ISSN 0280-5316
ISRN LUTFD2/TFRT--5848--SE

Force Controlled Grinding -The Cutting Edge

Olof Sörnmo

Department of Automatic Control
Lund University
December 2009

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
December 2009

Document Number
ISRN LUTFD2/TFRT--5848--SE

Author(s)
Olof Sörnmo

Supervisor
Anders Robertsson Automatic Control, Lund
Rolf Johansson Automatic Control, Lund (Examiner)

Sponsoring organization

Title and subtitle
Force Controlled Grinding-The Cutting Edge. (Kraftreglerad knivslipning)

Abstract
Sharpening knives by hand is both time-consuming and exhausting, and may still not always yield perfect results. This thesis investigates the possibility of sharpening knives with the use of a force-controlled industrial robot, regardless of the knifes shape. The procedure is performed by first identifying the shape of the knife, using Matlab Simulink to simulate the identifiation; two different types of force control are evaluated. Using an ABB IRB140B robot, the best performing controller is then used to identify a real knife. Based on the shape recorded by the robot, grinding experiments were successfully performed.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
56

Recipient's notes

Security classification

Contents

1 Introduction

1.1 Background

Robots are becoming more common in today's society, whether it is at home, in electronics or in the industry. The definition of a robot is not unequivocal, however a robot generally satisfies one or more of these conditions:

- It has the ability to translate or rotate around one or more axes, and can sense its environment and interact with it.
- It is programmable to act without the direct intervention of a human.
- It has artificial intelligence to some degree, being able to make simple decisions.

The type of robot that is dealt with in this thesis is the industrial robot. In today's industry, robots are being used frequently for a lot of different applications, such as drilling, welding, grinding, painting and assembly. In all of these applications, the robot is required to come into physical contact with the work object in order to complete its task. Some of these tasks, drilling for example, require the robot to exert a specific force on the work object. If the force is too strong, the work object or the robot might break, and if the force is too weak, the task will not be executedproperly. It is therefore crucial that the exerted force can be controlled in a well-behaved manner. In applications such as grinding, which is to be dealt with in this thesis, not only the force needs to be controlled precisely, but the motion of the robot as well.

1.2 Problem formulation

The topic of this thesis is to sharpen a knife with unknown shape, using a robot with force control. To accomplish this, the shape must be identified, first in a simulated environment using an arbitrary curve as a model of the knife. Next, the controller from the simulations is tested on the real robot. The recorded shape of the knife will be used as a reference for the grinding procedure. The identification should be performed without regard to the orientation of the knife.

1.3 Method

In order to model and control a robot, a theoretical study in the following areas is required:

- Robot frames and transformation matrices
- Kinematics and differential kinematics
- Force control

A brief introduction to these areas can be found in Section 2 on page 9.

The problem is divided into the following parts:

1. To model the robot and knife using MATLAB Simulink. To design a controller that identifies the shape of the knife regardless of what knife model is chosen. An aspect to consider is whether to mount the knife on the robot or to fix it in space during the identification. In simulations, both scenarios are to be evaluated.
2. To calibrate the robot so as to determine the force sensor frame and to be able to compensate for gravitational effects on the tool. Also, to identify the orientation of the knife.
3. To create a Simulink model that records the knife's shape using the real robot, with the same identification controller from the simulated model. Modification of the controller may be necessary due to friction and/or model errors. In order to be able to grind the knife later, the knife needs to be fixed on the robot during the actual shape identification.
4. To create a Simulink model that executes the grinding of the knife with the real robot, using the recorded shape from the identification as reference.

1.4 Simulation platform

In this thesis, simulations were mainly done in MATLAB with the extension Simulink, which is a graphical interface for creating models by using user- or predefined blocks. The models can both be simulated and exported to run on a real robot system, using *Real Time Workshop*. For the models in this thesis, predefined blocks from the *extctrl*-library [1] that describe the robot characteristics and dynamics, were used.

1.5 Experimental platform

Experiments were executed using an *ABB IRB-140B* robot, equipped with a *JR3 force sensor*, available in the robot laboratory at the Department of Automatic Control at LTH, Lund University.

1.5.1 ABB IRB-140B robot

The ABB IRB-140B robot is one of the smaller industrial robots, with a reach of 0.81 m and a maximum payload of 6 kg [2]. A serial industrial robot like the IRB-140B consists of several links, connected to each other with different types of joints. This specific robot consists of 6 joints of two different types, which gives it a total of six degrees of freedom (DOF), see Figure 1. A robot needs at least three DOF to move freely in space. The other three DOFs enable the robot to have an arbitrary tool orientation, which means that the robot can approach a point from any direction, given that the direction is inside the robot's working space. At the end of the last link, a small metal plate is fixed, called *flange*, where tools or force sensors are attached [3].

The robot system is controlled and powered by an ABB IRC5 cabinet, which has internal controllers for each joint [4]. When an external controller is connected to the robot system, the output from the controller is given as a position and velocity reference for each joint, which serves as input to the internal controllers. During calibrations, the robot is sometimes manually moved, or *jogged*, by using the robot's control panel; the *FlexPendant*, see Figure 2. In order to jog the robot, the *dead man's switch* on the FlexPendant must first be pressed. Then, using the joystick on the FlexPendant, the robot can be jogged in a linear motion or in each joint separately.

1.5.2 JR3 Force sensor

A JR3 force sensor of model 100M40AI63 is mounted on the flange, see Figure 3, in order to provide measurement feedback to the force controllers. The sensor measures forces and torques in the x-, y- and z-directions, with a sampling rate of 8 kHz and an uncertainty of about ± 2 N in force [5]. The measurements from the sensor are accessed from an external computer connected to the sensor.

1.5.3 Real Time Workshop and Opcom

To run experiments on the real system, the Simulink models must be converted to C-code, using a Real Time Workshop-plugin which generates the code. The converted model is then loaded in to a graphical interface called *Opcom*, that communicates with the robot through the IRC5 cabinet. The interface is displayed in Figure 4.

As shown in Figure 4 the interface has four modes: *unload*, *load*, *submit* and *obtain*. To load a model into the interface, the model path is entered and *load* is selected. Once the model has been loaded, it starts running immediately. When the mode *submit* is selected, the model gains access to

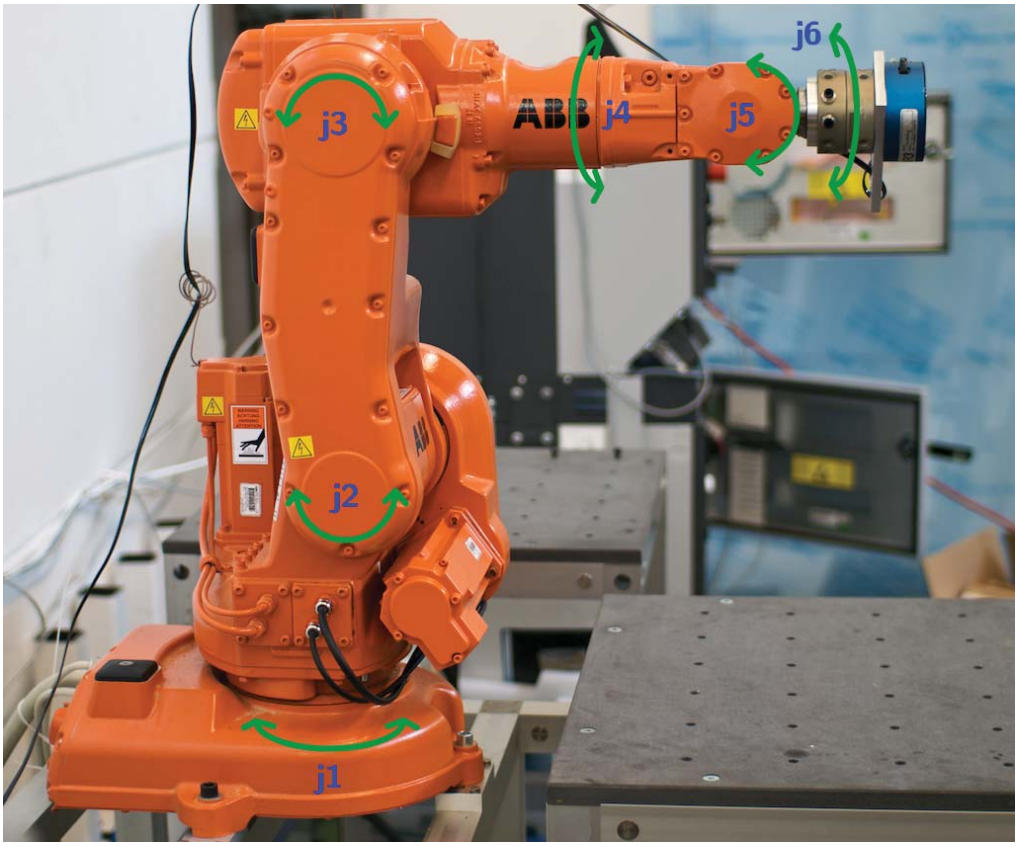

Figure 1: The IRB-140B robot in its home position

the signals sent from the robot system, such as joint angles, joint velocities and force/torque measurements. In this mode, it is not possible for the model to affect the robot system in any way. In the last mode, *obtain*, the inputs to the robot system are switched from the internal references to the outputs of the model, which gives the model full control over the robot. Since the model starts executing as soon as it is loaded and its outputs do not affect the robot in *load* or *submit*, possible integrators in the model will continue to grow until *obtain* is activated. This will in most cases lead to a too large input to the robot, outside of its constraints. To prevent this, a boolean variable called *fswitch* is introduced in the model, which set to zero by default. Once the value of *fswitch* is changed to one, the controllers become active. The value of *fswitch* can be changed directly in the Opcom interface, under parameters.

Once the experiments are done, the model is unloaded by selecting *unload*.

Figure 2: The robot control panel, the FlexPendant

2 Theory

2.1 Robot frames and transformation matrices

2.1.1 Robot frames

In order to describe the position and orientation of the tool attached to the robot, a set of frames has to be introduced. These frames are three-dimensional Cartesian coordinate systems, and are defined to be orthogonal. The first frame is the only one that is fixed in space and is attached to the base of the robot, therefore it is called *base frame*.

Next, a frame is fixed to the flange called the *flange frame*, with the z-direction normal to the flange surface. In case there is a force sensor attached to the flange, the *sensor frame* has to be defined, which is done in the flange frame. Further, a tool is attached to the force sensor or flange and at the tip, called the *tool center point (TCP)*, the *TCP frame* is fixed. This frame is also expressed in the flange frame, but does not necessarily have to be fixed throughout the robot motion. The frame definitions are found in Figure 5.

In order to define these frames, certain concepts of linear algebra are recalled. An arbitrary point in the three dimensional room can described in the base frame by the vector

Figure 3: The JR3 force sensor mounted on the robot

$$\boldsymbol{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

where p_x, p_y, p_z are the point coordinates. A point does not have an orientation, so assume instead that the point is the corner of a cube, with a frame attached to it, with the frame's origin in this point. Further, the cube is oriented so that its z-direction does not coincide with the z-direction of the base frame, see Figure 6.

The vector $\boldsymbol{p}$ can only describe *where* the origin of the cube frame is located in the base frame, and not *how* it is oriented. The orientation of the frame is represented using a *rotation matrix*, which is introduced in the next section.

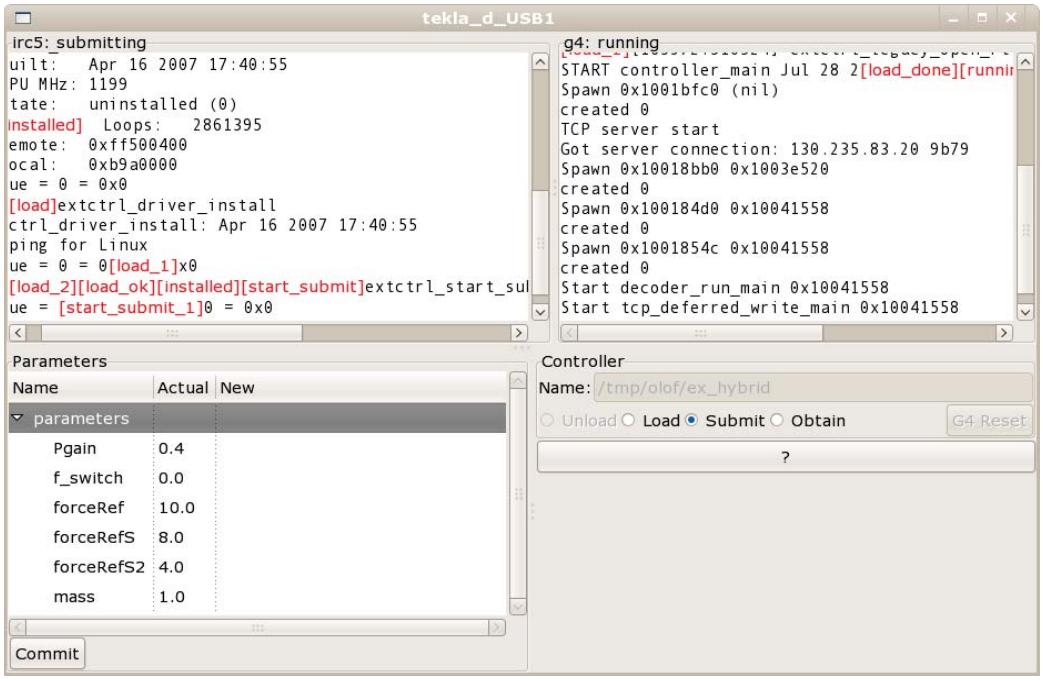

Figure 4: The Opcom interface

2.1.2 Rotation matrices

To describe the orientation of an object or a frame, the rotation matrix has to be introduced. Assume that the orientation of the cube in Figure 6 is to be described in the base frame. This is done by expressing the cube frame's unit vectors in the base frame, and concatenating them into a rotation matrix R:

$$\boldsymbol{r}_x = \begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{bmatrix} \quad \boldsymbol{r}_y = \begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix} \quad \boldsymbol{r}_z = \begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{bmatrix}$$

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{r}_x & \boldsymbol{r}_y & \boldsymbol{r}_z \end{bmatrix} = \begin{bmatrix} r_{xx} & r_{yx} & r_{zx} \\ r_{xy} & r_{yy} & r_{zy} \\ r_{xz} & r_{yz} & r_{zz} \end{bmatrix} \tag{1}$$

The rotation matrix $\boldsymbol{R}$ describes how much the base frame unit vectors needs to be rotated in each direction to have the same orientation as the cube frame.

