Robotic Manipulation Report

Group A1 (Group Baymax)
Gian-Luca Fenocchi 01869614, Aishwarya Anand 01852814,
Arthika Sivathasan 01921568

March 2023

Contents

1 Task One - Modelling the Robot

1.1 Forward Kinematics

1.1.1 Frame Assignment and DH Table

Within the project, the following nomenclature and frame assignment was utilised for the robot:

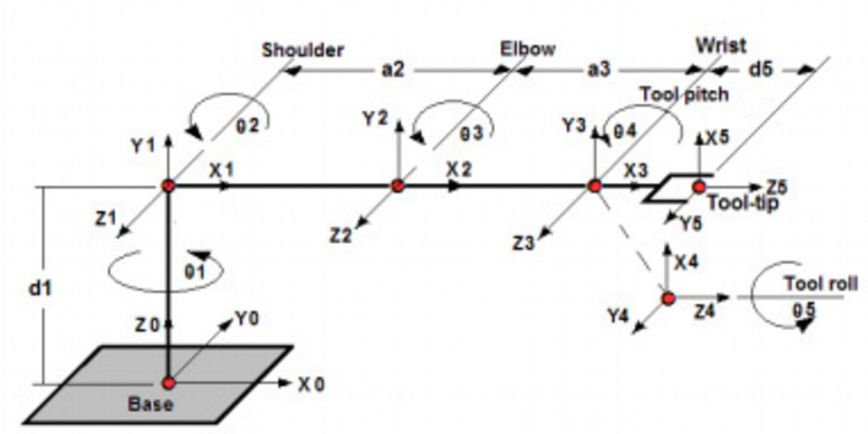

Figure 1: Shows the frame assignment of the robot and the nomenclature used to refer to the different joints. It can be seen that the Z axes always point outwards and X points along the link. Servos 2-4 move in the Z-XY plane while Servo 1 sets the position of the robot in the XY plane.

.

The DH-table was created by analysing the documentation of the Open Manipulator X. The DH-table was completed as shown to the left, where the different theta angles are variables which are used to set the position and orientation of the robot.

Parameters	α	**a**	**d**	θ
L1	0	0	0.077	θ_1
L2	90	0	0	θ_2
L3	0	0.130	0	θ_3
L4	0	0.124	0	θ_4
L5	0	0.126	0	θ_5

Table 1: DH Table

When considering the frame convention used in our kinematics, it was useful to consider how the inverse kinematics could be broken down into sub-problems. By considering this idea, it was imperative that all the frames seen in the upper arm (Servo 2 - 4), all took positive Z in the same way. This orientation was chosen since it greatly simplified the DH-table and allowed us to observe the upper arm in a 2D plane. An alternate orientation that could have worked would be to align X along the links of the upper arm but reverse the z axis with every alternate link. However, this would have added complexity in the DH-table since we would have to add additional alpha values to map the frames along the X axis.

1.1.2 Simulation

The robot was created using line objects and the frames were created using quiver objects in MATLAB. α was chosen to be +90 degrees so that all the θ angles (rotations around the Z axes) would be positive. The X-axes (red) for the frames always follow the link. Z (blue)

is the axle of the servo around which the different theta angles are measured and Y is at a right angle to these two axes.

As seen in Figure 2, all the Z frames are pointing out of the plot and X follows along the link. It is important to note that there are two overlaying quivers at the base, one of which is orientated in the usual XYZ position following the right-hand rule where Z points vertically upwards. The other frame has a 90-degree rotation around the X-axis, forcing the Z-axis to point out of the page. This second frame sets the convention for the other frames.

The forward kinematics model that has been created is a general model, that shows an appreciation of the position of the robot in XYZ dimension relative to each joint angle. However, other offsets will have to be specified during the calibration of the real robot arm to make our forward kinematics model more true to our real robot arm.

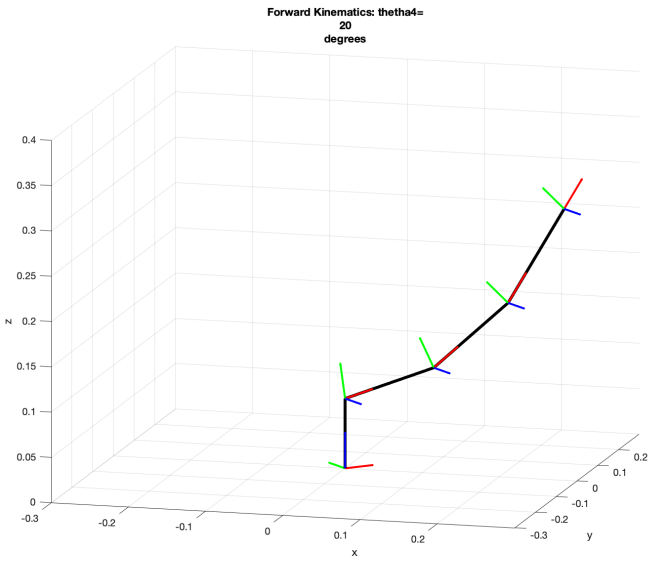

Figure 2: Shows Forward kinematics of the robot with the angle of the wrist servo being 20 degrees

1.2 Inverse Kinematics

Figure 3 below illustrates how the inverse kinematics was split into two main sub-problems highlighted in the two different colours used.

