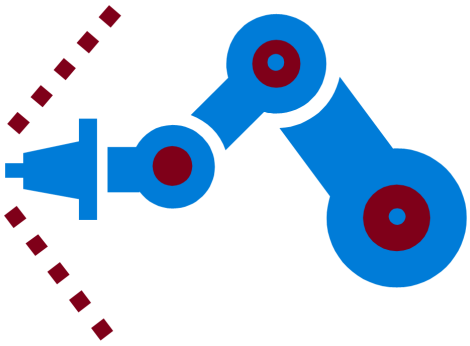
# Design specification
# Modeling and control of an industrial robot

Version 1.0

Author: AP, JK, TA
Date: October 11, 2011



**Status**

| | | |
|---|---|---|
| Reviewed | Alexander Pettersson | 2011-10-11 |
| Approved | Patrik Axelsson | 2011-10-11 |


| | | | |
|---|---|---|---|
| Course name: | Control Project Laboratory | E-mail: | toban607@student.liu.se |
| Project group: | Industrial robot | Document responsible: | AP, JK, TA |
| Course code: | TSRT10 | Author's E-mail: | alepe490@student.liu.se |
| Project: | Industrial robot | Document name: | designspecifikation.pdf |

## Project Identity


| | |
|---|---|
| **Group E-mail:** | toban607@student.liu.se |
| **Homepage:** | http://www.isy.liu.se/edu/projekt/reglerteknik/2011/industrirobot/ |
| **Orderer:** | Patrik Axelsson, Linköping University |
| | **Phone:** 013 284474, **E-mail:** axelsson@isy.liu.se |
| **Customer:** | Mikael Norrlöf, ABB Robotics |
| | **Phone:** 021 346017 , **E-mail:** mino@isy.liu.se |
| | Johan Sjöberg, ABB Corporate Research |
| | **Phone:** 013 284028 , **E-mail:** johans@isy.liu.se |
| **Course Responsible:** | David Törnqvist, Linköping University |
| | **Phone:** 013 281882, **E-mail:** tornqvist@isy.liu.se |
| | Daniel Axehill, Linköping University |
| | **Phone:** 013 284042, **E-mail:** daniel@isy.liu.se |
| **Project Manager:** | Tobias Andersson |
| **Advisors:** | André Carvalho Bittencourt, Linköping University |
| | **Phone:** 013 282622 , **E-mail:** andrecb@isy.liu.se |


## Group Members

| Name | Responsibility | Phone | E-mail (@student.liu.se) |
|---|---|---|---|
| Tobias Andersson (TA) | Project manager | 0730530440 | toban607 |
| Alexander Pettersson (AP) | Documents | 0737767682 | alepe490 |
| Jonas Källman (JK) | Design | 0739575719 | jonka615 |
| Victor Ingeström (VI) | Mechanical modeling | 0704926973 | vicin977 |
| Kristofer Klasson (KK) | Tests | 0738184392 | krikl150 |
| Gabriella Ahlbert (GA) | Information | 0705911501 | gabah362 |
| Andreas Samuelsson (AS) | | 0730651177 | andsa897 |
| Anders Gällsjö (AG) | | 0733681099 | andga726 |

## Document History

| Version | Date | Changes made | Sign | Reviewer |
|---|---|---|---|---|
| 0.1 | 2011-10-03 | First draft. | AP | André Carvalho Bittencourt |
| 0.2 | 2011-10-06 | First revison. | AP | Patrik Axelsson |
| 0.3 | 2011-10-09 | Second revison. | AP | Patrik Axelsson |
| 0.4 | 2011-10-10 | Third revison. | AP | Patrik Axelsson |
| 1.0 | 2011-10-11 | First version. | AP | Patrik Axelsson |

# Contents

# 1 Introduction

The goal of this project is to design and implement models and a control system for a robotic serial manipulator, see Figure 1. This document will present the design solutions for the various components in the system with a high level of detail. The purpose of this document is to serve as a guideline for the project group during the implementation phase of the project.

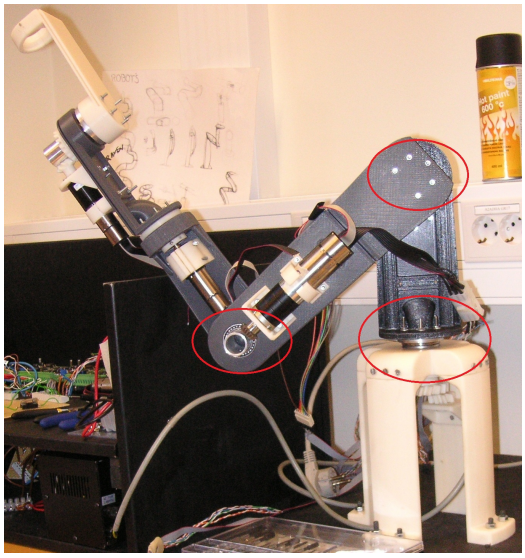


Figure 1: A picture of the miniature industrial robot.

# 2 Overview of the system

The system consists of 3 subsystems, a robot connected to a computer with a Labview interface via control electronics (that control the motors), models and a control system. An overview of the system is shown in Figure 2. The robot and the Labview interface were designed and implemented in a previous student project, which means that the design of the models and the control system will be the focus of this document.

