

Tennessee Technological University

Mechanical Engineering Department

Prof. Dr. Stephen Canfield

Intro to Robotics – ME 4140

Final Project Report

Adriano Henrique Rossette Leite – adrianohrl@unifei.edu.br

Michael Robson Araujo Leite – michael.robson.al@gmail.com

Matheus Mirapalheta Longaray – longaray.matheus@gmail.com

Intro to Robotics – Final Project

The robotic arm was designed to work with six degrees of freedom. It consists of a base and six more links where these links form six revolute joints. The three last joints form a wrist configuration.

First, each link was modeled separately using the 3D CAD Design Software SolidWorks and saved as STL files. Once designed, all the links were assembled to inspect some arm’s motion aspects such as the limit angles of the joints. These limit angles would be used to determine the workspace of the arm. It can be useful if the inverse kinematics presents two solutions for any angle. So, the workspace could be used to verify whether both angles/solutions are valid or not.

|  |  |  |
| --- | --- | --- |
| C:\Users\Michael Robson\Desktop\base.png C:\Users\Michael Robson\Desktop\link2.png | C:\Users\Michael Robson\Desktop\link4.png | https://lh3.googleusercontent.com/iZP3Qb6OvVi1I-qwTa5dgKGKU_jcja-Jw1AmYYdI0M7WOPGcv21glSB--H4gYphj-SjfK3gLaLYbXF3xZtTjjM1W37YAfwrlGJOhxHIGyMkURRk0V1cQJwxtEJY8 |
| C:\Users\Michael Robson\Desktop\link1.pngC:\Users\Michael Robson\Desktop\link3.png |
| C:\Users\Michael Robson\Desktop\link5.png C:\Users\Michael Robson\Desktop\link6.png |

Figure 1 – Links and the assembled robotic arm.

The STL files were imported to our MATLAB code. Since the links were placed on the same origin in space, it was not assembled as it was supposed to be. So, transformation matrixes were used to move/rotate the links in order to set up the robot properly.

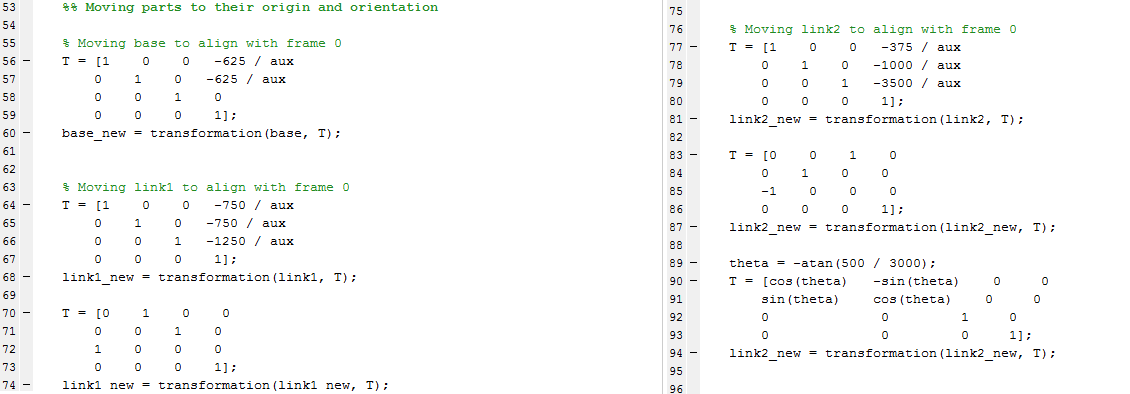


Figure 2 – Adjustment matrixes to set up the arm.

The forward kinematics was analytically solved using the D&H Table as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i |  |  |  |  |  |
| 1 | 0 | 90 |  |  |  |
| 2 |  | 0 |  |  |  |
| 3 |  | -90 | 0 |  |  |
| 4 | 0 | 90 |  |  |  |
| 5 | 0 | -90 | 0 |  |  |
| 6 | 0 | 0 |  |  |  |

Figure 3 – Final D&H Table according to the measurements of the assembled links.

The D&H table would provide the transformation matrixes which were used to solve the forward kinematics.

The inverse kinematics was solved analytically for all joint variables. Some minor changes on the original design were needed in order to simplify the inverse kinematics problem. And, whenever there were two solutions for a variable, the closest one to the prior value was chosen.

One of the changes was the re-design of the link attached to the base. This change allowed the variable which is the distance of the axis and along axis. Since , the inverse kinematics solution was simplified and it was easier to find an analytical result.

|  |  |
| --- | --- |
| Z  ***X*** | Z  ***X*** |

Figure 4 – Adjustment in the design of the first link to align the frames zero and one.

Now, with the inverse kinematics solved, it was possible to set a path for the robot to execute a motion.

Using path planning, the robotic arm was programmed to execute two different tasks.

The first task is drawing a sphere. This gives an idea of welding. The end-effector aims the sphere’s center the whole time while performs the task. The Jacobian Matrix, which is used to transfer Cartesian to spherical coordinates, was used to calculate the end-effector’s orientation.

