Feedback Control Design for MARLO, a 3D-Bipedal Robot

by

Alireza Ramezani

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2013

To Nasrin.

ACKNOWLEDGEMENTS

I have received support and encouragement from a great number of individuals. With this dissertation, my Ph.D. journey at the University of Michigan is coming to an end. Working with Professor Jessy Grizzle, my mentor, colleague, and friend, has made this a thoughtful and rewarding endeavor. I would like to thank Jessy for his invaluable support as my research project evolved from an idea to a completed study. Jessy's patience and mathematical prowess offered me the chance to learn and digest complex control design concepts. His persistent attendance in the lab and participation in experiments has always been heartwarming and encouraging.

I would like to extend my appreciation to my dissertation committee, Professors Bernstein, Remy, Gordon, and Hulbert, for their support over the past year. Special thanks goes to professor Remy for the one-one-one meetings after my Ph.D. proposal presentation. I found his constructive advice on techniques of optimization both interesting and helpful.

I appreciate the tremendous efforts of Professor Jonathan Hurst and his research team at Oregon State University for the arduous task of designing and manufacturing MARLO. Moreover, I am thankful to Professor Hurst for hosting our University of Michigan team during the design process and initial testing of MARLO.

During the first year of my studies I enjoyed the chance of working with Dr. Koushil Sreenath, now a faculty member at CMU, and Dr. Hae-Won Park, now a research scientist at MIT. From the task of preparing MABEL for demos, repairing the rotation limiter, helping on lab tours, experimenting late into the night, and from the Dexter-Ann Arbor half-marathon run in mid-summer with all the sweating and pain, Koushil and Hae-Won created memorable times. I owe each of them a debt of gratitude.

I am thankful to Dr. Kaveh Akbari Hamed, research scientist at the University of Michigan, for being a supportive colleague. His efforts at revising my journal papers, debugging my optimization codes, and designing feedback strategies are invaluable intellectual contributions to my Ph.D. work. I am also thankful to M.S. student Xinning Shen for her contributions to embedded programming, GUI design, and interfacing a joystick into xPC target. Her joystick program in particular helped me to obtain new auto-start results. I am thankful to Angela Farrehi and Dr. Hsiao-Wen Lo as they prepared me to face the challenges of studying abroad. Last but not least, I am thankful to everyone who challenged me to improve during my time at Michigan.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

ABSTRACT

Feedback Control Design for MARLO,
a 3D-Bipedal Robot

by

Alireza Ramezani

Chair: Professor Jessy W. Grizzle

This work develops feedback controllers for bipedal walking in 3D on level ground, both in simulation and experimentally. MARLO is a new robot that has been designed for the study of 3D-bipedal locomotion, with the aim of combining energy efficiency, speed, and robustness with respect to natural terrain variations in a single platform. The robot is highly underactuated, having six actuators and, in single support, 13 degrees of freedom. Its sagittal plane dynamics are designed to embody the spring loaded inverted pendulum (SLIP), which has been shown to provide a dynamic model of the body center of mass during steady running gaits in a wide diversity of terrestrial animals. A detailed dynamic model is used to optimize walking gaits with respect to the cost of mechanical transport (cmt), a dimensionless measure of energetic efficiency. A feedback controller is designed that balances the robot during the *quiet standing* mode, and another feedback policy is developed such that the robot can take a *transition step* from quiet standing to walking. A feedback controller is designed that stabilizes steady-state 3D walking gaits, despite the high degree of underactuation of the robot. These controllers are combined through a state machine that handles switching among the three controllers controllers. In experiments on planarized

(2D) and untethered (3D) versions of the robot with point feet and passive feet (prosthetic feet) walking over flat ground or on a ramp with a shallow slope, the adaptability of the designed controller to the environment (planar or untethered, flat ground or ramp), and to the morphology of the robot (point feet or passive feet), is demonstrated. In experiments on a planarized version of the robot with passive feet, the controller yielded stable walking after starting from quiet standing, autonomously and without any intervention from the operator. In experiments on an untethered (3D) version of the robot, the controller was able to balance the robot over flat ground or on a shallow ramp during the quiet standing mode. In addition, the controller yielded six-untethered "human-like" steps after starting from quiet standing, autonomously without any intervention from the operator.

CHAPTER I

Introduction

Currently, no robotic solutions can compete with the efficiency, speed and agility of mammalian locomotion. The speed and agility of a cheetah chasing its quarry and the accuracy of a mountain goat climbing up craggy hill sides are examples of animal locomotion that are inspiring roboticists. The mechanisms of mammalian walking and running are quite complicated and involve the use of various tendons on the back as energy storing springs to enhance energy efficiency. Mammalian legged locomotion remains one of engineering's most challenging questions: *how to design machines that can efficiently navigate rough terrain with agility and speed?*

This chapter begins with an overview of the mainstream approaches to achieving stable bipedal locomotion in machines. It is followed by the objectives of this work and an explanation of how they relate to the existing literature. The chapter concludes with the organization of the dissertation.

1.1 Literature Overview

1.1.1 Time-independent vs. Time-dependent control strategies

Two major trends in feedback policy design are apparent in the robot locomotion literature: "time-dependent" and "time-independent" control policies. Time-dependent policies rely on an external clock to generate a rhythmic walking pattern [1], [2], [3], [4]. Typically, the state of the clock is used to parameterize a set of joint trajectories, which are then implemented on the robot via a classical control method, such as PD control or computed torque. The design of the rhythmic external reference for the joints has been based on several methods, including "bio-inspired neural oscillators," the "zero moment point," and "passive dynamics." As discussed below, in some cases the state of the robot is fed back to adjust the phase of the external clock.

Time-independent policies, on the other hand, seek to generate a rhythmic motion without recourse to an external clock [5]. Several approaches have been taken, including a naturally oscillating element (i.e., spring) in the robot itself [6]; using virtual models and Jacobians to define torques that keep the torso upright while advancing the robot's body forward [7]; using feedback to shape the potential energy of a robot so that it walks as if on sloped ground [8, 9]; and using a gait-timing variable that is a monotonically increasing function of the robot's configuration during a walking pattern [10].

1.1.2 Biologically Inspired Schemes

Roboticists have sought inspiration in nature for creating the rhythmic motions associated with walking. An important theme in this regard are "central pattern generators" (CPGs), which are networks of neurons that produce periodic signals without external forcing. Such oscillators are believed to originate rhythmic activities such as walking gaits, wing flapping, fish swimming, hearts beating, and respiration. Mathematical models of neural oscillators presented in [11, 12, 13, 14, 15, 16, 17] suggest a mutual inhibition network composed of a continuous-variable neuron model in order to generate oscillation.

A specific instance of this is Matsuka's neural oscillator [12], which consists of two simulated neurons in mutual inhibition; when gains are tuned properly it exhibits oscillatory behavior. Matsuka's oscillator was used in the KIST humanoid robot. A reduced model of the robot, including virtual dampers and springs that capture the dominant dynamic response of the robot in the frontal plane, was coupled with the neural oscillator model in order to balance the robot.

Buchanan [14] proposed a circuit for the lamprey locomotor network consisting of three bilateral pairs of inter-neurons (two inhibitory and one excitatory) and a bilateral pair of motoneurons. Similar models have been employed in robotics to generate rhythmic motions [18, 19, 20, 21, 22, 23, 24, 25]. Collins and Richmond [26] designed a series of computer experiments to test whether a hard-wired CPG can produce multiple phase-oscillation patterns that correspond to natural animals gaits. In order to address their hypothesis, they modeled a quadrupedal locomotor CPG as a system of four coupled nonlinear oscillators and tested the ability of these models to generate three common quadrupedal motions — walking, trotting and bounding. They addressed and analyzed three common oscillator models of the Stein neuronal model, Van der Pol oscillator and FitzHugh-Nagumo model as the unit oscillator for the quadrupedal locomotor. They attempted to generate transitions between the different gaits by changing the networks' driving signal and by altering the internal parameters of the CPG oscillators.

In another work, the "musculo-skeletal" system proposed by Taga [27] consisted of a chain of rigid links confined to the sagittal plane and walked on compliant ground. It used a neural rhythmic generator composed of six oscillators connected in an inhibitory fashion. The robot was modeled as an inverted pendulum actuated with antagonistic muscles. Two sensors measured the angle of the inverted pendulum with respect to the ground when it was leaning forward and backward. A stretch-reflex-like feedback strategy was responsible for the entrainment between the activity of the neural oscillator and the inverted pendulum representation of the robot.

Or *et al.* [18] proposed a combined CPG-ZMP (central pattern generator and zero moment point) approach to generate walking gaits using a WABIAN-based robot model with flexible spine. Their model was a human-size robot with relatively similar human-like physiology which hosted a two-stage controller. Primarily, a ZMP-based pattern generator generated a walking pattern for the joints of the robot; then a CPG model generated rhythmic motions for the flexible spine. The controller retrieved the force data to measure ZMP (discussed below) and verified that the ZMP was inside the support polygon. If the robot was on the verge of toppling, an inhibitory signal was sent to reduce the neural activity within the CPG by dropping the global excitation.

1.1.3 Passive Dynamics

Passive dynamics was introduced and studied primarily with the motivation of energy efficiency. The secondary motivation for studying passive dynamics is the fact that passive walkers often demonstrate a natural walking pattern. In robots designed based on passive dynamic principles, the dissipation of energy due to the impact or damping is compensated for with variations in the total potential energy level of the the robot. Realizing a rhythmic walking pattern and maintaining a stable cyclic gait without reliance on active control effort was first studied by McGeer (1990) [28], in an attempt to replicate energetically efficient walking patterns comparable to humans. As McGeer pointed out in his original paper, passive walkers can be traced back to a diagram by Fallis in 1888 for a toy walker [29], which produced leg clearance by waddling side to side. Formal analysis of passive walking seems to have begun with Mochon and McMahon (1980), who noted that human walking is at least partially passive because "no signal is supplied to a leg during its swing phase" [30]. However, McGeer provided the first mathematical model for un-powered walking down a shallow slope, showing clearly how "gravity and natural dynamics alone" (meaning no actuation is required) could generate rhythmic gaits. McGeer called this "passive walking." He proved the correctness of his model and mathematical analysis by building a bipedal robotic mechanism based on his passive walking principle. McGeer's mechanism did not rock from side to side to achieve swing foot clearance as did the toy by Fallis. Instead, with each step, small motors retracted the swing foot end far enough to clear the ground.

Similar to McGeer's studies, Goswami [31, 32] studied the passive gait of an un-powered compass-like legged robot with point masses and knee-less legs, which was kinematically equivalent to a double inverted pendulum. Through the systematic study of the compass-gait model on inclined slopes, he showed that in response to a continuous perturbation of the gait parameters, such as ground slope angle, a symmetric and asymptotically stable gait of the un-powered robot gradually evolves through a regime of bifurcations characterized by progressively more complicated asymmetric gaits, eventually reaching a state where no consecutive steps are similar.

Kuo [33] proposed a simple controller for stabilizing a 3D-passive walker after showing the passive system was unstable. The passive walking machine consisted of two legs and a pelvis and walked forward by placing one foot on the ground and riding on the stance leg, which rolled forward as an inverted pendulum mounted on the stance foot. The swing legs was moving in a pendular arc, bringing the foot forward so that it was making ground contact. He defined three DoF in his passive walker: two were analogous to those of a planar two-legged walker without knees similar to McGeer's work, and the third DoF allowed the machine to move side to side in the frontal plane of walking. Assessment of the stability of the machine, showed that there were unstable modes in the dynamics of the passive walker corresponding to the lateral side-to-side motion. Five potential controllers with minimal control authority, reflecting the philosophy of passive walker design, were proposed: the first three controllers suggested direct modifications of the frontal trajectory; two controllers suggested foot-placement in the frontal plane of walking.

The bipedal kneed passive dynamic walker by Collins [34, 35] was similar to McGeer's passive walker in that it too could walk only over shallow slopes. The mechanism was distinguished from its predecessors, however, by walking stably in 3D. Compensation for the yaw and roll instability were achieved through detailed mechanical design instead of actuation, sensing, and feedback. In fact, yaw compensation was achieved by friction (to increase reactive torques), a flexible ankle (to achieve better ground contact), and counter swinging arms (to negate the yaw motion induced by the forward motion of the swing leg). Side-to-side lean (roll) compensation was also achieved through specific mechanical designs such as the use of arc-shaped feet, together with freely laterally moving arms with high friction joints, which seemed to help the dissipation of lateral side-to-side energy and resulted in stable passive walking.

Cornell's Ranger developed by Ruina [36] is an example of a self-contained (computing unit and batteries on-board), tether-free, four-legged and knee-less biped that could walk on flat ground. The mechanism was designed to have favorable "passive dynamics," meaning that it could walk stably down shallow slopes without actuation.

The robot's overall control strategy was similar to that in the work of Mura and Shimoyama [4], where the controller consisted of an open-loop time-based pattern generator and a feedback controller to stabilize a set point. The nominal trajectories were based on minimization of an energy metric. The feedback controller applied corrections at the end of each step on the basis of measurements made at the beginning of the step. The nominal trajectories and feedback controller were designed to exploit the robot's "natural dynamics."

1.1.4 Zero Moment Point (ZMP)

The most widely used control scheme in the robotics community is based on the ZMP or "zero moment point" criterion introduced by Vukobrativić *et al.* [37]. To define the zero moment point, assume the robot is in single support, with the stance foot flat on the ground and not sliding. The point on the ground where the resultant of the ground reaction forces acting on the stance foot produces zero moment in the ground plane is the zero moment point or ZMP. The "support polygon" of the robot is defined to be the convex hull of the stance foot. Vukobrativić's ZMP criterion is that the vertical component of the ground reaction force be positive and the zero moment point lie in the support polygon. When this is true, the foot will not rotate and will act as a base of support for the robot. To ensure that the foot is in fact not sliding, the resultant of the ground reaction forces must lie in the friction cone. Due to its very nature, the ZMP criterion leads to flat-footed waking.

Walking patterns that satisfy the ZMP criterion have been computed both off-line and on-line, as illustrated in Vukobratović *et al.* [38], Vukobratović and Juricic [39], Goswami [40], Wieber [41], Hemami and Farnswort [42], Arakawa and Fukuda [43], Hirai *et al.* [44], to name just a few of the hundreds of references that employ this method. In most of the applications of the ZMP method, the ankle torque is the primary actuator for shifting the ZMP safely within the support polygon. The maximum torque that can be compensated for is limited by the size of the support polygon. For this reason, robots using this method of control are fitted with large feet.

WABOT-1 [45] was the first powered robot with human morphology and of human size. It was able to communicate with humans in Japanese and had external sensors for measuring the distance to objects around it. WABOT-1 demonstrated successful tether-free walking motions using a controller based on the ZMP criterion. Its walking motion was realized with two levels of control action, one dealing with the relative motion of the links with respect to the torso, and a second level concerned with the overall dynamic balance of the mechanism, so that during the walking motion the ZMP was adjusted to keep the whole foot in contact with the ground.

The bipedal robot WABIAN [46, 47] used a family of smooth polynomials, determined with angular velocity, angular acceleration and height of the swing leg with respect to the ground, to generate a smooth motion of the swing leg during the single support phase. A moment compensation controller based on the ZMP criterion sought to counter the destabilizing moments generated by the moving links of the robot. The characterization of the ZMP as the point where the resultant ground force has zero moment about the ground plane was used to compute a set of nonlinear equations relating the position of the ZMP to a reference frame and a system of point masses representing the links of the robot. Using an on-line recursive calculation, the moment compensation controller minimized the errors that represented the divergences of two decoupled linearized ZMP equations from the desired pre-computed ZMP positions.

The ZMP approach was employed in the development of KHR-2 [48] with the desire to produce an anthropomorphic robot with light weight, backlash-free actuators (harmonic drives), simple kinematic structure and low power consumption. It consisted of two arms, two legs, a torso and a head. The robot used a multistage dynamic walking controller based on the ZMP criterion and consisted of a real time damping controller, ZMP-based balance compensator, and a landing position controller. The damping controller, designed on the basis of a simplistic inverted pendulum model of the robot aimed to damp out the oscillations generated during the single support phase. A linear fifth-order ZMP compensator designed on the basis of pole-placement techniques compensated for the movement of the ZMP within the support polygon by moving the torso forward or backward and the pelvis side-to-side. The landing controller regulated both the swing foot landing position and orientation as well as landing time.

Johnnie [49], an anthropomorphic robot equipped with 6-axis force sensors in the feet, applied ZMP for both desired trajectory generation and dynamic balance achievement. A reduced dynamic model of Johnnie lumping its overall mass at the center of gravity was used during trajectory generation to represent the relation between the motion of the CoM and the position of the ZMP. The objective of its ZMP compensator was to satisfy a set of constraints that described the unilateral contact between the foot and the ground. The unilateral constraints were used to formulate a "linear complementary problem" (LCP) whose solution met the ZMP constraints. The resulting trajectories were tracked via computed torque, as in Hemami and Katbab [50], Lee and Liao [51], Hurmuzlu [52], Yang [53], Jalics *et al.* [54], Lum *et al.* [55], Song and Guo [56], Park [57], and Taga [58].

In [59], Furusho and Sano proposed a hierarchical ZMP-based control structure in order to balance a nine-link tether-free bipedal robot called BLR-G2. They considered the frontal and sagittal dynamics separately and designated a controller to each dynamic. An optimal regulator strategy (Frank [60], Saidouni and Bessonnet [61]) balanced the reduced linearized frontal dynamics of the robot after selecting a performance index and deriving the optimal control law. The sagittal controller was composed of a torque controller and a position controller, which switched from one to the other based on the actual state of the robot.

1.1.5 Raibert's Controller

Despite the long-standing investigations of the potential advantages of active stability and intermittent support, the slow progress in building robots that employ these techniques highlights the difficulty of the task. The first dynamically stable machine with an intermittent support point seems to have been Raibert's hopper, a monopedal spring-legged machine employing an intuitive control scheme to maintain balance and move forward [6]. The hopping machine was composed of a pneumatic leg attached to a rolling-cage torso through a set of gimbals where the rolling-cage carried the electronics, gyros, accelerometer and pressure regulators. Turning to the control strategy, the robot employed three simple and decoupled strategies to regulate attitude, forward motion velocity and the height of hopping. The robot controlled forward running by positioning the foot with respect to the position of the COM. The attitude control was achieved during the moment that the foot was in contact with the ground by applying torque at the hip joint. The pneumatic leg provided the system with enough energy to bounce in a periodic motion while absorbing the energy at the impact moment, hence achieving energetic efficiency during hopping. The height of hopping was regulated by the regulation of air pressure of the pneumatic leg.

1.1.6 Passivity-based control

Motivated by the passive walkers, Spong and Bullo [9] demonstrated that stable walking could be achieved in a fully actuated biped by "potential energy shaping." Specifically, for the case of a three dimensional biped with n DoF and n independent actuators, they showed that changes in the ground slope define a group action on the configuration manifold and that the kinetic energy and the impact dynamics are invariant under this group action. Moreover, it is possible to use feedback to modify the potential energy of the system in order to achieve invariance of a passive limit cycle (i.e., a limit cycle that would be present in the biped walking down a slope and using no actuation). They introduced a potential energy forming ([62, 63, 64]) controller which ensured the biped is invariant under the slope changing action and they referred to that as "control symmetry" because the Lagrangian of the open-loop system was not invariant under the group action, but became invariant after feedback [65].

1.1.7 Virtual model control

Pratt *et al.* [5], introduced the notion of virtual model control and illustrated it on two bipedal walking robots, the Spring Turkey and the Spring Flamingo. The control scheme was based on the simulation of virtual components (e.g., spring, damper, mass, latch, bearing, nonlinear potential field, and dissipative field) in order to generate joint torques that create the same effect that the virtual components would have created if had they actually been presented and connected to the robot. For example, to maintain the torso upright, the virtual model control conceptually attached a spring and damper between the top of the torso and the world frame.

Pratt *et al.* control strategy assumed that all the joints of the robot were actuated. This limited the applicability of the method to robots with actuated feet and gaits that maintain the foot flat on the ground. This limitation is overcome with the next method we discuss.

1.1.8 Hybrid Zero Dynamics

In [10, 66], Grizzle *et al.* introduced a control design method with provable stability properties for a planar three-link bipedal robot with passive point feet, torso, hip and two identical legs. They used the monotonically increasing horizontal position of the hip to reparametrize the desired trajectories which encode the angles of the joints of the robot (e.g., leg angles); hence, this work falls in the category of time-independent control schemes.

The core of this gait design and stabilization methodology was the judicious choice of a set of holonomic constraints (output functions) which were enforced with a feedback law. Constraints enforced via feedback are called “virtual constraints”. In general, the maximal internal dynamics of a system that are compatible with the output being identically zero is known as the “zero dynamics” [67]. The importance of the zero dynamics is multi-fold as it allows provable stability analysis of the closed-loop system, and second, it reduces the dimension of the dynamic model, making it possible to conduct fast parametric walking pattern design based on nonlinear optimization methods. In contrast to Grizzle’s early work where only the zero dynamics corresponding to the swing phase is considered, Westervelt’s work [68, 69] showed that the zero dynamics, in general, is not invariant under the impact map and as a result it is not possible to relate the stability properties of the zero dynamics to the stability properties of the closed-loop system unless the required invariance conditions are met. Meeting these conditions results in the notion of the “hybrid zero dynamics” (HZD) .

Chevallereau *et al.* [70] reported the first successful walking experiment based on HZD using Rabbit an anthropomorphic bipedal robot with point feet. Later, Westervelt *et al.* [71] used Rabbit to extensively study the performance of HZD based controllers which induced stable walking on underactuated robots. They studied the stability properties of the resulting walking motions in terms of lower-dimensional sub-dynamics of the robot’s full dynamics.

After successful biped walking experiments, HZD-based control design was employed for biped running. A continuous HZD controller was employed by Chevallereau *et al.* [72] in order to achieve a stable running gait with Rabbit. The controller consisted of two parts, a continuous action controller based on HZD and a discrete action which adjusted the coefficients in the continuous portion of the controller in order to achieve landing objectives that ensure the existence of HZD. They studied the proposed running control law in an extensive simulation study by computing the Poincaré return map for 10 different running trajectories to demonstrate the feasibility of a stable running gait with Rabbit. Morris *et al.* [73] employed a feedback law based on HZD framework in order to achieve dynamic running with Rabbit. They reported six consecutive running steps with long stride length, flight phase, upright posture and average speed of $0.6[\frac{m}{s}]$. Recent successful experimental studies based on the HZD framework were reported by Sreenath *et al.* [74] on running of an anthropomorphic bipedal robot with point feet called MABEL . Park *et al.* [75] reported implementation of HZD-based control strategies for achieving a stable walking gait with MABEL over highly rough terrain.

1.2 Objectives of the Study

The main goals of the study are twofold — analytical and experimental. The analytical portion includes the design of stable walking gaits for MARLO, a tether-free anthropomorphic bipedal robot of human size, which is energy efficient and relatively fast, walking on flat ground of at least $1[\frac{m}{s}]$. The experimental portion primarily focuses on achieving a stable walking gait with the robot in the lab, with no attention paid to the energy efficiency.

The analytical studies addressed in this study are motivated by the energetically efficient, fast and agile walking pattern of biologic systems. The desire here is to replicate such motions with a robot physiologically similar to animals. Various robotic missions require robots that can operate outdoors relying only on on-board power supplies such as batteries or fuel tanks with limited energy capacity. During missions where access to additional energy resources is highly unlikely, power budgets are tight and energy efficiency plays a very critical role in fulfilling the objectives of the missions. Currently, operation time for legged robots is limited. BigDog is a rough-terrain robot that walks, runs, climbs and carries heavy loads. It has four legs that articulate like those of an animal, with compliant elements to absorb shock and recycle energy from one step to the next. BigDog has demonstrated recovery from falling after a large perturbation applied to it, and it has demonstrated complex navigation tasks through rough terrain. However, the robot is powered by a gasoline engine that drives an energy consuming hydraulic actuation system, and has only been able to achieved 12.8 miles without refueling. In 2010, Ranger [36] walked non-stop for 14.3 miles. In 2012, by improving the control algorithm and the electronics, Ranger's energy use was reduced by 43 % over that of 2010 and Ranger walked a 40.5 mile ultra-Marathon of 186,076 steps. However, Ruina's Ranger demonstrated a slow walking speed of approximately 1.32 mph over flat ground and could not clear obstacles higher than a few millimeters.

1.2.1 Theoretical Milestones

The above contrasting examples of BigDog and Ranger helped establish the overall goal of the MARLO project, which is to demonstrate walking gaits that are comparable to the speed and robustness of BigDog's walking gait and energy efficiency of Ranger's strides. However, this study tries to primarily assess the feasibility of these objectives through a theoretically organized framework after taking into account the robot's design and morphology. Theoretical milestones include developing a dynamic model of the robot, developing an optimization algorithm, developing a gait initiation feedback policy, developing a periodic walking feedback policy and developing an algorithm to verify the stability of the periodic walking gait. In the following sections, each theoretical milestone is described briefly and the relevant studies are cited as a means to highlight the similarities and differences to prior studies in perspective.

1.2.1.1 Dynamic Model

Developing a model capturing the dynamic characteristics of the robot for control algorithm design is seen as a preliminary step in a "model based controller design" approach for MARLO. To develop the model, three methodologies are possible: (1) modeling based on data-driven methods known as "system identification," (2) physical modeling based on known differential algebraic equations governing robot dynamics, [1] and (3) modeling based on the combination of system identification and physical modeling. A combination of these three methods is used by Park *et al.* [76] to model MABEL. Such a model provides a simulation environment for verifying the performance and functionality of the developed walking controller algorithms before implementing them on the actual robot.

1.2.1.2 Optimization Algorithm

The method used in this study is inspired by Westervelt *et al.* [77] and Sreenath *et al.* [78], who reported successful optimal gait design for the planar robots Rabbit and MABEL, based on HZD optimization. In this study, the HZD method is taken into 3D and series elastic actuation, which together more than double the dimension of the underlying mathematical model. After a careful choice of the controlled variables, the zero dynamics is computed for MARLO. It is 14 dimensional, whereas the zero dynamics for MABEL was dimension 4. The increased dimensionality makes the associated gait design problems much more numerically challenging. One way that we deal with this challenge is to introduce a new family of virtual constraints based on non-uniform rational B-splines, or NURBs for short. Using the zero dynamics and the new family of curves for the virtual constraints, gaits are designed through numerical optimization subjected to equality and inequality constraints (e.g., unilateral contact points, friction).

1.2.1.3 Gait Initiation Feedback Strategy

Few works have tackled gait initiation, a transition from a stand-still configuration to periodic walking, in bipedal locomotion and a relatively large number of studies consider periodic walking in the first place. The current studies on gait initiation strategies use ZMP-based algorithms and deal mainly with fully actuated anthropomorphic robots with big flat feet, where achieving and maintaining stability is trivial. Among the very few gait initiation

[1]Method of Lagrange.

studies, Wight *et al.* [79] introduced a measure called “foot placement estimator” (FPE) to restore balance to an unstable system. They defined a region of stability and, based on these regions, the derivations of FPE took place, which it was later extended to a complete gait cycle using the combination of a state machine and a series of simple linear control strategies.

Even robots such as MABEL and Rabbit, with physical design similarities[2] to MARLO, and similar control architectures, employed only a periodic walking controller, with gait initiation taking place through a push by the operator. As such, these robots are not fully autonomous. In fact, since Rabbit and MABEL’s walking motions are constrained to the sagittal plane of walking with a boom supporting these robots in the frontal plane, the push force does not cause lateral instability. In contrast, to these studies, the high degree of underactuation in MARLO highlights the necessity for a gait initiation feedback policy to ensure a transitory step from stand-still to periodic walking.

1.2.1.4 Planar and Tether-free Periodic Walking Feedback Strategy

The perspective and the relevance of MARLO’s feedback scheme to the prior studies has been addressed previously; however, it is worth looking at MARLO’s feedback from a different perspective. In fact, MARLO’s feedback strategy attempts to stabilize the robot without over-reliance on external sensory systems (e.g., vision sensors, laser range finder). In this context, MARLO’s walking controller has similar responsibilities to those of a human’s spinal controller, which regulates reflex actions. Evidence exists of both spinal and brain regulations of walking in human-beings, and based on [80], spinal regulated motions are involuntary and usually happen quickly in response to environmental stimuli where the human brain has no time to perform data processing and high level path planning, whereas brain control strategies are employed as the human brain receives sensory feedback data such as eyes and it involves the time consuming path planning tasks that affect the perfor-

[2]They are all categorized as underactuataed bipedal robots. In contrary to Rabbit , MABEL has compliant components, i.e., springs.

mance of human motions. Therefore, the design objective of MARLO's walking control lies in a context different from that of ASIMO, HRP-2 and similar androids, where visual and sonar feedback are employed.

1.2.1.5 Walking Stability Verification

Determining the existence of stability properties (i.e., asymptotically, exponentially stability) of a periodic orbit for a model including nonlinear dynamics and impact dynamics (hybrid nonlinear dynamics) is practical using classical methods that lead to rigorous conclusions. The goal is to use Poincaré maps to determine the stability characteristics of the robot in a closed-loop with the controller algorithm.

1.2.2 Experimental Milestones

Experimental milestones include the development of a mobile computing kernel, implementation and verification of an autonomous planar walking strategy, and implementation and verification of an autonomous tether-free walking strategy. The milestones follow.

1.2.2.1 Mobile Signal Processing Kernel

The mobile signal processing kernel aims to interface MARLO's controller algorithm with numerous sensors and actuators through a high bandwidth network of microprocessors known as slave units [81]. MARLO's communication architecture, which resembles a modular electronic data acquisition mechanism, facilitates the expansion of this network. The network is composed of seven electronic printed circuit boards (PCB), each called a medulla and each hosting two microprocessors. Each medulla is hard wired to the sensors and actuators of the specific parts of the robot (e.g., inertial measurement unit (IMU)). At each sample time, a dual level communication involving medulla-sensor communication, using a universal asynchronous receiver/transmitter (USART), and medulla-network communication, using a serial peripheral interface bus (SPI), takes place.

1.2.2.2 Autonomous Planar Walking

An autonomous planar walking experiment involves MABEL-like walking experiments with MARLO using an external power supply and a lateral support provided by a boom; during these experiments, the frontal plane angles of the hips will be held to a constant value to mimic a planar walker.

1.2.2.3 Autonomous Tether-Free Walking

Autonomous tether-free walking in the laboratory takes place with the help of an overhead crane found in industrial environments. The hoist is running on a "fixed" rail and the only possibilities, as long as the robot is connected to the hoist through a chain-pulley mechanism, are walking forwards and backwards in the laboratory, unlike planar walking, where external power supplies are used, tether-free walking utilizes internal power (i.e., Lipo batteries).

1.3 Organization of the Dissertation

The dissertation is organized as follows. Chapter II describes the design philosophy of MARLO and gives an overview of the mechanism and the electronics. Similarities and differences of MARLO with respect to existing robots are discussed in Section 2.1.1. In Chapter III, after introducing the modeling hypotheses, both a planar and a 3D dynamic model are developed using the method of Lagrange. In Chapter IV, after addressing the selection of virtual constraints adapted to the morphology of the robot, zero-dynamics-based optimization is used to design energetically efficient walking gaits. Chapter IV analyzes the walking efficiency of the 3D model of MARLO, measured by the "coefficient of mechanical transport" (cmt) [35], for average walking speeds varying from 0.5 to $1.4[\frac{m}{s}]$. Chapter V introduces a family of curves, non-uniform rational B-spline (NURB), to parameterize the virtual constraints. Properties of NURB curves over Bézier polynomials for gait design are highlighted, with NURB curves shown to improve the energetic efficiency of MARLO. Chapter VI presents a continuous-time time-invariant feedback controller based on virtual constraints and input-output linearization to asymptotically stabilize one of the optimal walking gaits obtained using the optimization method. Stability is checked with a Poincaré section analysis. The continuous-time feedback controller is augmented with an event-based controller to stabilize gaits in the 3D model. Chapter VII addresses feedback control design for two modes of "quiet standing," and "quiet standing to walking transitory step." In Chapter VIII, the experimental results for planar and 3D walking experiments are presented and discussed. Finally, the concluding remarks, contributions of the dissertation, and future works are presented in Chapter IX.

CHAPTER II

MARLO

This chapter is organized as follows. First, an overview of the design philosophy of MARLO is given followed by a mechanical-design-focused literature review with the purpose of comparing and contrasting MARLO's design with respect to current and prior robots. Next, MARLO's mechanism and electronics are explained in separate sections, where the mechanism section describes the mechanical components, e.g. torso, legs, hips and springs and the electronics section focuses on body-frame sensors, inertial sensors and the communication architecture linking these components. Finally, the last section summarizes the material provided in this section.

2.1 Testbed Overview

The bipedal robot [82] MARLO shown in Fig. 2.1 is a version of the ATRIAS 2.1bipedal robot series, a collaborative effort of Oregon State University, Carnegie Mellon University, and the University of Michigan. The robot has been conceived for energy efficiency, speed, and robustness with respect to natural terrain variations, without over-reliance on external sensing, such as vision. The robot is untethered, with all computation and power on board, using electric motors, batteries, and mechanical springs for cyclic gait-energy storage.

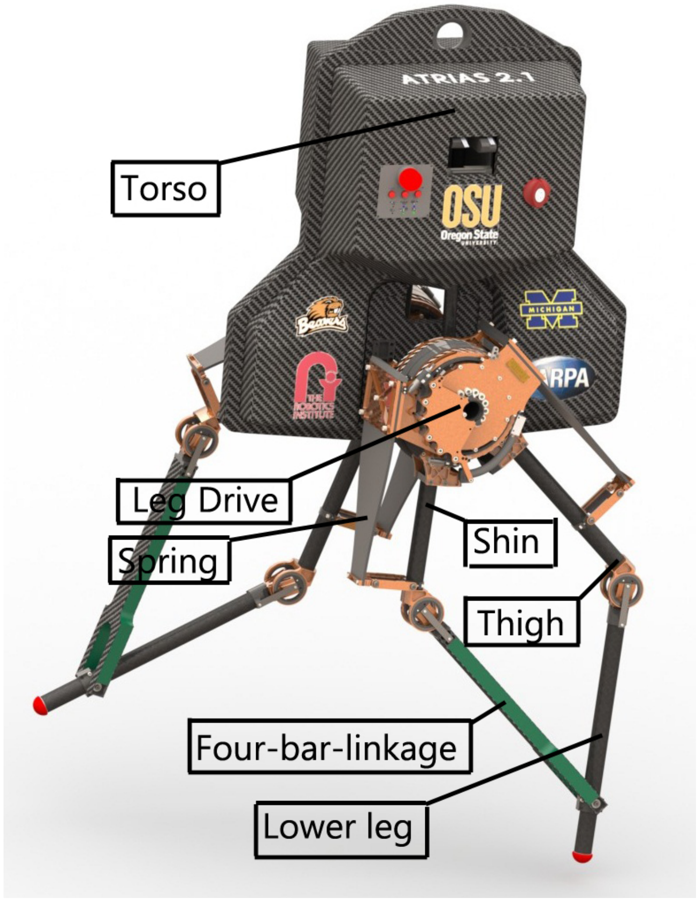


Figure 2.1: MARLO, a tether-free bipedal robot with point feet and springs.

2.1.1 Design philosophy

MARLO's sagittal plane dynamics is designed to embody the "spring-loaded inverted pendulum" (SLIP) model, which has been shown to approximate the body CoM motion during the "steady-state running gaits" of a wide diversity of terrestrial animals [83, 84, 85, 86, 87]. Successful running robots, such as the Planar Hopper, ARL Mono Pod II and CMU Bowleg Hopper, also exhibit SLIP model behavior [88, 89, 90]. An earlier machine by Hurst and Grizzle, the planar bipedal robot MABEL, also approximates a SLIP model: the robot achieved a peak walking speed of $1.5[\frac{m}{s}]$ [91], a peak running speed of $3[\frac{m}{s}]$ [92], and walked over terrain with variations exceeding 15% of leg length [75, 93]. These robots demonstrate that the spring-mass model is a successful and promising approach to machine design for robotic running. It is also promising for walking gaits: the same spring-mass model arranged in a bipedal pair has also been shown, in simulation, to reproduce the steady-state dynamics of human walking [94]. MARLO takes the spring-mass design philosophy into 3D, and combines it with tether-free electric actuation.

A number of robots have been built for the purposes of walking and running, [95, 96, 89, 70, 97]. One class of robots relies on large actuators and active control to implement a variety of behaviors. Examples include robots with rigid transmissions such as Rabbit and ASIMO [70, 97]. While these robots are capable of an aerial phase, it is at the expense of heavy actuators with high energetic cost and potentially unpredictable dynamic behavior at ground impact.

At the other end of the spectrum, the McGeer walker and similar robots use only passive elements. While these robots can efficiently walk down shallow slopes, they are extremely sensitive to disturbances and hard code only a single behavior [98, 28].

There are robots that fall in between those two extremes. For instance, the Cornell's Ranger adds minimal actuation at the ankle to an otherwise passive machine and can maintain a very efficient walking gait on flat ground. Yet this and similar machines [26, 35] retain the extreme sensitivity of passive dynamic walkers to disturbances and the focus on only one dynamic behavior.

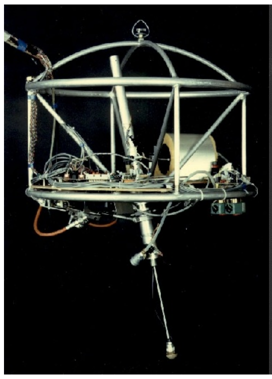
The MIT Leg Lab's Spring Flamingo walking biped and Boston Dynamics' BigDog walking and running quadruped both use series springs, a passive dynamic element, to improve the performance of their actuators in certain situations. Like the rigid-transmission robots, however, they attempt to create all gait dynamics through software control [99, 100]; the springs on these robots are primarily for force sensing and mechanical filtering purposes, and are essentially a soft load cell, acting as a force sensor for the low-level controllers [101]. Although this approach can result in impressive agility and robustness surpassing a rigid system, the energetic cost is still very high. For example, Big Dog uses a gasoline engine as a power source.

Recently, robots have been purposely built with high-bandwidth actuators and no springs, with the objective of implementing compliance in software. Examples include the hydraulically actuated HyQ [102] and MIT's electrically powered Cheetah [103].

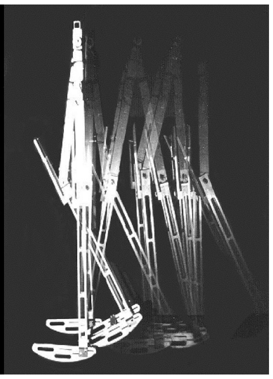
The jury is still out as to which design philosophy will provide the best tradeoffs in terms of agility, efficiency, and ease of control. MARLO is very much in the camp of seeking to combine the advantages of passive dynamics with actuation and feedback control for achieving a wide range of legged mobility. Its design exploits springs for energy-efficient steady-state locomotion and accommodating large disturbances, while using powerful actuators to achieve legged dexterity and gait stability when needed during transient behaviors.

(a) ASIMO

(b) Rabbit



(c) Raibert Hopper

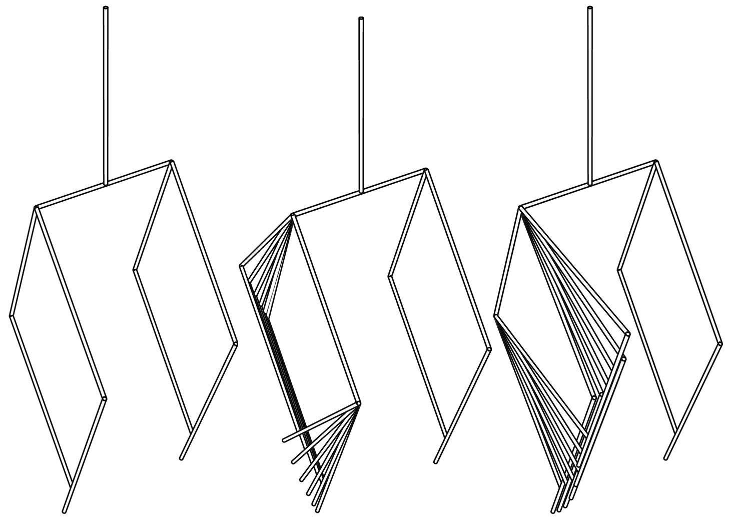


(d) McGeer Walker

Figure 2.2: Legged robot with various morphology.

2.1.2 Mechanism

MARLO's mechanical design is aimed at matching the key features of a SLIP model. While a massless limb is an abstract mathematical concept, MARLO's legs account for less than 10% of the total weight of the robot, making them as massless as physically possible. Each leg is formed by a parallelogram mechanism composed of lightweight carbon fiber tubes. The placement of two coaxial motor drives, which actuate the two topmost links of the parallelogram mechanism, is such that the mass concentration takes place at the hip, which is the point where the two legs are pinned to each other, thus enhancing the SLIP-like features of the robot. Through the parallelogram mechanism, both motor drives contribute to the leg angle and leg length changes. The design results in a smaller choice of motor drives because the stance force, which is acting at the end of the stance leg, is compensated for through the contribution of both motors. If, on the other hand, separate motors were designated for leg angle and leg length, a single motor would compensate for the stance force, requiring a larger motor. Figure 2.3a illustrates the parallelogram mechanism; Figures 2.3b and 2.3c show the contribution of the two topmost carbon fiber tubes in the leg angle and leg length changes. Figure 2.4 shows the contribution of the hip join in the lateral motions of the leg.



(a) 4-bar parallelogram (b) Shin contribution (c) Thigh contribution

Figure 2.3: **(a)** Parallelogram mechanism. **(b, c)** Contribution of the two topmost carbon fiber tubes in the leg angle and leg length changes, respectively.

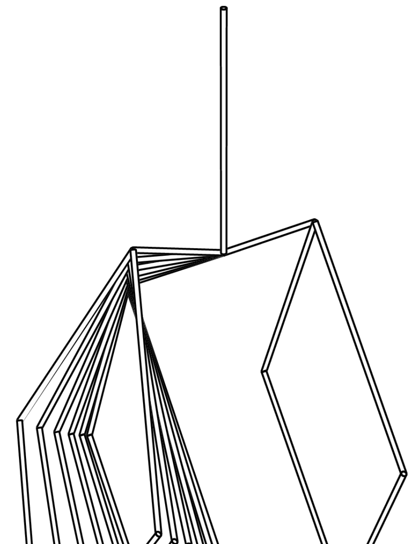


Figure 2.4: Contribution of the right hip joint in lateral motions of the leg.

To construct a “series-elastic actuator” (SEA), a fiberglass leaf spring, with stiffness chosen to coincide with the natural frequency of the preferred locomotion gait, and sized large enough to store gait energy, is placed between each of the upper links of the parallelogram mechanism and a motor drive output shaft. The series-elastic actuators in the legs are powered by electric motors mounted in the hip through a 50:1 harmonic drive. Hence the reflected inertia of the rotors of the motors dominate leg rotation, while the light legs and springs dominate the impact dynamics. In other words, the parallelogram mechanism allows the leg to be extremely lightweight, which minimizes the impact losses of the foot during touchdown of each stride, and maximizes the gait energy that can be stored in the springs and recycled. MARLO’s SEA is illustrated in Fig. 2.5. The overall configuration of the parallelogram mechanism with mass-less carbon fiber tubes and the motor harmonic drives is dynamically equivalent to a SLIP model. Most commonly, the SLIP model assumes a linear spring stiffness; however, due to the configuration of the parallelogram and fiberglass leaf spring, the equivalent spring actuating the two topmost carbon fiber tubes has a nonlinear stiffness.

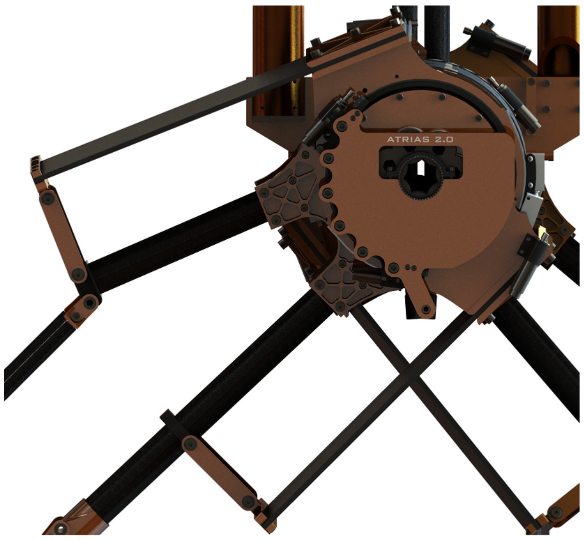


Figure 2.5: MARLO’s series-elastic actuator composed of a motor, a harmonic drive, fiberglass leaf spring, and carbon fiber tube.

In order to extend the robot’s motion from planar to 3D, the left and right legs are hinged to the torso through revolute joints at the hip, with a common axis that enables the legs to rotate normal to the sagittal plane. The point where the legs attach is not equal to the CoM of the robot, which is less in keeping with the features of a SLIP model. Each leg is independently actuated in the frontal plane by a brushless DC-motor, which is considerably smaller than those driving the legs in the sagittal plane. The motor is attached to the hip through a rigid carbon fiber tube, with the motor being mechanically attached to a semi-circular part using 4 wire-reinforced timing belts. The equivalent gear ratio of this hip actuation mechanism is 26.7:1 and is shown in Fig. 2.6. Each hip is approximately 15 cm wide, so that the total width of the hips when parallel is 30 cm.

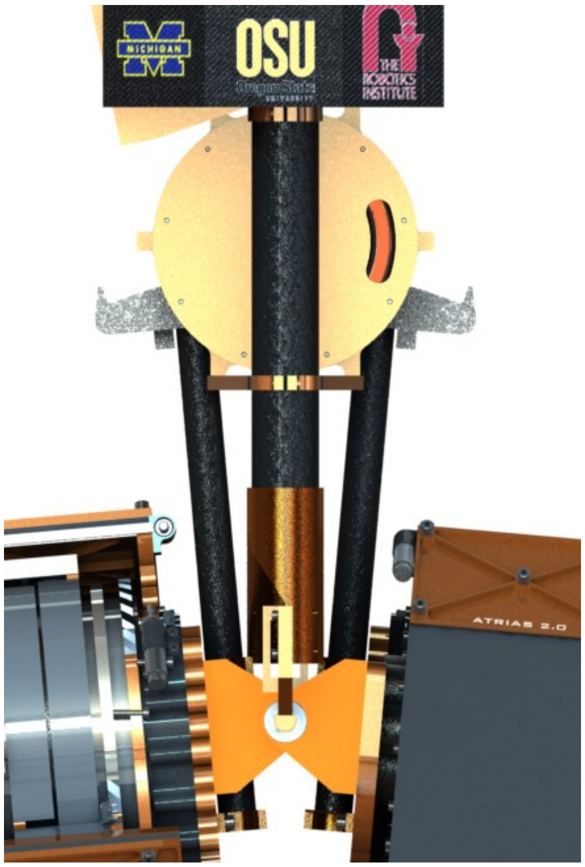


Figure 2.6: Hip actuation mechanism.

As indicated above, the hip actuation motors are mounted on the torso, which accounts for approximately 40% of the mass of the robot. Moreover, the torso houses the on-board real-time computing, amplifiers, and batteries.

2.1.3 MARLO's DoF and DoU

MARLO's legs each have four DoF in the sagittal plane, where two DoF arise from the two upper links of the parallelogram used in the leg construction, and two DoF arise from the fiberglass springs in the SEAs. Each leg's motions in the frontal plane is attached to the torso through a revolute joint, which adds one more DoF to each leg for a total 5 DoF for each leg. Because each leg has 5 DoF and 3 actuators, there are 2 degrees of underactuation (DoU) per leg.

Six DoF are associated with the translation and orientation of the torso when MARLO is unconstrained (e.g., in flight phase). However, during a single support phase, where one foot is in contact with the ground surface, the $(x - y - z)$ positions of the torso are dependent on other configuration variables; hence, overall three DoF are associated with the torso. Because MARLO's feet are unactuated, there is no direct control over the orientation of the torso, resulting in 3 additional DoU. It follows that the robot is highly underactuated. Indeed, when MARLO is not in contact with the environment, it has sixteen DoF, whereas when it is in contact with the ground, it has thirteen DoF. It has six independent actuators and seven DoU.

2.1.4 Electronics

As illustrated in Fig. 2.7, MARLO's electronic system includes microprocessors, encoders, thermistors, limit switches, motors and DC/DC converters. The specific sensors available on the robot depend on whether it is setup for planar or 3D operation.

In planar mode, it is attached to a boom to constrain the roll and yaw orientations. The boom, which is attached to the ground support structure on one side and to the torso on the other side, confines the walking motion of the robot to its sagittal plane. The boom's circular motion due to the robot's walking around the ground support structure is measured with the Z-boom-encoder (yaw), and the boom's lateral motion due to the robot's motions in the frontal plane of walking is measured with the X-boom-encoder (roll). The "pitch-boom-encoder," which is mounted on the boom and coupled with the motions of the torso in the sagittal plane of the robot using pulleys and timing belts, measures the pitch angle of the torso with respect to the boom. In 3D mode (meaning the robot is not supported by the boom mechanism), the roll, pitch and yaw angles of the torso are measured with an "inertial measurement unit" (IMU) attached to the torso.

Absolute encoders mounted on the harmonic drives measure the angles of the two top-most links of the parallelogram as well as the two fiberglass leaf springs, all relative to the robot's torso. Incremental encoders mounted on the frontal-plane hip-actuation motors measure the hip angles with respect to the torso; an absolute encoder in the hip actuators is used for calibration purposes. The limit switches, together with "E-stop" switches (emergency stop) mounted on several points on the harmonic drives and hip motor actuators, act as a safety mechanism by killing the robot's power when the actuators reach the limits of their work-space or when an emergency situation arises. The thermistors measure the internal temperature of certain electronic components of the robot, such as the amplifiers and motors.

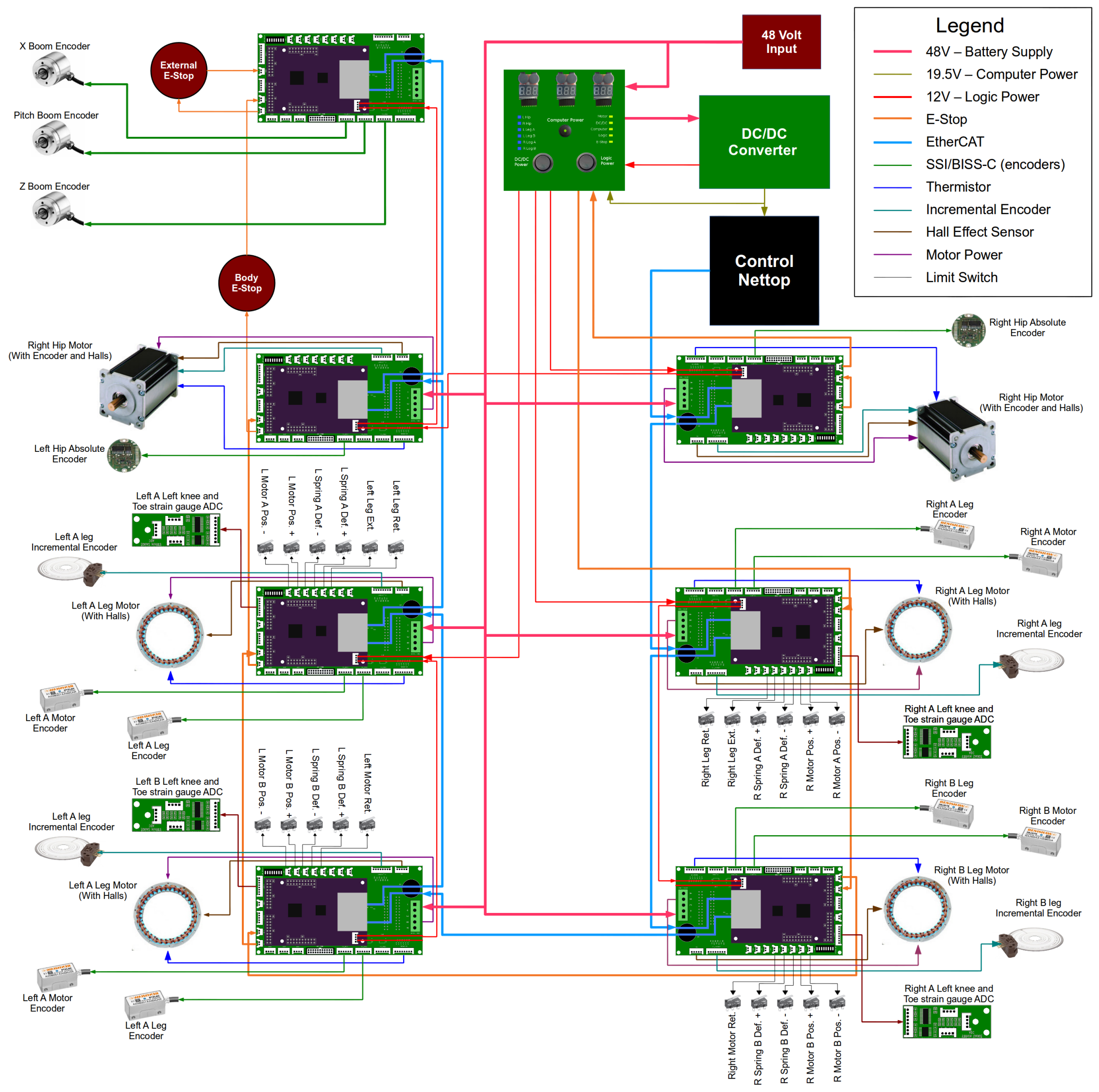


Figure 2.7: MARLO’s electronic schematic.

