MONITORING THE END-EFFECT POSITION OF THE THREE-JOINT ROTARY MANIPULATOR IN MATLAB SIMMECHANICS USING THE PID ALGORITHM

Nguyen Van Tan(1), Xuan Vinh Ha(2)

(1) School of Engineering - Technology, Thu Dau Mot University
(2) Faculty of Electric - Electronic Engineering, Hue Industrial College
Corresponding author: tannv@tdmu.edu.vn

DOI: 10.37550/tdmu.EJS/2024.01.524

Article Info

Volume: 6
Issue: 01
March 2024
Received: Dec. 28th, 2023
Accepted: Feb. 5th, 2024
Page No: 122-135

Abstract

Industrial robots have become one of the effective support tools for human labor. Robots are a solution to replace humans in repetitive tasks and in environments where humans cannot work. Robots have become one of the factors responding to the Industrial Revolution 4.0. Automatic control devices require high-precision control quality. Therefore, in this paper, we focus on researching controlling the position of the actuator accurately based on the PID algorithm. First, we study the forward and inverse kinematics of a three-joint robot. Second, we design the robot model on inventor software and transfer the 3d model in inventor software to Matlab Simmechanics. Third, modeling robot model on Simulink to simulate and evaluate the results achieved.

Keywords: matlab, PID controller, robot, simmechanics

1. Introduction

With the development of science, technology, and information technology as well as meeting the 4.0 industrial revolution to improve production productivity. Therefore, developing robotic systems, and improving accuracy and responsiveness to controllers is extremely necessary. Therefore, many researchers have created a lot of analysis software, determining robot coordinate systems, and calculating forward kinematics, and inverse kinematics of robots like the authors. (Ambuja Singh et al., 2016; Chittawadigi et al., 2011; Othayoth et al., 2017; Rajeevlochana et al., n.d.; Singh et al., 2016) used roboanalyzer software to analyze, Calculate and evaluate the performance of various types of robots. Authors (Ambuja Singh et al., 2016; Singh et al., 2016) The authors used RoboAnalyzer to simulate forward and inverse kinematics and illustrate and prove the correctness and reasonableness of the models. The authors (Chittawadigi et al., 2011; Rajeevlochana et al., n.d.) applied RoboAnalyzer to analyze the inverse and forward dynamics of a general series controller. The important contribution of this paper is to develop algorithms using object-oriented modeling methods and recursive formulations based on the Decoupled Natural Orthogonal Complement (DeNOC). The authors selected the KUKA KR5 Robot to simulate and collecte data and verified against the results obtained using the Dynamic Simulation module of Autodesk Inventor. The author (Othayoth et al., 2017) and his colleagues used RoboAnalyzer software to represent Denavit Hartenberg (DH) parameters to determine the structure of the robot and model the movement characteristics of the mechanism and robot behavior. The benefits of using RoboAnalyzer to overcome some of the challenges of learning about robotics in a classroom setting will also be discussed. The author

