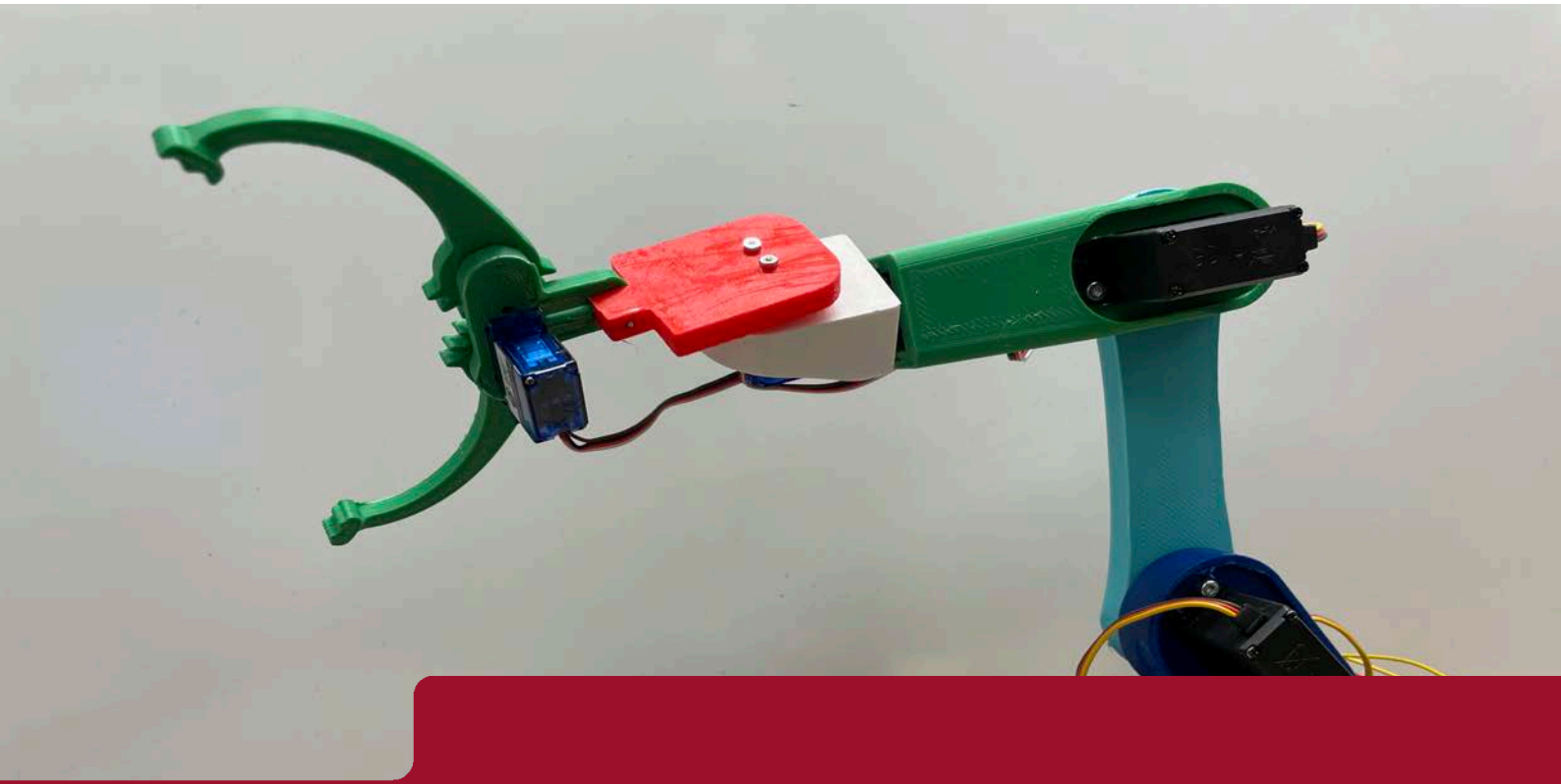
DEGREE PROJECT IN MECHANICAL ENGINEERING,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2021

Multipurpose Robot Arm

ALEXANDER ARONSSON

FAHIM PIRMOHAMED



KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT

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 angels 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 kanditatexamensarbete 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 manuell 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.

Resultaten visar att the manuella kontrollmetoden var enkel att implementera, snabb samt var tillförlitlig. Den gav precis styrning av alla vinklar för varje servomotor, vilket också innebar att den gav god styrning av varje frihetsgrad. Den numeriska metoden, mer bestämt Newton Raphson's metod, vidgade möjligheterna att kontrollera armen men var långsammare. Den tredje och sista lösningen var att använda ett neuralt nätverk, fuzzy logic. Detta visade sig vara ett kraftfullt sätt att styra roboten med låg latens. Det neurala nätverket visade sig dock vara opålitligt, men metoden visade stor potential för vidare utveckling och kan då prestera mycket bättre än de andra två metoderna.

Slutsatsen var att det neurala nätverket var den generellt bästa metoden för att kontrollera och manövrera 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

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

List of Tables

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 devoloped: 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.

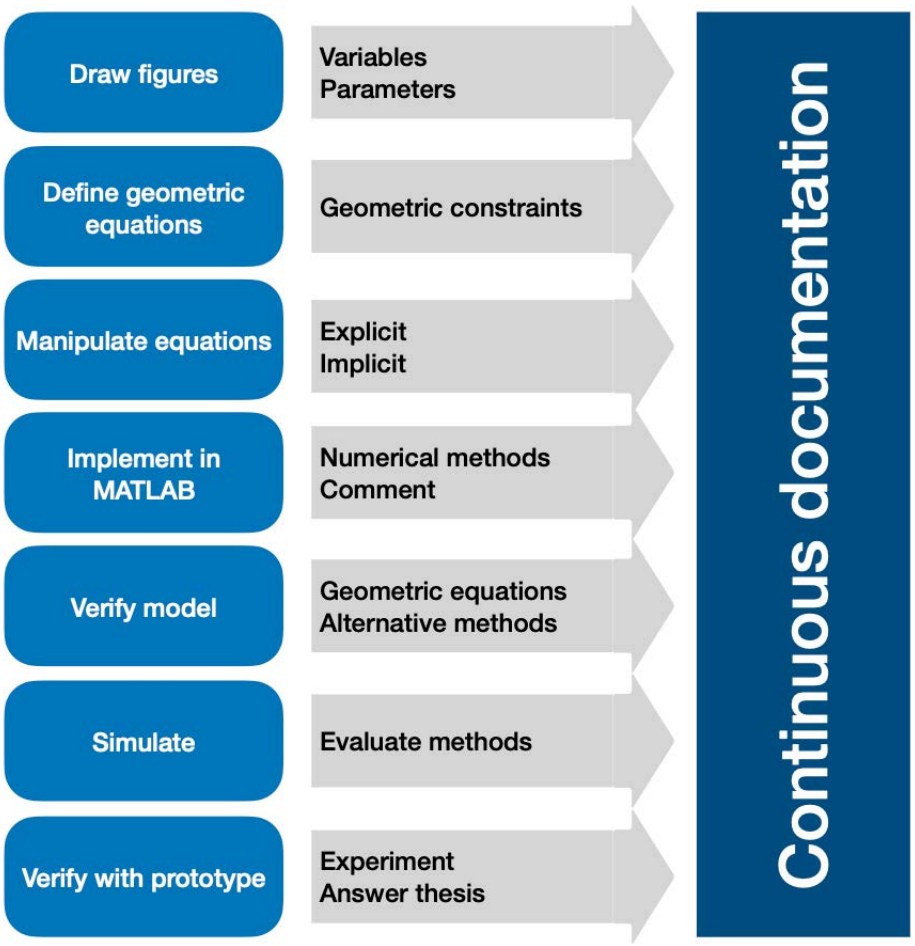


Figure 1.1. Method for developing software, drawn in Pages.

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 microcontrollers 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.

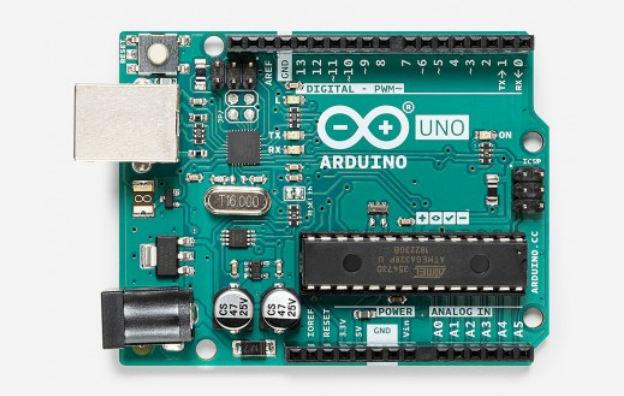


Figure 2.1. Arduino Uno Rev3 [4].

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].

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2.1}$$

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.

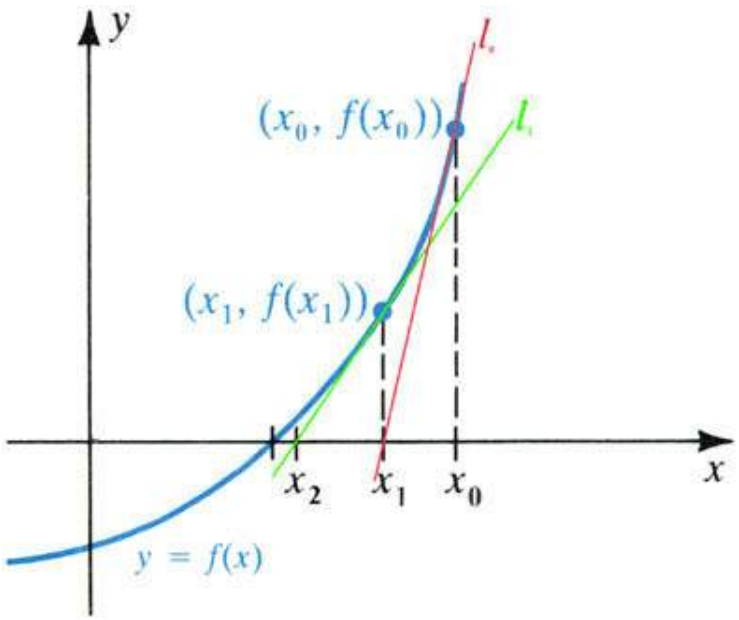


Figure 2.2. Illustrative graph of Newton Raphson's method [12].

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 fuzzyfing 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].

