**ECE 5425**  **Project Report**

**Group #8. Li Lin & Zhaofeng Tian**

1. **Abstract**

In this study, a three-links robot is built.

1. **Introduction**

As automation is growing fast all over the world, robotics is one of the most promising industries that would contribute significantly to human society. In the auto industry, the robot arm is commonly used to grab and move mechanical parts, while also help weld parts in the workspace; As a product of mechanical engineering, electrical engineering, and computer science, the robot arm can be commanded to do high accuracy motions which outperform human beings. Thus, the robot arm is used in the medical industry as well, to perform operations for the patients; In terms of STEM education, a three-link robot arm could interest kids very much, and inspire them to chase STEM dream in the future.

In this study, we propose a three-link robot arm called “G8 robot arm”, and showcase how we build it in physical world, create simulation environment in Matlab App designer, and how we optimize DH parameters using the comparisons between measurements of the distance from end-effector to clipboard frame origin and corresponding calculation results from simulation.

Rest report sections are arranged as follows. In section III, our methods of building robot in physical world and the simulation environment are proposed; In section IV, our results of DH parameter optimization are shown; Conclusion and Discussion are respectively put in sections V and VI.

1. **Methods**

In this section, the details of building process are illustrated step by step, which synchronizes with the project-required steps.

In the first step, we arrange the base on the clipboard such that we have the maximum joint ranges and that the robot can reach all corners of the given board with this additional criterion, we shorten one of the joint lengths of your robot arm by the smallest digit from all group members user id. So, as the smallest digit in my group ids is “1”, we shorten link 1 and 2 by 1 hole from the max length shown as the Fig. 1.

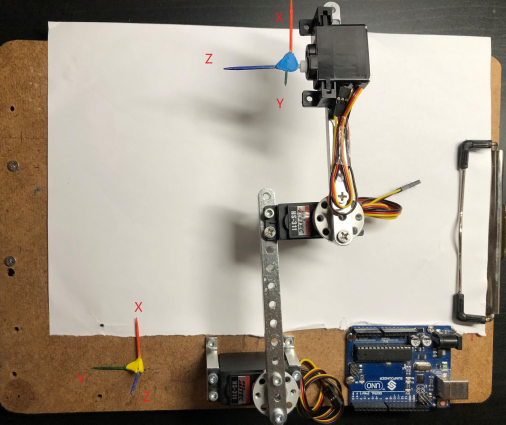
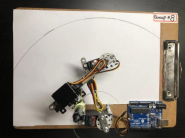


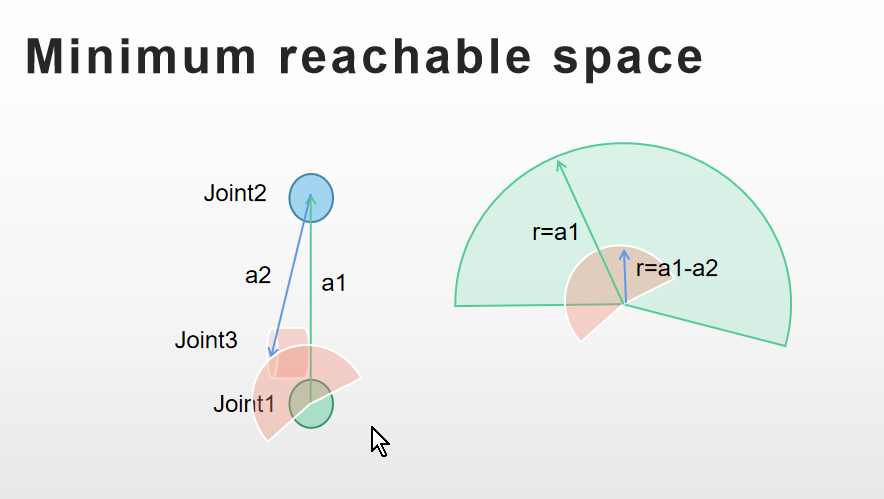
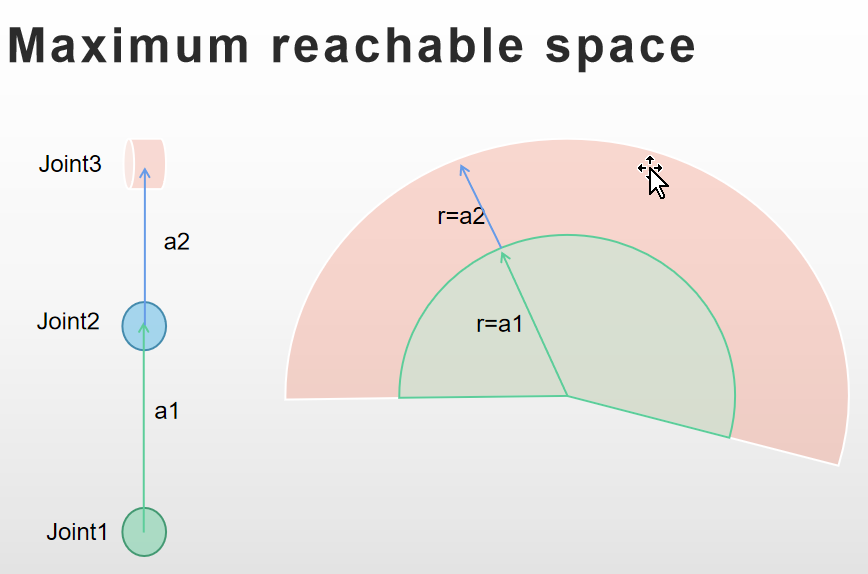
Fig. 1 Robot arm setup

