Multipurpose Robot Arm

ALEXANDER ARONSSON
FAHIM PIRMOHAMED
Multipurpose Robot Arm

Bachelor’s Thesis at ITM

ALEXANDER ARONSSON
FAHIM PIRMOHAMED

Bachelor’s Thesis at ITM
Supervisor: Nihad Subasic
Examiner: Nihad Subasic

TRITA-ITM-EX 2021:34
Abstract

Today’s society is facing a large increase of automation and smart devices. Everything from coffee machines to fridges include some kind of electronics and embedded systems.

The focus of this Bachelor’s thesis was to dive deeper into how these automated devices can be controlled and more specifically a robot arm. The main purpose revolved around constructing a robotic arm that could be controlled through three different methods using MATLAB. These three were manual control, numerical analysis control and with a neural network based control. The prototype was created by assembling six servo motors onto 3D-printed parts. The arm consisted of three main parts which were a base, an arm and a gripper. The system was controlled by an Arduino micro-controller connected to a computer.

The results show that the manual control method was easy to implement, fast and reliable. It allows control of all the angles for each servo motor, which also means controlling each individual degree of freedom. The numerical way, using Newton-Raphson’s method, broadened the abilities to control the arm but was slower. The third and final solution was to use fuzzy-logic. This ended up being a powerful method allowing for great control with low latency. While unreliable, the method showed great potential and with refinement could surpass the others.

The conclusion was that the neural network method was the overall best method for controlling and manoeuvring the robot arm using MATLAB.

Keywords: Mechatronics, Robot Arm, MATLAB, Fuzzy logic, Microcontroller
Referat

Multifunktions robotarm

Dagens samhälle står inför en stor ökning av automatisering och smarta produkter. Allt från kaffemaskiner till kyl och frys innehåller någon form av elektronik och inbäddade system.

Det huvudsakliga syftet med detta kandidatexamsarbete var att gräva djupare i hur dessa automatiserade produkter kan kontrolleras och mer specifikt i detta fall, en robotarm. Projektet handlade om att konstruera en robotarm som kunde styras och kontrolleras genom tre olika metoder i programmet MATLAB. Dessa tre har vi valt att kalla manuellt kontroll, numerisk kontroll och neuralt nätverksbaserad kontroll. Prototypen tillverkades genom att montera sex servomotorer på 3D-utskrivna delar. Armen bestod av tre huvuddelar, en bas, en arm och en gripklo. Systemet styrdes av en Arduino mikrokontroll ansluten till en dator.


Slutsatsen var att det neurala nätverket var den generellt bästa metoden för att kontrollera och manövera robotarmen via programmeringsprogrammet MATLAB.

Nyckelord: Mekatronik, Robotarm, MATLAB, Fuzzy logic, Mikrokontroller
Acknowledgements

First of all, we would like to express our deepest gratitude towards our supervisor, Nihad Subasic, for his invaluable guidance, help and feedback throughout this thesis project. Additionally, we would like to thank Staffan Qvarnström and Thomas Östberg for their help providing everything from components, to thoughts and ideas when needed.

Alexander Aronsson & Fahim Pirmohamed
May 2021
Contents

1 Introduction ............................................. 1
   1.1 Background ........................................... 1
   1.2 Purpose ............................................. 1
   1.3 Scope ............................................... 1
   1.4 Method ............................................. 2

2 Theory .................................................. 4
   2.1 Microcontrollers ..................................... 4
   2.2 Servo-motors ........................................ 5
   2.3 Newton Raphson’s Method ............................ 6
   2.4 Fuzzy logic .......................................... 7
   2.5 Degrees of freedom .................................. 8
   2.6 Kinematics .......................................... 9

3 Prototype .............................................. 10
   3.1 Electronics ......................................... 10
      3.1.1 Microcontroller ................................ 10
      3.1.2 Servo-motors ................................... 11
      3.1.3 Power supply .................................... 11
   3.2 Hardware ........................................... 11
      3.2.1 Base unit ....................................... 12
      3.2.2 Arm unit ........................................ 12
      3.2.3 Gripper .......................................... 13
   3.3 Software ............................................ 13
      3.3.1 Manual control .................................. 14
      3.3.2 Numerical approach ............................. 15
      3.3.3 Neural network approach ....................... 15

4 Results ................................................. 16
   4.1 Manual Control ...................................... 16
   4.2 Numerical approach ................................ 17
   4.3 Neural network ...................................... 18

5 Discussion ............................................. 19
List of Abbreviations

2D Two Dimensional
3D Three Dimensional
ANFIS Adaptive Neuro Fuzzy Inference System
CAD Computer-Aided Design
DC Direct Current
IDE Integrated Development Environment
I/O Input/Output
DOF Degrees Of Freedom
KTH Kungliga Tekniska Högskolan
PLA Polyactic Acid
PWM Pulse Width Modulation
RAM Random Access Memory
ROM Read Only Memory
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Method for developing software, drawn in Pages</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Arduino Uno Rev3 [4]</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Illustrative graph of Newton Raphson’s method [12]</td>
<td>6</td>
</tr>
<tr>
<td>2.4</td>
<td>Fuzzfing input value [15]</td>
<td>7</td>
</tr>
<tr>
<td>2.5</td>
<td>Free-body in 3D and 2D Space [16]</td>
<td>8</td>
</tr>
<tr>
<td>2.6</td>
<td>Modified schematic picture for our prototype [16]</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>Circuit diagram for our prototype made in Tinkercad</td>
<td>10</td>
</tr>
<tr>
<td>3.2</td>
<td>Completed construction, captured with iPhone 12 Pro by authors</td>
<td>12</td>
</tr>
<tr>
<td>3.3</td>
<td>Schematic figure for robot arm, drawn in Pages</td>
<td>13</td>
</tr>
<tr>
<td>3.4</td>
<td>Manual interface made in MATLAB app-designer</td>
<td>14</td>
</tr>
<tr>
<td>4.1</td>
<td>Representation of controlling one servo at a time to reach end-point, drawn in Pages</td>
<td>17</td>
</tr>
<tr>
<td>4.2</td>
<td>Comparison of coordinate system, left picture from Mathworks [20], right picture made in draw.io</td>
<td>18</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Evaluation of methods. .............................................. 16
Chapter 1

Introduction

1.1 Background

A crucial part of today’s automated factories is the robotic arm. Useful in a variety of ways from assembling car parts to cutting using a CNC head. These arms usually have six degrees of freedom, allowing them to move throughout 3D space. The early robotic arms performed mostly simple tasks such as making the same repetitive weld over and over [1]. However as technology advanced they could do a much wider variety of tasks depending on the tool attached to their head. Using sensors to detect flaws and imperfection while changing tools on the go. This proved to be very valuable and soon became a staple in every modern factory. The key advantages that the robot arms bring to the factories are, improved cost efficiency, decreased production time and improved quality [2]. It is predicted that robot arms will serve an even more important role in the future coupled with AI, not only in factories but also in our households [3].

1.2 Purpose

The purpose of this Bachelor’s Thesis project is to construct a prototype of a multipurpose robotic arm that is able to be controlled using different methods in MATLAB. The prototype is expected to be able to perform desired movements and simple tasks. The research question aimed to be answered are the following:

- How can the robot arm be controlled using MATLAB?

1.3 Scope

This project had limited resources which therefore put boundaries on the prototype. The main focus is to construct a fully working robot arm that has the ability to perform a limited amount of different tasks. Time is neither wasted on making a large scale arm that can move larger objects. It is rather focused on
making a small model that shows the potential and principle of our robot. The controller that will be using is an Arduino microcontroller [8]. The prototype will be constructed with five degrees of freedom while the computational methods will be limited to two. This is to simplify but also allow for further development in the future.

1.4 Method

The method that has been used in this Bachelor’s thesis consisted of three phases.

- **Information gathering**
  The first phase was at an early stage where information was gathered on how to construct a robot arm and what components that were necessary in doing so. Time was also spent on studying what components that were not crucial to complete the aimed research. Among the necessary components were the Arduino, servo motors and micro servos.

- **Prototype construction**
  The second phase consisted of designing the arm in CAD and 3D-printing the drawings to make a real life prototype. Relevant CAD-files were found online and modified for this project specifically [5]. This ended up saving time since there was no need to design the arm from the ground and up. This part also consisted of assembling the electronics, that is, connect the servos to the Arduino. The parts were later put together with the rest of the major components by assembling the PLA parts with the two servo-motors, the claw and the three micro servo-motors.

- **Software development**
  The third and final phase consisted of programming the Arduino to perform desired tasks. This was first done with Arduinos IDE to limit the amount of variables and check that all the servos were turning. Later, focus was shifted onto the use of MATLAB and their Arduino support package [6]. This provided flexibility and interactivity in controlling the robot arm as MATLAB eases the designing of apps and performing calculations. Three control methods were developed: manual control, numerical analysis based control and neural network based control. While developing these methods the same process was used for all of them. This process can be seen in figure 1.1.
1.4. METHOD

![Diagram of method for developing software](image)

**Figure 1.1.** Method for developing software, drawn in Pages.
Chapter 2

Theory

2.1 Microcontrollers

A microcontroller could be described as a very small computer. More technically a small microcontroller typically includes the following [7]:

- Central processing unit
- Memory for the program Read-only memory (ROM) that retains its data even when power from the microcontroller is removed.
- Memory for data This is known as random-access memory (RAM) and changes its data during the course of the microcontroller’s operation
- Address and data buses These link the subsystems of the microcontroller and transfers data together with instructions.
- Clock Keeps all the systems of the microcontroller in sync.

The microcontroller that is used in this project is the Arduino Uno which is a single-board microcontroller based on the ATmega328P microchip. It has 14 Digital I/O Pins, six of which that can utilize PWM. There are also six Analog Input Pins which can also be used as Digital I/O Pins however they also have an A/D converter with 10-bits resolution [8]. This makes them optimal to use for sensory input that varies the voltage with the reading, for example a light sensitive resistor.

The recommended input voltage of the Arduino Uno is 7-12V and the operating voltage is 5V. The DC Current per I/O Pin is 20 mA and for the 3.3V Pin 50mA [8]. This is important to consider when for example driving many servo motors. If the required current becomes high relative to this, using an external power supply for the servos should be considered. An Arduino can be seen in figure 2.1.
2.2. SERVO-MOTORS

A servo-motor is a motor that allows precise control of motion through electrical impulses. The servo uses feedback to control a DC-motor using PWM [9]. The feedback adjusts the output by measuring the difference between the desired and final position to achieve high accuracy [10]. In more technical detail, the motor gets powered until the output shaft is at its requested position. It then stops, if the current position is not correct, then the motor continues to move in the right direction.

One of the benefits granted is also that it is very energy-efficient for its small size [9]. Therefore, it is very useful for a small dimension arm.

A standard servo-motor for small applications consists of the following elements:

- DC-motor
- Gearbox
- Potentiometer
- Control circuit

Due to having a control circuit included within the servo it becomes easier to control compared to a DC-motor alone. The control circuit sends the PWM signal and controlling the servo using an Arduino becomes as trivial as sending the angle data.
2.3 Newton Raphson’s Method

The Newton Raphson method is based on Taylor expansion and is mostly known for root finding. It is a numerical method for approximating the zeros to a non-linear function given as \( f(x) \). To find the zeros you select a start value, \( x_0 \), preferably a value close to the expected root. You then calculate the derivative for \( f(x_0) \), which becomes \( f'(x_0) \) to get the tangent line. You will now be able to find the next value \( x_1 \) which typically is closer to the exact solution using equation 2.1 below, here \( n \) is the number of the iteration and \( f \) is a non-linear function [11].

\[
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}
\]  

(2.1)

The method is iterative, which means that it can be repeated infinite times to come infinitely close to the exact solution [11]. An illustrative graph can be seen in figure 2.2.

