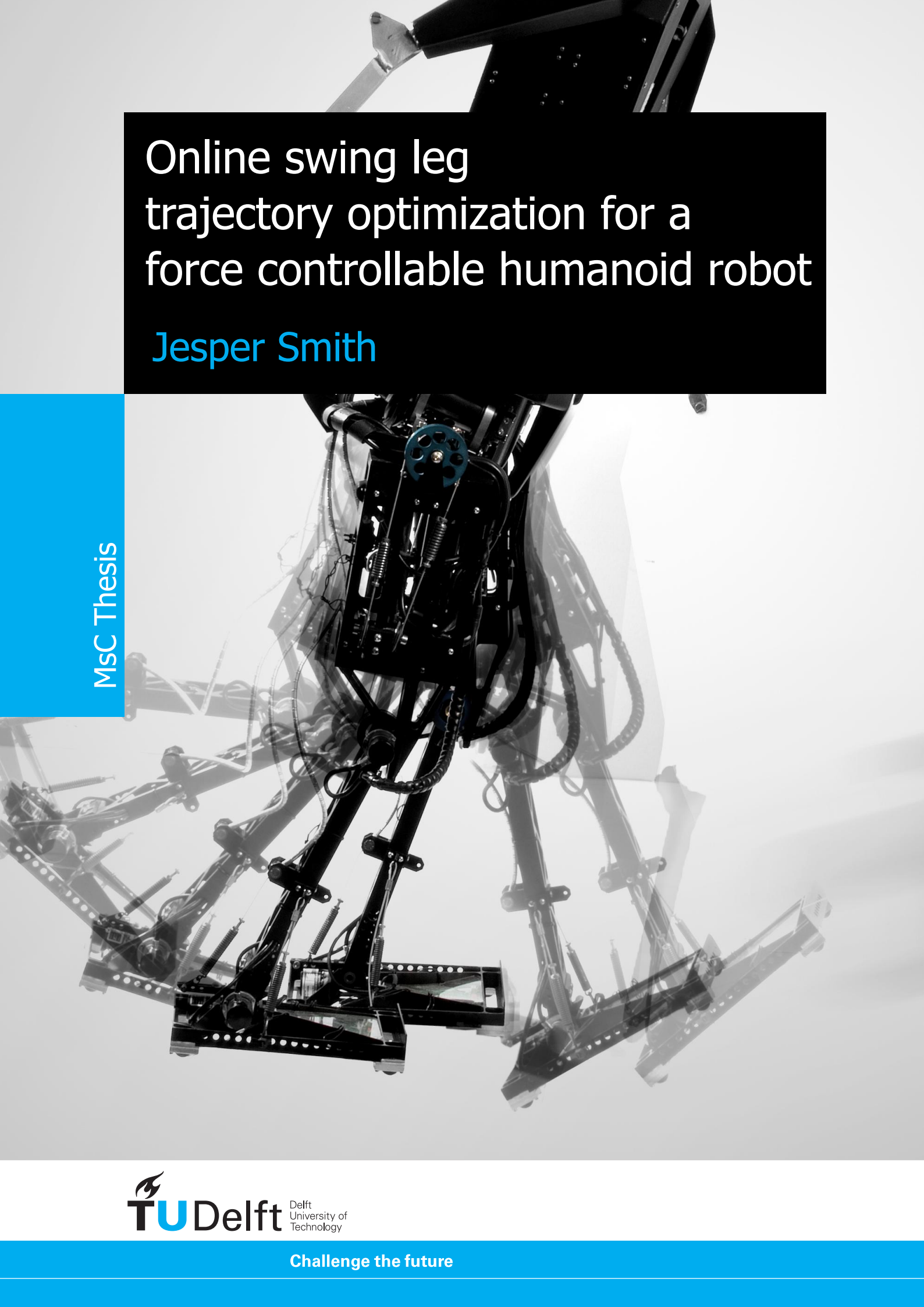




Online swing leg trajectory optimization for a force controllable humanoid robot

Jesper Smith

MSc Thesis

Online swing leg trajectory optimization for a force controllable humanoid robot

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Mechanical
Engineering at Delft University of Technology

Jesper Smith

August 20, 2012

Abstract

Humanoid robots are well suited to function in a world designed for humans. To function autonomously, it is important that the robot be robust against disturbances. Quick and accurate positioning of the swing foot allows the robot to recover from large disturbances. This thesis presents a novel controller that provides accurate foot positioning capability while reducing control effort compared to the existing swing leg controller. The presented controller uses model predictive control techniques to optimize the joint trajectories. Two cost functions are compared in a simulation study, the minimum of the sum of squared torques and the minimum sum of squared torque changes. The minimum torque change cost function has a 20% lower mechanical energy cost compared to the minimum torque cost function, but suffers from a significantly larger positioning error at the end of the step and reduced ground clearance. Compared to the existing controller, the proposed model predictive controller with a minimum torque cost function decreases the mechanical energy requirements by approximately 35% for a forward step while improving the accuracy. Negative work done by the actuator to decelerate the foot at the end of the swing is significantly reduced compared to the foot trajectory controller. The main disadvantage of the proposed model predictive controller is high computational cost. Therefore, a secondary controller is introduced, which plans joint trajectories using quintic splines through a number of via-points. Mechanical energy consumption during the swing is comparable to the existing controller, but bandwidth requirements are reduced. In conclusion, the proposed model predictive controller with a minimum torque cost function is able to quickly and accurately position the foot, while reducing total energy consumption by exploiting the natural dynamics of the swing leg.

Table of contents

Introduction	1
1 Background	3
1.1 Swing foot position control for various walking algorithms	3
1.1.1 Joint position control	5
1.2 Actuators	5
1.3 Conclusion	6
2 Model	7
2.1 Introduction	7
2.2 Kinematic Model	8
2.2.1 Forward kinematics	9
2.2.2 Inverse kinematics	13
2.3 Conclusion	18
3 Capture Point estimation	19
3.1 Introduction	19
3.2 Center of Mass Position	20
3.3 Estimating Center Of Mass Velocity	20
3.3.1 Sensor data	20
3.3.2 Kalman State Estimation	21
3.3.3 From IMU velocity to CoM velocity	24
3.4 Conclusion	25
4 Model Predictive Swing Controller	26
4.1 Introduction	26
4.2 Predicting final desired swing location	27
4.2.1 Predicting the pelvis position at the end of the step	27
4.3 Joint trajectory parametrization	29
4.4 Cost function	30
4.4.1 Minimum torque	30
4.4.2 Minimum torque change	30
4.5 Optimization algorithm	31
4.6 Constraints	31
4.6.1 Final joint angles and velocities	31
4.6.2 Joint position, velocity and torque limits	32
4.6.3 Ground clearance	33
4.7 Jacobian of the cost function	33

4.8	Initial values	34
4.9	Numerical considerations	34
4.10	Computational considerations	35
4.11	Conclusion	35
5	Minimum jerk controller	36
5.1	Trajectory parametrization	36
5.2	Choosing the Via-Points	37
5.3	Adjusting the trajectory	37
5.4	Trajectory tracking	38
5.5	Conclusion	39
6	Implementation	40
6.1	Simulation environment	40
6.2	Control software on the physical robot	40
6.2.1	Computational considerations	41
6.3	Conclusion	42
7	Results	43
7.1	MPC parameters	43
7.1.1	Number of constant acceleration sections	44
7.1.2	Minimum Swing Duration	44
7.2	Controller performance	45
7.2.1	Forward step	45
7.2.2	Forward step with a changing desired step location	56
7.2.3	Push recovery	61
7.3	Capture point calculation	70
7.4	Conclusion	73
8	Conclusion	75
A	Frame definition for TULip	77
B	XSens	79
C	Quintic (5th order) spline	81
C.1	Introduction	81
C.2	Constraints	81
C.3	Derivation	82
C.4	Writing the linear system of equations in matrix form	83
C.5	Boundary conditions	85
C.6	Solving for the coefficients	86

Acknowledgments

First of all, I would like to thank my adviser, Martijn Wisse, for his advice during this thesis. I would especially like to thank you for introducing me to the field of humanoid robotics. Your enthusiasm and motivation made my time at the Delft Biorobotics Laboratory an irreplaceable experience.

I would like to thank the Florida Institute for Human and Machine Cognition (IHMC) for providing me with the opportunity to work on cutting edge robotics research while finishing my masters degrees. I would like to thank all my colleagues there for their support. Specifically, I would like to thank Jerry Pratt for his insights and commentary. I would also like to thank Twan Koolen for our many discussions and providing invaluable commentary on this thesis.

I would also like to thank everyone at the Delft Biorobotics Laboratory and the Dutch Robotics team. Tomas de Boer for helping me select this research topic and our many late night coding sessions leading up to RoboCup. Guus Liqui Lung, Jan van Frankenhuyzen, Eelke van Breda, Gijs van der Hoorn, and the staff of the precision workshop for their support and willingness to always fix TULip when it was most needed. Jeroen Fransman, Sebastiaan Kiemel, Boris Lenseigne and everyone working on TULip for providing a great atmosphere to work and study in.

Finally, I would like to thank my family, for without their continuing support I would not have been able to take all the opportunities that have come across my path.

Pensacola,
August 20, 2012

Introduction

Humanoids robots are well adapted to function in a world that has been designed for humans. During interactions with the environment, unknown disturbances will be applied on the robot. Keeping balance under these conditions is essential for the successful deployment of humanoid robots.

Moving one of the feet to a new position allows the robot to recover from large disturbances. Suitable step locations for recovery from large disturbances can be determined using capturability [1]. A central concept is the capture point, defined as any point on the ground where the robot can step in order to bring itself to a complete stop [2]. The distance between the foot of the robot and the capture point increases with longer step durations. During the step, the location of the capture point is constantly updated to compensate for further disturbances. Quick and accurate control of the swing foot position is required for the robot to be robust against large disturbances. This thesis provides a novel foot position control strategy for force controllable bipedal robots.

Passive dynamics have been successfully exploited in the control of bipedal robots to attain a stable walking gait. Passive dynamic walkers can walk down a slope with no actuation [3]. Flat ground walking is possible with only positive actuator effort [4]. The goal of thesis is to create a controller that takes advantage of the dynamics of the swing leg to reduce control effort while providing accurate foot positioning capability in order to step quickly and accurately.

Planning the foot trajectory can be done in task space and joint space. In task space, the foot trajectory is described in Cartesian coordinates and the joint angles are calculated from the desired foot positions. This approach is very flexible and is well suited to plan foot trajectories that avoid obstacles. However, around singular configurations of the leg¹ the desired joint velocities and accelerations required to accurately track the trajectory are very high [5]. Therefore, our approach is to describe the foot trajectory as function of the joint angles. This method does not result in unrealistically high joint velocities when the leg is in singular configurations, and allows walking with a stretched knee.

In this thesis we propose two distinct controllers. The first controller is a model predictive controller using online optimization of the swing leg motion and the second one is a simplified controller that can be used on a robot that has limited computational power. The controllers are designed to be implemented on two robots that use capturability to control balance, M2V2 and TULIP. An illustration of the robots is provided in figure 1.

In chapter 1, we give an overview of the state of the art of current methods to control the swing leg and their drawbacks. In chapter 2 we introduce the

¹An example of a singular configuration for a leg is when the knee is stretched.

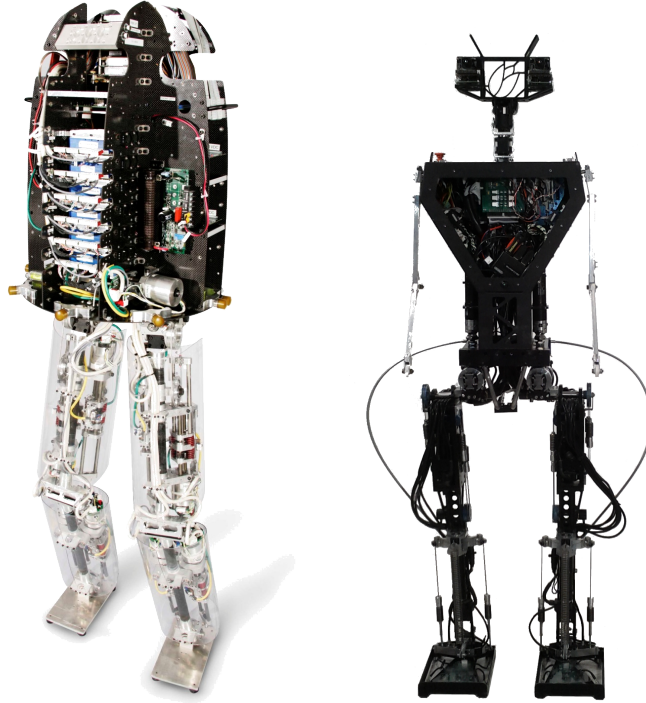


Figure 1: Humanoid robots. M2V2 is shown on the left and TULip is shown on the right.

model of our robot. The calculation of the capture point is described in chapter 3 after which we will describe the model predictive controller in chapter 4 and the reduced complexity controller in chapter 5. Details of the implementation of the controllers is given in chapter 6. The performance of the controllers is compared using simulations in chapter 7.

Chapter 1

Background

This chapter provides background information required to start the research into fast swing leg algorithms. We first review existing existing methods to accurately control the swing leg and the foot position for various walking algorithms, followed by an overview of the key characteristics of the actuators that are used in M2V2 and Tulip, the two robots on which the new controllers will ultimately be implemented in this thesis.

1.1 Swing foot position control for various walking algorithms

Most walking robots use the zero moment point (ZMP) balance criteria to generate walking patterns. Walkers based on the control of the ZMP plan a Center of Mass trajectory that keeps the Center of Pressure (CoP) well inside the support polygon of the feet. Foot placement is predetermined by the programmer or a planning algorithm which plans several steps ahead. This method has successfully been applied to the control of several bipedal robots, including Honda Asimo [6], HRP-2 [7], Aldebaran Nao [8], and KAIST Hubo [9]. Advances in the walking algorithms mean that these robots are stable even if perturbed by small disturbances [7]. Stiff position controlled joints are used to accurately control the center of mass position. During the swing phase, a foot trajectory is defined from the current location towards the desired step location in Cartesian space. Usually, a spline interpolation is employed to create a smooth trajectory for the foot. Given a robot with six degrees of freedom in each leg, this results in twelve unknown joint positions and twelve constraints (position and orientation of the foot, position of the CoM and orientation of the upper body). Using inverse kinematics, the corresponding joint positions can be determined. Because the desired foot position is typically fixed for the ZMP approach, it is not suitable for our problem where the desired foot position changes during the swing motion.

The current foot trajectory controller on M2V2 plans the foot trajectory in Cartesian space. The trajectory consists of three phases: take-off, cruise, and landing. Specified are the maximum acceleration and velocity of the foot, the slope of the take-off and landing phase and the desired ground clearance height. During the take-off phase, the foot accelerates towards the desired step

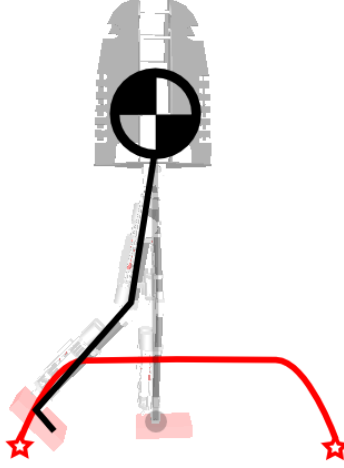


Figure 1.1: The desired foot trajectory for M2V2. The trajectory consists of three phases, a take-off phase, a constant height cruise phase and a landing phase.

location on the take-off slope until it reaches the desired ground clearance and transitions to the cruise phase. The foot moves towards the desired step location until the landing slope will take the foot towards the desired step location. In the landing phase, the foot decelerates and comes to a stop before touching down on the ground, see figure 1.1. The desired joint angles are determined from the trajectory using inverse kinematics and stiff position control is used to track the desired joint angles. The disadvantage of this controller is that it plans foot trajectories in Cartesian coordinates, leading to very large desired joint accelerations and velocities near singular configurations. Because the maximum joint accelerations and velocities are limited, it is not possible to accurately track the trajectory near singular configurations. This leads to large end of step positioning errors, especially if the leg is almost stretched at the end of the swing. Therefore, this control algorithm does not give adequate performance for accurate control of the swing foot position.

Joint space trajectory control is applied on various Limit Cycle Walkers, such as Meta and Flame [10]. Joint trajectories are defined by polynomial interpolation. Stability is attained by finding a limit cycle, where the state of the robot is the same at the end of each step. Foot placement is not adjusted during the swing motion. By defining joint trajectories instead of a foot trajectory, singular configurations do not lead to high accelerations and these robots can walk with stretched knees. Using joint trajectories with polynomial interpolation avoids problems with singular configurations and the resulting joint accelerations are therefore very useful for swing foot position control. The control algorithm as implemented for Meta and Flame does not incorporate changing desired foot positions and can therefore not be used for our control problem.

1.1.1 Joint position control

Traditional linear control, such as P(I)D control or full state feedback of the individual joints is a straightforward control technique to control the joint positions and consequently the swing foot position. The main advantage is that the only inputs are desired joint position and optionally desired joint velocity. A dynamic model of the robot is not necessary for the implementation of these controllers. Tracking behavior is accurate in only a small area of the work space due the non-linear dynamics of the swing leg. Therefore, linear control techniques are not suitable for accurate control of the swing foot position, particularly when the control gains are limited by the actuators.

Controlling the swing leg is a multi-input, multi-output (MIMO) problem. Using an accurate model of the inverse robot dynamics, joints torques can be calculated as a function of joint accelerations. The system behavior as a function of joint accelerations is decoupled and linear [11]. A PD controller with feed forward is used to determine the joint accelerations $\ddot{q}$ from the desired joint position q_d , joint velocity $\dot{q}_d$ and acceleration $\ddot{q}_d$

$$\ddot{q} = \ddot{q}_d + K_v(\dot{q}_d - \dot{q}) + K_p(q_d - q) \quad (1.1)$$

Where $\dot{q}$ is the measured joint velocity and q is the measured joint angle. The tracking error of this controller is theoretically zero, however inaccuracies in the dynamic model and discretization make this impossible to attain in practice.

1.2 Actuators

Stiff position control is necessary to accurately track joint trajectories. Robots based on ZMP control are designed for stiff position control, however M2V2 and TULip are designed for force control. This leads to the question if accurate tracking of joint trajectories is possible using actuators designed for force control.

To provide high precision torque control, both TULip and M2V2 are designed with series elastic actuators (SEAs) [12]. On M2V2 all joints are implemented with SEAs, while TULip has SEAs on the hip pitch, knee, ankle pitch and ankle roll and electric motors on the hip yaw and roll.

Series elastic actuators have an elastic element of known stiffness between the output shaft of the actuator and the joint axle. The deflection of the elastic element is measured (either directly or indirectly by the difference between the actuator shaft and joint axle angle) and used to calculate the joint torque. A PID controller is used to control the joint torque using the measured deflection. This allows accurate control of the joint torques.

The usage of SEAs allows the robots to use very flexible balance algorithms as described in [1]. The main disadvantage of SEAs over directly coupled actuators is that they have limited force bandwidth. This reduces the stiffness of the position control loop and makes tracking the desired foot trajectory inaccurate. Therefore, traditional foot position controllers are less suited for usage on M2V2 and TULip.

1.3 Conclusion

This chapter provided the required background on existing swing leg controllers and on the actuators used in our robots. With this information, we can conclude that Cartesian based foot position control schemes, while successfully used in numerous robots, suffer from singularities and the corresponding joint velocities and accelerations. This makes Cartesian based foot position control not suited for our control strategy. Joint trajectory control schemes avoid singularities, but have less control over the foot position during the swing. Tracking the desired joint angles can be done using P(I)D control, but for better tracking performance it is necessary to use a model based controller that includes the dynamics of the leg.

Force controllable actuators based on series elastic actuators provide high fidelity force control, but have reduced bandwidth compared to directly coupled actuators. The limited bandwidth reduces the stiffness of the position control loop, reducing the accuracy of the foot placement.

In the coming chapters, we will introduce a control scheme based on joint trajectory control that accurately positions the foot with SEAs.

Chapter 2

Model

2.1 Introduction

To accurately control the swing foot position of the robot while taking advantage of the passive dynamics of the swing leg it is desirable to have an accurate model of both the kinematics and dynamics of the robot. Using a forward kinematic model we can calculate the foot position as a function of the joint angles, while inverse kinematics are used to calculate desired joint angles as function of a desired foot position. M2V2 and Tulip are lower body humanoid robots with 6 degrees of freedom in each leg. For both robots, the ordering of the joint axis is the same, allowing us to use a common kinematic model. The kinematic structure of the robot is shown in figure 2.1. An existing forward kinematic model is available for both robots, this model is described in 2.2.1. We have derived an analytical expression for the inverse kinematics corresponding to the joint ordering employed by M2V2 and Tulip. The derivation is given in 2.2.2.

An inverse dynamics model allows the calculation of the joint torques resulting in a desired joint acceleration, given the joint angles and velocities and the external forces working on the robot. For our controller, we leverage the existing implementation of the inverse dynamics model to calculate the desired joint torques [13].

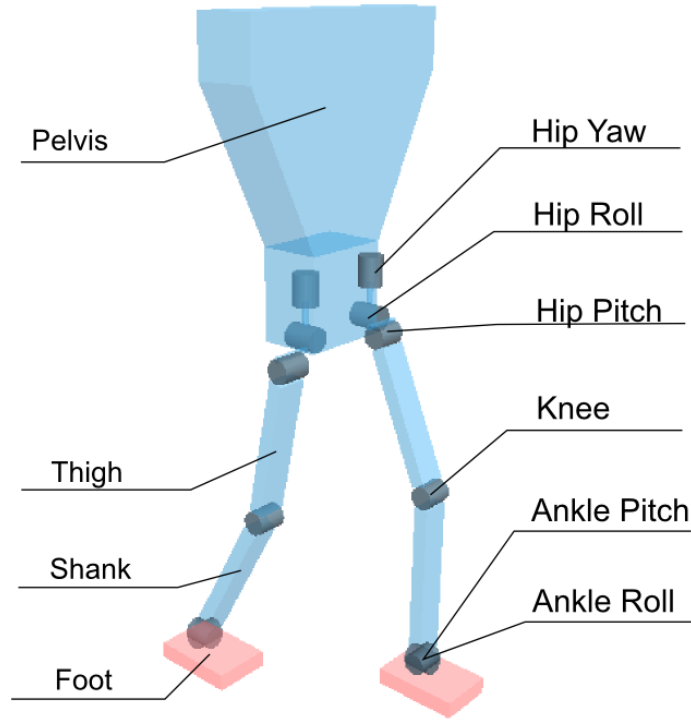


Figure 2.1: Link and joint names for the kinematic structure of both M2V2 and TULip. Note that the ankle roll and ankle pitch axis intersect. In our model, the order of the joints is hip yaw, hip roll, hip pitch, knee, ankle pitch and ankle roll.

2.2 Kinematic Model

There are two parts that make up the kinematic model of the robot, a forward kinematic model and an inverse kinematics calculator. The forward kinematics model, described in section 2.2.1, can be used to calculate the position of an arbitrary point on the robot, for example the swing foot position, as function of the joint angles. Using the forward kinematics model, we can also calculate the velocity of an arbitrary point on the robot using the joint positions and velocities, as described in section 2.2.1.

The inverse kinematic model, described in section 2.2.2, is used to calculate joint angles corresponding to a desired swing foot position. Likewise, desired angular velocities can be calculated from a desired foot velocity as described in 2.2.2.

2.2.1 Forward kinematics

The kinematic model of the robot consists of rotational joints and rigid links and follows the structure proposed by Duindam [14]. All joints are connected to their preceding joint by a link. A virtual root joint with six degrees of freedom is defined at the pelvis, connecting the robot to the world. At each joint, we define a reference coordinate frame (frame for short). The relation between the frames is described by a homogenous transformation matrix. A homogenous transformation matrix is a 4×4 matrix describing the translation and orientation between two frames [14]. We denote the homogeneous transformation matrix that transforms vectors from frame A and frame B as $\mathbf{H}_A^B$. The transformation matrix consists of a 3×3 rotation matrix $\mathbf{R}$ describing the orientation and a 3×1 vector $\mathbf{p}$ describing the translation.

$$\mathbf{H}_A^B = \begin{bmatrix} \mathbf{R}_A^B & \mathbf{p}_A^B \\ \mathbf{0} & 1 \end{bmatrix} \quad (2.1)$$

We notate a point expressed in frame A with the Cartesian coordinate x , y and z as

$$\mathbf{Q}^A = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2.2)$$

To transform the point $\mathbf{Q}^A$, expressed in frame A , to frame B we pre-multiply it by the transformation matrix $\mathbf{H}_A^B$.

$$\mathbf{Q}^B = \mathbf{H}_A^B \mathbf{Q}^A \quad (2.3)$$

Given the transformation matrix between frame B and C , $\mathbf{H}_B^C$, expressing the point $\mathbf{Q}^A$ in frame C becomes

$$\mathbf{Q}^C = \mathbf{H}_B^C \mathbf{Q}^B = \mathbf{H}_B^C \mathbf{H}_A^B \mathbf{Q}^A \quad (2.4)$$

From this, we can deduce that the transformation matrix describing the relation between frame A and frame C is

$$\mathbf{H}_A^C = \mathbf{H}_B^C \mathbf{H}_A^B \quad (2.5)$$

Finally, in order to express the point $\mathbf{Q}^C$, given in frame C , in frame A we just need to invert the transformation matrix.

$$\mathbf{Q}^A = (\mathbf{H}_A^C)^{-1} \mathbf{Q}^C \quad (2.6)$$

The inverse of a homogeneous transformation matrix $\mathbf{H}$ can be shown to be [14]

$$\begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \quad (2.7)$$

Rotation around an axis i by q radians can be described by a homogeneous transformation matrix $\mathbf{H}_i$ with zero translation.

$$\mathbf{H}_i(q) = \begin{bmatrix} \mathbf{R}_i(q) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \quad (2.8)$$

For a rotation q around the three primary axis, x , y and z , the corresponding rotation matrices are

$$\mathbf{R}_x(q) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos q & -\sin q \\ 0 & \sin q & \cos q \end{bmatrix} \quad (2.9)$$

$$\mathbf{R}_y(q) = \begin{bmatrix} \cos q & 0 & \sin q \\ 0 & 1 & 0 \\ -\sin q & 0 & \cos q \end{bmatrix} \quad (2.10)$$

$$\mathbf{R}_z(q) = \begin{bmatrix} \cos q & -\sin q & 0 \\ \sin q & \cos q & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.11)$$

Frames of TULip The frames of TULip are defined in figure 2.2. The position of the joint $i + 1$ with respect of the joint i is defined by the frame $i + 1$. The constant transformation matrix $\mathbf{H}_i^{i+1}$ consists of the translation $\mathbf{p}_i^{i+1}$ between joint i and $i + 1$.

$$\mathbf{H}_i^{i+1} = \begin{bmatrix} \mathbf{I} & \mathbf{p}_i^{i+1} \\ \mathbf{0} & 1 \end{bmatrix} \quad (2.12)$$

For numerical values describing the frame definitions of TULip, see appendix A.