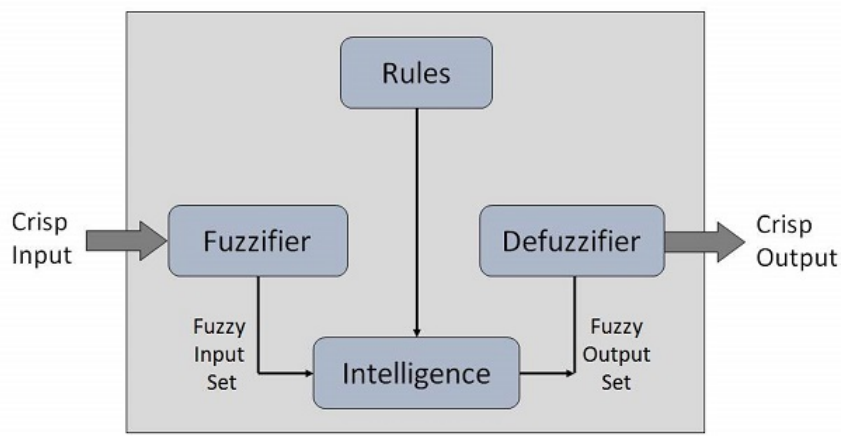


Figure 2.3. Fuzzy Logic Systems Architecture [13].

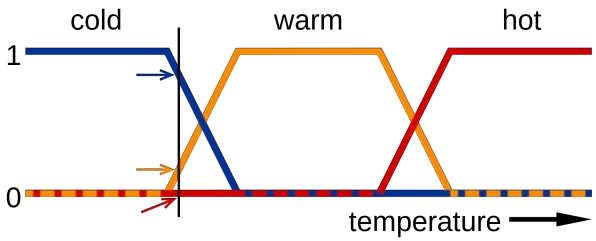


Figure 2.4. Fuzzfing input value [15].

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.

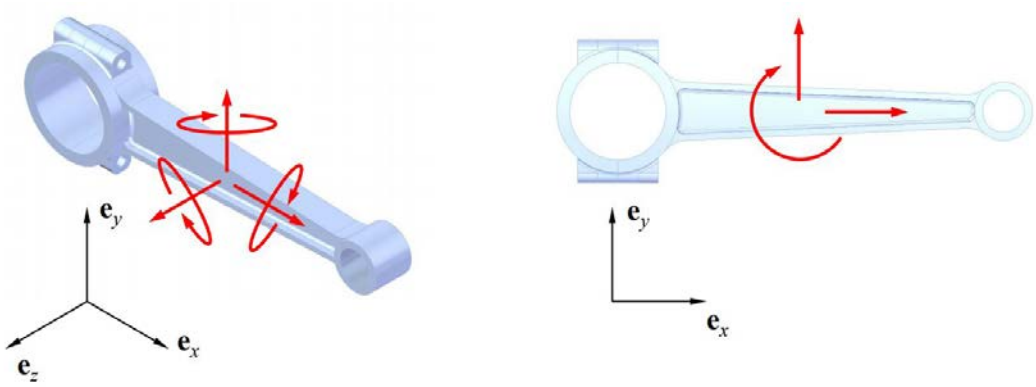


Figure 2.5. Free-body in 3D and 2D Space [16].

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

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

F	Degrees of freedom
N	Number of links (including stand/support)
f	Degrees of freedom in joints
m_f	Number of joints with f degrees of freedom

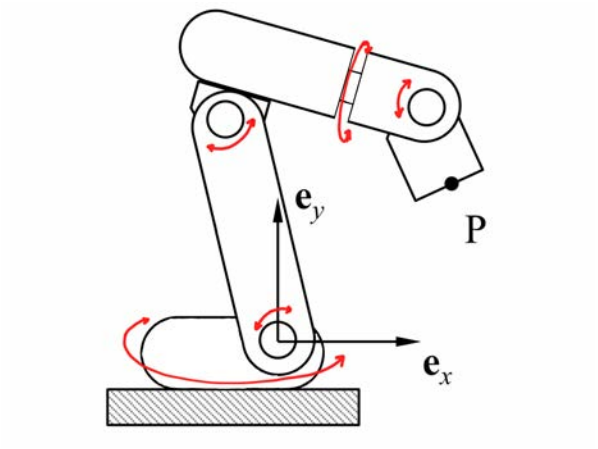


Figure 2.6. Modified schematic picture for our prototype [16].

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.

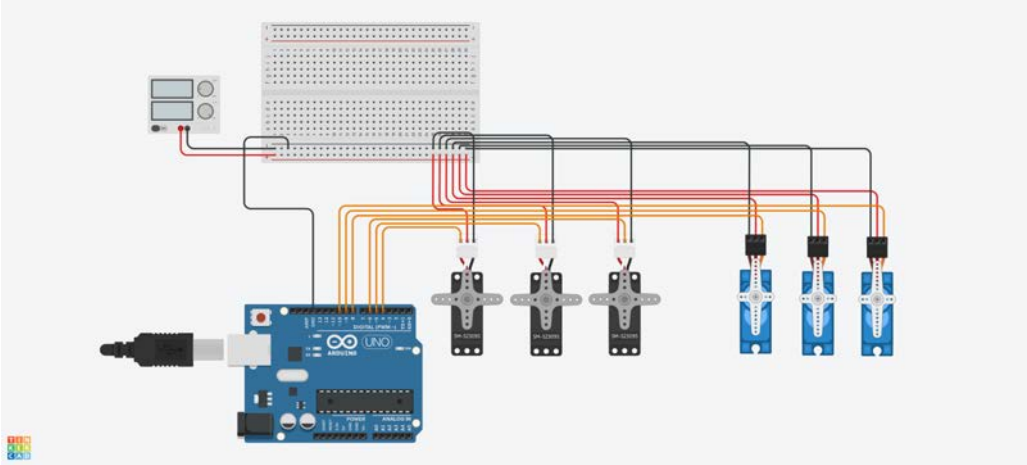


Figure 3.1. Circuit diagram for our prototype made in Tinkercad.

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.

3.1.2 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.

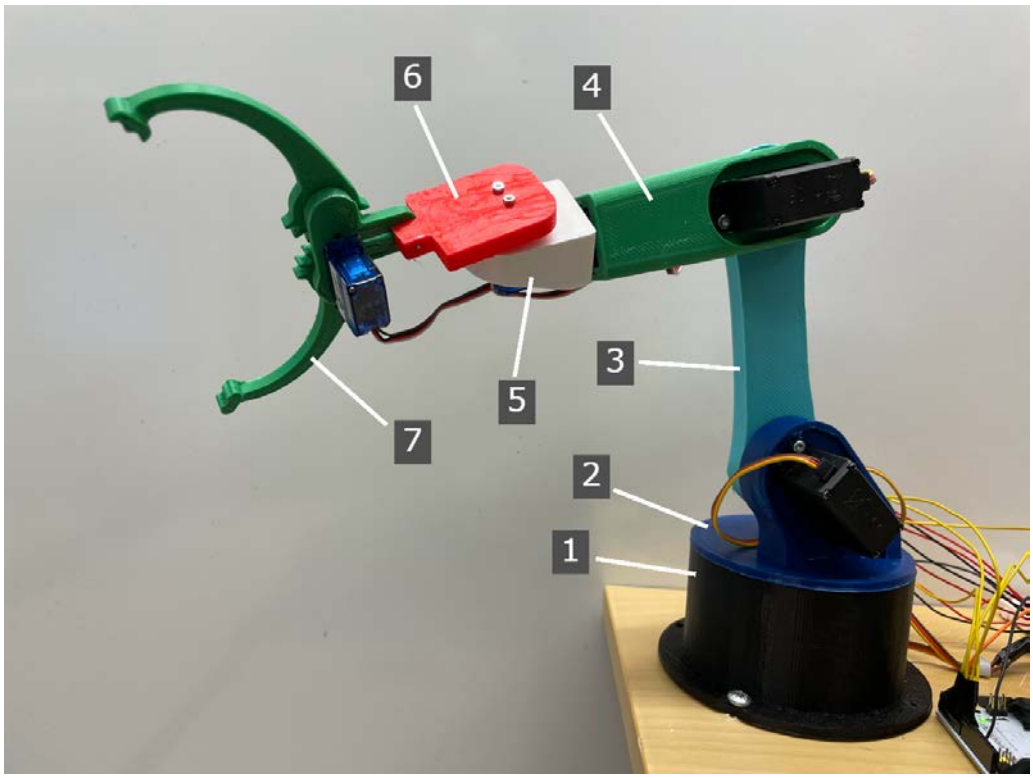


Figure 3.2. Completed construction, captured with iPhone 12 Pro by authors.

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.

3.2.3 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.

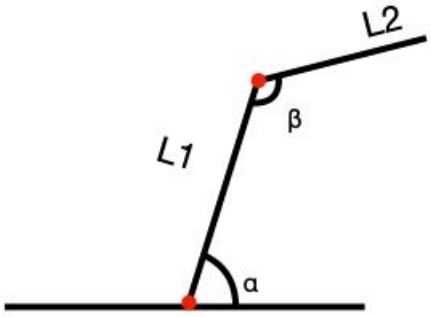


Figure 3.3. Schematic figure for robot arm, drawn in Pages.

α	Angle of servo 1, rotating link 3
β	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.

$$X = L_1 \cdot \cos(\alpha) + L_2 \cdot \cos(\alpha + \beta) \tag{3.1}$$

$$Y = L_1 \cdot \sin(\alpha) + L_2 \cdot \sin(\alpha + \beta) \tag{3.2}$$

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.

Figure 3.4. Manual interface made in MATLAB app-designer.

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

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

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 α and β. 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:

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

$$\bar{J}_1 = \frac{\delta F}{\delta \alpha} \tag{3.4}$$

$$\bar{J}_2 = \frac{\delta F}{\delta \beta} \tag{3.5}$$

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.

Method	**Control**	**Reliability**	**Speed**	**Total**
Manual control	4/10	10/10	10/10	24/30
Numerical analysis	10/10	8/10	8/10	26/30
Neural network	10/10	6/10	10/10	26/30

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.

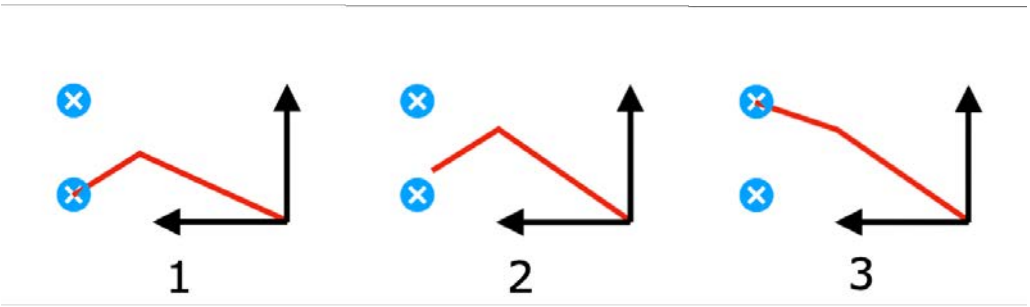


Figure 4.1. Representation of controlling one servo at a time to reach end-point, drawn in Pages.