![Figure 2.2. Illustrative graph of Newton Raphson’s method [12].](image-url)
2.4 Fuzzy logic

Fuzzy logic is a form of logic for computers that expand on the simple binary "True or False". It allows for values in between True (1) and False (0) thus can be used to represent and manipulate uncertain information. It gives the computer more human-like decision making abilities [13]. The computer can then consider all available data and take the best possible decision based on the specific given input. This allows the computer to "guess" what is the most probable output for a certain input. This is done in a series of steps as seen in figure 2.3. The first step is taking a crisp input, for example desired position or temperature, and converting it into a fuzzy input using a fuzzifier [14]. If there are three crisp input temperatures that are defined as "cold", "warm" and "hot". It is possible by fuzzifying them "blur the lines" between them and define what is colder than warm but warmer than cold. This can be seen as the red arrow in figure 2.4. Rules and intelligence based on the specific system that is controlled is then applied on the fuzzified input. Lastly the defuzzifier converts the fuzzy value back to crisp output [15].

![Fuzzy Logic Systems Architecture](image)

**Figure 2.3.** Fuzzy Logic Systems Architecture [13].

![Fuzzfing input value](image)

**Figure 2.4.** Fuzzfing input value [15].
2.5 Degrees of freedom

The degrees of freedom of a body are determined by the number of independent variables needed to determine the body’s position [16]. For example, a free-body in 3D-Space has six degrees of freedom consisting of three rotations and three translations as seen in figure 2.5.

![Figure 2.5. Free-body in 3D and 2D Space [16].](image)

To calculate the degrees of freedom of a mechanism or construction, the Grüber-Kutzbach’s criterion in equation 2.2, is applied.

\[
F = 3 \cdot (N - 1) - \sum_{f=1}^{2} (3 - f) \cdot m_f \tag{2.2}
\]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F)</td>
<td>Degrees of freedom</td>
</tr>
<tr>
<td>(N)</td>
<td>Number of links (including stand/support)</td>
</tr>
<tr>
<td>(f)</td>
<td>Degrees of freedom in joints</td>
</tr>
<tr>
<td>(m_f)</td>
<td>Number of joints with (f) degrees of freedom</td>
</tr>
</tbody>
</table>
2.6. KINEMATICS

The robot arm only consists of rotational joints which have 1 degree of freedom (rotation around one axis). Using equation 2.2 gives us \( F = 3 \cdot (N - 1) - 2 \cdot m_1 \).

Here, \( N = 6 \), \( m_1 = 5 \). This gives \( F = 5 \), which means that each individual joint needs to be positioned to get the desired position. It also means that the robot can’t fully move around every point in 3D-space because as previously mentioned, a free-body in 3D-space has six degrees of freedom [16]. A modified schematic picture of the prototype can be seen in figure 2.6.

2.6 Kinematics

Kinematics comes from the greek word "kinēsis" meaning "movement, motion". It is the part of mechanics that describes a body’s motion without regard to the reaction/effect of the motion. Forward kinematics is to calculate the location of the end point from the specified angles of the particular joints. This can be done very easily using a mechanism’s geometric equations and is considered a trivial task. Most of the time however, the opposite is what is desired. That is calculate the specific angles of the joints for a certain end point. For example which angles the servos should rotate to so that the end point of the arm moves to \( [x, y, z] = [1, 2, 3] \). Inverse kinematics can be achieved by using the same geometric equations and constrains as the forward kinematics but calculating backwards [16].
Chapter 3

Prototype

3.1 Electronics

The current section informs about the electrical parts that were used in the project. A microcontroller, three servo-motors, three micro servo-motors and a power supply were used, a circuit diagram can be seen in figure 3.1.

![Circuit diagram for our prototype made in Tinkercad.](image)

**Figure 3.1.** Circuit diagram for our prototype made in Tinkercad.

3.1.1 Microcontroller

To control and send instructions to our servo motors an Arduino Uno is used. This microcontroller works as an intermediary between the computer and the servos. The computer calculated (in MATLAB) the desired angles and sends it to the Arduino where all the servos are attached to the digital pins D4-D10.
3.2. HARDWARE

3.1.2 Servo-motors

The robotic arm is driven by six servo motors in total. Three standard sized servo motors of model MG966R Tower-Pro and three micro servo motors of model SG90. Rotation of the servo motors is achieved by sending data using the digital pins of the Arduino Uno.

3.1.3 Power supply

The $V_{cc}$ pin on the ATmega328P (the microchip on the Arduino Uno) has an absolute maximum ratings of 200 mA [17]. Apart from that there are also limitations by the voltage regulator. The stall current of our servos are 2.5A (standard) and 650 mA (micro) respectively [18][19]. Although the actual current draw will be lower than the stall current, with the amount of servos and the limitations provided, it is highly recommended to use an external power supply. This will make sure the Arduino does not get damaged from over-current and ensure that the servo’s torque is not bottle-necked by low current. Our solution was to use a 6V DC adapter and solder wires on the end. Those wires are then attached to the breadboard and supply power to the servos. It is also important to make sure the ground of the Arduino is also connected to the breadboard so that the power supply and Arduino share a common ground.

3.2 Hardware

The following section contains the hardware parts used in constructing the robotic arm. All servo motors are attached with a servo horn transferring the torque from the servo to the next link. A picture of the final product can be seen in figure 3.2 below.
3.2.1 Base unit

The robotic base unit contains two parts displayed as number 1 and 2 in figure 3.2, a base cylinder colored black and a round plate colored blue. The base cylinder has a servo-motor attached inside connected to the round plate to rotate the arm 180 degrees. The round plate is also connected to a servo-motor to move the arm-part 180 degrees around the second DOF.

3.2.2 Arm unit

The arm unit consists of four parts and is displayed as number 3, 4, 5 and 6 in figure 3.2. The first part of the arm showed as number 3 is connected to the base unit. On top of number 3 is another servo to link up between part 3 and 4. Number 5 and 6 are connected to and driven by a micro servo that gives the robot an elbow and a wrist when it comes to rotation. Worth noticing is also that every joint connected to a standard servo can only move 180 degrees where as every joint connected to a micro-servo can move 270 degrees.
3.3. SOFTWARE

3.2.3 Gripper

The gripper is the final part of the robot known as number 7 in figure 3.2. It is driven by a micro-servo that is attached on one of the claws. Rotating the micro servo transfers torque to the other arm of the claw via inter-meshing cogs. This leads to opening and closing the gripper.

3.3 Software

As previously mentioned we used MATLAB to send data to the Arduino and thereby controlling the robot arm. We had three main approaches for this. The general method for developing all of them consisted of 7 steps as shown in figure 1.1. Start by drawing the schematic figures for the robot arm and defining variables and known parameters. We chose to simplify the problem by only including the two links which are predominantly responsible for the end position, that is, link 3 and link 4 in figure 3.2. We also simplified to two dimensions because rotation of the base translates the 2D system to 3D by rotating the xy-plane.

\[
\begin{align*}
X &= L_1 \cdot \cos(\alpha) + L_2 \cdot \cos(\alpha + \beta) \\
Y &= L_1 \cdot \sin(\alpha) + L_2 \cdot \sin(\alpha + \beta)
\end{align*}
\]

Figure 3.3. Schematic figure for robot arm, drawn in Pages.

\(\alpha\) Angle of servo 1, rotating link 3
\(\beta\) Angle of servo 2, rotating link 4
\(L_1\) Length of link 3
\(L_2\) Length of link 4

From this we can define the geometric equations for the simplified schematic. These are the equations that define the end position of our robot arm.
3.3.1 Manual control

The first approach, manual control, does not require the previously defined equations. It is achieved by first, linking the Arduino and servos in MATLAB. Secondly, creating a custom interface in MATLAB app-designer.

![Manual interface made in MATLAB app-designer.](image)

The bottom sliders in figure 3.4 allow for manual and individual control of each servo’s rotation.

<table>
<thead>
<tr>
<th>Bas</th>
<th>Arm</th>
<th>Axel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rot</th>
<th>Hand</th>
<th>Klo</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Bas**: Servo rotating link 2
- **Arm**: Servo rotating link 3
- **Axel**: Servo rotating link 4
- **Rot**: Servo rotating link 5
- **Hand**: Servo rotating link 6
- **Klo**: Servo rotating link 7 (open/close gripper)

The top boxes allows saving of specific angles in a vector that can later be “played back” using the next button. This allows the manual control to precisely measure and save certain positions and repeat them on command. Thereby are the angles
3.3. SOFTWARE

calculated manually with trial and error. This is achieved by moving the arm and seeing where it ends up then saving when at the desired position.

3.3.2 Numerical approach

The second approach is a numerical approach. Here, the equations used are equation 3.1 and equation 3.2 to solve for $\alpha$ and $\beta$. This was done via the Newton Raphson’s method. It is then possible, by creating a Newton Raphson function in MATLAB, to solve for which angles are required for any specific end position. It defines:

\[
\vec{F} = \begin{bmatrix}
L_1 \cdot \cos(\alpha) + L_2 \cdot \cos(\alpha + \beta) - X \\
L_1 \cdot \sin(\alpha) + L_2 \cdot \sin(\alpha + \beta) - Y
\end{bmatrix} \tag{3.3}
\]

\[
\vec{J}_1 = \frac{\delta F}{\delta \alpha} \tag{3.4}
\]

\[
\vec{J}_2 = \frac{\delta F}{\delta \beta} \tag{3.5}
\]

This solves for the angles which can be sent to the Arduino.

3.3.3 Neural network approach

Using fuzzy logic a neural network was constructed to resolve the inverse kinematics by using the forward kinematics. Hence this can skip the requirement of constructing analytical equations. This is especially useful for more complex mechanisms, for example three or four links. Using MATLAB’s Fuzzy Logic Toolbox to create a fuzzy interference system, more specifically ANFIS. Following this, the forward kinematics were calculated across our range of inputs. The inputs and outputs is in this case the coordinates and the angles. This data is then used to ”train” the ANFIS. After ”training” the network the command ”evalfis” can be used, which when used with a ”trained” ANFIS allows the network to deduce what possible angles are required for a desired position.
Chapter 4

Results

A prototype of a robotic arm was constructed using six servo motors allowing for
five degrees of freedom. Using this prototype made it possible to evaluate the
methods not only using simulation but also real world performance. A general
simulation was made in Acumen that shows how the robot arm can move. The
source code for this simulation can be found in appendix A.

The methods were evaluated in three aspects (1-10 where higher score is better):
control, reliability and speed. Control is evaluated by how easy the robot arm
is to move to the end position. Reliability is determined by the success rate the
method has of moving the arm to the correct end position. Speed is evaluated by
time from command sent to program until when the arm starts to move.

<table>
<thead>
<tr>
<th>Method</th>
<th>Control</th>
<th>Reliability</th>
<th>Speed</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual control</td>
<td>4/10</td>
<td>10/10</td>
<td>10/10</td>
<td>24/30</td>
</tr>
<tr>
<td>Numerical analysis</td>
<td>10/10</td>
<td>8/10</td>
<td>10/10</td>
<td>26/30</td>
</tr>
<tr>
<td>Neural network</td>
<td>10/10</td>
<td>6/10</td>
<td>10/10</td>
<td>26/30</td>
</tr>
</tbody>
</table>

Table 4.1. Evaluation of methods.

4.1 Manual Control

Controlling the robot using manual control allowed for precise control of all the
angles for each individual servo motor. Due to that each servo provides one DOF,
this also resulted in precise control of each individual degree of freedom. This
method did however only allow for active control of one servo at a time. For
example, moving the arm vertically up had to be done in two steps. First by
moving servo 1 then servo 2, this can be seen in figure 4.1. This was partly
resolved by creating a vector which stores values for all the servos, then moving all
of them with a single button press. This does make the servos all move at the same
time. It does not however constrain the robot arm to only move in for example
the vertical or horizontal axis. This resulted in difficulty moving the arm to the desired position. The source code for this method can be found in appendix B.

4.2 Numerical approach

The numerical approach resulted in a way to solve all the required angles for a specific position. This allowed for calculating how to move the arm straight horizontally or vertically. In the following method this was achieved with a series of steps. For example moving the arm vertically (y-axis) resulted in the following required steps made by the MATLAB program:

- Use the forward kinematics to calculate the current position
- Add a step in the Y-axis \( Y = Y + \Delta Y \)
- Run the Newton-Raphson function to calculate the required angles
- Send angles to the respective servos

The numerical approach broadly resulted in good control of the robotic arm. The main downside was the speed, since calculations had to be run for every move. Source code for the numerical approach can be found in appendix C.
4.3 Neural network

Using a slightly modified version of the ANFIS network, written by Mathworks [20], a neural network was developed for the robotic arm. The main modifications that were done was to translate the coordinate system to our desired one. The differences can be seen in figure 4.2. The main changes being that the range of $\beta$ is rotated 90° clockwise and the range of $\alpha$ is 180° instead of 90°.