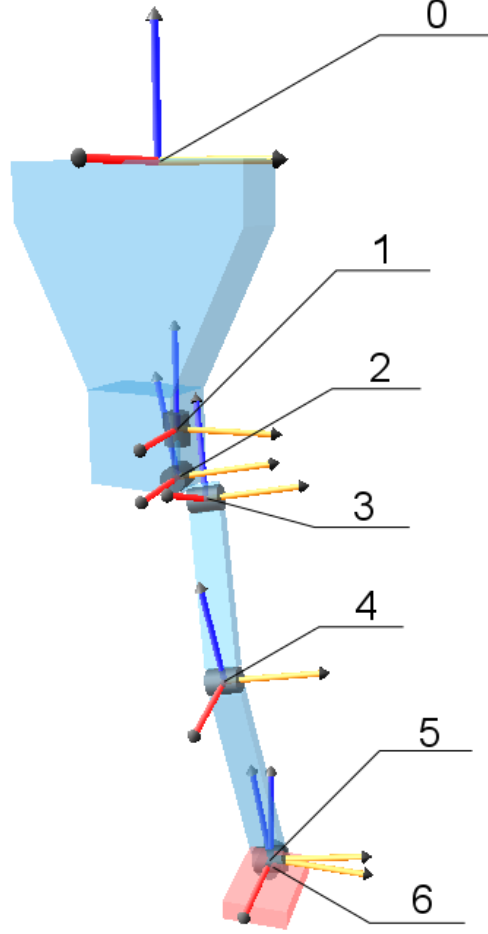


Figure 2.2: Frames as defined for Tulip. The root frame (0) is attached to the pelvis of the robot, at the backside of the top of the body. Frame (1) is attached to the outgoing axle of the hip yaw actuator. Frame (2) is attached to the axle of the hip roll actuator. Frame (3) is attached to the axle of the hip pitch actuator. Frame (4) is attached to the axle of the knee actuator. Frame (5) is attached to the axle of the hip pitch actuator and frame (6) is attached to the axle of the hip roll actuator.

The set of joint angles q for each leg is defined as

$$q = [q^1 \quad q^2 \quad q^3 \quad q^4 \quad q^5 \quad q^6]^\top \quad (2.13)$$

Where q^1 is the hip yaw angle, q^2 the hip roll angle, q^3 the hip pitch angle, q^4 the knee angle, q^5 the ankle pitch angle and q^6 the ankle roll angle. Note that due the frame definitions $\mathbf{R}_i^{i-1}$ depends on q^i and $\mathbf{p}_i^{i-1}$ is fixed.

Velocity kinematics

Analogous to transformation matrices describing the orientation between two reference frames, we can describe the relative velocity between two reference frames using a twist. A twist $T_j^{k,i}$ describes the relative velocity of frame j to frame i , expressed in frame k . A twist is a 6-by-1 vector, consisting of an angular part $\omega_j^{k,i} \in \mathbb{R}^3$ and a linear part $v_j^{k,i} \in \mathbb{R}^3$ [14].

$$T_j^{k,i} = \begin{bmatrix} \omega_j^{k,i} \\ v_j^{k,i} \end{bmatrix} \quad (2.14)$$

We can change the frame the twist is expressed in with the following relation [14]

$$T_j^{m,k} = \text{Ad}_{H_i^m} T_j^{i,k} \quad \text{with} \quad \text{Ad}_H := \begin{bmatrix} R & 0 \\ \tilde{p}R & R \end{bmatrix} \quad (2.15)$$

To determine the twist $T_j^{m,k}$ of an arbitrary body, we use the geometric Jacobian [14]. The geometric Jacobian from body j , relative to body i expressed in body k , $J_j^{k,i}$, is used to calculate the twist $T_j^{k,i}$

$$T_j^{k,i} = J_j^{k,i} \dot{q} \quad (2.16)$$

Velocity kinematics for the robot model The relative twists for each individual joint in the leg are

$$T_1^{1,0} = [0 \ 0 \ 1 \ 0 \ 0 \ 0]^T \dot{q}^1 \quad (2.17)$$

$$T_2^{2,1} = [1 \ 0 \ 0 \ 0 \ 0 \ 0]^T \dot{q}^2 \quad (2.18)$$

$$T_3^{3,2} = [0 \ 1 \ 0 \ 0 \ 0 \ 0]^T \dot{q}^3 \quad (2.19)$$

$$T_4^{4,3} = [0 \ 1 \ 0 \ 0 \ 0 \ 0]^T \dot{q}^4 \quad (2.20)$$

$$T_5^{5,4} = [0 \ 1 \ 0 \ 0 \ 0 \ 0]^T \dot{q}^5 \quad (2.21)$$

$$T_6^{6,5} = [1 \ 0 \ 0 \ 0 \ 0 \ 0]^T \dot{q}^6 \quad (2.22)$$

$$(2.23)$$

The geometric Jacobian for the leg, used to find the twist between the root body and the foot $T_6^{0,0}$, is

$$J_6^{0,0} = \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} & \text{Ad}_{H_1^0} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} & \text{Ad}_{H_2^0} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} & \text{Ad}_{H_3^0} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} & \text{Ad}_{H_4^0} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} & \text{Ad}_{H_6^0} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix} \quad (2.24)$$

2.2.2 Inverse kinematics

Two methods can be used to calculate the joint angles as function of the desired foot position, a numerical optimization using the forward kinematic model and an algebraic derivation. An algebraic derivation of the inverse kinematics has several advantages over a numerical solution

- A numerical solution can become unstable, especially near singular configurations of the leg.
- An algebraic solution can provide feedback on the reason why a certain position is not reachable.
- Multiple solutions to the inverse kinematics problem are possible. An algebraic solution can distinguish between feasible and infeasible solutions based on joint limits.
- An algebraic solution can be faster to compute

A major disadvantage of an algebraic solution is that it needs to be derived based on the kinematic configuration of the robot and is not guaranteed to exist. It can be shown that for all leg configurations with 6 degrees of freedom where three axes intersect it is possible to find an algebraic solution to the inverse kinematics [15]. However, both M2V2 and TULip have no three axes intersecting and there is no solution to calculate the joint angles as function of the desired homogeneous transformation matrix $\mathbf{H}_6^0$ between the pelvis and the foot.

To find a solution to the algebraic inverse kinematics problem, the complexity of the problem is reduced by defining the hip yaw angle e_{yaw} . The hip yaw angle is approximately equal to the foot yaw angle with respect to the pelvis. By defining the hip yaw angle, we need 5 variables to describe the remaining 5 degrees of freedom. These are the foot position with respect to the pelvis $\mathbf{p}_6^0$, the desired foot pitch with respect to the pelvis e_{pitch} and the desired foot roll with respect to the pelvis e_{roll} . The transformation matrix $\mathbf{H}_6^0$ can be calculated from the desired translation $\mathbf{p}_6^0$, the desired hip yaw angle e_{yaw} and the desired foot pitch e_{pitch} and foot roll e_{roll} with respect to the pelvis.

$$\mathbf{H}_6^0 = \mathbf{R}_z(e_{yaw}) \begin{bmatrix} \mathbf{I} & \mathbf{p}_6^0 \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{R}_y(e_{pitch}) \mathbf{R}_x(e_{roll}) \quad (2.25)$$

First we derive the hip roll, hip pitch and knee angles from the desired translation $\mathbf{p}_6^1$. Then we calculate the rotation $\mathbf{R}_4^0$ and use that to determine $\mathbf{R}_6^4$ and the corresponding ankle angles.

Hip yaw angle The hip yaw angle q_0 is defined to be the desired hip yaw angle e_{yaw}

$$q_0 = e_{yaw} \quad (2.26)$$

Hip roll angle The hip roll angle is determined using the geometric relation shown in figure 2.3. The distances l_a and l_b are the third and second component (z and x) from the translation between frame 2 and frame 5, $\mathbf{p}_2^5$. Note that $\mathbf{H}_6^0$ can be written as

$$\mathbf{H}_6^0 = \mathbf{H}_1^0 \mathbf{H}_z(q_0) \mathbf{H}_2^1 \mathbf{H}_x(q_1) \mathbf{H}_3^2 \mathbf{H}_y(q_2) \mathbf{H}_4^3 \mathbf{H}_y(q_3) \mathbf{H}_5^4 \mathbf{H}_y(q_4) \mathbf{H}_6^5 \mathbf{H}_x(q_5) \quad (2.27)$$

From this equation, we can extract the transformation between the pelvis and the hip roll frame $\mathbf{H}_2^0$ and use that to find the translation between the hip roll frame and the ankle $\mathbf{p}_5^2$.

$$\mathbf{H}_2^0 = \mathbf{H}_1^0 \mathbf{H}_z(q_0) \mathbf{H}_2^1 \quad (2.28)$$

$$\mathbf{p}_5^2 = (\mathbf{H}_2^0)^{-1} \mathbf{H}_6^0 \mathbf{H}_5^{\bar{c}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (2.29)$$

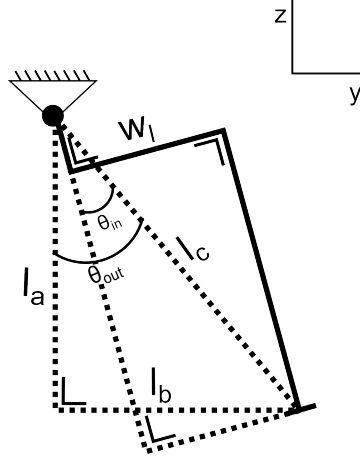


Figure 2.3: Determining the hip roll angle. A right triangle is spanned between the hip roll actuator and the ankle. Its sides have a length of l_a and l_b while the hypotenuse has a length of l_c . w_l is the distance between the hip roll actuator and the hip pitch actuator. The hip roll angle is equal to $\theta_{out} - \theta_{in}$.

With $\mathbf{p}_5^2$, we can calculate two angles θ_{in} and θ_{out} as defined in figure 2.3. The hip roll angle q_1 is

$$q_1 = \theta_{out} - \theta_{in} \quad (2.30)$$

First, we calculate the length of the sides of the dotted triangle in figure 2.3

$$l_a = -[0 \ 0 \ 1 \ 0] \mathbf{p}_5^2 \quad (2.31)$$

$$l_b = [0 \ 1 \ 0 \ 0] \mathbf{p}_5^2 \quad (2.32)$$

$$l_c = \sqrt{l_a^2 + l_b^2} \quad (2.33)$$

l_c corresponds to the distance between the origins of frame 2 and 5. We can check if the desired foot position is kinematically feasible here by making sure l_c is less than the distance between the hip roll actuator and the ankle when the leg is fully stretched.

The angles θ_{in} and θ_{out} are calculated using simple geometry

$$\theta_{in} = \arcsin\left(\frac{w_l}{l_c}\right) \quad (2.34)$$

$$\theta_{out} = \text{atan2}(l_b, l_a) \quad (2.35)$$

Note that the arctan function takes only one argument and cannot distinguish between diametrically opposite directions. For example, take two 2-D vectors $\mathbf{a} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$. The counterclockwise angle between $\mathbf{a}$ and the x-axis is $\frac{\pi}{4}$ and the counterclockwise angle between $\mathbf{b}$ and the x-axis is $-\frac{3\pi}{4}$. However, if we use the arctan function, then $\arctan(\frac{1}{1}) = \arctan(\frac{-1}{-1}) = \frac{\pi}{4}$. Therefore, we use the $\text{atan2}(x, y)$ function in the implementation of the inverse kinematics. $\text{atan2}(x, y)$ is defined as

$$\text{atan2}(x, y) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases} \quad (2.36)$$

Hip pitch angle We use the hip roll angle to find the translation between frame 3 and frame 5 $\mathbf{p}_5^3$.

$$\mathbf{H}_3^0 = \mathbf{H}_2^0 \mathbf{H}_x(q_1) \mathbf{H}_3^2 \quad (2.37)$$

$$\mathbf{p}_5^3 = (\mathbf{H}_3^0)^{-1} \mathbf{H}_6^0 \mathbf{H}_5^6 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (2.38)$$

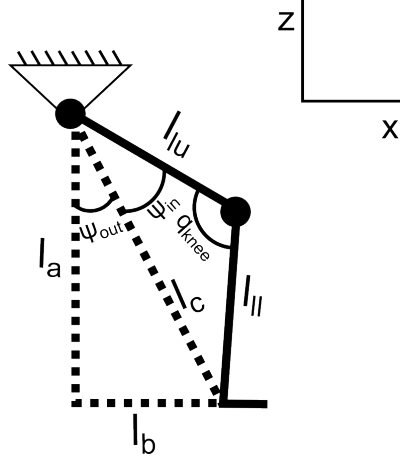


Figure 2.4: Deriving hip pitch and knee angles. The hip pitch angle q_3 is equal to $-(\psi_{out} + \psi_{in})$. l_{lu} is the upper leg length and l_{ll} lower leg length.

Using figure 2.4, we can determine the distance between the origin of frame 2 and frame 5 l_c from $\mathbf{p}_5^3$.

$$l_a = [0 \quad 0 \quad 1 \quad 0] \mathbf{p}_5^3 \quad (2.39)$$

$$l_b = [0 \quad 0 \quad 1 \quad 0] \mathbf{p}_{3,x}^5 \quad (2.40)$$

$$l_c = \sqrt{l_a^2 + l_b^2} \quad (2.41)$$

We check if the foot position is kinematically feasible here by making sure l_c is less than the upper leg length l_{lu} plus the lower leg length l_{ll} . Using the rule of cosines we can now find ψ_{in}

$$\psi_{in} = \arccos\left(\frac{l_{ul}^2 + l_l^2 - l_c^2}{2l_{ul}l_l}\right) \quad (2.42)$$

ψ_{out} is found by using the properties of a right triangle

$$\psi_{out} = \text{atan2}(l_b, l_a) \quad (2.43)$$

Finally the hip pitch angle q_3 is

$$q_3 = -(\psi_{out} + \psi_{in}) \quad (2.44)$$

Knee angle The knee angle is found using the rule of cosines and the geometry defined in 2.4

$$q_{knee} = \pi - \arccos\left(\frac{l_{ll}^2 + l_{ul}^2 - l_c^2}{2l_{ll}l_{ul}}\right) \quad (2.45)$$

Ankle pitch and roll By multiplying the rotation matrices of the hip yaw, hip roll, hip pitch and the knee together, we get the orientation of the frame before the ankle pitch joint, frame $\bar{5}$. Because the ankle roll and ankle pitch intersect, we can easily determine the ankle angles to get to the desired orientation of frame 6 (the foot).

$$\mathbf{R}_5^0 = \mathbf{R}_z(q_0)\mathbf{R}_x(q_1)\mathbf{R}_y(q_2)\mathbf{R}_y(q_3) \quad (2.46)$$

$$\mathbf{R}_6^{\bar{5}} = (\mathbf{R}_5^0)^{-1}\mathbf{R}_6^0 \quad (2.47)$$

From the rotation matrix $\mathbf{R}_6^{\bar{5}}$ we extract the yaw-pitch-roll Euler angles using [11].

$$r_{pitch} = \text{atan2}(-\mathbf{R}_6^{\bar{5}}(3, 1), \sqrt{\mathbf{R}_6^{\bar{5}}(1, 1)^2 + \mathbf{R}_6^{\bar{5}}(2, 1)^2}) \quad (2.48)$$

$$r_{yaw} = \text{atan2}\left(\frac{\mathbf{R}_6^{\bar{5}}(2, 1)}{\cos(r_{pitch})}, \frac{\mathbf{R}_6^{\bar{5}}(1, 1)}{\cos(r_{pitch})}\right) \quad (2.49)$$

$$r_{roll} = \text{atan2}\left(\frac{\mathbf{R}_6^{\bar{5}}(3, 2)}{\cos(r_{pitch})}, \frac{\mathbf{R}_6^{\bar{5}}(3, 3)}{\cos(r_{pitch})}\right) \quad (2.50)$$

Where $\mathbf{R}_6^{\bar{5}}(i, j)$ is the element at the i^{th} row and the j^{th} column of $\mathbf{R}_6^{\bar{5}}$.

Note that r_{yaw} should equal zero because there is no degree of freedom in the yaw axis in the ankle.

The resulting ankle angles are

$$q_4 = r_{roll} \quad (2.51)$$

$$q_5 = r_{pitch} \quad (2.52)$$

Inverse velocity kinematics

Using the Jacobian, not only can we find the twist between the root body and an arbitrary body, we can also use it to find desired joint velocities given a desired twist.

$$\dot{\mathbf{q}} = J_i^{-1}T_i^{0,0} \quad (2.53)$$

However, if the Jacobian is singular it is not possible to invert the matrix and there is no solution. An example of a singular configuration of the leg would be any situation where the knee is stretched. Physically, this means that it is not possible to produce an arbitrary velocity of the foot. We would like to find a set of joint velocities $\dot{\mathbf{q}}$ that minimizes the error $e = \|J_i\dot{\mathbf{q}} - T_i^{0,0}\|$ between the actual twist of the foot and the desired twist $T_i^{0,0}$. At and near singular configurations, the desired joint velocities in $\dot{\mathbf{q}}$ will be very large to minimize the error e . Proposed in [16], large joint velocities can be balanced against a large residual error by minimizing the sum

$$\|J_i\dot{\mathbf{q}} - T_i^{0,0}\|^2 + \alpha^2\|\dot{\mathbf{q}}\| \quad (2.54)$$

Where α is the dampening parameter. The sum in 2.54 can be written in matrix form as

$$\left\| \begin{bmatrix} J_i \\ \alpha \mathbf{I} \end{bmatrix} \dot{q} - \begin{bmatrix} T_i^{0,0} \\ \mathbf{0} \end{bmatrix} \right\|^2 \quad (2.55)$$

and the unique set of joint velocities that minimize $\dot{q}_a$ is given by

$$\dot{q}_a = \begin{bmatrix} J_i \\ \alpha \mathbf{I} \end{bmatrix}^+ \begin{bmatrix} T_i^{0,0} \\ \mathbf{0} \end{bmatrix} = (J_i^\top J_i + \alpha^2 \mathbf{I})^{-1} J_i^\top T_i^{0,0} \quad (2.56)$$

This technique is referred to as the damped least squares pseudo inverse. By using the approximation $\dot{q}_a$ large joint velocities are avoided, but near singular configurations the desired joint velocity becomes zero. This can become problematic if very high feedback gains are used for the desired joint velocity in comparison to the feedback gains for the joint position. The desired joint velocity will be low or zero, while in order to accurately track a desired foot trajectory the velocities have to be quite large.

2.3 Conclusion

We have implemented a general framework to calculate the forward kinematics of the robot, which allows us to easily request the position of any part of the robot as function of the joint angles. This framework is not dependent on a particular configuration of joints in the leg and can easily be adapted for different robots.

The algebraic inverse kinematic solution allows a quick and accurate calculation of the desired joint angles as a function of the desired foot position. However, this solution is dependent on the configuration of the robot leg and is not general to all robots. For new robot designs, it is advised to have the joint axis in the hip intersect such that a full solution to the inverse kinematics is possible.

Desired joint velocities can be found from the desired foot velocity using the Geometric Jacobian and the usage of the Damped-Least-Squares pseudo-inverse guarantees that numerical issues around singular configurations are avoided.

Chapter 3

Capture Point estimation

3.1 Introduction

Our controller has to accurately position the foot on the capture point while taking the passive dynamics of the leg into account. To control the swing leg such that it steps on the capture point we first need an accurate estimate of the capture point of the robot with respect to the world.

To determine the capture point, we model the robot as a 3D inverted pendulum. This allows us to quickly calculate the instantaneous capture point (iCP). The iCP is defined as [1]

The point on the ground that enables the system to come to a stop if it were instantaneously placed and maintain its point foot there.

The instantaneous capture point can be calculated from the position and velocity of the center of mass (CoM) using

$$\mathbf{r}_{ic} = \mathbf{P}\dot{\mathbf{r}} + \frac{\dot{\mathbf{r}}}{\omega_0} \quad (3.1)$$

Where $\mathbf{r}_{ic}$ is the location of the capture point, $\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, $\mathbf{r}$ is the location of the center of mass and $\omega_0 = \sqrt{\frac{g}{z_0}}$ with z_0 the height of the CoM and g the gravity constant.

However, it is not possible to instantaneously place the foot on the robot on an arbitrary point. That means that the instantaneous capture point is not necessary a capture point. The dynamics of the iCP as function of time are given explicitly by [1]

$$\mathbf{r}_{ic}(\Delta t) = [\mathbf{r}_{ic} - \mathbf{r}_{ankle}]e^{\omega_0\Delta t} + \mathbf{r}_{ankle} \quad (3.2)$$

The capture point is given by the location of the iCP at time Δt , where Δt is the desired step duration. If the robot steps on that point the CoM of the robot will come to a rest above the step location.

In this chapter we will describe how to find the CoM state necessary to calculate the instantaneous capture point and the capture point at time t .

3.2 Center of Mass Position

The center of mass of the robot is a linear combination of the center of mass position of all rigid bodies. Given two rigid bodies with masses m_1 and m_2 and a center of mass location p_{CoM}^1 and p_{CoM}^2 , the combined center of mass $p_{CoM,combined}^0$ of these two bodies expressed in frame 0 is

$$p_{CoM,combined}^0 = \frac{m_1 \mathbf{H}_1^0 p_{CoM}^1 + m_2 \mathbf{H}_2^0 p_{CoM}^2}{m_1 + m_2} \quad (3.3)$$

Using this formula recursively, we can find the center of mass of the robot $p_{CoM,robot}^0$ by combining the center of masses of the i individual rigid bodies in $O(n)$ using 3.4.

$$p_{CoM,robot}^0 = \frac{m_1 \mathbf{H}_1^0 p_{CoM}^1 + m_2 \mathbf{H}_2^0 p_{CoM}^2 + \dots + m_i \mathbf{H}_{i-1}^0 p_{CoM}^{i-1} + m_i \mathbf{H}_i^0 p_{CoM}^i}{m_1 + m_2 + \dots + m_{i-1} + m_i} \quad (3.4)$$

3.3 Estimating Center Of Mass Velocity

3.3.1 Sensor data

Tulip has two (sets) of sensors that can deduct the velocity of the pelvis, the inertial measurement unit (IMU) and the pelvis velocity kinematics. To improve the velocity estimation, we combine the data from both sensors. We can use the body velocity kinematics to find the bias of the IMU, compensating the drift due to the integration of the accelerations.

Inertial Measurement Unit Tulip has one XSens Inertial Measurement Unit (IMU) on its pelvis. The IMU consists of three sensors, a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer. Internally, the IMU uses the output of these three sensors to estimate its orientation in the world. Integration of the accelerations in time gives an estimate of the velocity, however due to sensor bias and errors in the numerical integration the velocity found using integration will drift in time. Another disadvantage is that accelerometers tend to display a significant amount of sensor noise. For information how we have configured the XSens, see appendix B.

Pelvis velocity kinematics Assuming the stance foot is stationary on the ground and does not slip, the foot frame is stationary with respect to world and we can determine the pelvis velocity with respect to the world using the Jacobian defined in equation 2.24. First we calculate the twist $T_6^{0,0}$ using equation 2.16. We can transform this twist to the pelvis velocity with respect to the foot, $T_0^{0,6}$ by negation of the twist [17]

$$T_0^{0,6} = -T_6^{0,0} \quad (3.5)$$

To accurately combine the pelvis velocity kinematics with the data from the IMU we define a new frame called the IMU frame, which has a constant

offset from the pelvis frame. Using equation 2.15 we change the frame $T_0^{0,6}$ is expressed in to the IMU frame

$$T_0^{IMU,6} = \text{Ad}_{H_0^{IMU}} T_0^{0,6} \quad (3.6)$$

As long as at least one foot is fixed with relation to the ground, the calculated velocity of the pelvis depends on the accuracy of the measured joint angles. However, during walking the foot might slip or rotate with respect to the ground making the calculation of the pelvis velocity using only the joint velocities unreliable.

3.3.2 Kalman State Estimation

We use a Kalman state estimator to fuse the two measurements of the upper body velocity together. We assume the velocity along the three axis (x, y and z) is decoupled. The velocity along the z-axis is not used in the control of the robot. Therefore, we use two separate velocity estimators, one for the velocity along the x-axis and one along the y-axis. The internal state of the velocity estimator consists of two variables, the current velocity of the pelvis $v[n]$ and the acceleration sensor bias $b[n]$.

$$\mathbf{x}[n] = \begin{bmatrix} v[n] \\ b[n] \end{bmatrix} \quad (3.7)$$

The input of the system is the current acceleration of the pelvis $a[n]$, as measured by the accelerometer in the IMU

$$\mathbf{u}[n] = \begin{bmatrix} a[n] \\ 0 \end{bmatrix} \quad (3.8)$$

The system dynamics are modeled as

$$\mathbf{x}[n] = \mathbf{F}\mathbf{x}[n-1] + \mathbf{B}\mathbf{u}[n] + \mathbf{w}[n] \quad (3.9)$$

where

- the state matrix $\mathbf{F} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}$
- the input matrix $\mathbf{B} = \begin{bmatrix} \Delta t & 0 \\ 0 & 0 \end{bmatrix}$
- the process noise, modeled as a normal distribution with a covariance of $\mathbf{Q}$ and zero mean $\mathbf{w}[n] \sim N(0, \mathbf{Q})$

At each time step n , a measurement $\mathbf{z}[n]$ is taken of the internal state $\mathbf{x}[n]$

$$\mathbf{z}[n] = \mathbf{H}\mathbf{x}[n] + v_m[n] \quad (3.10)$$

Where

- the observation model which transforms the internal state to the measured state, $\mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$

- the measured velocity $v_m[n] = [0 \ 0 \ 0 \ 1 \ 0 \ 0] T_0^{IMU,6}$ for the sagittal plan and $v_m[n] = [0 \ 0 \ 0 \ 0 \ 1 \ 0] T_0^{IMU,6}$ for the coronal plane
- the observation noise, modeled as Gaussian noise with a covariance of $\mathbf{R}$, $\mathbf{v}[n] \sim N(0, \mathbf{R})$

The Kalman filter consists of a prediction and an update step.

