Optimal Throwing Motion of Three-Link Underactuated Robot

A Study of Virtual Holonomic Constraints

Brede Løvik Lillehammer

Master of Science in Engineering Cybernetics (2 year))
Submission date: June 2014
Supervisor: Anton Shiriaev, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

This thesis will continue from *TTK4551 Specialization Project*. A reduced dynamic model was created for a three link robot consisting of actuated shoulder and elbow joint, and passive wrist joint. This corresponds to joint two, three and five on KUKA lightweight robot 4+ the model based upon, resulting in planar motion only.

The thesis will focus on analysing the achieved model, plan a throwing motion of the robot resulting in optimal pitch of an object, run simulations of the system and implement on the KUKA robot for experiments.

Supervisor: Professor Anton Shiriaev
Co-supervisor: Sergey Kolyubin

Preface

This report presents the work for *TTK 4900 Master Thesis* from my fourth and final semester at the department of Engineering Cybernetics at the Norwegian University of Science and Technology spring 2014. The work here is a continuation of the *Specialization Project TTK4551* carried out during the fall semester 2013, and parts of that report, mostly background theory, is reused here. The thesis focus lays in creating a dynamical model to simulate an optimal motion for longest possible pitch with a robot, which allows me to explore different kinds of interesting subjects I have learned through these two years.

I would like to thank my supervisors professor Anoton Shiriaev and Sergey Kolyubin for assistance and guidance throughout our many meetings and testing on robot.

Brede Løvik Lillehammer
Trondheim, June 3, 2014.

Abstract

Control design for weakly or underactuated systems, i.e. more degrees of freedom than number of control inputs available, has proven to be much more complex than fully actuated due to the new constraints it impose. This thesis will use a three degree of freedom model with passive spring articulated wrist joint for maximizing a ball pitch. A suitable dynamic model is found by the forward kinematic approach, using joint two, three and five of KUKA lightweight robot 4+ as foundation, resulting in planar motions. Then, by using virtual holonomic constraints, the dynamics is reduced in such a way that the wrist is controlled through shoulder and elbow joint. With these new dynamics, the optimization toolbox in MATLAB is used, searching for a motion that will optimize the pitch distance. This report contains both numerical research of optimal trajectory and results of these trajectories implemented on the robot for experiments. A variety of different motions are found, however, the Matlab-function *fmincon* finds local minimum, which means the results perhaps are only close to the optimal motion for longest pitch.

Sammendrag

Kontrol design for svakt eller underaktuerte systemer, dvs. flere frihetsgrader enn antall kontroll-innganger tilgjengelig, har vist seg å være mye mer komplekse enn systemer med bare aktuerte frihetsgrader, grunnet de nye restriksjonene som påføres systemet. Denne oppgaven vil bruke en model av tre frihetsgrader med passiv springladet håndledd for å maksimere et ballkast. En passende model er laget ved hjelp av "forward kinematic"-metoden hvor ledd to, tre og fem fra KUKA lightweight robot 4+ er tatt som grunnlag. Dette resulterer i todimensjonale bevegelser. Så, ved å bruke "virtual holonomic constraints", vil det dynamiske systemet bli redusert på en slik måte at håndleddet blir styrt ved hjelp av skulder- og albuleddet. Med den nye dynamikken vil optimaliserings verktøykassen til MATLAB bli brukt for å lete etter bevegelser for system som resulterer i lengst mulig kast. Denne rapporten inneholder både numerisk forskning av optimal bevegelse og resultater av disse bevegelsene implementert i roboten for eksperimentering. En god del forskjellige bevegelser er funnet, men Matlab-funksjonen "fmincon" finner bare locale minimumspunkt. Dermed er resultatene sannsynligvis bare nære den optimale kastebevegelsen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Up to this date, many studies have been done for a various types of pitches. Both for hitting a specific target and for distance throwing, with and without underactuated dynamics. For target throwing, (Senoo et al.)[1] have done numerical and experimental studies on a fully actuated 3 degree of freedom (DOF) robot arm where they present a strategy for high-speed swing motion. In contrast to fully actuated systems, systems of one or more underactuated DOF will be much harder to solve. However, both (Yedeg et al.)[2] and (Mettin and Shiriaev)[3] have done numerical studies of two-link system with objective of finding an optimal motion of shoulder joint that result in the longest possible pitch with passive spring articulated elbow joint. The task is solved using an interior point method with gradient supplied by a discrete adjoint method, and by reducing the system dynamics with virtual holonomic constraints respectively for the two reports.

This thesis, as (Mettin and Shiriaev)[3], will also make use of virtual holonomic constraints to reduce the system dynamics. However, the model will be based on the KUKA lightweight robot 4+, which is installed at the NTNU robot lab. Furthermore, the model will use joint two, three and five of a total of seven possible joint available, where joint five will be passive and spring articulated. More of the problem description is explained in chapter 1.1.

1.1 Objective

For the specialization project, the objective was to create a model for the dynamics of the KUKA lightweight robot 4+. The last joint i.e. q_3 of the generalized coordinates showed in figure 1.1 below, is used as a passive, spring articulated wrist. Use of a passive joint makes the system underactuated, which means there are more degrees of freedom than available independent control inputs. Consequently, motion planning and control design are much more complex. As shown in the figure below, the joint angle of q_1 is measured with respect to the x-axis, while q_2 and q_3 are measured with respect to the links they're connected to.

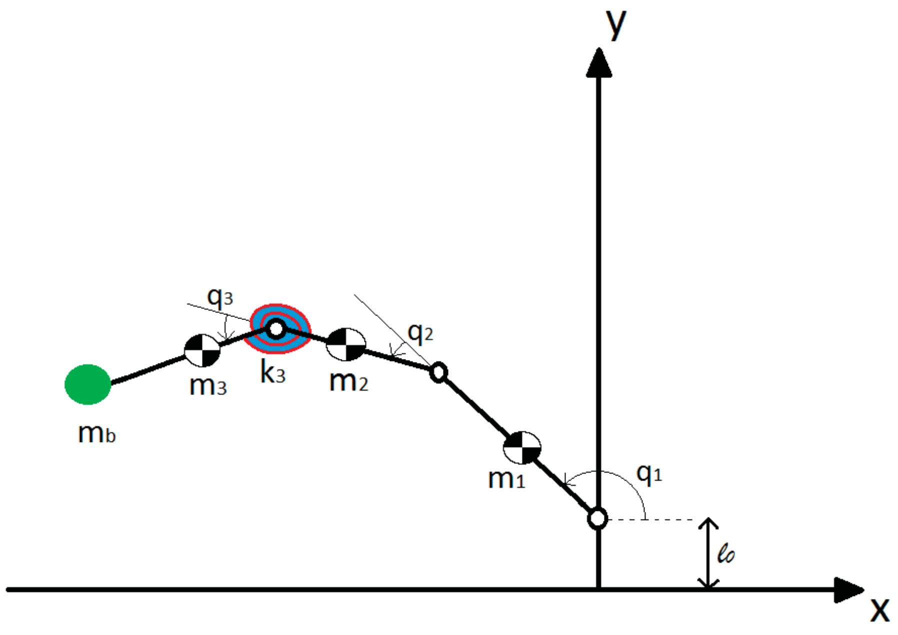


Figure 1.1: Underactuated 3-DOF robot manipulator.

As already mentioned in the introduction, only three out of seven joints available will be used. The rest are frozen with respect to the links connecting them, which means they can be seen as extensions of the links connecting the generalized coordinates together. This is also why joint one in the figure start at an elevated height l_0. For actively holding the ball during pitch phase, the robot is equipped with the end-effector *3-Fingered Adaptive Robot Gripper* from Robotiq.

The objective for this thesis however, is to utilize the model from the specialization project for finding motions for each joint that results in the longest pitch possible. Resulting motions are then to be used as target trajectory for the robot. This can be done in two ways, feed joint angles or torques directly to the robot. Consequently, control design will not be studied, as the robot uses its internal controllers to follow desired trajectories.

1.2 Software

For this thesis, three computer software's have been utilized to complete the objective. Here, a description of each program will be given.

1.2.1 Maple 17

Maple, developed by Maplesoft, is a powerful computing software for doing numerical/symbolic computations, visualization and much more. As a result of its great possibilities for symbolic computation, this software is used to derive a model for the relevant robot manipulator.

1.2.2 SolidWorks 2013

SolidWorks is a CAD-software developed by Dassault Systèmes SolidWorks Corp. used for developing 3D and electrical design, simulation and more. This software is used for finding physical parameters such as length of links, mass properties, center of mass and inertias for the robot, as KUKA labs only provide a model for the robot and not the exact parameters.

1.2.3 MATLAB R2014a

MATLAB, developed by MathWorks, is a software for both numerical and symbolic computations, modelling, simulation, data analysis and much more. For the thesis, MATLAB is used in a optimization procedure for achieving long pitch distance and for simulating the robot motion after trajectories are found.

1.3 Outline

In chapter 2, the background theory used for the project and thesis will be introduced. First, a description of how to find the dynamics of a system by using the Euler-Lagrange equations, with forward and velocity kinetamics, are explained. Then an introduction to virtual holonomic constraints and how to reduce the system dynamics with these are explained.

Chapter 3 will go through the kinematics of the ball in pitch- and ballistic phase. Some assumptions are made and explained. Next, step by step of how to create the dynamic model for the KUKA light weight robot, both with equation of motion and its reduced form by using virtual holonomic constraints. Physical parameters used in the model are discussed, and in the end, some explanations of how to create the objective function is introduced.

The Bézier polynomial will be introduced in chapter 4, followed by some explanation of the optimization task and its efficiency. Lastly, some characteristics of the system at hand will be discussed.

Results from pitching simulations and experiments are presented in chapter 5, and in chapter 6, these results are discussed and compared to other work.

In the end, a conclusion and suggestions for further work are presented in chapter 7.

Appendix A contains physical parameters, calculated and extracted from CAD-files, and constraints found in the datasheet. In appendix B, step by step how to solve the reduced dynamics with respect to $\dot{\theta}$ is given. Lastly, appendix C give an overview of what can be found in the digital appendix of pdf-files, CAD-models, Maple-scripts and MATLAB-code.

Chapter 2

Background Theory

In this chapter, theory and methodology used to find the dynamical model for the KUKA robot is introduced. First, an introduction of forward kinematics solved with the Denavit-Hartenberg convention will be given, followed by the velocity kinematics, use of equation of motion for the dynamics, and virtual holonomic constraints to find the reduced system dynamics.

2.1 Kinematics

2.1.1 Forward kinematics

The problem of kinematics is basically to find the geometric relations between joints, and position and orientation of the end-effector without considering what forces and torques that produce the motions (Spong et al.)[4]. These relations can be found by both inverse and forward kinematics, but only the latter will be discussed here.

A robot manipulator such as KUKA lightweight robot 4+ is a sequence of links and joints. All joints for this robot are revolute and can only rotate about one axis. This means the homogeneous transformation matrix can be written as

$$A_i = \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ 0 & 1 \end{bmatrix}, \tag{2.1}$$

where R_i^{i-1} and o_i^{i-1} is the 3×3 rotation matrix and 3×1 translation for link $i-1$ to i respectively, have no generalized coordinates of the prismatic type.

A normal homogeneous transformation would need six parameters. However, by using the Denavit-Hartenberg convention, the matrix is simplified, only using up to four parameters. To identify these parameters, the coordinate frames follows two features:

- "The axis x_i is perpendicular to the axis z_{i-1}." (Spong et al.)[4]
- "The axis x_i intersects the axis z_{i-1}." (Spong et al.)[4]

By using these two features, the forward kinematics is a simple task when following the Denavit-Hartenberg recipe. This gives, for each link, the parameters a_i, α_i, d_i and θ_i which are link length, link twist, link offset and joint angle respectively. The homogeneous transformation matrix can then be given by

$$A_i = \begin{bmatrix} c_{\theta i} & -s_{\theta i}c_{\alpha i} & s_{\theta i}s_{\alpha i} & a_i c_{\theta i} \\ s_{\theta i} & c_{\theta i}c_{\alpha i} & -c_{\theta i}s_{\alpha i} & a_i s_{\theta i} \\ 0 & s_{\alpha i} & c_{\alpha i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

However, it is important to notice that there are no unique way of assigning the coordinate frames. Still, the transformation matrix that contains position and orientation of end-effector with respect to the base frame,

$$T_n^0 = A_1 \cdot A_2 \cdot \ldots \cdot A_n \tag{2.3}$$

where n is the number of joints of the robot, will always be the same if done correctly.

2.1.2 Velocity kinematics

In previous chapter, the forward kinematics defined a matrix of Cartesian coordinates and joint positions. In this chapter, the next step of kinematics will be explained, namely the Jacobian, containing linear and angular velocities of joints and end-effector.

For every joint in a robot manipulator, prismatic or revolute, there exist a Jacobian matrix describing the velocities from joint space to Cartesian space (Spong et al.)[4]. The linear and angular velocity for joint i are given by $v_i = \dot{o}_i = J_{vi}\dot{q}$ and $\omega_i = J_{\omega i}\dot{q}$ respectively. Here, $\dot{q}$ is the joint variable differentiated with respect to time and J_v and J_ω are the Jacobian's.

Solving the Jacobian's is a trivial matter once the forward kinematics have been found. All that is needed is to extract two vectors from the transformation matrix and solve a crossproduct for linear velocity if the joint is revolute. The first is the vector z_i, which is the three first elements of the third column in the transformation matrix T_i^0. The second is the vector o_i which is the three first elements of fourth column in the transformation matrix and contains the origin coordinates for joint $i \in 1, ..., n$, where n is the number of joints for the robot. The Jacobian's are both a $3 \times n$ matrix and is calculated as shown in equation (2.4) and (2.5) and stacked as vectors shown in (2.6) and (2.7). (Equations from (Spong et al.)[4]).

$$J_{v_i} = \begin{cases} z_{i-1} \times (o_n - o_{i-1}), & \text{if joint i is revolute} \\ z_{i-1}, & \text{if joint i is prismatic} \end{cases} \tag{2.4}$$

$$J_{\omega_i} = \begin{cases} z_{i-1}, & \text{if joint i is revolute} \\ 0, & \text{if joint i is prismatic} \end{cases} \tag{2.5}$$

$$J_v = \begin{bmatrix} J_{v_1} & J_{v_2} & ... & J_{v_n} \end{bmatrix} \tag{2.6}$$

$$J_\omega = \begin{bmatrix} J_{\omega_1} & J_{\omega_2} & ... & J_{\omega_n} \end{bmatrix} \tag{2.7}$$

In the next chapter, about dynamics, these computations will prove to be very important.

2.2 Dynamics

Kinematics describes motion of joints and end-effector without even considering forces and torques made on the manipulator. On the other hand, the dynamics takes all this into consideration and gives the relation between forces, torques, position, velocities and accelerations of joints. Hence, it is called the equation of motion.

To calculate the equations of motion, the first step is to find the Lagrangian $\mathcal{L}$. The Lagrangian is defined as the total kinetic and potential energy difference in terms of the generalized coordinates of the robot, i.e. sum of the difference of kinetic and potential energy of every link. This gives the Lagrangian as equation (2.8).

$$\mathcal{L} = K - P \tag{2.8}$$

Kinetic energy is the sum of two terms: rotational energy of the body through center of mass, and translational energy of center of mass. This is given by equation (2.9).

$$K = \frac{1}{2}\dot{q}^T \left[\sum_{i=1}^{n} \left\{ m_i J_{v_i}(q)^T J_{v_i} + J_{\omega_i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega_i}(q) \right\} \right] \dot{q} \tag{2.9}$$

$$= \frac{1}{2}\dot{q}^T M(q)\dot{q} \tag{2.10}$$

Here, the generalized coordinates $q = [q_1, \cdots, q_n]$ is given by the joint variables found from the Denavit-Hartenberg convention as mentioned in 2.1.1. J_{v_i} and J_{ω_i} are the Jacobians from chapter 2.1.2, m_i is mass respective to each link, I_i is the inertia matrix and R_i is the orientation transformation from world frame coordinates to link i. In order to have good dynamics, the inertia matrix must be precise. This can prove to be difficult, as the shape and density of links usually are not uniform.

