Declarative Models for Self-Calibrating Robots

Johannes Olsson
Pontus Sikström

LUND UNIVERSITY
Department of Automatic Control
Abstract

The use of industrial robots in manufacturing often requires good calibration approaches and tools to obtain and maintain the desirable accuracy of the robot, for the robot to be able to execute a task in a satisfactory manner. For high-accuracy methods of calibration, it is necessary to first identify the robot’s stiffness parameters. This thesis will focus on a method that is based on a clamping procedure. In this procedure, the robot’s end effector is clamped to a fixed object and load is applied by the motors of the joints while measurements of torque and position of the motors are performed by the robot’s own sensors. These measurements are then used to identify the stiffness parameters of the robot. The purpose of this thesis is to develop a model library written in the modeling language Modelica to be used for robot simulations, and especially to simulate these clamping experiments. In the modeling, focus will lie on dynamical behaviour like compliance, backlash, and friction and how these effect the results of the clamping experiments.

The modularity aspect has been a central part in the progress and the model library developed is component based, i.e., the parts can be connected in various ways to model different types of robots. For model validation, many simulations have been performed with different dynamic model parameters to evaluate the results and how they compare to the expected dynamic behaviour. Also, experimental data from experiments with ABB’s robot IRB140 have been used to judge how well the models can reproduce results from real experiments.

The simulation results indicate that the models capture much of the dynamical behaviour in a satisfying way. The effects of joint compliance, friction, backlash, compliance in the clamping device, and variations in the clamping configuration are all of importance for the results of the clamping simulations. When comparing to data provided from clamping experiments of the IRB140 robot, the results of the simulations show that the models are able to reproduce much of the robot’s behaviour observed in the experiments.

Keywords: Robotics, Robot Modeling, Robot Calibration, Self-Calibrating Robots, Clamping, Declarative Modeling, Modelica.
Acknowledgments

This thesis was written during the fall of 2016 at the Department of Automatic Control at Lund University, in cooperation with Cognibotics AB, Lund.

We are very grateful to our supervisor Björn Olofsson for his exceptional support during the work with this thesis. Our assisting supervisors at Cognibotics AB, Andreas Stolt and Olof Sörnmo, are acknowledged for their valuable suggestions and inputs. Also, we would like to thank Klas Nilsson and Cognibotics AB for giving us the opportunity to write this thesis.

We would like to thank our wonderful friends and families for their great support throughout our studies at Lund University. A special thanks to Rebecca Tencic, William Lindeberg, Gustav Ek, Jonas Eliasson, Niklas Hansson, Emma Sjöborg, and Filip Thor for their invaluable friendship and for motivating us to always perform at our best. Niklas and Filip have also been great lunch partners at the university and have inspired us during the work with this thesis.

Lund, February 2017
Johannes Olsson and Pontus Sikström
# Contents

## 1. Introduction
1.1 Background .......................................................... 9
1.2 Problem Formulation .............................................. 9
1.3 Subject Foundation ................................................. 10
1.4 Delimitations ........................................................ 10

## 2. Theory
2.1 Manipulator Basics ................................................. 12
2.2 Wrists ................................................................. 13
2.3 Clamping Experiments ............................................. 15
2.4 Newton-Euler Formulation ....................................... 16
2.5 Elasticity Models ................................................... 18
2.6 Friction .............................................................. 18
2.7 Backlash .............................................................. 19
2.8 Controller ............................................................ 20
2.9 Modelica Language .................................................. 21

## 3. Modelling
3.1 Overview ............................................................. 25
3.2 Frame (connector) .................................................. 26
3.3 Inertial Frame ....................................................... 27
3.4 Flange (connector) ................................................ 27
3.5 Link ................................................................. 27
3.6 Joint ................................................................. 27
3.7 Revolute Joint ...................................................... 28
3.8 Revolute Axis ...................................................... 29
3.9 Clamping Device ................................................... 32
3.10 Controller .......................................................... 32
3.11 Path Planning ...................................................... 33
3.12 Wrist .............................................................. 35

## 4. Results
4.1 Dymola and JModelica.org ...................................... 36
### Contents

4.2 ABB IRB140 Experiments .................................................. 37
4.3 Simulation Setup .............................................................. 39
4.4 Joint Compliance .............................................................. 43
4.5 Friction ........................................................................ 45
4.6 Backlash ....................................................................... 48
4.7 Orthogonal Compliance ...................................................... 50
4.8 Compliance in the Clamped End .......................................... 52
4.9 Clamping Configuration ..................................................... 53
4.10 Simulated Results Compared to Experimental Data ............... 55

5. Discussion and Conclusions ................................................. 58
5.1 Simulation Results .......................................................... 58
5.2 Friction ................................................................. 59
5.3 Link Elasticity ............................................................... 59
5.4 Joint Types ................................................................. 60
5.5 Future Work ............................................................... 60
5.6 Conclusions ............................................................... 60

Bibliography ................................................................. 62
A. Modelica Models ............................................................ 64
A.1 Frame ............................................................... 64
A.2 InertialFrame ............................................................ 64
A.3 Flange ............................................................... 65
A.4 Link ............................................................... 65
A.5 Joint ............................................................... 66
A.6 RevoluteJoint ........................................................... 67
A.7 RevoluteAxis ........................................................... 69
A.8 Motor ............................................................... 70
A.9 Gear ............................................................... 70
A.10 ControlBus .............................................................. 71
A.11 ClampingPoint ........................................................... 71
A.12 Controller .............................................................. 72
A.13 Wrist ............................................................... 73
A.14 ClampingPathPlanning ................................................... 74
A.15 getOrthVectors .......................................................... 75
1

Introduction

This chapter introduces the background and describes the purpose of this thesis. The delimitations of the work will also be set.

1.1 Background

The demand for industrial robots has increased drastically the past decade. Between 2010 and 2015, the average robot sales increase per year was at 16% worldwide and this increase is predicted to continue the coming years [World Robotics 2016]. With higher sales and more application scopes added for each year, the demand for high accuracy also increases. One advantage of high-accuracy robots is that they can be programmed to reach certain positions in a reference coordinate system with good accuracy without being manually taught the positions. This reduces the need for human assistance when training a robot to perform a task.

Reliable calibration methods and tools are important to achieve and maintain high precision and accuracy. There are many different such calibration methods and this thesis focuses on a method that is based on experiments performed while the robots end-effector is clamped to a fixed object. By giving angle references to the regulator of the joints, while the end effector is clamped to a fixed point in space, it is possible to identify compliance in the joints that are trying to move. The measurements can be done with the robot’s sensors alone, without any need for external equipment, this is what makes the robot self-calibrating. This is a major advantage of this type of calibration method compared to many other calibration methods which often need expensive external equipment, like advanced cameras to measure the positions and orientations of different parts of the robot.

1.2 Problem Formulation

To be able to make relevant simulations regarding the precision and accuracy of robots, good models are required. It is desirable to be able to experiment with identification algorithms on models, instead of on actual manipulators, to estimate elas-
ticity parameters and other relevant dynamics. A suitable model library can also be used to find models where simulation results correspond to experimental data. It is both cheaper and more convenient to use computer simulations as an alternative to experiments on actual manipulators. Additionally, robot models can be used to predict the movement of a robot that in a certain task is experiencing external forces and torques. This prediction can be important when working with computer tools such as CAM (Computer-Aided Manufacturing).

The overall purpose of this thesis is to develop a Modelica library used for modeling and simulating manipulators. The models should be able to simulate clamping experiments of the robots, with results close to the results of experiments with actual robots. Another important aspect is the modularity, i.e., the manipulator models should be component based and it should be easy to modify the structure and component parameters to simulate different types of robots. Good models can be a great aid in the work to determine which dynamical effects of the manipulator that give rise to what kind of effects in the experiment data. This could lead to a better understanding of the resulting data from clamping experiments.

1.3 Subject Foundation

The work of this thesis has been carried out in cooperation with the company Cognibotics. Cognibotics is a company based in Lund, Sweden that specializes in methods and services for determination of robot properties such as backlash, friction, and non-linear compliance and improving robot accuracy in applications [Cognibotics, 2017]. One of the techniques investigated by the company is based on the method described in Section 1.1, where the robot’s end-effector is clamped fixed to another stiff object. It is of interest to evaluate how dynamic models of the robot, defined in a declarative modeling language, can be of use both in the identification phase and the subsequent calibration and pose compensation phases. This question is the foundation of this thesis. A description of the methods used for determination of robot properties can be found in [Nilsson, 2012].

The importance of different dynamical phenomena in robot models has been researched at Linköping University in the Master Thesis [Niglis and Öberg, 2015] with many topics associated to the work in this thesis. The thesis from Linköping University discusses friction and elasticity in greater depth while the work of this thesis is more focused around the implementation of the models. Also, a different model language, Simulink [Simulink], is used in the Linköping thesis.

1.4 Delimitations

Robot modeling is a very complex subject and the work with a model library as the one developed in this thesis needs to be delimited in some aspects for the task to be accomplished within the time frame of the thesis.
1.4 Delimitations

The models will only support one kind of joints, the revolute joint. This is the most common joint type for industrial robots [Ross et al., 2011, Ch. 2]. Other types of joints that are common in robots are prismatic, spherical, and helical joints, but modeling of these is not covered in this thesis.

The kinematics and dynamics of parallel manipulators will not be covered either. This means that the model library only supports modeling of serial manipulators (robots with links connected in series).

Compliance is a dynamic behaviour of great importance when modelling robots. The compliance is typically caused by the components of the joints—the gearbox, for example—and by link elasticity. In this thesis, only joint compliance will be considered. Both compliance in the rotational direction of the revolute joint and in directions orthogonal to this are included. The links will be considered as stiff and thus, link elasticity will not be modeled.

The modelling language Modelica is used for all the modeling in this thesis. Other simulation tools, like Simulink [Simulink], could have been tried as well but Modelica was judged to be the preferred one. Modelica supports DAE-based (differential algebraic equation) modelling which means that both ODEs (ordinary differential equation) and algebraic states can be used. This is an advantage compared to simulation tools that only support ODEs, since it often demands more of the user when not being able to use algebraic states in the model equation set-up.

and a description of some of the advantages of Modelica can be found in Section 2.9.
2

Theory

This chapter provides basic information about robot manipulators in general and about the clamping experiments that this thesis will revolve around. Some theory about the dynamics that are used in the models will also be explained. Furthermore, an introduction to the modelling language Modelica, that is used to implement the models, is presented.