Prediction During the prediction step, the measured acceleration is used to update the internal state $\bar{\mathbf{x}}[n]$ and the predicted error covariance $\bar{\mathbf{P}}[n]$. The estimated error covariance $\mathbf{P}[n]$ is determined in the update step, and is initialized to $\mathbf{0}$ at the first iteration.

$$\bar{\mathbf{x}}[n] = \mathbf{F}\mathbf{x}[n-1] + \mathbf{B}\mathbf{u}[n] \quad (3.11)$$

$$\bar{\mathbf{P}}[n] = \mathbf{F}\mathbf{P}[n-1] + \mathbf{Q}[n] \quad (3.12)$$

Update During the update step the prediction of the internal state is combined with the current observation of the true system state to improve the estimated system state. First, the measurement residual $\bar{\mathbf{y}}[n]$ and the residual covariance $\mathbf{S}[n]$ are determined

$$\bar{\mathbf{y}}[n] = v_m[n] - \mathbf{H}\bar{\mathbf{x}}[n] \quad (3.13)$$

$$\mathbf{S}[n] = \mathbf{H}\bar{\mathbf{P}}[n]\mathbf{H}^\top + \mathbf{R}[n] \quad (3.14)$$

Using the residual covariance the optimal Kalman gain $\mathbf{K}[n]$ can be determined

$$\mathbf{K}[n] = \bar{\mathbf{P}}[n]\mathbf{H}^\top \mathbf{S}[n] \quad (3.15)$$

Using the optimal Kalman gain, we update the internal state estimate $\mathbf{x}[n]$ and the estimated covariance $\mathbf{P}[n]$

$$\mathbf{x}[n] = \bar{\mathbf{x}}[n] + \mathbf{K}[n]\bar{\mathbf{y}}[n] \quad (3.16)$$

$$\mathbf{P}[n] = (\mathbf{I} - \mathbf{K}[n]\mathbf{H})\bar{\mathbf{P}}[n] \quad (3.17)$$

The estimated velocity of the pelvis v is

$$v = [0 \ 1] \bar{\mathbf{x}}[n] \quad (3.18)$$

Noise estimation

The matrix $\mathbf{Q}[n]$ and $\mathbf{R}[n]$ are the process and measurement noise respectively. For the velocity estimation, we assume that the noise is time-independent.

The process noise $\mathbf{Q}$ is the covariance matrix of the measured acceleration a_m and the accelerometer bias. We assume that the measured acceleration and accelerometer bias are linearly independent. The maximum accelerometer bias is given as $\epsilon = 0.02ms^{-2}$ in the XSens documentation. The variance, or expected

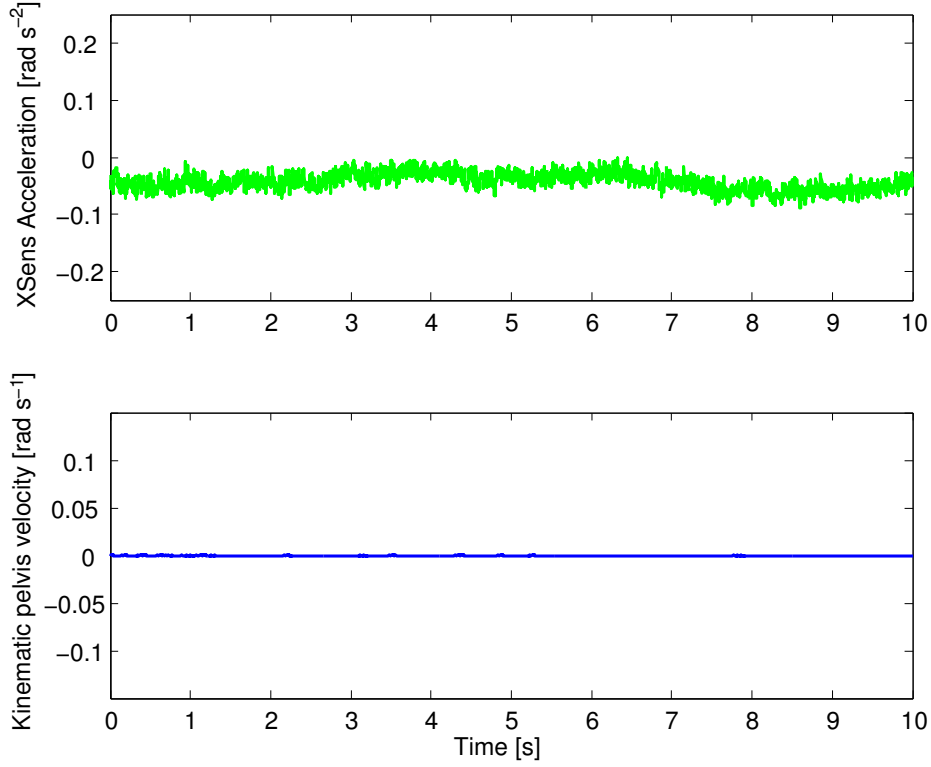


Figure 3.1: Measurement of the raw accelerometer data a_m from the XSens and the velocity of the pelvis calculated with pelvis velocity kinematics v_m . Both signals were recorded while the robot was suspended in the air with all joints stationary.

squared deviation, is ϵ^2 . Sampling at every $t = n\Delta t$ requires the individual covariances to be multiplied by Δt^2 .

$$\mathbf{Q} = \Delta t^2 \begin{bmatrix} \text{Var}(a_m) & 0 \\ 0 & \epsilon^2 \end{bmatrix} \quad (3.19)$$

The measurement noise is the variance of the velocity found using the pelvis velocity kinematics v_m .

$$\mathbf{R} = \text{Var}(v_m) \quad (3.20)$$

To determine the variance of a_m and v_m , we logged both signals for 10 seconds. During these 10 seconds, the robot was suspended in the air, the joints were stationary and motor power was off. The results of this measurement in the sagittal plane are shown in figure 3.1.

The variances of a_m and v_m in the sagittal plane are

$$\text{Var}(a_m) = 0.2455 \cdot 10^{-3} \quad (3.21)$$

$$\text{Var}(v_m) = 0.1016 \cdot 10^{-4} \quad (3.22)$$

3.3.3 From IMU velocity to CoM velocity

The CoM twist with respect to the world expressed in world, $T_{CoM}^{world,world}$ depends on the CoM twists of all rigid bodies. Defining $T_{CoM,i}^{world,world}$ as the CoM twist of body i with respect to the world, expressed in world, we can calculate the total center of mass twist of z rigid bodies using

$$T_{CoM}^{world,world} = \frac{\sum_{i=0}^z (m_i T_{CoM,i}^{world,world})}{\sum_{i=0}^z m_i} \quad (3.23)$$

However, finding the twists of each individual body with respect to world requires an accurate velocity estimation for each rigid body CoM. We have chosen not to derive the individual CoM velocities and instead approximate the center of mass velocity. This approximation consists of two parts

- Compensation of rotation around the IMU using a linear model
- The difference between the pelvis velocity and the CoM velocity using differentiation of the CoM position

Compensation of rotation around the IMU using a linear model Although care has been taken by placing the IMU near the Center of Mass of the robot, there is an offset between the CoM location and the IMU location. Therefore, any rotational velocity of the IMU will result in a linear velocity of the CoM. For simplicity, we use a very simple model to describe the relation between the rotational velocity of the IMU and the velocity of the CoM.

$$\dot{r}_{CoM} = \dot{x}_{imu} - c_{\omega_y, \dot{x}} \omega_y \quad (3.24)$$

$$\dot{r}_{CoM} = \dot{y}_{imu} + c_{\omega_x, \dot{y}} \omega_x + c_{\omega_z, \dot{y}} \omega_z \quad (3.25)$$

Where $\dot{r}_{CoM}$ and $\dot{r}_{CoM}$ are the estimated velocities of the CoM on the x-axis and y-axis respectively, $\dot{x}_{imu}$ and $\dot{y}_{imu}$ the velocities of the IMU estimated using the Kalman filter on the x-axis and y-axis respectively and ω_x , ω_y and ω_z the rotational velocities measured by the IMU around the x, y and z-axis respectively.

The coupling factors, $c_{\omega_y, \dot{x}}$, $c_{\omega_x, \dot{y}}$ and $c_{\omega_z, \dot{y}}$ describe the relation between the rotational velocity of the IMU and the CoM velocity. The values of the coupling factors are the respective distance between the IMU and the CoM when all joint positions are zero. For TULip, the values are

$$c_{\omega_y, \dot{x}} = 0.19$$

$$c_{\omega_x, \dot{y}} = 0.0$$

$$c_{\omega_z, \dot{y}} = 0.085$$

Center of mass velocity expressed in the IMU frame The CoM of the robot moves with respect to the IMU. For example, imagine the robot rotating its leg outward, moving its feet to the side. The CoM will move due to the leg being extended. To find the CoM velocity, we differentiate the position of the CoM in the IMU frame. To reduce the noise of this differentiation, we filter the resulting velocity with a 20Hz low pass Butterworth filter. The resulting velocity is added to the estimated velocity of the IMU.

3.4 Conclusion

We estimate the instantaneous capture point location by modeling the robot as a Linear Inverted Pendulum. To calculate the iCP, we need the CoM position and velocity. The CoM position is straightforward to calculate given the transformation between the rigid bodies. A Kalman filter is used to find an accurate estimation of the pelvis velocity, which is then transformed to the CoM velocity using coupling factors to compensate for the rotation of the pelvis with respect to the Center of Mass. Using the dynamics of the iCP we can determine a suitable capture point to place the feet of the robot to come to a standstill.

Chapter 4

Model Predictive Swing Controller

4.1 Introduction

To successfully control the swing leg for push recovery using capturability, the swing controller has to meet the following requirements

- Accurately place the foot at the desired step location in the given time.
- Handle unexpected changes in the desired step location during the swing.

Controlling the foot position in Cartesian coordinates gives us a very flexible approach with respect to the foot motion. However singular configurations, such as a stretched knee, result in joint velocities that exceed the maximum velocity of any actuator for certain motions of the foot. Near singular configurations, joint velocities and accelerations necessary to execute certain motions become very high. To avoid this, our controller describes the joint trajectories as function of joint angles, velocities and accelerations. The corresponding foot position is calculated using forward kinematics.

To handle unknown changes in step location, as well as disturbances, we propose to use a non-linear model predictive controller (NMPC). A basic NMPC controller consists of the following steps.

1. Measure the current system state.
2. Minimize a given cost function over a finite prediction horizon to find optimal input signals to the system.
3. Apply the first part of the optimal input signal.
4. At the start of the next control tick, continue with step 1.

We assume that the state of the robot is known at every moment in time.

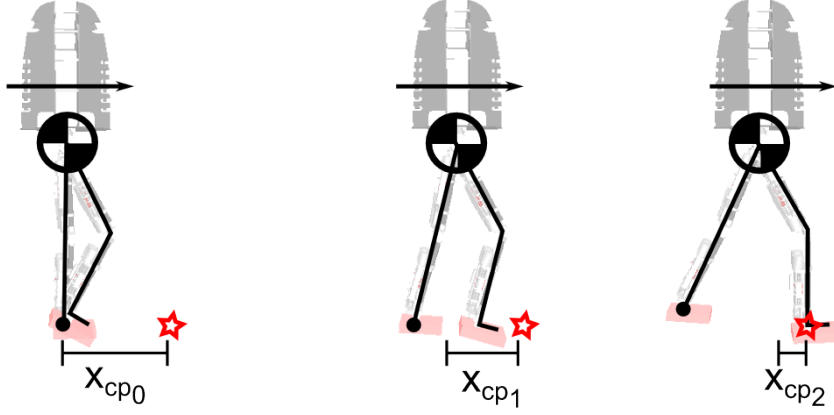


Figure 4.1: Motion of the pelvis of the robot during the step. The location of the desired step location with respect to the upper body changes during the step, because the pelvis moves forward.

4.2 Predicting final desired swing location

The design of the walking controller of M2V2 and TULip splits the control in two parts

- Balance controller
- Swing leg controller

The balance controller keeps the robot upright and controls the location of the instantaneous capture point, while the swing controller controls the swing foot location. The position of the swing foot is a function of the pelvis location, which changes during the swing motion.

In order to parametrize the joint trajectories, we require the desired joint angles and angular velocities at the end of the step. During the swing, the upper body moves with respect to the stance foot. See figure 4.1.

The transformation between the desired step location, called $\bar{ds}$, and the pelvis location at the end of the step, called ${}^0\mathbf{H}_0^{ds}$ is a function of the transform between the current location of the pelvis and the pelvis location at the end of the step $\mathbf{H}^{\bar{0}_0}$.

$$\mathbf{H}_{ds}^{\bar{0}} = \mathbf{H}_0^{\bar{0}} \mathbf{H}_{ds}^0 \quad (4.1)$$

This means that to accurately calculate the joint angles at the end of the swing, we need to predict the transformation matrix $\mathbf{H}_0^{\bar{0}}$. In the next section we will describe our pelvis position prediction algorithm.

4.2.1 Predicting the pelvis position at the end of the step

To predict $\mathbf{H}_0^{\bar{0}}$, we first model the dynamics of the robot using a simple model. The next step is to use the robot model to predict the translation between the

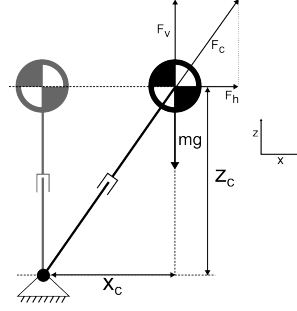


Figure 4.2: 2D Linear Inverted Pendulum Model (2D-LIPM)

current pelvis position and the pelvis position at the end of the step. Finally, we use the predicted translation to return $\mathbf{H}_0^0$.

Modelling the robot dynamics

The dominant dynamics of a bipedal robot with one leg on the ground can be modeled by an inverted pendulum connecting the ground contact point with the center of mass [18]. A widely used pendulum model to control bipedal walking robots is the Linear Inverted Pendulum Model (LIPM) [18, 19, 20]. The LIPM model, shown in figure 4.2, is based on the following assumptions [19]

- All mass is concentrated in a point mass corresponding to the CoM of the robot.
- The center of mass is at the constant height z_c .
- There is no torque at the base joint of the pendulum.

The equations of motion for the LIPM are

$$m\ddot{z} = F_v - mg = 0 \quad (4.2)$$

$$m\ddot{r} = F_h \quad (4.3)$$

$$\frac{F_v}{F_h} = \frac{z_c}{r} \quad (4.4)$$

$$m\ddot{r} = \frac{mgr}{z_c} \quad (4.5)$$

$$\ddot{r} = \omega^2 r \quad (4.6)$$

Where $\omega = \sqrt{\frac{g}{z_c}}$. Writing 4.6 in state-space form yields

$$\begin{bmatrix} \dot{r} \\ \ddot{r} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \omega^2 & 0 \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \end{bmatrix} \quad (4.7)$$

The explicit solution to 4.7 is

$$\begin{bmatrix} r(t) \\ \dot{r}(t) \end{bmatrix} = \begin{bmatrix} \cosh(\omega t) & \omega^{-1} \sinh(\omega t) \\ \omega \sinh(\omega t) & \cosh(\omega t) \end{bmatrix} \begin{bmatrix} r(0) \\ \dot{r}(0) \end{bmatrix} \quad (4.8)$$

Predicting the pelvis translation

The dynamics of the LIPM are decoupled so we can calculate the position of the pelvis separately in the sagittal and coronal plane. The position of the pelvis

$p^0 = \begin{bmatrix} x^0 \\ y^0 \\ z^0 \end{bmatrix}$ can be found using the forward kinematic model and the velocity of

the pelvis $\dot{p}^0 = \begin{bmatrix} \dot{x}^0 \\ \dot{y}^0 \\ \dot{z}^0 \end{bmatrix}$ is found using equation 3.18.

The estimated pelvis position at time t in the sagittal plane, $x_e^0(t)$, is found using equation 4.7, with $r(0) = x^0$ and $\dot{r}^0 = \dot{x}^0$. In the coronal plane the estimated pelvis position at time t , $y_e^0(t)$ can be calculated using the same equation 4.7 with $r(0) = y^0$ and $\dot{r}(0) = \dot{y}^0$.

The transform $\mathbf{H}_0^0$ becomes

$$\mathbf{H}_0^0 = \begin{bmatrix} 1 & 0 & 0 & x_e^0(t) \\ 0 & 1 & 0 & y_e^0(t) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.9)$$

With this transformation matrix, we can calculate the transformation matrix $\mathbf{H}_{ds}^0$ and use that in combination with the inverse kinematics calculator to find the desired joint angles at the end of the step.

4.3 Joint trajectory parametrization

The swing leg consist of six individual joints, namely hip yaw, hip roll, hip pitch, knee, ankle pitch and ankle roll. During the swing motion, the rotation around the z-axis (yaw) of the leg is relatively small and the inertia of the leg around the z-axis is low when the leg is near a straight-down configuration. Besides that, the foot of the robot has a small inertia relative to the rest of the leg. By excluding the hip yaw and ankle trajectories from the optimization we reduce the complexity and decrease the computational cost. Instead, we interpolate the joint angles of the hip yaw, ankle pitch and ankle roll using a quintic spline.

The trajectories of the hip roll, hip pitch and knee are then split in n sections with a constant acceleration. The input to our cost function is the acceleration of the joint at any section on the joint trajectories. By using the joint accelerations at each section, we are able to determine the joint position and velocity at any section and the final joint angle $q[n]$ and velocity $\dot{v}[n]$ is given by the following relation

$$\dot{q}[n] = \dot{q}[0] + \sum_{i=1}^n \ddot{q}[i] dT \quad (4.10)$$

$$q[n] = q[0] + \sum_{i=0}^{n-1} \dot{q}[i] dT + \sum_{i=1}^n \frac{1}{2} \ddot{q}[i] dT^2 \quad (4.11)$$

Where dT is the time taken for each section, or $\frac{t}{n}$ with t the remaining swing time.

For the three joints, we write down the state vector at time step i as

$$\mathbf{q}[i] = \begin{bmatrix} q_1[i] \\ q_2[i] \\ q_3[i] \end{bmatrix} \quad (4.12)$$

The vector of the joint accelerations at all time steps is

$$\ddot{\mathbf{q}} = [\ddot{\mathbf{q}}[0] \quad \ddot{\mathbf{q}}[1] \quad \dots \quad \ddot{\mathbf{q}}[n-1] \quad \ddot{\mathbf{q}}[n-2]] \quad (4.13)$$

4.4 Cost function

The goal of the model predictive controller is to generate a swing motion that minimizes actuator effort while accurately placing the foot on the desired step location. To describe actuator effort, we can employ several criteria

1. Minimum torque, see section 4.4.1.
2. Minimum torque-change, see section 4.4.2.

Assuming there is no regenerative braking, we are interested in the absolute value of actuator effort for every constant acceleration section. Therefore, we write the optimization problem as a sums-of-squares problem

$$f(\ddot{\mathbf{q}}) = \sum_{i=1}^n \mathbf{c}(\mathbf{q}[i], \dot{\mathbf{q}}[i], \ddot{\mathbf{q}}[i])^\top \mathbf{c}(\mathbf{q}[i], \dot{\mathbf{q}}[i], \ddot{\mathbf{q}}[i]) \quad (4.14)$$

Where $\mathbf{c}(\mathbf{q}[i], \dot{\mathbf{q}}[i], \ddot{\mathbf{q}}[i])$ is the optimization criterion.

4.4.1 Minimum torque

A minimum of joint torques as optimization criterion reduces the maximum torques required for the swing motion. Using an accurate dynamic model of the robot we can calculate the desired joint torque at each section of the trajectory using

$$\tau(\mathbf{q}[i], \dot{\mathbf{q}}[i], \ddot{\mathbf{q}}[i]) = \mathbf{M}(\mathbf{q}[i])\ddot{\mathbf{q}}[i] + \mathbf{C}(\mathbf{q}[i], \dot{\mathbf{q}}[i]) \quad (4.15)$$

Where $\mathbf{M}(\mathbf{q})$ is the inertia matrix of the system and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ encompasses the Coriolis forces and gravity. The optimization criterion for minimum torque is

$$\mathbf{c}(\mathbf{q}[i], \dot{\mathbf{q}}[i], \ddot{\mathbf{q}}[i]) = \tau(\mathbf{q}[i], \dot{\mathbf{q}}[i], \ddot{\mathbf{q}}[i]) \quad (4.16)$$

4.4.2 Minimum torque change

A minimization of the torque-change limits the bandwidth requirements of the actuator. The torque-change optimization criterion is the difference between two consecutive elements of the torque vector $\tau(\mathbf{q}[n], \dot{\mathbf{q}}[n], \ddot{\mathbf{q}}[n])[i]$ found in 4.15

$$\mathbf{c}(\mathbf{q}[i], \dot{\mathbf{q}}[i], \ddot{\mathbf{q}}[i]) = \tau(\mathbf{q}[i], \dot{\mathbf{q}}[i], \ddot{\mathbf{q}}[i]) - \tau(\mathbf{q}[i-1], \dot{\mathbf{q}}[i-1], \ddot{\mathbf{q}}[i-1]) \quad (4.17)$$

4.5 Optimization algorithm

There are only a few high-quality optimization packages available in Java for minimizing the sum-of-squares optimization problems. The one we have chosen to use is a Java port by the Forest Service of the US Department of Agriculture of the FORTRAN package MINPACK [21]. MINPACK implements the Levenberg-Marquardt algorithm for unconstrained non-linear systems and has the ability to numerically evaluate the Jacobian of the cost function as well as using a provided (analytical) Jacobian of the cost function¹ [22].

The Levenberg-Marquardt algorithm interpolates between a gradient decent and the Gauss-Newton method. It is more robust than the plain Gauss-Newton method and can find an optimal solution even if the initial conditions are far from the optimum.

4.6 Constraints

There are various constraints on the swing leg that must be met during the swing motion. These are

- The final joint angles and velocities of the swing leg should equal the desired joint angles and velocities, see section 4.6.1.
- The joint angles, velocities and torques should not exceed their respective limits, see section 4.6.2.
- The foot cannot hit obstacles or the ground before the end of the swing, see section 4.6.3.

We incorporate the constraints by rewriting the cost function such that it is unconstrained. This allows us to use an unconstrained optimization algorithm, giving us a large amount of optimization algorithms and implementations to choose from.

4.6.1 Final joint angles and velocities

The final joint angles and velocities are a linear combination of the accelerations of each section in the trajectory as given by equation 4.11 and 4.10. Therefore we can write down the constraint on the final joint angles as a linear equality constraint. For one joint, given the final desired joint angles $q[n]$ and velocities $\dot{q}[n]$ we can write the equality constraint in the form $\mathbf{Ax} = \mathbf{b}$

$$\begin{bmatrix} dT & dT & \dots & dT & dT \\ dT^2(n-1) + \frac{1}{2}dT^2 & dT^2(n-2) + \frac{1}{2}dT^2 & \dots & dT^2 + \frac{1}{2}dT^2 & \frac{1}{2}dT^2 \end{bmatrix} \begin{bmatrix} \ddot{q}[1] \\ \ddot{q}[2] \\ \vdots \\ \ddot{q}[n-1] \\ \ddot{q}[n] \end{bmatrix} = \begin{bmatrix} \dot{q}[n] \\ q[n] \end{bmatrix} \quad (4.18)$$

¹The Jacobian of the cost function is distinct from the geometric Jacobian from the kinematic model. See 4.7 for the definition.

Using a simple trick [23], we can rewrite the cost function such that the optimization problem becomes unconstrained. We optimize $\mathbf{f}(\ddot{\mathbf{q}})$ by allowing $\ddot{q}[1]$ to $\ddot{q}[n-2]$ to vary freely, and we can determine $\ddot{q}[n-1]$ and $\ddot{q}[n]$ from 4.18. Using the desired angle $q[n]$ and angular velocity $\dot{q}[n]$ we can determine $\ddot{q}[n-1]$ by the following relation

$$q[n-1] = q[n-2] + \dot{q}[n-2]dT + \frac{1}{2}\ddot{q}[n-2]dT^2 \quad (4.19)$$

$$q[n] = q[n-1] + \dot{q}[n-1]dT + \frac{1}{2}\ddot{q}[n-1]dT^2 \quad (4.20)$$

$$q[n] = x[n-2] + \frac{\dot{q}[n]dT}{2} + \frac{3\dot{q}[n-2]dT}{2} + \ddot{q}[n-2]dT^2 \quad (4.21)$$

$$\ddot{q}[n-1] = \frac{q[n] - q[n-2]}{dT^2} - \frac{\dot{q}[n] + 3\dot{q}[n-2]}{2dT} \quad (4.22)$$

The velocity constraint is used to determine $\ddot{q}[n]$:

$$\dot{q}[n] = \dot{q}[n-2] + \ddot{q}[n-1]dT + \ddot{q}[n]dT \quad (4.23)$$

$$\ddot{q}[n] = \frac{\dot{q}[n]}{dT} - \frac{\dot{q}[n-2]}{dT} - \ddot{q}[n-1] \quad (4.24)$$

$$(4.25)$$

By making the last two accelerations in the trajectory depend on the previous accelerations, we can treat the optimization problem as unconstrained concerning the final desired joint angles and velocities.

4.6.2 Joint position, velocity and torque limits

Joint angle limits are introduced with a linear contact model. For the minimum torque and minimum torque-change optimization criteria we add an extra torque as in equation 4.26.

$$\bar{\tau}[n] = \begin{cases} \tau[n] - K_p(q_{max}[n] - q[n]) + K_d(0 - \dot{q}[n]), & \text{if } q[n] > q_{max}[n] \\ \tau[n] - K_p(q_{min}[n] - q[n]) + K_d(0 - \dot{q}[n]), & \text{if } q[n] < q_{min}[n] \\ \tau[n], & \text{otherwise} \end{cases} \quad (4.26)$$

Where K_p is the stiffness of the limit stop, K_d the damping parameter, $\ddot{q}[n]$ the acceleration the joint has to exert to overcome the limit stop and $\bar{\tau}[n]$ the extra torque the joint has to exert to overcome the limit stop. Because the very high acceleration or force associated with violating the limit stop, the optimization algorithm will converge to a solution where the joint angle constraint is not violated.

