Developing Controllers for Biped Humanoid Locomotion

Yariv Bachar

Master of Science

School of Informatics

University of Edinburgh

2004

Abstract

This dissertation aspires to intensify the humanoid robots' mobility, particularly, their bipedal dynamic stability. In principle, humanoid dynamics have a highly nonlinear nature, suggesting that closed-form solutions are insufficient. Furthermore, the uncertainty of disturbances bound with the environment amplifies this complex issue. The key aspect is how to manage the large number of degrees-of-freedom entailed by such systems. In effect, the humanoid's complex kinematic structure yields a critical increase to the controller's computational cost. Another aspect is how the control should manipulate the robot's joints to accomplish desired balanced in the environment affected by reaction forces and moments. Albeit many studies have tried to deal with these issues, yet the silver bullet solution has not been discovered, holding back the humanoids from accomplishing human expectations.

This thesis introduces the concept of the ZMP Jacobian which approximately represents a relationship between the Zero Moment Point (ZMP), the prominent dynamic stability criterion, and the humanoid's posture. Effectively, this Jacobian maps whole-body joint angle velocities to ZMP velocity in Cartesian space. Using this concept, a method which adjusts predefined motion trajectories to enhance performance and dynamic stability during bipedal locomotion, is presented and termed – the *ZMP Jacobian Counterbalance*.

The method is then verified by various simulated experiments on a walking, 22 degrees-of-freedom humanoid, using simulation software with accurate physics. The obtained results prove that the method developed in this dissertation is capable of improving the dynamic balance of humanoid bipedal locomotion. At the end, a discussion of the limitations, significance, and future development to the method is presented, suggesting that humanoid locomotion research has still many open interesting issues to be tackled.

Acknowledgements

First, I would like to thank my supervisor, Dr. Sethu Vijayakumar, for his exceptional patience and understanding with my quirks, elongated e-mails, and questions. His astuteness and experience enabled me to see many of the obstacles which arose from this research in a constructive and positive perspective.

I am also very grateful to Dr. Olivier Michel, the founder of Cyberbotics Ltd. and creator of Webots from the Biologically Inspired Research Group (BIRG) in EPFL, Lausanne, Switzerland, for his continuous advice and generous assistance with numerous difficulties I encountered throughout the development of this project. In addition, I would like to thank other members of BIRG, namely Pascal Cominoli and Jean-Philippe Egger for their willingness to help and collaborate with this study.

I thank other members of the Statistical Machine Learning and Motor Control (SLMC) group for their helpful discussions and listening, especially my friends, Timothy Rost and Alex Arthur, for sharing their motion planning for the humanoid model, and Marc Toussaint for his guidance with ODE.

I would like to express my love and gratitude to my soul mate, Karen Moore, with her endless love, support and inspiration, which kept me going, over the past year. In addition, I am indebted to all my friends for their motivation and encouragement.

Finally, my most profound thanks are to my parents, Shmuel and Shlomit, and my sister Anat, for their love and understanding throughout the years.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Yariv Bachar*)

Table of Contents

1 Introduction 1

1.1 Background 1

1.2 Biped Locomotion Control 3

1.2.1 Contributing Research Fields to Biped Locomotion 3
1.2.2 Research Motivation for Biped Locomotion 5
1.2.3 Biped Locomotion Principles 6
1.2.4 Review of Biped Locomotion Control Methods 8

1.3 Research Goals 15

1.4 Thesis Outline 16

2 The Zero-Moment Point 18

2.1 Definition 18

2.2 The ZMP Concept 20

2.2.1 The ZMP's relation to the Support Polygon 23
2.2.2 Various ZMP Interpretations 24
2.2.3 ZMP and CoP 25

2.3 Applicability of the ZMP 27

2.3.1 Offline ZMP Based Motion Planning 27
2.3.2 Online ZMP Based Motion Modification 28
2.3.3 Input ZMP Based Manipulation 29

2.4 ZMP Variants and Extensions 30

2.5 Derivation of the ZMP 33

3 Simulation Environment 37

3.1 Real vs. Simulated Robots 37

3.2 Webots and Environment Implementations 39

3.2.1 Overview of Webots 40
3.2.2 Robot Control and Environment Behaviour 42
3.2.3 Implementing Customised Physics 44

3.3 The Robot Model 47

3.4 Sony QRIO-like Specification 48

4 The ZMP Jacobian Counterbalance 52

4.1 Introduction 52

4.2 The ZMP Jacobian Concept 54

4.3 The Humanoid Motion Generation 55

4.3.1 Motion Generation Strategies 56
4.3.2 Intuitive Motion Trajectories Generation 57
4.3.3 Trajectory Interpolation 58

4.4 ZMP Jacobian Counterbalance Motion Control 60

4.4.1 Computing the ZMP Jacobian Mapping 62
4.4.2 Computing the Correction Vector 63
4.4.3 Convergence to Corrected Motion Trajectories 64

4.5 Summary 65

5 Experimental Results 67

5.1 Introduction 67

5.2 Experiment 1: Counterbalance using Torso Joints 68

5.3 Experiment 2: Counterbalance using Torso and Hips Joints 74

5.4 Experiment 3: Adjusting Trajectories for Slope Walk 77

5.5 Experiment 4: Counterbalance under External Disturbance 81

6 Discussion 87

6.1 Introduction 87

6.2 Contribution and Significance 87

6.3 Known Problems and Limitations 88

6.4 Knowledge Gained in the Scope of this Thesis 90

7 Conclusion 93

7.1 Summary 93

7.2 Future Development 94

7.3 Conclusions 96

Bibliography 97

A Applying Kinematics using Denavit-Hartenberg Convention 105

A.1 Background 105

A.2 The Denavit-Hartenberg Notation 107

A.2.1 Algorithm 109

A.3 Applying D-H Convention to Fujitsu's HOAP-2 110

B Simple Trajectory Smoothing 116

B.1 Problem Definition 116

B.2 Implementation 117

B.2.1 Performing Interpolation 119
B.2.2 Extending to Whole-Body Interpolation 119
B.2.3 Adjusting Interpolations to Fit Corrections 121

C Computing Matrix Pseudo-Inverse using SVD 124

C.1 Background 124

C.1.1 The Moore-Penrose Pseudo-Inverse 124
C.1.2 Singular Value Decomposition of a Matrix 125

C.2 Algorithm 126

List of Figures

Figure 1.1: Research fields contributing to biped locomotion 4

Figure 1.2: Biped system with kinematics chains set 7

Figure 1.3: Categorisation of Biped Locomotion Control. 9

Figure 2.1: Zero-Moment Point original definition 18

Figure 2.2: Vukobratovic et al. Biped Shuffle Kinematic Model 19

Figure 2.3: Forces and Moments acting on the biped foot 20

Figure 2.4: The support polygon in three typical cases 22

Figure 2.5: Three typical cases depicting the relationship between ZMP and CoP ... 26

Figure 2.6: The Enhanced ZMP concept 31

Figure 2.7: The Virtual Horizontal Plane (VHP) concept 32

Figure 2.8: Definitions of vectors for a walking mechanism 35

Figure 3.1: Webots development cycle of robot simulation 39

Figure 3.2: The supervisor updating the CoM and ZMP 44

Figure 3.3: The original Sony QRIO robot compared to the Webots model. 49

Figure 3.4: joint positions and length specification of the Sony QRIO-like model ... 51

Figure 4.1: The motion definitions XML file format. 58

Figure 4.2: Comparing the various implemented interpolations results 60

Figure 4.3: The ZMP Jacobian Counterbalance method outline 61

Figure 4.4: Error-bar plots of confidence intervals on residuals 62

Figure 4.5: The iterative execution scheme of the algorithm 64

Figure 5.1: The initial configuration of the experiment. 68

Figure 5.2: The torso joints' trajectories in the regular run and actual runs 71

Figure 5.3: The torso joints' trajectories adjustments during compensation runs. 71

Figure 5.4: The CoM and ZMP trajectories in the regular and actual runs. 72

Figure 5.5: Snapshots of the experiment in which compensation is on torso joints. . 73

Figure 5.6: Torso pitch and hip roll trajectories in the regular and actual run..... 75

Figure 5.7: Torso pitch and hip roll trajectories adjustments during compensations. 75

Figure 5.8: The CoM and ZMP trajectories in the regular and actual runs. 76

Figure 5.9: The slight downhill stage configuration 77

Figure 5.10: The support-polygon shrinks due to the contact dynamics 78

Figure 5.11: The torso trajectories in the regular and actual runs (downhill)..... 79

Figure 5.12: The torso adjustment during compensation runs (downhill) 79

Figure 5.13: CoM and ZMP trajectories in the regular and actual runs..... 80

Figure 5.14: Torso joints' trajectories in the regular run and actual runs 83

Figure 5.15: CoM and ZMP trajectories in the regular and actual runs..... 83

Figure 5.16: CoM and ZMP trajectories in regular and actual runs..... 84

Figure 5.17: Snapshots of the external disturbance experiment. 85

Figure A.1: How to determine the positive sense of angles α_i and θ_i. 108

Figure A.2: The Webots model of Fujitsu's humanoid HOAP-2. 111

Figure A.3: HOAP-2 robot model specification. 112

Figure A.4: Coordinate frames assignment according to D-H notation..... 113

Figure B.1: Pseudo-code of initialisation of trajectory smoothing. 120

Figure B.2: Pseudo-code for whole-body simultaneous interpolation..... 121

Figure B.3: Pseudo-code for whole-body interpolation adjustment method 123

Figure C.1: Moore-Penrose pseudo-inverse calculation pseudo-code. 126

List of Tables

Table 3.1: The robot's weight parameters. 49

Table 3.2: Link lengths specification of the project's Sony QRIO-like model. 49

Table 3.3: Joints and limits specification of the Sony QRIO-like model. 50

Table 4.1: The convergence of the corrections. 65

Table A.1: Link Lengths specification of HOAP-2. 111

Table A.2: The extracted D-H parameters for HOAP-2 robot model. 114

Chapter 1

Introduction

1.1 Background

This thesis tries to tackle the problem of biped humanoid robot locomotion dynamics and control. Humanoid robots are anthropomorphic robotic systems which try to imitate human capabilities, in order to achieve tasks. It is expected that in the future humanoids will have the ability to adapt to human environments (e.g. offices, homes, and hospitals) and become, along with other various mechanical and automated equipments, adequate and helpful assistants to humans. Many researchers have paid great attention on the potential of humanoid robots. This is, in part, due to their anthropoid shapes which are suitable for human-supporting actions such as: carrying building materials and tools in construction sites; atomic power plant inspection; domestic household tasks etc. (Huang *et al.*, 2001; Nagasaka *et al.*, 1999).

In order to succeed in such real world environments, humanoid robots need to possess stable dynamic biped locomotion. However, the humanoid robots of today still do not satisfy the aforementioned demands and their level of dynamically stable mobility is insufficient in the context of the real and uncertain environment. Hence humanoids cannot cope with unexpected external forces or sudden contacts with the environment. This is partially attributable to the fact that development and implementation of responsive control algorithms has been, so far, scarce (Sugihara *et al.*, 2002). Hereby, locomotion control of humanoids research has still a long way to go.

Humanoids are extremely complex and non-linear dynamical systems, suggesting that there is no closed-form solution for controlling them. This is ascribed to the following problems:

1. The inertia frame lacks fixed points, thus causing the humanoids to be *under-actuated* systems. Inherently, at each contact point, a conversion from internal joint forces to external reaction forces is needed by the humanoid's interaction with its environment (Sugihara & Nakamura, 2003).
2. Moreover, humanoids are multi-body systems with numerous degrees-of-freedom (DOFs) (usually over 20 joints). As such, they have frenzied dynamics, thereby requiring a complicated coordinate frame handling (Sugihara & Nakamura, 2003).
3. Humanoids are *structure-varying* systems, i.e. the link connectivity of the system changes with contact state throughout the interaction of the humanoid with its environment. Indeed, humanoid bipedal locomotion contains three various kinematic chains. Dealing with such situations in a conventional way requires the robot to plan all potential link structures beforehand and shift between them while executing (Nakamura & Yamane, 2000).

Furthermore, achieving high-level behaviours and low-level dynamic stability simultaneously increases the mentioned complexity.

Vukobratovic and Borovac (2004) stated that regardless of structure or degrees-of-freedom, humanoids share some fundamental characteristics of biped locomotion. Firstly, the humanoid robot might rotate around any of the foot boundaries due to external disturbances (considered equivalent to a passive DOF). Secondly, humanoid bipedal gaits are symmetrical. Finally, biped locomotion systems interchange single- and double-support phases at a fixed interval. In the statically-stable *double-support phase* the biped is supported by both feet while in the statically-unstable *single-support phase* only one foot contacts the ground and the other swings from back to front. This explains the structure-varying principle as the humanoid alters its structure from an open to a closed kinematic chain throughout one walking cycle.

The circumstances mentioned above should all be considered in the development and synthesis of artificial gait systems such as the humanoid (Vukobratovic & Borovac, 2004).

1.2 Biped Locomotion Control

1.2.1 Contributing Research Fields to Biped Locomotion

Biped locomotion is one of the principal topics in the design and development of humanoid robots and has been the focus of many studies, from a variety of viewpoints, in the last several decades. In this section, the various contributing research fields, which actively investigate biped locomotion control, will be reviewed. These are depicted in **Figure 1.1**. The following presents those research fields:

- **Robotics:** is "a branch of engineering that involves the conception, design, manufacture, and operation of robots. This field overlaps with electronics, computer science, artificial intelligence, mechatronics, nanotechnology, and bioengineering".
- **Artificial Intelligence:** "The capability of a device to perform functions, which are normally associated with human intelligence, such as reasoning and optimisation, through experience. Note: AI is the branch of computer science that attempts to approximate the results of human reasoning by organising and manipulating factual and heuristic knowledge. Areas of AI activity include expert systems, natural language understanding, speech recognition, vision, and robotics".
- **Cybernetics:** "Based on the Greek '*kybernetes*' meaning steersman or governor, cybernetics is the science or study of control, or regulation mechanisms in human and machine systems, including computers".

- **Biomechatronics:** is "the interdisciplinary study of biology, mechanics, and electronics. Biomechatronics focuses on the interactivity of biological organs (including the brain) with electromechanical devices and systems".
- **Sport Medicine:** "the branch of medicine concerned with the treatment of injuries or illness resulting from athletic activities".
- **Rehabilitation Medicine:** rehabilitation and augmentation of human's motion ability.

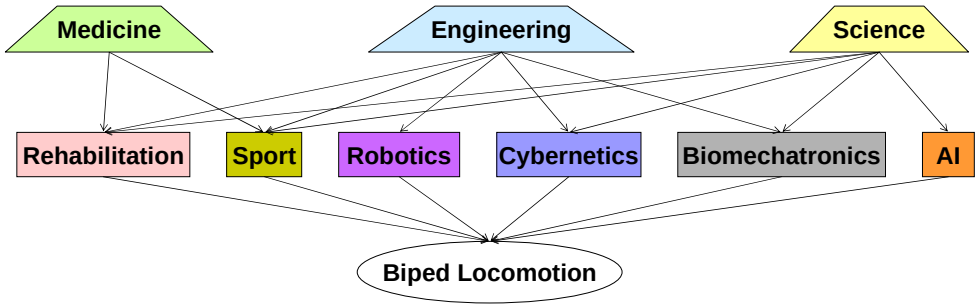


Figure 1.1: Research fields contributing to biped locomotion (adapted from Sugihara, 2004).

Many researchers have studied biped locomotion, mainly from two perspectives. The first perspective is of "human science", where biped robots are designed with similar structures to the human lower limbs and their control algorithms are implemented in such a way that they resemble the human being's walk. Yamaguchi *et al*. (1998, 1997) explains that by means of this practice, the human walking mechanism is verified using a robotics engineering investigative perspective. The authors also indicate that the experiments and outcomes from that perspective feedback the a variety of research fields contained in engineering, science, and medicine such as rehabilitation and sports medicine, cybernetics and biomechatronics.

The second perspective of these studies investigates the development of humanoid robots, which will hopefully evolve to become assistants to humans. Sugihara (2004) states that this perspective is a synthetic approach in which engineering is stirred by artificial biomimetic systems craftsmanship, in contrast to the analytic approach

adopted by medical and scientific interests which emphasise examination of human mechanisms.

1.2.2 Research Motivation for Biped Locomotion

Devising robotic systems, which can essentially move around their environment, is obviously stimulating and exhilarating, but apart from that, there exist two significant motives for biped locomotion investigation. The first is intrinsically *mobility*. The need for vehicles, which can move in difficult terrain, is very important. Raibert (1986) mentions that conventional wheeled robots surpass on surfaces like roads, but present poor mobility in uneven or spongy terrains, thus they can only access half of the earth's landmass. Many researchers have stated (Huang *et al.*, 1999; Huang *et al.*, 2001; Park & Cho, 2000) that biped robots possess higher mobility than wheeled robots, especially when moving in rough terrains. Zhang *et al.* (2003) adds that contrasted to other legged robots (e.g. quadrupeds), bipeds are more dextrous and have higher motion flexibility in complicated environments with obstacles.

One of the reasons why bipeds offer better mobility in rough terrains is that they use isolated footholds which optimise support and grip, whereas wheels or tracks demand continuous support paths. Hence, a wheel or a track negotiates with the worst terrain while a bipedal system selects the best footholds in the accessible terrain. In addition, bipedal vehicles are advantageous due to their dynamic suspension that separates the body trajectory from the feet's trajectories. Consequently, the bipedal system's load (upper body) moves in a smooth manner, even though the terrain is articulately diverse (Raibert, 1986). Moreover, biped robots are flexible when dealing with obstacles (Huang *et al.*, 2001).

The second motive for biped systems research, as Raibert suggests (1986), is to expand our comprehension of the human locomotion. Humans present a wide variety and complexity in the way they swing, hold, fly, and thrust their bodies while preserving their balance, orientation, and velocity. The human performance is remarkable from mechanical engineering, sensory-motor integration, or

computational perspectives. In addition, humans can move reliably through difficult terrains such as forests, swamps, and jungles.

Although we are proficient in the way we use our legs for locomotion, we still do not possess enough understanding in the control principles that bring about efficient locomotion (Lim *et al.*, 2001). In order to learn more about the possible mechanisms for human locomotion, scientists can engage in constructing biped (humanoid) robots. The idea being that, if a human and a biped robot achieve comparable locomotion, then their control and structure ought to work out similar problems. Theories, algorithms, and methods developed for biped robots can direct biological research by proposing specific models for verification, experimentation, and simulation.

1.2.3 Biped Locomotion Principles

1.2.3.1. Dynamic Complexity

Study of biped locomotion systems requires simplification because these systems (and especially anthropomorphic ones) represent highly complex dynamic systems in aspects of mechanical structure and control. Humans possess approximately 350 muscle pairs for execution of complete skeletal activity, thus, even if the human system is idealised to a rigid levers system with simple torque generation acting at each joint, it still involves extreme dynamical complexity. In fact, there is no unique dependence in the interaction between force and movements, and this originates from the evidence that the force-movement relationship is generated, in a biomechanical sense, from a second-order differential equation that demands two initial values, position and velocity, in order to be solved. These two constants may lead to different results during the same initial nervous stimuli. Therefore, any attempt to realise a biped system based on a trivial emulation of human patterns is certainly preposterous (Vukobratovic *et al.*, 1990).

1.2.3.2. Unpowered Degree-of-Freedom

Consider a typical bipedal system consisting of particles connected to form one or more kinematic chains (**Figure 1.2**). Each joint is actuated except for the virtual joint zero (a virtual DOF produced by the foot and ground contact). This joint cannot be affected directly by any of the actuators acting on the system. Vukobratovic *et al*. (1990) stated that the total reaction force is the only effecting element on this joint (under the assumption that the friction is strong enough to prevent slipping), this force is dependent on the whole system's dynamics. Therefore, the unpowered joint's motion is controlled only by the motion of the rest of the system's (powered) joints. The stability of the overall system will not be preserved in case of strong disturbances, which will cause the overall system to rotate around the foot edges and fall, despite realisation of the exact change of joints' angles. This presence of an unpowered DOF is crucially influential on the biped stability.

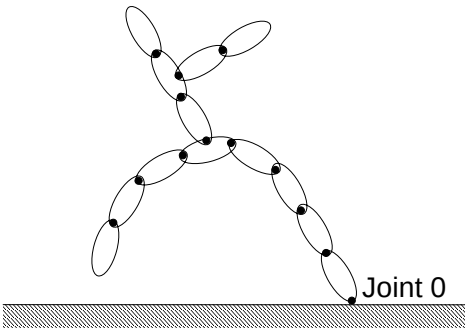


Figure 1.2: Biped system with kinematics chains set (Vukobratovic *et al.*, 1990).

1.2.3.3. Repeatability of Motion

An additional principle of biped locomotion is the repeatability of motion. Vukobratovic *et al*. (1990) define one step of bipedal locomotion as the period in which motion is repeated again and again. As anticipated, positions and velocities at the beginning and at the end of the gait are identical, and as Faconti (2003) states, these conditions are termed as the "*repeatability conditions*", which are satisfied by biped motions. Vukobratovic *et al*. also add that these conditions

originate from biped locomotion's nature and impose extra constraints on the possible solution of motion synthesis.

1.2.3.4. Changeability of Kinematic Structure

Biped locomotion consists of a perpetual interchange between phases when the system is supported on one foot and when both feet are contacting the ground. Many researchers define these phases as the "*single-support phase*" and "*double-support phase*" correspondingly (Kajita *et al.*, 2002; Shih & Gruver, 1992; Vukobratovic & Borovac, 2004). Each of these phases depicts a different kinematic structure of the legs. As Denk and Schmidt (2001) explain, during the single-support phase, when only one of the legs is in contact and the other is in a swing phase, the biped forms an open kinematic chain. On the other hand, in pre-swing and heel-contact, both feet contact the ground and form a closed kinematic chain. These two cases are characterised by different dynamics and should be handled separately.

1.2.4 Review of Biped Locomotion Control Methods

Many studies have been conducted on biped locomotion control, and many biped locomotion methods have been proposed. This section will try to review these previous studies, by first classifying them, based on how they handle the difficulty of reaction force management under the continuous contact state changes with the ground's surface. In each category, several various methods that have been tried in the past will described. **Figure 1.3** depicts this categorisation graphically.

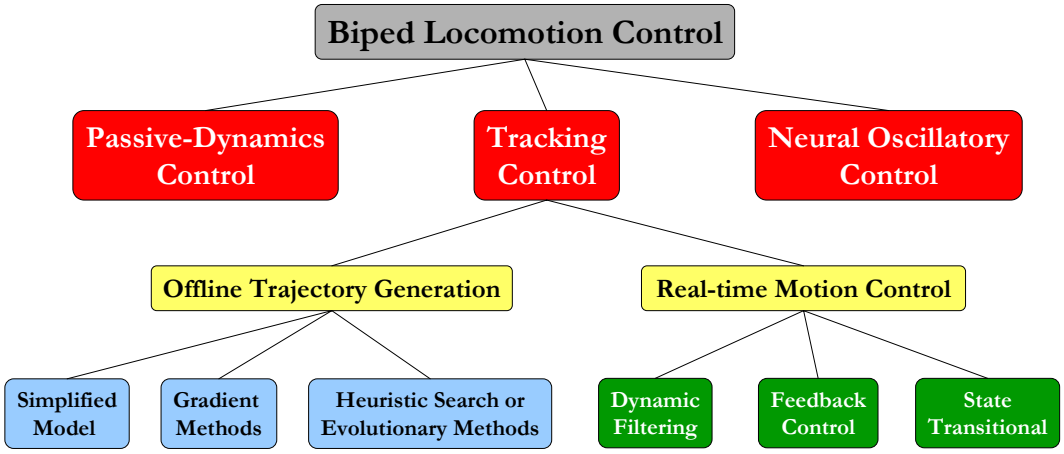


Figure 1.3: Categorisation of Biped Locomotion Control.

The first category referred to as "*tracking control methods*" (term used by Kajita *et al.*, 2003; Nakanishi *et al.*, 2003; Sugihara, 2004) in which the biped locomotion problem is separated into a reference motion pattern design and a stabilisation controller (some use the term compensation). Vukobratovic *et al.* (1970) presented this simple policy through their first model - a simple biped machine which shuffles lower limbs connected to an upper body lever. Later they improved this model to an anthropomorphic system (Vukobratovic & Stepanenko, 1972). Those authors stated that it is possible to plan a reference without the environment's model, according to a balance between the overall external force affecting the system and the generated moment of inertia by the system's motion. Thus, they proposed an overall indicator of the system's behaviour – a point where the mentioned balance is achieved and termed it the Zero-Moment Point (ZMP). The ZMP will be discussed in detail in chapter 2.

1.2.4.1. Offline Tracking Control Methods

This category can be split in terms of two points of view. The first point of view is from the planning or design of some reference. This sub-category shall be referred to as "*offline trajectory generation*" methods. These methods try to generate a dynamically stable walking pattern offline and they assume that the robot and

environment models are available. The current methods for computing dynamically stable trajectories can be branded to three types: (1) heuristic search or evolutionary methods (e.g. genetic algorithms), (2) problem optimisation such as gradient-descent methods, and (3) solving the problem using a simplified model and iteratively converging towards the actual model.

Most of the methods belonging to the latter type rely on the ZMP for pattern generation and control (see Hirai *et al.*, 1998; Huang *et al.*, 2001; Yamaguchi *et al.*, 1999). Those ZMP-based methods usually require precise knowledge of the robot's dynamics (e.g. mass, centre-of-mass location, and inertia of each link) to generate the walking patterns. Hence, they are dependent on the accuracy of the models.

Contrary, there are other methods, which use limited knowledge of the dynamics (e.g. total centre-of-mass location, total angular momentum). Since the controller knows little about the system structure, they rely on a feedback control. Those methods are usually termed as the inverted pendulum approach, since they frequently use variations of the inverted pendulum model. Kajita *et al*. (1991, 1991) suggested the Linear Inverted Pendulum Mode (LIPM) and generated centre-of-gravity (COG) trajectories using strict linearisation of the motion equation. In a further study, they extended to the 3D LIPM (Kajita *et al.*, 2002). Minakata *et al*. (1994) introduced the Virtual Inverted Pendulum Method (VIPM) and used it to vary the walking speed and step length of a "*bird-like walking*" robot in real-time. Several researchers have also combined motion generation through ZMP manipulation with inverted pendulum models. Napoleon *et al*. (2002) described the limitation of performance using ZMP feedback control with one-mass inverted pendulum model as it is a non-minimum phase system, bearing "*undershoot*" and "*waterbed*" effects/problems. Instead, they proposed a two-mass inverted pendulum model and provided the mathematical equations to depict the two-mass inverted pendulum model, in order to force actual ZMP trajectory to become closer to the reference ZMP trajectory.

Of the first type of offline trajectory generation methods, some studies have tried to design trajectories using *evolutionary algorithms*. The referential patterns are acquired through trials-and-error search such as a genetic algorithm (GA). Arakawa *et al*. (1997) aimed to generate natural motion of a biped robot, similar to a human walking in various environments. They applied a hierarchical method through energy optimisation consisting of a GA layer that minimised the total energy of all actuators, and an evolutionary programming (EP) layer that optimised interpolated configuration of the biped. By formulating the trajectory generation problem as an energy minimisation problem, they applied that hierarchical method. Interestingly, Mojon (2003) created a dynamical simulation of a biped humanoid robot using the platform which will be used in this research – Webots™ (see Chapter 3). This humanoid (Sony QRIO-like) generated walking motion using GA which optimised parameters of sinusoidal open loop control for each actuator (i.e. amplitude, frequency, phase, and offset). Although these heuristic and evolutionary methods necessitate less concern of dynamics from the designer, they usually fail to create reliable or natural motion patterns.

Nagasaka *et al*. (1999) described a method in which walking patterns based on the ZMP are generated by an optimal gradient method (OGM) in which at first the designer gives prescribed time trajectories of both feet, hands and a reference of ZMP. Then, initial trajectories of a trunk are determined based on a static walk. Finally, OGM optimises the horizontal motion of a trunk to reduce the deviation of the calculated ZMP from its reference. They also show that it can be applied to the yaw moment compensation problem and soles-ground shock compensation problem by simply changing the objective function of OGM. This is a unique example of the second type of offline trajectory generation.

1.2.4.2. Real-time Tracking Control Methods

The methods mentioned above, ignore the environment model and thus can considerably reduce the complexity of biped locomotion. Nevertheless, they are susceptible to real world uncertainties. Thus, the second point of view is from the

design of a controller. Let us refer to this sub-category as the *real-time* (*online*) *motion control* methods. Some of these controllers have been developed based on momentum *feedback* control and presented a combination with a preset reference obtained by the first sub-category methods (Hirai *et al.*, 1998; Huang *et al.*, 2001; Park & Cho, 2000). Hence, those are considered less robust against disturbances. Numerous other studies designed impressive biped locomotion controllers (Fujimoto *et al.*, 1998; Sorao *et al.*, 1997), but in fact, those yielded very complex systems due to the requirement to support various motions.

Instead, some methods used *dynamic filtering* to convert an input trajectory to a physically consistent and dynamic trajectory (Nakamura & Yamane, 2000; Yamane & Nakamura, 2000). In these methods, the authors generate motions for biped humanoid robots interactively using "*dynamics filter*" as a motion generator. They state that interactivity is the key issue of many humanoid applications working in a constantly changing environments with humans. Since their filter uses only temporal-local information, they can vary the preset reference trajectory by kinematic combination of several motions in response to the humanoid's interactions with the environment.

Other inspiring online motion control methods include: Nishiwaki *et al*. (2002) proposed an efficient online method to generate humanoid walking motions, which satisfy a desired upper body trajectories, while simultaneously carrying objects by subsequent updates to motion patterns and connecting them to the old ones in a stable manner, and Kagami *et al*. (2002) proposed a fast dynamically equilibrated trajectory generation method for a humanoid robots. Again, given an input motion and a desired ZMP trajectory, the algorithm generates a dynamically equilibrated trajectory. They denote three key issues: 1) an "*Enhanced ZMP*" constraint, which enables the stability calculation even when several limbs are contacting the environment, 2) a simplified robot model that represents the relationship between COG and ZMP, and 3) a convergence method, which eliminates approximation errors arising from the simplified model.

Several online methods have paid much attention to the fact that a gait, normally composed of several stages, in sense of foot support states and are defined as *state transitional* methods. Kagami *et al*. (2000) introduced "*AutoBalancer*" - an online dynamic balance stabilisation algorithm for humanoid robots which works by enumeration the possible contact state changes and compensating for dynamic instabilities using all joints in real-time. The authors formulate and solve the balance compensation problem as an optimisation problem. The solution is achieved for each state of feet-ground contact through particular constraints (mainly COG and moments of inertia constraints). The system is decomposed to 2 parts: a planner for transitions between states derived from the feet-ground contacts and a dynamic balance compensator which maintains dynamic stability through solving a constrained second order nonlinear programming optimisation problem. Pratt *et al*. (1997, 1998) described an intuitive motion control scheme for robots called Virtual Model Control (VMC). VMC is a language for describing interactive force behaviours. It uses simulations of virtual mechanical components to generate real actuator torques (or forces). These joints' torques create the same effect that the virtual components would have created if they had existed, thereby creating the illusion that the simulated components are connected to the real robot. Furuta *et al*. (2001) discussed four biped locomotion control strategies with various advantages (e.g. versatility, high-energy efficiency) in terms of final and initial states of the single- and double-support phases. However, these above methods usually are based on heuristic approaches which make the developed bipeds using those methods limited and less versatile in motion.

1.2.4.3. Passive Dynamics Control Methods

The second category of biped locomotion methods is established upon passive dynamics of the biped system. McGeer (1990) was the first to introduce the notion of passive dynamic walking. These methods essentially exploit the innate dynamics of the bipedal walking system which exhibit rhythmic cyclic gaits without actuation, external energy sources other than gravity, and without any

active feedback control (Kuo, 1999). The main attractive characteristic of passive dynamic bipeds is that they need not specify an actual trajectory according to which it is strictly controlled. If the biped deviates from the trajectory, the perturbation is progressively eliminated through the support transfer occurring in each step cycle (Kuo, 1999). Collins *et al*. (2001) extended this approach to a three dimensional passive walker device with specially curved feet, compliancy of the heel, and mechanically constrained arms, thus achieving harmonious and stable gait. But the main interest of this category of methods is the evolvement of periodic gaits, while usually neglecting agility and responsiveness of motion.