2.1 Manipulator Basics

The type of robots the models in this thesis describe are often referred to as manipulators and are built up by links and joints. There are many different types of joints, but in this thesis only revolute joints are concerned. A revolute joint is a one-degree-of-freedom kinematic pair which provides rotation around a single axis [Spong et al., 2005, p. 3-4], as illustrated in Figure 2.1.

![Figure 2.1](image)

**Figure 2.1** 2D representation of a revolute joint, where the arrow illustrates the rotation.

Links refer to the body parts between the joints (or, if the first or the last link, between a joint and a fixed/free end).

To be able to express the position and orientation of different parts of the robot, a set of coordinate frames is needed. Every coordinate frame is fixed to a certain part of the robot and if that part rotates or moves, the coordinate frame does so as
2.2 Wrists

The joints that control the orientation of the end-effector, located after the main joints of the manipulator, are usually referred to as the wrist of the manipulator. The joints in the wrist are most often of the revolute type and if their axes of rotation intersect at a common point in space they are referred to as a spherical wrist. Spherical wrists are becoming increasingly more popular [Spong et al., 2005]. While the first three joints of the manipulator are responsible for determining the position of the end effector, the wrist is what decides the orientation. There is often not enough room for all motors in the actual wrist, so they are many times located elsewhere on the manipulator, often closer to the base of the robot. In these cases, a mechanical train of gears are used to drive the wrist and the motors often result in motion in more than one of the joints in the wrist. With these configurations, a transmission matrix needs to be defined to describe the kinematic coupling of the wrist. This matrix is used to specify the relationships between the driving motors and the joints in the wrist by the relations

\[ q_{\text{arm}} = Gq_{\text{motor}} \]  
\[ \tau_{\text{motor}} = G^T \tau_{\text{arm}} \]
Figure 2.2  A sketch of a simple manipulator setup with three joints and three links.
where $q_{\text{arm}}$ and $q_{\text{motor}}$ are the three motor positions and three joint positions respectively, while $\tau_{\text{motor}}$ and $\tau_{\text{arm}}$ are the three motor torques and three joint torques, respectively. $G$ is the transmission matrix [Asada, 2005, Ch. 6].

## 2.3 Clamping Experiments

To accurately calibrate a robot, it is sometimes necessary to first identify the robot’s stiffness parameters, to manage to compensate for geometrical deviations that occurs because of gravity or process forces. This can be done in many different ways and most methods use external devices to apply loads to components of the robot and to measure positions. However, the clamping method concerned in this thesis does not need any external devices, since the loads are applied by the robot’s own actuators and all measurements are done by the robot’s own sensors. The stiffness parameters are estimated with the end-effector clamped to a fixed object.

![Figure 2.3 Setup for clamping [Nilsson, 2012]](image)

In the test setup in Figure 2.3, the end-effector 16 will be clamped to the fixed point 17. The measurements for identifying the stiffness parameters for one joint
are then acquired by using the motor of that joint’s axis to apply a load while the structure is clamped. All movements in the structure are then results of the elasticity of the components and not of joint movement. To get enough data for parameter estimation, the robot is clamped to a set of different fixed positions and measurements are done while applying loads to the different joints.

### 2.4 Newton-Euler Formulation

In the model library developed for this thesis, the Newton-Euler formulation [Spong et al., 2005, p. 215] is used to compute the force and torque affecting the links of the manipulator due to linear and angular motions. The Newton-Euler formulation is different from other common methods, like the Lagrangian formulation [Spong et al., 2005, p. 188], in the way that it uses a forward-backward recursion to analyze the force and the torque. This approach means that it treats one link at a time, to eventually be able to describe the manipulator as a whole, in contrary to the Lagrangian formulation where the states of all links are needed simultaneously to perform the analysis. In the Newton-Euler formulation, the equations are evaluated in a numeric way and the computational complexity in each step remains constant. This means that the computational complexity grows in a linear fashion with the number $N$ of joints and is computationally advantageous to other methods [Newton-Euler approach, 2016]. The Newton-Euler formulation also makes it easier to implement the models more genericly, since it is not as dependent of the manipulator setup as a whole to compute force and torque because all the computations are performed in each link separately. But of course, the forces and torques in one link depend on the forces and torques in neighbouring links. Before going into detail on the equations, some notations are introduced. Here frame $i$ is a coordinate frame that is rigidly attached to link $i$. Beginning link means the side closest to the base and end of the link is the side closest to the end-effector of the manipulator. The variables in the
2.4 Newton-Euler Formulation

formulation are as follows:

\[ f_i = \text{force exerted by link } i - 1, \text{ resolved in frame } i. \]

\[ \tau_i = \text{torque exerted by link } i - 1, \text{ resolved in frame } i. \]

\[ f_{i+1} = \text{force exerted by link } i + 1, \text{ resolved in frame } i. \]

\[ \tau_{i+1} = \text{torque exerted by link } i + 1, \text{ resolved in frame } i. \]

\[ m_i = \text{mass of link } i. \]

\[ a_{c,i} = \text{acceleration of the center of mass of link } i, \text{ resolved in frame } i. \]

\[ g_i = \text{acceleration due to gravity in frame } i. \]

\[ r_{i,ci} = \text{vector from beginning of link } i \text{ to the center of mass of link } i \text{ in frame } i. \]

\[ r_{i+1,ci} = \text{vector from end of link } i \text{ to the center of mass of link } i \text{ in frame } i. \]

\[ \omega_i = \text{angular velocity of frame } i \text{ with respect to the inertial frame resolved in frame } i. \]

\[ \alpha_i = \text{angular acceleration of frame } i \text{ with respect to the inertial frame resolved in frame } i. \]

\[ I_i = \text{inertia matrix of link } i \text{ about the center of mass of link } i \text{ in frame } i. \]

Let the number of links be \( N \) and the index \( i \) represent the \( i \):th link, from the base link \((i = 1)\) to the last link \((i = N)\). Since the free end is not subject to any forces or torques, set \( f_N^{N+1} = 0 \) and \( \tau_N^{N+1} = 0 \), then the following relations describe the force \( f_i \) and torque \( \tau_i \) in each link \( i \):

\[ f_i = f_{i+1}^{i} + m_i a_{c,i} - m_i g_i \quad (2.3) \]

\[ \tau_i = \tau_{i+1}^{i} - f_i \times r_{i,ci} + f_{i+1}^{i} \times r_{i+1,ci}^{i} + I_i \alpha_i + \omega_i \times (I_i \omega_i) \quad (2.4) \]

where \( i = N, N - 1, \ldots, 1 \). More details about the Newton-Euler formulation can be found on pages 215–222 in [Spong et al., 2005].

**The Forward Recursion** The backward recursion described in (2.3)-(2.4) is used to calculate force and torque in the manipulator. Now some new notation is introduced:

\[ \theta_i = \text{angle of frame } i \text{ with respect to the inertial frame, resolved in frame } i. \]

\[ \omega_{i-1}^{i} = \text{angular velocity of frame } i - 1 \text{ with respect to the inertial frame, resolved in frame } i. \]

\[ \alpha_{i-1}^{i} = \text{angular acceleration of frame } i - 1 \text{ with respect to the inertial frame, resolved in frame } i. \]

\[ y_{i-1}^{i} = \text{unit vector of frame } i - 1 \text{ expressed in frame } i. \]

\[ a_{c,i} = \text{acceleration of the end of the link}. \]
Before the backward recursion can be used to calculate the forces and torques, \( \omega_i \), \( \alpha_i \) and \( a_{c,i} \) need to be found. This is done by first setting \( \omega_0 = 0 \), \( \alpha_0 = 0 \), \( a_{e,0} = 0 \), and \( a_{c,0} = 0 \) since the base is not moving and then computing the following relations for \( i = 1, 2, ..., n \):

\[
\omega_i = \omega_{i-1} + y_{i-1} \dot{\theta}_i \quad (2.5)
\]

\[
\alpha_i = \alpha_{i-1} + y_{i-1} \ddot{\theta}_i + \omega_i \times y_{i-1} \dot{\theta}_i \quad (2.6)
\]

\[
a_{e,i} = a_{e,i-1} + \omega_i \times r_{i,i+1} + \omega_i \times (\omega_i \times r_{i,i+1}) \quad (2.7)
\]

\[
a_{c,i} = a_{c,i-1} + \omega_i \times r_{i,ci} + \omega_i \times (\omega_i \times r_{i,ci}) \quad (2.8)
\]

This part of the Newton-Euler formulation is known as the forward recursion. However, one big advantage of using Modelica for modelling is that these variables can be acquired directly without the need to solve the forward recursion equations and there is no need to perform any other kinematic calculations in the models. Chapter 6 in [Spong et al., 2005] provides more theory on the forward recursion.

### 2.5 Elasticity Models

Elasticity of a material can be described as its ability to resist a distorting force and to regain its original shape after the force is removed. In this thesis, elasticity is modeled both in the joint model and in the axis model, as described in Sections 3.7 and 3.8. The clamped point is also modeled with elasticity. The elasticity in the axis models is modeled with springs, with behavior according to Hooke’s law. Hooke’s law describes a linear relationship between deformation and force in the following way

\[
F = cx \quad (2.9)
\]

where \( F \) is the force exerted by the spring, \( c \) is the so called spring constant and \( x \) is the deformation [Khan Academy, 2017]. Hooke’s law can also be applied to rotational springs which is the type of spring that will be used in this thesis. For rotational springs the law describes a linear relationship between the torque and angular displacement

\[
\tau_{spring} = c\theta \quad (2.10)
\]

where \( \tau_{spring} \) is the torque exerted by the spring, \( c \) is the spring constant and \( \theta \) is the angular displacement.

### 2.6 Friction

Friction is the general force resisting movement of objects in contact. In this thesis, the friction torques in the bearings are defined by a function of angular velocity
\[ \tau_{\text{friction}} = f(w) \]. Here, \( \tau \) is the friction torque, \( w \) is the angular velocity and \( f \) is a function describing the relationship between the two. This function can be different for different parts of the robot. For manipulator applications, a phenomenon which is sometimes referred to as the stiction effect [Lewotsky, 2014] is often observed, where the friction torque is high when the velocity is zero but decreases when the static friction is overcome, see Figure 2.4.

![Graph of \( \tau = f(\omega) \)](image)

**Figure 2.4** Friction torque as a function of angular velocity where the stiction effect can be observed.

## 2.7 Backlash

Backlash can be defined as the maximum distance or angle through which a part of a mechanical system can be moved or rotated without applying appreciable force to the next part of the system [Bagdad, 2008]. In a manipulator, backlash typically appears in the gearboxes. Backlash is necessary to be able to design gears that rotates smoothly [KHK Stock Gears, 2016]. Figure 2.5 illustrates a gear backlash as the difference in size between a tooth space on one gear and a tooth on another.