If velocity and/or torque limits are exceeded during the swing motion, it is probably because the swing duration is set to be too low. By using a slightly conservative swing duration, a swing motion optimized for one of the optimization criteria should stay clear of any velocity and torque limits. Therefore we chose to not include any velocity and torque limits in the optimization algorithm, reducing the computational burden and subsequently the time required to execute the optimization algorithm.

4.6.3 Ground clearance

Due the kinematics of the leg, foot height does not map linearly to joint angles. Therefore, the ground clearance constraint is a non-linear constraint. To include this non-linear constraint in the optimization and write the optimization criterion as an unconstrained problem, we use a penalty function for the ground clearance. Assuming a linear ground contact model, when the foot hits the ground a large reaction force will occur on the foot. Using the forward kinematics model, we can calculate the distance between the foot and the ground $h_{foot}(\mathbf{q})$, as well as the foot velocity $v_{foot}(\mathbf{q}, \dot{\mathbf{q}})$. The penalty for violating the ground clearance constraint is written as an upward force acting $\mathbf{f}[n]$ on the foot, according to

$$\mathbf{f}[n] = \begin{cases} \begin{bmatrix} 0 \\ 0 \\ K_z h_{foot}(\mathbf{q}[n]) - B_z v_{foot}(\mathbf{q}[n], \dot{\mathbf{q}}[n]) \end{bmatrix}, & \text{if } h_{foot}(\mathbf{q}[n]) < 0 \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top, & \text{if } h_{foot}(\mathbf{q}[n]) \geq 0 \end{cases} \quad (4.27)$$

Using the Jacobian from the hip roll to the knee expressed in ankle roll frame $\mathbf{J}_1^{4,3}(\mathbf{q})$, we can determine the contribution of the ground reaction force on each joint using the Jacobian transpose. The extra joint torques required to counteract the ground reaction force thus become

$$\bar{\tau}[n] = -\mathbf{J}_1^{4,3}(\mathbf{q})^\top \mathbf{f}[n] \quad (4.28)$$

The extra joint torque $\bar{\tau}[n]$ is added to the cost of the trajectory as defined in section 4.4. If the foot violates the ground contact constraint, the joint torques to execute the trajectory become very large and the cost to execute the trajectory is increased. If K_z is large enough, the cost to violate the ground contact constraint is much higher than the cost to avoid it and the optimization algorithm will converge to a solution where the ground contact constraint is not violated.

4.7 Jacobian of the cost function

The Levenberg-Marquardt algorithm requires Jacobian of the cost function. The Jacobian $\mathbf{J}_c$ is defined as the first order partial derivatives of the optimization criterion $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ with respect to the acceleration vector $\ddot{\mathbf{q}}$

$$\mathbf{J}_c = \frac{\partial \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})}{\partial \ddot{\mathbf{q}}} \quad (4.29)$$

There are two ways to determine the Jacobian, an analytical and numerical approach.

The advantage of the analytical approach is that it can be much faster than a numerical evaluation. Given the equations of motion of the swing leg (equation 4.15), the derivative of the equation of motion at $n = 1$ is given by

$$\frac{\partial \tau[1]}{\partial \ddot{\mathbf{q}}[1]} = \mathbf{M}(\mathbf{q}[0]) \quad (4.30)$$

Evaluating just the mass matrix is less expensive than three evaluations of the equation of motion. However, at $n = 2$, the angle $q[1]$ and the angular velocity $\dot{q}[1]$ are dependent on $\ddot{q}[1]$. Assuming the initial velocity $\dot{\mathbf{q}}[1]$ is zero, the corresponding joint torques at $n = 2$ become

$$\tau[2] = \mathbf{M}(\mathbf{q}[1] + \frac{1}{2}\ddot{\mathbf{q}}[1]^2)\ddot{\mathbf{q}}[2] + \mathbf{C}(\mathbf{q}[1] + \frac{1}{2}\ddot{\mathbf{q}}[1]dT^2, \dot{\mathbf{q}}[1]dT) \quad (4.31)$$

The corresponding derivative is now a lot more complex

$$\frac{\partial \tau[2]}{\partial \ddot{\mathbf{q}}[2]} = \frac{\partial \mathbf{M}}{\partial \mathbf{q}} \frac{\partial \mathbf{q}[1]}{\partial \ddot{\mathbf{q}}[1]} + \frac{\partial \mathbf{C}}{\partial \mathbf{q}} \frac{\partial \mathbf{q}[1]}{\partial \ddot{\mathbf{q}}[1]} \frac{\partial \mathbf{C}}{\partial \dot{\mathbf{q}}} \frac{\partial \dot{\mathbf{q}}[1]}{\partial \ddot{\mathbf{q}}[1]} \quad (4.32)$$

This requires us to find derivatives of the $\mathbf{M}(\mathbf{q})$ and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ matrices. Finding these derivatives is time consuming and depends on order of the joints in the robot leg, making the optimization algorithm less general. The second reason not to pursue an analytical solution is that the computational complexity approaches or even exceeds evaluating the cost function once for each variable.

4.8 Initial values

At the beginning of the step, all unconstrained accelerations are initialized to zero as the starting point for the optimization algorithm.

$$\ddot{\mathbf{q}}[n] = \mathbf{0} \quad (4.33)$$

Each subsequent step, the desired acceleration vector from the previous time step is used as a starting point for the new optimization, with all values shifted one position to the left.

$$\ddot{\mathbf{q}}[n-1] = \ddot{\mathbf{q}}[n] \quad (4.34)$$

Using the result of the previous optimization as the starting point reduces the amount of iterations necessary to converge to the minimum, decreasing computational cost. This allows us to stop the optimization routine when the optimization algorithm takes longer than the allotted time to converge. In the next step, fewer iterations are necessary to reach the minimum.

4.9 Numerical considerations

The linear equality constraint on the accelerations guarantees the resulting foot trajectory reaches the desired step location. However, even a small position error between the desired joint angle and the current joint angle can lead to very high joint accelerations if the remaining swing time is low. Not only are these high accelerations infeasible, they also lead to an unstable system that oscillates around the desired joint angle. Our solution is to define a minimum remaining swing duration. The minimum remaining swing duration is chosen such that the system does not oscillate when keeping the foot at a constant location.

4.10 Computational considerations

The computational complexity of the algorithm depends on several factors. The most important are:

- Number of function evaluations
- Cost to derive the Jacobian

For minimum torque and minimum torque change the computational complexity consists of calculating the accelerations for all sections from the linear constraint and consequently evaluating the dynamic model for each section. The majority of the time is spent evaluating the dynamic model.

The number of function evaluations is dependent on the optimization algorithm, the number of variables, the distance between the initial conditions and the optimal solution, and the shape of the cost function. For the Levenberg-Marquardt algorithm, about n^3 arithmetic operations are needed to process each evaluation of the cost function, where n is the number of variables. To approximate the Jacobian, n function evaluations and mn^2 arithmetic operations are necessary, where m is the number of outputs. The time to evaluate the cost function will generally determine the total execution time [22].

The computational cost of the cost function heavily depends on the computational cost of the inverse dynamics algorithm. The inverse dynamics algorithm has a computational complexity of $O(n)$, where n is the number of joints [13]. The inverse dynamics algorithm is executed once for each constant acceleration section.

The time the optimization algorithm takes is minimized by reducing the number of joints trajectories to optimize and by using a course trajectory with only a few constant acceleration sections. Reducing the number of joints that are optimized from six to three reduces the time taken for the optimization by a factor of approximately two. A course solution is suitable to use in a Model Predictive Controller, because the optimization algorithm is re-evaluated every control tick such that even a course solution results in a smooth trajectory.

4.11 Conclusion

Within the proposed framework, we can easily change out different cost functions. We have implemented two different cost functions that strive to minimize the actuator effort, namely minimum torque and minimum torque change.

A parametrization of the trajectory using accelerations allows the usage of a linear equality constraint to guarantee the optimization will always converge to joint trajectories that bring the foot to the desired step location.

A linear ground contact model provides an elegant way to penalize ground contact in the torque/acceleration framework.

Evaluating the Jacobian analytically would not give a large computational benefit to a numerical approximation while using an analytical Jacobian reduces the flexibility to use the optimization on other leg configurations. Computational considerations require us to keep the number of constant acceleration sections to a minimum.

Chapter 5

Minimum jerk controller

The swing controller we have introduced uses a lot of computational effort. This necessitates the use of an off-board controller running on one of the fastest computers available on the market. A high bandwidth, low latency link is required between the robot and the controller, making a wireless connection infeasible. If we want to run one of the robots completely autonomously without a tether, this makes the usage of this controller impossible for now. Therefore, we introduce a second controller with a much lower computational effort, which minimizes joint jerk using spline interpolation of the joint angles to be used on autonomous humanoids with less available computing resources.

5.1 Trajectory parametrization

The mapping from a foot position in the pelvis reference frame to the individual joint angles is given by the inverse kinematics solution, described in section 2.2.2. Using the method described in 4.2 to get an accurate estimate of the body position at the end of the step, we can calculate the joint angles corresponding to the desired step location. Given the joint angles and the joint velocities corresponding to the desired step location, we can describe the foot trajectory as a function of all joint angles. Because the actuators have finite bandwidth, the joint jerk, the derivative of accelerations, has to be finite. Furthermore, the initial position, velocity and acceleration as well as the final position, velocity and acceleration of the joint is fixed. This means six constraints. A 5th order (quintic) polynomial provides an interpolation between those two points that does not violate any of those constraints. For the derivation of the quintic spline, see appendix C.

However, there is an extra constraint on the foot position. The foot should stay clear of the ground at any point in the trajectory. To allow an extra degree of freedom, we chose to employ a quintic spline. By adding extra points between the initial and final angles, we allow freedom in choosing the trajectories, while still having a finite jerk and ending up at the desired step location. This allows us to determine the foot trajectory by choosing suitable knot points.

Knot points in the joint space trajectory can be mapped to the foot position. We can also look at it the other way, multiple desired foot positions during the swing can be mapped to the knot-points of the trajectory. The intermediate

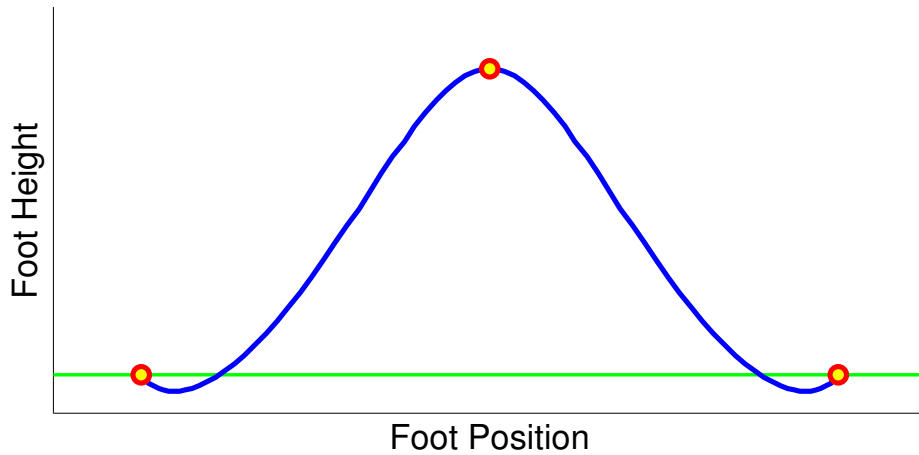


Figure 5.1: Resulting foot trajectory with one via-point. The ground is shown as a green line. Note that the final trajectory violates the ground clearance constraint.

desired foot positions are called via-points [24]. For an example of how the resulting foot trajectory will look using one via-point, see figure 5.1. Note that ground clearance is not guaranteed and depends on the position and number of via-points.

5.2 Choosing the Via-Points

To minimize the control effort necessary to track the trajectory, we chose to apply only one via-point. In figure 5.2, the placement of the via point is shown. A line is drawn between the current swing foot location and the desired step location. A 2 cm long line segment is drawn perpendicular to this line, facing away from the stance foot. The via point is placed on end of this perpendicular line. By moving the via-point away from the stance foot, we add extra clearance between the two legs, decreasing the likelihood of collisions between the feet.

5.3 Adjusting the trajectory

The angles corresponding to the via-point and desired step location are calculated using the inverse kinematics solution and an estimate of the pelvis position. However, the estimated pelvis position is based on a simple model and is updated continuously throughout the step. If we were to track the joint trajectories calculated at the beginning of the step, we would not reach the desired step location.

A straightforward solution would be to calculate a new via-point and a corresponding set of joint trajectories every control tick. However, the requirement that the foot has to pass the via-point and reach the desired point results in very high desired joint accelerations in practice. This happens when the foot is near a via-point and the velocity vector of the foot does not point directly to

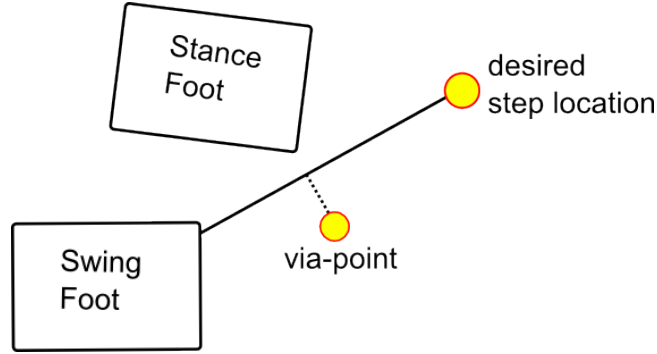


Figure 5.2: Via point placement on the ground. A line is drawn between the swing foot and the desired step location. The via point is placed on a line perpendicular to this control line, facing $2cm$ away from the stance foot.

the via-point, a very high acceleration is needed to move the foot through the via point. See figure 5.3 for a graphical representation.

To avoid these high accelerations, we first define a new reference frame named the *stance foot ZUp frame*. This frame has the x -axis pointing towards the toe of the stance foot and the z axis facing straight up in the opposite direction of gravity. The origin of this frame is the same as the origin of the stance ankle roll frame. During the swing motion this frame will stay stationary with respect to the world, assuming no slip of the stance foot. At the beginning of the step, we store the location of the via-point in the *stance foot ZUp frame* as well as the current joint angles, velocities and accelerations. During each control tick, the updated estimate of the pelvis position at the end of the step is used to calculate the joint angles corresponding to the saved via-point and desired step location. New joint trajectories are constructed from the saved angles at the beginning of the step, the angles corresponding to the via-point and the desired angles. These trajectories will only deviate slightly from the trajectories used for the previous control tick. Feedback control on the individual joints is used to compensate for the difference between the old and the new trajectory.

5.4 Trajectory tracking

The usage of quintic splines to describe the joint trajectories gives us a position, velocity and acceleration profile for each joint. We use a control strategy proposed by Craig [11] to accurately track the joint trajectories. The joint torque corresponding to a desired acceleration can be calculated using inverse dynamics, also taking into account the gravity working on the individual links. By choosing the desired acceleration to be the control output of the system, we can use the following PD controller to accurately control all joints

$$\ddot{q}_d = \ddot{q}_t + k_p(q_t - q_a) + k_d(\dot{q}_t - \dot{q}_a) \quad (5.1)$$

Where $\ddot{q}_d$ is the desired joint acceleration fed in to the inverse dynamics calculator, q_t , $\dot{q}_t$ and $\ddot{q}_t$ are the joint position, velocity and acceleration calculated

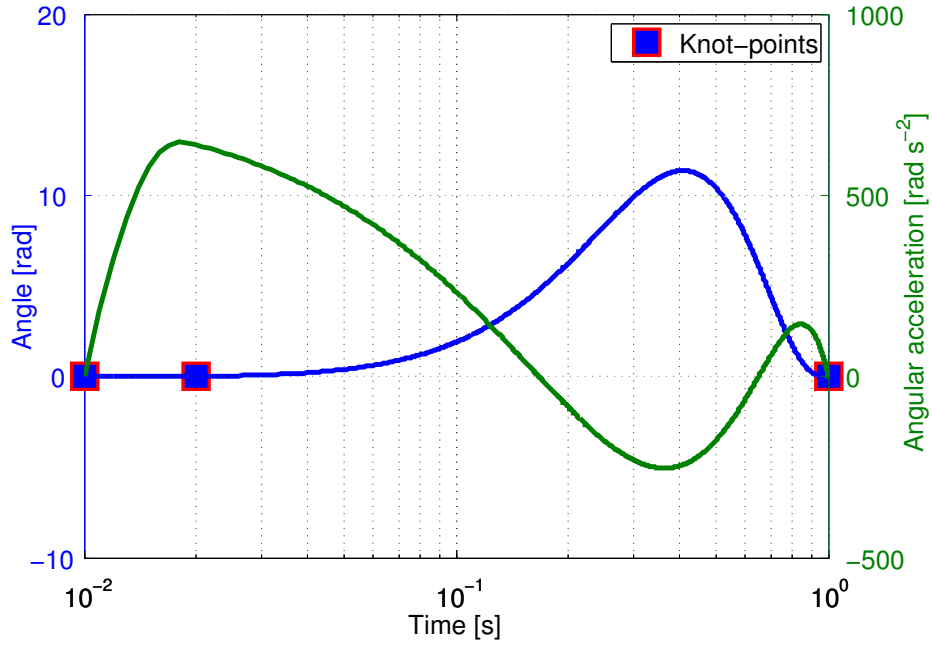


Figure 5.3: High desired joint acceleration due to the knot point laying close to the current joint angle. Notice the log scale.

from the joint trajectory, respectively and q_a and $\dot{q}_a$ are the actual joint position and velocity, respectively.

5.5 Conclusion

We introduced a second controller that guarantees smooth joint trajectories at a much lower computational costs than the model predictive controller. The drawback of this controller is that ground clearance at the beginning and the end of the trajectory is not guaranteed, making it possible that the feet will scuff the ground. Another drawback is that the controller cannot handle large changes in desired step location.

Chapter 6

Implementation

The control algorithms are tested in simulation before being run on the real robot. With the exception of the Kalman filter and the inverse kinematics, the control algorithms are all developed for and tested on M2V2. In this chapter we describe the simulation environment and the implementation of the control algorithm on the physical robot.

6.1 Simulation environment

A full dynamic simulation of M2V2 is available. The simulation is built on top of Simulation Construction Set (SCS) [25] and both the simulation environment and the controller are written in the Java programming language.

Joint accelerations are calculated from the desired joint torques using Featherstone's articulated rigid body algorithm [13]. A fourth order Runge-Kutta integrator with a 0.1 ms step size is used to simulate the motion of the robot. The ground contact forces are determined using a spring-damper ground contact model. Ground contact occurs between a fixed ground contact point on the feet and a ground surface $z = f(x, y)$. The simulation gives real-time feedback to the user about the current state of the robot and is able to plot all internal variables.

The position sensors on the robot are simulated by discretization of the actual values of the simulated joint angle. The discretization reduces the resolution of the simulated position sensors to correspond to the incremental encoders on the robot. The joint velocities are calculated by a numerical differentiation of the discretized signals.

The simulated actuators are limited by a torque-speed curve to simulate the behavior of the robot actuators. A low-pass filter is applied to the torque signal to simulate the limited bandwidth of the real series elastic actuators.

6.2 Control software on the physical robot

The control algorithms for the simulation can be run directly on the robot using the Robot Construction Set (RCS). RCS is built on top of the Real-Time Specification for Java (RTSJ) [26, 27] and uses a periodic thread to enable real

time control. Controllers written for the simulation environment run unchanged on the physical robot.

Sensor acquisition and the control signals for the motors are done using a number of ISA cards. ISA is an older bus protocol with a relatively low data rate, which gives us a maximum loop rate of 666.67 Hz for the low level control. The computer on the robot communicates with an off-board high performance computer using standard Ethernet that runs two control loops. One high-level control loop at a loop rate of 666.67 Hz and the optimization loop for the swing controller running 6 times slower at 111.11 Hz.

The on-board computer is based on the PC/104 standard and has an Intel Core Duo L2400 processor running at 1.6 Ghz with 2 Gb of memory. The Ethernet controller provides a 100 Mbps data link to the high level control computer. The high level control computer has an Intel Xeon X5660 processor running at 2.8 GHz with 12 Gb of memory.

6.2.1 Computational considerations

Java code is run on the Java Virtual Machine (JVM). Java is compiled into a platform-independent bytecode that is interpreted by the JVM. Interpretation of the bytecode is relatively slow compared to traditionally compiled languages like C++, because no optimizations are done on the bytecode before execution. Modern JVMs improve performance by using Just-in-Time (JIT) compilation. A JIT compiler will optimize and compile frequently executed parts of the code.

Memory allocated by the program is freed by the Garbage Collector when it is no longer referenced anywhere in the program. Care has to be taken by the garbage collector to not interfere with the execution of the control loop. In the next paragraphs we will show how JIT compilation and Garbage Collection affect determinism of the execution of the code and how we avoid these problems in our controller.

JIT Compilation

JIT compilation drastically improves the performance of the Java VM [28]. Our optimization algorithm is computationally expensive and is not fast enough to run interpreted. Several methods can be employed to guarantee compilation of the code to meet real-time deadlines at the first control ticks

- Ahead-Of-Time compilation
- Force JIT-compilation by executing it often enough

Ahead-Of-Time (AOT) compilation is available in Websphere Real Time [29]. Before running the program, it is compiled and optimized by a special compiler. This improves determinism, however throughput is slightly reduced because the AOT compiler doesn't have runtime information about the execution paths available, in contrast to the JIT compiler.

Initial tests with Websphere Real Time showed that the usage of AOT compilation reduces the execution time by approximately 30%. However, JIT compilation reduces the execution time by a factor of 2 or more. Therefore, we initialize the control modules by executing them many times using simulated input data until the execution time is satisfactory.

Garbage collection

A Garbage Collector keeps track of the memory used by the program and frees any memory that is no longer referenced anywhere in the program. During a garbage collection cycle, the Garbage Collector determines if an object is still used anywhere in the code and if not, frees the associated memory. We can make the distinction between three types of garbage collectors [30, 31]

- Stop-the-world collectors
- Incremental collectors
- Concurrent collectors

A stop-the-world collector will halt the execution of the program till the whole garbage collection cycle is done. This can take up to several seconds and will heavily impact the determinism of the controller. The advantage of the stop-the-world collector is that it uses the least amount of resources of all three kinds and has the highest throughput measured over the total lifetime of the program.

An incremental collector interleaves garbage collection with the execution of the main program, allowing useful work to be done during a collection cycle. In comparison to the stop-the-world collector, the time taken to perform garbage collection is increased, reducing overall throughput of the program but increasing the responsiveness.

A concurrent collector runs in parallel to the main thread and does not halt the execution of the main thread allowing for the greatest determinism at the cost of increased resource usage.

For control purposes determinism is more important than throughput, thus ruling out a stop-the-world collector. If the worst case pause times of the incremental collector are small enough, it can be used for real time control. For the highest determinism, it is necessary to use a fully concurrent garbage collector.

By avoiding the creation of new objects that are discarded at the end of one execution of the control loop, we reduce the amount of memory that needs to be freed and avoid the need to execute a garbage collection cycle. This improves throughput and makes the controller more deterministic at the cost of reduced flexibility while programming the controller.

6.3 Conclusion

Using an accurate dynamic simulation of the robot allows us to quickly and accurately test our control algorithm. Simulated sensors and actuators mimic the limitations of the Series Elastic Actuators in the real robot.

We develop the control algorithms in Java and run the same code on the real robot. The Java Real-Time System allows us to run the control loop at a sufficiently fast rate. Care is taken to reduce the amount of object creation and discarding in the control loop to reduce the overhead of the garbage collector.

Chapter 7

Results

The MPC and minimum jerk controllers are compared with the controller described in 1.1, which we will refer to as the foot trajectory controller. Using the Simulation Construction Set with an accurate model of M2V2 we simulate different scenarios for the robot. The Simulation Construction Set allows scripting of events enabling us to accurately reproduce tasks for the robot in order to compare the different controllers with each other. For the model predictive controller, the number of constant acceleration sections and the minimum swing duration as described in section 4.9 have to be determined. In section 7.1 we find these parameters for the simulation of M2V2. In section 7.2, we will compare the three controllers using simulations of three different scenarios

- Forward step, see section 7.2.1.
- Forward step with a change in desired foot step location during the swing, see section 7.2.2.
- Push recovery, see section 7.2.3.

The forward step and forward step with changing step location are performed while the robot is suspended in the air. These tests compare the performance of the swing controllers without interference from the balance algorithm. The push recovery tests the performance of the swing controllers while the pelvis is in motion.

To show that we can accurately predict the capture point, we have tested the CoM velocity estimation for the capture point calculation on TULIP. We will present the results in section 7.3.

The Oracle/Sun JRTS has a performance hit of about 50% compared to the normal Java VM, making it infeasible to run the model predictive controller on the real robot using available computational resources. We think that available computing power will increase sufficiently in the next 1 to 2 years to enable successful usage of the model predictive controller on a real robot.

7.1 MPC parameters

The model predictive controller is tuned by two separate variables. The first variable is the number of constant acceleration sections, which defines the maxi-

imum loop rate of the model predictive controller. Measurement of the optimization time versus the number of constant acceleration sections is given in section 7.1.1. The second variable is the minimum remaining swing time, determined in section 7.1.2.

7.1.1 Number of constant acceleration sections

The number of constant accelerations sections describing the joint trajectories is the major influence in the total required calculation time and thus determines the maximum loop rate. Increasing the number of sections results in a finer grained trajectory and will result in a lower overall cost as defined by the cost function. The required loop rate of 111.11 Hz, or a time per tick of 9 ms, puts an upper limit on the number of sections. We simulate two swing motions and measure the average calculation time per optimization. To account for artifacts from the JIT compiler and the garbage collector we throw away the data from the first step. On the real robot, this would be replaced by simulating one step with dummy data to warmup the optimization routines. For the average time taken per optimization step, see figure 7.1.

Number of sections	MPC - Minimum Torque	MPC - Minimum Torque Change
3	0.0012	0.0016
4	0.0034	0.0037
5	0.0058	0.0070
6	0.0099	0.011
7	0.014	0.016
8	0.019	0.021
9	0.025	0.026
10	0.041	0.033

Table 7.1: Time taken per optimization step for different cost functions averaged over 1.6 seconds for 3 to 10 sections with constant acceleration.

With a loop rate of 111.11 Hz, we cannot go above 5 piece-wise accelerations for any of the cost functions. All further tests are run with five piece-wise accelerations.