1.2.4.4. Neural-Oscillatory Control Methods

Finally, there is the "*neural-oscillatory control*" category. These control methods are based on the concept that biomimetic combination of artificial neural networks and the biped's dynamics can realise robust bipedal locomotion control in terms of external disturbances and energy consumption. Most of these methods use Central Pattern Generators (CPGs), neural oscillators proposed by Matsuoka (1987) which model the firing rate of two mutually inhibiting neurons depicted in a differential equations set. Taga *et al*. (1991) applied CPG for musculo-skeletal bipedal control and proved that an adaptive walking motion through various terrains could be realised from the interaction between the neural oscillatory controller and the body and environment dynamics.

Several other researchers extended this approach using similar strategies. Miyakoshi *et al*. (1998) achieved robust and adaptive locomotion while coordinating a redundant high DOF system under the strong effect of physical body dynamics. They extended Taga's work from 2D to 3D and also simplified the CPG control mechanism. One CPG controls the roll of the pelvis with respect to the trunk and two other CPGs control the "stamping" motion of the legs. Each leg's CPG flexor units excite the hip, knee, and ankle flexors of the ipsilateral leg and inhibit the flexor unit for the contra lateral leg. Entrainment is achieved with

the help of touch sensors on the feet and orientation sensors for the thighs that provide feedback to the CPG's.

Nakanishi *et al*. (2003) proposed a learning method for biped locomotion from human demonstration. The method adapts its frequency using rhythmic dynamical motion primitives as a CPG. In these primitives, the kinematic motion plans are depicted in a nonlinear differential equations set with well-defined attractor dynamics, and demonstrated trajectories are learned using locally weighted regression. The authors' numerical simulations illustrated the effectiveness and within a short term of walking, the simulation discovered an energy efficient walking frequency, roughly at the natural frequency of the combined robot-oscillator environment system.

1.3 Research Goals

The goal of this research is to investigate human-like locomotion and to develop a bipedal motion control method for simulated humanoid robots to enhance their mobility. The root of this control method is based on the Zero-Moment Point, the prominent dynamic stability criterion which has been thoroughly formulated, studied and used in the last 30 years in the robotics research community (see chapter 2). Due to the principles described in section 1.2.3, the humanoid locomotion still remains challenging and invigorating and has plenty more to advance in the future.

Many of the past studies concentrated on the typical periodic and stable biped locomotion, and they assumed a predetermined kinematic structure with fixed joint assignment, thus those studies were not generalised enough to be applied to other models. The fundamental approach used here to tackle the humanoid locomotion problem is to comprehend the common humanoid dynamics and to avoid specifics, which, according to the previous literature review, is predominantly encouraging to achieve best performance.

The development of this research pursued a *bottom-up* strategy, in which, the core functionalities of typical humanoids were created, optimised and tested first and on top of them, the higher level capabilities of control were constructed. Moreover, with reasoning that speed affects control performance and modular design allows easier modification and maintenance, the creation of each component was scrutinized in terms of both computational load and software modularity.

As a whole, this work aims to serve as a general dissertation on simulated humanoids control development, rather than a report of a specific model, and intends in helping future researchers to avoid the majority of initial complications involved in such work.

1.4 Thesis Outline

This dissertation is composed of seven chapters. This section shall briefly outline the main idea of each chapter.

Chapter 2 thoroughly describes the definition, concept and mathematical derivation of the Zero-Moment Point (ZMP), a prominent and reliable dynamic stability index, which is the root of this study. Furthermore, it examines previous research of biped locomotion related specifically to the ZMP and strives to give a solid understanding of the theory underlying humanoid locomotion control development.

Chapter 3 introduces the simulation environment used in this work. It briefly discusses the importance and relevance of using simulation, it then elaborates on the concerns involved in the creation of simulated control of humanoid locomotion and describes fundamental simulation oriented implementations required for such control. Finally, it reveals the robot model specifications used.

Chapter 4 presents the concept and a mathematical formulation of the ZMP Jacobian, a mapping between the full-body joint angles velocities to the ZMP velocity of the humanoid robot. It also explains the complex issues of designing gait motion

trajectories for humanoid robots. Then it develops a motion control method, termed the *ZMP Jacobian Counterbalance*, which adjusts predefined motion trajectories using the ZMP Jacobian to enhance performance and dynamic stability during bipedal locomotion.

In Chapter 5 several experiments results are illustrated which are established on the control method presented in Chapter 4. These experiments include among others: adapting walking trajectories of the redundant torso degrees-of-freedom, to improve stability and augmenting with more degrees that are redundant. The results of these experiments prove the feasibility and proficiency of the proposed method.

Chapter 6 provides a concise discussion of the known problems and limitations, and the contribution and significance of this dissertation. It also recaps the knowledge and understanding the author acquired during the course of this work.

Lastly, Chapter 7 concludes with a brief summary of this dissertation, and outlines several potential ideas for future development.

Chapter 2

The Zero-Moment Point

The previous chapter presented a broad background and literature survey of the biped locomotion in robotics. However, it only briefly mentioned the Zero-Moment Point (ZMP) along with a few examples of studies that revolved on its concept. Owing to the fact, that the principal goal of the project was to study and develop biped humanoid locomotion control based on the ZMP, it was decided to concentrate a full chapter to discuss in depth all the aspects of the ZMP, from its definition and concept, to its applicability (including previous studies) and variants, and finally its derivation.

2.1 Definition

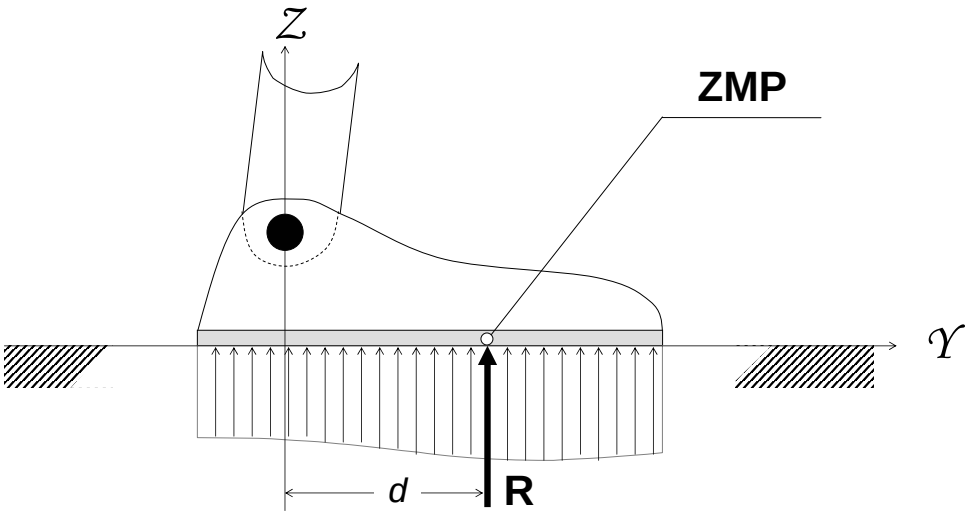


Figure 2.1: Zero-Moment Point original definition (Vukobratovic *et al.*, 1990).

The Zero-Moment Point (ZMP) was originally defined by Vukobratovic *et al*. (1972) although they developed the concept previously (Vukobratovic & Juricic, 1969). The ZMP's definition was as follows (see **Figure 2.1**):

> "*As the load has the same sign all over the surface, it can be reduced to the resultant force* ***R****, the point of attack of which will be in the boundaries of the foot. Let the point on the surface of the foot, where the resultant* ***R*** *passes, be denoted as the* ***zero-moment point*** *or* ***ZMP*** *in short.*"

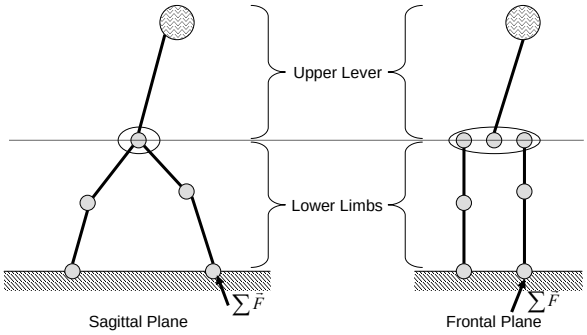


Figure 2.2: Vukobratovic *et al*.'s Biped Shuffle Kinematic Model.

In the original paper, they proposed a kinematic model in which the feet were in contact with the ground as shown in **Figure 2.2**. In order to maintain the support foot stationary and let the other foot swing forward i.e. make the system shuffle, an essential condition was mentioned where the ground reaction force must operate only on the supporting foot. Their original proposal was highly significant due to the following two facts. Primarily, it allows designing motion trajectories, which satisfy the mentioned condition while ignoring the environment's counteraction, as the total external force is dynamically equilibrated with the internal forces generated by the system's actuators. Secondly, motion design can be obtained by moment compensation in the planning because the horizontal component of the moment around the ZMP is zero (thus its name). In fact, in this preliminary model, the lower limbs' motion trajectory were analytic functions, hence, the upper limb's motion

compensating the moment around the support foot was computed using an iterative analytic solution.

Later, Vukobratovic et al. (1970) proposed feedback control for the same model. As mentioned previously, omission of the environment in the motion planning can simplify the problem, but it makes the system less robust against disturbances caused by slight model deviations. Therefore, the moment error should be removed using online feedback, so they proposed a real-time motion modification using a sensitivity matrix whose elements measured moment versus acceleration variation, to facilitate modification magnitude calculation as an inverse to the moment error.

2.2 The ZMP Concept

These initial investigations by Vukobratovic *et al*. showed that the ZMP could be an effective criterion for the control of biped locomotion, since it enabled an extension to the conventional control of manipulators while detaching motion planning phase and online sensory feedback. However, it is important to observe that the ZMP by itself never represents a sufficient condition since it includes only two components of the total external force that consists of six components.

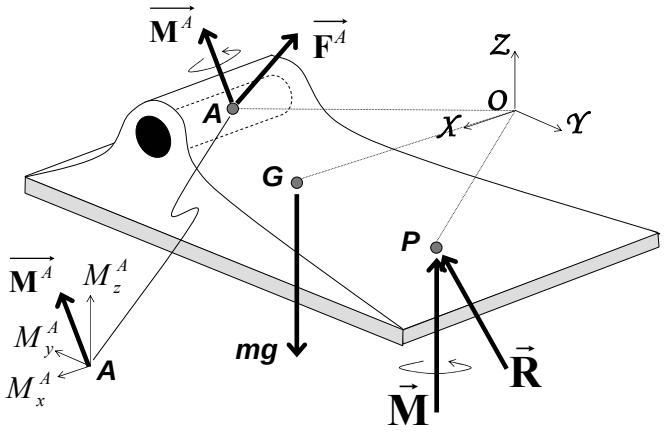


Figure 2.3: Forces and Moments acting on the biped foot (Vukobratovic & Borovac, 2004).

In order to explain the ZMP concept, Vukobratovic *et al*. (2004) analyse the locomotion during the single-support phase, in which the whole foot is contacting the ground. For simplicity, they separate the ankle of the support foot from the rest of the system and substitute its affect by force $\overrightarrow{\mathbf{F}^A}$ and moment $\overrightarrow{\mathbf{M}^A}$ (see **Figure 2.3**) and the point *G* the foot's centre of gravity. The ground reaction force is acting at point *P*, hence maintaining the whole system in balance. In addition, there is also the total ground reaction force $\vec{\mathbf{R}} = \left(R_x, R_y, R_z\right)$ and moment $\vec{\mathbf{M}} = \left(M_x, M_y, M_z\right)$. As detailed in that paper, due to the friction force and the fact that the foot is at rest, the horizontal reaction force (R_x, R_y) stands for the friction force balancing the horizontal component of force $\overrightarrow{\mathbf{F}^A}$. On the other hand, the vertical reaction moment M_z stands for the friction reaction forces' moment balancing the vertical component of the moment $\overrightarrow{\mathbf{M}^A}$ and the moment caused by $\overrightarrow{\mathbf{F}^A}$. Thus, assuming there is no slip, the static friction compensates for (R_x, R_y) and (M_z). On the other hand, R_z is the ground reaction component, which balances vertical forces.

Since the ground reaction force induced by the foot's action is oriented upwards, the horizontal components of all active moments can be compensated by only moving $\vec{\mathbf{R}}$'s position inside the convex hull[a] of the foot support area, also known as the Support Polygon (see **Figure 2.4** below). Hence, $\overrightarrow{\mathbf{M}^A}$'s horizontal components will shift the reaction force appropriately, in order to balance any overload[b].

In case the support polygon is too small to include the position $\vec{\mathbf{R}}$, in order to compensate the external moments, $\vec{\mathbf{R}}$ will only act at the foot's edge and the uncompensated remaining part will result in a rotation of the system about the foot's edge, i.e. overturning.

[a] The convex hull of a set of points is the smallest convex set, which includes those points.
[b] In **Figure 2.1**, a simple case in the $\mathcal{Y}$–$\mathcal{Z}$ plane is shown. The moment M^A_x is compensated by moving the acting point of R_z, whose magnitude is resolved using the overall acting forces equation of balance, by the distance *d*. Note that as long as $\vec{\mathbf{R}}$ is inside the foot's area, any increase in the moment will be compensated by moving the position of $\vec{\mathbf{R}}$, and M_x and M_y will be zero.

Thus, a *necessary* and *sufficient* condition for the bipedal locomotion system to remain in dynamic balance is for the point *P* (**Figure 2.3**) to satisfy is:

(2.1) $$M_x = M_y = 0$$

This equality to zero is the main reason for the name "*Zero-Moment Point*". Namely, as long as the ground reaction, due to the foot's rest, can be reduced to $\vec{\mathbf{R}}$ and M_z; the reaction force's acting point *P* depicts the ZMP.

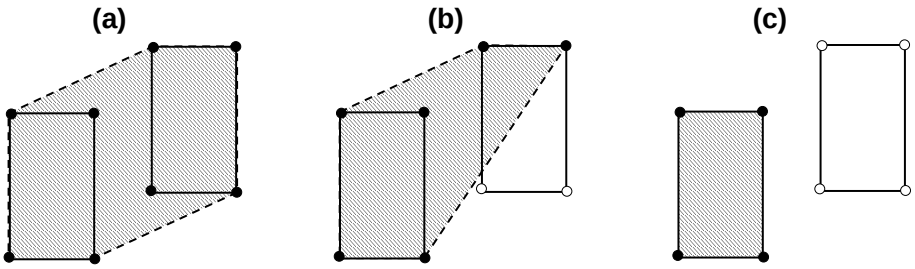


Figure 2.4: The support polygon in three typical cases, the biped's feet are represented as rectangles and the support polygon as the shaded area. Contact points with the ground are represented as black circles, while no contact is in white circles. (a) Both feet are fully supporting. (b) Right foot is only touching the ground at the forward tip. (c) Only Left foot is in contact with the ground.

The next step in understanding the ZMP concept is to understand how its location is related to dynamic balance. Once again, the reader is reminded that since the system is supported solely by the foot, the system's dynamic balance requires a full rest of the foot on the floor (**Figure 2.3**). Under this assumption, the static balance equations for the support foot are:

(2.2) $$\vec{\mathbf{R}} + \overrightarrow{\mathbf{F}^A} + m\vec{\mathbf{g}} = 0$$

(2.3) $$\overrightarrow{\mathbf{OP}} \times \vec{\mathbf{R}} + \overrightarrow{\mathbf{OG}} \times m\mathbf{g} + \overrightarrow{\mathbf{M}^A} + M_z + \overrightarrow{\mathbf{OA}} \times \overrightarrow{\mathbf{F}^A} = 0$$

Where $\overrightarrow{\mathbf{OP}}$, $\overrightarrow{\mathbf{OG}}$ and $\overrightarrow{\mathbf{OA}}$ are vectors from O_{xyz} (the ankle's coordinate system origin) to P, G, and the ankle's joint (A), respectively, and the foot's mass is m. By placing O_{xyz} at the point P and projecting Eq. (2.3) onto the $\mathcal{Z}$-axis, the vertical component of the ground reaction moment (actually, it is the ground friction moment) becomes:

$$M_z = M_f = -\left(M_z^A + (\overrightarrow{\mathbf{OA}} \times \overrightarrow{\mathbf{F}^A})_z \right) \tag{2.4}$$

This, in general, might be non-zero but can be reduced to zero by the appropriate dynamics of the overall system. However, the projection of Eq. (2.3) onto the $\mathcal{X}$-$\mathcal{Y}$ plane yields:

$$(\overrightarrow{\mathbf{OP}} \times \vec{\mathbf{R}})_{xy} + \overrightarrow{\mathbf{OG}} \times m\mathbf{g} + (\overrightarrow{\mathbf{M}^A})_{xy} + (\overrightarrow{\mathbf{OA}} \times \overrightarrow{\mathbf{F}^A})_{xy} = 0 \tag{2.5}$$

Eq. (2.5) provides a way to compute the position of the ZMP (i.e. the ground reaction force acting point - P) and it also depicts the foot's balance and yields the ZMP's position which will ensure dynamic stability, in terms of the overall bipedal system dynamics. Nonetheless, Eq. (2.5) does not satisfy the inverse problem: *Given the locomotion, is the system dynamically balanced*?

2.2.1 The ZMP's relation to the Support Polygon

In order to deal with the problem posed above, let us first look at the special relation of the P's position to the support polygon of the biped. If the position of P, computed from Eq. (2.5), is inside the support polygon, the system is dynamically balanced. Since the reaction force $\vec{\mathbf{R}}$ cannot act on the system if P is outside the support polygon, it is clear that, in practice, to ensure dynamic stability, P, which satisfies Eq. (2.5), cannot exist outside this polygon and must be within it.

The meaning of P which is outside the support polygon (in theory), is considered as follows. Knowing that P's position was acquired using the constraint in Eq. (2.1), it

is termed as a Fictitious ZMP (FZMP) (Vukobratovic & Borovac, 2004). Hence, in reality, ZMP only exists inside the support polygon. Eqs. (2.2) and (2.3) clearly show that the ZMP's position is dependant on the system's dynamics ($\overrightarrow{\mathbf{F}^A}$ and $\overrightarrow{\mathbf{M}^A}$).

If the system's dynamics change, so as the ZMP reaches the support polygon's edge, the point *P* will only stay the ZMP if no extra moments are acting at it. If not, the biped would start to rotate about the foot's edge and overturn. In this circumstance, the ground reaction force's acting point would be on the foot's edge, however, this point will cease to be ZMP, due to either $M_x \neq 0$ or $M_y \neq 0$.

Vukobratovic *et al*. (2004) further clarify the meaning of the ZMP being outside the support polygon, present a general method to determine the ZMP's position for a given biped motion using its dynamics model. In section 2.5, several derivations of the ZMP using various dynamics models shall be presented.

2.2.2 Various ZMP Interpretations

Although the ZMP concept has been so widely used and cited throughout a large number of research papers involved with biped locomotion, it was defined in many ways, which vary in their degree of detail. This is exemplified by a few sample interpretations collected along the comprehensive literature exploration for this study. Dasgupta and Nakamura (1999) present the following interpretation:

> "*ZMP is defined as that point on the ground at which the net moment of the inertial forces and the gravity forces has no component along the horizontal axes*."

Arakawa and Fukuda (1997) gave another interpretation adopted by Faconti (2003) and Napoleon *et al*. (2002):

> "*The ZMP is the point on the floor at which the moment T: (T_x, T_y, T_z) generated by the reaction force and the reaction torque satisfies T_x = 0, and T_y = 0.*"

Huang *et al*. (2001) expressed yet another version:

> "*The ZMP is defined as the point on the ground about which the sum of all the moments of the active forces equals zero*."

This thesis adopts Vukobratovic *et al*.'s (2004) claims that the ZMP has been vaguely related to the ground surface, regardless of referring to the support polygon. Furthermore, many researchers have failed to emphasise that a ZMP outside the support polygon has no practical meaning (as by definition a ZMP outside is non-existent).

2.2.3 ZMP and CoP

Many studies have confused the ZMP with the Centre-of-Pressure (CoP). This section is dedicated to elaborate the difference between the CoP and ZMP, as there is evident distinction between these two concepts, and in general, they need not be equivalent.

The CoP is defined as the point on the ground where the resultant of the ground-reaction force acts (Goswami, 1999). This is equivalent to the Centre of the Actual Ground-Reaction Force (C-ATGRF) definition used by Hirai *et al*. (1998). Whenever this resultant force during the motion is equilibrated with all active forces (e.g. Coriolis, inertia, gravity, centrifugal), its acting point becomes the ZMP. Hence, if and only if, the gait is dynamically stable, CoP and ZMP are the same.

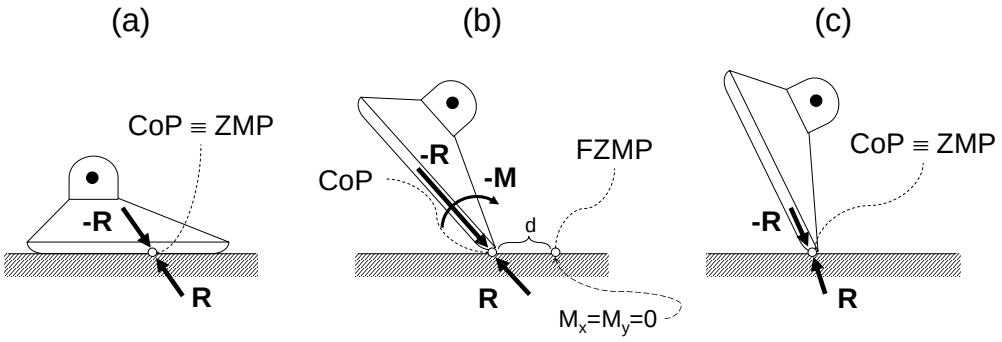


Figure 2.5: Three typical cases depicting the relationship between ZMP and CoP: (a) dynamically stable gait, (b) unstable gait, ZMP does not exist and the ground reaction force acting point is CoP; the point where $M_x = M_y = 0$ is outside the support polygon (FZMP). The system will rotate about the foot edge and fall over, and (c) tiptoe dynamic balance (Vukobratovic & Borovac, 2004).

To illustrate the ZMP and CoP relationship lets consider three typical cases of foot-ground contact (see **Figure 2.5**). When the gait is dynamically stable, the ZMP coincides with CoP (**Figure 2.5a**). **Figure 2.5b** depicts a different case where a disturbance occurs, causing the ground reaction force acting point to be at the foot's edge. Here, the perturbation moment will inevitably cause the bipedal system to rotate about the foot edge and it will fall over. Obviously, there exists only a Fictitious ZMP (FZMP), whose distance *d* from the foot edge is proportional to the perturbation's magnitude. Yet it is possible to realise a biped motion on the tiptoe (as in **Figure 2.5c**) using feet with a pinpoint area, and thus causing the ZMP to coincide once again with the CoP. However, this case does not portray a regular gait.

In short, the ZMP always coincides with the CoP in a dynamically stable gait. However, the CoP is not always ZMP (e.g. in a dynamically unstable gait). In addition, the FZMP never agrees with the CoP because, by definition, the CoP does not exist outside the support polygon.

2.3 Applicability of the ZMP

2.3.1 Offline ZMP Based Motion Planning

The first study which performed motion planning, based on the ZMP, was conducted by Vukobratovic *et al*. (1972) when they extended their shuffling system model to an anthropomorphic system, thus proposing a control method which focused on designing motion trajectories offline and then modifying them to achieve a desired ZMP online.

Along the years, motion planning based on ZMP criterion has broadly expanded (Arakawa & Fukuda, 1997; Huang *et al.*, 2001; Kagami *et al.*, 2000). However, due to those systems' complexity and large number of DOFs, those studies took on the ZMP as an assessment of dynamical feasibility of trajectories seemingly realising walking or by constraining it, to remain inside the supporting polygon. This was quite different from Vukobratovic *et al*.'s original application.

However, Yamaguchi *et al*. (1993; Yamaguchi *et al.*, 1998, 1999) were the first to solve the problem of acquiring trajectories by inverting some desired ZMP motion. They developed an offline motion modification algorithm by separating lower limbs' motion in frequency domain using Fast Fourier Transforms (FFT), and computing the upper body's motion, which compensated the moment about this desired ZMP. This method was especially adept in periodic motions cases. Using this principle, they built the Waseda Leg and later on WABIAN robots series.

Dasgupta and Nakamura (1999) proposed a method of generating feasible walking of humanoid robots by converting Human Motion Capture Data (HMCD) to a modified version which satisfies a desired ZMP with a 2-DOF waist joint. They generated the desired ZMP trajectory using an appropriate foot model, which agreed with the HMCD. By assuming periodic corrective motions represented as Fourier series, they formulated a steepest descent optimisation problem, which determined the unknown

coefficients of the Fourier series and considered the humanoid's complete non-linear dynamics.

Nagasaka *et al*. (1999) proposed another offline motion design method which was formulated as an optimisation problem. They termed it as the Optimal Gradient Method (OGM), which modified the trunk position's trajectory in order to realise the desired ZMP trajectory. They rightly claimed that this method was advantageous, due to its algorithmic simplicity and superior generality. Since the steepest descent vector is determined by numerical and not analytical methods, designers need only provide prescribed trajectories, an initial solution to the optimised trajectories, and the appropriate objective function. However, the drawback of this method was the severe computation time, so in later research Nagasaka improved it by proposing the idea of a Dynamics Filter, which consisted of several "filters" – each implementing a partial optimisation problem. Then Kagami *et al*. (2002) developed a fast trajectory generation method based on the Dynamics Filter idea. They used a simplified robot model so-called the mass-concentrated model that allows quantising the dynamics motion equation into trinomial form, thus they managed to reduce the computation time drastically.

Then again, none of the above-mentioned methods paid much attention to the generation of the ZMP trajectory, which has substantial influence on the generated locomotion. Park *et al*. (Park, 2003, 1998) suggested a design method for the referential ZMP trajectory by means of fuzzy logic.

2.3.2 Online ZMP Based Motion Modification

As mentioned in section 2.1, any reference-based motion control demands an additional online feedback control, in order to overcome errors resulting from external perturbations. Several studies developed control methods, which compensate the real-time error between desired and actual ZMP. For example, Park and Chung (1999) proposed an online ZMP compensation combined with impedance control. In their method when the actual sensed ZMP leaves a preset ZMP boundary,

the online planner generates the base link's trajectory in the vertical axis in order to compensate for required moment of the stability recovery. On the other hand, Okumura *et al*. (2003) introduced an adaptive ZMP compensation controller which utilised the principle of inertial loading while still preserving pre-assigned foot landing positions (which might be critical to the environmental needs of the biped).

On a different approach, Hirai *et al*. (1998) proposed a posture control composed of three sub-control methods: (1) Ground Reaction Force control by the ankle joint adjustment; (2) Model ZMP control for shifting the desired ZMP to an appropriate position to recover the robot's posture; and (3) Foot Landing Position control which corrects the relative upper body and feet positions jointly with the previous Model ZMP control.

Similarly, Lim *et al*. (2001) described a locomotion control, based on balance and impedance control. The balance component compensated for moments generated by the biped walking and operated during a complete walking cycle. The compensatory motion of trunk and waist was computed from lower-limbs, arms, head and ZMP trajectories. The parameters (damping, stiffness etc.) of the impedance controller component were adjusted in real-time according to gait phase.

In terms of d'Alembert's principle, these methods work as stabilisers and indirectly manipulate the ZMP, while the previous mentioned studies do not, since the only use the ZMP as an index of the equilibrium point.

2.3.3 Input ZMP Based Manipulation

Some studies inferred that the ZMP might be considered for manipulation in the force manipulation phase, due to its strong linkage with torques or forces. Fujimoto *et al*. (1998) developed a hierarchical control based on an indirect manipulation of the ground reaction force at the foothold. The system was decomposed to a reactive force controller and a body posture controller. This method incorporated physical constraints of contact force, ZMP and friction force.

Sorao *et al*. (1997) formulated a control strategy based on both COG and ZMP for the biped's support leg. The algorithm worked in two stages: 1) ZMP trajectory generation to realise an arbitrary COG trajectory; and 2) ZMP controller, which directly manipulated the ZMP of the support foot. The ZMP trajectory was generated based on the COG tracking error and used fuzzy logic. They claimed that the method enables realisation of an arbitrary COG trajectory by the ZMP position command and satisfies dynamic stability bypassing the nonlinear relationship between the ZMP and COG.

Sugihara *et al*. (2002) suggested a real-time motion generation which controls the COG by an indirect ZMP manipulation. They showed an online real-time response of their method which produced high-mobility of their humanoid platforms. The algorithm consisted of four components: referential ZMP planning, ZMP manipulation, COG velocity decomposition to joint angles, and local joint angles control. A year later, those authors (Sugihara & Nakamura, 2003) introduced the concept COG Jacobean, which relates whole-body motion to COG motion. They discussed, amongst the rest, the effectiveness of using COG Jacobean for ZMP manipulation and thus can be used for stability and responsiveness of a biped robot.

Interesting and ongoing research tried to combine ZMP compensation with Reinforcement Learning (Kim *et al.*, 2003). In this study, the authors introduce a novel two mode Q-learning which utilises both success and failure experience of the agent for convergence, and employ it in the proportional ZMP compensation of their humanoid's walking and standing posture.

All the methods and studies mentioned above consider the ZMP as the input variable they try to manipulate it to achieve some indirect effect.

2.4 ZMP Variants and Extensions

The original definition of the ZMP (section 2.1) was intended for a horizontal plane, but it should be generalised to a three-dimensional locomotion. Kagami *et al*. (2002)

explained that humanoid typed robots with four limbs can be supported by more than two contact points, thus they introduced the Enhanced ZMP idea which enabled different contact states in three-dimensional space. The derivation is quite straightforward, namely defining the enhanced contact points' plane and adapting the coordinate system appropriately (see **Figure 2.6**).

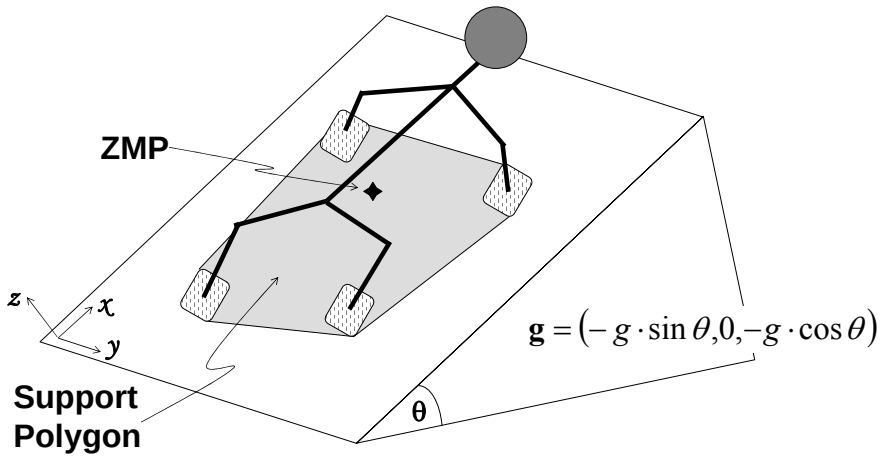


Figure 2.6: The Enhanced ZMP concept (Kagami *et al.*, 2002).

As mentioned previously, the ZMP is defined as a point on the ground's surface. However, the contact points between the environment and the humanoid are three-dimensional. Hence, a different framework is needed to incorporate ZMP feasibility to special cases in which this contact is three-dimensional. As discussed above, the Enhanced ZMP is projected on a plane created by three arbitrary points selected from all the contact points. Clearly, choosing these three points appropriately is a nontrivial task, and verifying the best combination of three points of all possible is computational expensive.

The concept of the Virtual Horizontal Plane (VHP) was also introduced (Sugihara *et al.*, 2002). It enables such a three-dimensional contact model. Each real contact point is substituted by an equivalent point on VHP regarding the forces acting to support the COG. The concept disregards inertial forces and such an equivalent point is the intersecting point of the line connecting the contact point and the COG with the VHP

(see **Figure 2.7**). This is because a force vector acting on the COG at a point p_i can be translated parallel to the transverse line of action. The support polygon on the VHP is the convex hull formed by those points.

