# Push Recovery Control for Force-Controlled Humanoid Robots

Benjamin Stephens

CMU-RI-TR-11-15

*Submitted in partial fulfillment of the*

*requirements for the degree of*

*Doctor of Philosophy in Robotics*

The Robotics Institute

Carnegie Mellon University

Pittsburgh, Pennsylvania USA

August 2011

Thesis Committee:

Christopher Atkeson (Chair)

Jessica Hodgins

Hartmut Geyer

Jerry Pratt (IHMC)

# Contents

# Abstract

Humanoid robots represent the state of the art in complex robot systems. High performance controllers that can handle unknown perturbations will be required if complex robots are to one day interact safely with people in everyday environments. Analyzing and predicting full-body behaviors is difficult in humanoid robots because of the high number of degrees of freedom and unstable nature of the dynamics. This thesis demonstrates the use of simple models to approximate the dynamics and simplify the design of reactive balance controllers. These simple models define distinct balance recovery strategies and improve state estimation. Push Recovery Model Predictive Control (PR-MPC), an optimization-based reactive balance controller that considers future actions and constraints using a simple COM model, is presented. This controller outputs feasible controls which are realized by Dynamic Balance Force Control (DBFC), a force controller that produces full-body joint torques. Push recovery, walking, and other force-based tasks are presented both in simulation and in experiments on the Sarcos Primus hydraulic humanoid robot.

# Acknowledgements

Chris Atkeson, my Ph.D. advisor, has been an inspiration and motivator for the majority of my thesis work. Whenever I complained about how difficult something was, he would call me out and make it sound simple. Not only was he able to teach me the gritty art of real robot control, but he showed me how to fit that work into a grander context of my thesis and the efforts of the greater robotics community.

Kevin Lynch, my undergraduate research mentor, got me started in robotics as a freshman. He hired me as a work study research assistant to work on several projects and gave me freedom to begin to pursue my own research interests. Kevin was actually one of the first Robograds and I am proud to follow in his footsteps.

Stuart Anderson, my Sarcos robot lab mate, provided a lot of interesting discussion and inspiration for my controls work. He is one of the most talented and brilliant roboticists I know. He also likes bikes which is cool.

My committee members, Jessica Hodgins, Hartmut Geyer and Jerry Pratt, each with their own exceptional accomplishments and expertise, set a high bar for me to achieve during my thesis work.

My cycling friends from Pittsburgh kept me healthy and sane throughout my five years at CMU. A Ph.D. is hard work and I needed to take a healthy break every now and then. Cycling became my hobby of choice the moment I joined the CMU cycling club and I have put a lot of passion into both aspects of my life. Thanks for those fast weekday excursions and those long weekend adventures.

My family doesn't really understand what I do but has always supported me along the way. I have been moving further and further from home, starting from moving to OKC for

high school, Chicago for undergrad, Pittsburgh for my Ph.D. and now on to Boston. I know they understand my desire to follow my passions and live my own life. I am always happy when they come to visit and look forward to returning home each chance I get.

# Chapter 1

# Introduction

In this thesis, simple models of biped dynamics are used to model force-based balance and directly applied to the control, planning and estimation of complex force-controlled humanoid robots performing push recovery and other tasks.

## 1.1 Summary

This thesis is concerned with the problem of **controlling push recovery for humanoid robots with full-body force-control** and is focused on overcoming two fundamental difficulties associated with these systems. First, the small base of support limits the forces and torques that can be generated to recover balance. For large pushes, a change of support can be achieved by stepping but cannot be performed instantaneously. Therefore, the controller should **reason about future actions and the effects of constraints** on the available contact forces. Second, the low impedance of the force-controlled joints allows the robot to achieve greater compliance during interaction with other objects and the environment, but results in less stable balance. The controller should also **continuously estimate the state**

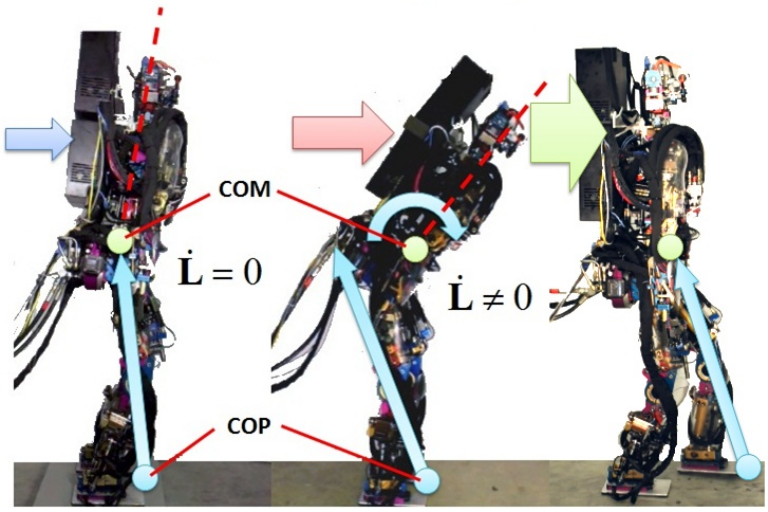


**Figure 1.1:** The three basic balancing strategies: Ankle strategy, hip strategy and stepping. The reaction force acts at the COP and applies a force and torque on the COM.

**of the system and correct its balance**.

## 1.2 Problem Definitions

**Humanoid robots** are robot systems that resemble humans. They are more anthropomorphic than typical robots such as industrial robotic arms. Often, but not always, humanoid robots locomote on two legs. Likewise, other features may include arms, hands, a head, and eyes. This thesis mainly addresses problems related to the coordination of the legs for balance and locomtion.

**Push Recovery** refers to maintaining balance when subjected to a large disturbance. In this thesis, we are mostly concerned with large, short duration impulses. Such pushes can quickly destabilize the system but do not alter the dynamics of the system. Recovery

is achieved through selection and execution of an appropriate strategy, as shown in Figure 1.1. Push recovery is an interesting topic because it explores the performance limits of the system. By achieving stable push recovery for large pushes, it is also likely that the stability of the system during normal, unperturbed operation will be improved.

**Control** refers to the process of executing actions based on the state of the system. The state can be made up of configuration variables that describe the physical condition of the robot or by superficial variables generated by a clock or other user-defined process. Actions include, but are not limited to, forces and torques that induce or resist motion. Control of humanoid robots is difficult because it involves the coordination of a large number of state variables and actions.

## 1.3 Types of Humanoid Balance Control

There have been many approaches to the problem of humanoid balance control. Two of the most common approaches to humanoid robot design, each resulting in a unique style of control, can generally be described as position-controlled robots and force-controlled robots, as shown in Figure 1.2. Advantages and disadvantages of these two approaches are described below.

**Position-controlled robots** are generally constructed using electric motors with high gear ratios. This allows for smaller, cheaper motors to operate efficiently while providing enough torque. The high reflected inertia and friction created by the gearbox creates a high impedance. High impedance can be useful for rejecting small disturbances. Controllers for these systems, such as the one depicted in Figure 1.3, generally exploit this feature by creating walking patterns that can be tracked very accurately. This effectively simplifies walking

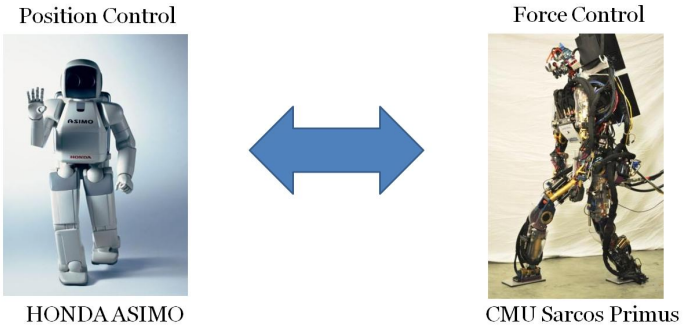


**Figure 1.2:** Position control and force control are two common approaches to humanoid robot design and control.

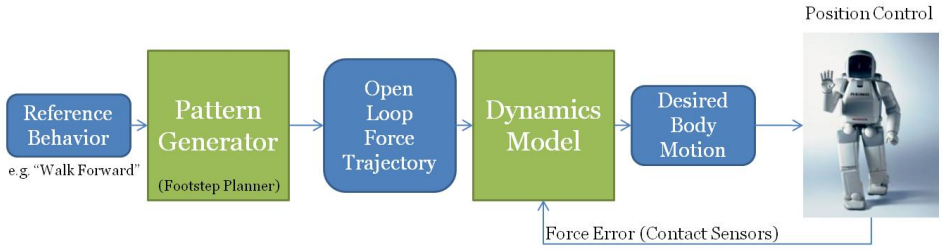


**Figure 1.3:** Position control.

control to footstep planning, for example. The high impedance also results in poor force control, which is useful for handling larger disturbances. To improve stability, compliance control is added to the feet using force-torque sensors. Because of the high joint impedance, low control bandwidth, and non-colocated sensors, the performance of this type of force control is limited.

**Force-controlled robots** are usually constructed with more direct-drive actuation to achieve a much lower impedance throughout the body and generally employ a reactive balance controller, as depicted in Figure 1.4. Full-body force control has several major advantages. It allows the controller to exploit the passive, or natural, dynamics of the system for

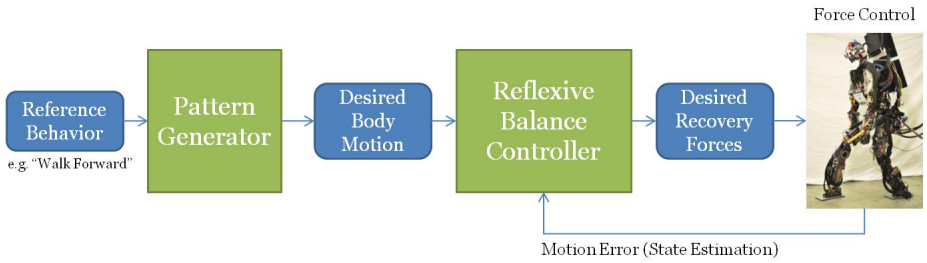


**Figure 1.4:** Force control is characterized by robots with force feedback throughout the body and a reactive closed-loop balance controller.

greater efficiency. The ability to distribute forces throughout the body can help reduce internal forces in closed kinematic loops. Most importantly, it allows for compliance to geometric and force disturbances allowing the system to handle unknown ground geometry or interact with dynamic objects such as a person. The disadvantage of force control, however, is that the controller must add the appropriate impedance as well as compensate for any dynamic coupling forces to successfully perform a task. Designing such a controller can be difficult given the complex dynamics of the robot.

One of the objectives of this thesis is to combine the advantages of these two approaches. Through extensive use of simplified models for planning, control, and estimation, some of the complexity of the robot dynamics can be masked. Likewise, reactive balance control can be achieved while maintaining compliance and performing useful tasks.

## 1.4 Motivation

There are few mechanical systems more complex than humanoid robots. The high number of degrees of freedom requires not only ingenious mechanical design, but control algorithms that scale. This thesis aims to **advance the state of the art in force-based control**

**for complex systems** by developing such algorithms. For a humanoid robot to operate in a complex, unknown environment and with people, the most important feature of the controller is that it be able to maintain its balance even when affected by unknown disturbances. Another motivation for this work is to **enable humanoid robots to perform general tasks**. Humanoid robots will be operating in a complex outdoor or cluttered indoor environments, requiring a high degree of reconfigurability and adaptability to react to rapidly changing contact states, obstacles and uncertainty.

This thesis also focuses on theory and methods of control that **simplify the design of humanoid robot behaviors**. More thorough analysis and controls can be developed for simplified models of the system and then mapped to the full system.

Finally, much of the work in this thesis was performed as part of the Quality of Life Technology Engineering Research Center, a National Science Foundation project aimed at transforming the lives of the older adults and people with disabilities through the development of innovative assistive and rehabilitative technology. The research in this thesis fell under the Mobility and Manipulation Research Thrust, which focused on developing controllers for **safe interaction and cooperation with humans**. In this regard, controllers in this thesis were designed to be compliant to unknown forces but able to recover without falling over.

## 1.5 Thesis Contributions

This thesis builds on the work of many previous researchers in the robotics field. As stated above, the goal of this work is to take dynamic behavior in humanoid robots to the next level. Below are specific contributions that will be described in this thesis:

**Push recovery strategy stability regions determined analytically using simple models** (See Section 3.3 and [112]): Using simple models of the robot dynamics, the stability of certain push recovery strategies can be described analytically. These analytic relationships can be used to determine which strategies to use based on the current state of the robot.

**Push Recovery Model Predictive Control** (See Section 4.3 and [116]): Model predictive control can be performed using simple models, a special objective function, and carefully chosen constraints to compute desired forces and footstep locations that account for future actions and constraints. This controller is run continuously online to reactively maintain balance, recover from pushes, and perform dynamic walking.

**Dynamic Balance Force Control** (See Section 5.4 and [115]): Full-body joint torques can be computed using simple models of the allowable contact and task forces and used to perform a variety of force-based tasks.

**State estimation with modeling error** (See Chapter 6 and [114]): Constructing Kalman filters by augmenting simple models with different types of modeling error results in improved COM state estimation, which is important for control.

**Implementations on a hydraulic humanoid robot:** The hydraulic humanoid robot is operated in force-control mode by first determining desired joint torques based on a rigid-body dynamics model and then converting to a hydraulic valve command using an independent joint-level force-feedback controller (See Section 7.4). Implementations presented in this thesis include standing balance, lifting heavy objects, push recovery using ankle and hip strategies and stepping, dynamic walking, and dancing using human motion capture.

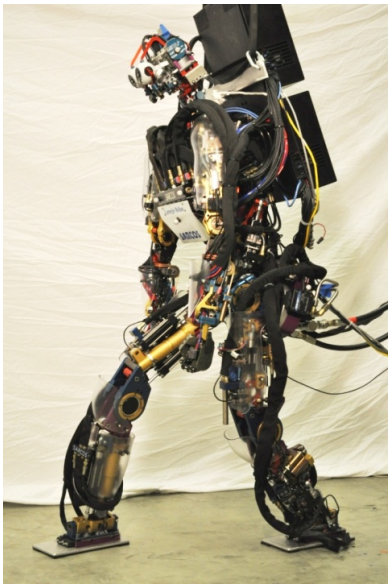

**Figure 1.5:** The Sarcos Primus humanoid robot is a hydraulically actuated experimental platform for control located at Carnegie Mellon University.

## 1.6 Hardware Platform

The Sarcos Primus humanoid robot, shown in Figure 1.5, is the main experimental platform for the research in this thesis. It was designed for the purpose of studying how the brain works when interacting with the real world [14]. The robot uses hydraulic actuators to control 33 degrees of freedom. It has potentiometers and load sensors at every joint to measure joint angle and force in the piston or linkage. A fast local force feedback control loop implements force control at each joint. For most of the balancing and walking experiments in this thesis, only the lower body degrees of freedom are controlled using force control while the rest are fixed using stiff PD position gains. More details on the robot and its controls are described throughout this thesis.

As stated in the previous section, one of the goals of this thesis is to improve the state

of the art in control of humanoid robots. To this end, all of the material presented in this thesis is **implemented on real hardware**. Simulation is often used as a starting point using an idealized model of the robot and the environment. Simulation alone is a complex problem, but often insufficiently approximates the real-world system, which is affected by other complex phenomena such as sensor noise, unmodeled dynamics, latency, calibrations, and perhaps other unknown sources. Therefore, one of the contributions of this thesis is what amounts to a collection of practical techniques for controlling force-controlled humanoid robots.

## 1.7 Outline

This thesis is outlined as follows. Following this introduction, a review of related work will be presented in Chapter 2. Next, Chapter 3 will include an overview of simplified models of biped balance. These models will be analyzed to predict bounds on stability that can be used to decide between different balance strategies, such as whether or not to take a step. In Chapter 4, it will be shown how model predictive control can use the simple model dynamics to control push recovery. The concept of Push Recovery Model Predictive Control (PR-MPC) will be presented, along with several extensions. In Chapter 5, algorithms for mapping controls for the simplified models to a humanoid robot will be shown. Dynamic Balance Force Control (DBFC) is used for a wide variety of tasks involving force control and balance. In Chapter 6, the problem of state estimation in force-controlled robot balance is considered using simplified models. Different types of modeling error are modeled and utilized for improved estimation. Finally, details related to real-world implementation of control on the Sarcos Primus hydraulic humanoid robot are presented in Chapter 7.

# Chapter 2

# Background

The problem of push recovery for a humanoid, be it a person or robot, requires a comprehensive understanding of kinematics, dynamics, and controls. In this chapter, an overview of related work on push recovery modeling and control in both robots and humans is presented. The Summary section points out specific works that play a major role in this thesis.

## 2.1 Models for Balance

Fundamental to push recovery are the various strategies used to recover. Hofmann [47] summarized three of these basic strategies, arguing that the key to balancing is controlling the horizontal motion of the COM. For small disturbances, simply shifting the center of pressure (COP) changes the tangential ground reaction force (GRF), which directly affects the motion of the COM. Because the location of the COP is limited to be under the feet, a second strategy is to create a moment about the COM, creating a momentarily larger tangential GRF. This leads to a new point, an effective COP, called the centroidal moment pivot (CMP). The third strategy used for large disturbances takes a step in order to increase

or move the feasible area for the COP (and the CMP).

Like Hofmann, many researchers have used simplified models to represent the complex dynamics of a humanoid. Some researchers focus on simple models that correspond to a lumped mass model at the center of mass [56][55][101]. The most notable of these models is the Linear Inverted Pendulum Model (LIPM), which is a linear model due to the assumption that the height of the COM is constant and there is zero angular momentum. Sometimes these lumped mass models are modified to include a rotational inertia term that models angular momentum, which can add significant capability to the system [38][62][97][112][71]. Instead of considering the control limits, Sugihara determined stability bounds given a fixed balance controller [120].

Inverted pendulums, particularly double inverted pendulums, have long been used as a model for balance [43]. These models are rigidly attached to the ground and every joint is actuated, making the dynamics simple. One exception is the Acrobot [111], which balances using an unactuated base (or ankle) joint. Many control techniques have been used for controlling inverted pendulum balance including linear feedback [35], integral control [63][113] and optimization [8]. Inverted pendulum models have also been used to optimize falling motions to minimize impact [27]. Simple models such as the Spring-Loaded Inverted Pendulum (SLIP) model have been used to model dynamic walking and running in humans[28][31].

Simple models are often used to study balance strategies [112]. Many researchers have used these model to define full-body balance strategies and stability measures using reduced coordinates such as the zero-moment point (ZMP) [128] and other ground reference points [37][95]. Pratt, *et al.* [97] used the LIPM to suggest the "capture point," a point about which the system is theoretically open-loop stable. This idea had been previously considered by Townsend [126]. That is, by placing a foot at this location a idealized system will come to

rest without any feedback control. They also formulated the "capture region," a collection of capture points where, by simple feedback, the system should be able to recover in one step. Sometimes the capture region is outside the range of motion and the robot takes more than one step in order to recover. This point is more useful than the instantaneous COP/ZMP as it relates to the evolution of the dynamics over time. This idea was extended by Wight, *et al.* [133] using inverted pendulum dynamics to predict the location of the capture point using the "Foot Placement Estimator."

## 2.2 Controlling Balance

Balance involves the manipulation of contact forces to produce a desired motion. Raibert suggested the use of a virtual leg which corresponded to the total reaction force and its points of contact [104]. This idea was developed on one-legged hoppers and extended to bipedal and quadrupedal robots [103]. The virtual linkage [135] is another simple model that represents multiple contacts, which has recently been used to control multiple contacts on humanoid robots [109]. The center of pressure in a 3D multi-contact context has also been considered [40]. Virtual model control [102] has been used for controlling bipedal walking [100]. Multi-contact force control has also been applied to multiple-leg walking systems to minimize horizontal forces [130] and obey friction constraints [61]. Force control of legged locomotion is closely related to force control in grasping [67].

Full-body inverse kinematics is often used either to directly control balance to complement a dynamic balance controller [121]. A floating-body model is useful for providing general solutions [83]. New algorithms have been developed to handle prioritization and hard constraints such as joint limits [60].

Force control of full-body humanoid balance has been considered more recently. Again, floating body inverse dynamics [82] is often used for generality. The inverse dynamics problem generally has many solutions, so a number of approaches have been applied. Passivity-based gravity compensation [50] is one of the simplest which assumes a quasi-static system and a minimum norm distribution of forces under the feet. Reformulation of the robot dynamics into a multi-objective optimization allows for solutions that obey constraints related to ground contact and joint limits [65][2]. Prioritized control [108] is often used to write controllers in operational or feature space, and can also be solved using a multi-objective optimization [20].

Controlling push recovery by stepping has been achieved using various techniques. Reinforcement learning has been used to learn corrections to the capture points calculated from simple models [105]. Other researchers have used human motion capture to determine models for balance and stepping [66]. Planning that considers the future has shown that sometimes the robot's task, reaching for a heavy object for example, may require a step to avoid falling [142]. Reflexes to deal with slipping and tripping have also been considered for hopping robots [12]. In addition, in extreme circumstances, taking a step can be used to redirect a fall from colliding with an obstacle or to help decrease the impact [143]. Similarly, foot placement can be corrected based on known safe footstep locations [16].

## 2.3 Locomotion

Biped locomotion has also been studied for many years [129][42]. Simple models have been widely used for control and estimation of humanoid robot locomotion. The Linear Inverted Pendulum Model (LIPM) [56] has been used in combination with a model-predictive con-

troller called "preview control" [53]. It has been shown that the preview control problem can be solved efficiently with quadratic programming [132] and even modified to include footstep placement [21]. Footstep planning for navigation [15] can be achieved by ignoring much of the robot dynamics and balance control, considering only a fixed set of foot placements at each step and searching over a fixed horizon. This work was extended to navigation among moveable objects [117].

Position control of humanoid robot walking [122][141][87] has been widely used and has been successfully applied to many robots [45][51][59][4]. The mechanical design of these robots is influenced by the desire to perform stiff and accurate trajectory control, often requiring harmonic drives with large gear ratios on electric motors. There have been extensions which include dynamic effects such as angular momentum [64][54].

Dynamic locomotion, using torque-based control, has also been widely studied using methods such as virtual model control [98], impedance control [92], and dynamics filtering [138]. For an overview of the dynamic locomotion problem, see [49]. In addition, the animation community has used momentum-based techniques [3] to optimize motions for balance and locomotion based on motion capture data [140].

In contrast, passive dynamics is an approach to analyzing locomotion that focuses on very little actuation [80][17]. For these simple systems, stability analysis is often expressed using a measure of gait sensitivity [46]. More recently, there have been efforts to embed passive dynamic systems into a nonlinear full-body controller [78].

## 2.4 Biomechanics

Biomechanics researchers have similarly used inverted pendulum models to explain balance [136]. The "hip strategy" and "ankle strategy," described by Horak and Nashner [48], have long been dominant descriptions of balance control in humans. Makai *et al.* studied the interaction of these "fixed-base" strategies with "change-of-support" strategies in human test subjects [77]. They argue that these strategies occur in parallel, rather than sequentially, and that humans will take a step before they reach the stability boundary for standing in place. An explanation for their findings is they only consider the position of the COM as the defining measure of stability. It was soon observed that the use of COM velocity in addition to position is a better measure of stability [90][81]. Pai, *et al.* used their simple dynamic model to predict the need for stepping in human experiments [89]. Optimal control, specifically LQR control, has also been used to explain human balance [41][68].

Human locomotion is often studied using simple models. Many studies are based on the concept of passive dynamic walking [80][17][137]. The simplest walking model [29] was used to study the energetics and preferred relationship between speed and step length [70]. Others have developed predictions of footstep location during walking using a simple model of the position and velocity of the pelvis [126][106]. Similarly, control of lateral balance has also been studied [69] and verified in human experiments [9]. Angular momentum has been shown to be highly regulated during human locomotion [94].

## 2.5 Animation

Animation shares a lot in common with robotics. Both fields attempt to reproduce natural and functional motions. In recent years, producing physically realistic animations has caused computer graphics researchers to consider not just kinematic correctness but also consider the role of contact forces, momentum and joint torques in simulation. Modern simulators are often implemented by solving an optimization at each timestep to compute physically realistic and meaningful controls and motions [2]. These systems often take advantage of simplified representations of momentum to efficiently generate realistic motions [25][3].

In animation, it is often desirable to take several example motions and extrapolate to new motions. Often these examples are taken from human motion capture [140][110][19][127]. Safanova [107] use databases of human motion capture combined with PCA decomposition to plan and optimize motions in low dimensional spaces. Grochow, et al. [39] generated "Style-Based Inverse Kinematics" based on a model of human poses learned from a motion capture database. Another technique involves interpolation of segmented trajectories [134]. Also important is the topic of re-targetting [32], or applying motions designed for one character to another with different kinematic or dynamic properties.

## 2.6 Force Control

Force control is desirable for balancing robots for several reasons. Pratt, *et al.* [99] suggested three major benefits of full-body force control. First, the low impedance allows the controller to exploit the natural dynamics of the system, such as a passive swing leg. By doing so, it may be possible to significantly reduce the control effort and simplify the controller. Second,

full-body force control allows for the distribution of forces throughout the body. For a system with closed kinematic loops, this means reducing the chance of slip and minimizing wasteful internal joint forces. Finally, the compliance offered by force control allows the system to comply with geometric uncertainty in the environment. Balancing on uneven terrain is achieved by controlling the net force on the body regardless of the specific pose of the robot.

The implementation of force control has been studied for many years. It is a difficult problem due to inherent instabilities in the control system and robot hardware. Eppinger, *et al.* , [24] explored a series of simplified models of a robot joint. Hardware design decision often dictate the choice of control. For robots with high-gain position feedback controllers, force sensing is usually added to the end effector where force is to be controlled. In general, the non-colocation of the actuator and force sensor, high reflected inertia, and friction all destabilize the system. The force feedback gain is usually reduced to ensure stability resulting in poor force control performance. Series elastic actuators [96] are an actuator design that was proposed to reduce these problems.

## 2.7 Trajectory Optimization

Trajectory optimization is a large field that takes a model of the system and computes a time-sequenced trajectory of states and/or controls according to some optimality criterion. There are several types of trajectory optimization methods. There are two important distinctions: global vs. local and simultaneous vs. sequential. Global optimization finds a consistent optimal solution throughout the entire state space but has a high computational cost. Local optimization finds only local minima but is generally cheaper to compute. Simultaneous trajectory optimization updates the entire trajectory all at once, often providing a better

handling of constraints. Sequential trajectory optimization usually involves a forward and backward pass that updates trajectory one timestep at a time producing more physically consistent trajectories but can be more numerically sensitive to the initial conditions, model and integration method. Model predictive control, or receding horizon control, is a method of control that performs these trajectory optimizations online.

Dynamic programming [10] is a global method that can be used to solve simple systems but runs into computational and memory limits for complex systems because it requires value or policy iteration. DP has been used to solve small sub-components of the dynamics for humanoid walking [131].

Another technique is Differential Dynamic Programming [52][123] which is local and sequential. DDP features much lower computational requirements and can output more physically realistic trajectories. Similar techniques such as iLQR [72], iLQG [124] iLDP [125] have been proposed.

Model predictive control in industry often use a simultaneous trajectory optimization technique [86][79]. The computational requirements are slightly higher but by solving simultaneously the algorithm usually handles constraints better.

Some have tried to exploit the lower computational and memory requirements of local trajectory optimization to create a library of trajectories capable of providing a more global solution. This involves both the creation of policies based on trajectory libraries [118] and generalization to contexts outside of the library, called policy transfer [119]. Trajectory library-based control has been applied to standing balance [73] and biped walking [75][74].

Another trajectory-based optimization deals with repeatedly improving a trajectory based on experiments on real hardware. Some approaches are Iterative Learning Control [6], Trajectory Learning [7][5] and Feedback Error Learning [36]. These approaches use a feedback

signal from a controller to update the feedforward trajectory created by the trajectory optimizer.

## 2.8 Summary

This thesis makes use of simple models, particularly the linear inverted pendulum by Kajita, *et al.* , [56] and the linear inverted pendulum plus flywheel model by Pratt, *et al.* , [97]. These models are used to describe strategies, like the ones summarized by Hofmann [47], and predict the stability using analytic bounds on COM position and velocity.

Once an appropriate strategy is chosen, a reactive controller can be designed using these same simple models. Model predictive control (MPC) is often applied for planning humanoid locomotion. This *Preview Control* approach was popularized by Kajita, *et al.* , [53], but required pre-planned footstep locations. The more recent work of Wieber, *et al.* , [132] and Herdt, *et al.* , [44] made footsteps variable to generate both COM trajectories and footstep locations simultaneously while obeying the ZMP constraints. In stiff position-controlled robots, this algorithm is used for walking pattern generation to achieve high level inputs such as desired footstep locations or walking velocities. The same basic algorithm is used in this thesis, but it is applied as low-level reactive balance control on a force-controlled robot by updating desired ZMP trajectories based on the current state of the robot. This idea is similar to more recent works in animation where locomotion is driven by a low-dimensional model such as Tsai, *et al.* , [127] and Coros, *et al.* , [18], which also use a linear inverted pendulum model, and Mordatch, *et al.* , [84] which uses an approximate spring-loaded inverted pendulum model.

Generating full-body torques is also accomplished through the use of simple models. This

is unlike methods that perform inverse dynamics, such as Mistry, *et al.* , [82], which require desired joint acceleration inputs. Instead, the commonality between simple model features, specifically COM acceleration, and the full robot can be used to generate full-body controls. The approach in this thesis is inspired by the work of Hyon, *et al.* , [50], which achieves reactive balance control by specifying desired forces on the COM. Virtual model controls, inspired by the work of Pratt, *et al.* , [98], are combined with this approach to perform generic tasks.

# Chapter 3

# Simple Models

This chapter describes simple dynamic models that can generally describe humanoid balance. The dynamics are analyzed to determine push recovery strategies and predict their stability. Later in this thesis, these simple models will be used for control and estimation purposes as well.

## 3.1 Introduction

We study humanoids as a way to understand humans. We want to understand what causes humanoids to fall, and what can be done to avoid it. Disturbances and modeling error are possible contributors to falling. For small disturbances, simply behaving like an inverted pendulum and applying a compensating torque at the ankle can be enough. As the disturbance increases, however, more of the body has to be used. Bending the hips or swinging the arms creates an additional restoring torque. Finally, if the disturbance is too large, the only way to prevent falling is to take a step.

In this chapter, we unify simple models used previously to explain humanoid balance and

control. In Section 3.1.1, we discuss previous work and in Section 3.1.2 we summarize our models and balance strategies. Section 3.2 describes lumped mass models that approximate the system as a point mass at each instant in time. In Section 3.3, the stability of standing balance is analyzed using these simple models. Finally, in Section 3.4, stepping is modeled as an additional strategy for push recovery.

**The main contributions of this chapter are the unification of models and strategies used for humanoid balance and the development of analytic stability regions that define the stability limits of each strategy**. These stability regions are defined as functions of reference points, such as the center of mass and center of pressure, that can be measured or calculated easily for both robots and humans. Both ankle and internal joint actuation are assumed to be available and used in balance recovery.

### 3.1.1 Related Work

The problem of postural stability in humanoids has been studied for many years. Vukobratovic, *et.al.* was the first to apply the concept of the ZMP, or zero moment point, to biped balance [129]. Feedback linearizing control of a simple double-inverted pendulum model using ankle and hip torques was used by Hemami, *et.al.* [42]. Stepping to avoid falling was also studied by Goddard, *et.al.* [33], using feedback control of computed constraint forces derived from Lagrangian dynamics.

Modern bipedal locomotion research has been heavily influenced by Kajita, *et.al.* and their *Linear Inverted Pendulum Model* (LIPM) [56]. It is linearized and constrained to a horizontal plane, so it is a one-dimensional linear dynamic system representing humanoid motion. When considering ankle torques and the constraints on the location of the ZMP, or

zero moment point, it has also been referred to as the "cart-on-a-table" model. An extension to the LIPM is the AMPM, or *Angular Momentum inducing inverted Pendulum Model* [64], which generates momentum by applying a torque about the center of mass (COM).

There have been numerous attempts to identify and model humanoid balancing and locomotion using simplified reference points [95]. Hofmann [47] studied humanoid control during walking and balancing tasks in his thesis. He argues that the key to balancing is controlling the horizontal motion of the COM, and there are three strategies for accomplishing this. For small disturbances, simply shifting the center of pressure (COP) changes the tangential ground reaction force (GRF), which directly affects the motion of the COM. Because the location of the COP is limited to be under the feet, a second strategy is to create a moment about the COM, creating a momentarily larger tangential GRF. This strategy suggests a new point, an effective COP, called the centroidal moment pivot (CMP) that takes into account the inertia of the upper body. Goswami, *et al.* [38] also considered the strategy of inducing non-zero moment about the COM to aid in balance. They define the *ZRAM point* (Zero Rate of change of Angular Momentum), which is identical to the CMP. The third strategy used for large disturbances takes a step in order to increase or move the feasible area for the COP (and equivalently the CMP). In stepping, the swing leg impact absorbs kinetic energy and the effective unsprung mass and impact velocity determine the magnitude of energy absorbed.

Pratt, *et al.* [101] attempted to formulate the control and stability of biped walking using more meaningful velocity formulations. They argued that traditional approaches, like Poincare Maps and ZMP trajectories, are not suitable for fast dynamic motions. They suggest that angular momenta about the center of pressure and the center of mass are important quantities to regulate. Their velocity-based formulations and "Linear Inverted

Pendulum Plus Flywheel Model" [97] were used to formulate the "capture region," where the robot must step in order to catch itself from a fall. Sometimes the capture region is outside the range of motion and the robot takes more than one step in order to recover. The model they use is simple and allows regulation of angular momentum, but in their formulation they do not consider impact energy losses or the effect of ankle torques.

### 3.1.2 Balance Models and Strategies

Here we explore different models and strategies used for humanoid balance. There are three basic strategies that can each be described by a unique class of simple models:

1. COP Balancing / Lumped Mass Models
2. CMP Balancing / Angular Momentum Models
3. Stepping / Stepping Models

Generally, these strategies can be employed sequentially from top to bottom, advancing to the next if the current strategy is inadequate. As shown in Figure 3.1, the effective horizontal force on the center of mass, equal to the horizontal ground reaction force, can be increased or moved (Stepping) if simple COP Balancing is not enough.

In this chapter, simple models will be used to determine stability regions that describe the stability limits of different push recovery strategies. Real humanoid robots will have many more degrees of freedom and complicated control problems, but by describing their motion in terms of these reduced quantities, such as center of mass and center of pressure, we create useful approximations. Reduced dimensionality also makes it easier to visualize motions, aiding in intuition and understanding.

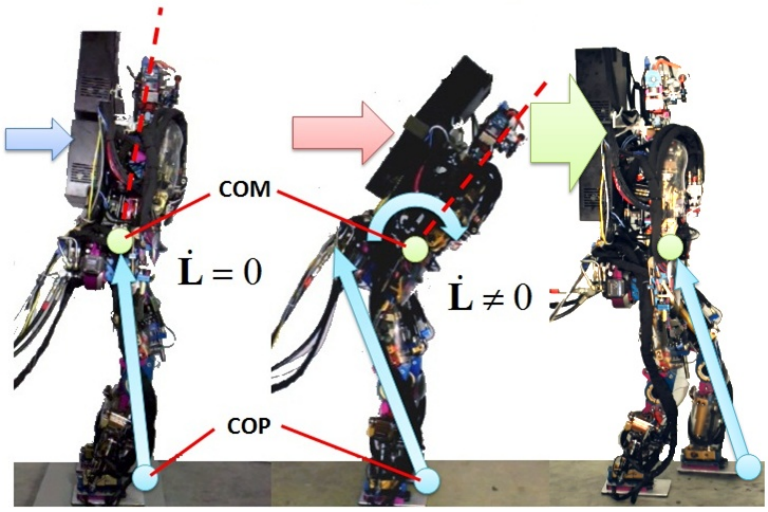


**Figure 3.1:** The three basic balancing strategies. The reaction force acts at the COP and applies a force and torque on the COM. From left to right are COP Balancing ("Ankle Strategy"), CMP Balancing ("Hip Strategy") and Stepping.

## 3.2 Lumped Mass Models

At any instant in time, a system of rigid bodies can be approximated as a lumped mass. At each instant, the sum of forces on the system results in a change of linear momentum, or, equivalently, an acceleration of the center of mass of the system. At the same time, the sum of moments about the COM results in a change of angular momentum of the system. Given a vector of contact forces and torques, $\mathbf{F}$, the relationship with the COM acceleration, $\ddot{\mathbf{C}}$, and change in angular momentum, $\dot{\mathbf{H}}$, is given by

$$\begin{bmatrix} \mathbf{D_1} \\ \mathbf{D_2} \end{bmatrix} \begin{pmatrix} \mathbf{F} \end{pmatrix} = \begin{pmatrix} m\ddot{\mathbf{C}} + \mathbf{F}_g \\ \dot{\mathbf{H}} \end{pmatrix} \tag{3.1}$$

where

$$\mathbf{D_1} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}, \tag{3.2}$$

$$\mathbf{D_2} = \begin{bmatrix} (\mathbf{P}_R - \mathbf{C})\times & (\mathbf{P}_L - \mathbf{C})\times & \mathbf{I}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix}, \tag{3.3}$$

and $\mathbf{P}_R$ and $\mathbf{P}_L$ are the positions of the feet, $\mathbf{r}\times$ represents the left cross product matrix, $m$ is the total mass of the system, and $\mathbf{F}_g$ is the gravitational vector force. For the case of a biped, $\mathbf{F}$ is partitioned to the right and left feet, respectively,

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}_R \\ \mathbf{F}_L \\ \mathbf{M}_R \\ \mathbf{M}_L \end{pmatrix}. \tag{3.4}$$

The first three equations of Eq. (3.1) sum the forces on the center of mass due to gravity and the ground contact points. The last three equations sum the torques about the center of mass to give the resulting change in angular momentum. Note that these equations can be easily extended to more than two contacts.

### 3.2.1 Linear Inverted Pendulum

For a lumped mass model, if it is assumed that there is no angular momentum, $\mathbf{H} = \mathbf{0}$, and no change in angular momentum in the system, $\dot{\mathbf{H}} = \mathbf{0}$, any forces that satisfy Eq. (3.1) do not generate angular momentum changes about the center of mass. If it is additionally assumed that the COM is at a constant height, $z = z_0$ & $\ddot{z} = 0$, the dynamics are identical to the well-known Linear Inverted Pendulum Model (LIPM) [56], illustrated in Figure 3.2. In this thesis, the planar LIPM dynamics are written as

$$m\ddot{x} = \frac{mg}{z_0}(x - x_c) \tag{3.5}$$

$$\dot{x}_c = u_x \tag{3.6}$$

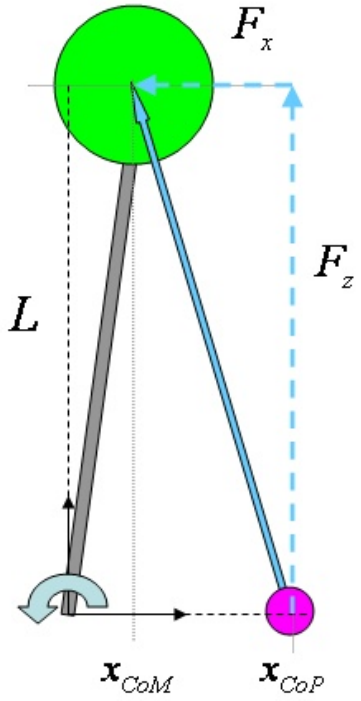


**Figure 3.2:** In the Linear Inverted Pendulum Model, the ground reaction force points from the center of pressure (COP) to the center of mass (COM), creating zero moment about the COM.

where $x$ is the horizontal position of the COM in the sagittal plane at a constant height, $z_0$, $x_c$ is the center of pressure (COP), and $u_x$ is the control signal for the center of pressure. The dynamics in the lateral direction are the same and decoupled from the sagittal direction.