7.1.2 Minimum Swing Duration

As described in 4.9, a minimum swing time needs to be defined to keep the maximum accelerations within bounds and keep the controller stable. A lower minimum remaining swing time results in a lower positioning error at the end of the swing. However, when the minimum remaining swing time is too small oscillations occur in the desired output torques and the system becomes unstable.

To find the minimum remaining swing time, the foot is controlled to hold a constant position using the model predictive controller. The minimum remaining swing time is then decreased until oscillations occur in the desired joint torques. The minimum remaining swing time for both costs functions is shown in table 7.2.

Cost function	Minimum time [s]
Minimum torque	0.27
Minimum torque-change	0.25

Table 7.2: Minimum remaining swing time.

7.2 Controller performance

7.2.1 Forward step

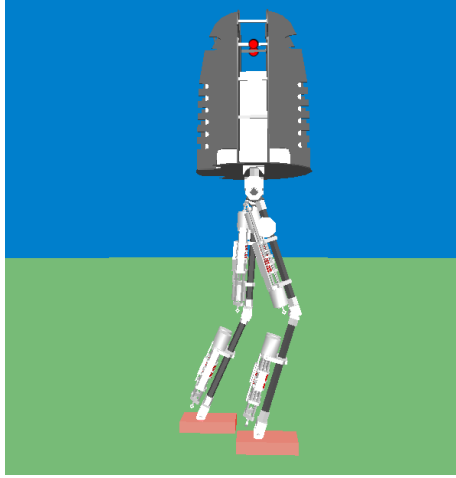


Figure 7.1: Initial conditions for the robot for the forward step with the left foot. The robot is suspended in the air by a force applied to the two attachment points depicted by a red circle on the robot.

To minimize the influence of the balance controller on the performance of the swing controller, the robot is suspended in the air as shown in figure 7.1 and the balance controller is disabled.

The robot is instructed to make a forward step to a constant desired step location. After the foot reached the desired step location, it should hold position. The desired swing duration is set to 0.4 s and the error at the end of the step is measured at $t = 0.7$ s. During this test, the right foot is considered the stance foot and is flat on the ground. The left foot is considered the swing foot. The initial position of the swing foot with respect to the stance foot $p(0)$ is

$$p(0) = \begin{bmatrix} -0.20\text{m} \\ 0.22\text{m} \\ 0.0\text{m} \end{bmatrix} \quad (7.1)$$

The final desired position with respect to the stance foot $p(0.4)$ is

$$p(0.4) = \begin{bmatrix} 0.20\text{m} \\ 0.22\text{m} \\ 0.0\text{m} \end{bmatrix} \quad (7.2)$$

We compare the results of the model predictive controller and minimum jerk controller with the foot trajectory controller as described in section 1.1 and implemented on M2V2. The simulations are run on the M2V2 model in Simulation Construction Set. The resulting swing motion is shown as a series of time-lapsed animations in figure 7.2.



Figure 7.2: From left to right a graphical representation of the state of the robot for $t = 0.0$ s, $t = 0.1$ s, $t = 0.2$ s, $t = 0.3$ s, and $t = 0.4$ s is shown for a forward step. The different controllers are shown from top to bottom: a) MPC - Minimum torque cost function, b) MPC - Minimum torque change cost function, c) Minimum jerk controller, and d) Foot trajectory controller. Note that the images are mirrored for clarity.

Actuator torques

The goal of the proposed controllers is to accurately position the foot on the desired step location while taking advantage of the natural dynamics of the swing leg. This would correspond to minimal actuator torques during most parts of the swing. The recorded actuator torques for the cost functions of the model predictive controller, the minimum jerk controller and the foot trajectory controller are plotted in figure 7.3.

The foot trajectory controller exhibits multiple sections of approximately continuous torque, in comparison with the model predictive controller which shows an initial torque step followed by a smooth trajectory. Clearly seen in the graphs is the jagged curve due to the reduced update rate for the joint torques of the model predictive controller in comparison to the minimum jerk and foot trajectory controllers. At the end of the swing motion ($t = 0.35$ to $t = 0.4$) a large positive braking torque is applied on the hip pitch for the foot trajectory controller. This will decelerate the foot so it will come to a stop before the end of the swing motion. The braking torque for the model predictive controller is much lower, avoiding negative work. The minimum jerk controller shows higher joint torques in the hip roll which are a result of extra clearance around the stance leg as described in 5.2.

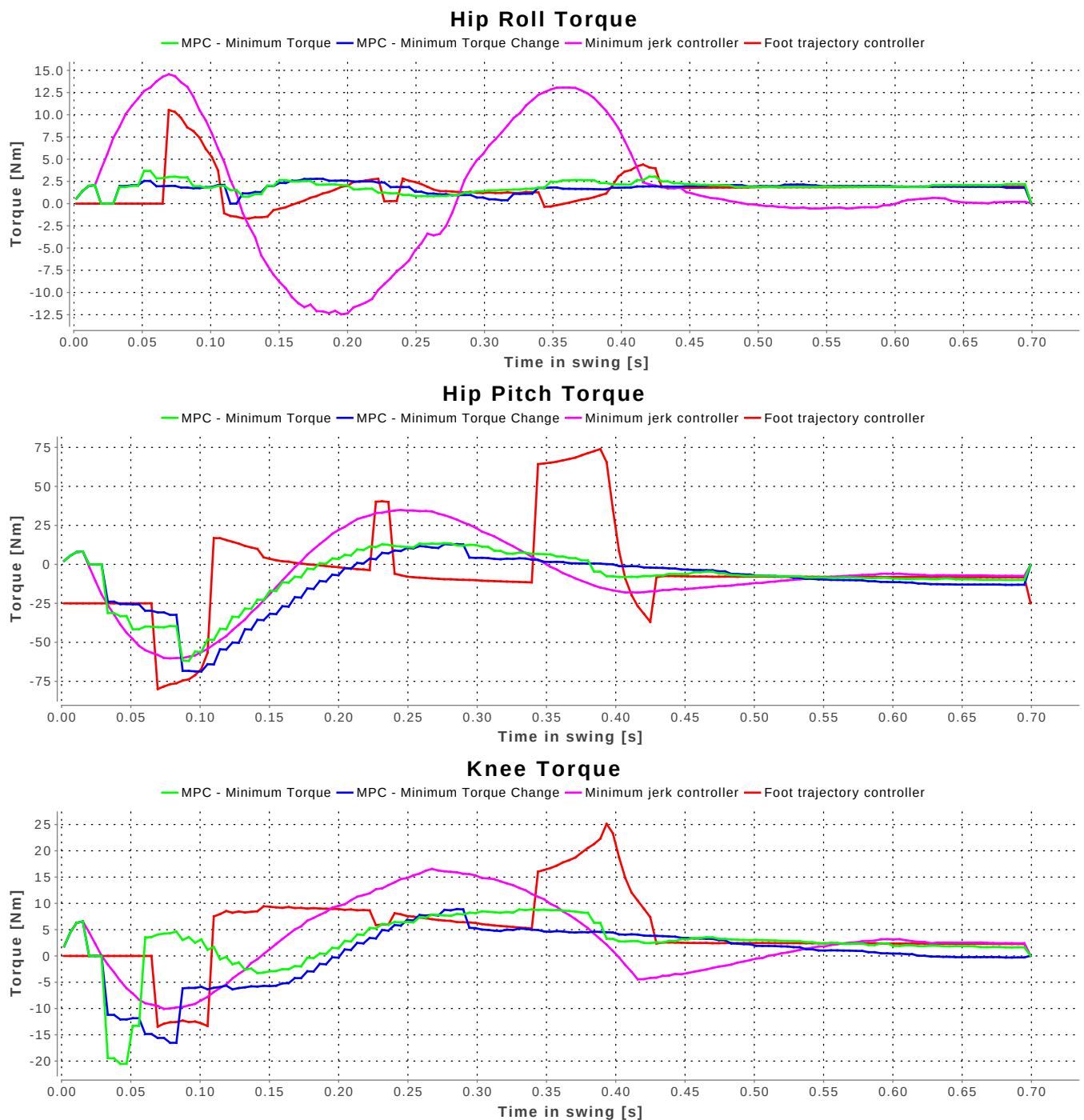


Figure 7.3: Actuator torques of the hip roll, hip pitch and knee for the cost functions of the model predictive controller, the minimum jerk controller and the foot trajectory controller.

Bandwidth

The maximum force control bandwidth of the Series Elastic Actuators is around 40 Hz for M2V2. If the desired torque trajectories exceed this bandwidth requirement, the actuators cannot physically execute the trajectory. In figure 7.4 we've taken the Fourier transform of the desired torque trajectories for the three joints. For the hip pitch, we can see that the magnitude of the frequency components between 10 – 50 Hz is two to three times larger for the foot trajectory controller than the other controllers. The frequency magnitude for both cost functions of the model predictive controller are comparable for the whole frequency range. The required actuator bandwidth for the model predictive controller is approximately 30 Hz. The large hip roll magnitude in the frequency range of 1 – 10 Hz for the minimum jerk controller can be attributed to the extra required clearance of the swing leg with respect to the stance leg.

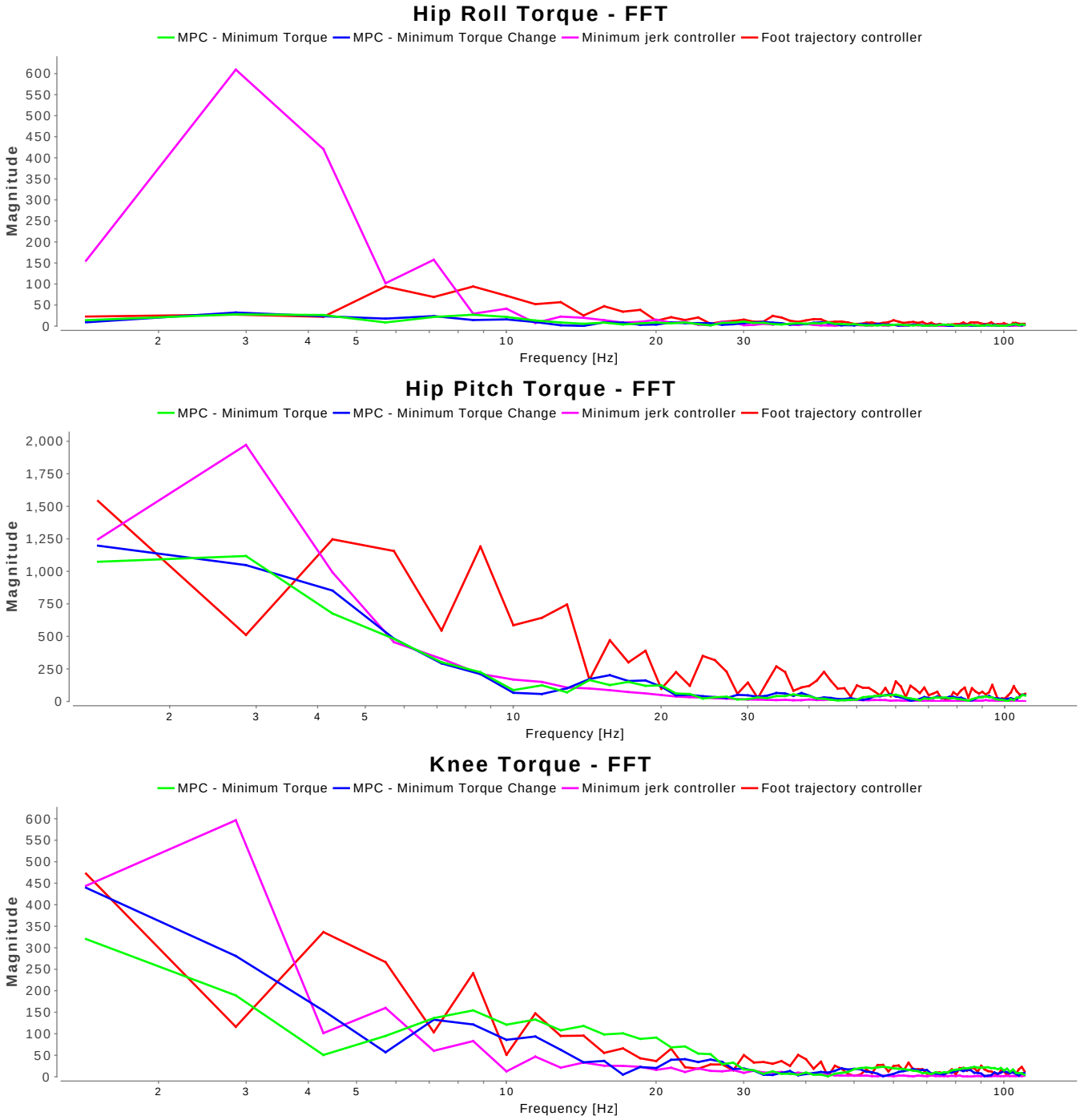


Figure 7.4: Frequency plots of the desired actuator torques as desired by both cost functions of the model predictive controller, the minimum jerk controller and the foot trajectory controller.

Joint trajectories

We recorded the actual joint positions and velocities during the swing motion. The results are shown in respectively figure 7.5 and 7.6. At the desired swing time ($t = 0.4\text{s}$) the model predictive controller is close to the desired step location, but both the foot trajectory and minimum jerk controllers still have a relatively large error. When we look at the joint velocities, we see that the velocity of both cost functions of the model predictive controller are near zero while the foot trajectory controller still has large joint velocities. This results in a foot that still has a considerable velocity with respect to the ground at the end of the swing.

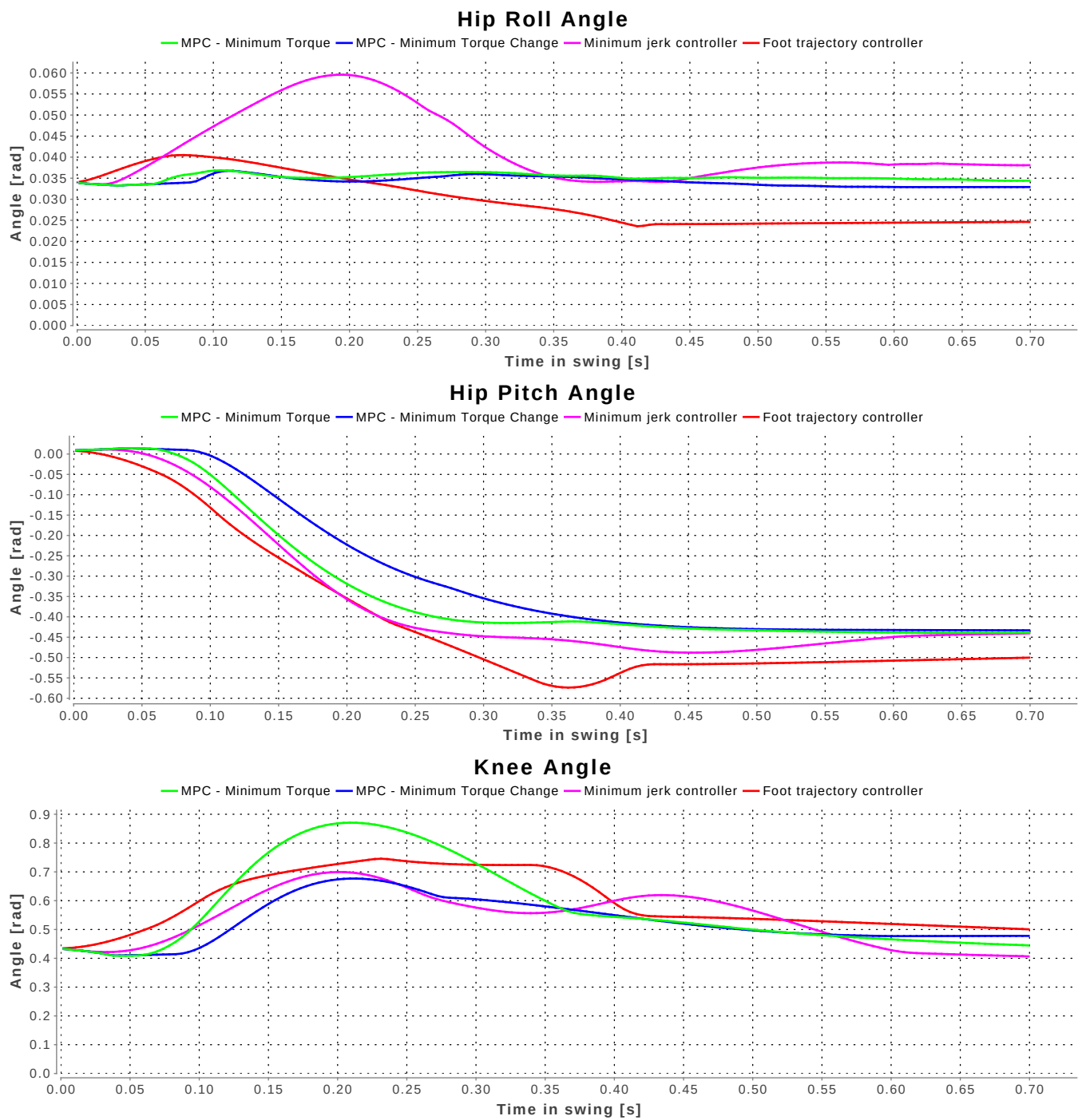


Figure 7.5: Position of the hip roll, hip pitch and knee for the cost functions of the model predictive controller, the minimum jerk controller and the foot trajectory controller.

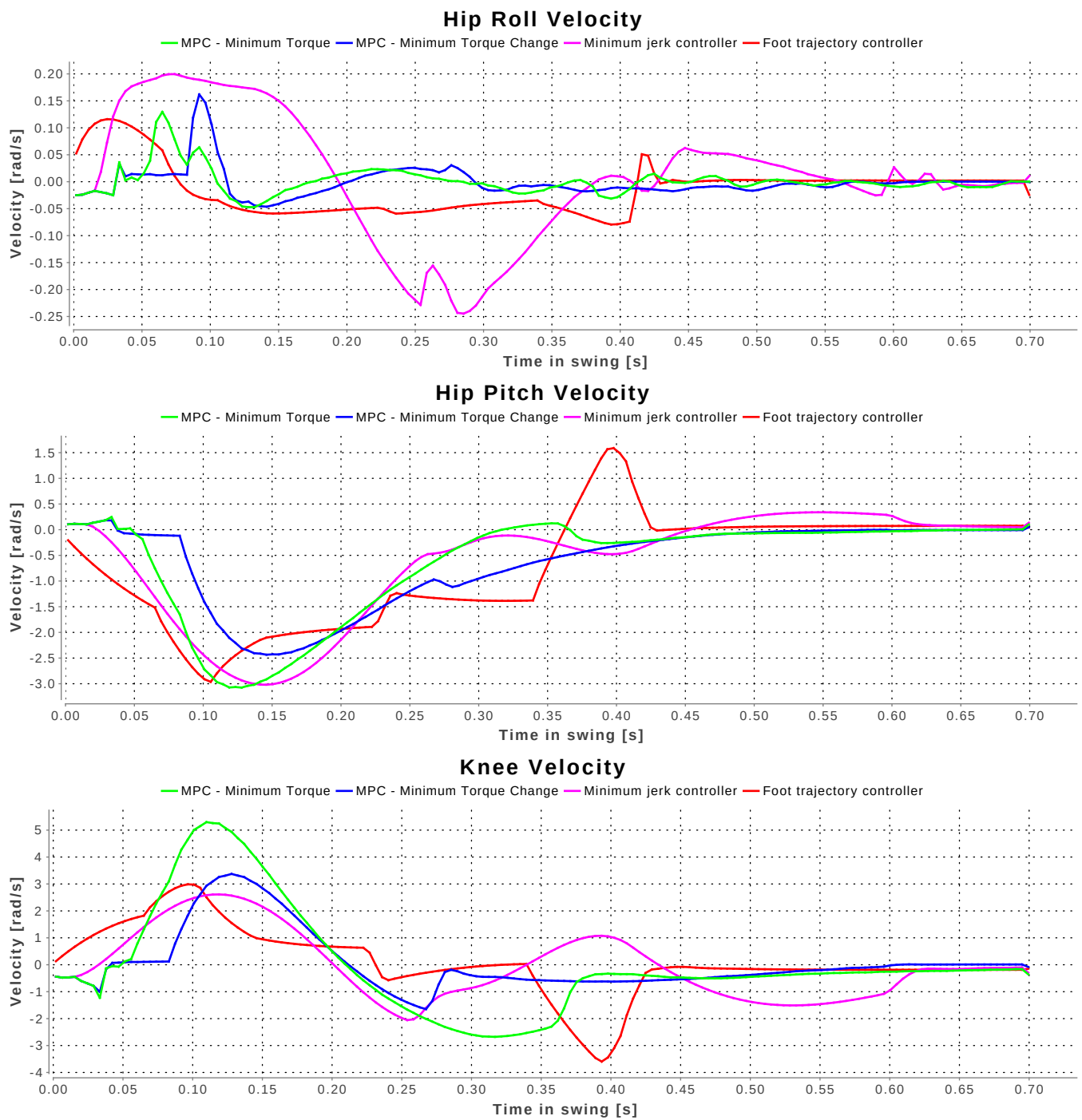


Figure 7.6: Velocities of the hip roll, hip pitch and knee for the cost functions of the model predictive controller, the minimum jerk controller and the foot trajectory controller.

Ground clearance

An important aspect of the performance of the swing controller is the ground clearance. The ground clearance measured in simulation is shown in figure 7.7.

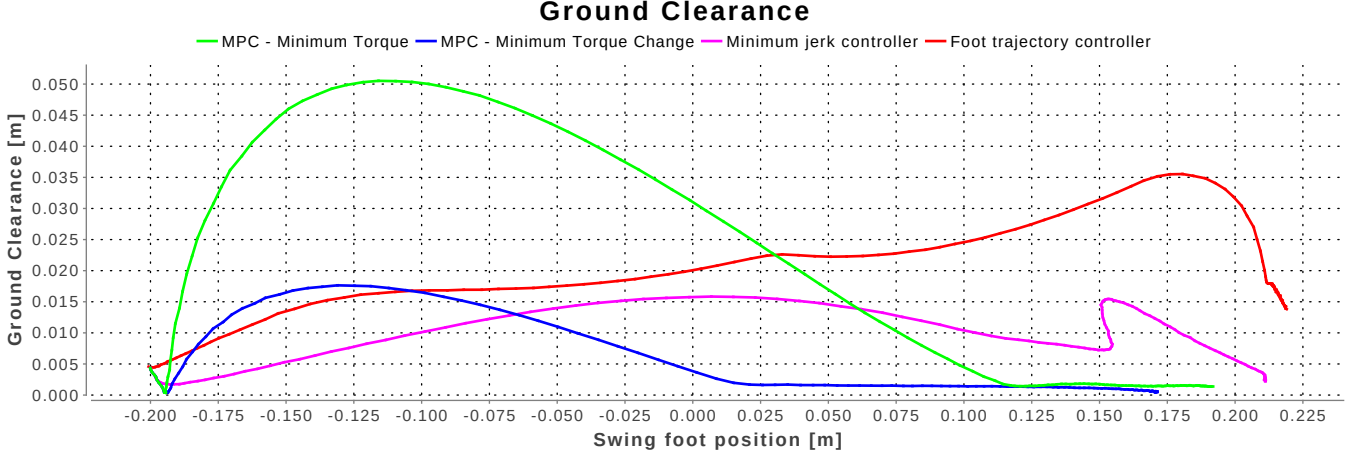


Figure 7.7: Ground clearance of the different controllers as function of the swing foot position in the sagittal plane.

The ground clearance for the model predictive controller depends on the cost function. The minimum torque cost function results in a ground clearance that is almost twice as large as the minimum torque change cost function. Near the end of the step, both cost functions prefer a motion of the foot that minimizes the distance between the ground and the foot. The ground clearance for the foot trajectory controller is more constant during the swing motion, however towards the end of the step it increases before putting the foot down on the ground. Because the foot overshoots the desired step location, the heel is planted in the ground first and the midpoint of the foot cannot reach the ground.

Performance characteristics

The relative performance of the three controllers and two cost functions is compared using four variables

- The peak torque. High peak torques are undesirable due increased actuator wear.
- The sum of squares of the torques during the trajectory.
- Mechanical energy, calculated as $\sum_{n=0}^z \|\tau_n \dot{q}_n \Delta t\|$.
- Error, the absolute positioning error at the end of the step.

A total of nine measurements are done, all starting from the same initial condition but with a different desired step location $p_i(0.4)$, where i is the index of the measurement. See table 7.3 for the desired step locations.

	x	y	z
$p_1(0.4)$	0.23 m	0.25 m	0.0 m
$p_2(0.4)$	0.23 m	0.22 m	0.0 m
$p_3(0.4)$	0.23 m	0.19 m	0.0 m
$p_4(0.4)$	0.20 m	0.25 m	0.0 m
$p_5(0.4)$	0.20 m	0.22 m	0.0 m
$p_6(0.4)$	0.20 m	0.19 m	0.0 m
$p_7(0.4)$	0.17 m	0.25 m	0.0 m
$p_8(0.4)$	0.17 m	0.22 m	0.0 m
$p_9(0.4)$	0.17 m	0.19 m	0.0 m

Table 7.3: Desired step locations for performance measurements.

The mean and the standard deviation σ of the peak torque, sum of squares of torques, mechanical energy and step error of 9 executions of the swing with different desired step locations are tabulated in 7.4. For the peak torque the highest torque measured in all three joints is taken. The sum of squares of torques and the mechanical energy is the combined value for all three joints.

Controller	Peak Torque [Nm] (σ)	$\sum \tau^2$ [Nm ²] (σ)	Energy [J] (σ)	Error [m] (σ)
MPC - Minimum Torque	61.2(10.1)	59139.5(11597.0)	14.4(1.43)	0.0109(0.00485)
MPC - Minimum Torque Change	66.0(6.22)	66665.5(6307.54)	11.7(0.956)	0.0315(0.00676)
Minimum jerk controller	62.9(13.4)	123838.0(41633.7)	21.6(4.23)	0.0117(0.00583)
Foot trajectory controller	79.6(0.764)	144094.0(1215.39)	22.3(0.557)	0.0223(0.0014)

Table 7.4: Comparison between the performance of the different controllers and cost functions for a forward swing. The mean and standard deviation over 9 runs

The Welch's t-test for sample sets with unequal variances is used to test if two measurements have a statistical significant difference. A difference between two values is considered significant if the p-value (p) is less than 0.05.