As the end-effector moves vertically along the sphere, the number of point in that circumference changes due to resolution issues. When the circumference is located near pole, the number of points is smaller. However, for those near the midst part, the number is greater and constant.

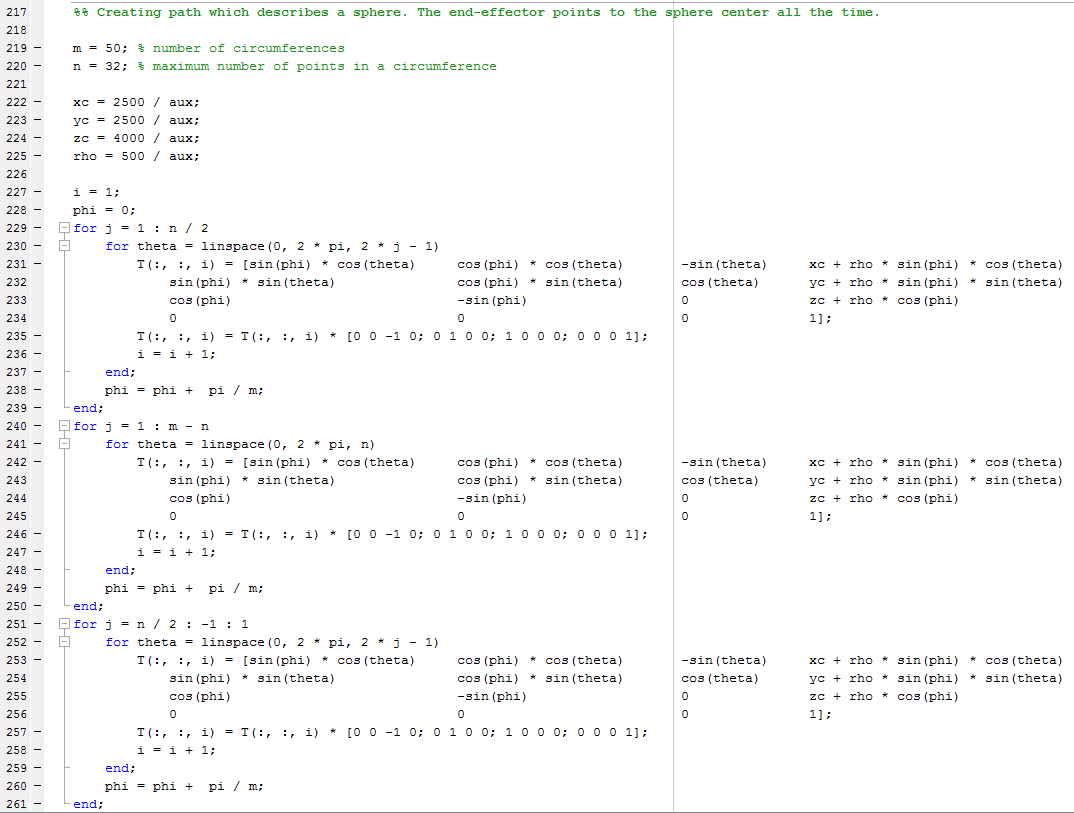
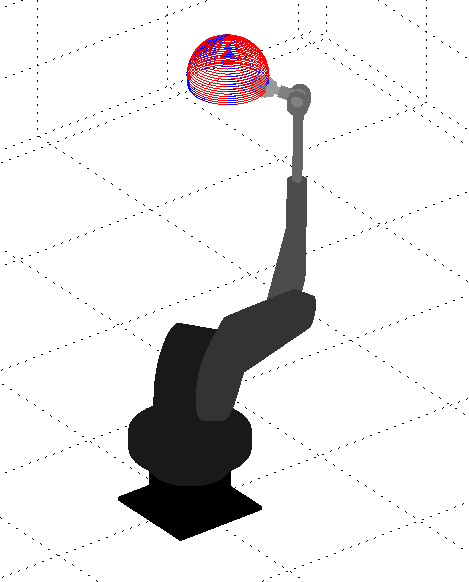


Figure 5 – Robotic arm performing the first task and its code.

The second task uses path planning as well. Now, the arm draws the phrase “THAT’S ALL FOLKS!” in the space. Then, it draws an ellipse around the phrase.

A function *phrasePath()* was created separately with each letter’s path. A string “letter” was declared to keep the phrase. Then, the function would draw each letter of the phrase according with the path created inside the function *phrasePath()*.

The end-effector aims to the phrase to draw it in the space. Considering each letter squared, the arm steps back after each movement (i.e. after draw the vertical line of the letter “L”, it steps back before start drawing the horizontal line).