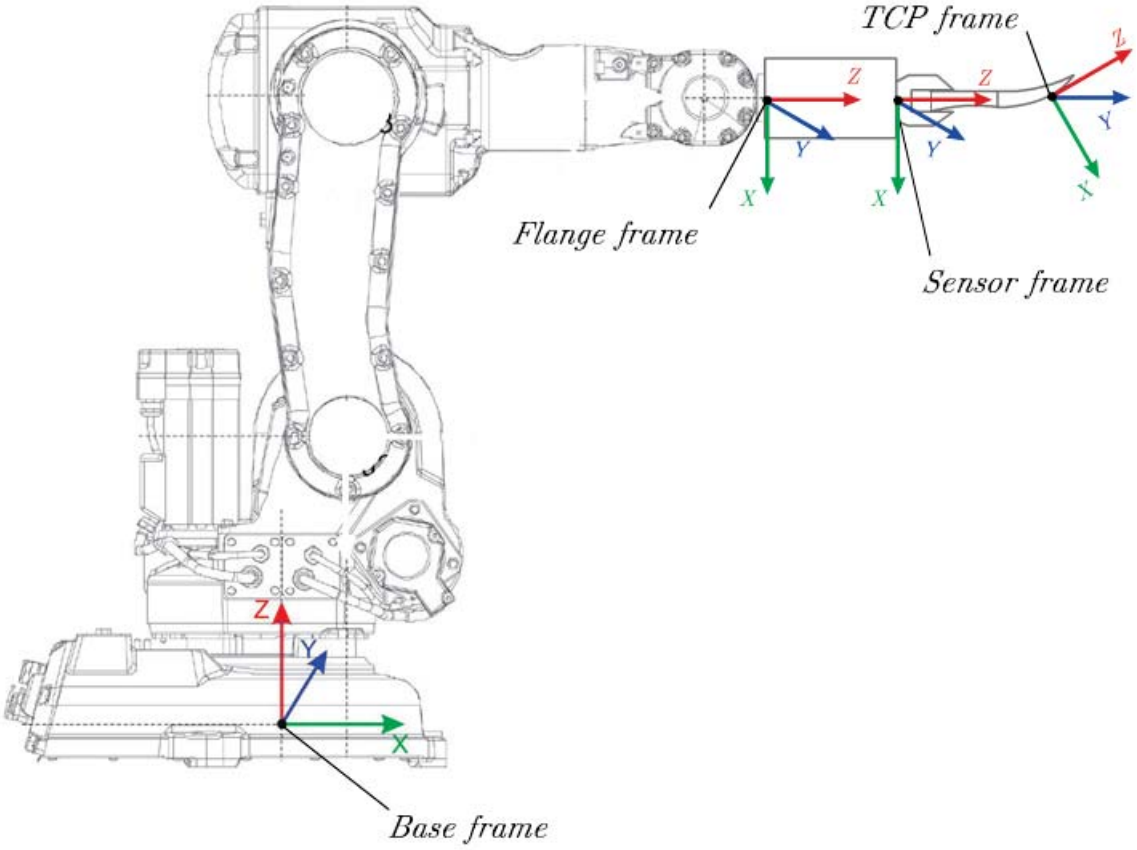

Figure 5: The frames defined on the robot.

2.1.3 Frame transformations

From the two previous sections we now have means to describe an arbitrary frame, by defining a point and an orientation in the base frame. A point $\boldsymbol{p}$ that is defined in a different frame than the base frame, for example the flange frame, needs to be transformed in order to be expressed in coordinates of the base frame. If the flange frame has the same orientation as the base frame, the point coordinates in the base frame are simply defined as:

$$\boldsymbol{p}_b = \boldsymbol{p}_f + \boldsymbol{O}_{b,f}$$

where $\boldsymbol{O}_{b,f}$ is the vector between the base frame origin and the flange frame origin.

If the flange frame is differently oriented than the base frame, the coordinates also need to be rotated. This is done by multiplying the rotation matrix

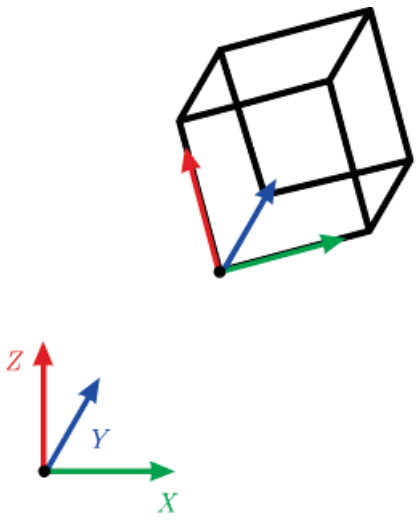

Figure 6: A cube defined in the base frame.

that describes the orientation of the flange frame, with the point coordinates. The total transformation is given by:

$$\boldsymbol{p}_b = \boldsymbol{R} \cdot \boldsymbol{p}_f + \boldsymbol{O}_{b,f} \tag{2}$$

It is often desirable to do several frame transformation in a row. This is easiest done by modifiying (2) so that a single matrix multiplication can represent the full transformation :

$$\begin{bmatrix} \boldsymbol{p}_b \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{O}_{b,f} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_f \\ 1 \end{bmatrix} \tag{3}$$

which is equivalent to

$$\boldsymbol{p}'_b = \boldsymbol{T}_{b,f} \cdot \boldsymbol{p}'_f \tag{4}$$

The matrix $\boldsymbol{T}$ in (4) is called a *T44 transformation matrix*, in this case from flange frame to base frame. To transform a point between frames, the T44 matrix is simply multiplied with the point vector that is to be transformed. The T44 matrix can also be used to describe the relation between different frames. For example, if the $\boldsymbol{T}_{f,T}$ and the $\boldsymbol{T}_{b,f}$ is known, the relation between the TCP frame and the base frame is defined as:

$$\boldsymbol{T}_{b,T} = \boldsymbol{T}_{f,T} \cdot \boldsymbol{T}_{b,f}$$

$\boldsymbol{T}_{f,T}$ is often unknown and has to be estimated. $\boldsymbol{T}_{b,f}$ is calculated using *forward kinematics*, which is considered in the next section.

2.2 Kinematics and differential kinematics

2.2.1 Forward kinematics

Assuming that all angles $\boldsymbol{q}$ between the different robot links are known, the position of the flange $\boldsymbol{p}$ in the base frame can be calculated using forward kinematics [6]. In order to determine the forward kinematics, one can use the standard *Denavit-Hartenberg convention*, which utilizes the *link length* a_i, *link twist* α_i, *link offset* d_i and *joint angle* θ_i to derive a T44 transformation matrix from link to link. The general form of the transformation matrix between link $i-1$ and i is given by:

$$\overset{\mathbf{i-1}}{\mathbf{i}}T = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i)cos(\alpha_i) & sin(\theta_i)sin(\alpha_i) & a_i cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i)cos(\alpha_i) & -cos(\theta_i)sin(\alpha_i) & a_i sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying the transformation matrices between all the links gives the final forward kinematics matrix; for a robot with n links, it is given by:

$$\boldsymbol{T}_{b,f}(\boldsymbol{q}) = {}^{0}_{n}T(\boldsymbol{q}) = {}^{n}_{i=1}\Pi {}^{i-1}_{i}T(\boldsymbol{q}) \tag{5}$$

Remembering the form of the T44 matrix from (3), the flange position is now given by the three first rows in the fourth column of the matrix $\boldsymbol{T}_{b,f}(\boldsymbol{q})$ from (5):

$$T = \begin{bmatrix} R & \cdots & \cdots & x_{flange} \\ \vdots & \ddots & & y_{flange} \\ \vdots & & \ddots & z_{flange} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

where x, y, z are the flange frame origin coordinates. To extract these coordinates from the T44 matrix, it is multiplied with a vector:

$$\begin{bmatrix} x_{flange} \\ y_{flange} \\ z_{flange} \\ 1 \end{bmatrix} = \begin{bmatrix} R & \cdots & \cdots & x_{flange} \\ \vdots & \ddots & & y_{flange} \\ \vdots & & \ddots & z_{flange} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

To transform a point defined in the flange frame to the base frame is done by simply multiplying the point coordinates with the T44 matrix:

$$\boldsymbol{p}_b = \boldsymbol{T}_{b,f}(\boldsymbol{q}) \cdot \boldsymbol{p}_f$$

2.2.2 Inverse kinematics

Inverse kinematics is used to calculate the joint angles, given the position of the flange. This is a lot more difficult than the forward kinematics problem, mainly because of the possibility of multiple or no solutions. In the scenario of no solutions, the desired flange position may be outside the robot's working space, or it may be in an impossible configuration. The possibility of the existence of multiple solutions is demonstrated in Figure 7. If there exists an infinite number of solutions, the robot is said to be in a *singularity*, see Section 2.2.4.

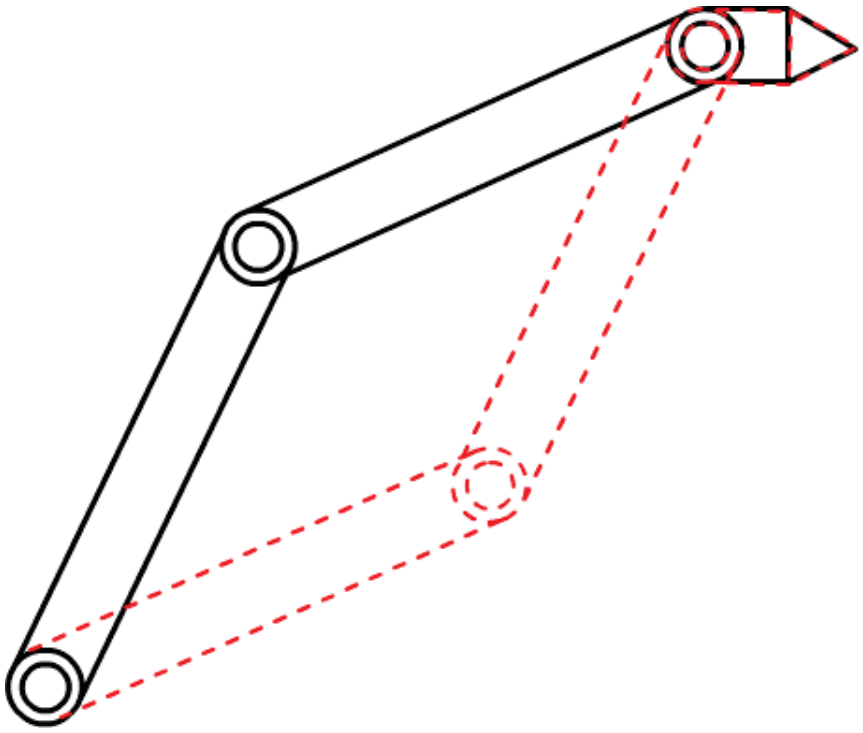

Figure 7: Two of the possible robot configurations to reach a given point and orientation.

Since it is desirable to have a continuous motion of the robot, the solutions far away from the current orientation of the robot need to be disregarded.

The last orientation is fed to the inverse kinematics and it chooses a new solution that is the closest match to the previous robot configuration.

2.2.3 Geometric Jacobian

The forward and inverse kinematics describe the relation between the joint angles and the flange position in coordinates of the base frame. It is necessary to define the linear and angular velocity of the flange, given the angular joint velocities. This is done by introducing the *geometric Jacobian* J which is used in the following relation, called the *differential kinematics equation*:

$$\begin{bmatrix} \dot{\boldsymbol{p}} \\ \boldsymbol{\omega} \end{bmatrix} = J(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{7}$$

where $\boldsymbol{\omega}$ is the angular velocity of the flange.

2.2.4 Joint singularities

A problem when using serial manipulator robots, such as the IRB140, is the presence of *joint singularities*. A singularity arises when there exists infinite solutions to the inverse kinematics problem, i.e. when a certain point and orientation can be reached in infinitely many ways. For example, when the angle of joint 5 is equal to zero, joint 4 and joint 6 will rotate around the same axis. This means that the position and orientation of the tool will be unchanged no matter what angle joint 4 or 6 has, as long as both joints have the same angle but with reversed sign.

When the robot enters a singularity, the Jacobian matrix defined in Section 2.2.3 becomes singular and the robot will not be able to leave the singularity using inverse kinematics. The home position of the robot is unfortunately defined as when all arm angles are equal to zero, thus the robot starts in a singularity. To avoid this, the angle of joint 5 can be changed slightly in order to leave the singularity before moving the robot using kinematics. It is important to make sure that the robot does not pass through or come close tosingularities during the execution of an experiment, because it will result in the robot behaving erratically and equipment may be damaged.

2.3 Force control

2.3.1 Direct force control

The simplest form of force control is the called *direct force control*. Given a force reference, the controller acts on the error between the measured and the desired force.

2.3.2 Hybrid force control

When direct force control and position control is combined, it is usually called *hybrid force control*. The robot is given a search direction in which the position controller will move the robot in, under circumstance that it is not in contact with an object, i.e. the output from the force sensor is zero. Once contact is achieved, the controller switches from position to force control. Using a given force reference, the controller will strive at keeping the force identical to the referecnce at all times and orientations.

2.3.3 Impedance control

When an object is subjected to a force, the ability of the object to resist motion is measured in mechanical impedance. The idea of impedance control is to control the ratio between force and position, instead of acting on them separately. The simplest form of impedance is the relation known as Hooke's law:

$$F = Kx \tag{8}$$

where x is the position, F is the resulting force at the contact point with the tool and K is a force/spring constant specific for the object in contact. A desired position x_d is introduced by substituting x with $\Delta x = x - x_d$:

$$F = K\Delta x \tag{9}$$

Equation (9) only gives the relation between force and position, and thus needs to be extended with a velocity term. The relation between force and velocity can be written as:

$$F = D\Delta\dot{x} \tag{10}$$

where D is a constant that represents damping. Combining (9) and (10) gives

$$F = D\Delta\dot{x} + K\Delta x \tag{11}$$

Using (11) together with Newton's second law

$$F = M\ddot{x} \tag{12}$$

where M is the object mass, the desired equation for the impedance becomes

$$F = M\ddot{x} + D\Delta\dot{x} + K\Delta x \tag{13}$$

where force, acceleration, velocity and position is incorporated. Given the desired position and velocity, the measurements of actual force, position and velocity, an acceleration can be calculated by rearranging (13) to:

$$\ddot{x} = \frac{1}{M}(F - D\Delta\dot{x} - K\Delta x) \tag{14}$$

The calculated acceleration $\ddot{x}$ is integrated once and twice in order to provide the velocity and position reference to the robot system. To determine and understand the parameters M, D and K, a physical interpretation of (13) is introduced as the motion of a spring-mass-damper system, shown in Figure 8.

