The 8th Science and Technology Symposium for OISP Student, 19 June 2021
Ho Chi Minh City University of Technology (HCMUT), VNU-HCM

Research on modelling and trajectory planning for a 5-DOF robot manipulator

Nguyen Quoc Thinh[1,2], Duong Xuyen Han[1,2], Dao Minh Nguyen[1,2], Phung Tri Cong[1]

[1]Faculty of Mechanical Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam

[2]Office for International Study Programs, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam

*Corresponding author: ptcong@hcmut.edu.vn

INTRODUCTION

This paper presents the development of a Graphic User Interface (GUI) in MATLAB to simulate and control a 5-DOF robotic arm through communication between a microcontroller unit and Matlab. Firstly, kinematic characteristics of the manipulator are analyzed, and the forward kinematics equations are established using Denavit-Hartenberg (D-H) method. Secondly, the working space is visualized in MATLAB and the inverse kinematics equations are derived using algebraic and geometric methods. Thirdly, velocity kinematics "Jacobian" and trajectory planning of the robotic arm are developed to analyze and control the robot's motion. Finally, the manipulator's movement is both simulated in MATLAB and tested on the real model to exhibit the effectiveness of the proposed method.

***Acknowledgement**: This research is funded by Office for International Study Programs (OISP), Ho Chi Minh City University of Technology (HCMUT), VNU-HCM under grant number* **(SVOISP-2020-CK-66)**. *We acknowledge the support of time and facilities from HCMUT, VNU-HCM for this study.*

METHODS AND IMPLEMENTATION

To simulate the robot motion, 3D CAD of the model was built in Solidworks and imported into MATLAB. Then, the working range (Fig. 1) is defined by setting the limits of all joints.

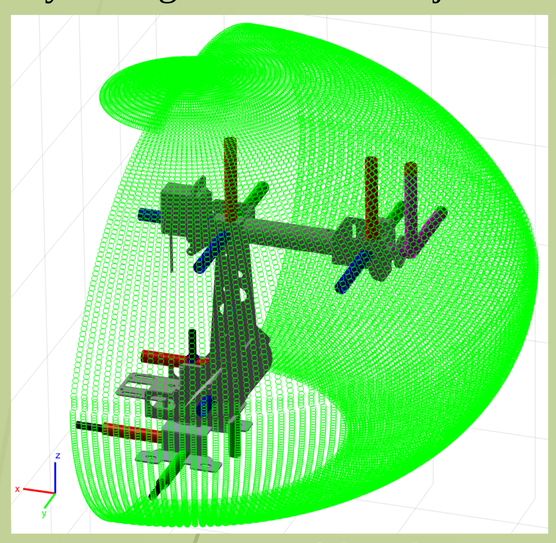


Fig. 1. Workspace of the robot arm

The GUI (Fig. 3) was developed by MATLAB programming to compute the forward, inverse kinematics, Jacobian and trajectory planning of the 5-DOF robot arm model. The motional simulation of the robot arm was also included of GUI to show the motion of model based on the theoretical analysis presented in this paper.