The ANFIS network proved to be very quick but unreliable. In some cases the network could guess the correct angles while in others it diverted from the correct position by between 0-20%. The success rate was through testing found to be around 60% where the network would "guess" correct. It was derived that the cases where the network guessed correct were usually within a specific range of angles. This range was $\alpha, \beta$ within $0 - \pi/2$. The source code for the ANFIS network can be found in appendix D.
Chapter 5

Discussion

The manual method while allowing for simple individual control proved to be very limiting when trying to move the arm to a specific end position. This is because it was hard to manually determine each angle to move to a certain position. Even with trial and error, a simple task such as moving the robot arm down showed to be very difficult. The speed and reliability of this method was however very great. The arm always moved straight away without latency because no calculations had to be done.

The numerical method’s ability to calculate the way for the arm to move vertically or horizontally proved very useful. This greatly increased the ability to control the arm and move it to the desired end position. There was however a noticeable latency from sending the desired end position until the arm starts to move. While this method proved to be very reliable within the range specified by its geometric equations, there were some issues. The biggest one being that sending a non-reachable position results in the Newton-Raphson function not converging. This leads to very sporadic commands being sent to the robot arm. The issue was combated by adding a max iteration count and not sending the commands to the arm if this max count was reached.

The neural network approach could control the arm in the same way as the numerical method. After the network had been "taught" it was much faster than the Newton-Raphson’s method. After sending a desired end position there was almost no latency until the arm started moving. Within the range that we got the neural network working in, it was reliable within a factor of 0.1%. The other advantage this method had over Newton-Raphson’s method was that sending an unreachable position to the neural network would still produce a sensible result. In other words the network will try to "guess" how to move the arm as close as possible to that point. This method also received the most total points which further displays its strengths.
Chapter 6

Conclusion & Improvements

The conclusion of this Bachelor’s thesis project is that we can control the robot arm with three methods. A manual, a numerical based and a neural network based method. The neural network method proved to be the most effective and best method.

For future work, you could design almost any mount instead of the gripper, perhaps a pen or another tool. Further development includes improving the manual interface to try and expand on the abilities to control the robotic arm without calculations. The numerical method could be changed to use a more effective and faster method instead of Newton-Raphson’s method. Furthermore "teaching" the neural network in a better way so that it becomes reliable in all available ranges of motion would be favourable. Lastly, broadening the scope so that the computational methods include additional degrees of freedom.


125. [Online]. Available from: https://doi.org/10.1051/matecconf/201712502025


Appendix A

Acumen Code

//KTH Royal Institute of Technology.
//Bachelor's Thesis in Mechatronics.
//Multipurpose robot arm.
//Multifunktions robotarm.

//Authors: Fahim Pirmohamed (fahimp@kth.se),
//Alexander Aronsson (alearo@kth.se).

//Course code: MF133X.
//Examiner: Nihad Subasic.
//TRITA: 2021:34.

//File for ACUMEN simulation of robot arm.

// Displaying a box

model Main(simulator) =
initially
x = 0,
x' = 0.2,

_3D = () // Orientation
always
x' = 0.2,

_3D = (Cylinder // Type of _3D object
center=(0,0,0.15) // Center point
radius = 0.4 // radius
length = 0.3 // length
color=red // Color
rotation=(pi/2,0,0)

Cylinder // Type of _3D object
center=(0,0,0.5) // Center point
radius = 0.1 // radius
length = 1 // length
color=red // Color
rotation=(pi/2,0,0)

Sphere // Type of _3D object
center=(0,0,1) // Starting point in [x,y,z] form
size=0.2 // Radius
color=cyan // Color in red-green-blue (RGB) intensity
rotation=(0,0,0)

Cylinder // Type of _3D object
center=(-0.5*cos(x),0.5-0.5*(1+sin(x)),1) // Center point
radius = 0.1 // radius
length = 1 // length
color=red // Color
rotation=(0,0,pi/2+x)

Sphere // Type of _3D object
center=(-cos(x),1-(1+sin(x)),1) // Starting point in [x,y,z] form
size=0.2 // Radius
color=cyan // Color in red-green-blue (RGB) intensity
rotation=(0,0,0)

Cylinder // Type of _3D object
center=(-cos(x),1-(1+sin(x)),0.75) // Center point
radius = 0.08 // radius
length = 0.5 // length
color=red // Color
rotation=(pi/2,0,pi/2+x) // Orientation
Appendix B

MATLAB Code Manual Control

Listing B.1. Source Code

```xml
<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
  <w:body>
    <w:p>
      <w:pPr>
        <w:pStyle w:val="code"/>
      </w:pPr>
      <w:r>
        <w:t><![CDATA[classdef KEXrobotManualVEK < matlab.apps.AppBase

          %KTH Royal Institute of Technology.
          %Bachelor's Thesis in Mechatronics.
          %Multipurpose robot arm.
          %Multifunktions robotarm.
          %Authors: Fahim Pirmohamed (fahimp@kth.se),
          %Alexander Aronsson (alearo@kth.se).
          %Course code: MF133X.
          %Examiner: Nihad Subasic.
          %TRITA: 2021:34.
          %File for MATLAB Manual Control.

          % Properties that correspond to app components
          properties (Access = public)
            UIFigure    matlab.ui.Figure
            ClearButton matlab.ui.control.Button
            CurrentEditField matlab.ui.control.NumericEditField
            CurrentEditFieldLabel matlab.ui.control.Label
            SavedEditField matlab.ui.control.NumericEditField
            SavedEditFieldLabel matlab.ui.control.Label
            GripToggleButton matlab.ui.control.Button
```

25
APPENDIX B. MATLAB CODE MANUAL CONTROL

resetbutton matlab.ui.control.Button
kloEditField matlab.ui.control.NumericEditField
kloEditFieldLabel matlab.ui.control.Label
handEditField matlab.ui.control.NumericEditField
handEditFieldLabel matlab.ui.control.Label
rotEditField matlab.ui.control.NumericEditField
rotEditFieldLabel matlab.ui.control.Label
kloSlider matlab.ui.control.Slider
kloSliderLabel matlab.ui.control.Label
handSlider matlab.ui.control.Slider
handSliderLabel matlab.ui.control.Label
rotSlider matlab.ui.control.Slider
rotSliderLabel matlab.ui.control.Label
nextButton matlab.ui.control.Button
saveButton matlab.ui.control.Button
moveButton matlab.ui.control.Button
axelEditField matlab.ui.control.NumericEditField
axelEditFieldLabel matlab.ui.control.Label
armEditField matlab.ui.control.NumericEditField
armEditFieldLabel matlab.ui.control.Label
axelSlider matlab.ui.control.Slider
axelSliderLabel matlab.ui.control.Label
armSlider matlab.ui.control.Slider
armSliderLabel matlab.ui.control.Label
basSlider matlab.ui.control.Slider
basSliderLabel matlab.ui.control.Label
basEditField matlab.ui.control.NumericEditField
basEditFieldLabel matlab.ui.control.Label

end

properties (Access = private)
    vBas = 90 % Description
    vArm = 90
    vAxel = 180
    vRot = 0
    vHand = 0
    vKlo = 200

    a
    s1
    s2
    s3
    s4
    s5
    s6
    savedBas = []
    savedArm = []
    savedAxel = []
    savedRot = []
    savedHand = []
    savedKlo = []

    g = 0
methods (Access = private)

function results = updateSliders(app)
    app.BasSlider.Value = app.vBas;
    app.ArmSlider.Value = app.vArm;
    app.AxelSlider.Value = app.vAxel;
    app.RotSlider.Value = app.vRot;
    app.HandSlider.Value = app.vHand;
    app.kloSlider.Value = app.vKlo;
end

function results = updateBoxes(app)
    app.BasEditField.Value = app.vBas;
    app.ArmEditField.Value = app.vArm;
    app.AxelEditField.Value = app.vAxel;
    app.RotEditField.Value = app.vRot;
    app.HandEditField.Value = app.vHand;
    app.KloEditField.Value = app.vKlo;
end

% Callbacks that handle component events
methods (Access = private)

% Code that executes after component creation
function startupFcn(app)
    app.a = arduino();
    app.s1 = servo(app.a,'D4');
    app.s2 = servo(app.a,'D5');
    app.s3 = servo(app.a,'D6');
    app.s4 = servo(app.a,'D7');
    app.s5 = servo(app.a,'D8');
    app.s6 = servo(app.a,'D9');
    writePosition(app.s1, app.vBas/180);
    writePosition(app.s2, app.vArm/180);
    writePosition(app.s3, app.vAxel/180);
    writePosition(app.s4, 0);
    writePosition(app.s5, 0);
    writePosition(app.s6, app.vKlo/270);
    app.updateSliders();
    app.updateBoxes();
end

% Value changed function: BasEditField
function BasEditFieldValueChanged(app, event)
    app.vBas = app.BasEditField.Value;
end
APPENDIX B. MATLAB CODE MANUAL CONTROL

%% Value changed function: ArmEditField
function ArmEditFieldValueChanged(app, event)
    app.vArm = app.ArmEditField.Value;
end

%% Value changed function: AxelEditField
function AxelEditFieldValueChanged(app, event)
    app.vAxel = app.AxelEditField.Value;
end

%% Button pushed function: MoveButton
function MoveButtonPushed(app, event)
    app.updateSliders();
    writePosition(app.s1, app.vBas/180);
    writePosition(app.s2, app.vArm/180);
    writePosition(app.s3, app.vAxel/180);
    writePosition(app.s4, app.vRot/270);
    writePosition(app.s5, app.vHand/270);
    writePosition(app.s6, app.vKlo/270);
end

%% Value changing function: BasSlider
function BasSliderValueChanging(app, event)
    changingValue = event.Value;
    app.vBas = changingValue;
    writePosition(app.s1, app.vBas/180);
    app.updateBoxes();
end

%% Value changing function: ArmSlider
function ArmSliderValueChanging(app, event)
    changingValue = event.Value;
    app.vArm = changingValue;
    writePosition(app.s2, app.vArm/180);
    app.updateBoxes();
end

%% Value changing function: AxelSlider
function AxelSliderValueChanging(app, event)
    changingValue = event.Value;
    app.vAxel = changingValue;
    writePosition(app.s3, app.vAxel/180);
    app.updateBoxes();
end

%% Button pushed function: SaveButton
function SaveButtonPushed(app, event)
    app.savedBas = [app.savedBas app.vBas];
    app.savedArm = [app.savedArm app.vArm];
    app.savedAxel = [app.savedAxel app.vAxel];
    app.savedRot = [app.savedRot app.vRot];
app.savedHand = [app.savedHand app.vHand];
app.savedKlo = [app.savedKlo app.vKlo];
app.SavedEditField.Value = app.SavedEditField.Value + 1;
end

% Button pushed function: NextButton
function NextButtonPushed(app, event)
idx = app.CurrentEditField.Value;
if idx <= length(app.savedBas)
    writePosition(app.s1, app.savedBas(idx)/180);
    writePosition(app.s2, app.savedArm(idx)/180);
    writePosition(app.s3, app.savedAxel(idx)/180);
    writePosition(app.s4, app.savedRot(idx)/270);
    writePosition(app.s5, app.savedHand(idx)/270);
    writePosition(app.s6, app.savedKlo(idx)/270);
    app.updateBoxes();
    app.updateSliders();
end

% Value changing function: RotSlider
function RotSliderValueChanging(app, event)
    changingValue = event.Value;
    app.vRot = changingValue;
    writePosition(app.s4, app.vRot/270);
    app.updateBoxes();
end

% Value changing function: HandSlider
function HandSliderValueChanging(app, event)
    changingValue = event.Value;
    app.vHand = changingValue;
    writePosition(app.s5, app.vHand/270);
    app.updateBoxes();
end

% Value changing function: kloSlider
function kloSliderValueChanging(app, event)
    changingValue = event.Value;
    app.vKlo = changingValue;
    writePosition(app.s6, app.vKlo/270);
    app.updateBoxes();
end

% Value changed function: HandEditField
function HandEditFieldValueChanged(app, event)
    app.vHand = app.HandEditField.Value;
end

