roll angle, it is possible to analyse in simulation, the maximum perturbation that the robot can stand before becoming unstable.

Figure 4.10: Estimation of poles of the Roll dynamics with RCP. For each lateral force applied to the robot, the poles estimated in several experiments are shown.

In this experiment, a trotting gait optimized for a speed of 0.2m/s at 1Hz, is analysed by applying a lateral perturbation during 1 second. In Figure 4.10, the poles of the second order system approximation for the roll angle are plotted for different lateral perturbation forces. In simulation, it could be tested that the estimated stability decreases as the

applied force increases, and the estimated stability also correlates well with the stability state of the robot (for example, robot has fallen or not). Therefore, it is possible to use these results as an idea of the robustness of the robot's gait and the maximum force that the robot can stand.

5 Conclusions

"When you see a good move, look for a better one."
— Emanuel Lasker, *German World Chess Champion , 1868-1941*

5.1 Summary

The purpose of this project was to implement a learning and adaptation layer over a parametrized gait generator for trotting on HyQ, a fully torque-controlled hydraulic quadruped robot designed for versatile movement. In this project, the parametrized gait generator (subject to optimization), was the Reactive Controller Framework, which generates elliptical trajectories for the feet based on Central Pattern Generators, and provides trunk stabilization control for stabilizing the attitude of the robot, among other functions.

Optimization of such a high dimensional problem cannot be performed by exhaustive search algorithms, therefore, the reinforcement learning algorithm PI^2 is used, due to its known capabilities of being able to handle high dimensional problems, fast convergence, and foundations on solid theoretical principles.

The cost function used for guiding the learning, was designed as simple as possible, but expressive enough in order to be able to optimize the most relevant parameters, namely the parameters of the gait generator (WCPG parameters) and the parameters that define the robot-environment interaction and trunk stabilization control (GAIN parameters).

An additional element of the framework is the use of adaptive frequency oscillators in order to synchronize desired impedance commands with the locomotion phase of the robot. This is an important element, in order to be able to vary the impedance gains of the robot, and in this way make the robot's trotting gait robuster.

One of the most important results is that the learning method was formulated in a principled way and has been able to optimize in simulation a trotting gait at different speeds, in terms of good speed tracking performance, energy efficiency, locomotion within the feet workspace, stability and robustness.

As a by-product, an idea of how to optimally select the number of basis functions used in a parametrized policy (for impedance gains, in our case) by means of the Bayesian Information Criterion Score and Gaussian Mixture Models has been presented. Finally, an idea for using Rhythmic Control Policies in a different way (characterizing stability of a nonlinear system) has been tested. Both ideas have proven to be useful and are

a first step in the direction of open research questions regarding optimal selection of the number of basis functions for a parametrized policy and determining stability in quadruped locomotion.

5.2 Conclusions

It has been shown that the optimization in simulation of a trotting gait, based on solid theoretical principles, making use of reinforcement learning by using Policy Improvements with Path Integrals Learning Algorithm (PI^2) and adaptive frequency oscillators, is highly effective.
The algorithm optimizes directly feedback terms by learning variable impedance schedules for the robot-environment interaction, and trunk stabilization parameters. It also learns indirectly feed-forward terms by optimizing the WCPG parameters that generate the desired feet elliptical trajectories. It has been shown that the algorithm has scaled very well to this very high dimensional problem, that optimizes the parameters for the entire locomotion cycle (stance and flight phase).
The learning algorithm has generated policies for different locomotion speeds, achieving a stable locomotion gait with limit cycle and an energy efficient locomotion frequency.
The issue of specifying a target impedance is not trivial, therefore learning is necessary. The learning algorithm has learned a variable impedance schedule, that gives the robot the compliance needed for the interaction with the environment. It provides enough stiffness during swing phase and compliance during stance phase, trading off in this way, the leg objectives of high performance trajectory tracking and robustness for the interaction with the environment.
The algorithm has not been tested in the real robot, therefore, the next step, in order to validate the results obtained in this project, will be to perform learning on the real robot. This will allow to push HyQ to its performance limits, taking into account also not modelled dynamics.