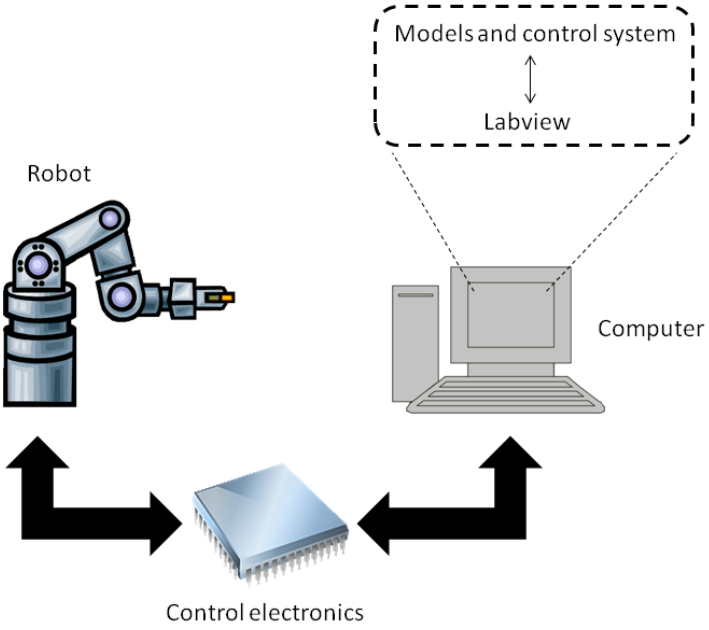



Figure 2: An overview of the system.

The robot will be modeled and controlled with 3 degrees of freedom, i.e. the 3 first joints marked in Figure 1. Thus, the tool position but not its orientation will be controlled. The computer will have Labview and Matlab/Simulink installed. Labview will be used to connect with the control electronics that are connected to the robot. The models and control system with the user interface will be implemented in Matlab/Simulink and run in Labview using the Simulation Interface Toolkit [1].

# 3 Models

In this section, the different models will be explained. The robot is modeled by the forward and inverse kinematics and by a flexible dynamic model.

## 3.1 Geometry

First, the geometry of the robot has to be defined. In Figure 3, the coordinate systems are defined (in a CAD drawing) and in Figure 4 the arm lengths are defined. All coordinate systems are Cartesian right-handed systems. $\{A\}$ is the fixed room coordinate system which the tool position will be given in. The coordinate system $\{B\}$ has its origin in the same point as $\{A\}$ and they share the $z$-axis. However, $\{B\}$ rotates around the $z$-axis and the angle between the $x$- and $y$-axes of $\{A\}$ and $\{B\}$ is denoted by $\theta_1$. The coordinate system $\{C\}$ has its origin at the bottom of the second arm and is fixed in this arm. $\theta_2$ is the angle between the $x_B$- and $x_C$-axis. The $x$-axis of $\{C\}$ is fixed along the arm and the $y$-axis is perpendicular to it. Furthermore, the second joint rotates around the $z$-axis of $\{C\}$. The coordinate system $\{D\}$ has its origin at the bottom of the third arm. Similar to $\{C\}$, the $x_D$-axis is fixed in the arm and the third joint rotates around the $z_D$-axis. The angle difference between the $x_C$- and $x_D$-axis is denoted by $\theta_3$. $\{E\}$ has the same orientation as $\{D\}$ but has its origin at the top of third arm, and serves as the coordinate system of the tool.

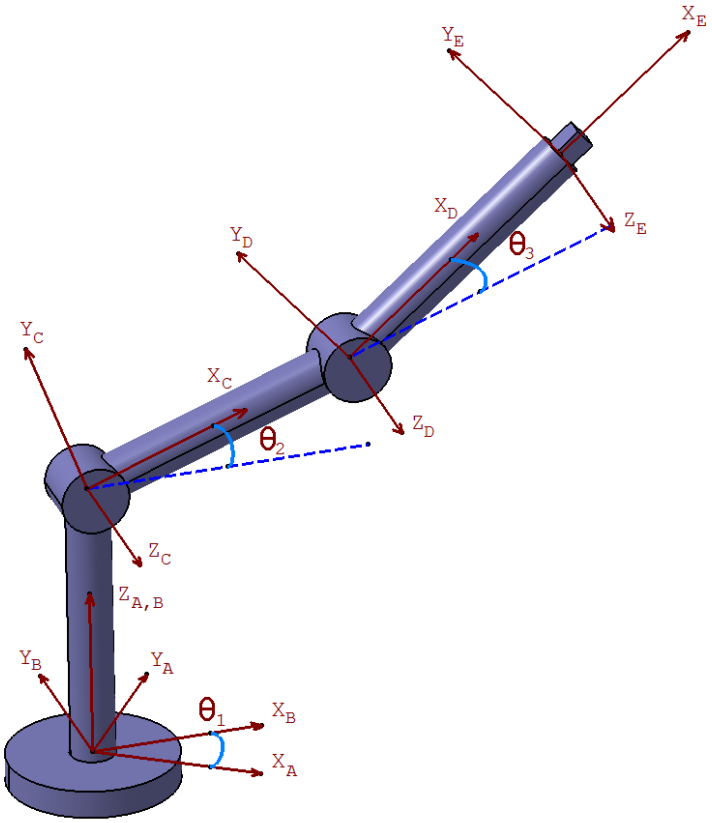



Figure 3: The geometry of the robot.