(Krisbudiman et al., 2021) and co-authors used RoboAnalyzer software to teach programming as an interactive way to program industrial robots. It involves the use of a handheld control device called a teach pendant that can be used to control the movement of the robot. It provides a very convenient method to teach trajectories to robots. One of the advantages of teaching pendant programming is that the operator does not need to learn any special programming language. Therefore, the teaching pendant can also be used as an effective tool in educational robotics. These devices are usually proprietary and work for a specific robot. A combination of matlab software and RoboAnalyzer software to analyze and simulate robot dynamics of the author team (Krisbudiman et al., 2021). Forward and inverse kinetic calculations were performed on Matlab software. Then, use RoboAnalyzer as a robot arm simulator to verify the calculation results. In addition, there are also several authors using MapleSim software to analyze and calculate and simulate the kinematics and dynamics of robots such as (C. C. et al., 2020) used the software to demonstrate the development of the 3-Link manipulator. The design parameters are randomly selected and symbolic tools in Maplesim are used to model the robot arm. Model simulations were performed, and the values using the Proportional Integral Derivative (PID) controller were adjusted, and the robot movement effect was displayed graphically. (Tan N.V, n.d.) used numerical simulation software to show the results. This work provides a potential basis for realizing robotic arms in the fields of industry, education and research, which is of great significance in improving production efficiency as well as supporting teaching and research in the field of robotics. The author (Gurel, 2018) applied MapleSim software to consider a 5-DOF manipulator for the mentioned multi-body simulation. The inverse kinematic equations determining the joint variables according to end-effect positions are derived using a geometric approach. Robots are mainly used for tasks such as picking and placing, welding, and painting. All of these tasks have one thing in common: they require the robot to follow a certain trajectory. To validate the model of the 5-DOF robot, the trajectory tracking mission was demonstrated. Meanwhile, the authors (Shaogang & Farah, 2017) utilized general-purpose presentation software for multi-body dynamics modeling and simulation in Maple and Maplesim. This method is illustrated by modeling a 2DOF Robot arm in Maple using the Newton-Euler formula construction algorithm. The drag-and-drop physical modeling tool in Maplesim is used to simulate the dynamics of the same robotic system. Results from both the Maple and Maplesim models are compared. The article is presented as a tutorial using Maple and Maplesim is a great environment for modeling and simulating multi-body systems. Moreover, a popular software called Simmechanics integrated with Matlab is used as a 3D simulation tool to collect robot simulation process data as the author (Yuan Shaoqiang et al., 2008) and colleagues used software to model and simulate an inverted pendulum. Simulation results of the SimMechanics physical model and the mathematical model for a simple inverted pendulum are compared. Furthermore, the full-state feedback controller is designed to meet performance requirements. It turns out that SimMechanics can be used for nonlinear systems and unstable robotics. The author (Mashali et al., 2020) and his colleagues designed a robot arm to perform packaging tasks. Kinematic and kinematic studies were performed for the 2R robot arm. The results of the kinematic study are that the angular displacement, angular velocity, and angular acceleration for each joint are determined and exported to the kinematic study to obtain torque and power consumption. Dynamic study was performed with the help of MATLAB, MATLAB/SimMechanics, and Solidworks were used to simulate and analyze the dynamics of the robot arm. The author (Nguyen et al., 2023) and co-colleagues utilized it to simulate problems related to kinematics and dynamics for an exoskeleton robot arm with 5 degrees of freedom (DoF). Graphical simulation takes advantage of the SolidWorks, and Catia design software as well as the calculation and simulation capabilities of the SimMechanics Toolbox in Matlab. The core of the proposed graphical simulation is an algorithm to solve the kinematics and kinematics problems of a developing upper limb rehabilitation robot. The authors used the proposed optimization-based algorithm to solve the inverse kinematics (IK) problem for the backup robot model. The end-effect trajectory is imported from measurement data. The joint variable solutions obtained before entering the dynamics problem have been smoothed to ensure feasibility in later calculations. A process for solving inverse dynamics problems using physical models by combining the power of SolidWorks and SimMechanics software is also proposed. This process ensures that the Robot's design can be changed and updated to calculate kinematics quickly and easily. To evaluate this procedure, we also compare these kinetic results with those when applying the Lagrange–Euler formula. All these calculation and simulation processes have been integrated into graphical simulation software to demonstrate efficiency and user-friendliness. Together with the author (Bahani et al., 2023) and colleagues used Simmechanics software to evaluate the inverse kinematics of a two-robot collaborative system using artificial intelligence and simulation tools. based on Matlab/SimMechanics, specifically the Levenberg-Marquardt (LM) optimization method. Therefore, an artificial neural network (ANN) is built that will replace the controller of a six-degree-of-freedom (6-DOF) collaborative robot. Therefore, the entire collaborative system was designed on SolidWorks, taking into account all the dimensions needed for the kinematic model, which was then converted into Matlab/SimMechanics and thanks to model manipulation in this software , we will be able to extract the joints and operational data of the cooperative system in its workspace. The robot system's kinematics database is built on Matlab to train the ANN and deploy it in Matlab/SimMechanics. In summary, the advantages of multi-body simulation packages such as SimMechanics and MapleSim can be summarized under four headings. They improve design, reduce development time, lower costs, and reduce development risk.

Therefore, in this article, we study forward kinematics and inverse kinematics equations of the robot 3R and use Simmechanics software to analyze, simulate, evaluate results, and propose solutions for the system to obtain the end-effect position control desired result. Considering the model is impacted by noise at the joints is a future research goal to minimize the impact of the environment using algorithms. 2.1 Arm design

2. Modeling Robot Arm

2.1 Arm design

Considering robot arm with three rovolution joint (RRR or 3R) as shown in figure 1