**Why make COP a state?** There are several reasons why one might want to model the COP as a state and the derivative of the COP as the control input. First of all, the rate of change of the COP can be limited to give smooth COP trajectories. In using this model for optimal control, such as in **Chapter 4**, it is simple to assign a cost to this derivative. Another reason for making the COP a state is that it is often a measured quantity, and is treated as such in **Chapter 6**. Finally, the hydraulic robot described in Section 1.6 is

actually a high-order dynamic system and cannot instantaneously change the COP because it takes time to accumulate force in the actuators.

The LIPM allows us to explore a simple balancing strategy which uses ankle torque to apply a restoring force, while other joints are fixed. This strategy is often referred to as the "ankle strategy." The location of the COP is approximately proportional to ankle torque, and therefore the limits on the position of the COP correspond to torque limits at the ankle.

Whether the robot will fall or not can be predicted by the state-space location, $(x, \dot{x})$, of the center of mass. We want to find the limits on the state of the center of mass to avoid falling while obeying the constraints of the COP. If no angular momentum is permitted, then the ground reaction force points from the COP to the COM, as shown in Figure 3.2. Then the acceleration of the COM is given by Eq. (3.5),

$$\ddot{x} = \frac{g}{z_0}(x - x_c) \tag{3.7}$$

where $x$ and $x_c$ are the locations of the COM and COP, respectively. If we assume the vertical motion of the COM is negligible, then $F_z = mg$.

### 3.2.2 Linear Inverted Pendulum with Constant Force

The sagittal LIPM dynamics of Eq. (3.5) can be modified slightly to include an external force, $f_x$, and written as

$$m\ddot{x} = \frac{mg}{z_0}(x - x_c) + f_x \tag{3.8}$$

$$\dot{x}_c = u_x \tag{3.9}$$

Again, the dynamics in the lateral plane are identical and decoupled.

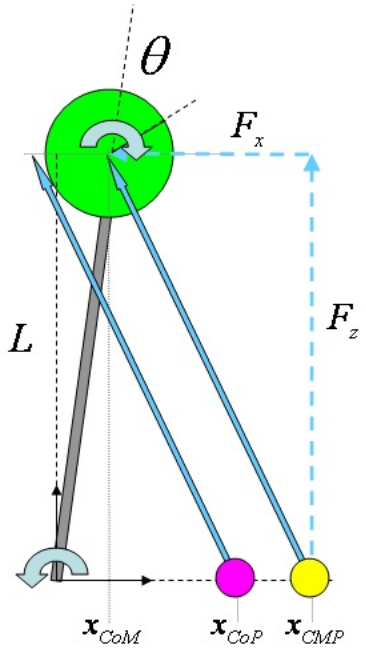


**Figure 3.3:** Internal joint torques create a torque about the COM. This torque is reflected by the ground reaction force, which acts at the COP but does not necessarily point through the COM. The equivalent force that points through the COM begins from the CMP.

**How is this used?** This model is useful for understanding the effects of long-duration push forces or, as will be described in **Chapter 6**, small modeling error.

### 3.2.3 Linear Inverted Pendulum with Flywheel

The dynamics of the upper body, particularly the torso and arms, can play an important role in push recovery. These joints can be used to apply a torque about the COM. The CMP, or centroidal moment pivot, is equal to the COP in the case of zero moment about the center of mass. For the LIPM, this is always the case. For a non-zero moment about the COM, however, the CMP can move beyond the edge of the foot while the COP still remains

under the foot. The result is a greater horizontal recovery force on the COM. This effect can be studied by approximating the upper body as a flywheel that can be torqued directly, as shown in Figure 3.3. This model is an extension of the model used by Pratt *et al.* [97] to allow for an ankle torque.

The dynamics of this system can be written as

$$m\ddot{x} = \frac{mg}{z_0}(x - x_c) - \frac{\tau_y}{z_0} \tag{3.10}$$

$$I_y\ddot{\theta}_y = \tau_y \tag{3.11}$$

$$\dot{x}_c = u_x \tag{3.12}$$

where $\tau_y$ is the torque about the COM in the sagittal plane, $\theta_y$ is the angle of the flywheel and $I_y$ is the inertia of the flywheel. The system can be modeled with decoupled dynamics in the two horizontal directions, which is equivalent to two flywheels.

**How is this used?** This model is used in Section 3.3.2 to determine stability bounds for the hip strategy, and in Section 5.6.2 to perform posture control.

## 3.3 Stability Regions

### 3.3.1 COP Balancing

In this section, we derive the stability region of the Linear Inverted Pendulum Model with the ankle torque saturated such that the COP is at the edge of the foot at $x_{\text{COP}} = \delta^{\pm}$, where $\delta^-$ and $\delta^+$ are the back and front edges of support region, respectively. In this case, the

maximum acceleration is given by Eq. (3.5),

$$\ddot{x}_{\text{COM}}^{\max} = \frac{g}{z_0}(x_{\text{COM}} - \delta^{\pm}), \tag{3.13}$$

where $(x_{\text{COM}} - \delta^{\pm})$ represents distance to the edges of the base of support. This ordinary differential equation has a solution of the form

$$x(t) = \left(\frac{f}{\omega^2} + x_0\right)\cosh(\omega t) + \frac{\dot{x}_0}{\omega}\sinh(\omega t) - \frac{f}{\omega^2} \tag{3.14}$$

where $\omega^2 = g/z_0$ and $f = -g\delta/z_0$ with initial conditions $x(0) = x_0$ and $\dot{x}(0) = \dot{x}_0$. This can be combined with the constraint that the COM must be able to come to a stop over the base of support and simplified to give the analytic relationship

$$\delta^- < \frac{\dot{x}_{\text{COM}}}{\omega} + x_{\text{COM}} < \delta^+ \tag{3.15}$$

which is equivalent to a constraint that the capture point [97] remains inside the base of support. This constraint represents a stability region in COM state space, as shown in Figure 3.4, that can be constantly monitored. If the state of the humanoid is outside of this stability region, then ankle torque alone cannot restore balance and either a different balance strategy is needed or a step should be initiated to prevent falling. These stability regions are evaluated on a single inverted pendulum using a PD ankle controller with saturation. Trajectories for this controller are plotted on top of the stability regions in Figure 3.5.

### 3.3.2 CMP Balancing

In this section, the stability region of a Linear Inverted Pendulum plus Flywheel Model is investigated. Compared to Eq. (3.13), which is the maximum acceleration using only ankle torque, the maximum acceleration using both ankle torque and a flywheel torque is defined

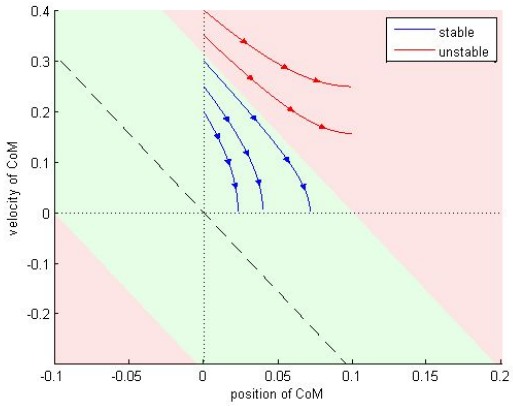


**Figure 3.4:** These are open loop trajectories starting from standing straight up and experiencing an impulse push and using a saturated ankle torque only. All trajectories that start in the green region are stable and all that start in the red will fall.

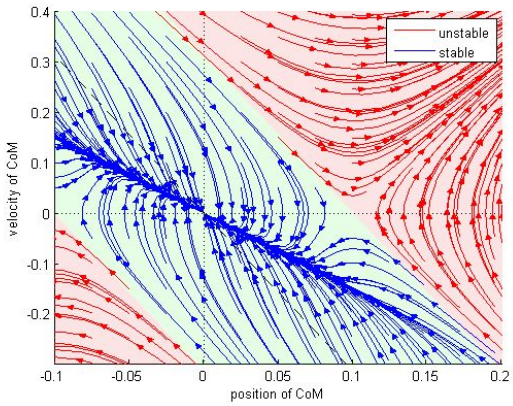


**Figure 3.5:** These are feedback trajectories created by a PD controller (with torque limits) on the ankle joint of a single inverted pendulum. The stability region derived from the linear inverted pendulum model closely predicts the stability of the nonlinear inverted pendulum.

by Eq. (3.10),

$$\ddot{x}^{\max} = \frac{g}{z_0}(x - \delta^{\pm} \pm \frac{\tau_{\max}}{mg}), \tag{3.16}$$

The additional torque term, due to the momentum generating flywheel, allows for a greater maximum horizontal acceleration. Considering only the sagittal (x-direction) dynamics, the dynamics of Eq. (3.10) and Eq. (3.11) can be written simply as

$$\ddot{x} - \omega^2 x = -\omega^2 \left(\delta - \frac{\tau_y}{mg}\right) \tag{3.17}$$

$$I_x \ddot{\theta} = \tau_y \tag{3.18}$$

However, the flywheel really represents inertia of the torso and upper body and should be subject to joint range constraints. Initially, a large torque can be used to accelerate the COM backwards. Soon after, however, a reverse torque must be applied to keep the flywheel from exceeding joint position limits. The goal of the flywheel controller is to return the system to a region that satisfies Eq. (3.15) that avoids exceeding the joint limit constraints. If this cannot be achieved, then a step will be required.

Like COP balancing, we determine the stability regions by considering the largest control action possible. For CMP balancing, this is accomplished using a saturated ankle torque and bang-bang control of the flywheel. Afterwards, we develop a practical controller using optimal control that demonstrates the effect of these stability regions.

### Bang-Bang Control

A bang-bang control input profile to the flywheel can be used to generate analytic solutions. As in [97], the bang-bang function, $\tau_{bb}(t)$, applies maximum torque for a period of $T_1$ then maximum negative torque for a period of $T_2$ and is zero afterwards. This function can be

written as

$$\tau_{bb}(t) = \tau_{\max} + 2\tau_{\max}u(t - T_1) - \tau_{\max}u(t - T_2), \tag{3.19}$$

where $u(t-T_*)$ is a step function at time, $t = T_*$. The goal is to return the system to a state that satisfies Eq. (3.15) at $t = T_2$,

$$\delta^- < \frac{\dot{x}(T_2)}{\omega} + x(T_2) < \delta^+ \tag{3.20}$$

If the system is returned to this state, COP balancing can be used to drive the system back to a stable state and the flywheel can be moved back to zero. We combine Eqs. (3.16) & (3.20) to solve for bounds. We assume the flywheel starts with $\theta(0) = 0$ and $\dot{\theta}(0) = 0$. First, by integrating Eq. (3.18) using the bang-bang input, $\tau_{bb}(t)$, and imposing the constraint, $\dot{\theta}(T_2) = 0$, we get

$$\dot{\theta}(T_2) = \frac{\tau_{\max}}{I}(T_2 - 2(T_2 - T_1)) = 0, \tag{3.21}$$

which requires that $T_2 = 2T_1 = 2T$. Additionally, integrating once more, and using the joint limit constraint, $\theta(2T) < \theta_{\max}$, we find that

$$\theta(2T) = \frac{\tau_{\max}}{I}T^2 < \theta_{\max} \tag{3.22}$$

which means $T$ has a maximum value,

$$T_{\max} = \sqrt{\frac{I\theta_{\max}}{\tau_{\max}}} \tag{3.23}$$

We can again use the derivations from Eq. (3.14) to solve the differential equation in Eq. (3.17). We find that

$$\begin{aligned} x(2T) &= \delta + (x_0 - \delta)\cosh(2\omega T) + \frac{\dot{x}_0}{\omega}\sinh(2\omega T) + \frac{\tau_{\max}}{mL\omega^2}\left(-\cosh(2\omega T) + \cosh(\omega T) - 1\right) \\ \dot{x}(2T) &= \omega(x_0 - \delta)\sinh(2\omega T) + \dot{x}_0\cosh(2\omega T) + \frac{\tau_{\max}}{mL\omega}\left(-\sinh(2\omega T) + 2\sinh(\omega T)\right) \end{aligned}$$

See [97] for more detailed derivations. Now, inserting this into the right-side inequality in Eq. (3.20) and using the identity, $\sinh(x) + \cosh(x) = e^x$, we get

$$\left(x_0 - \delta^+ + \frac{\dot{x}_0}{\omega} - \frac{\tau_{\max}}{mg}\right) e^{2\omega T} + \frac{2\tau_{\max}}{mg} e^{\omega T} - \frac{\tau_{\max}}{mg} < 0 \tag{3.24}$$

$$\left(x_0 - \delta^- + \frac{\dot{x}_0}{\omega} + \frac{\tau_{\max}}{mg}\right) e^{2\omega T} - \frac{2\tau_{\max}}{mg} e^{\omega T} + \frac{\tau_{\max}}{mg} > 0 \tag{3.25}$$

If we assume the worst-case, $T = T_{\max}$, then the inequalities from above become

$$\delta^- - \frac{\tau_{\max}}{mg}(e^{\omega T_{\max}} - 1)^2 < x_0 + \frac{\dot{x}_0}{\omega} < \delta^+ + \frac{\tau_{\max}}{mg}(e^{\omega T_{\max}} - 1)^2 \tag{3.26}$$

If the system is within this stability region, then the system can be stabilized using CMP balancing control. If the system is outside, however, it will have to take a step to avoid falling over. Of course, these bounds overlap the bounds in Eq. (3.15), so the whether or not to use CMP balancing for small perturbations is a decision to be made. However, keeping the torso and head upright is usually a priority, so COP balancing should be used whenever possible. Figure 3.6 and Figure 3.7 show simulations of the bang-bang controller.

### Optimal Control

We can alternatively solve this problem using optimal control. Using this method, we can test whether or not the bounds defined using the bang-bang controller above are realistic. The equations of motion from Eqs. (3.17) & (3.18) can be transformed into the first-order system,

$$\begin{pmatrix} \dot{x} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{\theta} \end{pmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \omega^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{pmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\omega^2 & -(mL)^{-1} \\ 0 & I^{-1} \end{bmatrix} \begin{pmatrix} \delta \\ \tau \end{pmatrix} \tag{3.27}$$

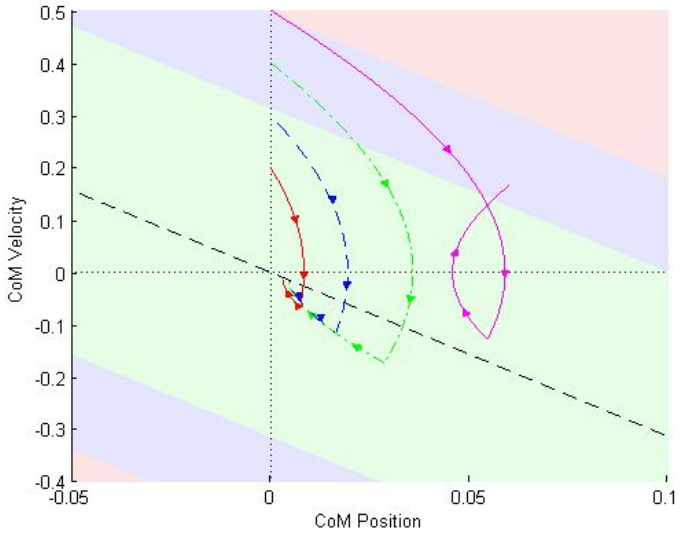


**Figure 3.6:** The phase plot shows how the system responds to bang-bang control of the flywheel along with the stability regions defined by Eq. (3.15) and Eq. (3.26). The extended blue region represents the area where CMP balancing will be successful but COP balancing will not. The flywheel trajectories are shown in Figure 3.7. All trajectories, except the one that starts outside of the stable regions, can be stabilized by bang-bang control.

We can discretize these equations, into $X_{i+1} = AX_i + BU_i$, and generate optimal controls. The quadratic cost function used is

$$J = \sum_{i}^{N} \gamma^{(N-i)} \left\{ X_i^T Q X_i + U_i^T R U_i \right\} \tag{3.28}$$

where $X_i$ is the state, $U_i = (\delta_i, \tau_i)^T$ is the action, and $\gamma < 1$ is a discount factor. The discount factor increases the cost with $i$. Figure 3.8 and Figure 3.9 show simulations of the

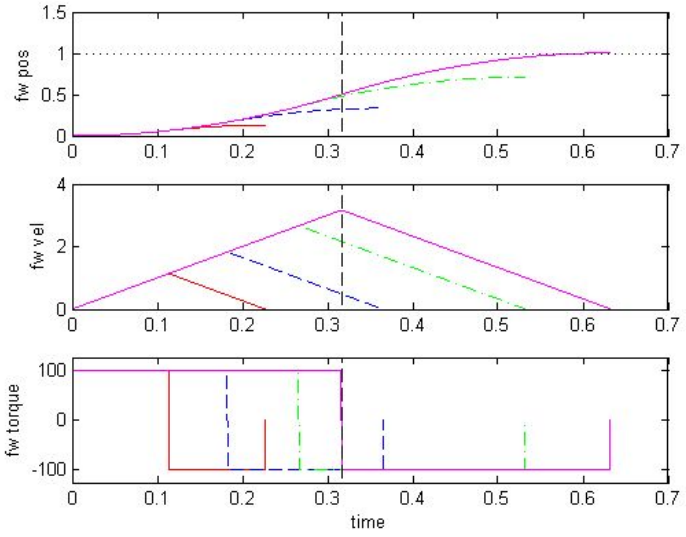


**Figure 3.7:** These time plots show the flywheel response to bang-bang control for the corresponding trajectories in Figure 3.6. In this case, $\theta_{\max} = 1.0rad$, $I = 10kgm^2$ and $\tau_{\max} = 100Nm$. The vertical dashed line represents $T_{\max}$, the maximum time for which the flywheel can be spun up.

optimal control, where the manually adjusted parameters we used were

$$Q = \text{diag}[10^3, 10^3, 10, 10]$$

$$R = \text{diag}[10^{-2}, 10^{-5}]$$

$$\gamma = 0.9$$

Receding horizon control, with a look-ahead time of $1.0s$ and a timestep of $0.02s$, makes $N = 50$. At each timestep, a quadratic programming problem is solved to generate the optimal control trajectory over the $1.0s$ horizon. Notice that one of the trajectories starts inside the COP Balancing region and the optimal controller does not use the flywheel while another starts outside of the stable regions and the controller cannot stabilize it, as expected.

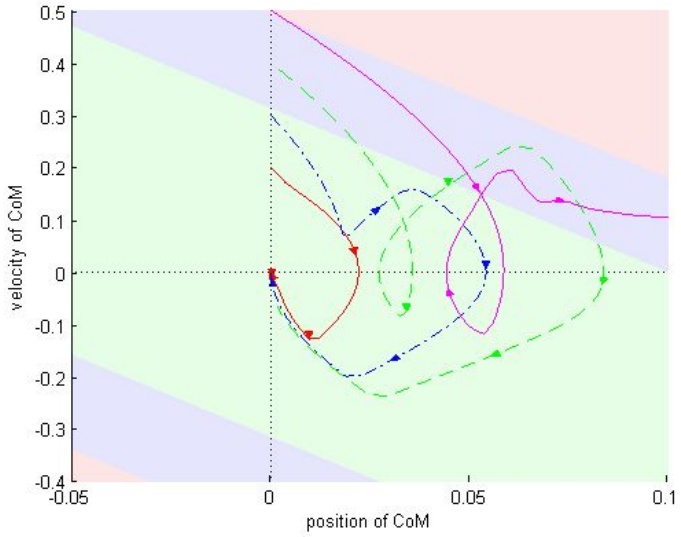


**Figure 3.8:** The phase plot shows how the system can be driven back into the stable region using optimal control. The flywheel trajectories and torques are shown in Figure 3.9. Notice that the unstable starting states are correctly predicted.

## 3.4 Stepping

For even larger disturbances, no feasible body movement without stepping will result in balance recovery. Taking a step moves the support region to either the area between the rear foot contact and the forward foot contact (double support) or the area under the new stance foot (single support). Because a larger disturbance requires a larger restoring force, this generally leads to larger steps. However, the location of the best foot placement is also affected by kinematic constraints and impact dynamics.

In this section two simple models for stepping will be considered. First we consider the LIPM with two finite-size feet. The simple dynamics allow us to analyze the effect of the stepping strategy using linear 2D dynamics and energy formulations [57]. Using this

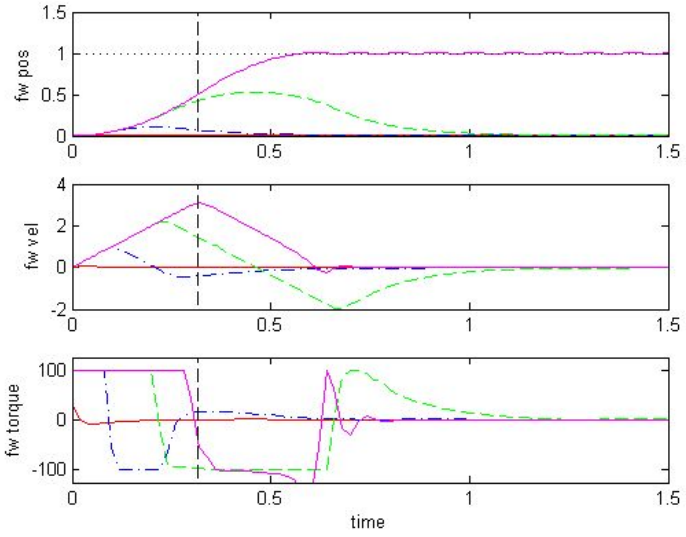


**Figure 3.9:** Optimal flywheel state trajectories and torques for the trajectories in Figure 3.8. The initial state of the flywheel in each case is zero in each case. Notice that for the small perturbation, the optimal controller does not use the flywheel much.

model, it is easy to build an intuitive understanding of the relationship between the motion of the COM and COP as well as the consequences of not being able to move the swing foot instantaneously. Second we consider Hofmann's model [47] which is very similar but models the effects of energy loss associated with impacts.

### 3.4.1 LIPM Stepping

In this section we analyze the dynamics of stepping, which changes the base of support by moving the feet. The high level step recovery policy is fixed and illustrated in Figure 3.10. It is assumed that the system will take 2 steps that result in the system ending at the home pose with zero velocity. The home pose is assumed to be with the two feet coplanar and a prescribed width, $W$, apart with the COM over the center. Given this stepping policy, a

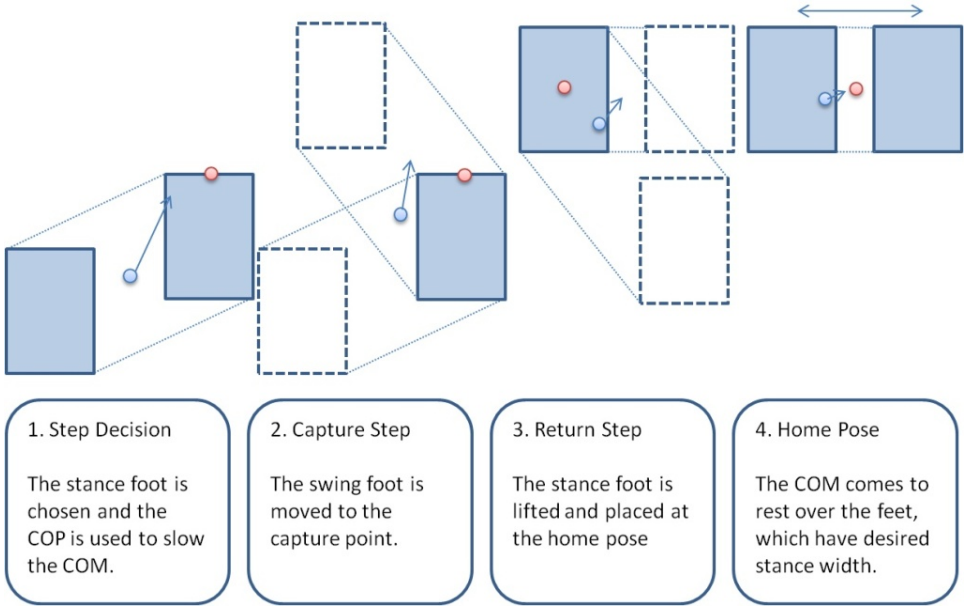


**Figure 3.10:** The 2D stepping policy involves taking two steps such that the feet end parallel and a predetermined width apart.

unique solution for the capture step location can be determined. The result is an open-loop stable trajectory that ends at the home pose.

**Sagittal Step Location.** First, the location of the step in the sagittal direction can be determined by inspecting the stepping policy in Figure 3.11. The system starts outside of the stable region and the stance foot location, $x_S = 0$, is the forward-most foot. The COP, $x_P$, is moved to the edge of the foot in order to slow the COM.

To derive the desired sagittal step location, consider the orbital energy of the system [57]. The trajectory from $(x_1, \dot{x}_1)$ to $(x_2, \dot{x}_2)$ has a constant orbital energy,

$$\frac{1}{2}\dot{x}_1^2 - \frac{\omega^2}{2}(x_1 - x_P)^2 = \frac{1}{2}\dot{x}_2^2 - \frac{\omega^2}{2}(x_2 - x_P)^2 \tag{3.29}$$

Eq. (3.29) is one equation with two unknowns, $x_2$ and $\dot{x}_2$. An additional equation can be

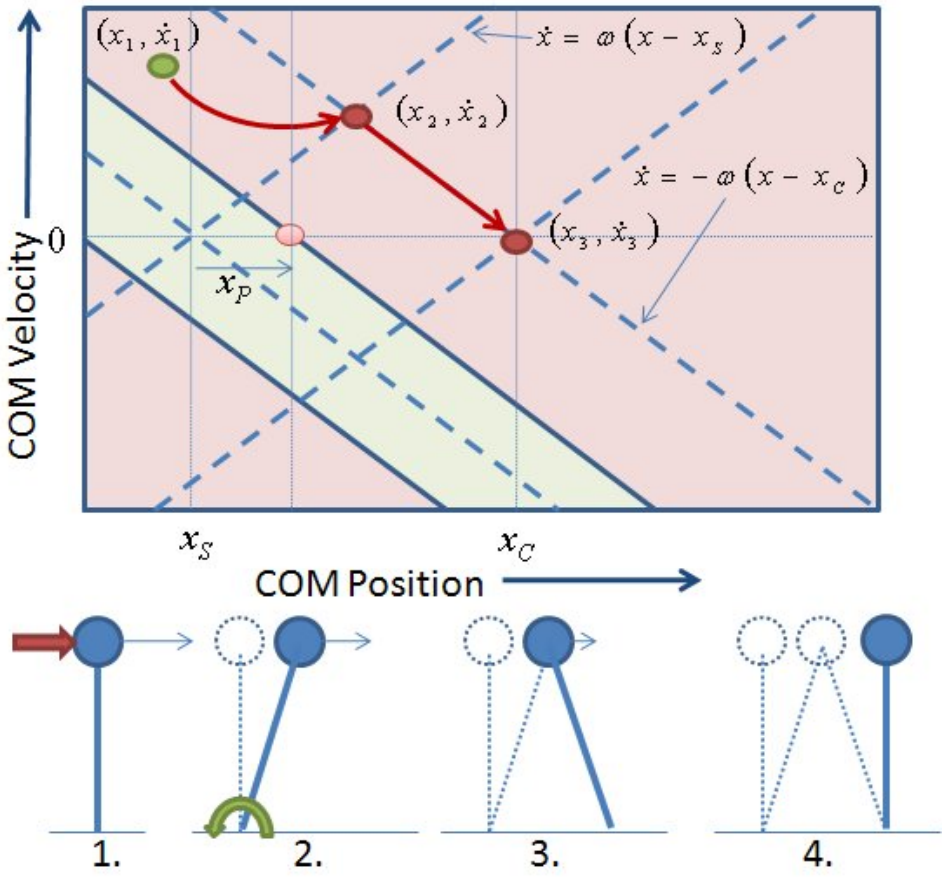


**Figure 3.11:** The sagittal stepping policy uses maximum ankle torque to slow the body, then takes a step at the state that results in an open-loop stable recovery back to the home pose with zero velocity. The trajectory from $(x_1, \dot{x}_1)$ to $(x_2, \dot{x}_2)$ follows a constant orbital energy. It is also assumed that the legs are the same length at impact, which constrains the location of $(x_2, \dot{x}_2)$.

added by assuming that the legs are the same length at impact, which corresponds to

$$\dot{x}_2 = \omega x_2 \tag{3.30}$$

Eq. (3.29) and Eq. (3.30) can be used to solve for $x_2$,

$$x_2 = x_1 + \frac{\dot{x}_1^2 - \omega^2 x_1^2}{2\omega^2 x_P} \tag{3.31}$$

Then the capture step location can be calculated generally as

$$x_C = 2x_2 \tag{3.32}$$

which is relative to the stance foot location, $x_S$.

The time of the capture step, $T_C$, is defined implicitly in the solution in Eq. (3.31). This is the time when the capture point coincides with the swing foot when the legs are symmetric. The value of $T_C$ can be computed numerically using a root-finding algorithm.

**Lateral Step Location.** Given the strategy in Figure 3.10 and the time of the step $T_C$, the lateral state of the COM at the time of the step, $(y_2, \dot{y}_2)$, is defined by a similar orbital energy relationship,

$$\frac{1}{2}\dot{y}_1^2 - \frac{\omega^2}{2}\left(y_1 - y_P\right)^2 = \frac{1}{2}\dot{y}_2^2 - \frac{\omega^2}{2}\left(y_2 - x_P\right)^2 \tag{3.33}$$

which is a first order differential equation. Integrating the equation to $T_C$ gives the lateral state at impact.

The lateral step location is chosen to ensure that the system returns to the home pose with zero velocity during the return step, as shown in Figure 3.12. The trajectory from $(y_2, \dot{y}_2)$ to $(y_3, \dot{y}_3)$ also has a constant orbital energy,

$$\frac{1}{2}\dot{y}_2^2 - \frac{\omega^2}{2}\left(y_2 - y_C\right)^2 = \frac{1}{2}\dot{y}_3^2 - \frac{\omega^2}{2}\left(y_3 - y_C\right)^2 \tag{3.34}$$

The end condition requires that

$$\dot{y}_3 = 0$$

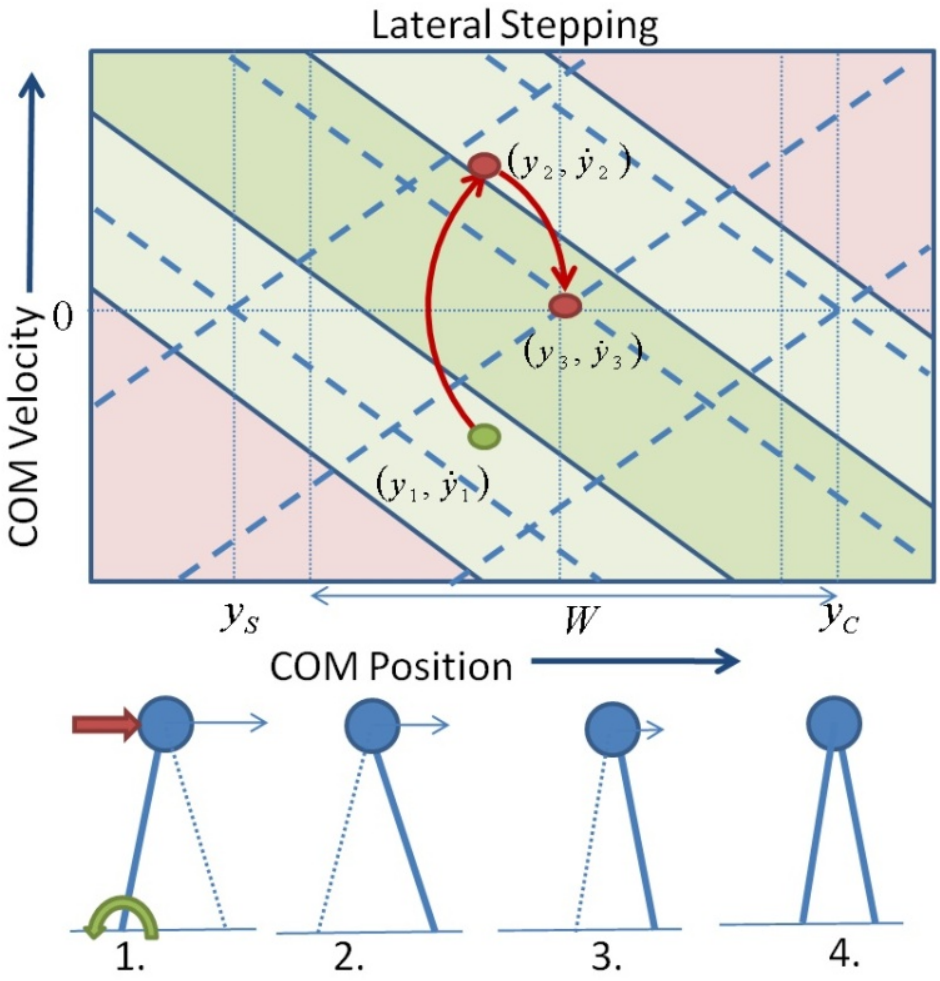


**Figure 3.12:** The lateral stepping policy steps so that after the step the feet return to a home pose.

and

$$y_3 = y_C \pm \frac{W}{2}$$

This assumption can be used with Eq. (3.34) to solve for $y_C$,

$$y_C = y_2 \pm \frac{1}{\omega}\sqrt{\dot{y}_2^2 + \omega^2 \left(\frac{W}{2}\right)^2}$$

where the sign is chosen according to sign $(\dot{y}_2 + \omega y_2)$.

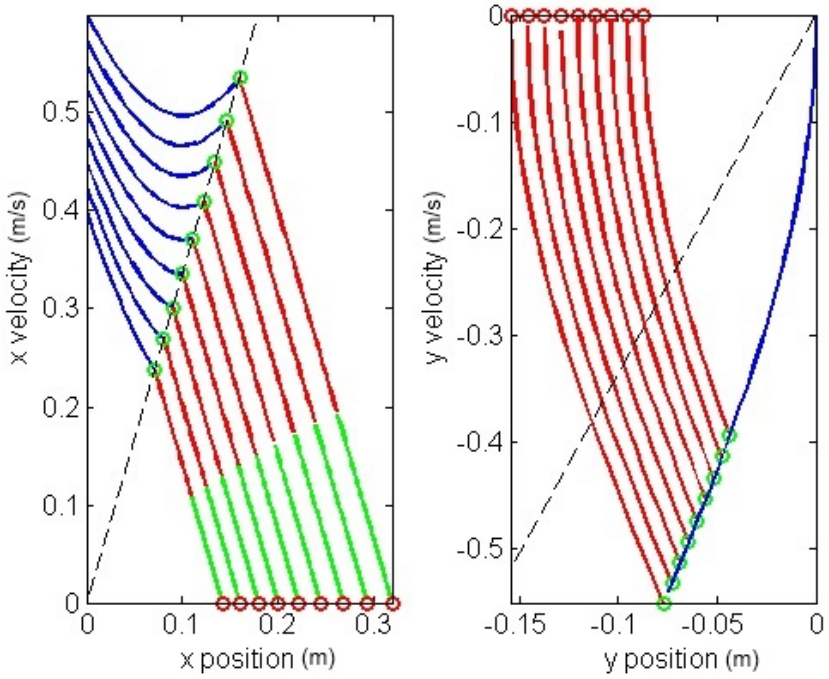


**Figure 3.13:** Comparing the stepping trajectories for different initial conditions corresponding to increasing push size. The blue portion of the trajectory corresponds to the capture step, the red to the return step. The curves in the trajectories are due to the constant orbital energies. Discontinuities correspond to changes in the location of the COP.

**LIPM Stepping Trajectories.** The location of the capture step $(x_C, y_C)$ fully determines the open-loop trajectory of the COM during step recovery. Figure 3.13 shows several of these trajectories which start at the home pose with initial velocity perturbations in the $x$-direction.

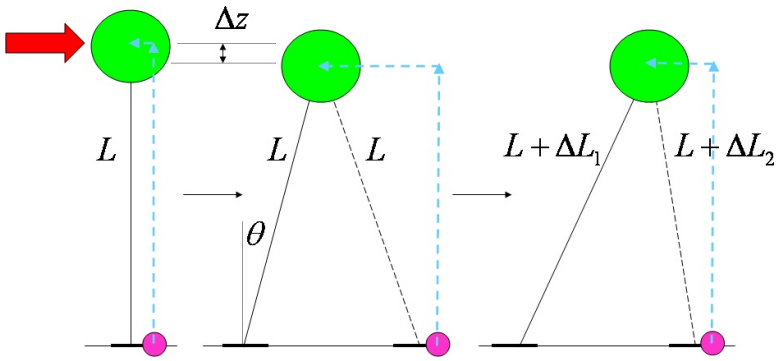


**Figure 3.14:** The model used for step-out has a point mass at the hip with massless, extendable legs. After impact, the body moves horizontally only.

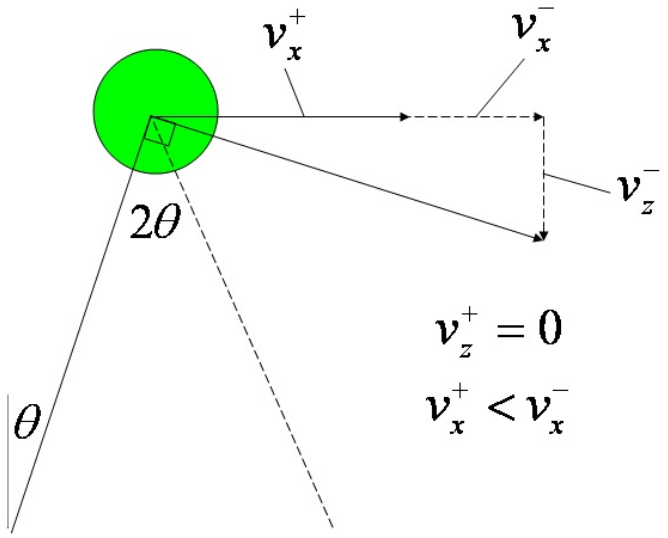


**Figure 3.15:** The impact model used assumes that the vertical velocity of the COM instantaneously goes to zero during the impact. The impulsive force that results in this change also results in a decrease in horizontal velocity.