In fact, the second arm is located "on the outside" of the second and third joint, thus creating a displacement in the $y_A$-direction. The third joint is however displaces the third arm the same distance in the opposite direction, so that the second and third joint are aligned in the $x_C$-axis. Consequently, the geometry can be regarded as the one defined in Figure 3.

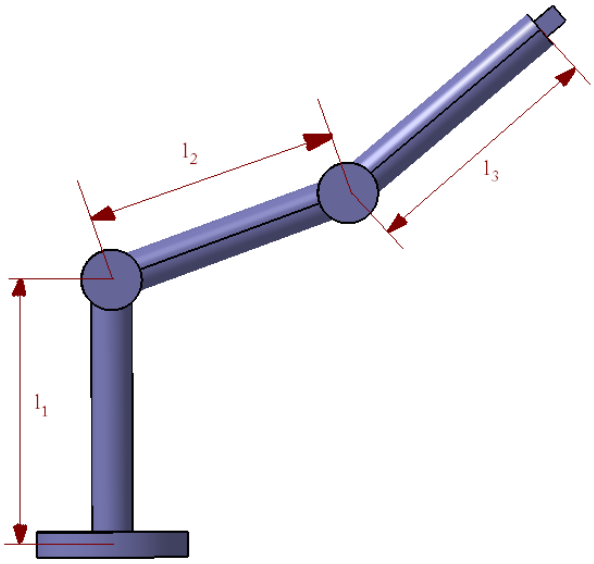



Figure 4: Definition of arm lengths.

## 3.2 Forward kinematics

The forward kinematics is a model that calculates the tool position in room coordinates given the angles of the joints [2]. The orientation of a coordinate system (CS) $\{B\}$ relative to another CS $\{A\}$ can be described by a rotational matrix,

$$ {}^A_B R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{3.1} $$

If a coordinate system is both translated and rotated relative to another, this can be described by a 4×4-matrix,

$$ {}^A_B T = \begin{bmatrix} {}^A_B R & {}^A P_T \\ 0 & 1 \end{bmatrix}, \tag{3.2} $$

where ${}^A P_T$ is the translation of the origin of $\{B\}$ expressed in $\{A\}$ as a 3×1-vector. With this notation, a point $P$ in CS $\{B\}$ can be expressed in $\{A\}$ by

$$ {}^A P = {}^A_B T \, {}^B P. \tag{3.3} $$

For the geometry defined in Section 3.1, the transformation matrices for the coordinate systems become

$$ {}^A_B T = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.4} $$

$$ {}^B_C T = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.5} $$

$$ {}^C_D T = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & l_2 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.6} $$

and

$$ {}^D_E T = \begin{bmatrix} 1 & 0 & 0 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.7} $$

where $l_i$, $i = 1, 2, 3$, are the arm lengths. The tool position in room coordinates is given by

$$ {}^A P = {}^A_E T \, {}^E P = {}^A_B T \, {}^B_C T \, {}^C_D T \, {}^D_E T \, {}^E P, \tag{3.8} $$

where ${}^E P$ is the tool center position in CS $\{E\}$. For example, with

$$ {}^E P = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T, \tag{3.9} $$

the tool center position as function of the joint angles can be determined from (3.8) and becomes

$$ {}^A P = \begin{bmatrix} \cos\theta_1(\cos\theta_2(l_3\cos\theta_3 + l_2) - l_3\sin\theta_2\sin\theta_3) \\ \sin\theta_1(\cos\theta_2(l_3\cos\theta_3 + l_2) - l_3\sin\theta_2\sin\theta_3) \\ l_3\cos\theta_2\sin\theta_3 + \sin\theta_2(l_3\cos\theta_3 + l_2) + l_1 \\ 1 \end{bmatrix}. \tag{3.10} $$

## 3.3 Inverse kinematics

The inverse kinematics model will give every possible joint angle combination to reach a given point ${}^A P = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$. For every reachable point except where $x = y = 0$ there are two possible values of $\theta_1$; one where the x-axis in CS $\{B\}$ is directed towards the point and one where it is directed away from the point. For the special case $x = y = 0$ the point is reachable for any values of $\theta_1$.

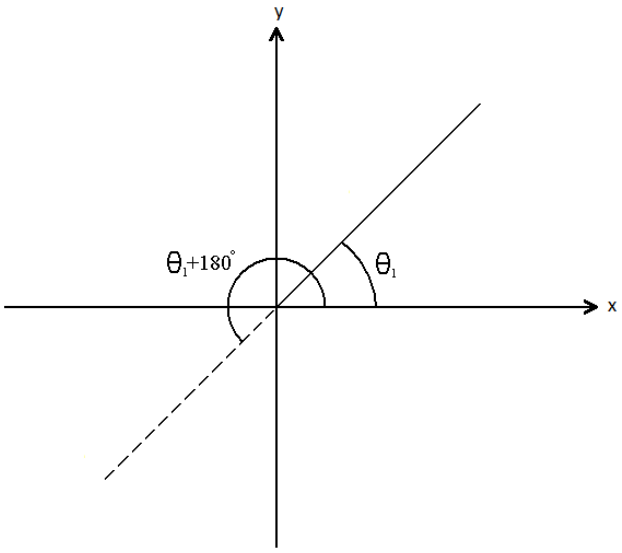



Figure 5: The two possible sets of $\theta_1$.