One effect of backlash in the gearboxes in a manipulator is that, when a motor change direction, there will be a short period of time when the teeth in the gears are not in contact and the transmission force is zero. The size of the backlashes in a manipulator will affect the accuracy of the robot, since the controllers are using measurements of the motor positions only, and it is therefore an important behaviour to consider in the models.
Chapter 2. Theory

2.8 Controller

To control the joint movement of the manipulator, the model library developed for this thesis provides cascade controllers with a structure visualized in Figure 2.6, where the inner loop controls the speed and the outer loop controls the position.

Since only revolute joints are used in the models, $\dot{q}$ and $q$ represent angular velocity and angle, respectively. The feedback used for the outer loop is the joint angle and the feedback for the inner loop is the joint velocity, both computed from measurements of the motor encoders.
2.9 Modelica Language

Modelica is a declarative, object oriented modelling language for complex systems [Modelica language specification, 2014]. In object oriented languages, code can easily be reused through inheritance and instantiation of models. Modelica is free and it contains components for many different types of systems, including mechanical and electrical systems. It supports ordinary differential equations and algebraic equations [Modelica language specification, 2014]. However, it has no support for partial differential equations, but for the purpose of this thesis, this will not be a problem.

Declarative Models

A declarative programming language is one that describes what needs to be computed rather than how to compute it [Declarative modelling, 2016]. This greatly simplifies the code writing. The Modelica language is equation based and code is built up by declarations, equations, and connections. The order of the equations does not matter.

Examples

The ordinary differential equation

\[ \dot{x} = x, \quad x(t_0) = 0 \]

can be simulated using Modelica with the following model:

```model Basic1
    Real x(start = 0);
    equation
        x = der(x);
end Basic1;
```

Here, the real variable \( x \) is declared with a starting value. Note that line four is not a declaration but rather a definition of relations, meaning that it can be written in a lot of different ways. The following model accomplishes exactly the same thing:

```model Basic2
    Real x(start = 0);
    equation
        0 = x - der(x);
end Basic2;
```

Models can be as complex or as simple as the task requires. Models can contain instances of other models.

Connectors  

Another key type of models that will be used in this thesis is connectors. Connectors are instances of the connector class. A connector is used to connect
Chapter 2. Theory

component variables in a desired way. A connector used to describe an electric pin with voltage, \( v \), and current, \( i \), can look like this:

```modelica
cconnector Pin
Voltage v;
flow Current i;
end Pin;
```

Connectors may contain two kinds of variables, flow variables and non-flow variables [Modelica language specification, 2014, p. 99-104]. Flow variables are variables that sum to zero within the connector. In this case the current is marked as a flow variable for the Kirchoff’s law to be obeyed. Flow variables are defined to have a positive sign when the flow is going into the connector. Non-flow variables are variables that are the same on both sides of the connector, in this example voltage is a non-flow variable. Illustrated in Figure 2.7 is an example of two connected pins where the connect equation has been used to connect the two pins. Listing 2.1 shows two ways of connecting the pins.

![Pin example](image)

**Figure 2.7** Pin example.

```modelica
equation // Manually defining relationship between variables.
   pin1.v = pin2.v;
   pin1.i + pin2.i = 0;
```

```modelica
equation // Using the connect command to define relationship.
   connect (pin1, pin2);
```

**Listing 2.1** How the connect command can simplify the models.

Using the connect command in an equation section greatly simplifies writing the code.

**Blocks** Another type of model that will be used is blocks. They are input/output blocks, mainly used to build block diagrams. Blocks are handy whenever a component is needed that uses input or produces output. Controllers can be built easily with blocks, as demonstrated in the source code in Appendix A.12, which imple-
ments the control structure in Figure 2.6. Below is an example of a basic block that takes an input signal and outputs the product of a constant and the input value.

```modelica
block Gain
    parameter Real k(start=1);
    Interfaces.RealInput u;
    Interfaces.RealOutput y;
equation
    y = k*u;
end Gain;
```


3

Modelling

The library developed for this thesis, used to create manipulator models, consists of connectors, links, joints, and many other submodels. This chapter will review all the submodels one by one. These components are shown in Table 3.1

In some of the library’s submodels, components and functions from the Modelica standard library are used. The functions are from the package Modelica.Mechanics.MultiBody.Frames which is documented in [Modelica documentation, 2011] and the models used are:


which are documented in [Modelica documentation, 2011].

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame</td>
<td>Connector used to connect links and joints.</td>
</tr>
<tr>
<td>Inertial Frame</td>
<td>Model of a reference coordinate frame.</td>
</tr>
<tr>
<td>Flange</td>
<td>Connector used to connect 1D rotational components.</td>
</tr>
<tr>
<td>Link</td>
<td>Link model.</td>
</tr>
<tr>
<td>RevoluteJoint</td>
<td>Joint model.</td>
</tr>
<tr>
<td>ClampingPoint</td>
<td>Model of the fixed point for clamping.</td>
</tr>
<tr>
<td>RevoluteAxis</td>
<td>Model of the driving axis of a joint.</td>
</tr>
<tr>
<td>Motor</td>
<td>Motor model.</td>
</tr>
<tr>
<td>Gear</td>
<td>Gear model.</td>
</tr>
<tr>
<td>Controller</td>
<td>Model of a motor controller.</td>
</tr>
<tr>
<td>ControlBus</td>
<td>Model of a control bus used by the controller.</td>
</tr>
<tr>
<td>ClampingPathPlanning</td>
<td>Model of path and trajectory planning for clamping experiments.</td>
</tr>
</tbody>
</table>
3.1 Overview

As described in Section 2.1, coordinate frames attached to each link of the robot are used to describe the robot’s kinematics. Due to the nature of the Modelica language and its use of connectors, our models have one frame (connector) at each end of both the link models and the joint models. Figure 3.1 illustrates a visualization of an example model setup with the connection points between the frames in the links, joints, and the surrounding environment.

![Figure 3.1 Example of manipulator structure with connection points.](image)

Connections 1a, 2a, and 3a in Figure 3.1 are the connections between the end of a link and the beginning of next joint. Connections 1b, 2b, and 3b are the connections between the end of a joint and the beginning of next link. Connection 0 is the connection between the beginning of the first link of the manipulator and the ground (inertial frame). Connection 4 is the connection between the end of the last link and the clamping device. Each joint has an axis with subcomponents to drive it and to model the dynamics. The structure of an axis is visualized in Figure 3.2.
Figure 3.2 Structure of an axis model (real world robots don’t usually include a speed sensor but instead derives the angle).

The motor model is used to pass a torque input to the inertia model, which computes the angular acceleration of the motor caused by the torque input. To model the dynamics of the joint, submodels of friction, backlash, and elasticity (elastoBacklash models both backlash and elasticity) are used. A gear model is connected at the end of the axis to translate motor rotation into joint rotation. A speed sensor and an angle sensor measure the angular velocity and position of the motor flange and pass the values to a control bus. The control bus is a connector that can be connected to a regulator to pass variables between the joint and the regulator. All of the axis models are described in more detail in Section 3.8.

### 3.2 Frame (connector)

Some points of the manipulator need to be connected by frames that describe the position and orientation of a specific section of the manipulator and update said positions and orientations according to the kinematics and dynamics of the system. These points are essentially:

- The fixed points, i.e., the inertial frame and the fixed point representing the clamped end.
- The beginning and end of each link.
- The joints.

Each joint actually has two frames with equal positions. One frame that is equal, both in position and orientation, to the previous frame (most often the end of the previous link) and one that is connected to the next link and is able to rotate around
3.3 Inertial Frame

The inertial frame represents the point where the first link of the manipulator, the base, is connected to the ground. This is the reference frame for expressing most of the length vectors and positions for setting up the overall manipulator model and clamping point. This model consists of an instance of the connector Frame, which has fixed position and fixed rotation. For source code, see Appendix A.2.

3.4 Flange (connector)

The flange is used to connect components with one degree of freedom. The model holds two variables, one for angle (rotational) and one for torque. This connector is used to link the axis components, which only rotate with one degree of freedom, with respect to each other. For source code, see Appendix A.3.

3.5 Link

The link model consists of two frames, representing the end points of the link, separated by a length vector. In order for Modelica to properly simulate the links, carefully specified starting values for the velocity, angular velocity, and position are required. The backwards part of the Newton-Euler equations specified in Section 2.4 is used to model the link’s force and torque. However, the forward recursion is not needed as the frame model contains information about the frame’s orientation and position. Finding \( \omega_i, \alpha_i, \) and \( a_{c,i} \) is easy using the features of the Modelica language by specifying the correct relationships between derivatives of variables. Using the Newton-Euler equations for each link, it is possible to compute the cut force and torque, which is then transmitted using the frames.

The frame that is referred to as frame \( i \) in the equations from Section 2.4 is attached to the side of the link that is closest to the base of the manipulator [Spong et al., 2005, p. 219]. In our model, the calculations are therefore performed with respect to the first frame of the link, which is named `frame_beg` in the source code in Appendix A.4. The link model takes the parameters specified in Table 3.2.

3.6 Joint

The joint model comprises a revolute joint and a revolute axis model and connects them to each other. These models are described in the following two sections.
Table 3.2 Parameters for the link model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l$</td>
<td>m</td>
<td>length vector</td>
</tr>
<tr>
<td>$cm$</td>
<td>m</td>
<td>vector from the first frame to the center of mass</td>
</tr>
<tr>
<td>width</td>
<td>m</td>
<td>width (and height) of the link, for animation</td>
</tr>
<tr>
<td>$I$</td>
<td>kgm$^2$</td>
<td>inertia matrix</td>
</tr>
<tr>
<td>$m$</td>
<td>kg</td>
<td>mass</td>
</tr>
</tbody>
</table>

Parameters for the revolute joint model and for the revolute axis model are passed directly to this model. For parameter specification, see the sections for revolute joint and revolute axis models, respectively.

3.7 Revolute Joint

The joint is modeled with two frames of identical position but different rotation. The rotation is computed using the equations of motion. The elasticity of the joint is separated into two parts. First, compliance in the axis of rotation is considered. This is not modeled directly in the joint model, but instead as a part of the axis model described in Section 3.8. Secondly, compliance in the plane orthogonal to the rotational axis is considered. This is modeled directly in the joint, with two springs and two dampers connected in parallel. Figure 3.3 shows a joint and a coordinate system where the joint rotational axis is marked, as well as the orthogonal axes. The rotation in the orthogonal axes would be fixed if the link was perfectly stiff but since the model is not stiff, small positional deviations can occur in these axes because of the orthogonal compliance.