When approaching the problem of finding the inverse kinematics for the robotic arm, the problem was split into two sub-problems: solving the inverse kinematics for the spinning base of the robot and solving the inverse kinematics for the rotations in the arm of the robot. This approach was taken since the rotations in the arm of the robot, all occurred in the same 2D plane that was different to the plane in which the robot spun. This can be seen in the diagram to the right, where the first sub-problem is highlighted in red and the second in green.

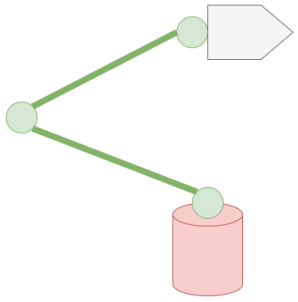

Figure 3: Shows the sub problems the robotic arm was split into to solve for the inverse kinematics

.

To solve the inverse kinematics of the robotic arm, a geometric and analytical method was taken. This was achieved by drawing out the angles that were required to be calculated and finding the trigonometric equations that would solve for them, as seen in the diagram below, see Figure 4.

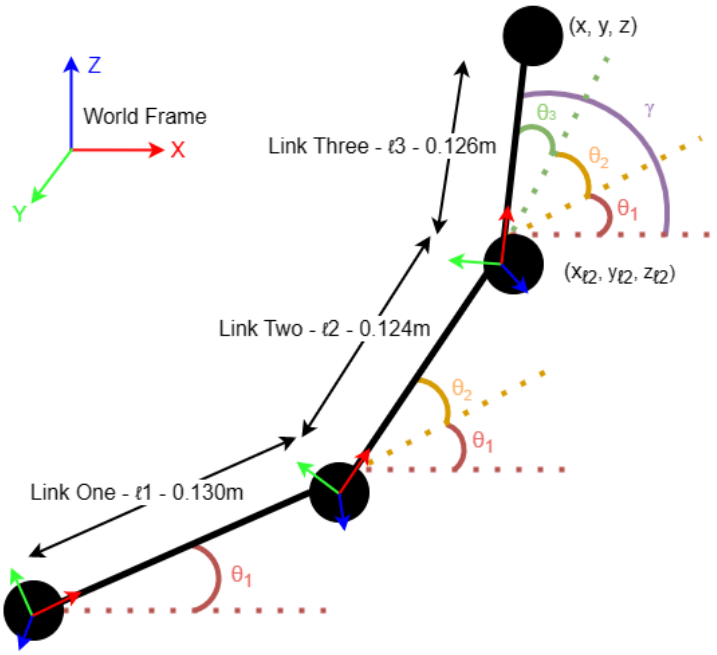

Figure 4: Diagram showing the different angles that need to be solved for during our Inverse Kinematics calculations

.

It is important to note that the angle of the wrist or head of the robot was not considered a parameter that had to be solved for. In implementation, it was decided that this angle would be an input parameter that the user would decide, similar to the other input parameter, the target position of the robot. This decision would allow the user to have control over how the robot's head would approach an object, which would be a necessary parameter to control in the following tasks. However, this also meant that in calculations, the lack of a head/wrist must be accounted for. For example, the inverse kinematics calculations should now solve for the target values of the second link (see Figure 4), ignoring the head/wrist of the robot (See Appendix for more detail).

It is also important to note that the inverse kinematics returned two solutions for each position: an elbow up solution and an elbow down solutions. Due to physical restrictions of the robot, the elbow up solution was always taken.

To validate the accuracy of the inverse kinematics, simulations were performed using the forward kinematics to ensure that the solutions from both kinematic models were consistent. Once the simulation results were satisfactory, physical trials were conducted with the robot to confirm the correctness of the inverse kinematics. This said, at first there were some minor discrepancies between the simulation and physical robot that are explored in a later section.

1.3 Drawing a Square

This task was completed by using a few while loops to generate edges of a square in the X, Y or Z plane. To generate a square in the XY plane, Z would be fixed at a constant value. "While Loops" could be used to generate the X and Y coordinates for the edges. For example, an edge parallel to the Y axis would be created by stepping through values of Y

while keeping X constant. Alternatively, an array of coordinates could be generated by using linspace and then accessed using a for loop. Before drawing the line for the edge, it was useful to plot the end effector location to check whether a square was really being drawn. Often at an unreasonable γ angle (See Figure 4) that prohibited inverse kinematics from reaching the desired location (either resulting in an error in computation of tan or going to the nearest possible coordinate) would result in an odd shape being drawn so it was useful to see the exact movement of the end effector. The lines which make up the edges of the squares were eventually generated by passing through the previous and recent end effector position and then updated in the next iteration.