As seen in Figure 5, $\theta_1$ can be expressed as

$$\theta_1 = \text{atan2}\,(y, x)\,, \tag{3.11}$$

where atan2 is a Matlab command that keeps track of which quadrant the point is in. For every value of $\theta_1$ there are two possible sets of $\theta_2$ and $\theta_3$, except the case where $\theta_3 = 0$, then there is only one set of angles for every $\theta_1$. This means that there are four possible sets of angles in most cases. To derive $\theta_2$ and $\theta_3$ help angles and vectors need to be defined, as shown in Figure 6.

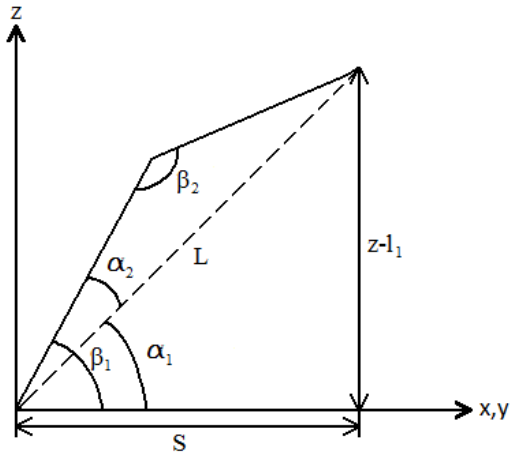



Figure 6: Definition of the help angles.

In the plane shown in Figure 6 the distances $S$ (which is in the $xy$-plane) and $L$ are

calculated as

$$S = \sqrt{x^2 + y^2}. \tag{3.12}$$

and

$$L = \sqrt{S^2 + (z - l_1)^2}. \tag{3.13}$$

According to Figure 6 and the law of cosines, the help angles can be expressed as

$$\alpha_1 = \operatorname{atan2}\left(z - l_1, x\right), \tag{3.14}$$

$$\alpha_2 = \arccos\left(\frac{l_1^2 - l_2^2 + L^2}{2 l_1 L}\right), \tag{3.15}$$

$$\beta_1 = \alpha_1 + \alpha_2 \tag{3.16}$$

and

$$\beta_2 = \arccos\left(\frac{l_1^2 + l_2^2 - L^2}{2 l_1 l_2}\right). \tag{3.17}$$

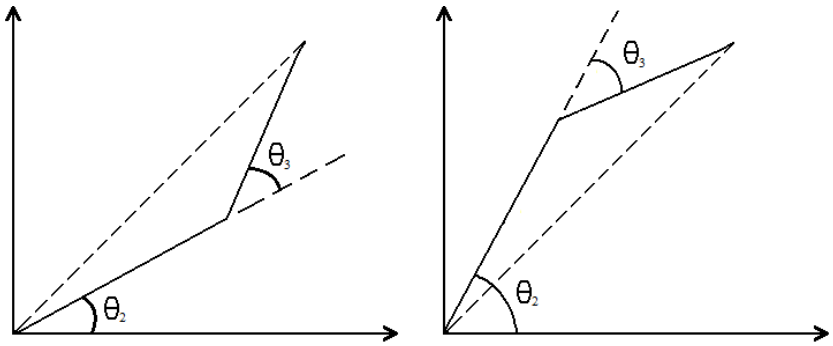



Figure 7: Two possible sets of the angles $\theta_2$ and $\theta_3$.

By combining Figure 6 and 7, one get

$$\theta_2 = \alpha_1 + \alpha_2 \tag{3.18}$$

and

$$\theta_3 = 180 - \beta_2. \tag{3.19}$$

With these formulas, the joint angles can be calculated from the position and therefore defines the inverse dynamics.

## 3.4 Flexible dynamic model

The flexible dynamic model explains the motion of the robot when forces act on it. The model is on the form

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) + D(\dot{q}) + K(q) + F(\dot{q}) = \tau, \tag{3.20}$$

where $q = [q_a \;\; q_m]^T$ are the arm and the motor angles, $M$ is the joint-space inertia matrix, $C$ includes Coriolis and centrifugal forces, $G$ includes external forces (gravity), $D$ includes damping, $K$ includes spring forces and $F$ includes friction. $\tau$ is the applied torque [2].

The rigid body contributions to the dynamic model, which is the $M$, $C$ and $G$ matrices, will be calculated using the Newton – Euler dynamic formulation as specified in [2]. All necessary iterations and equations to calculate $M$, $C$ and $G$ will be calculated using Mupad which is a symbolic math toolbox in Matlab [3]. All the robot link properties such as mass, inertia tensor and center of gravity are given in a data sheet provided by IEI, Linköping University.