### 3.4.2 Hofmann Stepping

Hofmann's model [47] is represented by a point mass at the hip and massless, extendable legs. The legs can act like dampers, both instantaneously at impact and continuously during

double support. In this model, we cannot ignore the vertical velocity of the mass, because it affects the impact. However, after the impact occurs, we assume the COM moves horizontally only, and the models we have previously used apply. For this to occur, we assume both legs are at length $L$, just before impact and change accordingly so that the COM moves horizontally, as shown in Figure 3.14.

Considering the point mass, $m$, at the end of the pendulum of length $L$, then the horizontal and vertical velocities are given by

$$\begin{pmatrix} v_x \\ \\ v_z \end{pmatrix} = -L \begin{pmatrix} \cos\theta \\ \\ \sin\theta \end{pmatrix} \dot{\theta} \tag{3.35}$$

During the impact, an impulsive force causes a discontinuous change in these velocities, as illustrated in Figure 3.15, given by the relationship,

$$\frac{\triangle v_x}{\triangle v_z} = \frac{v_x^+ - v_x^-}{v_z^+ - v_z^-} = \tan\theta \tag{3.36}$$

We assume that the vertical velocity goes to zero, or $v_z^+ = 0$, so $\triangle v_z = -v_z^-$. So the change in horizontal velocity is given by

$$\triangle v_x = -v_z^- \tan\theta = L\dot{\theta}\sin\theta\tan\theta \tag{3.37}$$

**Choosing Step Length**

The best choice for step length is the step that results in a transition into a state that is the most robust. Upon impact, there is a discontinuous jump in the COM phase plane. The new position of the COM is behind the stance foot and the velocity is the same, so the jump is horizontal, as illustrated by the dotted lines in Figure 3.16. After the impact, we want the system to be stabilized using COP or CMP Balancing. The best step is one that lies in the middle of the stable region. We want to find the step that results in the relationship,

$$v_x^+ + x\sqrt{\frac{g}{z}} = 0 \tag{3.38}$$

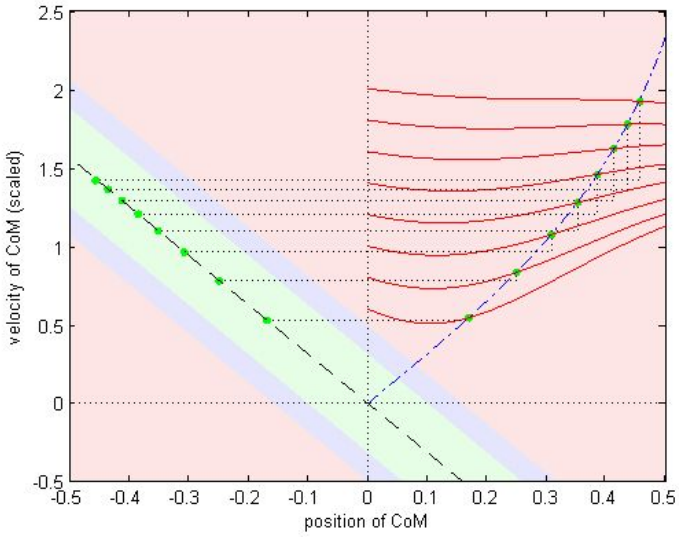


**Figure 3.16:** Starting from an unstable state, taking a step causes a discontinuous jump into a stable region. The transitions shown are the ones that satisfy Eq. (3.38). In this figure, velocities are scaled by $\sqrt{z}$ so each system behaves like $L = 1$.

which places the system on the black dashed line in Figure 3.16. A system starting on this line will be open-loop stable using zero ankle torque. Because there is a large margin for errors and disturbances, this is the most robust foot placement.

We can use Eqs. (3.35), (3.37) & (3.38) to derive a relationship for the step distance. The relationship, in terms of the angle $\theta$, just before impact, is

$$\dot{\theta} = \omega \frac{\sin\theta \cos^{1/2}\theta}{\cos 2\theta} \tag{3.39}$$

Eq. (3.39) is the equation of the blue dash-dotted line in Figure 3.16 that intersects the trajectories of the pendulum at the optimal step distance. There is no analytic solution for the trajectory of the pendulum, but numerical integration is used to find this intersection. Figure 3.17 shows the optimal step distance for starting at a range of initial horizontal COM

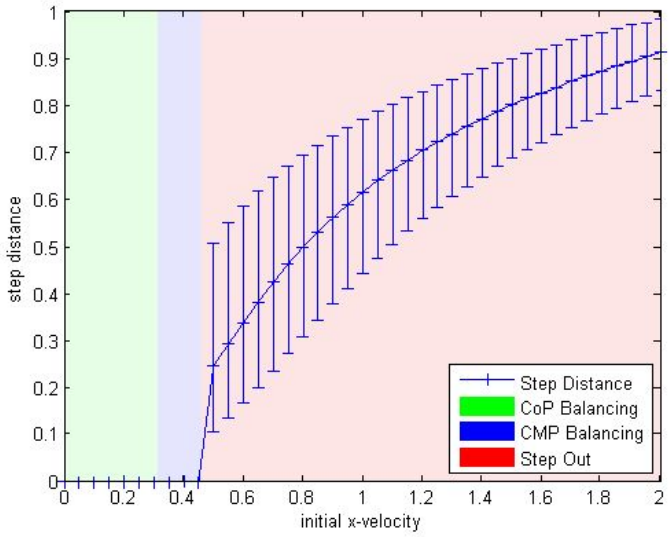


**Figure 3.17:** The optimal step distance vs. the initial horizontal velocity of the COM. The shaded regions show where the various recovery strategies should be employed. The error bars show the bounds on step distance that result in a stable state after impact. The optimal step distance is not half way between the bars because of the nonlinear relationship.

velocities. For illustration, error bars are attached to each point showing the range of step distances that would also result in a stable state.

## 3.5 Discussion

There are several assumptions made in this chapter. In the case of stepping, massless legs that instantly appear at the point of contact were assumed, whereas real legs have inertia that must be accelerated and also causes internal forces. Swing leg masses could be integrated into simple models if needed to better model this effect. Nonlinear pendulum-like models are very

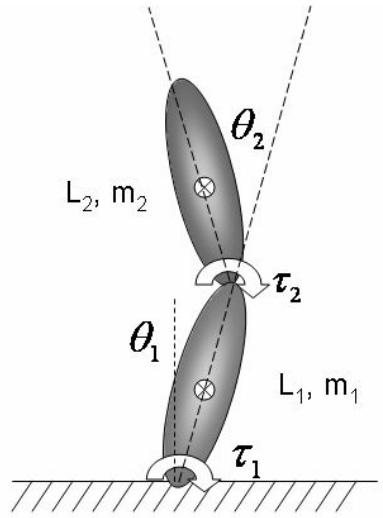


**Figure 3.18:** A double inverted pendulum with foot can apply torques at the ankle and hip joints. The location of the center of pressure must be underneath the foot.

realistic but linearized versions may also closely approximate the dynamics. Throughout the rest of this thesis, only the simplest models, like the LIPM and LIPM+Flywheel models, are considered, but other models could be easily substituted in many cases. Other models have also been used for modeling and performing push recovery and are described below.

### 3.5.1 Double Inverted Pendulum

A double inverted pendulum is a fully-actuated, unconstrained planar dynamic systems with nonlinear equations of motion of the form,

$$M(\theta)\ddot{\theta} = \tau - N(\theta, \dot{\theta}) \tag{3.40}$$

where $\theta$ is a vector of joint angles, $M$ is the inertia matrix, $\tau$ are the joint torques, and $N$ is a vector containing the gravitational, centripetal and coriolis forces.

These planar inverted pendulum models, like the one in Figure 3.18, can be used to

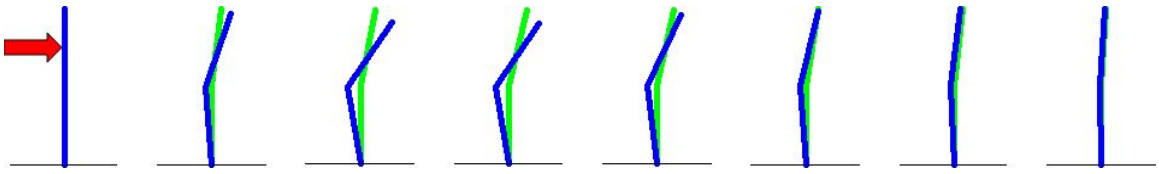

**Figure 3.19:** Animation comparing the responses of an ankle-strategy controller (light green) and a hip-strategy controller (dark blue) to a 17Ns impulse. The hip-strategy controller produces a large bend at the hips, similar to the same strategy used by humans.

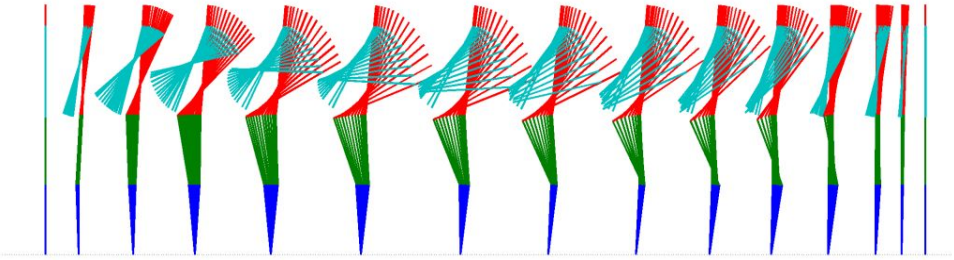

**Figure 3.20:** A four-link inverted pendulum models ankle, knee, hip and shoulder joints. These different figures represent optimal trajectories for recovering from multiple different forward pushes.

derive sagittal balance controllers, such as the hip strategy controller depicted in Figure 3.19 and described in more detail in [113]. Often in these models there is no explicit foot; it is assumed to be flat on the ground at all times, contributing no kinetic or potential energy. The definition of the center of pressure still applies as long as it remains within the area the foot would cover.

### 3.5.2 Multiple Pendulums

Multiple inverted pendulums can also be used to approximate the dynamics of a full robot. A four-link model is used in Figure 3.20, which illustrates how multiple optimal trajectories can potentially be used to automatically define a control policy for push recovery.

## 3.6 Conclusion

Simple models have been shown to approximate the motion of humanoids in the case of recovering from large disturbances. These simple models are built upon the prior work of biomechanics and robotics researchers. From these models, simple bounds on stability are defined that can be applied to complex humanoid robots or human subjects to predict a fall or choose a balance strategy.

In the remaining chapters, simple models of robot dynamics (namely the LIPM and LIPM + Flywheel) will be used to perform optimal control (Chapter 4), generate full-body torques (Chapter 5), and state estimation (Chapter 6).

# Chapter 4

# Push Recovery Model Predictive Control

In the previous chapter, simple models of robot dynamics were derived and used to predict the stability of different balance recovery strategies such as the ankle and hip strategy as well as stepping. In this chapter, optimal control will be used to produce push recovery controls using these models. Later, in Chapter 5, these controls will be mapped into full-body torque controls to control the humanoid robot.

## 4.1 Introduction

Humanoid robots, while operating in complex environments, can be expected to encounter uneven ground, dynamic obstacles, and humans. Force controlled robots, as opposed to stiff position controlled robots, can be compliant to unknown disturbances, resulting in safer and more robust operation. For small disturbances, standing balance is sufficient. However, for locomotion and large disturbances, the robot needs to step. The tight coupling between

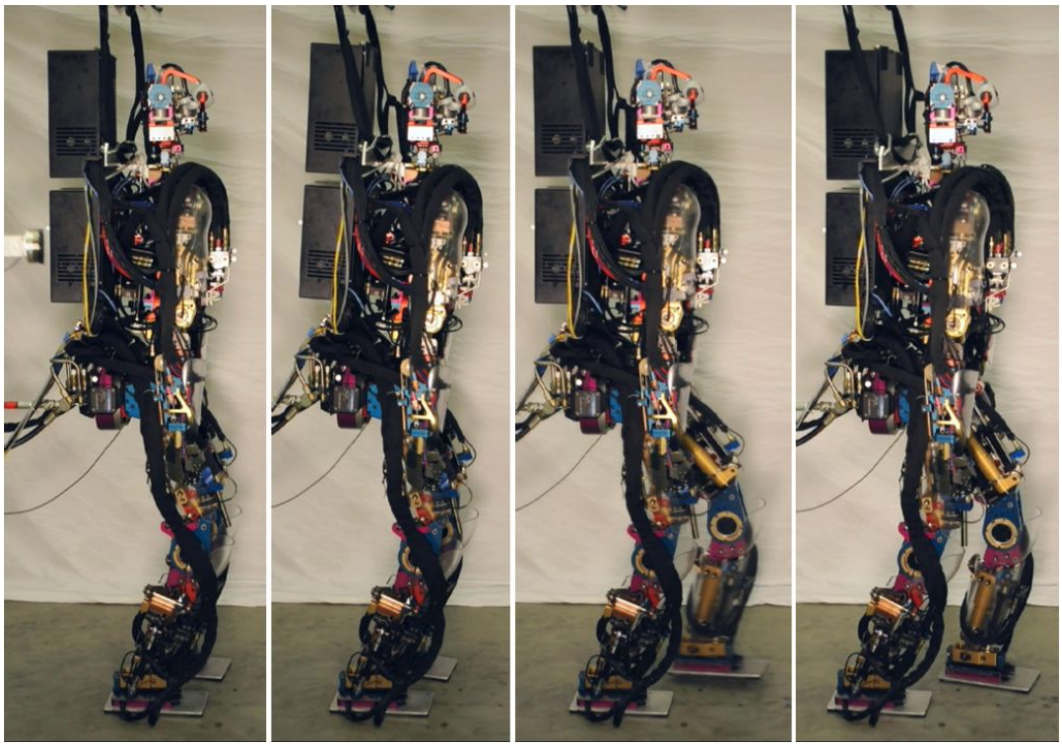

**Figure 4.1:** The controller presented in this chapter allows a humanoid robot to recover from a large perturbation by stepping. It is applied to the Sarcos Primus hydraulic humanoid robot pictured being pushed from behind by a force-measuring stick.

balance control and choice of footstep location makes this a challenging problem. This chapter presents a method for planning and controlling stepping in a force controlled robot, as shown in Figure 4.1.

The dynamics of balance are often described using simple models of the center of mass [112]. Given a robot with stiff joint position control and a known environment, the most common approach to balance is to generate a stable trajectory of the COM and then track it using inverse kinematics (IK) [122]. Stable walking patterns for walking robots are often generated using model predictive control (MPC), a trajectory optimization technique [86].

Using the LIPM, the trajectory optimization simplifies to a quadratic programming (QP) problem. The simplest case is one that assumes the footstep locations are given ahead of time, as in the approaches of Kajita, *et al.* , [53] and Wieber, *et al.* , [132]. These implementations of MPC for walking have been used only for pattern generation. That is, the controller generates an open-loop reference motion that is closely tracked. With a well-designed open-loop motion and known environment, stable walking can be achieved. This type of control is common amongst stiff position-controlled robots and is convenient for automatically generating patterns with varying speeds and objectives. The more recent work by Herdt, *et al.* , [44], emphasizes this control approach in the paper entitled "Walking Without Thinking About It".

Stepping strategies have been considered by several authors. Simple models have been used to define stable footstep locations, known as Capture Points [97]. Robots with stiff position control that expect small disturbances often solve footstep planning separately [15]. For situations when desired footstep locations cannot be known in advance, such as in the presence of large disturbances, motion and footstep planning can be performed simultaneously using a QP-based MPC algorithm [21][44].

In this chapter, an MPC algorithm called Push Recovery Model Predictive Control (PR-MPC) is presented. The basic MPC algorithm that is run inside the controller is the same as the one presented by Diedam, *et al.* , [21]. However, instead of open-loop pattern generation, the optimization is recomputed online using the current estimate of the state of the system to form a feedback controller that outputs a desired COP and footstep locations to maintain balance, recover from pushes and perform dynamic walking. This type of reactive balance control is necessary for robots with force-controlled joints, such as the Sarcos humanoid robot. **The main contribution of this chapter is the formulation of PR-MPC and**

**implementation of real-time control on the Sarcos Primus humanoid robot**.

This chapter is organized as follows. First in Section 4.2, the concept and notation for MPC is presented. PR-MPC is presented in Section 4.3 by describing a special form of the objective function and constraints. Some implementation details that allow this controller to run in realtime are also presented. This basic algorithm can be extended in a number of ways. For example, Section 4.4 describes an algorithm that builds on the work on stability boundaries in Chapter 3 and performs multiple optimizations of different strategies in parallel called Multi-Strategy PR-MPC. In addition, Section 4.5 presents an algorithm for trajectory optimization in the presence of a learned disturbance is presented along with basic results. Finally, in Section 4.6 some experimental results on the Sarcos humanoid robot are shown.

## 4.2 Model Predictive Control

Model predictive control (MPC), also known as receding horizon control, is a general term for a type of control that uses a dynamics model to predict the behavior of the system given a sequence of actions to be applied [86]. At its core, MPC uses a trajectory optimization technique to output an optimal trajectory from the initial state. Because the model and the real system will differ slightly, the real system will diverge from the predicted trajectory. For this reason, the trajectory optimization is periodically re-run online to update the predicted optimal trajectory. The real power of MPC is the consideration of future constraints. Because the algorithm is optimizing a trajectory into the future, it can predict appropriate actions to avoid violating constraints in the future.

Given a sequence of control inputs, $\bar{\mathbf{U}}$, a linear model, $\mathbf{X}_{t+1} = \mathbf{A}\mathbf{X}_t + \mathbf{B}\mathbf{U}_t$ can be

converted into a sequence of states, $\bar{\mathbf{X}}$, for the next $N$ timesteps,

$$\bar{\mathbf{X}} = \bar{\mathbf{A}}\mathbf{X}_t + \bar{\mathbf{B}}\bar{\mathbf{U}}, \tag{4.1}$$

where

$$\bar{\mathbf{X}} = \left(\mathbf{X}_{t+1}^T, \ldots, \mathbf{X}_{t+N}^T\right)^T, \tag{4.2}$$

$$\bar{\mathbf{U}} = \left(\mathbf{U}_{t}^T, \ldots, \mathbf{U}_{t+N-1}^T\right)^T, \tag{4.3}$$

and $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ have the form

$$\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^{N-1} \\ \mathbf{A}^N \end{bmatrix} \tag{4.4}$$

and

$$\bar{\mathbf{B}} = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & & \vdots \\ \mathbf{A}^{N-2}\mathbf{B} & \mathbf{A}^{N-3}\mathbf{B} & & \mathbf{B} & \mathbf{0} \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} \cdots & \cdots & \mathbf{AB} & \mathbf{B} \end{bmatrix} \tag{4.5}$$

respectively. Eq. (4.1) is a simple function that maps from the current state, $\mathbf{X}_t$, to a trajectory, $\bar{\mathbf{X}}$, where the actions over the trajectory, $\bar{\mathbf{U}}$ are the parameters. In order to determine the best trajectory, a cost function is constructed. A cost function is a scalar function that scores the trajectory. Model predictive controllers are typically derived by defining a quadratic cost on the elements of $\bar{\mathbf{X}}$ and $\bar{\mathbf{U}}$ such that

$$J = \frac{1}{2}\bar{\mathbf{X}}^T\bar{\mathbf{Q}}\bar{\mathbf{X}} + \frac{1}{2}\bar{\mathbf{U}}^T\bar{\mathbf{R}}\bar{\mathbf{U}} \tag{4.6}$$

By inserting Eq. (4.1) into Eq. (4.6), the cost can be written as a function of $\bar{\mathbf{U}}$, such that

$$J = \frac{1}{2}\bar{\mathbf{U}}^T \left(\bar{\mathbf{R}} + \bar{\mathbf{B}}^T \bar{\mathbf{Q}} \bar{\mathbf{B}}\right) \bar{\mathbf{U}} + \mathbf{X}_t^T \bar{\mathbf{A}}^T \bar{\mathbf{Q}} \bar{\mathbf{B}} \bar{\mathbf{U}} + \frac{1}{2}\mathbf{X}_t^T \bar{\mathbf{A}}^T \bar{\mathbf{Q}} \bar{\mathbf{A}} \mathbf{X}_t \tag{4.7}$$

which has a simple quadratic form,

$$J = \frac{1}{2}\bar{\mathbf{U}}^T \mathbf{H} \bar{\mathbf{U}} + \mathbf{f}^T \bar{\mathbf{U}} + J_0 \tag{4.8}$$

where

$$\mathbf{H} = \bar{\mathbf{R}} + \bar{\mathbf{B}}^T \bar{\mathbf{Q}} \bar{\mathbf{B}}, \tag{4.9}$$

$$\mathbf{f^T} = \mathbf{X}_t^T \bar{\mathbf{A}}^T \bar{\mathbf{Q}} \bar{\mathbf{B}} \tag{4.10}$$

and

$$J_0 = \frac{1}{2}\mathbf{X}_t^T \bar{\mathbf{A}}^T \bar{\mathbf{Q}} \bar{\mathbf{A}} \mathbf{X}_t \tag{4.11}$$

Note that the $J_0$ term is constant with respect to $\bar{\mathbf{U}}$ and has no effect on the location of the minimum of $J$.

Without constraints, the optimal trajectory can be found by differentiating Eq. (4.8) with respect to $\bar{\mathbf{U}}$, setting the result to zero, and solving for the optimum trajectory, $\bar{\mathbf{U}}^*$, which becomes

$$\bar{\mathbf{U}}^* = -\mathbf{H}^{-1}\mathbf{f} \tag{4.12}$$

However, constraints are often imposed on the system in the form of equality constraints,

$$\mathbf{C}_{eq} \bar{\mathbf{U}} = \mathbf{b}_{eq} \tag{4.13}$$

or inequality constraints

$$\mathbf{C}_{in} \bar{\mathbf{U}} \leq \mathbf{b}_{in} \tag{4.14}$$

State constraints of the form

$$\mathbf{C}_{in}\bar{\mathbf{X}} \leq \mathbf{b}_{in} \tag{4.15}$$

can also be implemented simply by using Eq. (4.1) to convert into functions of $\bar{\mathbf{U}}$ such that

$$\mathbf{C}_{in}\bar{\mathbf{B}}\bar{\mathbf{U}} \leq \mathbf{b}_{in} - \mathbf{C}_{in}\bar{\mathbf{A}}\mathbf{X}_t \tag{4.16}$$

In the presence of constraints, a quadratic programming (QP) solver is required. QP solvers iteratively solve simpler sub-problems to find the optimal result that simultaneously satisfies the constraints. More details about solving the QP problem will be discussed in Section 4.3.3. There are several types of solvers, each of which is usually suited for a specific form of the problem. For example, active sets, interior points, conjugant gradient or simplex methods can be used [88].

## 4.3 Push Recovery Model Predictive Control

In this section, Push Recovery Model Predictive Control (PR-MPC) is described. This controller operates on the actual state of the robot to generate reactive balance controls, such as desired COP and next footstep locations, by solving a trajectory optimization considering future actions and constraints.

**What is the model?** For all of the implementations presented, the model is the LIPM with 2D dynamics given by Eq. (3.5). However, the math in this chapter is generally independent of the model. It is straightforward to implement any other linear model. Nonlinear models can also be handled by linearizing along the nominal trajectory and iterating the optimization (also known as Sequential Quadratic Programming (SQP)).

### 4.3.1 Objective Function

The goal of PR-MPC is to bring the COM to rest over the centroid of the support region with both feet on the ground. The footstep locations are made variable, allowing the controller to adaptively choose step locations for push recovery. The timings of the stepping phases, including both single and double support, are fixed ahead of time, as is the number of steps to look ahead and the choice of initial swing foot.

**A note on fixed timings:** Timings are fixed because they appear nonlinearly in Eq. (4.1). Non-constant timings can be handled in several ways. The simplest way is to recompute the trajectory with different step timings and choose the best. The other choice is to make the step timings a variable and reformulate the MPC into an SQP problem.

The objective function for PR-MPC, which guides the COM to stop over the center of the feet, can be written as

$$J = \frac{w_1}{2}\left|\left|\mathbf{C}_X^{\text{goal}} - \mathbf{C}_X\right|\right|^2 + \frac{w_2}{2}\left|\left|\dot{\mathbf{C}}_X\right|\right|^2 + \frac{w_3}{2}\left|\left|\mathbf{U}_X\right|\right|^2 + \frac{w_4}{2}\left|\left|p_X^{\text{ref}} - p_X\right|\right|^2 + \frac{w_1}{2}\left|\left|\mathbf{C}_Y^{\text{goal}} - \mathbf{C}_Y\right|\right|^2 + \frac{w_2}{2}\left|\left|\dot{\mathbf{C}}_Y\right|\right|^2 + \frac{w_3}{2}\left|\left|\mathbf{U}_Y\right|\right|^2 + \frac{w_4}{2}\left|\left|p_Y^{\text{ref}} - p_Y\right|\right|^2 \tag{4.17}$$

where $\mathbf{C}_X$ and $\mathbf{C}_Y$ are vectors of COM positions in 2D over the next $N$ timesteps, $\left(\mathbf{C}_X^{\text{goal}}, \mathbf{C}_Y^{\text{goal}}\right)$ is the goal position, $\dot{\mathbf{C}}_X$ and $\dot{\mathbf{C}}_Y$ are the velocities, $\mathbf{U}_X$ and $\mathbf{U}_Y$ are the inputs (in this case, the derivatives of the COP), and $p^{\text{ref}}$ and $p$ are vectors of the next $M$ reference and actual footstep locations, respectively. For the examples in this section, it will be assumed that $M = 1$, but it is easily generalized to more footsteps. Reference footstep locations can be used to bias towards a desired step width or step length.

As shown in Figure 4.2, if $p_X^0$, $p_X^1$ and $p_X^2$ are the swing foot, stance foot and next footstep

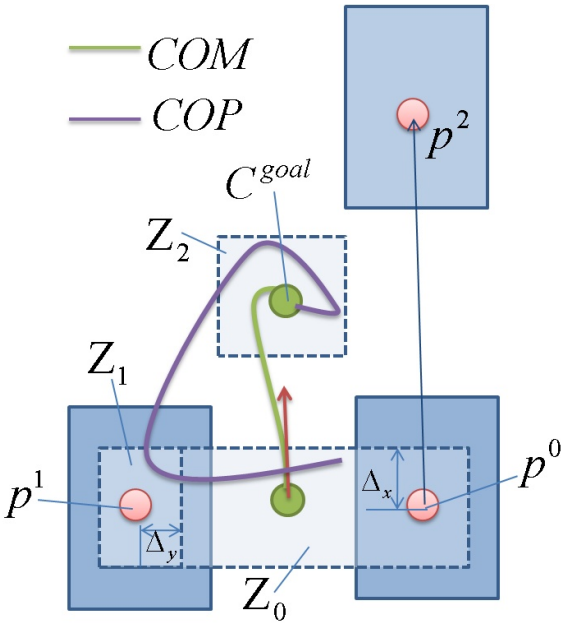


**Figure 4.2:** Illustration of one step lookahead with right leg stepping. The COP is used to shift the COM slightly to initially put more weight on the left foot. The COM comes to rest at the midpoint between the two feet after the step.

locations, respectively, then the COM goal position, $C_X^{\text{goal}}$ is given by

$$\mathbf{C}_X^{\text{goal}} = \frac{\mathbf{O}_N}{2}\left(p_X^1 + p_X^2\right) \tag{4.18}$$

$$\mathbf{C}_Y^{\text{goal}} = \frac{\mathbf{O}_N}{2}\left(p_Y^1 + p_Y^2\right) \tag{4.19}$$

where $\mathbf{O}_N$ is a vector of ones $N$ long. The desired position is located half way between the location of the final footstep locations.

The variables in Eq. (4.17) can be written as functions of the initial state, $\mathbf{X}_t$, and the

unknown control sequence, $\bar{\mathbf{U}}$, such that

$$\mathbf{C}_X = \mathbf{A_{cx}}\mathbf{X}_t + \mathbf{B_{cx}}\bar{\mathbf{U}} \tag{4.20}$$

$$\mathbf{C}_Y = \mathbf{A_{cy}}\mathbf{X}_t + \mathbf{B_{cy}}\bar{\mathbf{U}} \tag{4.21}$$

$$\dot{\mathbf{C}}_X = \mathbf{A_{vx}}\mathbf{X}_t + \mathbf{B_{vx}}\bar{\mathbf{U}} \tag{4.22}$$

$$\dot{\mathbf{C}}_Y = \mathbf{A_{vy}}\mathbf{X}_t + \mathbf{B_{vy}}\bar{\mathbf{U}} \tag{4.23}$$

where $\mathbf{A_{cx}}$ and $\mathbf{B_{cx}}$ are matrices corresponding to the appropriate rows of Eq. (4.4) and Eq. (4.5), respectively. Using this notation, Eq. (4.17) can be converted into a quadratic equation in the form of Eq. (4.8). This equation is quadratic in the inputs: control input $\bar{\mathbf{U}}$ over the next $N$ timesteps and the next $M$ footstep locations. Therefore, the dimensionality of the quadratic programming problem is $2(N + M)$.

### 4.3.2 Constraints

Constraints play a major role in the computation of the optimal trajectory. If it were not for the inequality constraints on the COP and footstep locations, the cost function in Eq. (4.17) could be solved using Eq. (4.12). Instead, inequality constraints in the form of Eq. (4.14) are added and the problem is solved using constrained quadratic programming.

It should also be noted that because the locations of the footsteps are variables in the optimization, meaning the constraints on the COP after the first step are unknown, the true problem is nonlinear. To maintain a simple form and take advantage of fast QP solvers, simple conservative constraints are chosen to approximate the true constraints. The chosen constraints are conservative because they represent an area smaller than, but strictly inside, the true double support constraints.

As shown in Figure 4.2, a COP constraint region, $Z_i$, is defined for each phase of the step. The first phase is assumed to be a short double support phase used to shift the weight over to the stance foot. The constraint region associated with this phase, $Z_0$, is known exactly because the current locations of the feet are known. Likewise, the constraint region during the swing phase, $Z_1$, is also known. However, the final constraint region, $Z_2$, is not known because the location of the footstep is not known in advance. A hand-crafted linear approximation of the final constraint region is shown. Notice also that the constraints assume smaller-than-actual-size feet. This assumption is useful to help compensate for some of the unmodeled dynamics of the humanoid robot.

These constraints can be written in a compact form for use in standard quadratic programming software. Let $\mathbf{O}_i$ be defined as a vector of zeros and ones with the ones corresponding to the timesteps of the $i$-th phase, with the first phase being the initial double support phase. For a single footstep lookahead, $i$ will range from 0 to 2. Notice that $\sum_i \mathbf{O}_i = \mathbf{O}_N$. The constraints on the $X$-trajectory of the COP, $\mathbf{Z}_X$, can be written as

$$\mathbf{O}_0 \left(\min\left(p_X^0, p_X^1\right) - \Delta_x\right) + \mathbf{O}_1 \left(p_X^1 - \Delta_x\right) + \mathbf{O}_2 \left(\mathbf{C}_X^{\text{goal}} - \Delta_x\right) \leq \mathbf{Z}_X \tag{4.24}$$

$$-\mathbf{O}_0 \left(\max\left(p_X^0, p_X^1\right) + \Delta_x\right) - \mathbf{O}_1 \left(p_X^1 + \Delta_x\right) - \mathbf{O}_2 \left(\mathbf{C}_X^{\text{goal}} + \Delta_x\right) \leq -\mathbf{Z}_X \tag{4.25}$$

where $\Delta_x$ is the distance from the center of the foot to the edge in the $x$-direction. If $\mathbf{Z}_X$ is factored just as in Eq. (4.20) such that

$$\mathbf{Z}_X \;=\; \mathbf{A_{zx}} \mathbf{X}_t + \mathbf{B_{zx}} \bar{\mathbf{U}} \tag{4.26}$$

then Eq. (4.24) and Eq. (4.25) can be re-written in the form of Eq. (4.14),

$$-\mathbf{B_{zx}} \bar{\mathbf{U}} + \mathbf{O}_2 \frac{1}{2} p_X^2 \leq -\left(\mathbf{O}_0 \min\left(p_X^0, p_X^1\right) + \mathbf{O}_1 p_X^1 + \mathbf{O}_2 \frac{1}{2} p_X^1 - \mathbf{O}_N \Delta_x\right) + \mathbf{A_{zx}} \mathbf{X}_t \tag{4.27}$$

$$\mathbf{B_{zx}} \bar{\mathbf{U}} - \mathbf{O}_2 \frac{1}{2} p_X^2 \leq \left(\mathbf{O}_0 \max\left(p_X^0, p_X^1\right) + \mathbf{O}_1 p_X^1 + \mathbf{O}_2 \frac{1}{2} p_X^1 + \mathbf{O}_N \Delta_x\right) - \mathbf{A_{zx}} \mathbf{X}_t \tag{4.28}$$

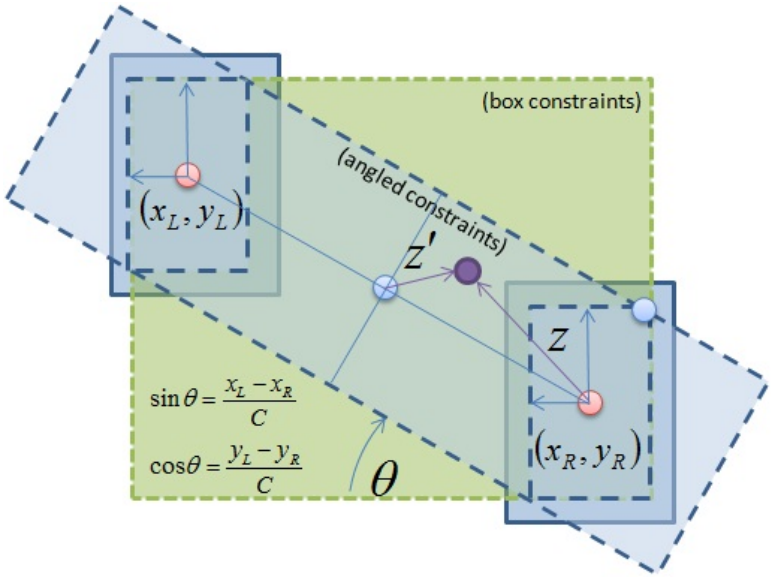


**Figure 4.3:** COP constraints during double support are divided into box constraints and angled constraints. If the feet positions are known, the COP can be rotated into the angled frame to give the angled constraints.

where Eq. (4.18) was substituted for $\mathbf{C}_X^{\text{goal}}$ and the footstep location $p_X^2$ was moved to the left side of the equation to signify that it is a variable as in Eq. (4.14).

### Double Support Constraints

During most of the phase, the constraints in Eq. (4.24) and Eq. (4.25) take a simplified form because they are simple box constraints. However, if the optimization begins with a double support phase (or there are no steps at all), the true constraints can be derived because the positions of both feet are known. Figure 4.3 shows the constraints during this initial phase. The constraints are made up of box constraints given simply by the maxima and minima of the edges of the feet (as in Eq. (4.27) and Eq. (4.28)) and angled constraints created by rotating constraints according to the positions of the feet. If $z'_x$ is the COP in the $x$-direction

of the rotated frame, then the angled constraints are

$$-\Delta \leq z'_x \leq \Delta \tag{4.29}$$

where $\Delta$ is the distance from the center to the corner of the foot. $z'_x$ can be found by rotating relative to the right foot such that

$$z'_x = (z_x - x_R)\, cos\theta - (z_y - y_R)\, sin\theta \tag{4.30}$$

where

$$\theta = tan^{-1} \frac{x_L - x_R}{y_L - y_R}$$

Eq. (4.30) can be re-written as

$$z'_x = \frac{1}{C}\left((z_x - x_R)\,(y_L - y_R) - (z_y - y_R)\,(x_L - x_R)\right) \tag{4.31}$$

such that $C = \sqrt{(x_L - x_R)^2 + (y_L - y_R)^2}$. $z_x$ and $z_y$ are the unknowns in this equation which can be rearranged into

$$Cz'_x = \begin{bmatrix} (y_L - y_R) & -(x_L - x_R) \end{bmatrix} \begin{pmatrix} z_x \\ z_y \end{pmatrix} - x_R\,(y_L - y_R) + y_R\,(x_L - x_R) \tag{4.32}$$

Again using the factored notation in Eq. (4.26), Eq. (4.32) can be combined with Eq. (4.29) and converted into vector equations

$$\begin{aligned} \left(\mathbf{A_{zx}X}_t + \mathbf{B_{zx}}\bar{\mathbf{U}} - \mathbf{O}_0 x_R\right)(y_L - y_R) - \left(\mathbf{A_{zy}X}_t + \mathbf{B_{zy}}\bar{\mathbf{U}} + \mathbf{O}_0 y_R\right)(x_L - x_R) &\leq C\Delta \\ -\left(\mathbf{A_{zx}X}_t + \mathbf{B_{zx}}\bar{\mathbf{U}} + \mathbf{O}_0 x_R\right)(y_L - y_R) + \left(\mathbf{A_{zy}X}_t + \mathbf{B_{zy}}\bar{\mathbf{U}} + \mathbf{O}_0 y_R\right)(x_L - x_R) &\leq -C\Delta \end{aligned}$$

which can be further converted to be consistent with Eq. (4.14),

$$mB_{zx}\bar{\mathbf{U}}\left(y_L - y_R\right) - \mathbf{B_{zy}}\bar{\mathbf{U}}\left(x_L - x_R\right) \leq$$

$$C\Delta - \left(\mathbf{A_{zx}}\mathbf{X}_t - \mathbf{O}_0 x_R\right)\left(y_L - y_R\right) + \left(\mathbf{A_{zy}}\mathbf{X}_t + \mathbf{O}_0 y_R\right)\left(x_L - x_R\right) \tag{4.33}$$

$$-\mathbf{B_{zx}}\bar{\mathbf{U}}\left(y_L - y_R\right) + \mathbf{B_{zy}}\bar{\mathbf{U}}\left(x_L - x_R\right) \leq$$

$$-C\Delta + \left(\mathbf{A_{zx}}\mathbf{X}_t + \mathbf{O}_0 x_R\right)\left(y_L - y_R\right) - \left(\mathbf{A_{zy}}\mathbf{X}_t + \mathbf{O}_0 y_R\right)\left(x_L - x_R\right) \tag{4.34}$$

It can be seen from this formulation how the double support constraints are nonlinear when the locations of the feet are unknown because of the multiplication of foot position variables with the inputs.

**Alternate Forms**

There are several different forms of the constraints that could be used to achieve slightly different results. For example, an intermediate double support phase could be modeled with a region based on both feet, such as $Z_2$ in Figure 4.2. Or it could be modeled such that all of the weight is instantly transferred to the next stance foot. Likewise, other behaviors such as push-off and heel-strike could be modeled by constraining the COP to be at the toes or heel at specific times during the phase.