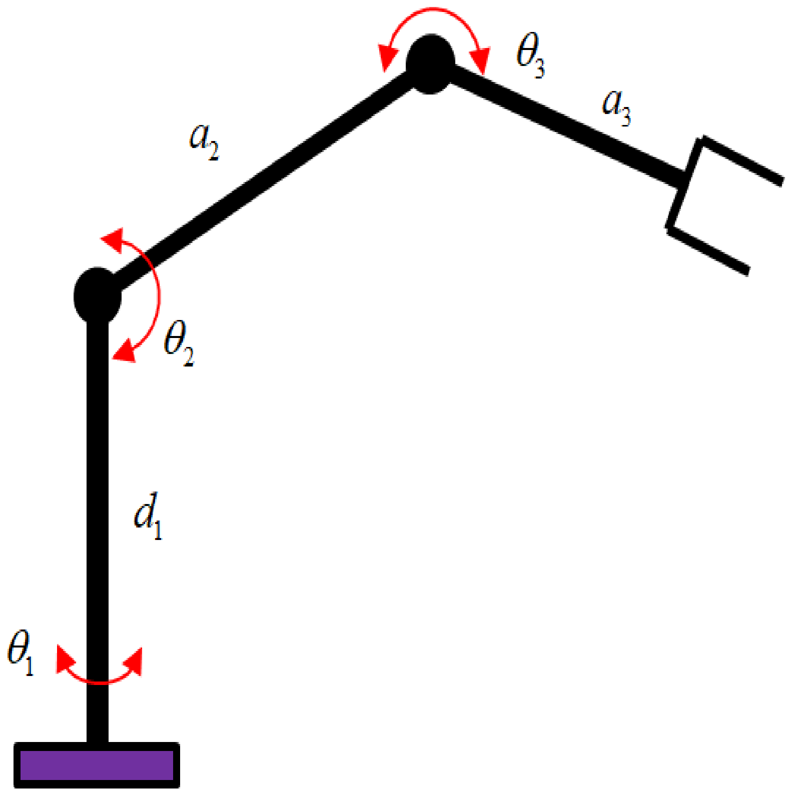


Figure 1. Scheme of the robot arm.

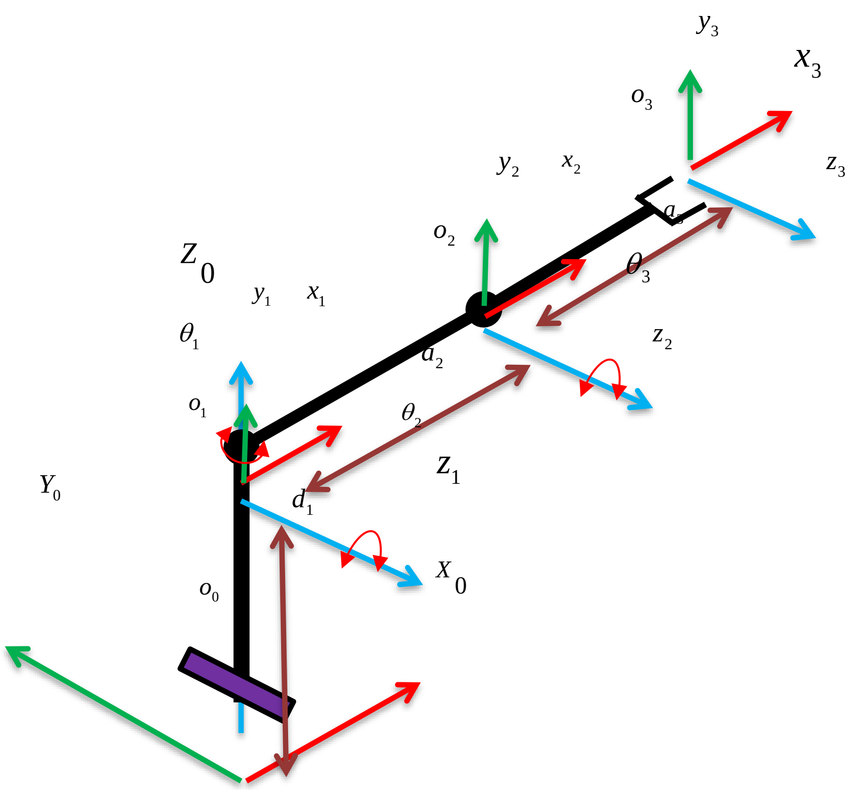


Figure 2. Scheme of coordinate of the robot arm.

2.2 Denavit- Hartenberg (DH)

After attaching the coordinate systems to the rotation joints of the robot as shown in Figure 2, we can set up the DH parameter table as shown in Table 1.