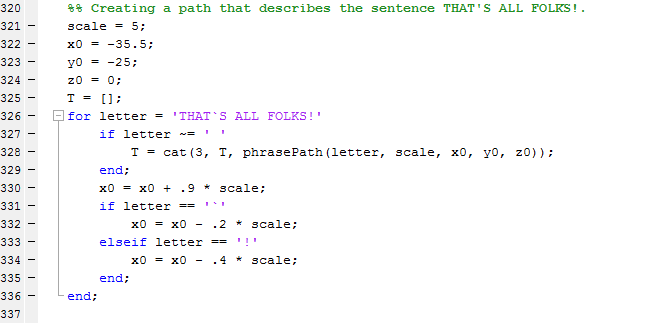
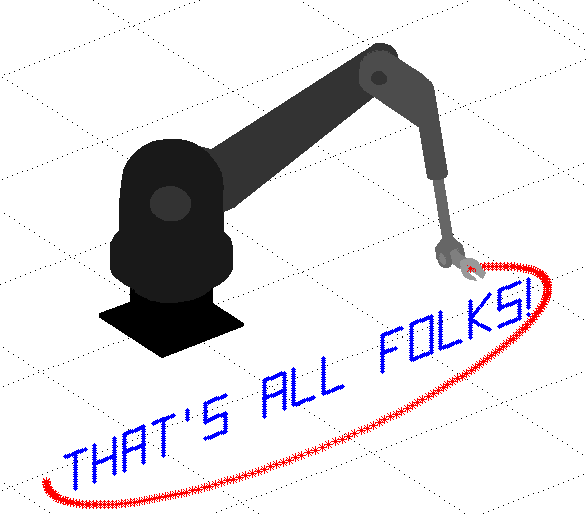


Figure 6 – Robotic arm performing the second task and its code.

As part of the project and using class notes, the *Jacobian* was calculated to plot the output velocity of each link. The following figure shows the respective velocity and orientation of each link.

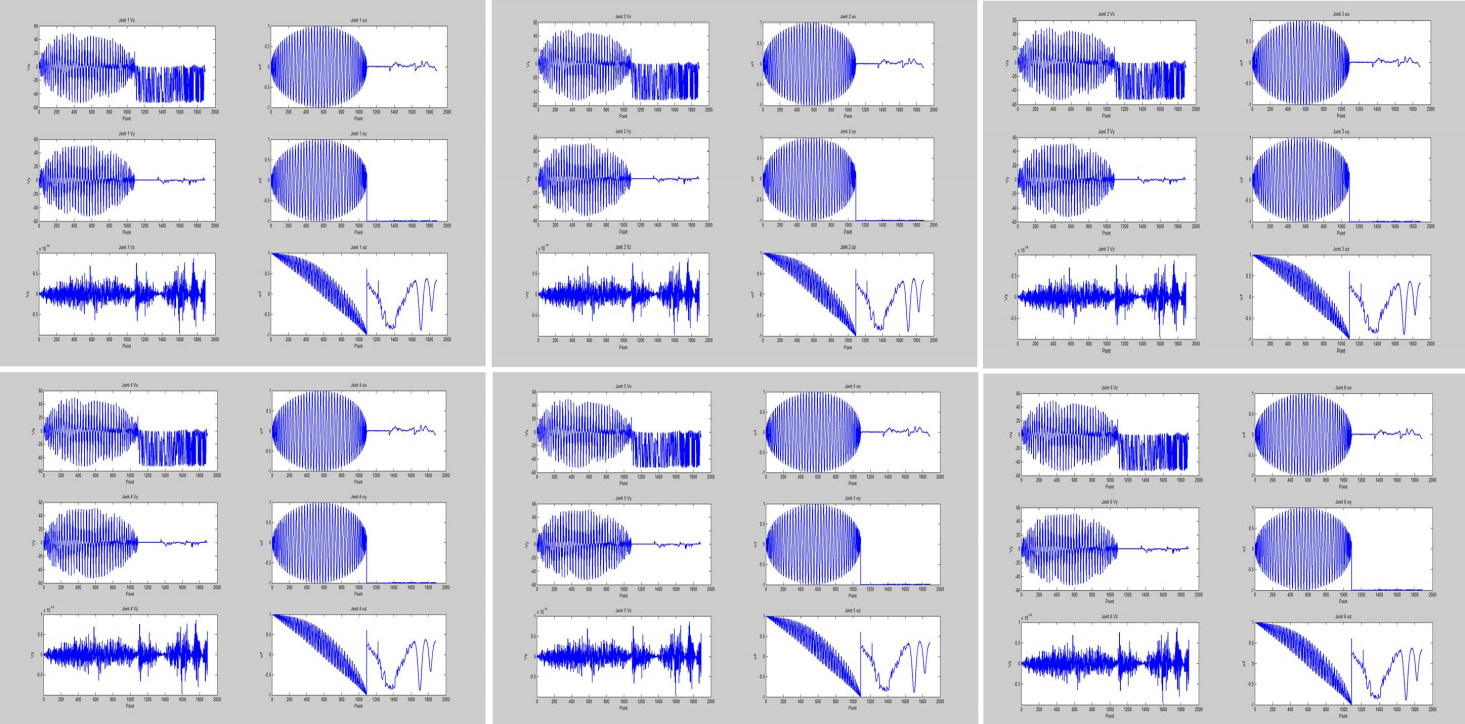


Figure 7 – Linear and angular velocity of each link.

Finally, we found various problems during the project’s execution and planning but everything went quite well. It is important to say that we learn and got a whole different perspective of robotics. Also, we had improved our MATLAB programming skills and designing ability using 3D CAD Design Software SolidWorks during this project.