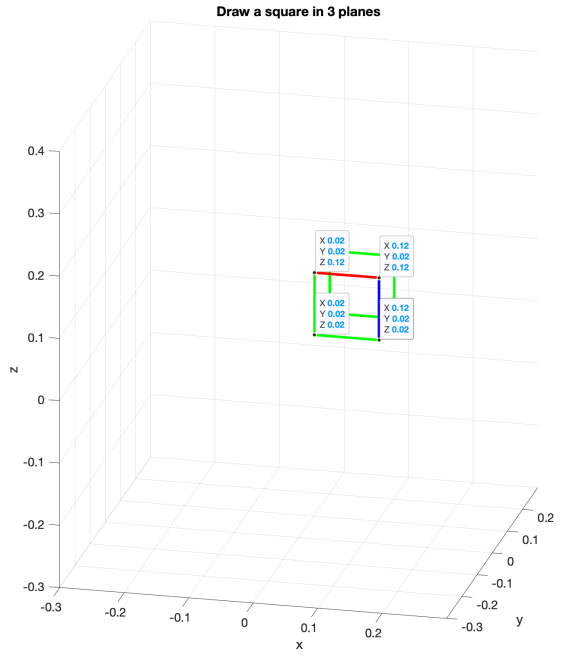

Figure 5: Showing the square drawn in 3 planes. The cube is 10x10x10

.

1.4 Mapping from Simulation to Real Robot

To map the simulation onto the real robot a new function was created that would transform the angles computed by the inverse kinematics to a 12 bit number (0 - 4095), that the servos could use to function. The angles that were created from the inverse kinematics were offset by the orientation of the servos and the mapping function would take coordinates rather than a distance in metres. The coordinates are taken as every square hole in the perspex board, which are accurately space at 25mm apart in the X and Y direction.

2 Task Two - Pick and Place

2.1 Graphical User Interface

A Graphical User Interface (GUI) was created to provide precise manual control and adjustment of the robot (Appendix 5.1). It enabled us to specify the block coordinates and choose from

different actions, including rotation, placement, gripping, or dropping. Additionally, the GUI generated a graph representing the inverse kinematics motions in real-time. This has proven to be incredibly useful in testing and debugging the robot arm's capabilities as well as verifying the inverse kinematics.

2.2 Task 2A - Translation

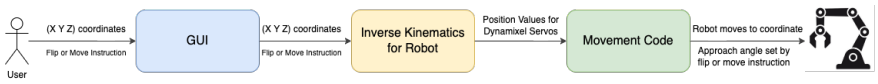

Figure 6: Shows the full chain of functions resulting in controlled robot movements

.

From the Figure 6 the user must only input: the coordinate values of the grid, the raised distance (Z) and provide the instruction of move or flip. The 'Inverse Kinematics Code for Robot' transforms the coordinates to a distance, by using the spacing of 2.5mm between each acceptable position in the X and Y plane. These distance values are then fed into the inverse kinematics code, created in Task 1 that produce angles from our MATLAB model. An offset of 10.62° to account for the offset between the shoulder and the elbow/wrist. After obtaining the correct angles, the angles are converted to position values that can be used to control the servos. A full revolution corresponds to 4095, with the 180° being 2048. The position values obtained are then sent to the movement code that inputs them as arguments in the move-to-goal function. Note that 180° corresponds to 12o'clock to each servo.

The user can also send an instruction to rotate the block. This will change the gamma angle of the inverse kinematics such that the robot gripper is parallel to the plane ($\gamma = 360°$) rather than perpendicular to the plane($\gamma = 270°$). This will allow the robot to rotate the block by 90° in the clockwise direction. To perform rotations of angles greater than 90°, the robot must pick, flip, and place the block down several times.

2.3 Task 2B - Rotation

One of the main problems faced in the implementation of this task was finding an algorithm that worked with the robot's limitation. For example, reaching certain block coordinates with the gripper faced down was impossible as the inverse kinematics could not provide a viable solution, which then meant the robot was unable to flip the block in the same spot. This problem was tackled by moving the block to a different coordinate position, flipping the block at this new coordinate and then placing the block back at the goal end position.

2.4 Task 2C - Stacking

This task had similar problems to the previous one with the added complexity of increasing the Z-value of the place position of the additional blocks such that the gripper would not knock over already stacked blocks. An intermediate position was added such that where inverse kinematic solutions existed, the robot would hover a few cms higher in the Z direction before slowly lowering, placing the block and hovering again a few cm higher than the place position. This prevented the blocks from being tipped over when the arm was returning to the rest position. Within the move function, conditional statements were added to make

sure the robot could always move to a viable position even if the intermediate position was not solvable due to the robot's own restrictions.

Furthermore, a difficulty experienced in all of Task 2 was understanding the relationship between the depth of a block when picked up by the gripper at an angle of 270° (straight down) to 360° (straight forward). Often the position of the end effector would have to be calibrated when replacing the block to the same position but at another end effector angle. For the stacking task, we wanted to experiment with this concept and try place the blocks on the diagonal, which was difficult since both X and Y would be needed to be calibrated well.