As you can see, we move the base frame along the Y axis for 0.08 meters to get a clipboard frame, where the one of two toothpick-made frames locates. The other toothpick-made frame locates at the end-effector frame origin. Then all the measurements are pointed from end-effector to the clipboard frame instead of the base frame, which should be noted and is set as default in latter measurements.

In the second step, we tried several possible angles to locate the end-effector, and finally choose the current setup shown as Fig. 1(a). Meanwhile, we have drawn the range limits of the end-effector on a piece of paper.



(a)



(b)

Fig. 2 Minimum and Maximum range of the end-effector.

FromFig.2 (b), we can get the reachable space of the end-effector as the red area shows.

In the step 3, we put the toothpick-made frames at the respective locations shown as Fig.1.

In the step 4, we measure the DH parameters for the robot arm following the textbook instructions. All measured DH parameters including theta value limit (joint angle limit) and offsets are shown in the Table. 1 below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | a  (m) | alpha (degree) | d  (m) | theta  (degree) | offset  (degree) |
| Joint 1 | 0.1 | 0 | 0.047 | -120 ~ 90 | -19 |
| Joint 2 | 0.073 | -90 | 0.02 | -120 ~ 90 | 16.8 |
| Joint 3 | 0.0 | 0 | 0.02 | 0 ~ 135 | 0 |

Table 1. DH parameters.

1. Firstly, we create a GUI in the Matlab App designer shown as Fig. 3 to tune the DH parameters and showcase the result by plotting the robot based on preset DH parameters. Note that Base Frame Translation section in the GUI, we move the previous base frame to the clipboard frame, so we need to multiply a translation matrix to get the new coordinates of these joints. Since we moved the base frame for 0.08 meters along the Y-axis of the previous base frame, all coordinate value of the joints in the new base frame, i.e., clipboard frame will have a translation along the counter-direction of Y-axis of new base frame, so the translation matrix has a Y value at

–0.08.

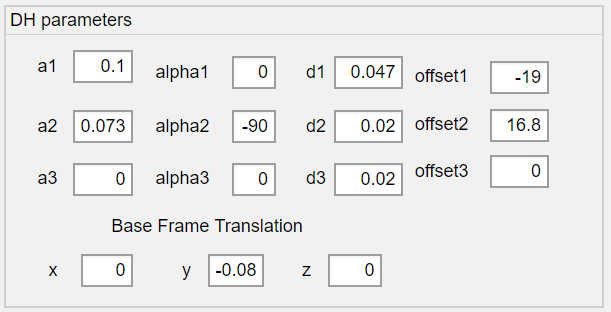


Fig. 3 DH parameters GUI

1. Secondly, to explain how the DH parameters are determined, here we are using a image to illustrate it. In the Fig. 4, all DH parameters are labeled on the links and joints. In the Fig.5, offsets are labeled.