Seven special purpose PCB boards, called medulla boards (shown in dark gray in Fig. 2.7), are used on the robot to connect and route sensing and actuation signals. Each board is composed of local computing units and an EtherCat chip. The boards are associated to specific parts of the robot (e.g., left leg or right hip), and receive signals from encoders, IMU, thermistors, and limit switches and send command signals to the power amplifiers. These medulla boards are interconnected using a daisy-chained EtherCAT network and allow MARLO’s electronics to be extended in a modular fashion.

2.1.5 Nominal parameters

Nominal values for the mass of various elements of the robots, inertia tensors, position of the CoM and the dimensions of the robot's components (e.g., torso, hip) were obtained numerically from the cad file. Parameters for the leg and hip actuator motor torque constants, the harmonic drive and hip actuator friction, the harmonic drive and hip actuator viscous damping, are based on the manufacturer data sheets. The leg motors' torque constants and rotor inertias were also estimated through system identification.[1] The spring stiffness and damping ratios were also estimated through system identification.

2.1.6 Concluding Remarks

MARLO's design philosophy reflects the motivations of energy efficiency and a natural walking pattern. The legs are composed of a carbon-fiber-based parallelogram mechanism actuated through leaf springs with powerful non-back-drivable harmonic drives in the sagittal plane and through back-drivable gearing in the frontal plane. The robot is designed to be capable of efficient, fast and robust walking patterns in natural environments. The mass distribution of MARLO's components and the design of the SEA mechanisms were done so that the equivalent dynamic model of the robot resembles a SLIP model.

[1]Courtesy of B. Buss and J. Grizzle.

CHAPTER III

Dynamic Model of Walking

This chapter develops four mathematical models for the study of walking of the bipedal robot MARLO. The dynamic models cover both 3D and planar modes of operation when the ground model is either rigid or compliant. In general, the rigid ground model is mostly used for gait design purposes and the compliant ground model is used for feedback evaluation purposes. The chapter begins with the modeling hypotheses for the robot itself, the gait, and the ground impacts. Next, the unconstrained dynamic model (i.e., no contact with the environment) is developed using the method of Lagrange. Then the constrained dynamics models, both 3D and planar, are derived using the method of Lagrange.

3.1 Modeling Hypotheses

Walking consists of alternating phases of single support and double support. During the single support phase, one leg is in contact with the ground, whereas in the double support phase, both legs are in contact with the ground. In single support, the leg in contact with the ground is called the stance leg and the non-contacting leg is the swing leg. In double support, the ambiguity concerning the swing leg is resolved by considering the retracting leg as the swing leg during the double support phase.

3.1.1 Gait Hypotheses for Walking on Rigid Ground

This subsection explains the modeling assumptions that are used for the robot walking over rigid ground; the assumptions for a compliant ground model are found in [104]. Conditions on the controller will ensure the following assumptions are met when the robot is in motion. During the single support phase, the stance leg acts as a pivot. The normal component of the ground reaction force at the pivot point is positive, where "Coulomb's friction law" holds, ensuring that no slippage of the stance point occurs. The double support phase is instantaneous (see below). In steady state motion, the walking gait is symmetric with respect to the two legs: with each step, the swing leg starts from strictly behind the stance leg and lands strictly in front of the stance leg. The robot's CoM position monotonically increases along the direction of walking. Unless otherwise mentioned, the gait takes place on level ground.

3.1.2 Planar Robot with Point Feet Hypotheses

For the planar dynamic model with trivial feet, the robot's motion is constrained to the sagittal plane of walking. The robot is composed of 9 rigid links connected by 8 revolute joints to form an open kinematic chain. The joints are rigid and frictionless. Each link has a distributed mass. The legs are symmetric and are connected to a common point called hip. Not all revolute joints are independently actuated, [1] and the point of contact between the end of the stance leg and the ground is unactuated.

3.1.3 Tehter-free Robot with Point Feet Hypotheses

For the tether-free dynamic model with point feet it is assumed that the robot is composed of 11 rigid links connected by 10 revolute joints. The joints are rigid and frictionless, forming an open kinematic chain; each link has a distributed mass; the robot's motions are considered both in the frontal plane and the sagittal plane; legs are symmetric and are con-

[1] The parallelogram mechanism has unactuated revolute joints.

nected to the hips. The hips are connected at a common point to the torso, not all revolute joints are independently actuated, and the point of contact between the end of the stance leg and the ground is unactuated.

3.1.4 Impact Hypotheses

Impact results when the swing leg contacts the ground. The impact is instantaneous and causes neither "rebound" nor "slipping." At the moment of impact, the stance leg lifts off the ground without any interaction. External forces acting during impact can be considered as impulses, while internal torques and forces cannot generate impulses. The impact impulse force may cause discontinuities in velocity, but cannot cause discontinuities in angles.

3.2 Modeling

3.2.1 MARLO's Unconstrained Dynamic Model

The Lagrangian for a multi-link rigid body robot with 1 DoF revolute joints such as MARLO is a functional acting on points in state space $x_f = (q, \dot{q}) \in \mathcal{X}_f = \mathbb{T}\mathcal{Q}_f$, where $\mathcal{Q}_f$ is the configuration space, an open subset of $\mathbb{T}^{10} \times \mathbb{R}^3$ and $\mathbb{T}^{10} = \mathcal{S}^1 \times \ldots \times \mathcal{S}^1$, and $\mathcal{S}^1$ is the unit circle. Coordinates are defined for the robot in general position, that is, when neither leg is in contact with the ground. Denote the generalized coordinate used in the unconstrained model by $\mathbf{q}_f = \{q_1, \ldots, q_i, \ldots, q_{16}\}$. The members of the generalized coordinate set follow.

Let a world frame **n** be defined and attach a Cartesian coordinate frame $\mathbf{b}_1 = (x_T, y_T, z_T)$ to the CoM of the torso, oriented so that when the torso is upright, the z-axis points upward and the y-axis points forward (Fig. 3.2). Euler angles are used to parametrize the orientation of the torso with respect to the world frame **n**. These angles are denoted by q_z, q_x, and q_y and are called yaw, pitch, and roll, respectively (Fig. 3.1). The yaw coordinate of the torso is also called heading. A Cartesian coordinate frame $\mathbf{b}_2$ is attached to the CoM of the right hip such that when the hip is perpendicular to the torso, the z-axis points upward and, the y-axis points forward (Fig. 3.2). A similar Cartesian coordinate is attached to the CoM of the left hip and is labeled $\mathbf{b}_3$. The angles of the right and left hips relative to the torso are denoted by q_{3R} and q_{3L}, respectively, as shown in Fig. 3.1. Because the four links in each leg form a parallelogram, only two coordinates are needed to parametrize them. The angles of the top two links relative to the hip are denoted by q_{1R} and q_{2R} for the right leg, and q_{1L} and q_{2L} for the left leg (Fig 3.1). All relative angles explained above are positive in the direction that respects the right-hand rule with one exception – the left hip joint angle. Two Cartesian coordinates $\mathbf{b}_4$ and $\mathbf{b}_5$ are attached to the CoM of the two topmost links of the right parallelogram mechanism such that the z-axis is along the link pointing downward and the y-axis pointing forward (Fig. 3.2). Likewise, two Cartesian coordinates $\mathbf{b}_6$ and $\mathbf{b}_7$ are attached to the CoM of the two topmost links of left parallelogram mechanism, Fig. 3.2.

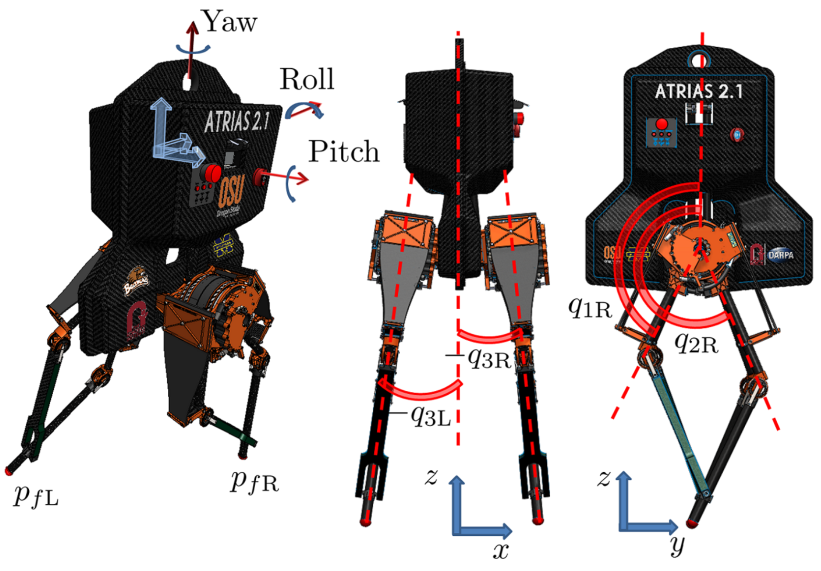


Figure 3.1: Angles used to uniquely determine the configuration of MARLO.

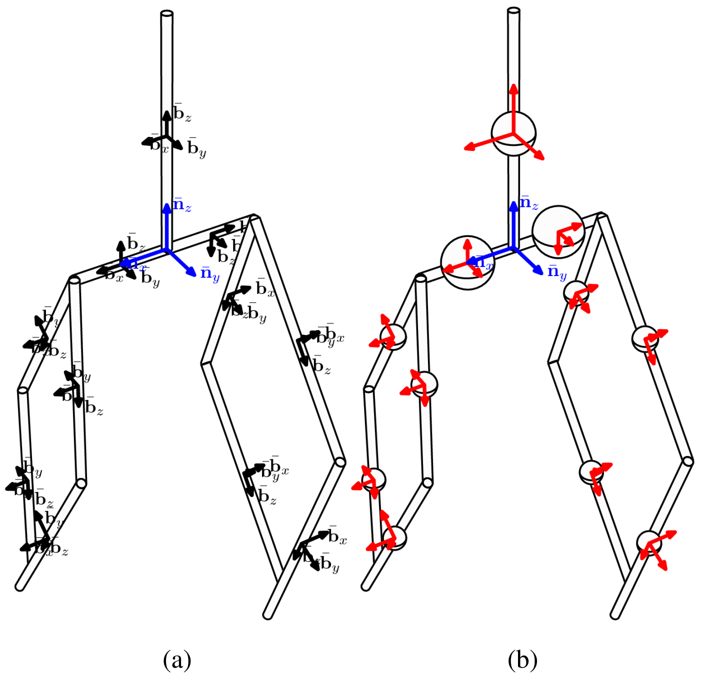


Figure 3.2: **(a)** Coordinate frames: blue is world coordinate and blacks are body coordinates. **(b)** Moment of inertia tensor associated with each link is computed in the local coordinate frame (red) and is then transformed into the world frame (blue).

Turning to the actuators, because the motors at the hips are connected to the body through a fixed gear ratio, the angles of the rotors relative to the torso are uniquely determined from q_{3R} and q_{3L}. On the other hand, the motors driving the legs are connected through springs, and hence additional coordinates are needed. For each motor, the angle of the output shaft of the harmonic drive is denoted by q_{gr}. These angles are used instead of the angles of the rotors of the motors because q_{gr1R}, q_{gr2R}, q_{gr1L}, and q_{gr2L} are in the same coordinate frame as the respective link angles of the legs. To complete the set of generalized coordinates, the position of an arbitrary point on the robot, $\mathbf{p}_e = (p_e^h, p_e^v, p_e^l)$, is added to the generalized coordinate set, $\mathbf{q}_e = (\mathbf{p}_e, q_z, q_y, q_x, q_{1R}, q_{2R}, q_{1L}, q_{2L}, q_{gr1R}, q_{gr2R}, q_{3R}, q_{gr1L}, q_{gr2L}, q_{3L})$. This completes the coordinates for the robot; see Fig. 3.1.

The Lagrangian for the unconstrained system $\mathcal{L}_f : \mathcal{X}_f \mapsto \mathbb{R}$ is defined as the difference of total kinetic energy and total potential energy; therefore,

$$\mathcal{L}_f := \mathcal{K}_f - \mathcal{V}_f \tag{3.2.1}$$

where $\mathcal{K}_f : \mathcal{X}_f \mapsto \mathbb{R}$ and $\mathcal{V}_f : \mathcal{Q}_f \mapsto \mathbb{R}$ are kinetic and potential energy, respectively. Using Hamilton's principle, the equation of motion is obtained from the Lagrangian,

$$\frac{d}{dt}\frac{\partial \mathcal{L}_f}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}_f}{\partial q_i} = f_i \tag{3.2.2}$$

where f_i represents generalized forces and generalized torques corresponding to the i-th generalized coordinate. The first element that must be calculated in order to compute (3.2.1) is the total kinetic energy of the system. Using the Cartesian coordinated frames defined previously $\{\mathbf{n}, \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{11}\}$, a set of rotation matrices $\mathcal{R}_{\mathbf{n},\mathbf{b}_1}$, $\mathcal{R}_{\mathbf{b}_1,\mathbf{b}_2}$, $\mathcal{R}_{\mathbf{b}_1,\mathbf{b}_3}$, $\mathcal{R}_{\mathbf{b}_2,\mathbf{b}_4}$, $\mathcal{R}_{\mathbf{b}_2,\mathbf{b}_5}$, $\mathcal{R}_{\mathbf{b}_3,\mathbf{b}_6}$, $\mathcal{R}_{\mathbf{b}_3,\mathbf{b}_7} \in \mathcal{SO}3$ are defined. $\mathcal{R}_{\mathbf{b}_i,\mathbf{b}_j}$ takes the vectors expressed in coordinate frame $\mathbf{b}_j$ and expresses the vector in the coordinate frame $\mathbf{b}_i$. Now that the transformation from one Cartesian coordinate to another Cartesian coordinate is well defined, let the kinetic energy of the robot's i-th link be

$$\mathcal{K}_{f,i} = \frac{1}{2}\int\limits_{V_i} \rho(\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i}) \left\|\dot{\hat{\mathbf{r}}}^{\epsilon}_{\mathbf{n}}\right\|^2 dV \tag{3.2.3}$$

where $V_i \in \mathbb{R}^3$ is the region occupied by the i-th link, $\rho(\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i})$ is the density of the i-th link at the point $\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i} \in V_i$, $\dot{\hat{\mathbf{r}}}^{\epsilon}_{\mathbf{n}}$ is the velocity of an element of i-th link with respect to the world frame $\mathbf{n}$, and $\|\|^2$ is the two-norm. Considering the rotation matrices $\mathcal{R}_{\mathbf{n},\mathbf{b}_i} \in \mathcal{SO}3$, it is possible to re-write $\hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}}$ as

$$\hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} = \hat{\mathbf{r}}_{\mathbf{b}_i \mapsto \mathbf{n}} + \mathcal{R}_{\mathbf{n},\mathbf{b}_i}\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i} \tag{3.2.4}$$

where $\hat{\mathbf{r}}_{\mathbf{b}_i \mapsto \mathbf{n}}$ is the vector starting from the origin of $\mathbf{n}$ and ending at the origin of $\mathbf{b}_i$. The time derivative of $\dot{\hat{\mathbf{r}}}^{\epsilon}_{\mathbf{n}}$ is given as,

$$\dot{\hat{\mathbf{r}}}^{\epsilon}_{\mathbf{n}} = \dot{\hat{\mathbf{r}}}_{\mathbf{b}_i \mapsto \mathbf{n}} + \dot{\mathcal{R}}_{\mathbf{n},\mathbf{b}_i} \hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i} \tag{3.2.5}$$

Now that $\hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}}$ and $\dot{\hat{\mathbf{r}}}^{\epsilon}_{\mathbf{n}}$ are known with respect to the world frame, substituting these terms in (3.2.3) gives,

$$\begin{aligned}\mathcal{K}_{f,i} =& \frac{1}{2} \int\limits_{V_i} \rho(\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i}) \left\| \dot{\hat{\mathbf{r}}}_{\mathbf{b}_i \mapsto \mathbf{n}} + \dot{\mathcal{R}}_{\mathbf{n},\mathbf{b}_i} \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} \right\|^2 dV \\ =& \frac{1}{2} \int\limits_{V_i} \rho(\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i}) \left(\left\| \dot{\hat{\mathbf{r}}}_{\mathbf{b}_i \mapsto \mathbf{n}} \right\|^2 + \left\| \dot{\mathcal{R}}_{\mathbf{n},\mathbf{b}_i} \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} \right\|^2 + 2 \left(\dot{\hat{\mathbf{r}}}_{\mathbf{b}_i \mapsto \mathbf{n}} \right) \left(\dot{\mathcal{R}}_{\mathbf{n},\mathbf{b}_i} \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} \right) \right) dV \\ =& \frac{1}{2} \int\limits_{V_i} \rho(\hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}}) \left(\left\| \dot{\hat{\mathbf{r}}}_{\mathbf{b}_i \mapsto \mathbf{n}} \right\|^2 + \left\| \mathcal{R}_{\mathbf{n},\mathbf{b}_i} [\Omega]_{\mathbf{b}_i} \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} \right\|^2 + 2 \left(\dot{\hat{\mathbf{r}}}_{\mathbf{b}_i \mapsto \mathbf{n}} \right) \left(\mathcal{R}_{\mathbf{n},\mathbf{b}_i} [\Omega]_{\mathbf{b}_i} \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} \right) \right) dV\end{aligned} \tag{3.2.6}$$

In (3.2.6), using the fact that $(\mathcal{R}_{\mathbf{n},\mathbf{b}_i})^{-1} \dot{\mathcal{R}}_{\mathbf{n},\mathbf{b}_i}$ is skew-symmetric, $\dot{\mathcal{R}}_{\mathbf{n},\mathbf{b}_i}$ can be written as follows,

$$\begin{aligned}\dot{\mathcal{R}}_{\mathbf{n},\mathbf{b}_i} =& \mathcal{R}_{\mathbf{n},\mathbf{b}_i} (\mathcal{R}_{\mathbf{n},\mathbf{b}_i})^{-1} \dot{\mathcal{R}}_{\mathbf{n},\mathbf{b}_i} \\ =& \mathcal{R}_{\mathbf{n},\mathbf{b}_i} [\Omega]_{\mathbf{b}_i}\end{aligned} \tag{3.2.7}$$

where $[\Omega]^{\vee}_{\mathbf{b}_i} = \bar{\Omega}_{\mathbf{b}_i} \in \mathbb{R}^3$ and the $\vee$ operator changes a skew-symmetric matrix into a vector. In (3.2.6), the kinetic energy associated with the i-th link is composed of three terms. The first term is due to the translational energy of the link,

$$K^{Translation}_{f,i} = \frac{1}{2} \int\limits_{V_i} \rho(\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i}) \left(\left\| \dot{\hat{\mathbf{r}}}_{\mathbf{b}_i \mapsto \mathbf{n}} + \dot{\mathcal{R}}_{\mathbf{n},\mathbf{b}_i} \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} \right\|^2 \right) dV \tag{3.2.8}$$

The second term is the rotational kinetic energy of the link,

$$\begin{aligned}
K^{Rotation}_{f,i} =& \frac{1}{2} \int\limits_{V_i} \rho(\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i}) \left(\left\| \mathcal{R}_{\mathbf{n},\mathbf{b}_i} [\Omega]_{\mathbf{b}_i} \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} \right\|^2 \right) dV \\
=& \frac{1}{2} \int\limits_{V_i} \rho(\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i}) \left(\mathcal{R}_{\mathbf{n},\mathbf{b}_i} [\Omega]_{\mathbf{b}_i} \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} \right)' \left(\mathcal{R}_{\mathbf{n},\mathbf{b}_i} [\Omega]_{\mathbf{b}_i} \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} \right) dV \\
=& \frac{1}{2} \int\limits_{V_i} \rho(\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i}) (\hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}})' \left([\Omega]_{\mathbf{b}_i} \right)' (\mathcal{R}_{\mathbf{n},\mathbf{b}_i})' \mathcal{R}_{\mathbf{n},\mathbf{b}_i} [\Omega]_{\mathbf{b}_i} \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} dV \\
=& \frac{1}{2} \int\limits_{V_i} \rho(\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i}) (\hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}})' \left([\Omega]_{\mathbf{b}_i} \right)' [\Omega]_{\mathbf{b}_i} \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} dV \\
=& \frac{1}{2} \int\limits_{V_i} \rho(\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i}) \left([\Omega]_{\mathbf{b}_i} \right)' (\hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}})' \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} [\Omega]_{\mathbf{b}_i} dV \\
=& \frac{1}{2} \left([\Omega]_{\mathbf{b}_i} \right)' \left(\int\limits_{V_i} \rho(\hat{\mathbf{r}}^{\epsilon}_{\mathbf{b}_i}) (\hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}})' \hat{\mathbf{r}}^{\epsilon}_{\mathbf{n}} dV \right) [\Omega]_{\mathbf{b}_i} \\
=& \frac{1}{2} \left([\Omega]_{\mathbf{b}_i} \right)' (\mathcal{J}_{i_{\mathbf{n}}}) [\Omega]_{\mathbf{b}_i}
\end{aligned} \tag{3.2.9}$$

where $\mathcal{J}_{i_{\mathbf{n}}}$ is the moment of inertia corresponding to i-th link with respect to the world frame **n**. Since the body coordinate frames are located at the CoM of i-th link, the third term in the $\mathcal{K}_{f,i}$ is zero. Therefore,

$$\begin{aligned}
\mathcal{K}_{f,i} =& K^{Translation}_{f,i} + K^{Rotation}_{f,i} \\
=& \frac{1}{2} m_i \left\| \dot{\hat{\mathbf{r}}}_{\mathbf{b}_i \mapsto \mathbf{n}} \right\|^2 + \frac{1}{2} [\Omega]'_{\mathbf{b}_i} \mathcal{J}_{i_{\mathbf{n}}} [\Omega]_{\mathbf{b}_i}
\end{aligned} \tag{3.2.10}$$

Now the terms $\dot{\hat{\mathbf{r}}}_{\mathbf{b}_i \mapsto \mathbf{n}}$ and $[\Omega]_{\mathbf{b}_i}$ are expressed in the configuration variables

$$\dot{\hat{\mathbf{r}}}_{\mathbf{b}_i \mapsto \mathbf{n}} = \frac{\partial \hat{\mathbf{r}}_{\mathbf{b}_i \mapsto \mathbf{n}}}{\partial q} \dot{q} := J_{\mathbf{b}_i} \dot{q} \tag{3.2.11}$$

where $J_{\mathbf{b}_i}$ is the Jacobian of the i-th body frame position $\mathbf{b}_i$ with respect to configuration vector q. $[\Omega]_{\mathbf{b}_i}$ is written as

$$
\begin{aligned}
[\Omega]_{\mathbf{b}_i} =& \sum_{i=1}^{N} (\mathcal{R}_{\mathbf{n},\mathbf{b}_i})^{-1} \frac{\partial \mathcal{R}_{\mathbf{n},\mathbf{b}_i}}{\partial q_i} \dot{q}_i \\
=& \left[\left((\mathcal{R}_{\mathbf{n},\mathbf{b}_i})^{-1} \frac{\partial \mathcal{R}_{\mathbf{n},\mathbf{b}_i}}{\partial q_1} \right) \dots \left((\mathcal{R}_{\mathbf{n},\mathbf{b}_i})^{-1} \frac{\partial \mathcal{R}_{\mathbf{n},\mathbf{b}_i}}{\partial q_N} \right) \right] \dot{q} \\
=& \mathcal{O}_i(q) \dot{q}
\end{aligned}
\tag{3.2.12}
$$

Therefore, $\mathcal{K}_{f,i}$ is,

$$\mathcal{K}_{f,i} = \frac{1}{2} \dot{q}' D_{f,i}(q) \dot{q} \tag{3.2.13}$$

where $D_{f,i}(q)$ is given by

$$D_{f,i} = \frac{1}{2} m_i J'_{\mathbf{b}_i}(q) J_{\mathbf{b}_i}(q) + \frac{1}{2} (\mathcal{O}_i)'(q) \mathcal{J}_{i_{\mathbf{n}}}(q) \mathcal{O}_i(q) \tag{3.2.14}$$

$D_{f,i}$ is a symmetric positive semi-definite matrix. The system's total kinetic energy is written as follows:

$$\mathcal{K}_f = \frac{1}{2} \sum_{i=1}^{M} \dot{q}' D_{f,i}(q) \dot{q} \tag{3.2.15}$$

The system's potential energy is composed of the potential energy of each link due to the gravitational field $V_{f,i}^{Gravity}$ and is written as

$$\mathcal{V}_f = \sum_{i=1}^{M} g m_i p_{cm,i}^{v}(q) \tag{3.2.16}$$

where g is the gravitational acceleration and $p_{cm,i}^{v}(q)$ the vertical position of the CoM of i-th link.

Now that both the total kinetic energy and total potential energy of the system are known, expanding (3.2.2) results in

$$\frac{d}{dt} \frac{\partial \mathcal{K}_f}{\partial \dot{q}_i} - \frac{\partial \mathcal{K}_f}{\partial q_i} - \frac{\partial \mathcal{V}_f}{\partial q_i} = f_i \tag{3.2.17}$$

Considering (3.2.15), the first term in (3.2.17) is re-written as follows:

$$\frac{d}{dt}\frac{\partial \mathcal{K}_f}{\partial \dot{q}_i} = \sum_{j=1}^{N} D_{i,j}(q)\ddot{q}_j + \sum_{j,k=1}^{N} \frac{\partial D_{i,j}(q)}{\partial q_k}\dot{q}_j\dot{q}_k \tag{3.2.18}$$

and

$$\frac{\partial \mathcal{K}_f}{\partial q_i} = \frac{1}{2}\sum_{j,k=1}^{N} \frac{\partial D_{k,j}(q)}{\partial q_i}\dot{q}_j\dot{q}_k \tag{3.2.19}$$

Substituting (3.2.18) and (3.2.19) in (3.2.17) results in:

$$\sum_{j=1}^{N} D_{i,j}(q)\ddot{q}_j + \sum_{j,k=1}^{N} \frac{\partial D_{i,j}(q)}{\partial q_k}\dot{q}_j\dot{q}_k - \frac{1}{2}\sum_{j,k=1}^{N} \frac{\partial D_{k,j}(q)}{\partial q_i}\dot{q}_j\dot{q}_k + \frac{\partial \mathcal{V}_f}{\partial q_i} = f_i \tag{3.2.20}$$

The second and third terms in (3.2.20) are components of the Coriolis matrix $\mathcal{C}_{f,i} \in \mathbb{R}^{N\times N}$, which is written as

$$\sum_{j,k=1}^{N} \frac{\partial D_{i,j}(q)}{\partial q_k}\dot{q}_j\dot{q}_k - \frac{1}{2}\sum_{j,k=1}^{N} \frac{\partial D_{k,j}(q)}{\partial q_i}\dot{q}_j\dot{q}_k = \sum_{j=1}^{N} \mathcal{C}_{f,i,j}\dot{q}_j \tag{3.2.21}$$

The effect of the gravitational field represented as $\mathcal{G}_{f,i} \in \mathbb{R}^N$ enters into the model as follows:

$$\mathcal{G}_{f,i} = \frac{\partial \mathcal{V}_f}{\partial q_i} \tag{3.2.22}$$

Re-writing (3.2.20) after substituting (3.2.21) and (3.2.22) in (3.2.20), the robot's equations of motion are given by

$$\mathcal{D}_f(q_f)\ddot{q_f} + \mathcal{C}_f(q_f,\dot{q}_f)\dot{q}_f + \mathcal{G}_f(q_f) = f(q_f,\dot{q}_f,u). \tag{3.2.23}$$

As previously mentioned, matrix $\mathcal{D}_f$ is the unconstrained mass-inertia matrix, $\mathcal{C}_f$ is the unconstrained matrix of Coriolis and centrifugal terms, $\mathcal{G}_f$ is the gravity vector for the unconstrained robot, and $f(q_f,\dot{q}_f,u)$ is the vector of generalized forces acting on the unpinned model of the robot. Using the principle of virtual work, $f(q_f,\dot{q}_f,u)$ can be written as

$$f(q_f, \dot{q}_f, u) = \mathcal{B}u + \mathcal{B}_{sp}(q_f)\tau_{sp}(q_f, \dot{q}_f), \tag{3.2.24}$$

where the matrices $\mathcal{B}$ and $\mathcal{B}_{sp}$ define how the motor torques $u = [u_{1R}, u_{2R}, u_{3R}, u_{1L}, u_{2L}, u_{3L}]'$ and the spring torques τ_{sp} enter the unconstrained model, respectively.

3.2.2 Tether-Free Model on Rigid Ground

MARLO's tether-free dynamic model is derived using the method of Lagrange when the ground is non-compliant (i.e., rigid). The model is composed of two phases: double support and single support. The single support model is formulated first and then the impact dynamics, mapping the state of the robot before impact to the states of the robot after impact, are derived. Finally, the hybrid model of walking, which combines the swing phase dynamics and impact dynamics, is presented.

3.2.2.1 Single Support Phase

When the robot is in single support, meaning one and only one leg is in contact with the ground, the coordinates are defined independent of which leg is eventually the stance leg. This is different from the approach followed in [105]. The Lagrangian for the constrained model of the robot with 1 DoF revolute joints is a functional acting on points in state space $x_f = (q, \dot{q}) \in \mathcal{X}_s = \mathbb{T}\mathcal{Q}_s$, where $\mathcal{Q}_s$ is the configuration space, an open subset of $\mathbb{T}^{10} \times \mathbb{R}^3$. Coordinates are defined for the robot when one leg is in contact with the ground. Denote the generalized coordinates used in the constrained model by $\mathbf{q}_s = \{q_1, \ldots, q_i, \ldots, q_{13}\}$. When in single support, the configuration variable $\mathbf{p}_e$ used in the unconstrained model is redundant and can be eliminated. Thus, for developing the single-support model with either the left or the right leg in stance, the configuration variables are $\mathbf{q}_s = (q_z$, q_y, q_x, q_{1R}, q_{2R}, q_{1L}, q_{2L}, q_{gr1R}, q_{gr2R}, q_{3R}, q_{gr1L}, q_{gr2L}, $q_{3L})$. Recall that the last six coordinates are independently actuated, whereas the first seven coordinates are un-actuated. Let $Q_s \subset SO(3) \times \mathbb{S}^{10}$ be a connected open subset giving the feasible set for the configuration variables.

The model and its feasible set of variables will of course depend on which leg is in stance. This will be made clear in Section 3.2.2.3 by appending a subscript "L" or "R" as appropriate. Because the mechanism in each leg forms a parallelogram, the linear and angular velocities of the two lower links are identical to the linear and angular velocities of the corresponding upper links. It follows that the Lagrangian of the model can be developed as if the robot were a pinned open kinematic chain.

To compute the Lagrangian, the kinetic energy and potential energy of each rigid body are calculated and expressed in the generalized coordinates. Summing over the rigid bodies then gives the total kinetic energy $\mathcal{K}_s$ and the total potential energy $\mathcal{V}_s$, yielding

$$\mathcal{L}_s(q_s, \dot{q}_s) := \mathcal{K}_s(q_s, \dot{q}_s) - \mathcal{V}_s(q_s). \tag{3.2.25}$$

Lagrange's equation then gives the standard robot equations

$$\mathcal{D}_s(q_s)\ddot{q}_s + \mathcal{C}_s(q_s, \dot{q}_s)\dot{q}_s + \mathcal{G}_s(q_s) = \Gamma_s, \tag{3.2.26}$$

where matrix $\mathcal{D}_s$ is the mass-inertia matrix, $\mathcal{C}_s$ is the matrix of Coriolis and centrifugal terms, $\mathcal{G}_s$ is the gravity vector, and Γ_s is the vector of generalized forces acting on the robot. Using the principle of virtual work, Γ_s can be written as

$$\Gamma_s = \mathcal{B}_s u + \mathcal{B}_{sp}(q_s)\tau_{sp}(q_s, \dot{q}_s) + \mathcal{B}_{yaw}(q_s)\tau_{yaw}(q_s, \dot{q}_s), \tag{3.2.27}$$

where the matrices[2] $\mathcal{B}_s$, $\mathcal{B}_{sp}$, and $\mathcal{B}_{yaw}$ define how the motor torques

$$u = [u_{1R}, u_{2R}, u_{3R}, u_{1L}, u_{2L}, u_{3L}]', \tag{3.2.28}$$

the spring torques τ_{sp}, and the stance-leg end yaw torque τ_{yaw} enter the model, respectively.

[2]Because of the way coordinates have been assigned, B_s is a constant matrix. Moreover, because the actuators are independent, B_s has (full) rank equal to the number of actuators, namely 6.

For $i \in \{1R, 2R, 1L, 2L\}$, the spring torque of the i-th series-elastic actuator is modeled as

$$\tau_{sp,i} = -k_{sp,i}(q_i - q_{gr,i}) - b_{sp,i}(\dot{q}_i - \dot{q}_{gr,i}), \tag{3.2.29}$$

where $k_{sp,i}$ denotes spring stiffness and $b_{sp,i}$ is a damping coefficient.

The primary joint friction is due to the harmonic drives used in the series-elastic actuators and is ignored in the model at the current time. References [106, 107] provide nonlinear models of power-loss at the harmonic drives and will be incorporated in future work when sufficient experimental data is available from the robot. The stance-leg end yaw torque is also modeled as viscous friction

$$\tau_{yaw} = -b_{yaw}\omega_{shin}(q_s, \dot{q}_s), \tag{3.2.30}$$

where ω_{shin} is the vertical component of the angular velocity about the stance shin and b_{yaw} is a constant.

Setting $x = (q_s; \dot{q}_s) \in \mathcal{TQ}_s$, the model in state-variable form is

$$\dot{x} = \begin{bmatrix} \dot{q}_s \\ \mathcal{D}_s^{-1}(q_s)(-\mathcal{H}_s(q_s, \dot{q}_s) + \mathcal{B}_s u) \end{bmatrix}, \tag{3.2.31}$$

where

$$\begin{aligned} \mathcal{H}_s(q_s, \dot{q}_s) =& \mathcal{C}_s(q_s, \dot{q}_s)\dot{q}_s + \mathcal{G}_s(q_s) - \mathcal{B}_{sp}(q_s)\tau_{sp}(q_s, \dot{q}_s) - \\ & \mathcal{B}_{fric}(q_s)\tau_{fric}(q_s, \dot{q}_s) - \mathcal{B}_{yaw}(q_s)\tau_{yaw}(q_s, \dot{q}_s). \end{aligned} \tag{3.2.32}$$

Equation (3.2.31) immediately leads to the state variable model

$$\dot{x} = \mathbf{f}_s(x) + \mathbf{g}_s(x)u. \tag{3.2.33}$$

3.2.2.2 Impact model

An impact occurs when the end of the swing leg contacts the ground. The impact dynamics is presented in the form of a map that takes the states of the robots before impact $x_s^- \in \mathcal{Q}_s$ and gives the states after the impact $x_s^+ \in \mathcal{Q}_s$. Let $p_v : Q_s \mapsto \mathbb{R}$ denote the vertical height of the swing leg above the ground so that the impact surface is

$$\mathcal{S} = \{x \in \mathcal{T}\mathcal{Q}_s \mid p_v(q_s) = 0, p_v(q_s) > 0\}. \tag{3.2.34}$$

The impact is modeled as a contact of two rigid bodies, using the methodology of [108]. (See [109] for the details) Consequently, the impact is instantaneous, the generalized configuration variables are constant across the impact, while the generalized velocities undergo a jump. Consider the unconstrained model of the robot again:

$$\mathcal{D}_f(q_f)\ddot{q}_f + \mathcal{C}_f(q_f, \dot{q}_f)\dot{q}_f + \mathcal{G}_f(q_f) = f(q_f, \dot{q}_f, u) + \delta\mathcal{F}_{imp}, \tag{3.2.35}$$

where $\delta\mathcal{F}_{imp} \in \mathbb{R}^4$ denotes the instantaneous impact force. The impact hypothesis mentioned previously (no-slipping, no-rebound and no-yaw-motion), yield the following equations

$$\begin{bmatrix} \dot{q}_f^+ \\ \mathcal{F}_v \end{bmatrix} = \begin{bmatrix} \mathcal{D}_f(q_f^-) & \mathcal{E}_v \\ -\mathcal{E}_v & O^{4\times4} \end{bmatrix}^{-1} \begin{bmatrix} \mathcal{D}_f(q_f^-)\dot{q}_f^- \\ 0^{4\times1} \end{bmatrix} \tag{3.2.36}$$

where $\mathcal{E}_v \in \mathbb{R}^4$

$$\mathcal{E}_v = \begin{bmatrix} \frac{\partial p_v(q_f^-)}{\partial q_f} \\ \frac{\partial \omega_{shin}(q_f^-, \dot{q}_f)}{\partial q_f} \end{bmatrix}. \tag{3.2.37}$$

The first thirteen entries of $\dot{q}_f^+$ should be used to initialize the single support phase after the impact. Moreover, the configuration angles before impact q_s^- are equal to the configuration angles after the impact q_s^+ since there is no rebound or jump in the angles.

This is written in the form of a map that results in the states of the robot after impact,

$$x_s^+ = \Delta(x_s^-), \tag{3.2.38}$$

where x_s^- is the value of the state just before the impact and x_s^+ is its value just after the impact.

3.2.2.3 Hybrid model

Combining the swing phase models and the impact models for the left and right legs results in the hybrid model

$$\begin{aligned} \Sigma_L : &\begin{cases} \dot{x} = \mathbf{f}_{s,L}(x) + \mathbf{g}_{s,L}(x)u, & x^- \notin \mathcal{S}_L \\ x^+ = \Delta_{L \to R}(x^-), & x^- \in \mathcal{S}_L \end{cases} \\ \Sigma_R : &\begin{cases} \dot{x} = \mathbf{f}_{s,R}(x) + \mathbf{g}_{s,R}(x)u, & x^- \notin \mathcal{S}_R \\ x^+ = \Delta_{R \to L}(x^-), & x^- \in \mathcal{S}_R. \end{cases} \end{aligned} \tag{3.2.39}$$

In the hybrid model, the dynamics evolve according to (3.2.33) until the swing leg impacts the ground. The impact map given by (3.2.38) is inactive until the state of the robot reaches the switching surface $\mathcal{S}$, at which point the impact map becomes active and results in jump (or discontinuity) in the velocity states. (3.2.39) gives the hybrid nonlinear system for 3D bipedal walking.

3.2.3 Tether-Free Model on Compliant Ground

The tether-free model with point feet walking over a compliant ground model is obtained from the unconstrained model of the robot introduced earlier with ground contact forces applied to the leg ends. Indeed, using the principle of virtual work and the unconstrained model results in

$$\mathcal{D}_f(q_f)\ddot{q}_f + \mathcal{C}_f(q_f, \dot{q}_f)\dot{q}_f + \mathcal{G}_f(q_f) = f(q_f, \dot{q}_f, u) + J'(q_f).\mathcal{F} \tag{3.2.40}$$

where $J'(q_f) \in \mathbb{R}^{6\times 16}$ is a Jacobian matrix of the robot's legs end-points,

$$J'(q_f) = \frac{\partial}{\partial q_f}\begin{bmatrix} x_R \\ y_R \\ z_R \\ x_L \\ y_L \\ z_L \end{bmatrix} \tag{3.2.41}$$

and $\mathcal{F} \in \mathbb{R}^6$ consists of tangential and normal forces acting at the ends of the robot's legs.

$$\mathcal{F} = \begin{bmatrix} \mathcal{F}_{R,x} \\ \mathcal{F}_{R,y} \\ \mathcal{F}_{R,z} \\ \mathcal{F}_{L,x} \\ \mathcal{F}_{L,y} \\ \mathcal{F}_{L,z} \end{bmatrix} \tag{3.2.42}$$

The tangential and normal forces $\mathcal{F}$ are modeled as in [104] with the following equations:

$$\begin{aligned} \mathcal{F}_x &= \mu_x(d, v).\left|\mathcal{F}_z\right| \\ \mathcal{F}_y &= \mu_y(d, v).\left|\mathcal{F}_z\right| \\ \mathcal{F}_z &= -\lambda_v^a.\left|z_G\right|^n.\dot{z}_G + k_\nu.\left|z_G\right|^n \end{aligned} \tag{3.2.43}$$

with

$$\begin{aligned} \dot{d} &= \nu - |\nu| . \frac{\sigma_{h_0}}{\alpha_{h_0}} . d \\ \mu_{x,y}(d,\nu) &= \sigma_{h_0} . d + \sigma_{h_1} . \dot{d} + \alpha_{h_2} . \nu \end{aligned} \tag{3.2.44}$$

The normal forces $\mathcal{F}_z$ have been modeled as a vertical nonlinear spring damper, where z_G is the penetration of the link into the ground, λ_ν^a is the damping coefficient of the vertical damper, k_ν is the stiffness of the vertical spring, and $n = 1.5$ is a coefficient characterizing the form of the surface in contact. The tangential forces $\mathcal{F}_{x,y}$ are in the form of a friction model with a non-constant friction coefficient. Specifically, the LuGre friction model has been used to evaluate the friction coefficients μ. The model supposes that the interaction between the leg end and the walking surface resembles a set of "bristles." If the average tangential force is sufficient, the springs and dampers used to model the bristle-like interaction will deflect; otherwise, the foot will slide. In the above, d is the deflection where ν is the relative velocity of the contacting surface; σ_{h_0} is the stiffness of the horizontal spring; α_{h_0} is the coefficient of static friction; σ_{h_1} is the damping coefficient of the horizontal damper; and α_{h_2} is the coefficient of viscous friction.

3.2.4 Planar Model on Rigid Ground

A sagittal plane model is obtained from the unconstrained model by imposing six holonomic constraints and setting the width of the hips to zero. The position and velocity of the stance leg-end point $p_{st} = (x; y; z) \in \mathbb{R}^3$ are set to zero. The yaw and roll coordinates of the torso, q_z and q_y, and their derivatives are set to zero, thereby constraining the torso to the sagittal plane. The hip coordinates q_{3R} and q_{3L} and their derivatives are also set to zero, so that the hip axis is perpendicular to the sagittal plane. The Lagrangian of the planar model is then the Lagrangian of the unconstrained model restricted to the surface:

$$\begin{aligned} \mathcal{TQ}_{s,2D} = \{(q_f; \dot{q}_f) \in \mathcal{TQ}_f \mid\ & p_{st} = 0, q_z = 0, q_y = 0, q_{3R} = 0, q_{3L} = 0, \\ & \dot{p}_{st} = 0, \dot{q}_z = 0, \dot{q}_y = 0, \dot{q}_{3R} = 0, \dot{q}_{3L} = 0\}. \end{aligned} \tag{3.2.45}$$

The control torques associated with q_{3R} and q_{3L} are removed, leaving four actuators. The mass of these actuators is retained in the model so that solutions of the tether-free and planar models can be compared.

3.2.4.1 Single Support Phase

The Lagrange equations for the constrained planar model at a point $q_{_{s,2D}} \in \mathcal{Q}_{_{s,2D}}$, $\dot{q}_{_{s,2D}} \in \mathcal{T}\mathcal{Q}_{_{s,2D}}$ is written as

$$\mathcal{D}_{_{s,2D}}(q_{_{s,2D}})\ddot{q}_{_{s,2D}} + \mathcal{C}_{_{s,2D}}(q_{_{s,2D}}, \dot{q}_{_{s,2D}})\dot{q}_{_{s,2D}} + \mathcal{G}_{_{s,2D}}(q_{_{s,2D}}) = \Gamma_{_{s,2D}}, \tag{3.2.46}$$

where matrix $\mathcal{D}_{_{s,2D}}$ is the mass-inertia matrix, $\mathcal{C}_{_{s,2D}}$ is the matrix of Coriolis and centrifugal terms, $\mathcal{G}_{_{s,2D}}$ is the gravity vector, and $\Gamma_{_{s,2D}}$ is the vector of generalized forces acting on the planar robot. Using the principle of virtual work, $\Gamma_{_{s,2D}}$ is written as

$$\Gamma_{_{s,2D}} = \mathcal{B}_{_{s,2D}}u + \mathcal{B}_{sp,_{s,2D}}(q_{_{s,2D}})\tau_{sp,_{s,2D}}(q_{_{s,2D}}, \dot{q}_{_{s,2D}}) \tag{3.2.47}$$

where matrices $\mathcal{B}_{_{s,2D}}$ and $\mathcal{B}_{sp,_{s,2D}}$ define how the motor torques $u = [u_{1R}, u_{2R}, u_{1L}, u_{2L}]'$ and the spring torques $\tau_{_{s,2D}}$ enter the model, respectively. For $i \in \{1R, 2R, 1L, 2L\}$, the spring torque of the i-th series-elastic actuator is modeled similar to the tether-free model.

Setting $x_{_{s,2D}} = (q_{_{s,2D}}; \dot{q}_{_{s,2D}}) \in \mathcal{T}\mathcal{Q}_{_{s,2D}}$, the model in state-variable form is

$$\dot{x}_{_{s,2D}} = \begin{bmatrix} \dot{q}_{_{s,2D}} \\ \mathcal{D}^{-1}_{_{s,2D}}(q_{_{s,2D}})(-\mathcal{H}_{_{s,2D}}(q_{_{s,2D}}, \dot{q}_{_{s,2D}}) + \mathcal{B}_{_{s,2D}}u) \end{bmatrix}, \tag{3.2.48}$$

where

$$\begin{aligned}\mathcal{H}_{_{s,2D}}(q_{_{s,2D}}, \dot{q}_{_{s,2D}}) =& \mathcal{C}_{_{s,2D}}(q_{_{s,2D}}, \dot{q}_{_{s,2D}})\dot{q}_{_{s,2D}} + \mathcal{G}_{_{s,2D}}(q_{_{s,2D}}) - \mathcal{B}_{sp,_{s,2D}}(q_{_{s,2D}})\tau_{sp,_{s,2D}}(q_{_{s,2D}}, \dot{q}_{_{s,2D}}) - \\ & \mathcal{B}_{fric,_{s,2D}}(q_{_{s,2D}})\tau_{fric,_{s,2D}}(q_{_{s,2D}}, \dot{q}_{_{s,2D}})\end{aligned} \tag{3.2.49}$$

Equation (3.2.49) immediately leads to the state variable model

$$\dot{x}_{s,2D} = \mathbf{f}_{s,2D}(x_{s,2D}) + \mathbf{g}_{s,2D}(x_{s,2D})u. \tag{3.2.50}$$

3.2.4.2 Impact Model

The impact map is once again developed using the method in [108]. Let $p_{v,s,2D} : Q_{s,2D} \mapsto \mathbb{R}$ denote the vertical height of the swing leg above the ground so that the impact surface is

$$\mathcal{S}_{s,2D} = \{x_{s,2D} \in \mathcal{TQ}_{s,2D} \mid p_{v,s,2D}(q_{s,2D}) = 0, p_v(q_{s,2D}) > 0\}. \tag{3.2.51}$$

The impact map for the planar model is obtained from the unconstrained model of the robot with the same approach explained previously, therefore,

$$x^+_{s,2D} = \Delta_{s,2D}(x^-_{s,2D}), \tag{3.2.52}$$

where $x^-_{s,2D}$ is the value of the state just before the impact and $x^+_{s,2D}$ is its value just after the impact.

3.2.4.3 Hybrid Model

A hybrid model is then formed, just as in (3.2.39). The model can be simplified to a single-phase system with impulse effects if leg swapping is incorporated into the impact map, as in [110], yielding

$$\Sigma_{s,2D} : \begin{cases} \dot{x}_{s,2D} = \mathbf{f}_{s,2D}(x_{s,2D}) + \mathbf{g}_{s,2D}(x_{s,2D})u, & x^-_{s,2D} \notin \mathcal{S}_{s,2D} \\ x^+_{s,2D} = \Delta_{s,2D}(x^-_{s,2D}), & x^-_{s,2D} \in \mathcal{S}_{s,2D} \end{cases}$$

3.2.5 Planar Model on Compliant Ground

The planar model with point feet walking over compliant ground is obtained from the unconstrained model of the robot after defining four holonomic constraints – that restrict the motion of the robot to the sagittal plane of walking

$$\mathcal{TQ}_{f,2D} = \{(q_f; \dot{q}_f) \in \mathcal{TQ}_f \mid q_z = 0, q_y = 0, q_{3R} = 0, q_{3L} = 0, \dot{q}_z = 0, \dot{q}_y = 0, \dot{q}_{3R} = 0, \dot{q}_{3L} = 0\}.$$

The Lagrangian restricted to $\mathcal{TQ}_{f,2D}$ for $q_{f,2D} \in \mathcal{Q}_{f,2D}$ and $\dot{q}_{f,2D} \in \mathcal{TQ}_{f,2D}$ results in

$$\mathcal{D}_{f,2D}(q_{f,2D})\ddot{q}_{f,2D} + \mathcal{C}_{f,2D}(q_{f,2D}, \dot{q}_{f,2D})\dot{q}_{f,2D} + \mathcal{G}_{f,2D}(q_{f,2D}) = f(q_{f,2D}, \dot{q}_{f,2D}, u) + J'(q_{f,2D}).\mathcal{F}$$

where $J'(q_{f,2D}) \in \mathbb{R}^{4\times 12}$ is a Jacobian matrix of the robot's legs end-points,

$$J'(q_{f,2D}) = \frac{\partial}{\partial q_{f,2D}} \begin{bmatrix} y_1 \\ z_1 \\ y_2 \\ z_2 \end{bmatrix}$$

and $\mathcal{F} \in \mathbb{R}^4$ consists of tangential and normal forces acting at the ends of the robot's legs,

$$\mathcal{F} = \begin{bmatrix} \mathcal{F}_{1,y} \\ \mathcal{F}_{1,z} \\ \mathcal{F}_{2,y} \\ \mathcal{F}_{2,z} \end{bmatrix}$$

The tangential and normal forces $\mathcal{F}$ are modeled similar to the tether-free model.

CHAPTER IV

Gait Design and Optimization

When a holonomic constraint is imposed through the action of an actuator rather than the internal forces of a physical constraint, it is said to be *virtual* [110, 69, 111]. Virtual constraints can be used to synchronize the links of a robot in order to achieve common objectives of walking, such as supporting the torso, advancing the swing leg in relation to the stance leg, or specifying foot clearance. Analogous to physical constraints, virtual constraints induce a reduced-dimensional model compatible with the constraints, called zero dynamics [112, 69]. When combined with parameter optimization, virtual constraints can be designed to achieve additional objectives such as walking at a desired speed and respecting bounds on ground reaction forces. Energy efficiency can also be increased, as explained in [69, 109, 105, 113].

This chapter begins with detailed derivations of the equations governing MARLO's zero dynamics using a general family of output functions that are only a function of the angles and not the angular velocities. Thereafter, a more specialized family of virtual constraints tied to MARLO's morphology are introduced, followed by a systematic gait design algorithm based on optimizing a metric characterizing MARLO's energetic efficiency subject to equality and inequality constraints. Simulations for nominal walking speeds of 0.5 to 1.4 $[\frac{m}{s}]$, in increments of 0.1 $[\frac{m}{s}]$, are presented. Walking at 1 $[\frac{m}{s}]$ is analyzed in greater detail. In addition, this chapter will provide guidelines and suggestions for improving the energetic efficiency of walking in the context of positive work and negative work before concluding with a summary note.

4.1 MARLO's Zero Dynamics

One virtual constraint per actuator is proposed in the form of an output that, when zeroed by a feedback controller, enforces the constraint. The constraints are written in the form

$$y = h(q_s, \alpha) = h_0(q_s) - h_d(\theta(q_s), \alpha), \tag{4.1.1}$$

where $h_0(q_s)$ specifies the vector of variables to be controlled, $h_d(\theta, \alpha)$ is the desired evolution of the controlled variables as a function of $\theta(q_s)$, and $\alpha = [\alpha_{j,i}] \in \mathbb{R}^{\,6\times(n+1)}$ is a matrix of real parameters to be chosen. The number of columns, $n+1$, is defined later in Section 5.1. A gait-timing variable $\theta(q_s)$ is used to replace time in parametrizing the motion of the robot. Thus, $\theta(q_s)$ is selected to be strictly monotonic (i.e., strictly increasing or decreasing) along normal walking gaits.