5.3 Future Work

First of all, it would be necessary to test the simulation results of this project on learning directly on HyQ, in order to evaluate the algorithm performance on the real robot.
In a next step, the feed-forward torques could be represented as a parametrized policy and directly optimized, such as the feedback gains were. Another possibility could be to try direct methods for optimizing feedforward commands and learning methods for learning feedback gains.
The energy efficiency could be better optimized by including hydraulic losses. In this way, it might be possible to find a new cost minimum.

A Appendix 1

"What I cannot create, I do not understand."

— Richard Feynman, *American physicist, 1918 - 1988*

A.1 Brownian Motion

In 1923, the American mathematician Norbert Wiener defined a Brownian Motion as a stochastic process $W(t)$ that satisfies the following assumptions [16]:

- *Independence:* $W(t+\Delta t) - W(t)$ is independent of $W(\tau)$ for all $\tau \leq t$.
- *Stationarity:* The distribution of $W(t+\Delta t) - W(t)$ does not depend on t.
- *Continuity:* $\lim_{\Delta t \to 0} \frac{P(|W(t+\Delta t) - W(t)| \geq \delta)}{\Delta t} = 0$ for all $\delta > 0$.

This definition implies that, if $W(t)$ is a Brownian motion, then $W(t) - W(0)$ is a normal random variable with mean μt and variance $\sigma^2 t$. Therefore, the density function of a Brownian motion can be written as:

$$f_{W(t)}(x) = \frac{1}{\sqrt{2\pi\sigma^2 t}} \exp - \frac{(x-\mu t)^2}{2\sigma^2 t}$$

A.2 Feynman-Kac Formula

The Feynman-Kac formula establishes a connection between the solution of parabolic partial differential equation (PDE) and its representation as stochastic differential equation (SDE) [67]. Given a PDE of the form:

$$\frac{\partial u}{\partial t}(x,t) + \mu(x,t)\frac{\partial u}{\partial x}(x,t) + \frac{1}{2}\sigma^2(x,t)\frac{\partial^2 u}{\partial x^2}(x,t) - V(x,t)u(x,t) + f(x,t) = 0$$

defined for all $x \in \mathbb{R}$ and $t \in [0,T]$, with final condition $u(x,T) = \Psi(x)$. The functions $\mu(x,t)$, $\sigma(x,t)$, $\Psi(x,t)$, $V(x,t)$ are known, T is the final time parameter, and $u(x,t)$ is the unknown function. In this case, the Feynman-Kac formula allows us to compute the solution of the PDE as a conditional expectation:

$$u(x,t) = \mathbb{E}^Q \left[\int_t^T \exp^{-\int_t^\tau V(X_\tau,\tau)\mathrm{d}\tau} f(X_r, r)\mathrm{d}r + \exp^{-\int_t^\tau V(X_\tau,\tau)\mathrm{d}\tau} \Psi(X_T) | X_t = x \right]$$

under the probability measure Q, and X defined by:

$$\mathrm{d}X = \mu(X,t)\mathrm{d}t + \sigma(X,t)\mathrm{d}W^Q$$

with initial condition $X(0) = x$.

A.2.1 Basic insights

This section provides basic intuition about the Backward Kolmogorov equation and its solution as an expectation using the Feynman-Kac Formula [16]. For the sake of simplicity, only the scalar case will be presented (the multidimensional case can be derived in the same way).

First of all, in the case that the dynamics of a system are given by the stochastic differential equation A.1a: if we are interested in an expected payoff Φ at the maturity time T, such as the one given by equation A.1b, then $u(x,t)$ solves equation A.1c.

Expected final-time payoff

$$dY = f(Y,t)dt + g(Y,t)dW \tag{A.1a}$$

$$u(x,t) = \mathbb{E}_{Y(t)=x}\left[\Phi(Y(T))\right] \tag{A.1b}$$

$$\frac{\partial u}{\partial t} + f(x,t)\frac{\partial u}{\partial x} + \frac{1}{2}g^2(x,t)\frac{\partial^2 u}{\partial x^2} = 0 \tag{A.1c}$$

for all: $t < T$ **with:** $u(x,T) = \Phi(x)$ (A.1d)

