ISSN 0280-5316
ISRN LUTFD2/TFRT--5850--SE

Kinematics and force control for a Gantry-Tau robot

Johan Friman

Department of Automatic Control
Lund University
January 2010

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

Document name
MASTER THESIS

Date of issue
January 2010

Document Number
ISRN LUTFD2/TFRT--5850--SE

Author(s)
Johan Friman

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

Sponsoring organization

Title and subtitle
Kinematics and force control for a Gantry-Tau robot. (Kinematik och kraftreglering för en parallelkinematisk robot)

Abstract
This master thesis present a simulation environment developed in Matlab/Simulink, which can be used for control design and code generation for real-time control experiments for the Gantry-Tau parallel robot. The Gantry-Tau is a new parallel robot concept which offers great rigidity, high speeds and large workspace. The kinematics library presented in this thesis is developed for the 5 degrees of freedom Gantry-Tau prototype used at the Robotics Lab, Department of Automatic Control, Lund. A force control design using dual sensors is presented. One force sensor si used in an operator lead-through design using an impedance control strategy. The other force sensor is used to protect the tool of the robot, by preventing large contact forces building up, force which potentially harmful for the tool and robot.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
52

Recipient's notes

Security classification

Contents

1 Acknowledgments

First and foremost I want to thank my supervisor Anders Robertsson. Widely regarded the most busy person at the department, he has always taken his time to help me, even passing by the robotics lab at weekends to help out. Much of the work in this thesis is based on researches conducted by the Ph.D student Isolde Dressler and I want to thank her for always answering all my questions and supporting my work. Anders Blomdell in depth knowledge of C and Fortran has been of great help at several occasions, particularly when exploring different linear algebra packages and investigating timing constrains for the Opcom application. During the work with this thesis much work was performed remotely and that would not have been possible without the help from Leif Andersson. He also supported me in questions regarding LaTeX.

2 Introduction

2.1 Problem formulation

The topic of this master thesis is to develop a simulation environment in Matlab/Simulink which also should be used for code generation for real-time control experiments in a force feedback scenario, controlling a Gantry-Tau robot using two force sensors. A lead-through scenario will be used to test the implemented controller. One of the force sensors is used to implement the lead-through controller and the other force sensor is use as a safe-guard, making sure the tool of the robot is not subjected to too big contact forces. In the implementation, different types of force controllers will be investigated and validated using a new type of parallel kinematic robot, the Gantry-Tau.

2.2 Motivation of work

In a lead-through scenario a human operator controlls the robot motion by applying a force, measured by a force sensor, to some kind of grip or handle that may or may not be attached to the robot. A big advantage using lead-through instead of conventional joystick from the operators point of view is that it is a very intuitive way of moving the robot. However, if no means of safety measurements are implemented in the robot system and the robot is only controlled by the operator's command, situations can occur that may damage the tool attached to the robot, the robot itself or in the worst case scenario harm the operator. A very basic security measure would be to limit the workspace for the robot, but objects put inside the workspace or people entering the workspace would still lead to potential hazards. An approach used in this thesis is to use an extra force sensor that measures the force acting on the tool of the robot. Using this information appropriate measures can be taken if the contact forces grow to large. Another application of this extra sensor is in situations where a constant contact force is desired, e.g. in cutting and grinding applications where unknown surfaces are being processed. To summarize this section the main motivation is to implement a control system that ensures that large undesired contact forces are avoided as a mean of ensuring robot and operator safety. The implementation may also be useful in applications involving contact forces in general.

2.3 Outline

In Chapter 2 a short introduction to the subject of this theses is given. Chapter 3 introduces different type of industrial robots and presents the Gantry-Tau parallel robot concept. Chapter 4 presents basic theory, with purpose to introduce the reader with preliminaries on the subject of robot modeling and force control. The chapter is also recommended for readers with experience of the subject as the chapter introduces frames and concepts used later in the report. Chapter 5 describes the methods used in the thesis. In Chapter 6, the developed

Simulink kinematics library for the GTP is presented. Chapter 7 and Chapter 8 presents simulation and experimental results respectively. A summary of the master thesis is presented in Chapter 9 and finally conclusion and future work are found in Chapter 10.

3 Industrial robotics

3.1 Brief History

For some decades now industrial robots, *IRB*s, have been used in industry, especially as a part of industrial production. For a period of time there was a strong belief that the industrial robots would totally replace the need for human labor in production. High expectations were put in the IRBs and advantages were seen, for example there would be no need for lighting when robots perform the work instead of the human counterpart. As of today we know that this scenario has not played out. The first industrial robots performed rather simple tasks, such as material transfers which only involved moving the robot arm from a point A to a point B etc. More advanced tasks, such as welding and grinding, require a more complex approach in robot control as the robot needs to take into account sensor information due to increased interaction with its environment.

3.2 Robot types

3.2.1 Serial robots

The serial robot is by far the most common robot type used in industry. A typical serial robot, the ABB IRB 2400, is shown in Fig. 1. These types of robots are based on serial chain of rigid links creating a open kinematic chain. Since each link contributes to robot reach, one of the main advantages with serial robots is their large workspace with respect to robot size. Another advantage compared to parallel robots is the reach around obstacles, since parallel robots generally have fewer degrees of freedom.

Figure 1: *Left:* a ABB IRB 2400, a typical serial robot. *Right*: The ABB IRB 360 FlexPicker, based on parallel structure[1]

3.2.2 Parallel robots

In many ways serial and parallel robots are dual, the weak points of one design is the strong point of the other. In general the strong points in a parallel design is greater rigidity, higher speeds and in some means higher accuracy. A substantial disadvantage with the conventional serial robot is the low ratio between load capacity and robot mass. This problem is due to each robot link contributes to the total moving mass. One type of parallel robot, the ABB FlexPicker, can be seen in Fig. 1. As a weak point parallel robots have in general a small workspace in relation to robot volume.

3.3 The Gantry-Tau concept

The Gantry-Tau robot is a parallel robot concept partly developed to find an approach suitable for small medium enterprises. The concept offers a large workspace compared to other parallel kinematic [10] and the possibility to reconfigure according to application. Figure 4 shows a 3 degree-of-freedom, 3 *DOF*, Gantry-Tau parallel robot based on a patent by ABB. The robot is a Gantry variation of the Tau parallel robot, hence the name. The basic variant has three degrees of freedom consisting of three kinematic chains. Each link consists of a cart moving along a track. One of the carts is shown in Fig. 2. The carts are connected to a mounting plate, shown in Fig. 3, using six light-weight carbon fiber links configured in clusters of 1, 2 or 3 links. The configuration is referred to as Tau-configuration which assures a constant orientated mounting plate regardless of cart positions. In a paper by Isolde Dressler [9] it is shown that the accuracy when mounting the tracks is not vital.

3.3.1 Additional Degrees of freedom

The original Ganty-Tau concept presented a robot with 3 degrees of freedom, i.e. *TCP* was not able to reorient. Some applications however require the ability to reorient the *TCP*. To meet the demand of a more versatile setup a number of solutions have been suggested to add additional degrees of freedom to the original design concept. The prototype used at the Robotics Lab, Department of Automatic Control, Lund, has by mounting a two link serial wrist at the mounting plate achieved a setup with 5-degrees of freedom. A pleasant feature using this solution is that the parallel chain and the serial chain can be solved separately, which considerably reduces the complexity when modeling the kinematics. More on this matter in Chapter 4.

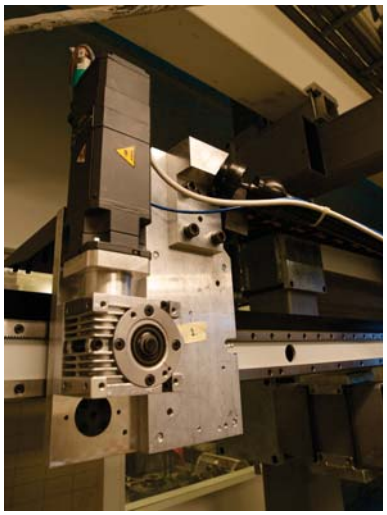

Figure 2: One of the three carts of the *GTP*. The cart consists of a electric motor running along a track. When modeled the cart are modeled as prismatic joints

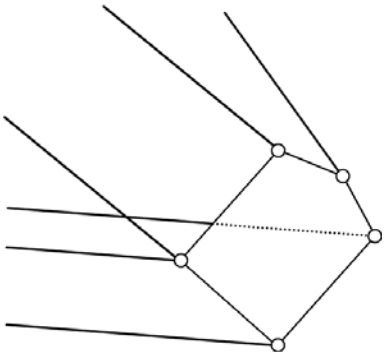

Figure 3: Shows the mounting plate which connects the carts using sets of 1, 2 or 3 parallel links. The mounting configuration is referred to as the Tau-configuration.

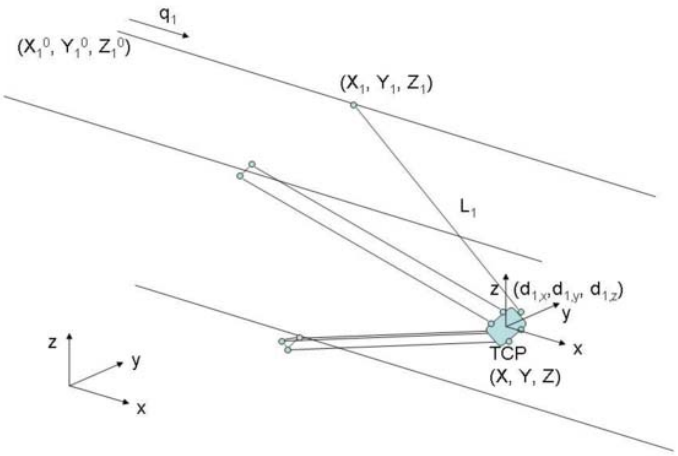

Figure 4: A schematic scheme of the 3 degrees of freedom Gantry-Tau parallel robot. The base frame and *TCP* frame is visible in the figure

4 Theory

The theory chapter in this report is written to give the reader the basic knowledge about the fundamental theory behind robotics [2]. The math behind robot modeling is briefly presented in the first part of the chapter. Section 4.4 involves robot kinematics, such as determining the tool center point, *TCP*, given the robot joint angles, also known as the forward kinematics problem, see section 4.4.1. The inverse kinematics problem: determining the robot joint angles given the position and orientation of *TCP* is presented in section 4.4.2. The velocity Jacobian, gives a relation between joint velocities and the velocities of the *TCP* and is introduced in section 4.4.3.