When the decoupling matrix (6.2.5) is invertible, see [112, 69, 109], zeroing of the virtual constraints in (4.1.1) via feedback creates a parametrized smooth manifold $\mathcal{Z}_\alpha$ that is invariant under the flow of the closed-loop single support dynamics. The dynamics restricted to this surface, that is, the dynamics compatible with the constraints, is called "zero dynamics". In each single-support phase, zero dynamics can be expressed in the form of a second-order system,

$$\mathcal{D}_{zero}(q_{zero}, \alpha)\ddot{q}_{zero} + \mathcal{H}_{zero}(q_{zero}, \dot{q}_{zero}, \alpha) = 0, \tag{4.1.2}$$

where $(q_{zero}, \dot{q}_{zero})$ are the unactuated variables in the Lagrangian model and constitute a set of local coordinates for the manifold $\mathcal{Z}_\alpha$. $\mathcal{D}_{zero}$ includes inertial terms and $\mathcal{H}_{zero}$ includes gravity and Coriolis terms. Because the actuators are doing work on the system when zeroing the outputs, the resulting dynamics may not be Lagrangian, although in special cases, the zero dynamics is Lagrangian [69]. When the virtual constraints in (4.1.1) have vector relative degree two[1] [112, 69], as is the case here, the dimension of q_{zero} equals the dimension of q_s minus the number of independent actuators. Hence, for the tether-free model used here, $q_{zero} = (q_z; q_y; q_x; q_{1R}; q_{2R}; q_{1L}; q_{2L})$.

4.1.1 Gait timing variable and coordinate partition

Let θ be defined by

$$\theta(q_s) = \mathcal{C}_0 q_s + \mathcal{C}_1, \tag{4.1.3}$$

for an appropriate row vector $\mathcal{C}_0$ and scalar $\mathcal{C}_1$. Let $q_{con} = h_0(q_s)$ denote the controlled variables in the output in (4.1.1) and suppose that q_{con} can be expressed as an affine function of the configuration variables,

$$q_{con} = \tilde{\mathcal{H}}_0 q_s + \tilde{\mathcal{H}}_1. \tag{4.1.4}$$

Let q_{zero} be a complementary set of variables satisfying (4.1.5),

$$q_{zero} = \hat{\mathcal{H}}_0 q_s + \hat{\mathcal{H}}_1, \tag{4.1.5}$$

and selected so that (4.1.6) is a set of generalized coordinates for $\mathcal{Q}_s$,

$$\bar{q} = \begin{bmatrix} q_{con} \\ q_{zero} \end{bmatrix} \tag{4.1.6}$$

Define

$$\mathcal{H}_0 = \begin{bmatrix} \tilde{\mathcal{H}}_0 \\ \hat{\mathcal{H}}_0 \end{bmatrix}, \ \mathcal{H}_1 = \begin{bmatrix} \tilde{\mathcal{H}}_1 \\ \hat{\mathcal{H}}_1 \end{bmatrix}, \tag{4.1.7}$$

[1]This means that the second derivatives of the six outputs in (4.1.1) depend on six inputs in a full rank or independent manner.

where it follows that $\mathcal{H}_0$ has (full) rank equal to the dimension of $\mathcal{Q}_s$. Defining $\mathcal{T}_0 = \mathcal{H}_0^{-1}$ and $\mathcal{T}_1 = -\mathcal{H}_0^{-1}\mathcal{H}_1$ leads to (4.1.8),

$$q_s = \mathcal{T}_0 \bar{q} + \mathcal{T}_1. \tag{4.1.8}$$

The coordinates expressed in (4.1.6) have been partitioned into those that are directly actuated and those that are not actuated. The first block of coordinates are sometimes referred to as the "controlled variables," while the second block are sometimes referred to as "unactuated" coordinates.

4.1.2 Key assumptions

The following conditions are assumed to hold.

1. The controlled variables q_{con} in (4.1.4), the gait-timing variable $\theta(q_s)$ in (4.1.3), and the desired evolution $h_d(\theta)$ in (4.1.1) have been selected so that the decoupling matrix is invertible.

2. The complementary variables q_{zero} of (4.1.5) have been selected so that $\mathcal{T}_0' \mathcal{B} = \begin{bmatrix} \bar{\mathcal{B}}_1 \\ 0 \end{bmatrix}$ and $\bar{\mathcal{B}}_1$ is 6×6, that is, it is square with the size determined by the number of actuators.

3. The gait-timing variable can be expressed in terms of q_{zero}, that is:

$$\theta(\bar{q}) = \bar{\mathcal{C}}_0 q_{zero} + \bar{\mathcal{C}}_1. \tag{4.1.9}$$

4.1.3 Model decomposition and zero dynamics

Expressing the mechanical model (3.2.26) in the coordinates (4.1.6) leads to:

$$\bar{\mathcal{D}}(\bar{q})\ddot{\bar{q}} + \bar{\mathcal{H}}(\bar{q}, \dot{\bar{q}}) = \bar{\mathcal{B}} u, \tag{4.1.10}$$

where

$$\bar{\mathcal{D}}(\bar{q}) = \mathcal{T}_0' \mathcal{D}_s(q_s) \mathcal{T}_0 \big|_{q_s = \mathcal{T}_0 \bar{q} + \mathcal{T}_1}, \tag{4.1.11}$$

$$\bar{\mathcal{H}}(\bar{q}, \dot{\bar{q}}) = \mathcal{T}_0' \mathcal{H}_s(q_s, \dot{q}_s) \mathcal{T}_0 \big|_{\substack{q_s = \mathcal{T}_0 \bar{q} + \mathcal{T}_1 \\ \dot{q}_s = T_0 \dot{\bar{q}}}} \tag{4.1.12}$$

and

$$\bar{\mathcal{B}} = \mathcal{T}_0' \mathcal{B}_s. \tag{4.1.13}$$

In the transformed coordinates, the dynamic model can be partitioned as,

$$\begin{bmatrix} \bar{\mathcal{D}}_{11}(\bar{q}) & \bar{\mathcal{D}}_{12}(\bar{q}) \\ \bar{\mathcal{D}}_{21}(\bar{q}) & \bar{\mathcal{D}}_{22}(\bar{q}) \end{bmatrix} \begin{bmatrix} \ddot{q}_{con} \\ \ddot{q}_{zero} \end{bmatrix} + \begin{bmatrix} \bar{\mathcal{H}}_1(\bar{q}, \dot{\bar{q}}) \\ \bar{\mathcal{H}}_2(\bar{q}, \dot{\bar{q}}) \end{bmatrix} = \begin{bmatrix} \bar{\mathcal{B}}_1 \\ 0 \end{bmatrix} u \tag{4.1.14}$$

The zero dynamics is the unactuated part of this model, namely,

$$\bar{\mathcal{D}}_{21}(\bar{q}) \ddot{q}_{con} + \bar{\mathcal{D}}_{22}(\bar{q}) \ddot{q}_{zero} + \bar{\mathcal{H}}_2(\bar{q}, \dot{\bar{q}}) = 0. \tag{4.1.15}$$

We now bring the virtual constraints into consideration,

$$0 = q_{con} - h_d(\theta), \tag{4.1.16}$$

that is,

$$q_{con} = h_d(\theta). \tag{4.1.17}$$

Computing the derivatives of q_{con} with respect to time so that we can substitute into (4.1.15) gives,

$$\dot{q}_{con} = \frac{\partial h_d(\theta)}{\partial \theta} \dot{\theta}, \tag{4.1.18}$$

and

$$\ddot{q}_{con} = \frac{\partial h_d(\theta)}{\partial \theta} \ddot{\theta} + \frac{\partial^2 h_d(\theta)}{\partial \theta^2} (\dot{\theta})^2. \tag{4.1.19}$$

(4.1.9) yields:

$$\dot{\theta} = \bar{c}_0 \dot{q}_{zero} \tag{4.1.20}$$

$$\ddot{\theta} = \bar{c}_0 \ddot{q}_{zero}. \tag{4.1.21}$$

Substituting these expressions into (4.1.15) and simplifying leads to:[2]

$$\mathcal{D}_{zero}(q_{zero})\ddot{q}_{zero} + \mathcal{H}_{zero}(q_{zero}, \dot{q}_{zero}) = 0, \tag{4.1.22}$$

where

$$\mathcal{D}_{zero} = \bar{\mathcal{D}}_{22} + \bar{\mathcal{D}}_{21}\frac{\partial h_d(\theta)}{\partial \theta}\bar{c}_0 \tag{4.1.23}$$

and

$$\mathcal{H}_{zero} = \bar{\mathcal{H}}_1 + \bar{\mathcal{D}}_{21}\frac{\partial^2 h_d(\theta)}{\partial \theta^2}(\dot{\theta})^2. \tag{4.1.24}$$

The control signal u compatible with the virtual constraints being zeroed can also be computed from (4.1.14), (4.1.19), and (4.1.21):

$$\begin{aligned} u^* =& \bar{\mathcal{B}}_1^{-1}\left\{\left[\bar{\mathcal{D}}_{12} + \bar{\mathcal{D}}_{11}\frac{\partial h_d(\theta)}{\partial \theta}\bar{c}_0\right]\ddot{q}_{zero} + \right. \\ & \left. \bar{\mathcal{H}}_1 + \bar{\mathcal{D}}_{11}\frac{\partial^2 h_d(\theta)}{\partial \theta^2}(\dot{\theta})^2\right\}. \end{aligned} \tag{4.1.25}$$

4.2 Specification of the Virtual Constraints

For the tether-free model of MARLO, the controlled variables, when the right leg is the stance leg, follow.

[2]When the decoupling matrix is invertible, $\mathcal{D}_{zero}$ is guaranteed to be invertible as well.

4.2.1 Leg Angles

The leg angle variables represent quantities in the sagittal plane, on the motor side of the series-compliant actuators; specifically, they correspond to the angles of the right and left legs relative to the torso when the spring deflection is zero:

$$h_0^{LegAngle}(q_s) = \begin{bmatrix} \dfrac{q_{gr1R} + q_{gr2R}}{2} \\ \dfrac{q_{gr1L} + q_{gr2L}}{2} \end{bmatrix} \tag{4.2.1}$$

4.2.2 Knee Angles

The knee angle variables represent quantities in the sagittal plane, on the motor side of the series-compliant actuators; specifically, they correspond to the relative angle of the thigh and shin links, when the spring deflection is zero:

$$h_0^{KneeAngle}(q_s) = \begin{bmatrix} \dfrac{q_{gr2R} - q_{gr1R}}{2} \\ \dfrac{q_{gr2L} - q_{gr1L}}{2} \end{bmatrix} \tag{4.2.2}$$

4.2.3 Hip Angles

The hip angle variables are the angles of the legs relative to the torso, in the frontal plane; recall that these variables are not actuated through springs.

$$h_0^{HipAngle}(q_s) = \begin{bmatrix} q_{3R} \\ q_{3L} \end{bmatrix} \tag{4.2.3}$$

Stacking the leg angle, knee angle and hip angle variables to form the vector of controlled variables defines $h_0(q_s)$ as

$$h_0(q_s) = \begin{bmatrix} \dfrac{q_{gr1R} + q_{gr2R}}{2} \\ \dfrac{q_{gr1L} + q_{gr2L}}{2} \\ q_{gr2R} - q_{gr1R} \\ q_{gr2L} - q_{gr1L} \\ q_{3R} \\ q_{3L} \end{bmatrix} = \begin{bmatrix} q_{grR}^{LA} \\ q_{grL}^{LA} \\ q_{grR}^{Knee} \\ q_{grL}^{Knee} \\ q_{R}^{Hip} \\ q_{L}^{Hip} \end{bmatrix}, \tag{4.2.4}$$

4.2.4 Gait-Timing Variable

The gait-timing variable is selected as

$$\theta(q_s) = \begin{cases} \theta_R(q_s) & \text{if the right leg is stance;} \\ \theta_L(q_s) & \text{otherwise,} \end{cases} \tag{4.2.5}$$

where $\theta_R(q_s)$ is the angle between the virtual leg[3] and the ground surface normal vector when the right leg is the stance leg. $\theta_L(q_s)$ is defined in an analogous manner when the left leg is the stance leg.

[3]Virtual leg is defined as a virtual line connecting the pivot point of the stance leg to the hip joint.

4.3 Systematic Gait Design, Optimization Method

As shown in Section 4.1, the process of deriving (4.1.22) provides a closed-form expression for the control input $u^*_\alpha(q_{zero}, \dot{q}_{zero})$,

$$\begin{aligned} u^* =& \bar{\mathcal{B}}_1^{-1} \left\{ \left[\bar{\mathcal{D}}_{12} + \bar{\mathcal{D}}_{11} \frac{\partial h_d(\theta)}{\partial \theta} \bar{\mathcal{C}}_0 \right] \ddot{q}_{zero} + \right. \\ & \left. \bar{\mathcal{H}}_1 + \bar{\mathcal{D}}_{11} \frac{\partial^2 h_d(\theta)}{\partial \theta^2} (\dot{\theta})^2 \right\}, \end{aligned} \tag{4.3.1}$$

that is, the control input that zeros the virtual constraints and creates the zero dynamics manifold $\mathcal{Z}_\alpha$

$$0 = h(q_s, \alpha) \Leftrightarrow h_0(q_s) = h_d(\theta, \alpha). \tag{4.3.2}$$

Solutions of the zero dynamics (4.1.22) are exact solutions of the full-dimensional model (3.2.31) in a closed-loop with u^*_α [109, 114, 105]. This motivates posing an optimization problem to select α, resulting in a periodic solution of the hybrid model (3.2.39), where the lower-dimensional dynamic equations of the zero dynamics (4.1.22) are integrated in place of the full dynamics (3.2.33) in a closed loop with u^*_α.

The cost function will be taken as the cost of mechanical transport cmt [35],

$$J(\alpha, q^-_{zero}, \dot{q}^-_{zero}) = \frac{1}{g M_{tot} d} \int_0^{T_I} \sum_{i=1}^{6} [p_i(t)]_+ dt, \tag{4.3.3}$$

where $(q^-_{zero}; \dot{q}^-_{zero})$ denotes the final condition of the zero dynamics. Furthermore, T_I is the step duration, d represents the distance traveled by the CoM in one step, g denotes the gravitational constant, M_{tot} is the total mass of the robot, $p_i(t)$ is actuator power, and $[p]_+ = p$ when $p \geq 0$ and equals zero otherwise. According to [35], for humans walking at approximately 1 m/s, cmt is approximately 0.05.

The parameter vector α and the initial conditions $(q^-_{zero}; \dot{q}^-_{zero})$ are chosen to minimize $J(\alpha, q^-_{zero}, \dot{q}^-_{zero})$ subject to the walking gait being periodic and symmetric, the ground reaction forces are feasible, and the speed is a desired value. Here, the minimization was carried out in MATLAB using the `fmincon` function.

If the optimization process is successful, it returns the (locally) optimized value for the parameters in the virtual constraints α^*, and the final conditions $(q^-_{zero}; \dot{q}^-_{zero})$ for a periodic solution of the hybrid model (3.2.39) corresponding to a walking gait with the specified properties. Chapter VI will discuss how the virtual constraints can be used to synthesize a feedback controller that asymptotically stabilizes the walking gait.

4.4 Optimal Solutions

The optimization framework of Sections 4.3 and 5.1 has been applied to the 3D model of Section 3.2.2 to compute walking gaits that are locally optimal with respect to cmt . The gaits may only be locally optimal because the dependence of the cost function (4.3.3) on the parameters of the virtual constraints and the final conditions of the zero dynamics is non-convex. Simulations for nominal walking speeds of 0.5 to 1.4 $[\frac{m}{s}]$, in increments of 0.1 $[\frac{m}{s}]$, are presented. Walking at 1 $[\frac{m}{s}]$ is analyzed in greater detail.

4.4.1 Walking Efficiency versus Speed

For comparison purposes, the reported cmt corresponding to human walking gait at $1[\frac{m}{s}]$ is 0.05, and the cmt-based efficiency of Cornell's Ranger is 0.04 at a speed of $0.6[\frac{m}{s}]$. Figure 4.1 shows the computed cmt versus speeds of 0.5 to 1.4 $[\frac{m}{s}]$, in increments of 0.1 $[\frac{m}{s}]$, when Bézier curves are used to parametrize the desired trajectories for the holonomic constraints in the tether-free model of MARLO while it walks over rigid flat ground with point feet. Figure 4.1 suggests that a quadratic relation exists between the walking speed and the cmt, and that the minimum cmt of 0.09 takes place at a speed of $0.5[\frac{m}{s}]$. Chapter V shows how using a new family of parametric curves makes it possible to improve the energetic efficiency of MARLO's walking gait.

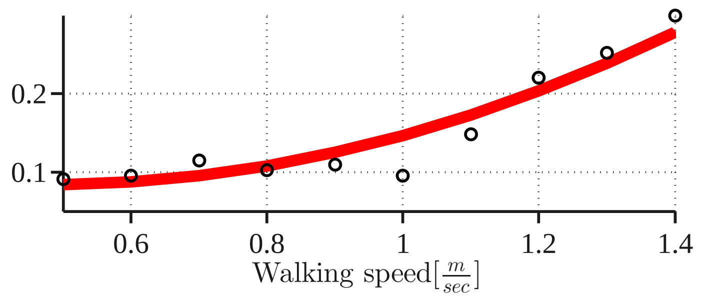
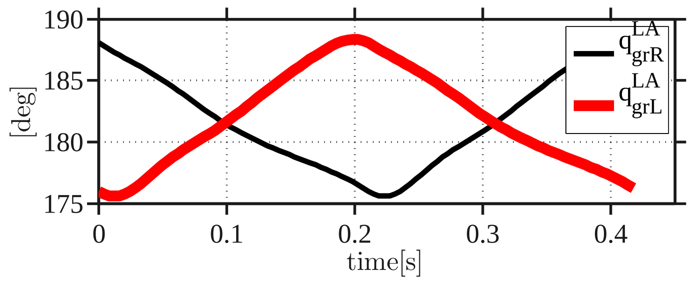


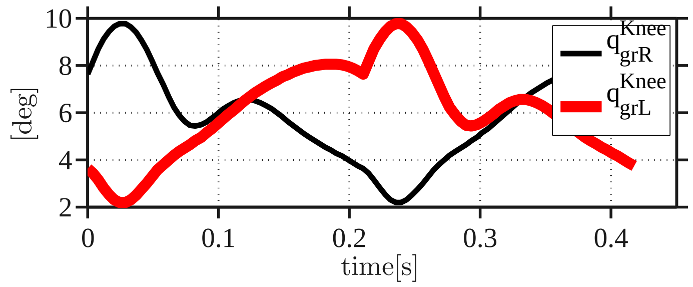
Figure 4.1: cmt versus walking velocity for nominal walking speeds of 0.5 to 1.4 $[\frac{m}{s}]$, in increments of 0.1 $[\frac{m}{s}]$, when Bézier curves are used as desired trajectories for the holonomic constraints.

4.4.1.1 Walking at 1 $[\frac{m}{s}]$

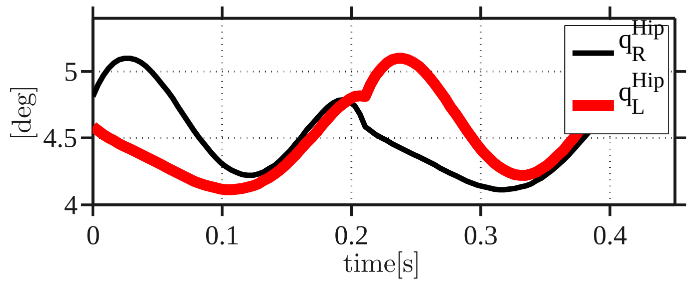
A periodic walking gait optimized at 1 $[\frac{m}{s}]$ is analyzed in more detail. In the following, the right leg is the stance leg while the left leg is the swing leg. The evolution of the virtual constraints is shown in Fig. 4.2. The motor-side right leg angle q_{grR}^{LA} decreases from $187[deg]$ to $175[deg]$ and the corresponding left leg angle q_{grL}^{LA} increases from $175[deg]$ to $187[deg]$. The motor-side right knee angle q_{grR}^{Knee} varies between $4[deg]$ and $10[deg]$ and the left knee angle q_{grL}^{Knee} varies from $2[deg]$ to $8[deg]$. In the frontal plane, the right hip joint angle q_R^{Hip} and the left hip joint angle q_L^{Hip} shown in Fig. 4.2c are essentially constant, being bounded between $4[deg]$ and $5[deg]$. Figure 4.3 shows the motor torques for u_{1R}, u_{1L}, u_{2R}, u_{2L}, u_{3R} and u_{3L}, while Fig. 4.4 shows the corresponding actuator power (product of torque and angular velocity at the motor shafts).



(a)

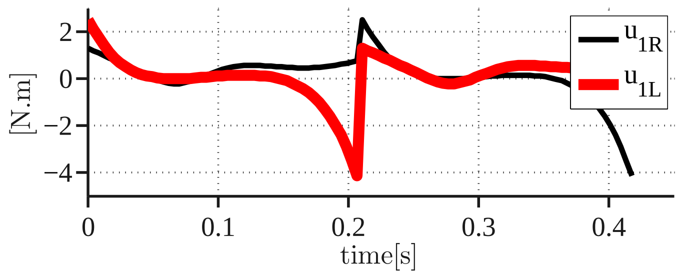


(b)

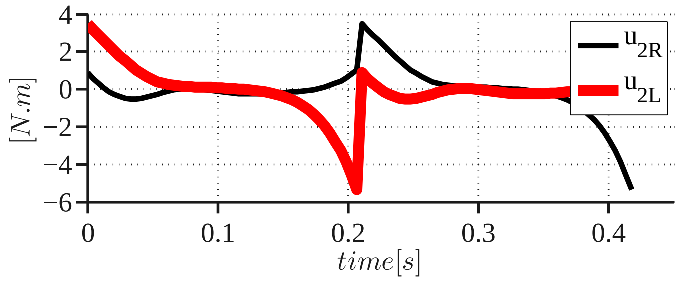


(c)

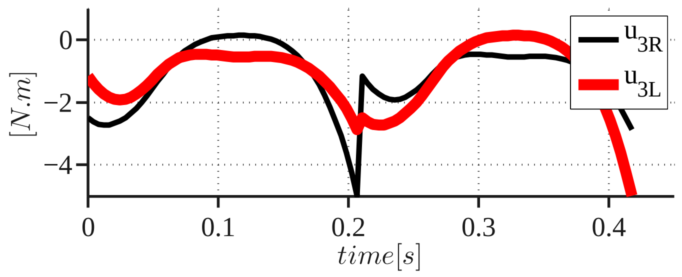
Figure 4.2: Evolution of **(a)** the right leg angle q_{grR}^{LA} and the left leg angle q_{grL}^{LA}, **(b)** the right knee angle q_{grR}^{Knee} and the left knee angle q_{grL}^{Knee}, **(c)** the right hip joint angle q_{R}^{Hip} and the left hip joint angle q_{L}^{Hip} during two steps for an optimal walking motion with the nominal velocity of $1.0[\frac{m}{s}]$ and cmt 0.096.



(a)

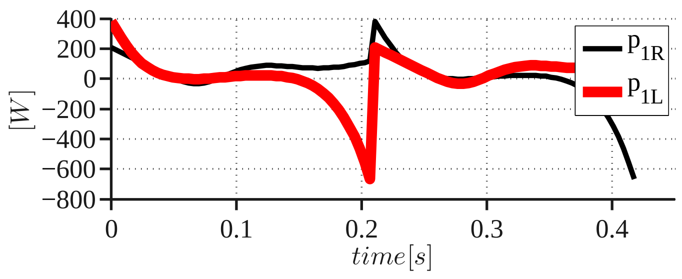


(b)

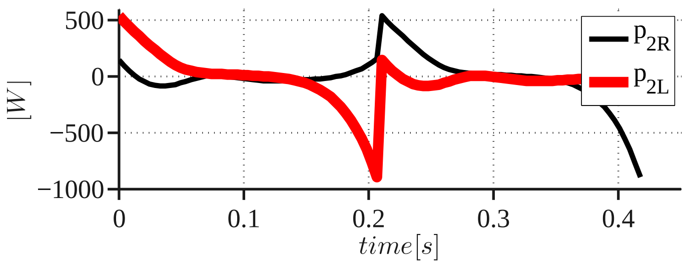


(c)

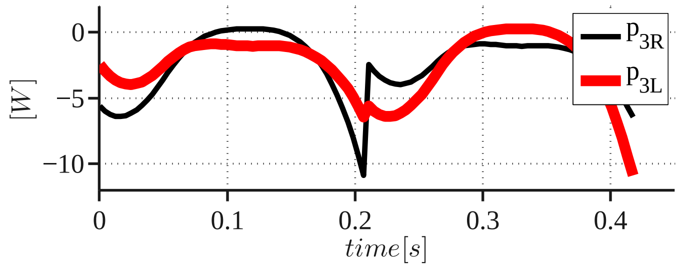
Figure 4.3: Control effort **(a)** u_{1R}, u_{1L}, **(b)** u_{2R}, u_{2L}, and **(c)** u_{3R}, u_{3L} during two steps corresponding to a fixed point with a nominal walking speed of $1.0[\frac{m}{s}]$ and cmt 0.096.



(a)



(b)



(c)

Figure 4.4: Actuation power **(a)** p_{1R}, p_{1L}, **(b)** p_{2R}, p_{2L}, and **(c)** p_{3R}, p_{3L} during two steps corresponding to a fixed point with a nominal walking speed of $1.0[\frac{m}{s}]$ and cmt 0.096.

Tables 4.1 and 4.2 present the amount of positive work $\mathbf{W}^{\oplus}$ performed by each actuator as well as the amount of negative work $\mathbf{W}^{\ominus}$ absorbed by each actuator over the course of a single step. A notable point is the large amount of negative work done by actuator u_1 of the stance leg. Due to the parallelogram mechanism, u_1 is positive to support the robot but moves in a negative direction to advance the body of the robot. In Tables 4.1 and 4.2, $\mathbf{p}^{max}$ and $\mathbf{u}^{max}$ denote the peak power and peak torque during the single step when the right leg is stance.

$\mathbf{u}_{iR}$	$\mathbf{W}^{\oplus}$	$\mathbf{W}^{\ominus}$	$\mathbf{p}^{max}$	$\mathbf{u}^{max}$
u_{1R}	0.154	3.605	84.518	4.154
u_{2R}	1.711	1.265	113.720	5.358
u_{3R}	0.429	0.608	42.834	4.989

Table 4.1: Right actuator's energy breakdown over one step for walking at $1.0[\frac{m}{s}]$; right leg is stance. The cmt is 0.096.

$\mathbf{u}_{iL}$	$\mathbf{W}^{\oplus}$	$\mathbf{W}^{\ominus}$	$\mathbf{p}^{max}$	$\mathbf{u}^{max}$
u_{1L}	1.339	1.487	174.23	4.154
u_{2L}	4.434	3.444	391.780	5.358
u_{3L}	0.283	0.330	27.298	4.989

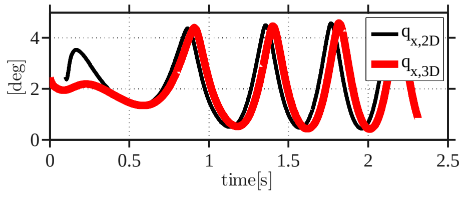
Table 4.2: Left actuator's energy breakdown over one step for walking at $1.0[\frac{m}{s}]$; right leg is stance. The cmt is 0.096.

4.5 Effects of Torso's CoM and Harmonic Drive Gear Ratio on Energy Efficiency of MARLO

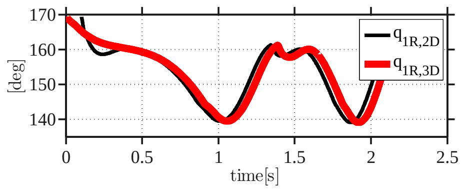
Section 3.2.4 obtained a planar model from the tether-free model by adding constraints to the Lagrangian. This section suggests that the planar and tether-free models are in agreement.

Consider an optimal walking motion with a cmt of 0.08 and an average walking speed $1.0[\frac{m}{s}]$, which is obtained using the HZD optimizer for the tether-free model. Next, by removing the yaw and roll coordinates q_z and q_y and their derivatives $\dot{q}_z$ and $\dot{q}_y$ from the final state $(q^-_{zero}, \dot{q}^-_{zero})$, and also removing the free parameters corresponding to q_{3R} and q_{3L} in the virtual constraint, an initial guess is obtained for the planar HZD optimizer. It is worth mentioning that the controlled variables for the planar model are equal to those of the tether-free model in the sagittal plane. Continuing the optimization process using the `fmincon` function results in a locally optimal solution for the planar model with cmt

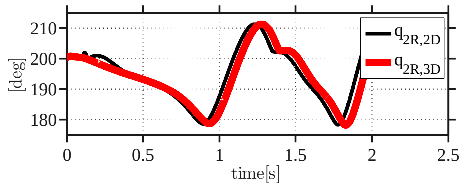
0.0778 and an average walking speed $1.0[\frac{m}{s}]$. The evolution of the sagittal plane angles for the planar and the tether-free models are shown in Fig. 4.5. It can be seen that the virtual constraints and initial conditions for the states of the tether-free model result in very good initial solutions for performing iterative gait optimization of the planar model, resulting in nearly identical predictions of cmt. Now that it has been shown that the planar model and the tether-free model are in agreement, a parametric design is addressed on the planar model.



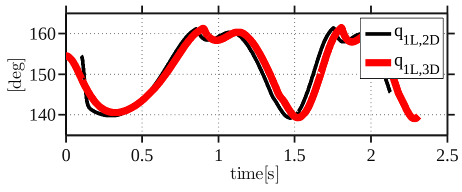
(a)



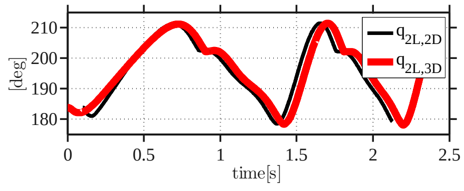
(b)



(c)



(d)



(e)

Figure 4.5: Comparison between the planar model and tether-free model.

An optimal solution for the tether-free model with cmt 0.0778 and a nominal walking speed of $1.0[\frac{m}{s}]$ is used as the initial condition for optimizing the planar model using HZD. The optimization of the planar model is repeated as the position of CoM of the torso is moved down by $0.3[m]$, which resulted in only a slight change in cmt value, an increase of approximately 0.5%. Next, the position of the CoM is returned to its nominal value and the gear ratio of the harmonic drive is modified to $30 : 1$ instead of $50 : 1$. When the optimization is redone, the cmt decreases by 38.6%. Figures 4.6 and 4.7 show the sagittal plane controllers u_{gr1R},u_{gr1L},u_{gr2R} and u_{gr2L}, respectively. These results suggest that modifications of the position of CoM may not vary the value of cmt on the tether-free model. In contrast, the changes in the harmonic drive gear ratio strongly affect the energetic efficiency of the tether-free model.

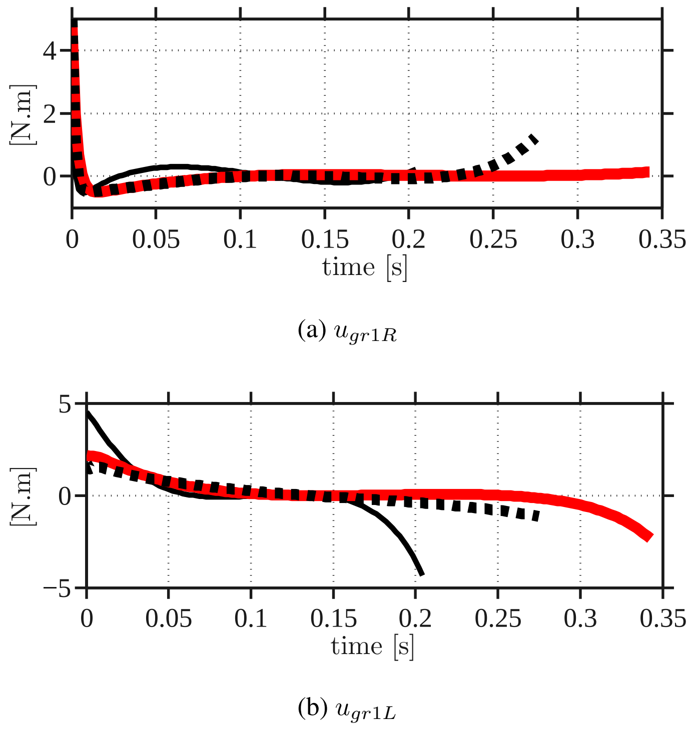
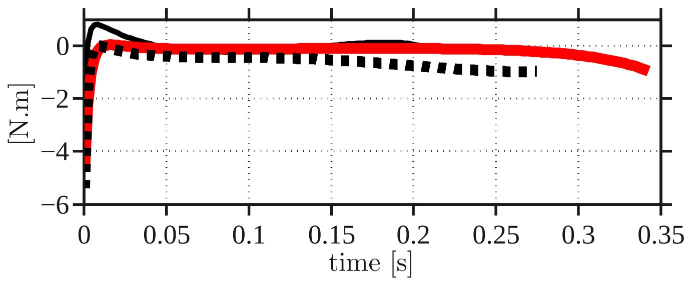
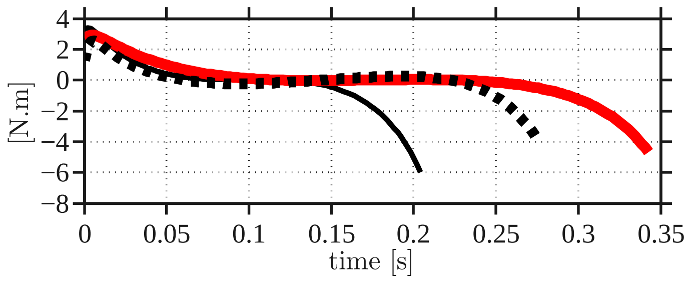


Figure 4.6: Controller effort for the planar model **(a)** u_{1R} and **(b)** u_{1L}, respectively. The black solid curve corresponds to the nominal gait. The red solid thick curve corresponds to the model with a modified position of the CoM, and the dashed black curve corresponds to the model with a modified gear ratio.

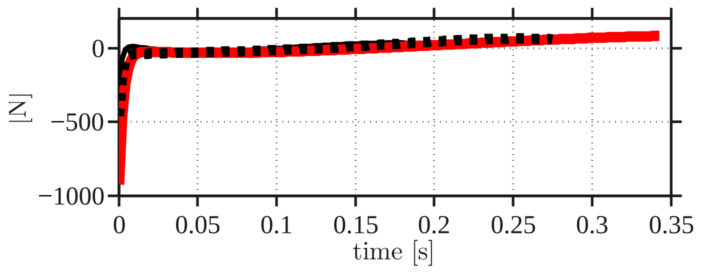


(a) u_{gr2R}

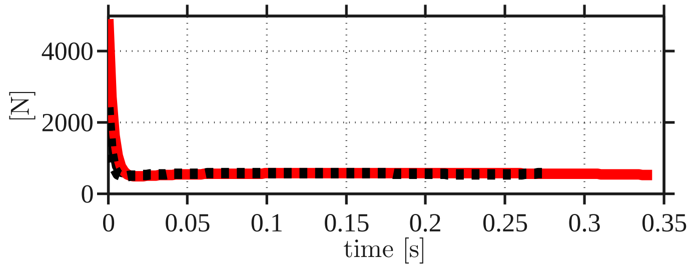


(b) u_{gr2L}

Figure 4.7: Controller effort for the planar model **(a)** u_{2R} and **(b)** u_{2L}, respectively. The black solid curve corresponds to the nominal gait. The red solid thick curve corresponds to the model with a modified position of the CoM, and the dashed black curve corresponds to the model with a modified gear ratio.



(a) F_t



(b) F_n

Figure 4.8: Forces at the end of stance leg for the 2D model, **(a)** F_t tangential force and **(b)** F_n normal force. Solid black curve corresponds to the nominal gait. Red solid thick curve corresponds to the model with a modified position of the CoM, and the dashed black curve corresponds to the model with a modified gear ratio.

CHAPTER V

Non Uniform Rational B-spline NURBs

As noted in the previous chapter, Bézier polynomials have been used in references [69], [109, Chap. 6], [91] and [105] to parametrize virtual holonomic constraints during gait design and optimization for robots such as Rabbit and MABEL. In this chapter, a more general family of parametric curves called NURBs is introduced for this purpose.

The chapter first explains how NURB curves are mathematically defined. Then, after introducing some of the useful features of NURB curves, the holonomic constraints corresponding to the leg angles, knee angles, and hip angles, explained in Chapter IV, are parametrized in terms of NURB curves before being employed in a constrained optimization based on HZD. Thereafter, the optimal cmt solutions based on NURB and Bézier desired trajectories for walking speed starting from 0.5 to $1.4[\frac{m}{s}]$ are compared to each other and relevant properties of NURB curves are discussed to explain the solutions. Finally, the last section suggests applications of NURB curves for walking controller design.

5.1 NURB Curves

A NURB curve is defined in a two-dimensional homogeneous coordinate space as follows. Let s be related to the gait timing variable by $s = \frac{\theta(q_s)-\theta^+}{\theta^- - \theta^+}$, where θ^+ and θ^- are the initial and final values of the gait timing variable θ on the periodic orbit, and let $P(s) = (s; h_{d,j}(s)) \in \mathbb{R}^2$ be the 2D position (i.e., graph) of the NURB curve for the j-th output. Then

$$P(s) = \frac{\sum_{i=1}^{n+1} B_i \, w_i \, \mathcal{N}_{i,k}(s)}{\sum_{i=1}^{n+1} w_i \, \mathcal{N}_{i,k}(s)} = \sum_{i=1}^{n+1} B_i \, \mathcal{R}_{i,k}(s), \tag{5.1.1}$$

where w_i, $i = 1, \cdots, n+1$ are non-negative real scalars, $B_i = (s_i; \alpha_{j,i}) \in \mathbb{R}^2$ is the 2D position of the i-th control vertex for the j-th output function, $\mathcal{N}_{i,k}(s)$ denotes the basis function of order k corresponding to the i-th control point defined by *Cox-de Boor* recursion, [115], namely,

$$\mathcal{N}_{i,0}(s) = \begin{cases} 1 & s_i \leq s < s_{i+1} \\ 0 & \text{otherwise}, \end{cases} \tag{5.1.2}$$

$$N_{i,k}(s) = \frac{(s - s_i)\, \mathcal{N}_{i,k-1}(s)}{s_{i+k-1} - s_i} + \frac{(s_{i+k} - s)\, \mathcal{N}_{i+1,k-1}(s)}{s_{i+k} - s_{i+1}} \tag{5.1.3}$$

where s_i are the values of the knot vector $\mathbf{s} = (s_1; s_2; \ldots; s_{n+1})$ with $s_i < s_{i+1}$; moreover, $n+1$ denotes the number of control points. It is worth noting that if the number of control points and the order of the basis functions are the same, then a NURB curve is equivalent to a Bézier curve [115].

For later use, the first derivative of a NURB curve can be expressed formally as

$$\frac{\partial P(s)}{\partial s} = \sum_{i=1}^{n+1} B_i \left(\frac{w_i \dot{\mathcal{N}}_{i,k}(s)}{\sum_{i=1}^{n+1} w_i \mathcal{N}_{i,k}(s)} - \frac{w_i \mathcal{N}_{i,k}(s) \sum_{i=1}^{n+1} w_i \dot{\mathcal{N}}_{i,k}(s)}{(\sum_{i=1}^{n+1} w_i \mathcal{N}_{i,k})^2} \right) \tag{5.1.4}$$

where $\dot{\mathcal{N}}_{i,k}$ is the first derivative of the basis function with respect to s.

5.1.1 Properties of NURBs

Important properties of NURB curves include:

1. Each rational basis function, $\mathcal{R}_{i,k}(s)$, is non-negative for all values of real number s.

2. For each real value of s, the sum of the rational basis functions, $\mathcal{R}_{i,k}(s)$, equals one:

$$\sum_{i=1}^{n+1} \mathcal{R}_{i,k}(s) = 1.$$

3. Except for $\mathcal{R}_{1,k}(s)$, all other rational basis functions have only one maximum.

4. A NURB curve of order k is C^{k-2} continuous.

5. A NURB curve generally follow the shape of its control polygon.

6. For $w_i > 0$ a NURB curve lies within the union of convex hulls formed by k successive control polygons.

7. An affine transformation map is applied to a NURB curve by applying the transformation to its control polygon.

As noted above, Bézier polynomials are a special case of NURB curves, and thus using NURB curves in place of Bézier polynomials allows more solutions when designing gaits via optimization, as in Chapter IV, which may result in a lower value of the cost function.

5.2 HZD Optimization with NURB Curves

Recall that in Chapter IV, the HZD gait design method based on virtual constraints and parameter optimization was explained. Here, the algorithm is modified for the vector of parameters α associated with NURB equations.

5.2.1 The vector of free parameters

The set of decision parameters is $\{q^-_{zero}; \dot{q}^-_{zero}; \mathbf{B}; \mathbf{w}; \mathbf{N}; k; n\}$, where $q^-_{zero} \in \mathcal{Z}$ and $\dot{q}^-_{zero} \in \mathcal{TZ}$ are the unactuated variables right before the moment of impact in the *Lagrangian* model. **B, w, N** $\in \mathbb{R}^{6\times(n+1)}$ are the matrices of control points, weight numbers and basis functions:

$$\mathbf{B} = \begin{bmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,n+1} \\ B_{2,1} & B_{2,2} & \cdots & B_{2,n+1} \\ \vdots & \vdots & \ddots & \vdots \\ B_{6,1} & B_{6,2} & \cdots & B_{6,n+1} \end{bmatrix}, \tag{5.2.1}$$

$$\mathbf{w} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n+1} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n+1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{6,1} & w_{6,2} & \cdots & w_{6,n+1} \end{bmatrix}, \tag{5.2.2}$$

and

$$\mathbf{N} = \begin{bmatrix} \mathcal{N}_{1,1,k}(s) & \mathcal{N}_{1,2,k}(s) & \cdots & \mathcal{N}_{1,n+1,k}(s) \\ \mathcal{N}_{2,1,k}(s) & \mathcal{N}_{2,2,k}(s) & \cdots & \mathcal{N}_{2,n+1,k}(s) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{N}_{6,1,k}(s) & \mathcal{N}_{6,2,k}(s) & \cdots & \mathcal{N}_{6,n+1,k}(s) \end{bmatrix}. \tag{5.2.3}$$

In the above matrices, $B_{i,j}$, $w_{i,j}$ and $\mathcal{N}_{i,j,k}(s)$ are associated with the j-th control vertex of the desired trajectory corresponding to the i-th control variable in the vector of variables to be controlled, $h_0(q_s)$. k is the order of the basis function in (5.1.3) and $n+1$ is the number of control points.

5.2.2 Relating full state to the zero dynamics

The single support phase state vector at the end of the step $x_s^- = (q_s^-; \dot{q}_s^-) \in \mathcal{TQ}_s$ is obtained using the decision parameters after ensuring that the decoupling matrix is invertible and the zero dynamics manifold $\mathcal{Z}$ is nonempty. Denote by q_s^-, $\dot{q}_s^-$, q_s^+ and $\dot{q}_s^+$ the configuration variables at the end and at the beginning of a step. When the virtual constraints are enforced using a feedback controller, (4.1.16) implies that the controlled variables can be written in terms of $B_{i,j}$, $w_{i,j}$ and basis functions $\mathcal{N}_{i,j,k}$, namely,

$$\begin{cases} q_{c,i} = \displaystyle\sum_{i=1}^{n+1} B_{i,j}\, \mathcal{R}_{i,j,k}(s) \\ \dot{q}_{c,i} = \displaystyle\sum_{i=1}^{n+1} B_{i,j}\, \frac{\partial \mathcal{R}_{i,j,k}(s)}{\partial s} \frac{\partial s}{\partial \theta} \frac{\partial \theta}{\partial q_z} \dot{q}_z \end{cases}$$

Using (4.1.8), q_s and $\dot{q}_s$ are,

$$q_s = \mathcal{T}_0 \bar{q} + \mathcal{T}_1$$

$$\dot{q}_s = \mathcal{T}_0 \dot{\bar{q}}$$

where $\bar{q}$ is given by (4.1.6).

Because s is computed from q_{zero}, (4.1.8) suggests an insertion map $\mathcal{F} : \mathcal{Z}_{\mathbf{B},\mathbf{N},\mathbf{w},n,k} \mapsto \mathcal{Q}_s$ from a zero dynamics manifold corresponding to $\mathbf{B}, \mathbf{N}, \mathbf{w}, n, k$ to the manifold of stance configuration variables. In addition, a projection map $\pi : \mathcal{Q}_s \mapsto \mathcal{Z}_{\mathbf{B},\mathbf{N},\mathbf{w},n,k}$ follows from (4.1.8). Using $\mathcal{F}$, π, Δ_s^{Right} and $\mathcal{T}_{q_z}\mathcal{F}$, [1] it is possible to obtain q_z^+ and $\dot{q}_z^+$ as illustrated in Fig. 5.1. Now, employing (4.1.22) and (4.3.3), the energetic efficiency of walking characterized in the form of cmt is minimized, subjected to a series of equality and inequality constraints.

$$\begin{tikzcd}
q_z^- \arrow[r, Rightarrow, "\mathcal{F}"] & q_s^- \arrow[r, Rightarrow, "\Delta_{q_s}^{Right}"] & q_s^+ \arrow[r, Rightarrow, "\pi"] & q_z^+ \\
\dot{q}_z^- \arrow[r, Rightarrow, "\mathcal{T}_{q_z^-}\mathcal{F}"] & \dot{q}_s^- \arrow[r, Rightarrow, "\Delta_{\dot{q}_s}^{Right}"] & \dot{q}_s^+ \arrow[r, Rightarrow, "\mathcal{T}_{\Delta_{\dot{q}_s}^{Right}\dot{q}_s^-}\pi"] & \dot{q}_z^+
\end{tikzcd}$$

Figure 5.1: Diagram shows $\mathcal{F} : \mathcal{Z}_{\mathbf{B},\mathbf{N},\mathbf{w},n,k} \mapsto \mathcal{Q}_s$ and $\pi : \mathcal{Q}_s \mapsto \mathcal{Z}_{\mathbf{B},\mathbf{N},\mathbf{w},n,k}$.

5.2.3 Selecting the orders

The number of control points in the control polygon and the order of the basis functions determine the local flexibility of a NURB curve. This point will be discussed in Section 5.3. Lower values for the order k of the basis functions results in more local behavior of a NURB curve, meaning that the control points have a smaller domain of effectiveness around the

[1]Is equivalent to the derivative of the differentiable map $\mathcal{F}$ at the point $q_z \in \mathcal{Z}_{\mathbf{B},\mathbf{N},\mathbf{w},n,k}$, $\mathcal{T}_{q_z}\mathcal{F} : \mathcal{T}_{q_z}\mathcal{Z}_{\mathbf{B},\mathbf{N},\mathbf{w},n,k} \mapsto \mathcal{T}_{\mathcal{F}(q_z)}\mathcal{Q}_s$.

control points. However, as the order decreases, a NURB curve loses smoothness; indeed, it is C^{k-2}. In practice, it is important to maintain a balance between the smoothness of a NURB curve and the local behavior of a NURB curve. In order to parametrize the desired trajectories of the virtual holonomic constraints, an order of $k = 3$ is chosen, since it is the lowest possible order that results in a differentiable trajectory.[2] Six control points are used therefore, if $k = 6$, the NURB curve reflects the behavior of a Bézier curve.

5.2.4 NURB-based parametrization of desired trajectories

The parameterizations of the desired trajectories for the holonomic constraints based on NURB curves requires the determination of the basis functions $\mathcal{N}i, j, k$ and knot vector $\mathbf{s}$. Recall from (5.1.3) that the basis functions are recursively defined. The dependency diagram shown below depicts the *Cox-de Boor* algorithm in a graphical sense. The diagram shows that in order to compute the basis function $\mathcal{N}i, 1, 3$, $\mathcal{N}i, 1, 2$ and $\mathcal{N}i, 2, 2$ must already have been computed. And, in order to compute $\mathcal{N}i, 1, 2$, $\mathcal{N}i, 1, 1$ and $\mathcal{N}i, 2, 1$ must first be computed. Hence, the basis functions that fall inside a triangle under the basis function $\mathcal{N}i, j, k$ are used in the recursive *Cox-de Boor* algorithm.

$$\begin{array}{llllllll}
\mathcal{N}_{i,1,3}\cdots & \mathcal{N}_{i,2,3}\cdots & \mathcal{N}_{i,3,3}\cdots & \mathcal{N}_{i,4,3}\cdots & \mathcal{N}_{i,5,3}\cdots & \mathcal{N}_{i,6,3} & & \\
\quad\vdots & \quad\vdots & \quad\vdots & \quad\vdots & \quad\vdots & \quad\vdots & & \\
\mathcal{N}_{i,1,2}\cdots & \mathcal{N}_{i,2,2}\cdots & \mathcal{N}_{i,3,2}\cdots & \mathcal{N}_{i,4,2}\cdots & \mathcal{N}_{i,5,2}\cdots & \mathcal{N}_{i,6,2}\cdots & \mathcal{N}_{i,7,2} & \\
\quad\vdots & \quad\vdots & \quad\vdots & \quad\vdots & \quad\vdots & \quad\vdots & \quad\vdots & \\
\mathcal{N}_{i,1,1}\cdots & \mathcal{N}_{i,2,1}\cdots & \mathcal{N}_{i,3,1}\cdots & \mathcal{N}_{i,4,1}\cdots & \mathcal{N}_{i,5,1}\cdots & \mathcal{N}_{i,6,1}\cdots & \mathcal{N}_{i,7,1}\cdots & \mathcal{N}_{i,8,1}
\end{array}$$

In addition, the dependency diagram and (5.1.2) show that in order to compute $\mathcal{N}_{i,8,1}$,

[2]This is because we are using outputs that have relative degree 2.

the knot points s_8 and s_9 are required. Hence, $\mathbf{s} = (s_1; \ldots; s_9)$ is defined such that $s_i = \frac{i}{9}$. To initiate the recursive computation of the basis functions, for $0 \leq s < s_1$,

$$\begin{aligned}
&\mathcal{N}_{i,1,1}(s) = 1 \\
&\mathcal{N}_{i,1,2}(s) = s \\
&\mathcal{N}_{i,1,3}(s) = \frac{(s)^2}{2} \\
&\mathcal{N}_{i,i,1}(s) = 0 && i \neq 1 \\
&\mathcal{N}_{i,i,2}(s) = 0 && i \neq 1 \\
&\mathcal{N}_{i,i,3}(s) = 0 && i \neq 1
\end{aligned}$$

and for $s_1 \leq s < s_2$,

$$\begin{aligned}
&\mathcal{N}_{i,2,1}(s) = 1 \\
&\mathcal{N}_{i,i,1}(s) = 0 && i \neq 2 \\
&\mathcal{N}_{i,1,2}(s) = (s_2 - s) \\
&\mathcal{N}_{i,2,2}(s) = (s_1 - s) \\
&\mathcal{N}_{i,i,2}(s) = 0 && i \neq 1,2 \\
&\mathcal{N}_{i,1,3}(s) = \frac{s(s_2 - s)}{2} \\
&\quad + \frac{(s_3 - s)(s - s_1)}{2} \\
&\mathcal{N}_{i,2,3}(s) = \frac{(s - s_1)^2}{2} \\
&\mathcal{N}_{i,i,3}(s) = 0 && i \neq 1,2,3
\end{aligned}$$

Using the dependency diagram and the *Cox-de Boor* algorithm, the values of the basis functions for the remaining regions are obtained, $s_i \leq s < s_{i+1}$. The parametrized desired trajectory is given in matrix form as follows:

$$h_d = \frac{1}{\sum_{j=1}^{n+1} w_{i,j}\mathcal{N}_{i,j,3}} \begin{pmatrix} w_{i,1}\mathcal{N}_{i,1,3} & \ldots & w_{i,6}\mathcal{N}_{i,6,3} \end{pmatrix} (\mathbf{B})'$$

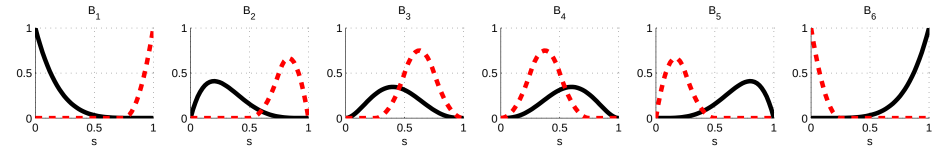
where **B** represents a matrix containing the control points for the desired trajectory.

5.3 NURB Curves vs. Bézier Curves

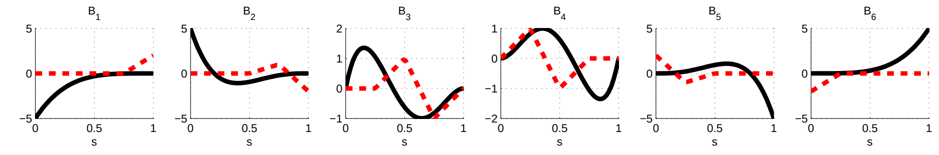
As [116] formally proves, Bézier polynomials are special cases of NURB curves. The first two of the following subsections show qualitatively how the use of NURB curves may result in more efficient walking gaits than Bézier polynomials, while the third subsection illustrates a quantitative advantage.