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 β is rotated 90° clockwise and the range of α is 180° instead of 90°.

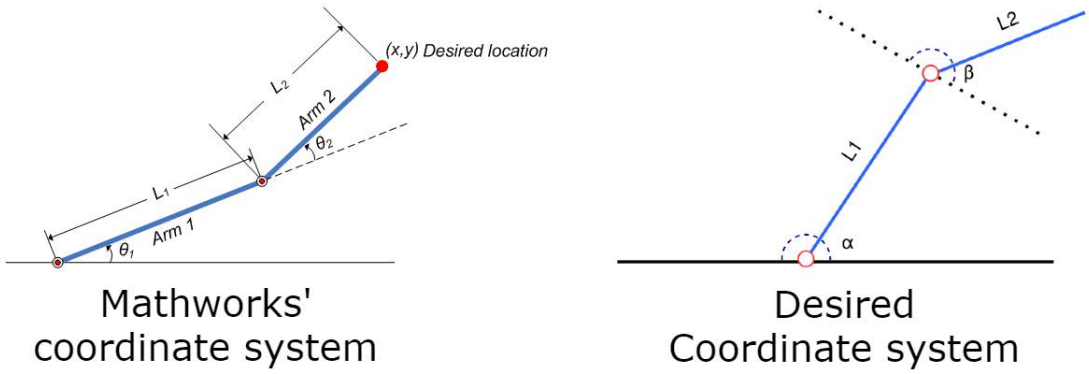


Figure 4.2. Comparison of coordinate system, left picture from Mathworks [20], right picture made in draw.io.

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 α, β 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.

Bibliography

[1] M. E. Moran. *Evolution of robotic arms* Journal of Robotic Surgery, 2007. [Online]. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4247431/

[2] Keystone Electronics Corp. *How Robots Have Changed Manufacturing* Keystone Electronics Corp Online, 2013. Available from: https://www.keyelco.com/blog-details.cfm?blog_id=40 Accessed: 2021-02-23

[3] K. Miller. *Assistive Feeding: AI Improves Control of Robot Arms* Stanford University, 2020. [Online]. Available from: https://hai.stanford.edu/blog/assistive-feeding-ai-improves-control-robot-arms

[4] Arduino. *About Us* Arduino website, 2020. Available from: https://www.Arduino.cc/en/Main/AboutUs Accessed: 2021-02-23

[5] How to Mechatronics. *Arduino Tutorials* How to Mechatronics website, 2021. Available from: https://howtomechatronics.com/tutorials/arduino/diy-arduino-robot-arm-with-smartphone-control/ Accessed: 2021-04-11

[6] Mathworks. *Hardware support* Mathworks website, 2021. Available from: https://se.mathworks.com/hardware-support/arduino-MATLAB.html Accessed: 2021-04-11

[7] J. Davies. *MSP430 MICROCONTROLLER BASICS* Newnes, 2008, ch. 1, sec. 3, pp. 5-6.

[8] Arduino. *Arduino UNO REV3* Arduino website, 2020. Available from: https://store.Arduino.cc/Arduino-uno-rev3 Accessed: 2021-03-29

[9] M. Sustek et al. *DC motors and servo-motors controlled by Raspberry Pi 2B,* Tomas Bata University, Faculty of Applied Informatics, Department of Automation and Control Engineering, 76005 Zlín, Czech Republic 2017, Vol.

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

[10] Ankur Bhargava. *Arduino controlled robotic arm,* University School of Information, Communication and Technology Guru Gobind Singh Indraprastha University Delhi, India 2017. [Online]. Available from: https://doi.org/10.1109/ICECA.2017.8212837

[11] Manolo Dorto. *Comparing Different Approaches for Solving Large Scale Power Flow Problems on the CPU and GPU with the Newton-Raphson Method* KTH Royal Institute of Technology, School of Electrical Engineering and Computer Science, Stockholm, Sweden, 2020. [Online]. Available from: http://kth.diva-portal.org/smash/record.jsf?pid=diva2:1523039

[12] J. M. Mahaffy. *Math 122 - Calculus for Biology II,* San Diego State University, USA 2000. Available from: https://jmahaffy.sdsu.edu/courses/f00/math122/lectures/newtons_method/newtonmethod.html Accessed: 2021-04-02

[13] Tutorialspoint *Artificial Intelligence - Fuzzy Logic Systems* Available from: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_fuzzy_logic_systems.htm Accessed: 2021-04-10

[14] Song, Inson et al. *Intelligent Parking System Design Using FPGA*, 2006, pp. 3. DOI:10.1109/FPL.2006.311249

[15] Wierman J. Mark *An Introduction to the Mathematics of Uncertainty*, Creighton University, 2012, pp. 115.

[16] Söderberg Anders. *Lecture from MF1064,* KTH Royal Institute of Technology, Sweden 2020.

[17] Atmel *ATmega328P Datasheet* 7810D-AVR, January 2015.

[18] Components101 *MG996R Servo Motor Datasheet* April 2019.

[19] Towerpro *SG90 Servo Datasheet*, 2021.

[20] Mathworks *Modeling Inverse Kinematics in a Robotic Arm*, 2021. Available from: https://se.mathworks.com/help/fuzzy/modeling-inverse-kinematics-in-a-robotic-arm.html Accessed: 2021-04-10