For rigid dynamics, the only potential energy the system have, is due to gravity. However, for the KUKA robot, all joints are spring articulated with a spring constant k_i in the range of 0.01-2000$[N/m]$ chosen by the user. The sum of potential energy is then given by equation (2.11).

$$P = \sum_{i=1}^{n} m_i g^T r_{c,i} + \frac{1}{2} k_i q_i^2 \tag{2.11}$$

Here, the generalized coordinates and masses are the same as for the kinetic energy, g^T is the gravity vector showing the direction of gravity through the base frame, $r_{c,i}$ is the coordinates of center of mass for link i and k_i is spring-constant of joint i. The potential energy from the spring is just Hook's law for torsional springs, where displacement of the spring is given by the joint variable q_i.

Now, the equation of motion given by (2.12) can be derived by using the Euler-Lagrange equation given by (2.13)

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + K_s(q) = B(q)\tau \tag{2.12}$$

where $M(q)$ is the $n \times n$ symmetric inertia matrix, $C(q,\dot{q})$ is the $n \times n$ centrifugal and Coriolis matrix, $G(q)$ is the $n \times 1$ gravity vector and $K_s(q)$ is the $n \times 1$ spring vector. $B(q)$ is a $n \times m$ constant matrix (m is the number of actuated joints), τ is the $m \times 1$ generalized force vector and q, $\dot{q}$ and $\ddot{q}$ are respectively the vectors of $n \times 1$ generalized coordinates and its 1st- and 2nd-order derivatives with respect to time.

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} = \tau_k \tag{2.13}$$

Here, the inertia matrix $M(q)$ is given by the terms involving the 2nd-order derivative of q with respect to time (also the same matrix found in the kinetic energy for Lagrange function). The centrifugal and Coriolis matrix $C(q, \dot{q})$ is given by the terms including products of the 1st-order derivative of q with respect to time and can be found by equation (2.14). Both the spring and gravity vector contains the terms only involving q and no derivatives. $K_s(q)$ is given by the terms including spring constants, while $G(q)$ is given by the terms including gravity.

$$c_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i = \sum_{i=1}^{n} \frac{1}{2}\left(\frac{\partial d_{kj}}{\partial q_j} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right)\dot{q}_i \tag{2.14}$$

2.3 Virtual holonomic constraints

As mentioned in chapter 1.1, the system have passive dynamics. Finding realizable motions and controller design may then prove to be a difficult task. Fully actuated systems can reach any point in the desired trajectory at any time, as long as constraint such as velocity and acceleration are not exceeded. On the other hand, systems with one or more passive degrees of freedom cannot. This is a result of fewer control inputs than generalized coordinates available.

To overcome the complexity of underactuated systems, the virtual holonomic constraint (VHC) approach can be used. This method is a powerful tool for motion planning, analysis and control of mechanical systems (Mettin)[5] and is often used for stabilizing periodic motion of underactuated system as for example a biped without actuation in the knees. In contrast to holonomic constraints, which represent a physical restriction on the generalized coordinates, VHC is constraints imposed on the system through feedback control. The main idea is to "specify a somewhat coordinated motion as a geometric function of the generalized coordinates" (Mettin et al.)[6]. This means the desired trajectory of the generalized coordinates $q(t)$ can be described by a scalar function, hereby called $\theta(t)$, instead of directly with respect to time. Explicit reference to time then disappears from the system as the generalized coordinates becomes functions of θ, $q(\theta)$. The simplest case is naturally where one degree of freedom is passive, however, it is possible to use the same method for n-degree system, given a controller that stabilizes (n-1) arbitrarily chosen geometrical relations imposed on the generalized coordinates(Shiriaev et al.)[8]

For using VHC on the system, first step is to introduce the scalar function $\theta(t)$. There are many ways to choose this function. For instance the arc length of the orbit made by the trajectory given by (2.15), where the star indicates the optimal evolution of each generalized coordinate,

$$q_1 = q_{1\star(t)}, \quad q_2 = q_{2\star}(t), \cdots, q_n = q_{n\star}(t) \quad t \in [0, T] \tag{2.15}$$

location of center of mass or even as one of the generalized coordinates itself (Mettin)[5]. Often, it is also desired that $\theta(t)$ is monotonic function of time.

Next step is to introduce the constraints. These constraints are geometric functions related to the generalized coordinates as shown in equation (2.16) and are virtual holonomic, if the relations are preserved through feedback control, and not through physical constraints (Mettin)[5]. The geometric constraints are often chosen as trigonometric or polynomial functions, such as for example Bézier polynomial.

$$q_1 = \phi_1(\theta), \quad q_2 = \phi_2(\theta), \cdots, q_n = \phi_n(\theta) \quad \theta = \theta_\star(t) \quad t \in [0, T] \tag{2.16}$$

For simplicity, the generalized coordinates from (2.16) can be put together in vector-form as shown below.

$$q = \Phi(\theta) = \begin{bmatrix} \phi_1(\theta) \\ \phi_2(\theta) \\ \vdots \\ \phi_n(\theta) \end{bmatrix} \tag{2.17}$$

As the equation of motion is described with q, its first and its second derivative with respect to time, i.e. joint angle, velocity and acceleration, the vector

$\Phi(\theta)$ must consist of C^2-functions as shown in (2.18) and (2.19).

$$\dot{q} = \begin{bmatrix} \phi_1'(\theta)\dot{\theta} \\ \phi_2'(\theta)\dot{\theta} \\ \vdots \\ \phi_n'(\theta)\dot{\theta} \end{bmatrix} \tag{2.18}$$

$$\ddot{q} = \begin{bmatrix} \phi_1''(\theta)\dot{\theta}^2 + \phi_1'(\theta)\ddot{\theta} \\ \phi_2''(\theta)\dot{\theta}^2 + \phi_2'(\theta)\ddot{\theta} \\ \vdots \\ \phi_n''(\theta)\dot{\theta}^2 + \phi_n'(\theta)\ddot{\theta} \end{bmatrix} \tag{2.19}$$

2.4 Reduced dynamics

By using the virtual holonomic constraints and the scalar function $\theta(t)$, the problem of finding $q_{1\star}(t), \cdots, q_{n\star}(t)$ is reformulated to a problem of finding $\phi_1(\theta), \cdots, \phi_n(\theta)$ and $\theta_\star(t)$. By assuming the virtual holonomic constraints are exactly satisfied, the new relations for q and its derivatives given by (2.17), (2.18) and (2.19) can be substituted to the equation of motion given by (2.12) and rewritten as

$$M(\Phi)(\Phi''\dot{\theta}^2 + \Phi'\ddot{\theta}) + C(\Phi, \Phi')\Phi'\dot{\theta}^2 + G(\Phi) + K_s(\Phi) = B(\Phi)\tau \tag{2.20}$$

As the system is underactuated, the $B(\Phi)$ is a $n \times m$ matrix, where $n - m$ is the number of unactuated degrees of freedom. This means there exist a matrix $B^\perp$ such that $B^\perp B(\Phi)\tau = 0 \quad \forall\ \Phi$ (Shiriaev et al.)[7]. Equation (2.20) can then be rearranged to

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) = 0 \tag{2.21}$$

where $\alpha(\theta)$, $\beta(\theta)$ and $\gamma(\theta)$ are given by

$$\alpha(\theta) = B^\perp M(\Phi(\theta))\Phi'(\theta) \tag{2.22}$$

$$\beta(\theta) = B^\perp \left[C(\Phi(\theta), \Phi'(\theta))\Phi'(\theta) + M(\Phi(\theta))\Phi''(\theta)\right] \tag{2.23}$$

$$\gamma(\theta) = B^{\perp}\left[G(\Phi(\theta)) + K_s(\Phi(\theta))\right] \tag{2.24}$$

The equation (2.21) is the reduced dynamics, often called α-β-$\gamma-$equation. Solutions of this equation defines motions achievable for the complete under-actuated system. This is a result of the virtual holonomic constraints that restrict the system dynamics to evolve on a two-dimensional sub-manifold $[\theta, \dot{\theta}] \in \mathbb{R}^2$, instead of the original state space $[q, \dot{q}] \in \mathbb{R}^{2n}$. Hence, the dynamics is represented by one variable, θ, instead of n-degrees of freedom the system consist of.

Chapter 3

Ball Kinematics and System Dynamics

In the first part of this chapter, the ball kinematics and pitch distance will be explained. Then the steps of finding kinematics and dynamics through equations of motion and the reduced dynamics with virtual holonomic constraints for the robot manipulator will be shown. All symbolic computations for the dynamical model is derived in Maple 17, and the scripts are attached in the digital appendix.

3.1 Pitching distance

As the objective of this thesis is to pitch a ball as far as possible with the robot manipulator as explained in the problem description, it is necessary to find the linear velocity of the end-effector resulting in the best pitch. The velocity is naturally the main factor to achieve a long pitch, however, release angle, extension of robot and release height also influence the results. From simple mathematics, it can be proved that the pitch angle resulting in the longest pitch, would be of 45° with respect to the horizontal line. On the other hand, this is true for pitching at ground level, which the gripper for this robot is not. Both for football throw-ins (Linthorne and Everett)[12] and shot put (Linthorne)[13], the optimal release angle is closer to 30° than 45° and varies for every single person, as there are difference in height, physique, obtainable speed and spin of the ball (in football). However, the main reason for the optimal angle being much lower than 45° is that it is possible to achieve a higher release velocity at this angle. Figure 3.1 show the new system, where the red axis θ rotated by ψ, indicates the release angle the pitch.

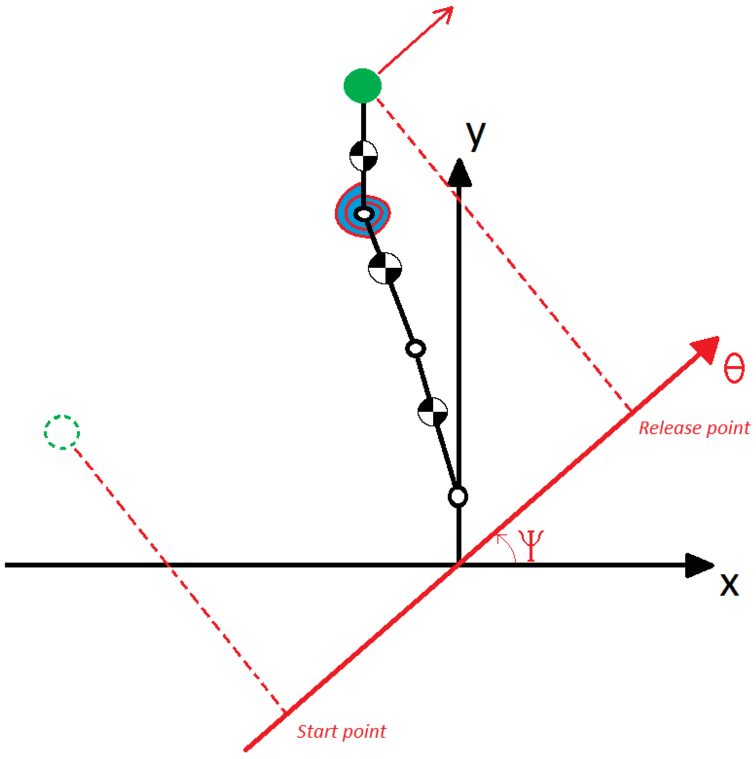


Figure 3.1: Underactuated 3-DOF robot manipulator with rotated axis for release angle.

During pitching phase, the ball held by the gripper at the end of link three can be found by the x and y coordinates given in equation (3.1). Velocity components in x and y direction are found by their derivatives with respect to time (not shown here as the vectors are trivial to compute and quite big).

$$p = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_1 cos(q_1) + l_2 cos(q_1 + q_2) + l_3 cos(q_1 + q_2 + q_3) \\ l_0 + l_1 sin(q_1) + l_2 sin(q_1 + q_2) + l_3 sin(q_1 + q_2 + q_3) \end{bmatrix} \tag{3.1}$$

Start position and velocity are given by p_b, $\dot{p}_b$, θ_b and $\dot{\theta}_b$ for their respective coordinates, while release position and velocity are given by p_r, $\dot{p}_r$, θ_r and $\dot{\theta}_r$

as shown below.

$$
\begin{aligned}
p_b &= [x_b, y_b]^T \quad \text{corresponding to} \quad \theta_b = \theta(0) \quad \text{at} \quad t = 0 \\
\dot{p}_b &= [\dot{x}_b, \dot{y}_b]^T \quad \text{corresponding to} \quad \dot{\theta}_b = \dot{\theta}(0) \quad \text{at} \quad t = 0 \\
p_r &= [x_r, y_r]^T \quad \text{corresponding to} \quad \theta_r = \theta(T_r) \quad \text{at} \quad t = T_r \\
\dot{p}_r &= [\dot{x}_r, \dot{y}_r]^T \quad \text{corresponding to} \quad \dot{\theta}_r = \dot{\theta}(T_r) \quad \text{at} \quad t = T_r
\end{aligned}
$$

At time T_r the gripper releases the ball, and it enters the ballistic flight phase. With release position and velocity as given by p_r and $\dot{p}_r$, the x- and y-coordinates during the flight is given by equation (3.2) and (3.3). This is correct by assuming there are no other external forces acting on the ball than gravity. In the real world, there would be air-resistance and possible ball rotation that would alter its trajectory as well.

$$x_f(t) = \dot{x}_r t + x_r \tag{3.2}$$

$$y_f(t) = y_r t - \frac{g}{2} t^2 + y_r \tag{3.3}$$

Using these two equations, the total distance travelled can be calculated for when the ball hits the x-axis, i.e. $y_f(t) = 0$. Solving equation (3.2) with respect to time t and substituting for time in equation (3.3), the distance travelled is given by equation (3.4), where $d = x_f(t)$.

$$d = \frac{\dot{x}_r \dot{y}_r}{g} + \sqrt{\left(\frac{\dot{x}_r \dot{y}_r}{g}\right)^2 + \frac{2\dot{x}_r^2 y_r}{g}} + x_r \tag{3.4}$$

It is important to notice that this distance is with respect to the origin point and not the release point. To find total distance travelled with respect to release point, x_r must be removed, as shown below.

$$d = \frac{\dot{x}_r \dot{y}_r}{g} + \sqrt{\left(\frac{\dot{x}_r \dot{y}_r}{g}\right)^2 + \frac{2\dot{x}_r^2 y_r}{g}} \tag{3.5}$$

By using one of these formulas, a simple example can be used to prove the optimal release angle varies with height. Assume for example you are standing on a cliff of 30 metres above the defined ground level. If you then throw a variety of rocks at initial velocity 5 m/s, the table below gives pitch distance accordingly to the release angle.

Table 3.1: Example of release angles versus pitch distance.

Release angle	**Pitch distance**
45°	7.8578 m
30°	9.3025 m
25°	9.6120 m
20°	9.8348 m
15°	9.9718 m
10°	10.0250 m
5°	9.9974 m
0°	9.8924 m

Here it can be seen that a release point of 10° result in a much longer pitch than 45°, however, it's only close to the optimal angle. By experimenting with the formula, it showed that the optimal release angle increase, as the initial velocity increases. Consequently, the optimal release angle varies with different conditions initial velocity and height of release point.

3.2 Model setup

Using KUKA lightweight robot 4+ to create a model, figure 3.2 shows one way of assigning the coordinate frames. From the problem description, only three joints will be used as generalized coordinates. These are $A_2 = q_1$, $A_3 = q_2$ and $A_5 = q_3$, while A_1, E_1, A_4 and A_6 are frozen with respect to the links connecting them. Resulting motion now only exist in a two dimensional space. As mentioned in chapter 2.1.1, there are more than one way of assigning the coordinate frames, hence, the frame assignment in figure 3.2 is not unique. However, the final transformation matrix should always be the same if computed correctly. Based on figure 3.2, the parameters a, α, d and ζ for each joint are found using the recipe from (Spong)[4, p. 110-111]. The DH-parameters for each respective joint are found in table 3.2.