% Value changed function: RotEditField
function RotEditFieldValueChanged(app, event)
    app.vRot = app.RotEditField.Value;
end

% Value changed function: KloEditField
function KloEditFieldValueChanged(app, event)
    app.vKlo = app.KloEditField.Value;
end

% Button pushed function: GripToggleButton
function GripToggleButtonPushed(app, event)
    if app.g == 0
        app.vKlo = 270;
        writePosition(app.s6, app.vKlo/270);
        app.g = 1;
    elseif app.g == 1
        app.vKlo = 200;
        writePosition(app.s6, app.vKlo/270);
        app.g = 0;
    end
    app.updateBoxes();
    app.updateSliders();
end

% Button pushed function: ResetButton
function ResetButtonPushed(app, event)
    app.CurrentEditField.Value = 1;
end

% Button pushed function: ClearButton
function ClearButtonPushed(app, event)
    app.savedBas = [];
    app.savedArm = [];
    app.savedAxel = [];
    app.savedRot = [];
    app.savedHand = [];
    app.savedKlo = [];
    app.SavedEditField.Value = 0;
    app.CurrentEditField.Value = 1;
end

end

% Component initialization
methods (Access = private)

% Create UIFigure and components
function createComponents(app)
    % Create UIFigure and hide until all components are created
    app.UIFigure = uifigure('Visible', 'off');
    app.UIFigure.Position = [100 100 640 480];
app.UIFigure.Name = 'MATLAB App';

% Create BasEditFieldLabel
app.BasEditFieldLabel = uilabel(app.UIFigure);
app.BasEditFieldLabel.HorizontalAlignment = 'right';
app.BasEditFieldLabel.Position = [24 437 26 22];
app.BasEditFieldLabel.Text = 'Bas';

% Create BasEditField
app.BasEditField = uieeditfield(app.UIFigure, 'numeric');
app.BasEditField.Limits = [0 180];
app.BasEditField.ValueChangedFcn = createCallbackFcn(app, @BasEditFieldValueChanged, true);
app.BasEditField.Position = [65 437 100 22];

% Create BasSliderLabel
app.BasSliderLabel = uilabel(app.UIFigure);
app.BasSliderLabel.HorizontalAlignment = 'right';
app.BasSliderLabel.Position = [41 190 26 22];
app.BasSliderLabel.Text = 'Bas';

% Create BasSlider
app.BasSlider = uislider(app.UIFigure);
app.BasSlider.Limits = [0 180];
app.BasSlider.ValueChangingFcn = createCallbackFcn(app, @BasSliderValueChanging, true);
app.BasSlider.Position = [88 199 150 3];

% Create ArmSliderLabel
app.ArmSliderLabel = uilabel(app.UIFigure);
app.ArmSliderLabel.HorizontalAlignment = 'right';
app.ArmSliderLabel.Position = [41 126 28 22];
app.ArmSliderLabel.Text = 'Arm';

% Create ArmSlider
app.ArmSlider = uislider(app.UIFigure);
app.ArmSlider.Limits = [0 180];
app.ArmSlider.ValueChangingFcn = createCallbackFcn(app, @ArmSliderValueChanging, true);
app.ArmSlider.Position = [90 135 150 3];
app.ArmSlider.Value = 90;

% Create AxelSliderLabel
app.AxelSliderLabel = uilabel(app.UIFigure);
app.AxelSliderLabel.HorizontalAlignment = 'right';
app.AxelSliderLabel.Position = [41 71 29 22];
app.AxelSliderLabel.Text = 'Axel';

% Create AxelSlider
app.AxelSlider = uislider(app.UIFigure);
app.AxelSlider.Limits = [0 180];
app.AxelSlider.ValueChangingFcn = createCallbackFcn(app, @AxelSliderValueChanging, true);
APPENDIX B. MATLAB CODE MANUAL CONTROL

app.AxelSlider.Position = [91 80 150 3];
app.AxelSlider.Value = 180;

% Create ArmEditFieldLabel
app.ArmEditFieldLabel = uilabel(app.UIFigure);
app.ArmEditFieldLabel.HorizontalAlignment = 'right';
app.ArmEditFieldLabel.Position = [191 437 28 22];
app.ArmEditFieldLabel.Text = 'Arm';

% Create ArmEditField
app.ArmEditField = uieeditfield(app.UIFigure, 'numeric')
app.ArmEditField.Limits = [0 180];
app.ArmEditField.ValueChangedFcn = createCallbackFcn(
    app, @ArmEditFieldValueChanged, true);
app.ArmEditField.Position = [234 437 100 22];

% Create AxelEditFieldLabel
app.AxelEditFieldLabel = uilabel(app.UIFigure);
app.AxelEditFieldLabel.HorizontalAlignment = 'right';
app.AxelEditFieldLabel.Position = [379 437 29 22];
app.AxelEditFieldLabel.Text = 'Axel';

% Create AxelEditField
app.AxelEditField = uieeditfield(app.UIFigure, 'numeric ')
app.AxelEditField.Limits = [0 180];
app.AxelEditField.ValueChangedFcn = createCallbackFcn(
    app, @AxelEditFieldValueChanged, true);
app.AxelEditField.Position = [423 437 100 22];

% Create MoveButton
app.MoveButton = uibutton(app.UIFigure, 'push ');
app.MoveButton.ButtonPushedFcn = createCallbackFcn(app, 
    @MoveButtonPushed, true);
app.MoveButton.Position = [65 333 100 22];
app.MoveButton.Text = 'Move ';

% Create SaveButton
app.SaveButton = uibutton(app.UIFigure, 'push ');
app.SaveButton.ButtonPushedFcn = createCallbackFcn(app, 
    @SaveButtonPushed, true);
app.SaveButton.Position = [234 333 100 22];
app.SaveButton.Text = 'Save ';

% Create NextButton
app.NextButton = uibutton(app.UIFigure, 'push ');
app.NextButton.ButtonPushedFcn = createCallbackFcn(app, 
    @NextButtonPushed, true);
app.NextButton.Position = [383 333 100 22];
app.NextButton.Text = 'Next ';

% Create RotSliderLabel
app.RotSliderLabel = uilabel(app.UIFigure);
app.RotSliderLabel.HorizontalAlignment = 'right';
app.RotSliderLabel.Position = [305 180 25 22];
app.RotSliderLabel.Text = 'Rot';

% Create RotSlider
app.RotSlider = uislider(app.UIFigure);
app.RotSlider.Limits = [0 270];
app.RotSlider.ValueChangingFcn = createCallbackFcn(app, @RotSliderValueChanging, true);
app.RotSlider.Position = [351 189 150 3];

% Create HandSliderLabel
app.HandSliderLabel = uilabel(app.UIFigure);
app.HandSliderLabel.HorizontalAlignment = 'right';
app.HandSliderLabel.Position = [296 126 34 22];
app.HandSliderLabel.Text = 'Hand';

% Create HandSlider
app.HandSlider = uislider(app.UIFigure);
app.HandSlider.Limits = [0 270];
app.HandSlider.ValueChangingFcn = createCallbackFcn(app, @HandSliderValueChanging, true);
app.HandSlider.Position = [351 135 150 3];

% Create kloSliderLabel
app.kloSliderLabel = uilabel(app.UIFigure);
app.kloSliderLabel.HorizontalAlignment = 'right';
app.kloSliderLabel.Position = [305 71 25 22];
app.kloSliderLabel.Text = 'klo';

% Create kloSlider
app.kloSlider = uislider(app.UIFigure);
app.kloSlider.Limits = [200 270];
app.kloSlider.ValueChangingFcn = createCallbackFcn(app, @kloSliderValueChanging, true);
app.kloSlider.Position = [351 80 150 3];
app.kloSlider.Value = 220;

% Create RotEditFieldLabel
app.RotEditFieldLabel = uilabel(app.UIFigure);
app.RotEditFieldLabel.HorizontalAlignment = 'right';
app.RotEditFieldLabel.Position = [25 386 25 22];
app.RotEditFieldLabel.Text = 'Rot';

% Create RotEditField
app.RotEditField = uieditfield(app.UIFigure, 'numeric');
app.RotEditField.ValueChangedFcn = createCallbackFcn(app, @RotEditFieldValueChanged, true);
app.RotEditField.Position = [65 386 100 22];

% Create HandEditFieldLabel
app.HandEditFieldLabel = uilabel(app.UIFigure);
app.HandEditFieldLabel.HorizontalAlignment = 'right';
% Create HandEditField
app.HandEditField = uieditfield(app.UIFigure, 'numeric');
app.HandEditField.ValueChangedFcn = createCallbackFcn(
    app, @HandEditFieldValueChanged, true);
app.HandEditField.Label = uilabel(app.UIFigure);
app.HandEditField.Label.Position = [383 386 25 22];
app.HandEditField.Label.Text = 'Hand';

% Create KloEditField
app.KloEditField = uieditfield(app.UIFigure);
app.KloEditField.ValueChangedFcn = createCallbackFcn(
    app, @KloEditFieldValueChanged, true);
app.KloEditField.Label = uilabel(app.UIFigure);
app.KloEditField.Label.Position = [383 386 25 22];
app.KloEditField.Label.Text = 'Klo';

% Create ResetButton
app.ResetButton = uibutton(app.UIFigure, 'push');
app.ResetButton.ButtonPushedFcn = createCallbackFcn(app, @ResetButtonPushed, true);
app.ResetButton.Label = uilabel(app.UIFigure);
app.ResetButton.Label.Position = [500 333 100 22];
app.ResetButton.Label.Text = 'Reset';

% Create GripToggleButton
app.GripToggleButton = uibutton(app.UIFigure, 'push');
app.GripToggleButton.ButtonPushedFcn = createCallbackFcn(app, @GripToggleButtonPushed, true);
app.GripToggleButton.Label = uilabel(app.UIFigure);
app.GripToggleButton.Label.Position = [66 267 100 22];
app.GripToggleButton.Label.Text = 'GripToggle';

% Create SavedEditFieldLabel
app.SavedEditFieldLabel = uilabel(app.UIFigure);
app.SavedEditFieldLabel.Position = [392 288 39 22];
app.SavedEditFieldLabel.Text = 'Saved';

% Create SavedEditField
app.SavedEditField = uieditfield(app.UIFigure, 'numeric');
app.SavedEditField.Editable = 'off';
app.SavedEditField.Label = uilabel(app.UIFigure);
app.SavedEditField.Label.Position = [451 288 23 22];

% Create CurrentEditFieldLabel
app.CurrentEditFieldLabel = uilabel(app.UIFigure);
app.CurrentEditFieldLabel.Position = [185 386 34 22];
app.CurrentEditFieldLabel.Position = [502 288 46 22];
app.CurrentEditFieldLabel.Text = 'Current';

% Create CurrentEditField
app.CurrentEditField = uieditfield(app.UIFigure, 'numeric');
app.CurrentEditField.Edible = 'off';
app.CurrentEditField.Position = [568 288 23 22];
app.CurrentEditField.Value = 1;

% Create ClearButton
app.ClearButton = uibutton(app.UIFigure, 'push');
app.ClearButton.ButtonPushedFcn = createCallbackFcn(app, @ClearButtonPushed, true);
app.ClearButton.Position = [234 267 100 22];
app.ClearButton.Text = 'Clear';

% Show the figure after all components are created
app.UIFigure.Visible = 'on';
end
end

% App creation and deletion
methods (Access = public)

% Construct app
function app = KEXrobotManualVEK

% Create UIFigure and components
createComponents(app)

% Register the app with App Designer
registerApp(app, app.UIFigure)

% Execute the startup function
runStartupFcn(app, @startupFcn)

if nargout == 0
    clear app
end
end

% Code that executes before app deletion
function delete(app)

% Delete UIFigure when app is deleted
delete(app.UIFigure)
end
end]]>
</w:t>
</w:r>
</w:p>
</w:body>
Appendix C