5.3.1 Local Action

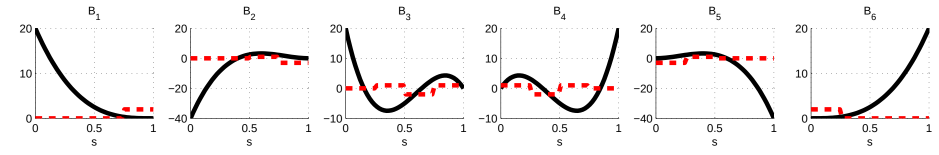
As the order of the NURB basis functions $\mathcal{N}_{i,j,k}(s)$ decreases, the control points affect the curve over smaller ranges of s (i.e., they act more locally), which has advantages. For example, Fig. 5.2 shows $\mathcal{N}_{i,j,k}(s)$, $\frac{\partial \mathcal{N}_{i,j,k}(s)}{\partial s}$ and $\frac{\partial^2 \mathcal{N}_{i,j,k}(s)}{\partial s^2}$ for $k = 6$ (solid line) and $k = 3$ (dashed line), with $n = 6$. When $k = 3$, B_4, B_5 and B_6 make no contribution to the shape of the desired trajectory at $s = 0$ because the basis functions corresponding to those control points vanish at $s = 0$. On the other hand, for $k = 6$, all of the control points contribute to the value of the trajectory at $s = 0$. Figure 5.2 also shows that when $k = 6$, at the boundaries where the control points B_1, B_2, B_5 and B_6 affect $h_d(q(s))$, their corresponding basis functions have large values for $\frac{\partial \mathcal{N}_{i,j,k}(s)}{\partial s}$ and $\frac{\partial^2 \mathcal{N}_{i,j,k}(s)}{\partial s^2}$. This can cause numerical instability of the solutions, as a numerical optimizer tries to adjust the control points to minimize cost while achieving periodicity of the solution.



(a) $\mathcal{N}_{i,j,k}(s)$



(b) $\frac{\partial \mathcal{N}_{i,j,k}(s)}{\partial s}$

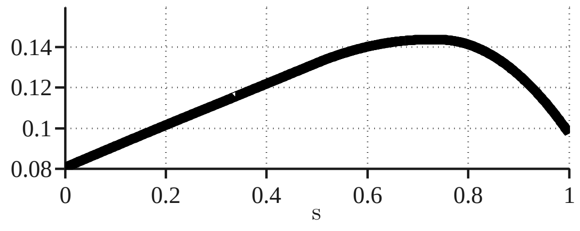


(c) $\frac{\partial^2 \mathcal{N}_{i,j,k}(s)}{\partial s^2}$

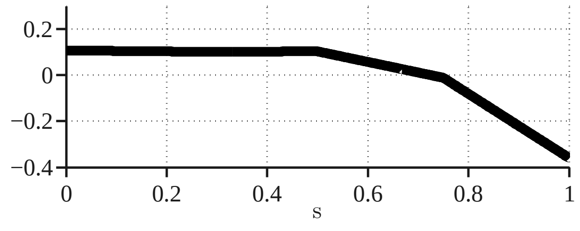
Figure 5.2: **(a)** Basis functions, **b)** first derivative of the basis functions, and **(c)** second derivative of the basis functions, all with respect to the timing variable s for each control point B_i when the order k is six (solid line) and three (dashed line).

5.3.2 Subphase

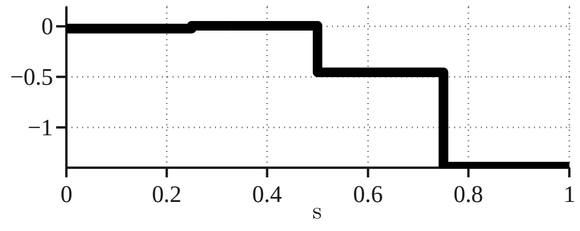
Because a Bézier polynomial is an analytic function (in the sense of globally convergent Taylor series), if it (or one of its derivatives) is constant over an open set, then it (or the derivative) is constant everywhere. When $k < n$, a NURB is no longer an analytic function, and thus it becomes possible to have non-trivial regions where the curve or one of its derivatives is constant, while the NURB is not globally constant. These regions are informally referred to as "subphases." Figure 5.3 illustrates this for one of the trajectories associated with the right leg knee angle, in the regions for $s \in [0\ 0.25]$, $s \in [0.25\ 0.5]$, $s \in [0.5\ 0.75]$ and $s \in [0.75\ 1]$. This figure shows that the velocity of the knee angle is commanded to a small value at the beginning of the stance phase. Consequently, the energy of the impact forces is directed into the springs. The energy stored in the springs is released later in the gait.



(a) $h_d(s)$



(b) $\frac{\partial h_d(s)}{\partial s}$



(c) $\frac{\partial^2 h_d}{\partial s^2}$

Figure 5.3: **(a)** Desired trajectory for the right (stance) leg knee angle q_{grR}^{Knee}, **(b)** first derivative of the desired trajectory and **(c)** the second derivative of the desired trajectory when the order of the NURB basis function is 3.

5.3.3 cmt

Figure 5.4 shows the computed cmt versus speed when Bézier and NURB are used as desired trajectories for the holonomic constraints, circles and squares, respectively. For comparison purposes, according to [35], the cmt of a human is estimated from experimental data to be 0.05 at a speed of 1 $[\frac{m}{s}]$, while according to [36], the experimentally estimated CMT of the Cornell Biped is 0.04 at a speed of 0.6 $[\frac{m}{s}]$. Interpolating a cubic polynomial through the NURB-based simulation data of MARLO results in a minimum cmt of 0.05 at

0.7 $[\frac{m}{s}]$. At $1[\frac{m}{s}]$, the simulated cmt of MARLO is 0.096, which means the robot would require approximately twice the power of a human when walking at $1[\frac{m}{s}]$, assuming once again that the harmonic drives are lossless. If the harmonic drives are assumed to be 70% efficient on average, then the motors in the sagittal plane must produce approximately 43% more power when realizing the walking gaits of Fig. 5.4. When this is taken into account, MARLO's estimated cmt at $0.7[\frac{m}{s}]$ is 0.071, or approximately one and a half times that of a human, while at $1[\frac{m}{s}]$, cmt would 0.13, or approximately two and a half times that of a human.

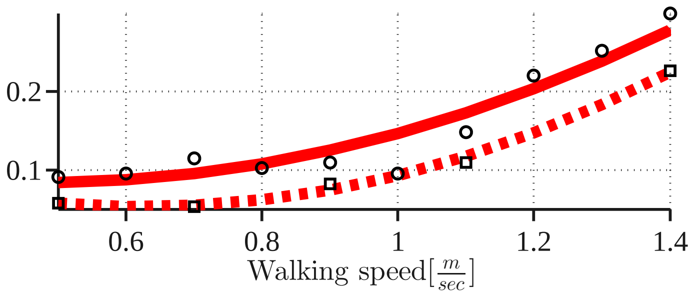


Figure 5.4: Circles represent computed cmt using Bézier curves for the desired trajectories versus walking speed from 0.5 to 1.4 $[\frac{m}{s}]$. The thick solid line is a cubic interpolation of these data. Squares represent computed cmt using NURB curves versus walking speeds 0.5, 0.7, 0.9, 1.1 and 1.4 $[\frac{m}{s}]$. The dashed line is a cubic interpolation of these data. Losses at the harmonic drives are ignored.

5.4 NURB Curves with Control Objectives

In addition to being useful in gait design via HZD optimization, the NURB curves introduced in this chapter have implications for achieving control objectives. For example, suppose an optimized periodic orbit $\mathcal{O}$ has been designed for the standard set of output functions in (4.2.4). Now let $\bar{h}_0$ represent a different choice of controlled outputs. If $\bar{h}_d$ can be found such that $\bar{y} = \bar{h}_0 - \bar{h}_d$ still vanishes on the periodic orbit, then $\bar{y}$ can still be used for implementing a feedback controller without re-doing the optimization. A straightforward way to obtain $\bar{h}_d$ is to regress (via least squares) a set of splines against $\bar{h}_0$ evaluated on the periodic orbit $\mathcal{O}$; see Section 6.5 of [109]. With Bézier curves, high degree polynomials are required, and experience has shown that this often leads to singularities in the decoupling matrix (see Chapter VI). In cases that we have investigated, NURBs have allowed us to avoid such singularity problems.

Two choices for controlled outputs are illustrated below, namely, the distance between the CoM and swing-leg-end and the orientation of the torso defined by roll, pitch or yaw. In addition, regression of experimentally measured actuator torque versus a gait-timing variable is used as a feed-forward term.

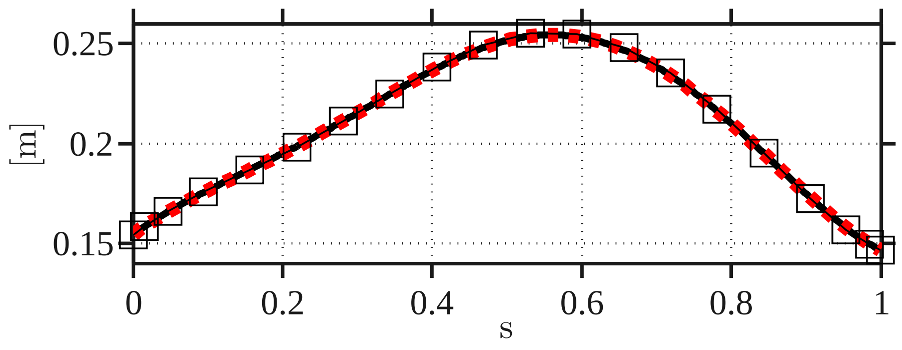
Reference [105] proposes a virtual constraint that specifies the distance between CoM and the swing-leg-end point as one of the variables to be controlled. This virtual constraint will be covered in greater detail in Chapter VI. The desired trajectories for this distance-based virtual holonomic constraint were parametrized using NURB curves with twenty basis functions of order $k = 3$. The vector **s** was selected as

$$\mathbf{s} = (s_1; \ldots; s_{23}),$$

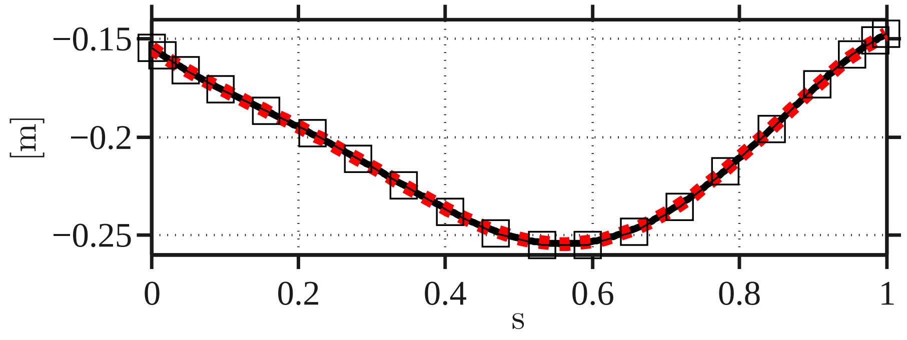
with $s_i = \frac{i}{18}$. Using the dependency diagram, the basis functions are computed recursively. Recalling (5.1.1), the desired trajectories for the distance-holonomic constraint are formally defined in terms of the basis functions, control polygon, weight vector and the knot vector.

$$h_d^{r|^{SE}_{CoM}} = \frac{1}{\sum_{j=1}^{20} w_{1,j}^{r|^{SE}_{CoM}} \mathcal{N}_{1,j,3}^{r|^{SE}_{CoM}}} \begin{pmatrix} w_{1,1}^{r|^{SE}_{CoM}} \mathcal{N}_{1,1,3}^{r|^{SE}_{CoM}} & \ldots & w_{1,20}^{r|^{SE}_{CoM}} \mathcal{N}_{1,20,3}^{r|^{SE}_{CoM}} \end{pmatrix} (\mathbf{B}^{r|^{SE}_{CoM}})' \quad (5.4.1)$$

Figure 5.5 illustrates the regressed NURB curve to the curve of the projected horizontal distance between the CoM and swing-leg-end; the red dashed curve represents the fitted NURB curve and the square marks represent the control points.



(a)

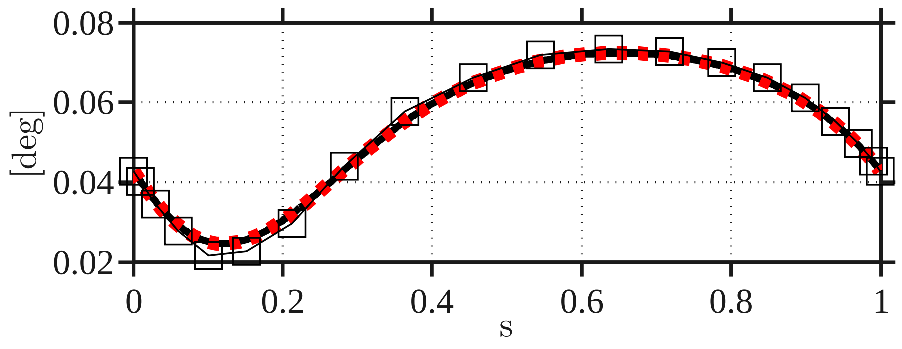


(b)

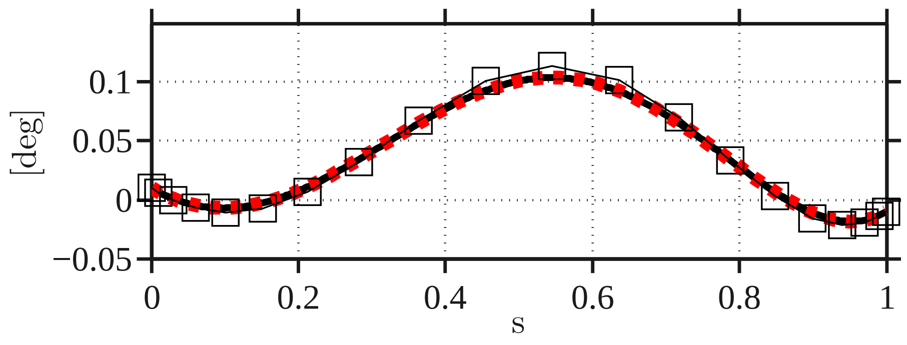
Figure 5.5: Regressed $r|_{CoM}^{SE}$ versus gait-timing variable when **(a)** right leg stance, and **(b)** left leg stance.

The same regressing algorithm was applied to parametrize the torso orientation angles, roll, pitch and yaw against the gait-timing variable. These parametrized orientations define the orientation of the torso with respect to a world frame and are shown in Fig. 5.6.

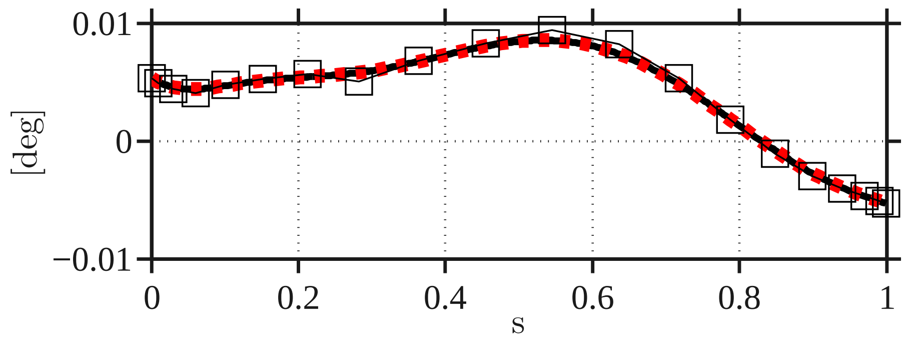
Figures 5.7 and 5.8 illustrate the regressed experimental actuators torque over gait-timing variable, which are used as feed-forward terms in the walking controller algorithm.



(a) pitch

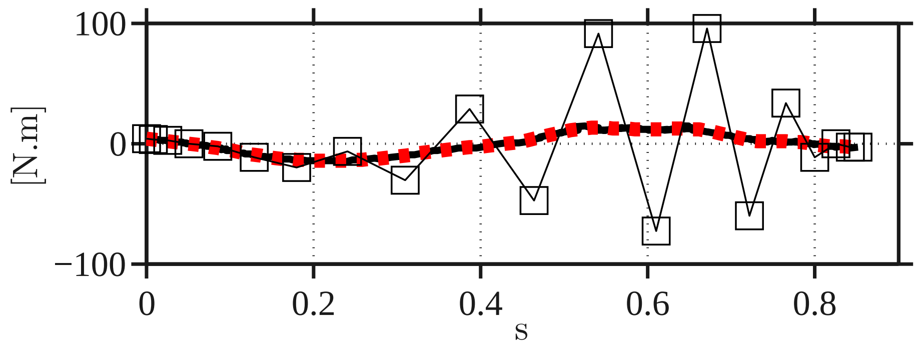


(b) roll

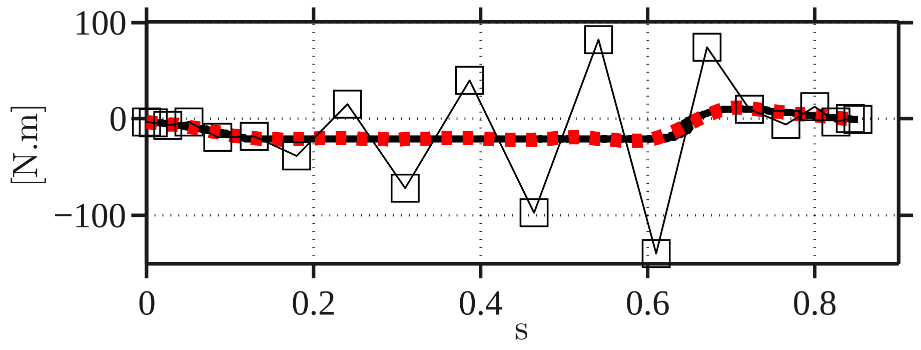


(c) yaw

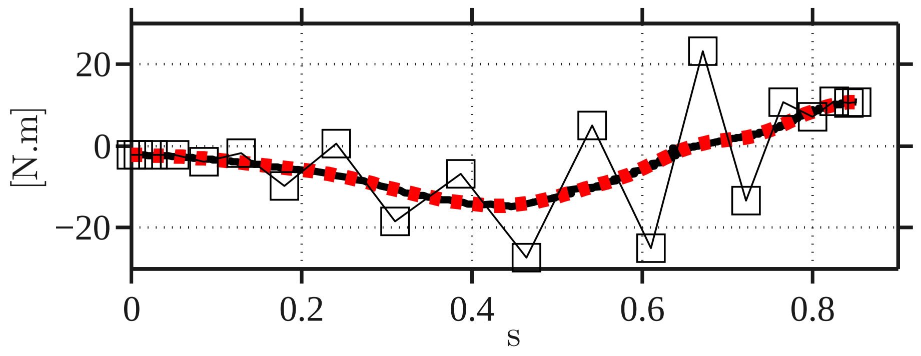
Figure 5.6: Regressed optimal torso angles versus gait-timing variable for **(a)** pitch, **(b)** roll and **(c)** yaw over a single step, when right leg is stance; the red dashed curve represents the fitted NURB curve; the square marks represent the control points.



(a) u_{1R}

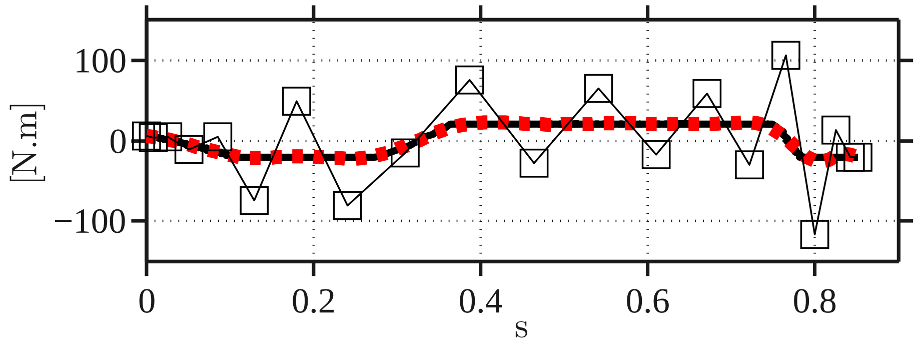


(b) u_{2R}

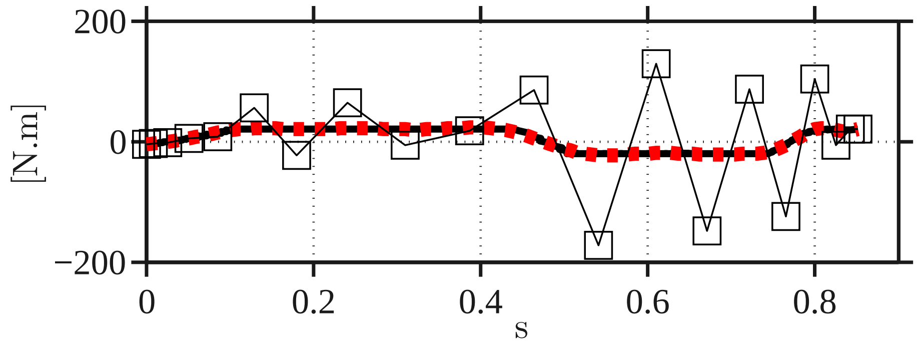


(c) u_{3R}

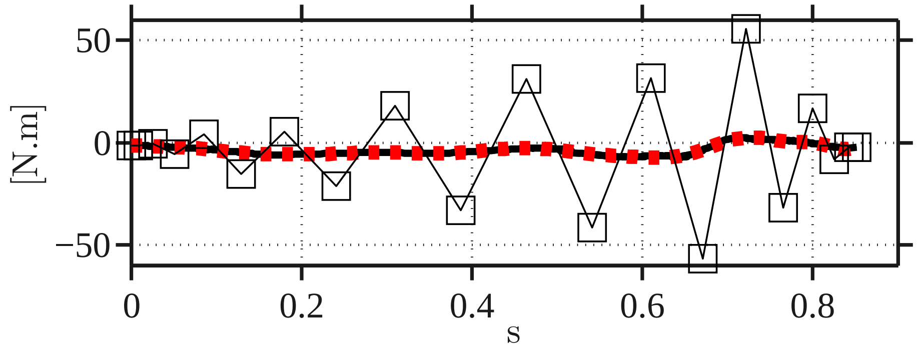
Figure 5.7: Regressed experimental actuation torques versus gait-timing variable for **(a)** u_{1R}, **(b)** u_{2R} and **(c)** u_{3R} over a single step; the red dashed curve represents the fitted NURB curve, the square marks represent the control points.



(a) u_{1L}



(b) u_{2L}



(c) u_{3L}

Figure 5.8: Regressed experimental actuation torques versus gait-timing variable for **(a)** u_{1L}, **(b)** u_{2L} and **(c)** u_{3L} over a single step; the red dashed curve represents the fitted NURB curve; the square marks represent the control points.

CHAPTER VI

Feedback Control

In this chapter an “input-output linearizing controller,” which is the main control loop on the robot, is presented and the choices of the virtual constraints and their effects on the stability of the walking gaits are presented. Two choices for the virtual constraints, one similar to Chapter IV and the other as proposed in [105], both result in an unstable walking gait. Stability of the gait is characterized with the help of the Poincaré method. Stability is achieved after applying an “event-based controller.

The controllers developed in this chapter are simulated with the 3D model of the robot, assuming point feet and a rigid ground model.

6.1 Input Output Linearizing Controller

Each of the gaits presented in Chapter IV comes with a set of outputs (i.e., virtual constraints), which vanish on the periodic orbit traced out by the robot’s states over the periodic walking motion. Let α^* denote the vector of the free parameters resulting from the optimization of the virtual constraints, giving

$$y = h(q_s, \alpha^*) = h_0(q_s) - h_d(\theta(q_s), \alpha^*). \tag{6.1.1}$$

A feedback controller is required to impose the constraints by driving the outputs to zero when the robot is away from the periodic orbit. Designing a feedback policy that enforces the virtual constraints introduced in Chapter IV is our first step in the design of the feedback controller.

6.2 Normal Virtual Constraints

Recall that the virtual constraints introduced in Chapter IV are referred to as normal virtual constraints. The proposed input-output linearizing controller enforces the normal virtual constraints. Consider the following output function representing a normal virtual constraint,

$$y = h(q_s). \tag{6.2.1}$$

The first derivative of y along the solutions of the single-support phase model (3.2.26) is simply

$$\dot{y} = \frac{\partial h(q_s)}{\partial q_s}\dot{q}_s, \tag{6.2.2}$$

and the second derivative is

$$\ddot{y} = -\frac{\partial h(q_s)}{\partial q_s}\mathcal{D}_s^{-1}(q_s)\mathcal{H}_s(\dot{q}_s, q_s)+ \tag{6.2.3}$$

$$\frac{\partial}{\partial q_s}\left(\frac{\partial h(q_s)}{\partial q_s}\dot{q}_s\right)\dot{q}_s + \frac{\partial h(q_s)}{\partial q_s}\mathcal{D}_s^{-1}(q_s)\mathcal{B}_s u. \tag{6.2.4}$$

The decoupling matrix is given by

$$\mathcal{A}(q_s, \dot{q}_s) = \frac{\partial h(q_s)}{\partial q_s}\mathcal{D}_s^{-1}(q_s)\mathcal{B}_s, \tag{6.2.5}$$

while u^* is obtained by solving for the input that sets $\ddot{y} = 0$, namely

$$u^*(q_s, \dot{q}_s) = \mathcal{A}^{-1}(q_s, \dot{q}_s) \left(\frac{\partial}{\partial q_s} \left(\frac{\partial h(q_s)}{\partial q_s} \dot{q}_s \right) \dot{q}_s - \frac{\partial h(q_s)}{\partial q_s} \mathcal{D}_s^{-1}(q_s) \mathcal{H}_s(\dot{q}_s, q_s) \right). \tag{6.2.6}$$

Assuming the decoupling matrix is invertible, input-output linearization yields the feedback controller

$$u(x) = u^*(x) - \mathcal{A}(x)^{-1}(\frac{1}{\epsilon^2}\mathcal{K}_P y + \frac{1}{\epsilon}\mathcal{K}_D \dot{y}), \tag{6.2.7}$$

which renders the input-output map linear, namely,

$$\ddot{y} + \frac{1}{\epsilon}\mathcal{K}_D \dot{y} + \frac{1}{\epsilon^2}\mathcal{K}_P y = 0. \tag{6.2.8}$$

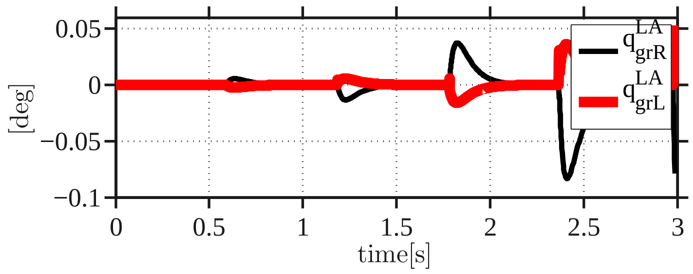
The control gains K_P and K_D are chosen such that the matrix

$$\begin{bmatrix} 0 & I \\ -\mathcal{K}_D & -\mathcal{K}_P \end{bmatrix} \tag{6.2.9}$$

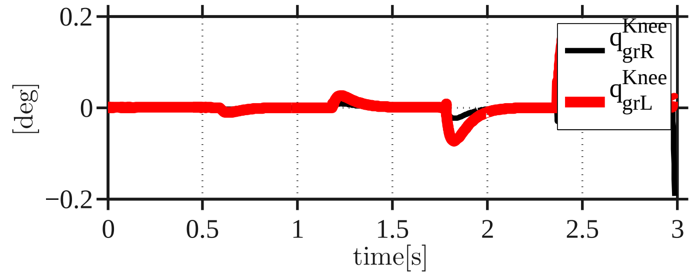
is Hurwitz; $\epsilon > 0$ is a tuning parameter.

6.2.1 Controller Performance

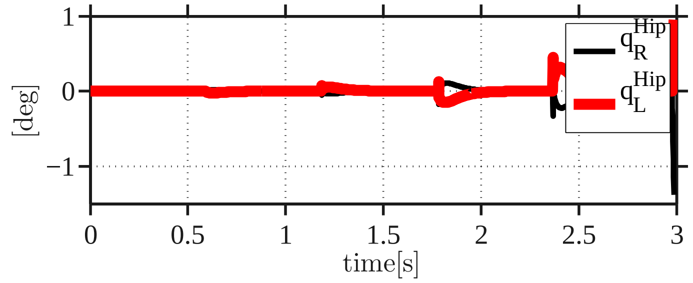
The feedback controller in (6.2.7) was implemented on the 3D model for the nominal walking speed of $1.0[\frac{m}{s}]$. Figure 6.1 illustrates the tracking error y and Fig. 6.2 shows the actuation torques corresponding to the harmonic derive and the hip actuators over five periodically stable steps. The maximum tracking error in leg angles occurs at q_{grR}^{LA} and is $-0.08[deg]$, while the maximum tracking error in knee angles occurs at q_{grL}^{Knee} and is $0.15[deg]$. For the hip, the maximum tracking error is associated with q_L^{Hip} and it is $0.35[deg]$. The maximum actuation torques for u_{1R} and u_{1L} are bounded between $-0.5[N.m]$ and $1.1[N.m]$. The maximum value for u_{2R} is $-1.5[N.m]$ and corresponding to the u_{2L}, it is $1.2[N.m]$. Hip actuators require more torque and the maximum required torque is $-6.5[N.m]$.



(a)

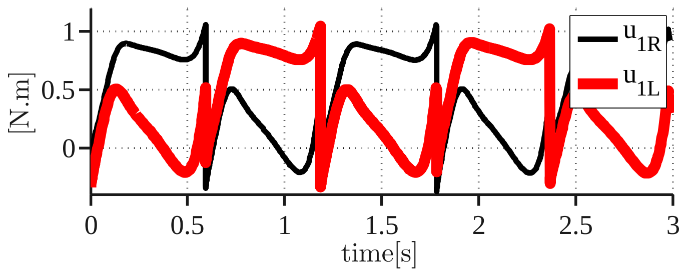


(b)

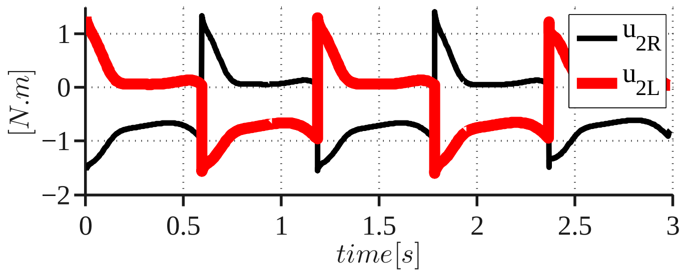


(c)

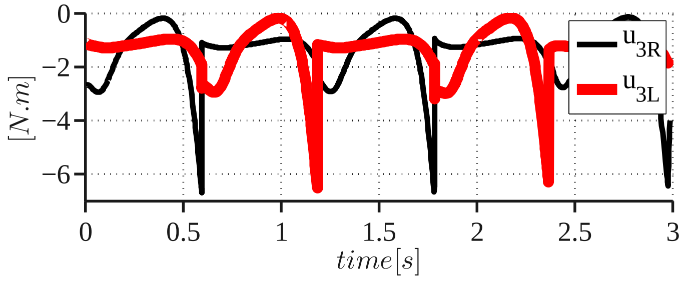
Figure 6.1: Tracking error for the proposed I/O linearizing controller that enforces the normal virtual constraints, **(a)** q^{LA}_{grR}, q^{LA}_{grL}, **(b)** q^{Knee}_{grR}, q^{Knee}_{grL}, and **(c)** q^{Hip}_{R}, q^{Hip}_{L}.



(a)



(b)

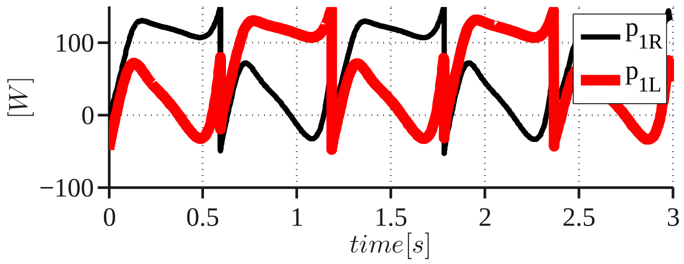


(c)

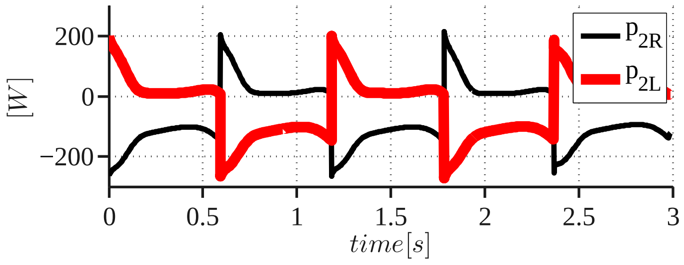
Figure 6.2: Actuation torque for the I/O linearizing controller that enforces the normal virtual constraints, **(a)** u_{1R}, u_{2R}, **(b)** u_{3R}, u_{1L}, and **(c)** u_{2L}, u_{3L}.

Figure 6.3 illustrates the power required by the leg motors, where p_{1R} and p_{1L} reach their maximum value at the end of each step. p_{2R} afnd p_{2L} are maximum at the beginning of each step and fluctuate between $-220[W]$ and $200[W]$. The hip actuators are consume less power, and with each step, their magnitude varies between $-8[W]$ and $3[W]$. Figure 6.4 illustrates the components of the ground reaction force acting at the end of the stance leg. The F_z component has jumps at the end of each step, which reflects the impulsive behavior of the impact model. The maximum magnitude associated with F_x and F_y is

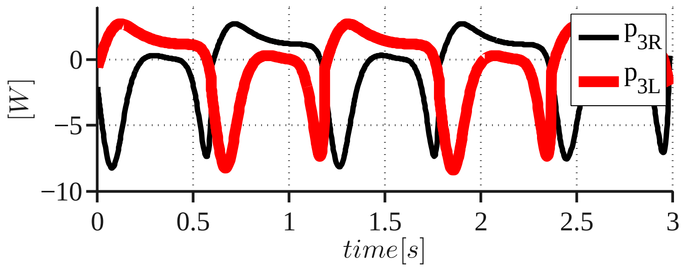
$60[N]$.



(a)



(b)



(c)

Figure 6.3: Actuation power for the I/O linearizing controller that enforces the normal virtual constraints, **(a)** p_{1R}, p_{2R}, **(b)** p_{3R}, p_{1L}, and **(c)** p_{2L}, p_{3L}.

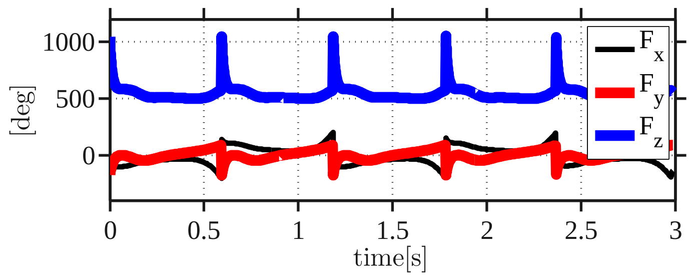


Figure 6.4: Components of the ground reaction force at the end of the stance leg, F_x, F_y, and F_z.

6.2.2 Poincaré Analysis

Recall that the switching surface is $\mathcal{S} := \{(q_s; \dot{q}_s) \in \mathcal{TQ}_s | p^v_{toe} = 0, p^v_{toe} > 0\}$. The stability of the fixed point is checked using a linearized Poincaré map $\mathcal{P} : \Delta\mathcal{S} \to \Delta\mathcal{S}$, where $x_{j+1} = \mathcal{P}(x_j)$ and $\Delta\mathcal{S}$ is the Poincaré section taken at the map of the switching surface by the impact. $x_j = (q_z; q_y; \cdots; \dot{q}_z; \dot{q}_y; \cdots) \in \Delta\mathcal{S}$ denotes the projection of the state vector for the full dynamic model during step j onto $\Delta\mathcal{S}$. Because the Poincaré section $\Delta\mathcal{S}$ is a hyperspace in $\mathbb{R}^{26}$, the Poincaré section has twenty-five independent components. To implement this, define a projection map $\Pi(x) = (q_z; q_y; \cdots; \dot{q}_z; \dot{q}_y; \cdots)$ that eliminates q_x (pitch angle) from the state vector; defining perturbations as $\delta x_j = \Pi(x_j - x^*)$, the linearization of the Poincaré map around the fixed point x^* results in the Jacobian $A = \frac{\partial P}{\partial x}$ of the Poincaré map. In particular,

$$A_j = \frac{\mathcal{P}(x^* + \Delta x_j) - \mathcal{P}(x^* - \Delta x_j)}{2\Delta x_j}, \tag{6.2.10}$$

where A_j is j^{th} column of $A = [A_1, A_2, \cdots, A_{25}]$ and $\Delta x_j = (0; \cdots; 0; \epsilon_j; 0; \cdots; 0)$. Using the output functions explained in (4.2.4) and computing the linearized Poincaré map for $\epsilon = 0.01$, the feedback controller of (6.2.7) makes the zero dynamics manifold invariant and attractive, however, due to the existence of compliant components and a high degree of under actuation, it does not make the zero dynamics manifold hybrid invariant [114]. As expected from [105], calculations show that one eigenvalue has a magnitude greater than one; hence, the gait is unstable under this controller.

6.3 Modified Virtual Constraints

When the normal virtual constraints are applied to the planar model (q_{3R} and q_{3L} removed), a stable gait is achieved. Based on this fact, one suspects that the instability of the 3D model may be due to the roll or yaw motions. From [105], the position of the CoM in the frontal plane is important. If at leg touchdown, the CoM is not between the feet but outside the position of the next supporting foot, the robot will topple sideways. Based on this physical intuition, the control of the variable q_3 (which regulates step width on the swing leg) was replaced by the control of the distance between the swing leg end and the CoM along the frontal direction. The modified virtual constraints are given by

$$h_0^{r|_{CoM}^{SE}}(q_s) = \begin{bmatrix} \dfrac{q_{gr1R} + q_{gr2R}}{2} \\ \dfrac{q_{gr1L} + q_{gr2L}}{2} \\ q_{gr2R} - q_{gr1R} \\ q_{gr2L} - q_{gr1L} \\ q_{3R} \\ p^h_{CoM} - p^h_{SE} \end{bmatrix} = \begin{bmatrix} q^{LA}_{grR} \\ q^{LA}_{grL} \\ q^{Knee}_{grR} \\ q^{Knee}_{grL} \\ q^{Hip}_{R} \\ r|^{SE}_{CoM} \end{bmatrix}, \qquad (6.3.1)$$

where $r|^{SE}_{CoM} = p^h_{CoM} - p^h_{SE}$ is the projected horizontal distance between swing-leg end SE and CoM into the frontal plane.

The optimal distance between the end of the swing leg and the CoM of the robot in the frontal plane is computed and is explained in Section 5.4 using a NURB curve. The modified virtual constraint is written in as,

$$y^{r|^{SE}_{CoM}} = h^{r|^{SE}_{CoM}}(q_s, \alpha^{*r|^{SE}_{CoM}}) = h_0^{r|^{SE}_{CoM}}(q_s) - h_d^{r|^{SE}_{CoM}}(\theta(q_s), \alpha^{*r|^{SE}_{CoM}}). \qquad (6.3.2)$$

Considering (6.3.2) and taking first and second time derivatives similar to (6.2.2) and (6.2.4), the decoupling matrix for the modified virtual constraint is given as

$$\mathcal{A}^{r|_{CoM}^{SE}}(q_s, \dot{q}_s) = \frac{\partial h^{r|_{CoM}^{SE}}(q_s)}{\partial q_s} \mathcal{D}_s^{-1}(q_s)\mathcal{B}_s, \tag{6.3.3}$$

while $u^{*r|_{CoM}^{SE}}$ is obtained by solving for the input that sets $\ddot{y}^{r|_{CoM}^{SE}} = 0$, giving

$$\begin{aligned} u^{*r|_{CoM}^{SE}}(q_s, \dot{q}_s) =& (\mathcal{A}^{r|_{CoM}^{SE}})^{-1}(q_s, \dot{q}_s) \left(\frac{\partial}{\partial q_s} \left(\frac{\partial h^{r|_{CoM}^{SE}}(q_s)}{\partial q_s} \dot{q}_s \right) \dot{q}_s - \right. \\ & \left. \frac{\partial h^{r|_{CoM}^{SE}}(q_s)}{\partial q_s} \mathcal{D}_s^{-1}(q_s)\mathcal{H}_s(\dot{q}_s, q_s) \right). \end{aligned} \tag{6.3.4}$$

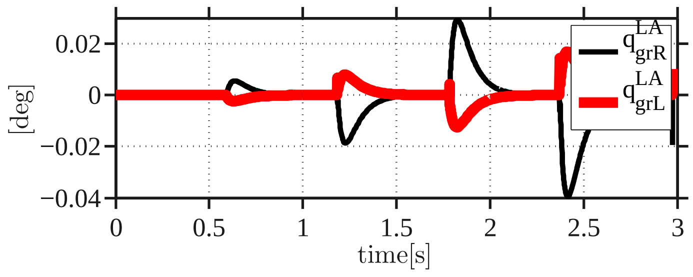
Similar to (6.2.7), at points where the decoupling matrix is invertible, input-output linearization yields the feedback controller

$$\begin{aligned} u^{r|_{CoM}^{SE}}(x) =& u^{*r|_{CoM}^{SE}}(x) - \mathcal{A}^{r|_{CoM}^{SE}}(x)^{-1}\left(\frac{1}{(\epsilon^{r|_{CoM}^{SE}})^2}\mathcal{K}_P^{r|_{CoM}^{SE}} y^{r|_{CoM}^{SE}} + \right. \\ & \left. \frac{1}{\epsilon^{r|_{CoM}^{SE}}}\mathcal{K}_D^{r|_{CoM}^{SE}} \dot{y}^{r|_{CoM}^{SE}}\right), \end{aligned} \tag{6.3.5}$$

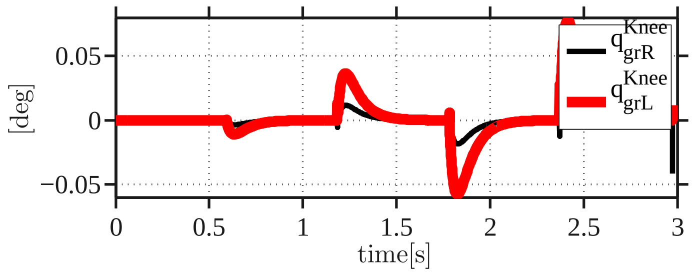
where $u^{r|_{CoM}^{SE}}(x)$ enforces the modified virtual constraints, and $\mathcal{K}_P^{r|_{CoM}^{SE}}$, $\mathcal{K}_D^{r|_{CoM}^{SE}}$ and $\epsilon^{r|_{CoM}^{SE}}$ are the control parameters for the modified virtual constraints.

6.3.1 Controller Performance

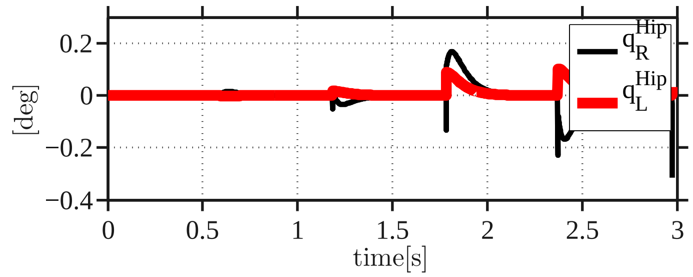
The feedback controller in (6.3.5) was implemented on the 3D model for the nominal walking speed of $1.0[\frac{m}{s}]$. Figure 6.5 illustrates the tracking error y and Fig. 6.6 shows the actuation torques corresponding to the harmonic drive and the hip actuators during five periodically stable walking steps. The maximum tracking error in the leg angles takes place at q_{grR}^{LA} and is $-0.04[deg]$, while the maximum tracking error in the knee angles takes place at q_{grL}^{Knee} and is $0.07[deg]$. The tracking error is larger in the hip actuators, where the maximum tracking error is $-0.18[deg]$ and is associated with q_R^{Hip}. For the actuation torques, the maximum occurs at the hip actuators, with a value of $-6.7[N.m]$. The magnitude of the actuation torques associated with the leg harmonic drive is $-1.5[N.m]$, and it occurs at u_{2L} or u_{2R}.



(a)

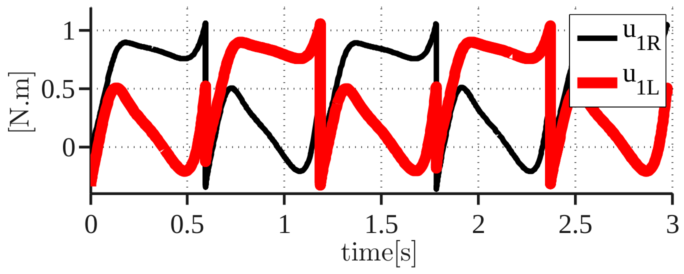


(b)

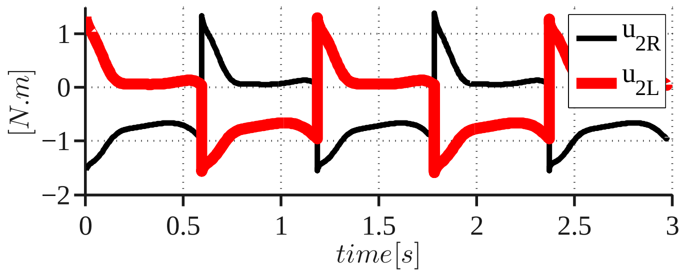


(c)

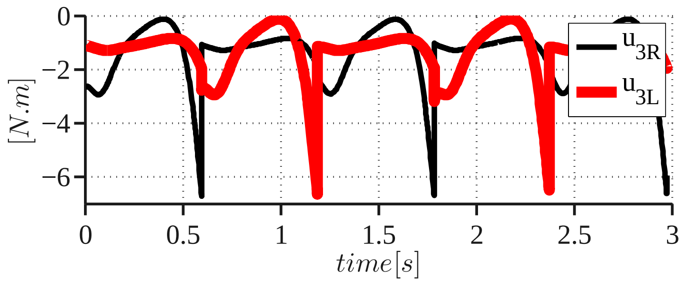
Figure 6.5: Tracking error for the proposed I/O linearizing controller that enforces the modified virtual constraints, **(a)** q_{grR}^{LA}, q_{grL}^{LA}, **(b)** q_{grR}^{Knee}, q_{grL}^{Knee}, and **(c)** q_{R}^{Hip}, q_{L}^{Hip}.



(a)



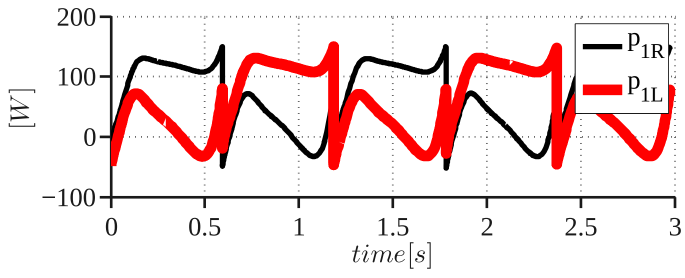
(b)



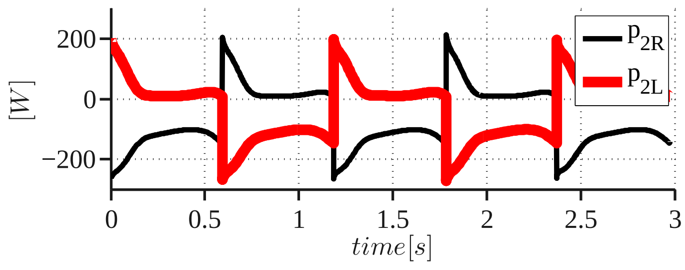
(c)

Figure 6.6: Actuation torque for the I/O linearizing controller that enforces the modified virtual constraints, **(a)** u_{1R}, u_{2R}, **(b)** u_{3R}, u_{1L}, and **(c)** u_{2L}, u_{3L}.

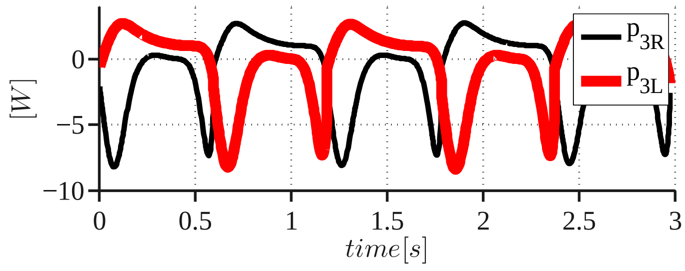
Figure 6.7 demonstrates the power required by the motor actuators during five periodically stable steps. Maximum torque occurs at u_{2R} or u_{2L}. At the first step, where the right leg is the stance leg, p_{2R} performs negative-work, meaning the corresponding harmonic drive is generating torque in the opposite direction from the motion. The minimum power demand takes place at the hip actuators and varies between $-8[N.m]$ and $5[N.m]$. Figure 6.8 shows the components of the ground reaction force acting at the end of the stance leg, where the normal component has jumps at the end of each step because of the impact map. F_z has an average value of $500[N]$; the tangential components are bounded between $-40[N]$ and $50[N]$. The relatively large difference between the magnitude of the tangential forces and the normal force is beneficial for the periodic walking because slippage is less likely.



(a)



(b)



(c)

Figure 6.7: Actuation power for the I/O linearizing controller that enforces the modified virtual constraints, **(a)** p_{1R}, p_{2R}, **(b)** p_{3R}, p_{1L}, and **(c)** p_{2L}, p_{3L}.

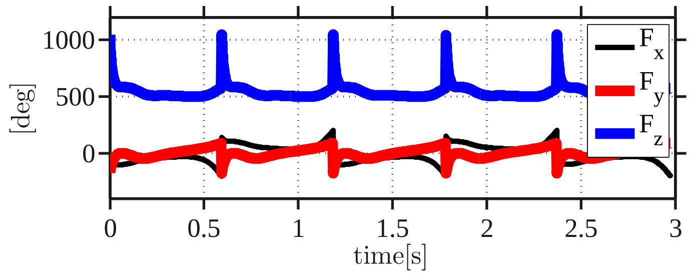


Figure 6.8: Components of the ground reaction force at the end of the stance leg, F_x, F_y, and F_z.

6.3.2 Poincaré Analysis

The linearized Poincaré map for the modified virtual constraint was computed following the procedure from the previous section. The calculations show that one eigenvalue still has a magnitude greater than one; hence, the gait is unstable under this controller.

6.4 Event-Based Controller

Next, an event-based controller is designed and integrated with the continuous input-output linearizing controller (6.3.5). To achieve this goal, the modified virtual constraint in (6.3.2) is augmented with a correction term,

$$\begin{aligned}\tilde{y}^{r|^{SE}_{CoM}} =& h^{r|^{SE}_{CoM}}(q_s, \alpha^{*r|^{SE}_{CoM}}) = \\ & h_0^{r|^{SE}_{CoM}}(q_s) - h_d^{r|^{SE}_{CoM}}(\theta(q_s), \alpha^{*r|^{SE}_{CoM}}) - \\ & h_{\text{corr}}^{r|^{SE}_{CoM}}(q_s, \beta^{r|^{SE}_{CoM}}).\end{aligned} \tag{6.4.1}$$

where $\tilde{y}^{r|^{SE}_{CoM}}$ is the augmented modified virtual constraint, and $\beta^{r|^{SE}_{CoM}}$ is the vector of corrective terms that are held constant over two steps and updated at the left-to-right impact event. Following a similar procedure as (6.2.2), (6.2.4) and (6.2.5), the input-output linearizing controller that enforces virtual constraints in (6.4.1) is given by

$$\begin{aligned}\tilde{u}^{r|^{SE}_{CoM}}(x) =&\tilde{u}^{*r|^{SE}_{CoM}}(x)- \\ &\tilde{\mathcal{A}}^{r|^{SE}_{CoM}}(x)^{-1}(\frac{1}{(\tilde{\epsilon}^{r|^{SE}_{CoM}})^2}\tilde{\mathcal{K}}_P^{r|^{SE}_{CoM}}\tilde{y}^{r|^{SE}_{CoM}}+ \\ &\frac{1}{\tilde{\epsilon}^{r|^{SE}_{CoM}}}\tilde{\mathcal{K}}_D^{r|^{SE}_{CoM}}\dot{\tilde{y}}^{r|^{SE}_{CoM}}),\end{aligned} \tag{6.4.2}$$

where $\tilde{u}^{*r|^{SE}_{CoM}}(x)$ is obtained by solving $\ddot{\tilde{y}}^{r|^{SE}_{CoM}} = 0$, and $\tilde{\mathcal{K}}_P^{r|^{SE}_{CoM}}$, $\tilde{\mathcal{K}}_D^{r|^{SE}_{CoM}}$ and $\tilde{\epsilon}^{r|^{SE}_{CoM}}$ are the controller tuning parameters.