Figure 3.3 The red arrows illustrates the orthogonal axes where any rotation is a result of orthogonal compliance in the joint.

The joint has a connector (flange) to which a driving axis (revolute axis) is connected. The revolute joint model takes the parameters specified in Table 3.3.
Table 3.3 Parameters for the revolute joint model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>—</td>
<td>rotational axis</td>
</tr>
<tr>
<td>$c_{orth}$</td>
<td>Nm/rad</td>
<td>rotational spring constant for orthogonal compliance</td>
</tr>
<tr>
<td>$d_{orth}$</td>
<td>Nms/rad</td>
<td>rotational damping constant for orthogonal compliance</td>
</tr>
</tbody>
</table>

3.8 Revolute Axis

The structure of the axis is described in Figure 3.2. This section reviews the parts — i.e., motor, inertia, friction, elasticity, backlash, gear, sensors, and control bus — in more detail. The revolute axis model connects to a revolute joint via a connector (flange).

Motor

The motor model is very simple. It takes a torque input to set the torque of the motor flange, which is connected to the next component of the axis. Thus, no detailed modelling of the motor dynamics is included.

Inertia

The inertia model used in the axis is from Modelica’s Mechanics library, Modelica.Mechanics.Rotational.Components.Inertia [Modelica documentation, 2011]. Given a parameter with the moment of inertia value, it uses the torque from the motor to compute a resulting angular acceleration.

Friction

In the revolute axis, the friction model is from Modelica’s Mechanics library, Modelica.Mechanics.rotational.Components.BearingFriction [Modelica documentation, 2011]. It takes a table of values for the friction torque, coupled to different rotational velocities, and makes a linear interpolation of the values, which are then used for the friction modelling. The user is able to provide a value for the peak friction experienced when in stationarity. The friction torque is defined as:

\[
\begin{align*}
\tau_{\text{friction}} &= k \omega - m, & \text{if } \omega < 0 \\
\tau_{\text{friction}} &= k \omega + m, & \text{if } \omega > 0 \\
|\tau_{\text{friction}}| &\leq p \cdot m, & \text{if } w = 0
\end{align*}
\]  

where $\tau_{\text{friction}}$ is the friction torque, $\omega$ is the angular velocity of the axis, $p$ is the peak friction value, and $k$ and $m$ are constants. A graph corresponding to this definition can be seen in Figure 3.4, compare this to Figure 2.4.
Chapter 3. Modelling

Elasticity and Backlash

The model for elasticity and backlash is from Modelica’s Mechanics library, Modelica.Mechanics.Rotational.Components.ElastoBacklash [Modelica documentation, 2011]. This model takes a spring constant and a damping constant as parameters as well as an angle for the backlash and then uses Hooke’s law to describe elasticity. The spring and the damper are connected in parallel and are then connected to the backlash in series, as illustrated in Figure 3.5.

Figure 3.4 Friction torque in the axis.

![Graph showing friction torque against angular velocity](image)

Figure 3.5 Elasticity and backlash in the axis.

![Diagram of elasticity and backlash model](image)
Table 3.4  The variables of control bus.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>motorVelocity</td>
<td>Measurement input to the controller.</td>
</tr>
<tr>
<td>motorAngle</td>
<td>Measurement input to the controller.</td>
</tr>
<tr>
<td>angle_ref</td>
<td>Reference to the controller.</td>
</tr>
<tr>
<td>velocity_ref</td>
<td>Reference to the controller.</td>
</tr>
<tr>
<td>motorTorque</td>
<td>Signal from the controller.</td>
</tr>
</tbody>
</table>

**Gear**

This model describes a gear with ratio $g$ and consists of two flanges with the following relationships:

\[
\tau_{arm} = g \tau_{motor} \quad (3.2)
\]

\[
g \phi_{arm} = \phi_{motor}. \quad (3.3)
\]

The dynamics of the gearbox, like backlash and elasticity, are not modeled here but are connected in series, as illustrated in Figure 3.2. The gear model consists of two flanges where the first flange is connected to the chain of axis components and the second flange is connected to the axis flange. The axis flange is the connector that then will be connected to the joint flange, as described in Section 3.7.

**Sensors and Control Bus**

Every instance of an axis has a control bus, an angle sensor, and a speed sensor. The control bus is a connector that holds all the variables used by the controller. These are described in Table 3.4. For the control bus source code, see Appendix A.10.

The two control bus variables motor angle and motor velocity are used as reference signals to the controller. They get their values from the angle sensor and the speed sensor in the axis, which are connected to the motor flange. The angle sensor and the speed sensor are from Modelica’s Sensor library, Modelica.Mechanics.Rotational.Sensors [Modelica documentation, 2011].

**Parameters**

The revolute axis model takes the parameters defined in Table 3.5.

**Friction Table**  The friction table is an $N \times 2$ matrix where angular velocity is entered in the left column and the corresponding friction torque is entered in the second column. The number of rows, $N$, can be any integer $> 1$ depending on how much data that are provided by the user. This matrix is then used in the friction model to define a function—a first-order polynomial—that describes the relation between the angular velocity and the friction torque. A linear interpolation is performed and a first-order polynomial is fitted to the input data. As an example, the
Table 3.5 Parameters for the revolute axis model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>Nm/rad</td>
<td>rotational spring constant</td>
</tr>
<tr>
<td>$d$</td>
<td>Nms/rad</td>
<td>rotational damping constant</td>
</tr>
<tr>
<td>$b$</td>
<td>rad</td>
<td>backlash</td>
</tr>
<tr>
<td>gearRatio</td>
<td>—</td>
<td>gear ratio</td>
</tr>
<tr>
<td>inertia</td>
<td>kgm$^2$</td>
<td>inertia (one dimension)</td>
</tr>
<tr>
<td>tau_pos</td>
<td>—</td>
<td>friction table</td>
</tr>
<tr>
<td>friction_peak</td>
<td>—</td>
<td>friction peak</td>
</tr>
<tr>
<td>breaksOn</td>
<td>—</td>
<td>(boolean) true if the axis should be fixed</td>
</tr>
</tbody>
</table>

friction table matrix

\[
\begin{pmatrix}
1 & 3 \\
2 & 5 \\
\end{pmatrix}
\]

represents a friction torque value of 3 Nm at angular velocity 1 rad/s and 5 Nm at angular velocity 2 rad/s, respectively. The first-order polynomial corresponding to this table is $\tau_{\text{friction}} = 2\omega + 1$, for $\omega \geq 0$.

### 3.9 Clamping Device

The clamping device is modeled as a stiff spring that is connected to the end of the manipulator. Every deviation of the manipulator end effector, in position or orientation, from the clamped point is responded to with a large opposing force proportional to the deviation. This model, with compliance, was chosen to capture the imperfections of the clamping of the manipulator. The spring and damper constants in this model are matrices so it is possible to change the compliance in a single direction if desired. There are two types of elasticity and damping represented by the matrices; both translational and rotational. So the stiffness parameters can be set in each direction and around each axis. The clamping point model takes the parameters specified in Table 3.6.

### 3.10 Controller

Modelica blocks, as described in Section 2.9, are used to represent the different parts of the controller, which are illustrated in the block diagram in Figure 2.6. The blocks are then connected to each other according to the control diagram, and the reference, feedback, and output blocks are connected to the axis control bus of the joint that should be controlled.
Table 3.6 Parameters for the clamping device.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{\text{lin}}$</td>
<td>N/m</td>
<td>spring constant for clamping point compliance</td>
</tr>
<tr>
<td>$d_{\text{lin}}$</td>
<td>Ns/m</td>
<td>damping constant for clamping point compliance</td>
</tr>
<tr>
<td>$c_{\text{rot}}$</td>
<td>Nm.rad</td>
<td>spring constant for clamping point compliance</td>
</tr>
<tr>
<td>$d_{\text{rot}}$</td>
<td>Nms.rad</td>
<td>damping constant for clamping point compliance</td>
</tr>
</tbody>
</table>

Table 3.7 Parameters for the controller.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_p$</td>
<td>–</td>
<td>gain of position controller</td>
</tr>
<tr>
<td>$k_s$</td>
<td>–</td>
<td>gain of speed controller</td>
</tr>
<tr>
<td>$T_s$</td>
<td>s</td>
<td>time constant of integrator of speed controller</td>
</tr>
<tr>
<td>ratio</td>
<td>–</td>
<td>gear ratio of the joint that is controlled</td>
</tr>
</tbody>
</table>

This controller uses feedback from the motor side, i.e., motor angle and motor velocity instead of joint angle and joint velocity. However, the reference signals are for the arm side, i.e., joint angle and joint velocity. Consequently, the reference signals are multiplied with the gear ratio of the gear box in the joint, to get the desired behavior. The reason for using measurements from the motor side is that most industrial robots have sensors on the motor side and not on the arm side.

The controller model takes the parameters specified in Table 3.7. For source code, see Appendix A.12.

3.11 Path Planning

To be able to simulate clamping experiments, suitable reference signals for the controller are needed. The path planning model handles this by using the Modelica block `Modelica.Blocks.Sources.KinematicPTP2` [Modelica documentation, 2011]. This block generates output signals for moving as fast as possible from a start position to an end position, given kinematic constraints. The constraints to be passed are maximum velocity and maximum acceleration of the joint. The outputs are angle reference, velocity reference, and acceleration reference.

By combining five KinematicPTP2 blocks, the path planning model outputs reference signals for moving back and forth, to a given maximum/minimum angle. Figures 3.6 and 3.7 show examples of two angle reference signals and velocity reference signals plotted over 15 seconds. The parameters used in the example are shown in Table 3.8.
Table 3.8  Parameters used in Figures 3.6 and 3.7.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum/minimum angle</td>
<td>$\pi/20$ ($\approx 0.157$ rad)</td>
</tr>
<tr>
<td>Maximum velocity</td>
<td>0.2 rad/s</td>
</tr>
<tr>
<td>Maximum acceleration</td>
<td>0.5 rad/s²</td>
</tr>
</tbody>
</table>

Figure 3.6  Example of angle reference plotted against time.

Figure 3.7  Example of velocity reference plotted against time.