Compared to the foot trajectory controller, there is a significant ($p < 0.01$) reduction in peak torque for the model predictive controllers and the minimum jerk controller. A significant reduction ($p < 0.01$) in the sum of squares of the joint torques is seen for the model predictive controller compared to the foot trajectory controller.

Mechanical energy consumption is significantly ($p < 0.01$) reduced for the model predictive controller. The minimum torque change cost function has the lowest overall energy consumption (19% smaller than the minimum torque cost function).

The model predictive controller with a minimum torque cost function and the minimum jerk controller both have a significant ($p < 0.01$) reduction of the error at the end of the step. The error is reduced by approximately 50%. However, the minimum torque-change cost function shows a significant increase ($p < 0.01$) in the error at the end of the step.

7.2.2 Forward step with a changing desired step location

To test handling a changing desired step location during the swing, we modify the previous test case by changing the final desired swing location at $t = 0.2s$. We move the desired step location 10 cm to the left such that the adjusted desired step location $\bar{p}(0.4)$ becomes

$$\bar{p}(0.4) = \begin{bmatrix} 0.20m \\ 0.32m \\ 0.0m \end{bmatrix} \quad (7.3)$$

With this test, we simulate a large disturbance on the robot during the swing motion that necessitates a change in desired step location. The minimum jerk controller is not able to cope with changes in step location during the swing and is excluded from this test. The resulting swing motion is shown as a series of time-lapsed animations in figure 7.8.

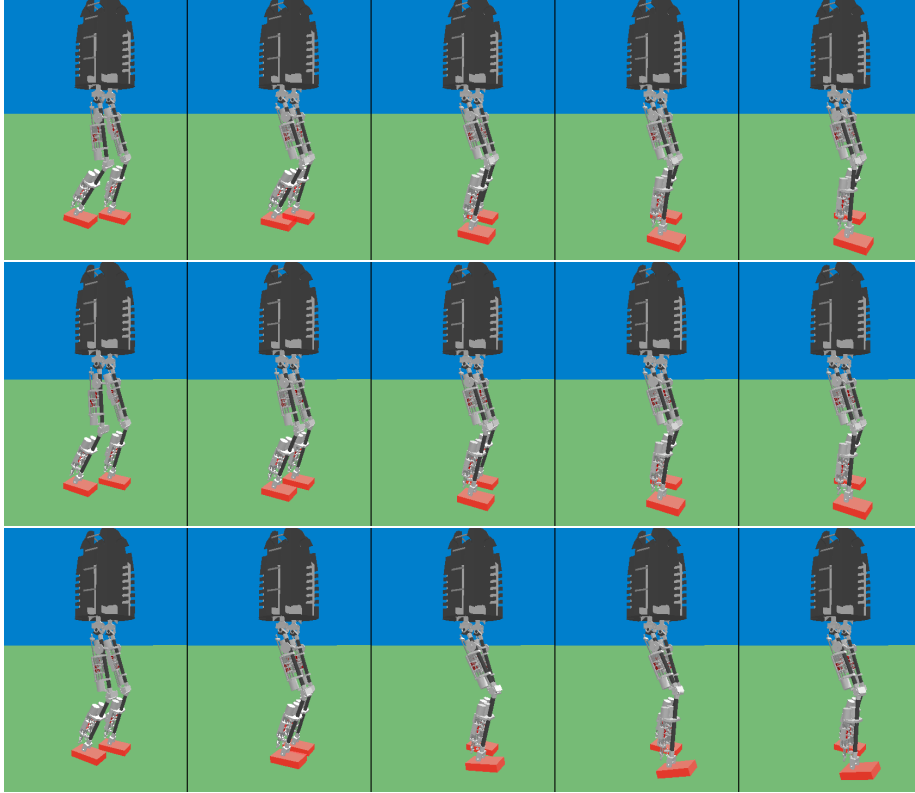


Figure 7.8: From left to right a graphical representation of the state of the robot for $t = 0.0$ s, $t = 0.1$ s, $t = 0.2$ s, $t = 0.3$ s, and $t = 0.4$ s is shown for a forward step with a changing desired step location. The desired step location is changed at $t = 0.2$ s, represented by the middle panels. The different controllers are shown from top to bottom: a) MPC - Minimum torque cost function, b) MPC - Minimum torque change cost function, and c) Foot trajectory controller. Note that the images are mirrored for clarity.

Actuator torques

The recorded actuator torques for the different cost functions of the model predictive controller and the foot trajectory controller for a changing desired step location during the swing are plotted in figure 7.9. At $t = 0.2\text{s}$ the desired step position changes. As expected, there is a peak in actuator torque for the hip roll for all three controllers at that point. However, the peak is much smaller for the model predictive controllers in comparison to the foot trajectory controller. The hip pitch and knee torque curves are comparable with the curve for the forward step without a changing desired step location. Just as in the forward motion without a changing desired step location we see large torques near the end of the swing motion ($t = 0.35$ to $t = 0.40$) for the foot trajectory controller that are not present for the model predictive controller.

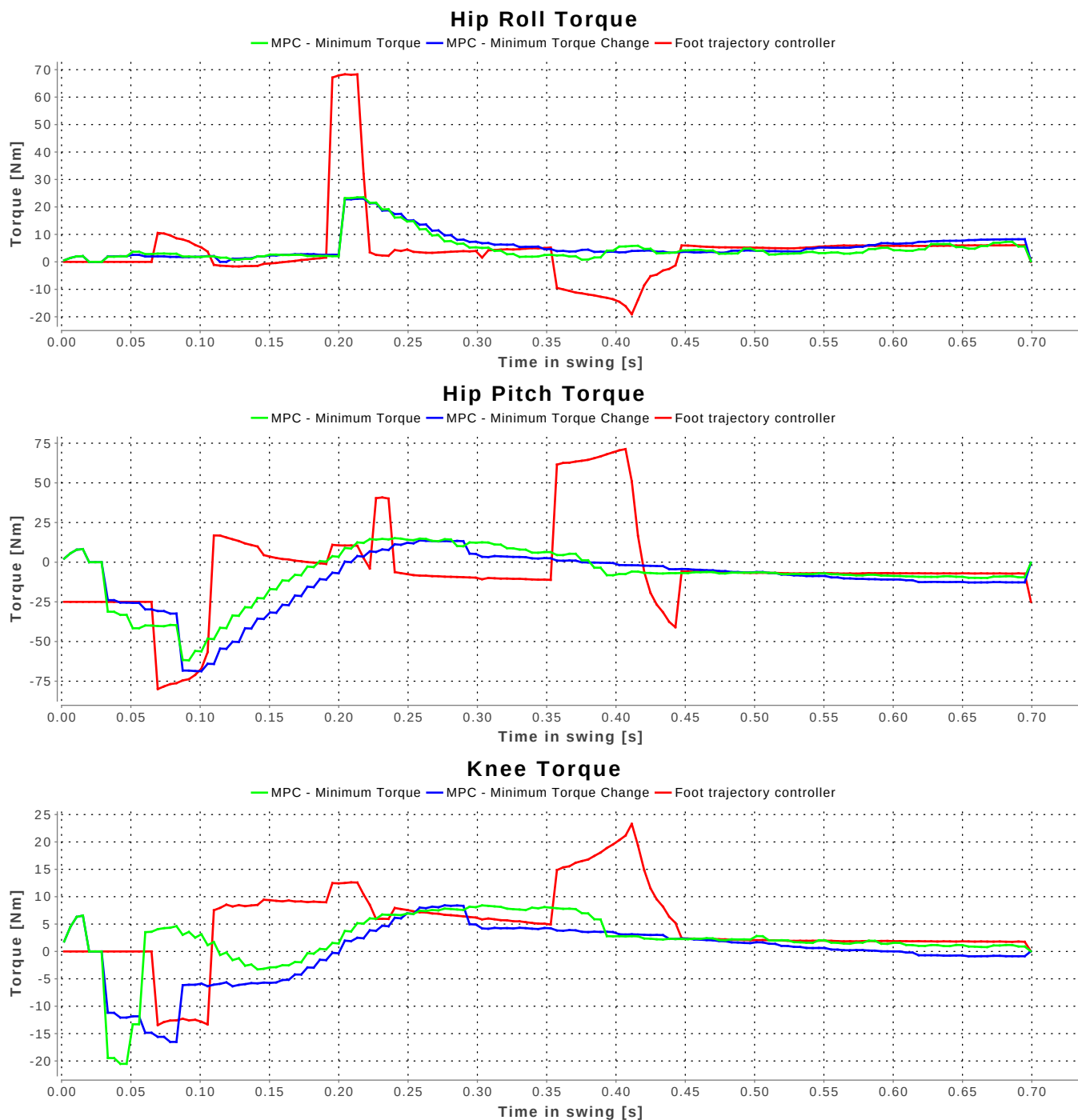


Figure 7.9: Actuator torques of the hip roll, hip pitch and knee for the cost functions of the model predictive controller and the foot trajectory controller for a swing with a changing desired step location.

Joint positions

The corresponding joint positions and velocities for the different cost functions of the model predictive controller and the foot trajectory controller for a changing desired step location during the swing are plotted in figure 7.10. We can see that at $t = 0.2$ s the hip roll angle starts changing from approximately 0.0 radians to approximately 0.14 radians to compensate for the changing desired step location. The slope of the hip roll angle for the model predictive controller is smaller, as expected from the lower torques.

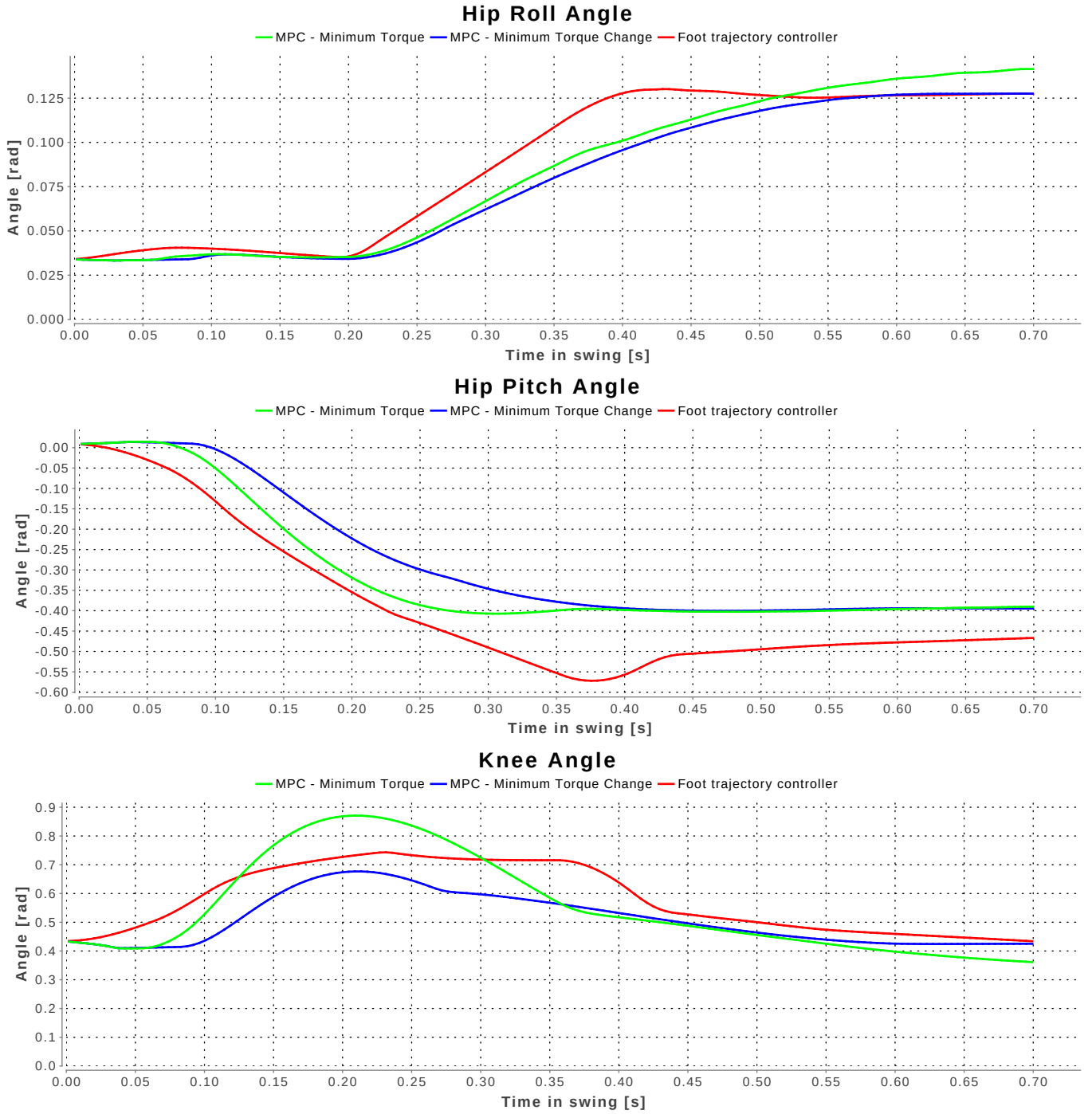


Figure 7.10: Position of the hip roll, hip pitch and knee for the cost functions of the model predictive controller and the foot trajectory controller for a swing with a changing desired step location.

Performance characteristics

For this task we also recorded the peak torque, sum of squared actuator torques, mechanical energy consumption and the error at the end of the step. The initial desired swing foot position $p_i(0.4)$ is given by table 7.3. At $t = 0.2$ s the desired step location is changed to $\bar{p}_i(0.4)$. The values for $\bar{p}_i(0.4)$ are given in table 7.5.

	x	y	z
$\bar{p}_1(0.4)$	0.2 m	0.30 m	0.0 m
$\bar{p}_2(0.4)$	0.2 m	0.275 m	0.0 m
$\bar{p}_3(0.4)$	0.2 m	0.25 m	0.0 m
$\bar{p}_4(0.4)$	0.17 m	0.3 m	0.0 m
$\bar{p}_5(0.4)$	0.17 m	0.275 m	0.0 m
$\bar{p}_6(0.4)$	0.17 m	0.25 m	0.0 m
$\bar{p}_7(0.4)$	0.15 m	0.3 m	0.0 m
$\bar{p}_8(0.4)$	0.15 m	0.275 m	0.0 m
$\bar{p}_9(0.4)$	0.15 m	0.25 m	0.0 m

Table 7.5: Desired changed step locations for performance measurements.

The mean value and the standard deviation σ are tabulated in 7.6.

Controller	Peak Torque [Nm] (σ)	$\sum \tau^2$ [Nm ²] (σ)	Energy [J] (σ)	Error [m] (σ)
MPC - Minimum Torque	62.3(10.4)	62577.0(10319.6)	14.9(0.937)	0.0117(0.00715)
MPC - Minimum Torque Change	65.4(5.07)	68113.5(4822.48)	11.5(0.766)	0.0377(0.00918)
Foot trajectory controller	78.0(0.647)	156058.0(2648.67)	22.2(0.649)	0.0184(0.00162)

Table 7.6: Comparison between the performance of the different controllers and cost functions for a forward swing with a changing desired step location.

For this task we also see a significant reduction ($p < 0.01$) in the peak torques for the model predictive controller. Likewise, the sum of squared joint torques is also significantly reduced for both cost functions ($p < 0.01$). Energy consumption during the swing is also significantly lower and again the minimum torque change cost function shows a significantly reduced energy consumption compared to the minimum torque cost function ($p < 0.01$).

The foot trajectory controller has a smaller error at the end of the step compared to the forward step without changing desired step location, however it is still significantly ($p \approx 0.02$) larger than the minimum torque change cost function. Again, the minimum torque change cost function has a significantly higher end of step error.

7.2.3 Push recovery

At the beginning of the push recovery task, the robot is standing on its right leg and a force is applied on the upper body, see figure 7.11. For a successful recovery, the robot has to accurately position its feet on the estimated capture point at the end of the swing motion. During this task, the pelvis moves with respect to the ground and an accurate estimation for the pelvis position at the end of the swing is necessary to calculate the final desired joint angles of the swing leg. We apply a force $\mathbf{F}$ for 0.2 seconds, where $\mathbf{F}$ is

$$\mathbf{F} = \begin{bmatrix} 0\text{N} \\ -70\text{N} \\ 0\text{N} \end{bmatrix} \quad (7.4)$$

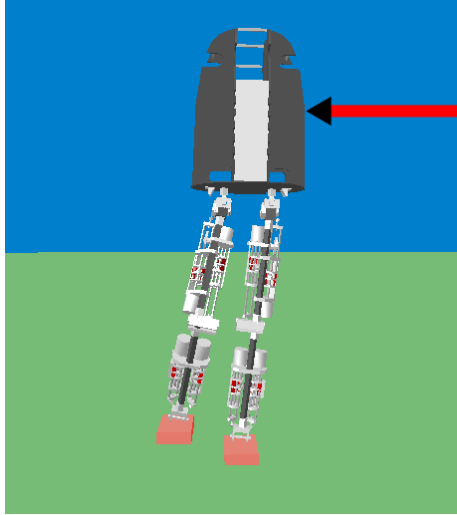


Figure 7.11: Initial conditions for the push recovery task, seen from the front of the robot. The robot is standing on its left leg and has its right leg in the air. A force is applied in the direction of the red arrow.

The resulting push recovery is shown as a series of time-lapsed animations in figure 7.12. All controllers are able to successfully regain balance after this disturbance.

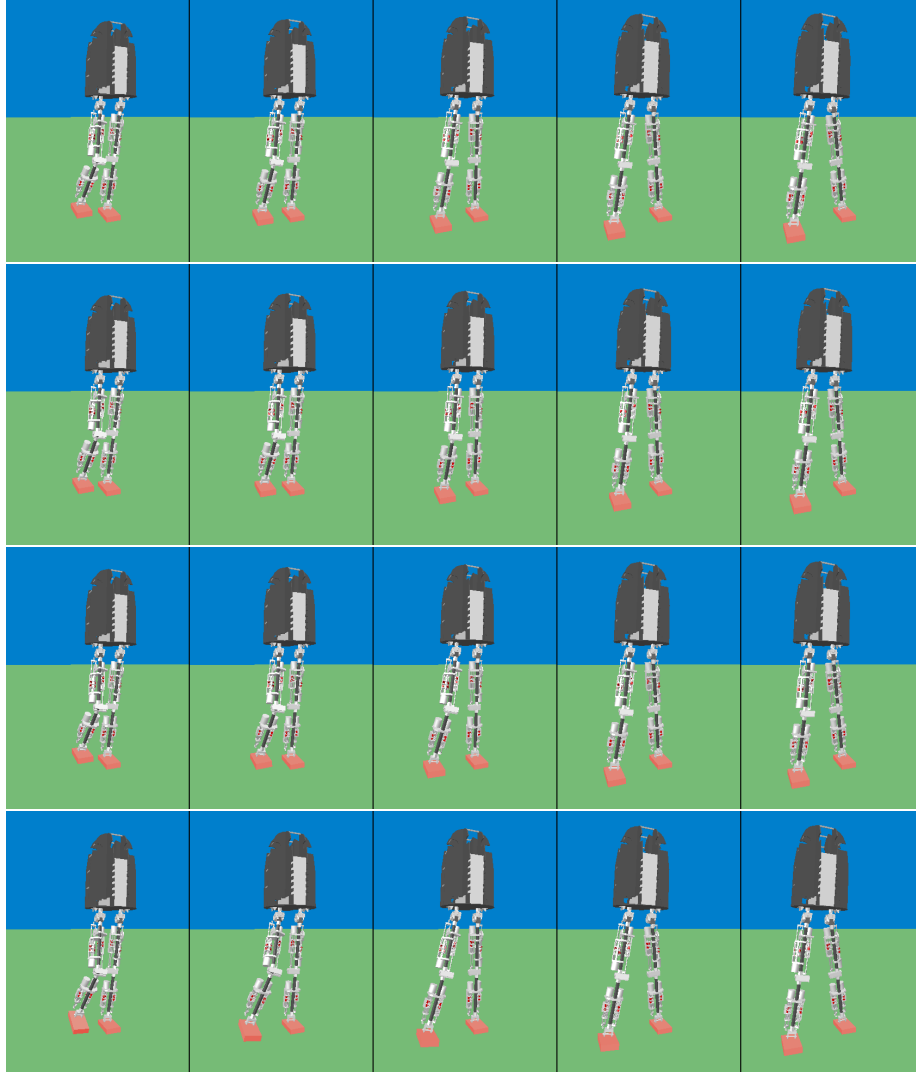


Figure 7.12: From left to right a graphical representation of the state of the robot at 0%, 20%, 40%, 60%, 80% and 100% of the motion is shown for push recovery. The different controllers are shown from top to bottom: a) MPC - Minimum torque cost function, b) MPC - Minimum torque change cost function, c) Minimum jerk controller, and d) Foot trajectory controller.

Pelvis position estimation

We measured the location of the pelvis with respect to world during the recovery. Ideally, the estimated location of the pelvis should stay constant during the recovery and be equal to the pelvis position at the end of the swing ($t = 0.65s$). In figure 7.13 we have plotted the estimation of the pelvis position at every moment in time with respect to the actual pelvis position. We can see that for the y-direction, the estimation is within 10% of the actual pelvis position for the better part of the swing duration. For the x-direction, there is a large

overshoot in the estimated pelvis position. During the push recovery, the center of pressure is actively controlled, influencing the estimation. Because the foot is longer than it is width, more control can be applied in the x-direction than the y-direction, negatively influencing the pelvis position prediction.

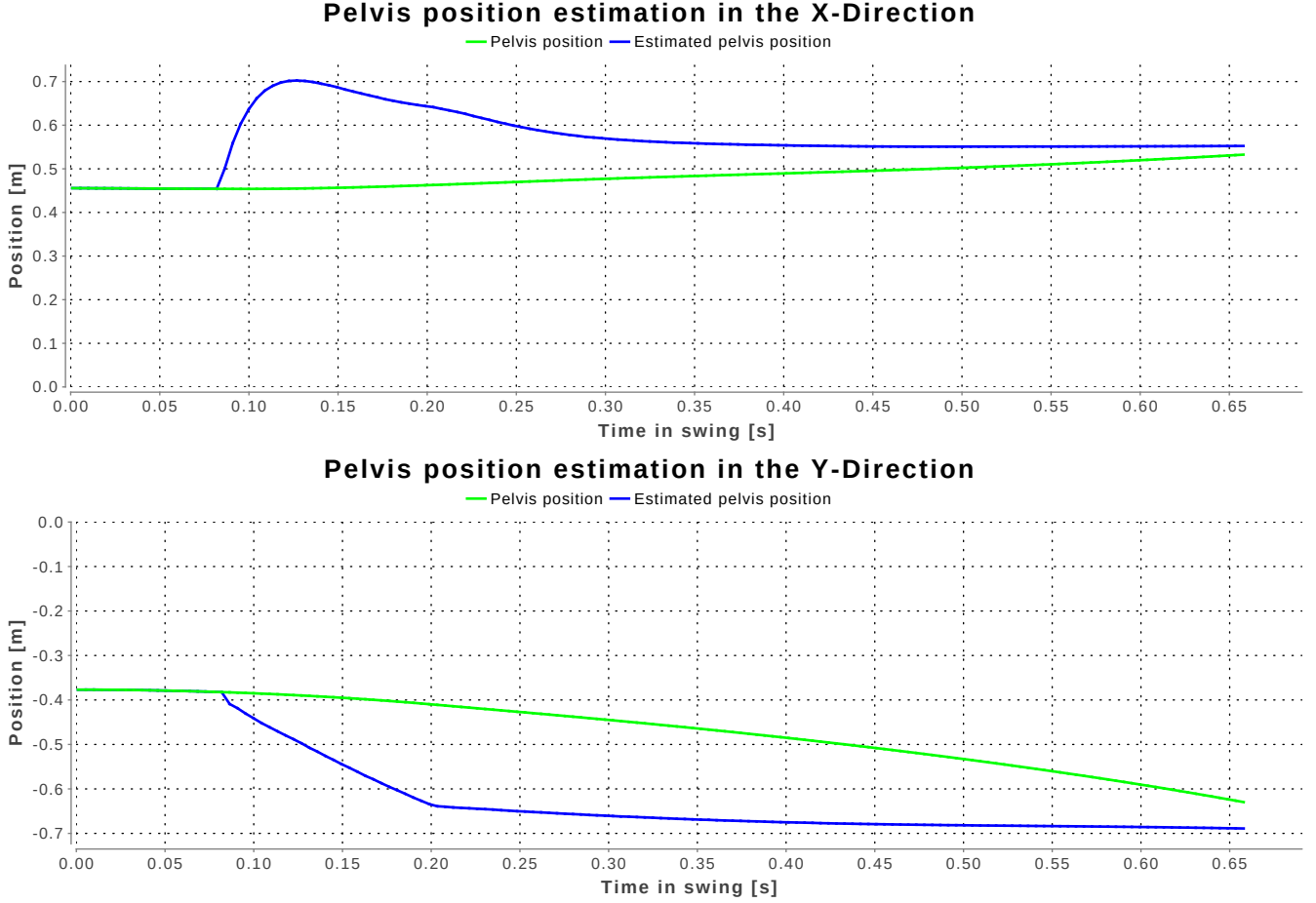


Figure 7.13: Pelvis position estimation during push recovery compared to the actual pelvis position.

Actuator torques

The desired actuator torques during push recovery are shown in figure 7.14. The actuator torques during push recovery are more constant than the foot trajectory controller, in comparison to multiple torque spikes in the foot trajectory controller.