3 Task Three - Trajectory Following

3.1 Gripper Design

The purpose of the gripper was to pick up a board marker pen and draw. The design was built around the idea of ergonomics and reducing plastic usage. The gripper is designed with two halves that pinch the pen with semicircle slots on each side, ensuring a secure grip. The semicircular slots are measured and made to fit the pen's conical shape. Compatibility with the robot's range of motion, weight reduction, and material strength are also crucial design considerations. Cut-outs in the gripper have been made to minimise the plastic and weight of the design, without compromising structural integrity. This solution meets all the criteria for successful and secure pen pick-up. Furthermore, a pen holder was designed to allow the gripper to easily pick up the pen.

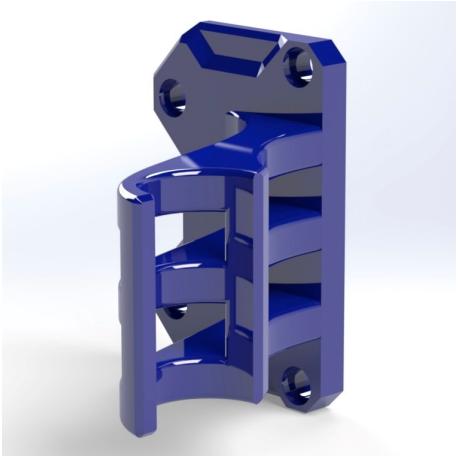

Figure 7: CAD Gripper Design

3.2 Drawing the Shape

The shape can be drawn in any order by calling the functions to draw a horizontal/vertical line, a diagonal line, or an arc. The starting and ending coordinates need to be given for the lines whilst the arcs ending coordinate is calculated by using the arc angle and radius. These functions all use linspace and by understanding and manipulating the trapezoidal nature of the time base profile of the drive mode, successive inverse kinematics solutions can be given to the robot so that it moves at a constant velocity.

If provided with more time, a further exploration into a solution involving cubic interpolation could be derived to help account for overshoots and jitters in the drawing (by minimising the effects of inertia and controlling acceleration). However, this was not deemed necessary since the motion near the end of the shape was slower anyways due to the trapezoidal time based profile. The arc can be drawn either clockwise or anticlockwise depending on the angle given and the number of points in the linspace is calculated from the length of the line in addition to the angle of the arc. We first simulated the robot before actually attempting to draw the shape with the real robot. This minimised the number of errors due to unsolvable inverse kinematics and allowed us to test several different orientations of the lines.

4 Task Four - Clothes Folding

4.1 Motivation

Our proposed task was to create a gripper and a robot that is able to pick and fold clothes. This was motivated by the fact that this action would benefit both industry and domestic use. For example big warehouses could use this robot to package clothing that is sold in stores. Similarly within a home, this robot can be used to achieve tedious household tasks efficiently and could be used by people who have arm/hand injuries and would be unable to complete such tasks.

Another reason behind our proposed task stemmed from the fact that a significant amount of manipulator research focuses on this particular task but typically utilising two arms. As a result, solving this task using only one arm was seen as a compelling challenge and an interesting concept.

4.2 Solution

Our solution to folding an item of clothing using only one arm was solved by introducing a mobile weight. This weight was used to hold the item of clothing down whilst a fold was being made. The weight was manoeuvred by using the robotic arm however, a key problem faced was that weight would often move whilst a fold was being made. This made the task of relocating the weight particularly challenging since it was incredibly important to keep track of the exact location of the weight after each fold. Through rigorous calibration this issue was solved. Another solution to this problem could have been to use a soft, pliable weight since by using a rigid weight it meant the precise location of the weight had to be found each time.

Other challenges faced were picking up and folding a non rigid body such as an item of clothing and pinching thin fabric from a surface, which was a big motivator in the gripper design.

4.3 Gripper Design

The gripper has been designed to pick up clothes by pinching them securely. The gripper consists of two halves with fine protrusions to provide a firm grip on the clothes. The gripper features cut-outs to reduce its plastic usage and overall weight while maintaining its structural integrity. This solution fulfills all the necessary criteria to successfully pick up the clothes. Additionally, the gripper was designed with a safety feature at the base to prevent over-clamping, avoiding potential damage.

Figure 8: Clothes Gripper

5 Appendix

5.1 Graphical User Interface

One of the key features of the GUI is its ability to set block coordinates. This means that the user can specify the exact location of the object that the robot needs to manipulate. This information is then used by the robot to determine the precise movement required to perform the desired action.

The GUI also provides the user with a range of different actions that the robot can perform. These include rotating, placing, gripping, or dropping. By selecting the appropriate action, the user can instruct the robot on what task to perform. In addition to these features, the GUI generates a real-time graph that depicts each of the inverse kinematics motions. This graph provides a visual representation of the robot's movements, which can be useful in identifying any issues or anomalies.