The associated Poincaré map, denoted by

$$x_{j+1} = \tilde{\mathcal{P}}^{r|^{SE}_{CoM}}(x_j, \beta_j^{r|^{SE}_{CoM}}) \tag{6.4.3}$$

is linearized around the fixed point $\tilde{x}^{*r|^{SE}_{CoM}}$ and $\beta^{*r|^{SE}_{CoM}}$ as

$$\begin{aligned}\delta x_{j+1} =&\frac{\partial\tilde{\mathcal{P}}^{r|^{SE}_{CoM}}(x, \beta^{r|^{SE}_{CoM}})}{\partial x}\delta x_j+ \\ &\frac{\partial\tilde{\mathcal{P}}^{r|^{SE}_{CoM}}(x, \beta^{D})}{\partial \beta^{r|^{SE}_{CoM}}}\delta\beta_j^{r|^{SE}_{CoM}}\end{aligned} \tag{6.4.4}$$

where $\delta\beta_j^{r|^{SE}_{CoM}} = \beta_j^{r|^{SE}_{CoM}} - \beta^{*r|^{SE}_{CoM}}$. A state feedback law given as

$$\delta x_j = -\tilde{\mathcal{K}}^{r|^{SE}_{CoM}}\delta\beta_j^{r|^{SE}_{CoM}} \tag{6.4.5}$$

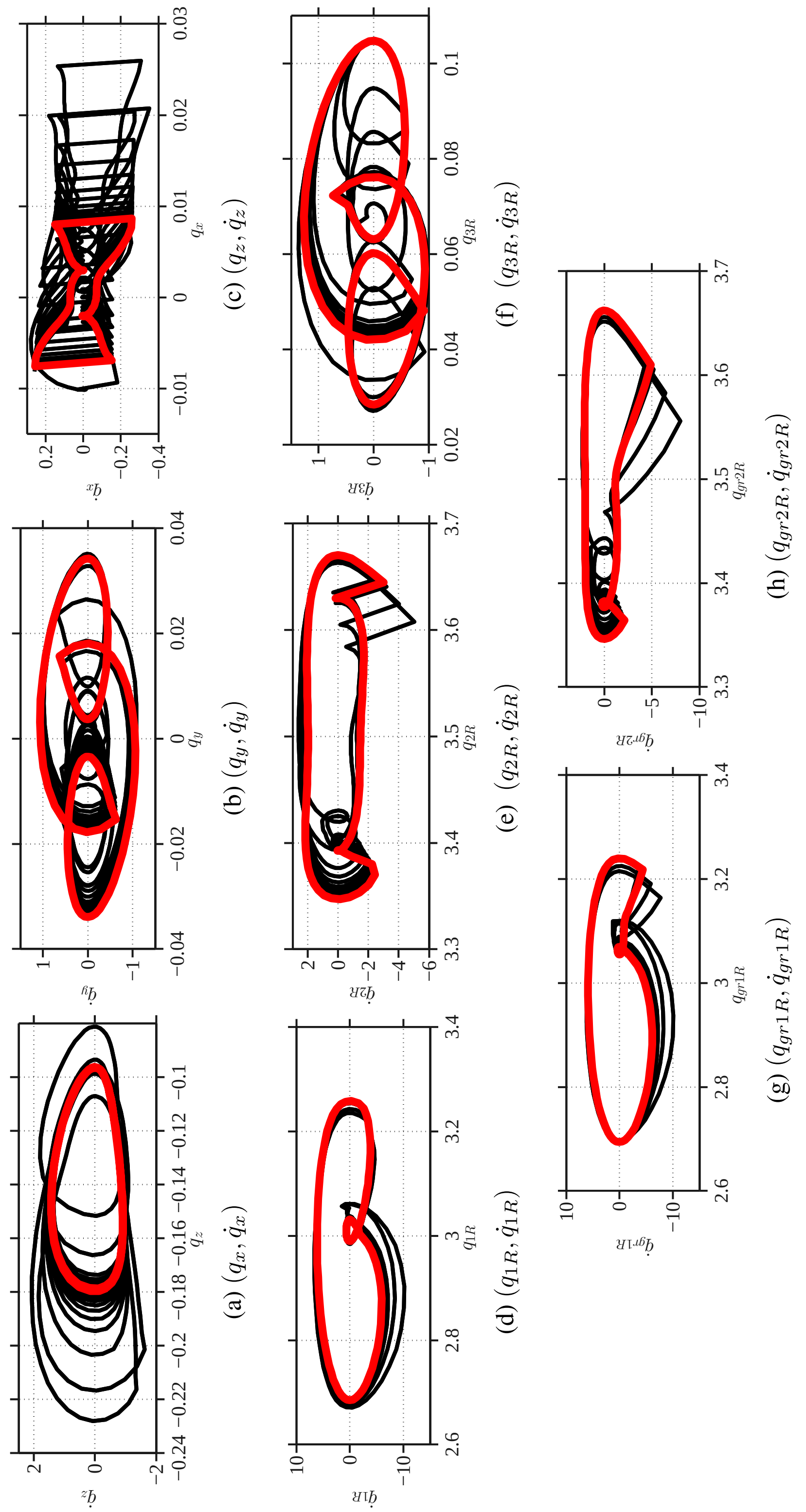
is designed such that

$$\textbf{eig}(\frac{\partial\tilde{\mathcal{P}}^{r|^{SE}_{CoM}}(x, \beta^{r|^{SE}_{CoM}})}{\partial x} - \frac{\partial\tilde{\mathcal{P}}^{r|^{SE}_{CoM}}(x, \beta^{r|^{SE}_{CoM}})}{\partial \beta^{r|^{SE}_{CoM}}}\tilde{\mathcal{K}}^{r|^{SE}_{CoM}}) \leq 1, \tag{6.4.6}$$

where the gain $\tilde{\mathcal{K}}^{r|^{SE}_{CoM}}$ is found through the Discrete Linear Quadratic Regulator Algorithm.

6.4.1 Poincaré Analysis

The Poincaré analysis for the closed loop system shows that the largest eigenvalues in magnitude for the closed loop system are $|\lambda_1| = 0.74$ and $|\lambda_2| = 0.066$, proving local exponential stability. Figure 6.9 shows the convergence of the phase portrait for $(q_x, \dot{q}_x)$, $(q_y, \dot{q}_y)$,$(q_z, \dot{q}_z)$, $(q_{1R}, \dot{q}_{1R})$, $(q_{2R}, \dot{q}_{2R})$, $(q_{3R}, \dot{q}_{3R})$,$(q_{gr1R}, \dot{q}_{gr1R})$ and $(q_{gr2R}, \dot{q}_{gr2R})$ after perturbing the fixed point.



(a) $(q_x, \dot{q}_x)$

(b) $(q_y, \dot{q}_y)$

(c) $(q_z, \dot{q}_z)$

(d) $(q_{1R}, \dot{q}_{1R})$

(e) $(q_{2R}, \dot{q}_{2R})$

(f) $(q_{3R}, \dot{q}_{3R})$

(g) $(q_{gr1R}, \dot{q}_{gr1R})$

(h) $(q_{gr2R}, \dot{q}_{gr2R})$

Figure 6.9: Convergence of the trajectories to the fixed point after perturbing the initial condition.

CHAPTER VII

Gait Initiation

Gait initiation consists of two parts – standing still, referred to here as quiet standing, and a transitory step from quiet standing to a periodic walking motion. To facilitate quiet standing, MARLO is fitted with passive feet, where the term passive refers to the fact that the ankles are locked in a fixed configuration, similar to many prosthetic lower limbs. The feet enable the robot to maintain balance, to a limited extent, in the sagittal and frontal planes of walking when no active balancing control effort is acting on the robot. However, in the face of small perturbations, such as a gentle push on the torso or a shallow-sloped ground surface, the robot can still topple. To avoid this, an active control algorithm is designed to achieve quiet standing. To facilitate the development of the gait initiation algorithms for quiet standing and the transitory step, MARLO's dynamic model is augmented with passive feet and a compliant ground model.[1]

The chapter begins with quiet standing. The strategies and challenges of quiet standing in a robot such as MARLO are discussed, followed by the presentation of a control policy designed to balance the robot at quiet standing. Next, quiet standing to walking transition is addressed. Various strategies and challenges are discussed, followed by the presentation of a family of control policies designed to fulfill the objectives of the transition from quiet standing to walking.

[1]Courtesy of K. Akbari Hamed and K. Galloway.

7.1 Quiet Standing

Quiet standing in humans consists of an upright posture of the body and is achieved using coordinated motion of the joints. During quiet standing, the nervous system's balancing controller compensates for internal perturbations, such as body deformation due to inhaling and exhaling, and for external perturbations, such as a backpack. The balancing controller slightly rocks the body forward and backward such that the upright posture is maintained.

7.1.1 Literature Overview and Strategies

In humans, the control system for standing is separate from that of walking [117, 118, 119, 120, 121]. In a similar manner, a separate controller for standing will be designed for MARLO. A standing controller has two objectives: (1) stabilize the inverted pendulum that characterizes the robot's posture at standing; and (2) coordinate the joints in the face of redundant DoF, meaning the robot can stand and maintain balance with different postures.

To stabilize the inverted pendulum characterizing standing, stiff ankle control is the intuitive solution. However, such an approach is not applied in humans because the overall stiffness of the series combination of the ankle and the "Achilles" tendon is dominated by the torque due to the gravity force exerted on the CoM, [121]. Reference [122] takes into account that the intrinsic stiffness of the human ankle is low and evaluates two types of controllers to achieve quiet standing in humans: a standard linear continuous PD controller, and an intermittent PD controller characterized by a switching function. It is found that a soft ankle combined with a feedback term that compensates for the flexibility of the Achilles can maintain the upright posture in humans. Moreover, there is functional merit to such a low-stiffness solution: a soft ankle can decrease impact shocks and can adapt to uneven terrain when walking. On the other hand, MARLO's morphology is not compatible with balance solutions that focus on joint coordination since the robot is underactuated and the actuated coordinates are simple 1 DoF revolute joints. Hence, stable quiet standing in

MARLO must focus on those strategies that are based on inverted pendulum models.

7.1.1.1 Ankle Strategy

MARLO has no actuator at the ankles and is therefore unable to use an ankle strategy for quiet standing.

7.1.1.2 Torso Strategy

In humans, a torso strategy in quiet standing is effective mainly because the torso consists of 50.80% [123] of the total body weight, whereas MARLO's torso contains only 29% of its total mass. Consequently, a torso strategy may not be very effective when applied to MARLO.

7.1.1.3 Hip Strategy

Humans do not instinctively move the hips to maintain stable quiet standing. On the other hand, a hip strategy that moves the CoM of the hips to stabilize the inverted pendulum characterizing quiet standing – instead of moving the torso – may be very effective in MARLO, where the hips account for 44% of the total weight.

	Torso	Hip	Ankle
Human	Y	N.E.	Y
MARLO	N.E.	Y	N.P.

Table 7.1: Quiet standing balancing strategies that MARLO and humans use, (Y) yes, (N.E.) not efficient, (N.P.) not possible.

7.1.2 Controller Design

Studies show that it is important to appropriately locate the CoP of the feet in order to maintain standing balance. At first glance, it seems that stabilization of quiet standing requires additional sensors, such as force sensors, to measure the CoP on the right and left feet. Here, assuming the ground surface is flat or nearly so, it is shown that such sensors are not necessary for achieving stable balance. Using a double inverted pendulum model, a feedback policy is designed such that stable quiet standing over flat ground and a gentle slope is achieved. The robustness of the feedback policy is verified with external perturbations such as instantaneous "push-pull" forces on the actual robot.

The intuitive explanation for the control performance follows. Suppose the leg angles are aligned with the torso and the robot is in vertical configuration with respect to the ground, an inverted pendulum configuration. Then changing the knee angle has relatively little effect on the y-position of the CoM, since the CoM moves up and down. On the other hand, as the bend in the knee increases, it forces the toe of the foot downward and moves the CoP forward on the foot. Conversely, the knees are straightened, the toe of the foot raise , and move the CoP backward. This happens because the angle of the foot is fixed relative to the shin link. This explains how the relative position of the CoP and CoM can be changed by a simple action, namely, knee bend, without making a measurement of the CoP. The relative difference between the position of CoM and CoP along the y-axis is proportional to the pitch accelerations of the above inverted pendulum.

The above intuitive operation of the quiet standing controller is mathematically formulated using "Newton's Law." Considering the forces and the moments acting on the robot at the inverted pendulum configuration, the relation between the position of the CoM, CoP and the angular acceleration of the inverted in the sagittal plane of walking is governed by

$$\frac{\mathcal{J}^{T}_{CoM} + \mathcal{J}^{H}_{CoM}}{\mathcal{W}_t}\ddot{q}_x = (p_{CoP,y} - p_{CoM,y}), \tag{7.1.1}$$

where $\mathcal{J}^{T}_{CoM}$ and $\mathcal{J}^{H}_{CoM}$ are the torso and hip moment of inertia around the CoM in y-z plane and $\mathcal{W}_t$ is MARLO's total weight. (7.1.1) relates the difference between the position of CoP and CoM in the sagittal plane $(p_{CoP,y} - p_{CoM,y})$ to the pitch acceleration $\ddot{q}_x$ in the context of balance control and suggests the following state feedback controller for the quantity $(p_{CoP,y} - p_{CoM,y})$,

$$\nu_{_{QS}} = -\frac{\mathcal{K}_P}{\epsilon^2}(q_x - q_x^*) - \frac{\mathcal{K}_D}{\epsilon}\dot{q}_x, \tag{7.1.2}$$

where q_x^* is the equilibrium pitch angle and $\mathcal{K}_P$, $\mathcal{K}_D$ and ϵ are controller gains to be chosen such that the following equation is Hurwitz

$$\frac{\mathcal{J}^T_{CoM} + \mathcal{J}^H_{CoM}}{\mathcal{W}_t}\ddot{q}_x + \frac{\mathcal{K}_D}{\epsilon}\dot{q}_x + \frac{\mathcal{K}_P}{\epsilon^2}(q_x - q_x^*) = 0 \tag{7.1.3}$$

(7.1.2) implies that adjusting $(p_{CoP,y} - p_{CoM,y})$ as a function of pitch angle and angular rate can maintain balance during quiet standing. In order to change the relative distance between the CoM and the CoP, the following feedback policy is proposed:

$$u_{_{QS}} = \frac{\partial h_{0,_{QS}}(q)}{\partial q}\mathcal{B}^{-1}(\frac{1}{\epsilon^2}\mathcal{K}_P y_{_{QS}} + \frac{1}{\epsilon}\mathcal{K}_D \dot{y}_{_{QS}}), \tag{7.1.4}$$

where $\mathcal{B}$ is a matrix that maps control commands to the leg harmonic drives and hip actuators. In (7.1.4), $\nu_{_{QS}}$ from (7.1.2) is added to the knee desired trajectories. Hence, using this, an offset value is added or subtracted to the knee virtual constraints, causing the knee angles of the robot to bend or stretch, resulting in the changes in the relative distance between the position of the CoM and CoP along the y-axis.

7.2 Transitioning from Quiet Standing to Walking

In this section, a transition from quiet standing to walking is designed with the help of optimization. A solution for the model is sought that joins the quiet standing position to the initial condition of a periodic walking motion. While in principle the transition solution could involve several walking steps, the work reported here seeks to make the transition in a single step called the "transition step."

The first attempt to obtaining a solution was to formulate a finitely parameterized optimization problem in the form of a two-point boundary value problem, with one boundary specified by the quiet standing posture and the other boundary corresponding to the initial condition of a pre-designed periodic walking motion with speed $0.3[\frac{m}{s}]$. A set of time-dependent output trajectories was parameterized with Bézier, and inverse dynamics was used to determine the torques to realize the trajectories. This direct approach did not work in the sense that a solution to the two-point boundary problem could not be found using `fmincon`.

An indirect approach was then formulated which did lead to a successful transition in a single step. The indirect approach to designing the transition step is composed of the following steps:

1. with the feet side-by-side and the robot in the standing position, the gait is initiated by straightening the knees, causing the robot to pitch forward in the sagittal plane;

2. the step length of the transition step is known from the posture at time zero of the periodic walking motion; the duration of the transition step is designed on the basis of a lumped-mass, double inverted pendulum model; and

3. with a non-zero initial velocity established, and a fixed value given for the step duration, the two-point boundary value problem could be solved with `fmincon`.

Moreover, the two-point boundary value problem could be solved using time-dependent output trajectories as well as time-independent output trajectories parameterized by the same phase variable used in the design of periodic orbits.

7.2.1 Non-zero pitch velocity

During quiet standing, the robot actively balances itself by adjusting the relative distance between the CoM and CoP along the y-axis. In this mode, the CoM horizontal velocity is zero. Initial energy is injected to the robot such that the CoM velocity reaches $0.1[\frac{m}{s}]$. The inverted pendulum model of MARLO requires this initial velocity in order to swing forward. This energy is injected through straightening the knees. Thereafter, the transitory step takes place as follows.

7.2.2 Transition duration, $\tau_{_{QS \mapsto W}}$

A reduced model, an inverted pendulum, is used to estimate the transitory step duration. The transitory duration is later used to design time-dependent desired trajectories. In the inverted pendulum, the ground reaction forces are exerted at the contact point between the flat foot and the ground. Gravitational forces act on the hip and torso CoM. The torque generated by the harmonic drives acts on the torso, which keeps the torso upright as the inverted pendulum swings forward during the quiet standing to walking transitory step. After applying Newton's Law, the equations of motion that reflect the dominant dynamic response associated with the torso are governed by a second-order differential equation,

$$\ddot{y}_T = \frac{\mathcal{W}_T + \mathcal{W}_H}{m_T z_H^*} y_T \quad = \beta y_T \tag{7.2.1}$$

where $\mathcal{W}_T$ and $\mathcal{W}_H$ are the torso and hip weights, respectively. m_T is torso mass and y_T is the torso CoM y position. The explicit solution to the above differential equation is given by:

$$y_T = \kappa_1 e^{\sqrt{\beta} t} + \kappa_2 e^{-\sqrt{\beta} t} \tag{7.2.2}$$

where $\kappa_1 = \frac{y_T|_{t=0}\sqrt{\beta} + \dot{y}_T|_{t=0}}{2\sqrt{\beta}}$, $\kappa_2 = \frac{y_T|_{t=0}\sqrt{\beta} - \dot{y}_T|_{t=0}}{2\sqrt{\beta}}$; $y_T|_{t=0}$ and $\dot{y}_T|_{t=0}$ are initial torso position and velocity at the beginning of the quiet standing to walking transitory step. Using (7.2.2), the time duration corresponding to the transitory step is calculated and is used to normalize the time-dependent gait-timing variable. Let $\sqrt{\beta} = 4.94$, step length $0.3[m]$, $y_T|_{t=0} = 0$, $\dot{y}_T|_{t=0} = 0.1[\frac{m}{s}]$, using (7.2.2) the estimated transition time is computed as $240[m.s]$.

7.2.3 Specification of controlled outputs

The controlled outputs, whether time dependent or independent, are written in the form used for the virtual constraints, namely,

$$\begin{cases} y_{_{QS\mapsto W}} = h_{0,_{QS\mapsto W}}(q_s) - h_{d,_{QS\mapsto W}}(\theta_{_{QS\mapsto W}}(t), \alpha^*) & \text{time-dependent} \\ y_{_{QS\mapsto W}} = h_{0,_{QS\mapsto W}}(q_s) - h_{d,_{QS\mapsto W}}(\theta_{_{QS\mapsto W}}(q_s), \alpha^*) & \text{time-independent} \end{cases} \tag{7.2.3}$$

where $h_{0,_{QS\mapsto W}}(q_s)$ specifies the vector of variables to be controlled, $h_{d,_{QS\mapsto W}}$ is the desired evolution of the controlled variables, and $\theta_{_{QS\mapsto W}}$ is the gait-timing variable.

The controlled variables are nominally selected as

$$h_{0,_{QS\mapsto W}}(q_s) = \begin{bmatrix} \dfrac{q_{gr1R} + q_{gr2R}}{2} \\ \dfrac{q_{gr1L} + q_{gr2L}}{2} \\ q_{gr2R} - q_{gr1R} \\ q_{gr2L} - q_{gr1L} \\ q_{3R} \\ q_{3L} \end{bmatrix} = \begin{bmatrix} q_{grR}^{LA} \\ q_{grL}^{LA} \\ q_{grR}^{Knee} \\ q_{grL}^{Knee} \\ q_R^{Hip} \\ q_L^{Hip} \end{bmatrix}, \tag{7.2.4}$$

7.2.3.1 Optimal time-dependent trajectories

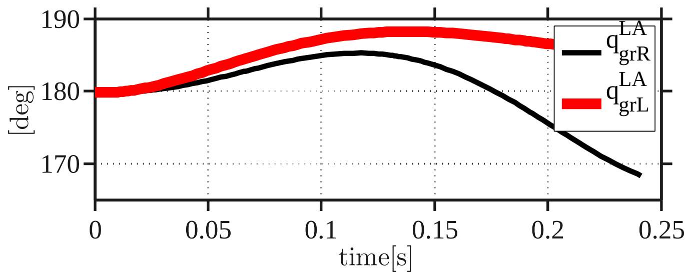
The gait-timing variable is selected as

$$\theta_{_{QS\mapsto W}}(t) = \frac{t}{\tau_{_{QS\mapsto W}}}, \tag{7.2.5}$$

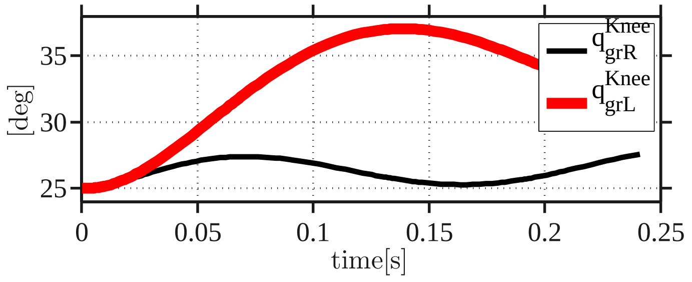
where t is zero at the beginning of the step and $\tau_{_{QS\mapsto W}}$ is the duration of the transition step determined from the double inverted pendulum model. The desired evolution of the controlled variables $h_{d,_{QS\mapsto W}}(\theta_{_{QS\mapsto W}}(t))$ is parameterized with degree 6 Bézier polynomials and inverse dynamics is used to determine torques that set the second derivative of the outputs to zero. The cost function is selected as cmt and is minimized subject to constraints:

1. positive normal ground reaction force.
2. $\left|\frac{F_{st}^{T}}{F_{st}^{N}}\right| < 0.7$.
3. $\left\| x^{-}_{_{QS\mapsto W}} - x^{+}_{W} \right\| = 0$, where, $x^{-}_{_{QS\mapsto W}}$ represent the state vector at the end of the transition step and x^{+}_{W} is the initial state vector for a periodic walking motion.

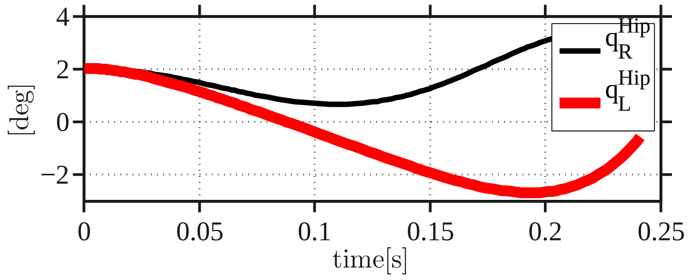
Figure 7.1 shows the evolution of the leg angles, knee angles and hip angles during the quiet standing to walking transition step. The transition step lasts $240[m.s]$ and during the transition, q_{grR}^{LA} decreases to $168[deg]$, while q_{grL}^{LA} increases to $188[deg]$. The value of q_{grR}^{Knee} varies between $25[deg]$ and $28[deg]$, while q_{grL}^{Knee} reaches $36[deg]$ to avoid scuffing. The hip angles both start from $2[deg]$ and q_{L}^{Hip} decreases to $-3[deg]$ during the step, causing the swing hip to move upward.



(a)



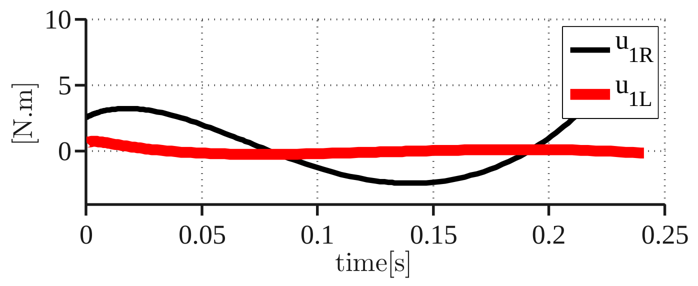
(b)



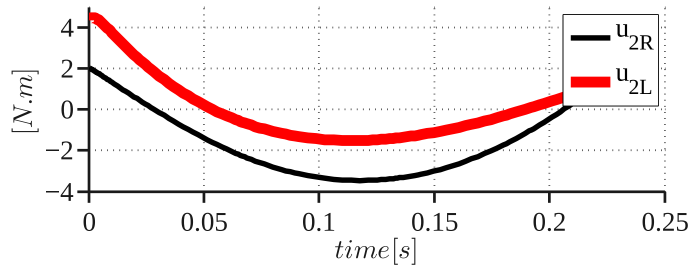
(c)

Figure 7.1: Evolution of variables during quiet standing to walking transition step **(a)** q_{grR}^{LA}, q_{grL}^{LA}, **(b)** q_{grR}^{Knee}, q_{grL}^{Knee}, and **(c)** q_{R}^{Hip}, q_{L}^{Hip}.

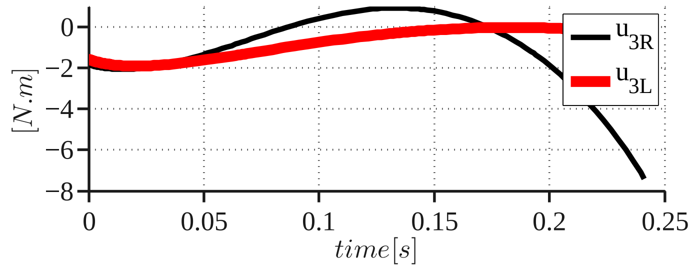
Figure 7.2 shows the computed control effort. The maximum actuation torque required by u_{1R} is $5[N.m]$ and for u_{1L} is $0[N.m]$. The maximum torque required by u_{2R} is $-4[N.m]$, whereas u_{2L} has a maximum value of $4[N.m]$. The stance hip actuator u_{3R} demands maximum torque of $-8[N.m]$. Figure 7.3 shows the power required corresponding to the motor actuators. It can be seen that the maximum power demand is associated with p_{1R} and occurs at the end of the step.



(a)

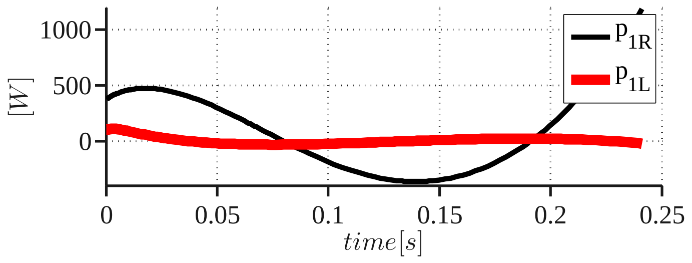


(b)

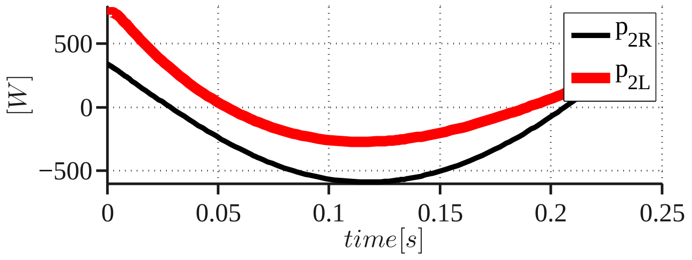


(c)

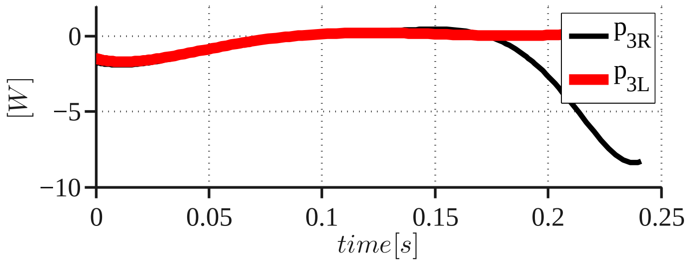
Figure 7.2: Control effort during quiet standing to walking transition step **(a)** u_{1R}, u_{1L}, **(b)** u_{2R}, u_{2L}, and **(c)** u_{3R}, u_{3L}.



(a)



(b)



(c)

Figure 7.3: Actuator power during quiet standing to walking transition step **(a)** p_{1R}, p_{1L}, **(b)** p_{2R}, p_{2L}, and **(c)** p_{3R}, p_{3L}.

7.2.3.2 Optimal time-independent desired trajectories

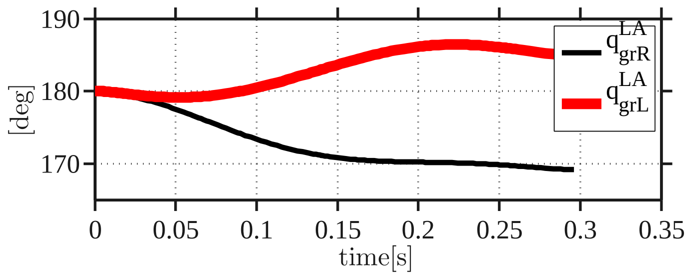
The gait-timing variable is selected as

$$\theta_{_{QS\mapsto W}}(q_s) = \begin{cases} \theta_{R,_{QS\mapsto W}} & \text{if the right leg is stance;} \\ \theta_{L,_{QS\mapsto W}} & \text{otherwise.} \end{cases} \tag{7.2.6}$$

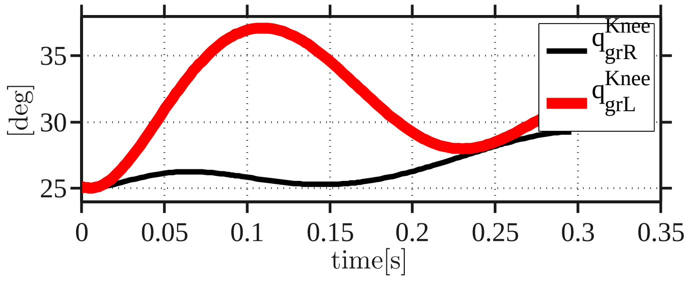
By selecting a time-independent gait-timing variable $\theta_{_{QS\mapsto W}}(q_s)$, the previously used zero dynamics calculations in Chapter IV are once again applicable, and the zero dynamics of the robot is used to find a transition step through optimization. The desired evolution of the controlled variables $h_{d,_{QS\mapsto W}}(\theta_{_{QS\mapsto W}}(q_s))$ is parameterized with degree 6 Bezier polynomials. The cost function is selected as cmt and is minimized subject to constraints:

1. positive normal ground reaction force.
2. $\left|\frac{F^T_{st}}{F^N_{st}}\right| < 0.7$.
3. $\left\|x^-_{_{QS\mapsto W}} - x^+_W\right\| = 0$, where, $x^-_{_{QS\mapsto W}}$ represent the state vector at the end of the transition step and x^+_W is the initial state vector for a periodic walking motion.

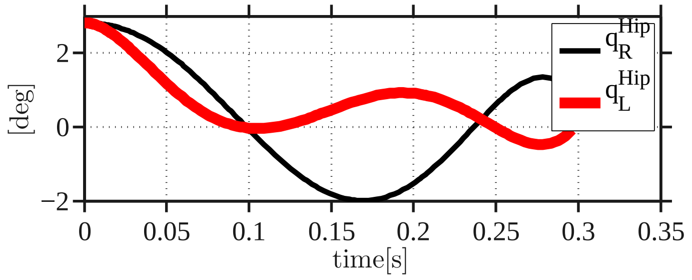
Figure 7.4 shows the evolution of the leg angles, knee angles and hip angles during the transition step. The step takes $300[m.s]$. During the step, q^{LA}_{grR} and q^{LA}_{grL} both begin at $180[deg]$. At the end of the step, q^{LA}_{grR} decreases to $169[deg]$ and q^{LA}_{grL} increases to $185[deg]$. The maximum knee angle corresponds to q^{Knee}_{grL} and is $37[deg]$, whereas q^{Knee}_{grR} slightly increases to $30[deg]$ toward the end of the step. Hip angles initiate from well above $2[deg]$ and vary between $-2[deg]$ and $2[deg]$ during the step.



(a)



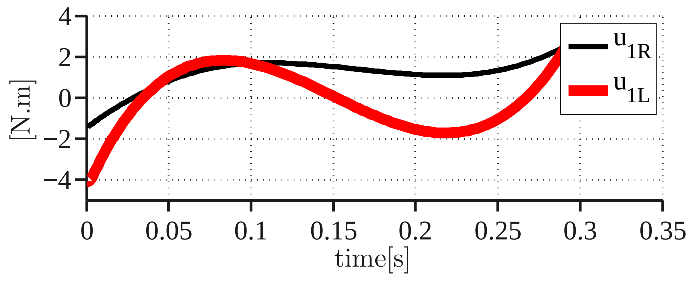
(b)



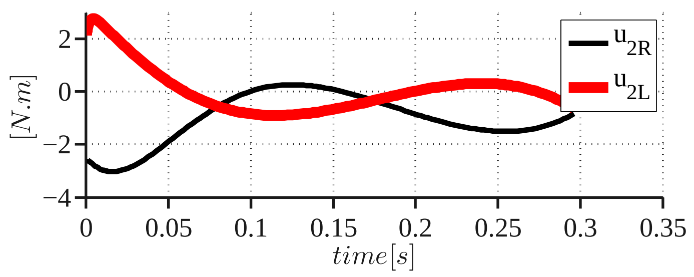
(c)

Figure 7.4: Evolution during quiet standing to walking transition step **(a)** q^{LA}_{grR}, q^{LA}_{grL}, **(b)** q^{Knee}_{grR}, q^{Knee}_{grL}, and **(c)** q^{Hip}_{R}, q^{Hip}_{L}.

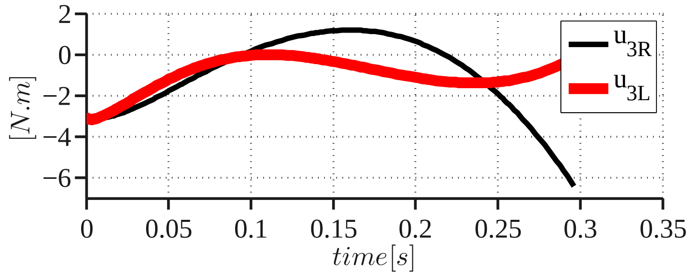
Figure 7.5 shows the actuator torques during the transition step. u_{1R} varies between $-2[N.m]$ and $2[N.m]$ and u_{1L} varies between $-4[N.m]$ and $2[N.m]$. u_{2R} and u_{2L} both start with $-2[N.m]$ and $2[N.m]$ and their magnitude decrease towards the end of the step. The maximum hip actuator is required by u_{3R}, which is $-6[N.m]$, and u_{3L} fluctuates between $-3[N.m]$ and $0[N.m]$. Figure 7.6 illustrates each actuator's required power during the transition step. p_{1R} and p_{2R} require the maximum power during the step, whereas the hip actuators have the lowest power demand.



(a)

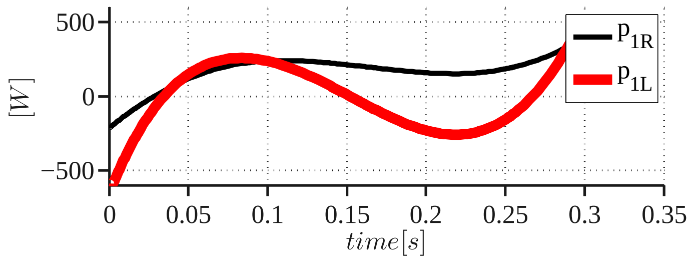


(b)

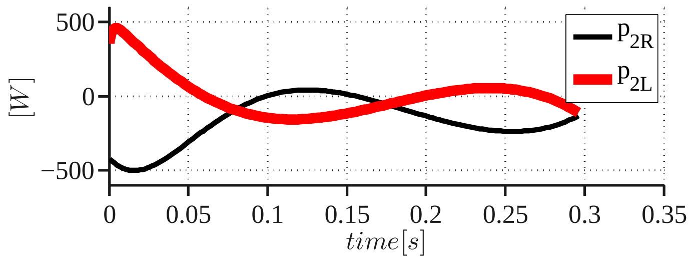


(c)

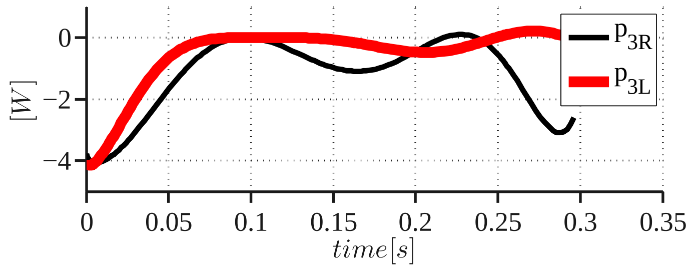
Figure 7.5: Control effort during quiet standing to walking transition step **(a)** u_{1R}, u_{1L}, **(b)** u_{2R}, u_{2L}, and **(c)** u_{3R},u_{3L}.



(a)



(b)



(c)

Figure 7.6: Actuator power during quiet standing to walking transition step **(a)** p_{1R}, p_{1L}, **(b)** p_{2R}, p_{2L}, and **(c)** p_{3R}, p_{3L}.

7.2.4 Controller Design

The controllers proposed in this section enforce the time-dependent output trajectories and time-independent virtual constraints described in the previous sections.

7.2.4.1 Time-dependent outputs

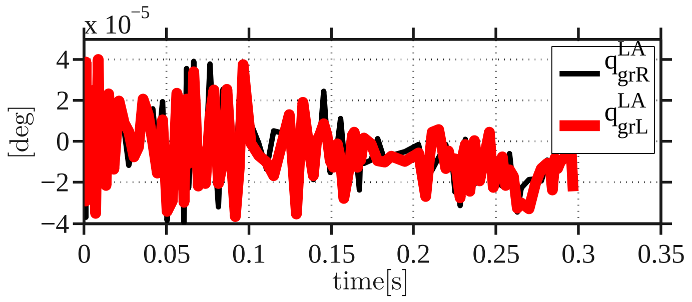
Recall that the gait-timing variable $\theta_{QS\mapsto W}$ is time-dependent, which means that the input-output linearizing control policy introduced in Chapter VI is not applicable. Therefore, a proportional-derivative PD controller is proposed for the quiet standing to walking transition step with the time-dependent gait-timing variable. Considering (7.2.3), the feedback policy $u_{_{QS\mapsto W}}(\frac{t_{_{QS\mapsto W}}}{\tau_{_{QS\mapsto W}}})$ is given by

$$u_{_{QS\mapsto W}} = \frac{\partial h_{0,_{QS\mapsto W}}(q_s)}{\partial q} \mathcal{B}^{-1}(\frac{1}{\epsilon^2}\mathcal{K}_P y_{_{QS\mapsto W}}(t_{_{QS\mapsto W}}, q_s) + \frac{1}{\epsilon}\mathcal{K}_D \dot{y}_{_{QS\mapsto W}}(t_{_{QS\mapsto W}}, q_s)) \tag{7.2.7}$$

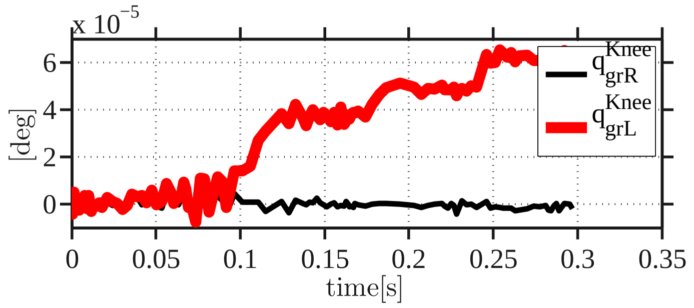
where $\mathcal{K}_P$, $\mathcal{K}_D$, ϵ are control tuning parameters. $\mathcal{B}$ is a matrix which maps control commands to the leg harmonic drives and hip actuators.

7.2.4.2 Controller Performance

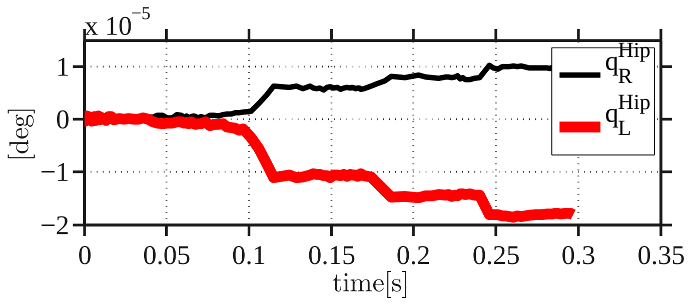
The controller in (7.2.7) was implemented on the tether-free model of the robot with point feet, where the ground model is a rigid ground model. Figure 7.7 demonstrates the tracking errors corresponding to the leg angle, knee angle and hip angle. The tracking error for all of these virtual constraints is on the order of 10^{-5} and it is seen the error increases toward the end of the step.



(a)



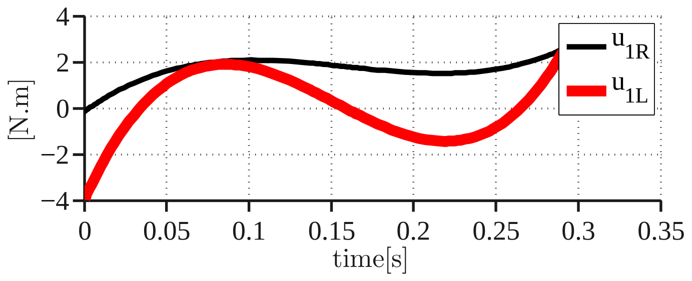
(b)



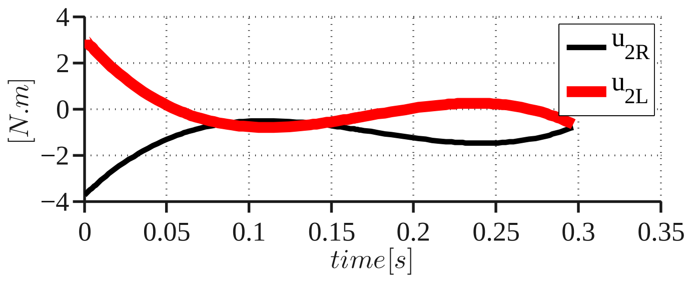
(c)

Figure 7.7: Tracking error during quiet standing to walking transition step **(a)** q_{grR}^{LA}, q_{grL}^{LA}, **(b)** q_{grR}^{Knee}, q_{grL}^{Knee}, and **(c)** q_{R}^{Hip}, q_{L}^{Hip}.

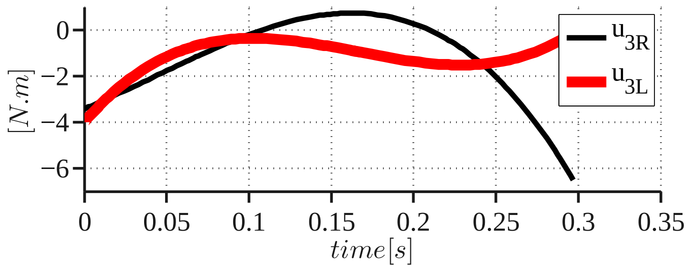
Figure 7.8 illustrates the control actuation torque corresponding to the motors during the transition step. u_{1L} requires $-4[N.m]$ to $3[N.m]$, whereas u_{1R} requires $0[N.m]$ to $3[N.m]$. u_{2R} and u_{2L} both start with $-4[N.m]$ and $3[N.m]$ at the beginning of the step and demand smaller torque commands towards the end of the step. Turning to the hip actuators, u_{3R} demands the maximum actuation torque of $-6[N.m]$ towards the end of the transition step.



(a)



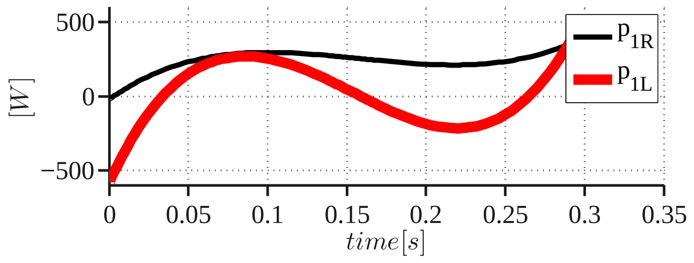
(b)



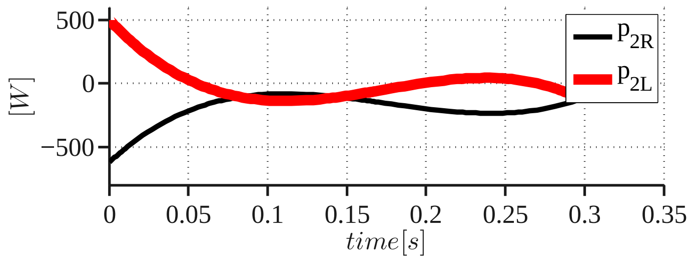
(c)

Figure 7.8: Actuation torque during quiet standing to walking transition step **(a)** u_{1R}, u_{1L}, **(b)** u_{2R}, u_{2L}, and **(c)** u_{3R}, u_{3L}.

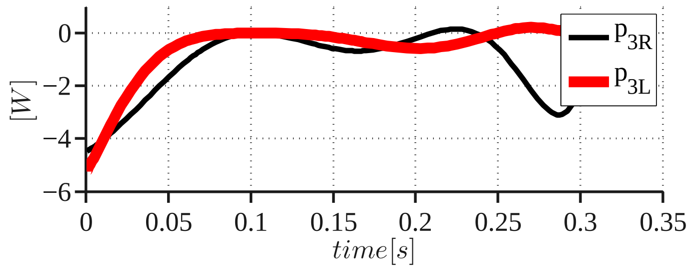
Figure 7.9 illustrates the power required by the robot's motors. It can be seen that a large amount of negative power is associated with p_{1L} and p_{2R} at the beginning of the step. This is due to the fact that the generated torques by these actuators are in the opposite direction of the rotation motion of the links associated with those actuators.



(a)



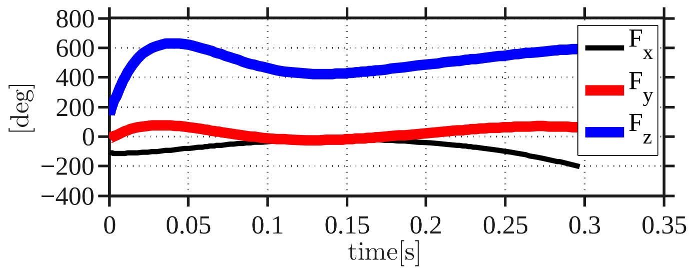
(b)



(c)

Figure 7.9: Actuation power during quiet standing to walking transition step **(a)** p_{1R}, p_{1L}, **(b)** p_{2R}, p_{2L}, and **(c)** p_{3R}, p_{3L}.

Figure 7.10 shows the components of the ground reaction force during the transition step. The ground reaction force acts at the contact point between the right leg and the ground. The small magnitude of the tangential components is desirable so that slippage does not occur during the transition step.



(a)

Figure 7.10: Components of the ground reaction force at the end of the stance leg during quiet standing to walking transition step.

7.2.4.3 Time-Independent/ Normal Virtual Constraints

Recall that as the gait-timing variable $\theta_{_{QS\mapsto W}}$ is time-independent, the input-output linearizing control policy introduced in Chapter VI applies. The decoupling matrix given by the following equation is invertible:

$$\mathcal{A}_{_{QS\mapsto W}}(q_s,\dot{q}_s) = \frac{\partial h_{_{QS\mapsto W}}(q_s)}{\partial q_s}\mathcal{D}_s^{-1}(q_s)\mathcal{B}_s, \tag{7.2.8}$$

while $u^*_{_{QS\mapsto W}}(x)$ is obtained by solving for the input that sets $\ddot{y}_{_{QS\mapsto W}} = 0$, giving

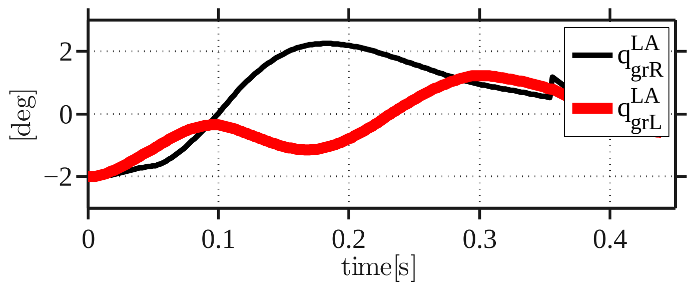
$$\begin{aligned}u^*_{_{QS\mapsto W}}(q_s,\dot{q}_s) =& \mathcal{A}^{-1}_{_{QS\mapsto W}}(q_s,\dot{q}_s)\left(\frac{\partial}{\partial q_s}\left(\frac{\partial h_{_{QS\mapsto W}}(q_s)}{\partial q_s}\dot{q}_s\right)\dot{q}_s - \right.\\ &\left.\frac{\partial h_{_{QS\mapsto W}}(q_s)}{\partial q_s}\mathcal{D}_s^{-1}(q_s)\mathcal{H}_s(\dot{q}_s,q_s)\right).\end{aligned} \tag{7.2.9}$$

Input-output linearization yields the feedback controller for the quiet standing to walking transition step,

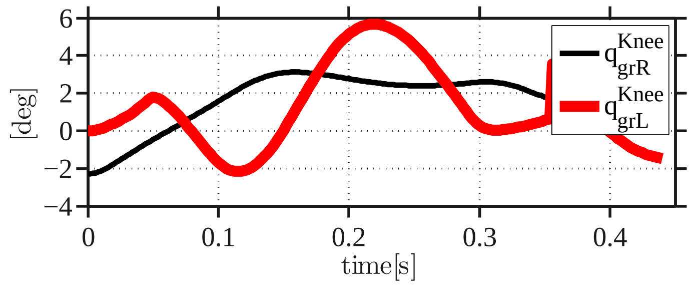
$$u_{_{QS\mapsto W}}(x) = u^*_{_{QS\mapsto W}}(x) - \mathcal{A}^{-1}_{_{QS\mapsto W}}(x)(\frac{1}{\epsilon^2}\mathcal{K}_P y_{_{QS\mapsto W}} + \frac{1}{\epsilon}\mathcal{K}_D \dot{y}_{_{QS\mapsto W}}). \tag{7.2.10}$$

7.2.4.4 Controller Performance

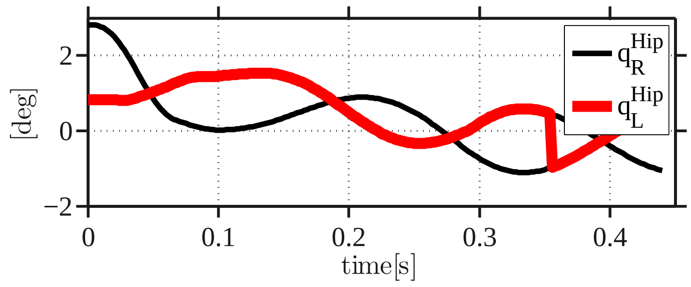
The controller in (7.2.10) was implemented on the tether-free model of the robot with prosthetic feet where the ground model is a compliant model. Figure 7.11 demonstrates the tracking errors corresponding to the leg angle, knee angle and hip angle. The maximum tracking error is associated with q_{grL}^{Knee}, which is $6[deg]$. The tracking error corresponding to q_{grR}^{LA} and q_{grL}^{LA} vary between $-2[deg]$ to $2[deg]$. The tracking error corresponding to q_{R}^{Hip} and q_{L}^{Hip} vary between $-1[deg]$ to slightly more than $2[deg]$.



(a)



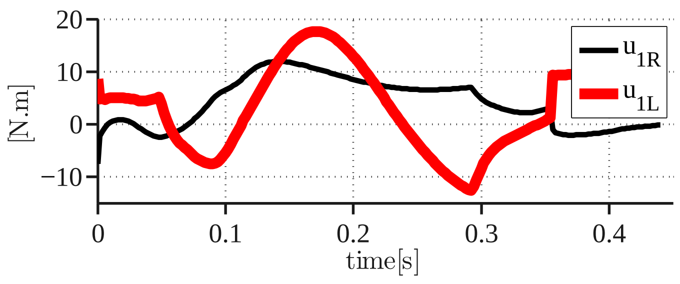
(b)



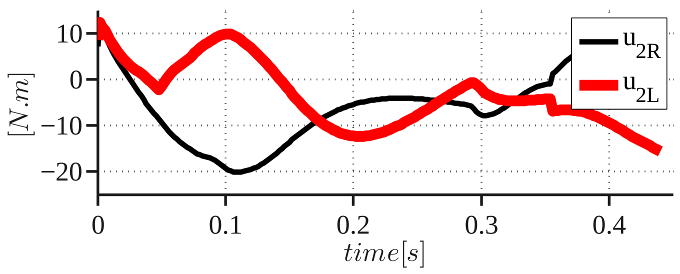
(c)

Figure 7.11: Tracking error during quiet standing to walking transition step **(a)** q^{LA}_{grR}, q^{LA}_{grL}, **(b)** q^{Knee}_{grR}, q^{Knee}_{grL}, and **(c)** q^{Hip}_{R}, q^{Hip}_{L}, black curve for right leg and red curve for left leg.