4.1 Modeling

Robots consist of chains of links connected with joints to form kinematic links. The two main type of joints are the revolute joint and the prismatic joint, see Fig. 5. These joints both have one degree of freedom, namely

$$q_i = \begin{cases} \theta_i & \text{if joint is revolute} \\ d_i & \text{if joint is prismatic} \end{cases} \tag{1}$$

Even though joints in many cases have more than one degree-of-freedom it is possible to model them as successive combinations of prismatic and revolute

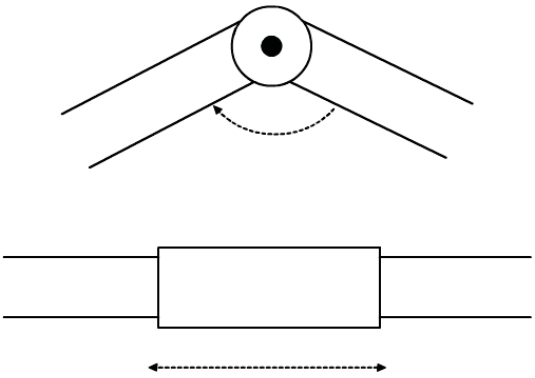

Figure 5: The upper figure shows two links connected with a revolute joint, and below shows two link connected with a prismatic joint. All type of joints can be modeled by connecting appropriate combinations of revolute and prismatic joints in series.

joints. To each link a frame is rigidly attached, used when referring to the robot, see 4.3.1

4.2 Representation

An arbitrary point in space, with respect to a given frame $o^0x^0y^0z^0$, can be represented by a vector p, where

$$p^0 = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2}$$

where x, y and z are the coordinates in the given frame. The superscript is used to clarify which frame of reference is used in the representation. Intiuitivly one can imagine that representing the point given another frame of reference, $o^1x^1y^1z^1$, the coordinates for the point given in 2 will differ. One way to describe a frame is to specify where in space the origin of the frame is located. Of course this point given the coordinates of the frame to be describe will have the coordinates: $(0, 0, 0)$. Given a different frame the point will however have a non-trivial coordinates.

To describe an object in space an orientation of the object is needed, hence a position is not sufficient. Orientation describes the relative rotation between two frames.
Frames are right orthogonal, normalized. The axes are oriented according the screw-rule and have coordinate axes of unit length.
One way to specify a rotation matrix is to specify the coordinate vectors for the axes of one frame with respect to the coordinate vectors of another frame. By

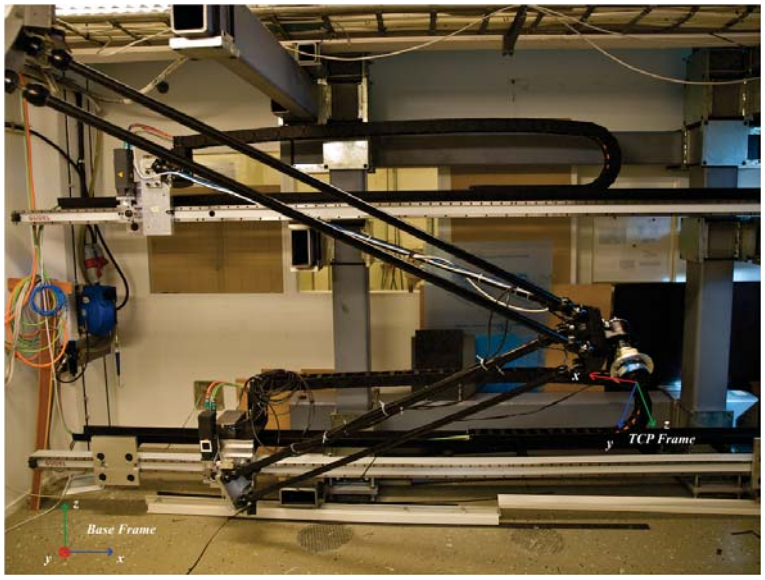

Figure 6: The relative position and orientation of the *base frame* and the *TCP frame* assuming that $q_5 = 0$ Note that the relative position and orientation between these frames are depends on the joint values $q_i, \quad i = 1...5$

projecting each of the coordinate vectors of the frame with respect to the new frame the following matrix can be constructed

$$R_1^0 = \begin{pmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{pmatrix} \tag{3}$$

R_1^0 is denoted the rotation matrix that projects the coordinates vector of frame $o_1x_1y_1z_1$ onto the frame $o_0x_0y_0z_0$ Vector and quaternion representation

4.3 Frames and transformations

4.3.1 Frames

- Base frame - The base frame is the global frame of reference. The base frame has a fixed position and orientation in the workspace.
- Flange frame - The flange, sometimes referred to as the mounting interface, is the mounting point for the tool on the robot.
- Sensor frames - To each sensor used a frame is attached. The sensor frame of the JR3 sensor is oriented with the X and Y axes in the plane of the sensor body and the Z axis perpendicular to the X and Y axes.

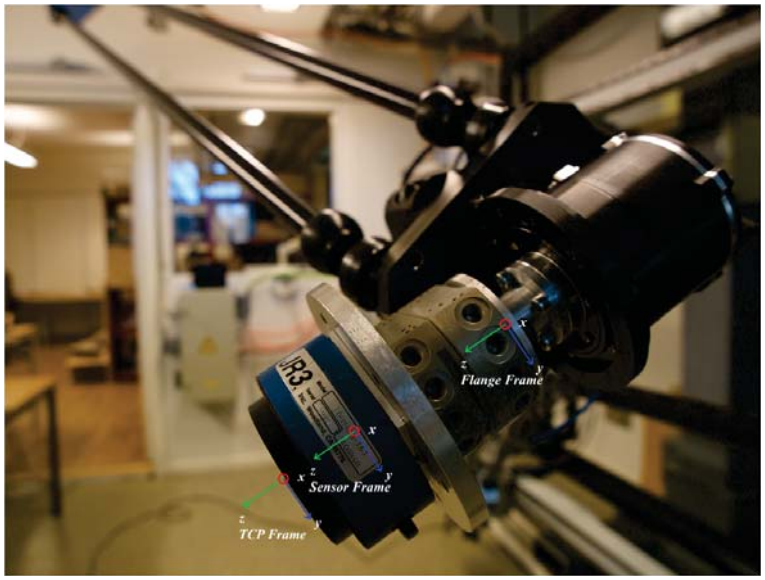

Figure 7: The relative position and orientation of the *flange frame*, *sensor frame* and *TCP frame*. Note that the relative position and orientation between these three frames are constant.

The geometrical center of the sensor is the reference point for the sensor frame. For the ATI sensor the sensor frame is also oriented with the X and Y axes in the plane of the sensor body and the Z axis perpendicular to the X and Y axes, but instead of the geometrical center of the sensor the reference point for the sensor frame is located on the surface of the sensor body.

- TCP frame - Tool Center Point, this frame is attached to the tool center point.

All the frames presented, except the base frame are rigidly attached to some specific part of the robot, sensor or tool.

4.3.2 Transformation matrices

Working with different frames requires the ability to express a vector in several frames since the relative position and orientation of the frames are dependent on the robot joint values. Let v_1^0 be a vector expressed in frame 0 and let v_1^1 be the same vector expressed in frame 1. A homogeneous transformation is a transformation that transforms a vector between different frames. A homogeneous transformation can be divided in two parts, one translation and one rotation. The translation part describes how the frames are oriented relative each other

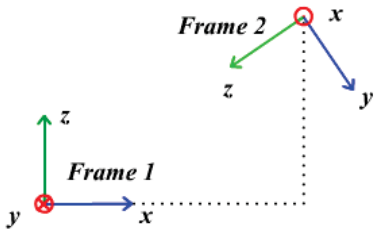

Figure 8: A vector given in *Frame 1* can be expressed in *Frame 2* using a homogeneous transformation. The x, y, z axis of *Frame 2* are expressed in *Frame 1*, representing the rotation part. The origin of *Frame 2* is expressed in *Frame 1*, representing the translation.

and the translation part describes the offset between the two frames, see Fig. 8.

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}, \quad R \in SO(3), \quad d \in \mathbb{R} \tag{4}$$

Homogeneous transforms combine rotation and translation and are used to perform coordinate transformation between different frames.

4.4 Kinematics

The *duality* between serial and parallel robots are also reflected in the mathematics surrounding the kinematics. As an example the forward kinematics problem is regarded as a quite easy problem to solve regarding serial robots, where as for parallel robots the same problem is more complex. For inverse kinematics the opposite holds, the inverse kinematics is generally more difficult to solve for a serial robot than for a parallel robots. This section discusses the kinematics from a parallel robot perspective, but some concepts for serial robots is also included as the robot wrist forms a serial chain.

4.4.1 Forward kinematics

The forward kinematics is the relationship between the individual joints, q, and the position and orientation of the tool or end effector [2]. A systematic approach when developing the forward kinematics is to assign a frame of reference for each of the link and derive homogeneous transformations between one link to another. A commonly used convention for selecting the frames of reference in robotics is the *Denavit-Hartenberg conventions* where the *link length* a_i, *link twist* α_i, *link offset* d_i and *joint angle* θ_i parameters are used to derive a T44 transformation matrix from link to link.

$$T_{i-1}^{i} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i)cos(\alpha_i) & sin(\theta_i)sin(\alpha_i) & a_i cos(\alpha_i) \\ sin(\theta_i) & cos(\theta_i)cos(\alpha_i) & -cos(\theta_i)sin(\alpha_i) & a_i sin(\alpha_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

The $T44$ in equation 5 is the transformation matrix between link $i-1$ and link i. In serial robotics the Denvite-Hartenberg conventions prove very useful because the final forward kinematics for a robot manipulator with n links is the product of the derived transformation matrices

$$T_{base}^{flange}(q) = \prod_{i=1}^{n} T_{i-1}^{i}(q) \tag{6}$$

However, 6 is only valid in a serial setup and consequently only be used for the wrist link 4 and 5 of the Gantry-Tau parallel robot.

For the 5-DOF Gautry-Tau robot the forward kinematics problem can be divided into two problems that can be solved separately.

- Find the relationship between the cart positions and the position and orientation of the mounting plate
- Find the relationship between the mounting plate and the position and orientation of the mounting flange

The relationship between the mounting plate and the position and orientation of the mounting flange can, as mentioned above, be solved using Denavite-Hartenberg conventions as described.
To derive the relation between the carts and the mounting interface other methods are used [10]. Let l_X denote the length of the links connected to cart X. For fixed cart positions A, B and C the intersection between three spheres with radius l_A, l_B and l_C and midpoints at respective cart describe all possible positions of the mounting plate. The midpoints of the spheres are located at respective cart position as mentioned

$$A = (x_a \quad y_a \quad z_a)_{base} \tag{7}$$
$$B = (x_b \quad y_b \quad z_b)_{base} \tag{8}$$
$$C = (x_c \quad y_c \quad z_c)_{base} \tag{9}$$

and the possible mounting plate positions are given by the intersection of the following spheres.

$$(x_a - x)^2 + (y_a - y)^2 + (z_a - z)^2 = l_A^2 \tag{10}$$
$$(x_b - x)^2 + (y_b - y)^2 + (z_b - z)^2 = l_B^2 \tag{11}$$
$$(x_c - x)^2 + (y_c - y)^2 + (z_c - z)^2 = l_C^2 \tag{12}$$

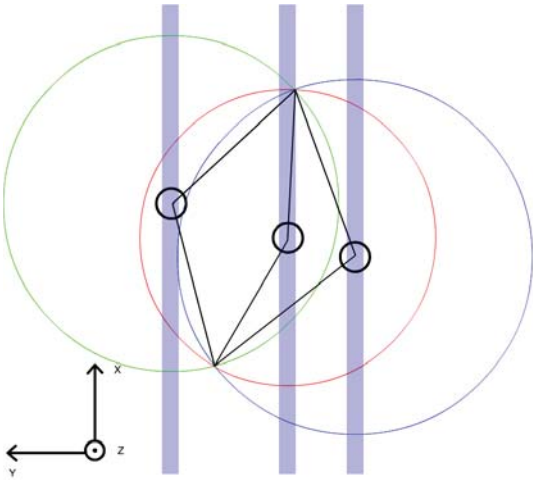

Figure 9: An illustration of the possibility of multiple solutions when solving the forward kinematics problem. Let each circle represent a cart that can move in the x-direction. Each circle represents possible link positions for the individual links in the x-y plane, hence the intersection points between the three circles represent possible mounting interface positions. The example illustrated in this Fig. has two possible solutions to the forward kinematics problem

As shown in [10] this geometrical problem can have multiple solutions, see Fig. 9. This is solved by defining different mounting plate configuration in which case a unique solution can be found if solutions exists for those cart positions. Due to the mechanical construction of the Gantry-Tau, 3.3, the orientation of the mounting plate in relation to the base frame will be fixed for all possible cart positions.