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

```
<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
```

```
35          ResetButton             matlab.ui.control.Button
36          KloEditField            matlab.ui.control.NumericEditField
37          KloEditFieldLabel       matlab.ui.control.Label
38          HandEditField           matlab.ui.control.NumericEditField
39          HandEditFieldLabel      matlab.ui.control.Label
40          RotEditField            matlab.ui.control.NumericEditField
41          RotEditFieldLabel       matlab.ui.control.Label
42          kloSlider               matlab.ui.control.Slider
43          kloSliderLabel          matlab.ui.control.Label
44          HandSlider              matlab.ui.control.Slider
45          HandSliderLabel         matlab.ui.control.Label
46          RotSlider               matlab.ui.control.Slider
47          RotSliderLabel          matlab.ui.control.Label
48          NextButton              matlab.ui.control.Button
49          SaveButton              matlab.ui.control.Button
50          MoveButton              matlab.ui.control.Button
51          AxelEditField           matlab.ui.control.NumericEditField
52          AxelEditFieldLabel      matlab.ui.control.Label
53          ArmEditField            matlab.ui.control.NumericEditField
54          ArmEditFieldLabel       matlab.ui.control.Label
55          AxelSlider              matlab.ui.control.Slider
56          AxelSliderLabel         matlab.ui.control.Label
57          ArmSlider               matlab.ui.control.Slider
58          ArmSliderLabel          matlab.ui.control.Label
59          BasSlider               matlab.ui.control.Slider
60          BasSliderLabel          matlab.ui.control.Label
61          BasEditField            matlab.ui.control.NumericEditField
62          BasEditFieldLabel       matlab.ui.control.Label
63      end
64
65
66      properties (Access = private)
67          vBas = 90 % Description
68          vArm = 90
69          vAxel = 180
70          vRot = 0
71          vHand = 0
72          vKlo = 200
73
74          a
75          s1
76          s2
77          s3
78          s4
79          s5
80          s6
81          savedBas = []
82          savedArm = []
83          savedAxel = []
84          savedRot = []
85          savedHand = []
86          savedKlo = []
87
88          g = 0
```

```
89      end
90
91      methods (Access = private)
92
93          function results = updateSliders(app)
94              app.BasSlider.Value = app.vBas;
95              app.ArmSlider.Value = app.vArm;
96              app.AxelSlider.Value = app.vAxel;
97              app.RotSlider.Value = app.vRot;
98              app.HandSlider.Value = app.vHand;
99              app.kloSlider.Value = app.vKlo;
100         end
101
102         function results = updateBoxes(app)
103             app.BasEditField.Value = app.vBas;
104             app.ArmEditField.Value = app.vArm;
105             app.AxelEditField.Value = app.vAxel;
106             app.RotEditField.Value = app.vRot;
107             app.HandEditField.Value = app.vHand;
108             app.KloEditField.Value = app.vKlo;
109         end
110     end
111
112
113     % Callbacks that handle component events
114     methods (Access = private)
115
116         % Code that executes after component creation
117         function startupFcn(app)
118             app.a = arduino();
119             app.s1 = servo(app.a,'D4');
120             app.s2 = servo(app.a,'D5');
121             app.s3 = servo(app.a,'D6');
122             app.s4 = servo(app.a,'D7');
123             app.s5 = servo(app.a,'D8');
124             app.s6 = servo(app.a,'D9');
125
126
127
128             writePosition(app.s1, app.vBas/180);
129             writePosition(app.s2, app.vArm/180);
130             writePosition(app.s3, app.vAxel/180);
131             writePosition(app.s4, 0);
132             writePosition(app.s5, 0);
133             writePosition(app.s6, app.vKlo/270);
134
135             app.updateSliders();
136             app.updateBoxes();
137         end
138
139         % Value changed function: BasEditField
140         function BasEditFieldValueChanged(app, event)
141             app.vBas = app.BasEditField.Value;
142         end
```

```
143
144         % Value changed function: ArmEditField
145         function ArmEditFieldValueChanged(app, event)
146             app.vArm = app.ArmEditField.Value;
147         end
148
149         % Value changed function: AxelEditField
150         function AxelEditFieldValueChanged(app, event)
151             app.vAxel = app.AxelEditField.Value;
152         end
153
154         % Button pushed function: MoveButton
155         function MoveButtonPushed(app, event)
156             app.updateSliders();
157
158
159             writePosition(app.s1, app.vBas/180);
160             writePosition(app.s2, app.vArm/180);
161             writePosition(app.s3, app.vAxel/180);
162             writePosition(app.s4, app.vRot/270);
163             writePosition(app.s5, app.vHand/270);
164             writePosition(app.s6, app.vKlo/270);
165         end
166
167         % Value changing function: BasSlider
168         function BasSliderValueChanging(app, event)
169             changingValue = event.Value;
170             app.vBas = changingValue;
171             writePosition(app.s1, app.vBas/180);
172             app.updateBoxes();
173         end
174
175         % Value changing function: ArmSlider
176         function ArmSliderValueChanging(app, event)
177             changingValue = event.Value;
178             app.vArm = changingValue;
179             writePosition(app.s2, app.vArm/180);
180             app.updateBoxes();
181         end
182
183         % Value changing function: AxelSlider
184         function AxelSliderValueChanging(app, event)
185             changingValue = event.Value;
186             app.vAxel = changingValue;
187             writePosition(app.s3, app.vAxel/180);
188             app.updateBoxes();
189         end
190
191         % Button pushed function: SaveButton
192         function SaveButtonPushed(app, event)
193             app.savedBas = [app.savedBas app.vBas];
194             app.savedArm = [app.savedArm app.vArm];
195             app.savedAxel = [app.savedAxel app.vAxel];
196             app.savedRot = [app.savedRot app.vRot];
```

```
197             app.savedHand = [app.savedHand app.vHand];
198             app.savedKlo = [app.savedKlo app.vKlo];
199 
200             app.SavedEditField.Value = app.SavedEditField.Value +
                    1;
201         end
202 
203         % Button pushed function: NextButton
204         function NextButtonPushed(app, event)
205                 idx = app.CurrentEditField.Value;
206                 if idx <= length(app.savedBas)
207                     writePosition(app.s1, app.savedBas(idx)/180);
208                     writePosition(app.s2, app.savedArm(idx)/180);
209                     writePosition(app.s3, app.savedAxel(idx)/180);
210                     writePosition(app.s4, app.savedRot(idx)/270);
211                     writePosition(app.s5, app.savedHand(idx)/270);
212                     writePosition(app.s6, app.savedKlo(idx)/270);
213                     app.updateBoxes();
214                     app.updateSliders();
215                     app.CurrentEditField.Value = app.
                            CurrentEditField.Value + 1;
216                 end
217         end
218 
219         % Value changing function: RotSlider
220         function RotSliderValueChanging(app, event)
221             changingValue = event.Value;
222             app.vRot = changingValue;
223             writePosition(app.s4, app.vRot/270);
224             app.updateBoxes();
225         end
226 
227         % Value changing function: HandSlider
228         function HandSliderValueChanging(app, event)
229             changingValue = event.Value;
230             app.vHand = changingValue;
231             writePosition(app.s5, app.vHand/270);
232             app.updateBoxes();
233         end
234 
235         % Value changing function: kloSlider
236         function kloSliderValueChanging(app, event)
237             changingValue = event.Value;
238             app.vKlo = changingValue;
239             writePosition(app.s6, app.vKlo/270);
240             app.updateBoxes();
241         end
242 
243         % Value changed function: HandEditField
244         function HandEditFieldValueChanged(app, event)
245             app.vHand = app.HandEditField.Value;
246         end
247 
248         % Value changed function: RotEditField
```

```
249         function RotEditFieldValueChanged(app, event)
250             app.vRot = app.RotEditField.Value;
251         end
252 
253         % Value changed function: KloEditField
254         function KloEditFieldValueChanged(app, event)
255             app.vKlo = app.KloEditField.Value;
256         end
257 
258         % Button pushed function: GripToggleButton
259         function GripToggleButtonPushed(app, event)
260             if app.g == 0
261                 app.vKlo = 270;
262                 writePosition(app.s6, app.vKlo/270);
263                 app.g = 1;
264 
265             elseif app.g == 1
266                 app.vKlo = 200;
267                 writePosition(app.s6, app.vKlo/270);
268                 app.g = 0;
269             end
270 
271             app.updateBoxes();
272             app.updateSliders();
273         end
274 
275         % Button pushed function: ResetButton
276         function ResetButtonPushed(app, event)
277             app.CurrentEditField.Value = 1;
278         end
279 
280         % Button pushed function: ClearButton
281         function ClearButtonPushed(app, event)
282             app.savedBas = [];
283             app.savedArm = [];
284             app.savedAxel = [];
285             app.savedRot = [];
286             app.savedHand = [];
287             app.savedKlo = [];
288             app.SavedEditField.Value = 0;
289             app.CurrentEditField.Value = 1;
290         end
291     end
292 
293     % Component initialization
294     methods (Access = private)
295 
296         % Create UIFigure and components
297         function createComponents(app)
298 
299             % Create UIFigure and hide until all components are
                    created
300             app.UIFigure = uifigure('Visible', 'off');
301             app.UIFigure.Position = [100 100 640 480];
```

```
302             app.UIFigure.Name = 'MATLAB App';
303
304             % Create BasEditFieldLabel
305             app.BasEditFieldLabel = uilabel(app.UIFigure);
306             app.BasEditFieldLabel.HorizontalAlignment = 'right';
307             app.BasEditFieldLabel.Position = [24 437 26 22];
308             app.BasEditFieldLabel.Text = 'Bas';
309
310             % Create BasEditField
311             app.BasEditField = uieditfield(app.UIFigure, 'numeric')
                    ;
312             app.BasEditField.Limits = [0 180];
313             app.BasEditField.ValueChangedFcn = createCallbackFcn(
                    app, @BasEditFieldValueChanged, true);
314             app.BasEditField.Position = [65 437 100 22];
315
316             % Create BasSliderLabel
317             app.BasSliderLabel = uilabel(app.UIFigure);