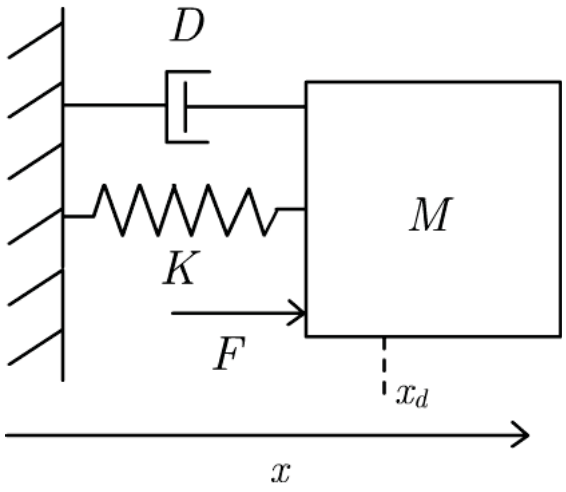

Figure 8: Spring-mass-damper system.

2.4 Recursive least squares algorithm

The recursive least squares (RLS) algorithm, adaptively estimates the filter coefficients in a linear model, by recursively minimizing the least squares error. A linear model is given by

$$y(t) = \boldsymbol{\phi}^{\boldsymbol{T}}(t)\boldsymbol{\theta}(t) + v(t) \tag{15}$$

where $\boldsymbol{\phi}(t)$ is the input vector to the filter, $\boldsymbol{\theta}(t)$ the filter coefficients, $y(t)$ the measured output of the filter and $v(t)$ an unknown noise source. The RLS algorithm is defined by [7] :

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + \boldsymbol{K}(t)(y(t) - \boldsymbol{\phi}^T(t)\hat{\boldsymbol{\theta}}(t-1)) \tag{16}$$

$$\boldsymbol{K}(t) = \boldsymbol{P}(t-1)\boldsymbol{\phi}(t)(I + \boldsymbol{\phi}^T(t)\boldsymbol{P}(t-1)\boldsymbol{\phi}(t))^{-1} \tag{17}$$

$$\boldsymbol{P}(t) = (I - \boldsymbol{K}(t)\boldsymbol{\phi}^T(t))\boldsymbol{P}(t-1) \tag{18}$$

where $\hat{\boldsymbol{\theta}}$ is a vector containing the estimates of the "real" filter coefficients, $\boldsymbol{P}$ the covariance matrix and $\boldsymbol{K}$ a gain matrix. To initialize the algorithm, $\boldsymbol{P}(0)$ is set to the unit matrix, and both $\boldsymbol{K}(0)$ and $\hat{\boldsymbol{\theta}}(0)$ are set to zero vectors.

3 Simulations

3.1 Models

As mentioned in Section 1.5.1, the robot system has internal controllers for each joint, accepting a position and a velocity reference as input. The robot model must therefore describe the controlled system for each joint separately. To simplify the modeling, all joints are considered equal and their transfer functions from position reference to position and velocity reference to velocity are identical for each joint. The model transfer functions are defined as:

$$G_{pos} = \frac{560s + 200000}{s^3 + 140s^2 + 10560s + 200000} \tag{19}$$

$$G_{vel} = \frac{28s + 10000}{s^3 + 140s^2 + 10560s + 200000} \tag{20}$$

By assuming that the knife can be modeled as a two-dimensional contour, defined by an arbitrary degree polynomial, the simulations can be reduced to a two-dimensional problem. To make sure that the simulation model works for all knife shapes, different polynomials were tested. Figures 9 and 10 show the two most frequently used knife models.

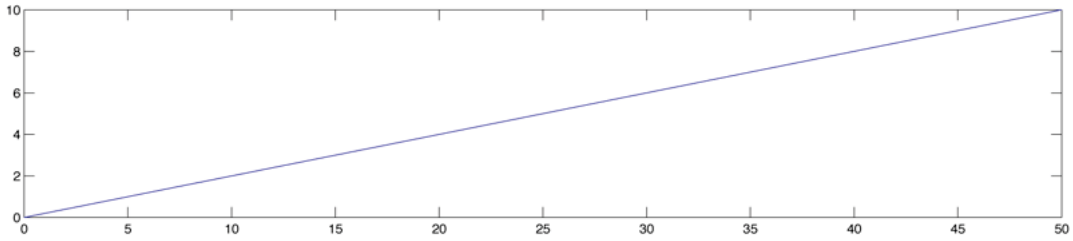

Figure 9: First order knife model.

To simulate a force measurement, the knife model finds the shortest distance between the knife and the tool. The shortest distance is always normal to the contour of the knife [8], and the same is true for the force that arises once contact is achieved. When the distance crosses zero, the tool is thought to deform the knife contour, and a force is modeled using (8), where x is the distance from the undeformed contour. The spring constant k can be chosen freely, but a too small spring constant will require the tool to deform longer into the knife contour to fulfill the force reference. This will create a displacement error in the simulated identification, thus a large k is to be recommended. Using the calculated force, the torque can be defined by taking

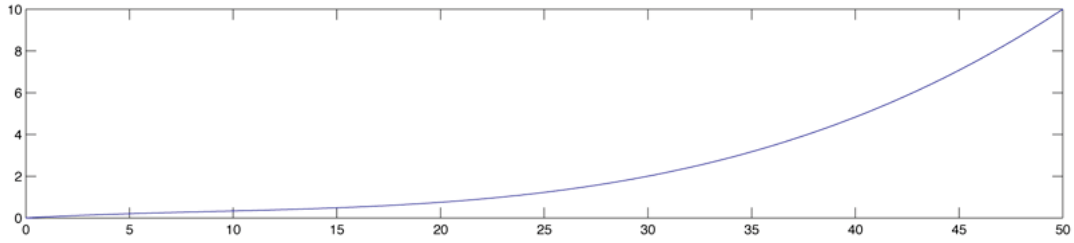

Figure 10: Third order knife model.

the vector product between the vector from the frame origin to the point of contact, and the force.

Finally, the calculated force and torque need to be transformed to the sensor frame, which is where the real robot will receive its measurements. Figure 11 shows how the force is calculated.

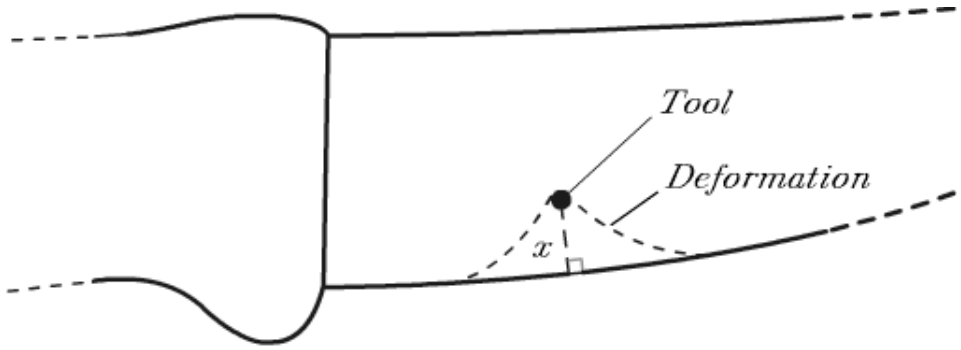

Figure 11: Simulation of force through deformation.

For simplicity in simulations, the sensor frame was set to coincide with the flange frame. Also, the tool used in the simulations was assumed to be just a line, instead of a cylinder with a certain radius.

Figure 40 shows an example of what a Simulink model can look like. The first red rectangle marks the controller block, in this case a hybrid controller, which uses the contact force measurements and force reference to calculate and output a reference velocity in Cartesian coordinates. This velocity is sent to the next block, marked by the second red rectangle, where it is transformed by the inverse-jacobian to joint angle velocities. The third and final rectangle marks the external outputs that control the robot, a joint angle and a joint angle velocity reference. In a simulated model, these outputs would instead lead to a block with a model of the robot.

3.2 Shape identification with knife fixed off-robot

In this scenario, the knife is fixed parallel to the base frame y-axis, with the knife's cutting edge facing the positive base frame z-direction. The tool is attached to the sensor, parallel to the base frame x-axis, and the TCP frame is defined arbitrarily along the tool.

The knife is placed so that it is not in contact with the tool initially and thus, when using the hybrid control scheme, the robot will start searching in a given direction. In this case, the search direction is set in the negative z-direction in the base frame. Once contact is achieved, i.e. when the force measurement from the force sensor does not equal zero, the controller switches from motion to force control. The force controller, which is chosen to be a PI controller with the transfer function

$$G_{PI} = K(1 + \frac{1}{sT_i}) \tag{21}$$

where K and T_i are tunable parameters. The PI-controller acts on the error between a given force norm reference and the measured force norm. The control signal from the force controller is multiplied with the normalized force measurement so that the control is exercised in the direction of the force. The force norm reference has to be positive so that the tool does not lose contact with the knife.

Once the knife is in contact with the tool, a new search direction is defined every sample period, described as a vector perpendicular to the force vector. Since the force vector is normal to the knife contour, the search direction will then be equal to the tangent of the knife contour. By using this kind of variable search direction combined with a force controller, the robot will follow the contour without losing contact with the knife. The robot movement is stored and will later be used for the grinding.

3.3 Shape identification with knife attached to robot

The knife is now instead fixed on the robot, with the blade assumed to be parallel to the flange frame xz-plane and the cutting edge facing the positive flange frame x-direction. The tool is fixed parallel to the base frame y-axis, somewhere below the attached knife.

This identification could be done in the same manner as in the previous scenario, but in order to prepare for the grinding procedure, it is decided that the identification should be modified. The modification consists of reorienting the knife during the identification so that the knife surface in contact is parallel to the tool surface. In terms of force, this is equal to a contact force

that only has a component in the base frame z-axis. The modification is demonstrated in Figure 12.

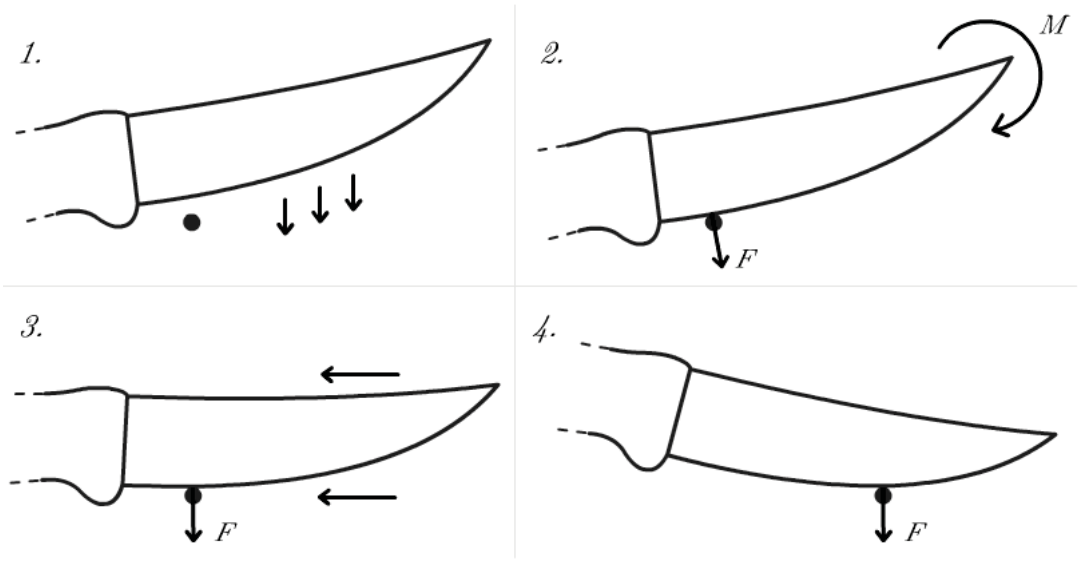

Figure 12: Knife identification with reorientation.

Just like in the previous scenario, the robot searches in a given direction until contact with the tool is achieved, at which point the force controller and a variable search direction is engaged. The reorientation is done by another controller, which uses the normalized force in the base frame to achieve the correct orientation. Since the force measurement is normalized, a force component equal to one in a direction means that the force vector is directed *only* in that direction. Knowing this, the force reference to the orientation controller is set to one. The controller, which is chosen to be proportional, uses the difference between the reference and the normalized force component in the z-direction of the base frame as input, and gives a torque as output.

The reorientation is to be done without interfering with the identification trajectory. If the reorientation is done around one of the axes of the flange frame, the point on the knife in contact with the tool will move and contact might be lost. Instead, the reorientation should be done around the actual point of contact, which is defined as the TCP. In simulations, the tool position is user-defined and can be used to reorient the knife. In the real robot setup, the tool position will not be known and it is therefore assumed that the tool position is unknown in simulations as well. An estimate of the TCP position is needed, and it can be obtained from the torque and force measurements.

If the knife is assumed to be just a straight contour (see Figure 13 for graphic representation), the following relation holds:

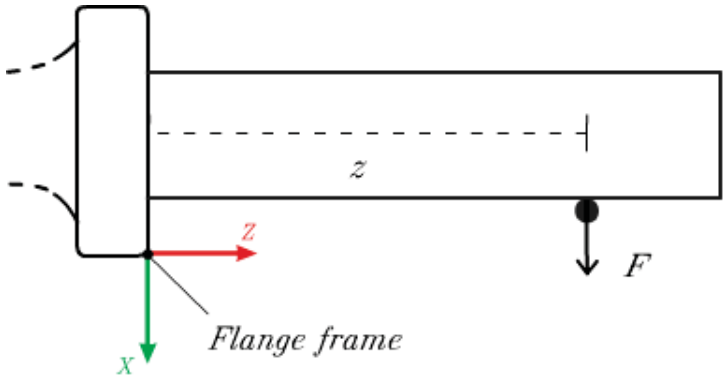

Figure 13: One-dimensional knife model in contact with tool.

$$M_y = F \cdot z \tag{22}$$

where M_y is the torque around the y-axis, F the force at the contact point and z the distance in the z-direction from the contact point to the sensor frame origin.

With this knife model, the contact point can easily be calculated using (22). Unfortunately, a real knife can hardly be modeled as in Figure 13, thus a method to find the contact point for an arbitrary knife contour is needed.