4.4.2 Inverse kinematics

The inverse kinematics end effector and the joint angles. In serial robotics the inverse kinematic problem is a more difficult problem than the forward kinematics due to the possibility of multiple solution or no solutions. Figure 10 illustrates a scenario where multiple solutions exists. The Gantry-Tau 5 DOF configuration has only two links in a serial chain. The relative joint positions of the serial chain however eliminates the elbow up and down problem.

In a parallel link setup the inverse kinematics problem is a a simpler problem than the forward kinematics problem, due to the fact that each cart position can be determined separately. Each cart position is calculated by finding the intersection between a sphere with midpoint at the mounting plate and the respective linear track. The diameter of the sphere is determined by the link

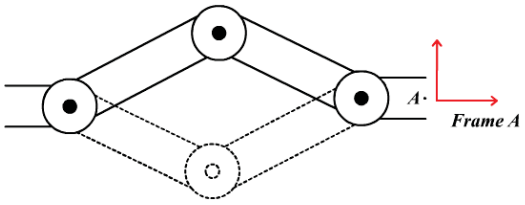

Figure 10: The elbow up/down phenomena. For given position and orientation of frame A, the inverse kinematics problem has several solutions. Figure shows the famous elbow up/elbow down problem often encountered working with serial robots.

length. Solving this problem normally yields two solutions for each cart which adds up to a total of up to eight different solutions for the inverse kinematics. As in the forward kinematics problem mounting plate configuration parameters are used to determine the solution. The inverse kinematics problem can be formulated as the following. Given the location of the TCP calculate A,B and C. This problem has eight solutions.

4.4.3 Velocity Jacobian

The Jacobian relates the joint velocities with the velocity of the flange. The relation is a function of the joint position, i.e the Jacobian is not constant. Let $J(q)$ be the Jacobian for the system, then the following relation holds

$$\dot{x} = J(q)\dot{q} \tag{13}$$

The Gantry-Tau robot has 5 joints, one for each cart and two in the serial link attached to the mounting interface. Expressing the flange velocities $\dot{x}$ in Cartesian coordinates with the flange frame as reference, six variables are needed. The translation in x, y, z direction, v_x, v_y, v_z and the angular velocities ω_x, ω_y and ω_z respectively. The set of linear equations to be solved to find the relation between joint and flange velocities are given by

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} j_{11} & j_{12} & j_{13} & j_{14} & j_{15} \\ j_{21} & j_{22} & j_{23} & j_{24} & j_{25} \\ j_{31} & j_{32} & j_{33} & j_{34} & j_{35} \\ j_{41} & j_{42} & j_{43} & j_{44} & j_{45} \\ j_{51} & j_{52} & j_{53} & j_{54} & j_{55} \\ j_{61} & j_{62} & j_{63} & j_{64} & j_{65} \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \\ \dot{q_5} \end{bmatrix} \tag{14}$$

where $j_{n,k}$ is the element n, k in the $J(q)$. In real application the inverse Jacobian is often of interest for instance to calculate the joint velocities for a given flange velocity. Since the Jacobian is a $6x5$ matrix it is not invertible. If $J(q)$ has full rank however, it is possible to derive the pseudo inverse of the matrix

$J(q)$ and thereby calculate the joint velocities.

$$\dot{x} = J(q)\dot{q}, \quad J(q) \in \Re^{mxn}, m > n \tag{15}$$

$$J(q)^T\dot{x} = J(q)^T J(q)\dot{x} \tag{16}$$

$$\dot{q} = \underbrace{(J(q)^T J(q))^{-1})}_{J(q)^\dagger} J(q)\dot{x} \quad \{q|rank(J(q)) = n\} \tag{17}$$

where $J(q)^\dagger$ is referred to as the pseudo-inverse of $J(q)$.

The Gantry-Tau parallel robot used in this thesis is implemented on the ABB IRC5 robot system. The IRB5 robot system is developed primarily for 6 DOF serial IRBs. Due to this fact, the robot system expects a 6x6 square Jacobian. Previous research by Isolde Dressler included implementing kinematics on the IRC5 robot system. In order to construct a Jacobian compatible with the robot system, a imaginary sixth joint was introduced making the Jacobian a 6x6 square matrix. Much of the work in this thesis is based on Isolde Dressler's research, so the same six axis solution was implemented in the kinematics library developed in this thesis. Even if the sixth joint is imaginary a great deal of care has to be consided when orienting the imaginary axis. An incorrectly introduced joint can lead to singularities in the solution. After the introduction of the sixth joint the set of equations to be solved expands to

$$\begin{bmatrix} j_{11} & j_{12} & j_{13} & j_{14} & j_{15} & j_{16} \\ j_{21} & j_{22} & j_{23} & j_{24} & j_{25} & j_{26} \\ j_{31} & j_{32} & j_{33} & j_{34} & j_{35} & j_{36} \\ j_{41} & j_{42} & j_{43} & j_{44} & j_{45} & j_{46} \\ j_{51} & j_{52} & j_{53} & j_{54} & j_{55} & j_{56} \\ j_{61} & j_{62} & j_{63} & j_{64} & j_{65} & j_{66} \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \\ \dot{q_5} \\ \dot{q_6} \end{bmatrix} \tag{18}$$

The problem can however be simplified by interpreting the physical meaning for each element of the Jacobian and take into account what is known about the Gantry-Tau setup. As described in section 3.3, the mounting interface of the Gantry-Tau parallel kinematic robot will keep a constant orientation for all cart positions, hence no angular velocities can be induced by only moving the carts.

$$\begin{bmatrix} j_{41} & j_{42} & j_{43} \\ j_{51} & j_{52} & j_{53} \\ j_{61} & j_{62} & j_{63} \end{bmatrix} = J_{translation[q1-q3]->rotation[x]} = 0 \tag{19}$$

The translation and rotation flange due the fifth joint is expressed by the following elements in the Jacobian matrix

$$\begin{bmatrix} j_{15} \\ j_{25} \\ j_{35} \\ j_{45} \\ j_{55} \\ j_{65} \end{bmatrix} = J_{\dot{q_5}->\dot{x}} \tag{20}$$

Since the fifth joint will always be align parallel with the z-axis of the *Flange* frame the only non zero element is j_{65} which maps a rotation of joint 5 with the rotation of the flange along the flange z-axis.
By taking these observation into account the complexity of the velocity Jacobian and the velocity Jacobian inverse is reduced.

4.5 Force sensors

4.5.1 The JR3 force sensors

In this thesis a control strategy using two force sensors will be presented. One of the sensors used in the experiments was the *JR3 100M40* [7]. The 100M40 is used measuring forces and torques acting on the sensor, F_{sensor}

$$F_{sensor} = \begin{pmatrix} F_x \\ F_y \\ F_z \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} \tag{21}$$

where F_x, F_y and F_z are the normal forces and τ_x, τ_y and τ_z are the torques acting on the sensor with respect to the x, y and z direction of the sensor frame. The measurement F_{sensor} is used as force feedback for the force controllers. The rated maximum load for the sensor is 400N and measurements are carried out by the sensor's internal electronics at a sampling rate of 8kHz. The two sensors were connected to a PC in the lab and the force measurements could then be accessed in real time through a network connection.

4.5.2 The ATI nano25 force sensor

In addition to the two JR3 sensors used, described in section 4.5.1, an ATI nano25 [8], sensor was used. As the name insists the physical dimension of the ATI nano25 are very small. With a diameter of merely 25mm and a height of 22mm, this sensor is one smallest force sensors in the market. The sensing range and noise levels of the ATI nano25 is similar to the JR3 100M40, but the ATI is much more robust to single axis overloads compared to the JR3. The robustness feature has made the ATI force sensor popular in robotic applications.

4.6 Force control

In tasks where contact between robot and the environment occur a convenient way of controlling the robot is to control the forces between the tool and the environment. Forces can also be used in the interaction between robot and operator in a so called, lead-through scenario. In a lead-through scenario the operator can, via applying forces to a force sensor, change the position and orientation of the robot end-effector. By doing so the operator has the ability

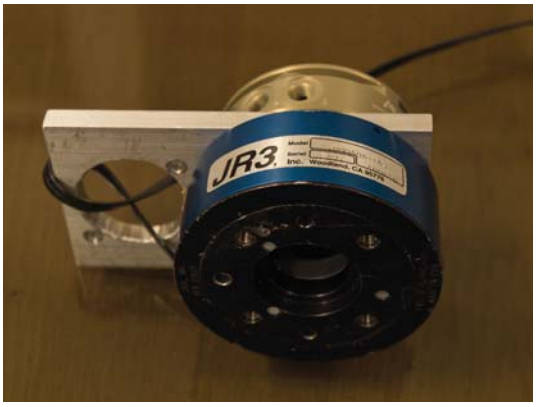

Figure 11: One of the two *JR3* force sensors used in experiments. The force sensor is attached to a mounting plate that allows mounting on the robot.

Figure 12: The *ATI nano25* force sensor used in experiments.

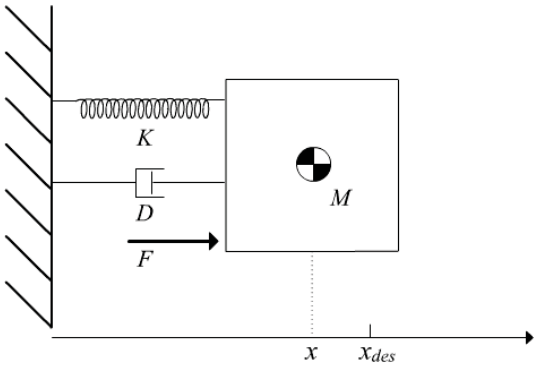

Figure 13: A spring-mass-damper system. A mass M actual position x and desired position x_{des}. The interaction with the environment is modeled a damper with corresponding damping constant D and a spring with spring constant K. A external force F is acting on the system.

to manipulate the end-effector in a way that is often more intuitive then using a joystick or control panel. In this section three different types of force controls will be described.