318             app.BasSliderLabel.HorizontalAlignment = 'right';
319             app.BasSliderLabel.Position = [41 190 26 22];
320             app.BasSliderLabel.Text = 'Bas';
321
322             % Create BasSlider
323             app.BasSlider = uislider(app.UIFigure);
324             app.BasSlider.Limits = [0 180];
325             app.BasSlider.ValueChangingFcn = createCallbackFcn(app,
                     @BasSliderValueChanging, true);
326             app.BasSlider.Position = [88 199 150 3];
327
328             % Create ArmSliderLabel
329             app.ArmSliderLabel = uilabel(app.UIFigure);
330             app.ArmSliderLabel.HorizontalAlignment = 'right';
331             app.ArmSliderLabel.Position = [41 126 28 22];
332             app.ArmSliderLabel.Text = 'Arm';
333
334             % Create ArmSlider
335             app.ArmSlider = uislider(app.UIFigure);
336             app.ArmSlider.Limits = [0 180];
337             app.ArmSlider.ValueChangingFcn = createCallbackFcn(app,
                     @ArmSliderValueChanging, true);
338             app.ArmSlider.Position = [90 135 150 3];
339             app.ArmSlider.Value = 90;
340
341             % Create AxelSliderLabel
342             app.AxelSliderLabel = uilabel(app.UIFigure);
343             app.AxelSliderLabel.HorizontalAlignment = 'right';
344             app.AxelSliderLabel.Position = [41 71 29 22];
345             app.AxelSliderLabel.Text = 'Axel';
346
347             % Create AxelSlider
348             app.AxelSlider = uislider(app.UIFigure);
349             app.AxelSlider.Limits = [0 180];
350             app.AxelSlider.ValueChangingFcn = createCallbackFcn(app
                    , @AxelSliderValueChanging, true);
```

```
351             app.AxelSlider.Position = [91 80 150 3];
352             app.AxelSlider.Value = 180;
353 
354             % Create ArmEditFieldLabel
355             app.ArmEditFieldLabel = uilabel(app.UIFigure);
356             app.ArmEditFieldLabel.HorizontalAlignment = 'right';
357             app.ArmEditFieldLabel.Position = [191 437 28 22];
358             app.ArmEditFieldLabel.Text = 'Arm';
359 
360             % Create ArmEditField
361             app.ArmEditField = uieditfield(app.UIFigure, 'numeric')
                    ;
362             app.ArmEditField.Limits = [0 180];
363             app.ArmEditField.ValueChangedFcn = createCallbackFcn(
                    app, @ArmEditFieldValueChanged, true);
364             app.ArmEditField.Position = [234 437 100 22];
365 
366             % Create AxelEditFieldLabel
367             app.AxelEditFieldLabel = uilabel(app.UIFigure);
368             app.AxelEditFieldLabel.HorizontalAlignment = 'right';
369             app.AxelEditFieldLabel.Position = [379 437 29 22];
370             app.AxelEditFieldLabel.Text = 'Axel';
371 
372             % Create AxelEditField
373             app.AxelEditField = uieditfield(app.UIFigure, 'numeric
                    ');
374             app.AxelEditField.Limits = [0 180];
375             app.AxelEditField.ValueChangedFcn = createCallbackFcn(
                    app, @AxelEditFieldValueChanged, true);
376             app.AxelEditField.Position = [423 437 100 22];
377 
378             % Create MoveButton
379             app.MoveButton = uibutton(app.UIFigure, 'push');
380             app.MoveButton.ButtonPushedFcn = createCallbackFcn(app,
                     @MoveButtonPushed, true);
381             app.MoveButton.Position = [65 333 100 22];
382             app.MoveButton.Text = 'Move';
383 
384             % Create SaveButton
385             app.SaveButton = uibutton(app.UIFigure, 'push');
386             app.SaveButton.ButtonPushedFcn = createCallbackFcn(app,
                     @SaveButtonPushed, true);
387             app.SaveButton.Position = [234 333 100 22];
388             app.SaveButton.Text = 'Save';
389 
390             % Create NextButton
391             app.NextButton = uibutton(app.UIFigure, 'push');
392             app.NextButton.ButtonPushedFcn = createCallbackFcn(app,
                     @NextButtonPushed, true);
393             app.NextButton.Position = [383 333 100 22];
394             app.NextButton.Text = 'Next';
395 
396             % Create RotSliderLabel
397             app.RotSliderLabel = uilabel(app.UIFigure);
```

```
398             app.RotSliderLabel.HorizontalAlignment = 'right';
399             app.RotSliderLabel.Position = [305 180 25 22];
400             app.RotSliderLabel.Text = 'Rot';
401 
402             % Create RotSlider
403             app.RotSlider = uislider(app.UIFigure);
404             app.RotSlider.Limits = [0 270];
405             app.RotSlider.ValueChangingFcn = createCallbackFcn(app,
                    @RotSliderValueChanging, true);
406             app.RotSlider.Position = [351 189 150 3];
407 
408             % Create HandSliderLabel
409             app.HandSliderLabel = uilabel(app.UIFigure);
410             app.HandSliderLabel.HorizontalAlignment = 'right';
411             app.HandSliderLabel.Position = [296 126 34 22];
412             app.HandSliderLabel.Text = 'Hand';
413 
414             % Create HandSlider
415             app.HandSlider = uislider(app.UIFigure);
416             app.HandSlider.Limits = [0 270];
417             app.HandSlider.ValueChangingFcn = createCallbackFcn(app
                    , @HandSliderValueChanging, true);
418             app.HandSlider.Position = [351 135 150 3];
419 
420             % Create kloSliderLabel
421             app.kloSliderLabel = uilabel(app.UIFigure);
422             app.kloSliderLabel.HorizontalAlignment = 'right';
423             app.kloSliderLabel.Position = [305 71 25 22];
424             app.kloSliderLabel.Text = 'klo';
425 
426             % Create kloSlider
427             app.kloSlider = uislider(app.UIFigure);
428             app.kloSlider.Limits = [200 270];
429             app.kloSlider.ValueChangingFcn = createCallbackFcn(app,
                    @kloSliderValueChanging, true);
430             app.kloSlider.Position = [351 80 150 3];
431             app.kloSlider.Value = 220;
432 
433             % Create RotEditFieldLabel
434             app.RotEditFieldLabel = uilabel(app.UIFigure);
435             app.RotEditFieldLabel.HorizontalAlignment = 'right';
436             app.RotEditFieldLabel.Position = [25 386 25 22];
437             app.RotEditFieldLabel.Text = 'Rot';
438 
439             % Create RotEditField
440             app.RotEditField = uieditfield(app.UIFigure, 'numeric')
                    ;
441             app.RotEditField.ValueChangedFcn = createCallbackFcn(
                    app, @RotEditFieldValueChanged, true);
442             app.RotEditField.Position = [65 386 100 22];
443 
444             % Create HandEditFieldLabel
445             app.HandEditFieldLabel = uilabel(app.UIFigure);
446             app.HandEditFieldLabel.HorizontalAlignment = 'right';
```

```
447             app.HandEditFieldLabel.Position = [185 386 34 22];
448             app.HandEditFieldLabel.Text = 'Hand';
449 
450             % Create HandEditField
451             app.HandEditField = uieditfield(app.UIFigure, 'numeric
                    ');
452             app.HandEditField.ValueChangedFcn = createCallbackFcn(
                    app, @HandEditFieldValueChanged, true);
453             app.HandEditField.Position = [234 386 100 22];
454 
455             % Create KloEditFieldLabel
456             app.KloEditFieldLabel = uilabel(app.UIFigure);
457             app.KloEditFieldLabel.HorizontalAlignment = 'right';
458             app.KloEditFieldLabel.Position = [383 386 25 22];
459             app.KloEditFieldLabel.Text = 'Klo';
460 
461             % Create KloEditField
462             app.KloEditField = uieditfield(app.UIFigure, 'numeric')
                    ;
463             app.KloEditField.ValueChangedFcn = createCallbackFcn(
                    app, @KloEditFieldValueChanged, true);
464             app.KloEditField.Position = [423 386 100 22];
465 
466             % Create ResetButton
467             app.ResetButton = uibutton(app.UIFigure, 'push');
468             app.ResetButton.ButtonPushedFcn = createCallbackFcn(app
                    , @ResetButtonPushed, true);
469             app.ResetButton.Position = [500 333 100 22];
470             app.ResetButton.Text = 'Reset';
471 
472             % Create GripToggleButton
473             app.GripToggleButton = uibutton(app.UIFigure, 'push');
474             app.GripToggleButton.ButtonPushedFcn =
                    createCallbackFcn(app, @GripToggleButtonPushed, true
                    );
475             app.GripToggleButton.Position = [66 267 100 22];
476             app.GripToggleButton.Text = 'GripToggle';
477 
478             % Create SavedEditFieldLabel
479             app.SavedEditFieldLabel = uilabel(app.UIFigure);
480             app.SavedEditFieldLabel.HorizontalAlignment = 'right';
481             app.SavedEditFieldLabel.Position = [392 288 39 22];
482             app.SavedEditFieldLabel.Text = 'Saved';
483 
484             % Create SavedEditField
485             app.SavedEditField = uieditfield(app.UIFigure, 'numeric
                    ');
486             app.SavedEditField.Editable = 'off';
487             app.SavedEditField.Position = [451 288 23 22];
488 
489             % Create CurrentEditFieldLabel
490             app.CurrentEditFieldLabel = uilabel(app.UIFigure);
491             app.CurrentEditFieldLabel.HorizontalAlignment = 'right
                    ';
```

```
492             app.CurrentEditFieldLabel.Position = [502 288 46 22];
493             app.CurrentEditFieldLabel.Text = 'Current';
494 
495             % Create CurrentEditField
496             app.CurrentEditField = uieditfield(app.UIFigure, '
                    numeric');
497             app.CurrentEditField.Editable = 'off';
498             app.CurrentEditField.Position = [568 288 23 22];
499             app.CurrentEditField.Value = 1;
500 
501             % Create ClearButton
502             app.ClearButton = uibutton(app.UIFigure, 'push');
503             app.ClearButton.ButtonPushedFcn = createCallbackFcn(app
                    , @ClearButtonPushed, true);
504             app.ClearButton.Position = [234 267 100 22];
505             app.ClearButton.Text = 'Clear';
506 
507             % Show the figure after all components are created
508             app.UIFigure.Visible = 'on';
509         end
510     end
511 
512     % App creation and deletion
513     methods (Access = public)
514 
515         % Construct app
516         function app = KEXrobotManualVEK
517 
518             % Create UIFigure and components
519             createComponents(app)
520 
521             % Register the app with App Designer
522             registerApp(app, app.UIFigure)
523 
524             % Execute the startup function
525             runStartupFcn(app, @startupFcn)
526 
527             if nargout == 0
528                 clear app
529             end
530         end
531 
532         % Code that executes before app deletion
533         function delete(app)
534 
535             % Delete UIFigure when app is deleted
536             delete(app.UIFigure)
537         end
538     end
539 end]]>
540             </w:t>
541          </w:r>
542       </w:p>
543    </w:body>
```