A different knife model subjected to a force is shown in Figure 14.

The force has now two components, f_x and f_z, which both contribute to the torque M_y around the y-axis, which is given by:

$$M_y = f_x \cdot z + f_z \cdot x \tag{23}$$

It is clear that (23) is an under-determined equation as both x and z cannot be calculated given only M_y, f_x, and f_z. Since the motion of the robot is known, (23) can be modified into a set of equations in order to estimate the coordinates when the knife first comes in contact with the tool, called x_0 and z_0.

$$M_y(t) = f_x(t) \cdot z_0 + f_z(t) \cdot x_0 \tag{24}$$

$$M_y(t+1) = f_x(t+1)(z_0 + \Delta z(t+1)) + f_z(t+1)(x_0 + \Delta x(t+1)) \tag{25}$$

where $\Delta x(t+1)$and $\Delta z(t+1)$ describe how much the robot has moved in each direction since contact was achieved, at the current time. Introducing

$$M'_y(t+1) = M_y(t+1) - f_x(t+1) \cdot \Delta z(t+1) - f_z(t+1) \cdot \Delta x(t+1) \tag{26}$$

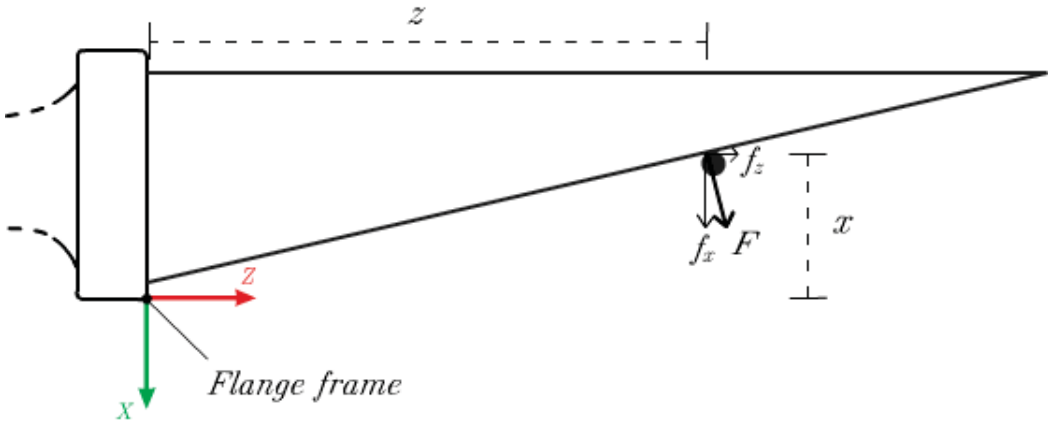

Figure 14: Two-dimensional knife model in contact with tool.

(25) is rewritten to:

$$M'_y(t+1) = f_x(t+1) \cdot z_0 + f_z(t+1) \cdot x_0 \tag{27}$$

By using an RLS algorithm together with (27), the values of x_0 and z_0 can be estimated. Comparing and rearranging (27) to match the form of (15):

$$M'_y(t+1) = \begin{bmatrix} f_x(t+1) & f_z(t+1) \end{bmatrix} \begin{bmatrix} z_0 \\ x_0 \end{bmatrix} + v(t+1) \tag{28}$$

The RLS algorithm requires excitation to work properly, if the measurements at time n and $n+1$ are identical, the equations will become singular, and thus the measured values that are input need to vary to some degree. This can be achieved by moving the knife until it is in contact, and then increasing the force until it reaches a given threshold. The RLS estimate is used to calculate the location of the tool in the base frame, which will give the desired TCP coordinates after the required frame transformations. Since the TCP is fixed in the base frame, the estimate is used to reorient the knife throughout the whole identification. As soon as the estimate from the RLS is ready, the TCP position in the flange frame is logged in order to be used as reference to the grinding procedure. Since the knife is fixed in the flange frame, the logged data will not be affected by the reorientation.

3.4 Hybrid control design

In order to get the best control performance, the parameters K and T_i of the PI-controller from (21) need to be tuned. To do this, a model describing the robot Cartesian position is needed, since the controller acts in Cartesian coordinates. An approximate model can be derived by assuming that the Cartesian position in one dimension can be described by the dynamics of a single joint, for small angles. Following this assumption, the transfer function G_{pos} from position reference to position is simply given by (19). Since the controller acts on the force norm error, the position is multiplied by the tool stiffness k_f in order to obtain the resulting force (according to (8)), assuming that the knife is in contact with the tool. Also, the control signal output from the controller is defined as a velocity reference, and it therefore needs to be integrated to a position reference. The system can be represented as a block diagram, shown in Figure 15.

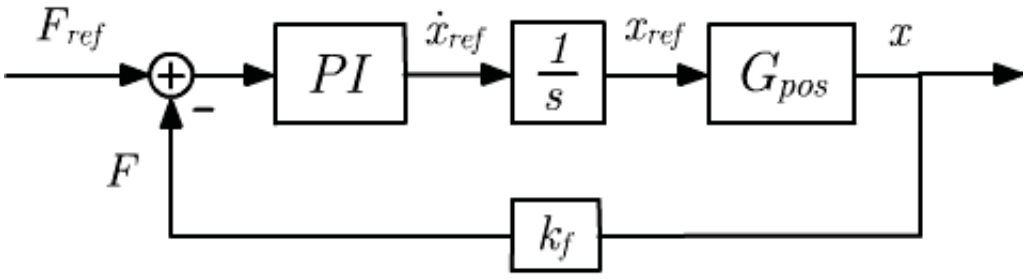

Figure 15: Force control block diagram.

Since the process is a fourth-order system, arbitrary pole placement with a PI-controller is not possible. With an initial choice of $K = 1$ and $T_i = 2$, the poles and zeros of the closed loop system are calculated and shown in Figure 16. From Figure 16, it is obvious that one of the complex-conjugated pairs of poles are poorly damped at $\zeta = 0.11$, which will result in an oscillative step response. Decreasing the influence of the integral part by setting $T_i = 10$ proves useless, the damping is only increased to $\zeta = 0.12$ while rendering the system slightly slower. By instead lowering the gain, the system will again become slower, but also a lot better damped. Choosing $K = 0.5$ and $T_i = 2$, the damping increases to $\zeta = 0.33$. Lowering the gain further to $K = 0.4$, the damping increases to $\zeta = 0.42$, and it becomes clear that there is a trade-off between controller speed and damping. Which is preferable depends on the application, and it will be discussed in the results sections.

This analysis is based on a tool stiffness set to $k_f = 50\ N/mm$. A material

that is less stiff will result in a less oscillating behaviour and thus easier control. The PI-controller parameters may therefore need some adjusting depending on what materials are used.

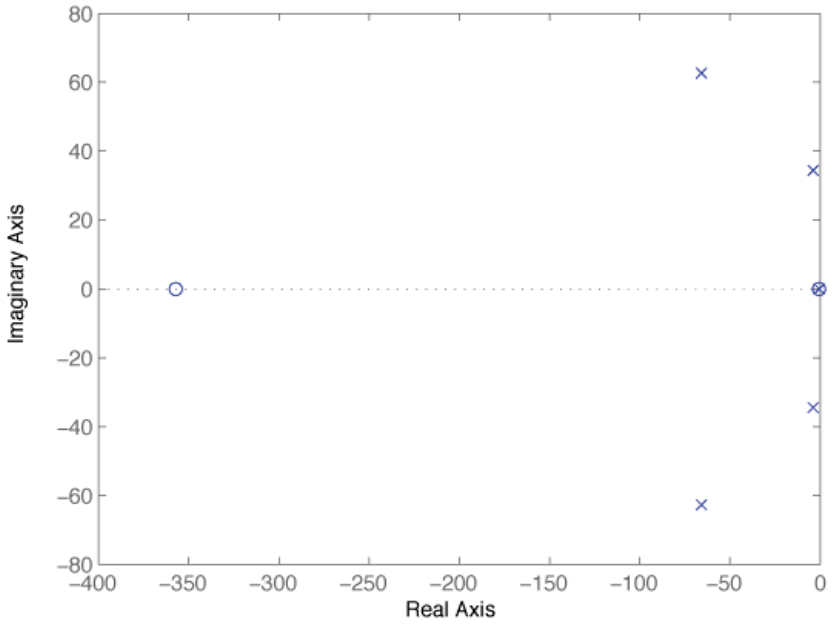

Figure 16: Pole-zero plot for the closed loop system with $K = 1$ and $T_i = 2$.

3.5 Shape identification using impedance control

Using the same setup and model as in the previous section, the hybrid force controller is now replaced by an *impedance controller*. Since the tool position is unknown, it is not possible to define a desired position of the knife. This means that K in (14) is set to zero at all times, the spring effect in Figure 8 is disabled. It is, however, known in which direction the tool is located, and thus a desired velocity can be defined. The impedance controller should act in all three dimensions, and therefore (14) is modified accordingly:

$$\ddot{\boldsymbol{n}} = \frac{1}{M}(\boldsymbol{F} - D\Delta\dot{\boldsymbol{n}}) \tag{29}$$

where $\boldsymbol{n} = \begin{bmatrix} x & y & z \end{bmatrix}'$.

Once the controller is given a desired velocity vector $\dot{\boldsymbol{n}}_d$, the robot will move in the desired direction until a force is measured. At this point, the desired velocity vector is changed to the vector perpendicular to the force

measurement, so that the tool will follow the knife contour. In order to keep a constant force throughout the identification, a force reference vector $\boldsymbol{F}_{ref}$ is introduced to (29):

$$\ddot{\boldsymbol{n}} = \frac{1}{M}(\boldsymbol{F} - \boldsymbol{F}_{ref} - D\Delta\dot{\boldsymbol{n}}) \tag{30}$$

Recalling the previous section, the hybrid controller acts on the force norm given a force norm reference, and the control signal manuevered the robot in the direction of the force. The impedance controller acts in each direction separately, thus to keep the force norm the same as the reference norm, $\boldsymbol{F}_{ref}$ is defined as:

$$\boldsymbol{F}_{ref} = \frac{\boldsymbol{F} \cdot f_{ref}}{||F||} \tag{31}$$

The parameters M and D are chosen as the desired physical properties of the spring-mass-damper system in Figure 8.

3.6 Simulation results

In all three force control simulations, the knife shape is identfied accurately, whether the knife is reoriented or not. The identified shape is shown in Figure 20. There is however, some differences in the control performance.

Following the analysis in Section 3.4, setting the PI-control parameters $K = 1$ and $T_i = 2$ will result in a fast but oscillating step response, which is is verified in simulations and shown in Figure 17. Instead using $K = 0.4$ and $T_i = 2$ results in a well damped and still quite fast system, see Figure 18. Lowering the gain further to $K = 0.3$ almost fully dampens the oscillations but renders the step response very slow. For the application of identifying knife shapes, the use of a highly damped controller is preferable to a faster controller since the main objective is to keep a *constant* force reference. Also, oscillations when identifying a shape will result in a less exact recorded shape. Thus, the controller with $K = 0.4$ and $T_i = 2$ is chosen to be evaluated in experiments.

In the simulations using impedance control, the results were not as good, most likely due to the fact that the impedance control parameters were difficult to tune. Since the hybrid controller performed satisfactory in simulations, it is decided that impedance control is not to be evaluated in experiments.

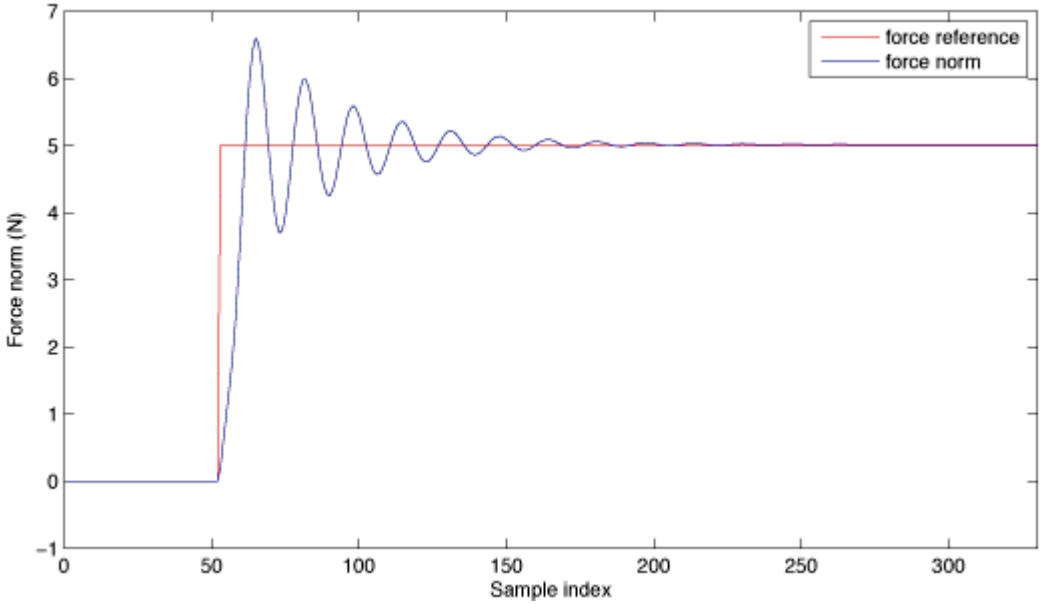

Figure 17: Hybrid force control with $K = 1$ and $T_i = 2$.