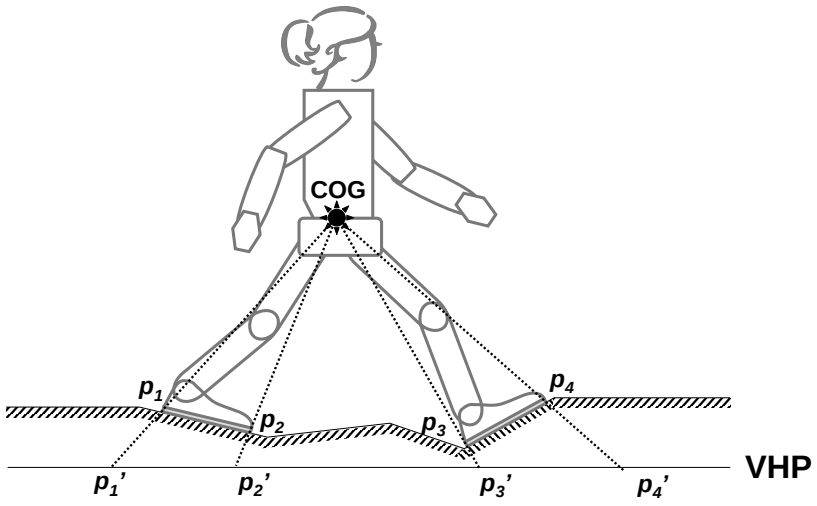


Figure 2.7: The Virtual Horizontal Plane (VHP) concept (Sugihara *et al.*, 2002).

Goswami (1999) presented the concept of the Foot-Rotation Indicator (FRI) point:

> "*The FRI point is defined as the point on the foot/ground contact surface, within or outside the convex hull of the foot-support area, at which the resultant moment of the force/torque impressed on the foot is normal to the surface*"

Goswami states that the FRI point has several important properties. Firstly, it indicates the foot rotation possibility. Second, the magnitude of the foot's unbalanced moment due to the impressed forces about a point *p* on the support-polygon's boundary is proportional to the distance between p and the FRI point. When the FRI is inside the support polygon, that moment counteracted and is precisely compensated; otherwise, that uncompensated moment is what causes the foot to rotate. In the same principle, the FRI point can indicate the direction of the foot's rotation. Lastly, it gives a measure to the stability margin of the robot according to

the minimum distance of the support polygon boundary from the FRI point when it is within the footprint. On the other hand, when it is outside, this measures the robot's instability. An impending foot rotation will be implied when the FRI point moves towards the support-polygon boundary. Nevertheless, no practical application study has been conducted to verify the FRI characteristics.

Besides that, the FRI point and the Fictitious ZMP (FZMP) (section 2.2.1) are equivalent in how they virtually fix the supporting foot in the inertia frame. However, the FRI point is discussed to be computed online, while the FZMP is measured offline and has already been applied in several conventional motion planning methods (Dasgupta & Nakamura, 1999; Kagami *et al.*, 2002; Nagasaka *et al.*, 1999; Yamaguchi *et al.*, 1993).

Many more variants and extensions to the ZMP are presented in the literature, such as local ZMP per limbs, other interpretations to the ZMP enhancement, studies of the stability margin etc. but those are less known in the humanoid and biped robotic community.

2.5 Derivation of the ZMP

Yamaguchi *et al*. (1993) summarised several underlying assumptions for the ZMP's position calculation as follows. They considered an *n*-degrees-of-freedom biped humanoid robot whose trunk was modelled as a 3-degrees-of-freedom joint. They also assumed: (a) the biped humanoid robot consists of a set of particles (links); (b) the floor is rigid and is not moved by any forces/moments, (c) the foot-ground contact region is a set of contact points, (d) the friction coefficients for rotating about X, Y and Z axes are almost zero at the contact points, and (e) the foot does not slide on the ground's surface.

Based on these assumptions and the ZMP definition presented in section 2.1, the ZMP has to satisfy the following equations:

$$\sum_{i}^{n}\left\{m_i\left(\mathbf{r}_i-\mathbf{p}\right)\times\left(\ddot{\mathbf{r}}_i+\mathbf{g}\right)+\mathbf{I}_i\dot{\boldsymbol{\omega}}_i+\boldsymbol{\omega}_i\times\mathbf{I}_i\boldsymbol{\omega}_i\right\}=\mathbf{M} \tag{2.6}$$

$$\mathbf{M}\times\mathbf{g}=\mathbf{0} \tag{2.7}$$

Where:

- n : is the number of links of the robot
- m_i : is the mass of link i
- $\mathbf{I}_i\equiv diag\left(I_{ix},I_{iy},I_{iz}\right)$: is the inertia tensor of link i
- $\mathbf{r}_i\equiv\left[x_i \quad y_i \quad z_i\right]^T$: is link i's centre-of-mass position
- $\boldsymbol{\omega}_i\equiv\left[\omega_{ix} \quad \omega_{iy} \quad \omega_{iz}\right]^T$: is link i's angular velocity
- $\mathbf{p}\equiv\left[x_{ZMP} \quad y_{ZMP} \quad z_{ZMP}\right]^T$: is the ZMP
- $\mathbf{M}\equiv\left[M_x \quad M_y \quad M_z\right]^T$: is the moment around the ZMP
- $\mathbf{g}\equiv\left[g_x \quad g_y \quad g_z\right]^T$: is gravity acceleration (i.e. $\mathbf{g}=\left[0 \quad 0 \quad -9.81\right]^T$).

Equations (2.6) and (2.7) can be united into one (Dasgupta & Nakamura, 1999):

$$\sum_{i}^{n}\left\{\left(\mathbf{r}_i-\mathbf{p}\right)\times m_i\ddot{\mathbf{r}}_i+\mathbf{I}_i\dot{\boldsymbol{\omega}}_i+\boldsymbol{\omega}_i\times\mathbf{I}_i\boldsymbol{\omega}_i-\left(\mathbf{r}_i-\mathbf{p}\right)\times m_i\mathbf{g}\right\}=\left(0,0,*\right)^T \tag{2.8}$$

Sugihara (2004) presented a simple case to compute the ZMP given $\mathbf{r}_i$, $\ddot{\mathbf{r}}_i$ and $\boldsymbol{\omega}_i$. He explains that since Eqs. (2.6) and (2.7) are composed of five independent equations, with six unknowns there is no unique solution, so usually in most of the conventional methods (Furuta *et al.*, 2001; Li *et al.*, 1992; Okumura *et al.*, 2003; Park & Cho, 2000; Takanishi *et al.*, 1985), the ZMP position is solved by giving a constant height for the ground h and by ignoring each link's inertia, thus:

$$x_{ZMP}=\frac{\sum_i m_i\left\{x_i\left(\ddot{z}_i+g_z\right)-\left(\ddot{x}_i+g_x\right)\left(z_i-h\right)\right\}}{\sum_i m_i\left(\ddot{z}_i+g_z\right)} \tag{2.9}$$

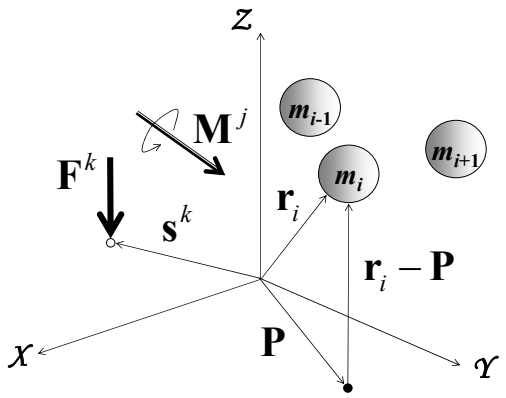
$$y_{ZMP}=\frac{\sum_i m_i\left\{y_i\left(\ddot{z}_i+g_z\right)-\left(\ddot{y}_i+g_y\right)\left(z_i-h\right)\right\}}{\sum_i m_i\left(\ddot{z}_i+g_z\right)} \tag{2.10}$$

Huang et al. (2001) chose not to ignore the inertia and assumed the constant height above is zero ($h = 0$) to yield an accurate version of Eqs. (2.9) and (2.10):

$$\mathbf{(2.11)} \qquad x_{ZMP} = \frac{\sum_i^n m_i \{x_i(\ddot{z}_i + g_z) - (\ddot{x}_i + g_x) z_i\} - I_{iy}\omega_{iy}}{\sum_i^n m_i(\ddot{z}_i + g_z)}$$

$$\mathbf{(2.12)} \qquad y_{ZMP} = \frac{\sum_i^n m_i \{y_i(\ddot{z}_i + g_z) - (\ddot{y}_i + g_y) z_i\} - I_{ix}\omega_{ix}}{\sum_i^n m_i(\ddot{z}_i + g_z)}$$

Equations (2.11) and (2.12) were used also by Zhang (2003).



$\mathbf{P} = [x_p \quad y_p \quad 0]^T$: position vector of **P**

$\mathbf{T} = [T_x \quad T_y \quad T_z]^T$: total torque acting at **P**

$\mathbf{M}^j = [M_x^j \quad M_y^j \quad M_z^j]^T$: external moment *j*

$\mathbf{F}^k = [F_x^k \quad F_y^k \quad F_z^k]^T$: external force *k*

$\mathbf{s}^k = [s_x^k \quad s_y^k \quad s_z^k]^T$: position where $\mathbf{F}^k$ is acting

Figure 2.8: Definitions of vectors for a walking mechanism (adapted from Takanishi *et al*., 1989).

Takanishi *et al*. (1989) presented a derivation for the ZMP equations which included external forces and moments according to the vectors' definitions as shown in **Figure 2.8**. Initially, the following equation of motion at an arbitrary point on the ground **P** is defined, which is acquired by applying d'Alembert principle:

$$\mathbf{(2.13)} \qquad \sum_i^n \{m_i(\mathbf{r}_i - \mathbf{p}) \times (\ddot{\mathbf{r}}_i + \mathbf{g})\} + \mathbf{T} - \sum_j \mathbf{M}^j - \sum_k \{(\mathbf{s}^k - \mathbf{p}) \times \mathbf{F}^k\} = \mathbf{0}$$

As mentioned earlier, the ZMP is the point in which the horizontal components of the moment are equal to zero. With this constraint, the authors modify Eq. (2.13) and retrieve (using simple mathematical operations) the X and Y components of the ZMP:

$$\textbf{(2.14)} \qquad x_{ZMP} = \frac{\sum_i^n m_i \left\{ x_i \left(\ddot{z}_i + g_z \right) - \left(\ddot{x}_i + g_x \right) z_i \right\} + \sum_j M_y^j + \sum_k \left[\mathbf{s}^k \times \mathbf{F}^k \right]_y}{\sum_i^n m_i \left(\ddot{z}_i + g_z \right) - \sum_k F_z^k}$$

$$\textbf{(2.15)} \qquad y_{ZMP} = \frac{\sum_i^n m_i \left\{ x_i \left(\ddot{z}_i + g_z \right) - \left(\ddot{x}_i + g_y \right) z_i \right\} + \sum_j M_x^j + \sum_k \left[\mathbf{s}^k \times \mathbf{F}^k \right]_x}{\sum_i^n m_i \left(\ddot{z}_i + g_z \right) - \sum_k F_z^k}$$

This approach was used in several studies by the authors and other researchers (Li *et al.*, 1991; Park & Rhee, 1998; Takanishi *et al.*, 1990).

Chapter 3

Simulation Environment

Due to the limited allocated time for this project, along with the fact that there was no physical humanoid robot available, it was essential to work using a simulation environment only. This chapter is organised as follows. First, it discusses general aspects of using a simulation environment. Second, it presents the specific simulation program chosen for this project along with the adaptations, contributions and enhancements implemented throughout the project. Finally, it describes the robot model used in this project along with its full specification.

3.1 Real vs. Simulated Robots

Developing autonomous robots is highly complicated and expensive, due to the following reasons. Firstly, this task requires a very calculated and thoughtful design; the developer has to decide on the type (biped, quadruped etc.), size, shape, weight, motor types, sensors and many more. Secondly, the components needed to build such robots are usually expensive due to their lightweight and hi-tech features, compactness, and materials. Thirdly, the building of such robots involves a vast interdisciplinary work in order to combine the numerous components of the robot. Usually, such work is a combination of many engineering fields: software, hardware, mechanical and bio-mechanics.

Therefore, the development of such robots is still governed by full-grown research companies and a small number of universities world-wide. Some examples include: Honda with its Asimo (Hirai *et al.*, 1998), Fujitsu with its HOAP series, Sony and

their SDR/QRIO and Aibo families (Fujita *et al*., 2003), and the Japanese Waseda university with their WL-XXX robotic family (Takanishi *et al.*, 1990; Yamaguchi *et al.*, 1996; Yamaguchi & Takanishi, 1997). In addition, until recently, the expectations of autonomous robots were very mediocre, so it was not always worthwhile to build and develop them.

Nowadays, the robotics field has evolved (although not completely) to a new state in which it has become more accessible to other research organisations, some with much lesser resources. This positive change in the field might be elucidated by the following reasons. First and foremost, the technologies acquired in robotics research in the past thirty years have now been well absorbed by the industry and academics, which enables some firms and universities to build robots at an affordable cost. Consequently, firms such as Sony or Fujitsu offer entertainment robots for a reasonable price.

Secondly, in this modern age computers have increased their computational power which enables the development simulation applications and toolkits. These tools make it possible to work on simulated robots directly, without the use of the physical robots at all. Initially, these tools were designed and developed for specific robots or specific research interest and in many cases were developed directly by robotics researchers. However, they have increasingly been evolving to a more general framework and several simulation software applications now exist which allow users to model and simulate virtual robots in a very realistic fashion (Kuffner *et al.*, 2000; Michel, 2004).

Besides the cost aspects compared to research on real robots, simulated robots have additional advantages which inherently derive from flexible nature of simulation. Primarily, simulation gives a complete and flexible control over the robot's model. In other words, the robot’s shape, weight, sensors, motors, and even appearance are modified straightforwardly to meet the researcher’s requirements, in contrast to physical robots’ adaptations which are very problematic. Moreover, the simulation’s

flexibility is also expressed in the control over the robot's virtual environment (sometimes referred to as "world"). In most simulation software, it is feasible to adjust parameters such as gravity, friction, and mass of objects, or to add external effects such as wind, water, and forces. These capabilities are hardly possible in real life experiments, and if they are, it demands a great deal of effort and resources. Finally, a very important benefit entailed by simulation is to release the researcher from any mechanical or electrical (i.e. hardware) restrictions imposed by physical robots. This means that the researcher can test computation expensive algorithms which would require a lengthily run-time on a real robot's microcontrollers (e.g. GAs). Also, it means that the user does not have to worry about damaging any of the robots components during experiments which enables her to perform better and unlimited testing. This aspect was especially critical for this project's needs since it was trying to develop biped locomotion, which naturally is accompanied by many failures leading the robot to tip over and fall.

In the next following sections, we shall take a glimpse into this project's simulation environment choice – *Webots*^{TM} – describe its main features and present specific adaptations, contributions, and enhancements implemented for this project.

3.2 Webots and Environment Implementations

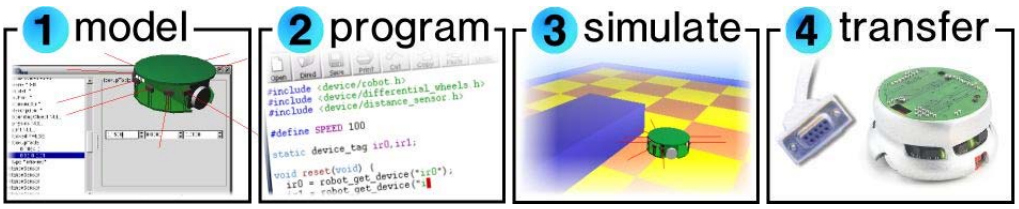


Figure 3.1: Webots development cycle of robot simulation.

The mobile robotics simulation software named Webots[c] enables convenient and rapid modeling, programming and simulating various types of mobile robots (Michel,

[c] Webots is developed by Cyberbotics Ltd., for further details see www.cyberbotics.com.

2004). In the same paper, the author (and proprietor of Webots) summarises the capabilities of Webots as follows which is also well-depicted in **Figure 3.1**:

> "*Webots lets you define and modify a complete mobile robotics setup, even several different robots sharing the same environment. For each object, you can define a number of properties, such as shape, color, texture, mass, friction, etc. You can equip each robot with a large number of available sensors and actuators. You can program these robots using your favorite development environment, simulate them and optionally transfer the resulting programs onto your real robots.*"

The following section will briefly overview the main components of this simulation toolkit, and later will explain how this software was used for this project's purposes.

3.2.1 Overview of Webots

In this section, the key features and capabilities of Webots will be listed, in order to put the immense work related to the simulation environment done in this project into context. Some features were not required for the purposes of this project. However, those are listed for completeness since many other researchers did require them and most probably will in the future.

Michel (2004) gracefully recaps Webots main features and functionalities:

- Webots allows the user to model and simulate *any* complex models of mobile robot and their environments. This ranges from wheeled and multiple legged robots to swimming and flying robots. This can be achieved in two ways, either using a graphical user interface (GUI) called the robot and world editor, or by directly creating a **.wbt* world model file (actually the editor manipulates his file). It should be noted, that modelling the robot in Webots is done as part of the world model. In addition, Webots is

deployed with various modelled robots (e.g. Khepera, Koala, and Sony's Aibo).

- Specifically for the robots, Webots contains a complete collection of various sensors, actuators and other devices usually used as robotic components ready to be plugged into the robot's model. This set comprises of: distance, range, light, and touch sensors, Global Positioning Sensor (GPS), inclinometers, cameras, receivers and emitters, servos (with feedback) etc.
- Webots enables the user to program the robot's controller, which will control the robot's behaviour and optionally a supervisor programming module, which will control and supervise the environment/experiments behaviour. Webots facilitates this by providing robust and well-known programming interfaces in C/C++ or in Java, enabling the programmer to interact with the robot and its environment through a predefined API. Furthermore, the controller can communicate with several third party software applications (e.g. Matlab®, Lisp®) using TCP/IP interface.
- Webots allows automatic transfers of controllers to some popular real mobile robots[d], such as Sony's Aibo®, Lego® Mindstorms®, Khepera®, and Koala®.
- In order to achieve accurate and realistic physical and dynamical simulation of complex robotic devices and their environments Webots relies on ODE (Open Dynamics Engine)[e]. Simulating realistic physics aspects is reflected in three manners: through the world model, through the robot's controller, and most of all through a customised physics programming module.
- Webots utilises virtual time to accomplish highly precise simulation. Therefore, it enables to run simulated experiments considerably faster than in reality. Moreover, the basic simulation time step can be adjusted according to the precision vs. speed trade-off. In addition, Webots enables users to interact with their simulated experiment during its run using the

[d] This feature was irrelevant for this project's purposes since the real robot model was unavailable and also Webots does not contain transfer capability to the robot used.
[e] For detailed information about ODE, see www.ode.org.

friendly and effective GUI, which includes step-by-step mode, changing viewpoint position, orientation and zoom, moving or rotating objects in the scene.

- Webots facilitates creation of AVI or MPEG simulation movies and/or PNG snapshots for web or presentations.

Many details concerning Webots have been left out and the interested reader can explore more of this software in the mentioned website. Obviously, Webots was merely an instrument and not the goal, but as you will see next, a great deal of effort was invested in figuring out Webots internals and taking advantages of its capabilities while establishing some key ideas and concepts of how to create an efficient simulation of a humanoid robot.

3.2.2 Robot Control and Environment Behaviour

The robot controller corresponds to a program embedded in a physical robot in reality. Hence, as explained by Olivier Michel (personal communication) its scope is strictly limited to a real robot's capabilities and cannot access environment information except for sensory data. However, the controller has access to any devices present on the robot such as cameras, the distance sensors, GPS and obviously the actuators (servos).

In the initial phases of this project it was still uncertain which humanoid robot model would be used to develop the balanced locomotion control. Therefore, it was decided to design the robot's controller to be extensible and adaptable to any type of humanoid robot. This would be quite straightforward and intuitive, if done using the object-oriented paradigm. Thus, along with performance considerations, it was natural to choose the C++ programming interface rather than the Java or C programming interfaces all provided by Webots. Many capabilities typical to a humanoid simulation control were organised together in an interface common to a humanoid (or biped) controller. These include: enabling of devices, reading/writing persistent data (including XML parsing implementation for user input motion

sequences), reading and setting joint angles, sending messages to other robots or supervisors, computing the support polygon, ZMP, CoM, and stability margin (see chapter 2), support phase determination, computing the robot's coordinate frame transformation in relation to world coordinates, and online plotting of diagrams (which uses an object-oriented interface to gnuplot[f]).

This approach, besides making the controller program organised, easy to follow and effortlessly maintainable yields other benefits. Primarily, it separates the fundamentals of controlling the simulated robot from the design of the biped locomotion algorithms, thus it allows the researcher to concentrate on the actual development of those algorithms. Secondly, it enables the developer to add several balanced walking mechanisms with various approaches and methods to the same robot, or to apply the same algorithms to different humanoids.

From a different aspect, in many cases the researcher needs to interact with the environment and the simulation continuously. As mentioned previously (section 3.2.1), the supervisor programming module enables to supervise the robotic experiment and to control behaviour of the environment. It can be used to change any property related to objects in the scene, particularly the robot and its components. In other words, it behaves in a god-like manner to: move objects, send/receive messages to/from robots, record robot trajectories, create new objects, display information etc. These capabilities were essential to this project's needs as explained in the following.

In some of the experiments conducted during the project, it was necessary to reset the robot's position to the initial location, in order to perform another iteration of the walking algorithm. The innate way to accomplish this in Webots was utilise the supervisor. The supervisor continuously listens on its receiver for incoming messages and whenever the robot requests it, the supervisor moves it back to its initial position at the beginning of the experiment. Using the same idea, the supervisor receives successive updates of the ZMP and CoM positions from the robot's controller (which

[f] For detailed description of *gnuplot*, see: http://www.gnuplot.info.

computes them), and after each such message the supervisor immediately visualises those positions in the Webots simulation window. The visualisation is twofold, on one hand the supervisor displays the actual 3D coordinates of the ZMP and CoM at the upper-left corner of the scene, and on the other hand it moves the corresponding crosshair indicators of the ZMP/CoM (added to the world model) to their updated positions represented in the absolute world coordinates. This feature is shown in **Figure 3.2**.

The supervisor was designed in a similar fashion to the controller described previously with the same concept in mind. Thus, it is easy to enhance the supervisor's features and add further capabilities.

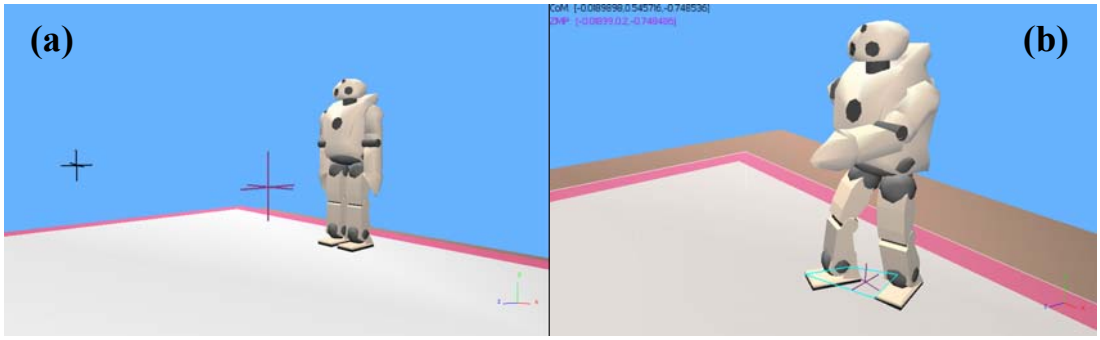


Figure 3.2: The supervisor updating the CoM and ZMP: (a) initially, the CoM and ZMP indicators (black and purple crosshairs respectively) are located at (-0.5, 0.5, 0) and (0.5, 0.5, 0). (b) During the run, the supervisor updates both the labels and indicators as received from the robot's controller.

3.2.3 Implementing Customised Physics

Section 3.2.1 mentioned that adjusting the realistic physics used in the simulation is threefold. This project employed the first method a bit, i.e. by modifying certain parameters in the world model, and mainly used the last method, namely by creating a customised physics control module. Let us first discuss the former and then concentrate on the latter.

By specifying a "Physics" node attached to any of the robot's components in the model, the user can specify density distribution or mass, inertia tensors, static and

kinematic friction coefficients, force dependent slip, and bounciness. It was decided to specify the strict mass of each component instead of letting ODE compute it, using the density distribution parameter, since this enables the verification of the ZMP calculation by comparing it to manual offline computation. In addition, it was left for Webots (using ODE) to compute implicitly the inertia tensors, which are required for the ZMP computation (see Chapter 2). Moreover, Webots allows setting the maximal force or acceleration, maximal velocity, and PID[g] gain parameter to be used by a servo device. Experiments with these parameters were carried out and the optimal values that perform accurate movement of the servos as specified by the controller were thus found.

On the other hand, Webots capability of creating a customised physics module for the simulations was fully utilised. This was achieved by creating a custom shared library which is loaded as a separate process by Webots at run-time and which contains function calls to the underlying ODE. This capability was especially useful to gather dynamics information (e.g. position, angular velocity, inertia tensors etc.) of every solid in the environment, specifically the robot's particles. Moreover, by accessing the underlying ODE collision detection system, it was possible to detect the contact points between the foot and ground at any simulation step. Then, by applying geometric convex-hull, the support polygon could be extracted. In addition, by using the customised physics module, it was possible to add external forces and disturbances to the environment (see section 5.5).

One might question the purpose of carrying out these non-trivial actions in an independent external process such as the physics module. Why should it not be performed by the robot itself? To answer this question the reader is referred to the aforementioned argument (beginning of section 3.2.2) which states that the robot's controller can only access information available as sensory data. Accessing

[g] A three mode control action in which a controller has time Proportionate, Integral (auto reset) and Derivative rate action. In this context, this parameter is used to compute the velocity from the requested final position. A small value yields a longer time needed to reach the target position, while a rather large value yields instabilities in reaching the target position.

equivalent information through sensors would imply attaching many of the available types of sensors (e.g. GPS, inclinometers) to each particle, which would be very costly. In addition, there are not sufficient sensors implemented to detect foot-ground contacts in the controller. Thus, this approach simplifies the collection of this vital information and delivers much more precise data.

Moreover, through the physics module it was possible to directly manipulate and display graphical 3D visualisations using OpenGL[h]. Specifically, this feature was utilised to display the support polygon of the robot at every simulation step (see **Figure 3.2**, the support polygon coloured cyan). However, in general, one could supplement with additional features such as displaying arrows representing forces or moments acting on the robot.

Yet, the customised physics shared library and the robot's controller program are two independent processes working concurrently under Webots during run-time, thus how can they share any data between them? Clearly, in any modern operating system, separate processes running in parallel cannot share internal data structures simply by using the same reference, pointer or even names since each process is allotted a non-overlapping part of the available memory for execution. In addition, even if those processes could do so, there would still be issues regarding mutual exclusion and concurrency. It was therefore inevitable to implement Inter-Process Communication (IPC) between the robot's controller and the physics share library.

Performing IPC in UNIX systems is achieved by several methods: Message Queues, Pipes, FIFOs, and Shared memory, which all have two variants, the POSIX one and the System V one. It was chosen to implement the Shared memory option since it is commonly the fastest (Stevens, 1999) and the System V variant was adopted since it gave the impression that it was easier to understand and much more comprehensive. In addition, to assure correctness through mutual exclusion, the corresponding

[h] A detailed explanation about OpenGL is presented in http://www.opengl.org.

System V Semaphores were implemented and thoroughly tested in the initial stages of development.

This section demonstrates the breadth of technologies involved in successfully accomplishing this project, and the depth of knowledge required to implement such technologies.

3.3 The Robot Model

So far, the discussion was not limited to a particular humanoid robot model and described simulation of any type of biped and humanoid robot. Naturally, in the initial stages of this project it was inevitable to decide on the robot models, which will be used. This question entails many sub-questions such as: How many and which DOFs should the robot have? Where should one place the joints in the robot's body? What parameters such as weight, size, mass, and joint limits should be set for the robot?

In theory, a competent and balanced locomotion control should be general enough to work sufficiently on any type of robot with humanoid characteristics. However, as seen in earlier chapters, sometimes the design of the robot's structure might affect the prosperity of the locomotion algorithm. For example, a humanoid model which lacks degrees-of-freedom in places crucial for the locomotion control (e.g. limited torso motion) might fail to walk stably, while in contrast, a model with passive design of the soles (Collins *et al.*, 2001) can enhance the locomotion stability. Hence, these issues, which are linked with the robot's model, make it problematic in evaluation of proposed algorithms.

In addition to the mentioned above, time scope for this project was very limited, and thus it was not affordable to take up time and create a robot model from scratch. Consequently, it was decided to use a robot model, which is firstly based on an existing humanoid robot and secondly, is already modelled for Webots. This decision

was undertaken due to the time constraint already mentioned along with the following reasons.

Most existing robots were created after many design iterations, the researchers significantly faced the above problems and dilemmas, and they invested a great deal of thought already – why not take advantage of that?! Moreover, many technical obstacles were confronted by the creators of such models for Webots and thus it could establish some help of support if collaboration with those creators was initiated. Finally, in the author's opinion, it is much more impressive to present a robust locomotion control on a well-known robot model, and this would make much more impact on the robotics community, rather than making some unknown robot walk balanced, especially when the proprietors of these popular robots still keep their technology so elusive. The next following sections will describe the exact robot model specification used in this project.

3.4 Sony QRIO-like Specification

On September 19, 2003, Sony Corporation officially launched their new small humanoid named QRIO. It is based on the SDR-4X prototype (Sony Dream Robot) presented in (Fujita *et al.*, 2003) and is extremely popular due to extensive publicity and numerous conferences it has been presented[i].

The modelling of the QRIO robot used here has been done for a preceding project (Mojon, 2003). Since then, the model has been extensively used by a vast forum of users through two world-wide programming competitions held by Cyberbotics Ltd., namely the "*Robot Judo Contest*" and the "*Roboka – Cyber Robot Tournament*". In the former, the author has taken part and reached the honourable 5th place using basic motion trajectory planning and tactics. Furthermore, this model was employed in

[i] The full specification of QRIO robot can be found in
http://www.sony.net/SonyInfo/News/Press/200203/02-0319E

another recent biped locomotion research combining CPG and GAs (Mojon, 2004). **Figure 3.3** compares the original QRIO model with the Webots prototype.

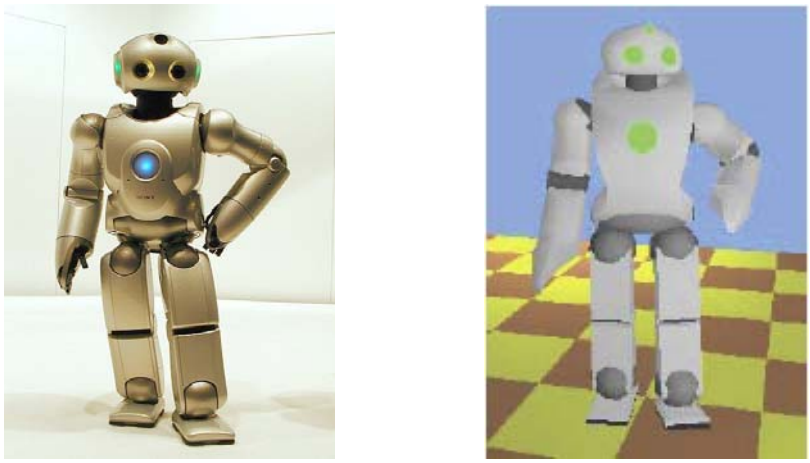


Figure 3.3: The original Sony QRIO robot compared to the Webots model.

It should be noted that the virtual model created for Webots is not completely accurate and was created based on images, video clips and available material, since Sony has not publicised an official document describing the exact specification and design which the built the robot upon (Olivier Michel, personal communication).

Table 3.1: The robot's weight parameters.

Link	***Weight [kg]***
Body	3.05
Waist	0.325
Hip	0.009 (×2)
Thigh	0.325 (×2)
Shank	0.4 (×2)
Foot	0.115 (×2)
Upper Arm	0.2 (×2)
Lower Arm	0.25 (×2)
Head	0.4
Total	**6.373**

Table 3.2: Link lengths specification of the Sony QRIO-like model.