### 4.3.3 Solving the QP

To solve for the step recovery trajectory, constrained quadratic programming is used to solve for the minimum of the objective function in Eq. (4.17) subject to the constraints in Eq. (4.24) and Eq. (4.25). When Eq. (4.1) is substituted into these equations, the result takes

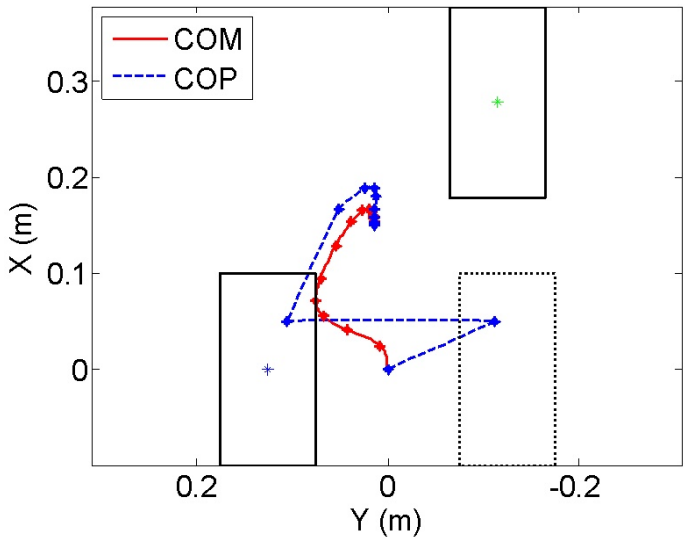


**Figure 4.4:** Example trajectory solved using the PR-MPC with $T = 0.2$, $N = 15$.

the form of Eq. (4.8) and the optimization problem becomes

$$\bar{\mathbf{U}}^* = \arg\min_{\bar{\mathbf{U}}} \bar{\mathbf{U}}^T \mathbf{H} \bar{\mathbf{U}} + \mathbf{f}^T \bar{\mathbf{U}} \tag{4.35}$$

$$\text{s.t. } \mathbf{D}\bar{\mathbf{U}} \leq \mathbf{d} \tag{4.36}$$

which is a standard form and can be solved by many existing constrained QP solvers. In this thesis, an active set dual method is used [34] using open-source C++ software [30]. $\bar{\mathbf{U}}^*$ can then be substituted back into Eq. (4.1) to obtain the COM trajectory. The $N$ points are connected using quintic splines to give continuous position, velocity and acceleration values along the trajectory. An example solution to this problem is shown in Figure 4.4 which resembles the expected behavior illustrated in Figure 4.2.

### 4.3.4 Computational Optimizations for Real-time Performance

PR-MPC is meant to be applied as a realtime controller. There are several speed optimizations that can be used to bring the solution time down to the sub-10ms time range. The easiest and most generally-applicable approach is to pre-compute parts of the problem.

**Cholesky Decomposition**: For many QP problems, the first step in the solution process is a Cholesky decomposition of the $\mathbf{H}$ matrix. If this matrix is constant, as is usually the case for linear dynamics, the Cholesky decomposition can be pre-computed off-line.

**Factoring Out $\mathbf{X}_t$**: The $\mathbf{f}$ vector in Eq. (4.7) is a function of the initial state, $\mathbf{X}_t$. Because the initial state is computed online, it can be factored out such that

$$\mathbf{f} = \mathbf{f}_{\mathbf{X}}\mathbf{X}_t$$

where

$$\mathbf{f}_{\mathbf{X}}^T = \bar{\mathbf{A}}^T\bar{\mathbf{Q}}\bar{\mathbf{B}}$$

is pre-computed. In most cases this can also be applied to the constraint matrices.

**Discretized Phase**: The dynamics change according to the time in the phase, because the step timings indicate when each foot is in contact with the ground. Solving exactly would mean pre-computing the Cholesky decomposition would not be possible. Instead, a separate $\mathbf{H}$ matrix is pre-computed for discrete times throughout a single phase and the Cholesky decomposition and factoring described above are performed for each. During execution, the current time in the phase is used to look up the closest pre-computed matrices.

**Other Optimizations**: Several other optimizations have been considered [22][23]. Choice of algorithm can be important for providing desired features such as the ability to warm-start (initialize with the previous solution), keep track of active constraints, and ensure that

intermediate iterations provide feasible solutions. These were not considered in this thesis; the problem was resolved from scratch each time.

### 4.3.5 Multi-Step Lookahead

It can be very beneficial to perform multiple footstep lookahead for step recovery and walking control. Table 4.1 shows the potential benefit of multiple footstep lookahead and re-planning, comparing a theoretical model to a simulated robot. For a simulated robot, which does not necessarily have the same dynamics as the real robot, optimizing over multiple steps into the future allows for recovery from much larger disturbances. For the model, the maximum push sizes are calculated using the bounds derived in Section 3.3. For each strategy, the model predicts a slightly higher maximum push than the simulation is able to achieve. Likewise, as the maximum push predicted by the model increases, the maximum push in the simulation also increases.

### 4.3.6 Re-Optimizing Online

Model predictive control is performed continuously online. The rate at which it is replanned is a tunable parameter. A simple method is to re-plan only at each touchdown. However, this could be done more often to correct for deviations from the planned trajectory due to additional disturbances or large modeling error. Figure 4.5 shows the size of push a simulated robot can recover from when planning at each touchdown vs. continuously. Figure 4.6 shows the added benefit of being able to adjust the desired footstep location if the robot receives a push mid-swing. In this simulation, the foot was hitting the ground earlier than predicted, but was still able to remain stable.

| Strategy | Model [1] (Ns) | Simulation [1] (Ns) |
|---|---|---|
| Ankle | 22 [2] | 18 |
| Hip | 31 [3] | 27 |
| 1-step | 39 [4] | 29 |
| 2-step | 42 [4] | 42 |

**Table 4.1:** [1] The model and simulation both have a total mass of $72kg$ and COM height of $1.0m$. All pushes in simulation are forward at the middle of the torso. [2]Foot size is $0.1m$ from mid-foot to the toe, [3]$\tau_{\max} = 40Nm$, $I = 4.0$, $\theta_{\max} = 0.5rad$, [4]Computed by running PR-MPC with increasing initial velocity until the system goes unstable with shift time $= 0.1s$, step time $= 0.4s$, max stance width $= 0.5m$.

### 4.3.7 Capture Point-Based Objective

The objective function in Eq. (4.17) can be modified in any number of ways. One possible modification is to make it correspond to the capture point [97] and re-write the objective to put the capture point over the middle of the base of support,

$$J = \frac{w_1}{2}\left|\left|\mathbf{CAP}_X^{\text{goal}} - \mathbf{CAP}_X\right|\right|^2 + \frac{w_2}{2}||\mathbf{U}_X||^2 + \frac{w_3}{2}\left|\left|p_X^{\text{ref}} - p_X\right|\right|^2 + \frac{w_1}{2}\left|\left|\mathbf{CAP}_Y^{\text{goal}} - \mathbf{CAP}_Y\right|\right|^2 + \frac{w_2}{2}||\mathbf{U}_Y||^2 + \frac{w_3}{2}\left|\left|p_Y^{\text{ref}} - p_Y\right|\right|^2 \tag{4.37}$$

where

$$\mathbf{CAP}_X = \mathbf{C}_X + \sqrt{\frac{L}{g}}\dot{\mathbf{C}}_X \tag{4.38}$$

is the capture point of the trajectory and

$$\mathbf{CAP}_X^{\text{goal}} = \mathbf{O}_0\frac{1}{2}\left(p_X^0 + p_X^1\right) + \mathbf{O}_1 p_X^1 + \mathbf{O}_2\frac{1}{2}\left(p_X^1 + p_X^2\right) \tag{4.39}$$

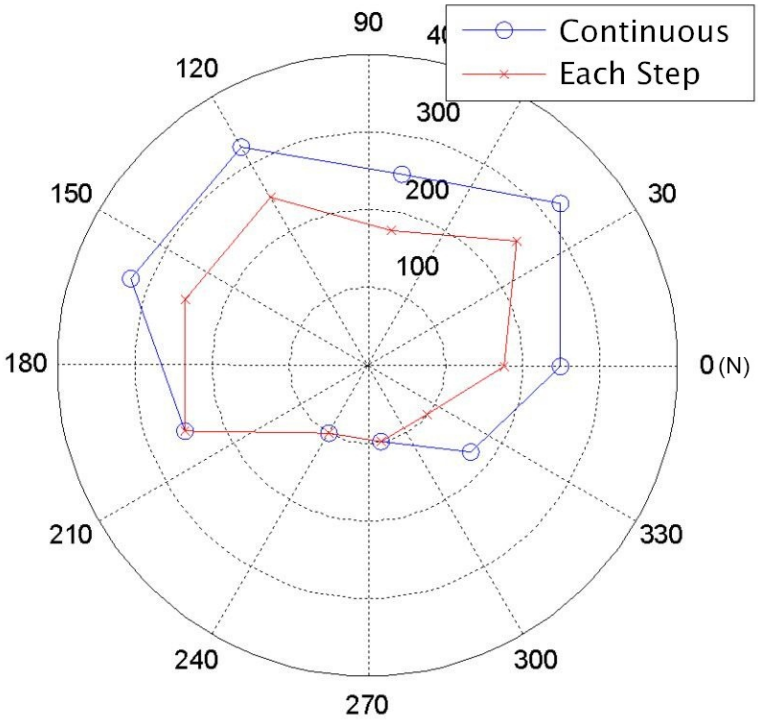


**Figure 4.5:** The maximum size of push (Newtons over a 0.1s period) for a simulated 72kg robot with and without continuous re-optimization.

is the capture point goal which corresponds to the middle of the base of support, using the notation from Section 4.3.1. This new goal causes the robot to move with a larger side-to-side amplitude and the capture point stays over the stance foot longer, as shown in Figure 4.7.

This new objective can also be used for walking by redefining $\mathbf{CAP}_X^{\text{goal}}$ from Eq. (4.39) to instead put the desired capture point in front of the midpoint of the base of support such that,

$$\mathbf{CAP}_X^{\text{goal}} = \mathbf{O}_0 \frac{1}{2}\left(p_X^0 + p_X^1 + \sqrt{\frac{L}{g}}\dot{\mathbf{C}}_X^{\text{des}}\right) + \mathbf{O}_1\left(p_X^1 + \sqrt{\frac{L}{g}}\dot{\mathbf{C}}_X^{\text{des}}\right) + \ldots \tag{4.40}$$

where $\dot{\mathbf{C}}_X^{\text{des}}$ is the desired walking speed. The "..." are used in this case because walking

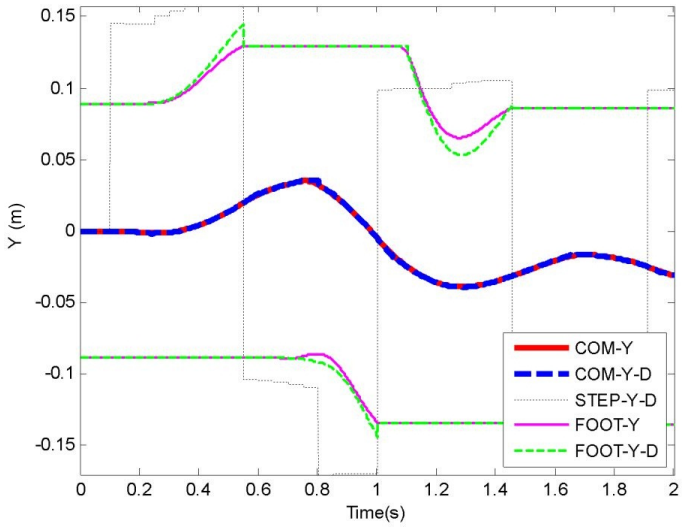


**Figure 4.6:** Full body simulation using PR-MPC with a high re-optimization rate to adjust the desired footstep location mid-swing. At $0.8s$ the robot is pushed to the side and the desired footstep location (FOOT-Y-D) output by the MPC is correspondingly moved further to the side.

usually requires optimization of more than one step into the future. It is straightforward to handle more steps.

## 4.4 Multi-Strategy PR-MPC

The linear dynamics of the LIPM allow for some analytic insight into the behavior of the system. In particular, stability margins can describe the states from which the system can recover without stepping [112], as shown in Figure 3.4. Until now, these analytic bounds have been used to determine when to step, then switching to a two-step lookahead version of PR-MPC. However, these analytic bounds can be difficult to determine for arbitrary footstep

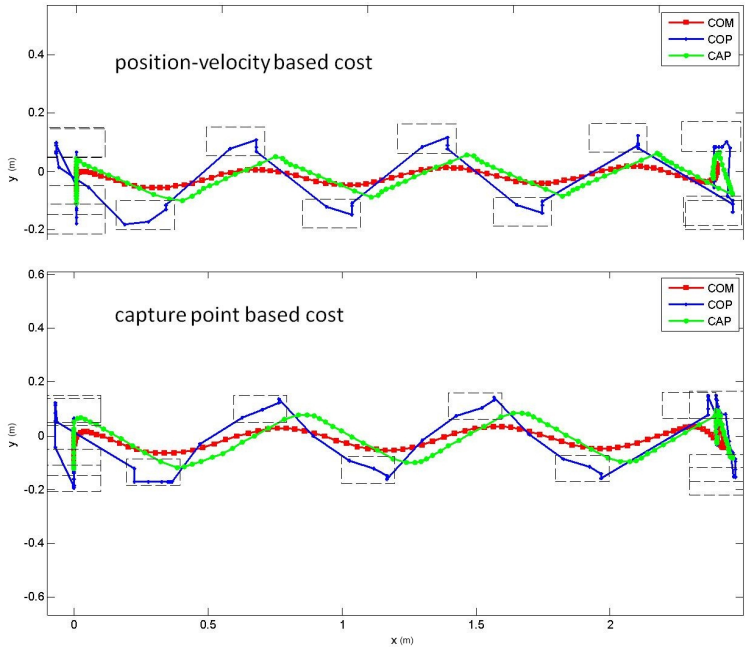


**Figure 4.7:** 1.0 m/s walking patterns automatically generated by PR-MPC. On the top is a walking pattern designed using a position-velocity based cost function and on the bottom is a pattern designed using a capture point based cost function. The bottom pattern results in the capture point moving closer to the foot.

locations and give little insight beyond deciding if a step is needed.

For this reason, a multiple strategy extension to PR-MPC can be designed, called Multi-Strategy PR-MPC (MSPR-MPC). By simultaneously optimizing multiple sets of constraints corresponding to different behaviors, the optimal strategy can be determined for arbitrary situations. In this thesis, we will consider the following strategies:

1. No Step (NS)
2. One Step - Left Foot First (LS1)
3. One Step - Right Foot First (RS1)

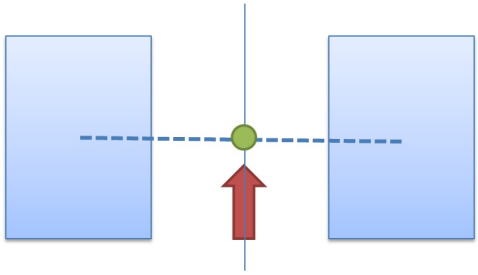

**Figure 4.8:** Test configuration with the COM centered between two parallel feet.

4. Two Steps - Left Foot First (LS2)

5. Two Steps - Right Foot First (RS2)

All strategies share a common objective function with the goal of bringing the COM to rest over the midpoint between the two feet. However the structure of the constraints for each strategy is different.

First we'd like to see how the MSPR-MPC relates to the analytic strategy policy shown in Figure 3.4. To do this, the configuration shown in Figure 4.8 with even feet is analyzed. The feet are $20cm$ in length, $10cm$ in width and $25cm$ apart. The optimal control strategy is determined for each initial condition in a grid corresponding to the position and velocity of the COM in the sagittal plane only. Figure 4.9 shows the result compared to the stability region from Figure 3.4. As intuitively expected, MSPR-MPC is slightly more conservative in deciding to take a step as opposed to the analytic region which was derived based on the theoretical limit. The figure also shows that taking a single step is optimal only for a small region of state space while taking two theoretically allows recovery from very large pushes.

Now consider the configuration in Figure 4.10 with uneven feet with a $5cm$ skew between them. Again the initial conditions are tested in a grid of COM positions and velocities in the sagittal plane only. Figure 4.11 shows results similar to Figure 4.9 but also demonstrates

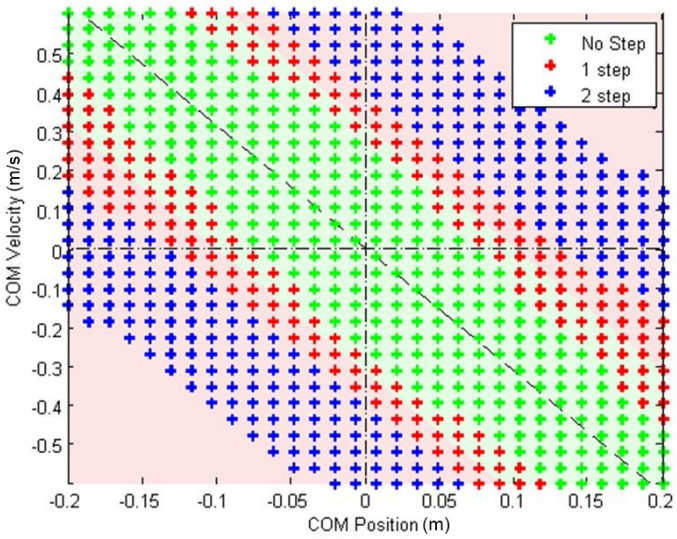


**Figure 4.9:** Optimal strategies plotted on top of the analytic stability region for two parallel feet. The one-step region overlaps the boundary which defines the limits of a pure ankle strategy.

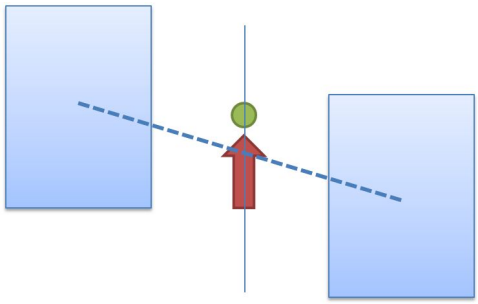

**Figure 4.10:** Test configuration with uneven feet. In this configuration it is predicted that there should be an asymmetry in the optimal strategies.

the ability of the controller to determine which foot should be the first stance foot.

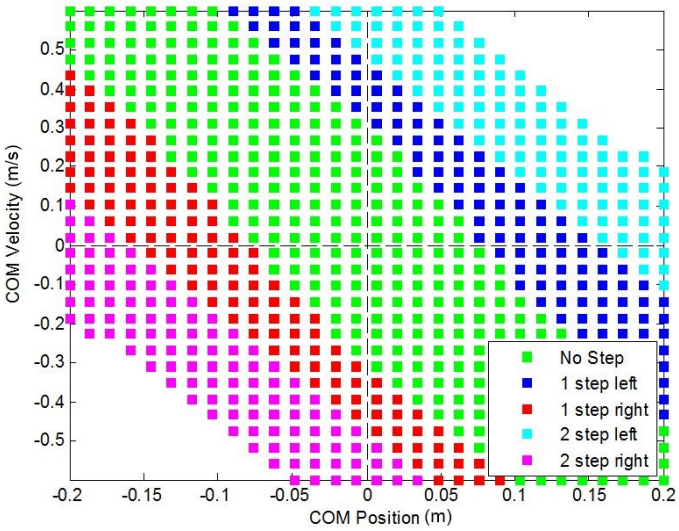


**Figure 4.11:** Optimal strategies for uneven feet. Generally, when pushed forward, MSPR-MPC determines that stepping with the right foot first is optimal. Likewise, when pushed backward, stepping with the left foot become optimal.

## 4.5 Perturbed PR-MPC

This section describes how to perform PR-MPC in the presence of a known disturbance. It is assumed that this disturbance is learned through experimentation using an algorithm such as Trajectory Learning [7]. Modeling error can be a significant source of error for any model-based controller, such as PR-MPC. This is especially true when the model is greatly simplified, such as when using the Linear Inverted Pendulum Model to approximate a humanoid robot. In the presence of modeling error, the expected dynamics will not match the observed dynamics. However, for motions that are repeated often, such as stepping or walking, it can be possible to learn a correction that compensates for the modeling error. Trajectory Learning is an approach wherein appropriate feedforward signals are learned

through repeated trials on real hardware. Learning is accomplished by recording the feedback signal in the controller during a training trial and adding it to the feedforward signal for future trials. This approach works well for systems with full control authority. However, a biped system is more complicated given the constrained and hybrid nature of the dynamics. For this reason, Perturbed PR-MPC is used. Put simply, the MPC optimizes the trajectory given the learned perturbation trajectory which acts as time-varying disturbance on the dynamics.

The linear dynamics can be written in the compact form,

$$\begin{pmatrix} \mathbf{X}_{i+1} \\ \mathbf{U}_{\text{pert}}(\phi) \end{pmatrix} = \begin{bmatrix} \mathbf{A} & \alpha\mathbf{B}_{\text{err}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{pmatrix} \mathbf{X}_i \\ \mathbf{U}_{\text{pert}}(\phi) \end{pmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{u}_i \end{pmatrix} \tag{4.41}$$

where $\mathbf{U}_{\text{pert}}(\phi)$ is the learned perturbation trajectory and $\mathbf{B}_{\text{err}}$ determines how the disturbance trajectory affects the dynamics. The parameter $\alpha$ can be used to vary the effect of the perturbation trajectory between trials. This trajectory is indexed by a phase variable $\phi$ to account for cyclic trajectories such as during walking. The number of control variables does not change so the optimal control problem does not increase in complexity.

The basics of Perturbed PR-MPC are demonstrated with two simple examples using the dynamics from Section 3.2.2 where the perturbation trajectory represents an external force on the COM. First, a fictitious sideways sinusoid force trajectory is given to the MPC with the goal of standing in place. The resulting optimized trajectory is shown in Figure 4.12 which, as expected, exhibits a sinusoidal COP trajectory which cancels the disturbance described by the perturbation trajectory. Second, a constant sideways force trajectory is passed to the MPC which takes a single step. The result is shown in Figure 4.13 which steps out to the side when its default behavior would be to step in place.

In this version of Perturbed PR-MPC, the perturbation trajectory is constant and learned

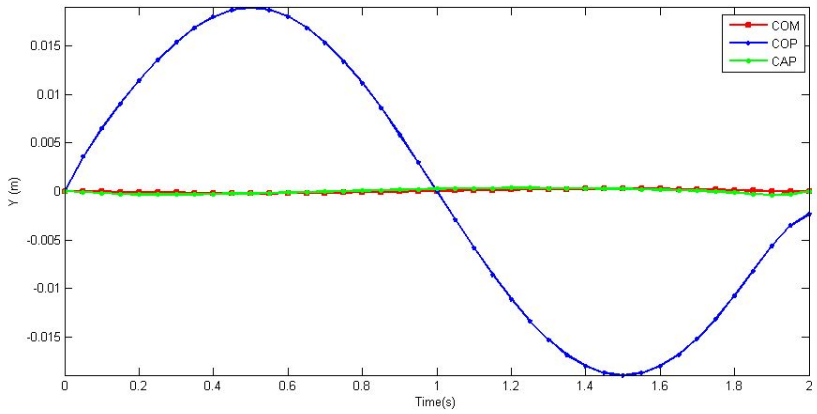


**Figure 4.12:** Optimized trajectory given an assumed sinusoidal perturbation.

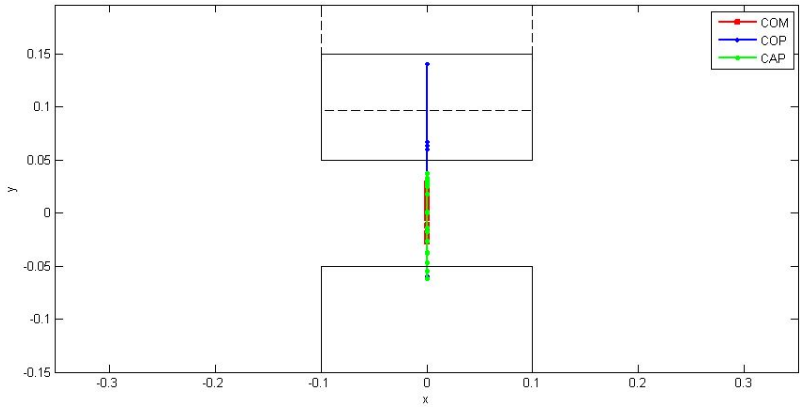


**Figure 4.13:** Given an assumed constant sideways disturbance, the default behavior is to step out to the side as opposed to stepping in place.

offline. A simple way to learn this trajectory is to perform multiple trials and average the error. Online learning could also be implemented by giving additional control authority to the MPC to modify the perturbation trajectory online.

### 4.5.1 Code Available

A copy of the MPC code used to solve these problems is available at `http://www.cs.cmu.edu/~cga/bstephens/mpcsolver.zip`. The code is written in C++, runs in Visual Studio 2010 and requires Eigen [1] for matrix math. There are a few Matlab files available for plotting outputs. The code is to be treated as untested research code and no support will be offered.

## 4.6 Results

The PR-MPC algorithm has been implemented on the Sarcos humanoid robot to create several behaviors that involve balance recovery by stepping. First, the problem of recovering from a large push while standing is presented. Then the problem of stepping while walking in place is demonstrated. Finally dynamic walking at variable speeds is shown.

### 4.6.1 Step Recovery

Step recovery experiments have been performed on the Sarcos humanoid robot. The robot was pushed from behind with a force-measuring device to sense the magnitude of the push. Figure 4.14 shows COM position and velocity and COP trajectories of a single push compared to the plan generated by PR-MPC. Foot swing trajectories are computed using minimum jerk splines ending at the desired footstep location.

Many push recovery experiments have been performed. Figure 4.15 shows 10 trials with forward push impulses ranging from 18-24Ns. For these 10 trials, half exhibit short steps of 10-15cm while the other half exhibit longer steps of 20-25cm. This apparent change in stepping strategy can be explained by Figure 4.16 which shows the forward step distance

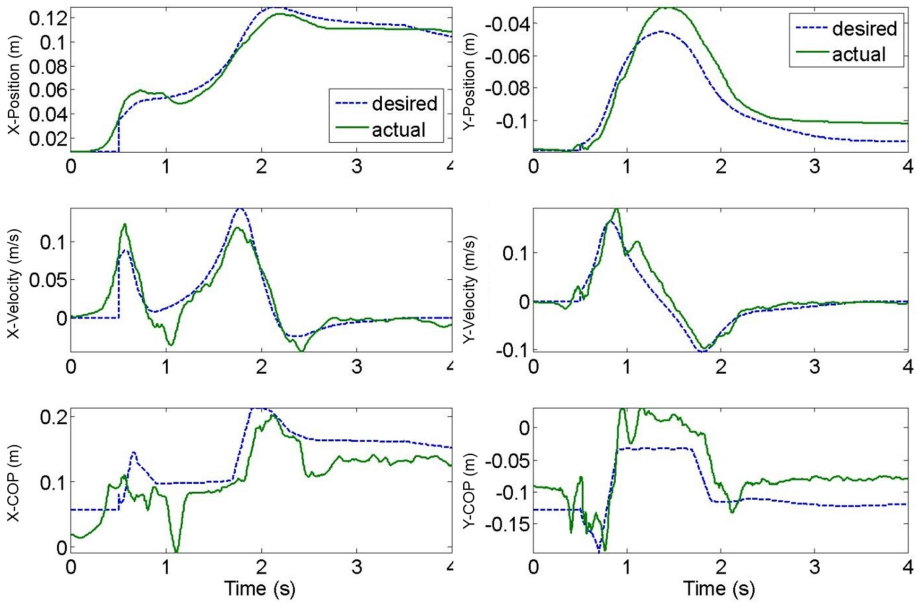


**Figure 4.14:** X and Y trajectories of the COM and COP from a push recovery experiment in which the robot was pushed from behind and takes a single step with the right foot.

predicted by PR-MPC for a given initial X velocity and constant external force. As the initial velocity increases, there is a change in strategy requiring drastically larger steps. This effect is largely due to the activation of additional constraints on the COP during the step.

A video of another push recovery experiment that shows the robot recovering from multiple pushes and coming to a stop can be found at `http://www.cs.cmu.edu/~cga/bstephens/videos/steprecovery.wmv`

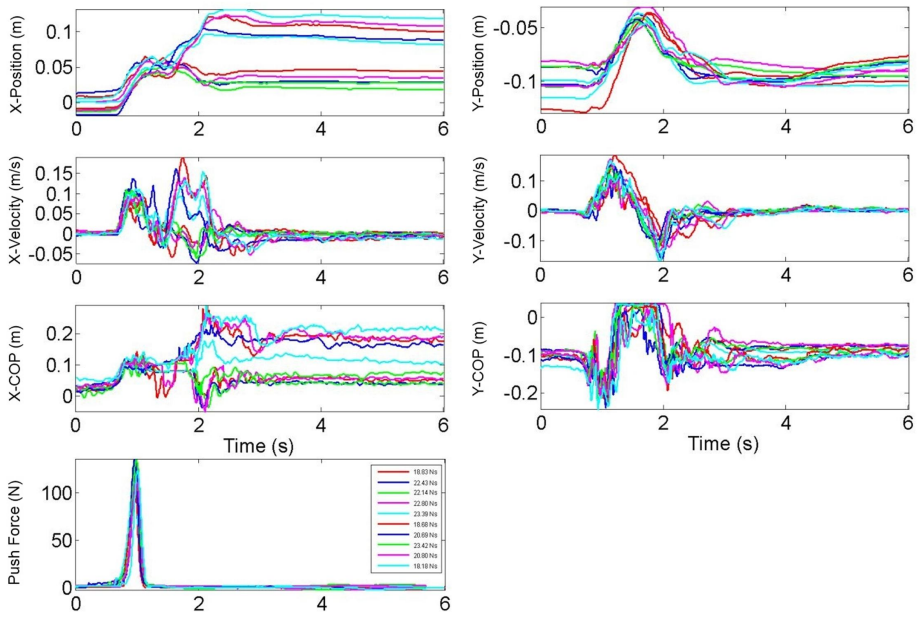


**Figure 4.15:** 10 X and Y trajectories of COM including measured push forces from experiment. Impulses were calculated by integrating the push force curve.

### 4.6.2 Walking in Place

Walking in place experiments have been performed on the Sarcos humanoid robot using PR-MPC. By continuously asking the robot to perform a two-step push recovery, even though it may not be pushed, walking in place can be achieved, as shown in Figure 4.17. Videos of this experiment showing the robot continuously walking in place and being pushed can be found at `http://www.cs.cmu.edu/~cga/bstephens/videos/steppinginplace.wmv` and `http://www.cs.cmu.edu/~cga/bstephens/videos/steppinginplace2.wmv`. Details related to the full-body controller will be described in Chapter 5.

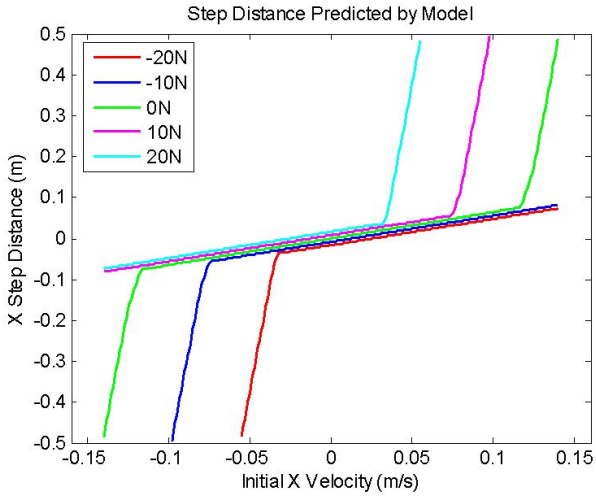


**Figure 4.16:** Theoretical relationship between initial forward velocity, constant external force and forward step distance using PR-MPC reveals an apparent change in strategy.

### 4.6.3 Forward Walking

Forward walking has been implemented on the Sarcos humanoid robot using PR-MPC with the capture-point-based objective function from Section 4.3.7. In this experiment, the desired velocity was slowly ramped from zero to 0.4m/s and back to zero. Figure 4.18 shows the resulting velocities and Figure 4.19 shows the COM, COP and capture point trajectories along with the feet.

Videos demonstrating this walking control are viewable at `http://www.cs.cmu.edu/~cga/bstephens/videos/walkingspeeds.wmv`

**Figure 4.17:** Screen capture from a video of the robot walking in place and being pushed.

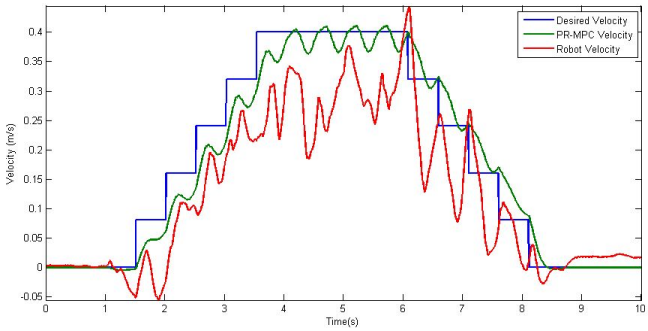


**Figure 4.18:** Varying desired walking velocity on the Sarcos humanoid.

## 4.7 Discussion

There are several limits of the current algorithm. As presented, it does not directly address foot yaw rotations or cross-stepping (stepping to the side by crossing legs). In order to

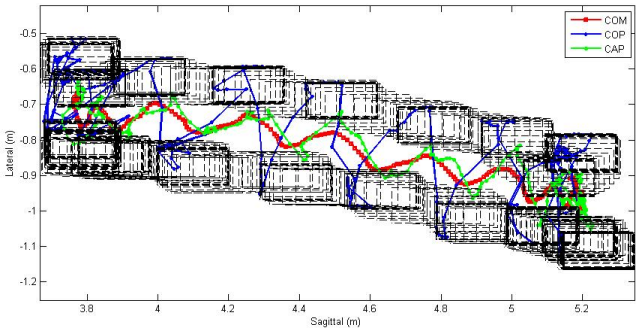


**Figure 4.19:** COM, COP and CAP trajectories during a walking experiment on the Sarcos humanoid robot.

use the efficient quadratic programming solution of PR-MPC, all of the constraints must be linear. The true double support constraints after touchdown are nonlinear, and this thesis only presents an approximation. Nonlinear constraints can be solved using sequential quadratic programming (SQP).

The computation speed of the algorithm has not been considered. The PR-MPC QP problem can be solved very efficiently by readily available algorithms and scales primarily with the number of lookahead timesteps, $N$. $N$ should be as small as possible, but the lookahead time must also be large enough. This optimization is currently solved with $N = 30$ with a timestep of $0.05s$ is used resulting in a lookahead of $1.5s$ and solution times under 5 milliseconds on a 2GHz CPU.

The controller presented in this chapter is independent of the exact model used. In all of the examples, LIPM dynamics are used. However, this model could easily be replaced by another model, such as a linear flywheel model that accounts for angular momentum [97][112]. Using this model in PR-MPC requires only minor changes. More complex nonlinear models could also be used but will generally require more time to solve.

## 4.8 Conclusion

This chapter presented an optimal control approach to reactive balance and push recovery control using simple models. Push Recovery Model Predictive Control (PR-MPC) is a trajectory optimization that considers future actions and constraints that is run online to reactively control balance. It uses the current COM state of the robot to determine actions such as desired COP and next footstep location.

**What's Next?** Chapter 5 will describe how controls developed for simple models (e.g. PR-MPC) can be mapped to a full-body humanoid robot. Chapter 6 will describe state estimation using simple models which serves as the input to a controller like those presented in this chapter.

# Chapter 5

# Full-Body Torque Control

This chapter is concerned with the problem of mapping controls from simple models to full-body torques for a humanoid robot. It will be shown that COM dynamics can play a major role in this process. Results performing a variety of tasks in simulation and hardware will be presented.

## 5.1 Introduction

While humanoid robots are very complex systems, the dynamics that govern balance are often described using simple models of the center of mass (COM) [112]. It has been shown through dynamic simulation that humanoid balance depends critically on controlling the linear and angular momentum of the system [76], quantities that can be directly controlled by contact forces. These insights suggest that balance is a fundamentally low-dimensional problem that can be solved by contact force control. This idea is the inspiration for the controller presented in this chapter, which is used to control full-body balance and other tasks on compliant force-controlled humanoid robots.

Given a robot with stiff joint position control and a known environment, the most common approach to balance is to generate a stable trajectory of the COM and then track it using inverse kinematics (IK) [122][141]. These trajectories can be modified online to produce whole body balance in the presence of small disturbances [121]. For environments with small uncertainty or small disturbances, the inverse kinematics can be modified to directly control the contact forces using force feedback [26]. Position-based controllers generally exhibit high impedance, and the speed at which they will comply to an unknown force is limited. Robots with low impedance joints can comply faster, but stable balance can be more difficult to control.

For compliant robots, there are a number of ways that contact force control can be achieved. Virtual model control (VMC), developed by Pratt, *et al.* , is a simple method that only uses a kinematic model. Desired contact forces are converted into joint torques assuming static loading using a Jacobian-transpose mapping [100]. Hyon, *et al.* , showed that under quasistatic assumptions and proper damping of internal motions the desired forces can be achieved [50]. If given the full constrained rigid-body dynamics model, desired joint accelerations can be converted into joint torques using inverse dynamics for improved tracking performance, as shown by Mistry, *et al.* , [82]. Other methods, such as the one by Abe, *et al.* , [2] reformulate the controller as a multi-objective optimization. For most objectives, these optimizations can be formulated as an efficient QP problems, as shown by de Lasa, *et al.* , [20].

This chapter presents another method, called Dynamic Balance Force Control (DBFC), which is summarized in Figure 5.1. Like [82], the full dynamic model is used and no quasistatic assumptions are made. However, like [100] and [50], the inputs are desired contact forces. Contact forces are computed independent of the full robot model based on a simple

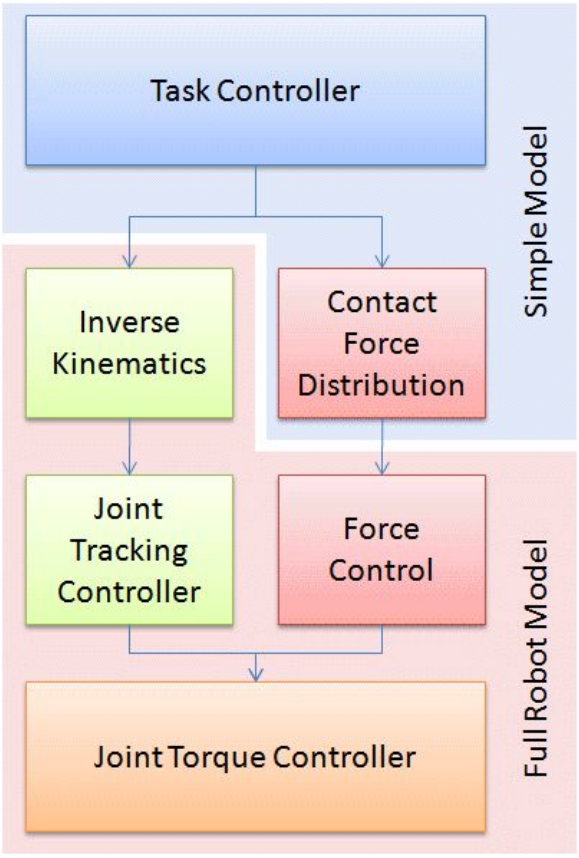


**Figure 5.1:** Block diagram of full control algorithm including DBFC.

COM dynamics model and external forces, such as the model described in Section 3.2.2. By solving for the contact forces first, the approach becomes similar to multi-objective control with a high priority, such as [20], on the desired COM dynamics. DBFC can be modified to compensate for non-contact forces using VMC-like controls. This modification, called DBFC-VMC, can be used to perform generic tasks such as posture control and manipulation. The outputs of the DBFC(-VMC) are full-body joint torques. **The main contributions of this chapter are the derivation of DBFC and application of controllers designed using simplified models to perform full-body task control**.