When adding the flexible matrices to the model, some approximations can be made. One approximation made is that the angular velocity of the rotors is only due to their own spinning. This simplifies the dynamic equations considerably as the dynamic couplings between the rotors and their links, only depend on the elastic torque transferred through the gearbox [6]. The elastic torque is modeled as

$$\tau_J = K(q_a - q_m), \tag{3.21}$$

where $q_a$ is the arm angles and $q_m$ is the motor angles which are transformed to the arm side via the gear ratio. $K$ is a positive definite, diagonal matrix containing the joint stiffnesses for each gearbox.

The damping in the gearboxes are modeled as

$$D(\dot{q}_a - \dot{q}_m), \tag{3.22}$$

where $D$ is a positive definite, diagonal matrix containing the damping coefficients of the gearboxes.

The friction is modeled as

$$F\dot{q}_m \tag{3.23}$$

where $F$ is a positive definite, diagonal matrix containing the friction coefficients. Both the link and motor angle velocities will affect the total friction but for convenience, the total effect of friction is collected and assumed to only be dependent on the motor angle velocities.

Gathering equations (3.20), (3.21), (3.22), (3.23) and rewriting them as one equation explaining the link movement and one equation explaining the motor movement results in

$$M_a(q_a)\ddot{q}_a + C_a(q_a,\dot{q}_a) + G_a(q_a) + K_a(q_a - q_m) + D_a(\dot{q}_a - \dot{q}_m) = 0 \tag{3.24}$$

and

$$B\ddot{q}_m - K_m(q_a - q_m) - D_m(\dot{q}_a - \dot{q}_m) + F_m(\dot{q}_m) = \tau, \tag{3.25}$$

where $B$ is a positive definite, diagonal mass matrix containing the moments of inertia for the motors. It is very important to note that both equations (3.24) and (3.25) are described on the link side of the robot. This means that $q_m$, $\tau$ and the moments of inertia in the $B$ matrix must be multiplied with the corresponding gear ratio. One can also note that the sign of the $K$ and $D$ matrices are different in the equations. This is natural considering that the $K$ and $D$ matrices in the arm equation is the reaction of the $K$ and $D$ matrices in the motor equation.

System identification will be used for unknown parameters in (3.24) and (3.25). The unknown parameters are the coefficients for the spring, damping and friction. This will be achieved by applying different (calculated) torques and thus obtaining different sets of angles, angular velocities and angular accelerations. By using least square methods or similar the parameters can be estimated.

The dynamic model is an important part of the project as it will be used in many instances later on, e.g. in simulation. To simulate the robot in Simulink, both the arm and motor angle accelerations in (3.24) and (3.25) must be solved by rewriting the equations to

$$\ddot{q}_a = M_a(q_a)^{-1}(-C_a(q_a, \dot{q}_a) - G_a(q_a) - K_a(q_a - q_m) - D_a(\dot{q}_a - \dot{q}_m)) \tag{3.26}$$

and

$$\ddot{q}_m = B^{-1}(\tau + K_m(q_a - q_m) + D_m(\dot{q}_a - \dot{q}_m) - F_m(\dot{q}_m). \tag{3.27}$$

Equations (3.26) and (3.27) calculates the arm and motor angle accelerations given the current robot configuration and torque input. The arm and motor angles and angle velocities can then be obtained by integrating the angle accelerations.

# 4 Control system

The control system will be used to control the movement of the joints. The controllers will be implemented in Matlab along with the models of the system so that testing can be performed before the controller is used on the robot. In Figure 8, a block diagram of the control system is shown. The trajectory planner calculates a reference signal on the form of the desired joint angles, $q_{a,ref}$, and is transformed to reference motor angles, $q_{m,ref}$, by the reference generator. The input to the torque controller will be $q_{a,ref}$, $q_{m,ref}$ and the measured motor angles $q_m$. The torque controller output will be the desired torque $\tau$ for the motors, which will be transformed into a current $i$ by the current controller. This current will be applied to the motors to obtain the right angles. The different blocks will be explained in more detail later in this chapter.

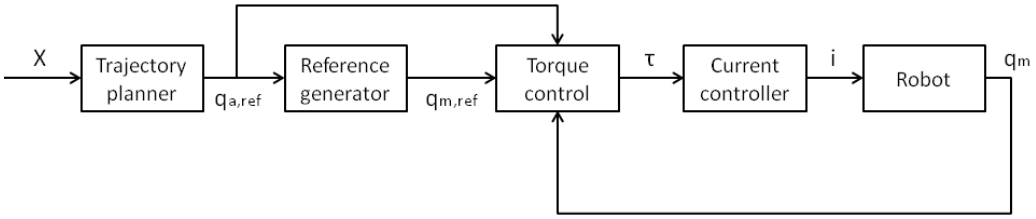



Figure 8: Block diagram of the control system.

## 4.1 User interface

The graphical user interface (GUI), which can be seen in Figure 9, will be implemented in Matlab and send the movement orders to the trajectory planner. The user interface has two modes, automatic and manual. The automatic mode will be used to send movement orders to the robot, either a joint target or a target in the Cartesian coordinate system.