544 </w:document>

Appendix C

MATLAB Code Newton-Raphson

Listing C.1. Source Code

```
    <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 appTestArm < 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 Numerical Method control.

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                    matlab.ui.Figure
        StepsizeEditField           matlab.ui.control.
            NumericEditField
        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
```

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

```
83              f = [L1*cos(a) + L2*cos(b) - x2; L1*sin(a) - L2*sin(b)
                    - y2];
84
85              while norm(f) > app.tol && i<app.imax
86                  i = i + 1;
87                  J11 = -L1*sin(a);
88                  J12 = -L2*sin(b);
89                  J21 = L1*cos(a);
90                  J22 = -L2*cos(b);
91                  J = [J11 J12; J21 J22];
92
93                  x = x-J\f;
94                  a = x(1);
95                  b = x(2);
96                  f = [L1*cos(a) + L2*cos(b) - x2; L1*sin(a) - L2*sin
                        (b) - y2];
97              end
98              if i == 10000
99                  app.resn = 'Non reasonable solution'
100             else
101                 app.resn = 'Reasonable solution'
102             end
103
104         end
105
106     end
107
108
109     % Callbacks that handle component events
110     methods (Access = private)
111
112         % Code that executes after component creation
113         function startupFcn(app)
114             axis(app.UIAxes,'square')
115         end
116
117         % Value changed function: DesiredXvalueEditField
118         function DesiredXvalueEditFieldValueChanged(app, event)
119             app.desiredX = app.DesiredXvalueEditField.Value;
120         end
121
122         % Button pushed function: CalculateButton
123         function CalculateButtonPushed(app, event)
124             str = cat(2,'Desired X,Y value is: ',num2str(app.
                    desiredX),' ',num2str(app.desiredY));
125             app.PositionTextArea.Value = str;
126
127             app.x = newtonrap(app,app.startX,app.desiredX,app.
                    desiredY,app.Arm1,app.Arm2);
128
129             app.x = app.x*180/pi;
130
131             str2 = cat(2,'Alpha,Beta is: ',num2str(app.x(1)),',',
                    num2str(app.x(2)),' ',app.resn);
```

```
132             app.AnglesTextArea.Value = str2;
133
134             if app.alphaCurrent > app.x(1)
135 alpha = (app.alphaCurrent:-0.5:app.x(1))*(pi/180);
136 else
137 alpha = (app.alphaCurrent:0.5:app.x(1))*(pi/180);
138 end
139 l = size(alpha);
140 interval = (1/(l(2)-1))*(app.betaCurrent-app.x(2));
141 beta = (app.betaCurrent:-interval:app.x(2))*(pi/180);
142 x0 = zeros(size(alpha));
143 y0 = zeros(size(alpha));
144 x1 = app.Arm1*cos(alpha);
145 y1 = app.Arm1*sin(alpha);
146 x2 = x1 + app.Arm2*cos(beta);
147 y2 = y1 - app.Arm2*sin(beta);
148 for k = 1:length(alpha)
149 plot(app.UIAxes,[x0(k) x1(k)],[y0(k) y1(k)],'ro-','LineWidth',2)
150 hold(app.UIAxes, 'on')
151 plot(app.UIAxes,[x2(1) x2(end)],[y2(1) y2(end)],'b--','MarkerSize
    ',10)
152 plot(app.UIAxes,x2(1),y2(1),'bx','MarkerSize',10)
153 plot(app.UIAxes,x2(end),y2(end),'bo','MarkerSize',10)
154 plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-','LineWidth',2)
155 hold(app.UIAxes, 'off')
156 %app.UIAxes.Xlim = [-0.5 0.5];
157 %app.UIAxes.Ylim = [-0.5 0.5];
158 %hold(app.UIAxes);
159 %plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-','LineWidth',2)
160 pause(0.018)
161 end
162 app.alphaCurrent = app.x(1);
163 app.betaCurrent = app.x(2);
164 app.AlphastartEditField.Value = app.x(1);
165 app.BetastartEditField.Value = app.x(2);
166 app.AlphaEditField.Value = app.x(1);
167 app.BetaEditField.Value = app.x(2);
168 end
169 % Value changed function: DesiredYvalueEditField
170 function DesiredYvalueEditFieldValueChanged(app, event)
171 app.desiredY = app.DesiredYvalueEditField.Value;
172 end
173 % Value changed function: AlphastartEditField
174 function AlphastartEditFieldValueChanged(app, event)
175 app.alphaStart = app.AlphastartEditField.Value;
176 app.alphaCurrent = app.AlphastartEditField.Value;
177 app.AlphaEditField.Value = app.AlphastartEditField.Value;
178 end
179 % Value changed function: BetastartEditField
180 function BetastartEditFieldValueChanged(app, event)
181 app.betaStart = app.BetastartEditField.Value;
182 app.betaCurrent = app.BetastartEditField.Value;
183 app.BetaEditField.Value = app.BetastartEditField.Value;
184 end
```

```
185 % Button pushed function: MoveButton
186 function MoveButtonPushed(app, event)
187 app.x = [app.alphaMove; app.betaMove];
188 str2 = cat(2,'Alpha,Beta is: ',num2str(app.x(1)),',',num2str(app.x
        (2)),' ',app.resn);
189 app.AnglesTextArea.Value = str2;
190 if app.alphaCurrent > app.x(1)
191 alpha = (app.alphaCurrent:-0.5:app.x(1))*(pi/180);
192 else
193 alpha = (app.alphaCurrent:0.5:app.x(1))*(pi/180);
194 end
195 l = size(alpha);
196 if (app.betaCurrent - app.x(2)) == 0
197 app.x(2) = app.x(2) - 0.01;
198 end
199 interval = (1/(l(2)-1))*(app.betaCurrent-app.x(2));
200 beta = (app.betaCurrent:-interval:app.x(2))*(pi/180);
201 x0 = zeros(size(alpha));
202 y0 = zeros(size(alpha));
203 x1 = app.Arm1*cos(alpha);
204 y1 = app.Arm1*sin(alpha);
205 x2 = x1 + app.Arm2*cos(beta);
206 y2 = y1 - app.Arm2*sin(beta);
207 for k = 1:length(alpha)
208  plot(app.UIAxes,[x0(k) x1(k)],[y0(k) y1(k)],'ro-','LineWidth',2)
209 hold(app.UIAxes, 'on')
210 plot(app.UIAxes,[x2(1) x2(end)],[y2(1) y2(end)],'b--','MarkerSize
        ',10)
211 plot(app.UIAxes,x2(1),y2(1),'bx','MarkerSize',10)
212 plot(app.UIAxes,x2(end),y2(end),'bo','MarkerSize',10)
213 plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-','LineWidth',2)
214 hold(app.UIAxes, 'off')
215 %app.UIAxes.Xlim = [-0.5 0.5];
216 %app.UIAxes.Ylim = [-0.5 0.5];
217 %hold(app.UIAxes);
218 %plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-','LineWidth',2)
219 pause(0.018)
220 end
221 app.alphaCurrent = app.x(1);
222 app.betaCurrent = app.x(2);
223 app.AlphastartEditField.Value = app.x(1);
224 app.BetastartEditField.Value = app.x(2);
225 end
226 % Value changed function: AlphaEditField
227 function AlphaEditFieldValueChanged(app, event)
228 app.alphaMove = app.AlphaEditField.Value;
229 end
230 % Value changed function: BetaEditField
231 function BetaEditFieldValueChanged(app, event)
232 app.betaMove = app.BetaEditField.Value;
233 end
234 % Button pushed function: XButton
235 function XButtonPushed(app, event)
236 app.desiredX = app.desiredX + app.stepsize;
```

```
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;
```

```
285 app.x = newtonrap(app,app.startX,app.desiredX,app.desiredY,app.Arm1
      ,app.Arm2);
286
287 app.x = app.x*180/pi;
288 str2 = cat(2,'Alpha,Beta is: ',num2str(app.x(1)),',',num2str(app.x
      (2)),' ',app.resn);
289 app.AnglesTextArea.Value = str2;
290 if app.alphaCurrent > app.x(1)
291 alpha = (app.alphaCurrent:-0.5:app.x(1))*(pi/180);
292 else
293 alpha = (app.alphaCurrent:0.5:app.x(1))*(pi/180);
294 end
295 l = size(alpha);
296 interval = (1/(l(2)-1))*(app.betaCurrent-app.x(2));
297 beta = (app.betaCurrent:-interval:app.x(2))*(pi/180);
298 x0 = zeros(size(alpha));
299 y0 = zeros(size(alpha));
300 x1 = app.Arm1*cos(alpha);
301 y1 = app.Arm1*sin(alpha);
302 x2 = x1 + app.Arm2*cos(beta);
303 y2 = y1 - app.Arm2*sin(beta);
304 for k = 1:length(alpha)
305 plot(app.UIAxes,[x0(k) x1(k)],[y0(k) y1(k)],'ro-','LineWidth',2)
306 hold(app.UIAxes, 'on')
307 plot(app.UIAxes,[x2(1) x2(end)],[y2(1) y2(end)],'b--','MarkerSize
      ',10)
308 plot(app.UIAxes,x2(1),y2(1),'bx','MarkerSize',10)
309 plot(app.UIAxes,x2(end),y2(end),'bo','MarkerSize',10)
310 plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-','LineWidth',2)
311 hold(app.UIAxes, 'off')
312 %app.UIAxes.Xlim = [-0.5 0.5];
313 %app.UIAxes.Ylim = [-0.5 0.5];
314 %hold(app.UIAxes);
315 %plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-','LineWidth',2)
316 pause(0.018)
317 end
318 app.alphaCurrent = app.x(1);
319 app.betaCurrent = app.x(2);
320 app.AlphastartEditField.Value = app.x(1);
321 app.BetastartEditField.Value = app.x(2);
322 app.AlphaEditField.Value = app.x(1);
323 app.BetaEditField.Value = app.x(2);
324 end
325 % Value changed function: StepsizeEditField
326 function StepsizeEditFieldValueChanged(app, event)
327 app.stepsize = app.StepsizeEditField.Value;
328 end
329 % Button pushed function: YButton
330 function YButtonPushed(app, event)
331 app.desiredY = app.desiredY + app.stepsize;
332 app.DesiredYvalueEditField.Value = app.desiredY;
333 str = cat(2,'Desired X,Y value is: ',num2str(app.desiredX),' ',
      num2str(app.desiredY));
334 app.PositionTextArea.Value = str;
```

```
335 app.x = newtonrap(app,app.startX,app.desiredX,app.desiredY,app.Arm1
        ,app.Arm2);
336 app.x = app.x*180/pi;
337 str2 = cat(2,'Alpha,Beta is: ',num2str(app.x(1)),',',num2str(app.x
        (2)),' ',app.resn);
338 app.AnglesTextArea.Value = str2;
339 if app.alphaCurrent > app.x(1)
340 alpha = (app.alphaCurrent:-0.5:app.x(1))*(pi/180);
341 else
342 alpha = (app.alphaCurrent:0.5:app.x(1))*(pi/180);
343 end
344 l = size(alpha);
345 interval = (1/(l(2)-1))*(app.betaCurrent-app.x(2));
346 beta = (app.betaCurrent:-interval:app.x(2))*(pi/180);
347 x0 = zeros(size(alpha));
348 y0 = zeros(size(alpha));
349 x1 = app.Arm1*cos(alpha);
350 y1 = app.Arm1*sin(alpha);
351 x2 = x1 + app.Arm2*cos(beta);
352 y2 = y1 - app.Arm2*sin(beta);
353 for k = 1:length(alpha)
354 plot(app.UIAxes,[x0(k) x1(k)],[y0(k) y1(k)],'ro-','LineWidth',2)
355 hold(app.UIAxes, 'on')
356 plot(app.UIAxes,[x2(1) x2(end)],[y2(1) y2(end)],'b--','MarkerSize
        ',10)
357 plot(app.UIAxes,x2(1),y2(1),'bx','MarkerSize',10)
358 plot(app.UIAxes,x2(end),y2(end),'bo','MarkerSize',10)
359 plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-','LineWidth',2)
360 hold(app.UIAxes, 'off')
361 %app.UIAxes.Xlim = [-0.5 0.5];