Overall, the GUI has proven to be an effective tool in testing the robot arm's capabilities. It provides a user-friendly interface for controlling the robot, enabling precise positioning and manipulation of objects. Additionally, the real-time graph provides a useful visual representation of the robot's movements, aiding in troubleshooting and identifying areas for improvement.

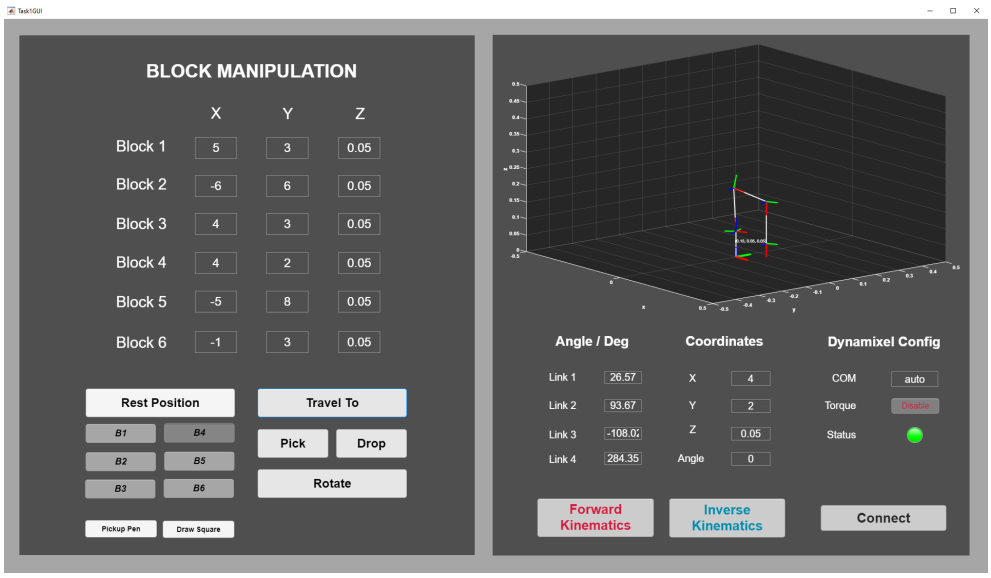

Figure 9: GUI design for Task 2

5.2 Mathematics for Inverse Kinematics

Calculating the inverse kinematics for the robotic arm began by first finding the target coordinates for the second link. This was done by solving the equations below that were derived geometrically. It is important to note that an offset of 0.077m was subtracted to compensate for the Z-offset the base provides.

From the above figure, the following equations were derived and solved:

$$Z_{difference} = \ell_3 sin(\gamma) \tag{1}$$

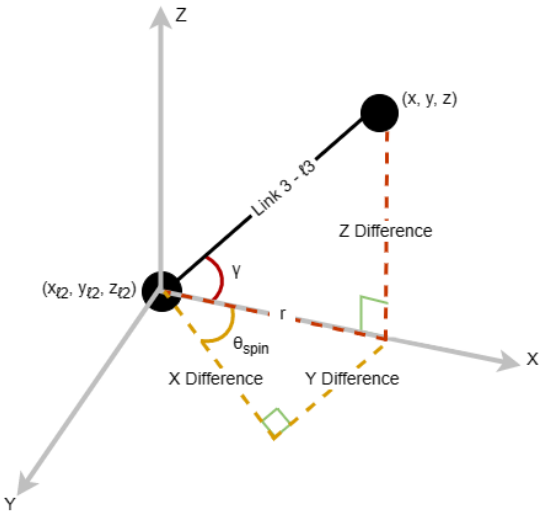

Figure 10: Diagram illustrating the geometrical approach taken to find the target coordinates.

.

$$r = \ell_3 cos(\gamma) \tag{2}$$

$$X_{difference} = rcos(\theta_{spin}) \tag{3}$$

$$X_{difference} = rcos(\theta_{spin}) \tag{4}$$

$$Y_{difference} = rsin(\theta_{spin} \tag{5}$$

$$(x_{\ell_2}, y_{\ell_2}, z_{\ell_2}) = (x, y, z) - (x_{diff}, y_{diff}, z_{diff}) \tag{6}$$

After finding the target coordinates for link two, the next step was to solve for θ_2 in the second sub-problem. The relevant labels and nomenclature for the angles and values can be seen in Figure 10.