Link	***Length [m]***	***Link***	***Length [m]***
ARM1	0.105	LEG1	0.048
ARM2	0.082	LEG2	0.024
ARM3	0.012	LEG3	0.013
ARM4	0.142	LEG4	0.116
BACK1	0.081	LEG5	0.104
BODY1	0.048	LEG6	0.048
BODY2	0.315	WAIST1	0.032
HEAD1	0.033	WAIST2	0.013
HEAD2	0.08		

Table 3.1 shows the weight specification for the robot's particles which slightly differ from the original Webots model. **Table 3.2** and **Table 3.3** along with **Figure 3.4** complete the specification of the virtual model of the QRIO robot used in this project.

Table 3.3: Joints and limits specification of the Sony QRIO-like model.

Part	***Joint Name***	***Motion***	***Joint Limits [rad]***	
			Min.	***Max.***
Torso (2 DOFs)	back_1	Torso pitch	-1.0	+2.0
	back_2	Torso roll	-0.5	+0.5
Left Leg (6 DOFs)	left_hip_1	Left hip yaw	-1.4	+1.7
	left_hip_2	Left hip pitch	-0.9	+1.6
	left_hip_3	Left hip roll	-0.5	+1.0
	left_knee	Left knee pitch	-0.2	+2.5
	left_ankle_1	Left ankle pitch	-0.7	+0.7
	left_ankle_2	Left ankle roll	-0.7	+0.7
Right Leg (6 DOFs)	right_hip_1	Right hip yaw	-1.4	+1.7
	right_hip_2	Right hip pitch	-0.9	+1.6
	right_hip_3	Right hip roll	-0.5	+1.0
	right_knee	Right knee pitch	-0.2	+2.5
	right_ankle_1	Right ankle pitch	-0.7	+0.7
	right_ankle_2	Right ankle roll	-0.7	+0.7
Left Arm (3 DOFs)	left_shoulder_1	Left shoulder pitch	-3.14	+3.14
	left_shoulder_2	Left shoulder roll	0.0	+3.14
	left_elbow	Left elbow pitch	0.0	+2.8
Right Arm (3 DOFs)	right_shoulder_1	Right shoulder pitch	-3.14	+3.14
	right_shoulder_2	Right shoulder roll	0.0	+3.14
	right_elbow	Right elbow pitch	0.0	+2.8
Head (2 DOFs)	neck	Neck yaw	-2.0	+2.0
	neck_tilt	Neck pitch	-0.3	+1.1

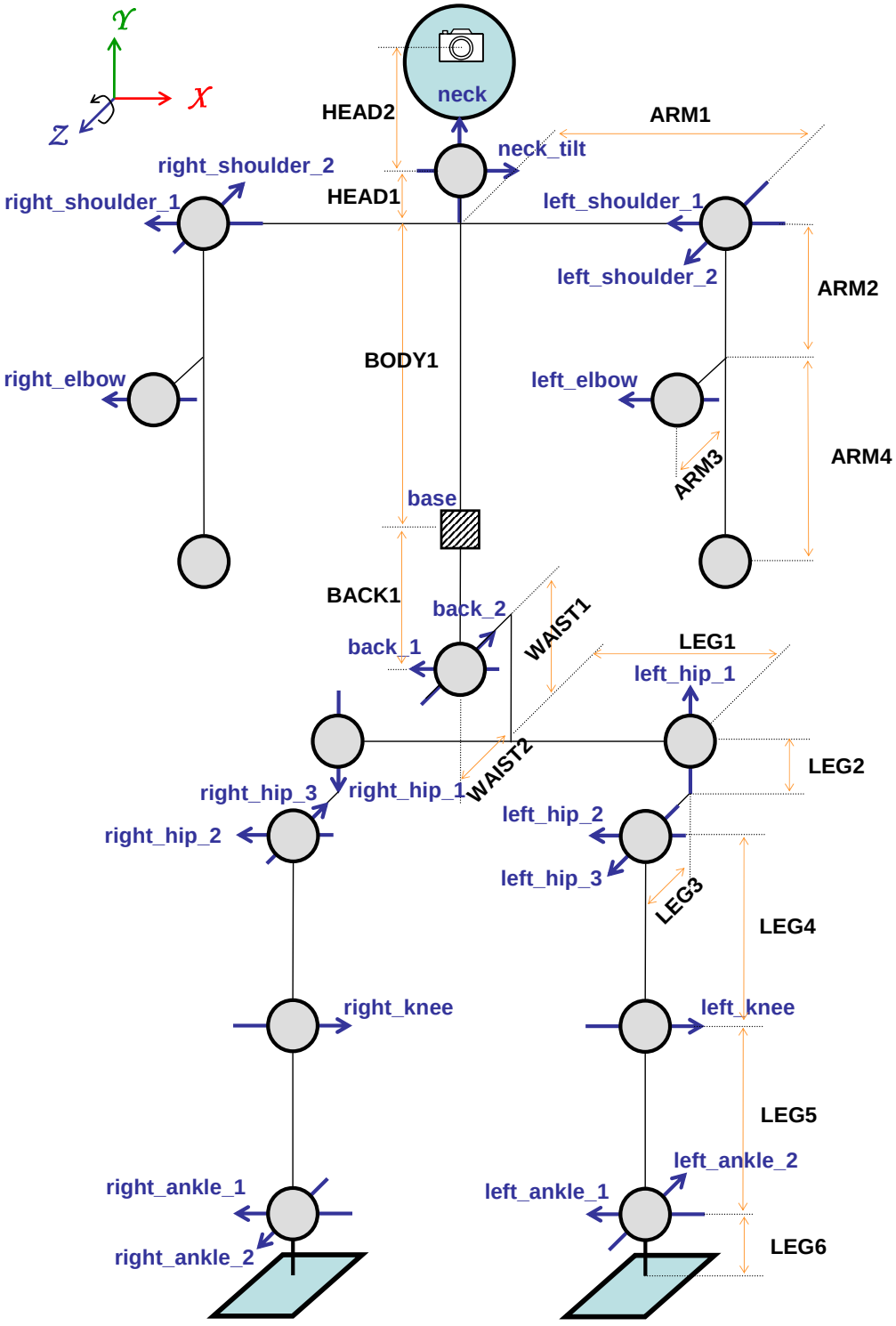


Figure 3.4: Joint positions and length specification of the Sony QRIO-like model.

Chapter 4

The ZMP Jacobian Counterbalance

4.1 Introduction

As discussed in Chapter 1, the prevalent biped research adheres to two approaches in order to accomplish effective and robust locomotion. In the first approach, the gait trajectories are computed *online* in agreement with the robot's intentions and sensory data (Kagami *et al.*, 2000; Kajita *et al.*, 1992; Park & Chung, 1999). In the second approach, a collection of trajectories is computed *offline* (Denk & Schmidt, 2001; Hardt *et al.*, 2002) and the robot chooses the most suitable predefined trajectory corresponding to the current state it is in. Biped locomotion appropriate gaits are usually subject to numerous constraints and in many cases the gait must satisfy various criteria such as smoothness, anthropomorphic nature, energy efficiency, and stability.

Several studies have shown that appropriate stride and stepping trajectories can be computed by advanced numerical optimal control methods which take into account various constraints such as feet-ground contact forces, dynamic stability, torques limits in the joints etc. (Kagami *et al.*, 2003; Zhu & Kawamura, 2003). In general, however, such calculations are hardly feasible in online approaches, given the present computational hardware. Therefore, most of the biped locomotion research concentrates on offline approaches to gait trajectories generation.

In this context, the robot's inability to perform motions for which the trajectories are not suitable or even unavailable is the main drawback of predefined trajectories. In relation to dynamic stability, if the predefined trajectories do not meet the dynamic

stability criteria (e.g. the ZMP is no longer inside the support polygon) in slightly different situations, it would be desirable to modify them in such a way that they become adequate. Hence, a significant challenge is some feedback-based online adaptation of predefined trajectories which enhances the bipedal locomotion stability and efficacy.

This chapter will propose the ***ZMP Jacobian Counterbalance*** method which adjusts the motion of specific body particles during the biped or humanoid walking according to the ZMP's Cartesian position and thus enhances the dynamic stability the robot's locomotion.

A few recent studies focused on online modification of predefined motion patterns to either ensure gait stability by having some control over the ZMP (Huang *et al.*, 2000), or for adapting predefined trajectories suitable for simple walking in a plane, to trajectories suitable to walking on slopes or climbing stairs (Kajita *et al.*, 2003). Modification of predefined trajectories using Jacobians was introduced and termed *Jacobi Compensation* in a recent work (Sobotka *et al.*, 2003; Wollherr *et al.*, 2003). This method allows shifting parts of the robot in Cartesian directions and thereby altering the humanoid's posture to compensate for errors or adapt trajectories, in order to make them applicable to situations which are different than those they were generated for. The use of this concept to adjust the ZMP's position and thereby improving the dynamic stability of a humanoid is considered novel.

This chapter is organised as follows: The following section explains the concept of the ZMP Jacobian. Section 4.3 discusses the complex problem of generating biped and humanoid motion and presents the strategy undertaken in this work. The bipedal motion control using the ZMP Jacobian Counterbalance is presented in Section 4.4. A brief summary of the chapter follows in Section 4.5.

4.2 The ZMP Jacobian Concept

In this section, the concept of the ZMP Jacobian for a general multi-body humanoid robot is formulated. The ZMP's position, $\boldsymbol{p}_{ZMP}$, of a rigid multi-body system is a general function of the system's joint angles $\boldsymbol{q}$ as follows:

$$\boldsymbol{p}_{ZMP} \equiv \boldsymbol{p}_{ZMP}(\boldsymbol{q}) \tag{4.1}$$

This relation implies that a Jacobian matrix exists which is formulated as:

$$d\boldsymbol{p}_{ZMP} = \frac{\partial \boldsymbol{p}_{ZMP}}{\partial \boldsymbol{q}} d\boldsymbol{q} \tag{4.2}$$

In other words, to modify the 3D Cartesian position of the ZMP, $\boldsymbol{p}_{ZMP} \equiv [x_Z \quad y_Z \quad z_Z] \in \Re^3$ by $\Delta\boldsymbol{p}_{ZMP}$ in 3D space, it is required to modify the joint angles vector of the multi-body $\boldsymbol{q} \in \Re^n$ by:

$$\Delta\boldsymbol{q} = f(\Delta\boldsymbol{p}_{ZMP}), \tag{4.3}$$

where $f(\bullet)$ is a function which transforms the Cartesian motion of the ZMP $\Delta\boldsymbol{p}_{ZMP}$ into the joint-space motion $\Delta\boldsymbol{q}$.

This $f(\bullet)$ relationship between the Cartesian motion of the ZMP and the joint-space motion is generally known as a Jacobian matrix, and here it is formulated as follows:

$$\boldsymbol{J}_{ZMP}(\boldsymbol{q}_a) = \left[\frac{\partial \boldsymbol{p}_{ZMP}}{\partial q_1} \quad \cdots \quad \frac{\partial \boldsymbol{p}_{ZMP}}{\partial q_n} \right]_{\boldsymbol{q}=\boldsymbol{q}_a} \in \Re^{3\times n} \tag{4.4}$$

This Jacobian is a function of the "*actual*" joint angles $\boldsymbol{q}_a \in \Re^n$ (i.e. the joint angles set at the robot's joints at each given moment), and it essentially maps the velocity vector $\dot{\boldsymbol{q}}$ in joint-space to the velocity of the ZMP $\dot{\boldsymbol{p}}_{ZMP} \in \Re^3$ in Cartesian space according to the following equation:

$$\dot{\boldsymbol{p}}_{ZMP} = \boldsymbol{J}_{ZMP}(\boldsymbol{q}_a)\cdot\dot{\boldsymbol{q}} \tag{4.5}$$

However, from the ZMP's definition and derivation in Chapter 1, it is known that the ZMP is always situated on the ground and only moves on the horizontal ground plane. Thus, there is no meaning to the vertical component of the ZMP, z_Z, in this context and Eq. (4.4) can ignore it and use only the horizontal position of the ZMP $\boldsymbol{p}^H_{ZMP} \in \Re^2$:

$$\boldsymbol{J}_{ZMP}(\boldsymbol{q}_a) = \left[\begin{array}{ccc} \dfrac{\partial \boldsymbol{p}^H_{ZMP}}{\partial q_1} & \cdots & \dfrac{\partial \boldsymbol{p}^H_{ZMP}}{\partial q_n} \end{array}\right]_{\boldsymbol{q}=\boldsymbol{q}_a} = \left[\begin{array}{ccc} \dfrac{\partial x_Z}{\partial q_1} & \cdots & \dfrac{\partial x_Z}{\partial q_n} \\ \dfrac{\partial y_Z}{\partial q_1} & \cdots & \dfrac{\partial y_Z}{\partial q_n} \end{array}\right]_{\boldsymbol{q}=\boldsymbol{q}_a} \in \Re^{2\times n} \tag{4.6}$$

As shall be seen in section 4.4, this Jacobian's inverse will be employed to compute the correction velocity in joint-space from a desired correction required for the ZMP, thereby obtaining the position (joint angles) adjustment in joint-space $\Delta\boldsymbol{q}$. But first, it is necessary to explain how one obtains such predefined motion trajectories, mentioned in section 4.1, which will undergo modification, and this will be presented in the following section.

4.3 The Humanoid Motion Generation

Generating motions for humanoid or even bipedal systems is a highly non-trivial problem, which encompasses a great deal of the robotics research community. Automatic generation or planning of full-body motion for humanoid robots is still a complex computational challenge due to the following reasons. First, humanoids are structured with a large number of degrees of freedom. Second, they possess intricate kinematic and dynamic models. Finally, stability constraints should be satisfied in order to prevent the humanoid from falling down during the motion.

Several approaches have been employed in the humanoid research scope to accomplish plans or trajectories for the robot's joints, which perform a desired

motion (e.g. walking). In the next section, some of these approaches and methods will be discussed, highlighting the difficulties in utilising them for the nature of this study.

4.3.1 Motion Generation Strategies

The simplest strategy to generate full-body motion trajectories for humanoids is to develop a special motion creator with a rich graphical user interface that enables the user to design and verify its motion trajectories for the robot used (Fujita *et al.*, 2003). There are obvious problems with such a strategy. Primarily, those applications are usually not very generalised and are too customised for a specific humanoid model. In addition, the effort and resources, which are required to build such an application, is tremendous and clearly inappropriate for this project's nature. Furthermore, such sophisticated tools are certainly efficient but they are distributed commercially and are not very accessible to the academic research. In contrast, the equivalent Open Source tools available have still not reached the level of maturity (Egger, 2004) and were found excessively time-consuming to use and produced poor motion plans.

Another strategy for generating humanoid motion involves capturing full-body motions of a human performer by using an optical tracking device (Dasgupta & Nakamura, 1999; Riley *et al.*, 2003; Ude *et al.*, 2004). This tracking device provides 3D locations of identified active markers, which are visible. However, this approach entails further complexities such as: (1) distinguishing the kinematic model human performer, (2) the imitated motion's joint angle trajectories must be estimated and generally are not accurate, and (3) appropriately transforming the captured motion to the kinematics of the humanoid. The latter is especially non-trivial and is in itself a whole branch of research. Besides, the mentioned problems one has to acquire such a human capture system, which is expensive.

Yet a different approach to produce motion trajectories for humanoids is by heavily relying on the kinematics and dynamics model of the humanoid entwined with obstacle avoidance algorithms. Such methods automatically generate collision-free

and stable motions (dynamic or static) given the full-body posture goals (Kuffner *et al*., 2003). The idea is that obstacle and balance constraints are imposed by incremental search of valid motions. Given initial and goal configurations which correspond to collision-free and statically-stable body postures, the motion plan produced can be further smoothed (e.g. by minimum-jerk constraints) and thus transformed into a collision-free and dynamically-stable motion for the entire body. However, there are some potential complications with this approach as described next.

Primarily, those methods require a full-knowledge of the environment for collision checking. In addition, the inverse kinematics and dynamics should be extracted accurately. An example of extracting necessary information for kinematics of a humanoid robot is presented in Appendix A. Furthermore, the inverse dynamics and kinematics are considered highly difficult and sometimes impractical and they can pose uniqueness and existence problems of solutions. Hence, it is necessary to solve the inverse kinematics/dynamics of the humanoid's motion using heuristic search algorithms and/or numerical optimisation techniques, which entails an expensive computational cost.

Due to all of the factors mentioned above, it was decided to resort to much simpler methods to achieve motion trajectories as presented in the following section.

4.3.2 Intuitive Motion Trajectories Generation

Fortunately, due to the author's participation in worldwide programming contest (as mentioned in Chapter 3Chapter 1) it was possible to utilise the previous efforts for the purpose of motion trajectories. The motions trajectories consist of various motion definitions, which specify control points (also known as *via points*) for each joint's angle positions along with a time frame sequence corresponding to these via points.

```
<?xml version="1.0" encoding="UTF-8" ?>
<GLOBAL_DEF>
    <DEFINE>
        <JOINT_DEF number="0"> joint_0_name </JOINT_DEF>
        ...
        <JOINT_DEF number="n"> joint_n_name </JOINT_DEF>
    </DEFINE>
    <MOTION_DEF>
        <SEQ id="FIRST_STEP" comment="Start from standing; right leg forward">
            <JOINT id="joint_i_name"> 0.2 0.15 0 </JOINT>
            ...
            <JOINT id="joint_m_name"> -0.3 -0.3 -0.3 </JOINT>
            <TIME_DEF type="interval"> 0.512 </TIME_DEF>
        </SEQ>

        <SEQ id="WALK_LEFT" comment="Start left leg behind; move forward">
            <JOINT id="joint_x_name"> -0.07 -0.16 0 </JOINT>
            ...
            <JOINT id="joint_y_name"> -0.3 -0.3 -0.3 </JOINT>
            <TIME_DEF type="explicit"> 0.512 1.024 1.280 </TIME_DEF>
        </SEQ>
        ...
    </MOTION_DEF>
</GLOBAL_DEF>
```

Figure 4.1: The motion definitions XML file format.

These motion definitions are stored in an XML file (see **Figure 4.1**) which is loaded by the robot's controller using the previously mentioned XML parsing capability implemented (section 3.2.2). In other words, this motion-planning file depicts for each motion type, which angle every joint will be at a specified time frame. Note that the creation of such motion definitions is a tremendously daunting task, since it requires manual tweaking of the aforementioned via points to produce sufficiently stable motions.

4.3.3 Trajectory Interpolation

Once basic gait motions are defined, it is essential to apply some smoothness to them, otherwise they would result in "*bang-bang*" motions when the humanoid tries to execute them. This is due to the fact that those motions are defined only in positional terms, and will clearly produce unstable and undesirable motion.

Smoothing these trajectories is better known as trajectory interpolation and is used not only robotics but also in computer graphics applications, astronomy, physics etc.

Thus, it was necessary to implement an interpolation method for the robot's motion control. Given the motion definitions (via points and time frames), the interpolation method will create a function which estimates the joint angle values within each two via points in the sequences.

Several well-known interpolation methods were implemented for this project, including: Akima's Cubic Spline method (Akima, 1970), Bezier[j] Cubic Spline method, Cardinal Cubic Spline method (Schoenberg, 1969), Fritsch-Carlson Monotone Cubic Spline method (Fritsch & Carlson, 1980) and a standard polynomial interpolation. In addition, the simple interpolation method described in Appendix A was implemented.

Figure 4.2 compares the various implemented methods on a sample sequence of via points taken from one of the motions. As shown in the figure, some of the mentioned interpolations have undesirable properties. For example, the Bezier interpolation does not necessarily pass through the given via points, while the Polynomial interpolation passes through them but is unstable within two via points, thus adding effects that the motion designer clearly will not want. Actually, the best results are achieved by the state-of-the-art Fritsch-Carlson interpolation, but it requires far more computation than the simple method. Thus, it was decided to employ the simple interpolation method (from section B.2.2) as it gracefully balances the accuracy-runtime tradeoff.

[j] A Bezier interpolation is a mathematically defined spline commonly used in 2D graphic applications and defined by four points: the initial and terminating position ("anchors") and two middle points ("handles"). The Bezier spline's shape is altered by moving the handles. The mathematical method to create Bezier splines was invented by Pierre Bézier in the late 1960's for the Renault automobiles manufacture.

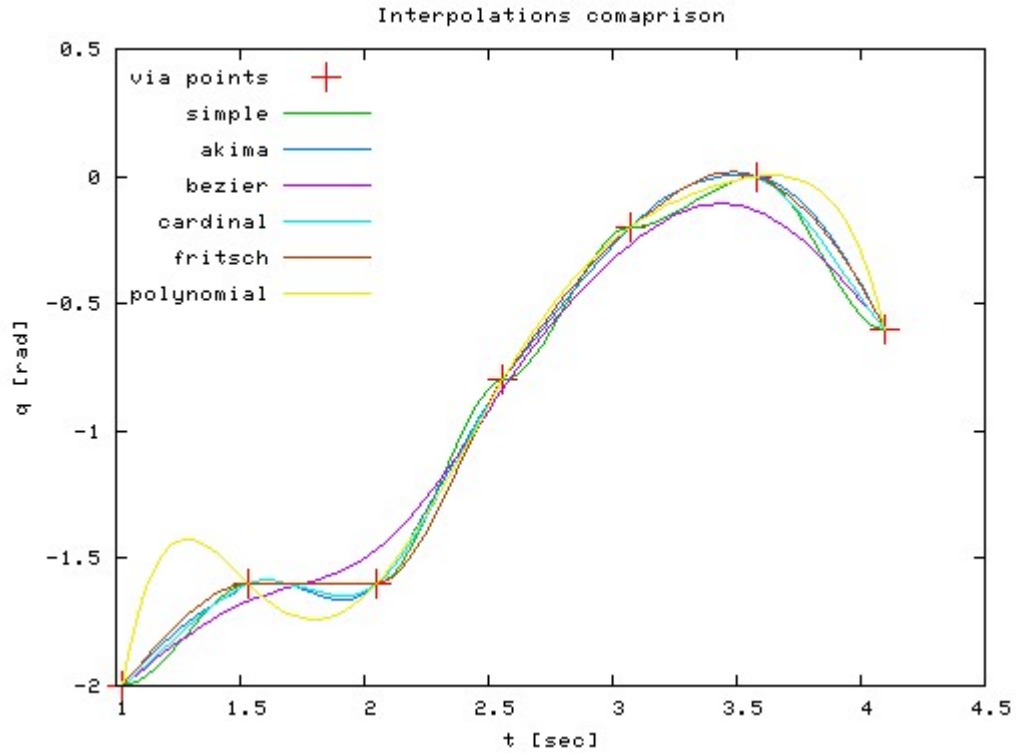


Figure 4.2: Comparing the various implemented interpolations results on a typical sequence of via points for a joint.

4.4 ZMP Jacobian Counterbalance Motion Control

Evidently, when predefined optimal motion control trajectories are given as input to the humanoid's controller, some deviations from the desired stability criteria or other constraints commonly occur. This may be explained due to the following: joint control noise, link dynamics flexibilities, gear loss or motor backlash, modelling errors, and environmental disturbances inflicted on the robot. Hence, the walking performance is degraded in terms of the control criteria, specifically here – the dynamic stability as represented by the ZMP. Normally, a control strategy involving sensory feedback has to be utilised in order to deal with those deviations.

In this section, the ZMP Jacobian Counterbalance method is described, which adapts the given predefined motion trajectories for the humanoid's walk, in terms of the ZMP position to reduce the dynamic stability criteria deviation and thus improve the

walking performance. Note that this method can be generalised to other task-dependent Cartesian deviations such as a specific joint position coordinates and the CoM Cartesian position instead of the ZMP's position.

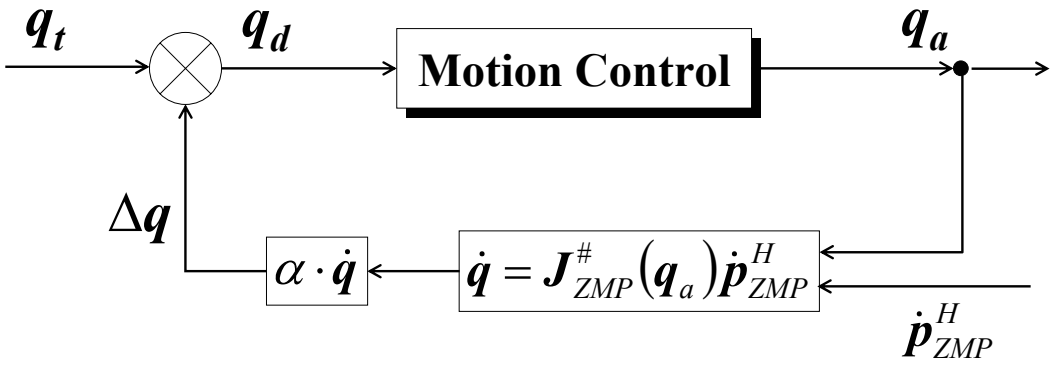


Figure 4.3: The outline of the ZMP Jacobian Counterbalance method. The predefined trajectory $\boldsymbol{q}_t$ is adjusted by linearly superimposing the correction vector $\Delta\boldsymbol{q}$ to create the desired posture $\boldsymbol{q}_d$. The required counterbalance correction $\Delta\boldsymbol{p}^{H}_{ZMP}$ is converted to the joint-space using the Jacobian's inverse (Sobotka *et al.*, 2003).

The aim of this method is to alter the humanoid's ZMP horizontal position $\boldsymbol{p}^{H}_{ZMP} \in \Re^2$ on the ground by $\Delta\boldsymbol{p}_{ZMP}$ whenever $\boldsymbol{p}^{H}_{ZMP}$ is outside the support polygon $\boldsymbol{S}$. The desired offset $\Delta\boldsymbol{p}_{ZMP}$ will be determined by the shortest distance of $\boldsymbol{p}^{H}_{ZMP}$ from the boundary of $\boldsymbol{S}$. The reason the ZMP position is moved only to the boundary and not further inside the support polygon is that the method does not want to cause serious side effects, which might cause the robot not to be able to perform the walk at all.

The joint angles vector $\boldsymbol{q}_t$, acquired from the predefined motion trajectories, should be adjusted by linearly superimposing a joint angles correction vector $\Delta\boldsymbol{q}$, related to $\Delta\boldsymbol{p}_{ZMP}$ as depicted in Eq. (4.3). As section 4.2 describes, this relation is the ZMP Jacobian $\boldsymbol{J}_{ZMP}(\boldsymbol{q}_a)$. Because of this linear superimposition, (see **Figure 4.3**) we obtain a new *desired* posture vector:

$$\boldsymbol{q}_d = \boldsymbol{q}_t + \Delta\boldsymbol{q} \qquad \textbf{(4.7)}$$

4.4.1 Computing the ZMP Jacobian Mapping

So far, the specific ZMP Jacobian instance used at each simulation step of the method has not been discussed. At each given simulation step in which a counterbalance is required, the method obtains an appropriate ZMP Jacobian to perform the corrections. However, how does it retrieve an appropriate Jacobian as such? What kind of relationship exists in the mapping $\boldsymbol{J}_{ZMP}(\boldsymbol{q}_a)$?

After a simple statistical examination performed, it was suggested that $\boldsymbol{J}_{ZMP}(\boldsymbol{q}_a)$ is a non-linear mapping. **Figure 4.4** illustrates the error-bar plots of the 95% confidence intervals on the residuals from the multiple linear regression (least squares fit) performed on a single column of $\boldsymbol{J}_{ZMP}(\boldsymbol{q}_a)$ (see Eq. (4.6)) for a sample of 20 consecutive $\boldsymbol{q}_a$ vectors. The particular column belongs to the torso pitch joint. The residuals appear in the plots in case order. Also, the *coefficient of determination* (R^2) is shown with values (59% and 65%) which clearly depict that a linear model is inappropriate, i.e. the $\boldsymbol{J}_{ZMP}(\boldsymbol{q}_a)$ mapping is highly non-linear. Moreover, one can see that *outliers* exist in the data, which makes the retrieval of a suitable ZMP Jacobian even harder.

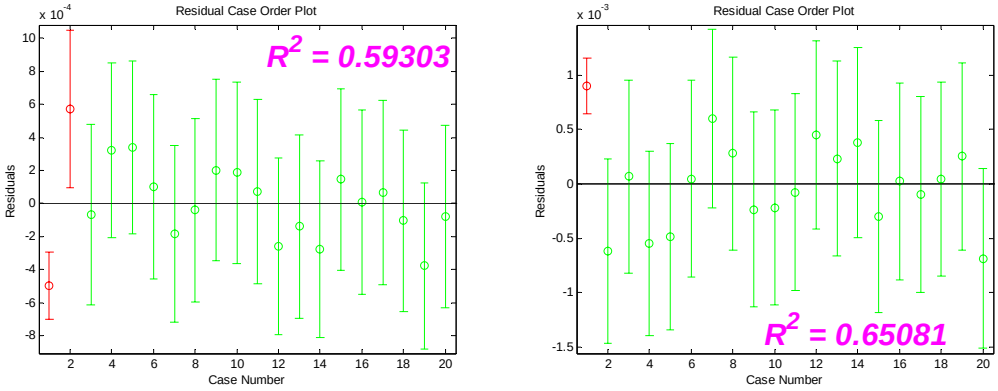


Figure 4.4: Error-bar plots of the 95% confidence intervals on the residuals from the least squares fit of $\partial \boldsymbol{p}^{H}_{ZMP}$ on ∂q_1 (torso pitch). The red error-bars indicate outliers.

In order to deal with the changing Jacobians given the changing posture vectors ($\boldsymbol{q}_a$), it was decided to store the $\boldsymbol{J}_{ZMP}(\boldsymbol{q}_a)$ mapping as a table and re-compute it after every

run of corrections. Furthermore, to overcome the non-linearity, *Nearest-Neighbour* (NN) method was used as follows. Given a new posture vector $\boldsymbol{q}$, at a simulation step demanding counterbalance, the controller would search the table for the nearest $\boldsymbol{q}_a$' entry to $\boldsymbol{q}$, in terms of squared distance, and pull out its corresponding $\boldsymbol{J}_{ZMP}(\boldsymbol{q}_a')$.

One might question the proficiency of using a simple NN method for non-linear interpolation such as here. However, as presented in Chapter 5, the positive results achieved seemed to justify this decision. In addition, other alternative ideas to this approach will be discussed in 7.2.

4.4.2 Computing the Correction Vector

As stated in section 4.2, the ZMP Jacobian relates the velocity vector $\dot{\boldsymbol{q}}$ in joint-space to the ZMP velocity in the horizontal plane as follows:

$$\dot{\boldsymbol{p}}^{H}_{ZMP} = \boldsymbol{J}_{ZMP}(\boldsymbol{q}_a)\cdot\dot{\boldsymbol{q}} \tag{4.8}$$

This relation holds the key to the whole method. From Eq. (4.8) it is possible to compute the correction velocity in joint-space given a desired correction required for the ZMP. This accomplished using the ZMP Jacobian's inverse (denoted by #):

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}_{ZMP}(\boldsymbol{q}_a)\cdot\dot{\boldsymbol{p}}^{H}_{ZMP} \tag{4.9}$$

The inverse ZMP Jacobian in Eq. (4.9) is obtained using the Moore-Penrose pseudo-inverse method, which minimises the Euclidian norm $\|\dot{\boldsymbol{q}}\|_2$ and is presented in Appendix C. This pseudo-inverse computes to:

$$\boldsymbol{J}^{\#}_{ZMP}(\boldsymbol{q}_a) = (\boldsymbol{J}_{ZMP})^T\left[\boldsymbol{J}_{ZMP}(\boldsymbol{J}_{ZMP})^T\right]^{-1} \text{ or } \left[(\boldsymbol{J}_{ZMP})^T\boldsymbol{J}_{ZMP}\right]^{-1}(\boldsymbol{J}_{ZMP})^T \tag{4.10}$$

depends if the Jacobian is row rank, or column rank deficient. The joint velocity vector from Eq. (4.9) is then integrated to obtain the posture correction $\Delta\boldsymbol{q}$ in joint space. However, since the simulation executes at a fixed time steps, all the velocities

are actually infinitesimal positional offsets. Hence, there is no need for integration. Also, as shown in **Figure 4.3**, only a fraction of the correction is taken ($0 < \alpha < 1$) to be $\Delta\boldsymbol{q}$ due to the following. Section 4.2 explains that the relationship between the ZMP's Cartesian motion and the joint-space motion depicted by the ZMP Jacobian is highly non-linear. Moreover, this Jacobian is computed numerically, which makes it susceptible by numerical noise. Therefore, if one should adjust the joint angles vector $\boldsymbol{q}_t$, by the full correction as obtained by Eq. (4.9); this might result in an extreme motion correction, which can cause more harm than good. In fact, this means that a gradient correction is used for this method, implying that several iterations are required to achieve the correct adjustments. Thus, the algorithm is executed iteratively until there are no corrections to be performed. **Figure 4.5** describes the general iterative execution scheme of the algorithm.