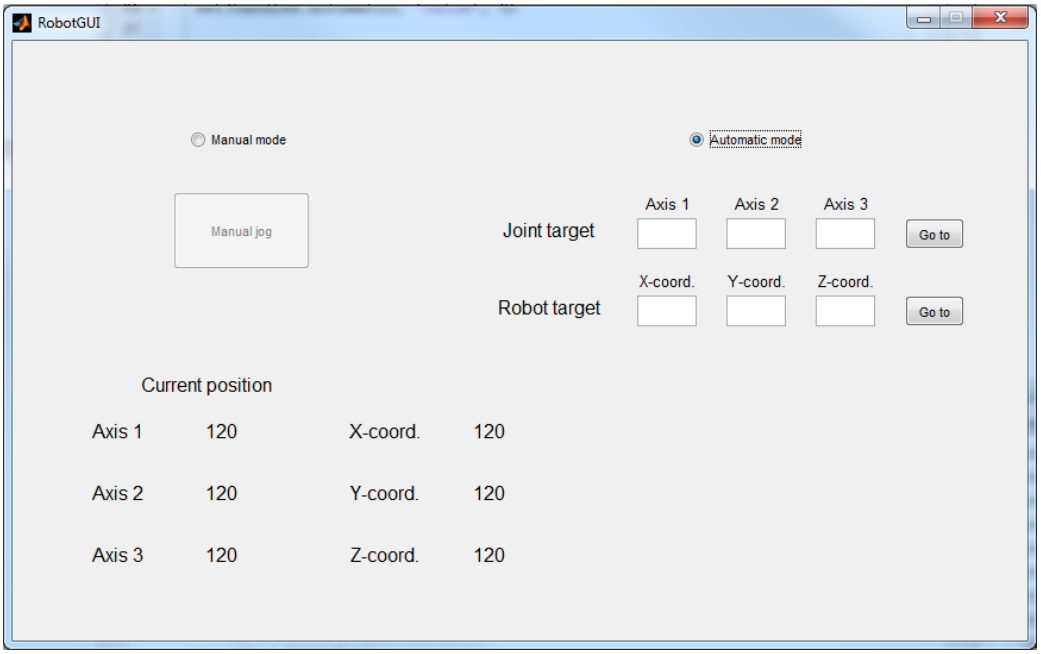



Figure 9: The user interface.

The manual mode will be used to jog one joint at a time by using the computer's keyboard. To make sure that the robot is not jogged by accident when in manual mode, a button has to be pushed in the GUI to open a second window that will detect the key strokes, see Figure 10. The *angle offset* setting decides the change in degrees for a joint angle when a key on the keyboard is pressed once.

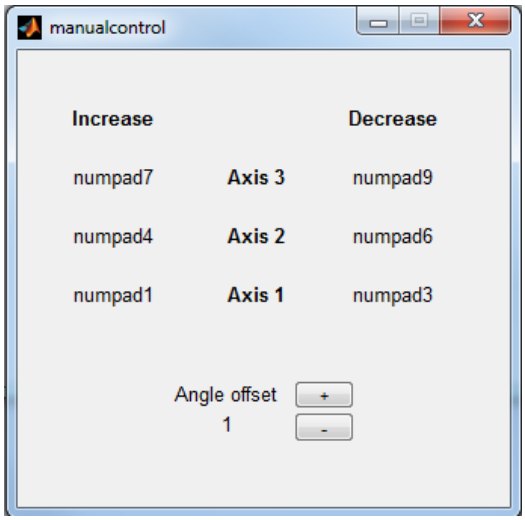



Figure 10: Manual control window.

## 4.2 Manual control

The manual control will not have a target position. The user can jog the different joints of the robot, by decreasing or increasing the joint angle until it reaches the desired position. The robot will then hold that position until a new order is given. Each keystroke will send an order to the trajectory planner to rotate the joint a given angle, which will be defined by the user as explained in section 4.1. Furthermore, the speed of the manual mode will be reduced to 25 % compared to joint control in automatic mode.

## 4.3 Torque control

The torque control will consist of a feed forward loop and a feedback loop, as shown in Figure 11.

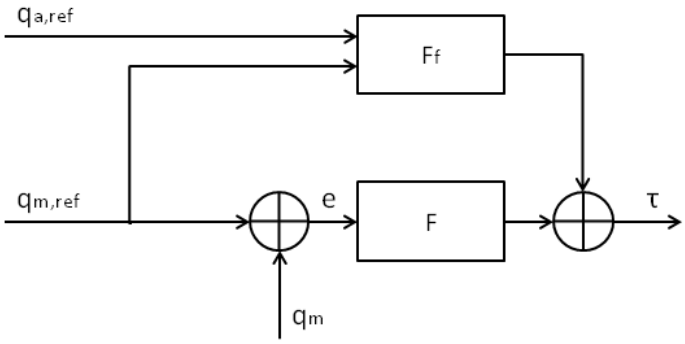



Figure 11: Block diagram of the torque control.

The control error $e$ is calculated as the difference between the reference and the measured motor angles,

$$e = q_{m,ref} - q_m, \tag{4.1}$$

and will be used as input to the feedback controller $F$. $F$ will be a diagonal PID controller and generates a torque as control signal.