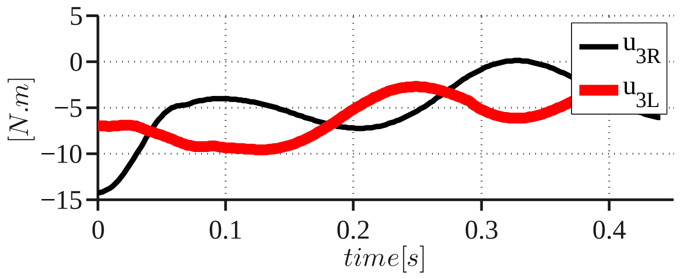
Figure 7.12 illustrates the control actuation torque corresponding to the robot's actuators during the quiet standing to walking transition step when the gait-timing variable is time-independent. u_{1L} varies between $-11[N.m]$ to $19[N.m]$ and u_{1R} varies between $-10[N.m]$ to $10[N.m]$. The maximum torque demanded by u_{2R} is $-20[N.m]$. The hip actuators demand a maximum $-15[N.m]$ at the beginning of the step.



(a)



(b)



(c)

Figure 7.12: Actuation torque during quiet standing to walking transition step **(a)** u_{1R}, u_{1L}, **(b)** u_{2R}, u_{2L}, and **(c)** u_{3R}, u_{3L}, black curve for right leg and red curve for left leg.

7.2.4.5 Time-Independent/ CoM Positioning

In Chapter VI, a similar modifed virtual constraint is considered for the quiet standing to walking mode, given as

$$h^{r|^{SE}_{CoM}}_{0,_{QS\mapsto W}}(q_s) = \begin{bmatrix} \dfrac{q_{gr1R}+q_{gr2R}}{2} \\ \dfrac{q_{gr1L}+q_{gr2L}}{2} \\ q_{gr2R}-q_{gr1R} \\ q_{gr2L}-q_{gr1L} \\ q_{3R} \\ p^h_{CoM}-p^h_{SE} \end{bmatrix} = \begin{bmatrix} q^{LA}_{grR} \\ q^{LA}_{grL} \\ q^{Knee}_{grR} \\ q^{Knee}_{grL} \\ q^{Hip}_{R} \\ r|^{SE}_{CoM} \end{bmatrix}, \tag{7.2.11}$$

where $r|^{SE}_{CoM}$ is the horizontal distance between the swing-leg-end and CoM in the frontal plane during the transition step, as previously explained in Chapter VI. The optimal distance between the swing-leg-end and the CoM of the robot in the frontal plane is computed as explained in Section 5.4: a NURB curve of order five with twenty control points is fit to this distance. The modified virtual constraint for the transition step is written as

$$\begin{aligned} y^{r|^{SE}_{CoM}}_{QS\mapsto W} =& h^{r|^{SE}_{CoM}}_{QS\mapsto W}(q_s, \alpha^{*,r|^{SE}_{CoM}}_{QS\mapsto W}) \\ &= h^{r|^{SE}_{CoM}}_{0,_{QS\mapsto W}}(q_s) - \\ &h^{r|^{SE}_{CoM}}_{d,_{QS\mapsto W}}(\theta^{r|^{SE}_{CoM}}_{QS\mapsto W}(q_s), \alpha^{*,r|^{SE}_{CoM}}_{QS\mapsto W}), \end{aligned} \tag{7.2.12}$$

where $y^{r|^{SE}_{CoM}}_{QS\mapsto W}$, $\alpha^{*,r|^{SE}_{CoM}}_{QS\mapsto W}$ are the modified virtual constraints and modified optimal gait parameters corresponding to the transition step. Considering (7.2.12) and taking first and second time derivatives similar to (6.2.2) and (6.2.4), the decoupling matrix for the modified virtual constraint is given as

$$\mathcal{A}^{r|^{SE}_{CoM}}_{QS\mapsto W}(q_s, \dot{q}_s) = \frac{\partial h^{r|^{SE}_{CoM}}_{QS\mapsto W}(q_s)}{\partial q_s} \mathcal{D}_s^{-1}(q_s)\mathcal{B}_s, \tag{7.2.13}$$

while $u^{*,r|^{SE}_{CoM}}_{QS\mapsto W}$ is obtained by solving for the input that sets $\ddot{y}^{r|^{SE}_{CoM}}_{QS\mapsto W} = 0$, giving

$$u^{*r|^{SE}_{CoM}}_{QS\mapsto W}(q_s,\dot{q}_s) = (\mathcal{A}^{r|^{SE}_{CoM}}_{QS\mapsto W})^{-1}(\frac{\partial}{\partial q_s}(\frac{\partial h^{r|^{SE}_{CoM}}_{QS\mapsto W}(q_s)}{\partial q_s}\dot{q}_s)\dot{q}_s - \frac{\partial h^{r|^{SE}_{CoM}}_{QS\mapsto W}(q_s)}{\partial q_s}\mathcal{D}_s^{-1}(q_s)\mathcal{H}_s(\dot{q}_s,q_s)). \tag{7.2.14}$$

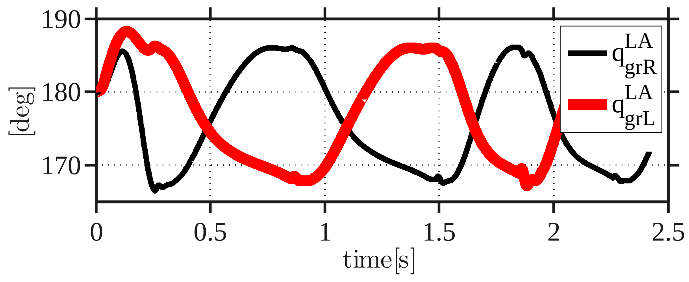
Similar to (6.2.7), since the decoupling matrix is invertible, input-output linearization yields the feedback controller

$$u^{r|^{SE}_{CoM}}_{QS\mapsto W}(x) = u^{*,r|^{SE}_{CoM}}_{QS\mapsto W}(x) - \left(\mathcal{A}^{r|^{SE}_{CoM}}_{QS\mapsto W}(x)\right)^{-1}(\frac{1}{\epsilon^2}\mathcal{K}_P y^{r|^{SE}_{CoM}}_{QS\mapsto W} + \frac{1}{\epsilon}\mathcal{K}_D \dot{y}^{r|^{SE}_{CoM}}_{QS\mapsto W}), \tag{7.2.15}$$

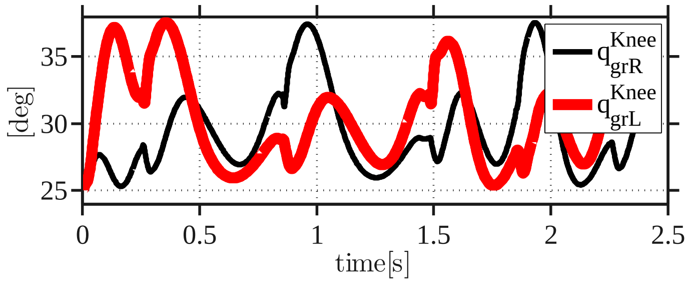
where $\mathcal{K}_P$, $\mathcal{K}_D$ and ϵ are the control parameters for the modified virtual constraints during the transition step.

7.2.4.6 Controller Performance

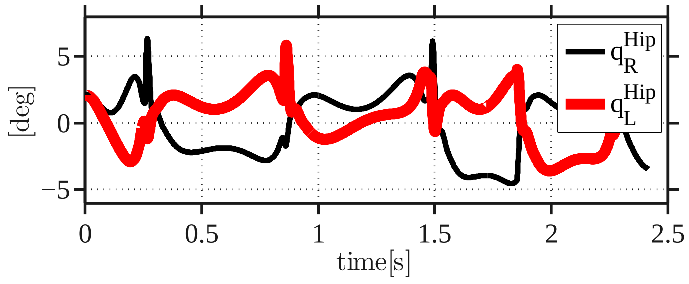
The controller (7.2.15) was implemented on the tether-free model of the robot with prosthetic feet, where the ground model is a compliant model. Figure 7.13 shows the time evolution of the leg angles, knee angles and hip angles for four consecutive steps after the transitory step. The transitory step takes $250[m.s]$. Figure 7.14 depicts the tracking performance of the controller where the errors are on order 10^{-1} at the beginning of each step and vanish toward the end of the step. Figure 7.15 illustrates the required control effort per actuators to enforce the transitory and periodic virtual constraints.



(a)

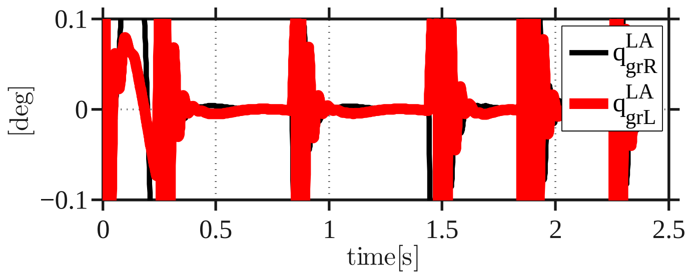


(b)

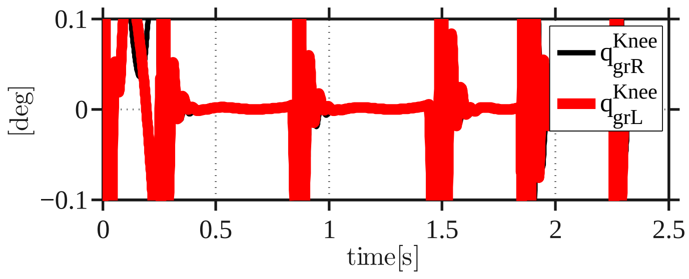


(c)

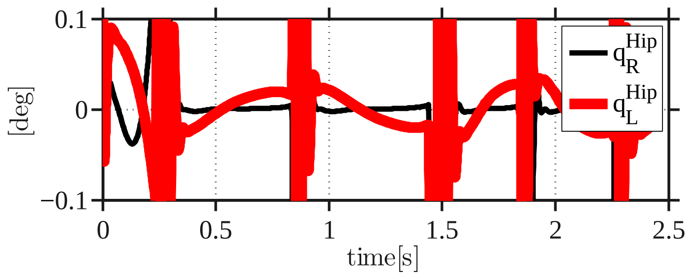
Figure 7.13: The evolution of the angles during quiet standing to walking transition step **(a)** q_{grR}^{LA}, q_{grL}^{LA}, **(b)** q_{grR}^{Knee}, q_{grL}^{Knee}, and **(c)** q_{R}^{Hip}, q_{L}^{Hip}, black curve for right leg and red curve for left leg.



(a)

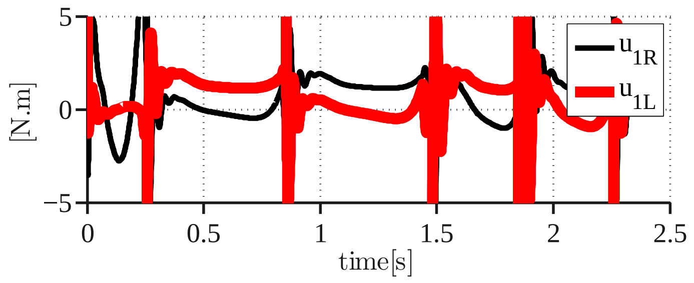


(b)

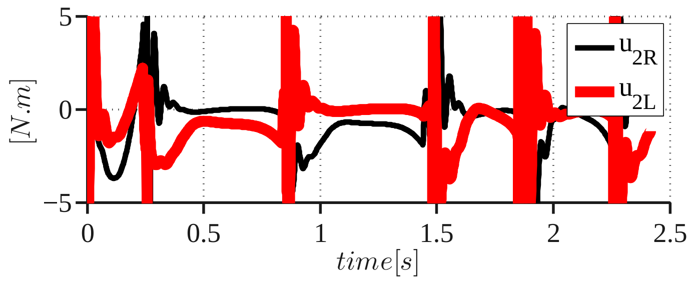


(c)

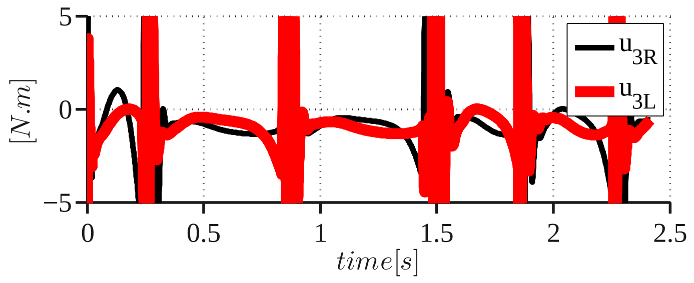
Figure 7.14: Tracking error during quiet standing to walking transition step **(a)** q^{LA}_{grR}, q^{LA}_{grL}, **(b)** q^{Knee}_{grR}, q^{Knee}_{grL}, and **(c)** q^{Hip}_{R}, q^{Hip}_{L}, black curve for right leg and red curve for left leg.



(a)



(b)



(c)

Figure 7.15: Actuation torque during quiet standing to walking transition step **(a)** u_{1R}, u_{1L}, **(b)** u_{2R}, u_{2L}, and **(c)** u_{3R}, u_{3L}, black curve for right leg and red curve for left leg.

CHAPTER VIII

Walking Experiments

This chapter integrates the individual controllers for standing, transitioning, and steady-state walking into an overall controller for autonomous locomotion. The controller is then experimentally deployed on both the planar and tether-free versions of MARLO. In planar mode, the robot's lateral and yaw motion is constrained, whereas in tether-free mode, the robot's full 3D dynamics come into play.

The chapter begins with a control architecture that integrates the control strategies developed throughout this study into a single control block and appropriate switching logic activates individual control strategies as a function of time and the state of the robot. Using this controller, autonomous planar walking is demonstrated on MARLO equipped with passive prosthetic feet. To be clear, the term autonomous means that the robot starts in a standing position and transitions to periodic walking without any intervention by the operator, such as a push used in robots such as MABEL and Rabbit. The results from autonomous tether-free walking are presented next. The maximum number of steps achieved is six natural human-like steps with walking speed of $0.3[\frac{m}{s}]$. Concluding remarks and discussion finish the chapter.

8.1 Embedded Controller Environment

For embedding the controller in the robot, xPC Target is employed, which executes Simulink and Stateflow models on a target computer for rapid control prototyping and real-time testing applications. It provides a library of drivers, real-time kernel and host target interface for real-time monitoring and data logging.

8.2 Overall Controller

Figure 8.1 and 8.2 show the basic architecture for each of the time-dependent and time-independent closed-loop strategies. A set of six variables to be controlled h_0 is constructed from the position variables of the robot. Desired values for these variables are specified by h_d. In all but one case, the function h_d is time-invariant, the lone exception being one of the controllers for the transition step. A HZD-based controller Γ is employed to drive the difference between h_0 and h_d to zero. In later discussion, the "tracking error" is defined as $h_0 - h_d$.

The state vector $x \in \mathbb{R}^{26\times 1}$ consists of the angles required to parameterize the robot in single support as well as their derivatives. The angles are directly measured, whereas the angular velocities are estimated, $S(q)$. The angles that relate the robot to the world frame are measured in two different ways, depending on the configuration of the platform. For the tether-free experiments, an IMU is used to measure the roll, pitch and yaw angles of the torso. For the planar experiments, the pitch angle of the torso can also be measured by an encoder between the robot's torso and the boom. The output vector from the controller in both planar and tether-free experiments has the same dimension, $u \in \mathbb{R}^{6\times 1}$. During planar walking, however, the commands to the hip actuators are constant set-points, whereas during tether-free walking experiments, the hip actuators track time- or state-varying trajectories.

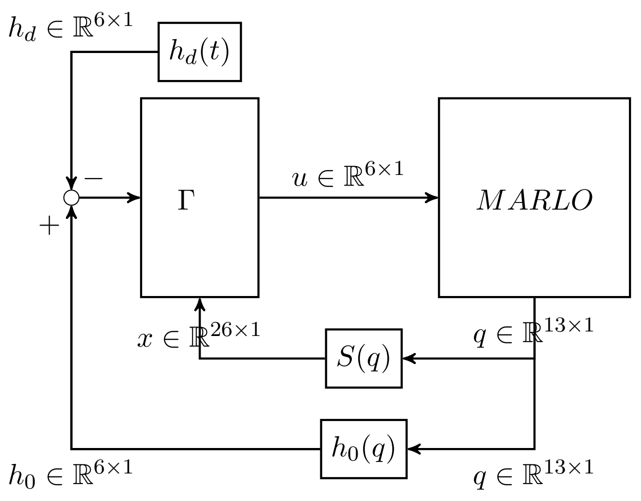

$h_d \in \mathbb{R}^{6\times1}$
$h_d(t)$
$-$
$+$
Γ
$u \in \mathbb{R}^{6\times1}$
$MARLO$
$x \in \mathbb{R}^{26\times1}$
$S(q)$
$q \in \mathbb{R}^{13\times1}$
$h_0(q)$
$h_0 \in \mathbb{R}^{6\times1}$
$q \in \mathbb{R}^{13\times1}$

Figure 8.1: Time-dependent closed-loop block-diagram.

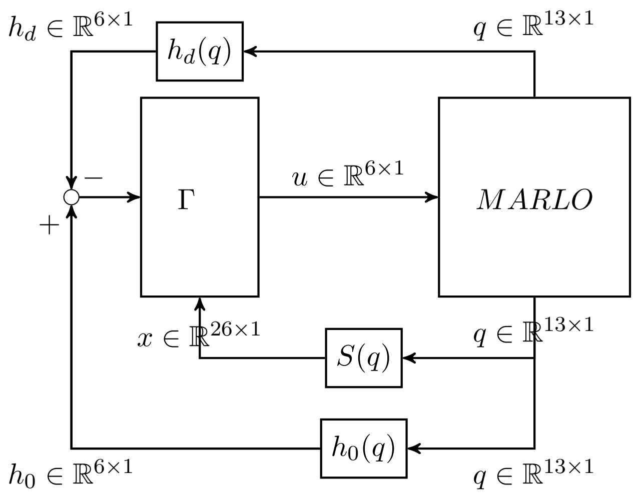

$h_d \in \mathbb{R}^{6\times1}$
$h_d(q)$
$q \in \mathbb{R}^{13\times1}$
$-$
$+$
Γ
$u \in \mathbb{R}^{6\times1}$
$MARLO$
$x \in \mathbb{R}^{26\times1}$
$S(q)$
$q \in \mathbb{R}^{13\times1}$
$h_0(q)$
$h_0 \in \mathbb{R}^{6\times1}$
$q \in \mathbb{R}^{13\times1}$

Figure 8.2: Time-independent closed-loop block-diagram.

8.2.1 Switching logic

Figure 8.3 shows the collection of controllers available for synthesizing a walking motion. The controllers are organized into three blocks: for quiet standing (top), transition step (middle), and periodic walking (bottom). Terminal S_3 includes a state-machine that selects the appropriate control output. Each experiment begins with the robot in quiet standing mode; see Section 7.1. This mode remains active until the operator presses a button on the graphical user interface (GUI). When the button is pressed, the knees are commanded to straighten, causing the robot to pitch forward. From here on, all decisions are made without operator intervention.

When the CoM horizontal velocity exceeds $0.1[\frac{m}{s}]$, the transition step is begun. The controller is pre-selected from (7.2.7), (7.2.10), and (7.2.15). After one step, the control mode automatically switches to one of the pre-selected controllers for periodic walking given in (6.2.7), (6.3.5), and (6.4.2). The experiment ends when either the robot falls or the operator kills the power.

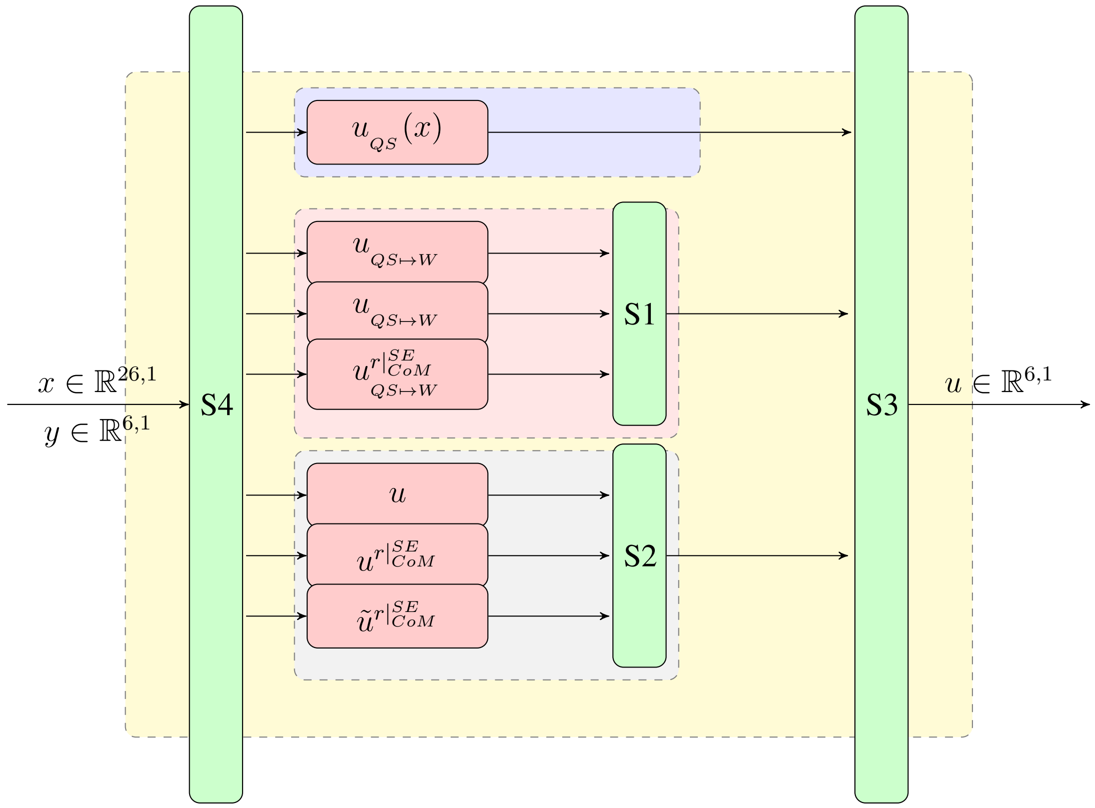


Figure 8.3: MARLO's walking controller Γ architecture.

8.2.2 Features added during experimentation

During the course of experimenting with the controller, two features were made available to the operator.

8.2.3 Double support delay

At the end of each step, the controller must swap the support leg from left to right or right to left. This leg swapping takes place when the spring deflection in the swing leg exceeds a pre-defined threshold. It was observed that while using spring deflection to detect when the swing impacting the ground worked well, it was nevertheless difficult to set a threshold for spring deflection that would ensure adequate ground reaction force on the swing leg before commanding it to take over the support of the robot (i.e., become the stance leg). The solution was to delay swapping for a fixed amount of time after impact was detected. This interval of time is called the "double support delay."

8.2.4 Regressed Feed-Forward Terms

Feed-forward is a well known means for reducing tracking error arising in PD controllers. Using the recorded torques from early walking experiments, it was possible to estimate off-line the nominal torques required for the transition step and for steady-state walking. Using NURBs and regression, such feed-forward torques were designed and added to the standard PD feedback terms of the controllers used in later experiments.

8.3 Autonomous Planar Walking with Passive Feet

The planar platform shown in Fig .8.4 uses a boom to constrain the motion of MARLO to the surface of a sphere. The resulting circular motion of the robot is different from a standard planar model in that the center of the hips traces out a circle on the floor. Consequently, the distance traveled by the inside leg during a step is, on average, shorter than the distance traveled by the outside leg. To reduce this effect, an offset value of $2[deg]$ is added to the right and left hip reference angles, causing both hips to move inwards, thus reducing the distance between the right and left leg ends in the frontal plane. The passive feet (prosthetic feet[1]) are attached to the robot so that it can stand upright with the legs parallel, just as it will have to do in the tether-free experiments.

8.3.1 Selected measurements and controller

The pitch angle of the torso with respect to the world frame was determined from the IMU. At the time of the experiment, the boom pitch encoder had been loaned to the group at Carnegie Mellon University and was thus unavailable. The controller architecture explained above is used. The experiment begins with the robot in quiet standing. The time-independent transition controller is used. A periodic walking controller with normal virtual constraints is selected because in planar mode, the hip angles are commanded to constant values. The designed walking speed is $0.3[\frac{m}{s}]$.

[1]Courtesy of B. Griffin

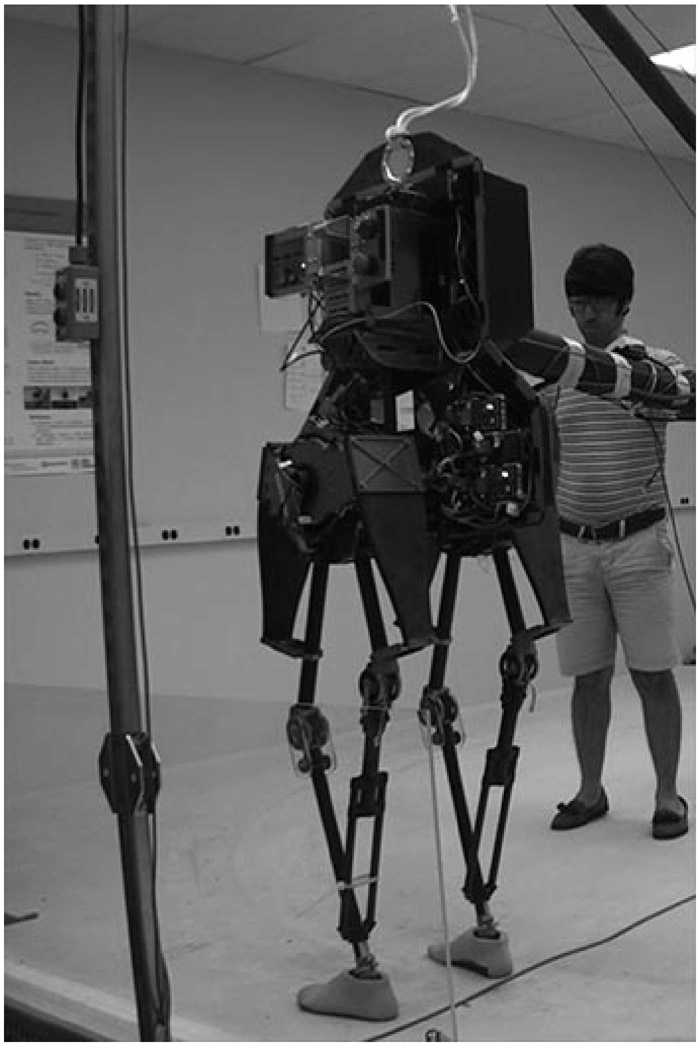


Figure 8.4: Planar walking experiment platform.

8.3.2 Results

The controller achieved autonomous stable walking, transitioning from the quiet standing mode and entering the basin of attraction of the periodic walking gait. Figure 8.5 shows the gait-timing variable versus time. The gait timing variable monotonically increases during each step and falls to zero at the beginning of the following.

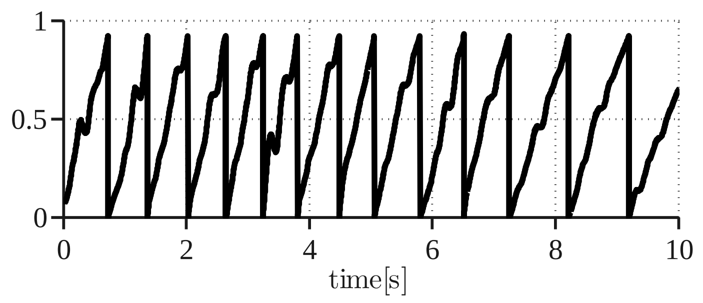
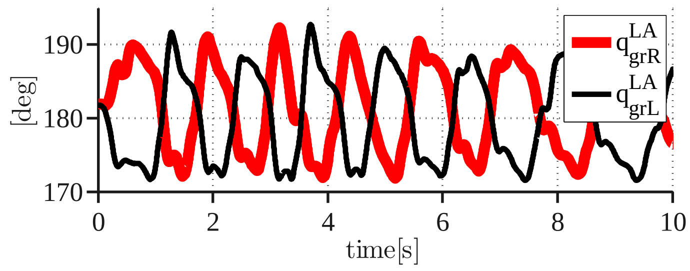
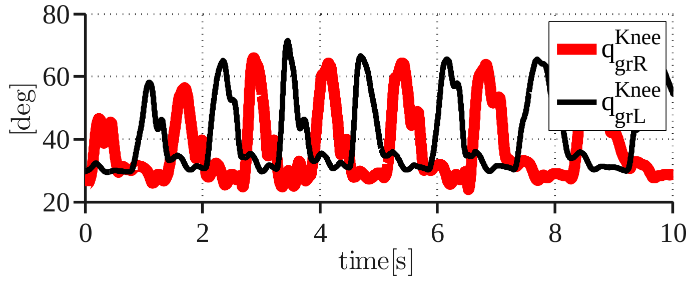


Figure 8.5: Gait-timing variable.

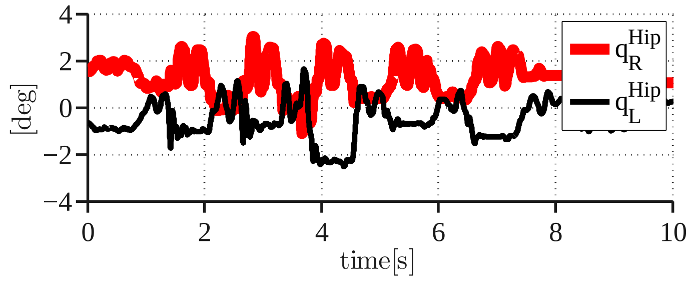
Figure 8.6 shows the leg angles, knee angles and hip angles for 13 steps and 10 seconds of walking. The transition from quiet standing to walking takes place at time zero. On the first step, q_{grR}^{LA} decreases to 175 $[deg]$ and q_{grL}^{LA} increases to 190 $[deg]$. On subsequent steps, this process alternates. The minimum knee angle is 25 $[deg]$ and the maximum knee angle is 70 $[deg]$. The hip angles vary between -3 $[deg]$ to 3 $[deg]$ and this is due to the shifting weight of the robot from one leg to the other. Between 3 seconds and 7 seconds, the gait appears to be in a steady-state regime. At 7 seconds the gait slows down, possibly due to a change in the height of the floor in the lab.



(a)



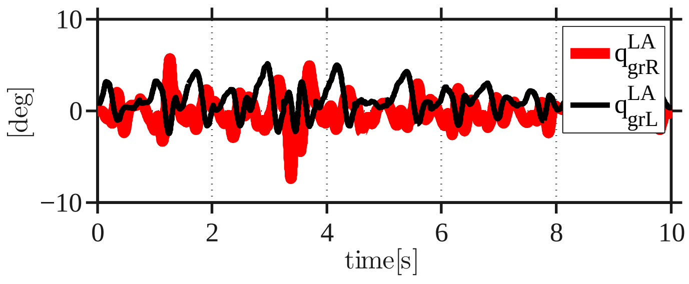
(b)



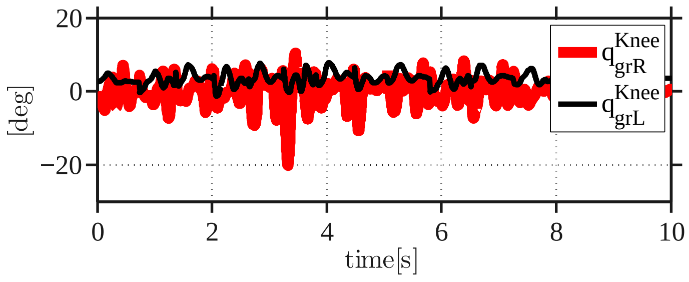
(c)

Figure 8.6: **(a)** q_{grR}^{LA}, q_{grL}^{LA}, **(b)** q_{grR}^{Knee}, q_{grL}^{Knee} **(c)** q_{R}^{Hip}, q_{L}^{Hip}.

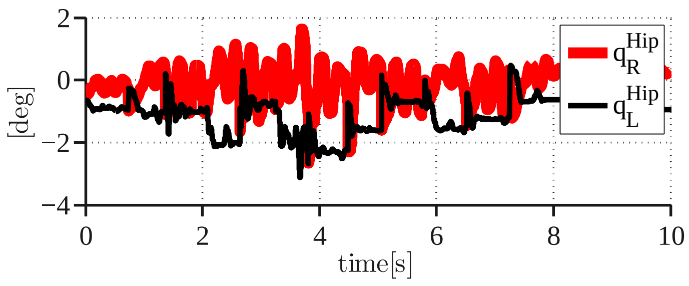
Figures 8.7 shows the tracking errors corresponding to the leg angles, knee angles and the hip angles. The maximum tracking error is associated with the left knee angle and is -20 $[deg]$. Despite this large error, the controller recovered and walking continued. Figure 8.8 shows the torque commands for the six actuators of the robot. The leg actuators are saturated to $\pm 4.5[N.m]$, whereas the maximum actuation torque on the hip actuator is $-2[N.m]$.



(a)

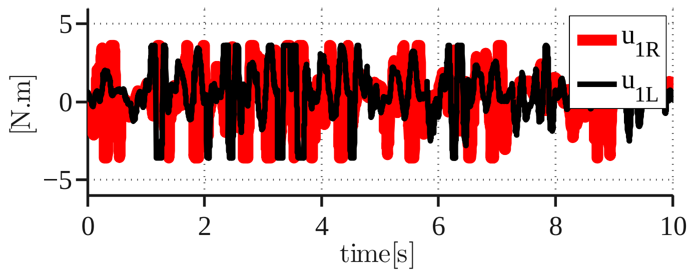


(b)

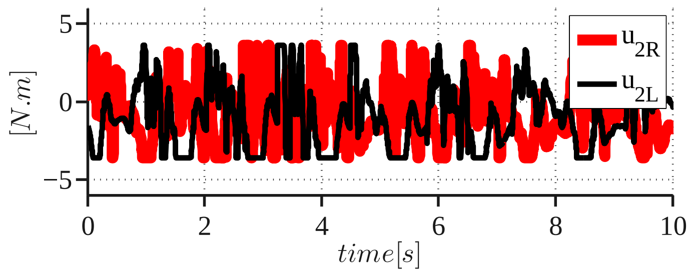


(c)

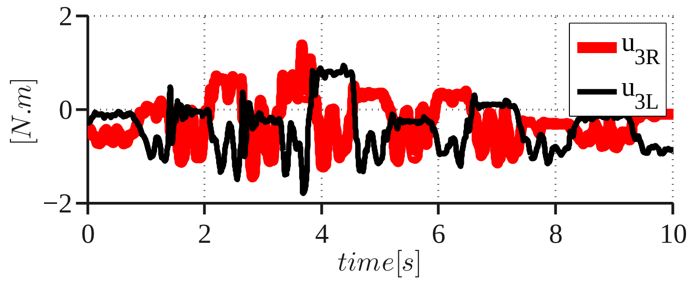
Figure 8.7: Tracking error for **(a)** q^{LA}_{grR}, q^{LA}_{grL}, **(b)** q^{Knee}_{grR}, q^{Knee}_{grL} **(c)** q^{Hip}_{R}, q^{Hip}_{L}.



(a)



(b)

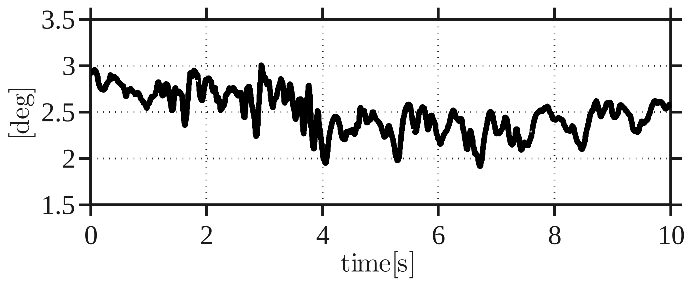


(c)

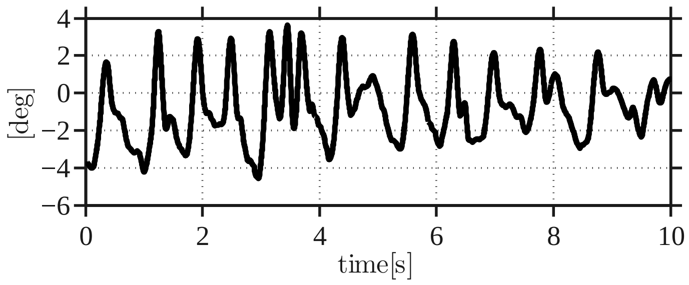
Figure 8.8: Actuation torque for the controller, **(a)** u_{1R}, u_{2R}, **(b)** u_{3R}, u_{1L} **(c)** u_{2L}, u_{3L}.

Figures 8.9 and 8.10 show the roll, pitch and yaw angles and their corresponding rates. In an ideal planar robot, the roll angle is not expected to vary. However, the end of the boom is actually moving on the surface of a sphere, and thus the roll angle varies between 2 $[deg]$ to 3 $[deg]$ as the height of the hips rises and falls; 1 $[deg]$ change in roll angle corresponds to roughly 3 cm of height change. The pitch angle starts at -4 $[deg]$ and ends at 2 $[deg]$ during the one-step transition from standing to walking. On the second step, the pitch angle varies from -4 $[deg]$ to 2 $[deg]$, and tends toward oscillating between -2 $[deg]$ and 2 $[deg]$.

Figure 8.11 and 8.12 show snapshots of walking on the boom. Figure 8.11 shows the single step transition from quiet standing to steady-state walking; Fig. 8.12 depicts the periodically stable planar walking after the transition.

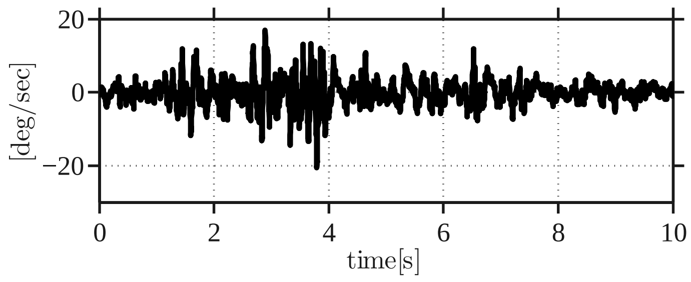


(a)

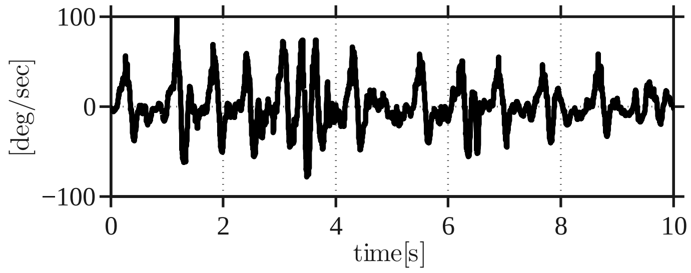


(b)

Figure 8.9: IMU angles, **(a)** roll, **(b)** pitch.

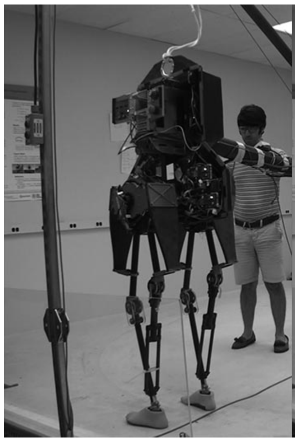


(a)

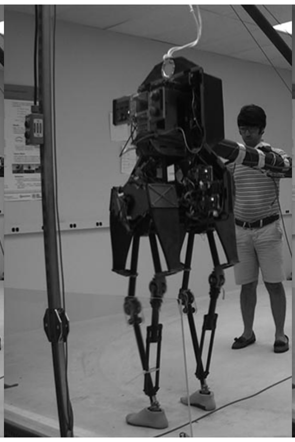


(b)

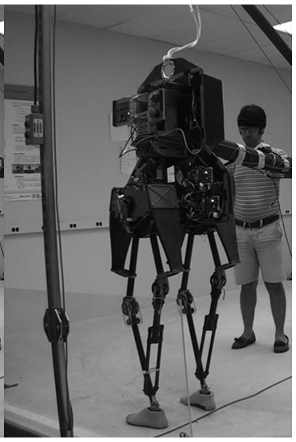
Figure 8.10: IMU rate angles, **(a)** roll-rate, **(b)** pitch-rate.



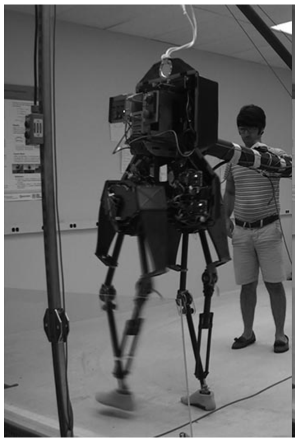
(t_3)



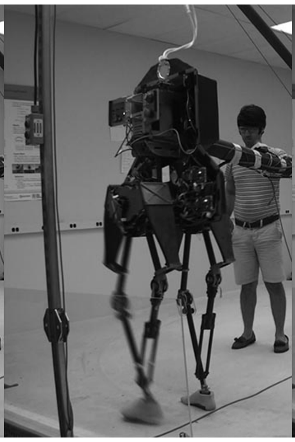
(t_2)



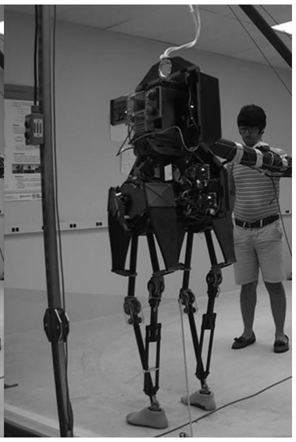
(t_1)



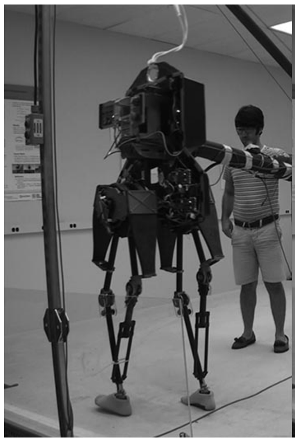
(t_6)



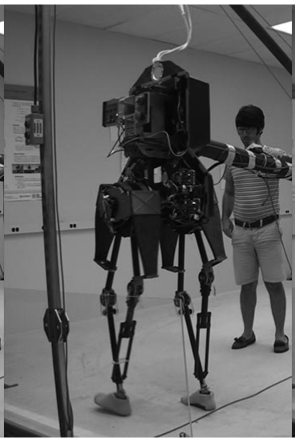
(t_5)



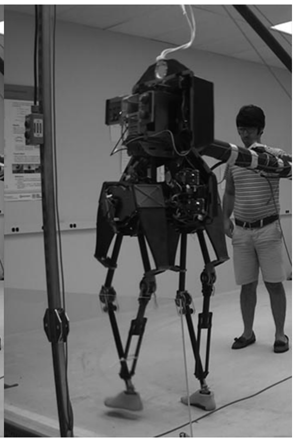
(t_4)



(t_9)

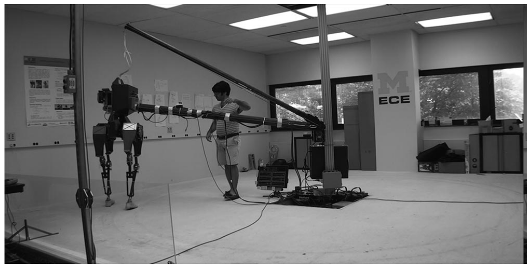


(t_8)

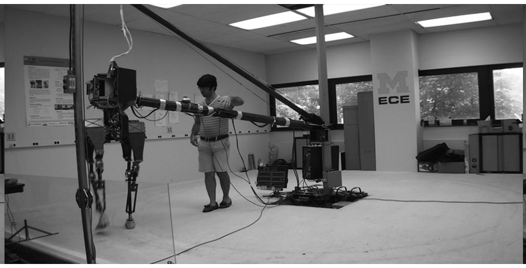


(t_7)

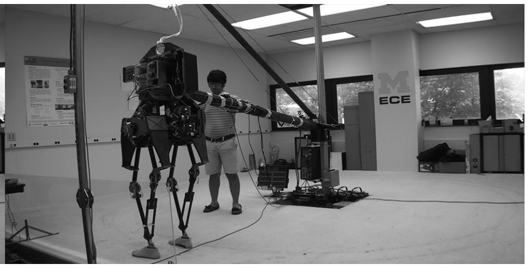
Figure 8.11: Planar transition step from quiet standing mode to periodic walking.



(t_3)

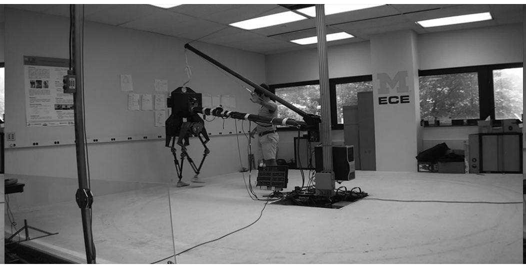


(t_2)

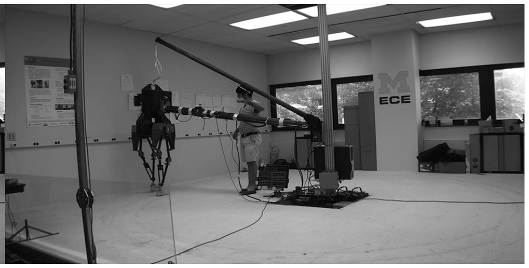


(t_1)

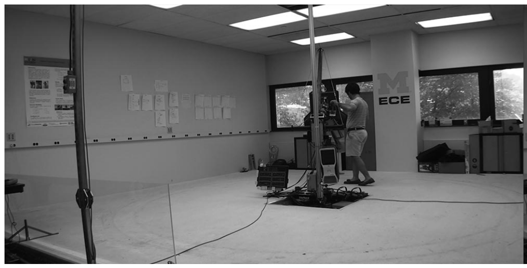
(t_6)



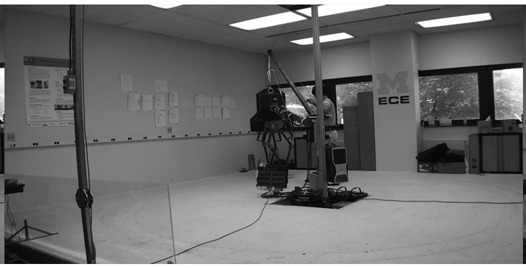
(t_5)



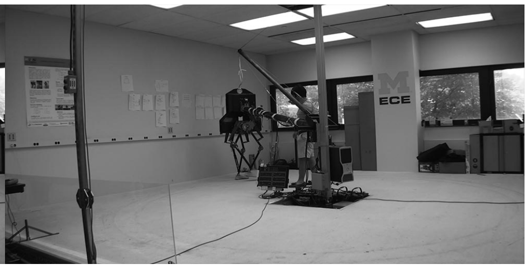
(t_4)



(t_9)



(t_8)



(t_7)

Figure 8.12: Planar stable periodic walking.

8.4 Autonomous 3D Walking with Passive Feet

The environment for the 3D experiments is shown in Fig. 8.13. An overhead gantry is used to provide a safety line (combination of a metal chain and a nylon rope) to the robot. The slack in the line is adjusted so that it catches the robot before its knees hit the floor. The line is not supporting the robot; if anything, it is acting as a disturbance on the torso. In these experiments, the robot is neither carrying its own power nor computation. Power is supplied by five 12-volt car batteries setting on the floor of the laboratory and connected to the robot via a cable. While the robot is designed to operate with on-board LiPo batteries, the present set of batteries do not supply adequate current. The computer unit serving as the target for the xPC Target system is not located on the robot. It too sets on the floor of the laboratory and is connected to the robot via an ethernet cable.

The robot is shown standing on a shallow ramp with an angle of $2[deg]$. The shallow ramp makes it easier to initiate the gait. In addition, it is covered with a non-slip rubber material.

The angle of the torso with respect to the world frame is measured with an IMU that is attached to the torso. The passive feet (prosthetic feet) used in this platform facilitate the stabilization at the quiet standing mode by increasing the area of the support polygon. In addition, it is possible that the passive feet provide some lateral stability to the robot. The angles of the passive feet with respect to the supporting link are adjusted through a series of set-screws located in the connector mechanism of each prosthetic passive foot.

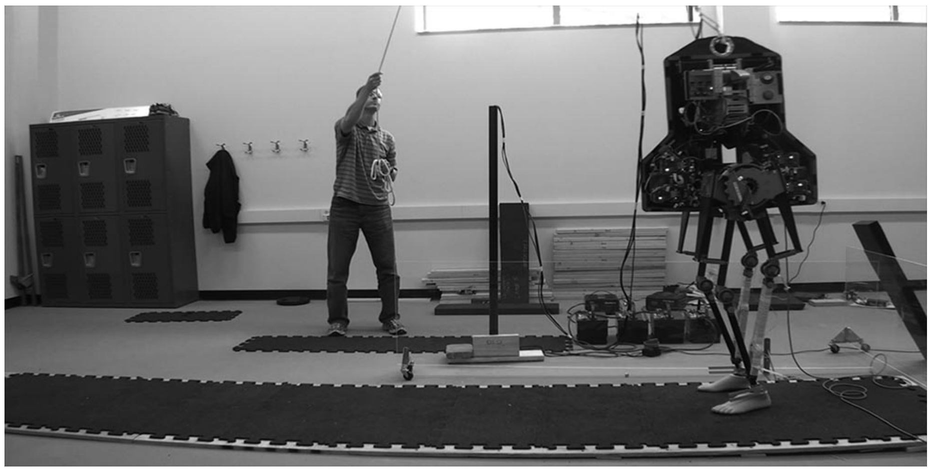


Figure 8.13: Tether-free walking experiment platform.

8.4.1 Six steps of walking

This section describes one particular experiment in detail. The time-independent transition controller based on the normal virtual constraints is selected by the operator. A periodic walking controller corresponding to an average speed of $0.3[\frac{m}{s}]$ and which enforces the normal virtual constraints is selected. In 3D, the robot initiated the gait from a quiet standing position and took five additional steps under the periodic-walking controller before falling on the 7th step.

Figure 8.14 shows the gait-timing variable versus time. The gait timing variable (nearly monotonically) increases during each step and falls to zero at the beginning of the next step.

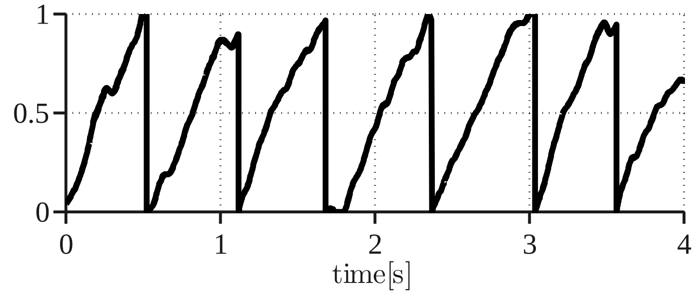
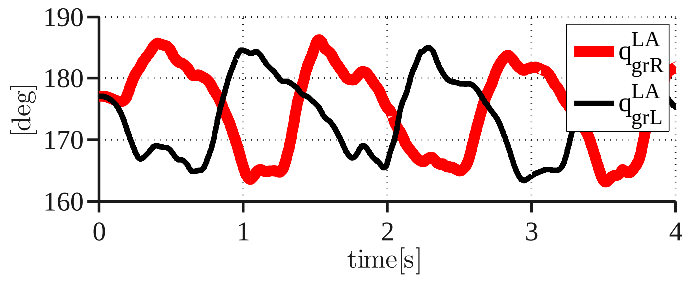
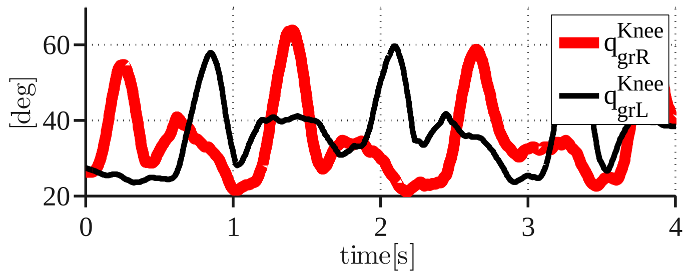


Figure 8.14: Gait-timing variable.

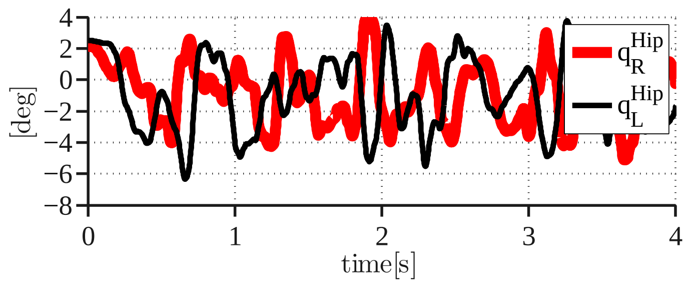
Figure 8.15 shows the leg angles, knee angles and hip angles for 7 steps and 4 seconds of walking. The transition from quiet standing to walking takes place at time zero. On the first step, q_{grR}^{LA} decreases to 175 $[deg]$ and q_{grL}^{LA} increases to 185 $[deg]$. On subsequent steps, this process alternates. The minimum knee angle is 20 $[deg]$ and the maximum knee angle is 65 $[deg]$. The hip angles vary between -6 $[deg]$ to 4 $[deg]$ and this is due to the shifting weight of the robot from one leg to the other.