362 %app.UIAxes.Ylim = [-0.5 0.5];
363 %hold(app.UIAxes);
364  %plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-','LineWidth',2)
365 pause(0.018)
366 end
367 app.alphaCurrent = app.x(1);
368 app.betaCurrent = app.x(2);
369 app.AlphastartEditField.Value = app.x(1);
370 app.BetastartEditField.Value = app.x(2);
371 app.AlphaEditField.Value = app.x(1);
372 app.BetaEditField.Value = app.x(2);
373 end
374 % Button pushed function: YButton_2
375 function YButton_2Pushed(app, event)
376 app.desiredY = app.desiredY - app.stepsize;
377 app.DesiredYvalueEditField.Value = app.desiredY;
378 str = cat(2,'Desired X,Y value is: ',num2str(app.desiredX),' ',
        num2str(app.desiredY));
379 app.PositionTextArea.Value = str;
380 app.x = newtonrap(app,app.startX,app.desiredX,app.desiredY,app.Arm1
        ,app.Arm2);
381 app.x = app.x*180/pi;
382 str2 = cat(2,'Alpha,Beta is: ',num2str(app.x(1)),',',num2str(app.x
        (2)),' ',app.resn);
```

```
383 app.AnglesTextArea.Value = str2;
384 if app.alphaCurrent > app.x(1)
385 alpha = (app.alphaCurrent:-0.5:app.x(1))*(pi/180);
386 else
387 alpha = (app.alphaCurrent:0.5:app.x(1))*(pi/180);
388 end
389 l = size(alpha);
390 interval = (1/(l(2)-1))*(app.betaCurrent-app.x(2));
391 beta = (app.betaCurrent:-interval:app.x(2))*(pi/180);
392 x0 = zeros(size(alpha));
393 y0 = zeros(size(alpha));
394 x1 = app.Arm1*cos(alpha);
395 y1 = app.Arm1*sin(alpha);
396 x2 = x1 + app.Arm2*cos(beta);
397 y2 = y1 - app.Arm2*sin(beta);
398 for k = 1:length(alpha)
399 plot(app.UIAxes,[x0(k) x1(k)],[y0(k) y1(k)],'ro-','LineWidth',2)
400 hold(app.UIAxes, 'on')
401 plot(app.UIAxes,[x2(1) x2(end)],[y2(1) y2(end)],'b--','MarkerSize
    ',10)
402 plot(app.UIAxes,x2(1),y2(1),'bx','MarkerSize',10)
403 plot(app.UIAxes,x2(end),y2(end),'bo','MarkerSize',10)
404 plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-','LineWidth',2)
405 hold(app.UIAxes, 'off')
406 %app.UIAxes.Xlim = [-0.5 0.5];
407 %app.UIAxes.Ylim = [-0.5 0.5];
408 %hold(app.UIAxes);
409 %plot(app.UIAxes,[x1(k) x2(k)],[y1(k) y2(k)],'ro-','LineWidth',2)
410 pause(0.018)
411 end
412 app.alphaCurrent = app.x(1);
413 app.betaCurrent = app.x(2);
414 app.AlphastartEditField.Value = app.x(1);
415 app.BetastartEditField.Value = app.x(2);
416 app.AlphaEditField.Value = app.x(1);
417 app.BetaEditField.Value = app.x(2);
418 end
419 end
420 % Component initialization
421 methods (Access = private)
422 % Create UIFigure and components
423 function createComponents(app)
424 % Create UIFigure and hide until all components are created
425 app.UIFigure = uifigure('Visible', 'off');
426 app.UIFigure.Position = [100 100 640 480];
427 app.UIFigure.Name = 'UI Figure';
428 % Create UIAxes
429 app.UIAxes = uiaxes(app.UIFigure);
430 title(app.UIAxes, 'Title')
431 xlabel(app.UIAxes, 'X')
432 ylabel(app.UIAxes, 'Y')
433 app.UIAxes.CameraPosition = [0 0 9.16025403784439];
434 app.UIAxes.CameraTarget = [0 0 0.5];
435 app.UIAxes.CameraUpVector = [0 1 0];
```

```
436 app.UIAxes.DataAspectRatio = [1 1 1];
437 app.UIAxes.PlotBoxAspectRatio = [1 1 1];
438 app.UIAxes.XLim = [-0.5 0.5];
439 app.UIAxes.YLim = [-0.5 0.5];
440 app.UIAxes.ZLim = [0 1];
441 app.UIAxes.CLim = [0 1];
442 app.UIAxes.XColor = [0.15 0.15 0.15];
443 app.UIAxes.XTick = [-0.5 0 0.5];
444 app.UIAxes.YColor = [0.15 0.15 0.15];
445 app.UIAxes.YTick = [-0.5 0 0.5];
446 app.UIAxes.ZColor = [0.15 0.15 0.15];
447 app.UIAxes.ZTick = [0 0.5 1];
448 app.UIAxes.GridColor = [0.15 0.15 0.15];
449 app.UIAxes.MinorGridColor = [0.1 0.1 0.1];
450 app.UIAxes.Position = [13 23 356 231];
451 % Create DesiredXvalueEditFieldLabel
452 app.DesiredXvalueEditFieldLabel = uilabel(app.UIFigure);
453 app.DesiredXvalueEditFieldLabel.HorizontalAlignment = 'right';
454 app.DesiredXvalueEditFieldLabel.VerticalAlignment = 'top';
455 app.DesiredXvalueEditFieldLabel.Position = [25 399 89 15];
456 app.DesiredXvalueEditFieldLabel.Text = 'Desired X value';
457 % Create DesiredXvalueEditField
458 app.DesiredXvalueEditField = uieditfield(app.UIFigure, 'numeric');
459 app.DesiredXvalueEditField.ValueChangedFcn = createCallbackFcn(app,
        @DesiredXvalueEditFieldValueChanged, true);
460 app.DesiredXvalueEditField.Position = [129 395 100 22];
461 % Create DesiredYvalueEditFieldLabel
462 app.DesiredYvalueEditFieldLabel = uilabel(app.UIFigure);
463 app.DesiredYvalueEditFieldLabel.HorizontalAlignment = 'right';
464 app.DesiredYvalueEditFieldLabel.VerticalAlignment = 'top';
465 app.DesiredYvalueEditFieldLabel.Position = [251 399 89 15];
466 app.DesiredYvalueEditFieldLabel.Text = 'Desired Y value';
467 % Create DesiredYvalueEditField
468 app.DesiredYvalueEditField = uieditfield(app.UIFigure, 'numeric');
469 app.DesiredYvalueEditField.ValueChangedFcn = createCallbackFcn(app,
        @DesiredYvalueEditFieldValueChanged, true);
470 app.DesiredYvalueEditField.Position = [355 395 100 22];
471 % Create AlphastartEditFieldLabel
472 app.AlphastartEditFieldLabel = uilabel(app.UIFigure);
473 app.AlphastartEditFieldLabel.HorizontalAlignment = 'right';
474 app.AlphastartEditFieldLabel.VerticalAlignment = 'top';
475 app.AlphastartEditFieldLabel.Position = [50 435 64 15];
476 app.AlphastartEditFieldLabel.Text = 'Alpha start';
477 % Create AlphastartEditField
478 app.AlphastartEditField = uieditfield(app.UIFigure, 'numeric');
479 app.AlphastartEditField.ValueChangedFcn = createCallbackFcn(app,
       @AlphastartEditFieldValueChanged, true);
480 app.AlphastartEditField.Position = [129 431 100 22];
481 % Create BetastartEditFieldLabel
482 app.BetastartEditFieldLabel = uilabel(app.UIFigure);
483 app.BetastartEditFieldLabel.HorizontalAlignment = 'right';
484 app.BetastartEditFieldLabel.VerticalAlignment = 'top';
485 app.BetastartEditFieldLabel.Position = [282 432 58 15];
486 app.BetastartEditFieldLabel.Text = 'Beta start';
```

```
487 % Create BetastartEditField
488 app.BetastartEditField = uieditfield(app.UIFigure, 'numeric');
489 app.BetastartEditField.ValueChangedFcn = createCallbackFcn(app,
        @BetastartEditFieldValueChanged, true);
490 app.BetastartEditField.Position = [355 428 100 22];
491 % Create PositionTextAreaLabel
492 app.PositionTextAreaLabel = uilabel(app.UIFigure);
493 app.PositionTextAreaLabel.HorizontalAlignment = 'right';
494 app.PositionTextAreaLabel.VerticalAlignment = 'top';
495 app.PositionTextAreaLabel.Position = [30 305 55 22];
496 app.PositionTextAreaLabel.Text = 'Position';
497 % Create PositionTextArea
498 app.PositionTextArea = uitextarea(app.UIFigure);
499 app.PositionTextArea.Editable = 'off';
500 app.PositionTextArea.Position = [100 269 150 60];
501 % Create CalculateButton
502 app.CalculateButton = uibutton(app.UIFigure, 'push');
503 app.CalculateButton.ButtonPushedFcn = createCallbackFcn(app,
        @CalculateButtonPushed, true);
504 app.CalculateButton.Position = [355 354 100 22];
505 app.CalculateButton.Text = 'Calculate';
506 % Create AnglesTextAreaLabel
507 app.AnglesTextAreaLabel = uilabel(app.UIFigure);
508 app.AnglesTextAreaLabel.HorizontalAlignment = 'right';
509 app.AnglesTextAreaLabel.VerticalAlignment = 'top';
510 app.AnglesTextAreaLabel.Position = [282 307 62 22];
511 app.AnglesTextAreaLabel.Text = 'Angles';
512 % Create AnglesTextArea
513 app.AnglesTextArea = uitextarea(app.UIFigure);
514 app.AnglesTextArea.Position = [359 271 150 60];
515 % Create MoveButton
516 app.MoveButton = uibutton(app.UIFigure, 'push');
517 app.MoveButton.ButtonPushedFcn = createCallbackFcn(app,
        @MoveButtonPushed, true);
518 app.MoveButton.Position = [429 174 100 22];
519 app.MoveButton.Text = 'Move';
520 % Create AlphaEditFieldLabel
521 app.AlphaEditFieldLabel = uilabel(app.UIFigure);
522 app.AlphaEditFieldLabel.HorizontalAlignment = 'right';
523 app.AlphaEditFieldLabel.VerticalAlignment = 'top';
524 app.AlphaEditFieldLabel.Position = [332 221 36 15];
525 app.AlphaEditFieldLabel.Text = 'Alpha';
526 % Create AlphaEditField
527 app.AlphaEditField = uieditfield(app.UIFigure, 'numeric');
528 app.AlphaEditField.ValueChangedFcn = createCallbackFcn(app,
        @AlphaEditFieldValueChanged, true);
529 app.AlphaEditField.Position = [383 217 76 22];
530 % Create BetaEditFieldLabel
531 app.BetaEditFieldLabel = uilabel(app.UIFigure);
532 app.BetaEditFieldLabel.HorizontalAlignment = 'right';
533 app.BetaEditFieldLabel.VerticalAlignment = 'top';
534 app.BetaEditFieldLabel.Position = [483 221 30 15];
535 app.BetaEditFieldLabel.Text = 'Beta';
536 % Create BetaEditField
```

```
537 app.BetaEditField = uieditfield(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.ButtonPushedFcn = createCallbackFcn(app, @XButtonPushed
        , 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.ButtonPushedFcn = createCallbackFcn(app,
        @XButton_2Pushed, 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.ButtonPushedFcn = createCallbackFcn(app, @YButtonPushed
        , 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.ButtonPushedFcn = createCallbackFcn(app,
        @YButton_2Pushed, true);
558 app.YButton_2.Position = [450 57 34 22];
559 app.YButton_2.Text = 'Y-';
560 % Create StepsizeEditFieldLabel
561 app.StepsizeEditFieldLabel = uilabel(app.UIFigure);
562 app.StepsizeEditFieldLabel.HorizontalAlignment = 'right';
563 app.StepsizeEditFieldLabel.Position = [339 78 51 22];
564 app.StepsizeEditFieldLabel.Text = 'Stepsize';
565 % Create StepsizeEditField
566 app.StepsizeEditField = uieditfield(app.UIFigure, 'numeric');
567 app.StepsizeEditField.ValueChangedFcn = createCallbackFcn(app,
        @StepsizeEditFieldValueChanged, true);
568 app.StepsizeEditField.Position = [398 78 32 22];
569 app.StepsizeEditField.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
```

```
585 clear app
586 end
587 end
588 % Code that executes before app deletion
589 function delete(app)
590 % Delete UIFigure when app is deleted
591 delete(app.UIFigure)
592 end
593 end
594 end]]>
595                 </w:t>
596             </w:r>
597         </w:p>
598     </w:body>
599 </w:document>
```