The proof goes as follows: for any function $\Phi(Y,t)$, Ito's lemma gives

$$d(\Phi(Y(t),t)) = \left(\frac{\partial \Phi}{\partial t} + f\frac{\partial \Phi}{\partial Y} + \frac{1}{2}g^2\frac{\partial^2 \Phi}{\partial Y^2}\right)dt + g\frac{\partial \Phi}{\partial Y}dW$$

Then by making $\Phi = u$ and integrating , we get to the following equation

$$u(Y(T),T) - u(Y(t),t) = \int_t^T \left(\frac{\partial \Phi}{\partial t} + f\frac{\partial \Phi}{\partial Y} + \frac{1}{2}g^2\frac{\partial^2 \Phi}{\partial Y^2}\right)ds + \int_t^T g\frac{\partial \Phi}{\partial Y}dW$$

By taking the expectation of this equation and by using equation A.1c, the right-half side of this equation drops to zero, and we are left with:

$$\mathbb{E}_{Y(t)=x}\left[\Phi(Y(T))\right] = u(x,t)$$

which concludes the proof.

For the same dynamical system, if we are interested in an expected discounted payoff Φ at the maturity time T, such as the one given by equation A.2b, then $u(x,t)$ solves equation A.2c.

Expected discounted final-time payoff

$$dY = f(Y,t)dt + g(Y,t)dW \tag{A.2a}$$

$$u(x,t) = \mathbb{E}_{Y(t)=x}\left[\Phi(Y(T))\exp\left(-\int_t^T V(Y(s),s)ds\right)\right] \tag{A.2b}$$

$$\frac{\partial u}{\partial t} + f(x,t)\frac{\partial u}{\partial x} + \frac{1}{2}g^2(x,t)\frac{\partial^2 u}{\partial x^2} - V(x,t)u = 0 \tag{A.2c}$$

for all: $t < T$ **with:** $u(x,T) = \Phi(x)$ (A.2d)

In this case the proof goes as follows:

$$d\left[\Phi(Y(t))\exp\left(-\int_t^r V(Y(s),s)ds\right)\right] = d\left[z_1 z_2\right] = z_1 dz_2 + z_2 dz_1 + dz_1 dz_2$$

$$dz_1 = -\exp\left(-\int_t^r V(Y(s),s)ds\right)V(Y(s),s)ds = -z_1 V(Y(s),s)ds$$

$$dz_2 = d(\Phi(Y(t)) = \left(\frac{\partial\Phi}{\partial t} + f\frac{\partial\Phi}{\partial Y} + \frac{1}{2}g^2\frac{\partial^2\Phi}{\partial Y^2}\right)dt + g\frac{\partial\Phi}{\partial Y}dW$$

Then by making $\Phi = u$, we get to the following result

$$d\left[u(Y(t),t)\exp\left(-\int_t^r V(Y(s),s)ds\right)\right] = \exp\left(-\int_t^r V(Y(s),s)ds\right)g\frac{\partial u}{\partial Y}dW$$

By integrating and taking the expectation of this equation, the right-half side of this equation drops to zero, and we are left with:

$$\mathbb{E}_{Y(t)=x}\left[\Phi(Y(T))\exp\left(-\int_t^T V(Y(s),s)ds\right)\right] = u(x,t)$$

and this concludes the proof.
Finally, if we are interested in a running payoff for a given function Ψ, such as the one given by equation A.3b, then $u(x,t)$ solves equation A.3c.

Expected running payoff

$$dY = f(Y,t)dt + g(Y,t)dW \tag{A.3a}$$

$$u(x,t) = \mathbb{E}_{Y(t)=x}\left[\int_t^T \Psi(Y(s),s)ds\right] \tag{A.3b}$$

$$\frac{\partial u}{\partial t} + f(x,t)\frac{\partial u}{\partial x} + \frac{1}{2}g^2(x,t)\frac{\partial^2 u}{\partial x^2} + \Psi(x,t) = 0 \tag{A.3c}$$

for all: $t < T$ **with:** $u(x,T) = 0$ (A.3d)

The proof is not different from the other cases and goes as follows:

$$\Phi(Y(t),t) = \int_t^T \Psi(Y(s),s)ds$$
$$d(\Phi(Y(t),t)) = \left(\frac{\partial\Phi}{\partial t} + f\frac{\partial\Phi}{\partial Y} + \frac{1}{2}g^2\frac{\partial^2\Phi}{\partial Y^2}\right)dt + g\frac{\partial\Phi}{\partial Y}dW$$