MATLAB Code Newton-Raphson

Listing C.1. Source Code

```matlab
%KTH Royal Institute of Technology.
%Bachelor’s Thesis in Mechatronics.
%Multifunktions robot arm.
%Authors: Fahim Pirmohamed (fahimp@kth.se),
%Alexander Aronsson (alearo@kth.se).
%Course code: MF133X.
%Examiner: Nihad Subasic.
%TRITA: 2021:34.
%
%File for MATLAB Numerical Method control.
%
%Properties that correspond to app components
%properties (Access = public)
    UIFigure    matlab.ui.Figure
    StepsizeEditField  matlab.ui.control.
    NumericEditField  matlab.ui.control.
    StepsizeEditFieldLabel  matlab.ui.control.Label
    YButton,2  matlab.ui.control.Button
    YButton  matlab.ui.control.Button
    XButton,2  matlab.ui.control.Button
    XButton  matlab.ui.control.Button
```

37
APPENDIX C. MATLAB CODE NEWTON-RAPHSON

35 BetaEditField matlab.ui.control.
36 NumericEditField matlab.ui.control.
37 BetaEditFieldLabel matlab.ui.control.Label
38 AlphaEditField matlab.ui.control.
39 NumericEditField matlab.ui.control.
40 AlphaEditFieldLabel matlab.ui.control.Label
41 MoveButton matlab.ui.control.Button
42 AnglesTextArea matlab.ui.control.TextArea
43 AnglesTextAreaLabel matlab.ui.control.Label
44 CalculateButton matlab.ui.control.Button
45 PositionTextArea matlab.ui.control.TextArea
46 PositionTextAreaLabel matlab.ui.control.Label
47 BetastartEditField matlab.ui.control.
48 NumericEditField matlab.ui.control.
49 BetastartEditFieldLabel matlab.ui.control.Label
50 AlphastartEditField matlab.ui.control.
51 NumericEditField matlab.ui.control.
52 AlphastartEditFieldLabel matlab.ui.control.Label
53 DesiredYvalueEditField matlab.ui.control.
54 NumericEditField matlab.ui.control.
55 DesiredYvalueEditFieldLabel matlab.ui.control.Label
56 DesiredXvalueEditField matlab.ui.control.
57 NumericEditField matlab.ui.control.
58 DesiredXvalueEditFieldLabel matlab.ui.control.Label
59 UIAxes matlab.ui.control.UIAxes
60 end
61
62 properties (Access = private)
63 desiredX = 0 % Description
64 desiredY = 0 % Description
65 alphaStart = 0 % Description
66 betaStart = 0 % Description
67 Arm1 = 0.3 % Description
68 Arm2 = 0.15 % Description
69 startX = [50*pi/180; 30*pi/180] % Startgissning
70 tol = 1e-12 % Description
71 imax = 10000 % Description
72 resn = 'False'
73 x = [50*pi/180; 30*pi/180]
74 alphaCurrent = 0
75 betaCurrent = 0
76 alphaMove = 0
77 betaMove = 0
78 stepsize = 0.01 % Description
79 end
80
81 methods (Access = private)
82 function x = newtonrap(app,x,x2,y2,L1,L2)
83 i = 0;
84 a = x(1);
85 b = x(2);
f = [L1*cos(a) + L2*cos(b) - x2; L1*sin(a) - L2*sin(b) - y2];

while norm(f) > app.tol & & i < app.imax
    i = i + 1;
    J11 = -L1*sin(a);
    J12 = -L2*sin(b);
    J21 = L1*cos(a);
    J22 = -L2*cos(b);
    J = [J11 J12; J21 J22];
    x = x - J\f;
    a = x(1);
    b = x(2);
    f = [L1*cos(a) + L2*cos(b) - x2; L1*sin(a) - L2*sin(b) - y2];
end
if i == 10000
    app.resn = 'Non reasonable solution'
else
    app.resn = 'Reasonable solution'
end
end
end
end

% Callbacks that handle component events
methods (Access = private)

% Code that executes after component creation
function startupFcn(app)
    axis(app.UIAxes,'square')
end

% Value changed function: DesiredXvalueEditField
function DesiredXvalueEditFieldValueChanged(app, event)
    app.desiredX = app.DesiredXvalueEditField.Value;
end

% Button pushed function: CalculateButton
function CalculateButtonPushed(app, event)
    str = cat(2,'Desired X,Y value is: ',num2str(app.desiredX),', ',num2str(app.desiredY));
    app.PositionTextArea.Value = str;
    app.x = newtonrap(app.app.startX,app.desiredX,app.desiredY,app.Arm1,app.Arm2);
    app.x = app.x*180/pi;
    str2 = cat(2,'Alpha,Beta is: ',num2str(app.x(1)),', ',num2str(app.x(2)),', ',app.resn);
end
APPENDIX C. MATLAB CODE NEWTON-RAPHSON

132 app.AnglesTextArea.Value = str2;
133 if app.alphaCurrent > app.x(1)
134     alpha = (app.alphaCurrent: -0.5:app.x(1))*(pi/180);
135 else
136     alpha = (app.alphaCurrent:0.5:app.x(1))*(pi/180);
137 end
138 l = size(alpha);
139 interval = (1/(l(2)-1))*(app.betaCurrent-app.x(2));
140 beta = (app.betaCurrent:-interval:app.x(2))*(pi/180);  
141 x0 = zeros(size(alpha));
142 y0 = zeros(size(alpha));
143 x1 = app.Arm1*cos(alpha);
144 y1 = app.Arm1*sin(alpha);
145 x2 = x1 + app.Arm2*cos(beta);
146 y2 = y1 - app.Arm2*sin(beta);
147 for k = 1:length(alpha)
148     plot(app.UIAxes,[x0(k) x1(k)],[y0(k) y1(k)],'ro','LineWidth',2)
149     hold(app.UIAxes,'on')
150     plot(app.UIAxes,[x2(1) x2(end)],[y2(1) y2(end)],'b--','MarkerSize',10)
151     hold(app.UIAxes,'off')
152     %app.UIAxes.Xlim = [-0.5 0.5];
153     %app.UIAxes.Ylim = [-0.5 0.5];
154     hold(app.UIAxes);
155     %plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro','LineWidth',2)
156     pause(0.018)
157 end
158 app.alphaCurrent = app.x(1);
159 app.betaCurrent = app.x(2);
160 app.AlphastartEditField.Value = app.x(1);
161 app.BetastartEditField.Value = app.x(2);
162 app.AlphaEditField.Value = app.x(1);
163 app.BetaEditField.Value = app.x(2);
164 % Value changed function: DesiredYvalueEditField
165 function DesiredYvalueEditFieldValueChanged(app, event)
166     app.desiredY = app.DesiredYvalueEditField.Value;
167 end
168 % Value changed function: AlphastartEditField
169 function AlphastartEditFieldValueChanged(app, event)
170     app.alphaStart = app.AlphastartEditField.Value;
171     app.alphaCurrent = app.AlphastartEditField.Value;
173 end
174 % Value changed function: BetastartEditField
175 function BetastartEditFieldValueChanged(app, event)
176     app.betaStart = app.BetastartEditField.Value;
177     app.betaCurrent = app.BetastartEditField.Value;
179 end
180 end
181 %V a l u e c h a n g e d f u n c t i o n : D e s i r e d Y v a l u e E d i t F i e l d 
182 function DesiredYvalueEditFieldValueChanged(app, event)
183     app.desiredY = app.DesiredYvalueEditField.Value;
184 end
185 % Value changed function: AlphastartEditField
186 function AlphastartEditFieldValueChanged(app, event)
187     app.alphaStart = app.AlphastartEditField.Value;
188     app.alphaCurrent = app.AlphastartEditField.Value;
190 end
191 % Value changed function: BetastartEditField
192 function BetastartEditFieldValueChanged(app, event)
193     app.betaStart = app.BetastartEditField.Value;
194     app.betaCurrent = app.BetastartEditField.Value;
196 end
197 40
% Button pushed function: MoveButton
function MoveButtonPushed(app, event)
    app.x = [app.alphaMove; app.betaMove];
    str2 = cat(2, 'Alpha, Beta is: ', num2str(app.x(1)), ', ', num2str(app.x(2)), ', ', app.resn);
    app.AnglesTextArea.Value = str2;
    if app.alphaCurrent > app.x(1)
        alpha = (app.alphaCurrent - 0.5:app.x(1))*(pi/180);
    else
        alpha = (app.alphaCurrent:0.5:app.x(1))*(pi/180);
    end
    l = size(alpha);
    if (app.betaCurrent - app.x(2)) == 0
        app.x(2) = app.x(2) - 0.01;
    end
    interval = (1/(l(2)-1))*(app.betaCurrent-app.x(2));
    beta = (app.betaCurrent:-interval:app.x(2))*(pi/180);
    x0 = zeros(size(alpha));
    y0 = zeros(size(alpha));
    x1 = app.Arm1*cos(alpha);
    y1 = app.Arm1*sin(alpha);
    x2 = x1 + app.Arm2*cos(beta);
    y2 = y1 - app.Arm2*sin(beta);
    for k = 1:length(alpha)
        plot(app.UIAxes,[x0(k) x1(k)],[y0(k) y1(k)],'r-o','LineWidth',2)
        hold(app.UIAxes,'on')
        plot(app.UIAxes,[x2(1) x2(end)],[y2(1) y2(end)],'b--','MarkerSize',10)
        plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'r-o','LineWidth',2)
        hold(app.UIAxes,'off')
        %app.UIAxes.Xlim = [-0.5 0.5];
        %app.UIAxes.Ylim = [-0.5 0.5];
        %hold(app.UIAxes);
        %plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'r-o','LineWidth',2)
    pause(0.018)
end
function AlphaEditFieldValueChanged(app, event)
    app.alphaMove = app.AlphaEditField.Value;
end
function BetaEditFieldValueChanged(app, event)
    app.betaMove = app.BetaEditField.Value;
end
% Button pushed function: XButton
function XButtonPushed(app, event)
    app.desiredX = app.desiredX + app.stepsize;
end

% Value changed function: AlphaEditField
function AlphaEditFieldValueChanged(app, event)
    app.alphaMove = app.AlphaEditField.Value;
end
% Value changed function: BetaEditField
function BetaEditFieldValueChanged(app, event)
    app.betaMove = app.BetaEditField.Value;
end
% Button pushed function: MoveButton
function MoveButtonPushed(app, event)
    app.x = [app.alphaMove; app.betaMove];
    str2 = cat(2, 'Alpha, Beta is: ', num2str(app.x(1)), ', ', num2str(app.x(2)), ', ', app.resn);
    app.AnglesTextArea.Value = str2;
    if app.alphaCurrent > app.x(1)
        alpha = (app.alphaCurrent - 0.5:app.x(1))*(pi/180);
    else
        alpha = (app.alphaCurrent:0.5:app.x(1))*(pi/180);
    end
    l = size(alpha);
    if (app.betaCurrent - app.x(2)) == 0
        app.x(2) = app.x(2) - 0.01;
    end
    interval = (1/(l(2)-1))*(app.betaCurrent-app.x(2));
    beta = (app.betaCurrent:-interval:app.x(2))*(pi/180);
    x0 = zeros(size(alpha));
    y0 = zeros(size(alpha));
    x1 = app.Arm1*cos(alpha);
    y1 = app.Arm1*sin(alpha);
    x2 = x1 + app.Arm2*cos(beta);
    y2 = y1 - app.Arm2*sin(beta);
    for k = 1:length(alpha)
        plot(app.UIAxes,[x0(k) x1(k)],[y0(k) y1(k)],'r-o','LineWidth',2)
        hold(app.UIAxes,'on')
        plot(app.UIAxes,[x2(1) x2(end)],[y2(1) y2(end)],'b--','MarkerSize',10)
        plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'r-o','LineWidth',2)
        hold(app.UIAxes,'off')
        %app.UIAxes.Xlim = [-0.5 0.5];
        %app.UIAxes.Ylim = [-0.5 0.5];
        %hold(app.UIAxes);
        %plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'r-o','LineWidth',2)
    pause(0.018)
end
end