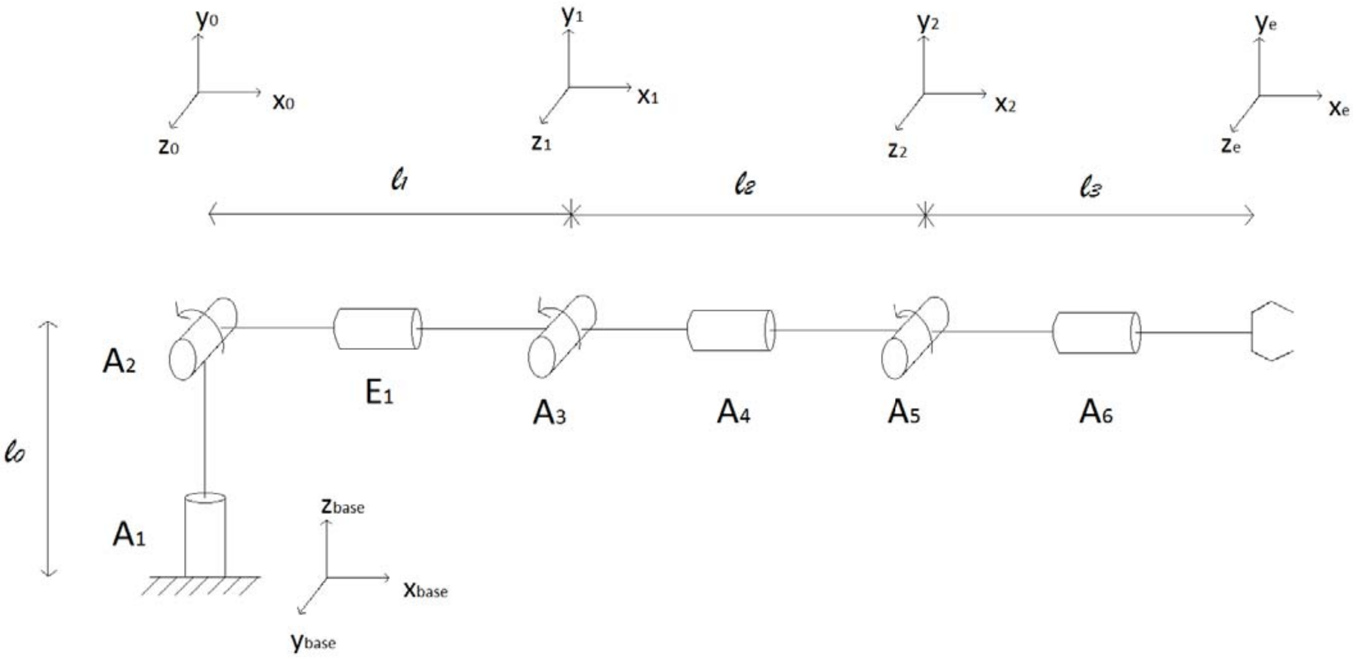


Figure 3.2: Representing KUKA lightweight robot 4+ with coordinate frames for each joint.

Table 3.2: DH-parameters from figure 3.2.

i	a_i	α_i	d_i	ζ_i
0	0	$\frac{\pi}{2}$	l_0	0
1	l_1	0	0	q_1^*
2	l_2	0	0	q_2^*
3	l_3	0	0	q_3^*

where q_1^*, q_2^* and q_3^* are joint variables.

From the DH-parameters in table 3.2, the transformation matrix are calculated using the Maple scripts in the digital appendix. It can be seen that the transformation matrix T_1^0 (named DH01 in the script) are given by transformation from A_1 to A_2, and not just transformation of A_2 alone. This is a result of a static first joint. As the base frame from figure 3.2 use z-axis in the vertical direction, this axis will be seen as the y-coordinate in the two dimensional model in figure 1.1 in the problem description. In the transformation T_3^0 (named DH03 in the script), it can be shown that the ball coordinates are exactly the same as the geometrically solved ball coordinates of vector (3.1). This proves the forward kinematics have been solved correctly by the Denavit-Hartenberg convention.

Once the forward kinematics are found, the Jacobians are a trivial matter to compute. The rotation about the z-axis together with the orientation vector of each respective generalized coordinate are extracted from the transformation matrices. For both linear and angular velocity Jacobians, the rotation z_0 and orientation o_0 is needed. This comes from the transformation matrix T_0^0 which is just the identity matrix. If interested, all the linear and angular velocity Jacobians are printed in the Maple script in the digital appendix.

The Lagrange-equation for the 3-DOF robot manipulator is given by equation (3.6).

$$\mathcal{L} = K_1 + K_2 + K_3 - P_1 - P_2 - P_3 \tag{3.6}$$

Here, K and P are calculated for their respective joints by using equation (2.9) and (2.11) respectively, as shown in chapter 2.2.

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{q}_1}\right] - \frac{\partial \mathcal{L}}{\partial q_1} = \tau_1 \quad \frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{q}_2}\right] - \frac{\partial \mathcal{L}}{\partial q_2} = \tau_2 \quad \frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{q}_3}\right] - \frac{\partial \mathcal{L}}{\partial q_3} = 0 \tag{3.7}$$

With the Euler-Lagrange equations as given by (3.7), the equations of motion can be calculated as explained in chapter 2.2. The equations of motion for the system is then given by (3.8). All matrices and vectors here are solved in Maple. As the matrices M and C and vectors G and K are quite big as well as there are not much to gain in studying them, they are not shown here.

$$M(q)\begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\\ \ddot{q}_3\end{bmatrix} + C(q,\dot{q})\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix} + G(q) + K_s(q) = \begin{bmatrix}\tau_1\\ \tau_2\\ 0\end{bmatrix} \tag{3.8}$$

As can be seen from the equations of motion above, the joints q_1 and q_2 are actuated by external torques τ_1 and τ_2 respectively, while the joint q_3 is unactuated.

3.3 Physical parameters

To have a very accurate model, the physical parameters are extremely valuable, which is why KUKA labs doesn't just give this information without having a really good reason for it. Instead, they provide CAD-models with roughly estimated parameters.

KUKA labs provided two models with slightly different parameters, and both are considered and used for the final model of this thesis. SolidWorks provide functions to easily measure lengths, masses, center of mass and inertias and can even calculate the same for more than one joint at the time. I.e. as link one in the model for this thesis consist of two links connected by a joint on the robot, they can be calculated as one entity. Assuming every link has uniformly mass density, SolidWorks gives principal moments of inertia. Using this information, inertia about the rotating joints can be calculated by the principal axis theorem given by equation (3.9) .

$$I = I_{cm} + ml_c^2 \tag{3.9}$$

Here, I_{cm} is the principal moment of inertia, m is the mass and l_c is the length from center of mass to the relevant rotating joint.

As already mentioned, the links of the model consist of more than one link of the real robot, as not all joints are used. Link one of the robot is link zero for the model, i.e. the base frame that elevates joint one. Link two and three of the robot creates link one for the model. Link four and five of the robot creates link two for the model, and link six together with the end-effector, gripping a ball, creates link three of the model. The physical parameters for link one and two are easily found through the CAD-model, however, it does not contain data for the gripper or the ball. Principal moment of inertia for the gripper and its center of mass are calculated on the lab, as some extra metal rings are used to attach it to the robot, which slightly changes all parameters from the datasheet. Now, roughly assuming a ball of diameter $0.07m$ and weight of $0.080kg$, center of mass for link 3 of the model are calculated by (3.10),

$$CoM = \frac{1}{M}\sum_{i=1}^{n} m_i r_i^2 \tag{3.10}$$

using the origin for the local coordinate system at the rotating wrist joint. Here, M is the total mass of objects, n is the total number of objects, m_i is mass of object i and r_i is the distance from center of mass for each respective object to the wrist joint coordinate system. Center of mass, principle moment of inertia and mass for the gripper with respect to a coordinate system of same orientation as the wrist joint, are given below (in the datasheet for the gripper, y- and z-axis are switched).

$$\begin{aligned} CoM_{gripper} &= \begin{bmatrix} -18.567 \\ 15.26 \\ 83.407 \end{bmatrix} mm \\ I_{cm,g} &= \begin{bmatrix} 0.00700 \\ 0.00700 \\ 0.00899 \end{bmatrix} kg \cdot m^2 \\ m_g &= 2.86073\ kg \end{aligned} \tag{3.11}$$

Adding total principal moment of inertia for all objects of link three, the total moment of inertia around the rotating joint are calculated using the parallel axis theorem. Result of these calculations and all other physical parameters used in the model are found in appendix A.

3.4 Virtual holonomic constraints

In this section, the task of finding the optimal evolution of each generalized coordinate $q_{1\star}(t)$, $q_{2\star}(t)$ and $q_{3\star}(t)$ for $t \in [0, T_r]$, assuming they exists, are reformulated to finding the scalar function $\theta_\star(t)$ and the virtual holonomic constraints with respect to θ. As explained in chapter 2.3, first step is to choose a scalar function $\theta(t)$. This is chosen as the ball position along the rotated axis from 3.1, which monotonically increases with respect to time and parametrize the trajectory as a function of the generalized coordinates. Through geometry, the scalar function is found to be

$$\theta = l_1 \cdot cos(q_1 - \psi) + l_2 \cdot cos(q_1 + q_2 - \psi) + l_3 \cdot cos(q_1 + q_2 + q_3 - \psi) \tag{3.12}$$

The C^2-smooth virtual constraints can now be written on the form of (3.13),

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \Phi(\theta) = \begin{bmatrix} \phi_1(\theta) \\ \phi_2(\theta) \\ f(\theta, \phi_1(\theta), \phi_2(\theta)) \end{bmatrix} \tag{3.13}$$

where $f(\theta, \phi_1(\theta), \phi_2(\theta))$ is equation (3.12) solved with respect to q_3 given by

$$q_3 = arccos\left(\frac{\theta - l_1 \cdot cos(\phi_1(\theta) - \psi) - l_2 \cdot cos(\phi_1(\theta) + \phi_2(\theta) - \psi)}{l_3}\right) + \psi - \phi_1(\theta) - \phi_2(\theta) \tag{3.14}$$

3.5 Virtually constrained dynamics

With the virtual holonomic constraints and the scalar function $\theta(t)$ as above, the dynamical system found by the equations of motion from (3.8) can be reformulated to a reduced form with the same methodology as explained in chapter 2.4. By substituting the generalized coordinates and its first and second derivative with respect to time (given the constraints are C^2-smooth) to the equations of motion, the new virtually constrained dynamics are found using equations (3.15), (3.16) and (3.17).

$$\alpha(\theta) = B^{\perp} M(\Phi(\theta))\Phi'(\theta) \tag{3.15}$$

$$\beta(\theta) = B^{\perp} \left[C(\Phi(\theta), \Phi'(\theta))\Phi'(\theta) + M(\Phi(\theta))\Phi''(\theta)\right] \tag{3.16}$$

$$\gamma(\theta) = B^{\perp} \left[G(\Phi(\theta)) + K_s(\Phi(\theta))\right] \tag{3.17}$$

When $B^{\perp} = [1, 1, 1]^T$ the virtually constrained dynamics are given by equation (3.18), (3.19) and(3.20). Now, the whole system is parametrized by the evolution of one variable, and the solutions exists in the phase-plane of $[\theta, \dot{\theta}]$.

$$\alpha_1(\theta)\ddot{\theta} + \beta_1(\theta)\dot{\theta}^2 + \gamma_1(\theta) = \tau_1(\theta) \tag{3.18}$$

$$\alpha_2(\theta)\ddot{\theta} + \beta_2(\theta)\dot{\theta}^2 + \gamma_2(\theta) = \tau_2(\theta) \tag{3.19}$$

$$\alpha_3(\theta)\ddot{\theta} + \beta_3(\theta)\dot{\theta}^2 + \gamma_3(\theta) = 0 \tag{3.20}$$

The reduced dynamics on the other hand are given by the unactuated part, i.e. when $B^{\perp} = [0, 0, 1]$ and given by the 2nd-order differential equation

(3.20). Script for calculating $\alpha(\theta)$, $\beta(\theta)$ and $\gamma(\theta)$ can be found in the digital appendix. Now, along any solution in the phase-plane $[\theta, \dot{\theta}]$, following equation holds (proof are given in appendix B).

$$\frac{d^2}{dt^2}\theta(t) = \frac{d}{d\theta}\frac{1}{2}\dot{\theta}(t)^2 \tag{3.21}$$

This makes it possible to convert the 2nd-order differential equation from (3.20) to a 1st-order ODE (3.22) by substituting $Y(\theta) = \dot{\theta}^2$.

$$\frac{1}{2}\alpha(\theta)\frac{d}{d\theta}Y(\theta) + \beta(\theta)Y + \gamma(\theta) = 0 \tag{3.22}$$

Assuming $\alpha(\theta) \neq 0$, the 1st-order ODE can be solved on general form

$$Y = \exp\left\{-2\int_{\theta_0}^{\theta}\frac{\beta(\tau)}{\alpha(\tau)}d\tau\right\}Y_0 - \int_{\theta_0}^{\theta}\exp\left\{-2\int_{s}^{\theta}\frac{\beta(\tau)}{\alpha(\tau)}d\tau\right\}\cdot 2\frac{\gamma(s)}{\alpha(s)}ds \tag{3.23}$$

Substituting back for Y then gives

$$\dot{\theta}(t)^2 = \underbrace{\exp\left\{-2\int_{\theta_b}^{\theta(t)}\frac{\beta(\tau)}{\alpha(\tau)}d\tau\right\}}_{\Psi_1}\dot{\theta}_b^2 - \int_{\theta_b}^{\theta(t)}\underbrace{\exp\left\{-2\int_{s}^{\theta(t)}\frac{\beta(\tau)}{\alpha(\tau)}d\tau\right\}\cdot 2\frac{\gamma(s)}{\alpha(s)}}_{\Psi_2}ds \tag{3.24}$$

which is a function of the squared velocity that is desired to maximize at time $t = T_r$ when $\theta(T_r) = \theta_r$ and $\dot{\theta}(T_r) = \dot{\theta}_r$. To find the highest possible velocity $\dot{\theta}_r$, the optimization problem is then to find the constraint functions $\phi_1(\theta)$ and $\phi_2(\theta)$ that maximize the velocity at release point θ_e.

$$\begin{aligned} J = \int_{\theta_b}^{\theta_e} & [\Psi_1(s, \phi_1(s), \phi_2(s), \phi_1'(s), \phi_2'(s), \phi_1''(s), \phi_2''(s))\dot{\theta}_b^2 \\ & + \Psi_2(s, \phi_1(s), \phi_2(s), \phi_1'(s), \phi_2'(s), \phi_1''s(s), \phi_2''(s))]ds \\ & + \Phi(\theta_b, \phi_1(\theta_b)\phi_2(\theta_b), \phi_1'(\theta_b), \phi_2'(\theta_b), \phi_1''(\theta_b), \phi_2''(\theta_b), \\ & \qquad \theta_e, \phi_1(\theta_e)\phi_2(\theta_e), \phi_1'(\theta_e), \phi_2'(\theta_e), \phi_1''(\theta_e), \phi_2''(\theta_e))\dot{\theta}_b^2 \to max \end{aligned} \tag{3.25}$$

If the maximizers $\phi_1(\theta)$ and $\phi_2(\theta)$ are C^2-smooth, they should by necessity satisfy the new Euler-Lagrange functions given by (3.26) and (3.27) (Arnold)[10].

$$\frac{d}{ds}\left[\frac{\partial}{\partial\phi_1'}\left(\Psi_1\dot{\theta}_b^2 + \Psi_2\right)\right] - \frac{\partial}{\partial\phi_1}\left(\Psi_1\dot{\theta}_b^2 + \Psi_2\right) = 0 \tag{3.26}$$

$$\frac{d}{ds}\left[\frac{\partial}{\partial \phi_2'}\left(\Psi_1 \dot{\theta}_b^2 + \Psi_2\right)\right] - \frac{\partial}{\partial \phi_2}\left(\Psi_1 \dot{\theta}_b^2 + \Psi_2\right) = 0 \tag{3.27}$$

Solving the new Euler-Lagrange equations explicitly requires 3rd- and 4th-derivative of $\phi_1(\theta)$ and $\phi_2(\theta)$, and a great of amount of computational power, as they are quite huge. Therefore, it will be solved by the approximated solution, using the performance index of (3.28) by safely assuming start velocity to be $\dot{\theta}_b = 0$. The objective function can then be rewritten to

$$J = \int_{\theta_b}^{\theta_e} \left[\Psi_2(s, \phi_1(s), \phi_2(s), \phi_1'(s), \phi_2'(s), \phi_1''(s), \phi_2''(s))\right] ds \to max \tag{3.28}$$

Chapter 4

Approximating Optimal Release-Velocity

As mentioned in last chapter, solving the explicit solutions of the Euler-Lagrange equations would be both hard and computational expensive. So, to approximate a solution, this chapter will introduce polynomials representing the geometric functions defining the virtual holonomic constraints, go through the optimization procedure for optimizing a pitch and analyse some characteristics for the reduced dynamics.