TABLE 1. DH parameter table of Robot 3R

Link	θ_i	d_i	a_i	α_i
1	θ_1	d_1	0	90^0
2	θ_2	0	a_2	0^0
3	θ_3	0	a_3	0^0

Where

$$d_1 = 250\,mm\,;\; a_2 = 207.5\,mm\,;\; a_3 = 180mm$$

Apply the formula for converting homogeneous matrices between coordinate systems according to DH law as follows

$$^{i-1}A_i = rotz(\theta_i)\times translz(0,0,d_i)\times translx(a_i,0,0)\times rotx(\alpha_i)$$

$$= \begin{bmatrix} C_i & -C_{\alpha_i}S_{\theta_i} & S_{\alpha_i}S_{\theta_i} & a_iC_{\theta_i} \\ S_i & C_{\alpha_i}C_{\theta_i} & -S_{\alpha_i}C_{\theta_i} & a_iS_{\theta_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Trong đó

$C_i = \cos(\theta_i)$; $S_i = \sin(\theta_i)$ với $i = 1,2,3...$ the i^{th} of the joint

Coordinate system conversion matrix between joint 1 and original coordinate system

$$^{0}A_1 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Similarly, the matrix converts the coordinate system between joint 2 to the coordinate system of joint 1

$$^{1}A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

And finally the matrix converts the coordinate system between joint 3 to the coordinate system of joint 2

$$^{2}A_3 = \begin{bmatrix} C_3 & -S_3 & 0 & a_3C_3 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where

$$S_1 = \sin(\theta_1);\ C_1 = \cos(\theta_1); S_2 = \sin(\theta_2);\ C_2 = \cos(\theta_2); S_3 = \sin(\theta_3);\ C_3 = \cos(\theta_3);$$

The homogeneous transformation matrix of the final actuator of the robot arm is calculated as follows

$$\begin{aligned} ^{0}A_3 &= {}^{0}A_1\, {}^{1}A_2\, {}^{2}A_3 \\ &= \begin{bmatrix} C_{23}C_1 & -S_{23}C_1 & S_1 & C_1\left(a_3C_{23} + a_2C_2\right) \\ C_{23}S_1 & -S_{23}S_1 & -C_1 & S_1\left(a_3C_{23} + a_2C_2\right) \\ S_{23} & C_{23} & 0 & d_1 + a_3S_{23} + a_2S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{5}$$

Where,

$$C_{23} = \cos(\theta_2 + \theta_3)\ ;\ S_{23} = \sin(\theta_2 + \theta_3)$$

The end-Effect position vector of the robot obtained as

$$\begin{cases} p_x = C_1\left(a_3C_{23} + a_2C_2\right) \\ p_y = S_1\left(a_3C_{23} + a_2C_2\right) \\ p_z = d_1 + a_3S_{23} + a_2S_2 \end{cases} \tag{6}$$

With $p = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$ is the position vector of the end-Effect

2.3 Inverse Kinematics

We define the position and orientation matrix of the end-effect for the robot arm as

$$^{0}T_4 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

In which

$n = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T$; $o = \begin{bmatrix} o_x & o_y & o_z \end{bmatrix}^T$ và $a = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T$ are vectors that determine the direction of the end-Effect

The kinematic equation of the RRR robot has the form

$$T_3 = {}^0A_1 \, {}^1A_2 \, {}^2A_3 \tag{8}$$

From equation (7), we multiply the inverse matrices of the transformation matrices between the coordinate systems of the joints, we have the following equations:

$$\left({}^0A_1\right)^{-1} T_3 = {}^1T_3 \tag{9}$$

From equation (9), the left side of the equation has the form

$$\left({}^0A_1\right)^{-1} T_3 = \begin{bmatrix} n_xC_1 + n_yS_1 & o_xC_1 + o_yS_1 & a_xC_1 + a_yS_1 & p_xC_1 + p_yS_1 \\ n_z & o_z & a_z & p_z - d_1 \\ n_xS_1 - n_yC_1 & o_xS_1 - o_yC_1 & a_xS_1 - a_yC_1 & p_xS_1 - p_yC_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

the right side of the equation obtains as

$${}^1T_3 = \begin{bmatrix} C_{23} & 0 & -S_{23} & a_2C_2 - d_3S_{23} \\ S_{23} & 0 & -C_{23} & a_2S_2 + d_3C_{23} \\ 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

Considering element (3,4) of the matrix in equation (10) and element (3,4) of the matrix in equation (11), we have

$$p_xS_1 - p_yC_1 = 0 \tag{12}$$

Thus, the angle θ_1 can be obtained as

$$\theta_1 = a\tan 2\left(p_y, p_x\right) \tag{13}$$

Moreover, from (6) Eq, we obtained as

$$p_x^2 + p_y^2 + \left(p_z - d_1\right)^2 = a_2^2 + 2a_2a_3C_3 + a_3^2 \tag{14}$$

Thus, from (14) Eq, we have

$$C_3 = \frac{\Delta - a_2^2 - a_3^2}{2a_2a_3} \tag{15}$$

With

$$\Delta = p_x^2 + p_y^2 + \left(p_z - d_1\right)^2$$

And

$$S_3 = \pm\sqrt{1 - \left(\frac{\Delta - a_2^2 - a_3^2}{2a_2a_3}\right)^2} \tag{16}$$