(a)



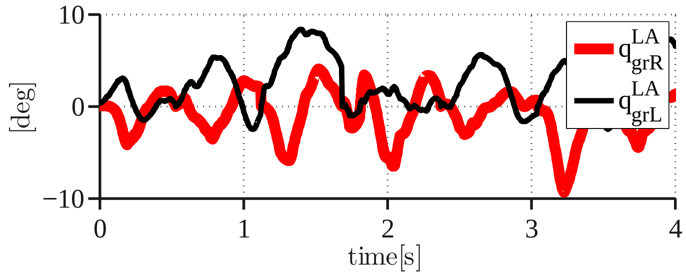
(b)



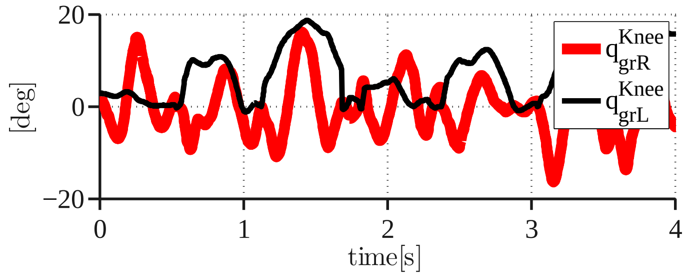
(c)

Figure 8.15: **(a)** q_{grR}^{LA}, q_{grL}^{LA}, **(b)** q_{grR}^{Knee}, q_{grL}^{Knee} **(c)** q_R^{Hip}, q_L^{Hip}.

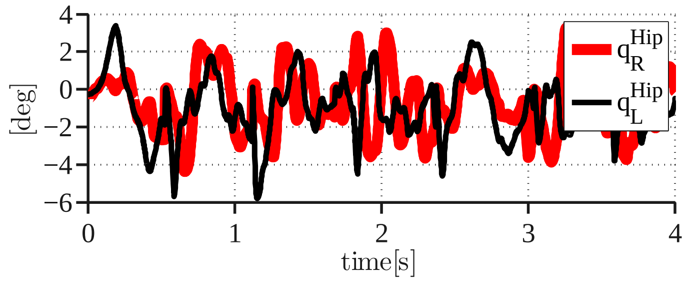
Figures 8.16 show the tracking errors corresponding to the leg angles, knee angles and the hip angles. The maximum tracking error is associated to the right knee angle and is 20 $[deg]$. Despite this large error, the controller recovered and walking was continued. Figure 8.17 shows the torque commands for the six actuators of the robot. The leg actuators are saturated to $\pm 2[N.m]$ whereas the maximum actuation torque on the hip actuator is slightly more than $-2[N.m]$.



(a)

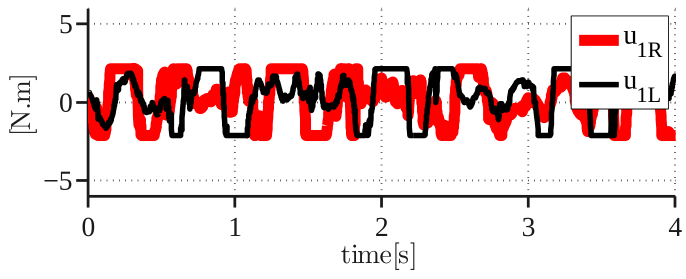


(b)

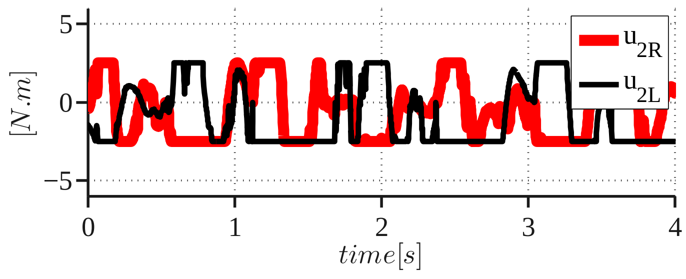


(c)

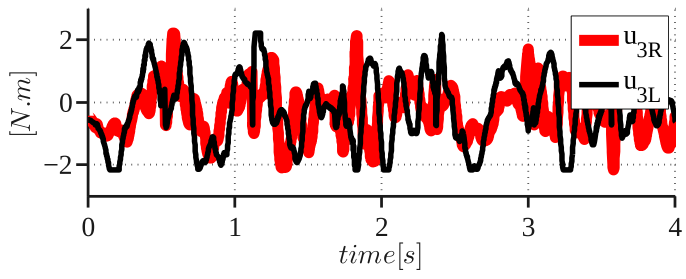
Figure 8.16: Tracking error for **(a)** q^{LA}_{grR}, q^{LA}_{grL}, **(b)** q^{Knee}_{grR}, q^{Knee}_{grL} **(c)** q^{Hip}_{R}, q^{Hip}_{L}.



(a)



(b)



(c)

Figure 8.17: Actuation torque for the controller, **(a)** u_{1R}, u_{2R}, **(b)** u_{3R}, u_{1L} **(c)** u_{2L}, u_{3L}.

Figures 8.18 and 8.19 show the roll and pitch angles and their corresponding rates. The roll angle mostly oscillates between $-4[deg]$ and $5[deg]$, but reaches a maximum of $8[deg]$ at $2[s]$, from which, however, the walking continues. The pitch angle starts from $-2[deg]$ and reaches $3[deg]$ at $0.25[s]$ before returning to $-2[deg]$ at the end of the first step. The pitch angle for the next consecutive steps follow a similar oscillating pattern, however, the average pitch angle increase over the steps, meaning the torso gradually rotates backward in the sagittal plane of walking. The roll rate generally oscillates between $-100[\frac{deg}{s}]$ and

$100[\frac{deg}{s}]$ with an exception being at $2[s]$ where the roll angle rate reaches $-200[\frac{deg}{s}]$. The pitch angle rate oscillates between $-80[\frac{deg}{s}]$ and $80[\frac{deg}{s}]$.

Figure 8.20 and 8.21 show snapshots of walking. Figure 8.20 shows the single step transition from quiet standing to steady-state walking and Fig. 8.21 depicts tether-free walking after the transition.

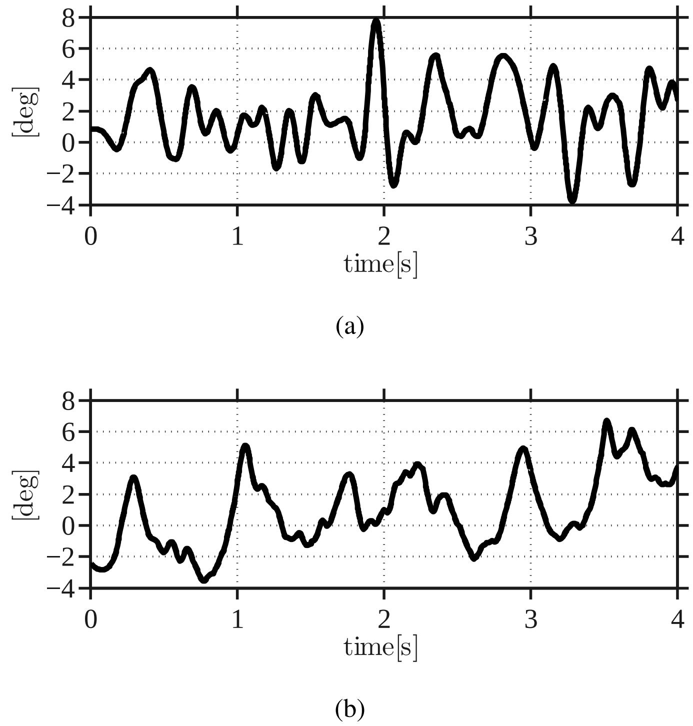
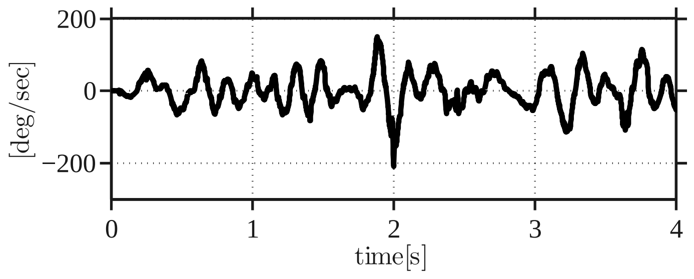
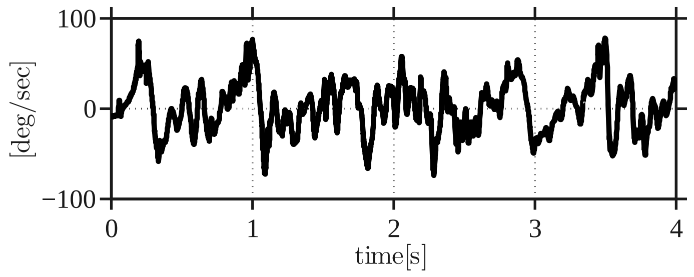


Figure 8.18: IMU angles, **(a)** roll, **(b)** pitch.

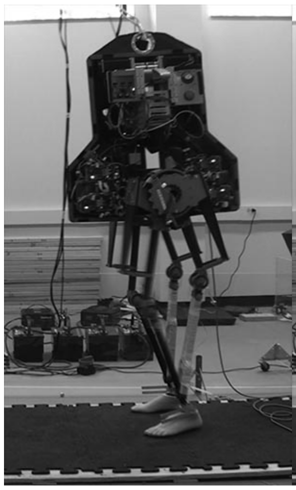


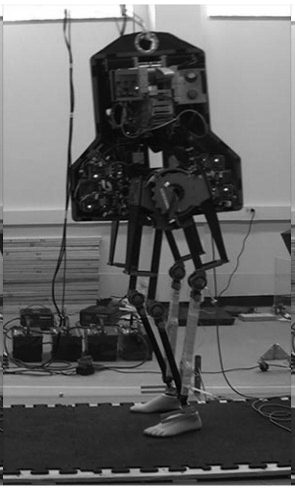
(a)

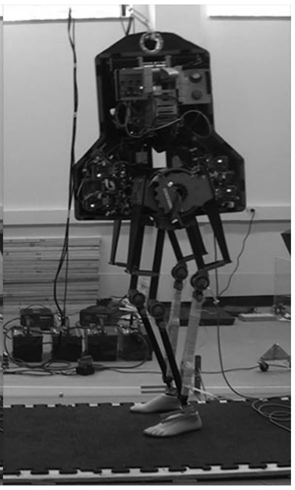


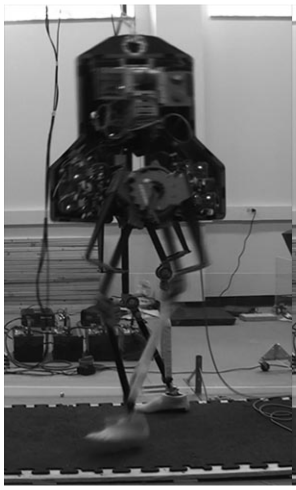
(b)

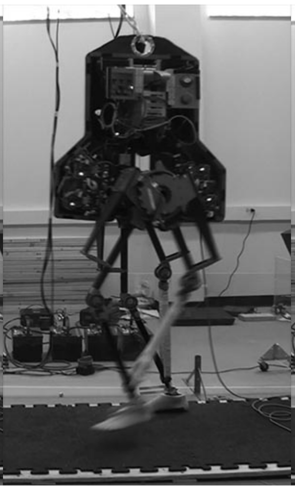
Figure 8.19: IMU rate angles, **(a)** roll-rate, **(b)** pitch-rate.

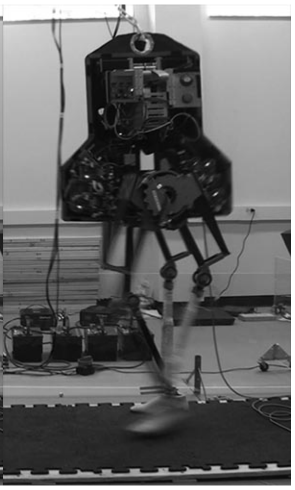

(t_3)

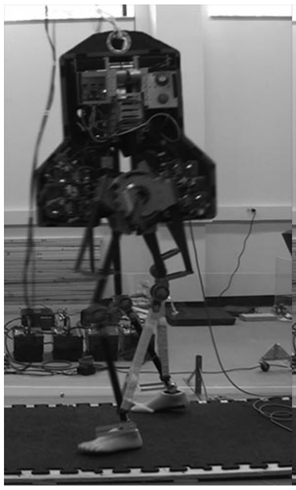

(t_2)

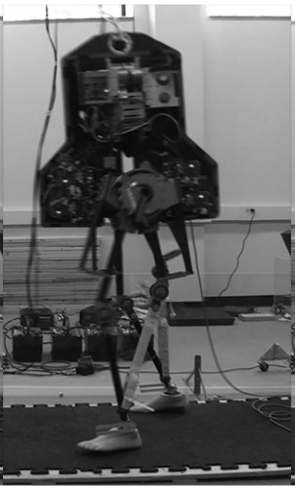

(t_1)


(t_6)


(t_5)


(t_4)


(t_9)


(t_8)

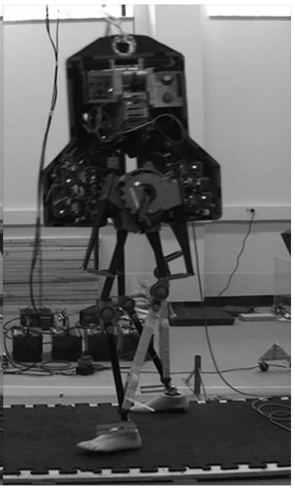

(t_7)

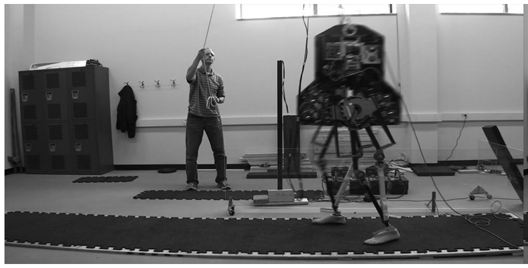
Figure 8.20: Tether-free transition step from quiet standing mode to periodic walking.

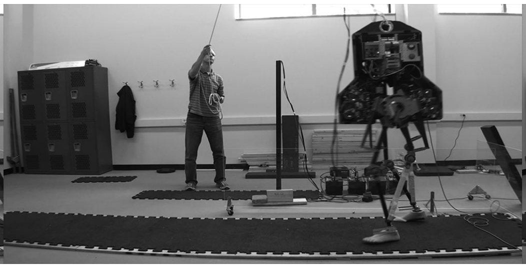
8.4.2 Discussion

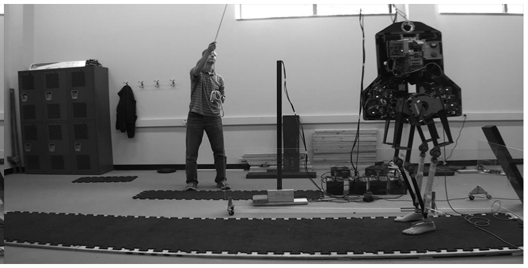
Several tether-free walking experiments with MARLO revealed natural and human-like steps, which were initiated autonomously without the intervention of an operator. Nevertheless, MARLO's current tether-free walking gaits are unstable and work is ongoing to stabilize them. In this section, a summary of the tether-free walking experiments is reported and compared to simulated results. Through the comparison between the simulated data and the experimental data, a hypothesis for the instability of the tether-free controller is presented.

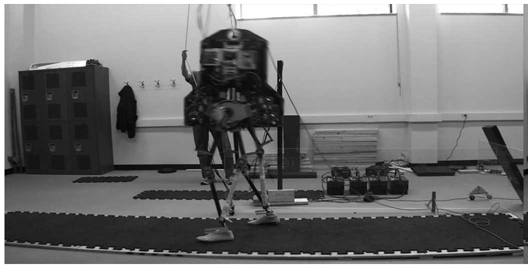
The controller (6.4.2) yielded an asymptotically stable periodic walking gait in our MATLAB simulation environment. Recall that this controller adjusts the lateral position of the swing leg so as to maintain a given distance between the CoM of the robot and the end of the swing leg. In simulation, the controller worked with both point feet on rigid ground and prosthetic feet on compliant ground. However, when the controller (6.4.2) was implemented on the actual robot, it lead to excessive vibrations of the hip actuation mechanism. In fact, the distance between the CoM and the swing leg-end is computed using the inverse kinematic equations obtained from the model of the robot. The derivative of this distance is "noisy" due to the numerical estimation of the angular velocity terms, which may have contributed to the observed vibrations. Consequently, the controller (6.2.7) which enforces the normal virtual constraints explained in Chapter VI was employed on the actual robot for tether-free walking.

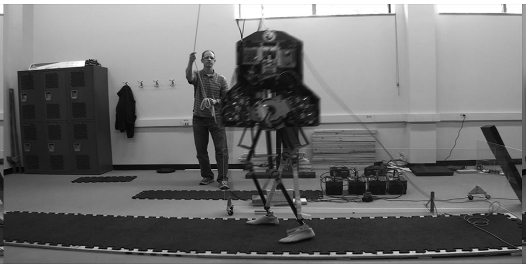
With the prosthetic feet on the robot, simulations of the untethered robot in closed-loop with the controller (6.2.7) were conducted with the compliant ground simulator. The results show that with a nominal torque limit of $5[N.m]$ for the actuators, stable walking is achieved. On the other hand, when the torque limit is reduced to $2[N.m]$, the robot falls after a few steps. The instability mechanism seems to be that the tracking error steadily increases on the stance hip, leading to the roll angle increasing and the robot toppling sideways. In the following, this behavior is compared to experimental data.

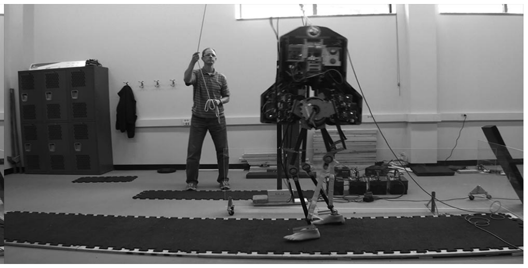

(t_3)

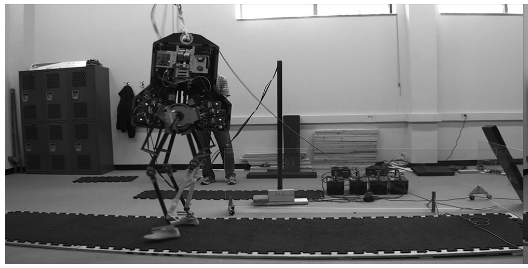

(t_2)

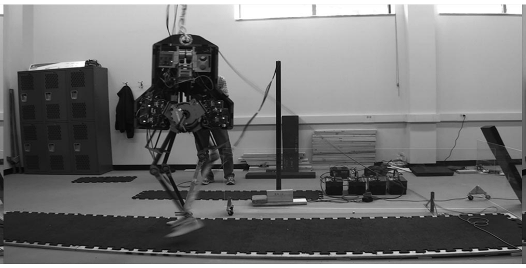

(t_1)


(t_6)


(t_5)


(t_4)


(t_9)


(t_8)

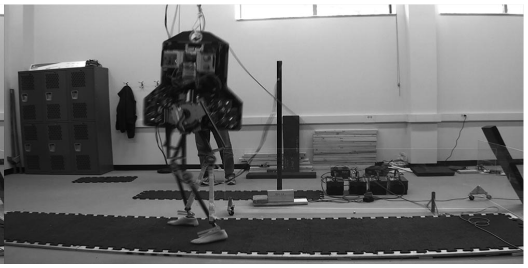
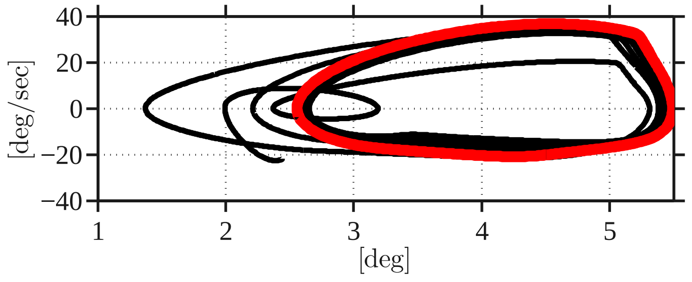

(t_7)

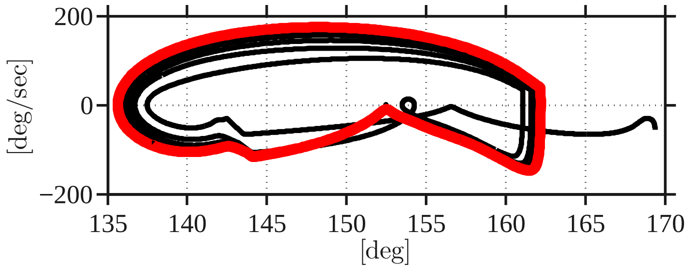
Figure 8.21: Six steps of 3D walking.

8.4.2.1 Planar walking

Even with all actuators limited to $2[N.m]$, the planar simulation model results in stable walking. Fig. 8.22 shows phase-portrait for q_x and q_{1R} over 15 steps. The simulated gait is converging to a periodic orbit. The persistence of stable walking in the sagittal plane model under severe restrictions on the torque is consistent with experiments.



(a) q_x



(b) q_{1R}

Figure 8.22: Planar phase-portrait (red curve is the fixed point).

8.4.2.2 3D walking

Simulation studies of the 3D model with (1) all actuators limited to $2[N.m]$ and (2) all actuators limited to $5[N.m]$ are compared against experimental results for *three steps proceeding the fall*. The last successful step is labeled $\mathcal{K}$, the step before that $\mathcal{K}-1$, and the one before that $\mathcal{K}-2$. Simulation with a limit of $2[N.m]$ results in an unstable walking gait where the robot topples sideways after four steps. While the harsh torque limits affect the tracking performance of each of the controlled variables, it is believed that the gait is most sensitive to errors in the stance hip frontal variable, q_3.

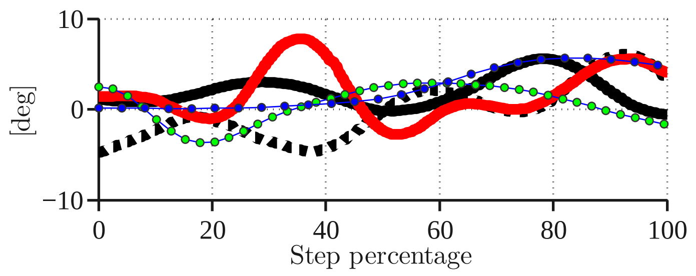
In Fig. 8.23, 8.24, 8.25 and 8.26 blue circles denote the simulation corresponding to a torque limit of $2[N.m]$ and the green circles correspond to $5[N.m]$, while the remaining three curves correspond to experimental data. The red line is the experiment reported in Section 8.4.1.

Figure 8.23 shows the roll angle. At 40% of step($\mathcal{K}$), the robot starts toppling sideways when the saturation is $2[N.m]$ (blue circles), and similarly for the two experiments represented by the black solid line and dashed black line; the roll angle increases, which means the robot is falling to the left side. When the torque limit is $5[N.m]$ (green circles) instability in the frontal plane is not observed. It is seen that the red curve follows the stable simulation. The red curve represents the evolution of the variables from the six step experiment presented in previous section. It was shown that the tracking errors associated to the six-step experiment are relatively small despite the limitations in the power amplifiers.

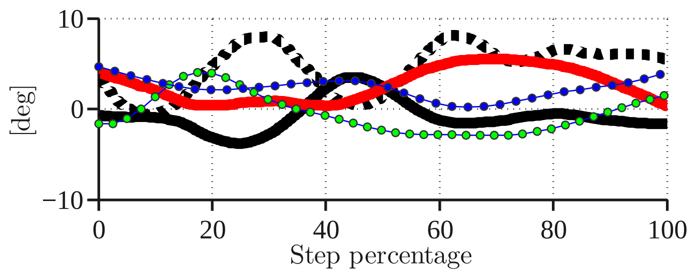
Turning to the pitch angles shown in Fig. 8.24, the experimental and simulated pitch angels vary between $-4[deg]$ and $6[deg]$ at the step($\mathcal{K} - 2$). In fact, the saturation level of the actuators does not affect the pitch angle evolution at steps($\mathcal{K} - 2$, $\mathcal{K} - 1$ and $\mathcal{K}$). It is seen that the six step experiment follows the simulated results, whereas, the black and dashed black experiments show a pitch instability that we believe is initiated by the roll instability in the robot.

Figures 8.25 and 8.26 indicate that when the saturation level is $2[N.m]$ (blue circles), the experimental and simulated tracking errors in the hip angles show a similar behavior. In Fig. 8.25, the hip tracking errors corresponding to the experiments and $2[N.m]$-simulation are larger than those of the $5[N.m]$-simulation. Moreover, it is seen that the six-step experiment (red curve) is in agreement with the $5[N.m]$-simulation (green circles). Figure 8.25 shows that at step ($\mathcal{K}$), $2[N.m]$-simulation -simulation (blue circles) and the experimental data have a large stance hip tracking error, where as the $5[N.m]$-simulation and the six-step experiment present significantly smaller stance hip tracking errors. Figure 8.26 shows the corresponding swing hip tracking errors. In contrast to the stance hip tracking errors, it is seen that the $2[N.m]$-simulation simulation (blue circles) and the $5[N.m]$-simulation simulation (green circles) are in agreement. In fact, since no force is acting at the swing leg-end, high saturation and low saturation yield similar tracking errors.

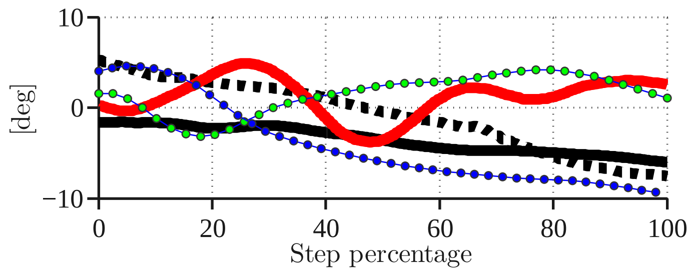
The above simulation study and the comparison made between the experimental and simulated results suggest that if the actuators (in particular, the hip actuators) fail to generate sufficient actuation torque, instability occurs. The minimum required actuation torque is unknown. Through simulation, however, an attempt was made to saturate the actuator output torque in order to model limitations dictated by the power amplifiers and motor drives. It was seen that when the generated torques are small, the hip tracking errors build up and consequently result in (frontal plane) instability.



(a) $\mathcal{K}-2$

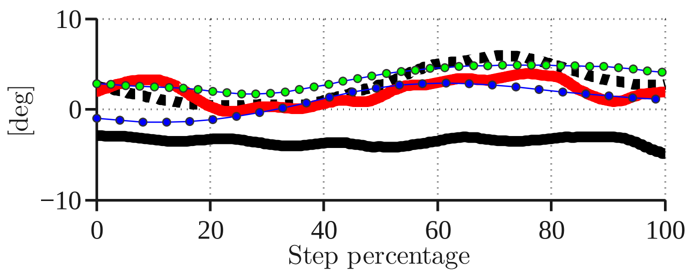


(b) $\mathcal{K}-1$

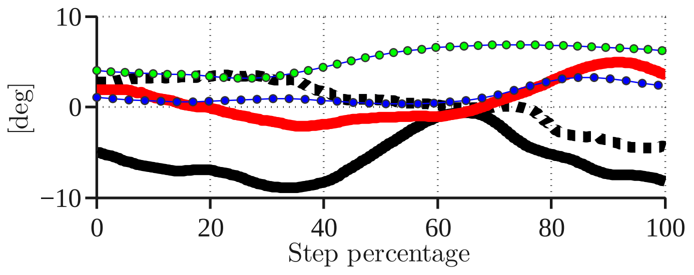


(c) $\mathcal{K}$

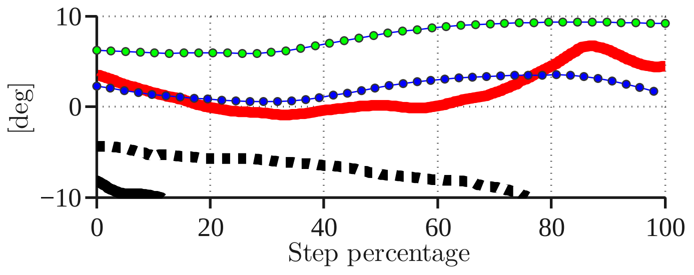
Figure 8.23: Roll angle evolution.



(a) $\mathcal{K}-2$

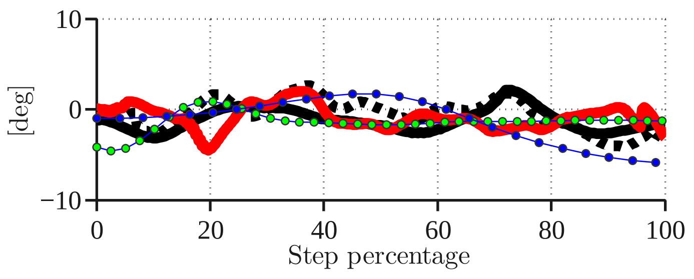


(b) $\mathcal{K}-1$

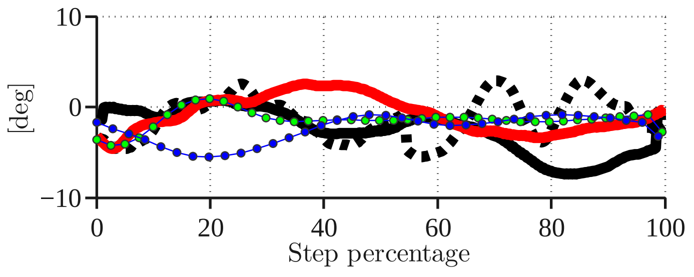


(c) $\mathcal{K}$

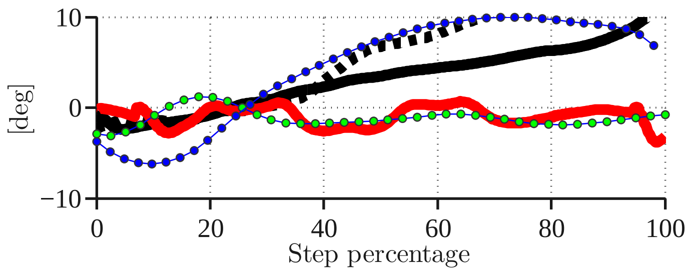
Figure 8.24: Pitch angle evolution.



(a) $\mathcal{K} - 2$

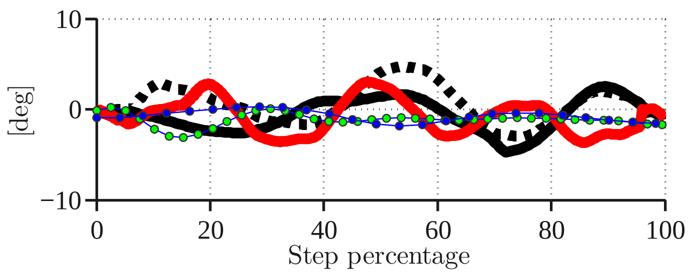


(b) $\mathcal{K} - 1$

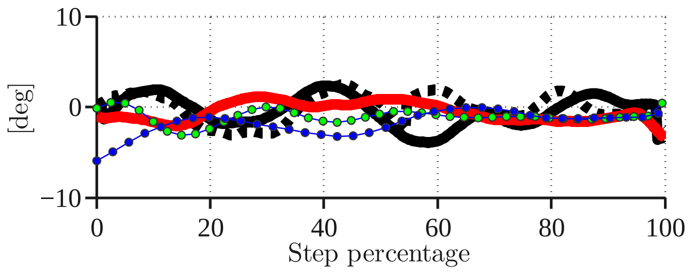


(c) $\mathcal{K}$

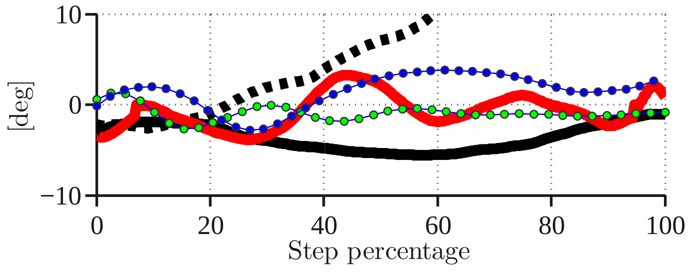
Figure 8.25: Hip angle tracking error corresponding to the stance leg.



(a) $\mathcal{K}-2$



(b) $\mathcal{K}-1$



(c) $\mathcal{K}$

Figure 8.26: Hip angle tracking error corresponding to the swing leg.

CHAPTER IX

Concluding Remarks

9.1 Summary of New Contributions

The new contributions of this study are two-fold: theoretical and experimental. The theoretical and experimental contributions tackled challenges in dynamic modeling, optimal gait design, controller design, and experimental testing on a 3D-bipedal robot, which has overall thirteen degrees of freedom and seven degrees of underactuation in single support phase.

Using the method of Lagrange, an unconstrained dynamic model of the robot was developed. Subsequently, four dynamic models (tether-free model with rigid ground, tether-free model with compliant ground, planar model with rigid ground, and planar model with compliant ground) were established after imposing proper holonomic constraints on the unconstrained dynamic model of the robot. It was demonstrated, through simulation, that the tether-free model and the planar model are in agreement.

Various choices of virtual constraints were identified and their implications on the stability of the associated hybrid zero dynamics (HZD) were analyzed. Due to the robot's 3D nature and the use of series elastic actuation, the state space dimension of its dynamic model was essentially double that of prior HZD-based studies, [70, 113]. The increased dimension was a considerable challenge in designing an HZD optimization algorithm for MARLO . While prior works [113] employed Bézier polynomials in their HZD optimization algorithms, a novel NURB-based family of virtual constraints is proposed in this work. A simulation study of optimal walking gaits for a range of walking velocities revealed energetic efficiency improvements when NURB curves are employed in comparison to Bézier curves. In particular, a NURB-based optimal walking gait with a cost of mechanical transport (i.e., cmt) of 0.05 at a nominal walking speed of $0.7[\frac{m}{s}]$ was obtained, which is comparable to the energetic efficiency of human-beings (cmt 0.05 for nominal walking speed of $1.0[\frac{m}{s}]$).

An HZD walking controller was designed for the tether-free model of the robot using a nominal choice of virtual constraints and the controller yielded unstable walking gaits. A virtual constraint proposed in [105] led to unstable walking gaits as well, proving that regulating the position of the CoM with respect to the swing foot will not guarantee the stability of a walking gait. Consequently, a dual-level event-based controller composed of a continuous loop and a discrete loop was developed that guaranteed stability of walking and the stability was checked using the method of Poincaré sections.

The HZD walking controller was implemented on the planarized version of MARLO at Oregon State University and a stable walking gait was demonstrated, similar to prior works by [113, 70]. The same controller also yielded stable walking gaits on the planarized copy of MARLO at the University of Michigan. During these experiments, the applicability of on-board power supplies was demonstrated when MARLO finished four laps using on-board batteries.

As noted in the literature review, several authors have demonstrated the suitability of HZD-based controllers in simulations. For example, Shih *et al.* [124] achieved asymptotically stable walking using HZD-based walking controller for a 3D-bipedal robot with 6 DoF and 2 DoU while [105, 125] showed asymptotically stable walking for a 3D-bipedal robot with 8 DoF and 2 DoU. In yet another work, Shih *et al.* [126] showed asymptotically stable walking for a 3D-bipedal robot with 9 DoF and 3 DoU. Nevertheless, these studies are simulation driven and the pioneering HZD-based experimental studies conducted by

[70] and [113] were on planar robots Rabbit (5 DoF and 1 DoU) and MABEL (7 DoF and 3 DoU), respectively. However, for the first time, in this study, an HZD-based controller was implemented experimentally on a 3D-bipedal robot with 13 DoF and 7 DoU. The controller yielded six human-like steps with walking speed of $0.3[\frac{m}{s}]$ in a 3D environment where an IMU was employed to measure the roll, pitch and yaw angles of the robot.

To begin the walking experiments, a gait initiation control strategy was designed and deployed. The strategy is composed of: (1) a quiet standing controller, which regulates the distance between CoM and CoP in order to balance the robot in a standing pose; and (2) a transition step controller, which takes the robot from a standing position to a posture and velocity "near" the fixed-point of a periodic walking gait. Two types of transitory steps, time-dependent and time-independent, were designed and tested. It was found that the time-dependent transition step may lead to premature impacts and scuffing of the swing leg. These problems were not observed in the time-independent transition step, and therefore it was mostly used in the experiments. The controller yielded a transitory step from a stand-still configuration to the fixed-point of periodic walking gait for both the planar and tether-free instantiations of MARLO. In the case of planar MARLO, the gait initiation control algorithm yielded asymptotically stable walking motions with a walking velocity $0.3[\frac{m}{s}]$ after the transitory step. The control policy completely removed any intervention of an operator during experiments. In the tether-free case, a maximum of 7 steps was achieved.

Raibert's famous hopper robots demonstrated tether-free underactuated running in the 1980's. In the intervening years, perhaps only Petman has demonstrated walking with partially actuated ankles, yielding two degrees of underactuation. The current work used totally passive ankles and compliant elements for energy efficiency. The analytical and experimental work demonstrate the feasibility of walking in a 3D environment using a high number of degrees of underactuation, namely seven. In addition, the work underlines the ability of HZD-based feedback designs to realize walking for robots with a wide range of morphologies, expanding from underactuated planar robots to underactuated 3D robots with compliant elements.

9.2 Perspective on Future Work

Attempts at embedding the controller with modified virtual constraints resulted in excessive vibrations in the hip actuator mechanism when the distance between the CoM and swing leg-end position was regulated by the controller. Experiments with the event-based controller yielded rapid unpredicted leg movements at the end of steps. Inverse kinematics is employed to compute the distance between CoM and swing leg-end after solving a series of symbolically defined equations during the controller run time. These computations are affected by the accuracy of the tether-free model and may not represent the actual distance between the CoM of the robot and the swing leg-end. An interesting first step to implement the controller with the modified virtual constraints would be to employ an electronic system capable of measuring the actual distance between the CoM of the robot and the swing leg-end, such as a “Vicon motion system”. In fact, the principal masses of the robot and the swing leg-end could be marked with reflectors; the associated Cartesian positions can be computed and sent to the controller to regulate the distance between the CoM and the swing leg-end.

The experimentally-demonstrated robustness of planar walking under HZD-based controllers for bipedal robots with various physiologies suggests that tether-free instability occurs in the frontal plane of walking. Current HZD-based control design considers a single gait-timing variable defined in the sagittal plane of walking and employed as a self-clocking mechanism to generate walking motions. In order to robustly compensate for perturbations in the frontal plane, explicit adjustment of the frontal hip angles is required. This can be done by introducing an independent frontal gait-timing variable associated to the frontal plane of walking where the hip joint motions are generated by enforcing virtual constraints which are driven by the frontal gait-timing variable.

Swing foot positioning with the current controller takes place through enforcement of the virtual constraints, which define a pre-computed ground clearance and angle of attack dependent on the value of the gait-timing variable. In various experiments, premature impact with improper angle of attack took place mainly because the foot positioning occurs with no knowledge of the configuration of the robot in the frontal plane. Embedding a ground clearance sensor in the feet may enable the controller to avoid premature impacts caused by frontal plane perturbations.

The dynamic model developed in this study captures the overall dynamic response of the robot. Nevertheless, the dynamics of the harmonic drives and the hip actuator mechanism, which reflect properties such as internal friction, and viscosity were not considered. Their consideration may lead to a more realistic model of the robot for gait design.

The energetic cost of mechanical transport used in this work represents the energetic efficiency associated with periodic walking on flat ground when there are no disturbances acting on the robot. In actual experiments, disturbances, such as unevenness in the terrain, act on the robot and perhaps, by incorporating the effects of such disturbances in HZD optimization, more realistic gaits would be obtained.

APPENDICES

APPENDIX A

NURB C Code

Following source codes[1] are used in HZD optimization using NURB curves.

```
/*
    Subroutine to generate a B-spline open knot vector with multiplicity
    equal to the order at the ends.

    c          = order of the basis function
    n          = the number of defining polygon vertices
    nplus2     = index of x() for the first occurence of the maximum
       knot vector value
    nplusc     = maximum value of the knot vector -- $n + c$
    x()        = array containing the knot vector
*/

knot(n,c,x)

int n,c;
int *x;

{
```

[1]Courtesy of David Rogers [116].

```
  int nplusc,nplus2,i;

  nplusc = n + c;
  nplus2 = n + 2;

  x[1] = 0;
    for (i = 2; i <= nplusc; i++){
        if ( (i > c) && (i < nplus2) )
        x[i] = x[i-1] + 1;
      else
        x[i] = x[i-1];
    }
}
```

```
/*  Subroutine to generate a B-spline uniform (periodic) knot vector.

    c           = order of the basis function
    n           = the number of defining polygon vertices
    nplus2      = index of x() for the first occurence of the maximum
        knot vector value
    nplusc      = maximum value of the knot vector -- $n + c$
    x[]         = array containing the knot vector
*/

#include  <stdio.h>

knotu(n,c,x)

int n,c;
int *x;

{
    int nplusc,nplus2,i;
```

```

   nplusc = n + c;
   nplus2 = n + 2;

   x[1] = 0;
   for (i = 2; i <= nplusc; i++){
       x[i] = i-1;
   }
}
```

```
/*  Subroutine to generate rational B-spline basis functions--open knot
    vector

   C code for An Introduction to NURBS
   by David F. Rogers. Copyright (C) 2000 David F. Rogers,
   All rights reserved.

   Name: rbais
   Language: C
   Subroutines called: none
   Book reference: Chapter 4, Sec. 4. , p 296

   c         = order of the B-spline basis function
     d          = first term of the basis function recursion relation
     e          = second term of the basis function recursion relation
   h[]       = array containing the homogeneous weights
     npts       = number of defining polygon vertices
     nplusc     = constant -- npts + c -- maximum number of knot values
     r[]        = array containing the rationalbasis functions
                  r[1] contains the basis function associated with B1 etc.
     t          = parameter value
     temp[]     = temporary array
     x[]        = knot vector
```

```
*/

#include <stdio.h>

rbasis(c,t,npts,x,h,r)

int c,npts;
float t;
int *x;
float *h;
float *r;

{
  int nplusc;
  int i,j,k;
  float d,e;
  float sum;
  float temp[36];

  nplusc = npts + c;

/*     printf("knot vector is \n");
    for (i = 1; i <= nplusc; i++){
       printf(" %d %d \n", i,x[i]);
    }
    printf("t is %f \n", t);
*/

/* calculate the first order nonrational basis functions n[i] */

  for (i = 1; i<= nplusc-1; i++){
      if (( t >= x[i]) && (t < x[i+1]))
      temp[i] = 1;
```

```
        else
        temp[i] = 0;
  }

/* calculate the higher order nonrational basis functions */

  for (k = 2; k <= c; k++){
        for (i = 1; i <= nplusc-k; i++){
            if (temp[i] != 0)    /* if the lower order basis function is
                zero skip the calculation */
                d = ((t-x[i])*temp[i])/(x[i+k-1]-x[i]);
            else
          d = 0;

            if (temp[i+1] != 0)     /* if the lower order basis function
                is zero skip the calculation */
              e = ((x[i+k]-t)*temp[i+1])/(x[i+k]-x[i+1]);
            else
            e = 0;

            temp[i] = d + e;
     }
  }

  if (t == (float)x[nplusc]){   /*     pick up last point   */
    temp[npts] = 1;
  }
/*
  printf("Nonrational basis functions are \n");
  for (i=1; i<= npts; i++){
    printf("%f ", temp[i]);
  }
  printf("\n");
```

```
*/
/* calculate sum for denominator of rational basis functions */

  sum = 0.;
  for (i = 1; i <= npts; i++){
        sum = sum + temp[i]*h[i];
  }

/* form rational basis functions and put in r vector */

  for (i = 1; i <= npts; i++){
       if (sum != 0){
           r[i] = (temp[i]*h[i])/(sum);}
     else
         r[i] = 0;
  }
}
```

```
/*
  Test program for C code from An Introduction to NURBS
  by David F. Rogers. Copyright (C) 2000 David F. Rogers,
  All rights reserved.

  Name: trbasis.c
  Purpose: test the rational basis function routine
  Language: C
  Subroutines called: knot.c
  Book reference: Chapter 4, Ex. 4.1 , Alg. p 296
*/

#include <stdio.h>

main()
```

```
{
  int i, index;
  int c,npts,nplusc;
  int x[22];
  float t;
  float h[20];
  float r[20];
  float sum;
  float hnew;

  t = 0.;

  printf("Input number of polygon points and order separated by a space
     npts c ");
  scanf("%d %d",&npts,&c);

  nplusc = npts + c;

  for (i=1; i <= npts; i++){
    h[i] = 1.0;
  }

  printf("If any homogeneous weighting factor is not 1.0, then \n");
  printf("input number index and value separated by a space, i.e., index
      h[index] \n");
  printf("Enter 0 0 to skip ");
  scanf("%d %f",&index,&hnew);

  if (( index > 0) && (index <= npts)) h[index]=hnew;

  knot(npts,c,x);
    printf("knot vector is \n");
    for (i = 1; i <= nplusc; i++){
        printf(" %d ", x[i]);
      }
      printf("\n");

      printf("Homogeneous weighting vector is \n");
      for (i = 1; i <= npts; i++){
        printf(" %f ", h[i]);
      }
      printf("\n");

    while(( t >= 0.) && (t <= (float)x[nplusc])){
      printf("Input the parameter value t (Control C to end) ");
      scanf("%f", &t);
      printf("The parameter value t is %f\n",t);
      rbasis(c,t,npts,x,h,r);
      printf("Rational basis functions are \n");
      sum = 0;
      for (i = 1; i <= npts; i++){
        sum = sum + r[i];
      }
      for (i = 1; i <= npts; i++){
        printf("%f ", r[i]);
      }
      printf("\n");
      printf("Sum of the Rational Basis functions is %f \n",sum);
    }
}
```

```
/* Subroutine to generate B-spline basis functions and their derivatives
     for uniform open knot vectors.
   C code for An Introduction to NURBS
   by David F. Rogers. Copyright (C) 2000 David F. Rogers,
   All rights reserved.
```

```

Name: dbasis.c
Language: C
Subroutines called: none
Book reference: Section 3.10, Ex. 3.18, Alg. p. 283


  b1        = first term of the basis function recursion relation
  b2        = second term of the basis function recursion relation
  c         = order of the B-spline basis function
  d1[]      = array containing the derivative of the basis functions
              d1[1]) contains the derivative of the basis function
                 associated with B1 etc.
  d2[]      = array containing the derivative of the basis functions
              d2[1] contains the derivative of the basis function
                 associated with B1 etc.
  f1        = first term of the first derivative of the basis function
      recursion relation
  f2        = second term of the first derivative of the basis
     function recursion relation
  f3        = third term of the first derivative of the basis function
      recursion relation
  f4        = fourth term of the first derivative of the basis
     function recursion relation
  npts      = number of defining polygon vertices
  n[]       = array containing the basis functions
              n[1]) contains the basis function associated with B1 etc
                 .
  nplusc    = constant -- npts + c -- maximum knot value
  s1        = first term of the second derivative of the basis
     function recursion relation
  s2        = second term of the second derivative of the basis
     function recursion relation
```

```
    s3          = third term of the second derivative of the basis
        function recursion relation
    s4          = fourth term of the second derivative of the basis
        function recursion relation
    t           = parameter value
    temp[]      = temporary array
    x[]         = knot vector
*/

#include  <stdio.h>
#include  <math.h>

dbasis(c,t,npts,x,n,d1,d2)

int c,npts;
int *x;
float t;
float *n,*d1,*d2;

{

  int nplusc;
  int i,k;
  float b1,b2;
  float f1,f2,f3,f4;
  float s1,s2,s3,s4;
  float temp[200];     /* allows for 35 defining polygon vertices */
  float temp1[200];
  float temp2[200];

  nplusc = npts + c;

/*     zero the temporary arrays */
```

```

  for (i = 1; i <= nplusc; i++){
    temp[i] = 0.;
        temp1[i] = 0.;
        temp2[i] = 0.;
  }

/* calculate the first order basis functions n[i] */

  for (i = 1; i<= nplusc-1; i++){
      if (( t >= x[i]) && (t < x[i+1]))
      temp[i] = 1;
      else
      temp[i] = 0;
  }

  if (t == (float)x[nplusc]){   /*     pick up last point   */
    temp[npts] = 1;
  }

/* calculate the higher order basis functions */

  for (k = 2; k <= c; k++){
      for (i = 1; i <= nplusc-k; i++){
          if (temp[i] != 0)    /* if the lower order basis function is
              zero skip the calculation */
              b1 = ((t-x[i])*temp[i])/(x[i+k-1]-x[i]);
          else
        b1 = 0;

          if (temp[i+1] != 0)     /* if the lower order basis function
              is zero skip the calculation */
            b2 = ((x[i+k]-t)*temp[i+1])/(x[i+k]-x[i+1]);
            else
            b2 = 0;

/*          calculate first derivative */

            if (temp[i] != 0)        /* if the lower order basis function
                is zero skip the calculation */
                f1 = temp[i]/(x[i+k-1]-x[i]);
            else
                f1 = 0;

            if (temp[i+1] != 0)      /* if the lower order basis function
                is zero skip the calculation */
                f2 = -temp[i+1]/(x[i+k]-x[i+1]);
            else
                f2 = 0;

            if (temp1[i] != 0)       /* if the lower order basis function
                is zero skip the calculation */
                f3 = ((t-x[i])*temp1[i])/(x[i+k-1]-x[i]);
            else
                f3 = 0;

            if (temp1[i+1] != 0)     /* if the lower order basis function
                is zero skip the calculation */
                f4 = ((x[i+k]-t)*temp1[i+1])/(x[i+k]-x[i+1]);
            else
                f4 = 0;

/*          calculate second derivative */

            if (temp1[i] != 0)       /* if the lower order basis function
                is zero skip the calculation */
                s1 = (2*temp1[i])/(x[i+k-1]-x[i]);
            else
                s1 = 0;

            if (temp1[i+1] != 0)      /* if the lower order basis function
                is zero skip the calculation */
                s2 = (-2*temp1[i+1])/(x[i+k]-x[i+1]);
            else
                s2 = 0;

            if (temp2[i] != 0)       /* if the lower order basis function
                is zero skip the calculation */
                s3 = ((t-x[i])*temp2[i])/(x[i+k-1]-x[i]);
            else
                s3 = 0;

              if (temp2[i+1] != 0)    /* if the lower order basis function
                  is zero skip the calculation */
                s4 = ((x[i+k]-t)*temp2[i+1])/(x[i+k]-x[i+1]);
            else
                s4 = 0;

              temp[i] = b1 + b2;
            temp1[i] = f1 + f2 + f3 + f4;
            temp2[i] = s1 + s2 + s3 + s4;
      }
    }

/* put in n array */

    for (i = 1; i <= npts; i++) {
        n[i] = temp[i];
        d1[i] = temp1[i];
        d2[i] = temp2[i];
    }
}
```

```
/*  Subroutine to generate a rational B-spline curve using an uniform
    open knot vector

  C code for An Introduction to NURBS
  by David F. Rogers. Copyright (C) 2000 David F. Rogers,
  All rights reserved.

  Name: rbspline.c
  Language: C
  Subroutines called: knot.c, rbasis.c, fmtmul.c
  Book reference: Chapter 4, Alg. p. 297

    b[]          = array containing the defining polygon vertices
                   b[1] contains the x-component of the vertex
                   b[2] contains the y-component of the vertex
                   b[3] contains the z-component of the vertex
  h[]      = array containing the homogeneous weighting factors
    k            = order of the B-spline basis function
    nbasis       = array containing the basis functions for a single
        value of t
    nplusc       = number of knot values
    npts         = number of defining polygon vertices
    p[,]         = array containing the curve points
                   p[1] contains the x-component of the point
                   p[2] contains the y-component of the point
                   p[3] contains the z-component of the point
    p1           = number of points to be calculated on the curve
    t            = parameter value 0 <= t <= npts - k + 1
    x[]          = array containing the knot vector
```

```
*/

rbspline(npts,k,p1,b,h,p)

int npts,k,p1;

float *b;
float *h;
float *p;

{
  int i,j,icount,jcount;
  int i1;
  int x[30];     /* allows for 20 data points with basis function of
      order 5 */
  int nplusc;

  float step;
  float t;
  float nbasis[20];
  float temp;


  nplusc = npts + k;

/*  zero and redimension the knot vector and the basis array */

  for(i = 0; i <= npts; i++){
     nbasis[i] = 0.;
  }

  for(i = 0; i <= nplusc; i++){
     x[i] = 0.;
```

```
    }

/* generate the uniform open knot vector */

  knot(npts,k,x);

/*
  printf("The knot vector is ");
  for (i = 1; i <= nplusc; i++){
    printf(" %d ", x[i]);
  }
  printf("\n");
*/

  icount = 0;

/*      calculate the points on the rational B-spline curve */

  t = 0;
  step = ((float)x[nplusc])/((float)(p1-1));

  for (i1 = 1; i1<= p1; i1++){

    if ((float)x[nplusc] - t < 5e-6){
      t = (float)x[nplusc];
    }

      rbasis(k,t,npts,x,h,nbasis);        /* generate the basis function
          for this value of t */
/*
    printf("t = %f \n",t);
    printf("nbasis = ");
    for (i = 1; i <= npts; i++){
        printf("%f    ",nbasis[i]);
    }
    printf("\n");
*/
    for (j = 1; j <= 3; j++){        /* generate a point on the curve */
      jcount = j;
      p[icount+j] = 0.;

      for (i = 1; i <= npts; i++){ /* Do local matrix multiplication */
        temp = nbasis[i]*b[jcount];
          p[icount + j] = p[icount + j] + temp;
/*
        printf("jcount,nbasis,b,nbasis*b,p = %d %f %f %f %f\n",jcount,
            nbasis[i],b[jcount],temp,p[icount+j]);
*/
        jcount = jcount + 3;
      }
    }
/*
    printf("icount, p %d %f %f %f \n",icount,p[icount+1],p[icount+2],p[
        icount+3]);
*/
      icount = icount + 3;
    t = t + step;
  }
}
```

```
/*  Subroutine to generate a rational B-spline curve using an uniform
    periodic knot vector

  C code for An Introduction to NURBS
  by David F. Rogers. Copyright (C) 2000 David F. Rogers,
  All rights reserved.

  Name: rbsplinu.c
  Language: C
  Subroutines called: knotu.c, rbasis.c, fmtmul.c
  Book reference: Chapter 4, Alg. p. 298

    b[]          = array containing the defining polygon vertices
                   b[1] contains the x-component of the vertex
                   b[2] contains the y-component of the vertex
                   b[3] contains the z-component of the vertex
  h[]      = array containing the homogeneous weighting factors
    k            = order of the B-spline basis function
    nbasis       = array containing the basis functions for a single
        value of t
    nplusc       = number of knot values
    npts         = number of defining polygon vertices
    p[,]         = array containing the curve points
                   p[1] contains the x-component of the point
                   p[2] contains the y-component of the point
                   p[3] contains the z-component of the point
    p1           = number of points to be calculated on the curve
    t            = parameter value 0 <= t <= npts - k + 1
    x[]          = array containing the knot vector
*/

rbsplinu(npts,k,p1,b,h,p)

int npts,k,p1;

float *b;
float *h;
float *p;