4 Experiments

4.1 Performing experiments

There are a number of steps involved in performing an experiment on the real robot system. Since unexpected output from the controller can be dangerous both to the robot system and its environment, thorough testing of the controller in the *submit*-mode is important, even if the controller is working in simulations. To test the controller, it is loaded into the Opcom by entering the path to where it was built from Simulink, and then set into the submit-mode. If force measurements are required, the sensor is activated by entering a command line in a terminal window, that connects the measurements computer with Opcom. The variable *fswitch* is set to 1 in Opcom so that the controller starts receiving measurements and executes. In order to store all the measurements that Opcom receives, another command line is entered, logging all measurements for a given period of time. Once the logging has started, the desired tests can be performed, such as examining the controllers behaviour when the robot is jogged, or when the tool is subjected to a force. The logged data is converted and loaded into Matlab, and can be analyzed to make sure that the controller is stable and behaves in the desired manner. When this is achieved, the next step is to switch to the *obtain*-mode, to allow

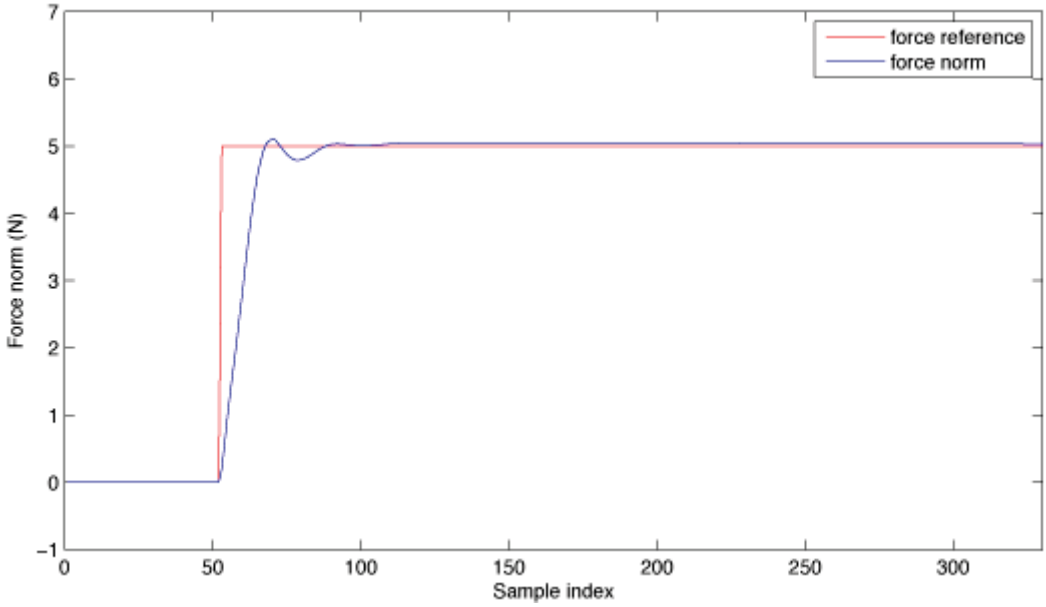

Figure 18: Hybrid force control with $K = 0.4$ and $T_i = 2$.

the controller to not only receive but to send signals. Before going into the obtain-mode, one must make sure that the robot can move unobstructedly and that the robot's workspace is clear of people.

The dead man's switch on the FlexPendant must be pressed at all times during the experiment, and the brakes must be released before the fswitch is set to 1. If this is done correctly, the controller is engaged and the robot system will receive position and velocity references, hopefully moving the robot in the desired direction. If the robot should move in an unexpected manner, the dead man's switch is simply released and the robot will stop immediately.

In the experiments presented in this Section, the knife is chosen to be fixed on the robot at all times. As a precaution, the experiments were firstly performed using a dummy knife instead of a real knife, see Figure 21. This was done to reassure that the experiment's general behavior was satisfactory, because of the obvious dangers of having a knife mounted on the robot. A cardboard box was used as a tool together with the dummy knife in order to avoid something breaking, should the robot move in the wrong direction the box will simply be deformed.

Once the experiments have been performed and found to give satisfactory results, the real knife can finally be mounted on the robot. The tool is also

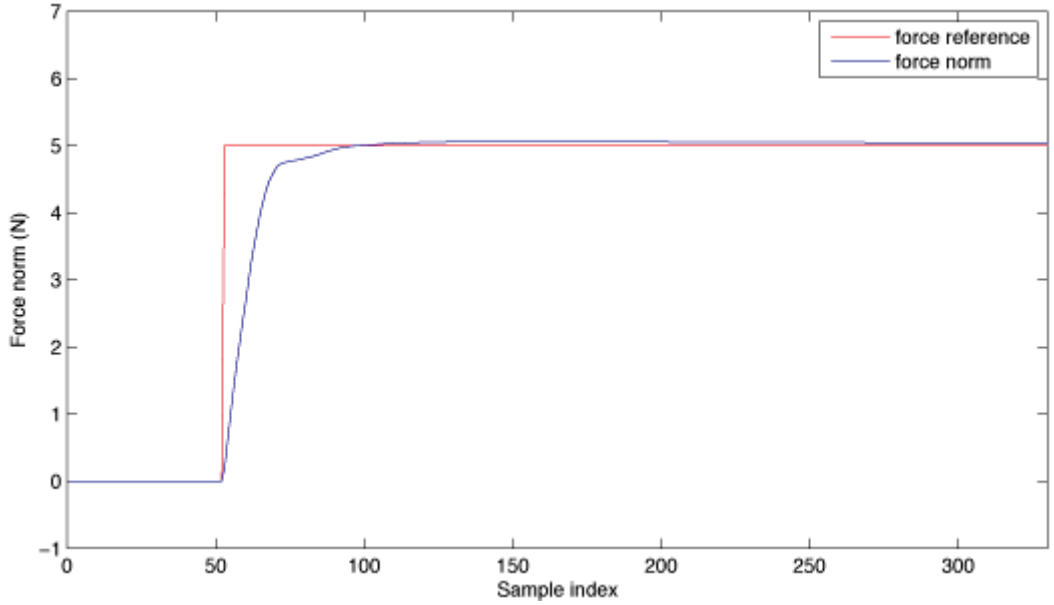

Figure 19: Hybrid force control with $K = 0.3$ and $T_i = 2$.

replaced to a much stiffer tool, see Figure 25 for an image of the setup.

4.2 Sensor calibration and gravity compensation

As mentioned in Section 3.1, the sensor frame was set to coincide with the flange frame to simplify simulations. On the real robot, the sensor is mounted directly on the flange, and the sensor frame position and orientation must be estimated.

The position of the sensor is simply measured, and it is assumed that the sensor is only offset in the flange frame z-direction. The position offset is measured from the end of the flange to the surface of the sensor. However, according to [9], the sensor coordinate system is located 20 mm from the sensor surface, thus the distance is compensated.

To determine the orientation of the sensor frame, it is firstly assumed that the sensor frame z-direction is aligned with the flange frame z-direction, and that the robot is in its home position. Secondly, a weight hanging by a thread is attached to the sensor, as close to the center as possible. Several force measurements are then executed, with different orientations around the z-axis of the flange. In order to only measure the force that arises from the hanging weight, the force sensor measurements needs to be reset before attaching the weight. When the robot is in its home position, the sensor frame z-axis

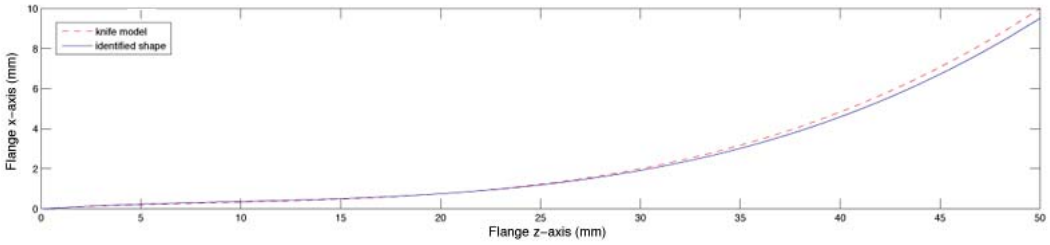

Figure 20: The shape of the knife, identified in simulations.

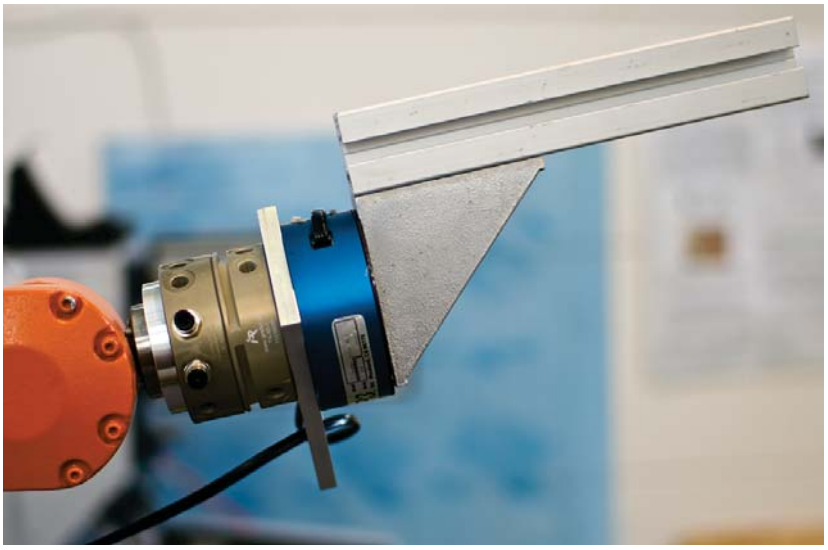

Figure 21: The dummy knife mounted on the robot.

is perpendicular to the gravitational field, thus the force measurements will only have components in the sensor x- and y-directions. Using the measurements, a least squares estimate of the deviation angle from the flange frame is calculated. To describe the sensor frame, the estimated angle is converted to a rotation matrix, and together with the position offset, a T44 transformation matrix is formed. To verify that the transformation is correct, the same weight is hung on the sensor once again. The force measurement is transformed from sensor to flange frame, and if the force is only directed in the flange x-axis, the transformation is correct.

As mentioned earlier, the force measurements are reset to only measure the external influence, and not the weight of the tool. This works fine as long as the orientation of the sensor is kept the same throughout the experiment.

During the shape identification and grinding, the robot is reoriented and it is therefore necessary to compensate for the gravitational influence.

The objective of gravity compensation is to keep the force measurements equal to zero when not in contact, no matter what orientation the robot has. To do this, the mass of the tool is required. The tool mass is easily determined by reorienting the tool and measuring the force in the sensor z-axis and compensating for the changing angle. Once the reorienting is done, a mean value of all the measurements gives the estimated mass of the tool. The uncompensated force measurements, tool mass and the flange T44 matrix is fed through a predefined Simulink gravity compensation-block from the extctrl library, and the compensated force measurement is given from the block.

4.3 Orientation calibration of knife

In simulations, the knife is defined to be perfectly aligned with the desired axes. In reality, the knife will always have a slight deviation in orientation. This orientation needs to be identified in order to allow the identification and grinding to be performed correctly.

Rotation around the flange frame y-axis has already been taken care of with the adoption of the orientation controller in Section 3.3. It is therefore only necessary to identify the rotation around the flange frame x- and z-axis. This is done by first assuming that the tool used in previous simulations is fixed off-robot, parallel to the base frame z-axis. Secondly, the robot searches for the tool until the knife is in contact with the tool. With the force measurements in the flange frame y- and z-directions, the rotation around the flange frame x-axis can be calculated using simple trigonometry, see Figure 22.

The deviation angle ϕ_1 is given by:

$$\phi_1 = tan^{-1}\left(\frac{f_y}{f_z}\right) \tag{32}$$

Since the force was normal to the knife surface, it was possible to easily calculate the rotation. In the case of rotation around the flange frame z-axis, the problem is not quite as trivial. Once the knife is in contact with the tool, the force measurements given from the sensor will always be normal to the tool surface. This means that the force will always be directed in the base frame y-direction, provided that the potential force component in the flange frame z-direction is disregarded, since it does not contribute to rotation around this axis, see Figure 23.

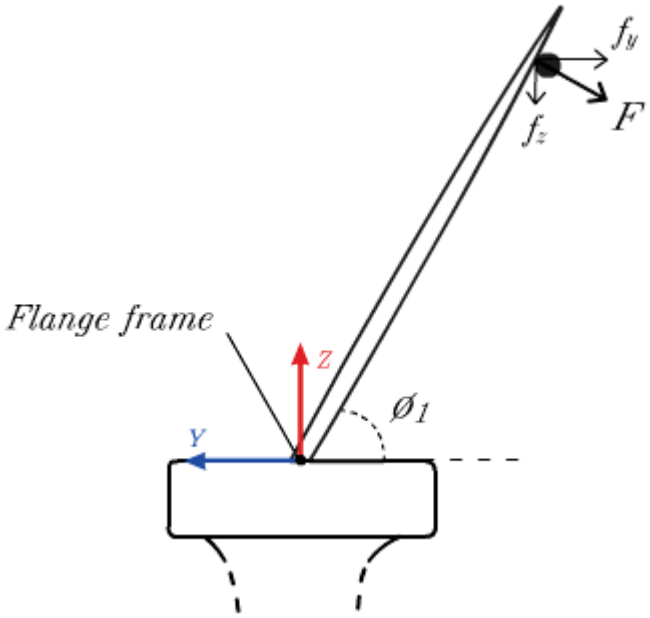

Figure 22: Simplified view of the knife fixed on the sensor, displayed in the negative base frame z-direction.

In Figure 23, the knife has been given an arbitrary rotation deviation ϕ_2 around the flange frame z-axis. However, it is from now on assumed that the deviation is *always* positive. When the knife is in contact with the tool, the sensor will give a force and a torque. If the knife is rotated around the point of contact, the other end of the knife will eventually come in contact with the tool, and the rotation will start to revolve around that point instead. Once this happens, the torque will change abruptly because of the sudden change of contact point, see Figure 24 for an example. This discontinuity in the torque is detected, and the flange orientation is recorded at that point, see Figure 24 for a graphical description of this method.

To do this, the point of contact needs to be identified in order to rotate around it. As mentioned before, the contact force will be directed normal to the tool surface. This concludes that the point of contact cannot be estimated using an RLS algorithm, since there is only one force component as excitation. If the knife is instead rotated around the known flange frame, contact will be lost. The addition of a force controller to this rotation will keep the knife in contact while it is rotating, and the desired rotation will be achieved.

Once ϕ_1 and ϕ_2 have been estimated, the robot can be reoriented so that the knife is perfectly aligned before engaging the identification.

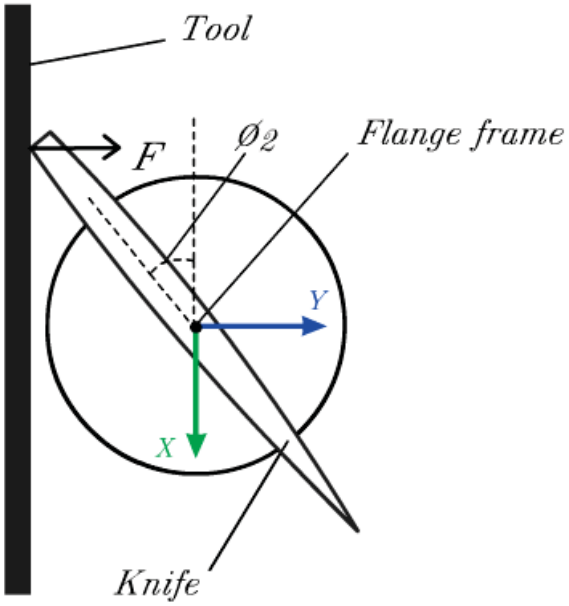

Figure 23: Simplified view of the knife fixed on the sensor, displayed in the negative base frame x-direction.