Finally, we can calculate the θ_3 angle as

$$\theta_3 = a\tan 2\left(\pm\sqrt{1-\left(\frac{\Delta - a_2^2 - a_3^2}{2a_2a_3}\right)^2}, \left(\frac{\Delta - a_2^2 - a_3^2}{2a_2a_3}\right)\right) \tag{17}$$

To compute the θ_2 angle, we can expand the (6) Eq as follows

$$C_1P_x + S_1P_y = \left(a_2 + d_3C_3\right)C_2 - d_3S_3S_2 \tag{18}$$

and

$$P_z - d_1 = \left(a_2 + d_3C_3\right)S_2 + d_3S_3C_2 \tag{19}$$

Solving the system of equations (18) and (19) we obtained

$$S_2 = \frac{\left(P_z - d_1\right)\left(a_2 + d_3C_3\right) - \left(C_1P_x + S_1P_y\right)d_3S_3}{d_3^2 + a_2^2 + 2a_2d_3C_3} \tag{20}$$

and

$$C_2 = \pm\sqrt{1 - S_2^2} \tag{21}$$

Finally, θ_2 get as

$$\begin{cases} \theta_2 = a\tan 2\left(S_2, C_2\right) \\ \theta_2 = a\tan 2\left(-S_2, C_2\right) \end{cases} \tag{22}$$

3. Simulation and Results

3.1 Modeling Robot RRR in Simulink

3.1.1 Robot 3R parameters Robot 3R Simulink

The 3R robot model is built in the inventor software as shown in figure 3

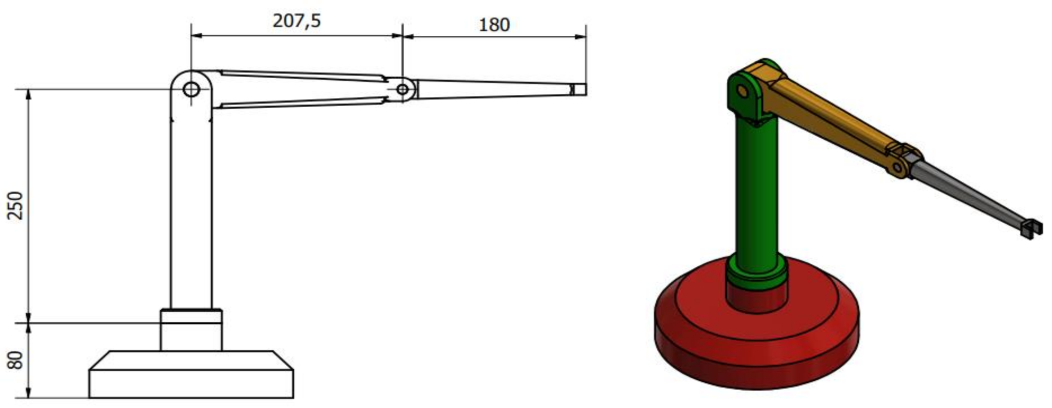


Figure 3. Robot 3R model

3.1.2 Robot 3R Simulink

The robot model to control position is modeled as follows:

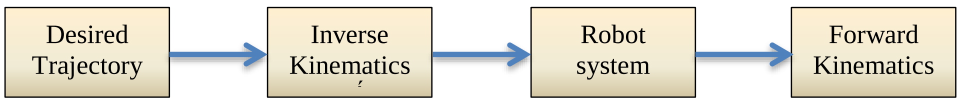


Figure 4. Robot position control diagram

From Figure 4, modeling of the Robot 3R is built in Matlab Simulink as below.