The rigid dynamics model (see Section 3.4) will be used to feed forward the reference angles $q_{a,ref}$ and $q_{m,ref}$, and the output from the feed forward block ($F_f$) will therefore also be a torque. The output signals from $F$ and $F_f$ will be summarized to the total desired torque for the motors. The desired torque will be the input to the next block, the current controller.

With a feed forward loop, the motors will generate a torque even when the control error is zero, so that the robot can maintain its angles in steady state. Furthermore, with a feed forward loop, the feedback controller $F$ can have a simple structure, i.e. a PID controller.

## 4.4 Current controller

Connected in cascade with the feed forward and the PID controller is a converter and a controller for the motor current. The torque given a current is modeled as

$$\tau_m = K_m i_a \tag{4.2}$$

where $\tau_m$ is the torque, $K_m$ is the torque constant and $i_a$ is the motor current [5]. $K_m$ has to be estimated by conducting experiments. In Figure 12, a block diagram of the current controller is shown. By inverting (4.2), the desired current $i_{ref}$ can be calculated. With a feedback from the actual current, a control error can be calculated and used as input to a PID controller. The output from the controller will be the applied current to the motors. Pulse-width modulation, PWM, will be used to obtain the current. The torque-current loop will be much faster than the other loops in the control system, therefore the current controller can be seen as a simple block that calculates the current from the desired torque as in Figure 8.

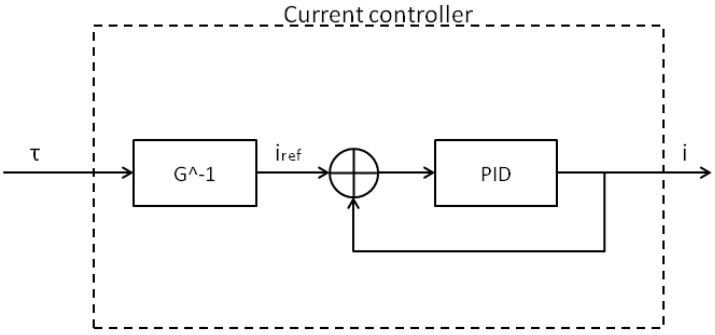



Figure 12: The current controller.

## 4.5 Trajectory planner

The trajectory planner works in two different modes. One where the robot moves the tool in a straight line from point A to point B and one when the joints move from one set of angles to another.

### 4.5.1 Moving the tool in a straight line


Figure 13: Block diagram of the trajectory planner.

The trajectory planner receives a target position and a maximum speed from the user interface (position B in Figure 14 and $v_{max}$ in Figure 16). Based on the current position of the robot's tool (position A in Figure 14), the length between position A and B can be calculated ($S_{AB}$). The length is then divided into $n$ small bits of size $\Delta S$, where bit number k has the position $S_k = S[k] = k\Delta S$ on the $S_{AB}$.

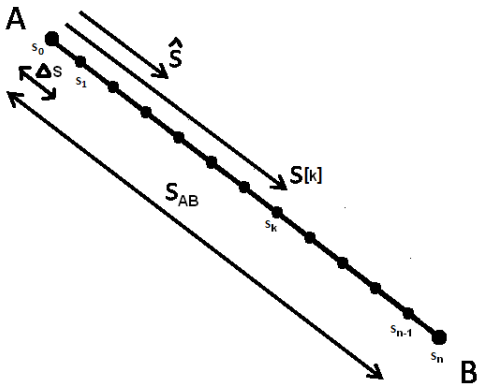



Figure 14: A trajectory between tool position A and B.

For each position in the position vector, $S_k$, the joint angles, $q_a$, are calculated by using the inverse kinematics and selecting the best set of angles of the ones that come out and putting them in a so called angle vector, $q_k = q[k]$, so that $q_k$ contains the joint angles for the different positions in $S_k$. This step is done by using the PGT toolbox [4] and an illustrative example is shown in Figure 15.

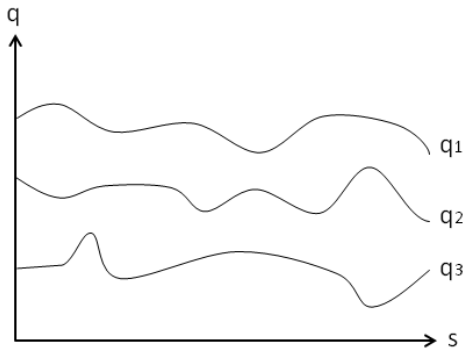



Figure 15: Joint angles as function of distance.

The robot tool should move along $S_k$ with the given speed $v_{max}$, but because the motors do not have infinite acceleration, the tool speed will optimally follow the curve in Figure 16.

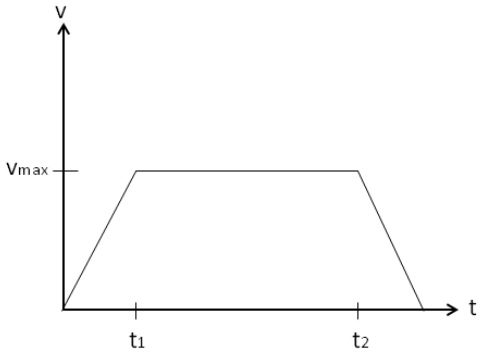



Figure 16: Velocity of the robot as function of time.