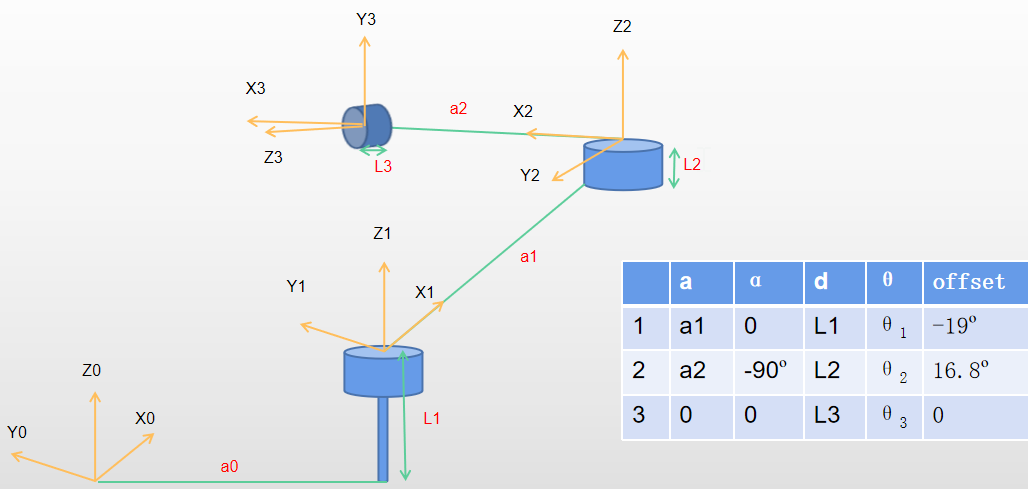


Fig. 4 DH parameters

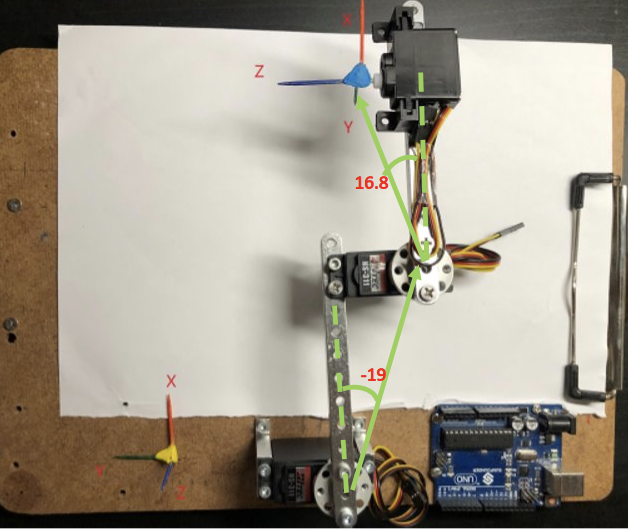


Fig. 5 offsets

1. To create the robot from DH parameters, we use a CreateRobot function shown as Fig. 6. This is a callback function of “CreateRobot” button, which draws the DH parameters to each Link by using L(i) = Revolute (DH parameters), where “Revolute” means to create a link with a revolute joint. Then we create a robot variable app. G8Robot to store the robot information which is generated by SerialLink function, a function to create a robot from links we build eariler (combine those L(i) to a full matrix, and feed it to the SerialLink() as a augment). Note that the joint angle limit is also presented in “qlim” label.

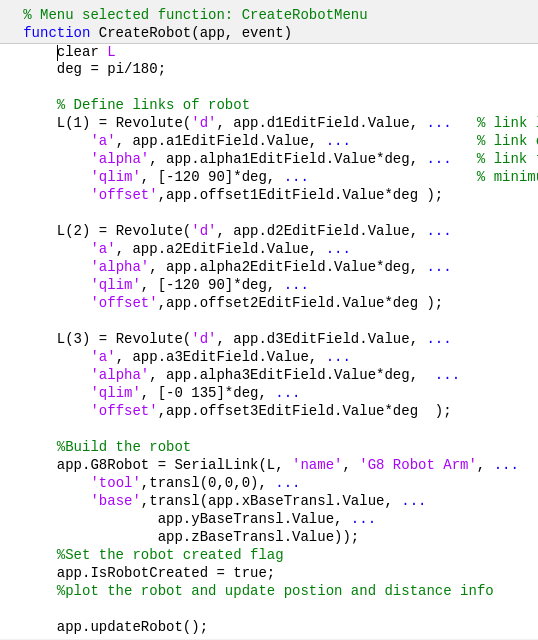


Fig. 6 Commands

1. The joint limits are declained in the “qlim” as an augment of Revolute(), meanwhile, in the latter design, we place a limit on the joint angle by setting limit values in the slider of the Matlab App designer.