4.6.1 Direct force control

When using a direct force control the aim of the control is to achieve a desired force by acting on the difference between actual force and the reference force. Using a PI controller yields the following control law in continuous time

$$e(t) = F_{desired}(t) - F_{actual}(t) \tag{22}$$

$$u = Ke(t) + \frac{K}{T_i}\int_0^t e(t)\mathrm{d}t \tag{23}$$

4.6.2 Hybrid force control

4.6.3 Impedance control

The aim with impedance control is to implement a feedback control algorithm imposing a desired cartesian impedance [5]. The control action is separated in two actions, one representing the motion control and the other representing the dynamic control [4]. In the dynamic control the aim is to make the robot interact with the environment as a specified mass-spring-damper system, see 13.

$$F = M(\ddot{x} - \ddot{x}_{des}) + D(\dot{x} - \dot{x}_{des}) + K(x - x_{des}) \tag{24}$$

Implemented as

$$F = M\ddot{x}_{ref} + D(\dot{x} - \dot{x}_{des}) + K(x - x_{des}) \tag{25}$$

where F is the force acting on the system, $\dot{x}_{des}$ and x_{des} is the desired speed and desired position of the robot, $\dot{x}$ and x is the actual speed and position of the robot. M, D and K are design parameters in the impedance controller.

$$\ddot{x}_{ref} = M^{-1}\big\{F - D(\dot{x} - \dot{x}_{des}) - K(x - x_{des})\big\} \tag{26}$$

The system will behave like the specified spring-mass-damper system if the robot system fulfills the equation above. Since the input to the robot system is reference positions and velocities the reference acceleration has to filtered. The following relations hold

$$\dot{x}_{ref} = \int \ddot{x}_{ref}\,\mathrm{d}x \tag{27}$$

$$x_{ref} = \iint \ddot{x}_{ref}\,\mathrm{d}x \tag{28}$$

In the implementation of the controller a discrete filter is used to filter the reference position and velocity. Given the sampling time h the following relations holds

$$\dot{x}_{ref}[k] = \ddot{x}_{ref}[0] + \sum_{i=1}^{k} \ddot{x}_{ref}[i]h \tag{29}$$

where $\ddot{x}_{ref}[0]$ is the initial acceleration. Filtering $\dot{x}_{ref}$ once more yields the position reference, given by

$$x_{ref}[k] = \dot{x}_{ref}[0] + \sum_{i=1}^{k} \dot{x}_{ref}[i]h \tag{30}$$

where $\dot{x}_{ref}[0]$ is the initial position. The tuning parameters in the controller are the spring constant K, the damping constant D and the mass M. How these parameters are chosen obviously changes the characteristics of the controller. In control theory the following characteristic equation is often used to determine the behavior of a second-order system

$$s^2 + 2\zeta\omega s + \omega^2 = 0 \tag{31}$$

where ζ is referred to as the damping ratio of the system and ω is referred to as the angular frequency of the undamped system. A system is said to be critically damped if $\zeta = 1$. A critically damped system that initially is in rest will when subjected to a step, converge without oscillating, i.e no overshoot. In robotic motion control overshoots there is often a desire to avoid overshoots. Using this knowledge about a second order system, a mapping between the impedance control design parameters and the damping ratio, ζ, and the angular frequency of the undamped system, ω is preferred. The mapping used in the thesis is the following

$$D = 2M\zeta\omega \tag{32}$$

$$K = M\omega^2 \tag{33}$$

As one may notice three design parameters are mapped into two. The extra degree of freedom from a control design point of view given by the third parameters, i.e M, is that the inertia of the system can be changed, as in the impact of K and D given a set of ζ and ω without a change of ratio between them. However one should keep in mind that the model is used to describe the behavior of the manipulator that ignores some aspects of the dynamics such as actuator dynamics and transmission. For further reading on impedance control implementation in robotics following three papers are highly recommended [4], [5] and [6].

5 Method

In this chapter methods used in this thesis will be presented. Section 5.1 gives a short introduction to the ABB IRC5 robot system and the Opcom application used to implement external controllers. Toolboxes and features used in Matlab/Simulink is presented in section 5.2. Control design, force control implementation and the safety-zone is presented in section 5.5. The signal processing of the raw force measurement signal are found in section 5.6. Finally, the experiential issues regarding gravity compensation and timing constrains are presented in section 5.7.

5.1 The IRC5 robot system

IRC5 is a robot system developed by ABB [1], used to communicate with the Gantry-Tau parallel robot. The IRC5 robot system uses internal controllers for each joint. The internal controllers run on axis computers which in turn communicate with one main computer. The main computer calculates reference position and velocities for each joint based on sensor information from the joint motors. The axis computers compute motor torques according to the reference joint and velocities received from the main computer.

During calibration the robot can be moved using the robot control panel, the FlexPendant, see Fig. 14. Moving the robot manually using the control panel is often referred to as jogging the robot. The FlexPendant can also be used to program the robot using the *RAPID* language. Simple RAPID programs were used in validating the developed kinematic library. In order to move the robot, jogging or using external controllers, the dead-man's switch located on the FlexPendant must be pressed.

5.1.1 Opcom

Opcom is a application developed to connect external controllers to the robot system [3]. Controllers designed in Simulink are compiled into C code and

Figure 14: The FlexPendant can be used to jog the robot or execute *RAPID* based programs

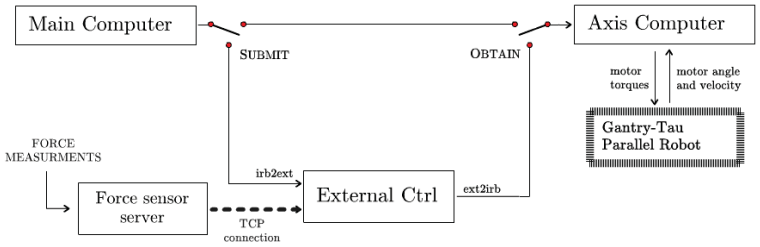

Figure 15: A schematic figure of the robot system setup

downloaded to the robot system. Opcom is the graphical user interface, *GUI*, used to load external controllers into the IRC5 robot system. Once the controller is loaded it can be operated in two modes, either in *Submit* mode or *Obtain* mode.

In *Submit* mode signals are sent from the main computer to the external controller but no signals are returned to the robot system. This is essential feature as it provides a safe way to test the external controllers with input from the real robot without any risk of damaging the robot or harming anyone inside the work cell. As described in section 5.2.2, the compiled controller is not directly based on the S-function but instead based on *tlc*-files. Hence, there is no guarantee that the compiled controller behaves in the same manner as the controller used in the Simulink simulations.

In *Obtain* mode the output from the external controller is used as input to the axis computers. Before any experiments are carried out in *Obtain* mode it is vital that thorough testing in *Submit* mode is carried out, verifying that the controller is behaving properly.

Important to note is that controllers loaded in Opcom starts running as soon as the model is loaded. To avoid integrator windup in the controller certain precautions are made. A boolean *fswitch* is introduced in the model, which when set to true activates the controller. The default value of *fswitch* is set to false. In order to avoid integrator windup the controllers are only activated(*fswitch* set to true) when control signal output is sent to the actuators, i.e. while Opcom operates in *Obtain* mode.

Prior to compiling *Real Time Workshop* provides the ability to define signals to be logged, which is important when testing and debugging the controller. In experiments carried out in this thesis to investigate if logging signals can effect the real-time performance of the controller.

5.2 Matlab/Simulink

5.2.1 S-functions

S-functions is a powerful tool to extend the capabilities of Matlab Simulink. A S-function is a language description of a Simulink block that can be written in

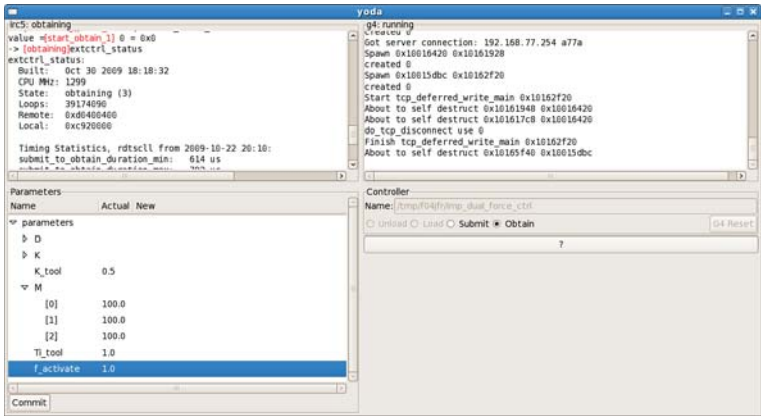

Figure 16: The graphical interface Opcom used to load external controllers into the robot system. Inlined parameters can be changed in real-time.

several programing languages. A S-function is after it is written compiled as a MEX-file. The Simulink blocks are in this thesis used for simulation of the robot kinematics and for designing the force controller.

5.2.2 Real-Time Workshop

Real-Time Workshop, *RTW*, is a tool package available in Matlab Simulink. Using the *RTW* toolbox enables the user to translate and generate C source code from Simulink models and Embedded Matlab code. The generated C source code is used to run real time applications. The source code, for the user defined S-functions, are not used directly by *RTW*, instead the *RTW* requires a target language compiler file, *tlc*. The designer has to take extra consideration when generating C code from user defined S-functions as the simulations are based on the C code for the S-functions whereas the generated source code are based on the *tlc* file.

5.3 Block development

Extctrl is a kinematic library [3] that supports several of the ABB serial robots. The library includes a set of S-functions that describe the dynamics of the robot. Such S-functions include blocks for forward and inverse kinematics and calculation of velocity Jacobian, blocks that can be used to design external controllers. Since the Gautry-Tau robot is in a prototype stage(at the time for this thesis no Simulink kinematics library was available) the main purpose of the thesis was to develop one. Throughout the development there was a desire to keep the S-function C code as generic as possible, i.e. in as large extent as possible using the same function calls and variables as used by the *extctrl* library for the serial ABB robots. The Ph.D student Isolde Dressler had prior to this

thesis developed C code for the Gantry-Tau, so much of the work consisted of combining Isolde Dressler's work with the generic structure of the *extctl* library to develope a kinematics library for the Gantry-Tau parallel robot. S-functions for the forward kinematics, inverse kinematics and Jacobian was developed, more information on the developed library can be found in Chapter 6.

5.4 Validation

A number of steps were taken to validate the code.