4.4 Shape identification

The model used to perform the shape identification on the real robot is basically the same as the simulation model, modified to connect with the robot system and the force sensor. Some additional modifications were however necessary to achieve optimal results. Firstly, the knife is reoriented to compensate for the deviation identified in Section 4.3, and the TCP frame orientation is defined accordingly. Further, as mentioned in Section 1.5.2, the measurements from the force sensor are known to be noisy, and it is therefore a good idea to low-pass filter the force measurements before feeding them to the controllers. Also, choosing a controller with a low gain will avoid amplification of the noise.

The shape identification was performed using hybrid control, an image of the experiment setup is shown in Figure 25.

4.5 Grinding procedure

When sharpening a knife manually, the knife is pressed with a constant force between two rotating grinding stones. The knife is then moved along its contour, while striving to keep the knife surface normal to the force, until

the blade ends. This procedure is repeated until the knife is considered to be sharp.

The grinding procedure is now to be done automatically using the robot, with the identified knife shape from the identification experiments used as reference. In order to make the robot follow the knife contour, the first set of coordinates from the recorded knife shape (TCP position) and the fix coordinates of the grinding stones are input to the inverse kinematics. The difference between the joint angles output from the kinematics forms the error for a proportional controller that controls each joint separately, moving the robot to the desired position. Once there, a discrete counter starts, sending a new TCP position reference to the inverse kinematics as the counter increases. Assuming that the controller moves the robot fast enough to reach the current position reference before receiving a new one, the robot will accurately follow the knife shape with a velocity defined by the discrete counter speed, and the sample time of the recorded knife shape.

In addition, the knife must be reoriented in order to keep the contact force normal to the knife surface. By differentiating the recorded knife shape, the angle of the knife is obtained and used to define the desired orientation of the TCP, in each point of the motion. To make sure that there is a force to keep normal to the knife surface and that it is kept constant, a force controller acting in the base frame z-direction is added. Since the force controller sends its control signal directly to the robot system, the path-following controller will try to counteract the force control, since it is under the impression that the robot has deviated from the path. The desired path's z-component is therefore continuously modified by adding the integral of the velocity reference, that is sent to the robot system from the force controller.

Ideally, if the recorded knife shape is perfectly accurate, the knife only has to start with the desired force and the perfect motion of the robot will keep the force constant. Since it may be very time consuming and difficult to record a perfect shape, the existence of a force controller is necessary, to correct for possible errors in the recorded shape. With an inexact shape the controller needs to be very fast in order to be able to compensate, resulting in a less stable system. Thus, a shape as accurate as possible is to be preferred. An image of the experiment setup is shown in Figure 26, and a close-up of the knife in contact with the grinding stones is shown in Figure 27.

4.6 Alternative methods for shape identification

The accuracy of the identified shape is very important for the control performance, and the assurance that the knife is sharpened evenly. It can therefore be a good idea to use alternative methods to extract the knife shape, either

just to verify the shape or to use as the actual reference.

An alternative method that was tested was the use of a *Heidenhain ST3078 linear encoder*. A linear encoder measures distance in one dimension with an accuracy of 1 μm, see Figure 28.

To identify the shape using a linear encoder, the robot is lowered until the encoder is almost fully pressed, and then the robot is moved linearly along the base frame x-axis. As the robot moves, the changing knife shape will allow the encoder to measure the z-component of the shape. The robot motion in the x-direction and the measurements in the z-direction from the linear encoder together form the recorded shape. Because of the high accuracy of the linear encoder, notches in the knife can be detected and give information on if the knife should be grinded specially or discarded.

Another possible method is to take a photo of the knife and use image analysis to extract the shape. A Matlab-algorithm for this was developed by Magnus Linderoth, using a picture such as in Figure 29 as input.

The algorithm uses white squares to map the knife transformation, i.e. if the picture was taken from angle and not exactly from above, the knife will look different. Also, the width of the squares are known, and can therefore be used to transform the scale from pixels to mm. The red background is used to contrast against the knife in order to simplify extraction of the contour. The result after the image analysis is shown in Figure 30.

The accuracy of the image analysis method is limited by the resolution of the camera and the distortion of the lens. One advantage over the other two identification methods is that the whole knife shape is identified, not just the cutting edge. This simplifies the decision on whether the knife can be sharpened once more or be discarded.

The advantage of using the force control identification is that it not only records the knife shape, but automatically estimates the knife position in the flange frame. The two methods described in this section only extracts the shape, and gives no information on where the knife is located. A fix transformation to the start of the shape is therefore needed for every knife.

4.7 Experiment results

4.7.1 Orientation calibration results

As expected, when the knife changes rotation point, a change in torque appears. However, when the knife is close to correctly aligned, the change in torque is not very prominent. This is most likely a result of the flexilibility of the knife, but it is easily solved by exaggerating the deviation by setting a fix rotating the knife before starting the calibration. With this solution, there

is an abrupt change that can be detected without problem, see Figure 31. Once the time of the change has been detected, it is mapped to the flange's orientation at that time, and the angle θ_2 is given. With the knife holder seen in Figure 26 that was used in the experiments, θ_2 was equal to 5° and θ_1 equal to zero.

4.7.2 Shape identification results

The hybrid force controller that was chosen from the simulations with $K = 0.4$ and $T_i = 2$, gives satisfactory results as shown in Figure 32. Increasing the gain to $K = 0.7$ gives even better performance, see Figure 33. Still, both controllers identify the knife shape more or less identical, the shape is shown in Figure 34. The limiting factor seems not to be the control performance but the force reference, since a larger force will result in more friction and thus add inaccuracy in the identification.

4.7.3 Grinding results

As mentioned in Section 4.5, the accuracy of the recorded knife shape is of great importance to the control performance during the grinding procedure. By looking at the control signal during a grinding experiment, one can get an estimate of the accuracy. From Figure 35, it is obvious that the shape is accurate except close to the tip of the knife. This can also be seen in a force norm plot, as shown in Figure 36. As opposed to the shape identification, the control performance during the grinding is very important in order to get an evenly sharpened knife. To improve the control performance and compensate for deviations in the shape, the controller needs to be faster, and thus the gain is raised to $K = 0.7$. Looking at Figure 37, this is evidently an improvement. Still, there is a remaining error towards the end of the knife, and in attempt to eliminate it the integral part influence is raised to $T_i = 1.4$. As seen in Figure 38, the error is reduced further, but the response is showing a slight increase in oscillations. The error may be completely eliminated by making the controller even faster, but at the cost of stability. It is therefore desirable to instead focus on obtaining a more exact knife shape, that will facilitate the control.

The large overshoot that is present in the beginning of all three experiments, is a result of switching controllers when the knife comes into contact. To reduce the overshoot, the switching condition is reviewed and improved, and the result is shown in Figure 39.

With some adjustments of the position and angle of the grinding machine and some minor recalibrations of the robot, the knife is after an experiment

sharpened satisfactory. There are naturally several things that can be improved to make the sharpening more than satisfactory, but the goal of this thesis is reached. See Section 5 for a discussion on possible improvements and additions.

5 Conclusions and future work

The purpose of this thesis was to investigate the possibility of identifiying an unknown knife's shape and sharpening it, using a force controlled industrial robot. Sharpening knives using an industrial robot has been done before, but with the use of position control instead of force control. Grinding using position control relies on perfect motion and calibrations to get the correct force throughout the sharpening, which requires a lot of time consuming programming. As this thesis proves, the use of force control when sharpening knives is possible and preferable, as it can correct for possible deviations and disturbances in force.

There are many improvements and additions that can be done in the future. Before performing the sharpening treated in this thesis, the knives need to be thinned in a procedure much similar to the sharpening. The knives also need to be polished at a specific angle, depending on how the sharpening was performed. Both of these procedures would be possible to implement with a force controlled robot, although the polishing will require more advanced methods with a six-dimensional motion and force control.

Another task to consider is quality control of the sharpness, either with some kind of vision-based test or by performing a destructive experiment. If it for example turns out that the knife has been grinded too much towards the end of the blade, a variable force reference or grind speed could be introduced, as a function of the knife shape coordinates. It would also be desirable to automatically determine if it is possible to the sharpen the knife once more, or if it is depleted.

To make the robot grinding suitable for the industry, the tasks need to become repeatable by designing a knife holder that can switch knife easily. Also, to make the grinding as cost-effective as possible, every robot motion should be speed optimized. To avoid having to calibrate the position of every new knife, which will be very time consuming, the knife holder needs to be very accurate.

References

[1] I. Dressler, "Force control interface for ABB s4," tech. rep., Department of Automatic Control, Lund University, Lund, Sweden, 2008.

[2] ABB, *IRB140B Data Sheet*.

[3] ABB, *ABB Home Page*, 2009.

[4] ABB, *IRC5 Data Sheet*.

[5] JR3, *JR3 100m40A Data Sheet*.

[6] L. Sciavicco and B. Siciliano, *Modeling and Control of Robot Manipulators*. New York: McGraw-Hill, 1996.

[7] K. J. Åström and B. Wittenmark, *Adaptive Control*. New York: Dover Publ., 2008.

[8] G. Sparr, *Linjär Algebra*. Lund: Studentlitteratur, 1997.

[9] JR3, *JR3 100m40AI63 Specifications*.

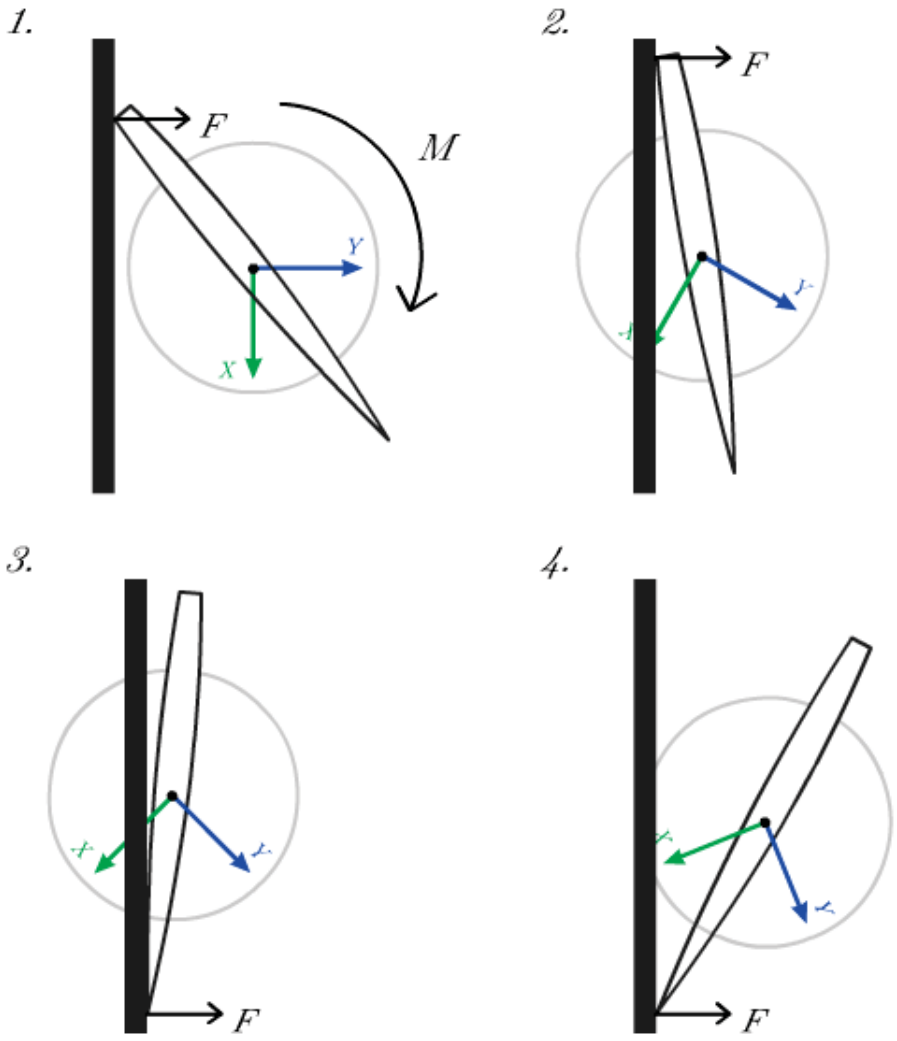

Figure 24: Method for identifying deviation around the flange z-axis.

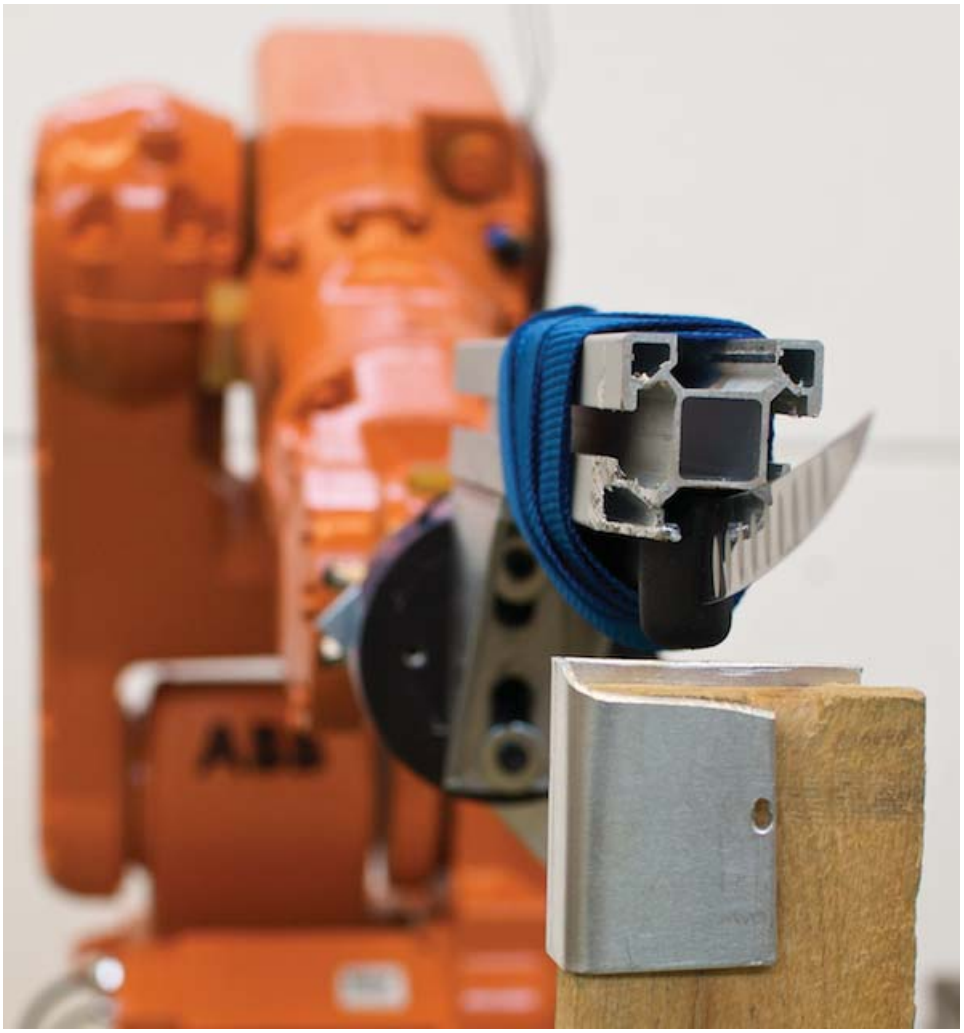

Figure 25: The experimental setup for identifying knife shapes.