For source code, see Appendix A.14.
3.12 Wrist

The wrist model has three flange connectors for external driving axes. These flanges drive the wrist according to the transmission matrix, as described in Section 2.2, that is given as a parameter when initializing the wrist. The compliance of the wrist is not modeled here but is taken care of by the external axes connected to the flanges. The wrist model only takes the parameter \textit{transmission}, describing the variable $G$ in Equations (2.1)-(2.2). This model of the wrist does not include any orthogonal compliance, for more on wrist modelling, see [Niglis and Öberg, 2015].
4

Results

This chapter provides results of simulations performed with manipulator models built with the model library discussed in the previous chapter. Focus lies on model validation and the effects of different dynamic behavior like friction, backlash, and joint compliance on the resulting joint/axis dynamics. During the simulations, results from all six joints of the robot have been considered. However, for readability reasons, all clamping curves are not presented here but results have been singled out to those who demonstrate the dynamic effects in the most visible way for the respective case.

4.1 Dymola and JModelica.org

Running the Modelica simulations can be done on many different platforms. This library has been tested in both Dymola [Dymola, 2017] and JModelica.org [Jmodelica.org, 2017] with similar results. An example of clamping curves from two identical model set-ups simulated in both Dymola and Jmodelica.org can be seen in Figure 4.1. In these simulations, JModelica.org is using the CVode solver [CVODE] and Dymola is using the Dassl solver [Petszold, 1982]. The results in this example are close to identical, and has been in all comparisons between the two platforms.

The results presented in this chapter all come from simulations performed in Dymola. The reason for using Dymola is mainly that it has a more accessible user interface and animation features, which makes it easier to evaluate simulation results.
4.2 ABB IRB140 Experiments

The results in this chapter are solely from simulations done with a model of ABB’s robot IRB140 [ABB IRB140 data, 2017]. The effects of parameters representing different dynamic behaviour have been examined and at the end of the chapter, simulation results will be compared to measured data from real clamping experiments with IRB140. Figure 4.2 shows a picture of the robot.
Figure 4.2  ABB IRB140 manipulator in the Robotics lab, shared by the Department of Automatic Control and Department of Computer Science at Lund University.

Figure 4.3 shows clamping curves from experimental data provided by Cognibotics. The character of these curves will be further examined and compared to simulated results in Section 4.10. For now, they only serve as an example of the characteristics of clamping curves.

The data visualized in the plots in Figure 4.3 come from experiments where the end-effector of the IRB140 was clamped to a fixed object and load is applied by the robots own motors, as described in Section 2.3.
4.3 Simulation Setup

The simulations in Sections 4.4-4.9 were performed using the clamping configuration shown in Figure 4.4 (full drawn lines). All joints have individual controllers and have been given a certain angle reference. The Modelica code for this simulation setup is shown in Listing 4.1. The six links, the three base joints, and the
Chapter 4. Results

wrist are all initiated individually with parameters defining the characteristics. For the links the parameters describe the dimensions and the weight and for the joints and wrist they define the elasticity, friction, gear ratio and inertia. After also initiating controllers, an inertial frame, path planners and a clamping device everything is connected in the equation section and ready for simulation.

```model IRB140Demo
    import C = Modelica.Constants;

    InertialFrame IF "Inertial frame";

    // Defining the links with length, width and mass.
    Link base(l={0,0.1,0},width=0.05,m=20);
    Link link1a(l={0.07,0,0},width=0.05,m=10);
    Link link1b(l={0,0.252,0},width=0.05,m=20);
    Link link2(l={0,0.360,0},width=0.05,m=20);
    Link link3(l={0.38,0,0},width=0.05,m=23);
    Link link4(l={0.065,0,0},width=0.05,m=5);

    // Defining all joints except for the wrist joints.
    Joint joint1(n={0,1,0},c=10,d=5,b=0.01,c_orth=10000000,tau_pos=
        [0,0.1;1,0.3],gearRatio=104.4,inertia=0.014);
    Joint joint2(n={0,0,1},c=20,d=2,b=0.01,c_orth=10000000,tau_pos=
        [0,0.1;1,0.3],gearRatio=104.4,inertia=0.014);
    Joint joint3(n={0,0,1},c=40,d=2,b=0.01,c_orth=10000000,tau_pos=
        [0,0.1;1,0.3],gearRatio=100.74,inertia=0.014);

    // Defining the wrist with transmission matrix.
    Wrist wrist(transmission=
        {{110.0/6649.0,0,0},{71.0/(22.0*6649.0/110.0*207675.0/3190.0),3190.0/207675.0,0},
        {(71.0*243360.0*22.0*207675.0/3190.0*64.0*22.0*4147.0)/(22.0*
            4147.0*22.0*6649.0/110.0*207675.0/3190.0*486720.0/7975.0),
        243360.0/(4147.0*207675.0/3190.0*486720.0/7975.0),
        7975.0/486720.0}});

    // Defining the axes that power the wrist.
    RevoluteAxis axis4(c=0.4,d=0.1,b=0.01,tau_pos=[0,0.1;1,0.3],
        gearRatio=1,inertia=0.014);
    RevoluteAxis axis5(c=2.5,d=0.5,b=0.01,tau_pos=[0,0.1;1,0.3],
        gearRatio=1,inertia=0.014);
    RevoluteAxis axis6(c=0.8,d=0.1,b=0.01,tau_pos=[0,0.1;1,0.3],
        gearRatio=1,inertia=0.014);

    // Defining the controller.
    Controller controller1(
        kp=0.008,
        ks=0.01,
        Ts=0.11,
        ratio = 104.4);
    Controller controller2(
        kp=0.008,
        ks=0.01,
```

4.3 Simulation Setup

Ts=0.12,
    ratio = 104.4);
Controller controller3(
    kp=0.008,
    ks=0.01,
    Ts=0.28,
    ratio = 100.74);
Controller controller4(
    kp=0.008,
    ks=0.002,
    Ts=0.15,
    ratio = 60.45);
Controller controller5(
    kp=0.008,
    ks=0.005,
    Ts=0.24,
    ratio = 65.1);
Controller controller6(
    kp=0.008,
    ks=0.001,
    Ts=0.12,
    ratio = 61.03);

// Defining a path for the controller.
ClampingPathPlanning path ( q_pos=C.pi/10, qd_max =0.4, qdd_max =0.8);
ClampingPathPlanning path2 ( q_pos=C.pi/6, qd_max =0.4, qdd_max =0.8);

// Defining the stiffness of the clamped end.
ClampingDevice cp ( c_rot=20000000*identity(3),
                c_lin=20000000*identity(3));

// Chain of connections from inertial frame through all links
// and joints all the way to the clamped end.
connect ( IF.frame_end , base.frame_beg);
connect ( base.frame_end , joint1.frame_beg);
connect ( joint1.frame_end , link1a.frame_beg);
connect ( link1a.frame_end , link1b.frame_beg);
connect ( link1b.frame_end , joint2.frame_beg);
connect ( joint2.frame_end , link2.frame_beg);
connect ( link2.frame_end , joint3.frame_beg);
connect ( joint3.frame_end , link3.frame_beg);
connect ( link3.frame_end , wrist.frame_beg);
connect ( wrist.frame_end , link4.frame_beg);
connect ( link4.frame_end , cp.frame);

// Connecting the controller to the control bus of the joint and
// to the angle reference and velocity reference
// of the path planner.
connect (controller1.controlBus.angle_ref , path.q);
connect (controller1.controlBus.velocity_ref , path.qd);
connect (controller1.controlBus , joint1.controlBus);
connect (controller2.controlBus.angle_ref , path.q);
connect (controller2.controlBus.velocity_ref, path.qd);
connect (controller2.controlBus, joint2.controlBus);
connect (controller3.controlBus.angle_ref, path.q);
connect (controller3.controlBus.velocity_ref, path.qd);
connect (controller3.controlBus, joint3.controlBus);
connect (controller4.controlBus.angle_ref, path2.q);
connect (controller4.controlBus.velocity_ref, path2.qd);
connect (controller4.controlBus, axis4.controlBus);
connect (controller5.controlBus.angle_ref, path2.q);
connect (controller5.controlBus.velocity_ref, path2.qd);
connect (controller5.controlBus, axis5.controlBus);
connect (controller6.controlBus.angle_ref, path2.q);
connect (controller6.controlBus.velocity_ref, path2.qd);
connect (controller6.controlBus, axis6.controlBus);

// Assigning the torque from the controller to the joint.
connect (joint1.torque, controller1.controlBus.motorTorque);
connect (joint2.torque, controller2.controlBus.motorTorque);
connect (joint3.torque, controller3.controlBus.motorTorque);
connect (axis4.torque, controller4.controlBus.motorTorque);
connect (axis5.torque, controller5.controlBus.motorTorque);
connect (axis6.torque, controller6.controlBus.motorTorque);

// Connecting the axes to the wrist.
connect (wrist.axis1, axis4.flange);
connect (wrist.axis2, axis5.flange);
connect (wrist.axis3, axis6.flange);

end IRB140Demo;

**Listing 4.1**  Modelica example code for an IRB140 setup.
4.4 Joint Compliance

Figure 4.4 Drawing of IRB140 with geometry specifications [ABB Robotics, 2017]. The positioning of the manipulator in this drawing is used in the following simulations.

4.4 Joint Compliance

First, the compliance along the direction of the rotational axis of the joint will be examined. Orthogonal compliance will be examined later in Section 4.7. The compliance in the axis is modeled by a rotational spring and a damper as described in Section 2.5. The relationship between the torque, angular displacement, and angular velocity is described by the equation

\[ \tau = c \theta + d \omega \]

where \( \tau \) is the torque, \( \theta \) is the angular displacement, and \( \omega \) is the angular velocity. In Figure 4.5, two different values for \( c \) are used, while the rest of the simulation settings are identical. No orthogonal compliance is modeled in these simulations and the compliance in the clamped end is small.
The two resulting clamping curves in Figure 4.5 demonstrate the effect of changing the spring constant. When doubling the value of the spring constant $c$, the slope of the clamping curve gets roughly twice as steep. The increase in steepness indicates that more torque is needed for the same rotation, which is an expected effect when having a stiffer spring. When performing a linear regression, the resulting lines are on the form $\tau = \beta_0 + \beta_1 \theta$ where the interesting part is $\beta_1$, representing the slope of the line. For the orange curve, $\beta_1 = 9.2389$ which is close to the expected value of 10. For the blue curve, $\beta_1 = 17.0773$ which differ a bit from the expected value of 20. However, it demonstrates that a doubling of the spring constant results in that the slope gets close to twice as steep, which is a good result.