% Value changed function: AlphaEditField
function AlphaEditFieldValueChanged(app, event)
    app.alphaMove = app.AlphaEditField.Value;
end
% Value changed function: BetaEditField
function BetaEditFieldValueChanged(app, event)
    app.betaMove = app.BetaEditField.Value;
end
% Button pushed function: XButton
function XButtonPushed(app, event)
    app.desiredX = app.desiredX + app.stepsize;
APPENDIX C. MATLAB CODE NEWTON-RAPHSON

```
237 app.DesiredXvalueEditField.Value = app.desiredX;
238 str = cat(2,'Desired X,Y value is: ', num2str(app.desiredX), ',' ,
num2str(app.desiredY));
239 app.PositionTextArea.Value = str;
240 app.x = newtonrap(app, app.startX, app.desiredX, app.desiredY, app.Arm1,
app.Arm2);
241 app.x = app.x*180/pi;
242 str2 = cat(2,'Alpha, Beta is: ', num2str(app.x(1)), ',' ,
num2str(app.x(2)), app.resn);
243 app.AnglesTextArea.Value = str2;
244 if app.alphaCurrent > app.x(1)
245 alpha = (app.alphaCurrent-0.5:app.x(1))*(pi/180);
246 else
247 alpha = (app.alphaCurrent:0.5:app.x(1))*(pi/180);
248 end
249 l = size(alpha);
250 interval = (1/(l(2)-1))*(app.betaCurrent-app.x(2));
251 beta = (app.betaCurrent:-interval:app.x(2))*(pi/180);
252 x0 = zeros(size(alpha));
253 y0 = zeros(size(alpha));
254 x1 = app.Arm1*cos(alpha); 
255 y1 = app.Arm1*sin(alpha);
256 x2 = x1 + app.Arm2*cos(beta);
257 y2 = y1 - app.Arm2*sin(beta);
258 for k = 1:length(alpha)
259 plot(app.UIAxes,[x0(k) x1(k)],[y0(k) y1(k)],'ro-', 'LineWidth',2)
260 hold(app.UIAxes,'on')
261 plot(app.UIAxes,[x2(1) x2(end)],[y2(1) y2(end)],'b--', 'MarkerSize', 10)
262 plot(app.UIAxes,x2(1), y2(1),'bx', 'MarkerSize', 10)
263 plot(app.UIAxes,x2(end), y2(end), 'bo', 'MarkerSize', 10)
264 plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-', 'LineWidth', 2)
265 hold(app.UIAxes,'off')
266 %app.UIAxes.Xlim = [-0.5 0.5];
267 %app.UIAxes.Ylim = [-0.5 0.5];
268 %hold(app.UIAxes);
269 %plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-', 'LineWidth', 2)
270 pause(0.018)
271 end
272 app.alphaCurrent = app.x(1);
273 app.betaCurrent = app.x(2);
274 app.AlphastartEditField.Value = app.x(1);
275 app.BetastartEditField.Value = app.x(2);
276 app.AlphaEditField.Value = app.x(1);
277 app.BetaEditField.Value = app.x(2);
278 end
279 % Button pushed function: XButton_2
280 function XButton_2Pushed(app, event)
281 app.desiredX = app.desiredX - app.stepsize;
282 app.DesiredXvalueEditField.Value = app.desiredX;
283 str = cat(2,'Desired X,Y value is: ', num2str(app.desiredX), ',' ,
num2str(app.desiredY));
284 app.PositionTextArea.Value = str;
```
app.x = newtonrap(app.app.startX, app.desiredX, app.desiredY, app.Arm1, app.Arm2);

app.x = app.x*180/pi;
str2 = cat(2, 'Alpha, Beta is: ', num2str(app.x(1)) ', ', num2str(app.x(2)) ', ', app.resn);
app.AnglesTextArea.Value = str2;

end

l = size(alpha);
interval = (1/(1(2)-1))*(app.betaCurrent-app.x(2));
beta = (app.betaCurrent:-interval:app.x(2))*(pi/180);
x0 = zeros(size(alpha));
y0 = zeros(size(alpha));
x1 = app.Arm1*cos(alpha);
y1 = app.Arm1*sin(alpha);
x2 = x1 + app.Arm2*cos(beta);
y2 = y1 - app.Arm2*sin(beta);

for k = 1:length(alpha)
    plot(app.UIAxes,[x0(k) x1(k)],[y0(k) y1(k)],'ro','LineWidth',2)
    hold(app.UIAxes,'on')
    plot(app.UIAxes,[x2(1) x2(end)],[y2(1) y2(end)],'b--','MarkerSize',10)
    plot(app.UIAxes,x2(1) ,y2(1) ,bx','MarkerSize',10)
    plot(app.UIAxes,x2(end) ,y2(end) ,bo','MarkerSize',10)
    hold(app.UIAxes,'off')
    if app.alphaCurrent > app.x(1)
    alpha = (app.alphaCurrent:-0.5:app.x(1))*(pi/180);
    else
    alpha = (app.alphaCurrent:0.5:app.x(1))*(pi/180);
    end
    app.UIAxes.Xlim = [-0.5 0.5];
    app.UIAxes.Ylim = [-0.5 0.5];
    %pause(0.018)

end

app.alphaCurrent = app.x(1);
app.betaCurrent = app.x(2);
app.AlphastartEditField.Value = app.x(1);
app.BetastartEditField.Value = app.x(2);
app.AlphaEditField.Value = app.x(1);
app.BetaEditField.Value = app.x(2);

% Value changed function: StepSizeEditField
function StepSizeEditFieldValueChanged(app, event)
    app.stepsize = app.StepsizeEditField.Value;
end

% Button pushed function: YButton
function YButtonPushed(app, event)
    app.desiredY = app.desiredY + app.stepsize;
    app.DesiredYvalueEditField.Value = app.desiredY;
    str = cat(2, 'Desired XY value is: ', num2str(app.desiredX) ', ', num2str(app.desiredY));
    app.PositionTextArea.Value = str;
end
APPENDIX C. MATLAB CODE NEWTON-RAPHSON

app.x = newtonrap(app,app.startX,app.desiredX,app.desiredY,app.Arm1,app.Arm2);
app.x = app.x*180/pi;
str2 = cat(2,'Alpha, Beta is:',num2str(app.x(1)),' ',num2str(app.x(2)),' ',app.resn);
app.AnglesTextArea.Value = str2;
if app.alphaCurrent > app.x(1)
alpha = (app.alphaCurrent:-0.5:app.x(1))*(pi/180);
else
alpha = (app.alphaCurrent:0.5:app.x(1))*(pi/180);
end
l = size(alpha);
interval = (1/(l(2)-1))*(app.betaCurrent-app.x(2));
beta = (app.betaCurrent:-interval:app.x(2))*(pi/180);
x0 = zeros(size(alpha));
y0 = zeros(size(alpha));
x1 = app.Arm1*cos(alpha);
y1 = app.Arm1*sin(alpha);
x2 = x1 + app.Arm2*cos(beta);
y2 = y1 - app.Arm2*sin(beta);
for k = 1:length(alpha)
plot(app.UIAxes,[x0(k) x1(k)],[y0(k) y1(k)],'ro','LineWidth',2)
hold(app.UIAxes,'on')
plot(app.UIAxes,[x2(1) x2(end)],[y2(1) y2(end)],'b--','MarkerSize',10)
plot(app.UIAxes,x2(1),y2(1),'bx','MarkerSize',10)
plot(app.UIAxes,x2(end),y2(end),'bo','MarkerSize',10)
plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro','LineWidth',2)
hold(app.UIAxes,'off')
%app.UIAxes.Xlim = [-0.5 0.5];
%app.UIAxes.Ylim = [-0.5 0.5];
%hold(app.UIAxes);
%plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro--','LineWidth',2)
pause(0.018)
end
app.alphaCurrent = app.x(1);
app.betaCurrent = app.x(2);
app.AlphastartEditField.Value = app.x(1);
app.BetastartEditField.Value = app.x(2);
app.AlphaEditField.Value = app.x(1);
app.BetaEditField.Value = app.x(2);
end

% Button pushed function: YButton_2
function YButton_2Pushed(app, event)
app.desiredY = app.desiredY - app.stepsize;
app.DesiredYvalueEditField.Value = app.desiredY;
str = cat(2,'Desired X,Y value is:',num2str(app.desiredX),' ',num2str(app.desiredY));
app.PositionTextArea.Value = str;
app.x = newtonrap(app,app.startX,app.desiredX,app.desiredY,app.Arm1,app.Arm2);
app.x = app.x*180/pi;
str2 = cat(2,'Alpha, Beta is:',num2str(app.x(1)),' ',num2str(app.x(2)),' ',app.resn);
app.AnglesTextArea.Value = str2;
if app.alphaCurrent > app.x(1)
alpha = (app.alphaCurrent: -0.5:app.x(1))*(pi/180);
else
alpha = (app.alphaCurrent:0.5:app.x(1))*(pi/180);
end
l = size(alpha);
interval = (1/(1(2)-1))*(app.betaCurrent-app.x(2));
beta = (app.betaCurrent:-interval:app.x(2))*(pi/180);
x0 = zeros(size(alpha));
y0 = zeros(size(alpha));
x1 = app.Arm1*cos(alpha);
y1 = app.Arm1*sin(alpha);
x2 = x1 + app.Arm2*cos(beta);
y2 = y1 - app.Arm2*sin(beta);
for k = 1:length(alpha)
plot(app.UIAxes,[x0(k) x1(k)],[y0(k) y1(k)],'ro-','LineWidth',2)
hold(app.UIAxes,'on')
plot(app.UIAxes,[x2(1) x2(end)],[y2(1) y2(end)],'b--','MarkerSize',10)
plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-','LineWidth',2)
hold(app.UIAxes,'off')
%app.UIAxes.Xlim = [-0.5 0.5];
%app.UIAxes.Ylim = [-0.5 0.5];
%hold(app.UIAxes);
%plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-','LineWidth',2)
pause(0.018)
end
app.alphaCurrent = app.x(1);
app.betaCurrent = app.x(2);
app.AlphastartEditField.Value = app.x(1);
app.BetastartEditField.Value = app.x(2);
app.AlphaEditField.Value = app.x(1);
app.BetaEditField.Value = app.x(2);
end
% Create UIAxes
app.UIAxes = uiaxes(app.UIFigure);
title(app.UIAxes, 'Title')
xlabel(app.UIAxes, 'X')
ylabel(app.UIAxes, 'Y')
app.UIAxes.CameraPosition = [0 0 9.16025403784439];
app.UIAxes.CameraTarget = [0 0 0.5];
app.UIAxes.CameraUpVector = [0 1 0];
APPENDIX C. MATLAB CODE NEWTON-RAPHSON