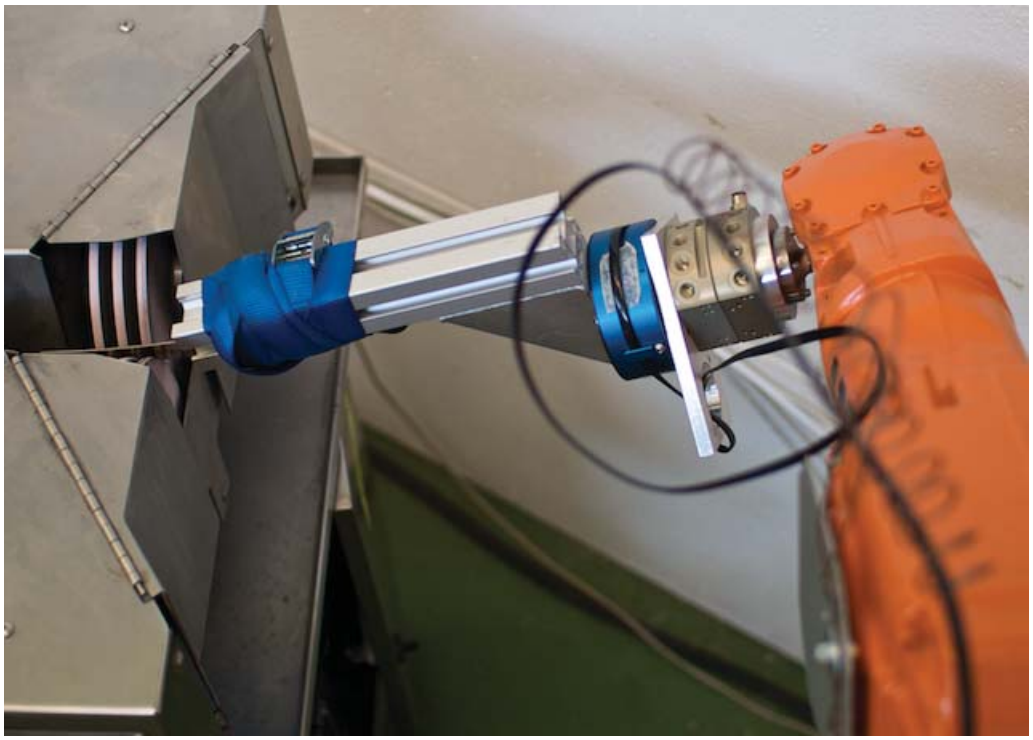

Figure 26: The experimental setup for the grinding.

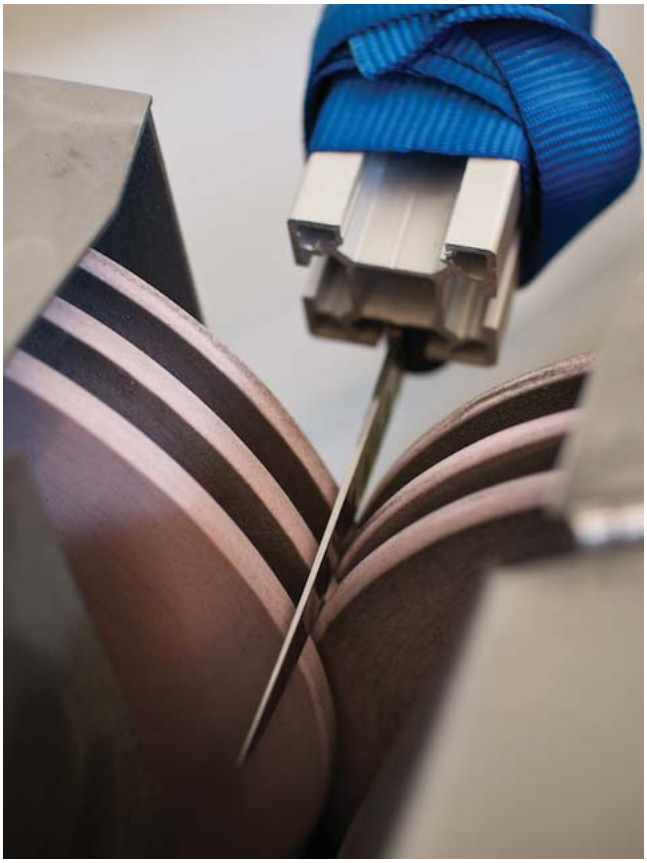

Figure 27: The knife being sharpened between the grinding stones.

Figure 28: The linear encoder in contact with a knife.

Figure 29: Input to the image analysis algorithm.

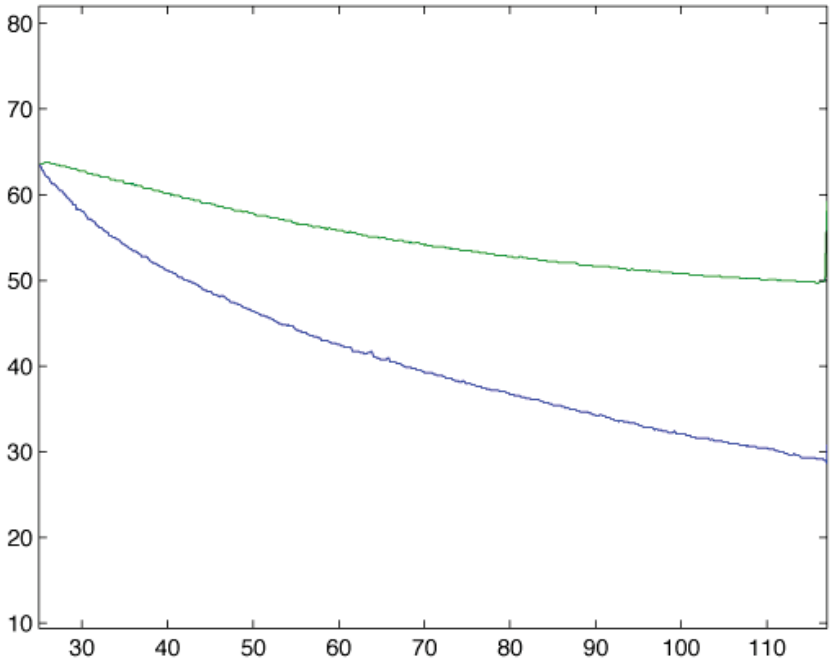

Figure 30: Output from the image analysis algorithm.

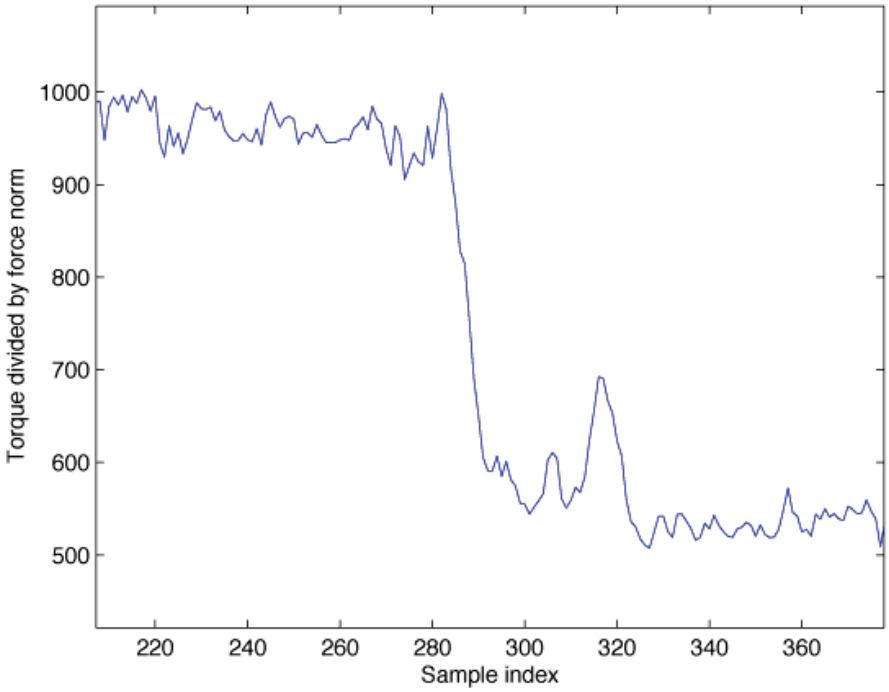

Figure 31: Plot showing the abrupt change in torque.

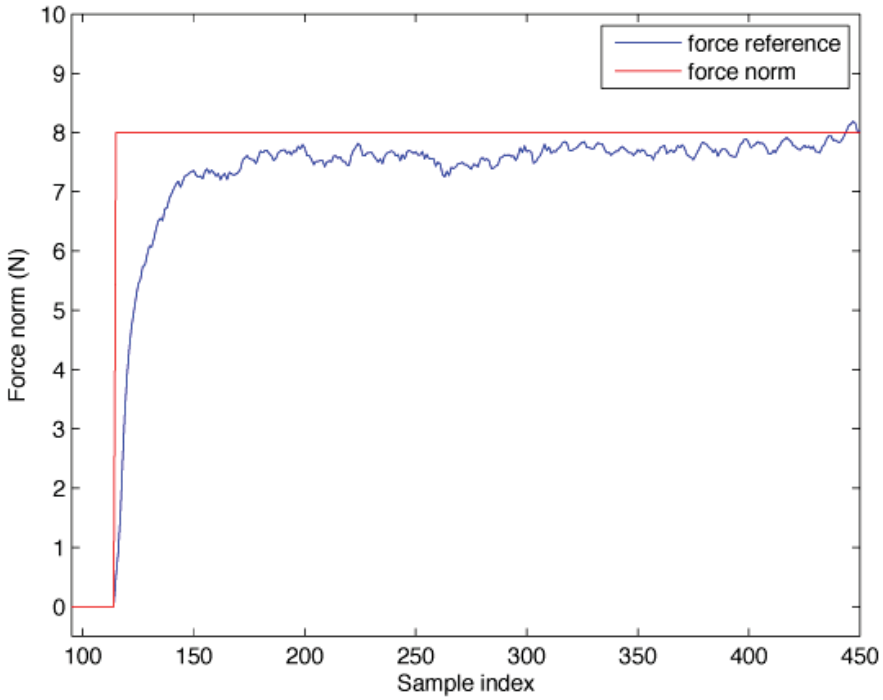

Figure 32: Shape identification force control using $K = 0.4$ and $T_i = 2$.

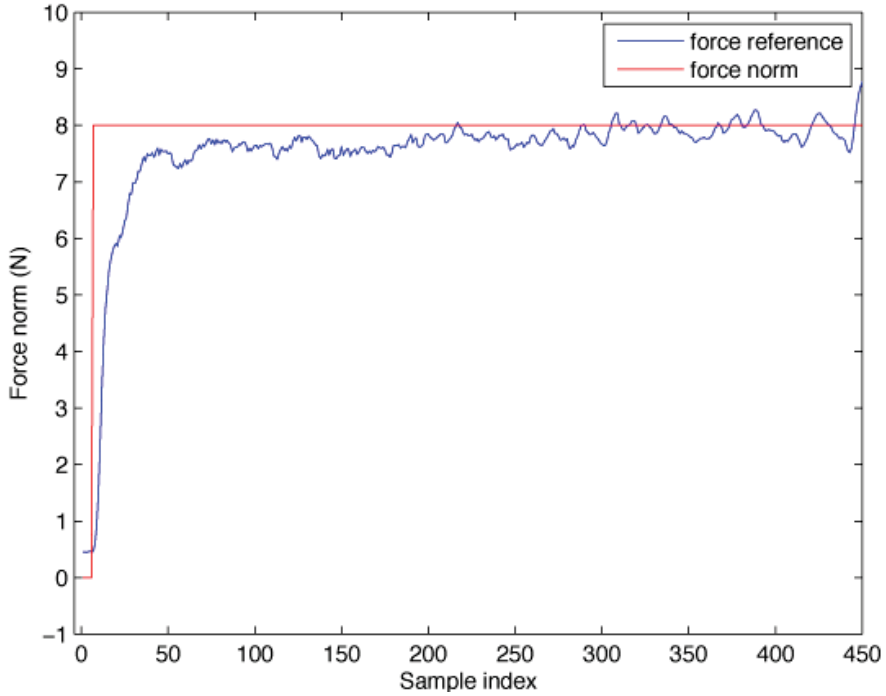

Figure 33: Shape identification force control using $K = 0.7$ and $T_i = 2$.