In the step 5, we are using sliders to control the join angle and plot the robot in the GUI. Shown as Fig.7, the robot could be updated by each change in the sliders using a callback function that calls app.updateRobot() (a method in the class) to use app.G8Robot.plot function to plot the updated robot.

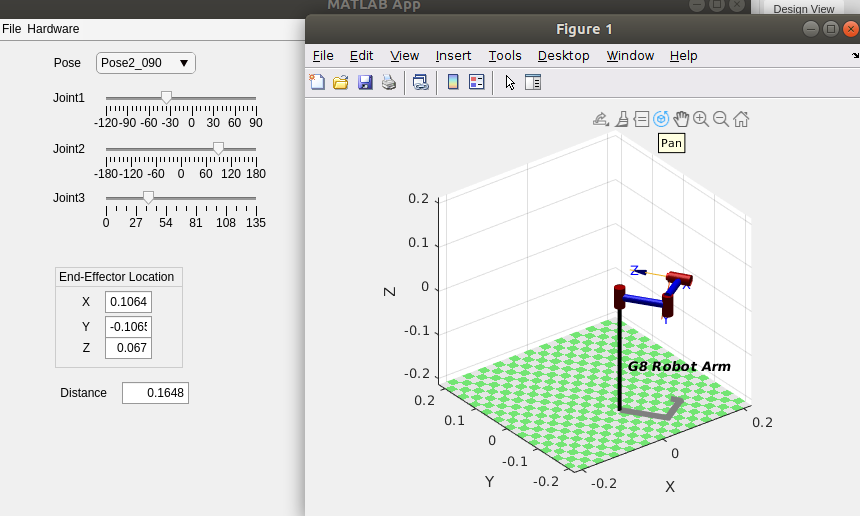


Fig. 7 Slider control

In the step 6, the codes used to implement keyboard cotrol are shown in the Fig.8, we use UIFigureKeyRelease(app, event) function to get the key input in the GUI window where envent.key can get exactly the key we input in the GUI. So, when detect arrow keys, the respective operations will be performed. In this case, we use leftarrow and rightarrow keys to control robot end-effector to move on the Y direction and use uparrow and downarrow to control robot end-effector to move on the X direction. For example, we get the leftarrow key as an input, the codes will update the end-effector transform matrix by changing the translation part with a motion along Y at 0.005, then get all joint angles stored in q2 by using inverse kinematic function ikine(), then update the slider value and call app.updateRobot() to plot the new robot configuration.