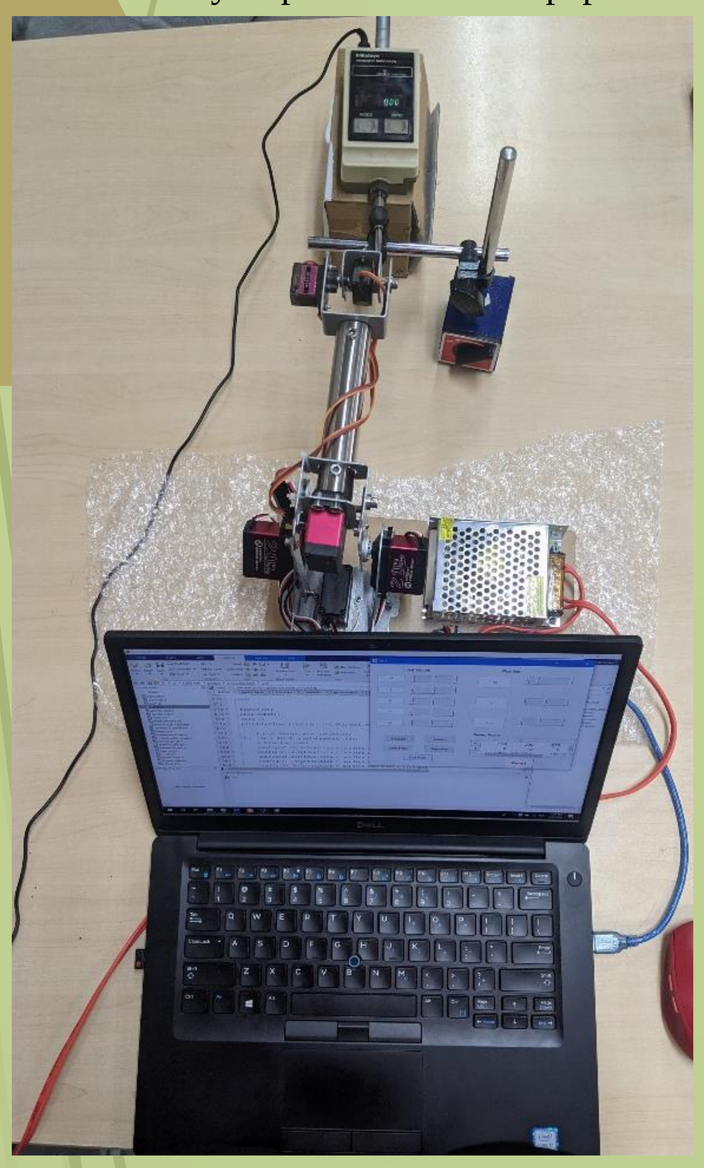


Fig. 2. Robot arm model experiment setup

SIMULATION AND EXPERIEMENTAL RESULTS

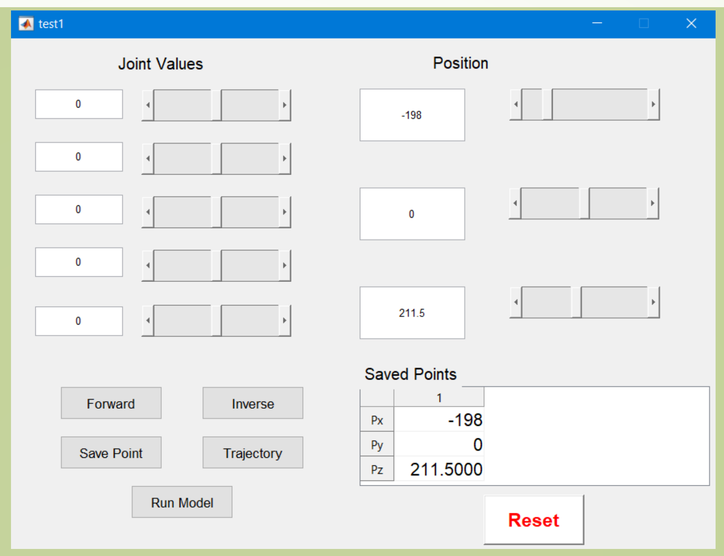


Fig. 3. Graphical User Interface

Firstly, user can adjust five 'Joint Values' in the boxes or sliders to achieve the end-effector position of the robot arm through forward kinematics, and the 'Forward' button is using for plotting the pose of the 3D robot.

Secondly, user can find the values of the joints at a specific end-effector position by setting the three 'Position' values in the boxes or sliders, the 'Inverse' button is used to show all the possible solutions of the inverse kinematics on the MATLAB Workspace area.

Thirdly, user able to generate trajectory through points by using 'Save Point' button to store the assigned points on 'Saved Points' table and 'Trajectory' button is for plotting the robot's motion and showing its end-effector Cartesian velocity v_x, v_y, v_z in a time scaling (Fig. 4).

Finally, the robot model can be controlled through connection between MATLAB and Arduino (Fig. 2), user can directly control in forward kinematics, inverse kinematics or create path for the end-effector to follow.

In order to verify the accuracy and precision of the model, we used a digimatic indicator to determine the errors of the end-effector in x, y, z axis. By controlling the robot doing the same task through MATLAB multiple times, and measure the errors in Cartesian coordinate system, we conclude that the error of the end-effector is $\pm 0.5mm$ in three directions.