Given a motion generator **MG** that provides predefined motion trajectories:

i. Perform "learning" run to compute $\mathbf{J}_{ZMP}(\mathbf{q}_a)$ for each $\mathbf{q}_a$ along the motion.

ii. Perform "compensation" runs until ZMP is never outside support-polygon:

- **(a).** Get $\mathbf{q}_t$ from the motion generator.
- **(b).** If ZMP inside the support-polygon then: $\mathbf{q}_d = \mathbf{q}_t$ and go to **(d)**.
- **(c).** Otherwise, compute $\Delta\mathbf{q}$ correction according to Figure 4.3.
  Also, $\mathbf{q}_d = \mathbf{q}_t + \Delta\mathbf{q}$.
- **(d).** Perform motion according to $\mathbf{q}_d$.
- **(e).** Adjust **MG** according to $\mathbf{q}_d \rightarrow$ **MG'**
- **(f).** Re-compute $\mathbf{J}_{ZMP}(\mathbf{q}_a)$ for each $\mathbf{q}_a$ along the motion.

iii. Perform "actual" final run using the updated **MG'** motion generator.

Figure 4.5: The iterative execution scheme of the algorithm.

4.4.3 Convergence to Corrected Motion Trajectories

The fact that the algorithm corrects in a gradient fashion might raise the question whether it will converge to a corrected motion trajectories, which satisfy the dynamic stability criteria of the ZMP being inside the support-polygon all along the motion. Thus, it is necessary to show why this iterative method of correcting only by a fraction is sound.

As will shortly be seen, if α is small enough, it should always reduce the required $\Delta \boldsymbol{p}_{ZMP}$. As explained in section 4.2, the ZMP Jacobian is computed numerically by giving finite tiny displacements to each joint and then computing the corresponding ZMP displacement. Hence, in the local neighbourhood of each Jacobian's calculation it is linear. This is, in fact, the reason why the ZMP Jacobian is a function of the actual joint angles vector $\boldsymbol{q}_a$ as only the local neighbourhood of $\boldsymbol{q}_a$ is this linearity assumed. Furthermore, due to this linearity assumption, moving the robot by $\alpha \cdot \dot{\boldsymbol{q}}$ should effectively reduce the positional error of the ZMP by approximately $\alpha \cdot \Delta \boldsymbol{p}_{ZMP}$. **Table 4.1** shows how eventually, after infinite number of iterations, the corrections will converge to the required correction of the ZMP.

Table 4.1: The convergence of the corrections.

Iteration	Correction	Remainder
1	$\alpha \cdot \Delta \boldsymbol{p}_{ZMP}$	$(1-\alpha) \cdot \Delta \boldsymbol{p}_{ZMP}$
2	$(1-\alpha)^1 \cdot \alpha \cdot \Delta \boldsymbol{p}_{ZMP}$	$(1-\alpha)^2 \cdot \Delta \boldsymbol{p}_{ZMP}$
…	…	…
i	$(1-\alpha)^{i-1} \cdot \alpha \cdot \Delta \boldsymbol{p}_{ZMP}$	$(1-\alpha)^i \cdot \Delta \boldsymbol{p}_{ZMP}$
…	…	…
Total	$= \sum_{i=1}^{\infty} (1-\alpha)^{i-1} \cdot \alpha \cdot \Delta \boldsymbol{p}_{ZMP} \approx (1-(1-\alpha))^{-1} \cdot \alpha \cdot \Delta \boldsymbol{p}_{ZMP} = \Delta \boldsymbol{p}_{ZMP}$	

However, since it is impractical to perform infinite iterations, the algorithm limits the use of the gradient factor α to compute the correction $\Delta \boldsymbol{q}$ in joint space. Whenever the desired ZMP correction's norm $\|\Delta \boldsymbol{p}_{ZMP}\|$ is less than a certain small value ξ the correction performed is $\Delta \boldsymbol{q} = \dot{\boldsymbol{q}}$.

4.5 Summary

This chapter presented the ZMP Jacobian Counterbalance method to adjust the predefined motion trajectories of chosen joints during the humanoid's walk according to the ZMP's horizontal correction and thereby enhance the humanoid's dynamic stability.

This method can be generalised to ample applications not necessarily related to the ZMP correction (Wollherr *et al.*, 2003). In the next chapter, some applications that are related to the ZMP correction, such as adjusting the torso joints' trajectories of the humanoid to enhance its walking dynamic stability will be presented. Another application might be the possibility to adapt predefined gait trajectories suitable for a horizontal plane to new trajectories suitable for walking on slopes.

This method uses a numerical computation of the ZMP Jacobian, thus it entails a great deal of computation and accuracy problems. However, it is independent of the feet-ground contact state which is clearly an advantage.

Chapter 5

Experimental Results

5.1 Introduction

This chapter demonstrates some motion control experiments, which employ the proposed method, explained in the previous chapter. The simulated experiments, all use the environment presented in Chapter 1section 3.2, with relevant minor changes to the world model in each experiment. The appearance, joint configuration and specification of the humanoid model in the simulations are the ones presented in section 3.4.

For each experiment, the initialisation, procedure, objectives and results shall be described, followed by a short analysis and discussion of the relevant points to that experiment. The first experiment utilises the method to adjust predefined motion trajectories of the pitch and roll torso joints of the robot, in order to enhance the dynamic stability of the humanoid's walk. The second experiment modifies the first by replacing the torso roll joint with both hips roll joints. It then compares the results and performance to the previous results obtained in the first experiment. The third experiment takes on a different approach by adapting optimal motion trajectories suitable for horizontal planes to dynamically balanced trajectories suitable for walking on a slope. Lastly, the fourth experiment uses the same setup as the first; however, it adds an external disturbance similar to head wind.

5.2 Experiment 1: Counterbalance using Torso Joints

In this experiment, the proposed method is applied to the humanoid's torso degrees-of-freedom. As described in section 3.4, the humanoid model used has two joints in the torso, a pitch joint and a roll joint. The robot's controller is given non-optimal predefined walking trajectories in which on several time instances of the simulation, the ZMP of the robot is found to be outside the support-region, i.e. as explained in section 2.2.1 it is practically an FZMP. The experiment is conducted on the standard simulation model, in which the humanoid starts from a specified starting point and performs several walking cycles using those motion trajectories, thereby constituting a single run (see **Figure 5.1**). Furthermore, the simulation step was set to 0.032 [msec].

Figure 5.1: The initial configuration of the experiment.

The procedure is as follows. The humanoid performs an initial *learning* run in which it computes the preliminary mapping between the posture vectors $\boldsymbol{q}_a$ and the Jacobians, required for the later counterbalancing. This phase is followed by an iterative scheme of runs, where the controller performs a *compensation*

(counterbalance) run followed by an immediate *Jacobian update* run. In each compensation run, whenever the controller detects that the ZMP is outside the support-polygon, it computes the current corrections for the two torso joints according to the algorithm described in section 4.4.2. These corrections are stored internally to adjust the motion trajectories in the successive compensation runs (see section B.2.3 in Appendix A for an explanation how this is done). Each Jacobian update run is used just for continuous adaptation of the mapping $\boldsymbol{J}_{ZMP}(\boldsymbol{q}_a)$ since it might change after each compensation run. Once, the humanoid achieves sufficiently corrected motion trajectories, it stops and saves the last corrections to those joints to a file so that the user can perform one final *actual* run which will employ the corrections obtained and prove the proficiency of the method.

The objective of this experiment was primarily to verify the feasibility of the proposed method. Thus, it aims in adjusting only two joints, which are considered *redundant* in terms of the motion. Also, these torso joints were chosen in favour of other joints since they clearly will have the largest affect on the robot's posture (these joints manipulate the heaviest particles of the robot).

Figure 5.2 compares the torso joints trajectories before and after applying the method. The oscillations in the torso joint angles in certain time instances are a result of the ZMP being outside the support-polygon. In addition, the gradual adjustments performed on the torso joints during the iterative compensation run are shown in **Figure 5.3**. One can observe, that those adjustments become larger with the number of iterations since they are accumulated during the iterations (this is especially apparent in the torso pitch joint). However, at a certain iteration the adjustment remains the same, since at that iteration the accumulated adjustment has become sufficient to maintain the ZMP inside the support-polygon. Clearly, due to these adjustments of the torso, the CoM and ZMP trajectories are changed. **Figure 5.4** illustrates a comparison of the ZMP and CoM trajectories between the *regular* run (without employing the method) and the *actual* run (the final run which is the result of the method). One can observe that the ZMP appeared to be outside the support-

polygon about 15 times (see in the ZMP plots of **Figure 5.4**), yet after only a few compensation iterations, this happened only 3-4 times – a substantial improvement in dynamic stability. **Figure 5.5** shows snapshots of the simulated motion.

From the results mentioned above, one can verify that the proposed method performs its aim, and improves the dynamic stability of the humanoid during its walk. However, not all dynamic instability occurrences are fixed, in a feasible number of iterations (and maybe never) and the reasons shall be explained in the next chapter.

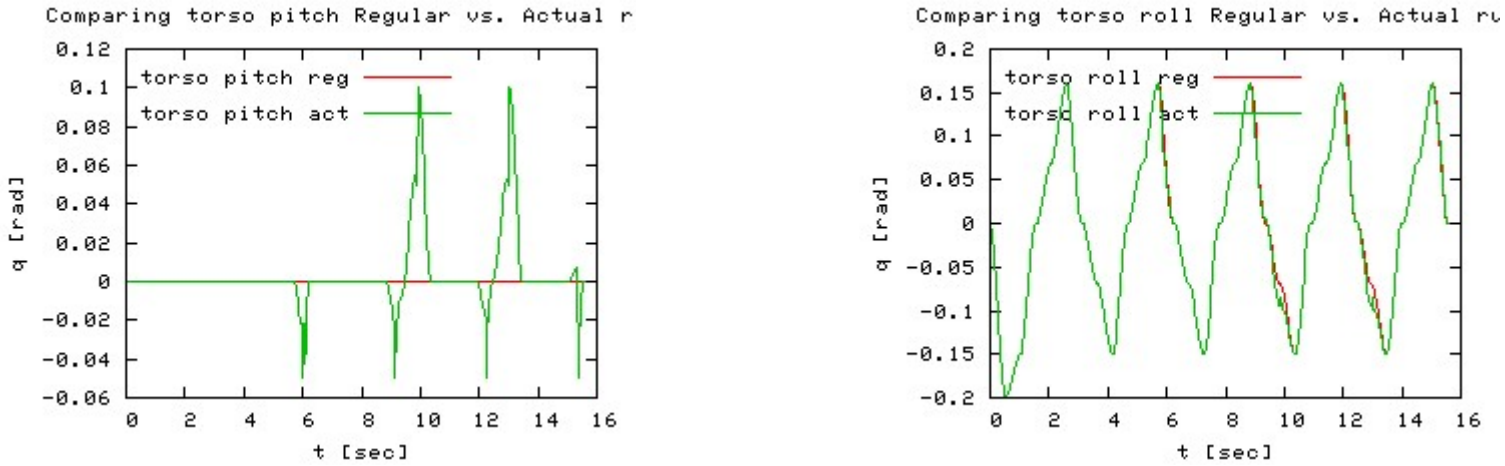


Figure 5.2: The torso joints' trajectories in the regular run compared with the actual final run.

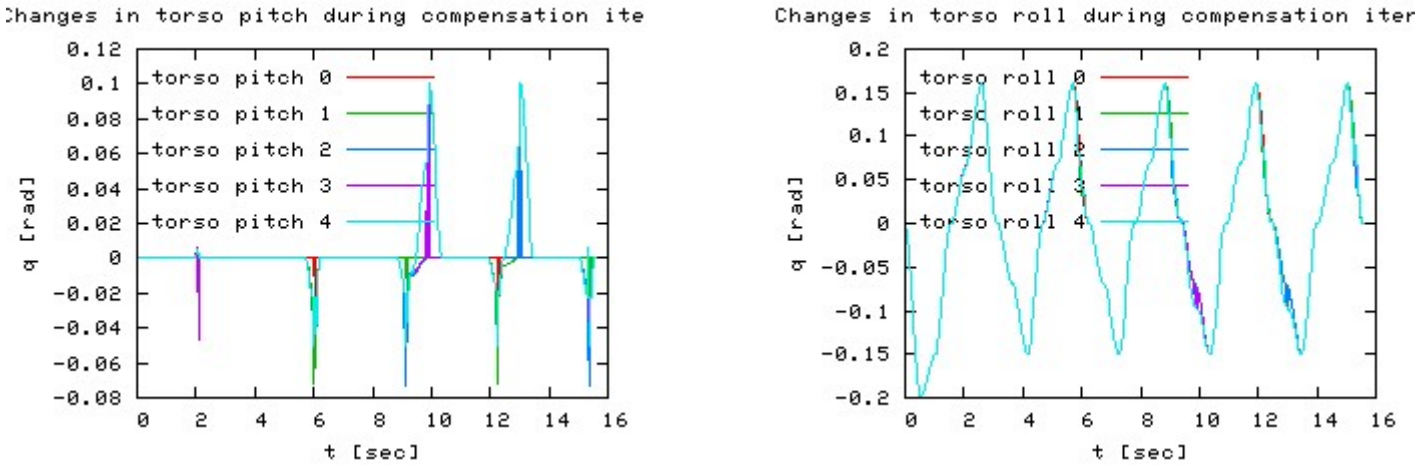


Figure 5.3: The torso joints' trajectories adjustments during the iterative compensation runs.

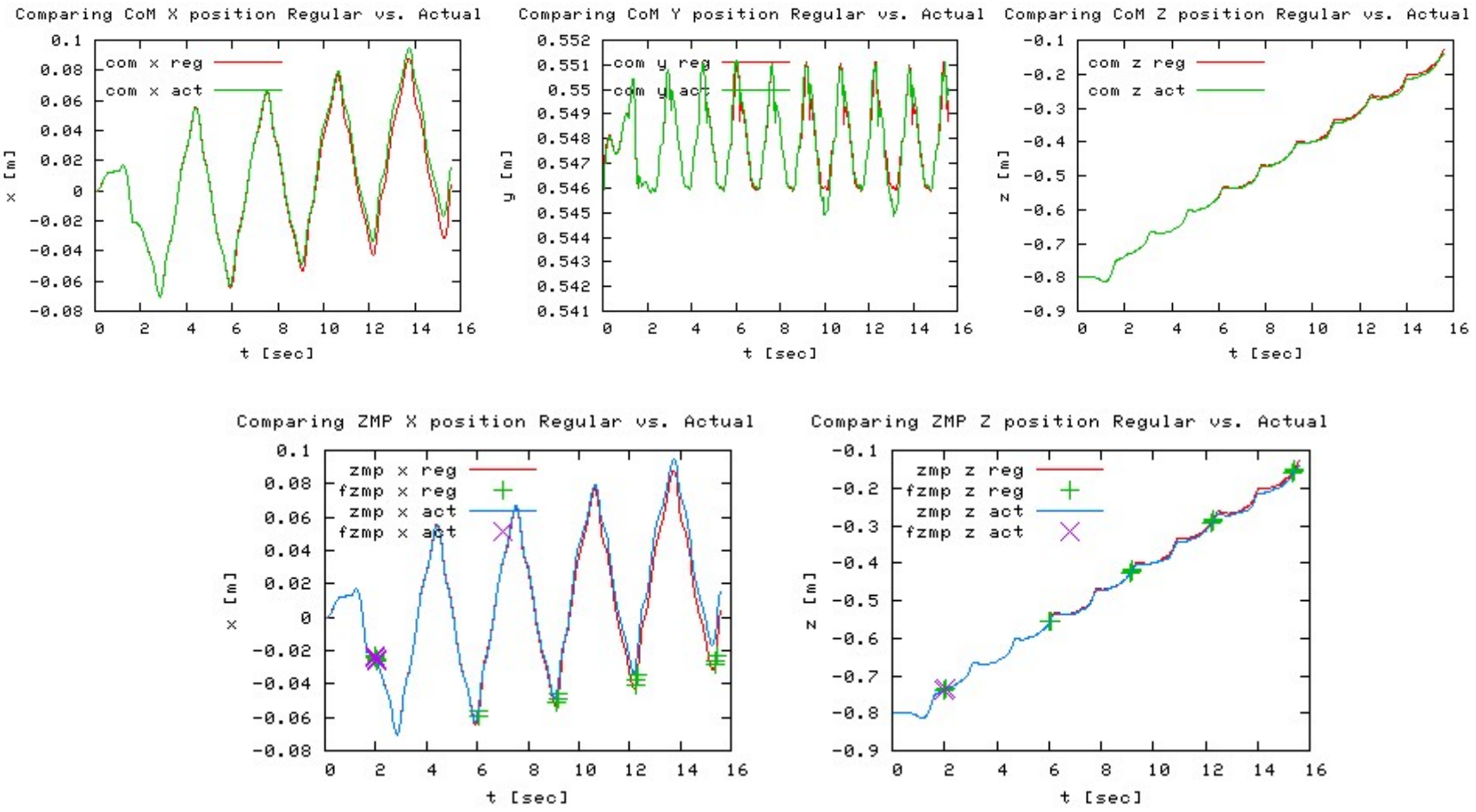


Figure 5.4: CoM and ZMP trajectories in the regular run compared with the actual final run. In the ZMP plots, the occurrences of FZMP are also illustrated.

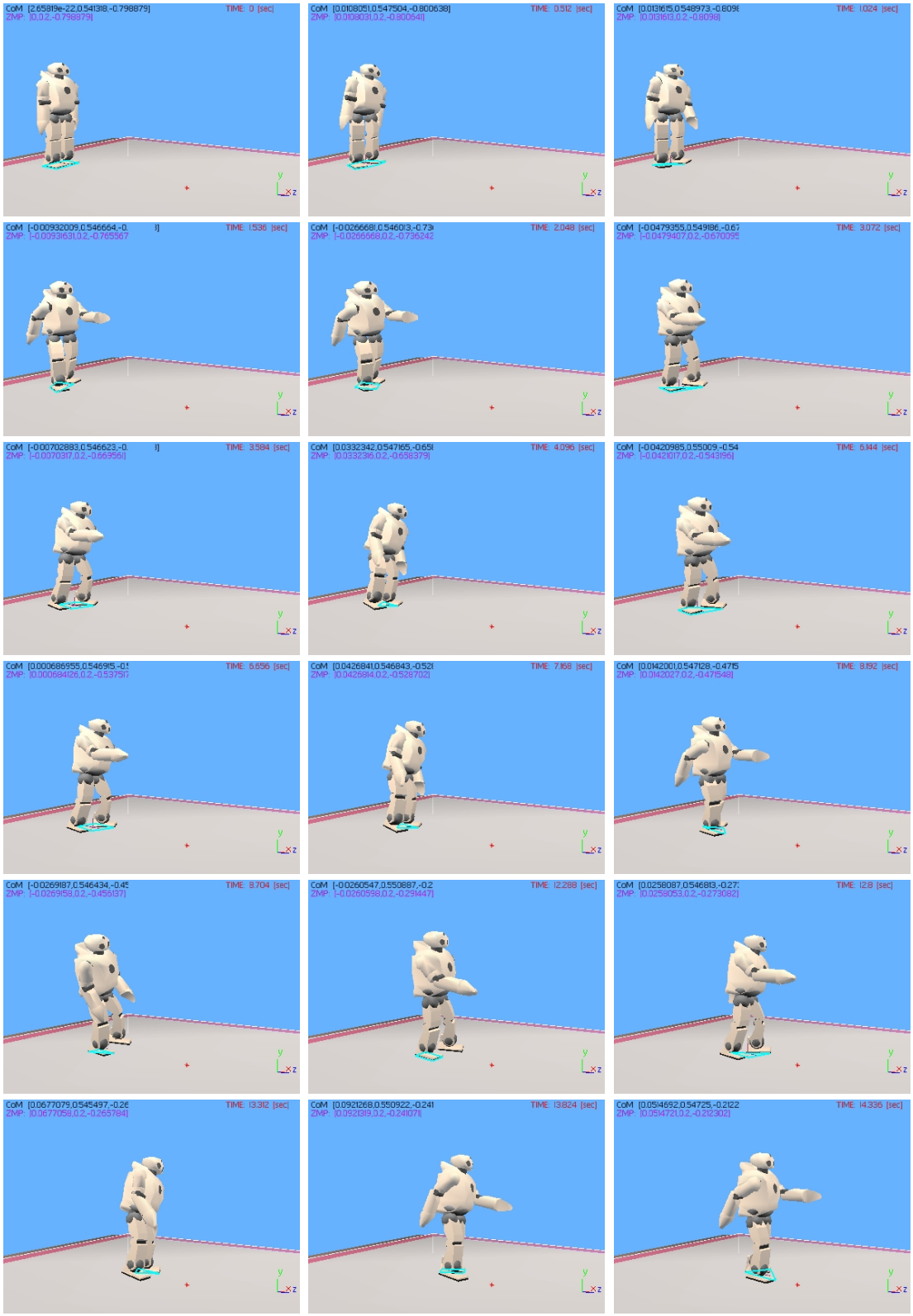


Figure 5.5: Snapshots of the walking experiment in which compensation is done only on torso joints.

5.3 Experiment 2: Counterbalance using Torso and Hips Joints

In this section, the method is tested with the hip roll instead of the torso roll in the previous experiment. As previously, the robot's controller is given non-optimal predefined walking trajectories in which on several time instances of the simulation, the ZMP of the robot is found to be outside the support-region. The experiment is carried out with the same configuration as before along with the same procedure except that this time the controller computes the corrections for the torso pitch joint and the two hip roll joints (left and right).

The objective here was to compare performance of the method applied on the torso and hip roll joints with the previous experiment. The use of the hips roll joints, which do not originally participate in the predefined motion trajectories, defines a proper counterbalance with what is also known in the literature as *redundant* degrees-of-freedom, i.e. joints that are inactive in the motion but are corrected to counterbalance the affects of the active joints.

As shown in **Figure 5.6**, the joints trajectories after applying the method contain oscillations in several time instances, which correspond to the FZMP occurrences. On the other hand, **Figure 5.7** shows the accumulation of corrections for the joints concerned during the iterative compensation run. As previously, due to these adjustments, the CoM and ZMP trajectories are modified. **Figure 5.8** compares the ZMP and CoM trajectories between the *regular* run and the *actual* run. In this experiment, the ZMP appeared to be outside the support-polygon around 10 times, and after a few compensations, this was reduced to five times. The simulated motion's snapshots were very similar to the previous experiment, thus, they are not shown.

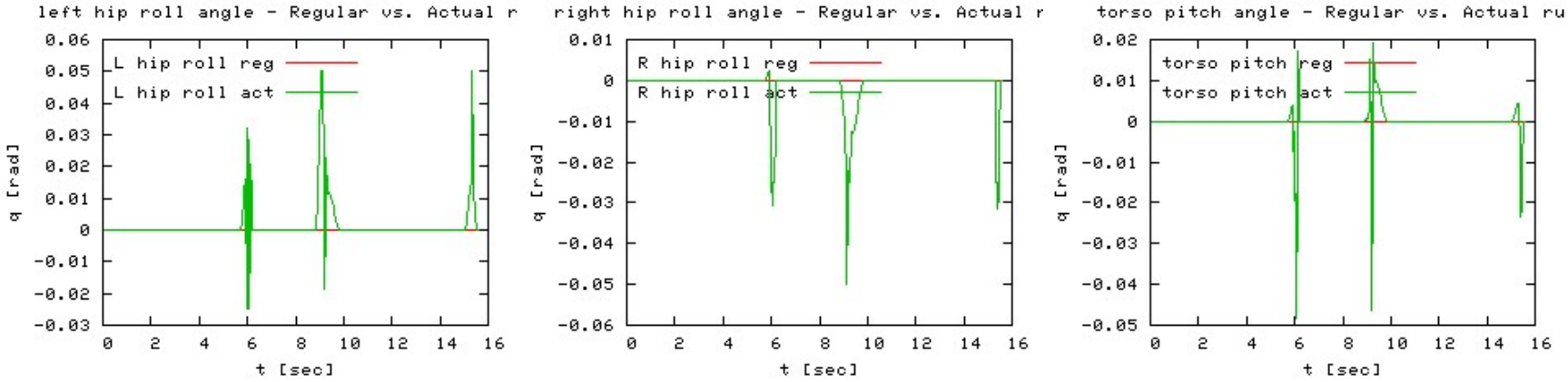


Figure 5.6: The torso pitch and hip roll joints' trajectories in the regular run compared with the actual final run.

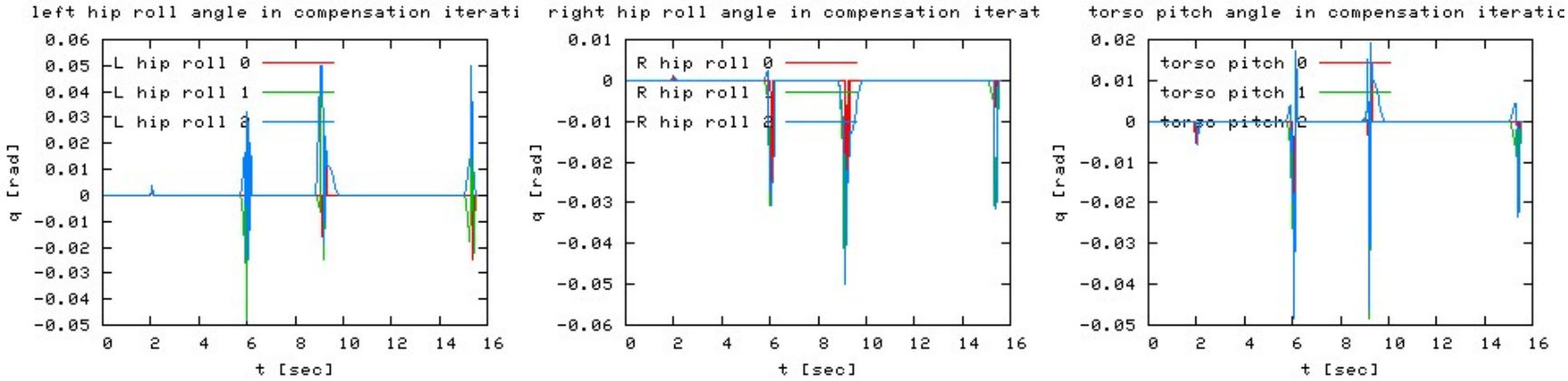


Figure 5.7: The torso pitch and hip roll joints' trajectories adjustments during the iterative compensation runs.

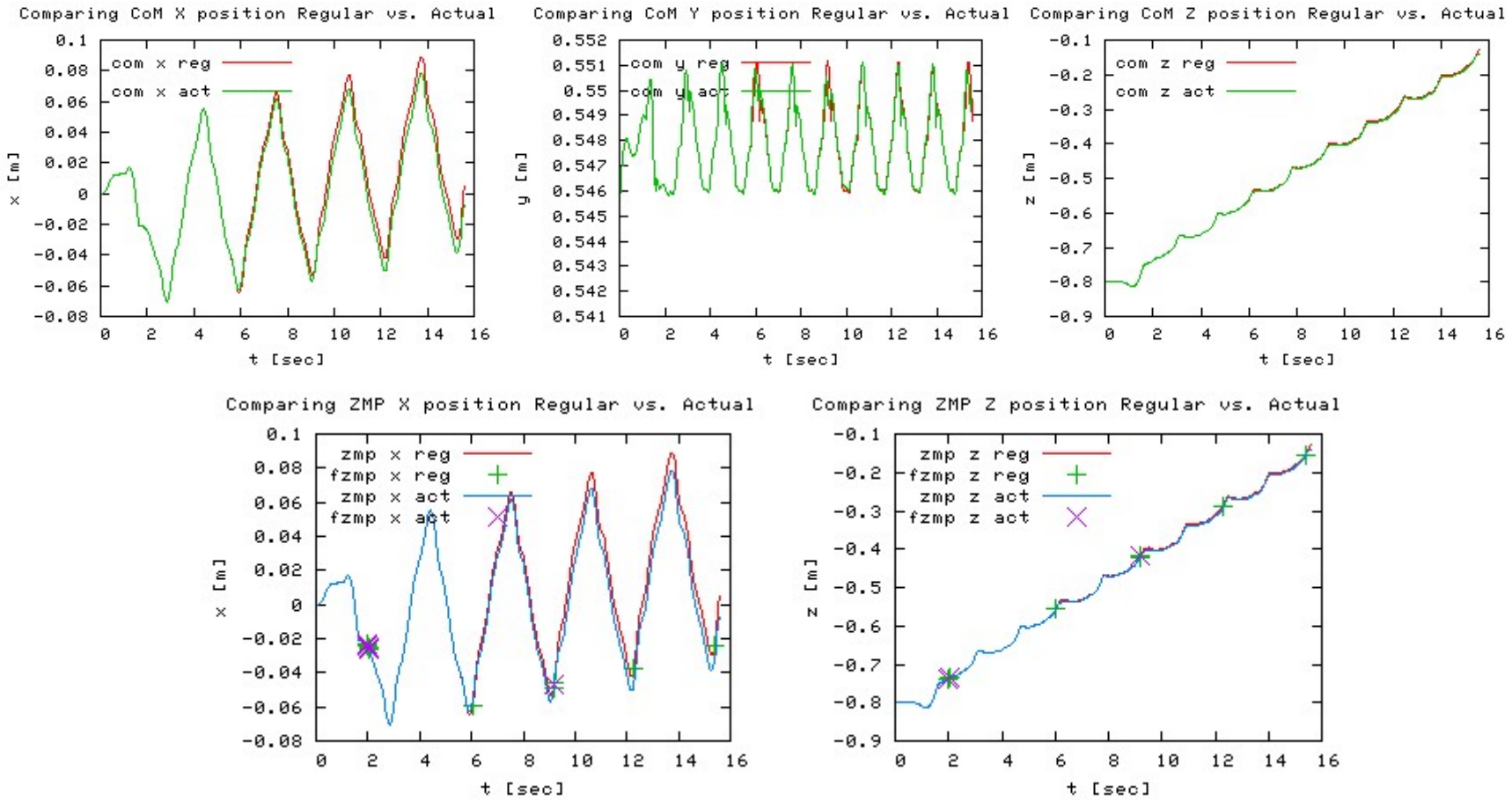


Figure 5.8: CoM and ZMP trajectories in the regular run compared with the actual final run. In the ZMP plots, the occurrences of FZMP are also illustrated.

The adjusted joints, which perform the counterbalance in this experiment, are all inactive in the original motion, which means that they are decoupled from the joints effectively performing the motion itself. This is an advantage, since the user can easily separate the corrected joints from the motion and obtain two versions of the motion – the original version and the corrected version. Moreover, one can observe that similar performance is achieved.

5.4 Experiment 3: Adjusting Trajectories for Slope Walk

This experiment verifies the applicability of the proposed method to adapt near optimal motion trajectories suitable for horizontal planes to dynamically balanced motion while walking on a slope. The configuration used in section 5.2's experiment was employed but with a sloped stage for the humanoid to walk on. This stage is illustrated in **Figure 5.9** with a slope of approximately 4º (0.063 [rad]). All the other parameters remained the same, namely the simulation step, the predefined motion trajectories and the corrected joints (torso joints). Moreover, the same procedure was followed, i.e. *regular*, *learning*, *compensation*, and *actual* runs.

Figure 5.9: The slight downhill stage configuration. The slope is approximately 4º.

As mentioned above, this experiment's objective was to test the feasibility of the proposed method to adapt optimal motion trajectories appropriate for horizontal planes to dynamically balanced trajectories for slopes. For simplicity, a downhill slope was chosen to avoid feet-ground impedance problems, which are unhandled by this method. Walking on slopes has been extensively studied in many biped and humanoid walking research (Fujimoto *et al.*, 1998; Kajita *et al.*, 2002; Zheng & Shen, 1990), thus it was appealing to test the proposed method with such a configuration.