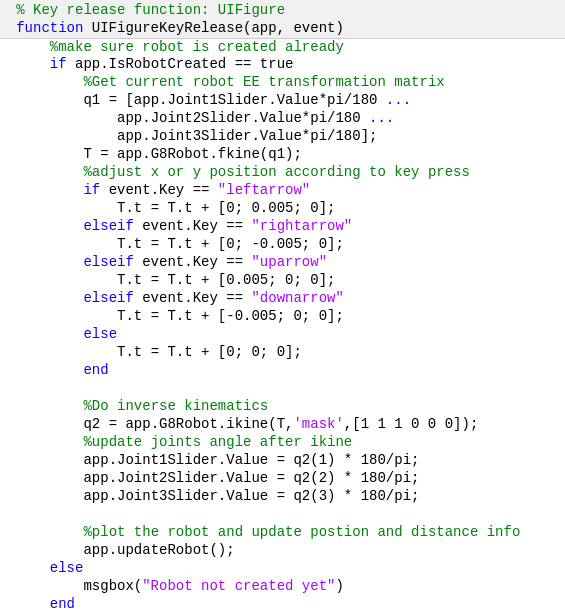


Fig.8 Keyboard Control

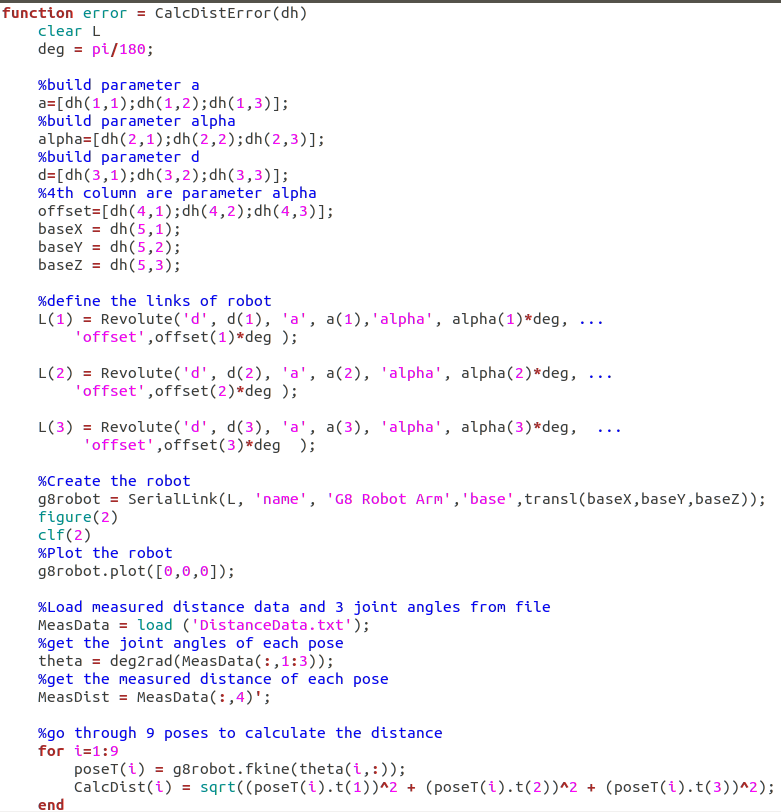
In the step 7, we determine 9 poses and measure the distance from end-effector to the new base frame (clipboard frame), also, we calculated respective distance by firstly using the fkine() function to get a transform matrix of end-effector and also get the translation part and then use the translation part to calculate the distance to the base frame, which is implemented in the class method app.updateRobot(). All the poses, measurements and calculations are listed in the Table. 2. Also, the poses and measurements of the distance are stored in the file DistanceData.txt.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Joint 1  (degree) | Joint 2  (degree) | Joint 3  (degree) | Measured Distance  (meters) | Calculated  Distance  (meters） |
| Pose\_1 | 0 | 0 | 0 | 0.212 | 0.2047 |
| Pose\_2 | 0 | 90 | 0 | 0.115 | 0.1095 |
| Pose\_3 | 90 | -90 | 0 | 0.233 | 0.2273 |
| Pose\_4 | 90 | 0 | 0 | 0.118 | 0.1119 |
| Pose\_5 | 90 | 90 | 0 | 0.087 | 0.07868 |
| Pose\_6 | 90 | -90 | 0 | 0.136 | 0.1296 |
| Pose\_7 | -90 | 0 | 0 | 0.264 | 0.2576 |
| Pose\_8 | -90 | 90 | 0 | 0.175 | 0.1759 |
| Pose\_9 | -90 | -90 | 0 | 0.231 | 0.2292 |

Table 2. Poses and distance

In step 8, we write an objective function shown as Fig.9. As an input of this function, dh is a matrix that combined all DH parameters. Then we use the DH paramters to create our robot and then load the poses and mesurements from DistanceData.txt. Then we use each pose to calculate a transform matrix of end-effector by using fkine() function, and fianlly calculate the distance from end-effector to the new base frame (clipboard frame) by get the translation part in the transform matrix and apply distance function to get the final calculated distance. Then use the obsolute value of the difference between measurements and calculated distance to be the error.

In summary, the objective function is to calculate the distance error between measurements and calculations, and then the error could be optimized in the later fminsearch() function.