    app.UIAxes.DataAspectRatio = [1 1 1];  
    app.UIAxes.PlotBoxAspectRatio = [1 1 1];  
    app.UIAxes.XLim = [-0.5 0.5];  
    app.UIAxes.YLim = [-0.5 0.5];  
    app.UIAxes.ZLim = [0 1];  
    app.UIAxes.CLim = [0 1];  
    app.UIAxes.XColor = [0.15 0.15 0.15];  
    app.UIAxes.XTick = [-0.5 0 0.5];  
    app.UIAxes.YColor = [0.15 0.15 0.15];  
    app.UIAxes.YTick = [-0.5 0 0.5];  
    app.UIAxes.ZColor = [0.15 0.15 0.15];  
    app.UIAxes.ZTick = [0 0.5 1];  
    app.UIAxes.GridColor = [0.15 0.15 0.15];  
    app.UIAxes.MinorGridColor = [0.1 0.1 0.1];  
    app.UIAxes.Position = [13 23 356 231];  
    % Create Desired X value Edit Field Label  
    app.DesiredXvalueEditFieldLabel = uilabel(app.UIFigure);  
    app.DesiredXvalueEditFieldLabel.HorizontalAlignment = 'right';  
    app.DesiredXvalueEditFieldLabel.VerticalAlignment = 'top';  
    app.DesiredXvalueEditFieldLabel.Position = [25 399 89 15];  
    app.DesiredXvalueEditFieldLabel.Text = 'Desired X value';  
    % Create Desired X value Edit Field  
    app.DesiredXvalueEditField = uieditfield(app.UIFigure, 'numeric');  
    app.DesiredXvalueEditField_ValueChangedFcn = createCallbackFcn(app,  
    @DesiredXvalueEditFieldValueChanged, true);  
    app.DesiredXvalueEditField.Position = [129 395 100 22];  
    % Create Desired Y value Edit Field Label  
    app.DesiredYvalueEditFieldLabel = uilabel(app.UIFigure);  
    app.DesiredYvalueEditFieldLabel.HorizontalAlignment = 'right';  
    app.DesiredYvalueEditFieldLabel.VerticalAlignment = 'top';  
    app.DesiredYvalueEditFieldLabel.Position = [251 399 89 15];  
    app.DesiredYvalueEditFieldLabel.Text = 'Desired Y value';  
    % Create Desired Y value Edit Field  
    app.DesiredYvalueEditField = uieditfield(app.UIFigure, 'numeric');  
    app.DesiredYvalueEditField_ValueChangedFcn = createCallbackFcn(app,  
    @DesiredYvalueEditFieldValueChanged, true);  
    app.DesiredYvalueEditField.Position = [355 395 100 22];  
    % Create Alpha start Edit Field Label  
    app.AlphastartEditFieldLabel = uilabel(app.UIFigure);  
    app.AlphastartEditFieldLabel.HorizontalAlignment = 'right';  
    app.AlphastartEditFieldLabel.VerticalAlignment = 'top';  
    app.AlphastartEditFieldLabel.Position = [50 435 64 15];  
    app.AlphastartEditFieldLabel.Text = 'Alpha start';  
    % Create Alpha start Edit Field  
    app.AlphastartEditField = uieditfield(app.UIFigure, 'numeric');  
    app.AlphastartEditField_ValueChangedFcn = createCallbackFcn(app,  
    @AlphastartEditFieldValueChanged, true);  
    app.AlphastartEditField.Position = [129 431 100 22];  
    % Create Beta start Edit Field Label  
    app.BetastartEditFieldLabel = uilabel(app.UIFigure);  
    app.BetastartEditFieldLabel.HorizontalAlignment = 'right';  
    app.BetastartEditFieldLabel.VerticalAlignment = 'top';  
    app.BetastartEditFieldLabel.Position = [282 432 58 15];  
    app.BetastartEditFieldLabel.Text = 'Beta start';
app.BetastartEditField = uieeditfield(app.UIFigure, 'numeric');
app.BetastartEditField.ValueChangedFcn = createCallbackFcn(app, @BetastartEditFieldValueChanged, true);
app.BetastartEditField.Position = [355 428 100 22];

app.PositionTextAreaLabel = uilabel(app.UIFigure);
app.PositionTextAreaLabel.HorizontalAlignment = 'right';
app.PositionTextAreaLabel.VerticalAlignment = 'top';
app.PositionTextAreaLabel.Position = [30 305 55 22];
app.PositionTextAreaLabel.Text = 'Position';

app.PositionTextArea = uitextarea(app.UIFigure);
app.PositionTextArea.Editable = 'off';
app.PositionTextArea.Position = [100 269 150 60];

app.CalculateButton = uibutton(app.UIFigure, 'push');
app.CalculateButton.ButtonPushedFcn = createCallbackFcn(app, @CalculateButtonPushed, true);
app.CalculateButton.Position = [355 354 100 22];
app.CalculateButton.Text = 'Calculate';

app.AnglesTextAreaLabel = uilabel(app.UIFigure);
app.AnglesTextAreaLabel.HorizontalAlignment = 'right';
app.AnglesTextAreaLabel.VerticalAlignment = 'top';
app.AnglesTextAreaLabel.Position = [282 307 62 22];
app.AnglesTextAreaLabel.Text = 'Angles';

app.AnglesTextArea = uitextarea(app.UIFigure);
app.AnglesTextArea.Position = [359 271 150 60];

app.MoveButton = uibutton(app.UIFigure, 'push');
app.MoveButton.ButtonPushedFcn = createCallbackFcn(app, @MoveButtonPushed, true);
app.MoveButton.Position = [429 174 100 22];
app.MoveButton.Text = 'Move';

app.AlphaEditFieldLabel = uilabel(app.UIFigure);
app.AlphaEditFieldLabel.HorizontalAlignment = 'right';
app.AlphaEditFieldLabel.VerticalAlignment = 'top';
app.AlphaEditFieldLabel.Position = [332 221 36 15];
app.AlphaEditFieldLabel.Text = 'Alpha';

app.AlphaEditField = uieeditfield(app.UIFigure, 'numeric');
app.AlphaEditField.ValueChangedFcn = createCallbackFcn(app, @AlphaEditFieldValueChanged, true);
app.AlphaEditField.Position = [383 217 76 22];

app.BetaEditFieldLabel = uilabel(app.UIFigure);
app.BetaEditFieldLabel.HorizontalAlignment = 'right';
app.BetaEditFieldLabel.VerticalAlignment = 'top';
app.BetaEditFieldLabel.Position = [483 221 30 15];
app.BetaEditFieldLabel.Text = 'Beta';

app.BetaEditField = uieeditfield(app.UIFigure, 'numeric');
app.BetaEditField.ValueChangedFcn = createCallbackFcn(app, @BetaEditFieldValueChanged, true);
app.BetaEditField.Position = [483 221 30 15];
app.BetaEditFieldLabel.Text = 'Beta';
APPENDIX C. MATLAB CODE NEWTON-RAPHSON

537 app.BetaEditField = ueditfield(app.UIFigure, 'numeric');
538 app.BetaEditField.ValueChangedFcn = createCallbackFcn(app, 
      @BetaEditFieldValueChanged, true);
539 app.BetaEditField.Position = [528 217 76 22];
540 % Create XButton
541 app.XButton = uibutton(app.UIFigure, 'push');
542 app.XButton.ButtonPressedFcn = createCallbackFcn(app, @XButtonPressed , true);
543 app.XButton.Position = [494 88 35 22];
544 app.XButton.Text = 'X+';
545 % Create XButton_2
546 app.XButton_2 = uibutton(app.UIFigure, 'push');
547 app.XButton_2.ButtonPressedFcn = createCallbackFcn(app, @XButton_2Pressed, true);
548 app.XButton_2.Position = [449 88 35 22];
549 app.XButton_2.Text = 'X-';
550 % Create YButton
551 app.YButton = uibutton(app.UIFigure, 'push');
552 app.YButton.ButtonPressedFcn = createCallbackFcn(app, @YButtonPressed, true);
553 app.YButton.Position = [495 57 34 22];
554 app.YButton.Text = 'Y+';
555 % Create YButton_2
556 app.YButton_2 = uibutton(app.UIFigure, 'push');
557 app.YButton_2.ButtonPressedFcn = createCallbackFcn(app, @YButton_2Pressed, true);
558 app.YButton_2.Position = [450 57 34 22];
559 app.YButton_2.Text = 'Y-';
560 % Create StepsizeEditFieldLabel
561 app.StepsSizeEditFieldLabel = uilabel(app.UIFigure);
562 app.StepsSizeEditFieldLabel.HorizontalAlignment = 'right';
563 app.StepsSizeEditFieldLabel.Position = [339 78 51 22];
564 app.StepsSizeEditFieldLabel.Text = 'Stepsize';
565 % Create StepsizeEditField
566 app.StepsSizeEditField = ueditfield(app.UIFigure, 'numeric');
567 app.StepsSizeEditField.ValueChangedFcn = createCallbackFcn(app, @StepsSizeEditFieldValueChanged, true);
568 app.StepsSizeEditField.Position = [398 78 32 22];
569 app.StepsSizeEditField.Value = 0.01;
570 % Show the figure after all components are created
571 app.UIFigure.Visible = 'on';
572 end
573 end
574 % App creation and deletion
575 methods (Access = public)
576 % Construct app
577 function app = appTestArm
578 % Create UIFigure and components
579 createComponents(app)
580 % Register the app with App Designer
581 registerApp(app, app.UIFigure)
582 % Execute the startup function
583 runStartupFcn(app, @startupFcn)
584 if nargout == 0
clear app
end
end

% Code that executes before app deletion
function delete(app)
% Delete UIFigure when app is deleted
delete(app.UIFigure)
end
end]]>
</w:t>
</w:r>
</w:p>
</w:body>
</w:document>
Appendix D

MATLAB Code Fuzzylogic

Listing D.1. Source Code

1     <w:document
2     xmlns:w="http://schemas.openxmlformats.org/wordprocessingml
3             /2006/main">
4     <w:body>
5     <w:p>
6         <w:pPr w:val="code">
7             <w:pStyle w:val="code"/>
8         </w:pPr>
9         <w:r>
10            <![CDATA<class def plotSIMONLY < matlab.apps.
11                  AppBase
12
13             %KTH Royal Institute of Technology.
14             %Bachelor’s Thesis in Mechatronics.
15             %Multipurpose robot arm.
16             %Multifunktions robotarm.
17             %Authors: Fahim Pirmohamed (fahimp@kth.se),
18             %Alexander Aronsson (alearo@kth.se).
19             %Course code: MF133X.
20             %Examiner: Nihad Subasic.
21             %TRITA: 2021:34.
22             %File for MATLAB ANFIS Control.
23             %Properties that correspond to app components
24             properties (Access = public)
25             UIFigure         matlab.ui.Figure
26             MoveButton      matlab.ui.control.Button
27             YEditField      matlab.ui.control.NumericEditField
28             YEditFieldLabel matlab.ui.control.Label
29             XEditField      matlab.ui.control.NumericEditField
30             XEditFieldLabel matlab.ui.control.Label
31
32     </w:r>
33     </w:p>
34     </w:body>
35     </w:document>
X_stepDownButton matlab.ui.control.Button
X_stepUpButton matlab.ui.control.Button
SolutionTextArea matlab.ui.control.TextArea
SolutionTextAreaLabel matlab.ui.control.Label
YSliderLabel matlab.ui.control.Label
YSlider matlab.ui.control.Slider
XSlider matlab.ui.control.Slider
XSliderLabel matlab.ui.control.Label
BetaSlider matlab.ui.control.Slider
BetaSliderLabel matlab.ui.control.Label
XYTextArea matlab.ui.control.TextArea
XYTextAreaLabel matlab.ui.control.Label
AlphaSlider matlab.ui.control.Slider
AlphaSliderLabel matlab.ui.control.Label
UIAxes matlab.ui.control.UIAxes

end

properties (Access = private)
alpha = 120 % Description
beta = 120
Arm1 = 0.5
Arm2 = 0.3
startX = [50*pi/180; 30*pi/180]
tol = 1e-12 % Description
imax = 10000 % Description
x = [50*pi/180; 30*pi/180]
cPos = 0
anfis1
anfis2
resn
a
s1
s2
s3
s4
s5
s6
vBas = 90 % Description
vArm = 90
vAxel = 180
vRot = 0
vHand = 0
vKlo = 200
xx
yy
end
APPENDIX D. MATLAB CODE FUZZYLOGIC

methods (Access = private)

function results = plotFig(app)
x0 = 0;
y0 = 0;
x1 = app.Arm1*cos(app.alpha*pi/180);
y1 = app.Arm1*sin(app.alpha*pi/180);
bo = app.beta;
x2 = x1 + app.Arm2*cos((app.alpha + bo)*pi/180);
y2 = y1 + app.Arm2*sin((app.alpha + bo)*pi/180);
plot(app.UIAxes,[x0 x1],[y0 y1],'ro-','LineWidth',2)
hold(app.UIAxes,"on")
plot(app.UIAxes,[x1 x2],[y1 y2],'ro-','LineWidth',2)
hold(app.UIAxes,"off")
end

function pos = calcPos(app)
x1 = app.Arm1*cos(app.alpha*pi/180);
y1 = app.Arm1*sin(app.alpha*pi/180);
bo = app.beta;
x2 = x1 + app.Arm2*cos((app.alpha + bo)*pi/180);
y2 = y1 + app.Arm2*sin((app.alpha + bo)*pi/180);
pos = [x2,y2];
end

function results = anfisSetup(app)
l1 = app.Arm1; % length of first arm
l2 = app.Arm2; % length of second arm
theta1 = 0:0.1:pi/2; % all possible theta1 values
theta2 = 0:0.1:pi; % all possible theta2 values
[THEtA1,THEtA2] = meshgrid(theta1,theta2); % generate a
grid of theta1 and theta2 values
BO = THEtA2;
X = l1*cos(THEtA1) + l2*cos(THEtA1 + BO); % compute
x coordinates
Y = l1*sin(THEtA1) + l2*sin(THEtA1 + BO); % compute
y coordinates
data1 = [X(:, Y(:, THETA1(:))]; % create x-y-theta1 dataset
data2 = [X(:, Y(:, THETA2(:))]; % create x-y-theta2 dataset