Then by making $\Phi = u$ and using equation A.3c, we get to the following equation

$$du(Y(t),t) = -\Psi(Y(t),t)dt + g\frac{\partial u}{\partial Y}dW$$

By integrating and taking the expectation of this equation, we are left with:

$$\mathbb{E}_{Y(t)=x}[u(Y(s),s)] - u(x,t) = \mathbb{E}_{Y(t)=x}\left[-\int_t^T \Psi(Y(s),s)ds\right]$$
$$u(x,t) = \mathbb{E}_{Y(t)=x}\left[\int_t^T \Psi(Y(s),s)ds\right]$$

Now we can interpret in a simple way, how PI^2 assigns the payoffs. In PI^2, we use the second case, where we are interested in an expected discounted payoff, where the final-time payoff $\Phi(Y(T))$ is a function of the final condition of the system and its weights are given by the integral of the immediate-time costs.
The Feynman-Kac Formula also allows to compute solutions for combinations of the payoffs presented before.

B Appendix 2

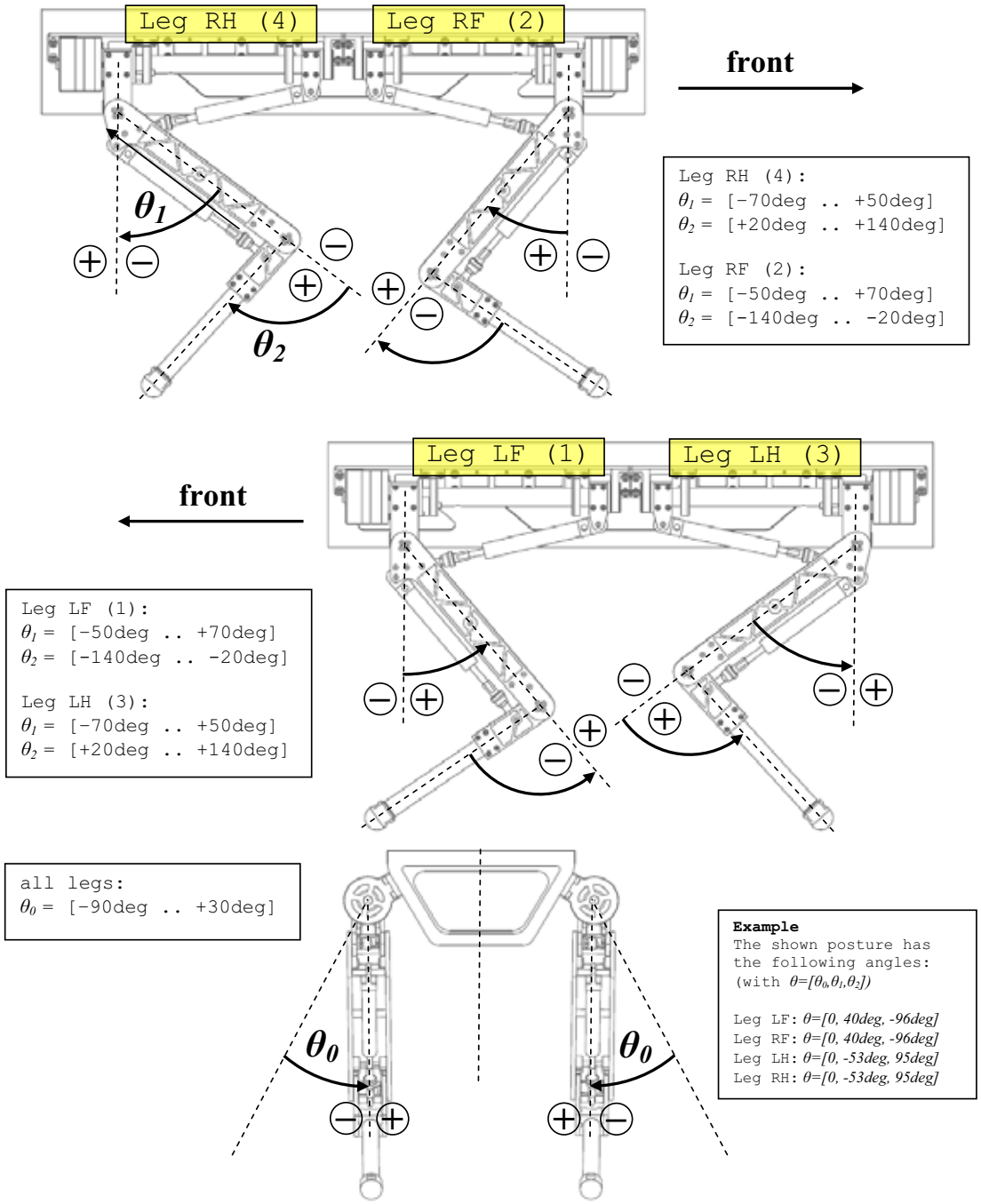
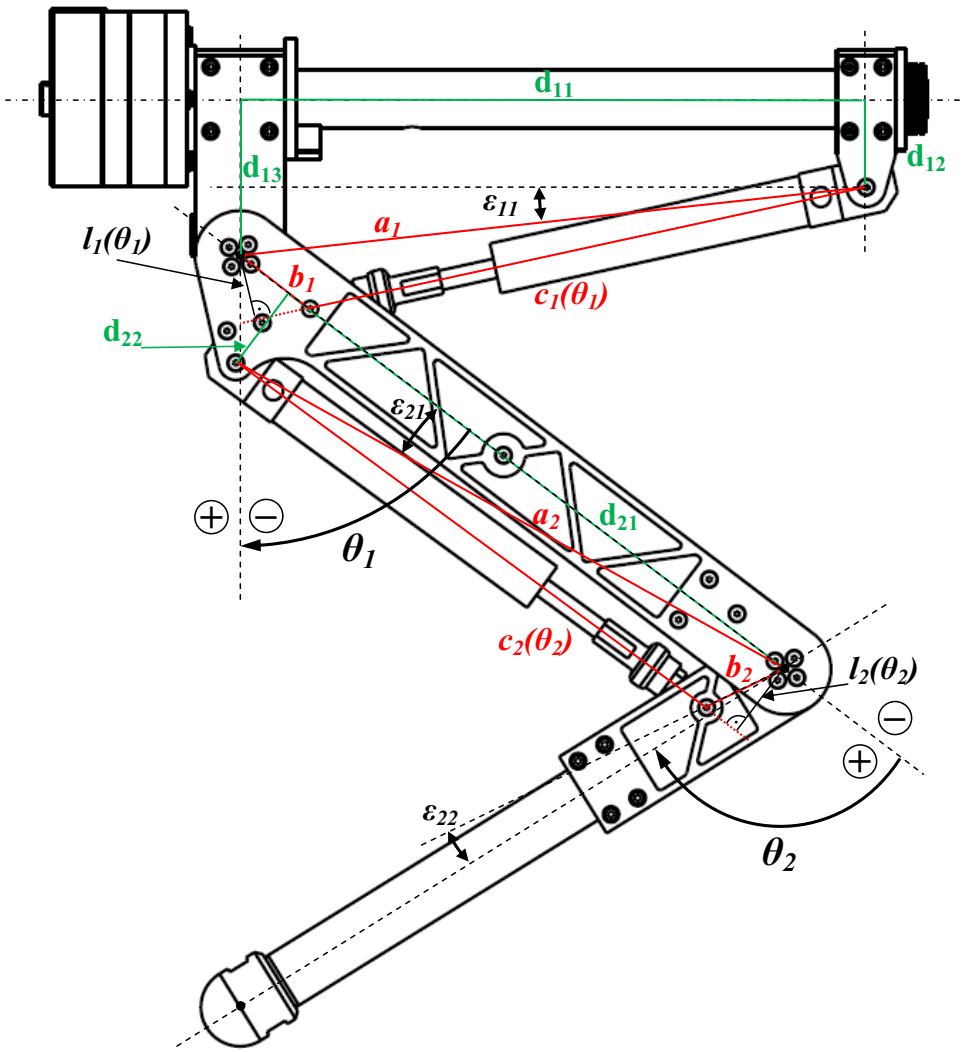


Figure B.1: Definition of the joint angles in *HyQ*. Taken from [60]