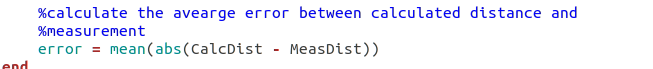
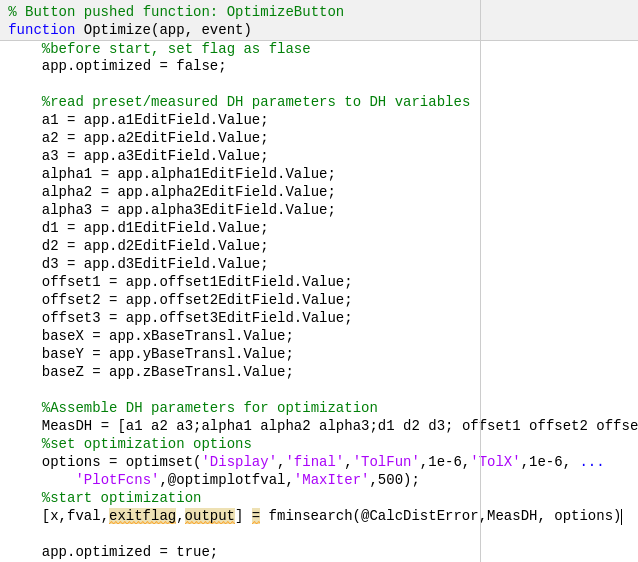


Fig.9 Objective Function

 Fig. 10 Optimization

In the callback function of “Optimize” button, we perform Optimize () function which uses fminsearch() function as its core function. The input parameters of fminsearch() function are the objective function (CalcDistError()), a metrix of DH parameters (MeasDH), options. While the objective function represents what you want to optimize, DH parameters are what can affect objective function and can be tuned. And options are about some setting of plotting requirements, tolerated error range (1e-6), and the number of iterations (500 interations). The output contains final error value.

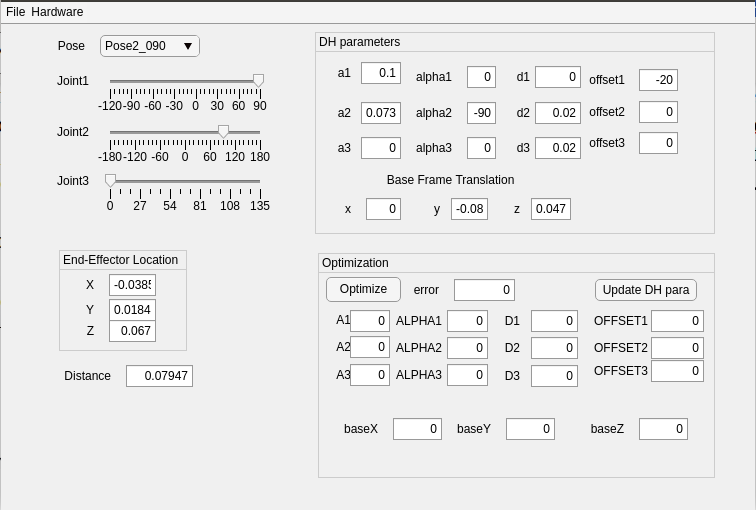
1. **Results**

In this section, we will explain our designed App, showcasing its functionality and show the results of optimization. This project has three main files:

* G8RobotArm.mlapp
* CalcDistError.m
* DistanceData.txt

The relationship among them is that .mlapp file is our designed App, so it could be regarded as the body of the project, .m file is used as the objective function in the fminsearch() function, while the .txt file contains 9 poses and corresponding distance measurements from end-effector to the new base frame, which is to be used in the CalcDistError() function.