Figure 4.6 shows the effects of varying the damping constant $d$. When increasing the damping constant, the angular velocity decreases, especially around the turning points at minimum/maximum motor angle. This too is expected behaviour, since a higher damping constant is supposed to slow down the motion. The speed of the motor of axis 2 in these simulations are between 0 and 0.4 rad/s and the most significant difference in speed of the two simulations occurs when the motor angle is in the range of 0.1 radians from the turning points.
4.5 Friction

The friction model used in the simulations describes the friction torque as a function of the angular velocity of the axis. This function is limited to be on the form of a polynomial of degree one as described in (3.1). Figures 4.7 and 4.8 show the difference in clamping curves when alternating the value for $k$ and $m$, respectively.

**Figure 4.6** Clamping curves from two simulations with different damping constants.
Chapter 4. Results

Figure 4.7  Clamping curves from two simulations with different values of $m$ in the friction equation (3.1).

Figure 4.8  Clamping curves from two simulations with different values of $k$ in the friction equation (3.1).
Figure 4.7 shows the effects of a higher value of \( m \). The vertical segments of the curve increases much, which means that the motor is standing still for a longer period of time. This is an expected result, since a higher value of \( m \) means that the friction torque function has a higher value at zero velocity and consequently, more torque is needed to transition from static friction into kinetic friction.

Figure 4.8 demonstrates the effects of a higher value of \( k \). Also here, the curve shows expected characteristics. A higher value of \( k \) should result in higher friction at all velocities, and more torque needed for the motor to drive. This is indicated by the wider hysteresis of the curve with a higher value of \( k \). Notice an important difference to Figure 4.5 — here, the motor eventually reaches the same position for both settings, whereas the curves in Figure 4.5 shows that a higher spring constant leads to a different end position. Friction does not influence the final position.

The model also has a parameter for peak friction. If the value of this parameter is \( \neq 1 \), it changes the size of the friction torque at zero velocity, and the friction torque is in that case determined by (3.1). Figure 4.9 shows the effects of using a peak-value \( \neq 1 \).

![Figure 4.9](image)

**Figure 4.9** Clamping curves from two simulations with different peak values in the friction equation (3.1).

The effects of a higher peak friction are similar to those of a higher value of \( m \). That is, a bigger segment of the curve is vertical which means that the motor is in static
Chapter 4. Results

mode for a longer time. It is a sensible result, since the friction torque around zero velocity is higher in both cases.

4.6 Backlash

The backlash parameter $b$ determines the size of the backlash in radians. Figure 4.10 illustrates the effect of backlash on the clamping curve. The two curves are results from two simulations with equal parameters and configuration, except one of them has a backlash of 0.01 radians in joint 2 and the other has no backlash.

![Axis 2 - clamping curve](image)

**Figure 4.10** Clamping curves showing the effect of backlash.

The appearance of the orange curve indicates that backlash in the model has the desired effect. When the torque is close to zero, an effect of the backlash is that the motor can rotate without much resistance. This makes the steepness of the clamping curve lower in a certain interval. In the plot above this is observed between motor angles of 0.01 and -0.05 radians. Figure 4.11 and 4.12 shows the effects of different values for the backlash parameter.

48
Figure 4.11  Clamping curves from two simulations with different backlash size.

Figure 4.12  Clamping curves from two simulations with different backlash size.

The effects on the clamping curves in Figure 4.12 when alternating the size of the backlash are as expected; a bigger backlash leads to a bigger segment of the
curve with lower steepness. In Figure 4.11, however, the size of the backlash does not seem to affect the appearance of the clamping curve much. For more analysis of this observation, see the discussion in Section 5.

4.7 Orthogonal Compliance

Each joint has a spring parameter, $c_{\text{orth}}$, that determines the amount of orthogonal compliance. Figure 4.13 shows the effects on the clamping curves when comparing two simulations with equal settings, except one of them has orthogonal compliance and one of them has not.

![Axis 2 - clamping curve](image)

**Figure 4.13** The effects of adding orthogonal compliance. Spring parameter $c_{\text{orth}} = 10^8$ Nm/rad for joints 1, 2, and 3.

There is definitely an observable effect of adding orthogonal compliance to the joints, and the clamping curves for all six joints were influenced in the simulations. As expected, the effects are very different depending on the configuration of the robot and the geometric relation between the rotational axes of the joints. The effects of joint 2, whose behavior is illustrated in Figure 4.13, are both a small decrease of the slope of the curve but also an increase of the motor rotation. When adding more compliance to a system, both of these effects seem reasonable.
Figure 4.14 Clamping curves from two simulations with different values of the orthogonal compliance spring parameter.

Figures 4.14 and 4.15 show the clamping curves for different values of $c_{\text{orth}}$.

Figure 4.15 Clamping curves from two simulations with different values of the orthogonal compliance spring parameter.
Chapter 4. Results

These clamping curves indicate a similar behavior as for the compliance in the rotational axis of the joint. However, in this case it is not the compliance of the joint itself that affects its clamping curve, but instead the compliance of the other joints of the manipulator. This will be discussed more in Chapter 5.

4.8 Compliance in the Clamped End

When the end-effector of the robot is clamped to a fixed object, there will always be a small amount of compliance in the clamping point. In the model for the clamping device, the compliance is modeled by a spring and the amount of compliance is determined by the spring constant $c$. In this experiment both the rotational spring and the linear spring is set to the same value $c$. Figure 4.16 shows the effects on the clamping curve when alternating the spring constant. In the simulation with the spring parameter, $c = 2 \cdot 10^7$ N/m (Nm/rad for the rotational spring), the end effector deviated 0.15 mm from the starting position and in the other simulation it moved 0.29 mm from the starting position. These are small differences but they seem to have a considerable influence on the appearance of the clamping curves. The effects are not trivial to interpret from the clamping curves, but a small change in maximum and minimum motor angle can be observed. The slope and the width of hysteresis of the curve, however, do not seem to be affected.
4.9 Clamping Configuration

The results of the clamping simulations appear to be quite sensitive to the configuration of the manipulator, i.e., what start position is used. Figure 4.17 shows simulations with equal parameters but different configurations. Clamping configuration 1 represents the configuration in Figure 4.4 with dashed lines and configuration 2 represents the solid drawn configuration in Figure 4.4.

**Figure 4.16** Clamping curves from two simulations with different values of the clamping device compliance.
Chapter 4. Results

The data in Figure 4.17 was generated with a model that included orthogonal compliance of the joints. Figure 4.18 show clamping curves from the same configurations but without orthogonal compliance of the joint and with a much stiffer clamping device.

Figure 4.17  Clamping curves from two simulations with different configurations.
4.10 Simulated Results Compared to Experimental Data

The data in Figures 4.17-4.18 are just one example and no definite conclusions can be made of just this particular comparison. Many more clamping configurations need to be evaluated to see a correlation between the configuration and the resulting clamping curves, but that is outside the scope of this thesis. Nevertheless, there seems to be a clear dependency but mostly in models including orthogonal compliance and compliance of the clamping device. This is expected since with more compliance in the system, the behaviour of the components of the robot are more affected by forces from surrounding components and gravitation.

4.10 Simulated Results Compared to Experimental Data

In Figure 4.19, clamping curves can be seen from simulations that were done using the same configuration as for the experimental results in Figure 4.3. The parameters used in Figure 4.19 are displayed in Table 4.1. An effort has been made to get the curves as similar as possible and the result is adequate in many aspects. The slopes of the clamping curves in the simulation were matched to that of the experimental data by adjusting the spring parameters in the rotational axes and in the orthogonal plane. Catching the behavior in the center of the curve in the experimental data was
done by adjusting the backlash in the simulation. Getting the desired behavior at the turning points was accomplished by tweaking the friction parameters. The overall behavior from the experimental data has been reconstructed in the simulations. However, the non-linear behaviour of axis 3 and 5 in the experimental curves could not be captured using a linear elasticity model.

**IRB140 clamping simulation**

![Clamping curves from a simulated clamping experiment. Compare with the experimental data in Figure 4.3.](image)

**Figure 4.19** Clamping curves from a simulated clamping experiment. Compare with the experimental data in Figure 4.3.
### Table 4.1 Parameters used for obtaining the results in Figure 4.19

<table>
<thead>
<tr>
<th>joint</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$c$</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>$d$</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>$b$</td>
<td>0.04</td>
</tr>
<tr>
<td>1</td>
<td>$\tau_{pos}$</td>
<td>[0,0.1; 1,0.4]</td>
</tr>
<tr>
<td>1</td>
<td>$friction_{peak}$</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>$c_{orth}$</td>
<td>5000000000</td>
</tr>
<tr>
<td>1</td>
<td>$d_{orth}$</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>$c$</td>
<td>8.2</td>
</tr>
<tr>
<td>2</td>
<td>$d$</td>
<td>0.58</td>
</tr>
<tr>
<td>2</td>
<td>$b$</td>
<td>0.01</td>
</tr>
<tr>
<td>2</td>
<td>$\tau_{pos}$</td>
<td>[0,0.12; 1,0.2]</td>
</tr>
<tr>
<td>2</td>
<td>$friction_{peak}$</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>$c_{orth}$</td>
<td>100000000</td>
</tr>
<tr>
<td>2</td>
<td>$d_{orth}$</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>$c$</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>$d$</td>
<td>0.001</td>
</tr>
<tr>
<td>3</td>
<td>$b$</td>
<td>0.01</td>
</tr>
<tr>
<td>3</td>
<td>$\tau_{pos}$</td>
<td>[0,0.1; 1,0.9]</td>
</tr>
<tr>
<td>3</td>
<td>$friction_{peak}$</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>$c_{orth}$</td>
<td>30000000</td>
</tr>
<tr>
<td>3</td>
<td>$d_{orth}$</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>$c$</td>
<td>0.32</td>
</tr>
<tr>
<td>4</td>
<td>$d$</td>
<td>0.005</td>
</tr>
<tr>
<td>4</td>
<td>$b$</td>
<td>0.01</td>
</tr>
<tr>
<td>4</td>
<td>$\tau_{pos}$</td>
<td>[0,0.05; 1,0.055]</td>
</tr>
<tr>
<td>4</td>
<td>$friction_{peak}$</td>
<td>1.7</td>
</tr>
<tr>
<td>5</td>
<td>$c$</td>
<td>1.4</td>
</tr>
<tr>
<td>5</td>
<td>$d$</td>
<td>0.001</td>
</tr>
<tr>
<td>5</td>
<td>$b$</td>
<td>0.05</td>
</tr>
<tr>
<td>5</td>
<td>$\tau_{pos}$</td>
<td>[0,0.08; 1,0.085]</td>
</tr>
<tr>
<td>5</td>
<td>$friction_{peak}$</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>$c$</td>
<td>0.16</td>
</tr>
<tr>
<td>6</td>
<td>$d$</td>
<td>0.001</td>
</tr>
<tr>
<td>6</td>
<td>$b$</td>
<td>0.04</td>
</tr>
<tr>
<td>6</td>
<td>$\tau_{pos}$</td>
<td>[0,0.05; 1,0.06]</td>
</tr>
<tr>
<td>6</td>
<td>$friction_{peak}$</td>
<td>1.4</td>
</tr>
</tbody>
</table>
5

Discussion and Conclusions

In this chapter, the results from the previous chapter are discussed and used to draw conclusions about the modelling and the clamping experiments.