Figure 5.11, **Figure 5.12**, and **Figure 5.13** show the typical results shown in previous sections. However, there seems to be much less improvement as the FZMP occurrences in the regular run were about 52 times, while after 3 compensation runs it was reduced to 33 times for the actual run. Moreover, if we continued with more compensation runs the robot becomes unstable and will eventually fall. This is because the FZMP occurrences are appear in consecutive time steps, which causes very jaggy oscillations to the torso joints, as seen in **Figure 5.12**. In addition, in certain phases during the walk, the corrections cause the support-polygon to shrink as a side effect. This, in actual fact, reduces the potential of dynamic balance (see **Figure 5.10**).

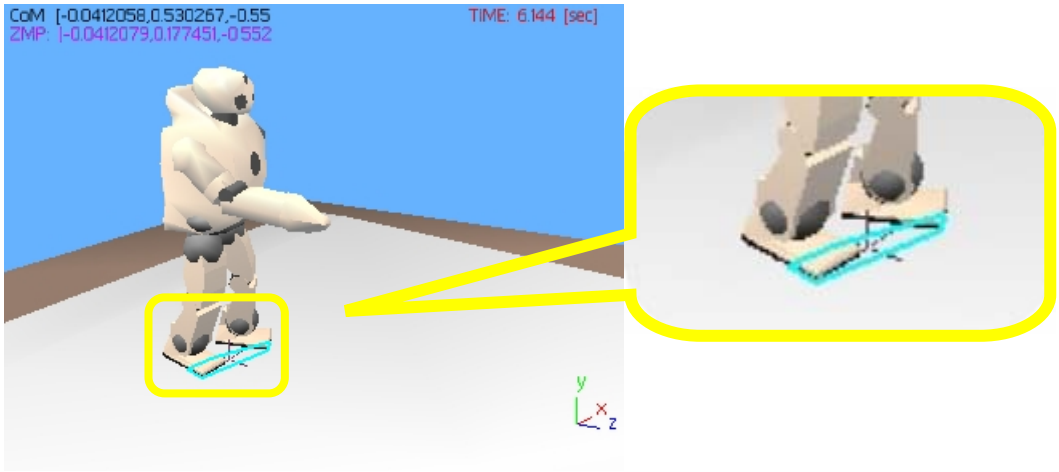


Figure 5.10: The support-polygon shrinks due to the contact dynamics affected by the slope and the torso motion corrections.

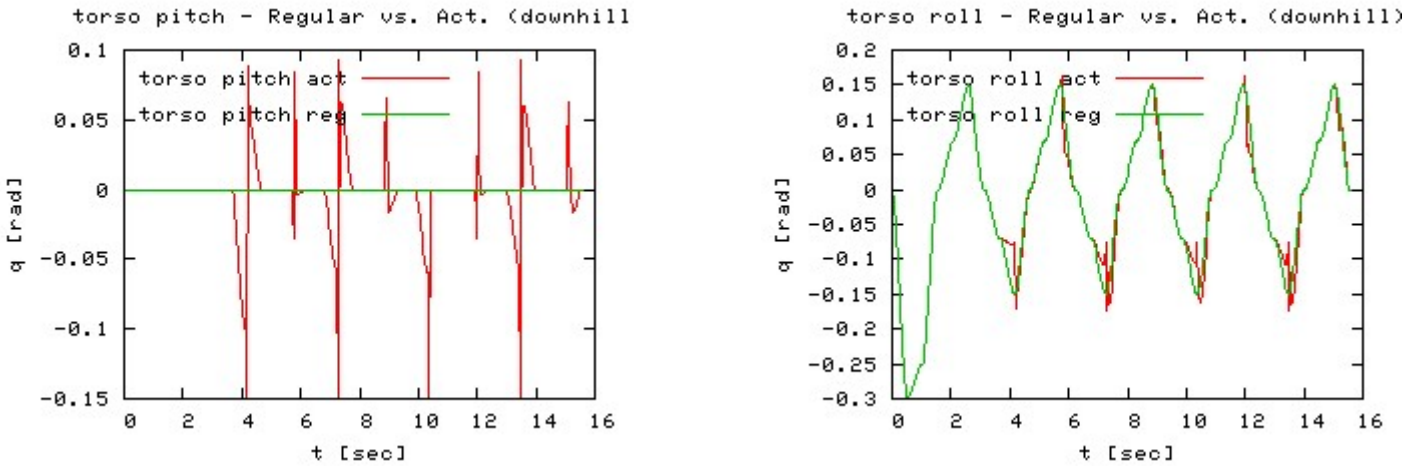


Figure 5.11: The torso joints' trajectories in the regular and actual runs for the downhill slope experiment.

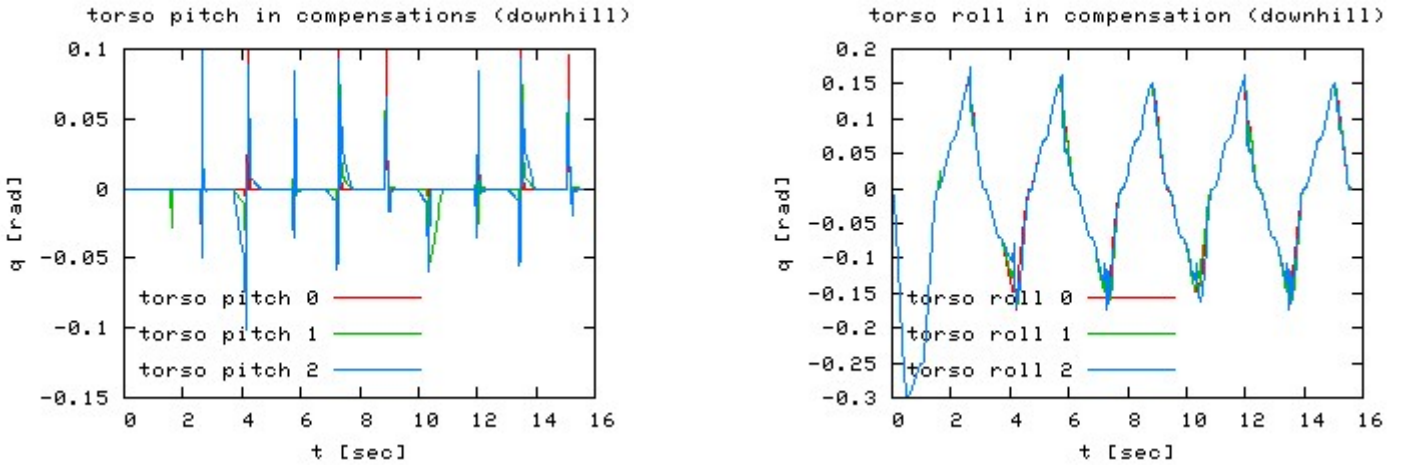


Figure 5.12: The torso joints' trajectories adjustment during the iterative compensation runs in the downhill slope experiment.

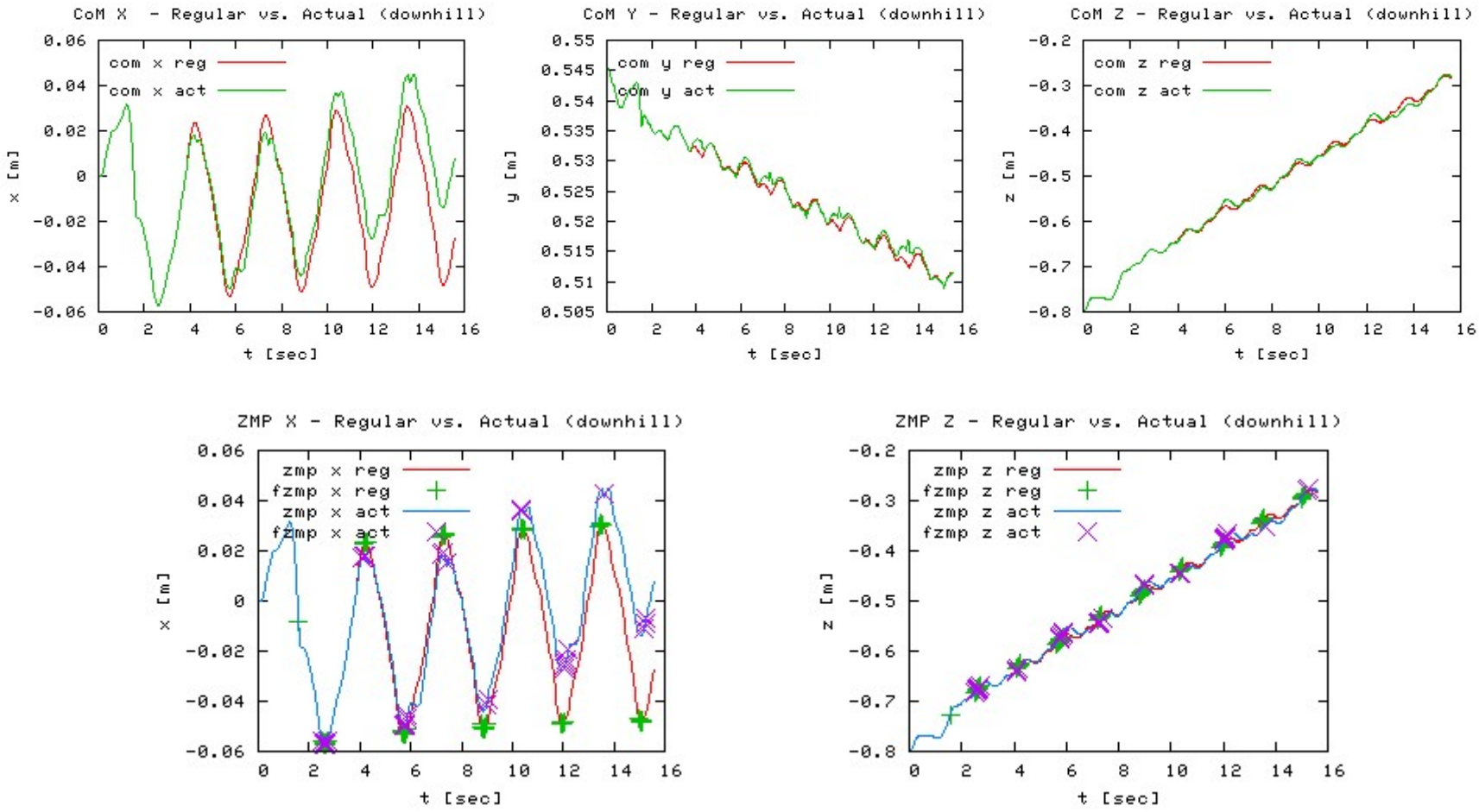


Figure 5.13: CoM and ZMP trajectories in the regular and actual runs. The ZMP plots also illustrate the FZMP occurrences.

As it seems, the proposed method is not robust enough with the tested configuration and its performance is very limited despite the reduction in FZMP occurrences. One improvement can be tried by augmenting the adjusted joints with the left and right ankle joints. Hopefully, this could lead the ankles to adjust themselves to the slope. Another suggestion is to use a different interpolation method for the trajectories' adjustments, which will smooth better the jaggy-like joint angle profiles and reduce the instability affect. As shown in chapter 4, the Bezier interpolation could be a potential candidate. Some of the limitations and improvement suggestions like the ones here will be discussed in more detail in the following chapters.

5.5 Experiment 4: Counterbalance under External Disturbance

This final experiment takes a slightly different approach and examines the method's performance under external disturbance. The same configuration from the first experiment is used but augmented with an external head wind force acting on the robot of -0.2 [N]. All the other parameters remained the same including the procedure.

The objective was to prove the robustness of the method, which will now try to counterbalance the applied external force, which effectively causes dynamic instability occurrences, even though the predefined motion trajectories are quite optimal in terms of dynamic balance. As mentioned earlier, this force is acting opposite to the walking direction, thus it pushes the ZMP position towards that direction (it is clear to see why by referring to the ZMP equations in section 2.5). Therefore, it was expected that the adjusted joints (the torso joints) will be corrected opposed to the external disturbance.

Figure 5.14, **Figure 5.15**, and **Figure 5.16** show the standard results as shown in previous experiments, while **Figure 5.17** presents the simulated motion's snapshots

with the yellow arrow representing the external force. Two interesting observations arise from these results. Firstly, the corrections computed for the torso pitch joint as shown in **Figure 5.14** and **Figure 5.15** are sensible, they cause the humanoid to bend its back forward to counterbalance the ZMP pushed back by the external force. Secondly, the FZMP occurrences are greatly reduced from 27 times to merely 4-5. This major reduction in the FZMP occurrences, to be precise, the improvement to the dynamic stability of the humanoid's walk is very impressive and shows the potential of the proposed method. However, note that the external force applied is constant and in case it was sudden or dynamically changing, this method would not have performed successfully due to its iterative scheme definition. Other directions of external constant force were experimented and exhibited similar success, however, only one example is shown here due to lack of space.

The observations mentioned above will be elaborated in the next chapter, along with other known problems and their implications.

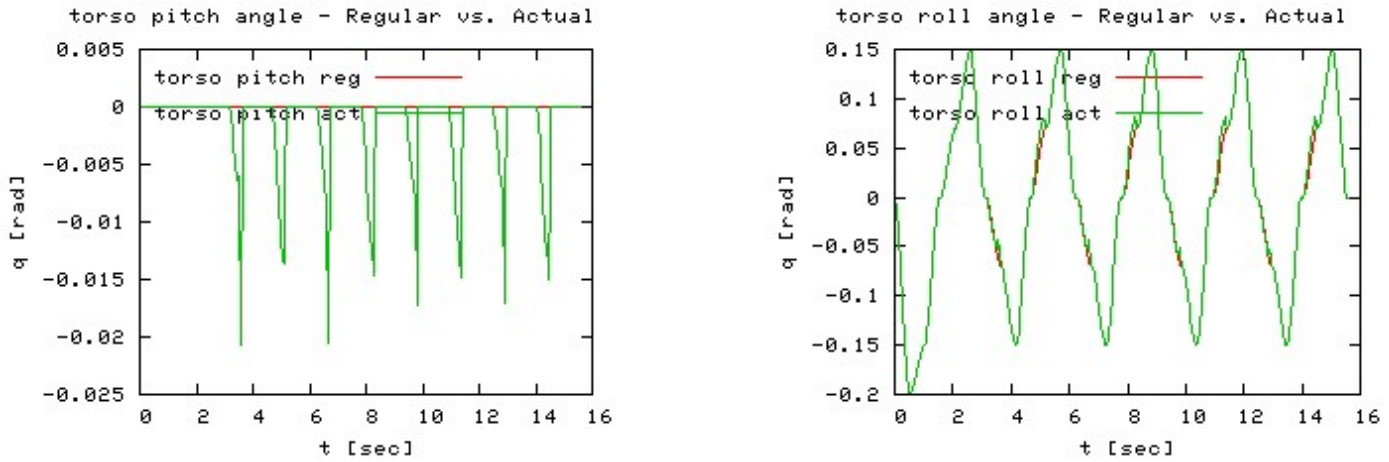


Figure 5.14: Torso joints' trajectories in the regular run compared with the actual final run. The pitch joint tries to counterbalance the external force.

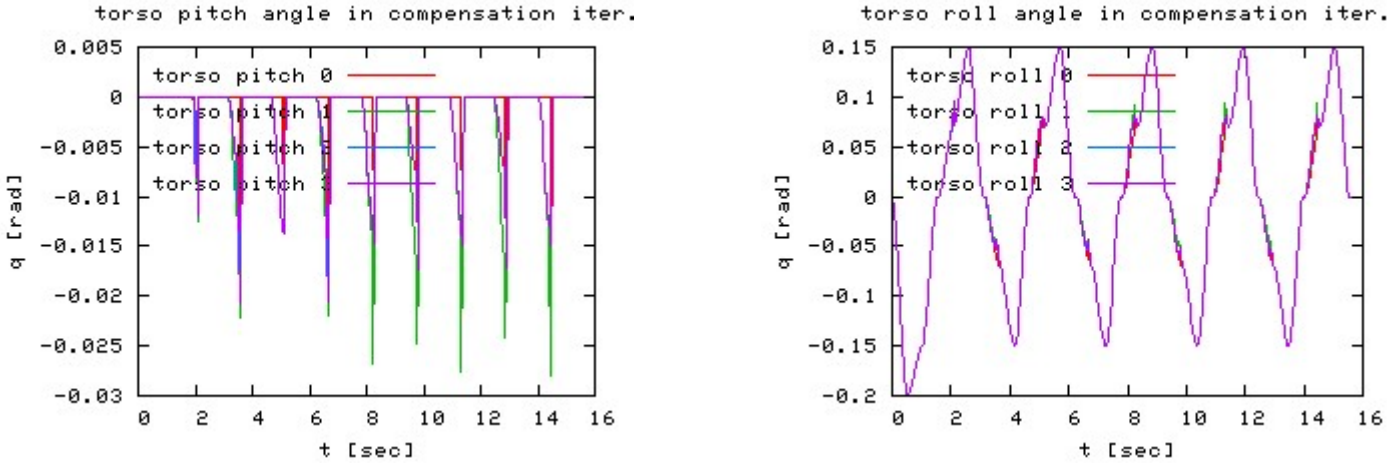


Figure 5.15: CoM and ZMP trajectories in the regular and actual runs. The ZMP plots also illustrate the FZMP occurrences.

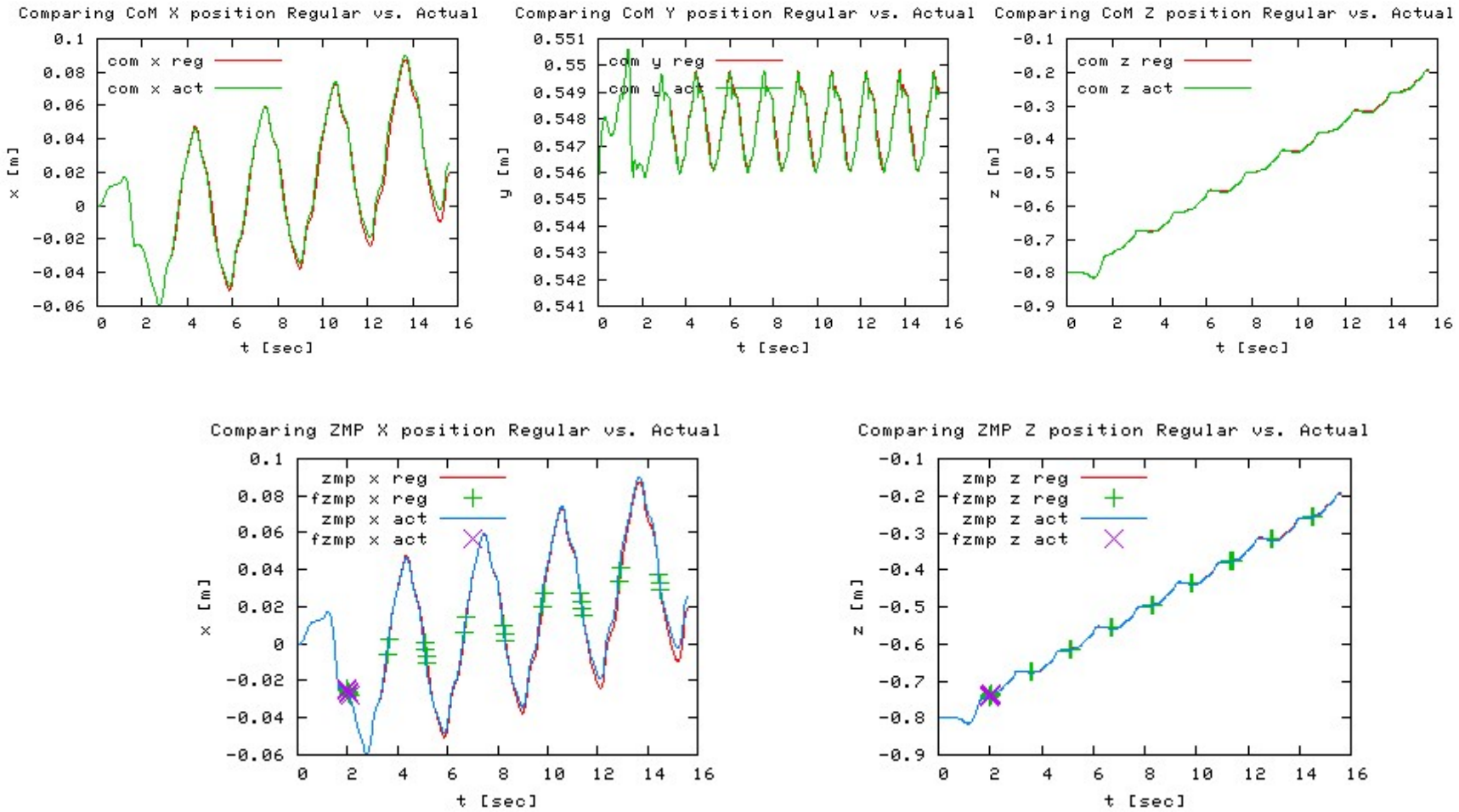


Figure 5.16: CoM and ZMP trajectories in regular and actual runs. The ZMP plots illustrate the substantial reduction in FZMP occurrences.

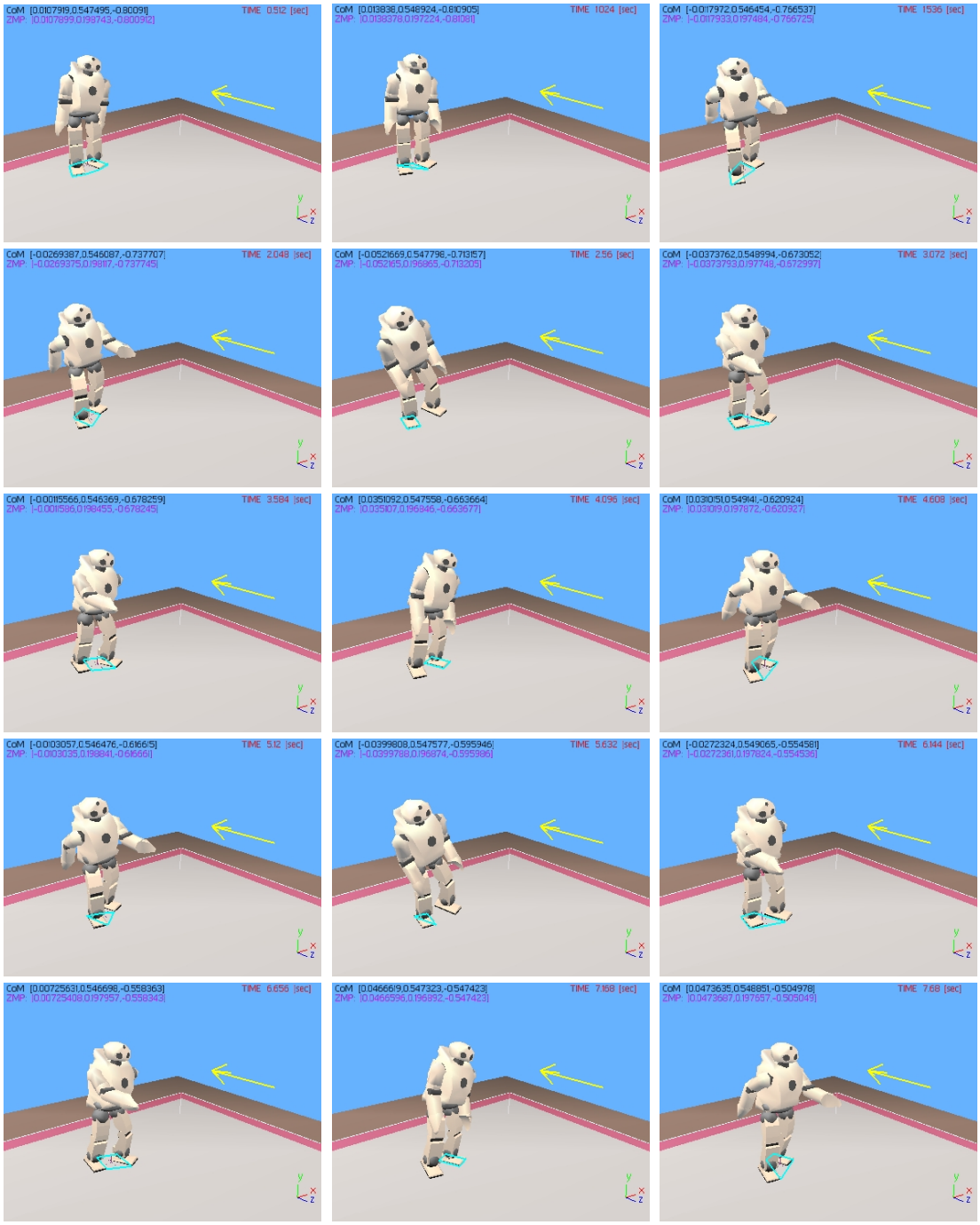


Figure 5.17: Snapshots of the external disturbance experiment. Compensation is done only using torso joints. The yellow arrow represents the external force acting on the robot (-0.2 [N]).

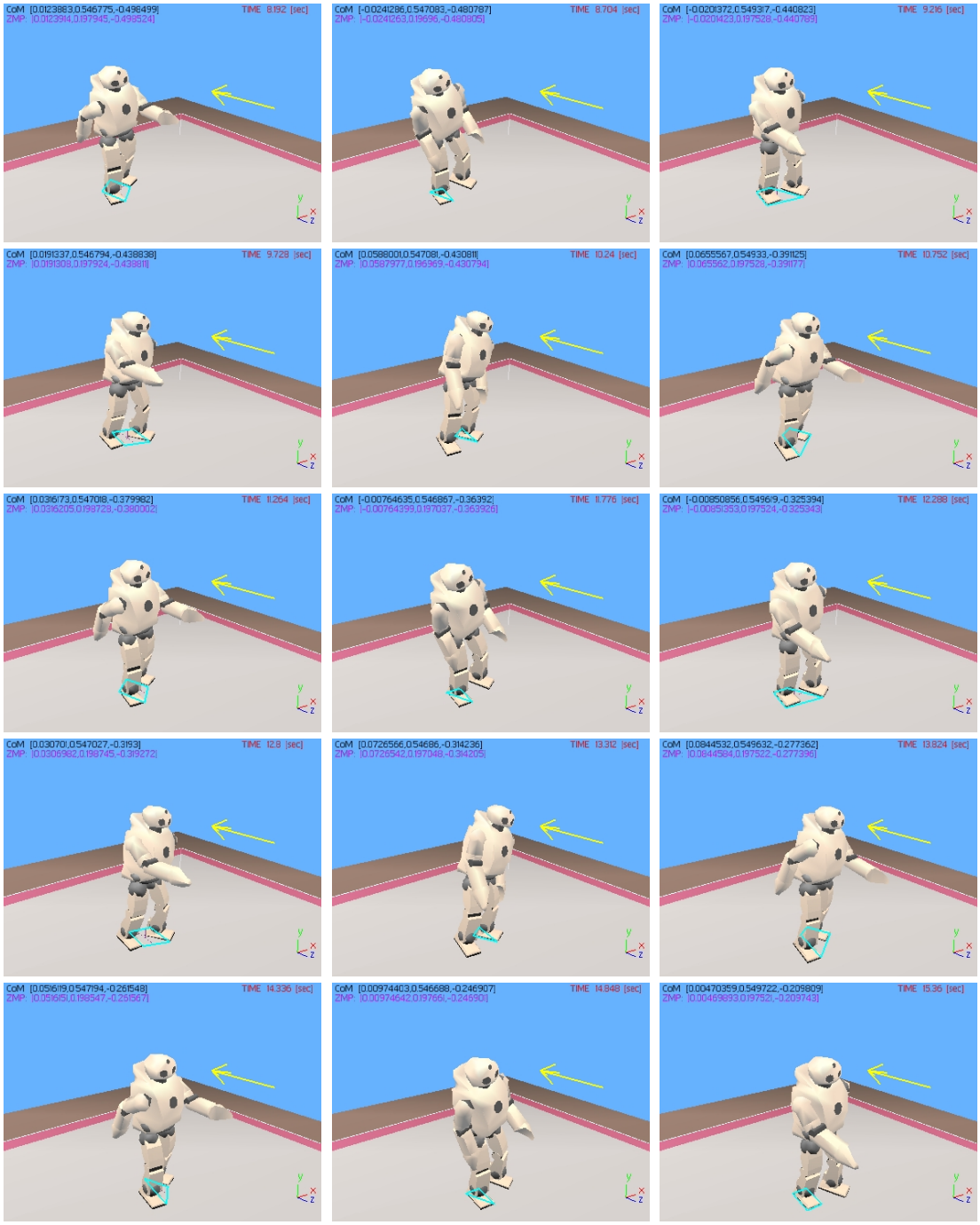


Figure 5.17: Snapshots of the external disturbance experiment. Compensation is done only using torso joints. The yellow arrow represents the external force acting on the robot (-0.2 [N]). (Continued).

Chapter 6

Discussion

6.1 Introduction

This chapter discusses several issues which arose from the development and experimental results of the research presented in this dissertation. The following section considers the various contributions and significance of the work presented in this dissertation to the humanoid locomotion research. Section 6.3 presents the main limitations and known problems of the method and implementation details described in chapters 3, 4 and 5. The last section briefly concludes the personal knowledge gained during the scope of this study.

6.2 Contribution and Significance

The method proposed and developed in this dissertation is significant from the following three perspectives:

1. The method does not switch its control manner or modify its procedure depending on the contact state changes. In other words, the method is versatile and independent of the feet-ground contact states, and behaves according to the same principle in both double-support and single-support phases of the bipedal walking.
2. The method indicates and assures the ZMP, inherently a kinematic parameter represented in Cartesian space, can characterise the humanoid system's global dynamics and can be utilised for the controller design of legged robots.

3. This dissertation has meticulously described the development of a simulated humanoid locomotion controller, while offering a large engineering contribution and paving the way in its combination of sophisticated humanoid simulation with ZMP based locomotion.

The extensive work presented in this dissertation has provided several contributions from various engineering aspects. From the theory point of view, it has presented a method, which integrates ZMP based locomotion control with a complex humanoid robot model with a large number of degrees-of-freedom. Moreover, it has initiated a new research approach, which concentrates on the ZMP in the development of humanoid locomotion, which will hopefully be pursued by future researchers. From the simulation environment perspective, it has designed a comprehensive framework for simulated humanoid control by identifying the core functionalities of such systems and composing them into a generic software design. In addition, by widely exploiting the main capabilities of a mobile robots simulation tool, this study has contributed to the improvement of it with bug discoveries, fix suggestions, and offering sample implementations for future users. Additional implementation achievements such as the Moore-Penrose pseudo-inverse implementation (Appendix C) and others have been proposed for open source software libraries.

6.3 Known Problems and Limitations

Naturally, the end product of the research presented in this dissertation bears some limitations and comprises a few known problems. These problems and limitations are categorised to limitations of the method described in chapter 4 and its algorithm, and problems stemming from implementation and experimental procedure presented in chapters 3 and 5.

The main limitation of the proposed method is the fact that it heavily relies on feasibly stable predefined motion trajectories as its input. It is clear, that this method cannot generate dynamically balanced walking trajectories from an input motion,

which causes the humanoid not to walk at all. The predefined motion trajectories have to possess static stability at least, which the method only promises to improve, in terms of dynamic stability. Moreover, as seen in previous chapter's experiments, in some situations, the method cannot adjust the predefined trajectories to become perfectly dynamic balanced. This issue also has some experimental and implementation aspects. The experimental procedure, in its current implementation, cannot deal with cases in which the humanoid tips over, since it is unable to detect such crises and restart the compensation iterations to correct the motion.

Another chief intricacy innate in the method is that it computes the ZMP Jacobians in a numerical fashion by giving finite tiny displacements to each of the adjusted joints and computing the corresponding ZMP displacement. This entails accuracy problems and large amount of computation. Evidently, since each corrected joint contributes one column to the ZMP Jacobian as formulated in section 4.2, the more joints applied in the method the ZMP Jacobians' become bigger and thus require more time to be computed.

Furthermore, the method does not give the user any hints concerning which joints to choose for adjustments. The user has to make an intelligent guess regarding the joints, which would be optimal for a specific motion trajectories and environment. This problem was illustrated in some of the experiments in the previous chapter, especially the slope walking experiment.

From the experimental design and implementation point of view, there are several other limitations. The way the ZMP Jacobian is retrieved for a given posture vector, in order to compute the desired joint corrections, is implemented as a lookup table, which effectively implements a discrete non-linear interpolation in the form of *Nearest Neighbour* method (as explained in section 4.4.1). This is highly approximated approach might bear further inaccuracies, which could cause unnecessary perturbations, resulting in the humanoid tipping over. Other options to accomplish this task will be discussed in the next chapter.