The below shows our GUI window,



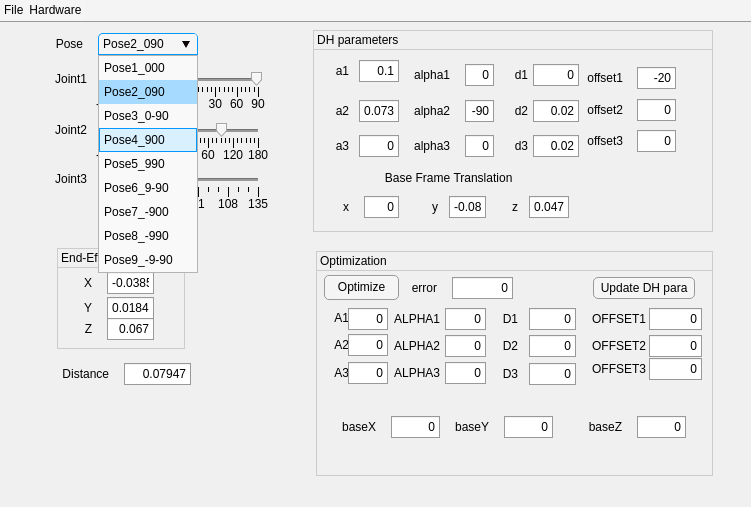
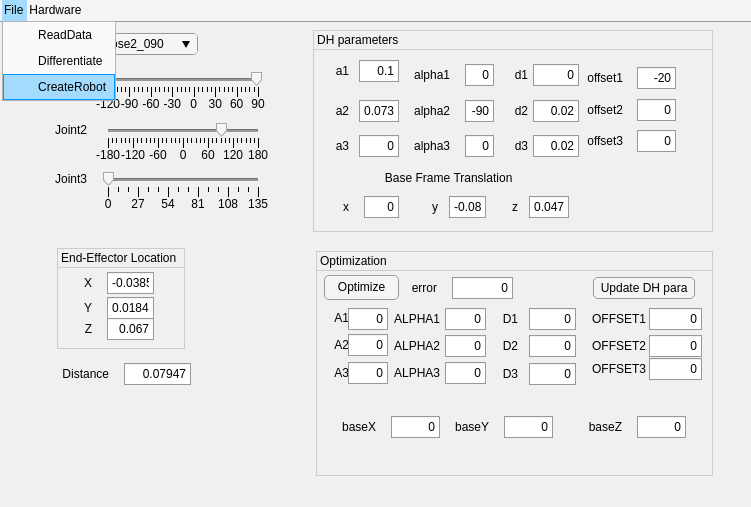


Fig. 11 GUI window.

And you can edit DH parameters, base frame translation parameters in it. Also, you can choose a pose and then press the “CreatRobot” button to create a robot. By tune the slider position, the robot can be replotted, which is showcased in Fig. 7. Then end-effector location and distance to the new base frame are also presented in the left side of the GUI window.

Next, we will showcase how optimization works in the GUI.

By pressing the “Optimize” button, the optimization of DH will be performed. Then the robot will be updated in the plot window, and the error will be improved in another window.

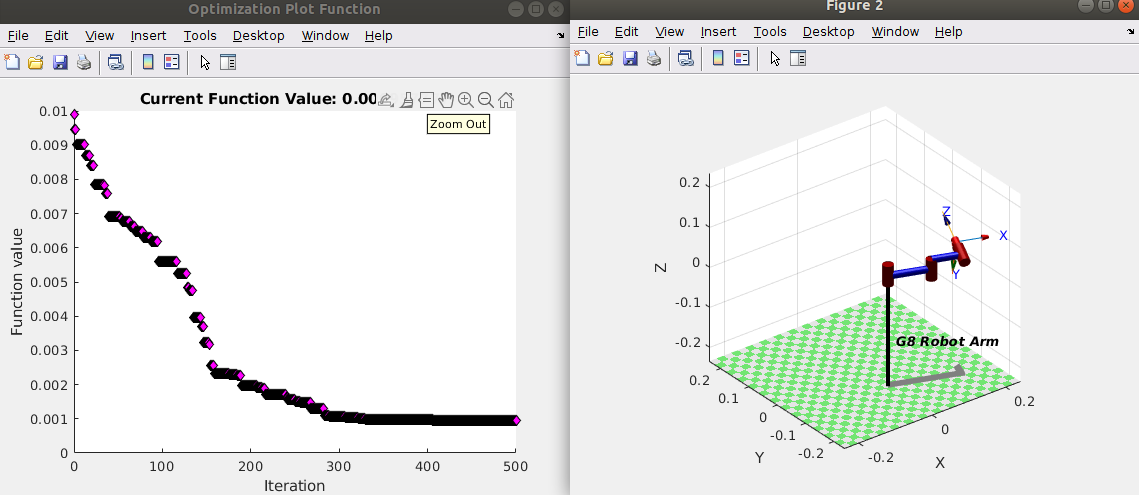


Fig. 12 Optimization window

After each optimization iteration, the DH parameters will be updated, and the robot will be replotted in the plot window. After 500 iterations, the error is reduced from 0.00991 to 0.00095, which is improved significantly.

1. **Conclusion**
2. **Discussion**