This chapter is organized as follows. In Section 5.2, desired contact forces are calculated

from a simplified model based on COM dynamics. In Section 5.4, it is shown how full-body joint torques can be calculated using DBFC. To perform more general tasks, DBFC-VMC is presented in Section 5.5. Results from experiments on a Sarcos humanoid robot are given in Section 5.6. Several examples showing a wide range of tasks are presented.

## 5.2 COM Inverse Dynamics

The COM dynamics of a general biped system with two feet in contact with the ground are instantaneously represented by a system of linear equations which sum the forces and torques on the COM. If $C = (x, y, z)^T$ is the location of the COM, then the dynamics can be written as

$$\begin{bmatrix} \mathbf{D_1} \\ \mathbf{D_2} \end{bmatrix} \begin{pmatrix} \\ \mathbf{F} \\ \\ \end{pmatrix} = \begin{pmatrix} m\ddot{\mathbf{C}} + \mathbf{F}_g \\ \\ \dot{\mathbf{H}} \end{pmatrix} \tag{5.1}$$

just as in Eq. (3.1).

These equations can be used to solve for a valid set of desired contact forces, $\mathbf{F}^*$, by solving a constrained optimization. If Eq. (5.1) is abbreviated as

$$\mathbf{KF} = \mathbf{u} \tag{5.2}$$

then the desired forces, $\mathbf{F}^*$, can be found by solving the quadratic programming problem,

$$\mathbf{F}^* = \arg\min_{\mathbf{F}} \mathbf{F}^T\mathbf{WF} \tag{5.3}$$

$$\text{s.t. } \mathbf{KF} = \mathbf{u} \tag{5.4}$$

$$\mathbf{BF} \leq \mathbf{c} \tag{5.5}$$

where $BF \leq c$ represents linear inequality constraints due to the support polygon and friction limits. $W = \text{diag}\,(w_i)$ can be used to weight certain forces more than others, for example to

penalize large horizontal forces. In order to keep the center of pressure under the feet, the constraints,

$$d_Y^- \leq \quad \tfrac{M_X}{F_Z} \quad \leq d_Y^+ \tag{5.6}$$

$$d_X^- \leq \; -\tfrac{M_Y}{F_Z} \; \leq d_X^+ \tag{5.7}$$

must be met for each foot, where $M_X$ and $M_Y$ are the total moments generated at the ground contact and $d_X^{\pm}$ and $d_Y^{\pm}$ represent the dimensions of the feet in the $x$ and $y$ directions, respectively. These equations can be re-written as linear constraints on the forces and torques.

Friction constraints can also be considered. However, the general form of these constraints is nonlinear in the forces,

$$\left| \frac{\sqrt{F_X^2 + F_Y^2}}{F_Z} \right| \leq \mu \tag{5.8}$$

A solution is to write a simple conservative approximation,

$$\left| \frac{F_X}{F_Z} \right| \leq \frac{\mu}{\sqrt{2}} \tag{5.9}$$

$$\left| \frac{F_Y}{F_Z} \right| \leq \frac{\mu}{\sqrt{2}} \tag{5.10}$$

which can be written as linear constraints. Higher order approximations to these constraints can also be used.

During single support, this optimization is not required because there are 6 equations and 6 unknowns, meaning the forces on the stance foot are completely determined by the desired accelerations.

It is possible that, given the constraints, there will be no valid solution to the optimization in Eq. (5.3). If that is the case, it is possible to re-write the quadratic optimization to solve

a constrained least squares problem,

$$\mathbf{F}^* = \arg\min_{\mathbf{F}} \left(\mathbf{KF} - \mathbf{u}\right)^T \left(\mathbf{KF} - u\right) + \mathbf{F}^T\mathbf{WF} \tag{5.11}$$

$$\text{s.t. } \mathbf{BF} \leq \mathbf{c} \tag{5.12}$$

which will always find a solution, but may not be able to achieve the desired accelerations.

Example code that performs this optimization is found at `http://www.cs.cmu.edu/~cga/bstephens/libmquad.zip`.

## 5.3 Biped Dynamics Equations

This section reviews the basics of biped dynamics. The equations of motion of a free-floating body can be written as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}\tau \tag{5.13}$$

where $\mathbf{q} \in \mathtt{R}^{n+6}$ is the configuration vector that includes $n$ joint angles and six root coordinates. $\mathbf{M}(\mathbf{q}) \in \mathtt{R}^{n+6\times n+6}$ is the inertia matrix, $\mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathtt{R}^{n+6}$ is a vector of nonlinear gravitational, centrifugal and coriolis terms, and $\mathbf{S} \in \mathtt{R}^{n+6\times n}$ maps the $n$ joint torques, $\tau$, to the appropriate rows of Eq. (5.13). Generally, these equations are written such that $\mathbf{S} = [\mathbf{0}, \mathbf{I}]^T$.

Constraints on the feet, such as contact with the ground, are represented by a set of equations

$$\mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} = \ddot{\mathbf{P}} \tag{5.14}$$

where $\mathbf{J}(\mathbf{q})$ is a Jacobian describing the positions and orientations of the centers of the feet and $\ddot{\mathbf{P}}$ is the accelerations of these features. For a static contact, $\ddot{\mathbf{P}} = \mathbf{0}$. Maintaining these

constraints requires applying forces to the system. In this case, Eq. (5.13) is re-written as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}\tau + \mathbf{J}^T(\mathbf{q})\mathbf{F} \tag{5.15}$$

where $\mathbf{F}$ represents the constraint forces. Eq. (5.14) and Eq. (5.15) form a coupled set of equations that can be written as a matrix equation that is linear in the accelerations, torques and forces,

$$\begin{bmatrix} \mathbf{M} & -\mathbf{S} & -\mathbf{J}^T \\ \mathbf{J} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \ddot{\mathbf{q}} \\ \tau \\ \mathbf{F} \end{pmatrix} = \begin{pmatrix} -\mathbf{N} \\ \dot{\mathbf{P}}_{\text{des}} - \dot{\mathbf{J}}\dot{\mathbf{q}} \end{pmatrix} \tag{5.16}$$

## 5.4 Dynamic Balance Force Control

The objective of DBFC is to determine the joint torques that will achieve some desired body motion. Rather than perfect joint trajectory control, this chapter is concerned with balance-related tasks, meaning that control of the motion of the COM and angular momentum is important. These quantities can be controlled by the contact forces. Presented below, DBFC is a model-based method for determining full-body joint torques based on desired COM motion and contact forces.

Eq. (5.16) can be augmented with the dynamics of the center of mass in Eq. (3.1), where the desired acceleration of the COM, $\ddot{\mathbf{C}}_{\text{des}}$, acceleration of the feet, $\ddot{\mathbf{P}}_{\text{des}}$, and change of angular momentum, $\dot{\mathbf{H}}_{\text{des}}$, are specified. This imposes virtual constraints such that all feasible solutions will have these desireds. Instantaneously, the result is a set of linear

equations,

$$\begin{bmatrix} \mathbf{M} & -\mathbf{S} & -\mathbf{J}^T \\ \mathbf{J} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_1 \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_2 \end{bmatrix} \begin{pmatrix} \ddot{\mathbf{q}} \\ \tau \\ \mathbf{F} \end{pmatrix} = \begin{pmatrix} -\mathbf{N} \\ \ddot{\mathbf{P}}_{\text{des}} - \dot{\mathbf{J}}\dot{\mathbf{q}} \\ m\ddot{\mathbf{C}}_{\text{des}} + \mathbf{F}_g \\ \dot{\mathbf{H}}_{\text{des}} \end{pmatrix} \tag{5.17}$$

For the rest of this chapter, we assume $\dot{\mathbf{H}}_{\text{des}} = \mathbf{0}$ for simplicity. This linear system can be solved by constrained quadratic programming to determine joint torques, $\tau$, using the same methods described in the previous section.

However, if we solve the bottom two lines of Eq. (5.17) separately, using Eq. (5.11) or some other method, to get a valid set of contact forces, $\mathbf{F}^*$, that obey the contact constraints, Eq. (5.17) can be simplified to

$$\begin{bmatrix} \mathbf{M}(\mathbf{q}) & -\mathbf{S} \\ \mathbf{J} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \ddot{\mathbf{q}} \\ \tau_{\text{dbfc}} \end{pmatrix} = \begin{pmatrix} -\mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{J}^T\mathbf{F}^* \\ -\dot{\mathbf{J}}\dot{\mathbf{q}} \end{pmatrix} \tag{5.18}$$

where $\ddot{\mathbf{P}}_{\text{des}} = 0$ if both feet are on the ground. This system can be written as $\mathbf{Gz} = \mathbf{f}$ and solved by a weighted pseudo-inverse,

$$\mathbf{z}^* = \left(\mathbf{G}^T\mathbf{G} + \mathbf{W}\right)^{-1}\mathbf{G}^T\mathbf{f} \tag{5.19}$$

where $\mathbf{W}$ is used to regularize the solution. The solution of this equation, $\mathbf{z}^T = \left[\ddot{\mathbf{q}}^T, \tau_{\text{dbfc}}^T\right]$, contains the full-body joint torques, $\tau_{\text{dbfc}}$, that are used for control. This equation is solved at every timestep and is simpler to solve than the full constrained dynamics in Eq. (5.17). A simple $\mathbf{W}$ is a diagonal matrix with very small values that keeps the solution from being too large.

**Why not solve Eq. (5.17) directly?** It is possible to write a controller that solves Eq. (5.17) directly at every timestep [2]. This approach can be useful if obtaining an $\mathbf{F}^*$ is

difficult. It gives the solver more variables to tune in order to solve the equations. The result may be unpredictable. For example, it may choose to distribute the contact forces differently given the current kinematic configuration of the robot. It may be difficult to constrain the extra degrees of freedom and keep the solver from exploiting the robot model, which may have significant error. The DBFC eliminates the extra degrees of freedom by using intuitive knowledge of the task to determine the contact forces *a priori*. This knowledge can be exploited in several contexts, as shown in Section 5.6.

## 5.5 Task Control

In addition to balance control, the ability to perform other tasks, such as posture control and manipulation is useful. Such tasks can be integrated into DBFC by including virtual task forces, just as in Virtual Model Control. These forces are reflected both in the contact forces and the joint torques. In the method described here, the compensating contact forces are first calculated using COM dynamics and then the joint torques are calculated using DBFC.

First, task forces and torques that affect the COM dynamics are used to offset the contact forces,

$$\begin{bmatrix} \mathbf{D}_1 \\ \mathbf{D}_2 \end{bmatrix} \mathbf{F} = \begin{pmatrix} m\ddot{\mathbf{C}}_{\text{des}} + \mathbf{F}_g + \sum_i \mathbf{F}^i_{\text{task}} \\ \sum_i (\mathbf{P}^i - \mathbf{C}) \times \mathbf{F}^i_{\text{task}} + \mathbf{M}^i_{\text{task}} \end{pmatrix} \tag{5.20}$$

where task forces, $\mathbf{F}^i_{\text{task}}$, and torques, $\mathbf{M}^i_{\text{task}}$, are applied at some specific point, $\mathbf{P}^i$, on the body. The cross product is included for when a task force also applies a torque about the COM. Solving for the contact forces, $\mathbf{F}^*$, in this case will compensate for the task forces. Now the robot joint torques can be found by solving the DBFC problem with the task forces

included,

$$\mathbf{G}\begin{pmatrix} \ddot{\mathbf{q}} \\ \tau_{\text{dbfc}} \end{pmatrix} = \begin{pmatrix} -\mathbf{N}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{J}^T\mathbf{F}^* + \sum_i J^{iT}_{\text{task}} W^i_{\text{task}} \\ -\dot{\mathbf{J}}\dot{\mathbf{q}} \end{pmatrix} \tag{5.21}$$

where $J^i_{\text{task}}$ is the 6-dof Jacobian associated with the point where the $i$-th task forces are applied and

$$W^i_{\text{task}} = \begin{pmatrix} \mathbf{F}^i_{\text{task}} \\ M^i_{\text{task}} \end{pmatrix} \tag{5.22}$$

## 5.6 Results

Several results are presented below using DBFC on a humanoid robot and in simulation. The examples are meant to describe both the performance and implementation details of the controllers. The same rigid-body dynamics model is used in simulation and in control of the real robot. Even though both the model and controller are 3D, for clarity only sagittal plane data is presented for each example.

Standing balance control is demonstrated in experiments performed on a Sarcos Primus humanoid robot. Walking control on the robot was presented in Section 4.6.3, but in this chapter is presented in simulation only for clarity. The simulator assumes a simple spring-ground contact model. The feet are assumed to be point feet, but have the ability to apply torques to the ground when in contact. The simulation automatically limits these contact torques to ensure the COP is always under the virtual foot.

Videos of these examples are available at `http://www.cs.cmu.edu/~cga/bstephens/videos/iros2010.mpg`.

### 5.6.1 Example: Balance Control

For standing balance control, the objective is simply to regulate the position of the COM, which is achieved by a PD controller,

$$\ddot{\mathbf{C}}_{\text{des}} = -K_p \left(\mathbf{C}_{\text{des}} - C\right) - K_d \dot{C} \tag{5.23}$$

This desired acceleration can be converted into contact forces using the method in Section 5.2. Solving Eq. (5.18) gives the appropriate joint torques. In practice, small feedback torques, $\tau_{\text{fb}}$, are also added to bias the joint angles and velocities to desired angles and velocities,

$$\tau_{\text{fb}} = \mathbf{K}_{qp} \left(\mathbf{q}_{\text{des}} - \mathbf{q}\right) + \mathbf{K}_{qd} \left(\dot{\mathbf{q}}_{\text{des}} - \dot{\mathbf{q}}\right) \tag{5.24}$$

For a standing balance task where the robot is pushed from behind, the result of the DBFC controller is shown in Figure 5.2. This figure shows the total desired and measured $F_X$ and $M_Y$ forces, as well as the state of the robot COM after the push.

### 5.6.2 Example: Torso Posture Control

Maintaining torso posture can be a desirable goal during standing balance. Often this is handled by special consideration of the hip joint torque [139]. This example shows how torso posture control can be achieved via DBFC-VMC and implemented on a humanoid robot. The posture, or torso angle, can be corrected by a torque, $\mathbf{M}_{\text{torso}}$, to the torso, as shown in Figure 5.3. There is no change to the linear force on the COM. This torso torque can be written as a simple PD controller,

$$\mathbf{M}_{\text{torso}} = \mathbf{K}_{\theta p} \left(\theta^{\text{pos}}_{\text{des}} - \theta_{\text{pos}}\right) - \mathbf{K}_{\theta d} \dot{\theta}^{\text{pos}} \tag{5.25}$$

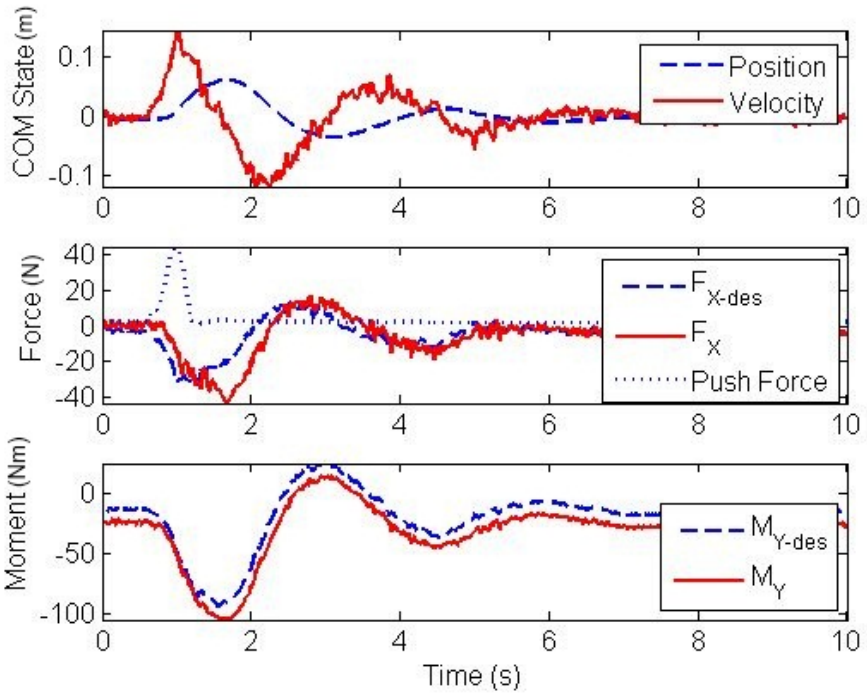


**Figure 5.2:** Example push recovery experiment using the DBFC approach in this chapter.

where $\theta^{\text{pos}}$ is the orientation of the torso. Using Eq. (5.20), $\mathbf{F}^*$ can be determined by solving

$$\begin{bmatrix} \mathbf{D}_1 \\ \mathbf{D}_2 \end{bmatrix} \mathbf{F} = \begin{pmatrix} m\ddot{\mathbf{C}}_{\text{des}} + \mathbf{F}_g \\ \mathbf{M}_{\text{torso}} \end{pmatrix} \tag{5.26}$$

and Eq. (5.21) can be used to determine the joint torques,

$$\mathbf{G} \begin{pmatrix} \ddot{\mathbf{q}} \\ \tau_{\text{dbfc}} \end{pmatrix} = \begin{pmatrix} -\mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{J}^T\mathbf{F}^* + \sum_i \mathbf{J}_{\text{torso}}^T \mathbf{M}_{\text{torso}} \\ -\dot{\mathbf{J}}\dot{\mathbf{q}} \end{pmatrix} \tag{5.27}$$

where $\mathbf{J}_{\text{torso}}$ is the Jacobian associated with the torso body to which the virtual torque is applied.

This controller was applied to the Sarcos Primus humanoid robot, which used an inertial measurement unit (IMU) attached to the hip to measure torso angle with respect to ground.

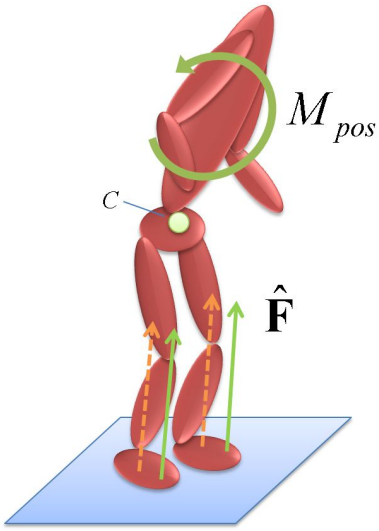


**Figure 5.3:** Posture control is achieved by a virtual torque on the torso. The original contact forces (dashed) are modified (solid) to compensate for this virtual torque.

Figure 5.4 shows the result of the balance controller when the robot is pushed from behind at the middle of the torso with and without posture control. Without posture control, the torso is underdamped and has a large deviation from upright. This is corrected by torso posture control with $\mathbf{K}_{\theta p} = 150$ and $\mathbf{K}_{\theta d} = 150$. The robot is also able to handle much larger pushes with this controller. Photos of the robot during this experiment are shown in Figure 5.5.

### 5.6.3 Example: Heavy Lifting

This task further demonstrates the capabilities of the DBFC-VMC controller. When lifting a heavy object as shown in Figure 5.6, DBFC-VMC can be used to generate the compensating joint torques. The VMC task controller can be defined to regulate the position of the object,

$$\mathbf{F}_{\text{lift}} = \mathbf{K}_{op}\left(\mathbf{P}_{\text{des}}^{\text{object}} - \mathbf{P}^{\text{object}}\right) \tag{5.28}$$

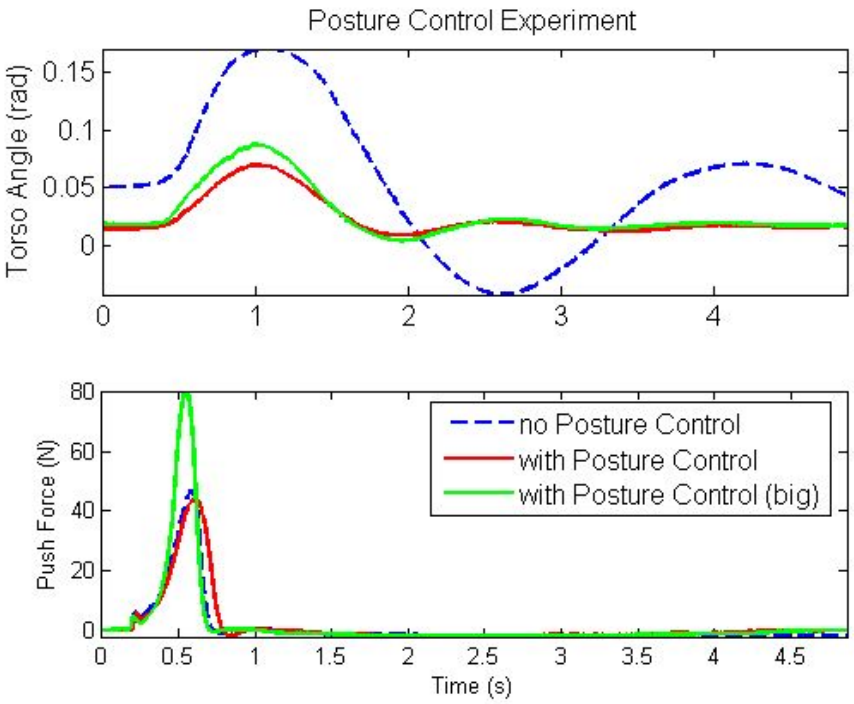


**Figure 5.4:** Torso angle with and without torso posture control. The force of the push was measured to compare the performance relative to push size.

Now Eq. (5.20) takes the form

$$\begin{bmatrix} \mathbf{D}_1 \\ \mathbf{D}_2 \end{bmatrix} \mathbf{F} = \begin{pmatrix} m\ddot{\mathbf{C}}_{\text{des}} + \mathbf{F}_g + \mathbf{F}_{\text{lift}} \\ \left(\mathbf{P}^{\text{object}} - \mathbf{C}\right) \times \mathbf{F}_{\text{lift}} \end{pmatrix} \tag{5.29}$$

The mass of the object is not needed for VMC control, but could be implemented in a feedforward manner if known, $\mathbf{F}^*_{\text{lift}} = \mathbf{F}_{\text{lift}} + m^{\text{object}} g$. In addition to the task force, a task torque is required because the gravitational force on the object applies a torque about the COM. Eq. (5.21) is again used to determine the joint torques,

$$\mathbf{G} \begin{pmatrix} \ddot{\mathbf{q}} \\ \tau_{\text{dbfc}} \end{pmatrix} = \begin{pmatrix} -\mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{J}^T \mathbf{F}^* + \sum_i \mathbf{J}^T_{\text{lift}} \mathbf{F}_{\text{lift}} \\ -\dot{\mathbf{J}}\dot{\mathbf{q}} \end{pmatrix} \tag{5.30}$$

where $\mathbf{J}_{\text{lift}}$ defines the point on the arms where the lift force is applied.

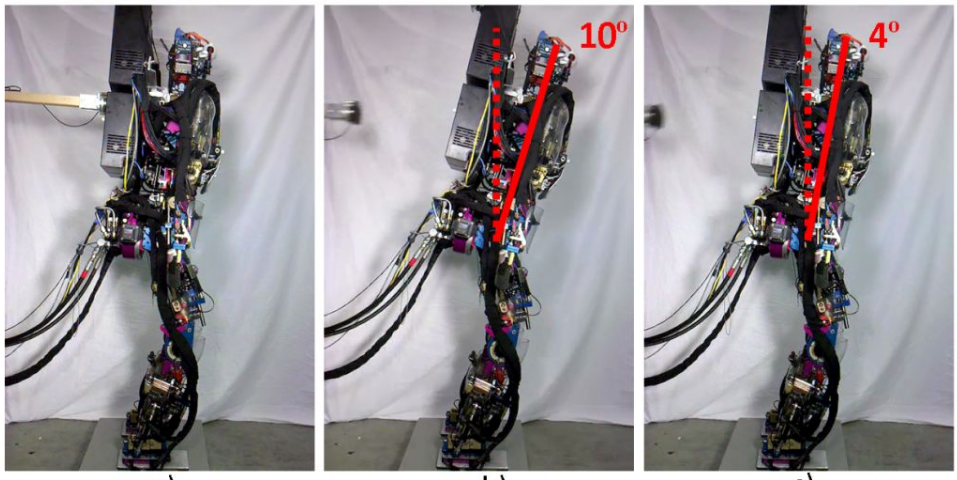


**Figure 5.5:** Comparing torso posture control responses. In a) the robot is pushed from behind to the middle of the torso with a force sensing stick. The effect of posture control can be seen in b) without using posture control and c) with posture control.

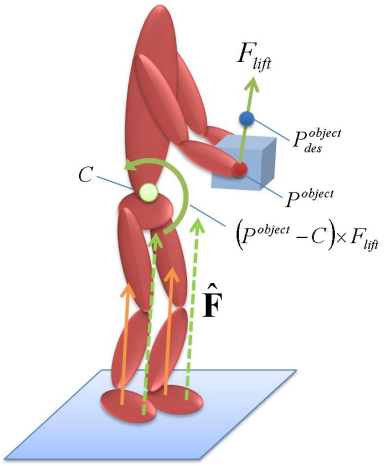


**Figure 5.6:** Lifting a heavy object requires both a task force and task torque to compensate for the gravitational force on the object.

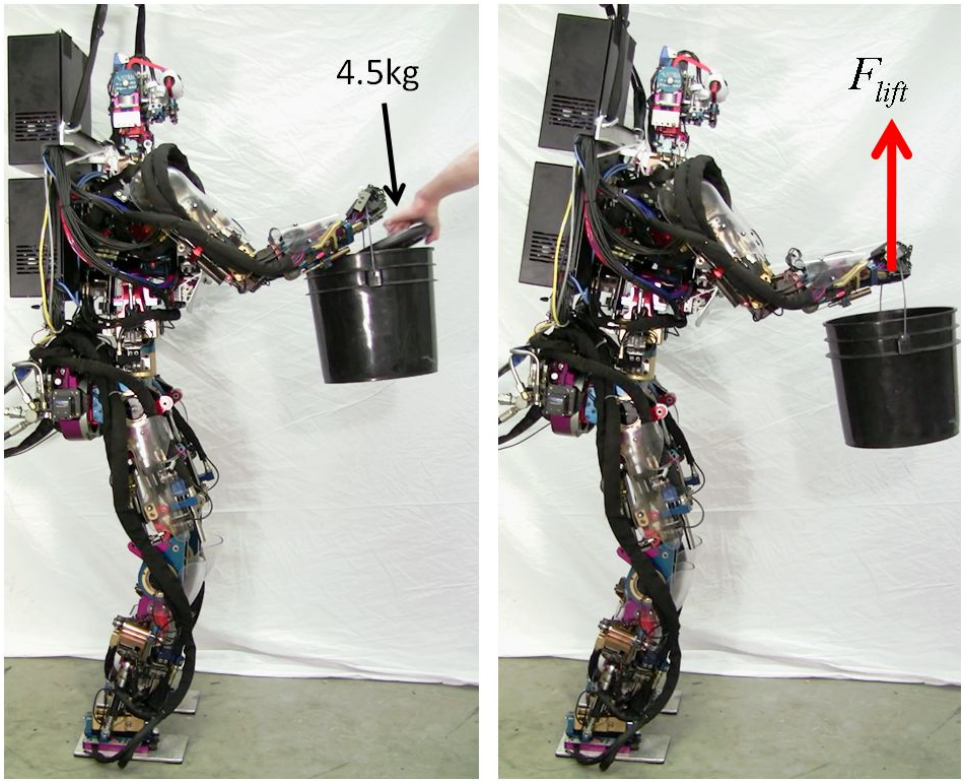


**Figure 5.7:** Experimental setup for evaluating the lifting task. A 4.5kg mass is dropped into the bucket. The robot applies a force, $\mathbf{F}_{\text{lift}}$, to the bucket to control the height of the bucket.

Figure 5.7 shows the experimental setup used to test this controller. A bucket is attached to the hand so only a vertical force is required. A 4.5kg mass is dropped into the bucket from a height of approximately 22cm. Without the VMC, the robot falls forward. However, when VMC is enabled with $K_p > 0$, the robot maintains its balance. A comparison of the performance for different $\mathbf{K}_{op}$ values is shown in Figure 5.8. As expected, $\mathbf{F}_{\text{lift}}$ increases with $\mathbf{K}_{op}$ and the steady state error is reduced. To eliminate the steady-state offset, an integrator could be added to Eq. (5.28).

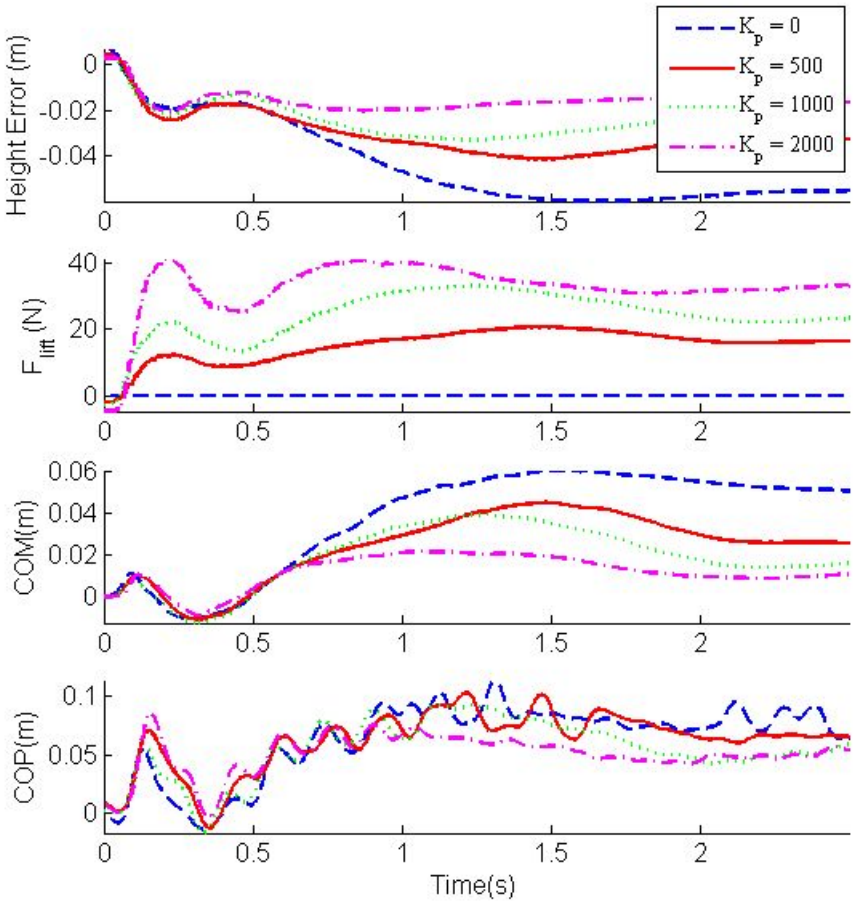


**Figure 5.8:** Comparison of heavy lifting experiments with a range of $K_p$ values.

### 5.6.4 Example: Force Feedback

The previous heavy object lifting task can be modified to incorporate force feedback control. The mass of the heavy object causes the effective COM of the entire system to move forward. The discrepancy between the model COM and the total COM can be sensed by measuring the center of pressure ($COP$) using the force-torque sensors on the feet. The desired COP, $COP_{\text{des}}$, is the center of the base of support. To compensate for the extra mass, the COM

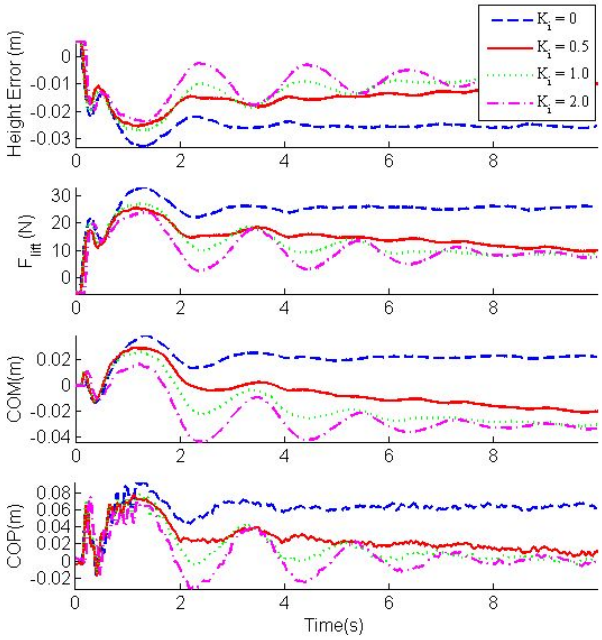


**Figure 5.9:** Comparison of force feedback controls for a range of $K_i$ values. $K_p = 1000$ and $T_0 = 700$ for all experiments.

of the robot is moved backwards using an integral control,

$$\mathbf{C}_{\text{des}} = \frac{K_i}{T_0} \int (\mathbf{COP}_{\text{des}} - \mathbf{COP})\, dt \tag{5.31}$$

where $K_i > 0$ and $T_0$ is a scaling factor.

Figure 5.9 shows the effect of the force feedback controller in Eq. (5.31). Without feedback ($K_i = 0$) the COP moves to the front of the foot where balance control is poor. As $K_i$ is increased, the COP is moved back to the center of the foot and the COM of the robot is moved backward. Doing so also results in better task performance because the robot has greater force control authority in this state. If $K_i$ is increased too much, oscillations in the COP can occur due to the dynamics of the robot. A COP controller that combines integral control and the dynamics of the robot [113] could be implemented for improved control.

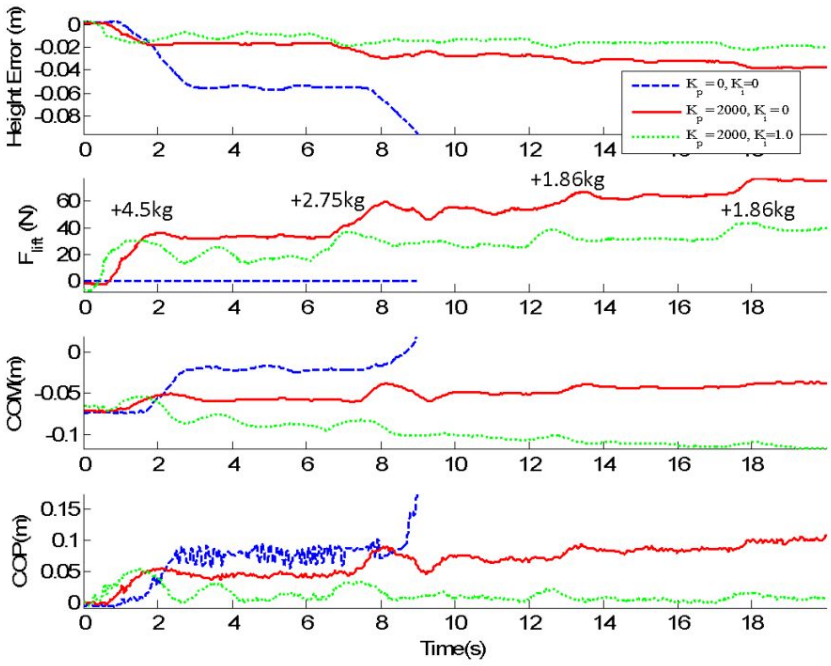


**Figure 5.10:** Comparison of different $K_p$ and $K_i$ gains for lifting multiple weights.

Figure 5.10 demonstrates control with and without force feedback when multiple weights are added to the bucket one at a time. For the $K_p = 0$ case, the robot falls over after the second weight is added. With $K_p = 2000$, the robot can remain standing. Without force feedback, both the COM and COP move forward with each weight. With force feedback, the COM is moved backward and the COP is regulated to the middle of the foot.

### 5.6.5 Example: Walking Control

This controller can be extended beyond standing balance to the control of locomotion such as walking. For this example, walking trajectories of the COM are generated using Push Recovery Model Predictive Control (PR-MPC), described in Chapter 4. This controller uses the LIPM dynamics to generate COM trajectories for walking. DBFC-VMC takes the output

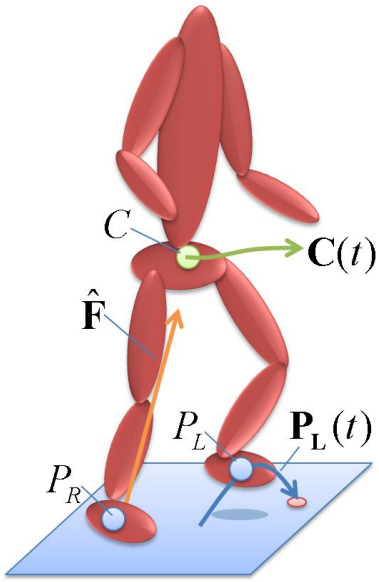


**Figure 5.11:** Walking control is achieved by tracking the desired COM and swing foot trajectories. Inverse kinematics is used to update the desired pose and DBFC-VMC is used to generate full-body joint torques.

of PR-MPC to generate full-body joint torques for walking.

PR-MPC can be thought of as a function, $\Phi$, of the current COM state, $(\mathbf{C}, \dot{\mathbf{C}})$ and foot locations, $\mathbf{P}_L$ and $\mathbf{P}_R$, that returns trajectories for the COM, $\mathbf{C}(t)$ and feet, $\mathbf{P_L}(t)$ and $\mathbf{P_R}(t)$, over the next several footsteps,

$$\Phi : \left\{\mathbf{C}, \dot{\mathbf{C}}, \mathbf{P}_L, \mathbf{P}_R\right\} \rightarrow \{\mathbf{C}(t), \mathbf{P_L}(t), \mathbf{P_R}(t)\} \tag{5.32}$$

as illustrated in Figure 5.11. In this example, footsteps are being planned three steps into the future and replanned just after each touchdown.