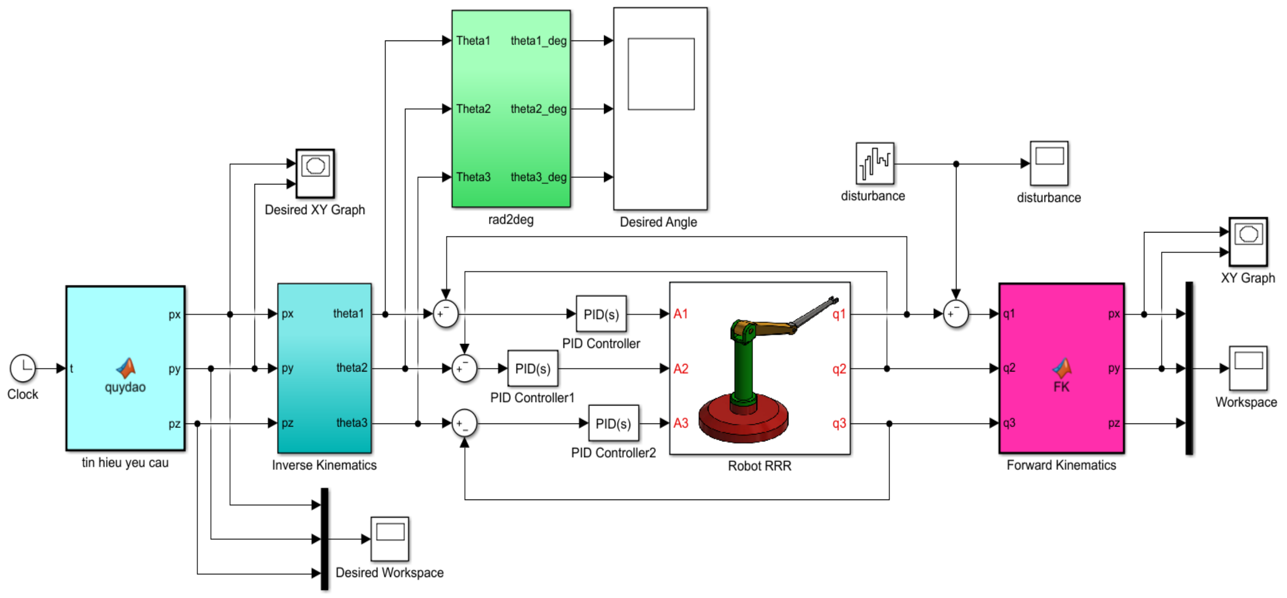


Figure 5. Robot position control diagram in Simulinks

Where:

The input trajectory is required in the form of equations as follows

$$\begin{cases} p_x = 30 + 20\cos(1.25\pi t - \pi / 6) \\ p_y = 30 + 20\sin(1.25\pi t - \pi / 6) \\ p_z = 100 + 20\cos(0.5\pi t) \end{cases} \quad (23)$$

Inverse kinematics is built from equations (13, 17 and 22) as shown

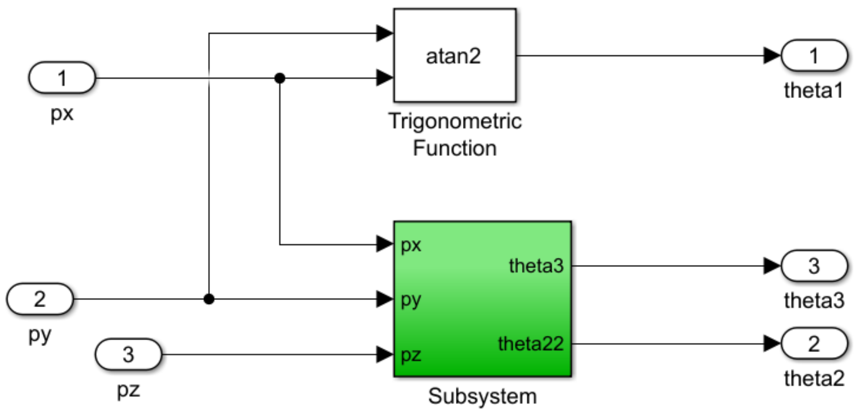


Figure 6. Inverse kinematics calculation diagram

Forward Kinematics is expressed by function of Toolbox Matlab as shown below

```
function [px,py,pz] = FK(q1,q2,q3)

d1=250;
a2=207.5;
d3=180;
t3=-q3;
px=cos(q1)*(a2*cos(q2)+d3*cos(q2+t3));
py=sin(q1)*(a2*cos(q2)+d3*cos(q2+t3));
pz=d1-(a2*sin(q2)+d3*sin(q2+t3));
```

The 3R robot model converted from the inventor software to the Matlab Simulinks environment is shown in Figure 7.