5.1 Simulation Results

The joint compliance spring parameter (Section 4.4) is largely what determines the slope of the clamping curve, since this is the parameter which determines how much torque is needed from the motor for the gearbox to be deformed. However, when observing the results from the orthogonal compliance of the joints (Section 4.7), the same type of effect can be seen, since looser springs for the orthogonal compliance allows the links to move when the other joints are excited. So there appears to be many parameters that can affect the clamping curves in similar ways, which makes it hard to identify what parameter that gave rise to one particular effect. Compare the effect of the joint compliance spring parameter in Figure 4.5 with the effect of the orthogonal compliance spring parameter in Figure 4.14, they both affect the slope of the curve in a similar way and introduce an uncertainty about what observed effects are connected to what parameters.

This uncertainty is also present when observing the width of the hysteresis of the curve. Considering the effect of the joint compliance damper parameter in Figure 4.6, it is easy to see that a higher value for the damping parameter adds to the width of the hysteresis of the curve. The same behaviour can be seen when looking at Figure 4.8, where the effect of the value of $k$ in the friction equation has been examined. These two parameters appear to give the same effect on the clamping curve. This should come as no surprise though, as they basically achieve the same thing in this simplified friction model. They both give more torque when the axis is moving faster.

The effects of the backlash component is not straightforward to analyze. Firstly, the inclusion of it in the model can be seen very clearly in the clamping curve (Fig-
5.2 Friction

When modeling friction torque as a function of angular velocity, it is certainly a limitation and a rough simplification to only use a first-order polynomial. There are a lot of dynamic behavior that will not be captured in this linear model. For instance, the behavior during the short time interval when an axis in the robot leaves a stationary state and starts to rotate is typically not well described by a linear function. The friction characteristics are also known to vary with temperature. This would mean that there could be a substantial difference between running a cold robot that has been idle for a while and a robot that is already warmed up. This, and a lot more on the topic of friction in robot models, can be found in [Niglis and Öberg, 2015].

During the clamping experiments, the manipulator is stationary and the joints will move very slowly. This led us to believe that additional emphasis should be put on the friction effects for angular velocities close to zero. In the work with the model library in this thesis, an attempt was made to implement a more advanced model. The focus of that implementation was on the friction effects when the angular velocity is close to zero. In that model, the torque at low velocities was described using a polynomial of higher degree. However, due to time limitations, a working model was never completed within this project.

5.3 Link Elasticity

The compliance in the joints does not fully describe the manipulators elastic behavior. Another source of compliance could be the elasticity in the links. It could be interesting to develop the link models in the library to also model the link stiffness and the dynamic behavior connected to the stiffness. This is a complicated subject and it was not possible to cover it in the scope of this thesis. If the link elasticity was modeled, it could possibly have some impact on the results of the simulations in this thesis, but since the load used in clamping experiments are quite low, one can argue that the effects of link elasticity would be quite low. More on the topic of link elasticity in robots can be found in [Nilsson et al., 2013].
5.4 Joint Types

In the model library developed for this thesis, only revolute type joints are included. In robot manipulators, those are not the only type of joints that are common. Another often seen joint type is the prismatic joint. Prismatic joints are joints that allow linear relative motion between two links [Spong et al., 2005]. Models for such joints were not developed for this thesis because of time limitations. However, if there had been more time available, it would had been interesting to develop such models since there are a lot of industrial robots that uses prismatic joints, or a combination of revolute and prismatic.

5.5 Future Work

The work of this thesis has been centred mostly around the development of the Modelica models and not as much around using the models for simulations, except for in model validation purposes. The results are from only one type of robot, ABB’s IRB140, and it would be interesting to set up models for other type of robots to evaluate the results. However, the model library was not built with the IRB140 in consideration and there is nothing indicating it would work worse for other robot types. It would also be interesting to extend the model library to be able to model parallel-bar robots, like the ABB IRB2400 robot [ABB IRB2400 data, 2017], which is currently not possible.

As discussed previously in this chapter, the friction models of the robot could be more advanced and that would possibly have an impact on the results, especially around the lowest motor velocities in the simulations. To improve the friction model would be of high priority if more work should be done with the models.

5.6 Conclusions

The overall purpose of this thesis was to establish models, to investigate how the dynamical effects of the manipulator affect the clamping curves in clamping experiments, and to validate them. The effects of compliance in the joint (both along the axis of rotation and in the plane orthogonal to this axis), the effects of backlash, the effects of compliance in the clamping device and the effects of the clamping configuration have all been studied and the models work well in many aspects. The effects of the friction, however, could be investigated further with a more advanced friction model (as discussed in Section 5.2).

The goal was to create a library that could be used to model and simulate many different types of manipulators within the delimitations set up prior to project start. All models built with six degrees of freedom or less have simulated successfully. The modularity aspect of the library was also discussed in the introduction to this thesis. This has been considered throughout the work with the model library and
all setups of manipulator models are made with submodels that can be reused for different types of manipulators.
Bibliography


A

Modelica Models

A.1 Frame

connector Frame

import SI = Modelica.SIunits;

SI.Position r[3]
    "Position of frame, expressed in inertial frame."

    "Orientation object of frame."

flow SI.Force f[3];
flow SI.Torque t[3];
end Frame;

Listing A.1  Modelica code for the connector Frame.

A.2 InertialFrame

model InertialFrame

    Frame frame_end;

    equation
        frame_end.R = Frames.nullRotation()
            "Defining null rotation of InertialFrame."
        frame_end.r = zeros(3) "Defining the position of origin."

        Connections.root(frame_end.R)
            "Defining root node in virtual connection graph."
end InertialFrame;

Listing A.2  Modelica code for the model InertialFrame.
A.3 Flange

connector Flange

import SI = Modelica.SIunits;
SI.Angle phi;
flow SI.Torque tau;
end Flange;

Listing A.3 Modelica code for the model Flange.

A.4 Link

model Link "Stiff link model."

import SI = Modelica.SIunits;


parameter SI.Length cm[3] = l/2
"Vector from frame_beg to center of mass."

parameter Real width = 0.1 "Only used for animation."

Frame frame_beg;
Frame frame_end;

SI.Velocity v[3](start={0,0,0});
SI.Acceleration a[3];

SI.AngularVelocity w[3](start={0,0,0});
SI.AngularAcceleration z[3];

parameter SI.Inertia I[3,3] = 0.01*identity(3)
"Inertia matrix."

parameter SI.Mass m= 10 "Mass of link."

constant SI.Acceleration g[3] = {0, -9.82, 0}
"Gravity acceleration vector."

shapeType="box",
color={255,0,0},
length=Modelica.Math.Vectors.norm(l),
width=width,
Appendix A. Modelica Models

height = width,
lengthDirection = l/Modelica.Math.Vectors.length(l),
widthDirection = {0,1,0},
r_shape = zeros(3),
r = frame_beg.r,
R = frame_beg.R) "Animation";

equation
// Velocity and acceleration of link center of gravity.
v = der(frame_beg.r + Frames.resolve1(frame_beg.R,cm));
a = der(v);

// Angular velocity and acceleration of frame_beg.
z = der(w);

// Force and torque equations, Newton-Euler backward recursion.
frame_beg.f = -frame_end.f + m*(Frames.resolve2(frame_beg.R,a - g));
frame_beg.t = -frame_end.t + cross(cm, frame_beg.f) +
cross(-frame_end.f, -l + cm) +I*z + cross(w, I*w);

// Defining potential root and non-breakable branch.
Connections.potentialRoot(frame_beg.R);
Connections.branch(frame_beg.R, frame_end.R);

// Link does not bend, rotation is equal.
frame_end.R = frame_beg.R;

// Definition of positional relationship between endpoints.
frame_end.r = frame_beg.r + Frames.resolve1(frame_beg.R, l);
end Link;

Listing A.4 Modelica code for the model Link.

A.5 Joint

model Joint "Joint model with actuator, modelled with compliance, friction, and backlash."
import SI = Modelica.SIunits;
import C = Modelica.Constants;

// Parameters to pass to RevoluteJoint and Revolute Axis.
parameter Modelica.Mechanics.MultiBody.Types.Axis n = {0,1,0};
parameter Real c_orth = 1000000;
parameter Real d_orth = 30;

parameter SI.RotationalSpringConstant c = 50;
parameter SI.RotationalDampingConstant d = 10;
parameter SI.Angle b = 0.01;
parameter Real gearRatio = 140;
A.6 RevoluteJoint

Listing A.5  Modelica code for the model Joint.

A.6 RevoluteJoint

model RevoluteJoint "Revolute joint with orthogonal compliance modelled by spring and damper in parallel."

import SI = Modelica.SIunits;
import C = Modelica.Constants;


parameter Real c_orth "Spring constant for orthogonal compliance."
parameter Real d_orth "Damping constant for orthogonal compliance.";
Appendix A. Modelica Models

SI.Angle q(start = 0) "Rotation angle, starting at zero."
"Angular velocity, starting at zero."

Real orthRotAxis1[3] "Axis orthogonal to n."
Real orthRotAxis2[3] "Axis orthogonal to n and orthRotAxis1."

SI.Angle orth1(start = 0)
"Angular displacement in first orthogonal axis."
SI.AngularVelocity wo1(start = 0, fixed = true);

SI.Angle orth2(start = 0)
"Angular displacement in second orthogonal axis."
SI.AngularVelocity wo2(start = 0, fixed = true);

Frame frame_beg;
Frame frame_end;

SI.Torque orthTorque1 "Torque in orthRotAxis1."
SI.Torque orthTorque2 "Torque in orthRotAxis2."

Flange flange "Flange for connecting external driving axis."

protected
Frames.Orientation R_rel
"Relative rotation of frame_beg and frame_end."

equation
(orthRotAxis1, orthRotAxis2)
= getOrthVectors(n) "Make orthRotAxis1 and orthRotAxis2 orthogonal to each other and to n."

// Define relationship between angle and angular velocity.
w = der(q);
wo1 = der(orth1);
wo2 = der(orth2);

frame_end.r = frame_beg.r;
-frame_end.t*n = flange.tau
"Torque in rotational axis, from external actuator."