The controller for this task is written as a trajectory-tracking controller with feed-forward accelerations,

$$\ddot{\mathbf{C}}_{\text{des}} = \ddot{\mathbf{C}}(t) + \mathbf{K}_{cp}\left(\mathbf{C}(t) - \mathbf{C}\right) + \mathbf{K}_{cd}\left(\dot{\mathbf{C}}(t) - \dot{\mathbf{C}}\right) \tag{5.33}$$

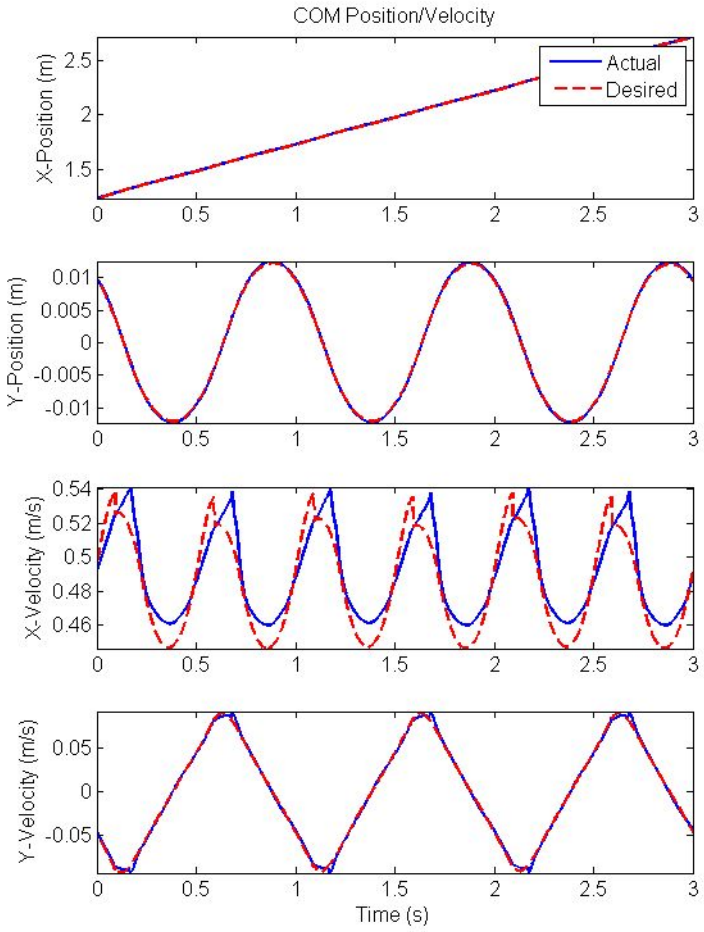


**Figure 5.12:** Trajectory of the COM generated by PR-MPC for a 0.5 m/s desired walking speed in simulation. Dashed lines represent desired positions/velocities and solid lines represent true state of the robot.

When substituted into Eq. (5.20), this generates the desired contact forces, $\mathbf{F}^*$. Joint torques can be found using Eq. (5.21), which can be re-written as

$$\mathbf{G}\begin{pmatrix} \ddot{\mathbf{q}} \\ \tau_{\text{dbfc}} \end{pmatrix} = \begin{pmatrix} -\mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{J}^T\mathbf{F}^* \\ \ddot{\mathbf{P}}(t) - \dot{\mathbf{J}}\dot{\mathbf{q}} \end{pmatrix} \tag{5.34}$$

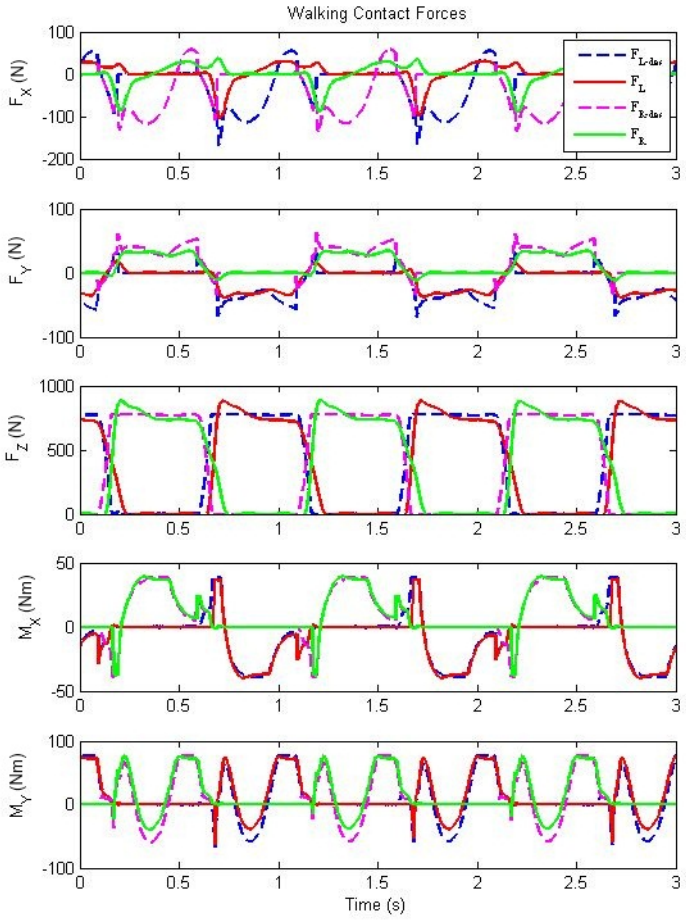


**Figure 5.13:** Contact forces during forward walking at 0.5 m/s in simulation. The dashed lines are the desired forces used by DBFC to create the full-body joint torques whereas the solid lines are the actual contact forces.

where $\ddot{\mathbf{P}}(t)$ is non-zero for the swing leg. The footstep trajectories can be defined in a number of ways. In this chapter, minimum jerk trajectories are defined between the takeoff position and desired footstep location with a predefined height of 5cm starting and ending with zero velocity. Joint trajectories are calculated using inverse kinematics [83], determining both

desired joint angles and joint velocities, and tracked using low gain PD control.

This walking controller has been successfully applied in simulation. Figure 5.12 shows the COM positions and velocities in the horizontal plane. In this simulation, PR-MPC is recomputed at every footstep. The desired COM trajectories calculated by PR-MPC are tracked very closely even though joint position gains are low and joint trajectory tracking is not perfect. Figure 5.13 shows the desired contact forces used by DBFC and the actual contact forces that result.

## 5.7 Discussion

The controller presented in this chapter is a model-based controller. It uses a 3D rigid-body dynamics model of the entire robot to determine forward and inverse dynamics. One of the problems with this type of controller is that the robot does not match the model. Not only can kinematics and mass properties be wrong, but unmodeled effects such as hydraulic fluid flow, deformable joints, and ground contact compliance can cause unpredicted behavior. These effects can be a significant problem for force-controlled robots performing dynamic motions. While the model used to control our robot is not perfect, reasonable force control is achieved, for example in Figure 5.2.

The simplest method for overcoming some modeling error is to include the feedback controller in Eq. (5.24). The structure used for feedback control can be a PD controller, which assumes all joints are independent, or a linear quadratic regulator (LQR) full-state feedback controller derived from a linearized model. In this chapter, this joint feedback controller is simply appended to the DBFC torques. A prioritized control approach could be used to limit the interaction between the two controllers [93][19].

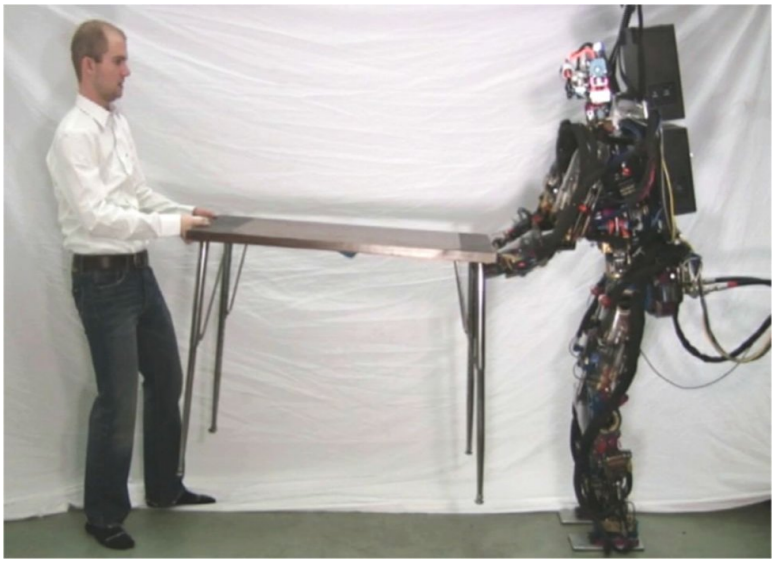

**Figure 5.14:** DBFC-VMC can be applied to cooperative manipulation tasks such as helping a person lift a table.

The DBFC-VMC controller can accommodate a wide variety of tasks. It also inherits from traditional VMC the ability to define tasks in a model-free way. As shown in Figure 5.6, a model of the object isn't needed, only a definition of a controller to grasp it. However, if a model is provided, DBFC-VMC can take advantage of the model to achieve improved control. It can be easily extended to related force control tasks like cooperative manipulation as shown in Figure 5.14 (see the video at `http://www.cs.cmu.edu/~cga/bstephens/videos/table.mpg`).

A slightly modified version of the task in Figure 5.6 would include grasping forces, which are internal to the system. To grasp the object steadily, both arms could apply an equal and opposite force. This force has no effect on the ground contact forces but does affect the joint torques in the two arms. This idea was tested in a full-body simulation shown in Figure 5.15

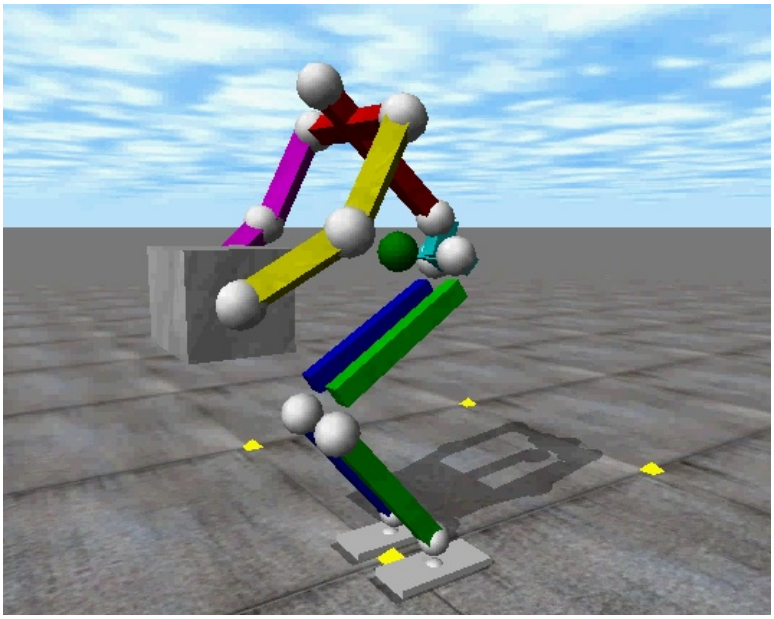

**Figure 5.15:** DBFC-VMC applied to a full-body robot in simulation lifting a heavy object.

(see the video at `http://www.cs.cmu.edu/~cga/bstephens/videos/liftsim.avi`). The lifting example also highlights a possible problem with the DBFC-VMC as presented. If the object is heavy enough to tip the robot over, adjustments such as leaning backwards or moving a foot to reshape the base of support may be required [142]. However, with proper placement of the COM and feet, DBFC is a simple choice for determining full-body joint torques.

The role of angular momentum is not emphasized in this chapter. Angular momentum is generated by applying a torque about the COM, for example by bending quickly at the hip or swinging the arms. Doing so can momentarily generate higher horizontal forces to aid balance. However, the humanoid form is not built to store angular momentum as in a

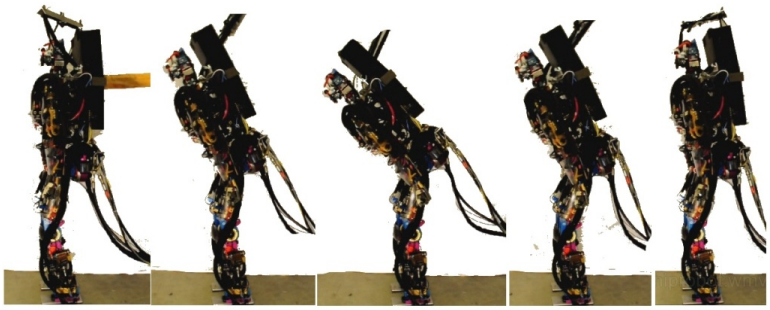

**Figure 5.16:** Screen captures of large push recovery using a hip strategy

flywheel; any angular momentum added to the system must also be removed quickly. In this chapter, the desired behavior is to minimize the generation of angular momentum by setting the desired change of angular momentum to zero, $\dot{\mathbf{H}}_{\text{des}} = \mathbf{0}$. In fact, it is generally useful to dissipate angular momentum by setting $\dot{\mathbf{H}}_{\text{des}} = -\mathbf{K}_p\mathbf{H}$. A multi-objective controller [2][20] can be used to better manage this tradeoff and result in the behavior shown in Figure 5.16 (see video at `http://www.cs.cmu.edu/~cga/bstephens/videos/hiprobot.wmv`).

## 5.8 Conclusion

This chapter presented a controller for mapping controls derived for a simple model to full-body joint torques for a humanoid robot. Dynamic Balance Force Control (DBFC) first computes feasible contact forces based on a desired COM acceleration and then computes full-body joint torques. This controller was demonstrated in force control tasks such as balancing, lifting heavy objects and walking.

# Chapter 6

# COM State Estimation

In this chapter, simple models are used to improve humanoid balance and push recovery by providing a better state estimate even in the presence of small modeling error. A good estimate of the COM position and velocity is important for choosing strategies, such as described in Section 3.3. The state estimator presented in this chapter was used for all hardware experiments in Chapters 4 and 5.

## 6.1 Introduction

The goal of this chapter is to provide a practical exploration of the effects of modeling error and unknown forces on state estimation for dynamically balancing humanoid robots. These disturbances have the effect that the observed dynamics do not match the expected dynamics, a discrepancy that cannot be simply represented as process noise. State estimators, such as a Kalman filter [58], are complicated by this because they rely on a prediction step that

uses the expected forward dynamics. Likewise, model-based control frameworks, such as model predictive control (MPC), use the expected dynamics to generate a control policy. This policy is also often in the form of a state feedback controller which requires estimates of the full state of the system. This chapter will explore how certain modeling errors can be overcome, either through changes to the process model in the estimator or the addition of specific sensing capabilities.

Humanoid balance is often studied using simplified models. One of the simplest and most widely-used is the Linear Inverted Pendulum Model (LIPM) [56] which treats the body as a lumped mass, centered at the center of mass (COM), that only moves horizontally. These assumptions result in a simple linear model of the dynamics, opening up the possibility of using any number of modern linear control and estimation techniques. One of the most widely used applications of the LIPM has been walking trajectory optimization using preview control [53],[132], which outputs a COM trajectory that satisfies the constraints that the center of pressure (COP) be within the base of support.

In general, many implementations of balance control on humanoid robots rely on stiff position-controlled joints and perfect tracking of the desired COM trajectory. However, these methods tend to fail in the presence of unmodeled disturbances such as uneven ground. This chapter focuses on robots with force-controlled joints, such as the Sarcos Primus humanoid, that can comply to disturbances but require reactive balance control. Balance for force-controlled robots is often achieved using force-control techniques [98],[50],[115], where feedback is required to control COM state.

The problem being explored in this chapter is that the state (position and velocity) of

the COM is not easily determined. Not only is there no direct measurement of joint velocity in our robot, but modeling error can be significant. Often, modeling error has been modeled by augmenting the process and/or output models with step disturbances [85],[91]. The construction of these augmented models can have both positive and negative effects. This chapter will consider both types of disturbances in the context of humanoid balance. First, an error in the measurement of the position of the COM can be represented by an output disturbance. Second, an unknown external force can be modeled by a state disturbance. **The main contribution of this chapter are the application of modeling-error-based Kalman filters to improving COM state estimation and control in humanoid balance**.

This chapter is outlined as follows. In Section 6.2, the different process dynamics models and sensor models are combined to form Kalman filters. Their performance is theoretically predicted by computing the observability of the system. Next, in Section 6.3, state estimators are designed and compared for different cases of modeling error. In Section 6.5, the estimators are applied to the state estimation of a force-controlled humanoid robot performing different balance tasks. Finally, in Section 6.6, an extension of these ideas is presented which additionally includes foot positions as variables allowing for a world-coordinates-based state estimate as the robot walks.

| Process Dynamics | Estimator (Process Model, Sensor Model) | | |
|---|---|---|---|
| | COM | COP | Dual |
| Naïve | $\mathbf{A}_0, \mathbf{C}_{1\|0}$ | $\mathbf{A}_0, \mathbf{C}_{2\|0}$ | $\mathbf{A}_0, \mathbf{C}_{12\|0}$ |
| External Force | $\mathbf{A}_1, \mathbf{C}_{1\|1}$ | $\mathbf{A}_1, \mathbf{C}_{2\|1}$ | $\mathbf{A}_1, \mathbf{C}_{12\|1}$ |
| COM Error | $\mathbf{A}_2, \mathbf{C}_{1\|2}$ | $\mathbf{A}_2, \mathbf{C}_{2\|2}$ | $\mathbf{A}_2, \mathbf{C}_{12\|2}$ |
| Dual | $\mathbf{A}_{12}, \mathbf{C}_{1\|12}$ | $\mathbf{A}_{12}, \mathbf{C}_{2\|12}$ | $\mathbf{A}_{12}, \mathbf{C}_{12\|12}$ |

**Table 6.1:** Different types of process dynamics models, $\mathbf{A}_j$, are combined with different sensing models, $\mathbf{C}_{i|j}$, to create Kalman filters.

## 6.2 System Modeling

In this chapter, different state estimators will be constructed by pairing process dynamics models, $\mathbf{A}_j$, with sensor models, $\mathbf{C}_{i|j}$, to form Kalman filter [58] state estimators. The various pairings of process models and sensing models are shown in Table 6.1.

The 1D Linear Inverted Pendulum [56] is used as the basic process dynamics. Just as in Eq. (3.5),

$$\ddot{x} = \omega^2 (x - x_c) \tag{6.1}$$

where $\omega = \sqrt{g/z_0}$, $x$ is the position of the center of mass (COM), $z_0$ is the constant height of the COM, and $x_c$ is the position of the center of pressure (COP). The LIPM dynamics are convenient because they can be written as a discrete time linear system in state space form,

$$\mathbf{x}_{k+1} = \mathbf{A}_0 \mathbf{x}_k + \mathbf{B}_0 u_k \tag{6.2}$$

where $\mathbf{x}_k = \{x, \dot{x}, x_c\}^T$ is the state vector and $u_k = \dot{x}_c$ is the input at timestep $k$. The

matrices in Eq. (6.2) are given by

$$\mathbf{A}_0 = \begin{bmatrix} 1 & T & 0 \\ \omega^2 T & 1 & -\omega^2 T \\ 0 & 0 & 1 \end{bmatrix} \tag{6.3}$$

and

$$\mathbf{B}_0 = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{6.4}$$

where $T$ is the timestep. This simple Euler approximation is used for clarity. It can be easily implemented for small $T$, such as the $T = 0.0025$s corresponding to the $400Hz$ control rate on the robot. $\mathbf{A}_0$ is the basic dynamics equation that will be used throughout this chapter, though some modifications will be proposed.

A linear measurement model is also assumed to have the form,

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k \tag{6.5}$$

where $\mathbf{y}_k$ is a vector of measurements. The structure of $\mathbf{C}$ depends on the type of sensors and state description.

### 6.2.1 Sensing Models

This section will describe various sensing models to be used in Eq. (6.5). The output matrices are denoted by $\mathbf{C}_{i|j}$ where $i$ denotes the sensor model (i.e. $S_i$) and $j$ denotes the state dynamics (i.e. $A_j$). Combinations of sensor models will be represented by $\mathbf{C}_{ik|j}$, for example $\mathbf{C}_{12|j}$.

**S1) Position -** Position sensors are the most common type of sensor. In this case, the position of the COM is calculated from a model of the mass distribution in the system (which is incorrect)

$$\mathbf{C}_{1|0} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{6.6}$$

**S2) Center of Pressure -** It is also common for balancing systems to have contact force sensors that measure the center of pressure of the system. In this case,

$$\mathbf{C}_{2|0} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{6.7}$$

**S3) Total COM Force -** A ground contact force sensor can provide the total horizontal force on the COM. In this case, Eq. (6.1) shows up in the output model, multiplied by the mass of the system,

$$\mathbf{C}_{3|0} = \begin{bmatrix} m\omega^2 & 0 & -m\omega^2 \end{bmatrix} \tag{6.8}$$

### 6.2.2 Modeling Errors

Two general types of error common to humanoid robots are considered here: an unknown COM offset in the measurement output and an unknown external force. The modified process and output models for these will be presented. These can be considered to be unknown parameters of the model. As with states such as the COM velocity, there are no direct measurements of these parameters available, so they are included in the dynamics model as extra states to be estimated.

**D1) Unknown COM Offset -** It is often the case in real systems that the true position

of the COM will be unknown. This error can be represented by an output step disturbance,

$$x_k = \hat{x}_k + \Delta x_k \tag{6.9}$$

where $\hat{x}$ is the estimated COM position and $\Delta x$ is an unknown offset. In this case, a COM offset can be appended to the state vector, $\mathbf{x} = \{x, \dot{x}, x_c, \Delta x\}^T$ and process model re-written as

$$\mathbf{A}_1 = \begin{bmatrix} 1 & T & 0 & 0 \\ \omega^2 T & 1 & -\omega^2 T & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.10}$$

which has the same dynamics as in Eq. (6.1).

**D2) External Force -** It also quite possible that an unmodeled external force could be perturbing the system. This force could represent an unknown contact with the world such as leaning against a wall, an interaction force such as shaking a hand, or an inertial frame force such as standing on a moving bus. In this case, the dynamics take the form,

$$\ddot{x} = \omega^2 (x - x_c) + u_x \tag{6.11}$$

where $u_x$ is an unknown external force. This model was also discussed in Section 3.2.2. In this case, an external force estimate can be appended to the process model, making the state,

$\mathbf{x} = \{x, \dot{x}, x_c, u_x\}^T$, which has the dynamics of Eq. (6.11) and the state space dynamics,

$$\mathbf{A}_2 = \begin{bmatrix} 1 & T & 0 & 0 \\ \omega^2 T & 1 & -\omega^2 T & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.12}$$

**D12) External Force and COM Offset -** If both of the above modeling errors are assumed, then both an unknown external force and COM offset are appended to the dynamics of the system,

$$\mathbf{A}_{12} = \begin{bmatrix} 1 & T & 0 & 0 & 0 \\ \omega^2 T & 1 & -\omega^2 T & T & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.13}$$

### 6.2.3 State Estimators

In this section, the process dynamics models from Sections 6.2.2 and 6.2.2 and sensing models from Section 6.2.1 are combined to form Kalman filters [58] denoted by $F_{i|j}$.

Process and measurement noise models are also constructed for the system to form the dynamics needed by the state estimator,

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \nu_x \tag{6.14}$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \nu_y \tag{6.15}$$

where $\nu_x$ is a vector representing the process noise of each of the components of $\mathbf{x}_k$ and $\nu_y$

| Variable | Process Noise | Output Noise |
|---|---|---|
| Position | $1e^{-8}$ | $1e^{-5}$ |
| Velocity | $1e^{-4}$ | |
| COP | $1e^{-4}$ | $1e^{-5}$ |
| COM Offset | $1e^{-4}/1e^{-8}$ | |
| External Force | $1e^{-4}$ | |
| Total Force | | 1 |

**Figure 6.1:** Table of covariance values used for the various estimators. These values were determined manually.

is a vector of measurement noise on each component of $\mathbf{y}_k$. The noise variables are assumed to be Gaussian processes with

$$\nu_x \sim N(\mathbf{0}, \mathbf{Q}) \tag{6.16}$$

$$\nu_y \sim N(\mathbf{0}, \mathbf{R}) \tag{6.17}$$

The covariance values used throughout this chapter are shown in Figure 6.1.

An estimator will be denoted by $F_{i|j}$ which corresponds to a process model $\mathbf{A}_j$ and a sensor model $\mathbf{C}_{i|j}$. Below are descriptions of various state estimators.

**F0) Naive Estimator ($F_{i|0}$) -** This estimator is referred to as *naive* because it has no representation of the modeling error besides the normal process noise.

**F1) COM Offset Estimator ($F_{i|1}$) -** For this estimator, the $A_1$ process model includes the COM offset as a variable and the output model is modified with a position measurement

derived from Eq. (6.9),

$$\mathbf{C}_{1|1} = \begin{bmatrix} 1 & 0 & 0 & -1 \end{bmatrix} \tag{6.18}$$

**F2) External Force Estimator ($F_{i|2}$) -** In this case, the $A_2$ process model is used but the output model is unchanged, except by adding an additional column for the force,

$$\mathbf{C}_{12|1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.19}$$

**F12) Dual Estimator ($F_{i|12}$) -** In this case, the $A_{12}$ process model is used and the output model includes two additional columns,

$$\mathbf{C}_{12|12} = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{6.20}$$

### 6.2.4 Observability

One way to partially predict the performance of different estimators is to consider the observability of the system defined by Eq. (6.14) and Eq. (6.15).

Given an estimator, $F_{i|j}$, observability is determined by evaluating [13]

$$O\left(F_{i|j}\right) = rank\left(\begin{bmatrix} \mathbf{C}_{i|j} \\ \mathbf{C}_{i|j}\mathbf{A}_j \\ \mathbf{C}_{i|j}\mathbf{A}_j^2 \\ \vdots \\ \mathbf{C}_{i|j}\mathbf{A}_j^{n-1} \end{bmatrix}\right) \tag{6.21}$$

where $n$ is the dimension of the state vector (including any offset parameters). If

| Process Model | Sensor Model (Detectability rank) | | |
|---|---|---|---|
| (state size) | **S1** | **S12** | **S123** |
| **$A_0$ (3) (Naïve)** | 3 | 3 | 3 |
| **$A_1$ (4)** | 3 (X) | 4 | 4 |
| **$A_2$ (4)** | 3 (X) | 4 | 4 |
| **$A_{12}$ (5)** | 3 (X) | 4 (X) | 4 (X) |

**Figure 6.2:** Table of observability given different process and output models. If the observability is equal to the size of the state vector then the state is fully observable. An (X) signifies that the system is not observable given these conditions.

$O\left(F_{i|j}\right) = n$, then the system is observable. The table in Figure 6.2 gives this value for several different process and sensor model pairs.

Note that the dual estimator (F12) is not observable even given all three sensors (S123); there are not enough independent measurements in the S123 sensor to distinguish between an external force and a COM measurement error.

## 6.3 State Estimator Comparisons

In this section, the performance of various estimators in the presence of different modeling errors is compared. The estimators take as inputs data from known ground truth state trajectories generated by sinusoids with an amplitude, $a = 1cm$, and frequency of $\beta = 2\pi\ rad/s$.

The state trajectories used have the form

$$x(t) = a \sin \beta t \tag{6.22}$$

$$\dot{x}(t) = a\beta \cos \omega t \tag{6.23}$$

$$x_c(t) = x(t) + \frac{1}{\omega^2}\left(a\beta^2 \sin \beta t + u_x\right) \tag{6.24}$$

and COM measurements of

$$y_1 = x(t) - \Delta x \tag{6.25}$$

$$y_2 = x_c(t) \tag{6.26}$$

### 6.3.1 Naive Estimation

First, the need for estimators which account for modeling error is demonstrated. In this section a naive state estimator (F0) is implemented. However, when generating the ground truth trajectories, modeling error in the form of a COM measurement offset (D1) or an external force (D2) is added. The performance using position-only (S1) and position-COP (S12) measurement models is compared.

Figures (6.3) and (6.4) show the results using the naive estimators in the presence of a COM offset and an unknown external force, respectively. Using S1, both the velocity and COP are poorly estimated. For the S12 case, the COP is correctly estimated but there are still errors in the COM velocity estimates. Both S1 and S12 fail in the D1 case to correctly estimate the COM position.

Because of this, it is desirable to modify the estimator to include additional states that correct for these steady state offsets.

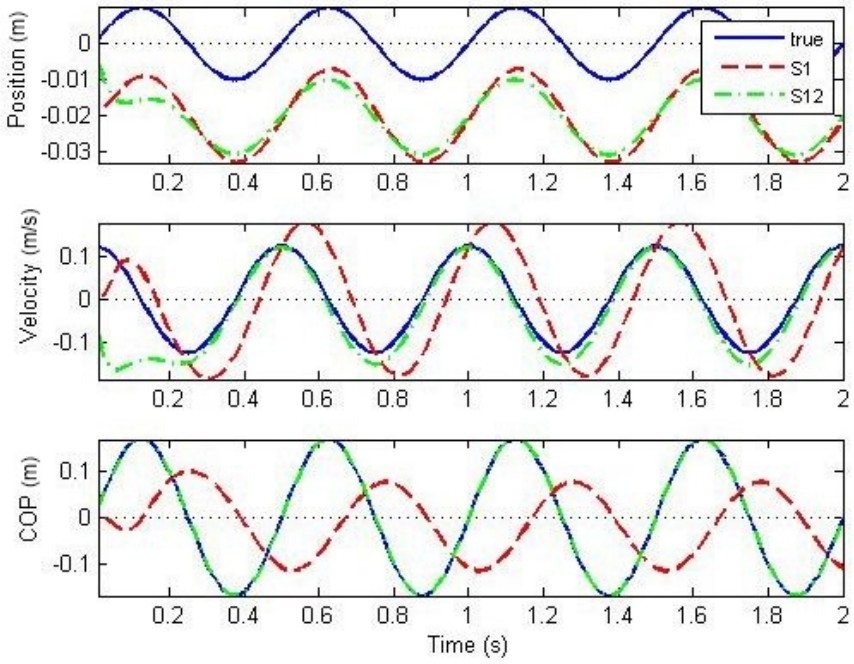


**Figure 6.3:** State estimation performance unknown COM offset (D1) comparing position-only measurement (S1) with position and COP measurement (S12). A COM offset of 2cm was used.

### 6.3.2 COM Offset Estimation (F1)

Results using an $F_{i|1}$ state estimator in the presence of an output error of $\Delta x = 2cm$ are shown in Figure 6.5. Using only S1, the system is unable to estimate the state properly, whereas using S12 the estimator quickly converges.

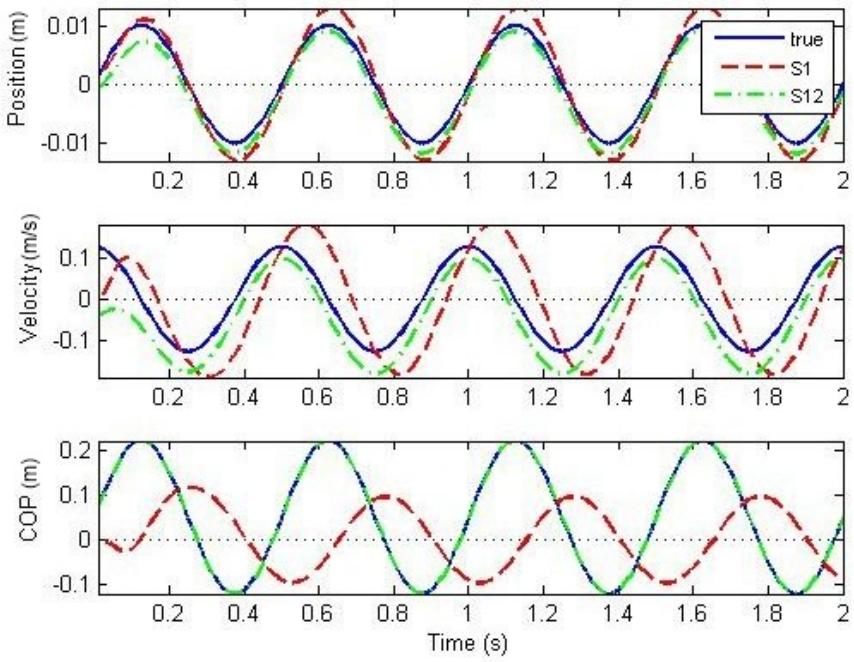


**Figure 6.4:** State estimation performance with unknown external force (D2) comparing position-only measurement (S1) with position and COP measurement (S12). The external force was set such that $u_x = 0.5$.

### 6.3.3 External Force Estimation (F2)

Results using an $F_{i|2}$ state estimator in the presence of a state disturbance of $u_x = 0.5m/s^2$ are shown in Figure 6.6. Again using the S1 sensor isn't enough to correctly estimate the state whereas the S12 succeeds.

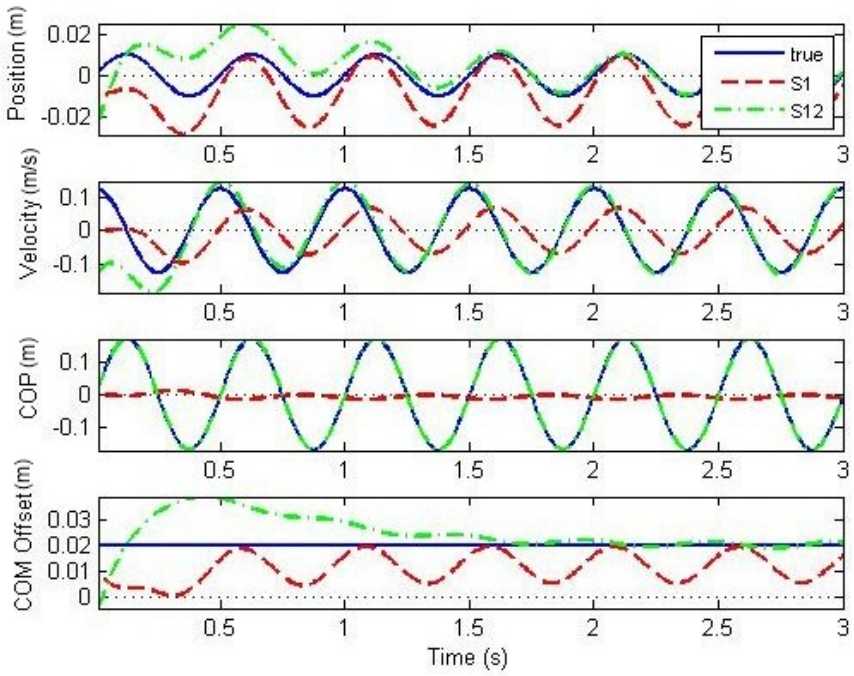


**Figure 6.5:** State estimation performance with unknown COM offset (D1) comparing position-only measurement (S1) with position and COP measurement (S12). The COM offset (with a true value of $\Delta x = 0.02m$) is added as an additional state variable in the process model.

## 6.4 Controlling With Modeling Error

A poor state estimate can lead to poor balance control. In the previous section, ground truth sinusoidal trajectories were generated analytically and used to test the state estimation. In this section, however, a controller is used to track the sinusoidal trajectories. It is assumed that the controller only has access to the state estimate.

A fixed gain linear quadratic regulator (LQR) controller, designed using the dynamics

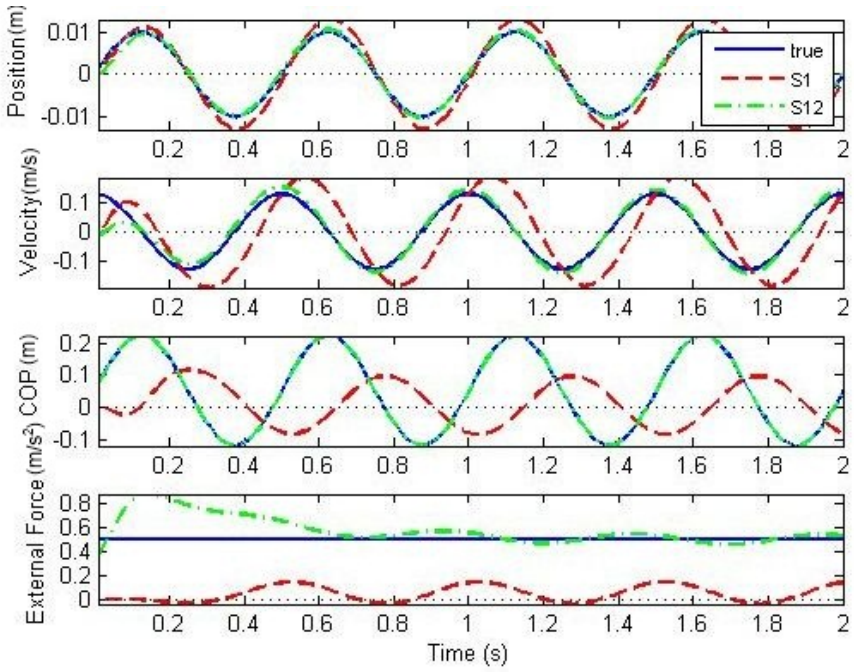


**Figure 6.6:** State estimation performance with unknown external force (D2) comparing position-only measurement (S1) with position and COP measurement (S12). The external force (with a true value of $u_x = 0.5m/s^2$) is added as an additional state variable in the process model.

from Eq. (6.3) and Eq. (6.4), is used. Desired trajectories are generated from the same sinusoid patterns above, except the terms, $u_x$ and $\Delta_x$, are set to zero because the modeling error isn't known in advance. Feedforward input commands are also calculated from these trajectories. The controller has the form,

$$u_k = \dot{x}_{ck} + \mathbf{K}\left(\mathbf{x}_k^d - \hat{\mathbf{x}}_k\right) \tag{6.27}$$

where $\mathbf{K}$ is a $(1 \times 3)$ LQR gain matrix and $\hat{\mathbf{x}}_k$ is the state estimate at timestep $k$. The LQR

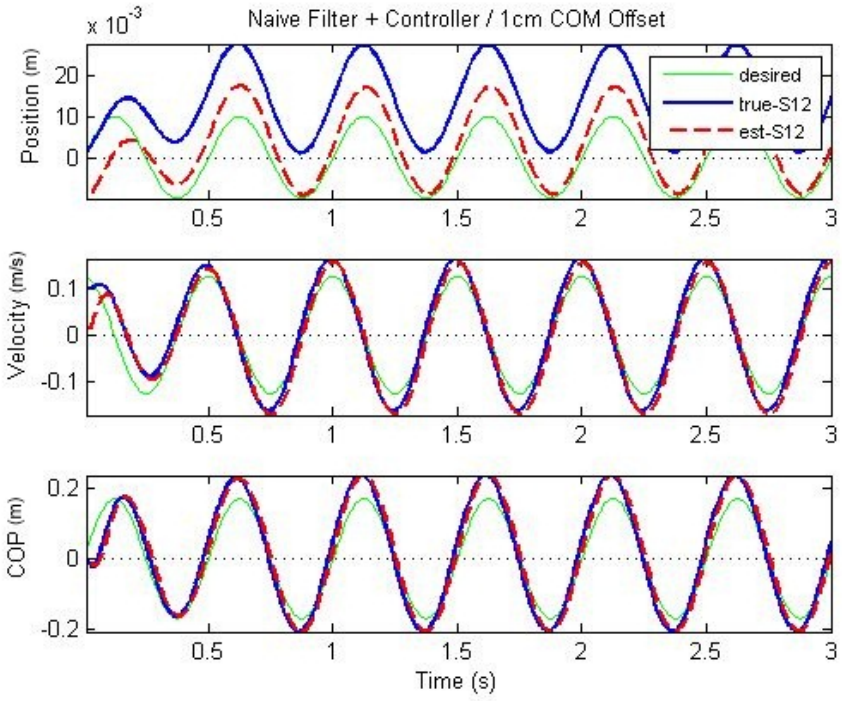


**Figure 6.7:** Controller in the presence of an unknown COM offset (D1) using naive estimator.

controller is designed with a state cost, $\mathbf{Q}_x = diag([1e^2, 1, 1])$, and a control cost of $\mathbf{R}_u = 1$.