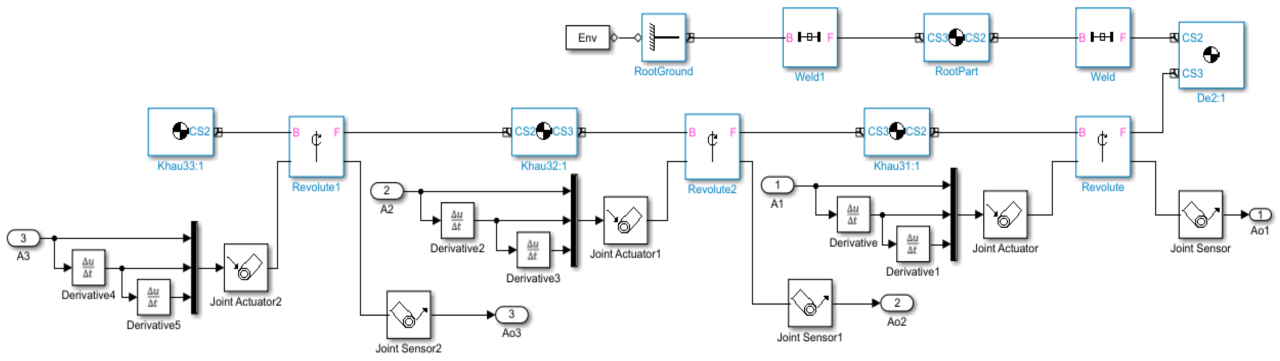


Figure 7. 3R robot model in Matlab Simulinks

3.2. Simulation of 3R Robot system

3.2.1 Simulation using PID controller for system without disturbance

With the desired coordinates given by equation (23), the angles are converted via the Inverse kinematics converter as shown in figure 8.

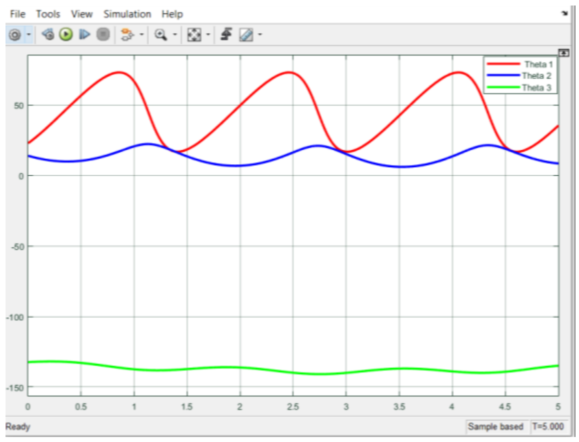
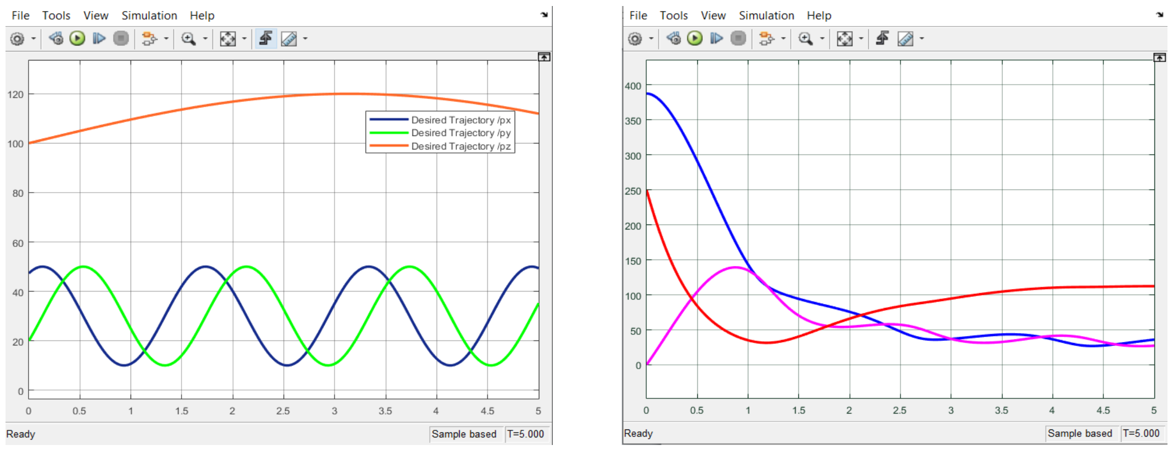


Figure 8. Joint angles converted by the Inverse kinematics converter

Using the tune method of Matlab Simulink, we can find the gain of the controllers as follows:

Controller 1: $k_p=0; k_i=75$
Controller 2: $k_p=0; k_i=70$
Controller 3: $k_p=0; k_i=55$

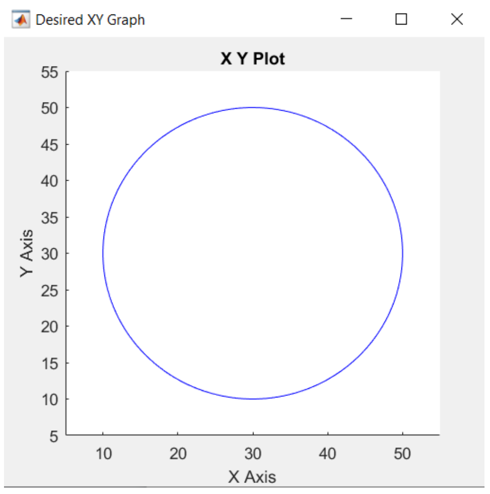
With these gains of the three PID controllers, the desired end-effect and the obtained end-effect coordinates are shown in Figure 9.