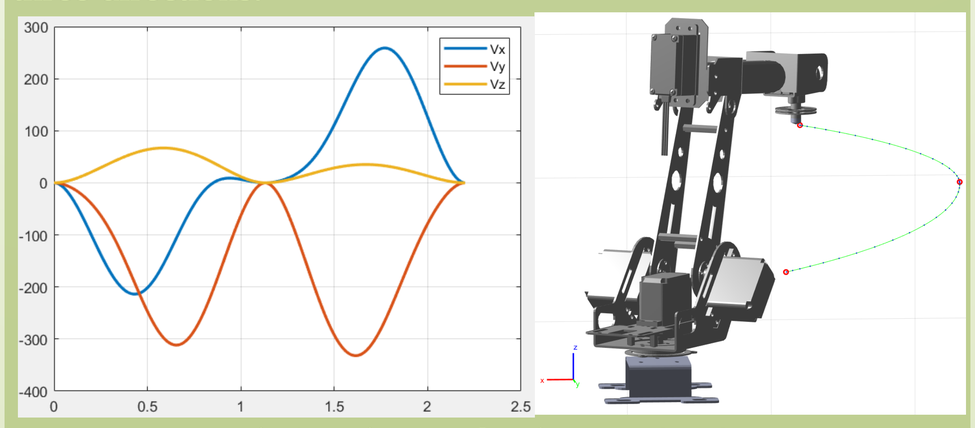
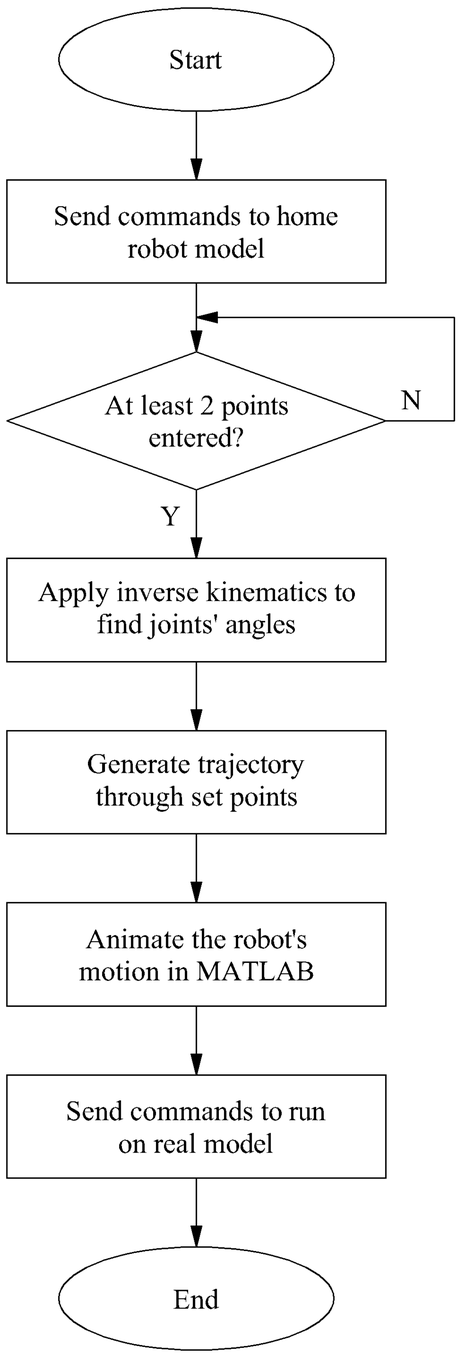


Fig.4. Trajectory planning on MATLAB

FLOWCHART

Flowchart for MATLAB program



CONCLUSIONS

In this research mathematical modeling and kinematic analysis of our robot model was developed and validated using MATLAB. Kinematics model was achieved with Denavit-Hartenberg (D-H) method. Kinematics and Jacobian solutions were fed into a trajectory planner using MATLAB GUI. The algorithm was implemented on the real robot through the connection between MATLAB and Arduino to verify simulation results.