### 6.4.1 Naive Control

First, the need to consider the modeling error during control in the presence of modeling error is demonstrated. Consider a system with a COM measurement offset (D1) but using an estimator with only the $\mathbf{A}_0$ process model. This result is shown in Figure 6.7. As expected, the system is stable but there is a steady state offset between the estimated COM and the true COM and the true COM does not reach zero.

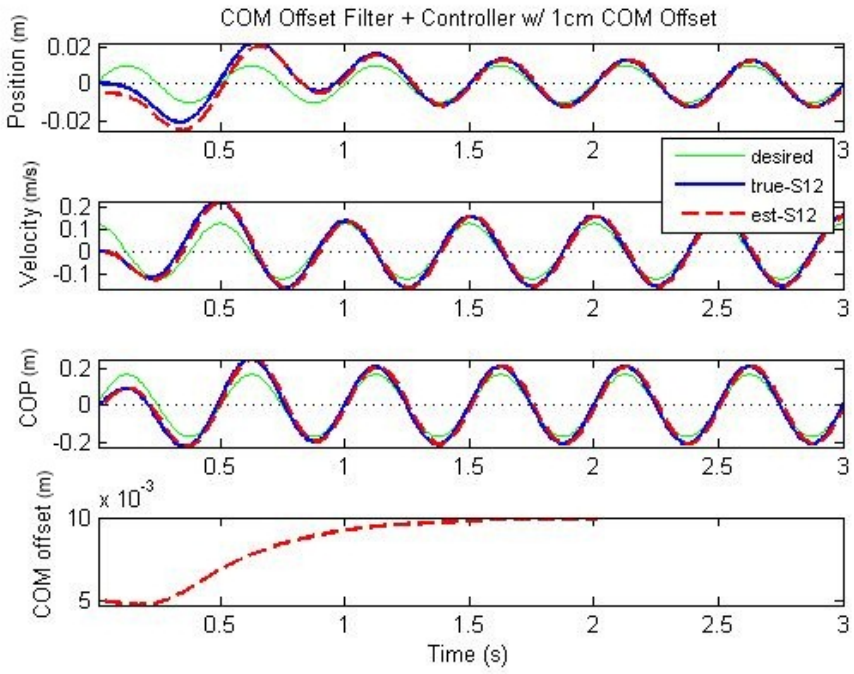


**Figure 6.8:** Controller in the presence of an unknown COM offset (D1) using COM offset estimator (F1).

### 6.4.2 COM Offset Control

In an attempt to eliminate the steady state offset between the true COM and the goal, an estimator that estimates the COM offset, $F_{12|1}$, can be implemented. The result of this is shown in Figure 6.8. The system simultaneously regulates to the goal and estimates the COM offset. Note that this controller requires the S12 sensor model.

### 6.4.3 External Force Compensation

Unlike the COM offset, an external force actually changes the dynamics of the system and represents an uncontrollable variable. The result is that while the state estimator will work properly, there will be a steady-state tracking error. For this reason, the steady state desireds are often recomputed based on the current estimate of the disturbance force[91]. In this case, the dynamics in Eq. (6.11) are inverted to determine an offset in the desired COP position such that

$$x^*_{ck} = x_{ck} + \frac{1}{\omega^2}\hat{u}_{xk} \tag{6.28}$$

$x^d_{ck}$ is the desired COP and $\hat{u}_{xk}$ is the estimated external force at timestep $k$. Results using this controller for a constant external disturbance are shown in Figure 6.9.

## 6.5 Experimental Results

The intended application of these state estimators is to better estimate the COM state of the Sarcos Primus humanoid robot shown in Figure 1.5. It can be expected that both types of modeling error (D1 and D2) will be present in the system. Because of the complexity and high dimensionality of the robot, an offset between the modeled COM and the true COM is likely. In addition, the hydraulic hoses that provide the pressurized fluid to the robot are known to apply external forces to the robot that are not measured. Below, the model of the robot used and its relation to the simplified models presented earlier is shown. The experimental setups are described with results.

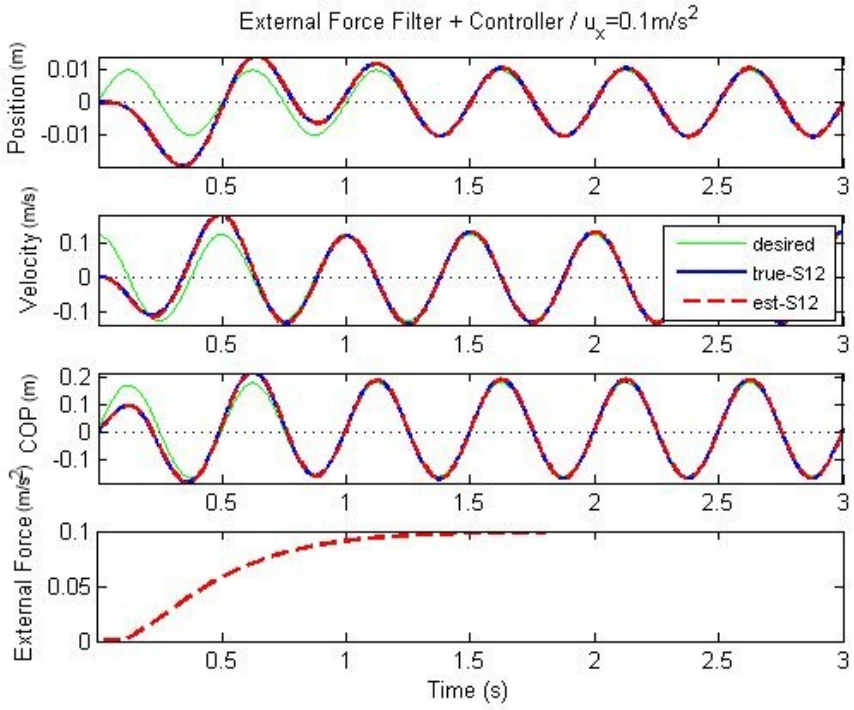


**Figure 6.9:** Controller compensating for a constant external force (D2) using an F2 estimator.

### 6.5.1 Robot Model

A rigid-body dynamics model of the robot is assumed. The model has 12 actuated joints (6 in each leg, excluding thigh rotation) and a free-floating root joint (the pelvis). The torso and upper body joints are assumed to be rigidly attached to the pelvis and are controlled using stiff PD control with constant desireds.

From the robot, the measurements received are joint angles, $\mathbf{q}$, from potentiometers, and pelvis orientation, $\mathbf{r}_0$, given by an IMU mounted to the pelvis. The pelvis position variables, $\mathbf{x}_0$ are not directly observable. However, by assuming that a foot is flat on the ground at a

known position these variables can be determined from the forward kinematics. There are two important features computed by the forward kinematics, including the foot centers,

$$x_L = f_L(\mathbf{0}, \mathbf{r}_0, \mathbf{q}) \tag{6.29}$$

$$x_R = f_R(\mathbf{0}, \mathbf{r}_0, \mathbf{q}) \tag{6.30}$$

and the COM position,

$$x = f_C(\mathbf{0}, \mathbf{r_0}, \mathbf{q}) \tag{6.31}$$

where $f_X(\mathbf{x}, \mathbf{r}, \mathbf{q})$ represents a forward kinematics function. In the above equations, $\mathbf{x}_0$ is set to zero. Therefore, assuming for example that the left foot is flat on the ground, the COM measurement is given by

$$\hat{x} = x - x_L \tag{6.32}$$

The COP measurement is computed by combining the measured COP under each foot using a 6-axis force/torque sensor. If $x_{pL}$ and $x_{pR}$ are the COPs and $F_{zL}$ and $F_{zR}$ are the vertical forces under the left and right foot, respectively, then

$$\hat{x}_c = \frac{(x_L + x_{pL}) F_{zL} + (x_R + x_{pR}) F_{zR}}{F_{zL} + F_{zR}} - x_L \tag{6.33}$$

if the total COP, again relative to the left foot.

### 6.5.2 Experiments

Two types of experiments were performed on the Sarcos Primus humanoid robot. In both experiments, the robot data was generated from a fixed balance controller using only low-pass filtered COM measurements and no state estimators. The state estimators were applied

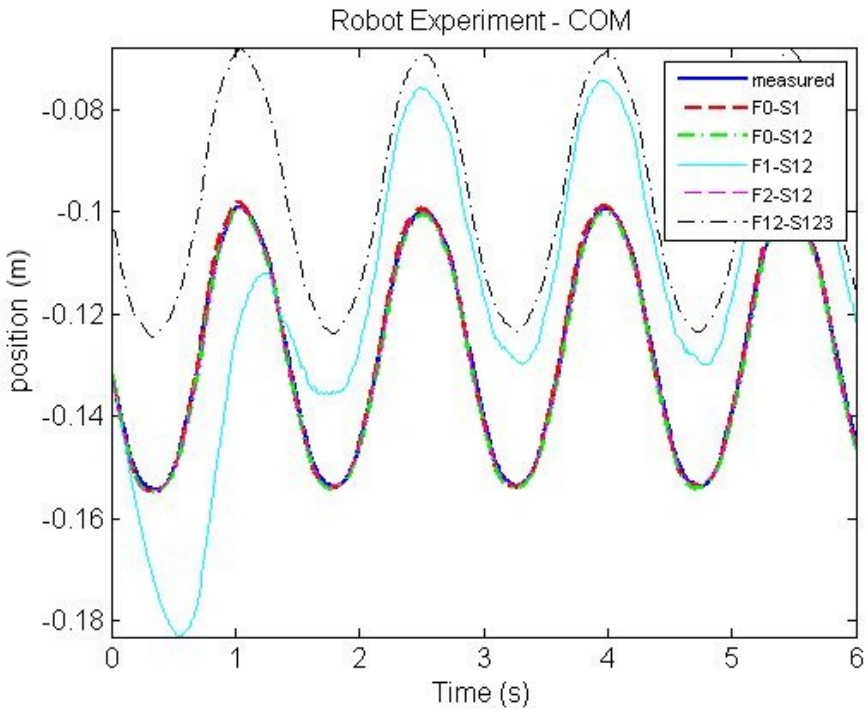


**Figure 6.10:** Comparing the COM position estimation of multiple estimators, two of which estimate COM offset errors.

to this data and performance was compared. For verification, the robot was standing on a force-measuring platform. Though it is predicted to fail, a $F_{123|12}$ estimator is included for comparison which uses the force measurements from the force platform and the output model in Eq. (6.8). The lower covariance for the COM offset is used for the $F_{123|12}$ estimator because this variable is likely to change much slower than the external force.

**Coronal Swaying -** For this experiment, the robot is standing on a force-measuring platform and swaying side to side in the coronal plane in a sinusoidal pattern. Using the measured COM and COP positions, several state estimators were applied to the data. Po-

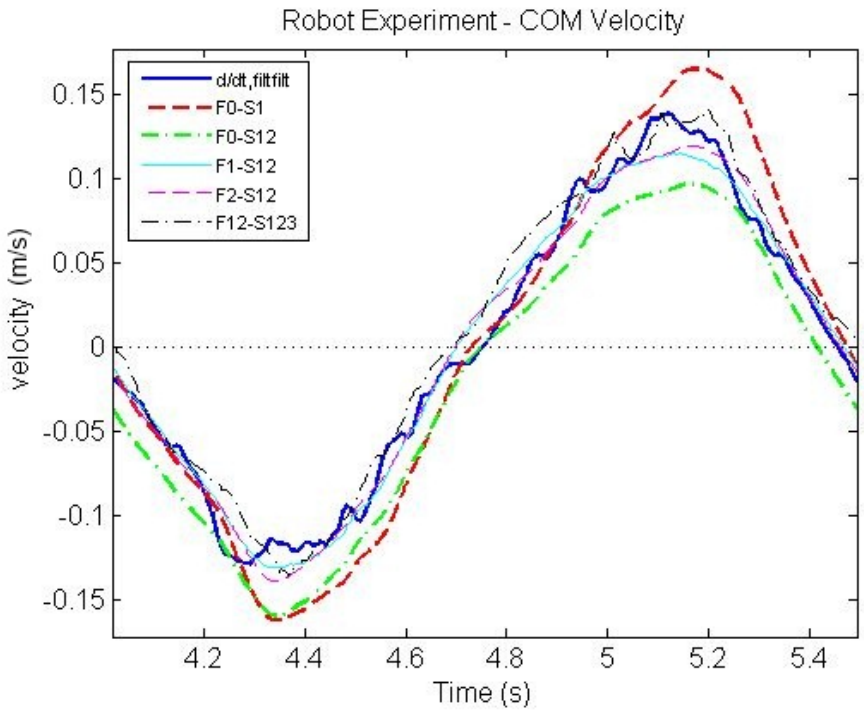


**Figure 6.11:** Zoomed in view of the COM velocity estimation for multiple estimators

sition estimates are shown in Figure 6.10. All estimators perform similarly, with $F_{i|1}$ and $F_{i|12}$ additionally estimating a significant COM offset of approximately $2.4cm$. Figure 6.11 shows a zoomed-in view of the velocity estimates after some time has passed. The velocities are compared to a computed velocity that is calculated by differentiating and applying a zero-phase low-pass filter. The velocity estimation is generally improved by the more advanced estimators. The table in Figure 6.12 compares the error between the estimates and the computed velocity.

**Sagittal Pushes -** For this experiment, the robot is standing on a force-measuring platform and pushed from the front in the sagittal plane. The push is performed using a stick

| Filter | RMS Velocity Error |
|---|---|
| $F_{1\|0}$ | 0.0055 |
| $F_{12\|0}$ | 0.0231 |
| $F_{12\|1}$ | 0.0015 |
| $F_{12\|2}$ | 0.0011 |
| $F_{123\|12}$ | 0.0084 |

**Figure 6.12:** Velocity estimation error of multiple estimators compared to a numerically calculated and filtered velocity.

with a 6-axis force/torque sensor mounted at the end point. There are actually two types of external forces in this experiment. There is a small, nearly-constant external force caused by the hydraulic hoses in addition to the large momentary pushes from the experimenter. The F2 estimator was applied to this data with varying covariances for the external force variable and the results are shown in Figure 6.13. As the process noise covariance is increased, the estimator more closely estimates the push force, but also begins to incorrectly estimate external forces during the transient recovery period between pushes.

## 6.6 Feet State Estimation Extension

The dynamics in Eq. (6.1) generalize the effect of multiple foot contacts by only considering the total center of pressure, $(x_c, y_c)$. However, this point is a result of the forces at each

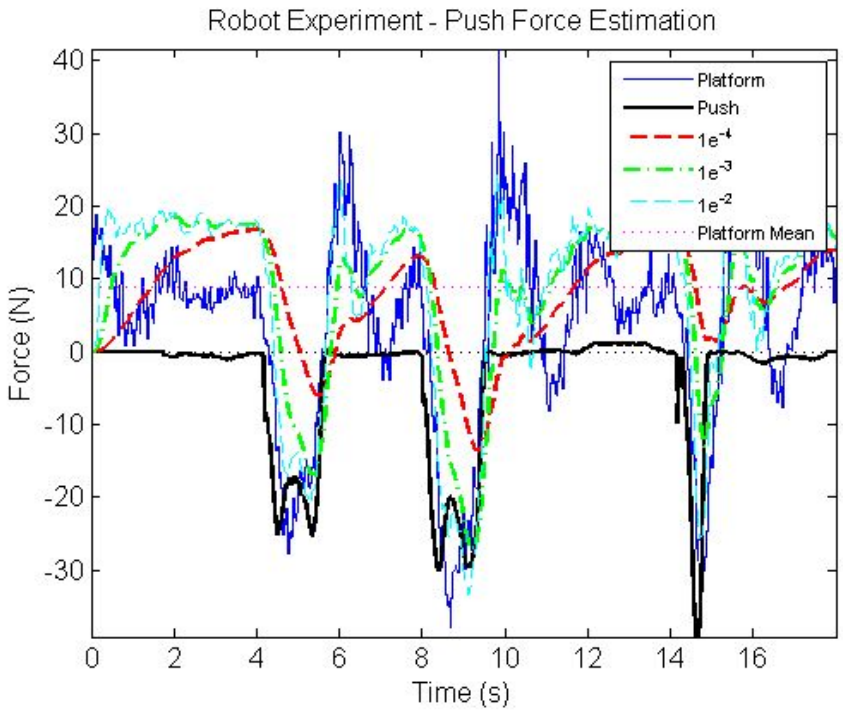


**Figure 6.13:** External force estimation while undergoing pushes in the sagittal plane for various process noise models.

contact point and can be re-written as

$$x_c = \alpha\left(x_L + x_{cL}\right) + (1-\alpha)\left(x_R + x_{cR}\right) \tag{6.34}$$

$$y_c = \alpha\left(y_L + y_{cL}\right) + (1-\alpha)\left(y_R + y_{cR}\right) \tag{6.35}$$

where $(x_L, y_L)$ and $(x_R, y_R)$ are the foot locations and $(x_{cL}, y_{cL})$ and $(x_{cR}, y_{cR})$ are the COPs under the left and right feet, respectively. $\alpha$ is a variable that represents the distribution of force between the two feet such that

$$\alpha = \frac{F_{zL}}{F_{zL} + F_{zR}}. \tag{6.36}$$

This parameter varies between 0 and 1. These variables define an alternative set of COM dynamics equations given by

$$m\ddot{x} = \frac{mg}{z_0}\left(x - \alpha\left(x_L + x_{cL}\right) - \left(1-\alpha\right)\left(x_R + x_{cR}\right)\right) + f_x \tag{6.37}$$

$$m\ddot{y} = \frac{mg}{z_0}\left(y - \alpha\left(y_L + y_{cL}\right) - \left(1-\alpha\right)\left(y_R + y_{cR}\right)\right) + f_y \tag{6.38}$$

These updated LIPM COM dynamics equations can be used for state estimation by constructing a Kalman filter. The dynamics can be discretized and reconstructed into state-space form,

$$\mathbf{X}_{t+1} = \mathbf{f}\left(\mathbf{X}_t\right) + \mathbf{B}\omega \tag{6.39}$$

where

$$\mathbf{X}_t = \left\{x, y, \dot{x}, \dot{y}, x_L, y_L, x_{cL}, y_{cL}, x_R, y_R, x_{cR}, y_{cR}, \alpha, f_x, f_y\right\}.$$

For the purposes of state estimation, it is assumed that there are no inputs to the system, only disturbances.

A new measurement model can also be considered. The measurements available include the centers of pressure and the relative position of the feet to the COM given by the robot

kinematics,

$$\mathbf{Y}_t = \begin{pmatrix} x_L - x \\ x_{cL} \\ y_L - y \\ y_{cL} \\ x_R - x \\ x_{cR} \\ y_R - y \\ y_{cR} \\ \alpha \end{pmatrix} \tag{6.40}$$

Process and measurement noise models are also constructed for the system to form the dynamics needed by the state estimator,

$$\mathbf{X}_{t+1} = \mathbf{f}(\mathbf{X}_t, \omega) \tag{6.41}$$

$$\mathbf{Y}_t = \mathbf{h}(\mathbf{X}_t, \nu) \tag{6.42}$$

where $\omega$ is a vector representing the process noise of each of the components of $\mathbf{X}_t$ and $\nu$ is a vector of measurement noise on each component of $\mathbf{Y}_t$.

A unique feature of this state estimation problem is that there is no direct measurement of the position of the COM, only the position relative to the feet, as in Eq. (6.40). However, there is an indirect measurement provided by the force weighting, $\alpha$. When there is a lot of weight on one foot, it can be assumed that that foot is fixed on the ground. Rather than switching dynamic models, the process model is modified based on the state of the system, such that the foot location has a low process noise when it is on the ground and a high

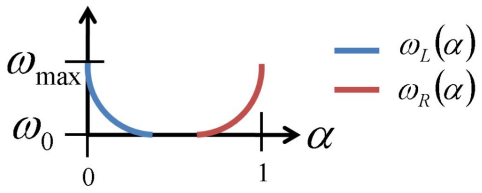


**Figure 6.14:** The process noise associated with the left and right foot variables are functions of the foot force weighting, $\alpha$, such that the process noise is $\omega_0$ when there is little or no weight on the foot and increases to $\omega_{max}$ as the weighting on that foot goes to zero.

process noise when in the air. This behavior can be accomplished by process noise models, $\omega_L(\alpha)$ and $\omega_R(\alpha)$, which are functions of the foot weighting and have a shape similar to that shown in Figure 6.14.

Note that because of the slight nonlinearity in the dynamics, a nonlinear Kalman filter, such as an extended Kalman filter is used to perform state estimation for this system.

The effect of the state dependent process model in Eq. (6.41) is demonstrated by Figure 6.15 and Figure 6.16. Both examples use the same data from a walking simulation. In Figure 6.15, the process noise model is constant. The estimate looks as if the legs are swinging back and forth. By using a state dependent process model, more realistic estimates like those in Figure 6.16 are achieved.

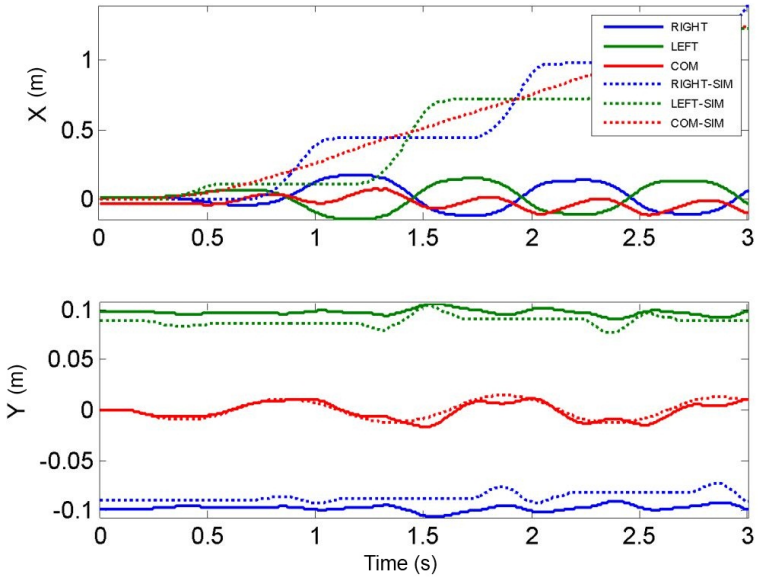


**Figure 6.15:** Simulation of walking without using state dependent process model. The dotted lines are a forward walking simulation. The solid lines are the estimates which are clearly incorrect.

## 6.7 Discussion

This chapter has presented motivational evidence for the use of disturbance-modeling state estimators on force-controlled balance systems using several examples, from both simulation and hardware. Some of the key insights gained are summarized:

- Naive estimation in the presence of modeling error results in inaccurate results.
- COM position only sensing (S1) almost always performs worse than when combined with COP sensing (S12).
- The presented models will not allow the estimation of both COM offset and external

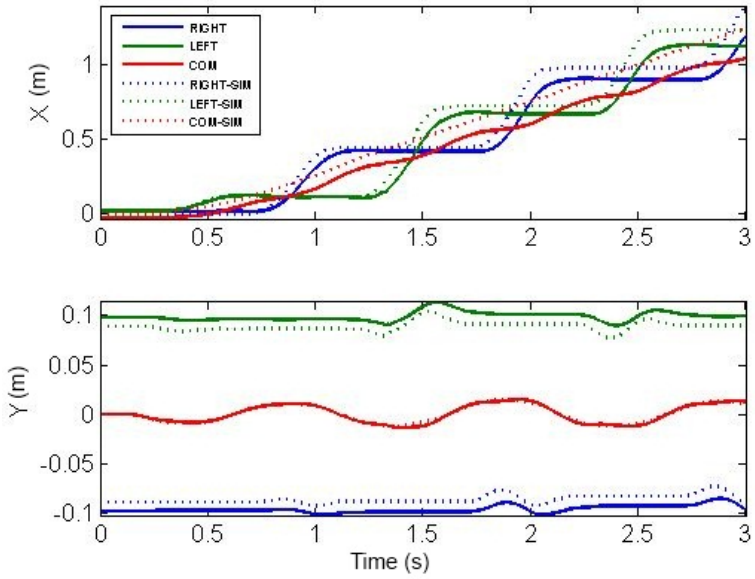


**Figure 6.16:** Simulation of walking using state dependent process model. The dotted lines are a forward walking simulation. The solid lines are the estimates which track the true values but drift slightly over time.

force disturbances (D12) simultaneously.

- There is a trade-off between precisely estimating the disturbance and the state.

The disturbance-modeling estimators presented use parametric models of the modeling error. However, the COM offset and external force disturbances are lumped-mass generalizations of real errors in the highly complex full-body robot. For example, perhaps there are small modeling errors in the COM position of every body in the rigid-body dynamics model of the robot. Active estimation of a COM offset allows for online correction of these small errors without explicitly relying on the full model. This is the major motivation for applying simple models such as the LIPM to humanoid robots.

One of the limitations, however, of the LIPM is that it does not model the effects of angular momentum and external torques on the system. Models have been suggested to model these effects, often through the addition of a flywheel with a rotational inertia to the LIPM model as described in Section 3.2.3. In this case the dynamics become

$$\ddot{x} = \omega^2 (x - x_c) - \frac{\tau}{mh} \tag{6.43}$$

$$I\ddot{\theta} = \tau \tag{6.44}$$

where a torque on the flywheel accelerates both the flywheel and the COM. The angle of the flywheel, $\theta$, can be roughly approximated by the angle of the torso. The added inertia increases the size of perturbations from which balance can be recovered, as described in Section 3.3.2.

Finally, time varying disturbances were only addressed briefly in Section 6.5. There are two approaches to handling this special class of disturbances. The first approach, suggested here, is careful tuning of the process model to change the rate at which disturbances are estimated. If the rate is too high, it is likely the system could go unstable. Another approach is to assume parametric models of the disturbances. If the system expects to experience sinusoidal disturbances, for example if it were standing on a moving bus or boat, it might be worthwhile to model the disturbances in that way instead of a constant.

When applying this state estimator in practice, the external force estimator (F2) is used. This is because it is easier to hand-tune a reasonable COM offset beforehand. In all of the examples presented in this thesis, the estimated external force is simply disregarded. This leads to improved COM position and velocity estimation but reduces the dependence on accurately estimating the external force. Integrating the estimated disturbance force into a

feedback controller was not implemented but could be useful in scenarios with low frequency force interaction. For example, the estimated force could be used as an additional input to a standing balance controller that adjusts the balance strategy based on the this knowledge [73].

## 6.8 Conclusion

This chapter presented the theory and design of state estimators that perform state estimation in the presence of modeling error by considering different disturbance models and sensor modalities. A variety of process and output models were constructed and compared. For a system containing modeling error, it was shown that a naive estimator (one that doesn't account for this error) results in inaccurate state estimates. Finally, estimators were applied to the state estimation on a force-controlled humanoid robot for a sinusoidal swaying task and a push recovery task.

# Chapter 7

# Sarcos Primus Humanoid Robot Control

## 7.1 Introduction

As described in Section 1.6, the Sarcos Primus hydraulic humanoid robot is the experimental platform used throughout this thesis. Most of this thesis has focused on simple models for approximating this system. This chapter will present some details specific to the full robot system, including descriptions of the kinematics and inverse kinematics algorithms, hydraulic torque control, and calibration methods.

## 7.2 Kinematics

Given a planned trajectory of the COM and footstep locations, robot joint trajectories can be generated using inverse kinematics. The planner only specifies the final footstep locations, so footstep trajectories are defined by fifth order spline trajectories starting and ending with zero velocity and acceleration.

A free-floating kinematic model is used to determine both desired joint angles and joint velocities. For full-body behaviors with multiple objectives, an optimization-based inverse kinematics solution is useful. In addition to COM and swing foot trajectories, the torso angle is desired to be vertical. A vector of feature velocities, $\dot{\mathbf{X}}_f$, is related to the joint velocities, $\dot{\mathbf{q}}$, by a full-body Jacobian,

$$\dot{\mathbf{X}}_f = \mathbf{J}(q)\dot{\mathbf{q}} \tag{7.1}$$

### 7.2.1 Inverse Kinematics

Due to the very complex kinematic structure of the Sarcos robot, analytic inverse kinematic solutions are difficult or impossible to compute. This is especially true for things like desired COM position. For this reason, an optimization-based inverse kinematics algorithm is used. The optimization attempts to minimize the objective function

$$\dot{\mathbf{q}}^* = \arg\min_{\dot{\mathbf{q}}} \left|\left|\dot{\mathbf{X}}_f^{\text{des}} - \mathbf{J}(q)\dot{\mathbf{q}}\right|\right|^2. \tag{7.2}$$

The desired feature velocities, $\dot{\mathbf{X}}_f^{\text{des}}$, can be defined to prevent drift by

$$\dot{\mathbf{X}}_f^{\text{des}} = \dot{\mathbf{X}}_f^{\text{plan}} + \mathbf{K}_{pf}\left(\mathbf{X}_f^{\text{plan}} - \mathbf{X}_f^{\text{int}}\right) \tag{7.3}$$

where $\dot{\mathbf{X}}_f^{\text{plan}}$ and $\mathbf{X}_f^{\text{plan}}$ are determined by the plan and $\mathbf{X}_f^{\text{int}}$ is calculated from the robot state found by integrating $\dot{\mathbf{q}}^*$. $\mathbf{K}_{pf}$ is a gain that controls the rate at which the solution is stabilized to the plan.

To solve this problem, a multi-objective optimization algorithm is used [60]. Objectives in the optimization are concatenated into a linear system of equations

$$\begin{bmatrix} \mathbf{J}_{bL} & \mathbf{J}_{qL} \\ \mathbf{J}_{bR} & \mathbf{J}_{qR} \\ \mathbf{J}_{bCom} & \mathbf{J}_{qCom} \\ \mathbf{J}_{bTor} & \mathbf{J}_{qTor} \\ \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{pmatrix} \dot{\mathbf{x}}_b \\ \dot{\mathbf{q}} \end{pmatrix} = \begin{pmatrix} \dot{\mathbf{X}}_L^{\text{des}} \\ \dot{\mathbf{X}}_R^{\text{des}} \\ \dot{\mathbf{X}}_{Com}^{\text{des}} \\ \dot{\mathbf{X}}_{Tor}^{\text{des}} \\ \dot{\mathbf{q}}^{\text{ref}} \\ \mathbf{0} \end{pmatrix} \tag{7.4}$$

where $\mathbf{J}_b$ is a Jacobian with respect to the floating base coordinates and $\mathbf{J}_q$ is a Jacobian with respect to the joint angle coordinates. $X_L$ and $X_R$ are the positions and orientations of the feet, $X_{Com}$ is the COM, $X_{Tor}$ is the orientation of the torso and $q^{\text{ref}}$ is a static reference pose. This system of equations can be written as

$$\mathbf{A}_{\text{IK}}\dot{\mathbf{x}} = \mathbf{b}_{\text{IK}} \tag{7.5}$$

Before this system is solved, the objectives are weighted by a matrix, $\mathbf{W}_{\text{IK}}$, such that

$$\mathbf{W}_{\text{IK}}\mathbf{A}_{\text{IK}}\dot{\mathbf{x}} = \mathbf{W}_{\text{IK}}\mathbf{b}_{\text{IK}} \tag{7.6}$$

Eq. (7.6) can be converted into a least squares problem

$$\dot{\mathbf{x}}^* = \arg\min_{\dot{\mathbf{x}}} \left|\left|\mathbf{W}_{\text{IK}}\mathbf{A}_{\text{IK}}\dot{\mathbf{x}} - \mathbf{W}_{\text{IK}}\mathbf{b}_{\text{IK}}\right|\right|^2 \tag{7.7}$$

which can be solved using any number of least squares solvers. However, the IK solution should also be constrained to avoid joint limits. There are linear inequality constraints of the form

$$\begin{bmatrix} \mathbf{0} & \mathbf{I}_{lim}T \\ \mathbf{0} & -\mathbf{I}_{lim}T \end{bmatrix} \begin{pmatrix} \dot{\mathbf{x}}_b \\ \dot{\mathbf{q}} \end{pmatrix} + \begin{pmatrix} \mathbf{q} - \mathbf{q}_{\min} \\ -\mathbf{q} + \mathbf{q}_{\max} \end{pmatrix} \geq \mathbf{0} \tag{7.8}$$

where $T$ is the step-size of the IK and $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$ are the minimum and maximum joint limits, respectively. Combined with these constraints, Eq. (7.7) can be solved using quadratic programming. Once a solution, $\dot{\mathbf{x}}^*$, is found, the reference joint positions are updated by integrating with timestep $T$.

In practice, $T$ is set to the timestep of the controller and the IK solution is updated once every timestep. The feedback term in Eq. (7.3) helps to regulate the solution. An alternative approach would be to use a smaller step and perform multiple iterations each timestep in order to exactly solve the IK problem. However, since the IK solution is primarily used to generate reference poses, exact optimal solutions at every timestep are not a priority.

### 7.2.2 Motion Capture Inverse Kinematics

Motion capture can be a very useful tool for programming humanoid robot motions that would otherwise be very difficult to encode by hand. The human doesn't have the same kinematics as the robot, so a reference pose cannot be extracted directly; an intermediate optimization is needed. Luckily, Eq. (7.4) only needs to be modified slightly to perform this optimization. Assuming a set of $M$ motion capture markers corresponding to features on the robot (e.g. shoulders, knees, etc.), an objective can be written to track the markers such

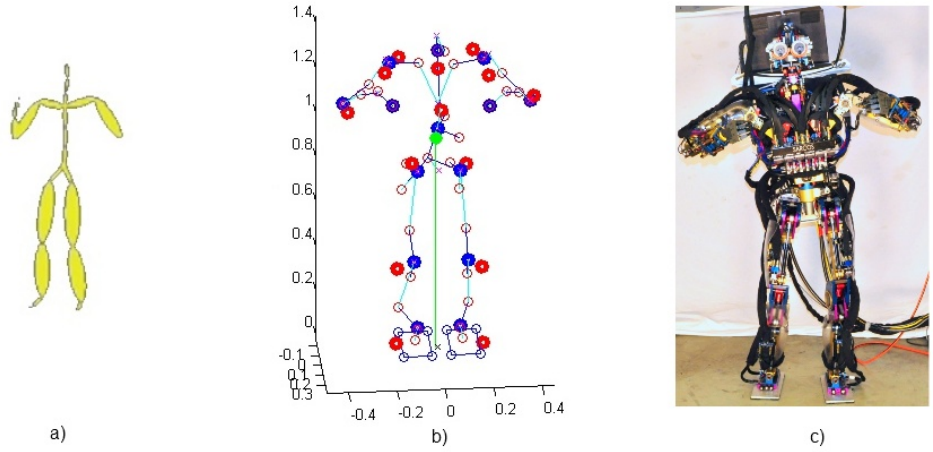


**Figure 7.1:** Implementing motion capture tracking inverse kinematics. a) Processed human motion capture output b) Marker tracking inverse kinematics c) Sarcos humanoid robot dancing.

that

$$\begin{bmatrix} \mathbf{J}_{b0}^{\text{mocap}} & \mathbf{J}_{q0}^{\text{mocap}} \\ \vdots & \vdots \\ \mathbf{J}_{bM}^{\text{mocap}} & \mathbf{J}_{qM}^{\text{mocap}} \end{bmatrix} \begin{pmatrix} \dot{\mathbf{x}}_b \\ \dot{\mathbf{q}} \end{pmatrix} = K_{mocap} \begin{pmatrix} \mathbf{X}_0^{\text{mocap}} - \mathbf{X}_0^{\text{IK}} \\ \vdots \\ \mathbf{X}_M^{\text{mocap}} - \mathbf{X}_M^{\text{IK}} \end{pmatrix} \tag{7.9}$$

where $\mathbf{J}_{bi}^{\text{mocap}}$ and $\mathbf{J}_{qi}^{\text{mocap}}$ are the Jacobians of the features on the robot to which the markers are attached, $X_i^{\text{mocap}}$ are the marker positions from the motion capture, and $X_i^{\text{IK}}$ are the positions of the robot features given the current IK solution.

This method was used to generate a "chicken dance" for the robot, shown in Figure 7.1 (see video at `http://www.cs.cmu.edu/~cga/bstephens/videos/chickendance.wmv`). The motion capture markers positions were obtained from the Carnegie Mellon University Motion Capture Database (see `http://mocap.cs.cmu.edu`).

Some example code is posted at `http://www.cs.cmu.edu/~cga/bstephens/mocapik.zip`.

## 7.3 Weighted-Objective Inverse Dynamics

This section describes the current implementation of the full-body controller on our Sarcos humanoid robot. This method is very similar to the multi-objective controller approach of Abe, *et al.* [2]. As described in Section 5.4, the biped dynamics equations in Eq. (5.16) can be augmented with additional virtual constraints, or objectives. The resulting set of equations can be solved for the joint torques that can realize those objectives. When many objectives are added, the system of equations may become over-constrained, and can be solved using weighted least-squares such as in Eq. (5.19). The weights have to be adjusted to describe the relative importance of the objectives. This weight-tuning is often done manually.