4.1 Bézier polynomial

To approximate a trajectory that optimizes the release velocity, the Bézier polynomial will be used. This is a way of curve fitting to approximate a motion, using polynomial functions. As the polynomial degree increases, the approximations will get closer to the real curve (the real curve will be given as the polynomial degree of ∞)[11]. Start point for every curve is always given by the first control point, e.g. a_0 of equation (4.1). End points of curves are also given by the last control points. Intermediate control points generally does not lie on the curve itself. Figure 4.1 shows a Bézier curve created of six control points. Here it can be seen that the blue smooth line, the Bézier curve, move towards the control points, connected by linear black lines.

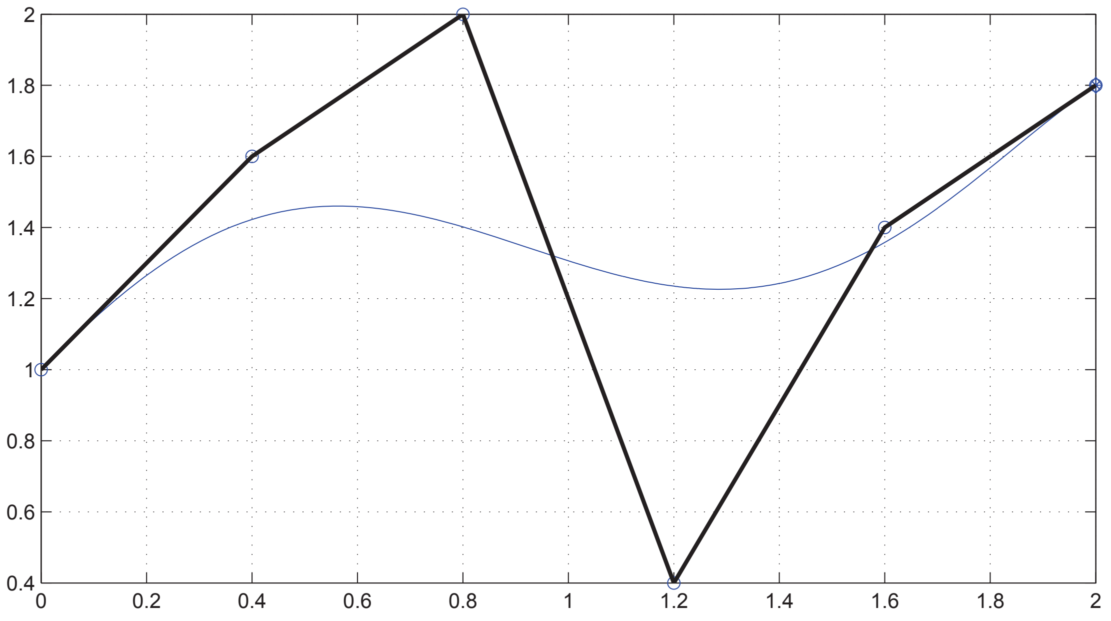


Figure 4.1: Example of Bézier curve.

For the virtual holonomic constraints $\phi_1(\theta)$ and $\phi_2(\theta)$, Bézier polynomial will be used. The general structure for these are given by equation (4.1) and (4.2) respectively.

$$\phi_1(\theta) = \sum_{k=0}^{M} a_k \frac{M!}{k!(M-k)!} s^k (1-s)^{M-k}, \quad s = \frac{\theta - \theta_b}{\theta_r - \theta_b} \tag{4.1}$$

$$\phi_2(\theta) = \sum_{k=0}^{M} b_k \frac{M!}{k!(M-k)!} s^k (1-s)^{M-k} \tag{4.2}$$

Here, the polynomial degrees are given by M, which is chosen as the degree of the geometric relations between the generalized coordinates[3]. The control points, consisting of scalar coefficients, are given by the vectors

$$a = [a_0, a_1, \cdots, a_M] \tag{4.3}$$

$$b = [b_0, b_1, \cdots, b_M] \tag{4.4}$$

In the search of longest possible pitch, these control points must be set in such a manner that motion created by $\phi_1(\theta)$ and $\phi_2(\theta)$ together, creates maximum release velocity $\dot{\theta}_r$ for the ball.

4.2 Optimization task

Finding the control points for the Bézier polynomials resulting in maximum velocity is a numerical optimization task. There are quite a few software to handle such tasks, e.g. Maple, Mathematica, MATLAB and more. For this thesis, a function called $fmincon$ from the optimization toolbox of MATLAB will be used. Fmincon is a gradient-based method for finding minimum point for nonlinear constrained multi-variable functions, and can use four different search algorithms: $trust - region - reflective$, $interior - point$, $active - set$ and sqp (sequential quadratic programming). However, it is important to notice that it finds local minimum points, not global. The trust-region-reflective method cannot be used, as constraints must be linear. Interior-point use a sequence of approximation minimization problems, which result in quite small steps throughout the optimization search of problem (4.6). Active-set and sqp are quite similar, however, active-set is a relatively inefficient method. Sqp on the other hand is the state of the art for nonlinear programming methods, and the main differences from the active-set method are [14]:

- Sqp has strict feasibility with respect to bounds.
- Sqp can attempt to take steps that fails (resulting in complex number, NaN or Inf) and recover from it.
- Sqp is much more efficient and attempts to obtain feasibility through second-order approximation of constraints if a step causes greater constraint violations.

Evaluating these features, it was not a difficult decision choose the sqp algorithm to use in the fmincon function for this thesis.

Originally in the specialization project, the optimization task was to find maximum release velocity by finding parameter vectors a and b as in (4.5).

$$\underset{a,b}{max}\ J,\quad s.t. \begin{cases} \dot{\theta}_b = 0 \\ \frac{1}{6}\pi \leq \phi_1(\theta) \leq \frac{7}{6}\pi \\ \mid \phi_2(\theta) \mid \leq \frac{2}{3}\pi \\ \mid q_3(\theta, \phi_1(\theta), \phi_2(\theta)) \mid \leq \frac{2}{3}\pi \\ \mid \theta - l_1 \cos(\phi_1 - \psi) - l_2 \cos(\phi_1(\theta) + \phi_2(\theta) - \psi) \mid \leq l_3 \\ \mid \phi_1'(\theta)\dot{\theta} \mid \leq \frac{11}{18}\pi \\ \mid \phi_2'(\theta)\dot{\theta} \mid \leq \frac{32}{45}\pi \\ \mid \dot{q}_3(\theta, \dot{\theta}, \phi_1(\theta), \phi_2(\theta), \phi_1'(\theta), \phi_2'(\theta)) \mid \leq \frac{46}{45}\pi \\ \mid \tau_1(\theta) \mid \leq 176 \\ \mid \tau_2(\theta) \mid \leq 100 \end{cases} \tag{4.5}$$

Here, $\tau_1(\theta)$ and $\tau_2(\theta)$ are found by solving for $\ddot{\theta}$ in equation (3.20) and substituting back to equation (3.18) and (3.19).

However, there are many conditions that must be considered before finding an optimal trajectory. There are spring coefficient, interval of $\theta \in [\theta_b, \theta_e]$ and release angle that also affect the pitching, though the main factor is to have a large release velocity. Reformulating the search for finding maximum pitch-distance instead of maximum release velocity, will ensure that the mixture of parameters and velocity results in the longest pitching distance. The new optimization problem can now be given by (4.6)

$$\max_{a,b,\theta_b,\theta_e,\psi,k_{spr}} \quad d(\max(\dot{\theta})) \tag{4.6}$$

using the same constraints as (4.5).

Efficiency wise for the optimization, there have been a few challenges. The dynamic model are quite huge, and for solving the reduced dynamics with respect to $\dot{\theta}^2$, exponential functions and double integral that are computational expensive must be solved. A few ways for computing this have been tried, and the best known solution is to solve equation (4.7) stepwise by creating vectors containing the whole interval of θ. The y_1^{-1}-function inside y_2 will have a double integral, which is solved using interpolation. For saving even more computation time, the objective function and nonlinear constraints are solved in a nested function. This results in about 30% faster computation. The structure of (4.7) is build upon the first equation of (B.9) from appendix B, which shows step by step how to solve the 2nd-order differential equation from (3.20) with respect to $\dot{\theta}^2$.

$$\begin{aligned} y_1(\theta_1, \theta_2) &= \exp\left(-2\int_{\theta_1}^{\theta_2} \frac{\beta_3(\tau)}{\alpha_3(\tau)} d\tau\right) \\ y_2(\theta_1, \theta_2) &= \int_{\theta_1}^{\theta_2} y_1^{-1}(\theta_1, s) \cdot 2\frac{\gamma_3(s)}{\alpha_3(s)} ds \\ \dot{\theta}^2(\theta_b, \theta) &= -y_1(\theta_b, \theta) \cdot y_2(\theta_b, \theta) \end{aligned} \tag{4.7}$$

Still, there are other factors in play for how fast the calculations are. If $\alpha(\theta)$ at some point in the interval of θ is zero, the integral struggles to give any solutions, as the denominators only consist of α-functions. Another crucial factor is the number of samples to be used. Usually, 100 samples gives quite accurate approximations (normally not more than a few centimetres difference from the real pitch distance), but constraints are not always perfectly upheld as a consequence. Figure 4.2 and 4.3 shows the approximated reduced dynamics of equation (3.20) of and arbitrary pitching motion using 100 and 10000 samples respectively. If the approximations had been perfect, they should be equal to zero for the hole interval, which they clearly are not. On the other hand, 10000 samples gives results much closer to zero than 100 samples, though there are quite a trade-off with respect to computation time. The starting spike is a result of unknown initial values for the acceleration of $\theta(t)$, $\ddot{\theta}(t)$.

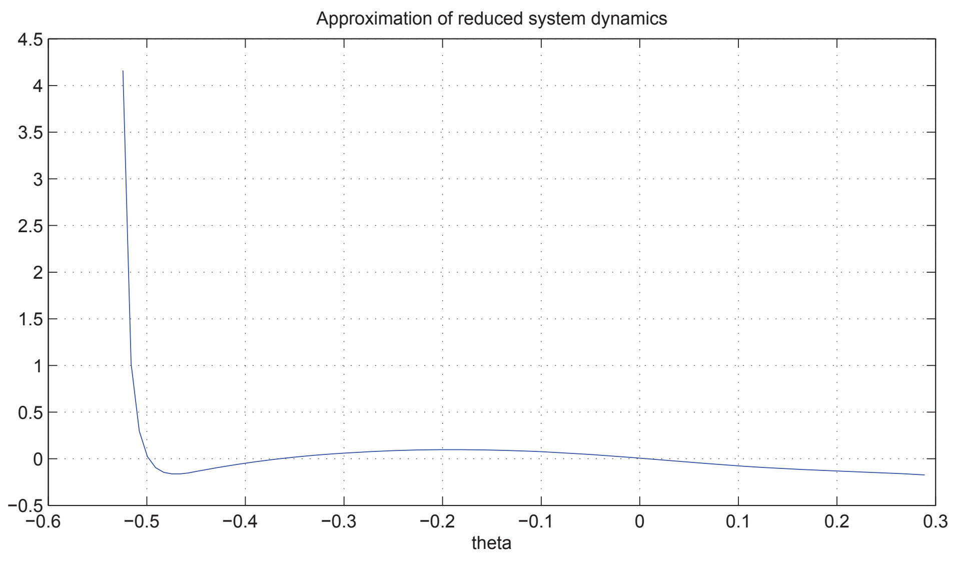


Figure 4.2: Approximation of the reduced dynamics using 100 samples.

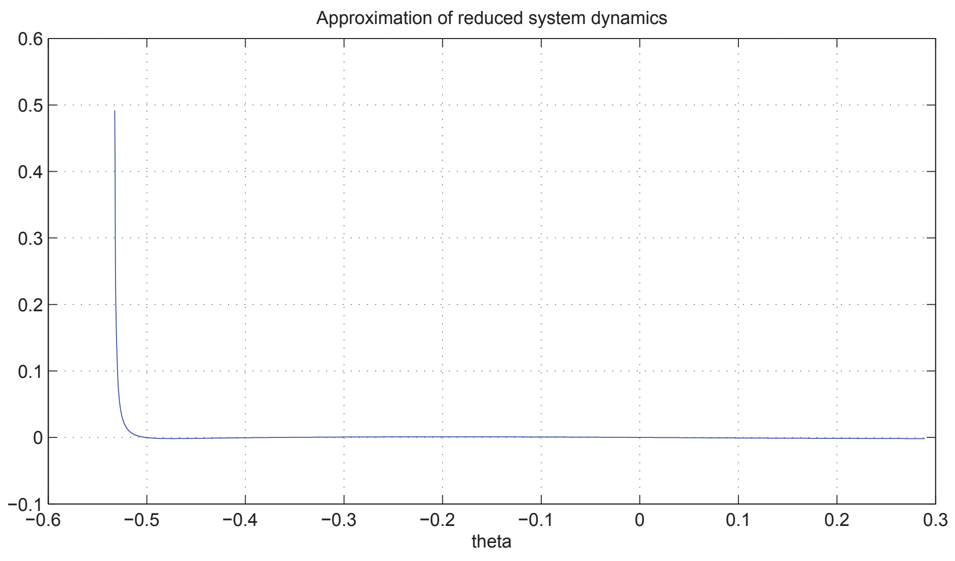


Figure 4.3: Approximation of the reduced dynamics using 10000 samples.

4.3 Characteristics of the reduced dynamics

For the reduced dynamics, almost all optimal trajectories found have the same characteristics. There are always at least one equilibrium point, and the leftmost equilibrium is always a center. If there exist more than one in the interval, it's usually a saddle point. In figure 4.4 an arbitrary trajectory with optimized virtual holonomic constraints are illustrated, using a variety of different initial conditions. This phase-portrait shows an interesting behaviour. There are possibilities for periodic motion, however, the optimized motion goes to zero (and then becomes complex) when θ increases. Other trajectories also goes to infinity outside the interval, but one of these to options always happens. In other words, desired motion for the robot is only stable for a small interval of θ.

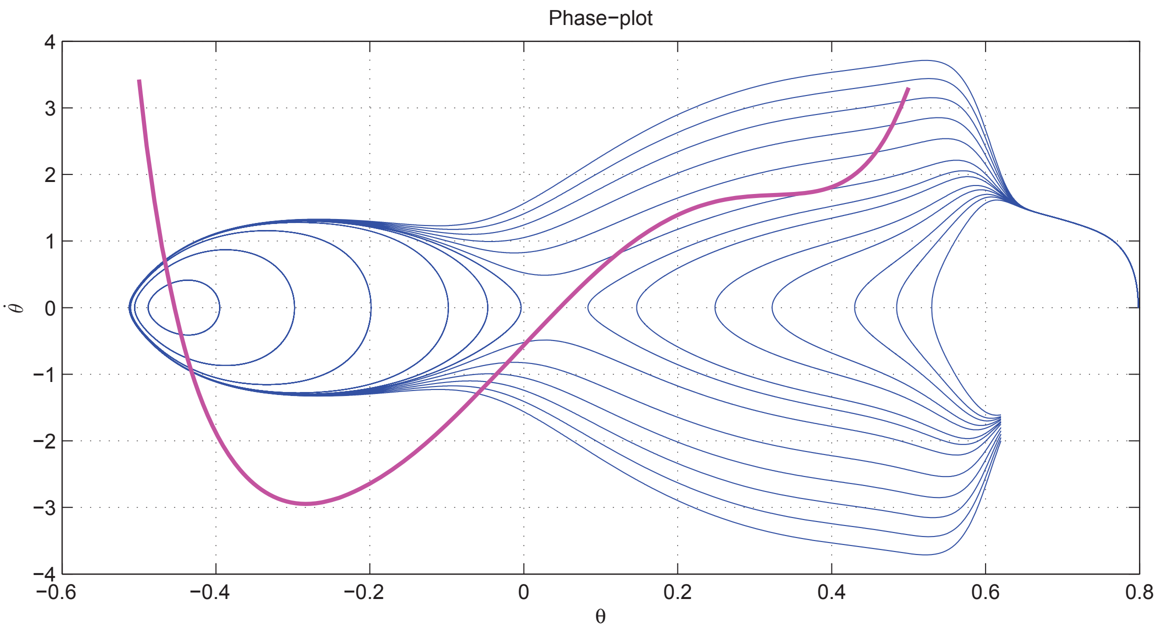


Figure 4.4: Phase-portrait of the system for an arbitrary pitching-motion, using a variety of initial conditions.