To get a smooth path trajectory for the tool movement along the line between A and B a Cubic or Quintic polynomial trajectory is calculated as described in Chapter 5.6.1 in [5]. The result of the polynomial which describes the tool position along $S_{AB}$ over time can be seen in Figure 17

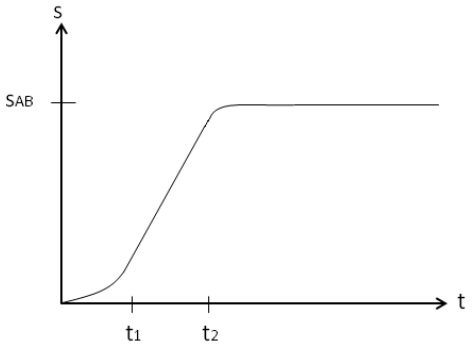



Figure 17: The tool's traveled distance as function of time.

With knowledge about the joint angles along $S_{AB}$ and the tool position along $S_{AB}$ over time, a mapping of the joint angles over time can be made, as seen in Figure 18. This is what is sent as a reference to the torque controller.

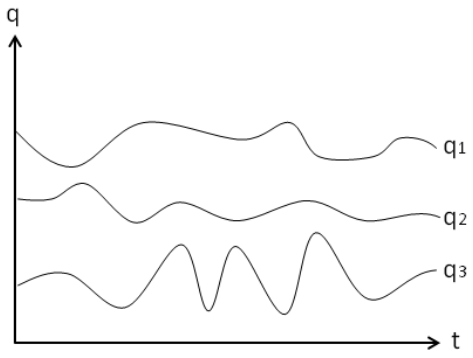



Figure 18: Angles as function of time.

### 4.5.2 Moving the joints from one set of angles to another

The trajectory planner receives a set of angles from the user interface. In this mode the angles goes straight to the new angle, making it possible to calculate and apply a quintic or cubical polynomial trajectory on each motor in order to obtain a smooth trajectory. If the polynomial trajectory is made with the same initial values accept start and stop angles, all the motors will arrive at their target angle at the same time. An example of the result of the polynomial which describes angles and the angular velocity over time can be seen in figure 19. This is what is sent as a reference to the torque controller.

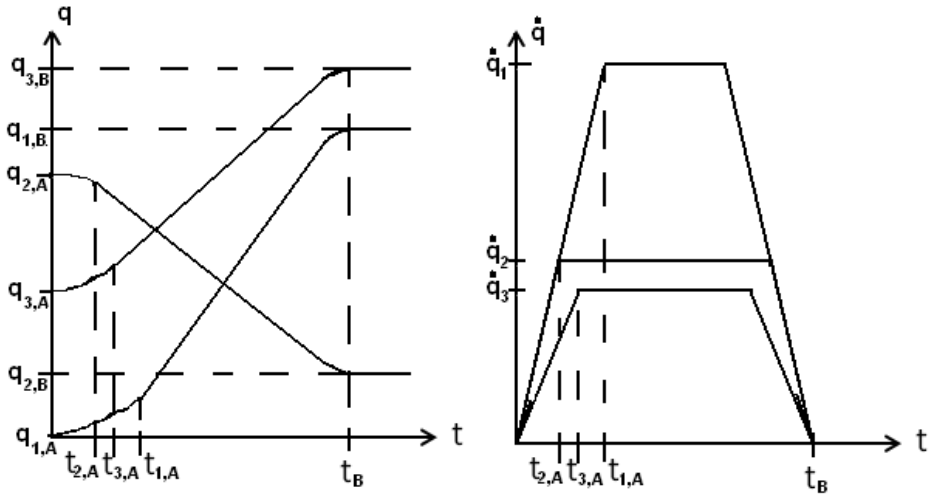



Figure 19: Angles (left) and angular velocity (right) as function of time.

## 4.6 Implementation

The robot is connected with the NI cRIO-9022 Real-Time Controller, which is connected to the computer with an Ethernet cable. NI 9022 is controlled by Labview, but the control system and models will be implemented in Simulink. Consequently, Labview must run the Simulink models in real time. This will be achieved by using the Simulation Interface Toolkit [1], which is a toolkit for Labview that makes it possible to run Simulink models in Labview by compiling the model to a real time target written in C.

# References

[1] National Instruments, *NI LabVIEW Simulation Interface Toolkit*, 2009.

[2] Craig, John J., *Introduction to robotics: Mechanics and control*. Pearson Education, 3rd Edition, 2005.

[3] The Mathworks Inc., *MuPAD*, 2011.

[4] Nyström, Maria and Norrlöf, Mikael, *PGT - A path generation toolbox for Matlab (v0.1)*, Department of Electrical Engineering, Linköping University, 2003.

[5] M. W. Spong, S. Hutchinson, M. Vidyasagar, *Robot Modeling and Control*. John Wiley & Sons, 1st Edition, 2005.

[6] Bruno Siciliano, Oussama Khatib, *Springer handbook of robotics*, 2008.