```

```
{
  int i,j,icount,jcount;
  int i1;
  int x[30];     /* allows for 20 data points with basis function of
      order 5 */
  int nplusc;

  float step;
  float t;
  float nbasis[20];
  float temp;


  nplusc = npts + k;

/*  zero and redimension the knot vector and the basis array */

  for(i = 0; i <= npts; i++){
      nbasis[i] = 0.;
  }

  for(i = 0; i <= nplusc; i++){
      x[i] = 0.;
    }

/* generate the uniform periodic knot vector */

  knotu(npts,k,x);

/*
  printf("The knot vector is ");
  for (i = 1; i <= nplusc; i++){
     printf(" %d ", x[i]);
  }
  printf("\n");

  printf("The usable parameter range is ");
  for (i = k; i <= npts+1; i++){
    printf(" %d ", x[i]);
  }
  printf("\n");
*/

  icount = 0;

/*      calculate the points on the rational B-spline curve */

  t = k-1;
  step = ((float)((npts)-(k-1)))/((float)(p1-1));

  for (i1 = 1; i1<= p1; i1++){

    if ((float)x[nplusc] - t < 5e-6){
      t = (float)x[nplusc];
    }

      rbasis(k,t,npts,x,h,nbasis);       /* generate the basis function
          for this value of t */
/*
    printf("t = %f \n",t);
    printf("nbasis = ");
    for (i = 1; i <= npts; i++){
      printf("%f  ",nbasis[i]);
    }
    printf("\n");
*/
```

```
    for (j = 1; j <= 3; j++){        /* generate a point on the curve */
      jcount = j;
      p[icount+j] = 0.;

      for (i = 1; i <= npts; i++){ /* Do local matrix multiplication */
        temp = nbasis[i]*b[jcount];
          p[icount + j] = p[icount + j] + temp;
/*
        printf("jcount,nbasis,b,nbasis*b,p = %d %f %f %f %f\n",jcount,
            nbasis[i],b[jcount],temp,p[icount+j]);
*/
        jcount = jcount + 3;
      }
    }
/*
    printf("icount, p %d %f %f %f \n",icount,p[icount+1],p[icount+2],p[
        icount+3]);
*/
      icount = icount + 3;
    t = t + step;
  }
}
```

```
/*
  Test program for C code from An Introduction to NURBS
  by David F. Rogers. Copyright (C) 2000 David F. Rogers,
  All rights reserved.

  Name: trbsplin.c
  Purpose: Test rational B-spline curve generator Chapter 4
  Language: C
  Subroutines called: rbspline.c
  Book reference:  Chapter 4, Ex. 4.1, Alg. p 297
```

```
*/
  main(){

  int i;
  int npts,k,p1;

  float b[31];  /* allows for up to 10  control vertices */
  float h[11];  /* allows for up to 10  control vertices */
  float p[103]; /* allows for up to 100 points on curve */

  npts = 5;
  k = 3;      /* third order, change for other orders */
  p1 = 11;    /* eleven points on curve */

  for (i = 1; i <= 3*npts; i++){
    b[i] = 0.;
  }

/*  set all homogeneous weighting factros to 1.0 */

    for (i=1; i <= npts; i++){
    h[i] = 1.0;
  }

/*  vary the homogeneous weighting factor 0, 0.25, 1.0, 5.0 */

  h[3] = 1;

  for (i = 1; i <= 3*p1; i++){
    p[i] = 0.;
  }

/*
```

```
  Define the control polygon, Ex. 4.1, p. 140 in the z=1 plane because
     this is three dimensional routine. x=b[1], y=b[2], z=b[3], etc.
*/
  b[1]=0;
  b[2]=0;
  b[3]=1;
  b[4]=1;
  b[5]=2;
  b[6]=1;
  b[7]=2.5;
  b[8]=0;
  b[9]=1;
  b[10]=4;
  b[11]=2;
  b[12]=1;
  b[13]=5;
  b[14]=0;
  b[15]=1;

  rbspline(npts,k,p1,b,h,p);

  printf("\nPolygon points\n");

  for (i = 1; i <= 3*npts; i=i+3){
     printf(" %f %f %f \n",b[i],b[i+1],b[i+2]);
  }

  printf("\nHomogeneous weighting vector is \n");
  for (i = 1; i <= npts; i++){
     printf(" %f ", h[i]);
  }
  printf("\n");


  printf("\nCurve points\n");

  for (i = 1; i <= 3*p1; i=i+3){
    printf(" %f %f %f \n",p[i],p[i+1],p[i+2]);
  }
}
```

```
/*
  Test program for C code from An Introduction to NURBS
  by David F. Rogers. Copyright (C) 2000 David F. Rogers,
  All rights reserved.

  Name: trbspliu.c
  Purpose: Test periodic rational B-spline curve generator Chapter 4
  Language: C
  Subroutines called: rbspline.c
  Book reference:  Chapter 4, Sec. 4.2, Fig. 4.6, Alg. p 298
*/
  main(){

  int i;
  int npts,k,p1;

  float b[31];  /* allows for up to 10  control vertices */
  float h[11];  /* allows for up to 10  control vertices */
  float p[103]; /* allows for up to 100 points on curve */

  npts = 5;
  k = 3;      /* third order, change for other orders */
  p1 = 11;    /* eleven points on curve */

  for (i = 1; i <= 3*npts; i++){
      b[i] = 0.;
   }

/*   set all homogeneous weighting factros to 1.0 */

      for (i=1; i <= npts; i++){
      h[i] = 1.0;
   }

/*   vary the homogeneous weighting factor 0, 0.25, 1.0, 5.0 */

   h[3] = 1;

   for (i = 1; i <= 3*p1; i++){
      p[i] = 0.;
   }

/*
   Define the control polygon, Ex. 4.1, p. 140 in the z=1 plane because
      this is three dimensional routine. x=b[1], y=b[2], z=b[3], etc.
*/
   b[1]=0;
   b[2]=0;
   b[3]=1;
   b[4]=1;
   b[5]=2;
   b[6]=1;
   b[7]=2.5;
   b[8]=0;
   b[9]=1;
   b[10]=4;
   b[11]=2;
   b[12]=1;
```

```
  b[13]=5;
  b[14]=0;
  b[15]=1;

  rbsplinu(npts,k,p1,b,h,p);

  printf("\nPolygon points\n");

  for (i = 1; i <= 3*npts; i=i+3){
    printf(" %f %f %f \n",b[i],b[i+1],b[i+2]);
  }

  printf("\nHomogeneous weighting vector is \n");
  for (i = 1; i <= npts; i++){
    printf(" %f ", h[i]);
  }
  printf("\n");


  printf("\nCurve points\n");

  for (i = 1; i <= 3*p1; i=i+3){
    printf(" %f %f %f \n",p[i],p[i+1],p[i+2]);
  }
}
```

```
#define S_FUNCTION_NAME BEZIER
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"

#include "bezier.h"

extern real_T my_alg(real_T u);  /* Declare my_alg as extern */
```

```c

/*
 * mdlInitializeSizes - initialize the sizes array
 */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetOptions(S,(SS_OPTION_USE_TLC_WITH_ACCELERATOR |
        SS_OPTION_WORKS_WITH_CODE_REUSE));
    ssSetNumSFcnParams( S, 0); /*number of input arguments*/

    if (!ssSetNumInputPorts(S, 2)) return;
    ssSetInputPortWidth(S, 0, 36);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    ssSetInputPortWidth(S, 1, 1);
    ssSetInputPortDirectFeedThrough(S, 1, 1);

    if (!ssSetNumOutputPorts(S,1)) return;
    ssSetOutputPortWidth(S, 0, 6);





    ssSetNumSampleTimes( S, 1);
}

/*
 * mdlInitializeSampleTimes - indicate that this S-function runs
 * at the rate of the source (driving block)
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}


/*
 * mdlOutputs - compute the outputs by calling my_alg, which
 * resides in another module, my_alg.c
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    // Inputs/outputs to the block
    int i;
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType wPtrs = ssGetInputPortRealSignalPtrs(S,1);


    real_T            *oPtrs    = ssGetOutputPortRealSignal(S,0);


    // Inputs/outputs to the function
    real_T afra[36];
    real_T s;


    for(i=0;i<36;i++){
        afra[i] = *(uPtrs[i]);
    }

    s = *(wPtrs[0]);

    bezier(afra, s, oPtrs);

 }
/*
 * mdlTerminate - called when the simulation is terminated.
 */
static void mdlTerminate(SimStruct *S)
{
}

#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c" /* MEX-file interface mechanism */
#else
#include "cg_sfun.h" /* Code generation registration function */
#endif
```

```
#define S_FUNCTION_NAME BEZIERD
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"

#include "bezierd.h"

/*
 * mdlInitializeSizes - initialize the sizes array
 */
static void mdlInitializeSizes(SimStruct *S)
{

    ssSetNumSFcnParams( S, 0); /*number of input arguments*/

    if (!ssSetNumInputPorts(S, 2)) return;
    ssSetInputPortWidth(S, 0, 36);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    ssSetInputPortWidth(S, 1, 1);
```

```c
    ssSetInputPortDirectFeedThrough(S, 1, 1);

    if (!ssSetNumOutputPorts(S,1)) return;
    ssSetOutputPortWidth(S, 0, 6);





    ssSetNumSampleTimes( S, 1);
}

/*
 * mdlInitializeSampleTimes - indicate that this S-function runs
 * at the rate of the source (driving block)
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}


/*
 * mdlOutputs - compute the outputs by calling my_alg, which
 * resides in another module, my_alg.c
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    // Inputs/outputs to the block
    int i;
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType wPtrs = ssGetInputPortRealSignalPtrs(S,1);


    real_T              *oPtrs     = ssGetOutputPortRealSignal(S,0);


    // Inputs/outputs to the function
    real_T afra[36];
    real_T s;


    for(i=0;i<36;i++){
        afra[i] = *(uPtrs[i]);
    }

    s = *(wPtrs[0]);

    bezierd(afra, s, oPtrs);

 }
/*
 * mdlTerminate - called when the simulation is terminated.
 */
static void mdlTerminate(SimStruct *S)
{
}

#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c" /* MEX-file interface mechanism */
#else
#include "cg_sfun.h" /* Code generation registration function */
#endif
```

```
#define S_FUNCTION_NAME BEZIERA
```

```
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"

#include "beziera.h"


/*
 * mdlInitializeSizes - initialize the sizes array
 */
static void mdlInitializeSizes(SimStruct *S)
{

    ssSetNumSFcnParams( S, 0); /*number of input arguments*/

    if (!ssSetNumInputPorts(S, 2)) return;
    ssSetInputPortWidth(S, 0, 36);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    ssSetInputPortWidth(S, 1, 1);
    ssSetInputPortDirectFeedThrough(S, 1, 1);

    if (!ssSetNumOutputPorts(S,1)) return;
    ssSetOutputPortWidth(S, 0, 6);





    ssSetNumSampleTimes( S, 1);
}

/*
 * mdlInitializeSampleTimes - indicate that this S-function runs
 * at the rate of the source (driving block)
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}


/*
 * mdlOutputs - compute the outputs by calling my_alg, which
 * resides in another module, my_alg.c
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    // Inputs/outputs to the block
    int i;
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType wPtrs = ssGetInputPortRealSignalPtrs(S,1);


    real_T           *oPtrs    = ssGetOutputPortRealSignal(S,0);


    // Inputs/outputs to the function
    real_T afra[36];
    real_T s;


    for(i=0;i<36;i++){
        afra[i] = *(uPtrs[i]);
    }

    s = *(wPtrs[0]);

    beziera(afra, s, oPtrs);

 }
/*
 * mdlTerminate - called when the simulation is terminated.
 */
static void mdlTerminate(SimStruct *S)
{
}

#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c" /* MEX-file interface mechanism */
#else
#include "cg_sfun.h" /* Code generation registration function */
#endif
```

```
#define S_FUNCTION_NAME NURB
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"

#define NPTS 20 // Num of control points,problem with dynamic alocation
    ..

/*
 * mdlInitializeSizes - initialize the sizes array
 */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetOptions(S,(SS_OPTION_USE_TLC_WITH_ACCELERATOR |
        SS_OPTION_WORKS_WITH_CODE_REUSE));
    ssSetNumSFcnParams( S, 0); /*number of input arguments*/

    if (!ssSetNumInputPorts(S, 5)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

     ssSetInputPortWidth(S, 1, 1);
    ssSetInputPortDirectFeedThrough(S, 1, 1);

     ssSetInputPortWidth(S, 2, 1);
    ssSetInputPortDirectFeedThrough(S, 2, 1);

    // for 25 npts
    ssSetInputPortWidth(S, 3, 3*NPTS+1);
    ssSetInputPortDirectFeedThrough(S, 3, 1);

     ssSetInputPortWidth(S, 4, 1);
    ssSetInputPortDirectFeedThrough(S, 4, 1);


    if (!ssSetNumOutputPorts(S,3)) return;
    ssSetOutputPortWidth(S, 0, 4);
    ssSetOutputPortWidth(S, 1, 4);
    ssSetOutputPortWidth(S, 2, 4);



    ssSetNumSampleTimes( S, 1);
}

/*
 * mdlInitializeSampleTimes - indicate that this S-function runs
 * at the rate of the source (driving block)
 */
```

```
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}


/*
 * mdlOutputs - compute the outputs by calling my_alg, which
 * resides in another module, my_alg.c
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    // Inputs/outputs to the block
    int i;
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType wPtrs = ssGetInputPortRealSignalPtrs(S,1);
    InputRealPtrsType xPtrs = ssGetInputPortRealSignalPtrs(S,2);
    InputRealPtrsType yPtrs = ssGetInputPortRealSignalPtrs(S,3);
    InputRealPtrsType zPtrs = ssGetInputPortRealSignalPtrs(S,4);



    real_T          *oPtrs   = ssGetOutputPortRealSignal(S,0);
    real_T          *pPtrs   = ssGetOutputPortRealSignal(S,1);
    real_T          *qPtrs   = ssGetOutputPortRealSignal(S,2);


    // Inputs/outputs to the function
    int npts,k,p1;
    float t;
    float b[3*NPTS+1];
    float p[4];
```

```
      float d[4];
      float dd[4];

      npts = *(uPtrs[0]);
     k = *(wPtrs[0]);
     p1 = *(xPtrs[0]);

      for(i=0;i<(3*NPTS+1);i++){
          b[i] = *(yPtrs[i]);
     }

      t = *(zPtrs[0]);



    fcn_bspline(npts,k,p1,b,t,p,d,dd);

    for(i=0;i<4;i++){
         oPtrs[i] = p[i];
         pPtrs[i] = d[i];
         qPtrs[i] = dd[i];
    }


  }
/*
 * mdlTerminate - called when the simulation is terminated.
 */
static void mdlTerminate(SimStruct *S)
{
}

#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c" /* MEX-file interface mechanism */
```

```
#else
#include "cg_sfun.h" /* Code generation registration function */
#endif
```

```
#include "mex.h"
void mexFunction(int nlhs, mxArray *plhs[],
        int nrhs, const mxArray *prhs[]) {

int n,c;
int *x;


    if (nrhs !=2)
        mexErrMsgTxt("Must have two input argument");
    if (nlhs !=1)
        mexErrMsgTxt("Must have  one output argument");



    n= mxGetScalar(prhs[0]);
    c = mxGetScalar(prhs[1]);

    /* Create an mxArray for the output data */
    plhs[0] = mxCreateDoubleMatrix(n+c, 1, mxREAL);
    x =  mxGetPr(plhs[0]);


    knot(n,c,x);


}
```

```
#include "mex.h"
```

```
void mexFunction(int nlhs, mxArray *plhs[],
          int nrhs, const mxArray *prhs[]) {

int n,c;
int *x;


    if (nrhs !=2)
        mexErrMsgTxt("Must have two input argument");
    if (nlhs !=1)
        mexErrMsgTxt("Must have  one output argument");



    n= mxGetScalar(prhs[0]);
    c = mxGetScalar(prhs[1]);

    /* Create an mxArray for the output data */
    plhs[0] = mxCreateDoubleMatrix(n+c, 1, mxREAL);
    x =  mxGetPr(plhs[0]);


    knotu(n,c,x)


}
```

```
#include "mex.h"
#define min(a, b) (((a) < (b)) ? (a) : (b))
#define max(a, b) (((a) > (b)) ? (a) : (b))


void mexFunction(int nlhs, mxArray *plhs[],
        int nrhs, const mxArray *prhs[]) {
    int c, npts,i;
    int *x;
    float *n,*h;
    float t;
    mwSignedIndex m,q;

    if (nrhs !=5)
        mexErrMsgTxt("Must have three four argument");
    if (nlhs !=1)
        mexErrMsgTxt("Must have one output argument");



    c = mxGetScalar(prhs[0]);
    t = mxGetScalar(prhs[1]);
    npts = mxGetScalar(prhs[2]);
    x =  mxGetPr(prhs[3]);
    h =  mxGetPr(prhs[4]);


    /* Check the dimensions */
    m = (mwSignedIndex)mxGetM(prhs[3]);
    q = (mwSignedIndex)mxGetN(prhs[3]);
    if (mxIsComplex(prhs[3]) ||
       (max(m,q) != npts+c+1) || (min(m,q) != 1)) {
        mexPrintf("\n arg[4] requires that input be a %d x 1 vector.",
            npts+c+1);
        mexErrMsgTxt("ERROR!!!!!!!!!!!!");
    }

     m = (mwSignedIndex)mxGetM(prhs[4]);
    q = (mwSignedIndex)mxGetN(prhs[4]);
    if (mxIsComplex(prhs[4]) ||
        (max(m,q) != npts+1) || (min(m,q) != 1)) {
         mexPrintf("\n arg[4] requires that input be a %d x 1 vector.",
             npts+1);
         mexErrMsgTxt("ERROR!!!!!!!!!!!!");
    }


    /* Create an mxArray for the output data */
    plhs[0] = mxCreateNumericMatrix(npts+1,1,mxSINGLE_CLASS,mxREAL);
    n = mxGetData(plhs[0]);
    rbasis(c,t,npts,x,h,n);



}
```

```
#include "mex.h"
#define min(a, b) (((a) < (b)) ? (a) : (b))
#define max(a, b) (((a) > (b)) ? (a) : (b))

void mexFunction(int nlhs, mxArray *plhs[],
         int nrhs, const mxArray *prhs[]) {

int npts,k,p1;
float *b;
float *h;
float *p;
mwSignedIndex m,n;
int i;

    if (nrhs !=5)
         mexErrMsgTxt("Must have five input argument");
    if (nlhs !=1)
        mexErrMsgTxt("Must have  one output argument");

    npts= mxGetScalar(prhs[0]);
    k = mxGetScalar(prhs[1]);
    p1 = mxGetScalar(prhs[2]);
    b = mxGetPr(prhs[3]);
    h = mxGetPr(prhs[4]);



    /* Check the dimensions */
    m = (mwSignedIndex)mxGetM(prhs[3]);
    n = (mwSignedIndex)mxGetN(prhs[3]);
    if (mxIsComplex(prhs[3]) ||
       (max(m,n) != 3*npts+1) || (min(m,n) != 1)) {
        mexPrintf("\n arg[4] requires that input be a %d x 1 vector.",3*
            npts+1);
        mexErrMsgTxt("ERROR!!!!!!!!!!!!");
    }

    m = (mwSignedIndex)mxGetM(prhs[4]);
    n = (mwSignedIndex)mxGetN(prhs[4]);
    if (mxIsComplex(prhs[4]) ||
       (max(m,n) != npts+1) || (min(m,n) != 1)) {
        mexPrintf("\n arg[5] requires that input be a %d x 1 vector.",
            npts+1);
        mexErrMsgTxt("ERROR!!!!!!!!!!!!");
    }


    /* Create an mxArray for the output data */
    plhs[0] = mxCreateNumericMatrix(3*p1+1, 1,mxSINGLE_CLASS, mxREAL);
    p =  mxGetData(plhs[0]);



    rbspline(npts,k,p1,b,h,p);






}
```

```
#include "mex.h"
#define min(a, b) (((a) < (b)) ? (a) : (b))
#define max(a, b) (((a) > (b)) ? (a) : (b))

void mexFunction(int nlhs, mxArray *plhs[],
        int nrhs, const mxArray *prhs[]) {

int npts,k,p1;
float *b;
float *h;
float *p;
mwSignedIndex m,n;
int i;

    if (nrhs !=5)
        mexErrMsgTxt("Must have five input argument");
    if (nlhs !=1)
        mexErrMsgTxt("Must have  one output argument");

    npts= mxGetScalar(prhs[0]);
    k = mxGetScalar(prhs[1]);
    p1 = mxGetScalar(prhs[2]);
    b = mxGetPr(prhs[3]);
    h = mxGetPr(prhs[4]);



    /* Check the dimensions */
    m = (mwSignedIndex)mxGetM(prhs[3]);
    n = (mwSignedIndex)mxGetN(prhs[3]);
    if (mxIsComplex(prhs[3]) ||
        (max(m,n) != 3*npts+1) || (min(m,n) != 1)) {
         mexPrintf("\n arg[4] requires that input be a %d x 1 vector.",3*
             npts+1);
         mexErrMsgTxt("ERROR!!!!!!!!!!!!");
    }

    m = (mwSignedIndex)mxGetM(prhs[4]);
    n = (mwSignedIndex)mxGetN(prhs[4]);
    if (mxIsComplex(prhs[4]) ||
        (max(m,n) != npts+1) || (min(m,n) != 1)) {
         mexPrintf("\n arg[5] requires that input be a %d x 1 vector.",
             npts+1);
         mexErrMsgTxt("ERROR!!!!!!!!!!!!");
    }


    /* Create an mxArray for the output data */
    plhs[0] = mxCreateNumericMatrix(3*p1+1, 1,mxSINGLE_CLASS, mxREAL);
    p =  mxGetData(plhs[0]);




    rbsplinu(npts,k,p1,b,h,p);




}
```

APPENDIX B

Passive Foot

The following passive feet are designed to replace prosthetic feet. The current prosthetic feet are large and not in agreement with the assumptions of point-foot gait design. Hence, the following design, which is significantly smaller than the current passive feet, was produced. The passive feet have a "roll-pitch-yaw" adjustable connector and these angles can be adjusted with respect to the shin link. They have shock absorbing springs on heel and toe. As such, smooth impacts are predicted.

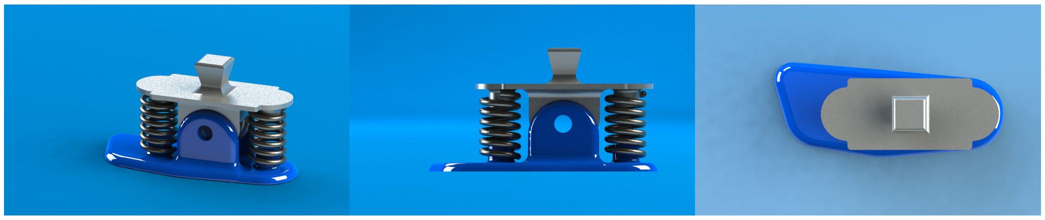
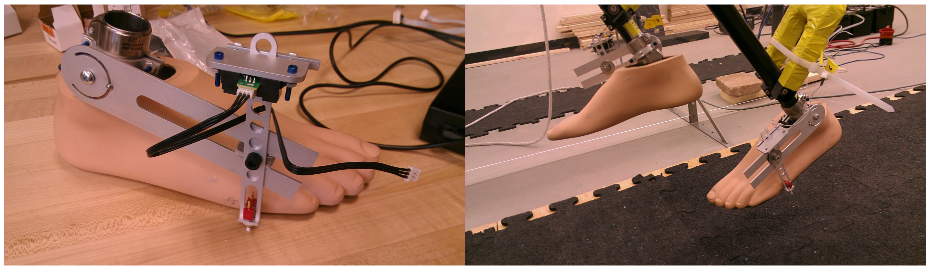


Figure B.1: Passive foot with small support surface, shock absorbing springs and roll-pitch-yaw adjustable connector.

APPENDIX C

Toe Switch, Sharp Sensor Support

Estimating the moment of impact is important for the controllers developed during the course of this study. Using the following mechanism, a Sharp distance sensor, which measures the height of the swing foot with respect to the ground, and a switch, which detects the moment of the impact, are mounted on the prosthetic feet. The components of the mount are designed such that the toe switch position is adjustable with respect to the prosthetic feet.



(a) Sharp sensor (b) toe switch

Figure C.1: **a**) shows a Sharp distance sensor mounted on the prosthetic foot. **b**) shows a toe switch mounted on the prosthetic foot.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Kajita, S., Yamaura, T., and Kobayashi, A., "Dynamic Walking Control of Biped Robot Along a Potential Energy Conserving Orbit," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 4, August 1992, pp. 431–437.

[2] Furusho, J. and Sano, A., "Sensor-based control of a nine-link biped," *International Journal of Robotics Research*, Vol. 9, No. 2, 1990, pp. 83–98.

[3] Katoh, R. and Mori, M., "Control method of biped locomotion giving asymptotic stability of trajectory," *Automatica*, Vol. 20, No. 4, 1984, pp. 405–414.

[4] Miura, H. and Shimoyama, I., "Dynamic Walk of a Biped," *The International Journal of Robotics Research*, 1984.

[5] Pratt, J., Chee, M., Torres, A., Dilworth, P., and Pratt, G., "Virtual model control: an intuitive approach for bipedal locomotion," *International Journal of Robotics Research*, Vol. 20, No. 2, February 2001, pp. 129–143.

[6] Raibert, M., Brown, H., and Chepponis, M., "Experiments in Balance with a 3D One-Legged Hopping Machine," *IJRR*, Vol. 3, June 1984, pp. 75–92.

[7] Pratt, J., Dilworth, P., and Pratt, G., "Virtual Model Control of a Bipedal Walking Robot," *Proc. of the IEEE International Conference on Robotics and Automation, Albuquerque, N.M.*, April 1997, pp. 193–198.

[8] Spong, M. and Bullo, F., "Controlled Symmetries and Passive Walking," *Proc. of IFAC World Congress, Barcelona, Spain*, July 2002.

[9] Spong, M. and Bullo, F., "Controlled Symmetries and Passive Walking," *IEEE Transactions on Automatic Control*, Vol. 50, No. 7, July 2005, pp. 1025–1031.

[10] Grizzle, J., Abba, G., and Plestan, F., "Proving asymptotic stability of a walking cycle for a five DOF biped robot model," *2nd Int. Conf. on Climbing and Walking Robots, CLAWAR-99, Portsmouth, U.K.*, September 1999, pp. 69–81.

[11] Schner, G., Jiang, W., and Kelso, J., "A Synergetic Theory of Quadrupedal Gaits and Gait Transitions," *Journal of Theoretical Biology*, Vol. 142, February 1990, pp. 359–391.

[12] K, M., "Sustained Oscillations Generated by Mutually Inhibiting Neurons with Adaptation," *Biological Cybernetics*, Vol. 52, 1985, pp. 367–376.

[13] Williams, T. L., "Phase Coupling by Synaptic Spread in Chains of Coupled Neuronal Oscillators," *Science*, Vol. 258, October 1992, pp. 662–665.

[14] Buchanan, J., "Neural Network Simulations of Coupled Locomotor Oscillators in the Lamprey Spinal Cord," *Biological Cybernetics*, Vol. 66, October 1992, pp. 367–374.

[15] Wadden, T., Hellgren, J., Lansner, A., and Grillner, S., "Intersegmental Coordination in the Lamprey: Simulations Using a Network Model Without Segmental Boundaries," *Biological Cybernetics*, Vol. 76, 1997, pp. 1–9.

[16] Hellgren, J., Grillner, S., and Lansner, A., "Computer Simulation of the Segmental Neural Network Generating Locomotion in Lamprey by Using Populations of Network Interneurons," *Biological Cybernetics*, Vol. 68, June 1992, pp. 1–13.

[17] Traven, H., Brodin, L., Lansner, A., Ekeberg, O., Wallen, P., and Grillner, S., "Computer Simulations of NMDA and non-NMDA Receptor- Mediated Synaptic Drive: Sensory and Supraspinal Modulation of Neurons and Small Networks," *Journal of Neurophysiology*, Vol. 70, August 1993.

[18] Or, J., "A hybrid CPG-ZMP control system for stable walking of a simulated flexible spine humanoid robot," *Neural Networks*, Vol. 23, April 2010, pp. 452–460.

[19] Hnaffa, P., Scesab, V., Ouezdoub, B. B., and Bruneaub, B., "Real time implementation of CTRNN and BPTT algorithm to learn on-line biped robot balance: Experiments on the standing posture," *Control Engineering Practice*, Vol. 19, January 2011, pp. 89–99.

[20] Miller, W., "Real-Time Neural Network Control of Biped Walking Robot," *IEEE*, February 1994.

[21] Santos, S. and Campo, C., "Biped Locomotion Control with Evolved Adaptive Center-Crossing Continuous Time Recurrent Neural Networks," *Neurocomputing*, Vol. 86, June 2012, pp. 86–96.

[22] Delcomyn, F., "Walking Robots and the Central and Peripheral Control of Locomotion in Insects," *Autonomous Robots*, 1999, pp. 259–270.

[23] Dgallier, S., Santos, C., Righetti, L., and Ijspeert, A., "Movement Generation Using Dynamical Systems : A Humanoid Robot Performing A Drumming Task," *IEEE*, 2006.

[24] Cruse, H., Bartling, C., Dereifert, M., Schmitz, J., Brunn, D., Dean, J., and Kindermann, T., "Walking: A Complex Behavior Controlled by Simple Networks," *Adaptive Behavior*, Vol. 3, March 1995, pp. 385–418.

[25] Wu, X. and Maa, S., "CPG-based Control of Serpentine Locomotion of a Snake-like Robot," *Mechatronics*, Vol. 20, March 2010, pp. 326–334.

[26] Collins, S. H., Wisse, M., and Ruina, A., "A 3-D Passive Dynamic Walking Robot with Two Legs and Knees," *International Journal of Robotics Research*, Vol. 20, 2001, pp. 607–615.

[27] Taga, G., "Emergence of bipedal locomotion through entrainment among the neuro-musculo-skeletal system and the environment," *Physica D: Nonlinear Phenomena*, Vol. 75, August 1994, pp. 190–208.

[28] McGeer, T., "Passive dynamic walking," *International Journal of Robotics Research*, Vol. 9, No. 2, 1990, pp. 62–82.

[29] Fallis, G., "Walking Toy," patent 376-588, US Patents, ST. Joseph, Missouri, January 1888.

[30] Mochon, S. and McMahon, T., "Ballistic walking: an improved model," *Mathematical Biosciences*, 1980, pp. 241–260.

[31] Goswami, A., Espiau, B., and Keramane, A., "Limit cycles and their stability in a passive bipedal gait," *Proc. of the IEEE International Conference on Robotics and Automation, Minneapolis, MN.*, April 1996, pp. 246–251.

[32] Thuilot, B., Goswami, A., and Espiau, B., "Bifurcation and chaos in a simple passive bipedal gait," *Proc. of the IEEE International Conference on Robotics and Automation, Albuquerque, N.M.*, April 1997, pp. 792–798.

[33] Kuo, A. D., "Stabilization of lateral motion in passive dynamic walking," *International Journal of Robotics Research*, Vol. 18, No. 9, 1999, pp. 917–930.

[34] Collins, S. H., Wisse, M., and Ruina, A., "A three-dimensional passive-dynamic

walking robot with two legs and knees," *International Journal of Robotics Research*, Vol. 20, No. 7, July 2001, pp. 607–615.

[35] Collins, S. H., Ruina, A., Tedrake, R., and Wisse, M., "Efficient bipedal robots based on passive-dynamic walkers," *Science*, , No. 307, 2005, pp. 1082–1085.

[36] Ruina, A., "Cornell Ranger, 2011 4-legged bipedal robot," `http://ruina.tam.cornell.edu/research/topics/locomotion_and_robotics/ranger/Ranger2011/index.html`, November 2012.

[37] Vukobratović, M., Borovac, B., and Potkonjak, V., "ZMP: A Review of some Basic Misunderstandings," *International Journal of Humanoid Robotics*, Vol. 3, No. 2, June 2006, pp. 153–175.

[38] Vukobratovic, M., Borovac, B., Surla, D., and Stokic, D., "Scientific fundamentals of robotics 7: Biped locomotion." *Springer*, 1990.

[39] Vukobratovic, M. and Juricic, D., "Contribution to the Synthesis of Biped Gait," *IEEE Transactions on Bio Medical Engineering*, Vol. BME-16, 1969.

[40] Goswami, A., "Postural Stability of Biped Robots and the Foot Rotation Indicator(FRI) Point," *The International Jouranl of Robotics Research*, Vol. 18, 1999, pp. 523–533.

[41] Wieber, P. B., "On the stability of walking systems," *Proceedings of the International Workshop on Humanoid and Human Friendly Robotics*, 2002.

[42] Hemami, H. and Fransworth, R. L., "Postural and Gait Stability of a Planar Five Link Biped by Simulation," *IEEE Transactions on Automatic Control*, 1977.

[43] Arakawa, T. and Fukuda, T., "Natural Motion Generation of Biped Locomotion Robot Using Hierarchical Trajectory Generation Method Consisting of GA, EP Layers," *IEEE Internation Conference On Robotics and Automation*, 1997.

[44] Hirai, K., Hirose, M., Haikawa, Y., and Takenake, T., “The development of Honda humanoid robot,” *Proc. of the IEEE International Conference on Robotics and Automation, Leuven, Belgium*, May 1998, pp. 1321–1326.

[45] Lim, H. and Takanishi, A., “Biped walking robots created at Waseda University: WL and WABIA family,” *Philosophical Transactions of th Royal Society*, 2007, pp. 49–64.

[46] LIM, H., OGURA, Y., and TAKANISHI, A., “Locomotion pattern generation and mechanisms of a new biped walking machine,” *The Royal Society*, November 2007, pp. 273–288.

[47] Ogura, Y., Aikawa, H., Shimomura, K., Kondo, H., Morishima, A., Lim, H., and Takanishi, A., “Development of a New Humanoid Robot WABIAN-2,” *IEEE International Conference on Robotics and Automation*, May 2006.

[48] KIM, J., PARK, I., and OH, J., “Experimental realization of dynamic walking of the biped humanoid robot KHR-2 using zero moment point feedback and inertial measurement,” *Advanced Robotics*, Vol. 20, 2006, pp. 707–736.

[49] Johnnie - The TUM Biped Walking Robot, “`http://www.lbm.mw.tum.de/index.php?id=182&L=1`,” August 2008.

[50] Hemami, H. and Katbab, A., “Constrained Inverted Pendulum Model For Evaluating Upright Postural Stability,” *Journal of Dynamic Systems, Measurment and Control*, Vol. 104/343, 1982.

[51] Lee, T. and Liao, J. H., “Trajectory Planning And Control of A 3 Link Biped Robot,” *IEEE*, 1988.

[52] Hurmuzlu, Y., “Dynamics of bipedal gait - Part 1: objective functions and the contact

event of a planar five-link biped," *Journal of Applied Mechanics*, Vol. 60, June 1993, pp. 331–336.

[53] Yang, J., "A Control Study of a Kneeless Biped Locomotion System," *Journal of the Franklin Institute*, Vol. 33, 1993, pp. 125–143.

[54] Jalics, L., Hemami, H., and Clymer, B., "A Control Strategy for Terrain Adaptive Bipedal Locomotion," *Autonomous Robots*, Vol. 4, 1997, pp. 243–257.

[55] Lum, H. K., Zribi, M., and Soh, Y. C., "Planning and control of a biped robot," *International Journal of Engineering Science*, Vol. 37, 1997, pp. 1319–1349.

[56] Songlow, J. and Guo, W., "A simplified hybrid force position controller method for the walking robots," *Robotica*, Vol. 17, 1999, pp. 583–589.

[57] Park, J. H., "Impedance Control for Biped Robot Locomotion," *IEEE Transactions on Robotics and Automation*, Vol. 17, 2001.

[58] Taga, G., "A model of the neuro-musculo-skeletal system for human locomotion," *Biological Cybernetics*, Vol. 73, 1995, pp. 113–121.

[59] Furusho, J. and Sano, A., "Development of Biped Robot," *Advances in Psychology*, Vol. 78, 1991, pp. 277–303.

[60] Frank, A., "An Approach to the Dynamic Analysis and Synthesis of Biped Locomotion Machines," *Medi. and Biol. Enginnering*, Vol. 8, 1970, pp. 465–476.

[61] Saidouni, T. and Bessonnet, G., "Generating globally optimized sagittal gait cycles of biped robot," *Robatica*, Vol. 21, 2003, pp. 199–210.

[62] Takegaki, M. and Arimoto, S., "A Newfeedback Method for Dynamic Control of Manipulators," *J. Dyna. Syst., Measure., Control*, Vol. 102, 1981, pp. 119–125.

[63] Koditschek, D. E., “The Application of Total Energy as a Lyapunov Function for Mechanical Control Systems,” *Dynamics and Control of Multibody Systems*, Vol. 97, 1989, pp. 131–157.

[64] Ortega, R., “Energy shaping revisited,” *Proc. IEEE Conf. Control Applications*, Sep. 2000, pp. 121–126.

[65] Grizzle, J. W. and Marcus, S., “The structure of nonlinear control systems possessing symmetries,” *IEEE Transaction on Automatic Control*, Vol. 30, March 1985, pp. 248–258.

[66] Grizzle, J., Plestan, F., and Abba, G., “Poincaré’s Method for Systems with Impulse Effects: Application to Mechanical Biped Locomotion,” *IEEE Conf. on Decision and Control*, December 1999, p. 1.

[67] Isidori, A., *Nonlinear Control Systems: An Introduction*, Springer-Verlag, Berlin, Germany, 2nd ed., 1989.

[68] Westervelt, E. R., Grizzle, J. W., and Koditschek, D. E., “Zero Dynamics of Planar Biped Walkers with One Degree of Under Actuation,” *IFAC World Congress*, Barcelona, Spain, July 2002.

[69] Westervelt, E., Grizzle, J., and Koditschek, D., “Hybrid Zero Dynamics of Planar Biped Walkers,” *IEEE Transactions on Automatic Control*, Vol. 48, No. 1, January 2003, pp. 42–56.

[70] Chevallereau, C., Abba, G., Aoustin, Y., Plestan, F., Westervelt, E. R., Canudas-de-Wit, C., and Grizzle, J. W., “RABBIT: A Testbed for Advanced Control Theory,” *IEEE Control Systems Magazine*, Vol. 23, No. 5, October 2003, pp. 57–79.

[71] Westervelt, E. R., Buche, G., and Grizzle, J. W., “Experimental Validation of a Framework for the Design of Controllers that Induce Stable Walking in Planar

Bipeds," *International Journal of Robotics Research*, Vol. 24, No. 6, June 2004, pp. 559–582.

[72] Chevallereau, C., Westervelt, E., and Grizzle, J., "Asymptotically Stable Running for a Five-Link, Four-Actuator, Planar Bipedal Robot," *International Journal of Robotics Research*, Vol. 24, No. 6, June 2005, pp. 431 – 464.

[73] Morris, B., Westervelt, E., Chevallereau, C., Buche, G., and Grizzle, J., *Fast Motions Symposium on Biomechanics and Robotics*, chap. Achieving Bipedal Running with RABBIT: Six Steps toward Infinity, Lecture Notes in Control and Information Sciences, Springer-Verlag, Heidelberg, Germany, 2006, pp. 277–297.

[74] Sreenath, K., Park, H., and Grizzle, J., "Embedding Active Force Control within the Compliant Hybrid Zero Dynamics to Achieve Stable, Fast Running on MABEL," *IJRR*, 2011.

[75] Park, H.-W., Sreenath, K., Ramezani, A., and Grizzle, J., "Switching Control Design for Accommodating Large Step-down Disturbances in Bipedal Robot Walking," *Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 331–3450.

[76] Park, H.-W., Sreenath, K., Hurst, J. W., and Grizzle, J. W., "Identification of a Bipedal Robot With a Compliant Drivetrain: Parameter Estimation for Control Design," *Control Systems Magazine*, Vol. 31, No. 2, April 2011, pp. 63–88.

[77] Westervelt, E., Buche, G., and Grizzle, J., "Inducing Dynamically Stable Walking in an Underactuated Prototype Planar Biped," *Proc. of the IEEE International Conference on Robotics and Automation, New Orleans, LA*, May 2004, See [127] for a reprint.

[78] Sreenath, K., Park, H.-W., Poulakakis, I., and Grizzle, J., "Design and experimental implementation of a compliant hybrid zero dynamics controller for walking on MABEL," *IEEE Conference on Decision and Control*, 2010, pp. 280–287.

[79] Wight, D., Kubica, E., and Wang, D., "Introduction of the Foot Placement Estimator: A Dynamic Measure of Balance for Bipedal Robotics," *Journal of Computational and Nonlinear Dynamics*, Vol. 3, January 2008.

[80] Collins, J. and Luca, J. D., "Open-loop and closed-loop control of posture: a random-walk analysis of center-of-pressure trajectories," *National Center for Biotechnology Information*, Vol. 95(2), 1993, pp. 308–18.

[81] "EtherCAT," `http://www.ethercat.org/`.

[82] Grimes, J. A. and Hurst, J. W., "The design of ATRIAS 1.0 a unique monoped, hopping robot," *Proceedings of the 2012 International Conference on Climbing and walking Robots and the Support Technologies for Mobile Machines*, pp. 548–554.

[83] Blickhan, R., "The Spring Mass Model for Running and Hopping," *Journal of Biomechanics*, Vol. 22, No. 11-12, 1989, pp. 1217–1227.

[84] McMahon, T. A. and Cheng, G. C., "The Mechanics of Running: How Does Stiffness Couple with Speed?" *Journal of Biomechanics*, Vol. 23, 1990, pp. 65–78.

[85] Farley, C. T., Glasheen, J., and McMahon, T. A., "Running Springs: Speed and Animal Size," *Journal of Experimental Biology*, 1993, pp. 71–86.

[86] Full, R. J. and Farley, C. T., "Musculoskeletal Dynamics in Rhythmic Systems - A Comparative Approach to Legged Locomotion," *Biomechanics and Neural Control of Posture and Movement*, edited by J. M. Winters and P. E. Crago, Springer-Verlag, New York, 2000.

[87] Seyfarth, A., Geyer, H., Gunther, M., and Blickhan, R., "A Movement Criterion for Running," *Journal of Biomechanics*, Vol. 35, November 2001, pp. 649–655.

[88] Raibert, M., *Legged Robots That Balance*, MIT Press, Cambridge, Mass., 1986.

[89] Zeglin, G. and Brown, H. B., “Control of a Bow Leg Hopping Robot,” *IEEE International Conference on Robotics and Automation*, May 1998.

[90] Ahmadi, M. and Buehler, M., “Control Passive Dynamic Running Experiment with the ARL Monopod II,” Vol. 22, No. 5, Oct. 2006, pp. 974–986.

[91] Sreenath, K., Park, H.-W., Poulakakis, I., and Grizzle, J. W., “Compliant Hybrid Zero Dynamics Controller for achieving Stable, Efficient and Fast Bipedal Walking on MABEL,” *International Journal of Robotics Research*, Vol. 30, No. 9, August 2011, pp. 1170–1193.

[92] Sreenath, K., Park, H.-W., and Grizzle, J., “Design and Experimental Implementation of a Compliant Hybrid Zero Dynamics Controller with Active Force Control for Running on MABEL,” *Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 51–56.

[93] Park, H.-W., Ramezani, A., and Grizzle, J., “A Finite-State Machine for Accommodating Unexpected Large Ground-Height Variations in Bipedal Robot Walking,” *IEEE Transactions on Robotics*, 2013, pp. 45–50.

[94] Geyer, H., Seyfarth, A., and Blickhan, R., “Compliant leg behaviour explains the basic dynamics of walking and running,” *Proc. R. Soc. Lond. B*, Vol. 273, 2006, pp. 2861–2867.

[95] Hodgins, J. K. and Raibert, M. H., “Adjusting step length for rough terrain locomotion,” *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 3, June 1991, pp. 289–298.

[96] Ahmadi, M. and Buehler, M., “The ARL Monopod II Running Robot: Control and Energetics,” *IEEE International Conference on Robotics and Automation*, May 1999, pp. 1689–1694.

[97] Corp., H., "Asimo Humanoid Robot, http://world.honda.com/ASIMO/," .

[98] McGeer, T., "Stability and control of two-dimensional biped walking," Tech. Rep. 1, Center for Systems Science, Simon Fraser University, Burnaby, B.C., Canada, 1988.

[99] Pratt, J. and Pratt, G., "Exploiting Natural Dynamics in the Control of a Planar Bipedal Walking Robot," *Proceedings of the Thirty-Sixth Annual Allerton Conference on Communication, Control, and Computing*, September 1998.

[100] Playter, R., Buehler, M., and Raibert, M., "BigDog," *Proceedings of SPIE International Society for Optical Engineering*, edited by G. R. Gerhart, C. M. Shoemaker, and D. W. Gage, Vol. 6230, SPIE, April 2006.

[101] Robinson, D. W., Pratt, J. E., Paluska, D. J., and Pratt, G. A., "Series Elastic Actuator Development for a Biomimetic Walking Robot," *IEEE/ASME international conference on advanced intelligent mechatronics*, September 1999, pp. 561–568.

[102] Boaventura, T., Semini, C., Buchli, J., Frigerio, M., Focchi, M., and Caldwell, D. G., "Dynamic Torque Control of a Hydraulic Quadruped Robot," *IEEE Conference on Robotics and Automation*, 2012, pp. 1189–1894.

[103] Kim, S., "Biomimetics Robotics Lab," `http://sangbae.scripts.mit.edu/biomimetics/videos/`, July 2012.

[104] Plestan, F., Grizzle, J. W., Westervelt, E. R., and Abba, G., "Stable walking of a 7-DOF biped robot," Vol. 19, No. 4, August 2003, pp. 653–668.

[105] Chevallereau, C., Grizzle, J., and Shih, C., "Asymptotically Stable Walking of a Five-Link Underactuated 3D Bipedal Robot," *IEEE Transactions on Robotics*, Vol. 25, No. 1, 2009, pp. 37–50.

[106] Tuttle, T. and Seering, W., "A nonlinear model of a harmonic drive gear transmission," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 3, 1996, pp. 368–374.

[107] Kennedy, C. and Desai, J., "Modeling and control of the Mitsubishi PA-10 robot arm harmonic drive system," *IEEE/ASME Transactions on Mechatronics*, Vol. 10, No. 3, 2005, pp. 263–274.

[108] Hurmuzlu, Y. and Chang, T., "Rigid body collisions of a special class of planar kinematic chains," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 22, No. 5, 1992, pp. 964–971.

[109] Westervelt, E. R., Grizzle, J. W., Chevallereau, C., Choi, J., and Morris, B., *Feedback Control of Dynamic Bipedal Robot Locomotion*, Control and Automation, CRC Press, Boca Raton, FL, June 2007.

[110] Grizzle, J. W., Abba, G., and Plestan, F., "Asymptotically Stable Walking for Biped Robots: Analysis via Systems with Impulse Effects," *IEEE Transactions on Automatic Control*, Vol. 46, January 2001, pp. 51–64.

[111] Canudas-de-Wit, C., "On the concept of virtual constraints as a tool for walking robot control and balancing," *Annual Reviews in Control*, Vol. 28, 2004, pp. 157–166.

[112] Isidori, A., *Nonlinear Control Systems: An Introduction*, Springer-Verlag, Berlin, Germany, 3rd ed., 1995.

[113] Sreenath, K., Park, H., Poulakakis, I., and Grizzle, J., "A Compliant Hybrid Zero Dynamics Controller for Stable, Efficient and Fast Bipedal Walking on MABEL," *International Journal of Robotics Research*, Vol. 30, 2011, pp. 1170–1193.

[114] Morris, B. and Grizzle, J. W., "Hybrid Invariant Manifolds in Systems with Impulse Effects with Application to Periodic Locomotion in Bipedal Robots," *IEEE Transactions on Automatic Control*, Vol. 54, No. 8, August 2009, pp. 1751–1764.

[115] Piegl, L. and Tiller, W., *The NURBS book*, Springer, 1997.

[116] Rogers, D. F., *An Introduction to NURBS with Historical Perspective*.

[117] Bock, P., Kearney, R., Forster, M., and Wagner, R., "Modulation of Stretch Reflex Excitability during Quiet Human Standing," *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, September 2004, pp. 1–5.

[118] Collins, J. and Luca, C. D., "Random Walking during Quiet Standing," *The American Physical Society*, Vol. 73, August 1994, pp. 764–767.

[119] Sasagawaa, S., Ushiyamab, J., Kouzakic, M., and Kanehisaa, H., "Effect of the hip motion on the body kinematics in the sagittal plane during human quiet standing," *Neuroscience Letters*, Vol. 450, January 2009, pp. 27–31.

[120] Blzqueza, M. T., de Saavedraa, F., Lallenaa, A., and Carpenab, P., "Study of the human postural control system during quiet standing using detrended fluctuation analysis," *Physica A: Statistical Mechanics and its Applications*, Vol. 388, May 2009, pp. 1857–1866.

[121] Loram, I. and Lakie, M., "Direct measurement of human ankle stiffness during quiet standing: the intrinsic mechanical stiffness is insufficient for stability," *The Journal of Physiology*, Vol. 15, December 2002, pp. 10411053.

[122] Asai, Y., Tasaka, Y., Nomura, K., Nomura, T., Casadio, M., and Morasso, P., " A Model of Postural Control in Quiet Standing: Robust Compensation of Delay-Induced Instability Using Intermittent Activation of Feedback Control," July 2009.

[123] Tozeren, A., *Human Body Dynamics: Classical Mechanics and Human Movement*, Springer, 2000.

[124] Shih, C., Grizzle, J., and Chevallereau, C., "Asymptotically Stable Walking of a Simple Underactuated 3D Bipedal Robot," *The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, Taipei, Taiwan, Novembre 2007, pp. 2766–2771.

[125] Grizzle, J., Chevallereau, C., and Shih, C., "HZD-Based Control of a Five-Link Underactuated 3D Bipedal Robot," *IEEE Conf. on Decision and Control*, Cancun, Mexico, December 2008.

[126] Shih, C.-L., Grizzle, J., and Chevallereau, C., "From Stable Walking to Steering of a 3D Bipedal Robot with Passive Point Feet," *Robotica–To appear*, 2012.

[127] Westervelt, E., "Eric Westervelt's publications," 2005, `www.mecheng.osu.edu/~westerve/publications/`.