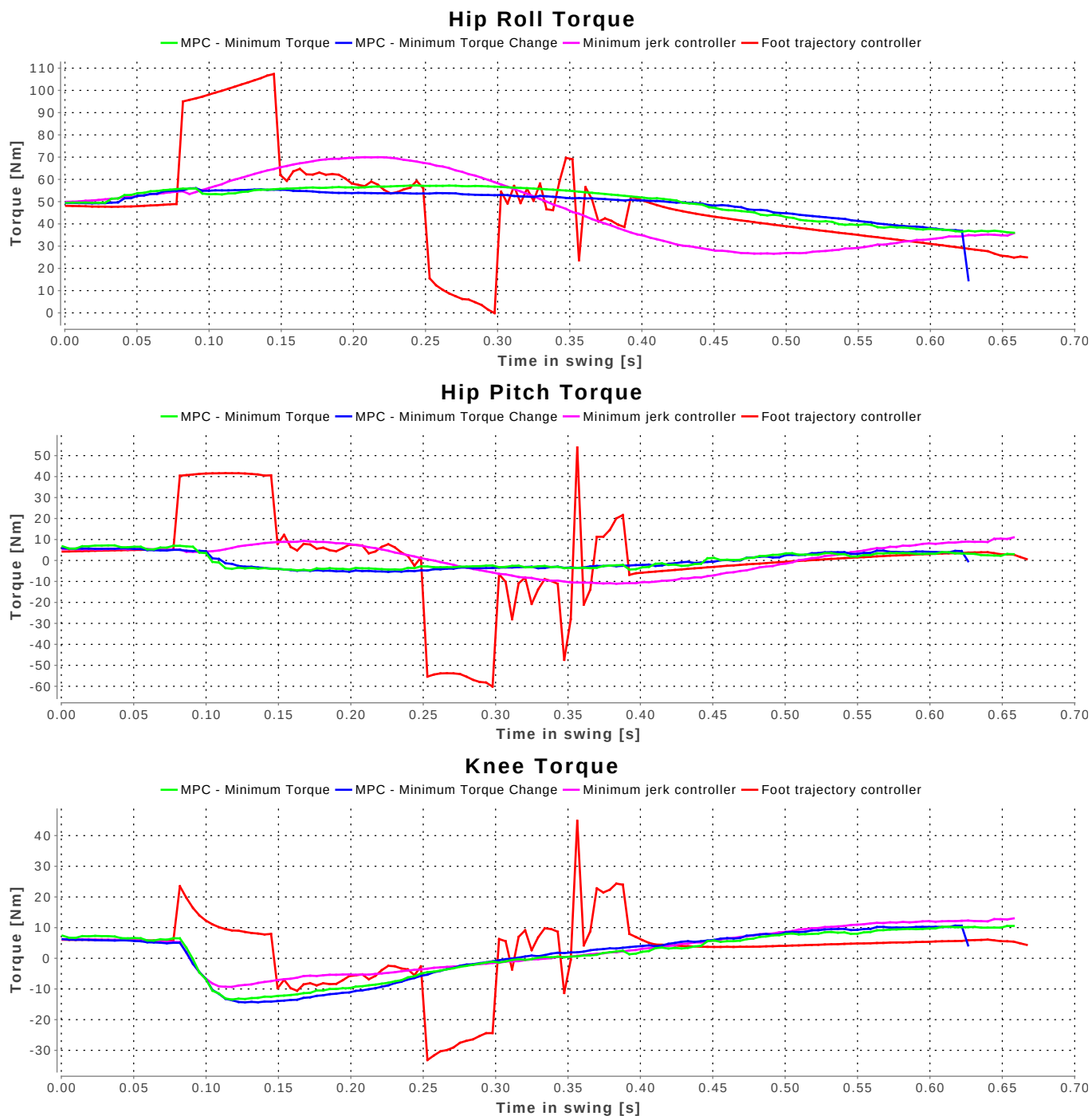


Figure 7.14: Actuator torques of the hip roll, hip pitch and knee for the cost functions of the model predictive controller and the foot trajectory controller during push recovery.

Bandwidth

Frequency plots of the desired actuator torques during push recovery are shown in figure 7.15. As expected from the desired torques of figure 7.14, the bandwidth requirements for the model predictive controller and minimum jerk controller are lower for all joint compared to the foot trajectory controller in the range from 3 – 100 Hz.

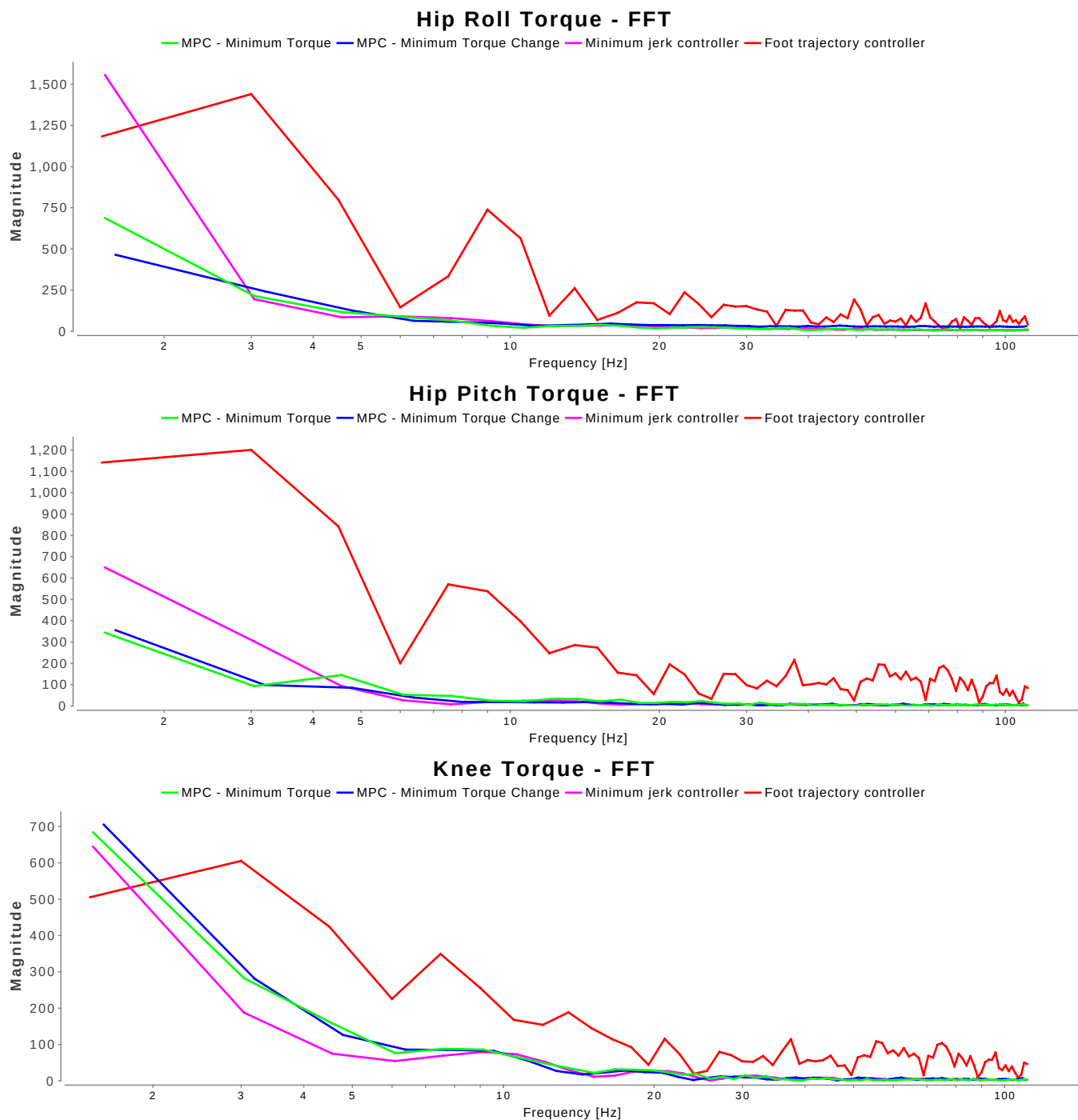


Figure 7.15: Frequency plots of the desired actuator torques as desired by both cost functions of the model predictive controller, the minimum jerk controller and the foot trajectory controller during push recovery.

Joint positons

The corresponding joint positions for the different cost functions of the model predictive controller and the foot trajectory controller during push recovery are plotted in figure 7.16. The joint angle trajectories for the model predictive controllers and the minimum jerk controller are almost the same, however the trajectories and final angles for the foot trajectory controller differ significantly for both the hip pitch and knee angle. The final angle at the hip pitch is approximately 0.03 radians smaller, while the final angle for the knee is approximately 0.03 radians larger. Due the kinematic configuration of the robot, this leads to a foot position that is slightly higher but at approximately the same position in the (x,y) plane.

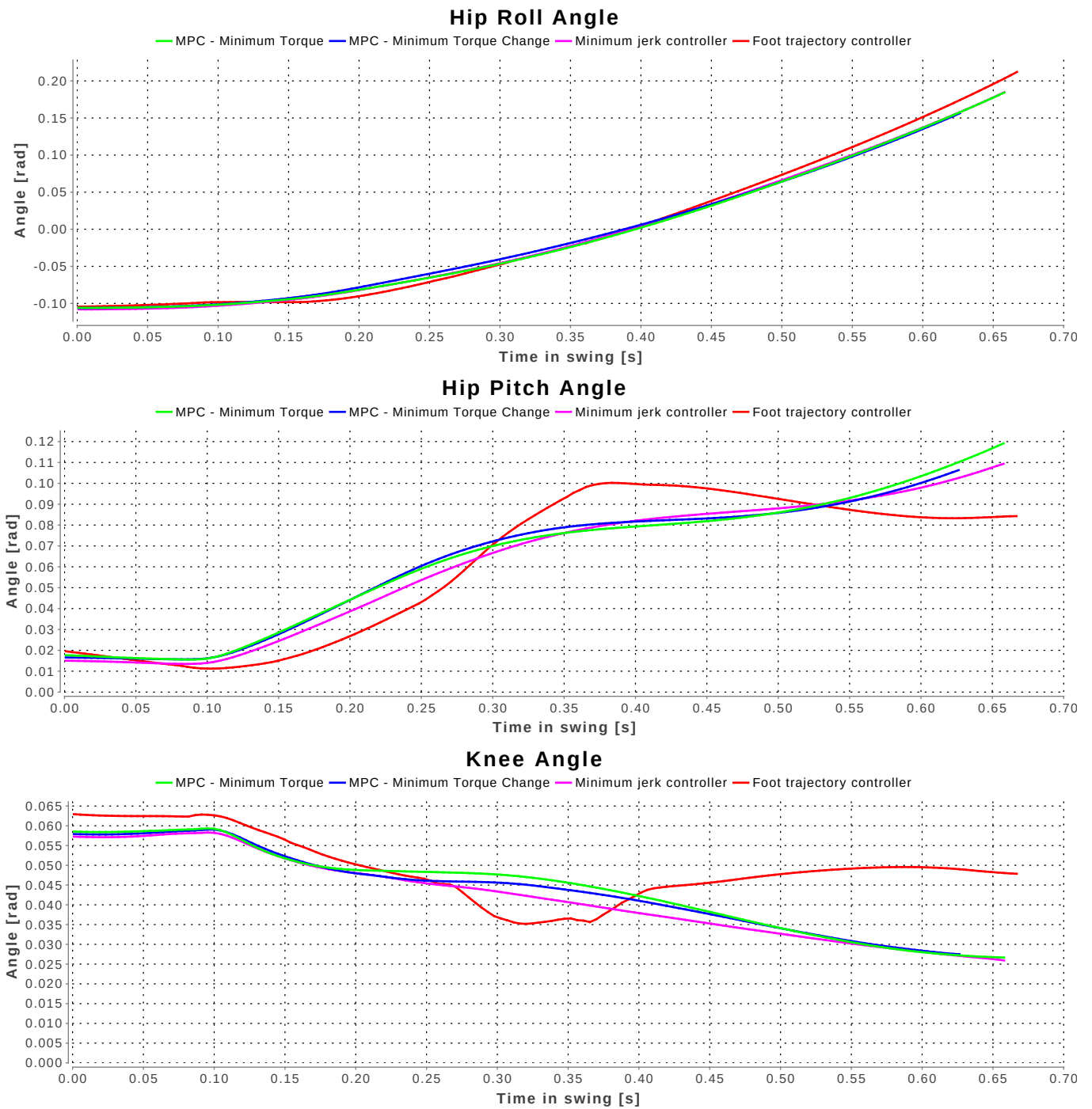


Figure 7.16: Position of the hip roll, hip pitch and knee for the cost functions of the model predictive controller and the foot trajectory controller during push recovery.

Performance characteristics

Push recovery performance is compared for a range of disturbance forces $\mathbf{F}_i$, see table 7.7.

	x	y	z
$\mathbf{F}_4$	0 N	-65 N	0 N
$\mathbf{F}_5$	0 N	-70 N	0 N
$\mathbf{F}_6$	0 N	-75 N	0 N

Table 7.7: Forces applied on the robot for push recovery

For the push recovery task, we measured the peak torques, sum of squares of the joint torques, mechanical energy consumption and the error at the end of the step. The mean values for the three trials as well as the standard deviation σ are tabulated in 7.8.

Controller	Peak Torque [Nm] (σ)	$\sum \tau^2$ [Nm ²] (σ)	Energy [J] (σ)	Error [m] (σ)
MPC - Minimum Torque	57.4(0.3)	385318.0(23941.2)	14.1(0.751)	0.0522(0.02)
MPC - Minimum Torque Change	56.0(0.211)	361814.0(22732.9)	12.9(0.639)	0.0877(0.0323)
Minimum jerk controller	70.0(0.315)	373333.0(12561.8)	12.6(0.133)	0.0636(0.00499)
Foot trajectory controller	107.0(0.186)	518506.0(14551.8)	15.5(0.022)	0.0429(0.00193)

Table 7.8: Comparison between the performance of the different controllers and cost functions during push recovery.

The MPC controller almost halves the peak torque required during push recovery. While in simulation, the corresponding torques for the foot trajectory controller are feasible these are unrealistically high for the physical robot. The error at the end of the step is slightly increased for the minimum torque cost function compared to the foot trajectory controller ($p \approx 0.015$), however this did not affect the ability to regain balance. The reduction in energy consumption for the minimum torque cost function compared to the foot trajectory controller is not significant.

7.3 Capture point calculation

To calculate the capture point, it is necessary to provide an accurate CoM position and velocity. Sources of error in the CoM position calculation are

- Inaccurate information on the CoM position of the individual links of the robot.
- Measurement errors in the joint position.
- Non-rigid links.
- Body orientation estimate.

To minimize the error in CoM position of the individual links, TULip was disassembled and the CoM of the individual links was measured [32]. The joint positions for the hip pitch, knee and ankle joints is measured by high precision

incremental encoders. Initial joint angles are set by fixating the joint angles using a calibration jig. The encoders are directly connected to the joint axle and the maximum error is determined by the resolution of the encoder (≈ 0.003 radians). The hip yaw and hip roll angle are measured by an incremental encoder on the motor shaft, which is coupled to the joint axle by a reduction gear. The error is determined by the maximum play in the gearbox (≈ 0.05 radians). Non-rigid links can result in a large error in CoM position. We have not determined significant bending of parts in the sagittal plane on TULip. The high precision incremental encoders in combination with an accurate measurement of the CoM position of the individual links will provide us with an accurate estimate of the CoM position in the sagittal plane. Due the large error in the hip roll angle, there will be a large error in the estimate of the CoM position in the coronal plane.

The biggest influence in the accuracy of the capture point calculation in the sagittal plane will be the CoM velocity estimation. We implemented the Kalman filter on TULip and logged the velocity of the IMU calculated using the velocity kinematics, the accelerations of the IMU and the velocity of the IMU estimated using the Kalman filter in the sagittal plane while the robot was actively balancing and being pushed. In figure 7.17, we plotted the measured velocity, the estimated velocity and the measured accelerations integrated over time. We can see that the drift of the accelerometer integration is compensated for and that the resulting velocity estimation is less noisy than just the pelvis velocity kinematics.

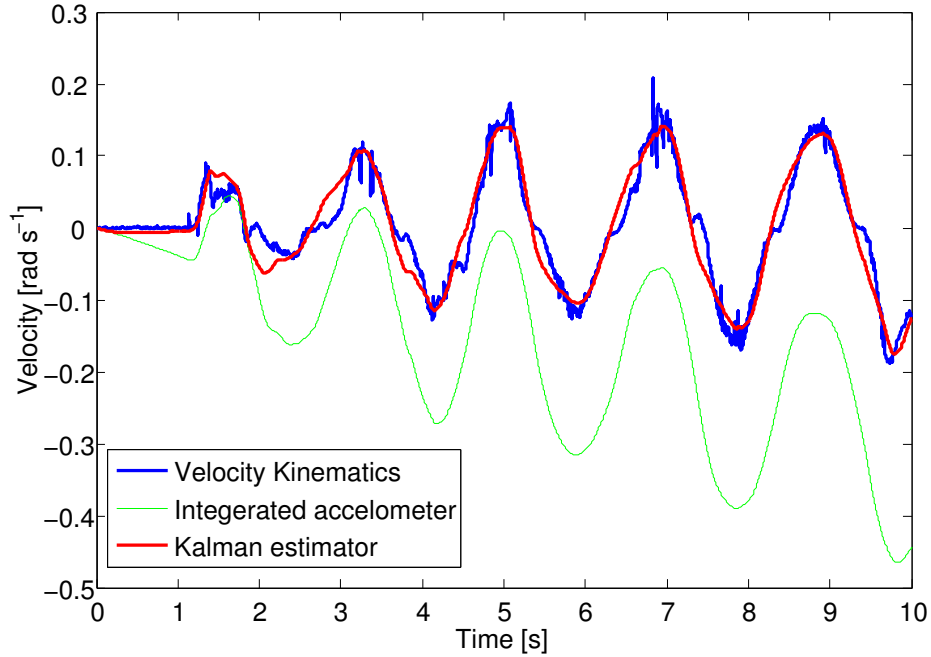


Figure 7.17: Velocity calculated with the velocity kinematics, calculated by integrating the IMU acceleration data and the velocity of the robot estimated using the Kalman filter in the sagittal plane.

When the robot is actively balancing, we see that the velocity of the IMU calculated using the velocity kinematics shows random spikes in the data. This heavily influenced the resulting velocity measurement of the robot. We manually increased the measurement noise covariance estimate $\mathbf{R}$ until the resulting velocity was satisfactory to be used for control purposes. See figure 7.18 for the adjusted Kalman filter results. The resulting noise estimate is $\mathbf{R} = 0.5 \cdot 10^{-3}$.

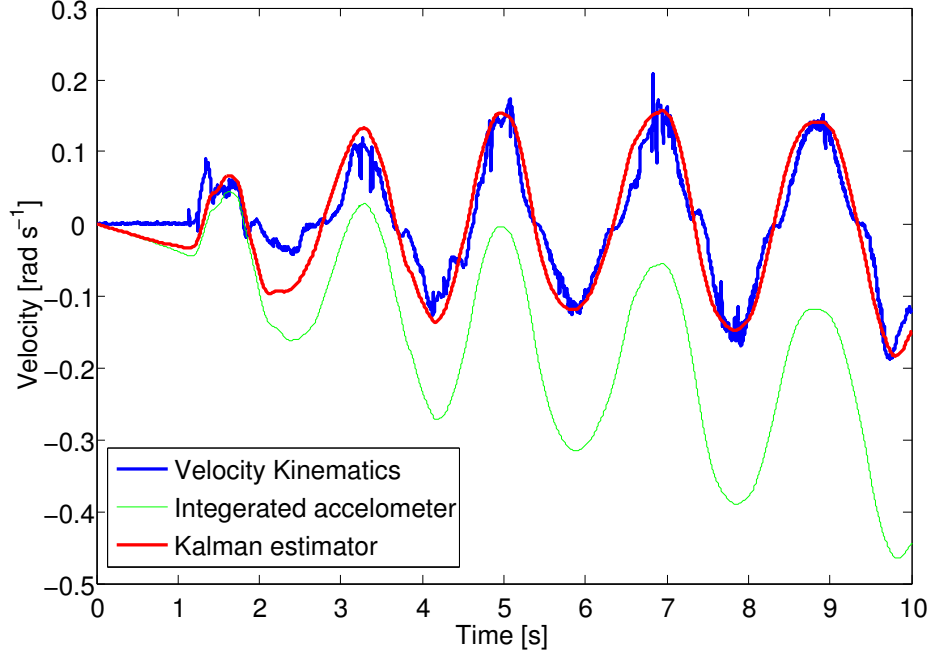


Figure 7.18: Velocity calculated with the velocity kinematics, calculated by integrating the IMU acceleration data and the velocity of the robot estimated using the Kalman filter with a manually tuned noise estimate in the sagittal plane.

If the pelvis has no angular velocity, the pelvis velocity is equal to the CoM velocity. However, because the CoM is not equal to the location of the IMU, the pelvis velocity differs from the CoM velocity. By using the coupling factors described in section 3.3.3, we can compensate for the angular velocity of the pelvis. In figure 7.19 we can see that the coupling factor reduces the CoM velocity if the rotational velocity of the body is large, but for a small rotational velocity the measured CoM velocity is hardly affected.

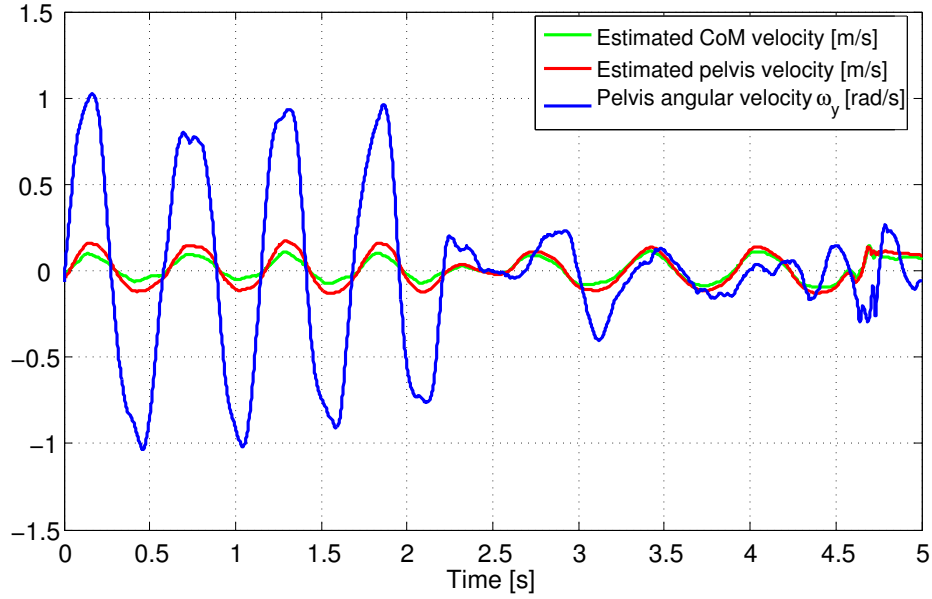


Figure 7.19: Measurement of estimated CoM velocity, estimated pelvis velocity and angular velocity of the pelvis while the robot was in double support and the operator pitched the upper body by hand. The CoM velocity is approximately 40% lower than the pelvis velocity due to the offset of the CoM from the IMU location.

7.4 Conclusion

The most suitable cost function for swing leg control is the minimum torque cost function. Compared to the minimum torque-change cost function, the minimum torque change cost function has an approximately 20% higher mechanical energy consumption for a forward step. However, the error at the end of the step is significantly larger for the minimum torque-change cost function and the ground clearance is the smallest of all controllers.

The goal of this thesis is to find a controller for the swing leg that accurately positions the foot while taking advantage of the dynamics of the swing leg to reduce control effort. By exploiting the dynamics of the swing leg, negative work can be avoided and the mechanical energy consumption is reduced. For the model predictive controller, braking torques at the end of the swing are much lower than the foot trajectory controller, reducing losses due to negative work. Furthermore, the mechanical energy consumption is reduced considerably compared to the foot trajectory controller.

For push recovery mechanical energy consumption is comparable to the model predictive controller with the minimum torque change cost function and the foot trajectory controller. The end of the step error is slightly increased. However, the peak torque is almost halved for the model predictive controller compared to the foot trajectory controller. Bandwidth requirements are also significantly reduced for the model predictive controller.

The minimum jerk controller is comparable to the foot trajectory controller in terms of energy requirements and peak torque. The bandwidth requirements

are reduced compared to the foot trajectory controller. Ground clearance depends on the chosen knot points, but is not constant during the step. A change in step location during the swing cannot be compensated for. The recommended usage for the minimum-jerk controller is in computationally constrained systems that have limited bandwidth actuators. In cases where the bandwidth of the actuators is high or there is a need to change the step location during the swing the foot trajectory controller will outperform the minimum jerk controller.

For the model predictive controller, computational constraints limit the number of constant acceleration sections to 5. Calculation time on the high level control computer is below 9 ms allowing a theoretical loop rate of 111 Hz. However, the JRTS real time virtual machine introduces a large performance penalty in comparison to the standard JVM and real-time control is not feasible at this moment. We expect that advances in processor speeds will make this controller computationally feasible in 1 to 2 years.

Finally, using a Kalman filter for pelvis velocity estimation allows us to get a good estimate for the capture point of the robot. Advantages of using a Kalman filter over just the velocity kinematics are increased noise rejection and robustness against small foot rotations and slip.

Chapter 8

Conclusion

We have proposed two controllers to replace the swing controller currently in use on M2V2, a model predictive controller and a minimum jerk controller. Two cost functions are implemented for the model predictive controller, a minimum torque cost function and a minimum torque change cost function. The minimum torque change cost function results in the lowest mechanical energy cost during the swing, but the minimum torque cost function is preferred over the minimum torque change cost function because it results in a larger ground clearance during the swing motion and a smaller error at the end of the step. The minimum torque cost function has an approximate 20% increase in mechanical energy cost compared to the minimum torque change cost function. Compared to the existing foot trajectory controller, the model predictive controller with the minimum torque cost function results in approximately 20% lower peak torques, a decrease in mechanical energy usage of approximately 35% and a significant reduction of the error at the end of the step for a forward swing motion.

The goal of this thesis is to design a controller that takes advantage of the dynamics of the swing to reduce actuator effort while providing accurate foot positioning capability for the swing foot. The proposed model predictive controller with a minimum torque change cost function is able to take a 40 cm step in 0.4 s with a positioning error between 1 cm and 2 cm. The resulting joint trajectories take advantage of the dynamics of the swing leg to come to a stop at the end of the swing with a much smaller active braking torque (negative work) compared to the foot trajectory controller. This results in the significant decrease in mechanical energy consumption.

The CoM velocity to accurately calculate the capture point is estimated by fusing the pelvis velocity kinematics and the accelerations measured by the IMU using a Kalman filter. The usage of the Kalman filter to estimate the CoM velocity greatly improves the noise characteristics over calculating the CoM velocity using the individual joint velocities. When the robot is disturbed, the proposed swing leg controller is used to quickly step on the capture point such that the robot can regain its balance.