- The forward kinematics was tested for a set of inputs for which Isolde Dressler had expected outputs.
- A simple test is to send joint values to forward kinematics block. The output is then sent to an inverse kinematics block and then compared to the input to the forward kinematics block. If everything is correct the output from the inverse kinematics block should match the input to the forward kinematics block.
- A *RAPID* program was made using the FlexPendant where the robot was moved between points spread around the entire workspace. The position of the *TCP* given by the robot system was compared to the position calculated by the forward kinematics block.
- The Jacobian could be evaluated by comparing small deviations in joint positions with the corresponding deviations of the *TCP*.

5.5 Control Design

5.5.1 Lead-through control

In normal lead-through scenario the spring damper is set to zero because it is desired that the robot act as a pure integrator. Setting the spring constant to zero has the effect that the mass in the mass-spring-damper system in section, 4.6.3, will have no effect.

5.5.2 Safety zone

A safety zone was implemented to give the operator a hint that the robots *TCP* is getting close to the outer borders of the robots work space. In normal operation the lead-through impedance controller operates without spring action, i.e. the robot stays at the position given by the operator. By implementing spring action at the outer borders of the work space the operator is given feedback by proportional increasing spring action as the *TCP* enters the safety zone. By having the different frames of reference the implementation was rather simple. A safety zone was defined in base frames x, y, z directions. By using the base frame as reference each direction can be implemented separately. For each direction a test is made deciding weather the *TCP* has entered the safety zone

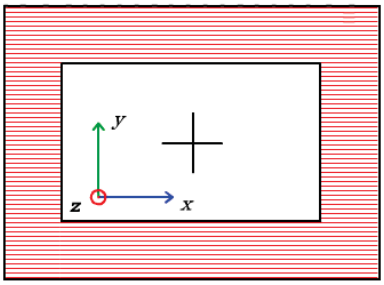

Figure 17: A schematic image of the safety zone. The outer square represents the entire robot workspace. The inner square represents the workspace subset where no spring force action is applied. The red zone is the zone where the spring force is activated. The force is proportional to the distance traveled into the safety zone and directed in the direction closes to the inner workspace.

or not. If that is the case the desired position is set to the safety zone boarder in *base frame* and then transformed to the *TCP frame*. If a spring constant $K > 0$ is applied, the difference between actual position and desired position will introduce spring force action, see equation 26. The spring force action will, if no other forces are applied, slowly move the robot the shortest way back to the safety zone inner border. This implementation sends intuitive feedback to the operator that the robot is approaching the work space borders. In some cases however the operator might want to work in the outer areas of the robot work space, so the safety zone implementation can be disabled by setting the *Safety Zone parameter* to zero. Figure 17 shows a schematic illustration of the safety zone.

5.5.3 Tool force sensor

5.5.4 Dual sensor setup

To further test the kinematic library a goal in set up in this thesis was to implement two force sensors to control the robot in a lead-through scenario with one sensor acting as safe-guard for the tool. If the force controller implemented for the lead-through force, F_{LT}, sensor is viewed as the main controller, MC, and the force controller acting on the input from the force acting on the tool force sensor, F_{tool}, is viewed as the secondary controller, SC, many different implementations can be taken into consideration.

5.5.5 Hybrid control

$$u = \begin{cases} MC(F_{LT}) & \text{if switch condition} = 0 \\ SC(F_{tool}) & \text{if switch condition} = 1 \end{cases} \tag{34}$$

where switch condition $\in 0, 1$ and is a function of F_{LT} and F_{tool}

5.6 The force sensor signal

The signal from the *JR3* force sensor is introduced to the Simulink model through the port *jr3_comedi*.

5.6.1 Resetting the sensor and gravity compensation

Due to the construction of the force sensor the sensor itself will contribute to the measured force signal. These forces, due to e.g. the weight of the sensor itself, are not desirable when conducting force control. Due to these forces a crucial step at start up is to reset the force sensor. In he *extctrl* library there is a block used for reseting the sensor for a given signal. However, if the force sensor is in some way reoriented the reset done at startup is no longer valid, this is because the force sensor frame and the gravity force vector no longer have the same relative orientation. In the case when the force sensor frame is reoriented a gravity compensation is needed. Luckily the *extctrl* library also has a block for implementing gravity compensation.

5.6.2 Noise components

The raw measurement from the *JR3* force sensor was subject to noise. The cable connecting the force sensor with the measurement PC was quite long so to a FFT (Fast Fourier transform) was performed on the F_z component of the signal in order to investigate if the signal was subject of some external noise source. Figure 18 shows the measurement along with a FFT. The sample frequency during the measurement was 250Hz, hence no frequencies over the Nyquist frequency was shown in the FFT. No frequency peak was observed in the FFT, so the noise can be regarded as white noise.

5.6.3 Filtering

A discrete first order low pass filter was implemented to smoothen the force measurement signal. A higher order filter was also implemented but in order not to introduce to much delay the first order filter was used throughout the experiments.

5.6.4 Dead-zone

When no force was applied, oscillations was observed in the control signal, see Fig. 19, due to noise elements making the force input to the controller changing sign. Even if the induced control signal oscillation was small this type of behavior can be harmful for the joint axis motors in the long run. A dead-zone L was implemented for the force measurement

$$F_{out} = \begin{cases} max(F_{in} - Lsign(F_{in}, 0) & ,\forall F_{in} > 0 \\ min(F_{in} - Lsign(F_{in}, 0) & ,\forall F_{in} \leq 0 \end{cases} \tag{35}$$

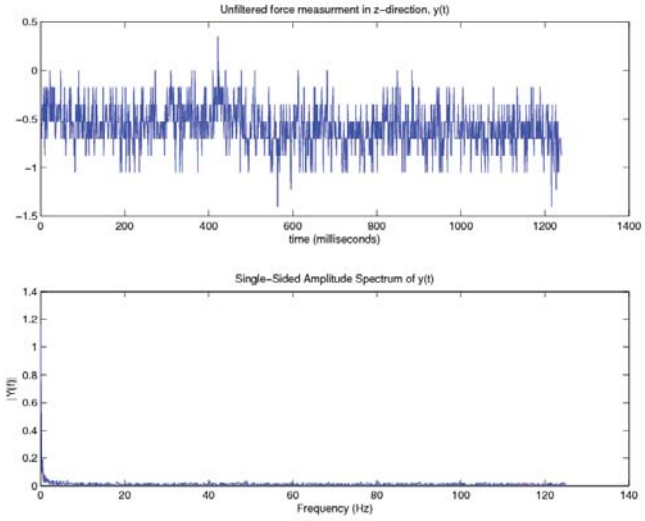

Figure 18: The upper plot shows the noisy force measurement signal after gravity compensation, no forces applied. The lower plot shows a single sided amplitude spectrum of the measurement. Sample frequency 250 Hz

where F_{in} is the force signal into the dead-zone block and F_out is the force signal out from the dead-zone. L is the size of the dead-zone and throughout the experiments a value between 0.5 N was used.

5.7 Experiments

5.7.1 Gravity compensation

The force sensor is reset before experiments in order to only measure external forces and not the weight of of the tool attached to the force sensor. This will work as long as the tool is not reoriented. As the tool is reoriented the tool weight will contribute to the force measurement signal. In the drill use case scenario, reorientation of the tool may occur and therefor a compensation for the gravity contribution to the force measurement is needed. The extctrl library includes a block for gravity compensation. Inputs to the block are tool mass, the flange T44 matrix and the output is a force signal that compensates the force contribution from the tool. The tool mass can be estimated by a force measurement experiment where the robot is jogged in a way such that the tool will contribute to the force measurement. The goal of gravity compensation is that the robot force measurements will stay at zero as long as no external forces are applied, regardless of robot orientation.

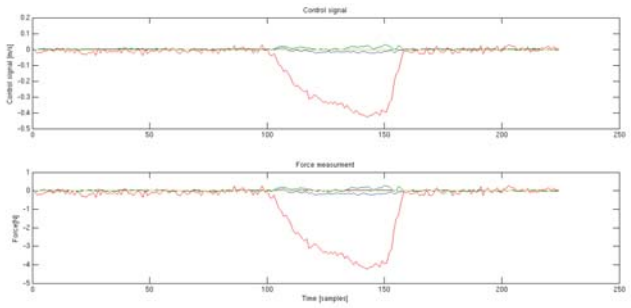

Figure 19: The upper plot shows the control signal using a PI controller when no dead zone is implemented. Due to noise in the force signal, the lower plot, a noisy control signal is obtained, which decreases accuracy and increases long term actuator tear.

5.7.2 Opcom Timing constrains

The entire *IRC5* robot system has a hard timing constrain for the entire control loop of 4 milliseconds, shown as D-A in Fig. 20. During the control loop the robot system reads information from the system's sensors. An external controller which is loaded using Opcom and set in submit mode, receives robot joint from the robot system and force measurements from the measurement server via a TCP connection. The time duration for receiving and copying the information is marked as subtask 1 in Fig. 20. In obtain mode the time duration for calculating output from the external controller and the time duration for sending the output to the robot system is also measured, marked as subtask 2 and 3 in Fig. 20. The control signal for the joint actuators are calculated by each respective axis computer, shown as the IRC5 activity done after Opcom in Fig. 20. If the loop duration exceeds 4 milliseconds the system will shut down and enter system failure mode.
Due to the timing constrains of the entire control loop, the Opcom application uses its own timing requirements to ensure that the Opcom face of the control loop does not take too much time to execute. Using default settings the maximum allowed duration from data copied to Opcom to a new control signal is calculated is set to 600 μs.

5.7.3 Timing statistics

After validating the kinematic library in simulations, the first experiments were conducted. Initially strange behavior was observed during execution. The robot would run perfectly and behave according to the simulation, but could unexpectedly get an over speed system failure. Much effort was put in trying to determine the cause of the behavior. Excessive logs were conducted, at first it seemed like the problem was caused by rapid changes in reference position and

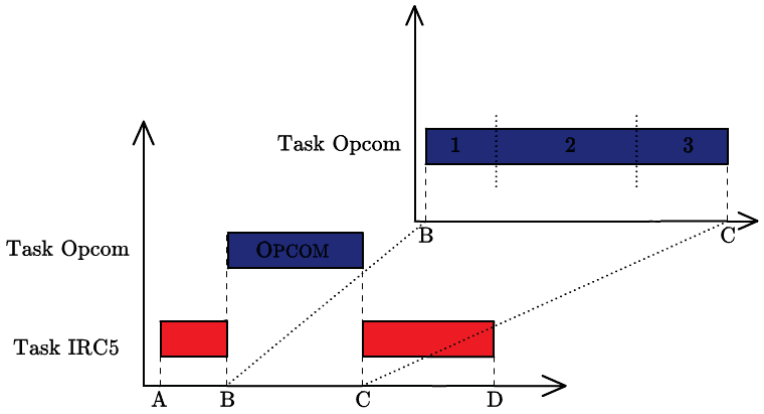

Figure 20: A schematic figure showing the Opcom task timing. The IRC5 task represents normal operation without an external controller. A hard timing constraint of 4 ms is imposed between A and D. To insure that the IRC5 timing constrains are not violated, a timing constraint is imposed on the Opcom task, B to C. The Opcom task can, when running in Obtain mode be divided into three subtasks. 1) Copy to Submit 2) Calculate Output 3) Copy from Obtain

velocity caused by the teach pendant even though the teach pendant was not used during the experiments. Finding the source of this behavior was tedious, partly due to the fact that the Robotic lab at the department of control does not have access to the robot system source code but only to the compiled system. So logs were sent back and fourth to helpful people at ABB robotics in Vsterrs. With the insight from ABB, we came to the conclusion to that the problem was due to timing constrains in Opcom was not being met, causing the robot system to reset. When the IRC5 robot system resets, default position and velocity references are given to the robot. That explains the steps seen in the logs.
The model was simplified in order to make sure that no timing constrains were violated, but after a series of experiment, the strange behavior was even observed in empty Simulink models that were compiled using real time workshop. The Opcom source code was modified to give the user rich data about the loop timings, trying to isolate the strange behavior. Finally, after great help from Anders Blomdell, a bug in the Opcom source code was found in the method timing the obtain loop of the Opcom application. The bug was due to timing variable in the internal processor was stored in two separate 32-bit variables instead of one 64-bit variable. In certain variable overload situation this internal structure would cause the Opcom application to believe that the obtain loop had broken the timing constraints. The Opcom application communicated the error message to the IRC5 robot system which in turn caused the robot system to reset.