C Appendix 3



$a_1 = 0.3219\ m = \sqrt{d_{11}^2+(d_{13}-d_{12})^2}$ $b_1 = 0.045\ m$ $\varepsilon_{11} = 6.24\ deg = \arctan((d_{13}-d_{12})/d_{11})$ $(\varepsilon_{12} = 0\ deg)$ $c_1(\theta_1) = \sqrt{a_1^2 + b_1^2 - 2 a_1 b_1 \cos(\pi/2+\theta_1+\varepsilon_{11})}$ $l_1(\theta_1) = a_1 \sin(\arccos((a_1^2+c_1(\theta_1)^2-b_1^2)/(2 a_1 c_1(\theta_1))))$	$d_{11} = 0.32\ m$ $d_{12} = 0.045\ m$ $d_{13} = 0.08\ m$
$a_2 = 0.3218\ m = \sqrt{d_{21}^2+d_{22}^2}$ $b_2 = 0.045\ m$ $\varepsilon_{21} = 8.04\ deg = \arctan(d_{22}/d_{21})$ $\varepsilon_{22} = 6.0\ deg$ $c_2(\theta_2) = \sqrt{a_2^2 + b_2^2 - 2 a_2 b_2 \cos(\pi-\theta_2-\varepsilon_{21}-\varepsilon_{22})}$ $l_2(\theta_2) = a_2 \sin(\arccos((a_2^2+c_2(\theta_2)^2-b_2^2)/(2 a_2 c_2(\theta_2))))$	$d_{21} = 0.3186\ m$ $d_{22} = 0.045\ m$

Figure C.1: Definition of leg geometry in *HyQ*. Taken from [60]

Bibliography

[1] A. Abourachid, M. Herbin, R. Hackert, L. Maes, and V. Martin. Experimental study of coordination patterns during unsteady locomotion in mammals. *Journal of Experimental Biology*, 210(2):366, 2007.

[2] R. McM. Alexander. The gaits of bipedal and quadrupedal animals. *The International Journal of Robotics Research*, 2:49–59, 1984.

[3] Victor Barasuol, Jonas Buchli, Claudio Semini, Marco Frigerio, Edson De Pieri, and Darwin Caldwell. A reactive controller framework for quadrupedal locomotion on challenging terrain. In *ICRA*, pages 2554–2561. IEEE, 2013.

[4] Victor Barasuol, VictorJukiano De Negri, and Edson Roberto De Pieri. Wcpg: A central pattern generator for legged robots based on workspace intentions. In *ASME*, 2011.

[5] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal. Learning variable impedance control. *International Journal of Robotics Research*, 2011.

[6] S. Calinon. *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press, 2009. EPFL Press ISBN 978-2-940222-31-5, CRC Press ISBN 978-1-4398-0867-2.

[7] S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 37(2):286–298, 2007.

[8] Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel van de Panne. Locomotion skills for simulated quadrupeds. *ACM Transactions on Graphics*, 30(4):Article TBD, 2011.

[9] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000.

[10] T.E. Duncan, H.F. Chen, and B. Pasik-Duncan. A kiefer-wolfowitz algorithm with randomized differences. *IEEE Transactions on Automatic Control*, 44:442–453, 1999.

[11] Peter Fankhauser. Optimizing robotic single legged locomotion with reinforcement learning. Master's thesis, Swiss Federal Institute of Technology Zurich, Switzerland, 2012.

[12] Peter Fankhauser, Marco Hutter, Christian Gehring, Michael Bloesch, Mark A. Hoepflinger, and Roland Siegwart. Reinforcement learning of single legged locomotion. *Neural Computation*, 10(10):2047–2084, 1997.

[13] Michele Focchi. *Hydraulic compliance control of the quadruped robot HyQ*. PhD thesis, Istituto Italiano di Tecnologia Genoa, 2013.

[14] Michele Focchi. *Strategies to improve impedance control performance of a quadruped robot*. PhD thesis, Istituto Italiano di Tecnologia Genoa, 2013.

[15] A. Gams, A. Ijspeert, S. Schaal, and J. Lenarčič. On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Autonomous Robots*, 1(1):3–23, 2009.

[16] H. P. Geering, G. Dondi, F. Herzog, and S. Keel. Stochastic systems, April 2011.