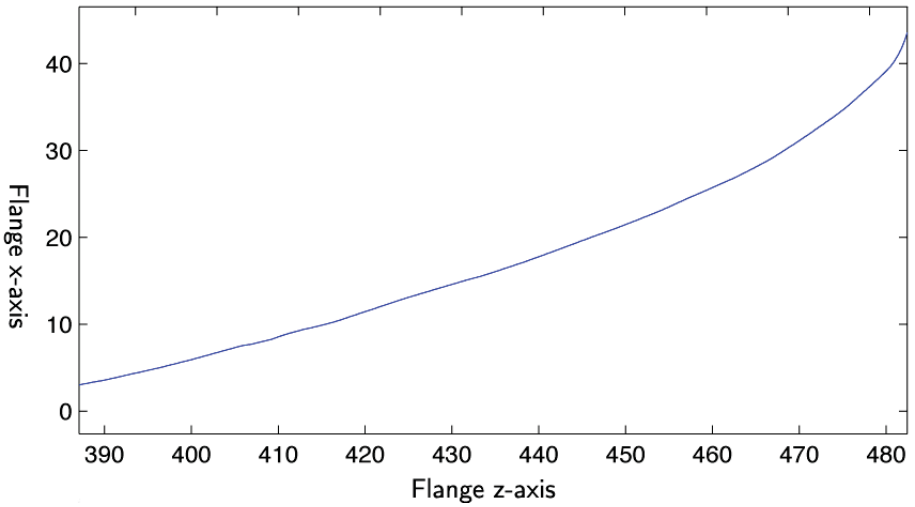

Figure 34: The identified knife shape.

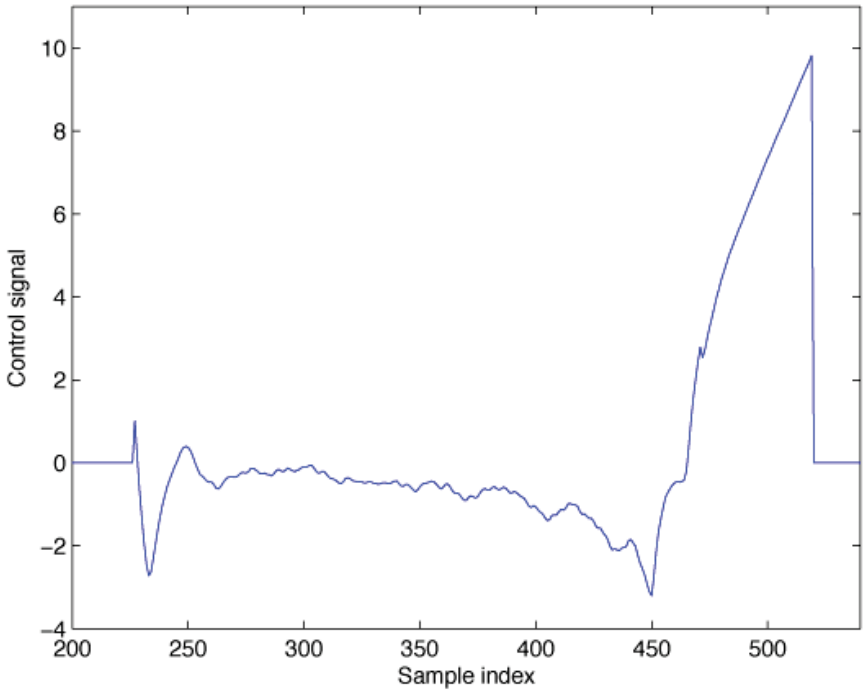

Figure 35: Control signal during grinding, using $K = 0.4$ and $T_i = 2$.

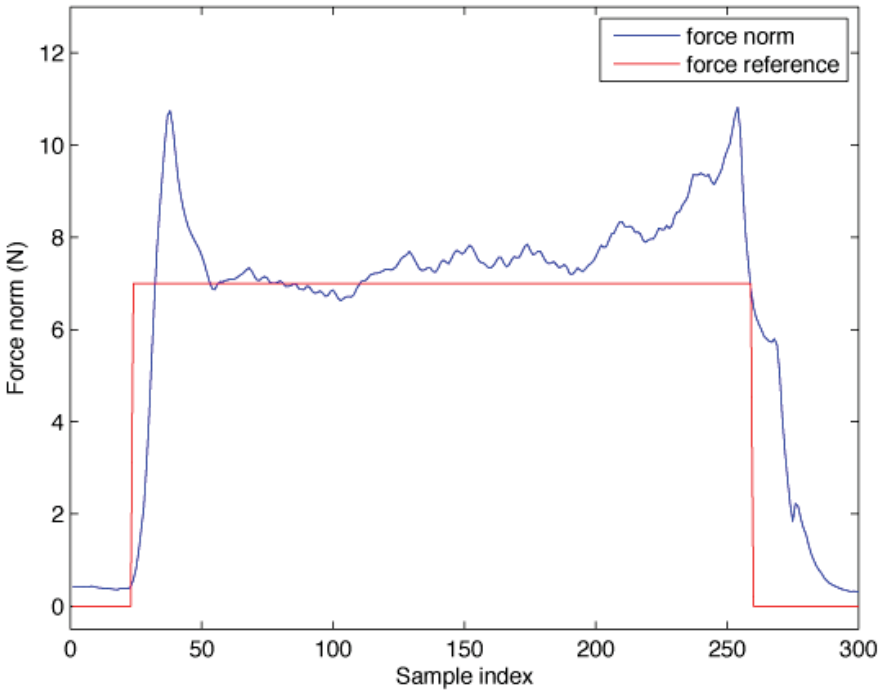

Figure 36: Force control during grinding, using $K = 0.4$ and $T_i = 2$.

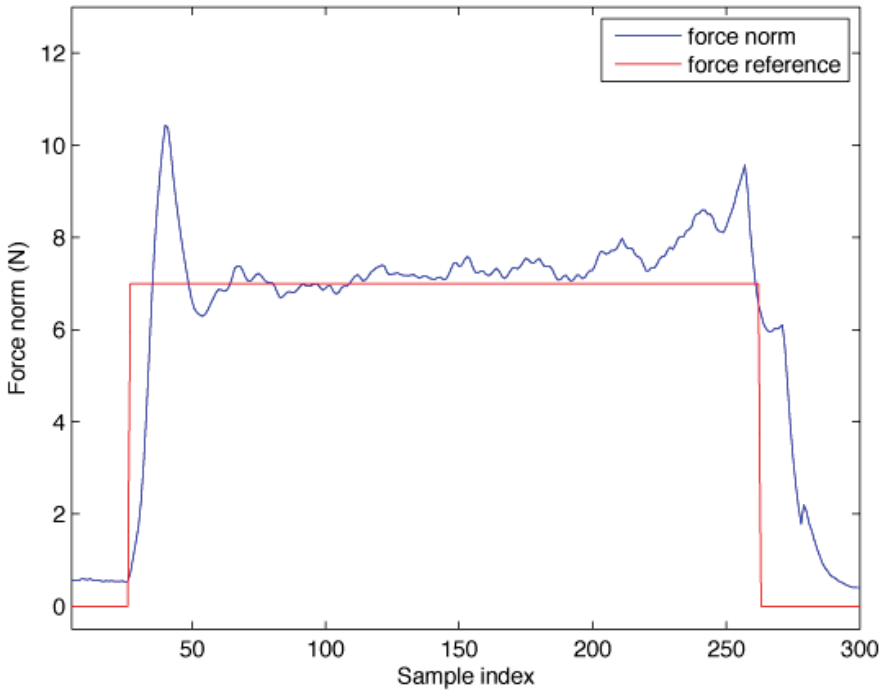

Figure 37: Force control during grinding, using $K = 0.7$ and $T_i = 2$.

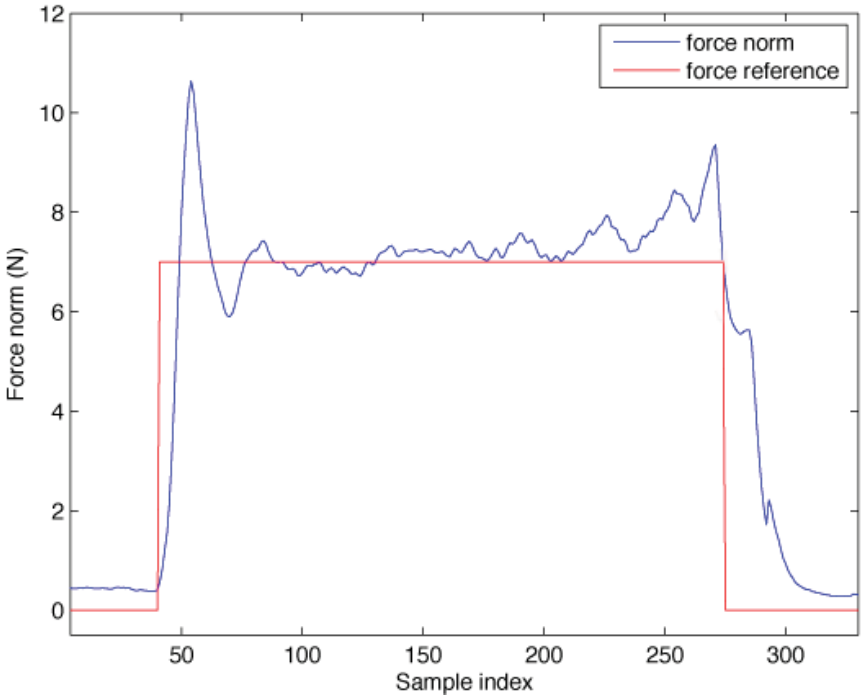

Figure 38: Force control during grinding, using $K = 0.7$ and $T_i = 1.4$.

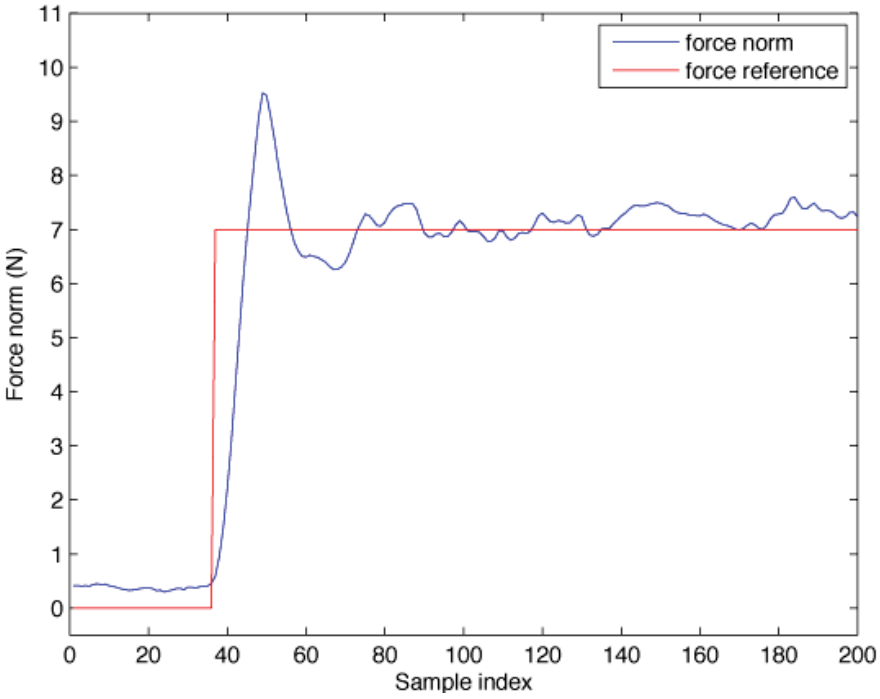

Figure 39: Attempt to reduce the overshoot.

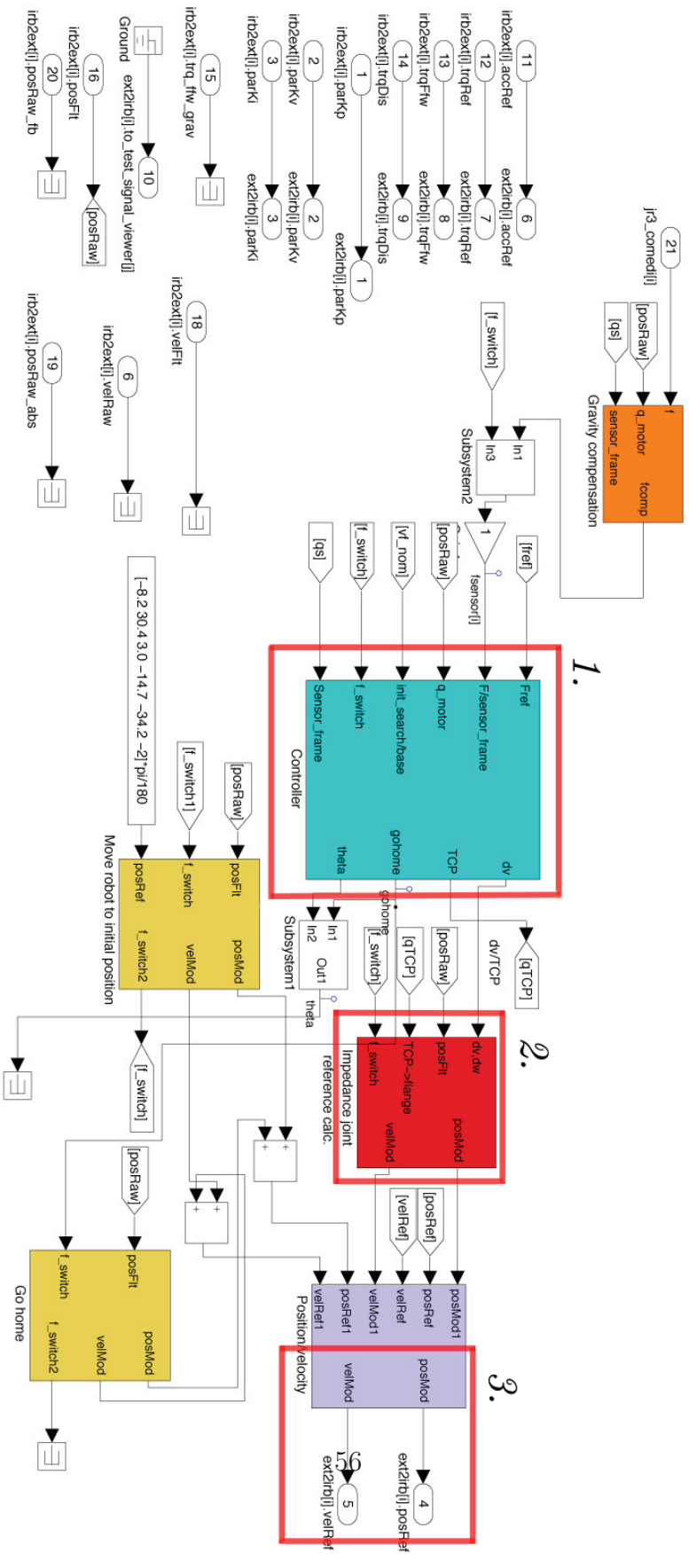

Figure 40: A simulink model example.