// Torque in orthogonal rotational axis.
frame_end.t*orthRotAxis1 = orthTorque1;
frame_end.t*orthRotAxis2 = orthTorque2;

// Hooke's law.
orthTorque1 = c_orth*orth1 + d_orth*wo1;
orthTorque2 = c_orth*orth2 + d_orth*wo2;

flange.phi = q;

// Relative rotation of joint in rotational axis and rotation in orthogonal axes because of compliance.
A.7 RevoluteAxis

model RevoluteAxis "Driving axis with gearbox containing elasticity, backlash, and friction. Sensors for speed and angle are available for controller."

import SI = Modelica.SIunits;
import Modelica.Mechanics.Rotational;

parameter SI.RotationalSpringConstant c = 50 "Spring constant for gearbox elasticity."
parameter SI.RotationalDampingConstant d = 10 "Damper constant for gearbox elasticity."
parameter SI.Angle b = 0.001 "Backlash of gearbox (rad)."
parameter Real gearRatio = 1 "Gear ratio of gearbox."
parameter Real inertia = 0.01 "Inertia of motor."
parameter Real tau_pos[:,2] = [0, 1; 1, 1.1] "Friction values."
parameter Real friction_peak = 1 "Friction peak in w = 0."
parameter Boolean breaksOn = false "Break parameter."

Motor motor(breaksOn = breaksOn);
Rotational.Components.Inertia I(J = inertia);
Rotational.Components.BearingFriction bearingFriction(tau_pos = tau_pos, useSupport = false, peak = friction_peak);
Rotational.Components.ElastoBacklash backlash(c = c, d = d, b = b);
Gear gear(ratio = gearRatio);

Rotational.Sensors.AngleSensor angleSensor;
Rotational.Sensors.SpeedSensor speedSensor;
Modelica.Blocks.Interfaces.RealInput torque;

Flange flange;
ControlBus controlBus;

equation
  connect(motor.flange, I.flange_a);
  connect(I.flange_b, bearingFriction.flange_a);
  connect(bearingFriction.flange_b, backlash.flange_a);
  connect(backlash.flange_b, gear.flange_beg);
Appendix A. Modelica Models

Listing A.7  Modelica code for the model RevoluteAxis.

A.8  Motor

Listing A.8  Modelica code for the model Motor.

A.9  Gear

Listing A.9  Modelica code for the model Gear.
equation
  flange_beg.phi = flange_end.phi * ratio;
  flange_beg.tau * ratio = -flange_end.tau;
end Gear;

Listing A.9  Modelica code for the model Gear.

A.10  ControlBus

expandable connector ControlBus "Connector used for communications between sensors and controller."
import SI = Modelica.SIunits;

  SI.AngularVelocity motorVelocity;
  SI.Angle motorAngle;

  SI.Angle angle_ref;
  SI.AngularVelocity velocity_ref;

  SI.Torque motorTorque;
end ControlBus;

Listing A.10  Modelica code for the model ControlBus.

A.11  ClampingPoint

model ClampingDevice "Clamping modelled by spring and damper in parallel, one for spatial displacement and one for rotational."
import SI = Modelica.SIunits;

  Frame frame;

  parameter Real[3,3] c_lin = 9001*identity(3)
    "Spring constants for linear springs.";
  parameter Real[3,3] d_lin = 9001*identity(3)
    "Damper constants for linear dampers.";

  parameter Real[3,3] c_rot = 9001*identity(3)
    "Spring constants for rotational springs.";
  parameter Real[3,3] d_rot = 9001*identity(3)
    "Damper constants for rotational dampers.";

  SI.Position dr[3](start = {0, 0, 0}, fixed = true)
    "Spatial displacement vector.";
  SI.Velocity v[3];

  SI.Angle q[3](start = {0, 0, 0}, fixed = true)
Appendix A. Modelica Models

"Angular displacement vector";
SI.AngularVelocity w[3];

equation

// Defining relationships between position and velocity.
der (frame.r) = v;
der (dr) = v;
der (q) = w;

// Hooke's law.
frame.t = c_rot*q + d_rot*w;
frame.f = c_lin*dr + d_lin*v;
Frames.angularVelocity2(frame.R) = w;
end ClampingDevice;

Listing A.11  Modelica code for the model ClampingPoint.

A.12 Controller

model Controller

import SI = Modelica.SIunits;
import Modelica.Blocks.Math;
import Modelica.Blocks.Continuous;

parameter Real kp=10 "Gain of position controller";
parameter Real ks=1 "Gain of speed controller";
parameter SI.Time Ts=0.01
    "Time constant of integrator of speed controller";
parameter Real ratio=1 "Gear ratio of gearbox";

Math.Gain gain1 (k=ratio);
Continuous.PI PI(k=ks, T=Ts,
    initType=Modelica.Blocks.Types.Init.InitialOutput);
Math.Feedback feedback1;
Math.Gain P(k=kp);
Math.Add3 add3 (k3=-1);
Math.Gain gain2 (k=ratio);

ControlBus controlBus;

equation

connect (gain1.y , feedback1.u1);
connect (feedback1.y , P.u);
connect (P.y , add3.u2);
connect (gain2.y , add3.u1);
connect (add3.y , PI.u);
connect (gain2.u , controlBus.velocity_ref);
connect (gain1.u , controlBus.angle_ref);
connect (feedback1.u2 , controlBus.motorAngle);
connect (add3.u3 , controlBus.motorVelocity);
connect (PI.y , controlBus.motorTorque);
// Derived from
end Controller;

Listing A.12  Modelica code for the model Controller.

A.13  Wrist

model Wrist "Wrist model with transmission matrix, external actuators are required to move it."

import SI = Modelica.SIunits;

Frame frame_beg;
Frame frame_end;

parameter Real transmission [3,3] "Transmission matrix.";
parameter Real controlJoint = 1
  "(1 = joint 4, 2 = joint 5, 3 = joint 6)";

Flange axis1;
Flange axis2;
Flange axis3;

// Rotation angle vector and angular velocity vector.
SI.Angle q[3](start = {0,0,0});
SI.AngularVelocity w[3](start = {0,0,0}, fixed = true);

protected
  Frames.Orientation R_rel;

equation
  w = der(q);

  frame_end.r = frame_beg.r;

  transpose(transmission)*(-frame_end.t) =
  {axis1.tau, axis2.tau, axis3.tau};

  transmission*{axis1.phi, axis2.phi, axis3.phi} = q;

  // Rotation of wrist.
  R_rel = Frames.axesRotations({1,2,3}, q, w);
  frame_end.R = Frames.absoluteRotation(frame_beg.R, R_rel);

  // Force and torque transfer.
  frame_beg.f = -Frames.resolve1(R_rel, frame_end.f);
  frame_beg.t = -Frames.resolve1(R_rel, frame_end.t);
end Wrist;

Listing A.13  Modelica code for the model Wrist.
block ClampingPathPlanning "Block for generating reference signals (angle and velocity) for clamping experiments."

import SI = Modelica.SIunits;
import C = Modelica.Constants;
import Modelica.Blocks.Sources;

Modelica.Blocks.Interfaces.RealOutput q;
Modelica.Blocks.Interfaces.RealOutput qd;

parameter Real q_pos = C.pi/20 "maximum/minimum angle";
parameter Real qd_max = 0.2 "maximum velocity";
parameter Real qdd_max = 0.5 "maximum acceleration";

Sources.KinematicPTP2 path1 {
    q_end={q_pos},
    qd_max={qd_max},
    qdd_max={qdd_max},
    startTime=0,
    q_begin={0});

Sources.KinematicPTP2 path2 {
    q_end={-q_pos},
    qd_max={qd_max},
    qdd_max={qdd_max},
    startTime=3,
    q_begin={q_pos});

Sources.KinematicPTP2 path3 {
    q_end={q_pos},
    qd_max={qd_max},
    qdd_max={qdd_max},
    startTime=6,
    q_begin={-q_pos});

Sources.KinematicPTP2 path4 {
    q_end={-q_pos},
    qd_max={qd_max},
    qdd_max={qdd_max},
    startTime=9,
    q_begin={q_pos});

Sources.KinematicPTP2 path5 {
    q_end={q_pos},
    qd_max={qd_max},
    qdd_max={qdd_max},
    startTime=12,
    q_begin={-q_pos});

equation
A.15 getOrthVectors

function getOrthVectors "Function returns two arbitrary vectors orthogonal to each other and to input vector."
input Real n[3];
output Real u[3];
output Real v[3];
algorithm
  // First vector u, is computed using crossproduct with input vector, n, and an arbitrary vector not parallel to n.
  if n[1] == 0 then
    u := cross({1,0,0}, n);
  elseif n[2] == 0 then
    u := cross({0,1,0}, n);
  else
    u := cross({0,0,1}, n);
  end if;
  u := Modelica.Math.Vectors.normalize(u);
  v := cross(n, u) "v is computed by crossing n with u."
end getOrthVectors;

Listing A.15 Modelica code for the function getOrthVectors.
Declarative Models for Self-Calibrating Robots

Abstract
The use of industrial robots in manufacturing often requires good calibration approaches and tools to obtain and maintain the desirable accuracy of the robot, for the robot to be able to execute a task in a satisfactory manner. For high-accuracy methods of calibration, it is necessary to first identify the robot’s stiffness parameters. This thesis will focus on a method that is based on a clamping procedure. In this procedure, the robot’s end effector is clamped to a fixed object and load is applied by the motors of the joints while measurements of torque and position of the motors are performed by the robot’s own sensors. These measurements are then used to identify the stiffness parameters of the robot. The purpose of this thesis is to develop a model library written in the modeling language Modelica to be used for robot simulations, and especially to simulate these clamping experiments. In the modeling, focus will lie on dynamical behaviour like compliance, backlash, and friction and how these effect the results of the clamping experiments.

The modularity aspect has been a central part in the progress and the model library developed is component based, i.e., the parts can be connected in various ways to model different types of robots. For model validation, many simulations have been performed with different dynamic model parameters to evaluate the results and how they compare to the expected dynamic behaviour. Also, experimental data from experiments with ABB’s robot IRB140 have been used to judge how well the models can reproduce results from real experiments.

The simulation results indicate that the models capture much of the dynamical behaviour in a satisfying way. The effects of joint compliance, friction, backlash, compliance in the clamping device, and variations in the clamping configuration are all of importance for the results of the clamping simulations. When comparing to data provided from clamping experiments of the IRB140 robot, the results of the simulations show that the models are able to reproduce much of the robot’s behaviour observed in the experiments.

Keywords