The pink plot is $\gamma(\theta)$ of the reduced dynamics for the interval of $\theta \in [-0.5, 0.5]$, for the same virtual holonomic constraints. The unique property of the γ-function is quite handy. All equilibrium points for the desired interval are easily found; as seen in the figure, $\gamma(\theta) = 0$ wherever there are an equilibrium.

$$\omega = \left[\frac{d}{d\theta}\frac{\gamma(\theta)}{\alpha(\theta)}\right]\Bigg|_{\theta=\theta_0} \tag{4.8}$$

Using this equation, where θ_0 is an equilibrium for the reduced system, it can be seen if it is center, saddle, stable or unstable focus without checking phase portrait. If $\omega < 0$, it is a saddle-point, is $\omega > 0$ it means the equilibrium is either a center, stable, or unstable focus. By linearizing the reduced dynamics given by (3.20), what kind of equilibrium it is can be found. Proof given by (Shiriaev et al.)[8]. A commonly used controller for underactuated systems is transversal linear feedback control, however, periodic motion is a necessity. Consequently, it can not be used. Though, as already mentioned, the KUKA lightweight robot 4+ have internal controllers. Using the *Fast Research Interface*, there are several types of controllers available. For experimenting, the *Joint Impedance Controller* will be used. This one is able to individually set desired stiffness and damping for all joint individually and following a predetermined trajectory can be done in three ways: position control, torque control, or both of these active at the same time.

Chapter 5

Results

In this chapter, numerical results achieved from optimization of pitching distance with the model of the KUKA lightweight robot 4+ and comparisons of experimental results are presented. For not damaging hardware, the robot have velocity constraint on all joints. Hence, optimization have mainly been focused to satisfy this. However, some optimization problems have been solved without the constraint on the wrist joint for the possibility of comparing simulations with a robot, having physically passive joint, as well as the results from (Mettin and Shiriaev)[3].

5.1 Simulations compared to experiments

In the search for optimal pitches, finding a feasible starting point is not always an easy task for this system. However, throughout many tries, quite a few motions have been found. These can of course be found in the digital appendix. Usually, the phase portrait have the same characteristics for motion as in figure 5.1, though another type of motion for the pitch is also found, shown in figure 5.8. The pitch resulting in the longest distance for both these types of motion, have been used for experimenting and are presented in chapter 5.1.1 and 5.1.2.

5.1.1 Pitch one

The first pitch used for testing on the robot is created with the parameters from table 5.1. This is in total 19 parameters that solve the optimization problem. Linear release velocity, shown in the phase portrait of figure 5.1, is found to be $\dot{\theta} = 3.954$ m/s at $\theta_e = 0.3772$ m and evolve over a period of 434.8 ms. This corresponds to a pitch distance of 2.95 m, which is confirmed by the animation in figure 5.2.

Table 5.1: Parameters used to compute the motion of pitch one.

Parameter	**Value**
$\dot{\theta}_b$	0 m/s
θ_b	-0.3377 m
θ_e	0.3772 m
Release angle	18.988°
Spring coefficient	21.9020
Bézier control point vector 1	$[1.732, 1.642, 1.505, 1.515, 1.426, 1.369, 1.311]$
Bézier control point vector 2	$[0.666, 0.648, 0.714, 0.541, 0.469, 0.399, 0.332]$

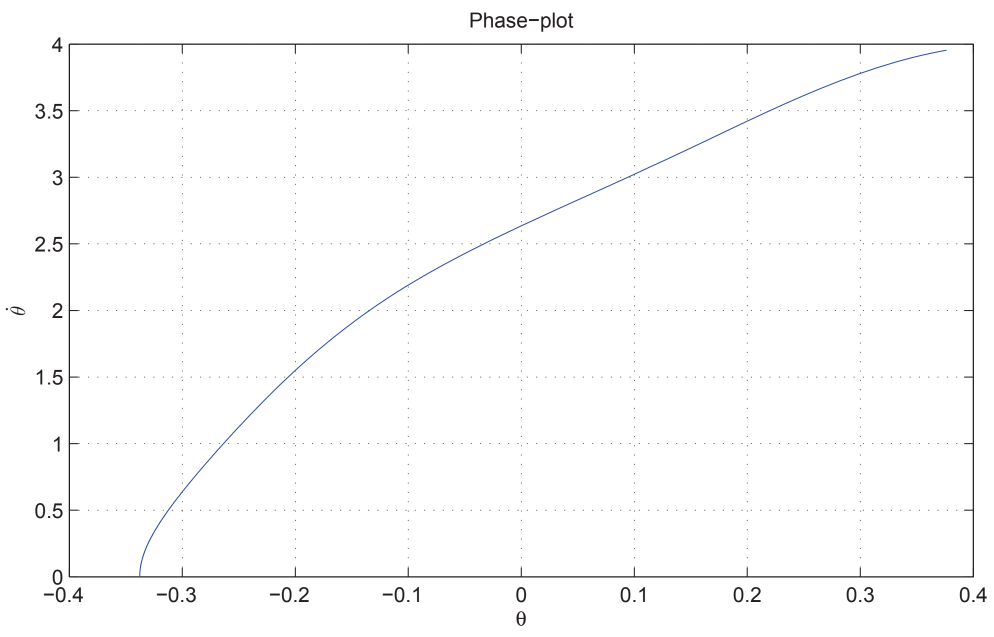


Figure 5.1: Phase portrait of pitch one.

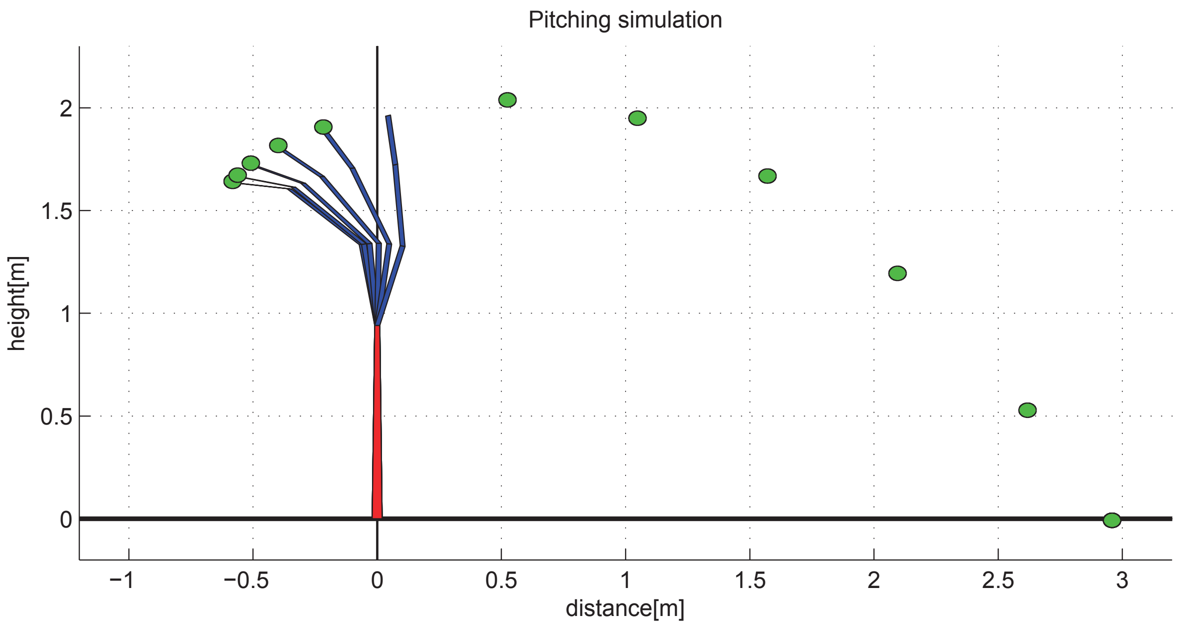


Figure 5.2: Animation of pitch one.

In figure 5.3, the simulated position for joint one are compared to the "user-defined" given back from the robot. In other words, the reference trajectory the robot tries to follow. Data for these two should be identical, however, the input data to the robot deviate to some degree from the original data. Same kind of deviation is on data for all joints and both pitches. This of course affect the trajectory to some degree.

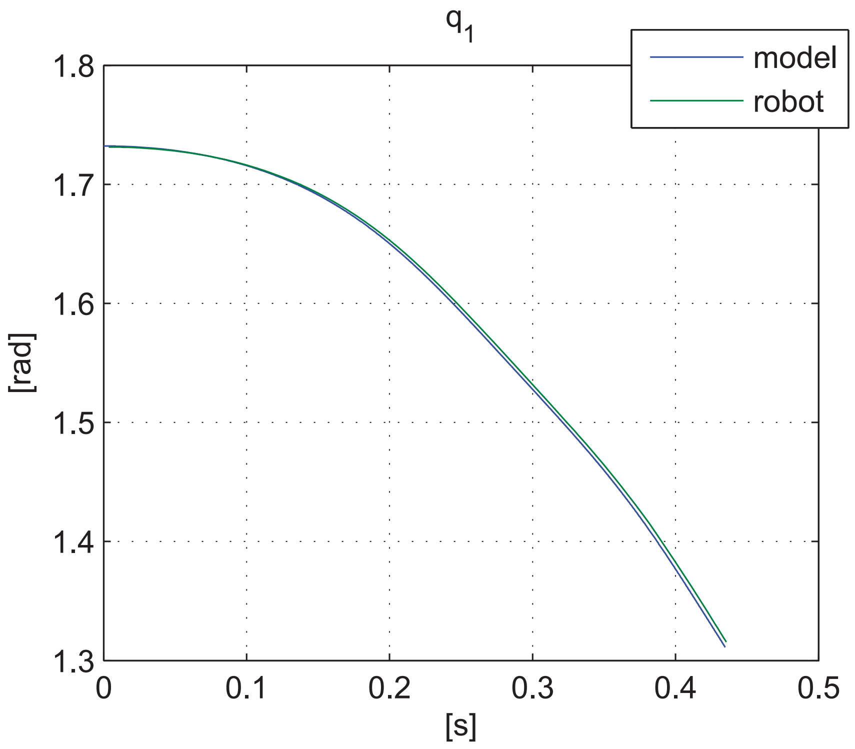


Figure 5.3: Simulated position for joint one plotted together with data the robot gives back as user defined data for the same joint.

Figure 5.4, 5.5 and 5.6 presents simulated data compared to measured for joint angle, velocity and torque respectively. From the angle comparison, it can be seen that there is deviation for all joints, especially number three. A portion of this is due to the reference trajectory being slightly different from simulated data. Consequently, the joint velocity also deviate from the simulation. However, for joint one and two, the end velocity actually reach the limits (the horizontal black lines). Again, joint three deviate quite a lot from the simulation, a total of 1.19 rad/s. As a consequence, from position and velocity, release angle is 24.2° and release velocity $\dot{\theta} = 3.72$ m/s, giving a pitch distance of 2.67 m in theory. The real distance on the other hand proved otherwise, and varied from approximately 3-3.4 m.

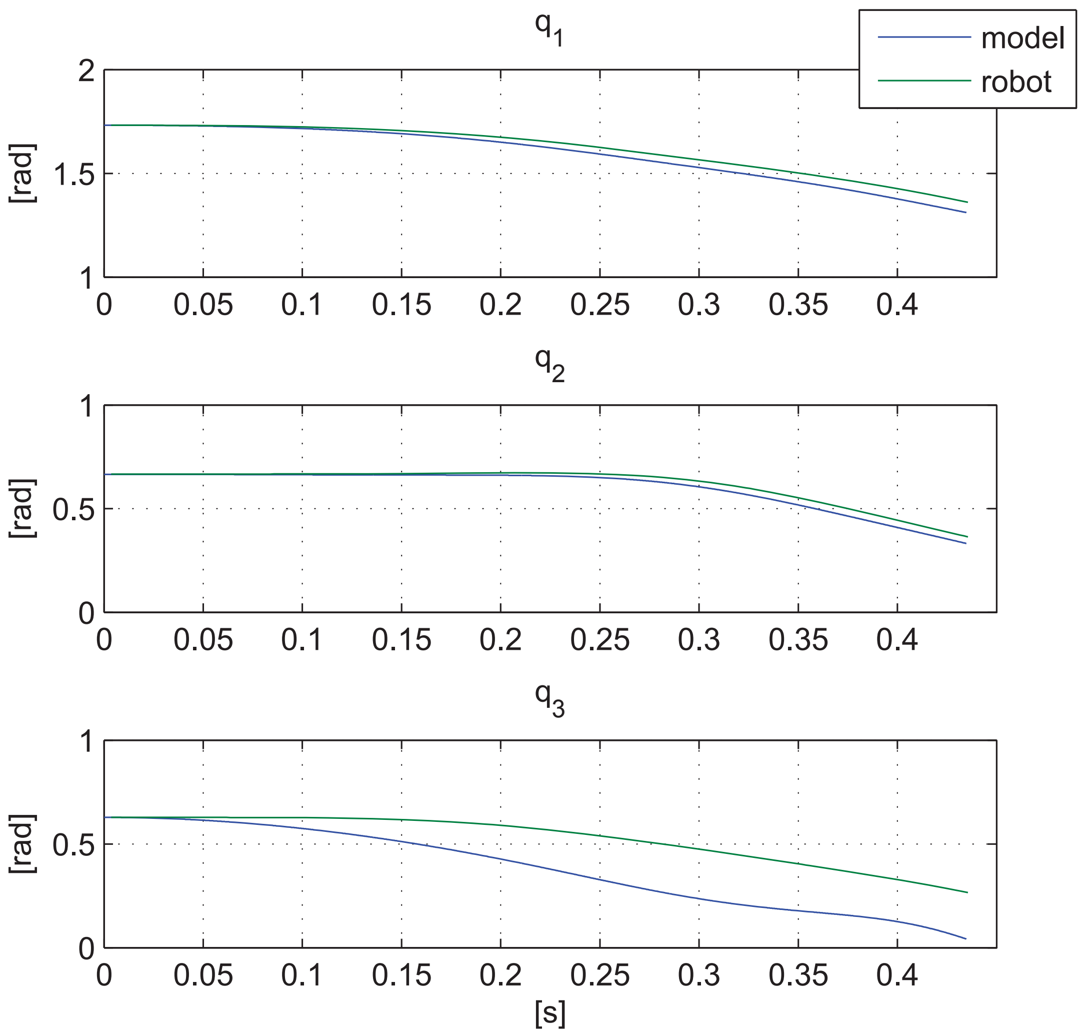


Figure 5.4: Simulated joint angles compared to measured joint angles of the robot.