Moreover, the inverse computation of the Jacobians might suffer from singularity issues. In some cases, a required correction to the ZMP offsets from the support-polygon might numerically occur around a singular posture. This can yield, once again, an incorrect and extreme joint correction through the pseudo-inverse result of the Jacobians (see Appendix C). This can be improved by an augmented inverse matrix method which is singularity robust (Sugihara, 2004) or by simply discarding such suggested corrections when the Jacobians are not fully-ranked.

Another primary limitation from the environment and implementation aspect is that it is immensely restricted to simulation and cannot be exported to a real humanoid. The environment used in this study, as described in chapter 3, required significant engineering to compose the physics module, robot's controller and the supervisor due to Webots limited sensory devices. Ideally, the humanoid should have relied only on its own sensors to acquire the dynamics information for the ZMP computation and not use the direct information inherently available from the simulation software (through ODE and the physics module). If that was the case, then the work accomplished here could be employed on a physical humanoid prototype.

Lastly, the results presented in the previous chapter suggest the feasibility of the proposed method. However, all the experiments were conducted on the same humanoid model with more or less the same corrected joints. Further experiments are required to prove the admissibility and generality of the method, such as applying it on a different humanoid like the HOAP-2 model presented in Appendix A.

6.4 Knowledge Gained in the Scope of this Thesis

Although the description of this research and its goal focus on humanoid locomotion control, it was astonishing to discover the breadth of knowledge and skills that were required for the work presented in this dissertation. Consequently, the author had to improve and/or acquire expertise in a variety of domains.

First, the theory of biped and humanoid locomotion had to be investigated in depth. This investigation started by a wide range of literature overview, which kept accumulating as the research progressed. As presented in chapter 1, this exploration went from general approaches to biped locomotion to ZMP focused studies. Besides furbishing of literature review skills and writing a large-scale dissertation, the author had to understand diverse locomotion control schemes, methods and algorithms and to become proficient in the ZMP related issues.

On a different level, this study involved a great deal of comprehension in mathematics, dynamics, and kinematics. The ZMP derivation itself, the core of this research, is quite complicated in terms of moments, accelerations, angular velocities and inertia tensors. The mathematics knowledge included several subjects such as matrix/vector algebra (e.g. Jacobians, Moore-Penrose pseudo-inverse), homogeneous coordinates and transformations, interpolation methods, and computational geometry (e.g. convex-hulls, squared distances).

From software design and implementation viewpoint, the work presented here blended many fields of computer science and software engineering: XML parsing, Shared Memory and Semaphores, Computer Graphics, Robotic Simulation etc. Moreover, this project exploited a large set of open source libraries, which included Arabica[11] XML parser toolkit, CGAL[12] (Computational Geometry Algorithms Library), ODE, OpenGL[13], NEWMAT and others. The main implication of this huge blend of software is that it required significant interfacing and setup. In addition, taken as a whole, the code developed during this work had very much the characteristics of a large-scale software project, demanding time-consuming coding, careful revision, and maintenance.

[11] See http://www.jezuk.co.uk/cgi-bin/view/arabica.
[12] See http://www.cgal.org.
[13] See http://www.opengl.org.

Aside from the aspects mentioned above, the author also acquired valuable time management and collaboration skills, as well as experience in high-level academic research.

Chapter 7

Conclusion

This dissertation has presented the development of simulated humanoid locomotion control. This chapter is dedicated to conclude the study presented here. It shall commence with a brief summary of the dissertation in the following section. Next, in section 7.2, it will discuss possible future developments to enhance the described methods. Lastly, it presents the overall conclusions of this research in section 7.3.

7.1 Summary

This section briefly summarises this dissertation by presenting the main ideas and concepts of each chapter as follows.

Chapter 1 explained the background and motivation for this research. It then methodically classified and reviewed the past works on biped and humanoid locomotion. Furthermore, it described the intrinsic difficulties and principles of this locomotion along with presenting achievements of other sophisticated controllers on in terms of mobility robustness.

Chapter 2 systematically explained every aspect of the ZMP, from its definition, to its derivation. It also took a deeper look into previous studies ZMP related biped and humanoid locomotion.

In chapter 3, the simulation environment used was introduced and explained. From a brief debate on using simulation for biped research, it continued to specify some issues relating to simulated humanoid control. It also presented several core

implementations, which were necessary to perform the locomotion control, and the humanoid prototype used.

In chapter 4, a motion controlling method based on the ZMP Jacobian concept was presented. This method tries to counterbalance dynamic instability of predefined motion trajectories by correcting specified joints to result in enhanced dynamically balanced locomotion. In addition, the complexities of gait motion trajectories design were explained.

Chapter 5 presented some experiments, which applied the proposed method from chapter 4 and their results. The results of these experiments substantiated the applicability and performance of the ZMP Jacobian Counterbalance method. Especially impressive was the successful experiment in which counterbalance under external disturbance was carried out.

In chapter 6, the known problems and limitations of the presented work were elucidated. Moreover, the significance and contributions of this dissertation were discussed, together with the knowledge and understanding gained by the author in the process of this research.

Next, the ideas for future development of this work will be suggested, followed by the conclusion of this dissertation.

7.2 Future Development

The research presented in this dissertation only touched the development of ZMP based humanoid locomotion control, and due to the short time available for it, is far from being conclusive. Therefore, it could be improved from three different aspects: the method relating the ZMP to the dynamic balance control, the implementation specifics, and experiments or applicability of the proposed method.

From the method's perspective, several ideas can be further developed. Firstly, one could try to develop the ZMP Jacobian analytically by linking it to the multi-bodied structure of a generalised humanoid using a better and more relevant model, or existing ones such as the Mass-Concentrated Model or the Inverted Pendulum variants. Computing the ZMP Jacobian analytically would improve the accuracy and computational cost of the controller as opposed to the current numerical calculation. In addition, the proposed method is certainly lacking any handling of impedance and ground reaction forces. Augmenting the method with reaction force manipulation strategies would enhance the performance of the method and enable it to avoid crisis situations of the humanoid tipping over.

Other ideas, linked to the current implementation, can extend the work done here. For example, the motion planning could be upgraded by applying inverse kinematics and dynamics (see section 4.3 and Appendix A). This would allow easier creation of diverse walking motions, which would be much more robust and balanced. Effectively, this would enable wider testing of the proposed method. Another prime direction for future improvement could be the interpolation of the ZMP Jacobian given a certain posture vector. At the moment, a simple Nearest Neighbour scheme is employed, which is a discrete non-linear interpolation. An alternative approach would try to utilise continuous methods like recursive regression learning. This approach is highly advantageous because the controller will not have to re-compute the ZMP Jacobians in-between compensation iterations and could simply retrieve approximate ones based on the regression method and the given posture.

The last aspect for extending this research is the experimental design. Two main concepts for extension are suggested. Firstly, the method could be applied to various humanoid models along with different configurations of the joints being adjusted. Experimenting with the method on different humanoid prototypes will further reinforce its potential. Unfortunately, in the short scope of this study it was not possible to carry out these extended experiments. Secondly, the performance measure of the dynamic stability achieved in the experiments was very simple; it

only took the number of FZMP occurrences into consideration. Another approach could incorporate far superior performance measures for the results (Kagami *et al.*, 2003; Sobotka *et al.*, 2003).

A different but interesting concept for future improvement to general research of humanoid control using simulation tools is the development of debugging framework for simulated robots. In most robotic simulations tools, it is impossible to debug the controller program as it is incorporated as a sub-process of the application. Finding a way to augment these tools with a debugging capability would make it much easier to develop complex controllers for humanoids.

7.3 Conclusions

While research still struggles to realise dynamically balanced humanoid locomotion to near human-like performance, this dissertation reveals a glimpse of the different complexities involved in achieving such a task. By choosing the ZMP as its core function to determine the dynamic stability, this dissertation has supported many other researchers in their belief that the ZMP based methods is the most promising approach. In addition, this dissertation proved that it is possible to devise humanoid locomotion control, which ignores the feet-ground contact phases and still performs well under various conditions. Moreover, the author believes that the biped locomotion research has attained sufficient maturity to concentrate more on the highest level of robotic systems – the humanoid. The research presented here, is only a small step to comprehend the full relationship between humanoid motion control and the ZMP. Hopefully, other future studies will advance further and bring the humanoid robots closer to their anticipated capabilities.

Bibliography

Akima, H. (1970). A new method of interpolation and smooth curve fitting based on local procedures. *Journal of the ACM, 17*(4), 589-&.

Arakawa, T., & Fukuda, T. (1997). *Natural motion generation of biped locomotion robot using hierarchical trajectory generation method consisting of ga, ep layers.* Paper presented at the IEEE Int'l Conf. on Robotics and Automation.

Collins, S. H., Wisse, M., & Ruina, A. (2001). A three-dimensional passive-dynamic walking robot with two legs and knees. *International Journal of Robotics Research, 20*(7), 607-615.

Cominoli, P. (2004). *Development of a physical simulation of a real humanoid robot* (Semester project). Lausanne: Swiss Federal Institute of Technology.

Craig, J. (1986). *Introduction to robotics: Mechanics and control*: Addison Wesley.

Dasgupta, A., & Nakamura, Y. (1999). *Making feasible walking motion of humanoid robots from human motion capture data.* Paper presented at the IEEE International Conference on Robotics and Automation.

Denavit, J., & Hartenberg, R. S. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics, 22*, 215-221.

Denk, J., & Schmidt, G. (2001, 22-24 November). *Synthesis of a walking primitive database for a humanoid robot using optimal control techniques.* Paper presented at the IEEE-RAS Int'l Conf. on Humanoid Robots, Tokyo, Japan.

Egger, J.-P. (2004). *Simulated robot programming contest* (Semester project). Lausanne: Swiss Federal Institute of Technology.

Faconti, D. (2003). *Isaac: Theoretical analysis and implementation of a humanoid robot.* Unpublished Masters Thesis, Politecnico of Torino, Chalmers University of Technology, Turin, Italy.

Fritsch, F. N., & Carlson, R. E. (1980). Monotone piecewise cubic interpolation. *Siam Journal on Numerical Analysis, 17*(2), 238-246.

Fujimoto, Y., Obata, S., & Kawamura, A. (1998). *Robust biped walking with active interaction control between foot and ground.* Paper presented at the IEEE International Conference on Robotics and Automation.

Fujita, M., Kuroki, Y., Ishida, T., & Doi, T. T. (2003). *Autonomous behaviour control architecture of entertainment humanoid robot SDR-4x.* Paper presented at the IEEE/RSJ Int'l Conference on Intelligent Robots and Systems (IROS2003).

Fujita, M., Kuroki, Y., Ishida, T., & Doi, T. T. (2003). *A small humanoid robot SDR-4x for entertainment applications.* Paper presented at the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2003).

Furuta, T., Tawara, T., Okumura, Y., Shimizu, M., & Tomiyama, K. (2001). Design and construction of a series of compact humanoid robots and development of biped walk control strategies. *Robotics and Autonomous Systems, 37*(2-3), 81-100.

Goswami, A. (1999). Postural stability of biped robots and the foot-rotation indicator (FRI) point. *International Journal of Robotics Research, 18*(6), 523-533.

Hardt, M., Wollherr, D., Buss, M., & von Stryk, O. (2002). *Design of an autonomous fast-walking humanoid robot.* Paper presented at the Proceedings of the 5th International Conference on Climbing and Walking Robots, Paris, France.

Hirai, K., Hirose, M., Haikawa, Y., & Takenaka, T. (1998). *The development of Honda humanoid robot.* Paper presented at the IEEE Int'l Conf. on Robotics and Automation.

Huang, Q., Kajita, S., Koyachi, N., Kaneko, K., Yokoi, K., Arai, H., et al. (1999). *A high stability, smooth walking pattern for a biped robot.* Paper presented at the IEEE International Conference on Robotics and Automation.

Huang, Q., Kaneko, K., Yokoi, K., Kajita, S., Kotoku, T., Koyachi, N., et al. (2000). *Balance control of a biped robot combining off-line pattern with real-time modification.* Paper presented at the IEEE Int'l Conf. on Robotics and Automation (ICRA '00).

Huang, Q., Li, K., & Nakamura, Y. (2001). *Humanoid walk control with feedforward dynamic pattern and feedback sensory reflection.* Paper presented at the IEEE Int'l Symposium on Computational Intelligence in Robotics and Automation.

Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Arai, H., Koyachi, N., et al. (2001). Planning walking patterns for a biped robot. *Robotics and Automation, IEEE Transactions on, 17*(3), 280-289.

Kagami, S., Kanehiro, F., Tamiya, Y., Inaba, M., & Inoue, H. (2000, 16-18 March). *Autobalancer: An online dynamic balance compensation scheme for humanoid robots.* Paper presented at the 4th International Workshop on

Algorithmic Foundations of Robotics (WAFR2000). Dartmouth College, Hanover, New Hampshire, USA.

Kagami, S., Kitagawa, T., Nishiwaki, K., Sugihara, T., Inaba, M., & Inoue, H. (2002). A fast dynamically equilibrated walking trajectory generation method of humanoid robot. *Autonomous Robots, 12*(1), 71-82.

Kagami, S., Mochimaru, M., Ehara, Y., Miyata, N., Nishiwaki, K., Kanade, T., et al. (2003). *Measurement and comparison of human and humanoid walking.* Paper presented at the IEEE International Symposium on Computational Intelligence in Robotics and Automation.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., et al. (2003). *Biped walking pattern generation by using preview control of zero-moment point.* Paper presented at the IEEE International Conference on Robotics and Automation (ICRA '03).

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Yokoi, K., & Hirukawa, H. (2002). *A realtime pattern generator for biped walking.* Paper presented at the IEEE International Conference on Robotics and Automation (ICRA '02).

Kajita, S., & Tani, K. (1991). *Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode.* Paper presented at the IEEE International Conference on Robotics and Automation.

Kajita, S., & Tani, K. (1991). *Study of dynamic biped locomotion on rugged terrain-theory and basic experiment.* Paper presented at the Fifth Int'l Conference on Advanced Robotics, "Robots in Unstructured Environments" (ICAR '91).

Kajita, S., Yamaura, T., & Kobayashi, A. (1992). Dynamic walking control of a biped robot along a potential energy conserving orbit. *IEEE Transactions on Robotics and Automation, 8*(4), 431-438.

Kim, J.-H., Park, K.-H., Jang, J.-S., Kim, Y.-D., Lee, B.-J., & Kim, K.-P. (2003). *Humanoid robot hansaram: Schemes for zmp compensation.* Paper presented at the International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS2003), Singapore.

Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M., & Inoue, H. (2003). *Motion planning for humanoid robots.* Paper presented at the Proceedings of the 11th Int'l Symp. of Robotics Research (ISRR '03).

Kuffner, J. J., Jr., Kagami, S., Inaba, M., & Inoue, H. (2000). *Graphical simulation and high-level control of humanoid robots.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2000).

Kuo, A. D. (1999). Stabilisation of lateral motion in passive dynamic walking. *International Journal of Robotics Research, 18*(9), 917-930.

Li, Q., Takanishi, A., & Kato, I. (1991). *A biped walking robot having a zmp measurement system using universal force-moment sensors.* Paper presented at the IEEE/RSJ International Workshop on Intelligent Robots and Systems, "Intelligence for Mechanical Systems".

Li, Q., Takanishi, A., & Kato, I. (1992). *Learning control of compensative trunk motion for biped walking robot based on zmp stability criterion.* Paper presented at the IEEE/RSJ Int'l Conference on Intelligent Robots and Systems.

Lim, H.-o., Setiawan, S. A., & Takanishi, A. (2001). *Balance and impedance control for biped humanoid robot locomotion.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems.

Matsuoka, K. (1987). Mechanisms of frequency and pattern control in the neural rhythm generators. *Biological Cybernetics, 56*(5-6), 345-353.

McGeer, T. (1990). Passive dynamic walking. *International Journal of Robotics Research, 9*(2), 62-82.

Michel, O. (2004). Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems, 1*, 39-42.

Minakata, H., & Hori, Y. (1994). *Realtime speed-changeable biped walking by controlling the parameter of virtual inverted pendulum.* Paper presented at the 20th International Conference on Industrial Electronics, Control and Instrumentation (IECON94).

Miyakoshi, S., Taga, G., Kuniyoshi, Y., & Nagakubo, A. (1998). *Three dimensional bipedal stepping motion using neural oscillators-towards humanoid motion in the real world.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS98).

Mojon, S. (2003). *Realisation of a physic simulation for a biped robot* (Semester project report). Lausanne: Swiss Federal Institute of Technology Lausanne (EPFL).

Mojon, S. (2004). *Using nonlinear oscillators to control the locomotion of a simulated biped robot.* Unpublished Diploma, Swiss Federal Institute of Technology Lausanne (EPFL), Lausanne.

Nagasaka, K., Inoue, H., & Inaba, M. (1999). *Dynamic walking pattern generation for a humanoid robot based on optimal gradient method.* Paper presented at

the IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC '99).

Nakamura, Y., & Yamane, K. (2000, 6-8 September). *Interactive motion generation of humanoid robots via dynamics filter.* Paper presented at the 1st IEEE-RAS International Conference on Humanoid Robots (Humanoids2000). MIT, Boston, Massachusetts, USA.

Nakanishi, J., Morimoto, J., Endo, G., Schaal, S., & Kawato, M. (2003, 27 - 31 October). *Learning from demonstration and adaptation of biped locomotion with dynamical movement primitives.* Paper presented at the IEEE/RSJ Int'l Conference on Intelligent Robots and Systems (IROS '03). Las Vegas, USA.

Napoleon, Nakaura, S., & Sampei, M. (2002). *Balance control analysis of humanoid robot based on zmp feedback control.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and System.

Nishiwaki, K., Kagami, S., Kuniyoshi, Y., Inaba, M., & Inoue, H. (2002). *Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired zmp.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and System.

Okumura, Y., Tawara, T., Endo, K., Furata, T., & Shimizu, M. (2003). *Realtime zmp compensation for biped walking robot using adaptive inertia force control.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2003).

Park, J. H. (2003). Fuzzy-logic zero-moment-point trajectory generation for reduced trunk motions of biped robots. *Fuzzy Sets and Systems, 134*(1), 189-203.

Park, J. H., & Cho, H. C. (2000). *An online trajectory modifier for the base link of biped robots to enhance locomotion stability.* Paper presented at the IEEE International Conference on Robotics and Automation (ICRA '00).

Park, J. H., & Chung, H. (1999). *ZMP compensation by online trajectory generation for biped robots.* Paper presented at the IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC '99).

Park, J. H., & Rhee, Y. K. (1998). *ZMP trajectory generation for reduced trunk motions of biped robots.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems.

Pratt, J., Dilworth, P., & Pratt, G. (1997). *Virtual model control of a bipedal walking robot.* Paper presented at the IEEE International Conference on Robotics and Automation.

Pratt, J., & Pratt, G. (1998). *Intuitive control of a planar bipedal walking robot.* Paper presented at the IEEE Int'l Conference on Robotics and Automation.

Raibert, M. H. (1986). *Legged robots that balance*: Massachusetts Institute of Technology.

Riley, M., Ude, A., Wade, K., & Atkeson, C. G. (2003). *Enabling real-time full-body imitation: A natural way of transferring human movement to humanoids.* Paper presented at the IEEE International Conference on Robotics and Automation (ICRA '03).

Schoenberg, I. J. (1969). Cardinal interpolation and spline functions. *Journal of Approximation Theory, 2*(2), 167-206.

Shih, C. L., & Gruver, W. A. (1992). Control of a biped robot in the double-support phase. *IEEE Transactions on Systems Man and Cybernetics, 22*(4), 729-735.

Sobotka, M., Wollherr, D., & Buss, M. (2003). *A Jacobian method for online modification of precalculated gait trajectories.* Paper presented at the 6th International Conference on Climbing and Walking Robots, Catania, Italy.

Sorao, K., Murakami, T., & Ohnishi, K. (1997). *A unified approach to zmp and gravity centre control in biped dynamic stable walking.* Paper presented at the IEEE/ASME International Conference on Advanced Intelligent Mechatronics.

Spong, M. W. (1989). *Robot dynamics and control*: John Wiley & Sons, Inc.

Stevens, W. R. (1999). *UNIX network programming: Inter-process communications* (Second ed. Vol. 2): Prentice Hall PTR.

Sugihara, T. (2004). *Mobility enhancement control of humanoid robot based on reaction force manipulation via whole body motion.* Unpublished PhD Thesis, University of Tokyo, Tokyo.

Sugihara, T., & Nakamura, Y. (2003, 4-8 March). *Whole-body cooperative cog control through zmp manipulation for humanoid robots.* Paper presented at the 2nd International Symposium on Adaptive Motion of Animals and Machines (AMAM2003), Kyoto, Japan.

Sugihara, T., Nakamura, Y., & Inoue, H. (2002). *Real-time humanoid motion generation through zmp manipulation based on inverted pendulum control.* Paper presented at the IEEE International Conference on Robotics and Automation (ICRA '02).

Taga, G., Yamaguchi, Y., & Shimizu, H. (1991). Self-organised control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological Cybernetics, 65*(3), 147-159.

Takanishi, A., Ishida, M., Yamazaki, Y., & Kato, I. (1985, 9-10 September). *The realisation of dynamic walking by the biped walking robot wl-10rd.* Paper presented at the International Conference on Advanced Robotics (ICAR1985).

Takanishi, A., Takeya, T., Karaki, H., & Kato, I. (1990). *A control method for dynamic biped walking under unknown external force.* Paper presented at the IEEE International Workshop on Intelligent Robots and Systems, "Towards a New Frontier of Applications" (IROS '90).

Takanishi, A., Tochizawa, M., Karaki, H., & Kato, I. (1989). *Dynamic biped walking stabilised with optimal trunk and waist motion.* Paper presented at the IEEE/RSJ International Workshop on Intelligent Robots and Systems, "The Autonomous Mobile Robots and Its Applications" (IROS '89).

Ude, A., Atkeson, C. G., & Riley, M. (2004). Programming full-body movements for humanoid robots by observation. *Robotics and Autonomous Systems, 47*(2-3), 93-108.

Vukobratovic, M., & Borovac, B. (2004). Zero-moment point - thirty-five years of its life. *International Journal of Humanoid Robotics, 1*(1), 157-173.

Vukobratovic, M., Borovac, B., Surla, D., & Stokic, D. (1990). *Biped locomotion: Dynamics, stability, control and application (scientific fundamentals of robotics)*.Berlin: Springer Verlag.

Vukobratovic, M., Frank, A. A., & Juricic, D. (1970). On the stability of biped locomotion. *IEEE Transactions on Biomedical Engineering, BM17*(1), 25-&.

Vukobratovic, M., & Juricic, D. (1969). Contribution to synthesis of biped gait. *IEEE Transactions on Biomedical Engineering, BM16*(1), 1-6.

Vukobratovic, M., & Stepanenko, J. (1972). On the stability of anthropomorphic systems. *Mathematical Biosciences, 15*(1-2), 1-37.

Wollherr, D., Buss, M., Hardt, M., & von Stryk, O. (2003). *Research and development towards an autonomous biped walking robot.* Paper presented at the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2003).

Yamaguchi, J.-I., Takanishi, A., & Kato, I. (1993). *Development of a biped walking robot compensating for three-axis moment by trunk motion.* Paper presented

at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93).

Yamaguchi, J., Inoue, S., Nishino, D., & Takanishi, A. (1998). *Development of a bipedal humanoid robot having antagonistic driven joints and three DOF trunk.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems.

Yamaguchi, J., Kinoshita, N., Takanishi, A., & Kato, I. (1996). *Development of a dynamic biped walking system for humanoid development of a biped walking robot adapting to the humans' living floor.* Paper presented at the IEEE International Conference on Robotics and Automation.

Yamaguchi, J., Soga, E., Inoue, S., & Takanishi, A. (1999). *Development of a bipedal humanoid robot-control method of whole body cooperative dynamic biped walking.* Paper presented at the IEEE International Conference on Robotics and Automation.

Yamaguchi, J., & Takanishi, A. (1997). *Development of a biped walking robot having antagonistic driven joints using nonlinear spring mechanism.* Paper presented at the IEEE International Conference on Robotics and Automation.

Yamane, K., & Nakamura, Y. (2000). *Dynamics filter - concept and implementation of online motion generator for human figures.* Paper presented at the IEEE International Conference on Robotics and Automation (ICRA '00).

Zhang, R., & Vadakkepat, P. (2003, 1-3 October). *Motion planning of biped robot climbing stairs.* Paper presented at the Federation of International Robot-soccer Association (FIRA) Robot World Congress.

Zheng, Y. F., & Shen, J. (1990). Gait synthesis for the sd-2 biped robot to climb sloping surface. *IEEE Transactions on Robotics and Automation, 6*(1), 86-96.

Zhu, C., & Kawamura, A. (2003). *Walking principle analysis for biped robot with zmp concept, friction constraint, and inverted pendulum model.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003).

Appendix A

Applying Kinematics using Denavit-Hartenberg Convention

This appendix describes how to apply kinematics for a humanoid robot model according to the Denavit-Hartenberg (D-H) convention (Denavit & Hartenberg, 1955). Although the output of this appendix was not used in the project (as explained in section 4.3.2), quite an effort was invested to extract this data and it could contribute to other researchers involved in humanoid trajectory planning. An example of how to apply this convention is presented at the end of this appendix. Note that this appendix does not aim to explain the forward or inverse kinematics for robots and in the next section will only briefly mention several ideas and definitions to make the descriptions clear. The interested reader can refer to other books which thoroughly do so (Craig, 1986; Spong, 1989).

A.1 Background

The forward kinematics problem is concerned with determining the position and orientation of a manipulator's end-effector, given the values for the joint variables of that manipulator. On the other hand, the inverse kinematics problem is concerned with determining values for the joint variables, which will achieve some desired position, and orientation for the end-effector of the robot.

A robot manipulator is usually composed from a set of links, which are connected by various joints. Joints can be very elementary (e.g. revolute or prismatic joints) with

only a single DOF, or they can be more complicated (e.g. ball and socket joints) with two DOFs. In this context, all joints are of revolute type, which makes analysis much simpler. Therefore, each joint's movement is depicted by one real number, i.e. in our case the angle of rotation. The forward kinematics goal is to resolve the cumulative effect of all the joints' variables and the following section will present the well-known convention that provides a systematic procedure to achieve this goal.

Since each joint connects two links, a robotic manipulator having n joints will consist of $n + 1$ links, this is also known as a "*kinematic structure*" or "*kinematic chain*". It is common to number the joints from 1 to n, and to number the links from 0 (the base link) to n. Also, a joint variable denoted by q_i is associated for each joint as follows:

(A.1) $$q_i = \begin{cases} \theta_i & \text{joint } i \text{ is revolute} \\ d_i & \text{joint } i \text{ is prismatic} \end{cases}$$

In addition, kinematic analysis demands attaching a coordinate frame O_i to each link i, which inherently defines the coordinates of each point on link i, expressed in O_i, independently of the manipulator's motion. The coordinate frame O_0 (attached to the base) termed as the *inertial frame*. Now, let us define $\mathbf{A}_i(q_i)$ as the homogeneous transformation matrix, which represents the position and orientation of O_i with respect to $O_{i\text{-}1}$, which is a function of the joint variable. Furthermore, the homogeneous transformation matrix, which represents the position and orientation of O_j with respect to O_i, is defined as follows:

(A.2) $$\mathbf{T}_j^i = \begin{cases} \mathbf{A}_{i+1}\mathbf{A}_{i+2}\ldots\mathbf{A}_{j-1} & \text{if } i < j \\ \mathbf{I} & \text{if } i = j \\ \left(\mathbf{T}_i^j\right)^{-1} & \text{if } i < j \end{cases}$$

Thus, the position and orientation of the end-effector in the inertial frame are given by:

(A.3) $$\mathbf{T}_n^0 = \mathbf{A}_1(q_1)\ldots\mathbf{A}_n(q_n)$$

In theory, that is the fundamentals to forward kinematics analysis, i.e. establish the functions $\mathbf{A}_i(q_i)$, and multiply them sequentially. However, the Denavit-Hartenberg notation further simplifies the process by adding other several conventions, which will be described next.

A.2 The Denavit-Hartenberg Notation

In general, it is possible to perform the kinematic analysis by assigning arbitrary coordinate frames to each link. However, it is useful to choose ones in a methodical fashion. Denavit and Hartenberg (1955) proposed a convention named after them for selecting frames of reference robotic structures and since then it has been commonly used worldwide. In this convention, each $\mathbf{A}_i$ transformation is represented as a product of four basic transformations:

$$\mathbf{A}_i = \mathbf{R}_{\theta_i}^{z}\mathbf{T}_{d_i}^{z}\mathbf{T}_{a_i}^{x}\mathbf{R}_{\alpha_i}^{x}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(A.4)}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where a_i, d_i, α_i, θ_i are the parameters associated with link i and joint i and defined as the *link length*, *link offset*, *link twist*, and *joint angle*, respectively. Since the $\mathbf{A}_i$ is a function of a single variable, essentially three of the four parameters are constant for a given link, while the fourth parameter is the joint variable as depicted in Eq. (A.1).

Obviously, it is not possible to represent any arbitrary homogeneous transformation using only four parameters since such a transformation is characterised by six

unknowns. However, many textbooks describing the D-H notation explain how to determine which homogeneous transformations can be expressed in the form of Eq. (A.4) (Craig, 1986; Spong, 1989). They also introduce two constraints, which must be satisfied in order to accomplish the D-H method:

[1] Axis x_i should be perpendicular to axis z_i.

[2] Axis x_i should intersect axis z_i.

Only under these constraints, they claim that a unique configuration of a_i, d_i, θ_i, and α_i exists such that Eq. (A.4) holds. Clearly, since θ_i and α_i are angles, they actually imply that they are unique with respect to a multiple of 2π.

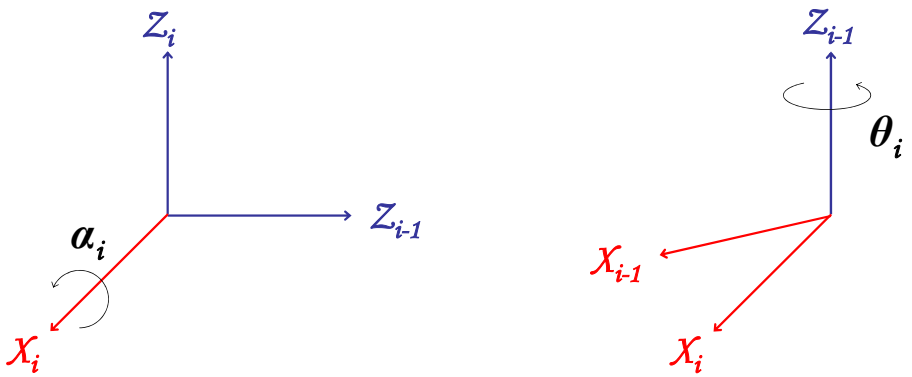


Figure A.1: How to determine the positive sense of angles α_i and θ_i.

Having established that each homogeneous transformation matrix satisfying the two constraints above can be expressed in the form (A.4), a physical interpretation to the four parameters in (A.4) can be described. The parameter a_i is the distance between axes z_{i-1} and z_i, measured along the axis x_i. The angle α_i is the angle between axes z_{i-1} and z_i, measured in a plane normal to x_i. This angle's positive sense is determined according to the right-hand rule from z_{i-1} to z_i as shown in **Figure A.1**. The parameter d_i is the distance between origin O_{i-1} and the intersection of x_i with z_{i-1} measured along the z_{i-1}. Finally, θ_i is the angle between x_{i-1} and x_i, measured in the plane normal to z_{i-1}. This angle's positive sense is also shown in **Figure A.1**.

Before summarising the procedure, one should note the following important facts. First, in some circumstances, placing frames 0...n in a way which satisfies the above two constraints will require situating the origin O_i of frame i in an unintuitive location. Second, choosing the various coordinate frames is not unique, despite the constraints above. Thus, it is possible to derive different, but equally correct, coordinate frame assignments for the robot's links which will all result in the same matrix $\mathbf{T}_n^0$, regardless of intermediate link frames assignment. Finally, in any case, whether joint i is revolute or prismatic, a_i and α_i are always constant. If joint i is prismatic, then θ_i is also a constant, while d_i is variable. Similarly, if joint i is revolute, then d_i is constant and θ_i is variable.