Appendix D

MATLAB Code Fuzzylogic

Listing D.1. Source Code

```
1    <w:document
2    xmlns:w="http://schemas.openxmlformats.org/wordprocessingml
         /2006/main">
3    <w:body>
4        <w:p>
5            <w:pPr>
6                <w:pStyle w:val="code"/>
7            </w:pPr>
8            <w:r>
9                <w:t>
10                   <![CDATA[classdef plotTestSIMONLY < matlab.apps.
                         AppBase
11
12 %KTH Royal Institute of Technology.
13 %Bachelor's Thesis in Mechatronics.
14 %Multipurpose robot arm.
15 %Multifunktions robotarm.
16 %
17 %Authors: Fahim Pirmohamed (fahimp@kth.se),
18 %Alexander Aronsson (alearo@kth.se).
19 %
20 %Course code: MF133X.
21 %Examiner: Nihad Subasic.
22 %TRITA: 2021:34.
23 %
24 %File for MATLAB ANFIS Control.
25
26
27     % Properties that correspond to app components
28     properties (Access = public)
29         UIFigure                matlab.ui.Figure
30         MoveButton              matlab.ui.control.Button
31         YEditField              matlab.ui.control.NumericEditField
32         YEditFieldLabel         matlab.ui.control.Label
33         XEditField              matlab.ui.control.NumericEditField
34         XEditFieldLabel         matlab.ui.control.Label
```

```
35          X_stepDownButton        matlab.ui.control.Button
36          X_stepUpButton          matlab.ui.control.Button
37          SolutionTextArea        matlab.ui.control.TextArea
38          SolutionTextAreaLabel   matlab.ui.control.Label
39          YSliderLabel            matlab.ui.control.Label
40          YSlider                 matlab.ui.control.Slider
41          XSlider                 matlab.ui.control.Slider
42          XSliderLabel            matlab.ui.control.Label
43          BetaSlider              matlab.ui.control.Slider
44          BetaSliderLabel         matlab.ui.control.Label
45          XYTextArea              matlab.ui.control.TextArea
46          XYTextAreaLabel         matlab.ui.control.Label
47          AlphaSlider             matlab.ui.control.Slider
48          AlphaSliderLabel        matlab.ui.control.Label
49          UIAxes                  matlab.ui.control.UIAxes
50      end
51
52
53      properties (Access = private)
54          alpha = 120 % Description
55          beta = 120
56          Arm1 = 0.5
57          Arm2 = 0.3
58
59          startX = [50*pi/180; 30*pi/180]
60          tol = 1e-12 % Description
61          imax = 10000 % Description
62          x = [50*pi/180; 30*pi/180]
63
64          cPos = 0
65
66          anfis1
67          anfis2
68
69          resn
70
71          a
72          s1
73          s2
74          s3
75          s4
76          s5
77          s6
78
79          vBas = 90 % Description
80          vArm = 90
81          vAxel = 180
82          vRot = 0
83          vHand = 0
84          vKlo = 200
85
86          xx
87          yy
88      end
```

```
89
90      methods (Access = private)
91
92          function results = plotFig(app)
93              x0 = 0;
94              y0 = 0;
95
96              x1 = app.Arm1*cos(app.alpha*pi/180);
97              y1 = app.Arm1*sin(app.alpha*pi/180);
98
99              bo =app.beta;
100
101             x2 = x1 + app.Arm2*cos((app.alpha + bo)*pi/180);
102             y2 = y1 + app.Arm2*sin((app.alpha + bo)*pi/180);
103
104             plot(app.UIAxes,[x0 x1],[y0 y1],'ro-','LineWidth',2)
105             hold(app.UIAxes, "on")
106             plot(app.UIAxes, [x1 x2],[y1 y2],'ro-','LineWidth',2)
107             hold(app.UIAxes, "off")
108         end
109
110
111
112         function pos = calcPos(app)
113
114             x1 = app.Arm1*cos(app.alpha*pi/180);
115             y1 = app.Arm1*sin(app.alpha*pi/180);
116
117             bo =app.beta;
118
119             x2 = x1 + app.Arm2*cos((app.alpha + bo)*pi/180);
120             y2 = y1 + app.Arm2*sin((app.alpha + bo)*pi/180);
121
122             pos = [x2,y2];
123         end
124
125         function results = anfisSetup(app)
126             l1 = app.Arm1; % length of first arm
127             l2 = app.Arm2; % length of second arm
128
129             theta1 = 0:0.1:pi/2; % all possible theta1 values
130             theta2 = 0:0.1:pi; % all possible theta2 values
131
132             [THETA1,THETA2] = meshgrid(theta1,theta2); % generate a
                    grid of theta1 and theta2 values
133
134             BO =THETA2;
135
136             X = l1 * cos(THETA1) + l2 * cos(THETA1 + BO); % compute
                    x coordinates
137             Y = l1 * sin(THETA1) + l2 * sin(THETA1 + BO); % compute
                    y coordinates
138
```

```
139             data1 = [X(:) Y(:) THETA1(:)]; % create x-y-theta1
                    dataset
140             data2 = [X(:) Y(:) THETA2(:)]; % create x-y-theta2
                    dataset
141
142             opt = anfisOptions;
143             opt.InitialFIS = 7;
144             opt.EpochNumber = 150;
145             opt.DisplayANFISInformation = 0;
146             opt.DisplayErrorValues = 0;
147             opt.DisplayStepSize = 0;
148             opt.DisplayFinalResults = 0;
149
150             app.SolutionTextArea.Value = '--> Training first ANFIS
                    network.';
151 app.anfis1 = anfis(data1,opt);
152 app.SolutionTextArea.Value = '--> Training second ANFIS network.';
153 opt.InitialFIS = 6;
154 app.anfis2 = anfis(data2,opt);
155 app.SolutionTextArea.Value = 'Training done';
156 end
157 end
158 % Callbacks that handle component events
159 methods (Access = private)
160 % Code that executes after component creation
161 function startupFcn(app)
162 axis(app.UIAxes,'square')
163 grid(app.UIAxes, 'on')
164 app.plotFig();
165 app.cPos = app.calcPos();
166 app.XYTextArea.Value = num2str(app.cPos);
167 app.XSlider.Value = app.cPos(1);
168 app.YSlider.Value = app.cPos(2);
169 app.anfisSetup();
170 app.a = arduino();
171 app.s1 = servo(app.a,'D4');
172 app.s2 = servo(app.a,'D5');
173 app.s3 = servo(app.a,'D6');
174 app.s4 = servo(app.a,'D7');
175 app.s5 = servo(app.a,'D8');
176 app.s6 = servo(app.a,'D9');
177 writePosition(app.s1, app.vBas/180);
178 writePosition(app.s2, app.vArm/180);
179 writePosition(app.s3, app.vAxel/180);
180 writePosition(app.s4, 0);
181 writePosition(app.s5, 0);
182 writePosition(app.s6, app.vKlo/270);
183 end
184 % Value changing function: AlphaSlider
185 function AlphaSliderValueChanging(app, event)
186 changingValue = event.Value;
187 app.alpha = changingValue;
188 app.plotFig();
189 app.cPos = app.calcPos();
```

```
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)
        ,app.Arm1,app.Arm2);
240 app.alpha = newAngles(1);
241 app.beta = newAngles(2);
242 app.cPos = app.calcPosOld;
```

```
243 app.SolutionTextArea.Value = app.resn;
244 app.plotFigOld();
245 end
246 % Button pushed function: X_stepDownButton
247 function X_stepDownButtonPushed(app, event)
248 xNew = app.cPos(1) - 0.01;
249 newAngles = app.newtonrapOld([app.alpha; app.beta],xNew,app.cPos(2)
        ,app.Arm1,app.Arm2);
250 app.alpha = newAngles(1);
251 app.beta = newAngles(2);
252 app.cPos = app.calcPosOld;
253 app.SolutionTextArea.Value = app.resn;
254 app.plotFigOld();
255 end
256 % Value changed function: XEditField
257 function XEditFieldValueChanged(app, event)
258 app.xx = app.XEditField.Value;
259 end
260 % Value changed function: YEditField
261 function YEditFieldValueChanged(app, event)
262 app.yy = app.YEditField.Value;
263 end
264 % Button pushed function: MoveButton
265 function MoveButtonPushed(app, event)
266 xy = [app.xx;app.yy];
267 app.alpha = evalfis(app.anfis1,xy)*180/pi;
268 app.beta = evalfis(app.anfis2,xy)*180/pi;
269 app.cPos = app.calcPos();
270 app.plotFig();
271 app.AlphaSlider.Value = app.alpha;
272 app.BetaSlider.Value = app.beta;
273 end
274 end
275 % Component initialization
276 methods (Access = private)
277 % Create UIFigure and components
278 function createComponents(app)
279 % Create UIFigure and hide until all components are created
280 app.UIFigure = uifigure('Visible', 'off');
281 app.UIFigure.Position = [100 100 640 480];
282 app.UIFigure.Name = 'MATLAB App';
283 % Create UIAxes
284 app.UIAxes = uiaxes(app.UIFigure);
285 title(app.UIAxes, 'Title')
286 xlabel(app.UIAxes, 'X')
287 ylabel(app.UIAxes, 'Y')
288 zlabel(app.UIAxes, 'Z')
289 app.UIAxes.XLim = [-1 1];
290 app.UIAxes.YLim = [-1 1];
291 app.UIAxes.Position = [26 19 349 322];
292 % Create AlphaSliderLabel
293 app.AlphaSliderLabel = uilabel(app.UIFigure);
294 app.AlphaSliderLabel.HorizontalAlignment = 'right';
295 app.AlphaSliderLabel.Position = [400 422 36 22];
```

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

```
346 % Create SolutionTextArea
347 app.SolutionTextArea = uitextarea(app.UIFigure);
348 app.SolutionTextArea.Position = [196 352 150 60];
349 % Create X_stepUpButton
350 app.X_stepUpButton = uibutton(app.UIFigure, 'push');
351 app.X_stepUpButton.ButtonPushedFcn = createCallbackFcn(app,
        @X_stepUpButtonPushed, true);
352 app.X_stepUpButton.Position = [269 422 77 22];
353 app.X_stepUpButton.Text = 'X_stepUp';
354 % Create X_stepDownButton
355 app.X_stepDownButton = uibutton(app.UIFigure, 'push');
356 app.X_stepDownButton.ButtonPushedFcn = createCallbackFcn(app,
        @X_stepDownButtonPushed, true);
357 app.X_stepDownButton.Position = [173 422 83 22];
358 app.X_stepDownButton.Text = 'X_stepDown';
359 % Create XEditFieldLabel
360 app.XEditFieldLabel = uilabel(app.UIFigure);
361 app.XEditFieldLabel.HorizontalAlignment = 'right';
362 app.XEditFieldLabel.Position = [10 431 25 22];
363 app.XEditFieldLabel.Text = 'X';
364 % Create XEditField
365 app.XEditField = uieditfield(app.UIFigure, 'numeric');
366 app.XEditField.ValueChangedFcn = createCallbackFcn(app,
        @XEditFieldValueChanged, true);
367 app.XEditField.Position = [50 431 29 22];
368 % Create YEditFieldLabel
369 app.YEditFieldLabel = uilabel(app.UIFigure);
370 app.YEditFieldLabel.HorizontalAlignment = 'right';
371 app.YEditFieldLabel.Position = [78 431 25 22];
372 app.YEditFieldLabel.Text = 'Y';
373 % Create YEditField
374 app.YEditField = uieditfield(app.UIFigure, 'numeric');
375 app.YEditField.ValueChangedFcn = createCallbackFcn(app,
        @YEditFieldValueChanged, true);
376 app.YEditField.Position = [118 431 29 22];
377 % Create MoveButton
378 app.MoveButton = uibutton(app.UIFigure, 'push');
379 app.MoveButton.ButtonPushedFcn = createCallbackFcn(app,
        @MoveButtonPushed, true);
380 app.MoveButton.Position = [26 388 100 22];
381 app.MoveButton.Text = 'Move';
382 % Show the figure after all components are created
383 app.UIFigure.Visible = 'on';
384 end
385 end
386 % App creation and deletion
387 methods (Access = public)
388 % Construct app
389 function app = plotTestSIMONLY
390 % Create UIFigure and components
391 createComponents(app)
392 % Register the app with App Designer
393 registerApp(app, app.UIFigure)
394 % 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
end]]>
                </w:t>
            </w:r>
        </w:p>
    </w:body>
</w:document>
```

TRITA ITM 2021:34