Solving for θ_2 consisted of recognising that the x_{ℓ_2} position and z_{ℓ_2}position could be written as a combination of the distances of the first and second link, seen below:

$$x_{\ell_2} = \ell_1 \cos(\theta_1) + \ell_2 \cos(\theta_1 + \theta_2) \tag{7}$$

$$z_{\ell_2} = \ell_1 \sin(\theta_1) + \ell_2 \sin(\theta_1 + \theta_2) \tag{8}$$

Rearranging these two equations gives us the following mathematics and an equation for θ_2:

$$x_{\ell_2}{}^2 + z_{\ell_2}{}^2 = \ell_1{}^2 + \ell_2{}^2 + 2\ell_1\ell_2 \cos(\theta_2) \tag{9}$$

$$\cos(\theta_2) = \frac{x_{\ell_2}{}^2 + z_{\ell_2}{}^2 - \ell_1{}^2 - \ell_2{}^2}{2\ell_1\ell_2} \tag{10}$$

It is important to note that the above mathematics considers the case in which the robot has no spin angle. However, after introducing a spin angle, the mathematics must change to account for this, which can be seen below:

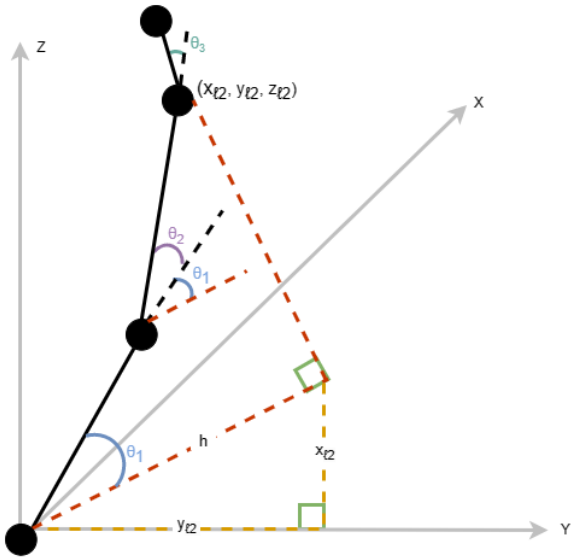

Figure 11: Diagram illustrating how the angles in the arm change after adding a spin angle in the base of the robot.

.

However, since we wish to utilise the $Atan2()$ function and the following final equation is used to find θ_2. Note how calculating for $\sin(\theta_2)$ gives two solutions: the elbow up solution and the elbow down solution.

$$\sin(\theta_2) = \pm\sqrt{1 - \cos(\theta_2)^2} \tag{11}$$

$$\theta_2 = Atan2(\sin(\theta_2),\ \cos(\theta_2)) \tag{12}$$

The final step is to find the equation that would solve for θ_1 which can be seen below:

$$k_1 = \ell_1 + \ell_2 \cos(\theta_2) \tag{13}$$

$$k_1 = \ell_2 \sin(\theta_2) \tag{14}$$

$$\theta_1 = Atan2(x_{\ell_2},\ z_{\ell_2}) - Atan2(k_1,\ k_2) \tag{15}$$

However, the final equation seen above, does not take into account a spin angle. Once a spin has been introduced, the equation changes to:

$$\theta_1 = Atan2(h,\ z_{\ell_2}) - Atan2(k_1,\ k_2) \tag{16}$$

5.3 CAD Models

5.3.1 Task Three - Gripper Design

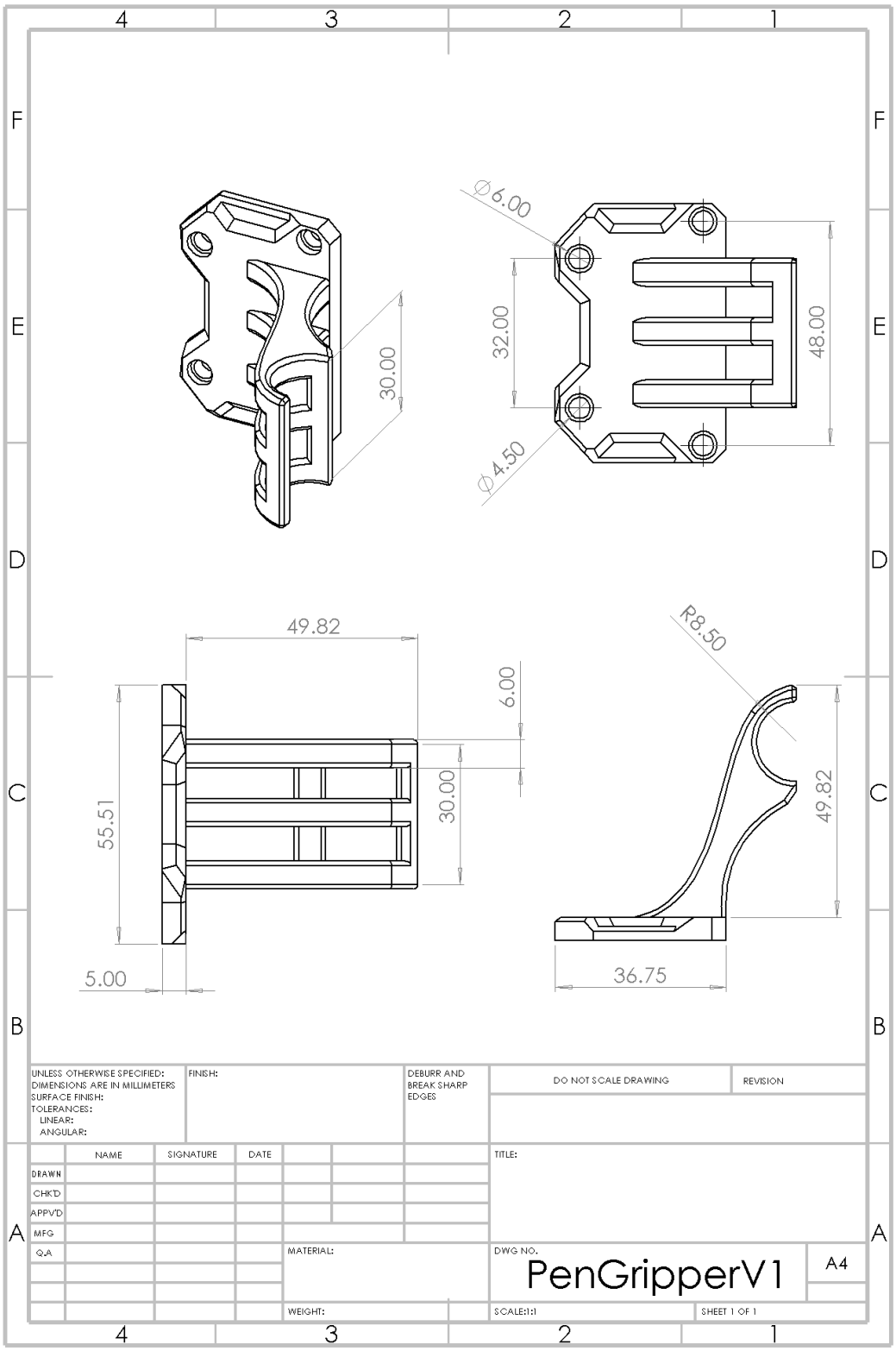

Figure 12: CAD design for the pen gripper

.