The objectives used in the weighted-objective inverse dynamics include linear and angular momentum regulation (as in Eq. (5.17)), torso angle regulation, torque minimization, and

desired COP control.

$$\begin{bmatrix} \mathbf{M} & -\mathbf{S} & -\mathbf{J}^T \\ \mathbf{J} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_1 \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_2 \\ \mathbf{J_T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D_C} \end{bmatrix} \begin{pmatrix} \ddot{\mathbf{q}} \\ \tau \\ \mathbf{F} \end{pmatrix} = \begin{pmatrix} -\mathbf{N} \\ \dot{\mathbf{P}}_{\text{des}} - \dot{\mathbf{J}}\dot{\mathbf{q}} \\ m\ddot{\mathbf{C}}_{\text{des}} + \mathbf{F}_g \\ \ddot{\mathbf{H}}_{\text{des}} \\ \mathbf{K_{p\theta}}\left(\theta_{\text{des}} - \theta\right) - \mathbf{K_{d\theta}}\dot{\theta} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} \tag{7.10}$$

where $\mathbf{J_T}$ is the Jacobian associated with the torso angle, $\mathbf{K_{p\theta}}$ and $\mathbf{K_{d\theta}}$ are torso gains, and $\mathbf{D_C}$ relates the contact forces to the COP such that

$$\mathbf{D_C} = \begin{bmatrix} 0 & 0 & P_{LX} - COP_{X-des} & 0 & -1 & 0 & 0 & 0 & P_{RX} - COP_{X-des} & 0 & -1 & 0 \\ 0 & 0 & P_{LY} - COP_{Y-des} & 1 & 0 & 0 & 0 & 0 & P_{RY} - COP_{Y-des} & 1 & 0 & 0 \end{bmatrix} \tag{7.11}$$

In order to solve Eq. (7.10), the objectives are weighted by multiplying each side of the equation by a matrix $\mathbf{W_I D}$ that is diagonal in the objective weights such that

$$\mathbf{W_I D} = \text{diag}\left(\left[\mathbf{w_{DYN}}^T, \mathbf{w_{CON}}^T, \mathbf{w_{COM}}^T, \mathbf{w_{ANG}}^T, \mathbf{w_{HIP}}^T, \mathbf{w_{TOR}}^T, \mathbf{w_{COP}}^T\right]\right) \tag{7.12}$$

Each of these weight vectors correspond to the objective rows in Eq. (7.10). Example values are given in Table 7.1. The dynamics and constraints are given the highest weight. When controlling a desired COP, the $x$ and $y$ components of the COM objective are given zero weights. The angular momentum objective can be given a low weight to allow the optimization to sacrifice angular momentum regulation in order to achieve better COM control.

| Weight | Value |
|---|---|
| $\mathbf{w_{DYN}}$ | 1.0 |
| $\mathbf{w_{CON}}$ | 1.0 |
| $\mathbf{w_{COM}}$ | $[0, 0, 1.0e^{-2}]$ |
| $\mathbf{w_{ANG}}$ | $1.0e^{-4}$ |
| $\mathbf{w_{HIP}}$ | $1.0e^{-2}$ |
| $\mathbf{w_{TOR}}$ | $1.0e^{-4}$ |
| $\mathbf{w_{COP}}$ | $1.0e^{-2}$ |

**Table 7.1:** Objective weight values for weighted objective inverse dynamics control.

Given an equation of the form $\mathbf{WAx} = \mathbf{Wb}$, the unknown vector $\mathbf{x}$ can be solved using quadratic programming. QP is used because of constraints on the elements of $\mathbf{x}$ such as COP constraints and torque limits. This QP optimization is run at every timestep, the torques are extracted from $\mathbf{x}$ and are applied to the robot. Generally, low gain PD controls are also added to these torques to stabilize the system and bias towards a reference pose.

Some example code is found at `http://www.cs.cmu.edu/~cga/bstephens/inverse_dynamics.zip`.

## 7.4 Hydraulic Actuator Torque Control

Hardware experiments are performed on a Sarcos humanoid robot. The robot uses linear hydraulic actuators with force feedback to perform compliant torque control on every joint[11].

Power is provided by an off-board pump with tethered hoses that connect to a manifold on the hip. There are potentiometers and force sensors at every joint, an inertial measurement unit (IMU) mounted on the hip, and 6-axis force sensors on each foot.

In this section, a simple controller is presented which is used to control the torque at the joint. This control is made difficult by the fact that the torque is not directly measured nor controlled.

Each joint, $i$, is controlled independently using a valve command, $v_i$, that controls the rate at which hydraulic fluid enters and exits the piston chambers. This valve command is made up of two parts,

$$v_i = v_i^{fb} + v_i^{ff} \tag{7.13}$$

where $v_i^{fb}$ is a feedback term calculated locally on the robot at 5kHz and $v_i^{ff}$ is a feedforward command calculated offboard and communicated to the robot at 400Hz. The local controller does simple linear feedback on the load, $u_i$, at each joint,

$$v_i^{fb} = -K_{ui} u_i \tag{7.14}$$

while the offboard controller adds the necessary command to control the desired load, $u_i^{\text{des}}$. In addition, a positive velocity feedback term is added to cancel the actuator dynamics. The feedforward command is

$$v_i^{ff} = K_{ui} u_i^{\text{des}} + K_{vi} \Psi_i \left( \dot{\theta}_i \right). \tag{7.15}$$

Eq. (7.14) plus Eq. (7.15) results in a proportional controller on the load at the joint. The function $\Psi_i \left( \dot{\theta}_i \right)$ turns joint angular velocity into linear velocity of the piston. This function takes into account the kinematics of the joint and represents the multiplication by

an instantaneous moment arm. Likewise, the desired load is calculated from the desired joint torque by a function, $u = \Phi(\tau)$, and combined with a slow integrator that compensates for any valve bias,

$$u_i^{\text{des}} = \Phi_i \left( \tau_i^{\text{des}} + K_{\tau i} \int \left( \tau_i^{\text{des}} - \Phi_i^{-1}(u_i) \right) dt \right) \tag{7.16}$$

Finally, the desired torque is a combination of the feedforward torques and low gain PD controls,

$$\tau_i^{\text{des}} = \tau_i^{\text{ff}} + K_{pi}(\theta_i^{\text{des}} - \theta_i) + K_{di}(\dot{\theta}_i^{\text{des}} - \dot{\theta}_i) \tag{7.17}$$

## 7.5 Dual Frequency Force Control

This section describes the current implementation of the torque controller on our Sarcos humanoid robot that uses two controllers, one onboard at high frequency and one offboard at low frequency to achieve improved joint-level control.

As described in the previous section, Eq. (7.15) is computed offboard and added to the local robot control signal, computed by Eq. (7.14), to achieve the desired joint force. In our current implementation, the local controller, which runs at a fast 5kHz rate, also includes small position and velocity gains, $K_{pi}$ and $K_{di}$, such that

$$v_i^{fb} = K_{pi}\left(\theta_i^{\text{des}} - \theta_i\right) - K_{di}\dot{\theta}_i - K_{ui}u_i \tag{7.18}$$

The full feedforward command, updated at 400Hz, is computed to cancel these position and velocity gains

$$v_i^{ff} = -K_{pi}\left(\theta_i^{\text{des}} - \theta_i\right) + K_{di}\dot{\theta}_i + K_{ui}u_i^{\text{des}} + K_{vi}\Psi_i\left(\dot{\theta}_i\right). \tag{7.19}$$

The result is still a torque controller, but the high frequency position and velocity gains tend to cancel out high frequency modes giving smoother motions.

## 7.6 Automatic Calibration

The robot also requires constant recalibration and offset adjustment. Calibration is accomplished through a two-step procedure to first correct the kinematic offsets and then the dynamic offsets.

### 7.6.1 Kinematic Offset Calibration

Kinematic offsets include the joint angle zero offsets of the potentiometers at every joint. Calibration is achieved by placing the feet at known positions and orientations relative to each other and solving an inverse kinematics-like problem. The orientation of the hip given by the IMU can also be used in the computation.

### 7.6.2 Dynamic Offset Calibration

There are many dynamic offsets in the system, including the foot force and torque sensors, the joint torque sensors and the unknown external wrench applied by the hydraulic hoses. These offsets can be estimated by fitting the dynamics equations. Ideally, the dynamics have

the form,

$$\begin{bmatrix} \mathbf{M}(\mathbf{q}) & -\mathbf{S} & -\mathbf{J}_C^T & -\mathbf{J}_H^T \\ \mathbf{J}_C & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \ddot{\mathbf{q}} \\ \tau \\ \mathbf{F}_C \\ \mathbf{F}_H \end{pmatrix} = \begin{pmatrix} -\mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) \\ -\dot{\mathbf{J}}_C \dot{\mathbf{q}} \end{pmatrix} \tag{7.20}$$

where $\mathbf{q}$ are the body and joint coordinates, $\mathbf{M}$ is the mass matrix, $\mathbf{S}$ is a selection matrix, $\mathbf{J}_C$ is the contact Jacobian, $\mathbf{J}_H$ is the Jacobian of the hose contact point, $\tau$ are the joint torques, $\mathbf{F}_C$ are the contact forces, $\mathbf{F}_H$ are the hose forces, and $\mathbf{N}$ are the centripetal and coriolis forces.

Assuming offsets of the form, $x = x^{\text{obs}} - x^{\text{off}}$, where $x^{\text{obs}}$ are the observations and $x^{\text{off}}$ are the offsets, Eq. (7.20) can be re-written as

$$\begin{bmatrix} \mathbf{M}(\mathbf{q}) & S & \mathbf{J}_C^T & \mathbf{J}_H^T \\ \mathbf{J}_C & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \ddot{q} \\ \tau^{\text{off}} \\ \mathbf{F}_C^{\text{off}} \\ \mathbf{F}_H^{\text{off}} \end{pmatrix} = \begin{pmatrix} -\mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{S}\tau^{\text{obs}} + \mathbf{J}_C^T \mathbf{F}_C^{\text{obs}} \\ -\dot{\mathbf{J}}_\mathbf{C} \dot{\mathbf{q}} \end{pmatrix} \tag{7.21}$$

Because the dynamics are linear in these offsets, this equation can be easily solved by a weighted least-squares of the form

$$\left(\mathbf{A}^T \mathbf{A} + diag(w)\right) \mathbf{z} = \mathbf{A}^T \mathbf{b} \tag{7.22}$$

where $diag(w)$ is a diagonal matrix of weights that bias each offset to zero. $\ddot{\mathbf{q}}$ is included in Eq. (7.21) to ensure constraint satisfaction. If the robot is static for the calibration, the weights associated with $\ddot{\mathbf{q}}$ can be large to make sure the solution results in zero acceleration. Likewise, if an acceleration estimate is known it can also be used.

In practice, the robot is balanced in place before this calibration is initiated. Eq. (7.21) is solved at each timestep using Eq. (7.22) for a short period (100 timesteps) and the solutions are averaged and the offsets held constant throughout the experiment. This calibration is repeated during every trial.

## 7.7 Conclusion

This chapter provided an overview of algorithms specific to whole body control of the Sarcos Primus humanoid robot.

# Chapter 8

# Conclusion

This thesis presents applications of simple models of biped dynamics for use in force-based balance and push recovery. These simple models are used for control, planning and estimation of complex force-controlled humanoid robots performing push recovery and other tasks. The primary problem addressed is **controlling push recovery for full-body force-controlled humanoid robots**. Simple models in terms of generalized quantities such as the COM and COP were analyzed to determine strategies for recovering balance. These strategies are found to be highly dependent on the constraints on the allowable ground reaction forces. To compensate for these strict constraints, a model predictive controller, PR-MPC, is used to **reason about future actions and the effects of constraints** in order to bring the system to rest. It was also demonstrated how this optimization can be tuned to control both step recovery and walking. These simple models allow the robot to **continuously estimate the state of the system and correct its balance** even in the presence of large disturbances. Finally, push recovery behaviors are implemented on the Sarcos humanoid

robot using a low-level force controller which mapped the controller for the simple model to full-body joint torques. The combination of this model-based force control and low gain reference motion tracking allows the robot to achieve greater compliance during interaction with other objects and the environment.

## 8.1 Push Recovery Comparison

Push recovery performance is compared in this section. Building on the comparison presented in Table 8.1, performance from robot experiments and human biomechanics literature are also compared. Since the masses of the systems are different, the maximum impulses are divided by the total mass. The result is the effective change in velocity due to the impulse. In some cases, the push force was not directly measured, so the change in velocity due to the push is estimated directly by inspecting the data. Data for this comparison was taken from Sarcos robot experiments not specifically designed to test the maximum push the robot could handle.

The pushes handled by the robot are significantly less than those handled by the simulation and predicted by the model. In several cases, this is a result of the controller being overly conservative in executing either a hip or stepping strategy. To be conservative, the controller assumes smaller than actual size feet. This causes stepping and hip strategies to be executed for smaller pushes. In practice, using such a conservative constraint helps keep the COP from hitting the edge of the foot, where rotation, especially in the yaw direction, is likely to occur.

| Strategy | Model [1] (m/s) | Simulation [1] (m/s) | Robot [5] (m/s) | Human (m/s) |
|---|---|---|---|---|
| Ankle | 0.31 [2] | 0.25 | 0.23 | 0.24 [8] |
| Hip | 0.43 [3] | 0.38 | 0.14 [6] | -- |
| 1-step | 0.54 [4] | 0.40 | 0.24 [6],[7] | -- |
| 2-step | 0.58 [4] | 0.58 | 0.24 [6],[7] | -- |

**Table 8.1:** [1]The model used in simulation is different, $m = 72kg$, [2]Using a $0.1m$ distance to the toe, $L = 1.0m$, [3]$\tau_{\max} = 40Nm$, $I = 4.0kg \cdot m^2$, $\theta_{\max} = 0.5rad$, [4]Computed by running PR-MPC with increasing initial velocity until the system goes unstable. Shift time $= 0.1s$, Step time $= 0.4s$, max stance width $= 0.5m$ [5]The robot has total mass of $95kg$, COM height of approximately $0.95m$, and $0.1m$ distance to the toe [6]These robot experiments were performed assuming the foot was half-size to ensure it did not rotate off the ground, so these numbers appear smaller [7]Estimated from data [8]Estimated from figures in (Pai, et al. 1997)

There are other likely causes for the controller being unable to handle larger pushes. Stepping for push recovery requires high performance swing leg control because the swing foot has to hit a target in a very short amount of time. High performance swing leg control can be difficult to achieve due to a poor dynamics model or slipping of the stance foot. Another limitation of the PR-MPC controller as presented is that it doesn't consider angular momentum. Large pushes to the torso can induce significant angular momentum requiring an adjustment of desired step location. This lack of angular momentum compensation coupled with poor swing foot control limited success recovering from large pushes.

The biomechanics literature is often concerned with predicting whether a step should

occur or not using simple dynamic models [77][90][89][81]. The simple one-link dynamic model from [90] was used to generate the ankle strategy limit in Table 8.1. The reason that this value is less than that predicted by the simple model from Section 3.3.1 is likely due to the inclusion of friction, joint torque limits, and muscle force limits. This biomechanics model was used to successfully predict stepping in human trials [89]. While many hip strategy experiments have been performed on humans, maximum push or perturbation sizes have not been reported. In the case of stepping, because it is difficult to constrain a person to take only one or only two steps, this data is also not available.

## 8.2 Discussion of Future Work

This thesis has presented methods for simplifying the design of very basic behaviors. Because of the complexity of these systems, implementation can still be challenging and tedious. Future applications will require **automatic generation of stable behaviors**. Not only should the developmental tools for programming and diagnosing behaviors be improved, but the controllers themselves should be able to generate new behaviors to adapt to new situations. In addition, controllers should improve performance and efficiency of behaviors through experience.

Not only will humanoid robot controllers need to improve, but the hardware and system integration need to be improved to allow **robots capable of real world application**. There are currently few examples of humanoid robots performing valuable tasks outside of the research laboratory. One of the most useful applications of humanoid robots is in

dangerous or otherwise unreachable scenarios, for example repairing a nuclear power plant or traveling to distant planets. These tasks will require robust and efficient systems capable enough to withstand abuse and repair or reconfigure themselves if needed.

Another application that has yet to be realized is the **testing of biomechanical models of humans**. Currently there is some effort by roboticists to use existing biomechanical models and observations of humans to inform the design of humanoid robot hardware and control. However, there is very little effort to use these results to inform and assist the efforts of the physiological fields. A greater focus on such two-way collaboration will lead to accelerated development of robust human-like control for robots and new tools and devices for evaluating and assisting humans.

Like all forms of automation, there are concerns surrounding not just the technological advances. Humanoid robot researchers of the future should be trained in the **sociological and ethical issues** raised by their work. A humanoid robot is perhaps theoretically capable of any task a person is, and perhaps at the same time capable of greater performance. Exciting as this may be, caution is needed by researchers and policy makers. Are humanoid robots going to replace factory workers? Will they replace soldiers and fight wars for us? Will we use them as avatars to live our daily lives? Who is to blame if the robot causes harm? In the end there will be some tasks for which a humanoid robot may be better suited and some tasks for which it is not. The question will ultimately be how to make safe and effective use of these systems while doing so ethically with responsible foresight of the consequences.

# Bibliography


[1] http://eigen.tuxfamily.org.

[2] Y. Abe, M. da Silva, and J. Popovic. Multiobjective control with frictional contacts. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 249–258, 2007.

[3] Y. Abe, C. K. Liu, and Z. Popovic. Momentum-based Parameterization of Dynamic Character Motion. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 194–211, 2004.

[4] K. Akachi, K. Kaneko, N. Kanehira, S. Ota, G. Miyamori, M. Hirata, S. Kajita, and F. Kanehiro. Development of humanoid robot HRP-3P. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 50–55, 2005.

[5] C. An, C. Atkeson, and J. Hollerbach. *Model-based control of a robot manipulator*. MIT Press, Cambridge, Mass., 1988.

[6] S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of Robots by learning. *Journal of Robotic Systems*, 1(2):123–140, Jan. 1984.

[7] C. Atkeson and J. McIntyre. Robot trajectory learning through practice. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1737–1742. Institute of Electrical and Electronics Engineers, 1986.

[8] C. Atkeson and B. Stephens. Multiple Balance Strategies from One Optimization Criterion. In *The IEEE-RAS 2007 International Conference on Humanoid Robots*, 2007.

[9] C. E. Bauby and A. D. Kuo. Active control of lateral balance in human walking. *Journal of Biomechanics*, 33:1433–1440, 2000.

[10] R. Bellman. Dynamic Programming. *Science*, 153:34–37, 1966.

[11] D. C. Bentivegna and C. G. Atkeson. Compliant control of a hydraulic humanoid joint. *Proceedings of the International Conference on Humanoid Robots*, pages 483–489, Nov. 2007.

[12] G. N. Boone and J. K. Hodgins. Slipping and Tripping Reflexes for Bipedal Robots. *Autonomous Robots*, 4(3):259 – 271, 1997.

[13] C. I. Byrnes and C. F. Martin. Global observability and detectability: An overview. *Modelling and Adaptive Control*, 1988.

[14] G. Cheng, S.-h. Hyon, J. Morimoto, A. Ude, G. Colvin, W. Scroggin, and S. Jacobsen. CB: A Humanoid Research Platform for Exploring NeuroScience. *International Conference on Humanoid Robots*, pages 182–187, Dec. 2006.

[15] J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade. Footstep Planning for the Honda ASIMO Humanoid. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Apr. 2005.

[16] J. Chestnutt, Y. Takaoka, M. Doi, K. Suga, and S. Kagami. Safe Adjustment Regions for Legged Locomotion Paths. In *Humanoid Robots*, 2010.

[17] S. H. Collins, M. Wisse, and A. Ruina. A Three-Dimensional Passive-Dynamic Walking Robot with Two Legs and Knees. *The International Journal of Robotics Research*, 20:607–615, 2001.

[18] S. Coros, P. Beaudoin, and M. van de Panne. Generalized biped walking control. *ACM Transactions on Graphics (TOG)*, 29(4):1–9, 2010.

[19] M. da Silva, Y. Abe, and J. Popović. Simulation of Human Motion Data using Short-Horizon Model-Predictive Control. *Computer Graphics Forum*, 27(2):371–380, Apr. 2008.

[20] M. de Lasa, I. Mordatch, and A. Hertzmann. Feature-Based Locomotion Controllers. *ACM Transactions on Graphics*, 29(3), 2010.

[21] H. Diedam, D. Dimitrov, P.-B. Wieber, K. Mombaur, and M. Diehl. Online Walking Gait Generation with Adaptive Foot Positioning Through Linear Model Predictive Control. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1121–1126. IEEE, 2008.

[22] D. Dimitrov, P.-B. Wieber, H. J. Ferreau, and M. Diehl. On the implementation of model predictive control for on-line walking pattern generation. *2008 IEEE International Conference on Robotics and Automation*, pages 2685–2690, May 2008.

[23] D. Dimitrov, P.-B. Wieber, O. Stasse, H. Ferreau, and H. Diedam. An optimized Linear Model Predictive Control solver for online walking motion generation. *2009 IEEE International Conference on Robotics and Automation*, (1):1171–1176, May 2009.

[24] S. Eppinger and W. Seering. Understanding bandwidth limitations in robot force control. In *Proceedings of the International Conference on Robotics and Automation*, pages 904–909. Institute of Electrical and Electronics Engineers, 1987.

[25] A. C. Fang and N. S. Pollard. Efficient synthesis of physically valid human motion. In *Proceedings of ACM SIGGRAPH*, volume 22, pages 417–426, 2003.

[26] Y. Fujimoto and A. Kawamura. Proposal of biped walking control based on robust hybrid position/force control. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2724–2730. IEEE, 1996.

[27] K. Fujiwara, S. Kajita, K. Harada, K. Kaneko, M. Morisawa, F. Kanehiro, S. Nakaoka, and H. Hirukawa. An optimal planning of falling motions of a humanoid robot. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 456–462. IEEE, 2007.

[28] R. J. Full and D. E. Koditschek. Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *The Journal of experimental biology*, 202(Pt 23):3325–32, Dec. 1999.

[29] M. Garcia, A. Chatterjee, A. Ruina, and M. Coleman. The Simplest Walking Model: Stability, Complexity, and Scaling. *Journal of Biomechanical Engineering*, 120:281–288, 1998.

[30] L. D. Gaspero. QuadProg++.

[31] H. Geyer, A. Seyfarth, and R. Blickhan. Compliant leg behaviour explains basic dynamics of walking and running. *Proceedings. Biological sciences / The Royal Society*, 273(1603):2861–7, Nov. 2006.

[32] M. Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42, 1998.

[33] R. Goddard, H. Hemami, and F. Weimer. Biped side step in the frontal plane. *IEEE Transactions on Automatic Control*, 28:179–187, Feb. 1983.

[34] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27(1):1–33, Sept. 1983.

[35] C. Golliday and H. Hemami. Postural stability of the two-degree-of-freedom biped by general linear feedback. *Automatic Control, IEEE Transactions on*, 21(1), 1976.

[36] H. Gomi and M. Kawato. Learning control for a closed loop system using feedback-error-learning. In *Proceedings of the Conference on Decisions and Control*, volume 6, pages 3289 – 3294, 1990.

[37] A. Goswami. Postural Stability of Biped Robots and the Foot-Rotation Indicator (FRI) Point. *The International Journal of Robotics Research*, 18(6):523–533, June 1999.

[38] A. Goswami and V. Kallem. Rate of change of angular momentum and balance maintenance of biped robots. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, volume 4, pages 3785–3790, Honda Res. Inst., Mountain View, CA, USA, 2004.

[39] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. *ACM Transactions on Graphics*, 23(3):522, Aug. 2004.

[40] K. Harada, S. Kajita, K. Kaneko, and H. Hirukawa. Zmp analysis for arm/leg coordination. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 75–81, 2003.

[41] J. He, W. Levine, and G. Loeb. Feedback gains for correcting small perturbations to standing posture. In *Proceedings of the 28th IEEE Conference on Decision and Control*, pages 518–526. IEEE, 1989.

[42] H. Hemami and P. Camana. Nonlinear feedback in simple locomotion systems. *IEEE Transactions on Automatic Control*, 21:855–860, Dec. 1976.

[43] H. Hemami, F. Weimer, and S. Koozekanani. Some aspects of the inverted pendulum problem for modeling of locomotion systems. *IEEE Transactions on Automatic Control,*, 18:658–661, Dec. 1973.

[44] A. Herdt, N. Perrin, and P.-B. Wieber. Walking without thinking about it. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 190–195. IEEE, Oct. 2010.

[45] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of Honda humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1321–1326, May 1998.

[46] D. G. E. Hobbelen and M. Wisse. A Disturbance Rejection Measure for Limit Cycle Walkers: The Gait Sensitivity Norm. *IEEE Transactions on Robotics*, 23:1213–1224, 2007.

[47] A. Hofmann. *Robust Execution of Bipedal Walking Tasks From Biomechanical Principles*. PhD thesis, MIT, Jan. 2006.

[48] F. B. Horak and L. M. Nashner. Central programming of postural movements: adaptation to altered support-surface configurations. *Journal of Neurophysiology*, 55:1369–1381, 1986.

[49] Y. Hurmuzlu, F. Genot, and B. Brogliato. Modeling, stability and control of biped robots - a general framework. *Automatica*, 40:1647–1664, 2004.

[50] S.-H. Hyon, J. G. Hale, and G. Cheng. Full-Body Compliant Human Humanoid Interaction: Balancing in the Presence of Unknown External Forces. *IEEE Transactions on Robotics*, 23:884–898, Oct. 2007.

[51] P. Ill-Woo, K. Jung-Yup, L. Jungho, and O. Jun-Ho. Mechanical design of humanoid robot platform KHR-3 (KAIST humanoid robot - 3: HUBO). In *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, volume 12, pages 321–326. IEEE, 2005.

[52] D. H. Jacobson and D. Q. Mayne. *Differential dynamic programming.* American Elsevier Pub. Co., New York, 1970.

[53] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point. In *Proceedings fo the International Conference on Robotics and Automation*, pages 1620–1626, Taipei, Taiwan, Sept. 2003.

[54] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Resolved momentum control: humanoid motion planning based on the linear and angular momentum. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, 2003.

[55] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In *Intelligent Robots and Systems, Proceedings. IEEE/RSJ International Conference on*, volume 1, pages 239–246. IEEE, 2001.

[56] S. Kajita and K. Tani. Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1405–1411, Apr. 1991.

[57] S. Kajita, T. Yamaura, and A. Kobayashi. Dynamic walking control of a biped robot along a potential energy conserving orbit. *IEEE Transactions on Robotics and Automation*, 8:431–438, Aug. 1992.

[58] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASMEJournal of Basic Engineering*, pages 35–45, 1960.

[59] K. Kaneko, F. Kanehiro, S. Kajita, K. Yokoyama, K. Akachi, T. Kawasaki, S. Ota, and T. Isozumi. Design of prototype humanoid robotics platform for HRP. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2431–2436, 2002.

[60] O. Kanoun, F. Lamiraux, P.-B. Wieber, F. Kanehiro, E. Yoshida, and J.-P. Laumond. Prioritizing linear equality and inequality systems: Application to local motion planning for redundant robots. *2009 IEEE International Conference on Robotics and Automation*, (2):2939–2944, May 2009.

[61] C. Klein and S. Kittivatcharapong. Optimal force distribution for the legs of a walking machine with friction cone constraints. *Robotics and Automation, IEEE Transactions on*, 6(1):73–85, 1990.

[62] T. Komura, A. Nagano, H. Leung, and Y. Shinagawa. Simulating pathological gait using the enhanced linear inverted pendulum model. *IEEE Transactions on Biomedical Engineering*, 52(9):1502–13, 2005.

[63] K. Kondak and G. Hommel. Control and online computation of stable movement for biped robots. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 1, pages 874–879. IEEE, 2003.

[64] S. Kudoh and T. Komura. C^2 continuous gait-pattern generation for biped robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1135–1140, Oct. 2003.

[65] S. Kudoh, T. Komura, and K. Ikeuchi. The dynamic postural adjustment with the quadratic programming method. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2563–2568, 2002.

[66] S. Kudoh, T. Komura, and K. Ikeuchi. Stepping motion for a human-like character to maintain balance against large perturbations. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2661–2666, 2006.

[67] V. Kumar and K. Waldron. Force distribution in closed kinematic chains. *Robotics and Automation, IEEE Journal of*, 4(6):657–664, 1988.

[68] A. Kuo. An optimal control model for analyzing human postural balance. *IEEE Transactions on Biomedical Engineering*, 42(1):87–101, 1995.

[69] A. Kuo. Stabilization of Lateral Motion in Passive Dynamic Walking. *International Journal of Robotics Research*, 18:917–930, 1999.

[70] A. D. Kuo. Energetics of Actively Powered Locomotion Using the Simplest Walking Model. *Journal of Biomechanical Engineering*, 124:113–120, 2002.

[71] S.-H. Lee and A. Goswami. Reaction Mass Pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots. In *IEEE International Conference on Robotics and Automation*, pages 4667–4672, Apr. 2007.

[72] W. Li and E. Todorov. Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems. In *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics*, pages 222–229, 2004.

[73] C. Liu and C. G. Atkeson. Standing balance control using a trajectory library. In *International Conference on Intelligent Robots and Systems*, pages 3031–3036, 2009.

[74] C. Liu, C. G. Atkeson, and J. Su. Biped walking control using a trajectory library. *submitted to Robotica*, 2010.

[75] C. Liu, C. G. Atkeson, and J. Su. Neighboring optimal control for periodic tasks for systems with discontinuous dynamics. *SCIENCE CHINA Information Sciences*, 54(3):653–663, 2010.

[76] A. Macchietto, V. Zordan, and C. R. Shelton. Momentum control for balance. *ACM Transactions on Graphics*, 28(3):1, 2009.

[77] B. E. Makai and W. E. McIlroy. The role of limb movements in maintaining upright stance: the "change-in-support" strategy. *Physical Therapy*, 77:488–507, May 1997.

[78] I. R. Manchester, U. Mettin, F. Iida, and R. Tedrake. Stable dynamic walking over uneven terrain. *The International Journal of Robotics Research*, 30(3):265–279, Jan. 2011.

[79] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[80] T. McGeer. Passive Dynamic Walking. *The International Journal of Robotics Research*, 9:62–82, 1990.

[81] M. L. Mille, M. W. Rogers, K. Martinez, L. D. Hedman, M. E. Johnson, S. R. Lord, and R. C. Fitzpatrick. Thresholds for Inducing Protective Stepping Responses to External Perturbations of Human Standing. *Journal of Neurophysiology*, 90:666–674, Apr. 2003.

[82] M. Mistry, J. Buchli, and S. Schaal. Inverse Dynamics Control of Floating Base Systems using Orthogonal Decomposition. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2010.

[83] M. Mistry, J. Nakanishi, G. Cheng, and S. Schaal. Inverse kinematics with floating base and constraints for full body humanoid robot control. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2008.

[84] I. Mordatch, M. de Lasa, and A. Hertzmann. Robust physics-based locomotion using low-dimensional planning. *ACM Transactions on Graphics (TOG)*, 29(4), 2010.

[85] K. Muske and T. Badgwell. Disturbance modeling for offset-free linear model predictive control. *Journal of Process Control*, 12(5):617–632, Aug. 2002.

[86] K. R. Muske and J. B. Rawlings. Model predictive control with linear models. *AIChE Journal*, 39(2):262–287, Feb. 1993.

[87] K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue. Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired ZMP. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System*, volume 3, pages 2684–2689, 2002.

[88] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[89] Y. C. Pai, B. E. Maki, K. Iqbal, W. E. McIlroy, and S. D. Perry. Thresholds for step initiation induced by support-surface translation: a dynamic center-of-mass model provides much better prediction than a static model. *Journal of biomechanics*, 33(3):387–92, Mar. 2000.

[90] Y. C. Pai and J. L. Patton. Center of mass velocity-position predictions for balance control. *Journal of Biomechanics*, 30:347–354, 1997.

[91] G. Pannocchia and J. B. Rawlings. Disturbance models for offset-free model-predictive control. *AIChE Journal*, 49(2):426–437, Feb. 2003.

[92] J. H. Park. Impedance Control for Biped Robot Locomotion. *IEEE Transactions on Robotics and Automation*, 17:870–881, Dec. 2001.

[93] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal. A unifying framework for robot control with redundant DOFs. *Autonomous Robots*, 24(1):1–12, 2007.

[94] M. Popovic, A. Hofmann, and H. Herr. Angular momentum regulation during human walking: biomechanics and control. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 3, pages 2405–2411. IEEE, 2004.

[95] M. B. Popovic, A. Goswami, and H. Herr. Ground Reference Points in Legged Locomotion: Definitions, Biological Trajectories and Control Implications. *The International Journal of Robotics Research*, 24:1013–1032, 2005.

[96] G. Pratt and M. Williamson. Series elastic actuators. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 399–406. IEEE Comput. Soc. Press, 1995.

[97] J. Pratt, J. Carff, S. Drakunov, and A. Goswami. Capture Point: A Step toward Humanoid Push Recovery. In *Proceedings of the International Conference on Humanoid Robots*, pages 200–207. IEEE, Dec. 2006.

[98] J. Pratt, P. Dilworth, and G. Pratt. Virtual model control of a bipedal walking robot. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1:193–198, Apr. 1997.

[99] J. Pratt, B. Krupp, and C. Morse. Series elastic actuators for high fidelity force control. *Industrial Robot: An International Journal*, 29(3):234–241, 2002.

[100] J. Pratt and G. Pratt. Intuitive control of a planar bipedal walking robot. In *IEEE International Conference on Robotics and Automation*, pages 2014–2021, Leuven, Belgium, 1998.

[101] J. Pratt and R. Tedrake. Velocity-Based Stability Margins for Fast Bipedal Walking. In *Fast Motions in Biomechanics and Robotics*, pages 299–324. Springer Berlin / Heidelberg, Heidelberg Germany, Sept. 2006.

[102] J. Pratt, A. Torres, P. Dilworth, and G. Pratt. Virtual Actuator Control. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1219–1226, Osaka, Japan, 1996.

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

[104] M. H. Raibert. *Legged robots that balance*. Massachusetts Institute of Technology, Cambridge, MA, USA, 1986.

[105] J. Rebula, F. Canas, J. Pratt, and A. Goswami. Learning Capture Points for Bipedal Push Recovery. In *Proceedings of the IEEE International Conference on Robotics and Automation*, page 1774, 2008.

[106] M. S. Redfern and T. Schumann. A model of foot placement during gait. *Journal of Biomechanics*, 27(11):1339–1346, Nov. 1994.

[107] A. Safonova. *Reducing the search space for physically realistic human motion synthesis*. PhD thesis, Carnegie Mellon University, 2006.

[108] L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pages 2641–2648, 2006.

[109] L. Sentis, J. Park, and O. Khatib. Modeling and Control of Multi-Contact Centers of Pressure and Internal Forces in Humanoid Robots. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[110] K. W. Sok, M. Kim, and J. Lee. Simulating Biped Behaviors from Human Motion Data. In *International Conference on Computer Graphics and Interactive Techniques*, 2007.

[111] M. Spong. Swing up control of the Acrobot. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, volume 46, pages 2356–2361. IEEE Comput. Soc. Press, 1994.

[112] B. Stephens. Humanoid Push Recovery. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2007.

[113] B. Stephens. Integral Control of Humanoid Balance. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2007.

[114] B. J. Stephens. State Estimation for Force-Controlled Humanoid Balance using Simple Models in the Presence of Modeling Error. In *IEEE International Conference on Robotics and Automation*, 2011.

[115] B. J. Stephens and C. G. Atkeson. Dynamic Balance Force Control for Compliant Humanoid Robots. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2010.

[116] B. J. Stephens and C. G. Atkeson. Push Recovery by Stepping for Humanoid Robots with Force Controlled Joints. In *Proceedings of the International Conference on Humanoid Robots*, 2010.

[117] M. Stilman and J. Kuffner. Navigation among movable obstacles: real-time reasoning in complex environments. In *Proceedings of the IEEE/RAS International Conference on Humanoid Robots*, volume 1, pages 322–341, 2004.

[118] M. Stolle and C. G. Atkeson. Policies based on trajectory libraries. In *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pages 3344–3349, 2006.

[119] M. Stolle, H. Tappeiner, J. Chestnutt, and C. G. Atkeson. Transfer of policies based on trajectory libraries. In *Proceedings of the IEEE/RSJ International Confrerence on Intelligent Robots and Systems*, pages 2981–2986, 2007.

[120] T. Sugihara. Standing stabilizability and stepping maneuver in planar bipedalism based on the best COM-ZMP regulator. *2009 IEEE International Conference on Robotics and Automation*, pages 1966–1971, May 2009.

[121] T. Sugihara and Y. Nakamura. Whole-body cooperative balancing of humanoid robot using COG Jacobian. In *Proceedings of the International Conference on Intelligent Robots and Systems*, number October, pages 0–5, 2002.

[122] A. Takanishi, T. Takeya, H. Karaki, and I. Kato. A control method for dynamic biped walking under unknown external force. In *IEEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, volume 29, pages 795–801. IEEE, 1990.

[123] Y. Tassa and T. Erez. Receding horizon differential dynamic programming. In *Advances in Neural Information Processing Systems*, 2007.

[124] E. Todorov and W. Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference, Proceedings of the*, pages 300–306. IEEE, 2005.

[125] E. Todorov and Y. Tassa. Iterative Local Dynamic Programming. In *Proceedings of the 2nd IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 90–95, 2009.

[126] M. A. Townsend. Biped gait stabilization via foot placement. *Journal of Biomechanics*, 18(1):21–38, 1985.

[127] Y.-Y. Tsai, W.-C. Lin, K. B. Cheng, J. Lee, and T.-Y. Lee. Real-Time Physics-Based 3D Biped Character Animation Using an Inverted Pendulum Model. *IEEE transactions on visualization and computer graphics*, 16(2):325–37, 2010.

[128] M. Vukobratovic. How to Control Artificial Anthropomorphic Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(5):497–507, 1973.

[129] M. Vukobratovic, A. A. Frank, and D. Juricic. On the stability of biped locomotion. *IEEE Transactions on Biomedical Engineering*, pages 25–36, Jan. 1970.

[130] K. Waldron. Force and motion management in legged locomotion. *Robotics and Automation, IEEE Journal of*, 2(4):214–220, 1986.

[131] E. Whitman and C. Atkeson. Control of Instantaneously Coupled Systems Applied to Humanoid Walking. In *Proceedings of the IEEE International Conference on Humanoid Robots*, 2010.

[132] P.-B. Wieber. Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 137–142, 2006.

[133] D. L. Wight, E. G. Kubica, and D. W. L. Wang. Introduction of the Foot Placement Estimator: A Dynamic Measure of Balance for Bipedal Robotics. *Journal of Computational and Nonlinear Dynamics*, 3:11009, 2008.

[134] D. J. Wiley and J. K. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17:39–45, 1997.

[135] D. Williams and O. Khatib. The virtual linkage: a model for internal forces in multi-grasp manipulation. In *IEEE International Conference on Robotics and Automation*, 1993.

[136] D. A. Winter. Human balance and posture control during standing and walking. *Gait and Posture*, 3:193–214, Dec. 1995.

[137] M. Wisse. *Essentials of dynamic walking; Analysis and design of two-legged robots*. PhD thesis, 2004.

[138] K. Yamane and Y. Nakamura. Dynamics Filter - concept and implementation of online motion Generator for human figures. *IEEE Transactions on Robotics and Automation*, 19:421–432, June 2003.

[139] K. Yin, K. Loken, and M. van de Panne. SIMBICON: simple biped locomotion control. *ACM Transactions on Graphics*, 26(3):105, 2007.

[140] K. Yin, D. K. Pai, and M. van de Panne. Data-driven Interactive Balancing Behaviors. *Pacific Graphics*, Oct. 2005.

[141] K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, and K. Tanie. Planning walking patterns for a biped robot. *IEEE Transactions on Robotics and Automation*, 17(3):280–289, June 2001.

[142] E. Yoshida, O. Kanoun, C. Esteves, and J.-P. Laumond. Task-driven Support Polygon Reshaping for Humanoids. In *IEEE-RAS International Conference on Humanoid Robots*, pages 208–213, 2006.

[143] S.-k. Yun, A. Goswami, and Y. Sakagami. Safe Fall: Humanoid robot fall direction change through intelligent stepping and inertia shaping. In *IEEE International Conference on Robotics and Automation*, pages 781–787, 2009.