The current setup available for M2V2 is not sufficiently powerful to run the model predictive controller, mostly because the Oracle/Sun JRTS has a large performance penalty compared to the normal Java VM. We expect that faster computers will be available in the 1-2 year time frame that are sufficiently fast to run the proposed controller.

The proposed minimum-jerk controller should only be used when the actuators have low force bandwidth and the robot is computationally constrained as the peak joint torques are higher than the model predictive controller and this controller cannot handle changing step locations.

Further work includes decreasing the calculation time to control a physical robot using our current setup, stepping over and on obstacles, implementing different cost functions based on the required task and using the model predictive controller to control arms in a constricted environment. There are several ways to decrease the calculation time. Parallel calculation of the joint torques for the constant accelerations sections using multiple threads should provide the most significant decrease in calculation time. Research topics for parallel calculation are smart algorithms for dividing the work load and implementing correct synchronization between the different threads. Other solution include profiling and hand optimization slow code paths, rewriting the controller to run without garbage collection and porting the controller to a more computationally efficient programming environment. Stepping over and on obstacles can be implemented by using a ground profile instead of assuming a flat ground for the ground clearance penalties. Research should focus on providing smooth ground profiles so the optimization algorithm converges quickly. Different cost functions can be used depending on the task. Possible cost functions are a minimization of swing time to regain balance after extremely large disturbances, maximization of ground clearance for navigating rough, unmapped terrain, and minimization of electrical energy consumption to increase the operating time of the robot. Finally, the proposed controller can be used to control the arms of the robot. A major research topic would be avoidance of obstacles, not only with the hand but also for the individual limbs.

Appendix A

Frame definition for TULip

This appendix lists the numerical values for the translations between the frames of TULip. The kinematic chain of the leg of TULip consists of 6 joints, see figure 2.1. The frames are defined as in figure 2.2. The data is taken from the SolidWorks model of TULip and should be accurate up till 0.1 mm for the physical robot.

The translation between the pelvis frame (0) and the hip yaw frame p_0^1 is

$$p_0^1 = \begin{bmatrix} 0.085 \\ 0.0759 \\ -0.392 \end{bmatrix} \quad (\text{A.1})$$

The translation between the hip yaw frame (1) and the hip roll frame (2) p_1^2 is

$$p_1^2 = \begin{bmatrix} 0.0 \\ 0.0 \\ -0.0705 \end{bmatrix} \quad (\text{A.2})$$

The translation between the hip roll frame (2) and the hip pitch frame (3) p_2^3 is

$$p_2^3 = \begin{bmatrix} 0.0 \\ 0.0378 \\ -0.04 \end{bmatrix} \quad (\text{A.3})$$

The translation between the hip pitch frame (3) and the knee frame (4) p_3^4 is

$$p_3^4 = \begin{bmatrix} 0.0 \\ 0.0 \\ -0.275 \end{bmatrix} \quad (\text{A.4})$$

The translation between the knee frame (4) and the ankle pitch frame (5) p_4^5 is

$$p_4^5 = \begin{bmatrix} 0.0 \\ 0.0 \\ -0.320 \end{bmatrix} \quad (\text{A.5})$$

Finally, the translation between the intersecting axis of the ankle pitch (5) and ankle roll frame (6) p_5^6 is zero

$$p_5^6 = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \end{bmatrix} \quad (\text{A.6})$$

Appendix B

XSens

The Inertial Measurement Unit (IMU) used on TULip is the XSens MTi. In this appendix we will describe how we configured the XSens to run at a maximum data rate and how the internal filters are configured.

The XSens MTi is connected to the onboard PC using a serial connection. This limits the data rate to $115200\text{bits} \cdot \text{s}^{-1}$. To minimize the amount of data sent over, we request a quaternion to describe the orientation (compared to a rotation matrix, a quaternion gives a non-ambiguous orientation with 4 floating point values in comparison to 9 for a rotation matrix). The complete data packet send by the XSens IMU consists of

- Preamble (1 byte)
- Address (1 byte)
- Message identifier (1 byte)
- Message length (1 byte)
- Data
 - Orientation (quaternion, 4 floats)
 - Acceleration (x-y-z, 3 floats)
 - Rotational velocity (x-y-z, 3 floats)
- Checksum (1 byte)

A float is 4 bytes long. This gives us a packet size of 45 bytes. The IMU sends data encoded with 10 bits per byte, thus 450 bits per packet. The maximum update rate Δt_{max} is

$$\Delta t_{max} = \frac{115200}{450} = 256\text{Hz} \quad (\text{B.1})$$

Tests showed that increasing the update rate above 200Hz resulted in random errors in the data stream or a complete loss of XSens data.

According to the XSens manual, there are five scenarios for the XSens, which control the different filter settings used internally in the IMU

- Human

- Human (Large Accelerations)
- Machine
- Machine (no magnetic field)
- Marine

For our humanoid robot, we used the Human scenario.

The gyroscopes and accelerometers inside the XSens need about 10 minutes to warm up and settle in steady state operation. Besides that, the internal filter needs about a minute to converge to the current orientation. We wrote a small daemon that continuously reads the XSens data and makes it available to the control software, such that if the controller starts the XSens is immediately usable.

If the XSens does not coincide with the CoM of the robot, rotations around the Center of Mass result in an acceleration of the IMU, reducing the accuracy of the orientation measurement. Therefore, we attached the XSens as close as possible to the (average) location of the CoM.

Appendix C

Quintic (5th order) spline

C.1 Introduction

Splines were first introduced by de Boor in 1987 [33]. A spline is a combination of polynomials that allow interpolation between many points. The advantage over a polynomial interpolation of a number of points is that the order of the spline does not increase when the number of points is increased, as opposed to a polynomial interpolation.

A fifth order (quintic) spline allows us to define an arbitrary number of points to interpolate. The points to interpolate are also called knot points. The location of knot point i is described by the variables x_i and y_i .

A quintic spline has six free variables, this allows us to choose an arbitrary initial value, derivative and second derivative as well as an arbitrary final value, derivative and second derivative.

A quintic spline consists of n 5th order polynomials, s_0 to s_n . The first five derivatives of the individual polynomials s_i are finite, see equation C.1.

$$\begin{aligned} s_i(x) &= a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 + e_i(x - x_i)^4 + f_i(x - x_i)^5 \\ \dot{s}_i(x) &= b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2 + 4e_i(x - x_i)^3 + 5f_i(x - x_i)^4 \\ \ddot{s}_i(x) &= 2c_i + 6d_i(x - x_i) + 12e_i(x - x_i)^2 + 20f_i(x - x_i)^3 \\ \dddot{s}_i(x) &= 6d_i + 24e_i(x - x_i) + 60f_i(x - x_i)^2 \\ \dots_i(x) &= 24e_i + 120f_i(x - x_i) \end{aligned} \tag{C.1}$$

C.2 Constraints

At the knot points, a quintic spline is guaranteed to be continuous to the 4th derivative. This allows us to define six constraints for each polynomial s_i , see equation C.2.

$$\begin{aligned}
s_i(x_i) &= y_i \\
s_i(x_i) &= s_{i-1}(x_i) \\
\dot{s}_i(x_i) &= \dot{s}_{i-1}(x_i) \\
\ddot{s}_i(x_i) &= \ddot{s}_{i-1}(x_i) \\
\dddot{s}_i(x_i) &= \dddot{s}_{i-1}(x_i)
\end{aligned} \tag{C.2}$$

C.3 Derivation

The values of a_i , b_i , c_i , d_i , e_i and f_i can be determined from the constraints given in equations C.2. First, we define the difference between the location of the two knot points, h_i

$$h_i = x_{i+1} - x_i \tag{C.3}$$

Substituting equation C.3 into C.1

$$s_i(x_i) = a_i = y_i \tag{C.4}$$

$$s_{i+1}(x_i) = a_{i+1} = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 + e_i h_i^4 + f_i h_i^5 \tag{C.5}$$

$$\dot{s}_{i+1}(x_i) = b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2 + 4e_i h_i^3 + 5f_i h_i^4 \tag{C.6}$$

$$\ddot{s}_{i+1}(x_i) = 2c_{i+1} = 2c_i + 6d_i h_i + 12e_i h_i^2 + 20f_i h_i^3 \tag{C.7}$$

$$\dddot{s}_{i+1}(x_i) = 6d_{i+1} = 6d_i + 24e_i h_i + 60f_i h_i^2 \tag{C.8}$$

$$\ddot{\ddot{s}}_{i+1}(x_i) = 24e_{i+1} = 24e_i + 120f_i h_i \tag{C.9}$$

First, we write a_{i+1} , b_{i+1} , c_{i+1} , d_{i+1} and f_i in terms of e_i , e_{i+1} and h_i .

$$120f_i h_i = 24e_{i+1} - 24e_i \tag{C.10}$$

$$f_i = \frac{e_{i+1} - e_i}{5h_i} \tag{C.11}$$

$$6d_{i+1} = 6d_i + 24e_i h_i + 60 \frac{e_{i+1} - e_i}{5h_i} h_i^2 \tag{C.12}$$

$$6d_{i+1} = 6d_i + 12e_i h_i + 12e_{i+1} h_i \tag{C.13}$$

$$2c_{i+1} = 2c_i + 6d_i h_i + 12e_i h_i^2 + 20 \frac{e_{i+1} - e_i}{5h_i} h_i^3 \tag{C.14}$$

$$2c_{i+1} = 2c_i + 6d_i h_i + 8e_i h_i^2 + 4e_{i+1} h_i^2 \tag{C.15}$$

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2 + 4e_i h_i^3 + 5 \frac{e_{i+1} - e_i}{5h_i} h_i^4 \tag{C.16}$$

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2 + 3e_i h_i^3 + e_{i+1} h_i^3 \tag{C.17}$$

$$a_{i+1} = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 + e_i h_i^4 + \frac{e_{i+1} - e_i}{5h_i} h_i^5 \quad (\text{C.18})$$

$$a_{i+1} = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 + \frac{4}{5} e_i h_i^4 + \frac{e_{i+1}}{5} h_i^4 \quad (\text{C.19})$$

From equation C.6 we can determine b_i

$$b_i h_i = a_{i+1} - a_i - c_i h_i^2 - d_i h_i^3 - \frac{4}{5} e_i h_i^4 - \frac{e_{i+1}}{5} h_i^4 \quad (\text{C.20})$$

$$b_i = \frac{a_{i+1}}{h_i} - \frac{a_i}{h_i} - c_i h_i - d_i h_i^2 - \frac{4}{5} e_i h_i^3 - \frac{e_{i+1}}{5} h_i^3 \quad (\text{C.21})$$

Substituting equation C.21 in equation C.17

$$b_{i+1} = \frac{a_{i+2}}{h_{i+1}} - \frac{a_{i+1}}{h_{i+1}} - c_{i+1} h_{i+1} - d_{i+1} h_{i+1}^2 - \frac{4}{5} e_{i+1} h_{i+1}^3 - \frac{e_{i+2}}{5} h_{i+1}^3 \quad (\text{C.22})$$

Then substituting equation C.22 and equation C.17 in equation C.6

$$\begin{aligned} & \frac{a_{i+2}}{h_{i+1}} - \frac{a_{i+1}}{h_{i+1}} - c_{i+1} h_{i+1} - d_{i+1} h_{i+1}^2 - \frac{4}{5} e_{i+1} h_{i+1}^3 - \frac{e_{i+2}}{5} h_{i+1}^3 = \\ & \frac{a_{i+1}}{h_i} - \frac{a_i}{h_i} - c_i h_i - d_i h_i^2 - \frac{4}{5} e_i h_i^3 - \frac{e_{i+1}}{5} h_i^3 + 2c_i h_i + 3d_i h_i^2 + 3e_i h_i^3 + e_{i+1} h_i^3 \end{aligned} \quad (\text{C.23})$$

Finally combining all the terms gives us the linear equality constraint that needs to be satisfied for all polynomials

$$\begin{aligned} & \frac{a_{i+2}}{h_{i+1}} - \frac{a_{i+1}}{h_{i+1}} - c_{i+1} h_{i+1} - d_{i+1} h_{i+1}^2 - \frac{4}{5} e_{i+1} h_{i+1}^3 - \frac{e_{i+2}}{5} h_{i+1}^3 = \\ & \frac{a_{i+1}}{h_i} - \frac{a_i}{h_i} + c_i h_i + 2d_i h_i^2 + 2\frac{1}{5} e_i h_i^3 + \frac{4}{5} e_{i+1} h_i^3 \end{aligned} \quad (\text{C.24})$$

C.4 Writing the linear system of equations in matrix form

First, move all a_i terms in equation C.24 to the left side

$$\begin{aligned} & \frac{a_{i+2}}{h_{i+1}} - \frac{a_{i+1}}{h_{i+1}} - \frac{a_{i+1}}{h_i} + \frac{a_i}{h_i} = \\ & c_i h_i + c_{i+1} h_{i+1} + 2d_i h_i^2 + d_{i+1} h_{i+1}^2 + 2\frac{1}{5} e_i h_i^3 + \frac{4}{5} e_{i+1} h_i^3 + \frac{4}{5} e_{i+1} h_{i+1}^3 + \frac{e_{i+2}}{5} h_{i+1}^3 \end{aligned} \quad (\text{C.25})$$

Next, write d_0 to d_{n-1} as function of c_0 , d_0 and e_0 to e_n

$$6d_{i+1} = 6d_i + 12e_i h_i + 12e_{i+1} h_i \quad (\text{C.26})$$

$$d_{i+1} = d_i + 2e_i h_i + 2e_{i+1} h_i \quad (\text{C.27})$$

$$(\text{C.28})$$

Writing equation C.27 in matrix form for $i = 0 \rightarrow n - 1$ results in

$$\begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 2h_0 & 2h_0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 1 & 2h_0 & 2h_0 + 2h_1 & \dots & 2h_{n-3} & 0 & 0 \\ 0 & 1 & 2h_0 & 2h_0 + 2h_1 & \dots & 2h_{n-3} + h_{n-2} & 2h_{n-2} & 0 \end{bmatrix} \begin{bmatrix} c_0 \\ d_0 \\ e_0 \\ e_1 \\ \vdots \\ e_{n-2} \\ e_{n-1} \\ e_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{bmatrix} \quad (\text{C.29})$$

The next step is to write c_0 to c_n as function of c_0 , d_0 and e_0 to e_n

$$c_{i+1} = c_i + 3d_i h_i + 4e_i h_i^2 + 2e_{i+1} h_i^2 \quad (\text{C.30})$$

Writing equation C.30 in matrix form for $i = 0 \dots n$ results in

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 4h_0^2 & 2h_0^2 & 0 & \dots & 0 & 0 \\ 1 & 0 & 4h_0^2 & 2h_0^2 + 4h_1^2 & 2h_1^2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 4h_0^2 & 2h_0^2 + 4h_1^2 & 2h_1^2 + 4h_2^2 & \dots & 2h_{n-2}^2 & 0 \\ 1 & 0 & 4h_0^2 & 2h_0^2 + 4h_1^2 & 2h_1^2 + 4h_2^2 & \dots & 2h_{n-2}^2 + 4h_{n-1}^2 & 2h_{n-1}^2 \end{bmatrix} \begin{bmatrix} c_0 \\ d_0 \\ e_0 \\ e_1 \\ e_2 \\ \vdots \\ e_3 \\ e_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 3h_i d_0 \\ 3h_i(d_0 + d_1) \\ \vdots \\ 3h_i(d_0 + d_1 + \dots + d_{n-2}) \\ 3h_i(d_0 + d_1 + \dots + d_{n-2} + d_{n-1}) \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_3 \\ c_4 \end{bmatrix} \quad (\text{C.31})$$

For clarity we did not expand d_0 to d_{n-1} , which are given in equation C.29.

The final step is to put equation C.25 in matrix forms for $i = 0 \dots n$. Again, for clarity we do not expand c_0 to c_n and d_0 to d_{n-1} given in equation C.29 and C.31.

$$\begin{aligned}
& \begin{bmatrix} 0 & 0 & 2\frac{1}{5}h_0^3 & \frac{4}{5}h_0^3 + \frac{4}{5}h_1^3 & \frac{1}{5}h_1^3 & 0 & 0 \\ 0 & 0 & 0 & 2\frac{1}{5}h_1^3 & \frac{4}{5}h_1^3 + \frac{4}{5}h_2^3 & \frac{1}{5}h_2^3 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\ 0 & 0 & 0 & 0 & 2\frac{1}{5}h_2^3 & \frac{4}{5}h_2^3 + \frac{4}{5}h_3^3 & \frac{1}{5}h_3^3 \end{bmatrix} \begin{bmatrix} c_0 \\ d_0 \\ e_0 \\ e_1 \\ e_2 \\ \vdots \\ e_3 \\ e_4 \end{bmatrix} \\
& + \begin{bmatrix} c_0h_0 + c_1h_1 + 2d_0h_0^2 + d_1h_1^2 \\ c_1h_1 + c_2h_2 + 2d_1h_1^2 + d_2h_2^2 \\ \vdots \\ c_2h_2 + c_3h_3 + 2d_2h_2^2 + d_3h_3^2 \end{bmatrix} = \begin{bmatrix} \frac{y_2}{h_1} - \frac{y_1}{h_1} - \frac{y_1}{h_0} + \frac{y_0}{h_0} \\ \frac{y_3}{h_2} - \frac{y_2}{h_2} - \frac{y_2}{h_1} + \frac{y_1}{h_1} \\ \vdots \\ \frac{y_4}{h_3} - \frac{y_3}{h_3} - \frac{y_3}{h_2} + \frac{y_2}{h_2} \end{bmatrix} \quad (C.32)
\end{aligned}$$

C.5 Boundary conditions

To completely constrain the quintic spline, we need to define four more constraints besides the knot points and the constraints given in equation C.2. These boundary conditions are the initial and final velocities and accelerations of the quintic spline.

$$\dot{s}_0 = \dot{y}_0 \quad (C.33)$$

$$\ddot{s}_0 = \ddot{y}_0 \quad (C.34)$$

$$\dot{s}_n = \dot{y}_n \quad (C.35)$$

$$\ddot{s}_n = \ddot{y}_n \quad (C.36)$$

The initial velocity $\dot{y}_0$ can be written in terms of a_0 , a_1 , c_0 , d_0 and e_0

$$\dot{y}_0 = b_0 = \frac{a_1}{h_0} - \frac{a_0}{h_1} - c_0h_0 - d_0h_0^2 - \frac{4}{5}e_0h_0^3 - \frac{e_1}{5}h_0^3 \quad (C.37)$$

$$\frac{a_1}{h_0} - \frac{a_0}{h_1} - \dot{y}_0 = c_0h_0 + d_0h_0^2 + \frac{4}{5}e_0h_0^3 + \frac{e_1}{5}h_0^3 \quad (C.38)$$

And the initial acceleration $\ddot{y}_0$

$$\ddot{y}_0 = 2c_0 \quad (C.39)$$

The final desired velocity $\dot{y}$ can be written in terms of a_n , e_n , a_{n-1} , c_{n-1} , d_{n-1} and e_{n-1}

$$\dot{s}_n = b_n = b_{n-1} + 2c_{n-1}h_{n-1} + 3d_{n-1}h_{n-1}^2 + 3e_{n-1}h_{n-1}^3 + e_nh_{n-1}^3 \quad (\text{C.40})$$

$$b_{n-1} = \frac{a_n}{h_{n-1}} - \frac{a_{n-1}}{h_{n-1}} - c_{n-1}h_{n-1} - d_{n-1}h_{n-1}^2 - \frac{4}{5}e_{n-1}h_{n-1}^3 - \frac{e_n}{5}h_{n-1}^3 \quad (\text{C.41})$$

$$\dot{y}_n = \frac{a_n}{h_{n-1}} - \frac{a_{n-1}}{h_{n-1}} + c_{n-1}h_{n-1} + 2d_{n-1}h_{n-1}^2 + 2\frac{1}{5}e_{n-1}h_{n-1}^3 + \frac{4}{5}e_nh_{n-1}^3 \quad (\text{C.42})$$

$$\dot{y}_n - \frac{a_n}{h_{n-1}} + \frac{a_{n-1}}{h_{n-1}} = c_{n-1}h_{n-1} + 2d_{n-1}h_{n-1}^2 + 2\frac{1}{5}e_{n-1}h_{n-1}^3 + \frac{4}{5}e_nh_{n-1}^3 \quad (\text{C.43})$$

And the final desired acceleration $\ddot{y}_n$

$$\ddot{y}_n = 2c_4 \quad (\text{C.44})$$

Combining equations C.38, C.39, C.43 and C.44 into matrix form gives us the linear set of equations to solve for the boundary conditions.

$$\begin{bmatrix} h_0 & h_0^2 & \frac{4}{5}h_0^3 & \frac{1}{5}h_0^3 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\frac{1}{5}h_3^3 & \frac{4}{5}h_3^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_0 \\ d_0 \\ e_0 \\ e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ c_3h_3 + 2d_3h_3^2 \\ 2c_4 \end{bmatrix} = \begin{bmatrix} \frac{a_1}{h_0} - \frac{a_0}{h_1} - \dot{y}_0 \\ \ddot{y}_0 \\ \dot{y}_3 - \frac{a_4}{h_3} + \frac{a_3}{h_3} \\ \ddot{y}_4 \end{bmatrix} \quad (\text{C.45})$$

C.6 Solving for the coefficients

The spline coefficients can be found by solving the linear set of equations given by equation C.32 and equation C.45. Using freely available matrix libraries we can solve the linear set of equations quickly and efficiently. This allows us to use the quintic spline to create trajectories in real time.

Bibliography

- [1] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. E. Pratt, “Capturability-Based Analysis and Control of Legged Locomotion, Part 1: Theory and Application to Three Simple Gait Models,” *International Journal of Robotics Research*, vol. Submitted, 2012.
- [2] J. E. Pratt, J. Carff, S. V. Drakunov, and A. Goswami, “Capture Point : A Step toward Humanoid Push Recovery,” in *International Conference on Humanoid Robots*, pp. 200–207, 2006.
- [3] T. McGeer, “Passive walking with knees,” *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pp. 1640–1645, 1990.
- [4] S. H. Collins, A. Ruina, R. Tedrake, and M. Wisse, “Efficient bipedal robots based on passive-dynamic walkers,” *Science (New York, N.Y.)*, vol. 307, pp. 1082–5, Feb. 2005.
- [5] S. R. Buss, “Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods,” tech. rep., 2004.
- [6] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, “the development of honda humanoid robot,” in *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, (Leuven, Belgium), pp. 1321–1326, 1998.
- [7] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, and K. H. K. Yokoi, “Biped walking pattern generation by using preview control of zero-moment point,” in *in Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1620–1626, 2003.
- [8] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafortade, B. Marnier, J. Serre, and B. Maisonnier, “Mechatronic design of NAO humanoid,” in *2009 IEEE International Conference on Robotics and Automation*, pp. 769–774, IEEE, May 2009.
- [9] J.-Y. Kim, I.-W. Park, J. Lee, M.-S. Kim, B.-k. Cho, and J.-H. Oh, “System Design and Dynamic Walking of Humanoid Robot KHR-2,” in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 1431–1436, IEEE, 2005.
- [10] D. G. E. Hobbelen, *Limit cycle walking*. PhD thesis, Delft University of Technology, 2008.

- [11] J. J. Craig, *Introduction to robotics: mechanics and control*. Prentice Hall, 2005.
- [12] J. E. Pratt and B. T. Krupp, “Series Elastic Actuators for legged robots,”
- [13] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer, 2007.
- [14] V. Duindam, *Port-Based Modeling and Control for Efficient Bipedal Walking Robots*. PhD thesis, University of Twente, Enschede, 2006.
- [15] D. L. Pieper, “The kinematics of manipulators under computer control, Computer Science Department,” 1968.
- [16] C. W. Wampler, “Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, pp. 93–101, Jan. 1986.
- [17] V. Duindam, *Port-Based Modeling and Control for Efficient Bipedal Walking Robots*.
- [18] 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 *Proceedings of the 2001 IEEE International Conference on Intelligent Robots and Systems*, vol. 1, (Maui, Hawaii), pp. 239–246, IEEE, 2001.
- [19] J. Engelsberger, C. Ott, M. A. Roa, A. Albu-Schaffer, and G. Hirzinger, “Bipedal walking control based on Capture Point dynamics,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4420–4427, IEEE, Sept. 2011.
- [20] J. E. Pratt, T. Koolen, T. De Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. D. Neuhaus, “Capturability-Based Analysis and Control of Legged Locomotion , Part 2: Application to M2V2 , a Lower Body Humanoid,” *International Journal of Robotics Research*, vol. Accepted, 2012.
- [21] FPL Statistics Group, “Nonlinear Optimization Java Package,” 2010.
- [22] B. S. Garbow, K. E. Hillstom, and J. J. More, “LMDIF1,” 1980.
- [23] S. Verrill, “Some tricks for converting linearly constrained nonlinear optimization problems into unconstrained problems (Unpublished),” tech. rep., Forest Products Laboratory, Madison, Wisconsin, 2002.
- [24] M. Spong and S. Hutchinson, “Robot modeling and control,” *Control*, 2006.
- [25] IHMC, “Simulation Construction Set,” 2012.
- [26] J. Gosling, G. Bollella, P. Dibble, S. Furr, and M. Turnbull, *The Real-Time Specification for Java*. Addison Wesley Longman, 1st editio ed., 2000.
- [27] E. J. Bruno and G. Bollella, *Real-Time Java Programming: With Java RTS*. Prentice Hall, 1st editio ed., 2009.
- [28] M. Paleczny, C. Vick, and C. Click, “The java hotspot TM server compiler,” p. 1, Apr. 2001.

- [29] J. Auerbach, D. F. Bacon, B. Blainey, P. Cheng, M. Dawson, M. Fulton, D. Grove, D. Hart, and M. Stoodley, “Design and implementation of a comprehensive real-time java virtual machine,” in *Proceedings of the 7th ACM & IEEE international conference on Embedded software - EMSOFT '07*, (New York, New York, USA), p. 249, ACM Press, Sept. 2007.
- [30] P. R. Wilson and P. R. Wilson, “Uniprocessor Garbage Collection Techniques,” 1992.
- [31] D. F. Bacon, P. Cheng, and V. T. Rajan, “The Metronome: A Simpler Approach to Garbage Collection in Real-Time Systems,” pp. 466–478, 2003.
- [32] T. De Boer, *Foot placement in robotic bipedal locomotion*. Phd thesis, Delft University of Technology, 2012.
- [33] C. De Boor, *A practical guide to splines*, vol. 27. Springer Verlag, 1978.