[17] Christian Gehring. c-optimizer, August 2013.

[18] Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics*, 32(6), 2013.

[19] Erico Guizzo. Hyq quadruped robot from italy can trot, kick. http://spectrum.ieee.org/automaton/robotics/industrial-robots/hyq-quadruped-robothttp://spectrum.ieee.org/automaton/robotics/industrial-robots/hyq-quadruped-robot, 2011. [Online; accessed 31-October-2013].

[20] I. Havoutis, C. Semini, J. Buchli, and D. Caldwell. Quadrupedal trotting with active compliance. In *Mechatronics (ICM), 2013 IEEE International Conference on*, pages 610–616, 2013.

[21] N. C. Heglund and C. R. Taylor. Speed, stride frequency and energy cost per stride: How do they change with body size and gait? *Journal of Experimental Biology*, 97:1–21, 1988.

[22] Kazuo Hirai, Masato Hirose, Yuji Haikawa, and Toru Takenaka. The development of honda humanoid robot. In *ICRA*, pages 1321–1326. IEEE Computer Society, 1998.

[23] Donald F. Hoyt and C. Richard Taylor. Gait and the energetics of locomotion in horses. *Nature*, 292(5820):239–240, jul 1981.

[24] Marco Hutter. *StarlETH - Design and Control of Legged Robots with Compliant Actuation*. PhD thesis, Swiss Federal Institute of Technology Zurich, 2013.

[25] IEEE. *Stochastic optimization with inequality constraints using simultaneous perturbations and penalty functions*, volume 4, 2003.

[26] A. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25:328–373, 2013.

[27] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *In Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS2002*, pages 958–963, 2002.

[28] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *In IEEE International Conference on Robotics and Automation (ICRA2002*, pages 1398–1403, 2002.

[29] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems*, pages 1523–1530. MIT Press, 2003.

[30] G. Jaegger, Marcellin Little D, and D. Levine. Reliability of goniometry in labrador retrievers. *AJVR*, 63:979–986, 2002.

[31] Nate Kohl and Peter Stone. Machine learning for fast quadrupedal locomotion. In *The Nineteenth National Conference on Artificial Intelligence*, 2004.

[32] Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

[33] John Mathews and Kurtis Fink. *Numerical Methods*. Prentice-Hall Inc., 2004.

[34] L. McDowell. *The Dog in action*. Orange Judd Publishing Company Inc., 1950.

[35] S. Meek, J. Kim, and M. Anderson. Stability of a trotting quadruped robot with passive, underactuated legs. In *IEEE International Conference on Robotics and Automation ICRA*, 2008.

[36] M. Mistry, J. Buchli, and S. Schaal. Inverse dynamics control of floating base systems using orthogonal decomposition. In *Robotics and Automation (ICRA), 2010 IEEE*, pages 3406–3412, 2010.

[37] Tom M. Mitchell. *Machine learning*. McGraw-Hill, New York, NY, 1997.

[38] Igor Mordatch, Emanuel Todorov, and Zoran Popovic. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.*, 31(4):43, 2012.

[39] Jun Morimoto and Christopher G. Atkeson. Nonparametric representation of an approximated poincare map for learning biped locomotion. *Auton. Robots*, 27(2):131–144, 2009.

[40] Jun Nakanishi, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47:79–91, 2004.

[41] Andrew Y. Ng and Michael I. Jordan. Pegasus: A policy search method for large mdps and pomdps. *CoRR*, abs/1301.3878, 2013.

[42] Andrew Y. Ng, H. Jin Kim, Michael I. Jordan, and Shankar Sastry. Inverted autonomous helicopter flight via reinforcement learning. In *In International Symposium on Experimental Robotics*. MIT Press, 2004.

[43] J. Gordon Nichol, Surya P. N. Singh, Kenneth J. Waldron, Luther R. Palmer, and David E. Orin. System design of a quadrupedal galloping machine. *I. J. Robotic Res.*, 23(10-11):1013–1027, 2004.

[44] Nanua P. and Waldron KJ. Energy comparison between trot , bound and gallop using a simple model. *J. Biomech Eng.*, pages 466–73, 1995.

[45] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.

[46] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *International Journal of Robotics Research*, 33(1):69 – 81, 2013.