6 Gantry-Tau Simulink Kinematics Library

Time wise a big part of this master thesis work revolved around developing and validating a Simulink library for the kinematics used by the Gantry-Tau parallel robot.

6.1 Numerical Linear Algebra

Much effort was put into finding and testing mathematical packages which could run the routines. As described in section 5.2.2, uses one code for simulation in Simulink and another code when compiling into c for real time application. For mathematical operations such as computing the matrix inverse a software package is used to perform numerical linear algebra. In simulations the software library *LINPACK* [12], which makes use of *BLAS* (Basic Linear Algebra Subprograms) [13], for performing basic vector and matrix operation.
The Power PC running the compiled real time application however had at the time for this thesis no support for *LINPACK*. An attempt implementing the software library *LAPACK*,(Linear Algebra PACKage), [14]. LAPACK can partly be view as a successor of LINPACK, also deping on BLAS rutines.

6.2 Motor to arm angle conversion

Signals from the IRC5 robot system that are used as input to the external controller, *irb2ext* are given as motor angles, q_m. The kinematic blocks in the Gantry-Tau library uses arm angles (joint angles), q, as inputs. Therefor a conversion between motor angles and arm angles are needed. For the Gantry-Tau robot the following diagonal matrix is used for conversion between motor and arm angles:

$$arm2motor = \begin{bmatrix} \frac{400}{1000} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{400}{1000} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{400}{1000} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{80\pi}{180} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{80\pi}{180} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{80\pi}{180} \end{bmatrix} \tag{36}$$

$$q = q_m(arm2motor) \tag{37}$$

where column 1 to 3 represents the conversion between motor angles given in radian to joint position in mm. Column 4 and 5 represents the conversion between degrees to radians. Since the 6th joint is imaginary the sixth column does not represent any physical conversion, the sixth diagonal element was set to 1. A similar conversion is used before signals are sent back to the robot system, *ext2irb*.

$$motor2arm = arm2motor^{-1} \tag{38}$$

$$q_m = (motor2arm)q \tag{39}$$

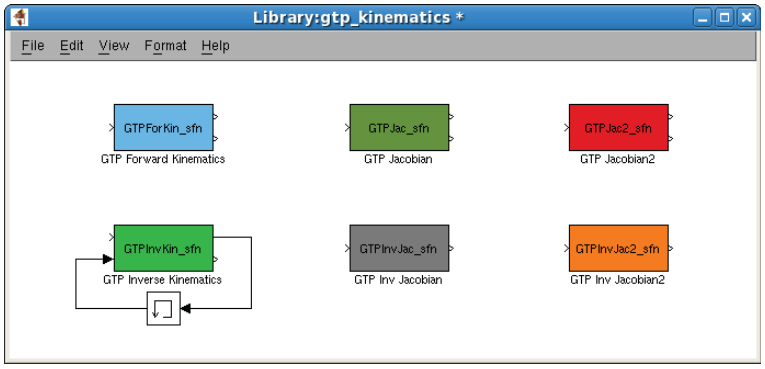

Figure 21: The developed kinematics library for the Gantry-Tau parallel robot. *Upper row from the left:* Forward kinematics block, Velocity Jacobian (implementing LAPACK), Velocity Jacobian (implementing svdcmp). *Lower row from the left:* Inverse Kinematics, Inverse Jacobian (implementing LAPACK), Inverse Jacobian (implementing svdcmp).

6.3 Kinematic blocks

6.3.1 Forward Kinematics

Input: Joint positions, being the three cart positions and the motor angles of joint 4 and 5.
Output 1: A *T44* transformation matrix (the transformation matrix between the base frame and the mounting interface (Flange)). The matrix is given as as row major vector
Output 2; Returns a integer error signals if errors occurred calculating the forward kinematics. Integer zero indicates that no errors detected

6.3.2 Inverse Kinematics

The developed inverse kinematics block does not use the exact methods explained in the theory section 4.4.2 to compute the inverse kinematics. As seen in Fig. 21, the block is uses together with a Memory function in order to use the previous joint positions, $q(k-1)$, as input to the block. One reason why this solution was used is to create a kinematics library that matches a generic kinematics library, in this case that means a kinematics library that in look and feel resembles the one used for the serial robots. For serial robots different configurations are normally used in different parts of the workspace and previous joint positions are used in order to select configuration and thereby choose which solution is the most likely. The GTP however, in current setup, only uses one hardware configuration at a time, and this parameter is stored in the *robotdata.h* configuration file, which means that previous joint positions are not necessary. But as mentioned a generic Simulink interface was emphasized and

future GTP setups might use multiple hardware configuration, thats why this solutions was chosen.
Input 1: A *T44* transformation matrix given as a row major vector
Input 2: Previous joint positions as a vector input
Output 1: Joint positions, three cart positions and motor angles of joint 4,5 and 6 (imaginary). Stored in a vector
Output 2; Returns a integer error signals if errors occurred calculating the inverse kinematics. Integer zero indicates that no errors detected

6.3.3 Jacobian

Input: Joint positions, the three cart positions and the motor angles of joint 4, 5 and 6. The joint positions are given as vector input
Output 1: The Jacobian stored in a *T66* given in row major vector format
Output 2; Returns a integer error signals if errors occurred calculating the Jacobian. Integer zero indicates that no errors detected

6.3.4 Inverse Jacobian

Inputs: The Jacobian, *T66*, stored as a row major vector
Output 1: The inverse Jacobian stored in a *T44* given in row major vector format

6.4 Validation

As mentioned in 5.4 a number of steps were taken to validate the kinematic library. One of the tests conducted was gather joint values spread around the entire robot workspacevia a *RAPID* program. The joint values was then used as input to the forward kinematics block. The output was then sent to an inverse kinematics block and then compared to the input to the forward kinematics block. As shown in Fig. **??**, the residuals from were kept small.

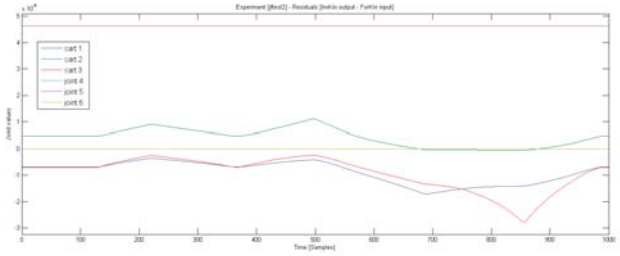

Figure 22: In the verification phase of the Gantry-Tau kinematics library residuals were studied to verify the forward and inverse kinematics. As seen the residuals are kept small.

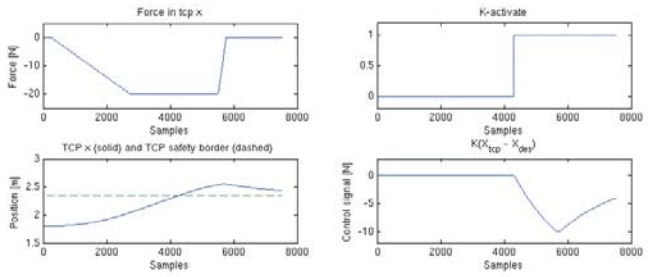

Figure 23: Safety zone implementation in simulation. *Upper left:* The force signal in the x direction in *TCP* frame. *Upper right:* The restoring force is activated as the *TCP* enters the safety zone *lower left. Lower right:* the restoring force is proportional to the perpendicular distance from the safety zone border.

7 Simulation

7.1 Simulation setup

7.2 Safety zone simulation

The safety zone was tested in simulation by introducing a force on the lead through force sensor resulting in robot movement toward the safety zone. As seen in Fig. 23 the "virtual force" and the introduced spring constant introduces resistance and gradually increases as the robot moves further into the safety zone. The lead through force is then removed to verify that the robot slowly moves back toward the safety zone limit. The K parameters is in this case used as a design parameter to adjust how aggressive the safety zone implementation is to be made.

7.3 Tool sensor force simulation

In simulation, a curve was used to define a surface. The surface is modeled as as spring, i.e, the force is proportional to the distance traveled in into the material

$$F = k\delta x \tag{40}$$

where δx is the distance traveled in the material and k is the a material dependent constant.

Let $f(x)$ be the couture of the modeled surface. To shortest distance from a point to a curve is the line perpendicular to the tangent line [11] of $f(x)$, i.e. a line with slope $\frac{1}{f'(x)}$. Assume that the shortest distance form a arbitrary point $(a, b) \quad a, b \in \Re$ to the curve $f(x)$ is the point $(x_0, y_o) \quad (x_0, y_0) \subset f(x)$. Let $g(x)$ define the linear equation describing the linear curve that intersects both (a, b) and (x_0, y_0). Since the slope of $g(x)$ is known, a equation system can be

setup to solve the equation for $g(x)$

$$g(x) = \frac{x}{-f'(x_0)} + c, \quad c \in \Re \tag{41}$$

$$(a, b) \subset g(x), \quad (x_0, y_0) \subset g(x) \quad \Rightarrow \quad c = \frac{a}{f'(x_0)} + b \tag{42}$$

The point of interest is the intersect between $f(x)$ and $g(x)$

$$f(x_0) = g(x_0) \quad \Rightarrow \quad f(x_0) = \frac{a - x_0}{f'(x_0)} + bx_0 = a - f'(x_0)(f(x) - b) \tag{43}$$

Depending on the curve $f(x)$, solving x_0 is an optimization problem. Knowing the intersection point the curve $f(x)$ that lays closest the point (a, b) the modeled force is calculated as described in 40 where the distance, d is computed using the famous *theorem of Pythagoras*

$$d = \sqrt{(a - x_0)^2 + (b - y_0)^2} \tag{44}$$

7.4 Dual sensor simulation

In simulation a constant force was applied to the lead-through sensor moving the TCP at constant speed toward the contour shown in Fig. 24. At contact the modeled force described in section 7.3 acts on the tool force sensor. It is assumed that the position and orientation of the TCP frame and force sensor frame coincide, hence no force transformation was used. Figure 24 shows results from a simulation using dual sensors. As seen the tool sensor controller, acting as a tool protection mechanism, prevents large contact forces building up. Figure 25 shows some initial oscillations at contact, probably caused by aggressive parameter setting of the PI-controller. The contact forces stabilizes at a point corresponding to the equilibrium between the control signal from the lead through sensor moving the tool into the contour and the control signal generated from the contact forces. By changing the design parameters of the tool sensor controller the magnitude of the contact forces can be tuned.