5.3.2 Task Three - Pen Holder Design

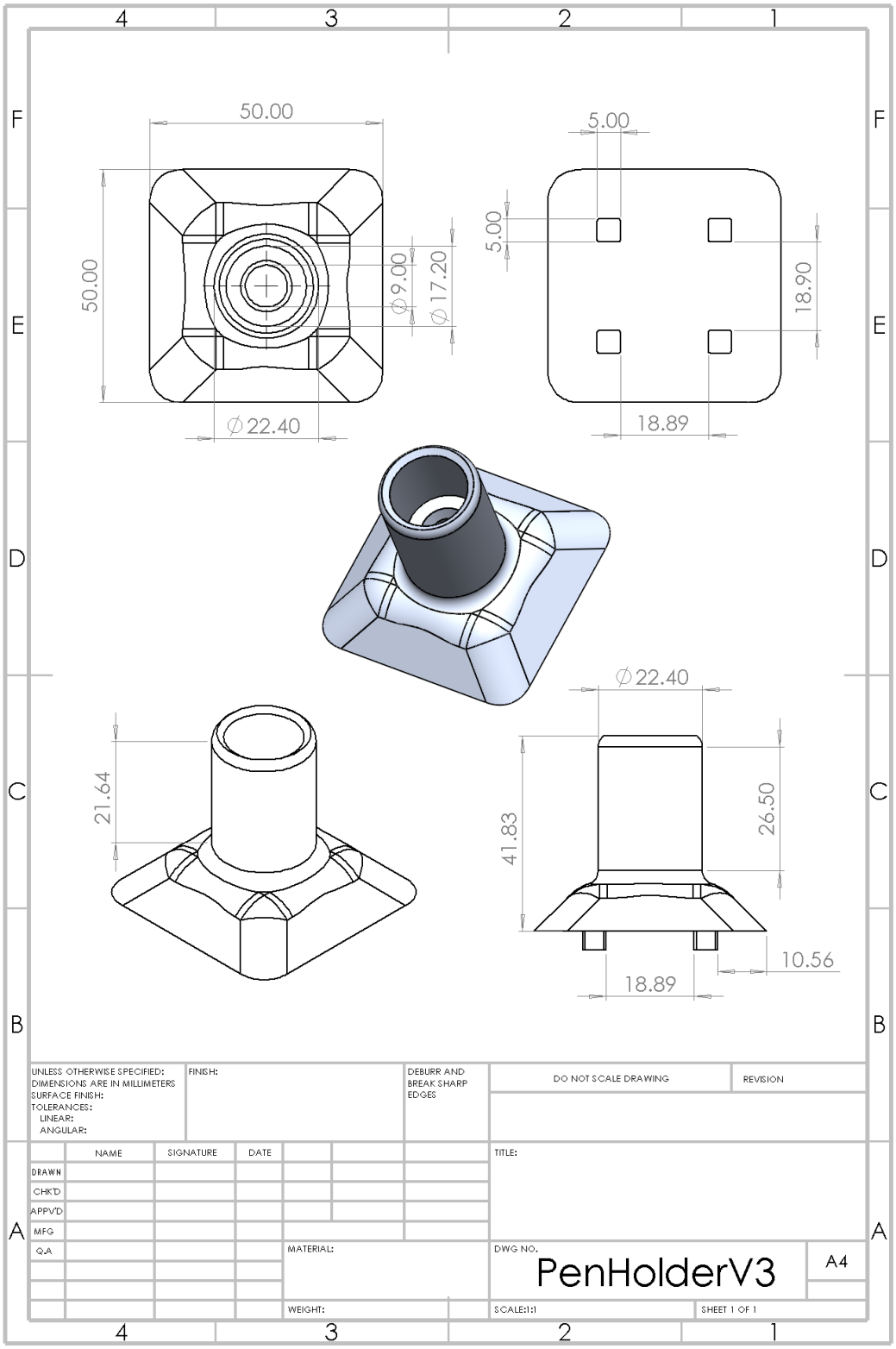

Figure 13: CAD design for the pen holder

.

5.3.3 Task Four - Gripper Design

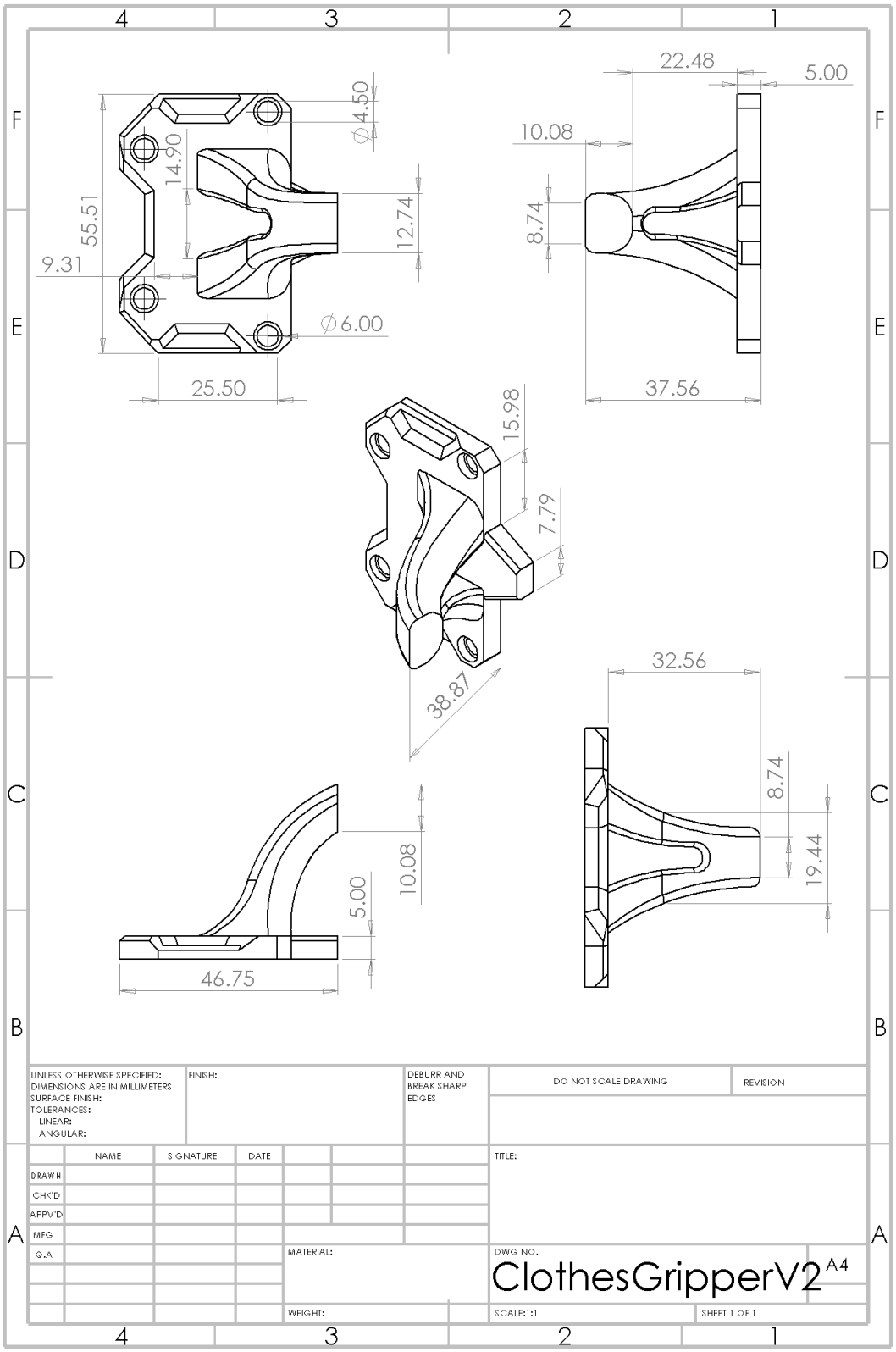

Figure 14: CAD design for the clothes gripper

.