opt = anfisOptions;
opt.InitialFIS = 7;
opt.EpochNumber = 150;
opt.DisplayANFISInformation = 0;
opt.DisplayErrorValues = 0;
opt.DisplayStepSize = 0;
opt.DisplayFinalResults = 0;

app. SolutionTextArea.Value = 'Training first ANFIS network.';
app.anfis1 = anfis(data1, opt);
app. SolutionTextArea.Value = 'Training second ANFIS network.';
opt. InitialFIS = 6;
app.anfis2 = anfis(data2, opt);
app. SolutionTextArea.Value = 'Training done';
end
end
% Callbacks that handle component events
% Code that executes after component creation
function startupFcn(app)
axis(app.UIAxes, 'square')
grid(app.UIAxes, 'on')
app.plotFig();
app.cPos = app.calcPos();
app.XYTextArea.Value = num2str(app.cPos);
app.XSlider.Value = app.cPos(1);
app.YSlider.Value = app.cPos(2);
app.anfisSetup();
app.a = arduino();
app.s1 = servo(app.a, 'D4');
app.s2 = servo(app.a, 'D5');
app.s3 = servo(app.a, 'D6');
app.s4 = servo(app.a, 'D7');
app.s5 = servo(app.a, 'D8');
app.s6 = servo(app.a, 'D9');
writePosition(app.s1, app.vBas/180);
writePosition(app.s2, app.vArm/180);
writePosition(app.s3, app.vAxel/180);
writePosition(app.s4, 0);
writePosition(app.s5, 0);
writePosition(app.s6, app.vKlo/270);
end
% Value changing function: AlphaSlider
function AlphaSliderValueChanging(app, event)
changingValue = event.Value;
app.alpha = changingValue;
app.plotFig();
app.cPos = app.calcPos();
APPENDIX D. MATLAB CODE FUZZYLOGIC

190 app.XYTextArea.Value = num2str(app.cPos);
191 app.XSlider.Value = app.cPos(1);
192 app.YSlider.Value = app.cPos(2);
193 writePosition(app.s2, app.alpha/180);
194 end
195 % Value changing function: BetaSlider
196 function BetaSliderValueChanging(app, event)
197 changingValue = event.Value;
198 app.beta = changingValue;
199 app.plotFig();
200 app.cPos = app.calcPos();
201 app.XYTextArea.Value = num2str(app.cPos);
202 app.XSlider.Value = app.cPos(1);
203 app.YSlider.Value = app.cPos(2);
204 writePosition(app.s3, app.beta/180);
205 end
206 % Value changing function: XSlider
207 function XSliderValueChanging(app, event)
208 changingValue = event.Value;
209 xNew = changingValue;
210 XY = [xNew, app.cPos(2)];
211 app.alpha = evalfis(app.anfis1,XY)*180/pi;
212 app.beta = evalfis(app.anfis2,XY)*180/pi;
213 %app.AlphaSlider.Value = app.alpha;
214 %app.BetaSlider.Value = app.beta;
215 app.cPos = app.calcPos();
216 app.plotFig();
217 app.XYTextArea.Value = num2str(app.cPos);
218 app.SolutionTextArea.Value = num2str(app.alpha);
219 %writePosition(app.s2, app.alpha/180);
220 %writePosition(app.s3, app.beta/180);
221 end
222 % Value changing function: YSlider
223 function YSliderValueChanging(app, event)
224 changingValue = event.Value;
225 yNew = changingValue;
226 XY = [app.cPos(1), yNew];
227 app.alpha = evalfis(app.anfis1,XY)*180/pi;
228 app.beta = evalfis(app.anfis2,XY)*180/pi;
229 %app.AlphaSlider.Value = app.alpha;
230 %app.BetaSlider.Value = app.beta;
231 app.cPos = app.calcPos();
232 app.plotFig();
233 app.XYTextArea.Value = num2str(app.cPos);
234 app.SolutionTextArea.Value = num2str(app.alpha);
235 end
236 % Button pushed function: X_stepUpButton
237 function X_stepUpButtonPushed(app, event)
238 xNew = app.cPos(1) + 0.01;
239 newAngles = app.newtonrapOld([app.alpha; app.beta], xNew, app.cPos(2),
240 app.Arm1, app.Arm2);
241 app.alpha = newAngles(1);
242 app.beta = newAngles(2);
243 app.cPos = app.calcPosOld;
app. SolutionTextArea.Value = app.resn;
app.plotFigOld();
end

% Button pushed function: X_stepDownButton
function X_stepDownButtonPushed(app, event)
xNew = app.cPos(1) - 0.01;
newAngles = app.newtonrapOld([app.alpha; app.beta], xNew, app.cPos(2),
    . Arm1, . Arm2);
app.alpha = newAngles(1);
app.beta = newAngles(2);
app.cPos = app.calcPosOld;
app.SolutionTextArea.Value = app.resn;
app.plotFigOld();
end

% Value changed function: XEditField
function XEditFieldValueChanged(app, event)
app.xx = app.XEditField.Value;
end

% Value changed function: YEditField
function YEditFieldValueChanged(app, event)
app.yy = app.YEditField.Value;
end

% Button pushed function: MoveButton
function MoveButtonPushed(app, event)
xy = [app.xx; app.yy];
app.alpha = evalfis(app.anfis1, xy)*180/pi;
app.beta = evalfis(app.anfis2, xy)*180/pi;
app.cPos = app.calcPos();
app.plotFig();
app.AlphaSlider.Value = app.alpha;
app.BetaSlider.Value = app.beta;
end

% Component initialization
methods (Access = private)

% Create UIFigure and components
function createComponents(app)
% Create UIFigure and hide until all components are created
app.UIFigure = uifigure('Visible', 'off');
app.UIFigure.Position = [100 100 640 480];
app.UIFigure.Name = 'MATLAB App';
% Create UIAxes
app.UIAxes = uiaxes(app.UIFigure);
title(app.UIAxes, 'Title')
xlabel(app.UIAxes, 'X')
ylabel(app.UIAxes, 'Y')
zlabel(app.UIAxes, 'Z')
app.UIAxes.XLim = [-1 1];
app.UIAxes.YLim = [-1 1];
app.UIAxes.Position = [26 19 349 322];
% Create AlphaSliderLabel
app.AlphaSliderLabel = uilabel(app.UIFigure);
app.AlphaSliderLabel.HorizontalAlignment = 'right';
app.AlphaSliderLabel.Position = [400 422 36 22];
APPENDIX D. MATLAB CODE FUZZYLOGIC

296  app.AlphaSliderLabel.Text = 'Alpha';
297  % Create AlphaSlider
298  app.AlphaSlider = uislider(app.UIFigure);
299  app.AlphaSlider.Limits = [0 180];
300  app.AlphaSlider.ValueChangingFcn = createCallbackFcn(app,
301      @AlphaSliderValueChanging, true);
302  app.AlphaSlider.Position = [457 431 150 3];
303  app.AlphaSlider.Value = 65;
304  % Create XYTextAreaLabel
305  app.XYTextAreaLabel = uilabel(app.UIFigure);
306  app.XYTextAreaLabel.HorizontalAlignment = 'right';
307  app.XYTextAreaLabel.Position = [415 300 30 22];
308  app.XYTextAreaLabel.Text = 'X, Y:';
309  % Create XYTextArea
310  app.XYTextArea = uitextarea(app.UIFigure);
311  app.XYTextArea.Position = [457 301 150 20];
312  % Create BetaSliderLabel
313  app.BetaSliderLabel = uilabel(app.UIFigure);
314  app.BetaSliderLabel.HorizontalAlignment = 'right';
315  app.BetaSliderLabel.Position = [403 361 30 22];
316  app.BetaSliderLabel.Text = 'Beta';
317  % Create BetaSlider
318  app.BetaSlider = uislider(app.UIFigure);
319  app.BetaSlider.Limits = [0 180];
320  app.BetaSlider.ValueChangingFcn = createCallbackFcn(app,
321      @BetaSliderValueChanging, true);
322  app.BetaSlider.Position = [454 370 150 3];
323  % Create XSliderLabel
324  app.XSliderLabel = uilabel(app.UIFigure);
325  app.XSliderLabel.HorizontalAlignment = 'right';
326  app.XSliderLabel.Position = [406 246 25 22];
327  app.XSliderLabel.Text = 'X';
328  % Create XSlider
329  app.XSlider = uislider(app.UIFigure);
330  app.XSlider.Limits = [-1 1];
331  app.XSlider.ValueChangingFcn = createCallbackFcn(app,
332      @XSliderValueChanging, true);
333  app.XSlider.Position = [452 255 150 3];
334  % Create YSliderLabel
335  app.YSliderLabel = uilabel(app.UIFigure);
336  app.YSliderLabel.HorizontalAlignment = 'center';
337  app.YSliderLabel.Position = [413 189 25 22];
338  app.YSliderLabel.Text = 'Y';
339  % Create YSlider
340  app.YSlider = uislider(app.UIFigure);
341  app.YSlider.Limits = [-1 1];
342  app.YSlider.ValueChangingFcn = createCallbackFcn(app,
343      @YSliderValueChanging, true);
344  app.YSlider.Position = [456 199 150 3];
345  % Create SolutionTextAreaLabel
346  app.SolutionTextAreaLabel = uilabel(app.UIFigure);
347  app.SolutionTextAreaLabel.HorizontalAlignment = 'right';
348  app.SolutionTextAreaLabel.Position = [131 388 50 22];
349  app.SolutionTextAreaLabel.Text = 'Solution';
% Create SolutionTextArea
app.SolutionTextArea = uitextarea(app.UIFigure);
app.SolutionTextArea.Position = [196 352 150 60];

% Create X_stepUpButton
app.X_stepUpButton = uibutton(app.UIFigure, 'push');
app.X_stepUpButton.ButtonPushedFcn = createCallbackFcn(app, @X_stepUpButtonPushed, true);
app.X_stepUpButton.Position = [269 422 77 22];
app.X_stepUpButton.Text = 'X_stepUp';

% Create X_stepDownButton
app.X_stepDownButton = uibutton(app.UIFigure, 'push');
app.X_stepDownButton.ButtonPushedFcn = createCallbackFcn(app, @X_stepDownButtonPushed, true);
app.X_stepDownButton.Position = [173 422 83 22];
app.X_stepDownButton.Text = 'X_stepDown';

% Create X_EditFieldLabel
app.XEditFieldLabel = uilabel(app.UIFigure); app.XEditFieldLabel.HorizontalAlignment = 'right';
app.XEditFieldLabel.Position = [10 431 25 22];
app.XEditFieldLabel.Text = 'X';

% Create X_EditField
app.XEditField = uieditfield(app.UIFigure, 'numeric');
app.XEditField.ValueChangedFcn = createCallbackFcn(app, @XEditFieldValueChanged, true);
app.XEditField.Position = [50 431 29 22];

% Create Y_EditFieldLabel
app.YEditFieldLabel = uilabel(app.UIFigure); app.YEditFieldLabel.HorizontalAlignment = 'right';
app.YEditFieldLabel.Position = [78 431 25 22];
app.YEditFieldLabel.Text = 'Y';

% Create Y_EditField
app.YEditField = uieditfield(app.UIFigure, 'numeric');
app.YEditField.ValueChangedFcn = createCallbackFcn(app, @YEditFieldValueChanged, true);
app.YEditField.Position = [118 431 29 22];

% Create MoveButton
app.MoveButton = uibutton(app.UIFigure, 'push');
app.MoveButton.ButtonPushedFcn = createCallbackFcn(app, @MoveButtonPushed, true);
app.MoveButton.Position = [26 388 100 22];
app.MoveButton.Text = 'Move';

% Show the figure after all components are created
app.UIFigure.Visible = 'on';
end
end

methods (Access = public)

% Construct app
function app = plotTestSIMONLY
% Create UIFigure and components
createComponents(app)
% Register the app with App Designer
registerApp(app, app.UIFigure)
% Execute the startup function
runStartupFcn(app, @startupFcn)
if nargout == 0
clear app
end
end
% Code that executes before app deletion
function delete(app)
% Delete UIFigure when app is deleted
delete(app.UIFigure)
end
end

<w:document>