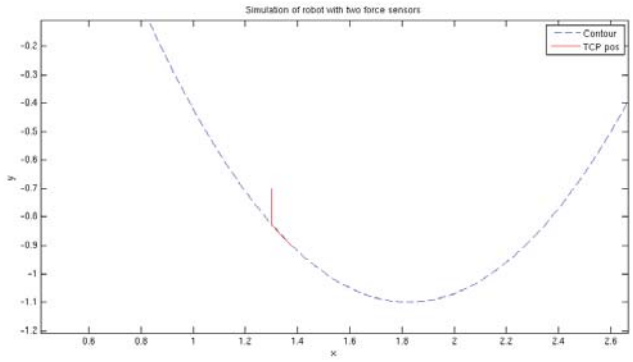

Figure 24: Simulation of dual controllers. A impedance controller was used for the lead-through force sensor and a PI controller was used for the tool sensor

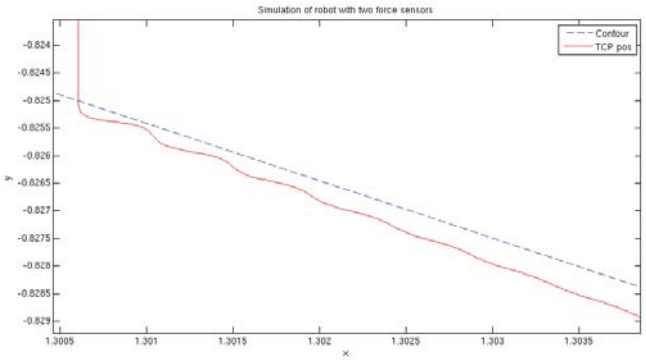

Figure 25: Zoom in of 24. Figure shows some initial oscillations at contact. These oscillation are probably caused by aggressive parameter setting of the PI-controller

8 Experiments

8.1 Drill user case scenario

A drill use case scenario was conducted to test the dual force setup. The ATI nano25 force sensor was intended as tool sensor due to better robustness to single axis overloads compared to the JR3 100M40 sensor. Initial tests however, showed very poor signal to noise ratio see Fig. 26. After trying to locate the reason for the poor performance without any success it was decided to use the JR3 sensor instead, see Fig. 27. The Simulink implementation of the controller is shown in Fig. 28.

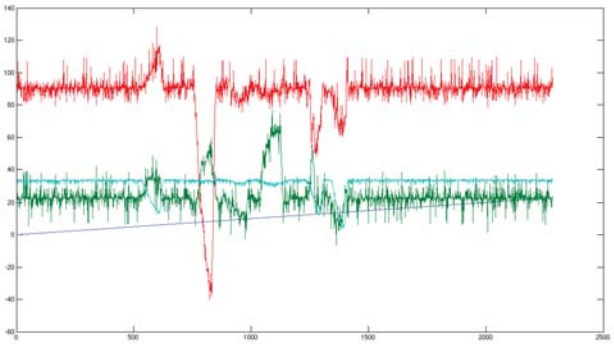

Figure 26: Shows force measurement from the ATI nano25 sensor (y-axis: force, x-axis: time). The cause of the high noise to signal ratio was not found and the JR3 100M40 sensor was used instead

8.1.1 TCP calibration

In order to transform the sensor measurement to forces acting on the tool, a TCP calibration was carried out. Figure 29 shows the relative position and orientation of the flange, sensor and TCP frames. The transformation between tool and flange was determined by a four-point TCP calibration. In a four-point calibration the robot is jogged to a fixed point in the robot work space, with four different orientations, resulting in four calibration points. For an arbitrary tool sensor orientation two additional calibration points are needed. Using four calibration points the tool frame is given the same orientation as the flange frame. The calibration resulted in the following transformation from flange to TCP

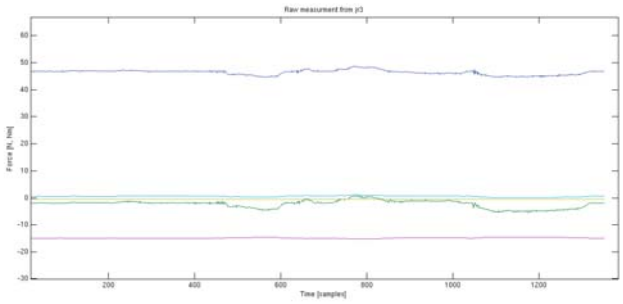

Figure 27: Shows a raw force measurement from the JR3 100M40 sensor. Compared to the raw measurements from the JR3, see Fig. 26, the JR3 sensor showed a better signal to noise ratio and was therefor used in the experiments instead for the ATI nano25 sensor.

$$H_{flange}^{TCP} = \begin{bmatrix} 1 & 0 & 0 & -71.47 \\ 0 & 1 & 0 & -58.00 \\ 0 & 0 & 1 & 294.56 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{45}$$

where the translations are given in mm.

8.1.2 Sensor to TCP force transformation

By changing the sign of the y-axis the force measurement signal was modified to achieve a right normalized orthogonal sensor frame. The transformation matrix between the modified force sensor frame to flange was approximated to

$$H_{flange}^{Sensor} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 135 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{46}$$

where the translations are given in mm.

The transformation matrices in Eq. 45 and 46 are used to derive the transformation matrix between TCP and sensor, H_{Sensor}^{TCP}, using the following relation

$$H_{TCP}^{Sensor} = (H_{flange}^{TCP})^{-1} H_{Sensor}^{TCP} \tag{47}$$

The Simulink implementation is shown in Fig. 30. Since the two transformations, TCP to flange and sensor to flange, do not depend on joint values these matrices were defined as constants in the model.

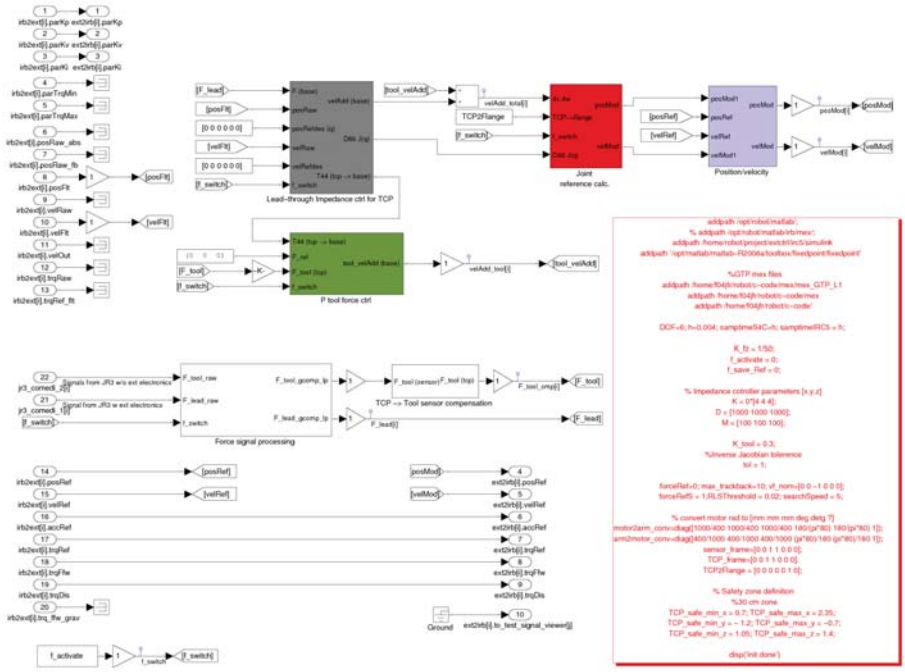

Figure 28: Shows the Simulink implementation used in the experiential setup. As inputs, the Simulink model takes signals from the IRC5 robot system, *irb2ext*, and force measurement signals, *jr3_comedi*. Two force controllers were used, one proportional controller acting as a tool protection controller and an impedance controller was used for operator lead-through. Modified position and velocity references are then returned to the robot system using the *ext2irb* outputs.

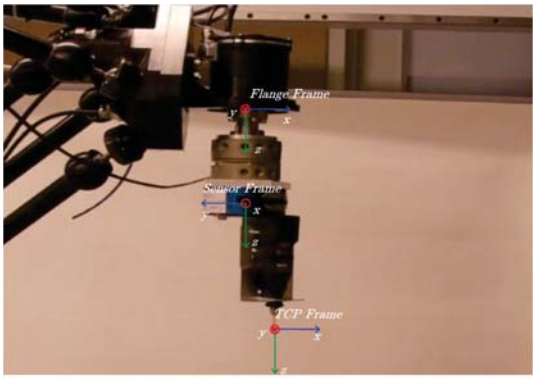

Figure 29: The figure shows the experimental setup showing the relation between flange, sensor and TCP frame. Note that the senor frame is left orthogonal.

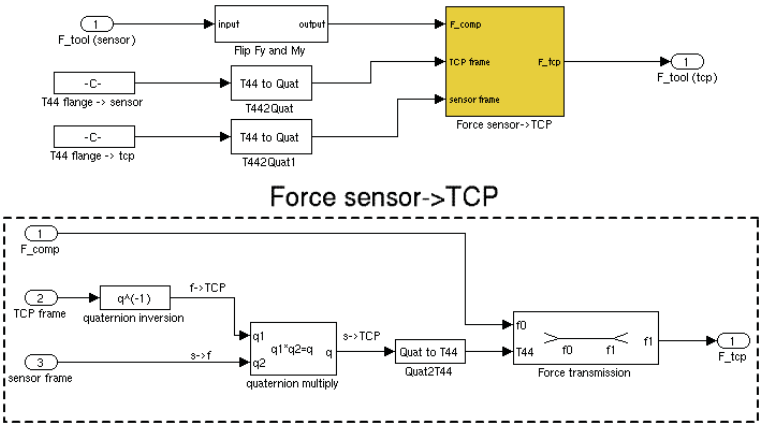

Figure 30: Figure shows the tool sensor to TCP transformation implemented in Simulink. *Upper section:* The force measured by the tool force sensor is taken as input and the force transformed to the TCP frame is given as output. *Lower section:* The implementation of Eq. 47 in a Simulink block using quaternions.