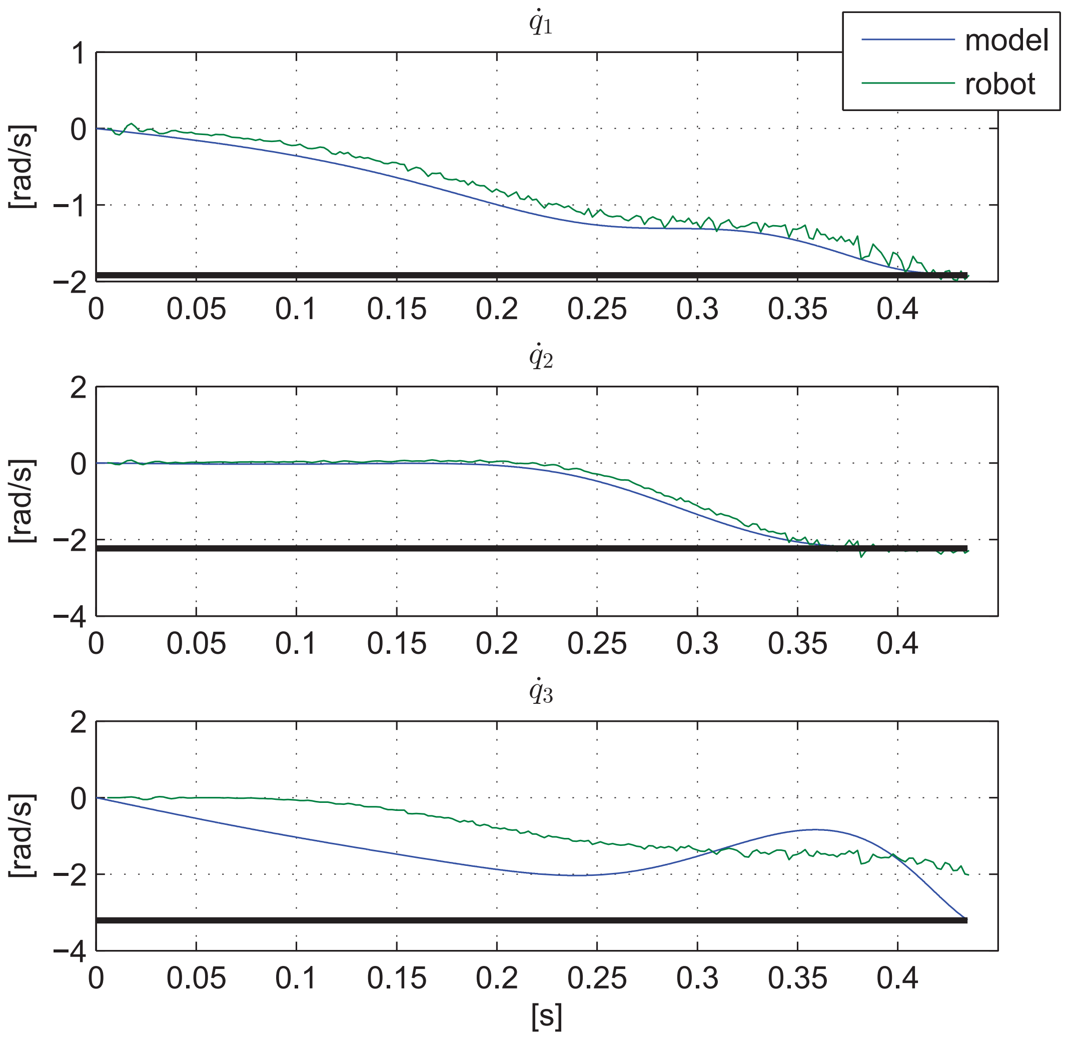


Figure 5.5: Simulated joint velocity compared to velocity of the robot.

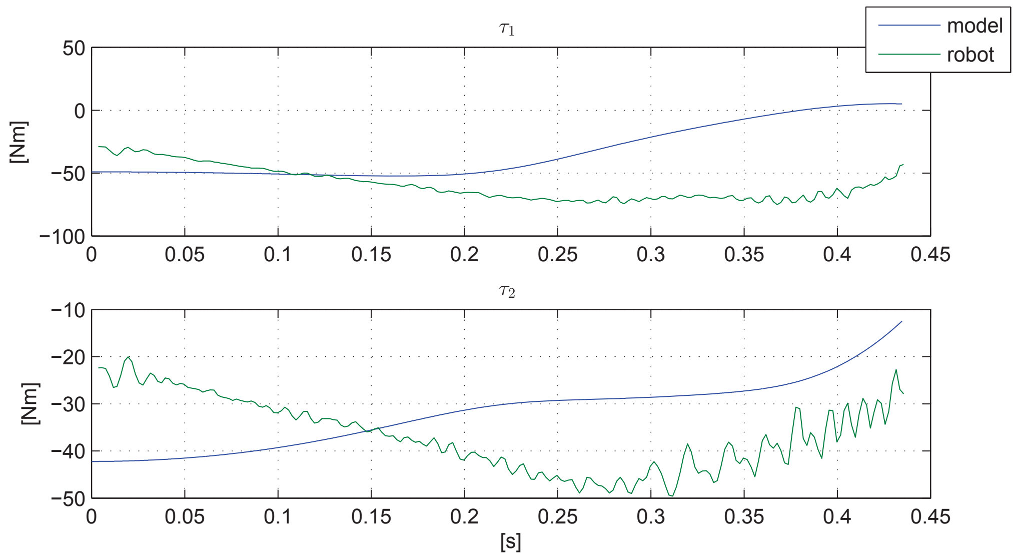


Figure 5.6: Simulated torque compared to measured torque of the robot.

For this pitch, both position and torque control was used. Figure 5.6 shows simulated torque compared to the measured, which can be seen to be quite different from each other. Position control alone were also tried, though there were only small differences in the measured torque.

A picture of the robot in its starting position can be seen in figure 5.7. Base frame is here elevated by 63 cm above ground. Video of the robot pitch can also be found in the digital appendix if interested.

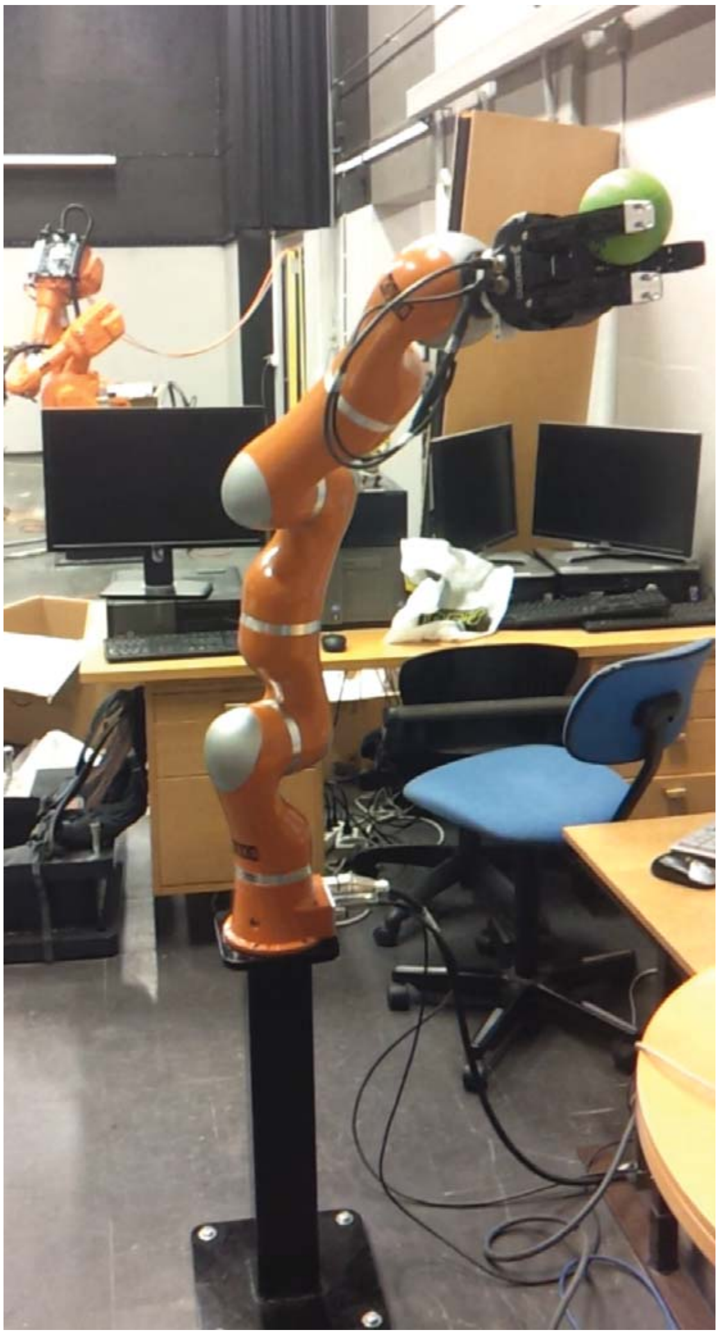


Figure 5.7: Start position of robot for pitch one.

5.1.2 Pitch two

The second pitch used for testing is created with the parameters from table 5.2, containing 17 parameters. Solving the reduced dynamics gives linear velocity of ball equal to $\dot{\theta} = 4.057$ m/s at release point of $\theta_e = 0.289$ m, evolving over a period of 588.3 ms. This is slightly longer time than pitch one, but does not give a longer distance thrown with respect to the base frame. The distance from base is 2.61 m, though, release point is here at -0.365 m. Animation of this trajectory is shown in figure 5.9. Compared to the first pitch, in this simulation, the robot is more stretched out at release point, resulting in higher linear velocity.

Table 5.2: Parameters used to compute the motion of pitch two.

Parameter	**Value**
$\dot{\theta}_b$	0 m/s
θ_b	-0.5326 m
θ_e	0.2890 m
Release angle	37.099°
Spring coefficient	20.2472
Bézier control point vector 1	$[2.298, 2.174, 2.056, 1.940, 1.859, 1.782]$
Bézier control point vector 2	$[0.637, 0.662, 0.349, 0.390, 0.292, 0.202]$

Figure 5.10, 5.11 and 5.12 presents simulated data compared to measured for joint angle, velocity and torque respectively for pitch two. As for pitch one, it can also be seen deviations here. The movement for joint three is also more dynamic compared to pitch one and struggles to follow the reference trajectory. Same goes for the velocity. Joint one and two isn't that bad, but joint three is way of, especially at release point. Theoretically from the measured data, release angle is 49.06° and $\dot{\theta} = 3.29$ m/s resulting in a pitch of 1.51 m. The real pitch distance on the other hand was approximately 2.5 m.

Compared to the first pitch, this one only used position control for following reference trajectory. However, when it comes to torque, the deviations are not really worse for pitch two.

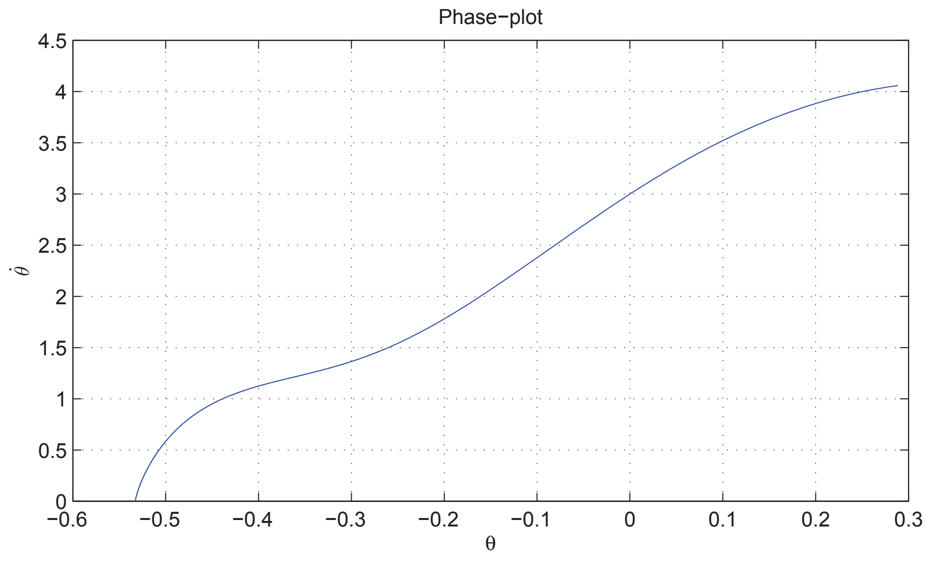


Figure 5.8: Phase portrait of pitch two.

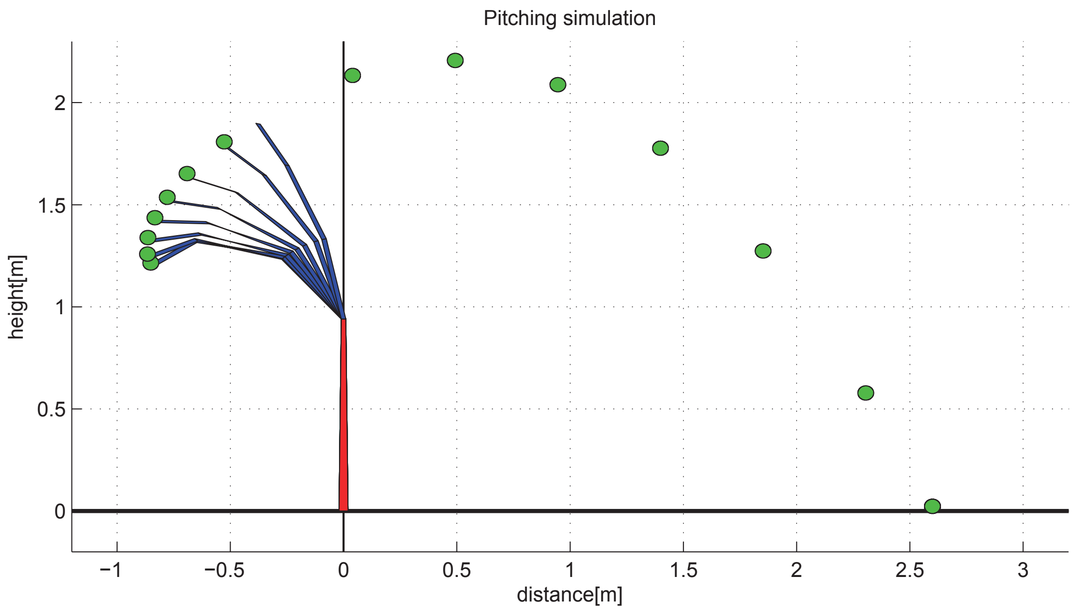


Figure 5.9: Animation of pitch two

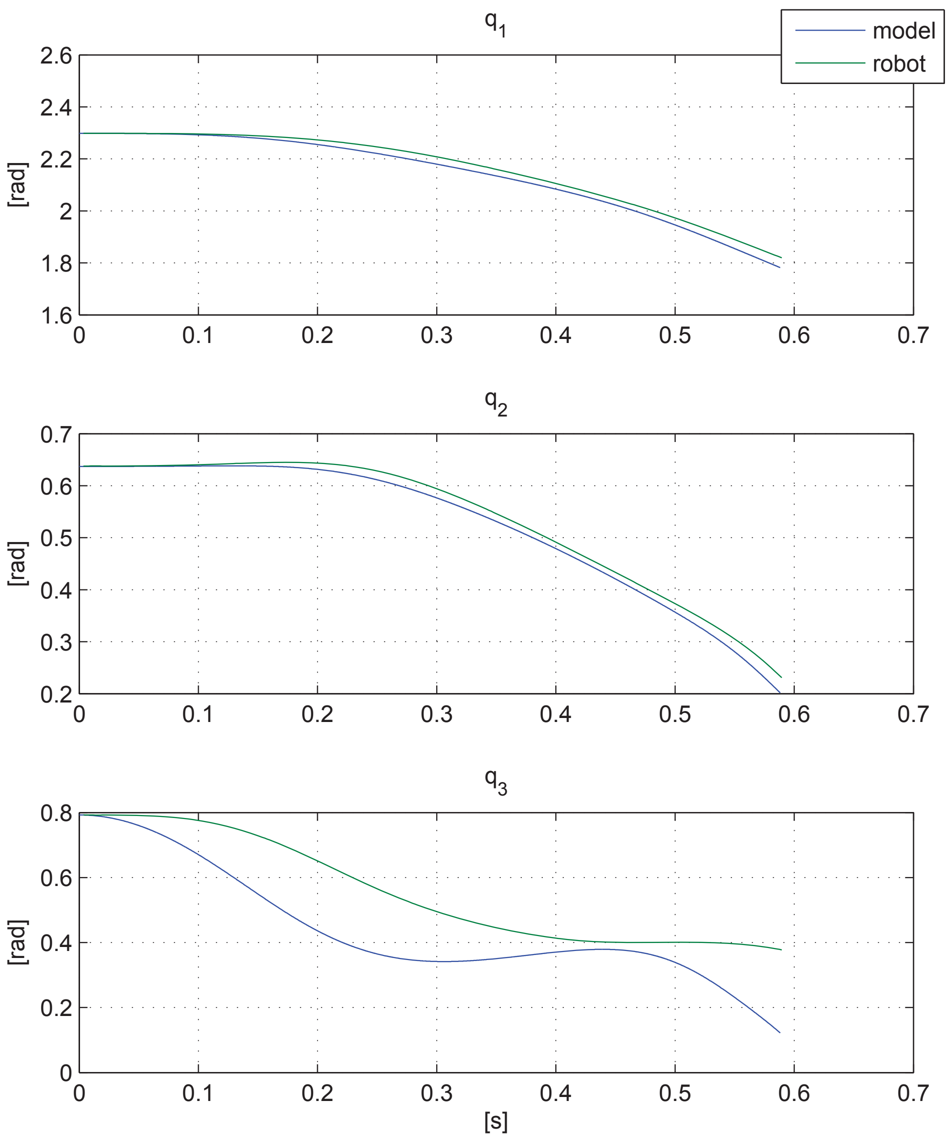


Figure 5.10: Simulated joint angles compared to measured joint angles of the robot.