A.2.1 Algorithm

Here, the method based on the D-H notation is summarised in the following algorithm for extracting the any manipulator's forward kinematics:

Step (i): Situate and label joint axes $z_0, ..., z_{n-1}$.

Step (ii): Establish the base frame and place the origin anywhere on the z_0-axis. Choose x_0 and y_0 axes arbitrarily to form a right-hand frame.

Repeat steps **(iii)-(v)** for $i = 1, ..., n - 1$:

Step (iii): Situate origin of O_i on the intersection of z_i and z_{i-1}'s common normal and z_i. In case z_i intersects z_{i-1} situate O_i at that intersection. However, if they are parallel, situate O_i arbitrarily along z_i.

Step (iv): Fix x_i along z_i and z_{i-1}'s common normal through O_i or in case z_i intersects z_{i-1} along the direction normal to both of them.

Step (v): Fix y_i as such to complete a right-hand frame.

Step (vi): Situate the end-effector frame O_n. If the n-th joint is revolute, set z_n along the same direction as z_{n-1}. Establish the origin O_n arbitrarily along z_n, preferably at the centre of the end-effector. Fix x_n and y_n conveniently to form a right-hand frame.

Step (vii): Write a table of link i's parameters a_i, d_i, α_i, θ_i:

- $a_i \equiv$ distance along x_i from O_i to the intersection of the x_i and z_{i-1} axes.
- $d_i \equiv$ distance along $z_{i-1}$from O_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic.
- $\alpha_i \equiv$ the angle between z_{i-1} and z_i measured about x_i (see **Figure A.1**).
- $\theta_i \equiv$ the angle between x_{i-1} and x_i measured about z_{i-1} (see **Figure A.1**). θ_i is variable if joint i is revolute.

Step (viii): Form the matrices $\mathbf{A}_i$ by plugging in the above parameters into (A.4).

Step (ix): Form $\mathbf{T}_n^0 = \mathbf{A}_1(q_1)\ldots\mathbf{A}_n(q_n)$ to give the position and orientation of the end-effector frame expressed in the base coordinate frame.

A.3 Applying D-H Convention to Fujitsu's HOAP-2

As an example, this section presents the process of extracting all the data necessary to implement forward or inverse kinematics for the Fujitsu HOAP-2 Webots model created in a previous project (Cominoli, 2004). The Webots model is shown in **Figure A.2**. In addition, **Figure A.3** presents the full specification of this robot's model along with **Table A.1** which specifies the link lengths values. Note that the kinematic structure of the robot was separated to five separate kinematic chains. This is due to the fact that the typical structure of a humanoid is what is also known as a *tree-like* kinematic structure, and now there is still no kinematics programming library, which sufficiently supports such structures. However, one could perform separate kinematic analyses by using the concept that the base frame is mobile. Thus, at each control step of the humanoid one should compute the relative current transformation of the base frame in world coordinates and multiply this transformation (or its inverse) when performing forward or inverse kinematics calculations. This figure also shows the result of performing step (i) + (ii) of the algorithm presented in section A.2.1.

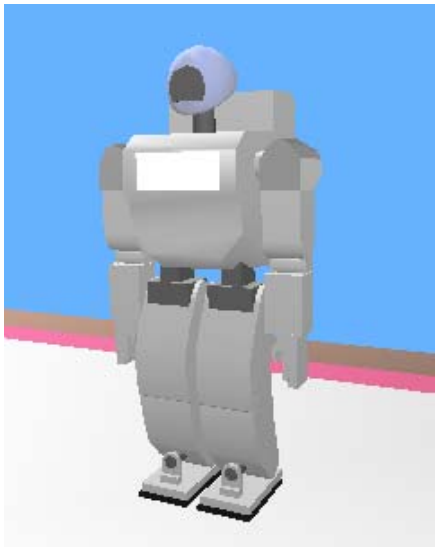


Figure A.2: The Webots model of Fujitsu's humanoid HOAP-2.

Table A.1: Link Lengths specification of HOAP-2.

Link	***Length [m]***
ARM1	0.1
ARM2	0.101
ARM3	0.146
ARM4	0.05
LEG1	0.039
LEG2	0.1
LEG3	0.1
LEG4	0.037
BODY1	0.09
BODY2	0.315
HEAD1	0.0025
HEAD2	0.085
WAIST1	0.055
WAIST2	0.034

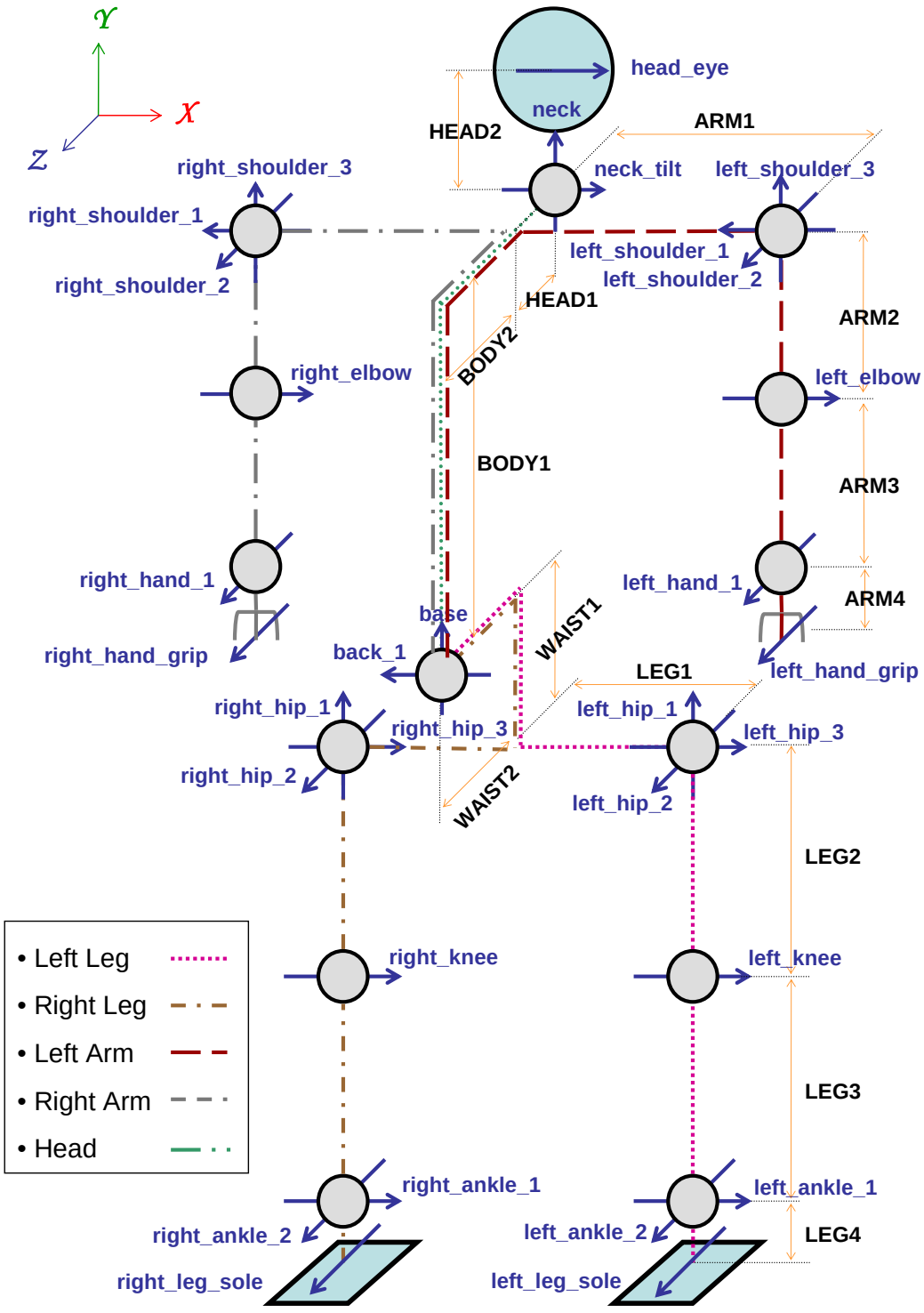


Figure A.3: HOAP-2 robot model specification.

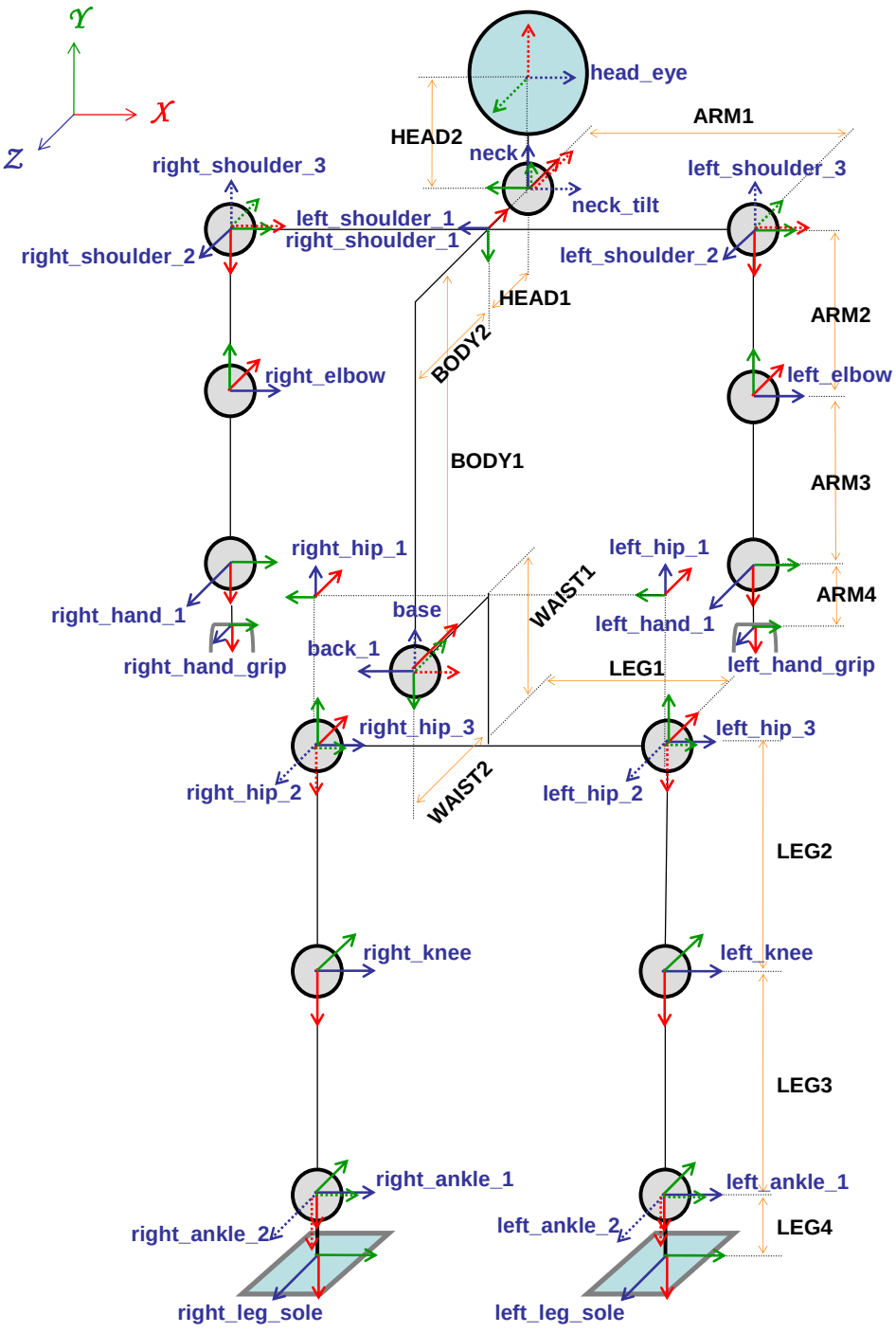


Figure A.4: Coordinate frames assignment according to D-H notation.

The consecutive steps (iii) – (vii) described in the algorithm presented in section A.2.1 and the coordinate frame assignments are depicted in **Figure A.4**. The results of step (vii) for each kinematic chain are presented in **Table A.2** which follows.

Table A.2: The extracted D-H parameters for HOAP-2 robot model.

Chain	***Link***	***a* [*m*]**	**α [rad]**	***d* [*m*]**	**θ [rad]**
Left Leg	back_1_0	0	-π/2	0	π/2
	left_hip_1_0	0.034	π/2	-0.039	0
	left_hip_3_0	0	π/2	-0.055	0
	left_hip_2_0	0	π/2	0	-π/2
	left_knee_0	0.1	-π/2	0	0
	left_ankle_1_0	0.1	0	0	0
	left_ankle_2_0	0	π/2	0	0
	left_leg_sole	0.037	0	0	0
Right Leg	back_1_0	0	-π/2	0	π/2
	right_hip_1_0	0.034	π/2	0.039	0
	right_hip_3_0	0	π/2	-0.055	0
	right_hip_2_0	0	π/2	0	-π/2
	right_knee_0	0.1	-π/2	0	0
	right_ankle_1_0	0.1	0	0	0
	right_ankle_2_0	0	π/2	0	0
	right_leg_sole	0.037	0	0	0
Left Arm	left_shoulder_1_0	0.0315	-π/2	0.09	π/2
	left_shoulder_2_0	0	-π/2	-0.1	π/2
	left_shoulder_3_0	0	-π/2	0	π/2
	left_elbow_0	0	π/2	-0.101	π/2
	left_hand_1_0	0.146	π/2	0	-π/2
	left_hand_gripper	0.05	0	0	0
Right Arm	right_shoulder_1_0	0.0315	-π/2	0.09	π/2
	right_shoulder_2_0	0	-π/2	0.1	π/2
	right_shoulder_3_0	0	-π/2	0	π/2
	right_elbow_0	0	π/2	-0.101	π/2
	right_hand_1_0	0.146	π/2	0	-π/2
	right_hand_gripper	0.05	0	0	0
Head	neck_0	0.0365	0	0.09	π/2
	neck_tilt_0	0	π/2	0	0
	head_center	0.085	0	0	π/2

With the above extraction of D-H parameters for this robot, one could use any kinematics package to perform the standard forward and inverse kinematics analysis. Specifically, one could perform trajectory planning for walking motions using inverse kinematics of the foot soles steps positions along with dynamically updating the base frame's position in world coordinates. However, the inverse kinematics is fundamentally a non-trivial optimisation problem due to uniqueness and joint limits problems, thus in the time scope of this project this idea was not used, and other options were applied as mentioned in section 4.3.

Appendix B

Simple Trajectory Smoothing

In this appendix, the method used to perform smooth joint trajectories for motion planning is presented. The aim was to perform trajectory smoothing dependent on the joint angles position and velocity only, without regarding other constraints such as acceleration and jerk. Thus, this method is termed *simple*. However, other well-known trajectory smoothing methods exists, usually closely linked to the mathematical concept of interpolation such as cubic, Bezier and Hermite splines. Let us start with the problem definition and later describe the underlying mathematics, which serves as the basis for this method.

B.1 Problem Definition

Given a sequence of joint angles (positions) of q_0, $q_{1\ldots}$ q_m for a particular joint, which defines some motion plan for that joint, how can one create a function or an interpolation that at any instance of time will output an intermediate position for the joint. Clearly, this is dependent on the time interval given for the whole motion and time intervals between each pair of positions. In addition, the solution relies on the type of function interpolated between the given positions. Various functions require different number of constraints in order to solve the problem, since there is different number of unknowns to solve for.

For this project's purposes, the trajectory smoothing method had to satisfy the following conditions:

[i]. The method had to be *efficient* in terms of computation and execution, since it was undesirable to slow down the motion control of the humanoid.

[ii]. The method had to provide *predictability* and *accuracy*. In other words, it should not degenerate near singularities.

[iii]. Joint position and velocity should be a smooth function of time.

[iv]. At time instance that corresponds to one of the control points given (i.e. the given joint positions) the interpolation should output exactly the control point value, since the person, who designed those control points, will expect the joint to have that angle at that specific time instance.

B.2 Implementation

To satisfy the four requirements above, it was decided to use a *cubic* function for each consecutive pair of control points in the given sequence for a joint, which has the following form:

(B.1) $$q_i(t) = a_i t^3 + b_i t^2 + c_i t + d_i$$

where $q_i(t)$ stands for the joint angle at time t using the cubic function i, and t is in some time interval $[t_i, t_{i+1}]$ also specified by the user. In addition, the pair of consecutive control points q_i and q_{i+1}. Such a cubic function has four unknown parameters, a_i, b_i, c_i, and d_i, which are called the coefficients. In order to solve the four coefficients, four independent equations were needed. By satisfying the Eq. (B.1) for both *start* and *end* positions two equations are generated. Taking the time derivative of Eq. (B.1) gives the following joint velocity equation:

(B.2) $$\dot{q}_i(t) = 3a_i t^2 + 2b_i t + c_i$$

Thus, by specifying the start and end velocity for the joint they can be plugged into Eq. (B.2) to obtain two additional constraints and thus the cubic's coefficients can be

solved. Nevertheless, how should one set the start and end velocities? A very simple but efficient scheme was chosen, which simply sets the start and end velocities to zero, as follows:

(B.3) $$\dot{q}_i = \dot{q}_{i+1} = 0$$

Clearly, since a separate cubic is fitted for each consecutive pair of control points q_i and q_{i+1} which is constrained by q_i, q_{i+1}, $\dot{q}_i$ and $\dot{q}_{i+1}$, the requirements [i], [iii] and [iv] are satisfied.

To avoid singularities and satisfy condition [ii], the coefficients for each cubic are solved in a time interval [$t_s = 0$, $t_e = t_{i+1} - t_i$]. Thus, by plugging the control points and velocities from Eq. (B.3) into Eqs. (B.1) and (B.2) on acquires:

(B.4) $$q_{i+1} = a_i t_e^{\,3} + b_i t_e^{\,2} + c_i t_e + d_i$$

(B.5) $$\dot{q}_{i+1} = 3a_i t_e^{\,2} + 2b_i t_e + c_i$$

(B.6) $$q_i = a_i t_s^{\,3} + b_i t_s^{\,2} + c_i t_s + d_i = d_i$$

(B.7) $$\dot{q}_i = 3a_i t_s^{\,2} + 2b_i t_s + c_i = c_i$$

And in matrix notation, it can be written as:

(B.8) $$\boldsymbol{H}_i \boldsymbol{x}_i \equiv \begin{bmatrix} t_e^{\,3} & t_e^{\,2} & t_e & 1 \\ 3t_e^{\,2} & 2t_e & 1 & 0 \\ t_s^{\,3} & t_s^{\,2} & t_s & 1 \\ 3t_s^{\,2} & 2t_s & 1 & 0 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = \begin{bmatrix} t_e^{\,3} & t_e^{\,2} & t_e & 1 \\ 3t_e^{\,2} & 2t_e & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = \begin{bmatrix} q_{i+1} \\ \dot{q}_{i+1} \\ q_i \\ \dot{q}_i \end{bmatrix} \equiv \boldsymbol{y}_i$$

In order to solve Eq. (B.8) to obtain the vector of coefficients for the cubic of pair (q_i, q_{i+1}), matrix inversion is used:

(B.9) $$\boldsymbol{x}_i = \left(\boldsymbol{H}_i\right)^{-1} \boldsymbol{y}_i$$

B.2.1 Performing Interpolation

Once the coefficients of all cubic functions $q_i(t)$ have been computed ($i \in \{0 \ldots m\}$) the following equation is used to perform the interpolation:

(B.10)
$$q(t) \underset{t_i \le t < t_{i+1}}{\equiv} q_i(t) = a_i(t - t_s)^3 + b_i(t - t_s)^2 + c_i(t - t_s) + d_i$$

B.2.2 Extending to Whole-Body Interpolation

So far, the fundamental mathematics involved in computing the cubic function to smooth a single pair of consecutive points (q_i, q_{i+1}) belonging to only one joint was described. With the intention of improving the computational efficiency of this trajectory smoothing mechanism for the humanoid, it was decided to extend the algorithm to all joints' trajectories, i.e. the humanoid's whole body, and thus perform interpolation simultaneously for all joints to get a vector of joint angles $\boldsymbol{q}$ for any given time instance t.

The idea is to perform the same computations as before but using matrix notation for the whole body. This means that the user can input a matrix $\boldsymbol{M}$ of dimensions $(n+1)\times(m+1)$, where $m+1$ is the number of control points for each joint, and $n+1$ is the number of joints as such:

(B.11)
$$\boldsymbol{M} = \begin{bmatrix} q_0^0 & \cdots & q_i^0 & \cdots & q_m^0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ q_0^j & \cdots & q_i^j & \cdots & q_m^j \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ q_0^n & \cdots & q_i^n & \cdots & q_m^n \end{bmatrix}$$

Furthermore, the user specifies a row vector $\boldsymbol{t}$ of length $m+1$, representing the time values for each control point as follows:

(B.12)
$$\boldsymbol{t} = \begin{bmatrix} t_0 & t_1 & \cdots & t_i & \cdots & t_m \end{bmatrix}$$

Given this initial data, one can compute and store four matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ of dimensions $m\times(n+1)$ which contain the coefficients for all joint-position pairs' cubic functions using Eqs. (B.8) and (B.9), such that:

$$a_i^j = \boldsymbol{A}_{j,i};\ b_i^j = \boldsymbol{B}_{j,i};\ c_i^j = \boldsymbol{C}_{j,i};\ d_i^j = \boldsymbol{D}_{j,i} \tag{B.13}$$

Figure B.1 presents a pseudo-code of initialising the sufficient data needed in order to perform the trajectory smoothing method for the whole-body joints.

```
function init (M : Matrix of n+1×m+1, t : RowVector of length m+1)
{
    A ← create_matrix(n+1,m) ; B ← create_matrix(n+1,m)
    C ← create_matrix(n+1,m) ; D ← create_matrix(n+1,m)
    for i = 1 : m ; do
        t_e ← t(i + 1) - t(i)
        t_s ← ε                    # ε is a very small no
        for j = 1 : n+1 ; do
            q_e ← M(j,i + 1) ; q_s ← M(j,i)
            v_e ← v_s ← 0
            x ← compute_coeff(t_s, t_e, q_e, q_s, v_e, v_s) # Eqs. (B.8) &(B.9)
            A(j,i) ← x(1) ; B(j,i) ← x(2) ; C(j,i) ← x(3) ; D(j, i) ← x(4)
        end
    end
    save A, B, C, D, t;
}
```

Figure B.1: Pseudo-code of initialisation of the method for whole-body trajectory smoothing. This pseudo-code assumes that the functions used exist. Matrices/Vectors are in **bold** face.

Interpolation is similar to the interpolation done for a single joint and single control points' pair. However, it is performed using the matrix notation, and thus outputs a vector $\boldsymbol{q}(t)$ of length $n+1$ which contains the current interpolated position (angle) for each joint (see **Figure B.2**).

```
function interpolate (t : Real) : returns ColumnVector of length n+1
{
    load A, B, C, D, t;
    i ← find index s.t. t(index) ≤ t < t(index + 1)
    dt ← t(i + 1) - t(i)
    # The following line is Eq. (B.10) in matrix notation:
    q = A(:, i) · dt^3 + B(:, i) · dt^2 + C(:, i) · dt + D(:, i)
    return q
}
```

Figure B.2: Pseudo-code for whole-body simultaneous interpolation. This pseudo-code assumes that the functions used exist. Matrices/Vectors are in **bold** face and "**X**(:, *i*)" means column *i* of **X**.

B.2.3 Adjusting Interpolations to Fit Corrections

It might well be that one would one to adjust the interpolations according to intermediate correction via points. A good example can be found in Chapter 4, in which the proposed method of this thesis needs to adjust the predefined motion trajectories. Moreover, these adjustments will probably be carried out quite frequently in the course of the run, thus, a robust and computationally efficient adjustment method should be provided to the trajectory smoothing control.

Here, the user provides a matrix $\boldsymbol{Q}_c$ of dimensions $(n+1)\times(p+1)$, where $p+1$ is the number of correction via points for each joint, and $n+1$ is the number of joints as such:

$$\boldsymbol{Q}_c = \begin{bmatrix} \partial \boldsymbol{q}_0 & \dots & \partial \boldsymbol{q}_i & \dots & \partial \boldsymbol{q}_p \end{bmatrix} \tag{B.14}$$

Furthermore, the user specifies a row vector $\boldsymbol{t}_c$ of length $p+1$, representing the time instances for each correction via point as follows:

$$\boldsymbol{t}_c = \begin{bmatrix} t_0^c & t_1^c & \dots & t_i^c & \dots & t_p^c \end{bmatrix} \tag{B.15}$$

In addition, the row vector $\boldsymbol{t}$ (length m+1), represents the time instances currently used in the trajectory smoothing as in Eq. (B.12).

As explained in the next following paragraphs the idea is fairly simple. The adjustment will create a merged time row vector $\hat{\boldsymbol{t}}$ of length r ($r \leq p+m+2$) which will contain the sorted union[n] of $\boldsymbol{t}$ and $\boldsymbol{t}_c$ as:

$$\textbf{(B.16)} \qquad \hat{\boldsymbol{t}} = \begin{bmatrix} \hat{t}_0 & \hat{t}_1 & \dots & \hat{t}_i & \dots & \hat{t}_r \end{bmatrix}$$

Furthermore, it creates a merged via points matrix $\hat{\boldsymbol{M}}$ of dimensions (n+1)×r which corresponds to $\boldsymbol{t}'$ as follows:

$$\textbf{(B.17)} \qquad \hat{\boldsymbol{M}} = \begin{bmatrix} \hat{\boldsymbol{q}}_0 & \dots & \hat{\boldsymbol{q}}_i & \dots & \hat{\boldsymbol{q}}_r \end{bmatrix}$$

Once the matrix $\hat{\boldsymbol{M}}$ and the row vector $\hat{\boldsymbol{t}}$ are created, the method simply re-initialises the trajectory smoothing with them. This will cause the new trajectory smoothing to act according to the old data adjusted with the new corrections. In the following page, **Figure B.3** presents the pseudo-code for the adjustment method and reveals how $\hat{\boldsymbol{t}}$ and $\hat{\boldsymbol{M}}$ are constructed.

[n] Note that union essentially ensures uniqueness.

```
function adjust (Q_c : Matrix of n+1×p+1, t_c : RowVector of length p+1)
{
    load t;       # has length m+1
    t' ← create_row_vector(p+m+2) ; M' ← create_matrix(n+1, p+m+2)
    i ← 1 ; j ← 1 ; r ← 1
    repeat until (i > m+1) ; do
        if ( j > p + 1 or  t(i) < t_c(j) ) { # t(i) is smaller or none left in t_c
            t'(r) ← t(i)
            M'(:, r) ← interpolate ( t(i) )
            i ← i + 1
        } else {          # t(i) is greater equal
            t'(r) ← t_c(j)
            M'(:, r) ← interpolate ( t_c(j) ) + Q_c(:, j) # add corrections
            j ← j + 1
            if ( t(i) == t_c(j) ) { i ← i + 1 }
        }
        r ← r + 1
    end
    init (M', t') # re-initialize the trajectory smoothing M' and t'
}
```

Figure B.3: Pseudo-code for whole-body interpolation adjustment method. The method uses the previous shown methods in Figure B.1 and Figure B.2.

Appendix C

Computing Matrix Pseudo-Inverse using SVD

In this appendix, an algorithm for computing the Moore-Penrose pseudo-inverse of a general matrix using Singular Value Decomposition (SVD) is presented. In this study's context, this method was used to compute the pseudo-inverse of the Jacobean matrices which in general case can be singular and rectangular (non-square). Due to this and the fact that most linear algebra libraries in C++[o] do not contain an implementation of the pseudo-inverse, it was necessary to implement such a method.

C.1 Background

The classical definition of matrix inversion is confined to square and non-singular (regular) matrices, thus the number of multivariate problems, which can be solved by matrix algebra, is very limited. In this thesis' context, the calculation of Jacobean matrices, which are not necessarily regular, demands an extension of the matrix inversion concept.

C.1.1 The Moore-Penrose Pseudo-Inverse

The matrix inversion has been extended to a more general level in the form of the Moore-Penrose pseudo-inverse. Its definition is as follows:

[o] NEWMAT version 11 was used in this project, and it still does not provide a pseudo-inverse implementation.

Let $\boldsymbol{A}$ be an arbitrary matrix of order $m \times n$, and $\boldsymbol{A}^+$ be a matrix of order $n \times m$. $\boldsymbol{A}^+$ is called the "*Moore-Penrose pseudo-inverse*" of $\boldsymbol{A}$, if it satisfies the four following conditions:

- $\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{A}$
- $\boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}^+$
- $\left(\boldsymbol{A}\boldsymbol{A}^+\right)^T = \boldsymbol{A}\boldsymbol{A}^+$
- $\left(\boldsymbol{A}^+\boldsymbol{A}\right)^T = \boldsymbol{A}^+\boldsymbol{A}$

In simple words, the first two conditions mean that all non-invertible properties of a matrix are neglected while inverting the rest and the other two conditions choose a suitable matrix $\boldsymbol{A}^+$ out of all those matrices, which satisfy the first two rules.

It is also true that $\boldsymbol{z} = \boldsymbol{A}^+\boldsymbol{x}$ is the minimum least-squares solution to the problem: $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{z}$. If the inverse of ($\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$) exists, then $\boldsymbol{A}^+ = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T$ this can be seen by multiplying both sides of $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{z}$ by $\boldsymbol{A}^{\mathrm{T}}$ to get $\boldsymbol{A}^T\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{z}$ and thus creating a symmetric matrix, which can be inverted to give: $\boldsymbol{z} = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{x} \equiv \boldsymbol{A}^+\boldsymbol{x}$.

C.1.2 Singular Value Decomposition of a Matrix

Trying to solve linear equation systems, which are singular by Gaussian elimination or by LU decomposition, will fail or result in unstable solutions. In such cases, the linear equation systems can be solved by Singular Value Decomposition (SVD).

SVD is based on the following linear algebra theorem: "A matrix $\boldsymbol{A}$ of dimensions $m \times n$, can be transformed into a product of three matrices $\boldsymbol{U}_{m\times n}$, $\boldsymbol{D}_{n\times n}$, and $\boldsymbol{V}^T_{n\times n}$, i.e. $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$. $\boldsymbol{U}$, $\boldsymbol{D}$, and $\boldsymbol{V}$ have specific properties: First, $\boldsymbol{U}$ and $\boldsymbol{V}$ have orthonormal columns such that $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$. Second, $\boldsymbol{V}$ is quadratic and finally, $\boldsymbol{D}$ is diagonal and its diagonal values are sorted in descending order".

C.2 Algorithm

As mentioned earlier, the algorithm is based on SVD of the input matrix. In order to neglect any singular values, a simple factor was used which serves as an indicator on how much singularity will be tolerated. In other words, any singular values less than the tolerance factor are treated as a zero. For simplicity, the method described here sets a default tolerance value and does not let the user determine it (or give it as an additional input). The pseudo-code of the algorithm is shown in **Figure C.1**.

```
function pseudo-inv (A : Matrix of m×n) : returns Matrix of n×m
{
    (U, D, V) ← SVD(A)          # get SVD of given matrix
    if (m > 1)                  { d ← get_diagonal(D) }
    else if (m == 1) { d ← D(1, 1) }

    tolerance ← max(m, n) · norm(A) · ε      # ε is a very small no.
    r ← count all elements e ∈ d s.t. e > tolerance
    if (r == 0)                 { A+ ← create_zero_matrix(n, m) }
    else                        { Σ ← create_diag_matrix(1/d(1),…,1/d(r))
                                  A+ ← V(:,1:r) · Σ · (U(:,1:r))T }
    return A+
}
```

Figure C.1: Moore-Penrose pseudo-inverse calculation pseudo-code.

Some clarifications must be made concerning the pseudo-code presented in **Figure C.1**. First of all, vectors and matrices are written in **bold** face and an expression such as "*x*:*y*" means all the range between *x* and *y*, while an expression ":" means all available range. Secondly, the pseudo-code assumes that all the functions used (e.g. SVD, create_zero_matrix, get_diagonal etc.) exist either in the matrix library employed or self-implemented. Lastly, the smallest precision of the C++ 'double' type was utilised for the value of ε, which is roughly 2.22e-16.

The implementation of Moore-Penrose matrix pseudo-inverse described above was proposed for addition to the NEWMAT[p] C++ matrix library, which was extensively utilised in this project.

[p] For more information about NEWMAT package, see: http://www.robertnz.net/nm11.htm.