Test name	Value	Description
K_x	0.012	Proportional gain in TCP x-direction
K_y	0.012	Proportional gain in TCP y-direction
K_z	0.012	Proportional gain in TCP z-direction
$DZ_{F_{xyz}}$	+/- 0.5 N	Contact force dead zone
$DZ_{\tau_{xyz}}$	+/- 0.5 N	Torque dead zone

Table 1: Shows the parameters used during the tool protection experiment

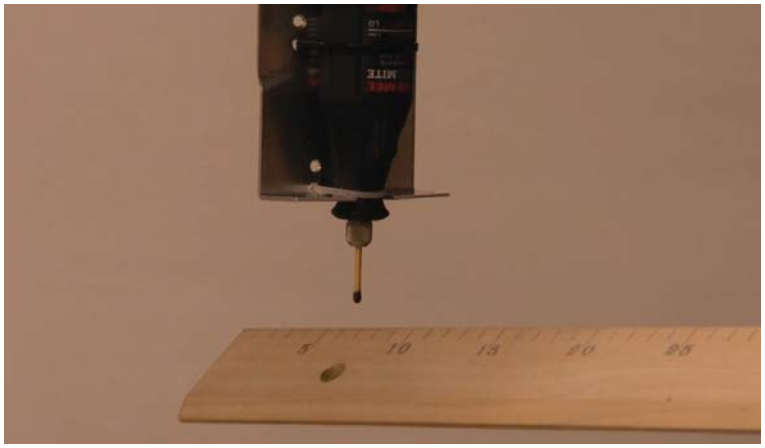

Figure 31: A tool protection experiment was conducted. The aim of the experiment was to prevent a match stick from breaking. The wooden object shown in the figure was used to subject the match stick to external forces

8.1.3 Tool protection experiment

A tool protection experiment was conducted using the setup shown in Fig 29. A dremel was used to attach a match stick. The match stick was used as a dummy tool. The goal in the experiment was to prevent the match stick breaking from external forces induced using a wooden stick, see Fig. 31. Three proportional controllers were used, one for each direction in the TCP frame. The parameters used during the experiment are shown in Table 1
The controllers were able to prevent the match from braking by generating a Cartesian velocity references to prevent too large contact forces, see Fig. 33. The velocity reference is then transformed from the TCP frame to Base frame. The velocity reference together with the Jacobian inverse is used to generate joint velocity and position references sent to the robot system.

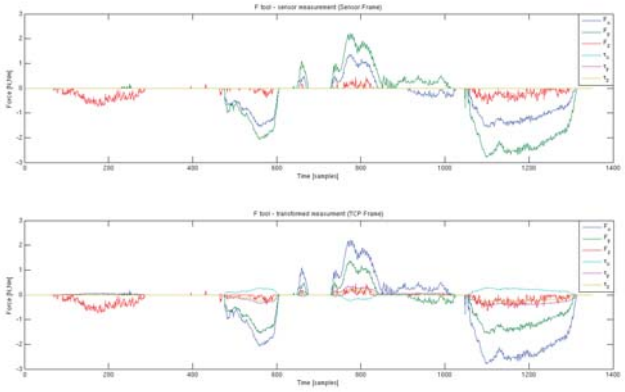

Figure 32: The figure to the *left* shows the force measurement in sensor frame. The force acting on the tool is derived using the force transformation described in section 8.1.2 is shown to the *right*

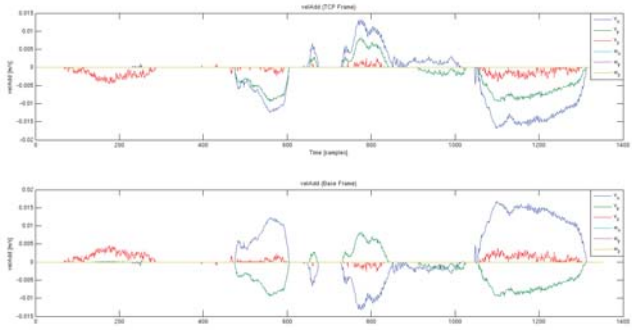

Figure 33: Shows the Cartesian velocity reference generated by the tool protection mechanism. The *left* plot shows the velocity refenence given in the TCP frame and the *right* plot shows the velocity refence given in base frame

8.2 Lead-through scenario with safety zone

The lead-through scenario was implemented with safety zone. Due to the strange behavior of the robot, caused by a bug in Opcom which was not discovered until a very late stage of this work, the lead-through testing was conducted with the the lead-through force sensor mounted outside the robot cell. Even if the force sensor was not mounted on the robot the lead-through scenario could be tested and the conclusion that it worked could be drawn. The intuitive feeling of the safety zone was however harder to test with the externally mounted force sensor, but it was possible to observe the spring force in action when approaching the work space borders.

8.3 Timing statistics

By modifying the Opcom application, richer information regarding execution time durations of the external controller could be collected. Results from the timing during experiments are presented in the appendix. For each experiment a *min duration*, *max duration* and a *average duration* is presented. Due to the code structure the average measurement is slightly different from the min and max measurements. Using the illustration shown in Fig. 20, the min and max duration measures the slowest and fastest execution of the entire opcom task, i.e. subtask 1,2 and 3. The average measurements is a measurement for the mean execution time for subtasks 2 and 3. As shown in 2 the real time performance of *LAPACK* and *SVDCMP.c* are quite similar. The worse case execution times are slightly better running *LAPACK*.

The next experiment was made to explore whether logging data had significant impact on the real time performance or not. As shown in 3. A interesting observation was that the external controller runs significantly slower with test points defined, even when no logging is performed. A possible explanation is that Real-Time Workshop compilation differs when test points are defined in order to support logging.

Test name	min[μs]	max[μs]	avg[μs]
lapack test1	80	139	65.98
lapack test2	81	139	66.25
lapack test3	81	140	65.89
svdcmp test1	81	141	65.89
svdcmp test2	80	139	66.53
svdcmp test3	79	142	65.90

Table 2: Shows the minimum, maximum and average duration execution time for three different models without test points defined. *lapack test1* runs *GTP Jacobian*, *lapack test2* runs *GTP Inv Jacobian* and *lapack test3* runs *GTP Jacobian* and *GTP Inv Jacobian* in series, all three tests with constant inputs. Equivalent models but using *GTP Jacobian2* and *GTP Inv Jacobian2* instead were used in the *svdcmp*tests

Test name	min [μs]	max [μs]	avg [μs]
svdcmp test 4.1	86	171	96.01
svdcmp test 4.2	91	167	99.04
svdcmp test 4.3	87	174	98.22
svdcmp test 4.4	86	145	75.06

Table 3: Shows the minimum, maximum and average duration execution time for a model using different logging behaviors. The model used in the test is equivalent with the model used for *svdcmp test3* in 8.3. In *svdcmp test 4.1* the logging is initiated seconds after measurements are started to observe if the initialization causes any peaks in loop duration. In *svdcmp test 4.2* logging is also started seconds after measurement are started, stopped seconds before the measurements are stopped to observe if stopping the logging causes any peaks. In *svdcmp test 4.3* logging is performed throughout the test. In *svdcmp test 4.4* no logging is performed, however test points are still defined. Interesting to note is that a model runs significantly slower with test points defined even without logging them.

Test name	min[μs]	max[μs]	avg[μs]
fkin test1	81	138	67.39
invkin test1	200	269	187.74

Table 4: Shows the minimum, maximum and average duration execution time for the forward kinematics block (*fkin test1* and the inverse kinematics block (*invkin test1*, given constant input. Note the poor performance of the inverse kinematics block. In this thesis there were no need for a inverse kinematics block in the model used for the experiments.

9 Summary

The aim of this master thesis was to develop a simulation environment for the Gantry-Tau parallel robot. The work resulted in a kinematics library including Simulink blocks for forward kinematics, inverse kinematics, Jacobian and inverse Jacobian. The kinematics library was validated and tested in both simulation and experiments. One force sensor was used to provide an impedance control based lead-through and the second force sensor supplied input to a tool protection controller. The dual sensor setup was tested in simulation. A safety zone concept design, to provide operator feedback, was also tested and validated in simulation with good results.

A tool protection experiment was conducted were a match stick was used as tool. The controller was able to prevent large contact forces building up and thereby preventing the match stick from breaking when subjected to external forces.

10 Conclusion and future work

Using the Gantry-Tau kinematics library for code generation for real-time applications on the IRC5 robot system demands good real-time performance. Timing benchmarks for the blocks included in the library shows that the kinematics blocks are somewhat CPU demanding, but the overall performance was good enough for the dual force sensors user scenario. In the scenario the inverse kinematics block was not used and this is the block with the worst real-time performance. Future work includes optimizing the C-code underlining the inverse kinematics block to improve performance in real-time applications. One suggestion in optimizing the code would be to use the fact that the configuration of the Gantry-Tau is fix, hence no need for previous joint positions. In future work a more theoretical pleasing implementation of the Jacobian should be evaluated. Instead of a imaginary sixth axis a pseudo-inverse implementation should be examined.

The drill use case scenario presented in this thesis was used mainly used to test the kinematics library. As the main effort was put in the development and validation work, work remains to improve the performance of the designed controller regarding optimizing the parameters for the scenario.

11 References

[1] The ABB Group. *http://www.abb.com*, 2009-11-07.

[2] M. W. Spong and M. Vidyasagar. *Robot Modeling and Control*. John Wiley & Sons, Inc., New York, 2005.

[3] I. Dressler. *Force control interface for ABB S4/IRC5*, LTH, 2009.

[4] N. Hogan. Impedance control: An approach to manipulation, part I theory. *J. Dynamic Systems Measurement and Control*, vol. 107, 1985, pp. 1-7.

[5] N. Hogan. Impedance control: An approach to manipulation, part II theory. *J. Dynamic Systems Measurement and Control*, vol. 107, 1985, pp. 8-16.

[6] N. Hogan. Impedance control: An approach to manipulation, part III theory. *J. Dynamic Systems Measurement and Control*, vol. 107, 1985, pp. 17-24.

[7] JR3 Inc. *http://www.jr3.com*, 2009-09-25.

[8] ATI Industial Automation. *http://www.ati-ia.com/*, 2009-12-01.

[9] M. Haage, I. Dressler, A. Robertsson, K. Nilsson, T. Brogårdh, R. Johansson. *Reconfigurable Parallel Kinematic Manipulator for Flexible Manufacturing*, Preprints 13th IFAC Symp. Information Control Problems in Manufacturing (INCOM2009), Moscow, Russia, June 3-5, 2009, pp. 145-150.

[10] L. Johannesson, V. Berbyuk, T. Brogårdh *Gantry-Tau A New Three Degrees of Freedom Parallel Kinematic Robot*, In Proc. of the 4th Chemnitz Parallel Kinematics Seminar, Chemnitz, Germany, April 20-21 2004, pp. 731-734.

[11] Gunnar Sparr *Linjär Algebra* Studentlitteratur, Lund, 1997.

[12] LINPACK. *http://www.netlib.org/linpack/*, 2009-11-10.

[13] BLAS, Basic Linear Algebra Subprograms *http://www.netlib.org/blas/*, 2009-11-10.

[14] LAPACK, Linear Algebra PACKage *http://www.netlib.org/lapack/*, 2009-11-10.