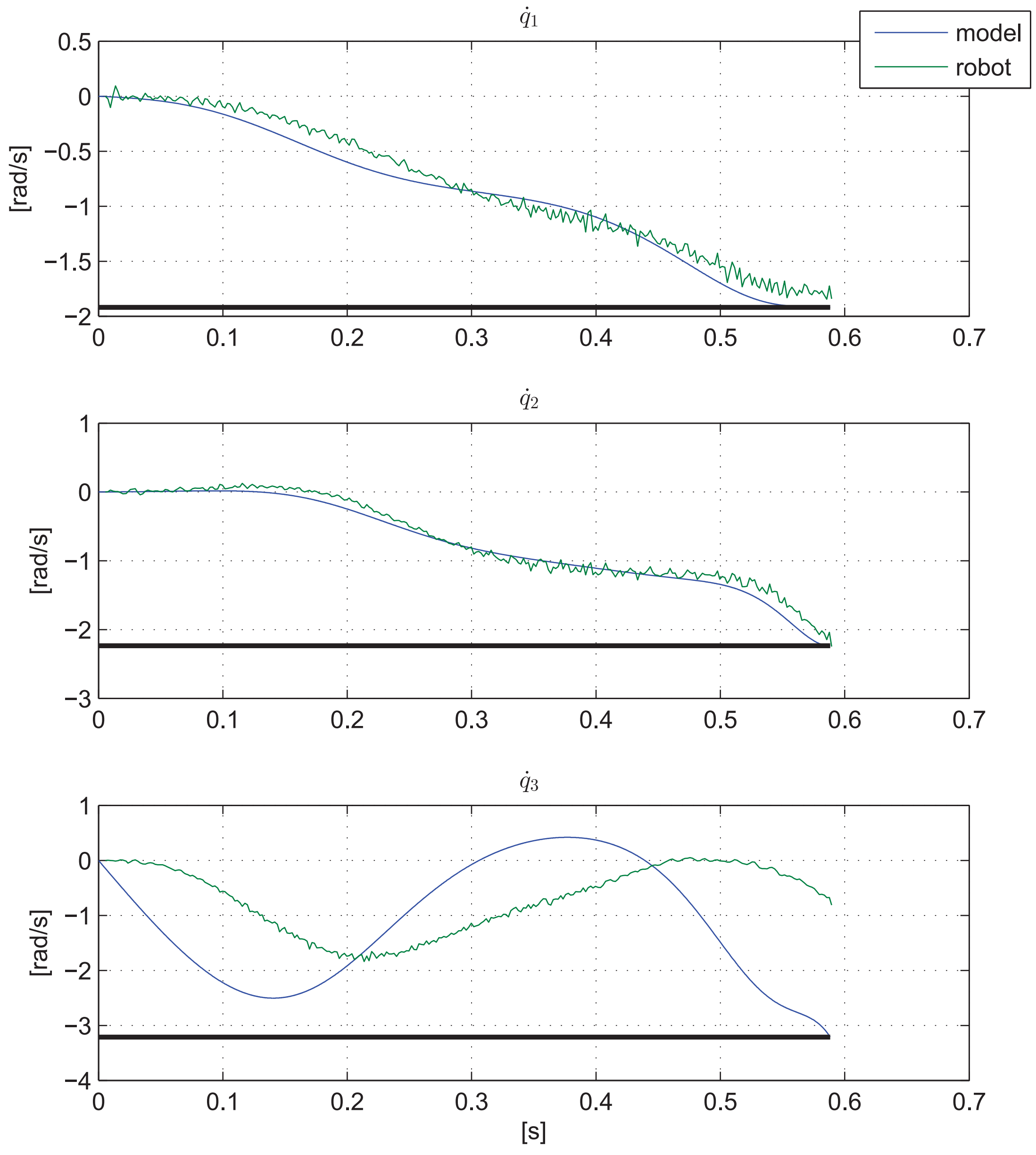


Figure 5.11: Simulated joint velocity compared to velocity of the robot.

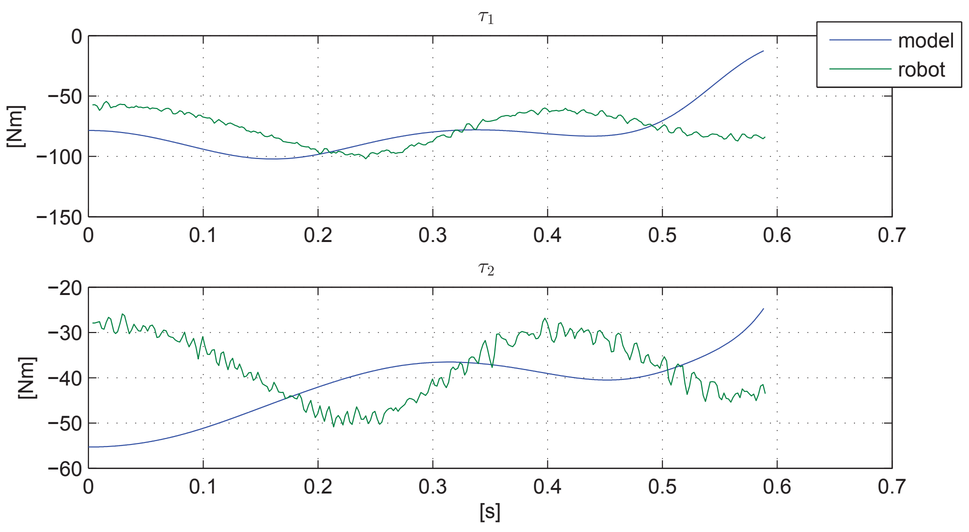


Figure 5.12: Simulated torque compared to measured torque of the robot.

5.2 Optimized pitch without velocity constraint on wrist joint

Without using velocity constraint on the wrist joint for the numerical search of optimized pitching motion, it opens up possibilities for much longer pitches. However, this also makes it harder to determine if solutions are close to the global maximum without doing a large amount of searches, or use another software for finding global points. Nonetheless, results from one such search will be presented here.

The longest pitch found throughout the search are created by adding the parameters from table 5.3 to the model. This is a total of 17 parameters solving the optimization problem and resulting in maximum velocity $\dot{\theta} = 9.893$ m/s at release point $\theta_e = 0.1026$ m as shown in the phase-portrait. The ball only move 0.291 m before it's released, corresponding to a time interval of 82.5 ms. This is roughly five and seven times shorter interval than pitch one and two respectively. As a result, joint one and two reach maximum velocity very fast and have small movement, while joint three have quite big motion. This can be verified by looking at the animation of figure 5.17.

Table 5.3: Parameters used to compute the pitching motion without velocity constraint on wrist joint.

Parameter	**Value**
$\dot{\theta}_b$	0 m/s
θ_b	-0.1822 m
θ_e	0.1026 m
Release angle	23.640°
Spring coefficient	20.5240
Bézier control point vector 1	$[1.332, 1.230, 1.283, 1.254, 1.250, 1.240]$
Bézier control point vector 2	$[1.135, 1.096, 1.212, 1.142, 1.159, 1.146]$

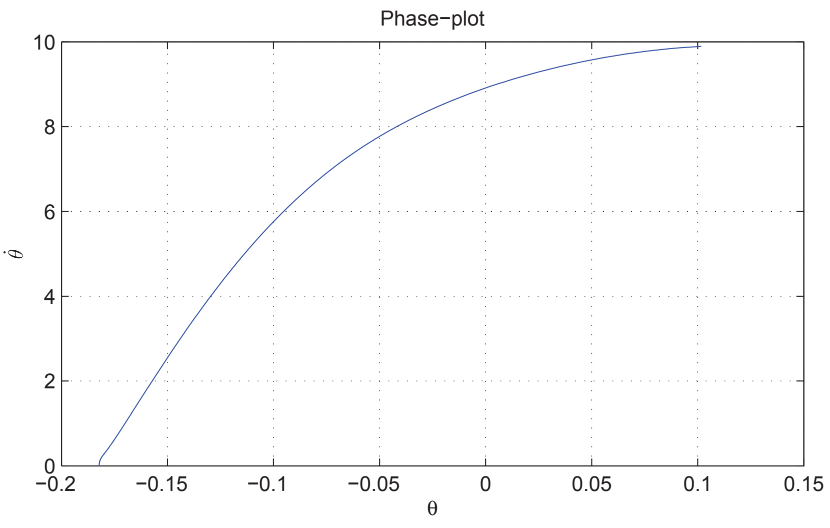


Figure 5.13: Phase portrait of the optimized pitch.

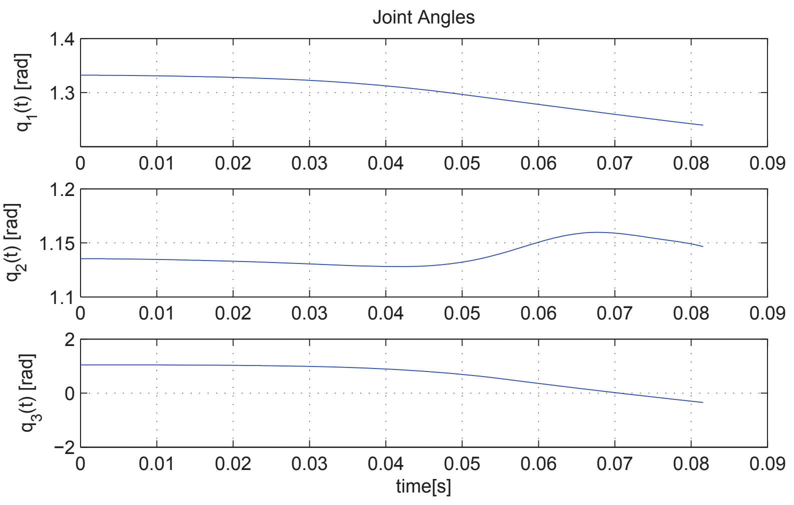
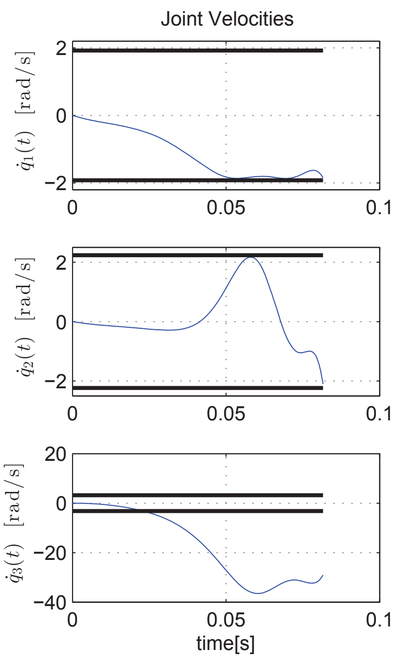


Figure 5.14: Joint angles for the optimized pitching motion.



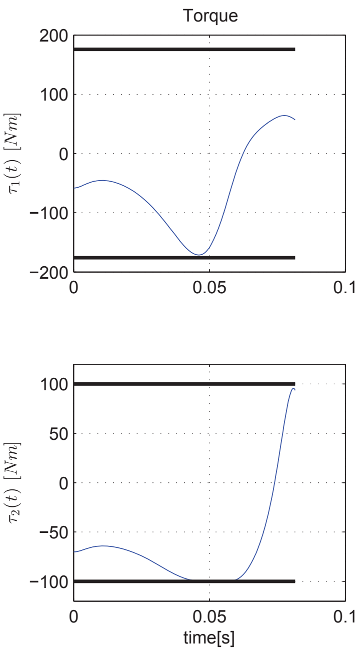


Figure 5.15: Joint velocities for the optimized pitching motion.

Figure 5.16: Torque for the optimized pitching motion.

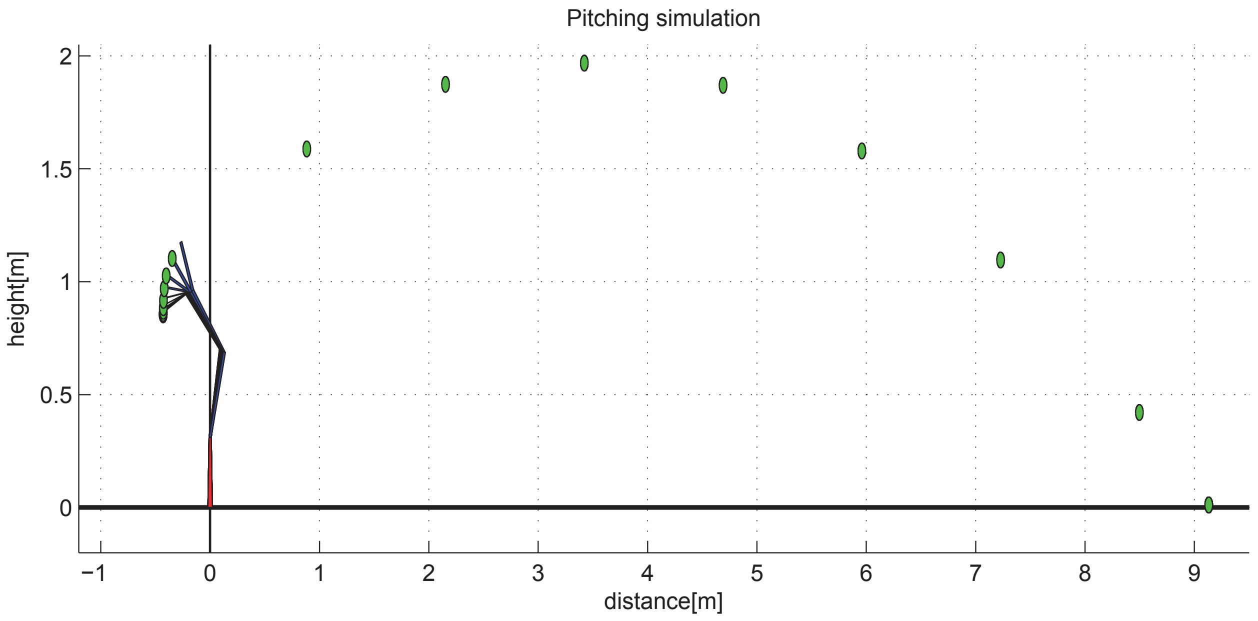


Figure 5.17: Animation of pitch.

Even though the pitch results just described are the absolute largest found (0.15 m larger than second largest), there are many very similar motions found, using a variety of starting points. If interested, data for these can be found in the digital appendix. This type of pitch has not been pursued very much, but it proves that there are potential for finding much larger pitch distances when the relevant constraint is removed.

Chapter 6

Discussion

Using a passive joint on a robot for pitching a ball as far as possible, it can be considered if KUKA lwr is the best robot to use. The robot itself has of course no physical passive joints, which adds extra weight and inertia. For not damaging hardware, there are strict constraints, especially on velocity. The wrist joint is also very small, though a quite heavy gripper is connected, extending the reach a little further. However, this also increase inertia, resulting in reduced acceleration. As constraints must be upheld for all joints at all time, it is possible to roughly estimate release velocity and pitch distance by hand. If all joints rotate at maximum speed and joint two and three (for the model) reach zero degrees, resulting in maximum extension, the linear velocity of gripper is approximately 4.18 m/s. From the base frame, this gives a pitch of 2.75 m at release angle 14°. In simulation of optimized pitch, the longest distance was 2.95 m from the base frame. This is 0.2 m longer, however, there are trade-off between release angle, speed and release point that result in longer pitch distance. So, by hand-calculation, it can be proved that it is physically not possible to throw much longer by using this robot.

In figure 5.7 displaying the robot for pitch one at the start position, it can be seen that the ball is not actively held by the gripper. During pitch testing, a couple of methods were tried for actively holding the ball and release it at the desired point, however, it did not work. It would seem like commands can only be sent to the gripper while the robot is standing still, which is quite inconvenient for the task at hand. Another method of gripping the ball with much force and slowly release it were also tried, but it proved to be difficult to synchronize the release and end point of trajectory. For this reason, the ball was placed on top of two fingers. This gives a slightly shifted center of mass, however, the ball is very light compared to the wrist joint and gripper, meaning it does not affect the pitching motion noticeably.