[47] Ioannis Poulakakis, James Andrew Smith, and Martin Buehler. Modeling and experiments of untethered quadrupedal running with a bounding gait: The scout ii robot. *I. J. Robotic Res.*, 24:239–256, 2005.

[48] Jerry Pratt. Fastrunner a robot ostrich. http://www.ihmc.us/groups/fastrunner/, 2013. [Online; accessed 28-December-2013].

[49] Marc Raibert. Bigdog, the rough-terrain quadruped robot. In Myung J. Chung, editor, *Proceedings of the 17th IFAC World Congress, 2008*, volume 17, 2008.

[50] Marc Raibert. Bigdog, the rough-terrain quadruped robot. In *Proceedings of the 17th IFAC World Congress, 2008*, volume 17, 2008.

[51] Marc H. Raibert. *Legged Robots That Balance*. Massachusetts Institute of Technology, Cambridge, MA, USA, 1986.

[52] Mark Raibert. Wildcat quadruped robot - boston dynamics. http://spectrum.ieee.org/automaton/robotics/military-robots/whoa-boston-dynamics-announces-new-wildcat-quadruped, 2013. [Online; accessed 28-December-2013].

[53] M.H. Raibert, M. Chepponis, and H. Benjamin Brown. Running on four legs as though they were one. *IEEE Journal of Robotics and Automation*, RA-2(2):70–82, June 1986.

[54] Ludovic Righetti. *Control of legged locomotion using dynamical systems: Design methods and adaptive frequency oscillators*. PhD thesis, Ecole Polytechnique Federale de Lausanne, 2008.

[55] Ludovic Righetti, Jonas Buchli, and Auke Jan Ijspeert. Dynamic hebbian learning in adaptive frequency oscillators. *Physica D: Nonlinear Phenomena*, 216(2):269 – 281, 2006.

[56] Eric Rombokas, Evangelos Theodorou, Mark Malhotra, Emo Todorov, and Yoky Matsuoka. Tendon-driven control of biomechanical and robotic systems: A path integral reinforcement learning approach. In *ICRA*, pages 208–214. IEEE, 2012.

[57] Manish Saggar, Thomas D'Silva, Nate Kohl, and Peter Stone. Autonomous learning of stable quadruped locomotion. In Gerhard Lakemeyer, Elizabeth Sklar, Domenico Sorenti, and Tomoichi Takahashi, editors, *RoboCup-2006: Robot Soccer World Cup X*, volume 4434 of *Lecture Notes in Artificial Intelligence*, pages 98–109. Springer Verlag, Berlin, 2007.

[58] Stefan Schaal. The sl simulation and real-time control software package. Computational Learning and Motor Control Laboratory, University of Southern California, March 2006.

[59] Stefan Schaal and Christopher G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(10):2047–2084, 1997.

[60] Claudio Semini. *HyQ - Design and Development of a Hydraulically Actuated Quadruped Robot*. PhD thesis, Istituto Italiano di tecnologia Genoa, 2010.

[61] Claudio Semini, Nikos G. Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin G. Caldwell. Design of hyq - a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers,Journal of Systems and Control Engineering.*, pages 831–849, August 2011.

[62] James. C. Spall. Adaptive stochastic approximation by the simultaneous perturbation method. *Automatic Control, IEEE Transactions on*, 45(10):1839–1853, 2000.

[63] James C. Spall and Senior Member. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37:332–341, 1992.

[64] Freek Stulp, Evangelos Theodorou, and Stefan Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *IEEE Transactions on Robotics*, 28(6):1360–1370, 2012.

[65] Russ Tedrake, Teresa Weirui Zhang, and H. Sebastian Seung. Learning to walk in 20 minutes. *Neural Computation*, 10(10):2047–2084, 1997.

[66] Wikipedia the free encyclopedia. Eadweard muybridge, February 2014.

[67] Wikipedia the free encyclopedia. Feynman-kac formula, February 2014.

[68] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11:3137–3181, 2010.

[69] Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8*, pages 514–520. MIT press, 1996.

[70] X. Zhang, H. Zheng, X. Guan, Z. Cheng, and L. Zhao. A biologically inspired quadruped robot structure and control. *IEEE Robotics and Biomimetics*, 2005.