In the comparisons between simulation and testing on the robot, it was presented that user defined data for position is not the same as the simulated data, even though they should be exactly the same. The reason for this occurrence is still unclear, however, this undoubtedly affect the final results for both pitches. Two different control methods are used. Torque and position control for pitch one and position control alone for pitch two. With respect to simulated torque compared to the measured, both are quite far of the target, so it is hard to say which controller works best. A test of position control for pitch one was also carried out. With an exception of noise however, the controllers have approximately the same deviation from desired trajectory. The reason for these deviations are most likely inaccurate model used in the optimization problem. Especially mass, which again affects inertia. If these two parameters are larger in the robot, which most likely are the case, it means each joint will not be able to accelerate as fast as desired. This does not seem to affect joint one and two very much, but joint three, the wrist joint, is way of target on position and velocity. The desired case would be to have the joint completely passive, but the robot actually use position control, together with a torsional spring. Since the gripper is quite heavy, it reduce the possible acceleration for the joint, resulting in large deviations. Especially pitch two, which is a more dynamic motion, seems to be playing catch-up with the reference when looking at the velocity.

For both pitch one and two, the measured data indicates release points of the robot are lower with respect to the x-axis than the simulated data. Consequently, release angle are higher and velocity lower. Still, the robot managed to pitch up to 3.4 m and 2.5 m respectively for pitch one and two. At the end of each pitch, the robot stops because torque limits are exceeded. This stop is quite abruptly, therefore the ball most likely get an extra nudge before it is released, resulting in a longer pitch.

In the numerical search, pitch distance is measured as length from base frame to landing point. Pitch one is the longest in terms of this definition, however, looking at pitches from release point, this is not the case. As can be seen in figure 5.5 and 5.11, all joints of the simulation have maximum angular velocity. Now, looking at the animation of each pitch, it can be seen that pitch two has higher release angle and is more stretched out, resulting larger velocity for the ball. Consequently, the total pitch distance with respect to release point is actually 2.98 m compared to pitch one of 2.9 m.

Compared to (Mettin and Shiriaev)[3], the robot used here is not optimized for pitching objects with a passive joint. The passive joint of the robot used in numerical study by Mettin and Shiriaev is long and light (without velocity constraint), while the wrist joint and gripper on the KUKA robot is heavy and short, resulting in small linear velocity. To even get close to the same pitching distance, the wrist joint would need an enormous rotational velocity, which is the case in figure 5.15 for pitch three in the results. Compared to the first two pitches, it can be seen in figure 5.16 that also torque is starting to constrain the motion. However, when the mass for the model is assumed to be quite inaccurate, this pitch will most likely break the torque limits. Therefore resulting in a smaller pitch distance too.

Chapter 7

Conclusion and Further Work

7.1 Conclusion

In the specialization project, the reduced dynamic system still hadn't been tested, and was therefore not proved to be correct. However, through analysis and simulations of the system, the model was found to be fully potential of finding trajectories for the robot, using geometric parametrization. Utilizing this for the optimization problem, many tests were conducted, resulting in quite a few different trajectories, both with and without velocity constraint for the wrist joint. As weight of object to be thrown increased, finding feasible starting points also got significantly harder.

As all joints easily reach maximum angular velocity, the degree of scalar control points for the Bézier polynomial is assumed to not affect the total pitch length very much, though it increases computational time of the optimization. For pitches without velocity constraint on the wrist joint on the other hand, increased control point degree may result in better motion for the first two joints, giving a longer pitch. However, this has not been tested much, as the main focus was to find motions to test on the robot.

Both through the optimization and hand-calculation, maximum pitch length proved to be just below three metres, and in simulation, the constricting factor is without a doubt velocity of all joints. Hence, it is physically not possible to throw much longer than the results presented, and it is assumed that the solution is close to the global optimum. However, as the model is not correct in terms of mass, it is still unknown if the torque also would constrict the motion. In fact, when experimenting with the robot, it always stopped at the end of the trajectory, giving a warning of torque limits exceeded. Therefore, it might be that the torque also constrain motion. Still, a more accurate model should be acquired to test this.

7.2 Further work

Even though the pitching simulation with all physical constraints active is most likely very close to the global optimum, better virtual holonomic constraints may add a few centimetres to the pitch. As these constraints are just an approximation, higher coefficient degree means it can get closer to the optimal motion. Consequently, each coefficient added results in longer computation time, so it must be considered if such a study is worth using time on. On the other hand, the computational time can be reduced by computing the gradient in parallel and adding it directly to the objective function. Physically it may not be so fascinating. However, analytically it would be interesting to see how close to the global optimum it would be possible to reach through this approximation.

As concluded, the model is not very accurate as a result of roughly estimated parameters from CAD-models, that may be inaccurate in itself. Both for this thesis and for other projects using this robot, it could benefit to know more about the physical parameters. Thus, it is highly recommended to do a study of parameter estimation.

The gripper, as already mentioned in chapter 6, proved to be difficult to synchronize with the robot. For the experiments, the gripper ended up not holding the ball actively, but were used more or less as the bucket for a catapult. In future work, the gripper should be used actively. Either by somehow manage to command the gripper while the robot is moving, or determine opening speed, force and orientation of fingers in such a way that the gripper get the command before the robot starts moving. Then slowly opens the fingers, resulting in the ball being release at the correct point.

Appendix A

Physical parameters

Table A.1: Parameters used for the model of KUKA lightweight robot 4+.

Parameter	**First link**	**Second link**	**Third link**
Mass(kg)	$m_1 = 4.60714$	$m_2 = 3.89944$	$m_3 = 4.5131$
Length(m)	$l_1 = 0.4$	$l_2 = 0.39728$	$l_3 = 0.24$
Length to CoM(m): x-axis	$l_{1c,x} = 0$	$l_{2c,x} = 0$	$l_{3c,x} = -0.01177$
Length to CoM(m): y-axis	$l_{1c,y} = -0.02522$	$l_{2c,y} = 0.0222$	$l_{3c,y} = 0.01109$
Length to CoM(m): z-axis	$l_{1c,z} = 0.198$	$l_{2c,z} = 0.1561$	$l_{3c,z} = 0.10649$
Inertia: x-axis	$I_{1,x} = 0.0086628$	$I_{2,x} = 0.007843$	$I_{3,x} = 0.01085$
Inertia: y-axis	$I_{1,y} = 0.096343$	$I_{2,y} = 0.063329$	$I_{3,y} = 0.01092$
Inertia: z-axis	$I_{1,z} = 0.2743713$	$I_{2,z} = 0.15196$	$I_{3,z} = 0.06355$

Parameter	**Quantity**
Gravitational constant	$g = 9.81$ m/s^2
Length of link 0	$l_0 = 0.3105$ m
Length of platform the robot is placed on	0.63 m

Table A.2: Constraints used for the model of KUKA lightweight robot 4+.

Quantity	**Constraints**
Configuration space [rad]	$q_1 \in [-\frac{1}{6}\pi, \frac{7}{6}\pi]$ $q_2 \in [-\frac{2}{3}\pi, \frac{2}{3}\pi]$ $q_3 \in [-\frac{2}{3}\pi, \frac{2}{3}\pi]$
Velocity [rad/s]	$\lvert\dot{q}_1\rvert \leq \frac{11}{18}\pi$ $\lvert\dot{q}_2\rvert \leq \frac{32}{45}\pi$ $\lvert\dot{q}_3\rvert \leq \frac{46}{45}\pi$
Torque [N/m]	$\lvert\tau_1\rvert \leq 176$ $\lvert\tau_2\rvert \leq 100$

Appendix B

Solving the reduced dynamics with respect to $\dot{\theta}$

The reduced dynamics written as

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) = 0. \tag{B.1}$$

Assuming $\alpha(\theta) \neq 0$ at any point of $\theta \in [\theta_b, \theta_e]$, it can be rearranged to

$$\ddot{\theta} + \frac{\beta(\theta)}{\alpha(\theta)}\dot{\theta}^2 + \frac{\gamma(\theta)}{\alpha(\theta)} = 0. \tag{B.2}$$

Now, defining $Y = \dot{\theta}^2$, the 2nd-order equation (B.2) can be converted to a 1st-order ODE using following relation:

$$\ddot{\theta} = \frac{d}{dt}\dot{\theta} = \frac{d}{d\theta}(\dot{\theta})\frac{d}{dt}\theta = \frac{d}{d\theta}(\dot{\theta})\dot{\theta} = \frac{d}{d\theta}\left(\frac{1}{2}\dot{\theta}^2\right) = \frac{1}{2}\frac{dY}{d\theta}$$

Substituting this relation to (B.2) now gives the nonhomogeneous ordinary differential equation

$$\frac{dY}{d\theta} + 2\frac{\beta(\theta)}{\alpha(\theta)}Y + 2\frac{\gamma(\theta)}{\alpha(\theta} = 0 \tag{B.3}$$

Simplified for further calculation, it can be written

$$\frac{dY}{d\theta} + p(\theta)Y = q(\theta) \tag{B.4}$$

where

$$p(\theta) = 2\frac{\beta(\theta)}{\alpha(\theta)} \quad \text{and} \quad q(\theta) = -2\frac{\gamma(\theta)}{\alpha(\theta)}$$

Now, solving for the homogeneous solution Y_h for (B.4).

$$\begin{aligned} \frac{dY_h}{d\theta} + p(\theta)Y_h &= 0 \\ \frac{1}{Y_h}\frac{dY_h}{d\theta} &= -p(\theta) \\ ln(Y_h) - ln(Y_0) &= -\int_{\theta_0}^{\theta} p(\tau)d\tau \\ Y_h &= \exp\left\{-\int_{\theta_0}^{\theta} p(\tau)d\tau\right\} Y_0 \end{aligned} \tag{B.5}$$

The particular solution Y_p can then be derived by using equation (B.5) with a variable Y_0, i.e.

$$Y_p = \exp\left\{-\int_{\theta_0}^{\theta} p(\tau)d\tau\right\} z(\theta) \tag{B.6}$$

Differentiating with respect to θ gives

$$\frac{dY_p}{d\theta} = -p(\theta)\text{exp}\left\{-\int_{\theta_0}^{\theta} p(\tau)d\tau\right\} z(\theta) + \exp\left\{-\int_{\theta_0}^{\theta} p(\tau)d\tau\right\} \frac{d}{d\theta}z(\theta) \tag{B.7}$$

Substituting equation (B.6) and (B.7) to (B.4) then gives

$$\begin{aligned} \frac{d}{d\theta}z(\theta) &= \exp\left\{\int_{\theta_0}^{\theta} p(\tau)d\tau\right\} q(\theta) \\ z(\theta) &= \int_{\theta_0}^{\theta} \exp\left\{\int_{\theta_0}^{s} p(\tau)d\tau\right\} q(s)ds \end{aligned} \tag{B.8}$$

Now, last part of equation (B.8) can be substituted back to (B.6) for finding the particular solution:

$$\begin{aligned} Y_p &= \exp\left\{-\int_{\theta_0}^{\theta} p(\tau)d\tau\right\} \int_{\theta_0}^{\theta} \exp\left\{\int_{\theta_0}^{s} p(\tau)d\tau\right\} q(s)ds \\ Y_p &= \int_{\theta_0}^{\theta} \exp\left\{-\int_{s}^{\theta} p(\tau)d\tau\right\} q(s)ds \end{aligned} \tag{B.9}$$

The general solution equation (B.4) can then be found by the sum of the homogeneous solution (B.5) and particular solution (B.9):

$$\begin{aligned}
Y &= Y_h + Y_p \\
Y &= \exp\left\{-\int_{\theta_0}^{\theta} p(\tau)d\tau\right\} Y_0 + \int_{\theta_0}^{\theta} \exp\left\{-\int_{s}^{\theta} p(\tau)d\tau\right\} q(s)ds \\
Y &= \exp\left\{-2\int_{\theta_0}^{\theta} \frac{\beta(\tau)}{\alpha(\tau)}d\tau\right\} Y_0 - \int_{\theta_0}^{\theta} \exp\left\{-2\int_{s}^{\theta} \frac{\beta(\tau)}{\alpha(\tau)}d\tau\right\} 2\frac{\gamma(s)}{\alpha(s)}ds
\end{aligned} \tag{B.10}$$

Consequently, re-substituting back for $Y = \dot{\theta}^2$ and $Y_0 = \dot{\theta}^2$ gives

$$\dot{\theta}^2(t) = \exp\left\{-2\int_{\theta_0}^{\theta(t)} \frac{\beta(\tau)}{\alpha(\tau)}d\tau\right\} \dot{\theta}_0^2 - \int_{\theta_0}^{\theta(t)} \exp\left\{-2\int_{s}^{\theta(t)} \frac{\beta(\tau)}{\alpha(\tau)}d\tau\right\} 2\frac{\gamma(s)}{\alpha(s)}ds \tag{B.11}$$

Appendix C

Digital appendix

C.1 Datasheets

In the ”Datasheet”-folder, instruction manual for gripper, operating instructions and software instruction for the KUKA lightweight robot 4+ can be found.

C.2 CAD-files

In the ”CAD”-folder, CAD-models for gripper and two different models for the KUKA lightweight robot 4+ can be found.

C.3 Maple script’s

In the ”Maple”-folder”, Maple code for the kinematics and dynamics, calculation of the alpha,beta and gamma equations for all equations of motion, calculation of phi and conversion to Matlab of these functions can be found.

C.4 MATLAB script’s

In the ”Matlab”-folder, scripts for running optimization procedure, simulation and animation can be found, as well as results from optimization and experiments.

C.5 Video files

In the ”Video”-folder, a video of pitch one and two can be found.

Bibliography

[1] T. Senoo, A. Namiki and M. Ishikawa. *High-Speed Throwing Motion Based on Kinetic Chain Approach.* In Proc. of the 2008 IEEE International Conference of Intelligent Robots and Systems, Nice, France, September 2008.

[2] E.L. Yedeg and E. Wadbro *State Constrained Optimal Control of a Ball Pitching Robot* Mechanism and Machine Theory 69, 337-349, 2013.

[3] U. Mettin and A.S. Shiriaev. *Ball-Pitching Challenge with an Underactuated Two-Link Robot Arm.* 18th IFAC World Congress vol. 18, 2011.

[4] M.W. Spong, S. Hutchinson and M. Vidyasagar. *Robot Modeling and Control.* John Wiley & Sons, 2006.

[5] U. Mettin. *Principles for Planning and Analyzing Motions of Underactuated Mechanical Systems and Redundant Manipulators.* Department of Applied Physics and Electronics, Umeå University, Doctoral thesis, 2009.

[6] U. Mettin, P.X. La Hera, L.B. Freidovich and A.S. Shiriaev. *Parallel Elastic Actuators as Control Tool for Preplanned Trajectories of Underactuated Mechanical Systems.* The International Journal of Robotics Research, 29(9), 1186–1198, 2010.

[7] A.S. Shiriaev, J.W. Perram and C. Canudas-de-Wit. *Constructive Tool for Orbital Stabilization of Underactuated Nonlinear Systems: Virtual Constaints Approach.* IEEE Transactions on Automatic Control, vol. 50 no. 8, 1164-1176, August 2005.

[8] A.S. Shiriaev, A. Robertson, J.W. Perram and A. Sandberg. *Periodic Motion Planning for Virtually Constrained Euler-Lagrange Systems.* System and Control Letters, vol. 55, 900-907, 2006.

[9] U. Mettin, A.S. Shiriaev, L.B. Freidovich and M. Sampei. *Optimal Ball Pitching with an Underactuated Model of a Human Arm.* In proc. 2010 IEEE International Conference on Robotics and Automation, Anchorage, USA, May 2010.

[10] V.I. Arnold. *Mathmatical Methods of Classical Mechanics(Graduate Texts in Mathematics).* Springer, New York, 2nd edition, 1989.

[11] Rida T. Farouki. *The Bernstein Polynomial Basis: A Centennial Retrospective* Department of Mechanical and Aerospace Engineering, University of California, Davis, CA 95616, March 3, 2012.

[12] N.P. Linthorne and D.J. Everett *Release Angle for Attaining Maximum Distance in the Soccer Throw-in* ....

[13] N.P. Linthorne *Optimum Release Angle in the Shot Put* Journal of Sports Science, vol. 19, 359-372, February 4, 2001.

[14] http://www.mathworks.se/help/optim/ug/constrained-nonlinear-optimization-algorithms.html