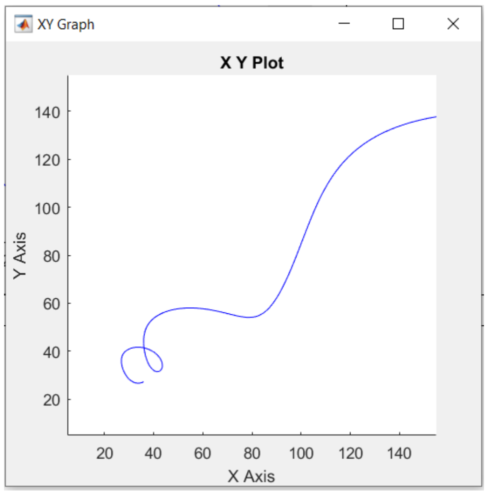
a) Desired end-effect coordinates *b) The obtained end-effect coordinates*

Figure 9. Desired end-effect coordinates compared to the obtained end-effect coordinates

The simulation results of desired end-effect coordinates compared to the obtained end-effect coordinates are depicted in 2D as shown in Figures 10a and 10b.



a) *Desired end-effect coordinates*



b) *The obtained end-effect coordinates*

Figure 10. *The feedback position signal is compared with the desired signal position in 2D*

The resulting desired end-effect coordinates is depicted in 3D as shown in Figure 11.

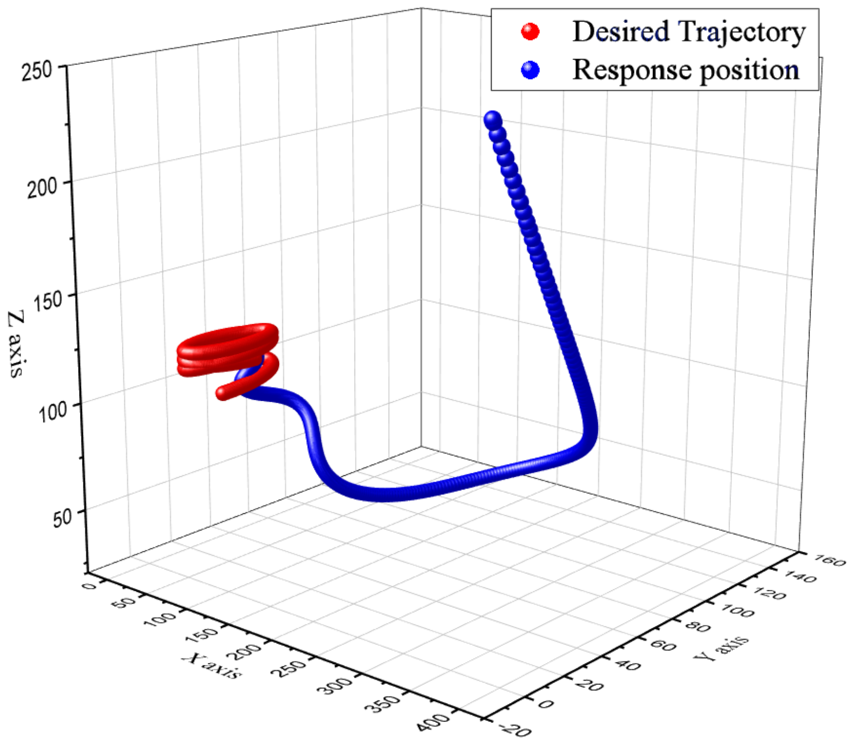


Figure 11. *The desired end-effect coordinates is compared to the obtained end-effect coordinates in 3D*

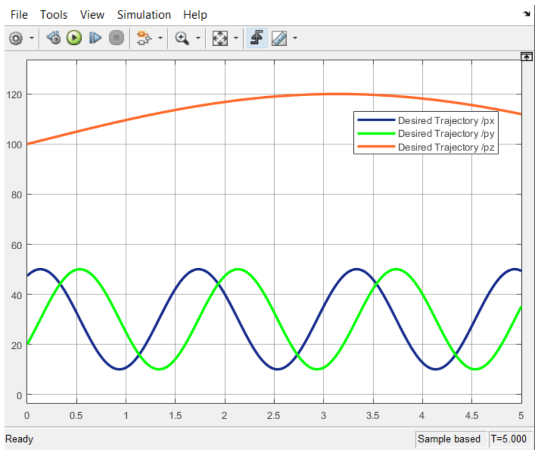
The feedback signal does not reach the desired signal, so we have to re-tune the gain values using Matlab Simulink's tune, we can find the gain of the controllers as follows:

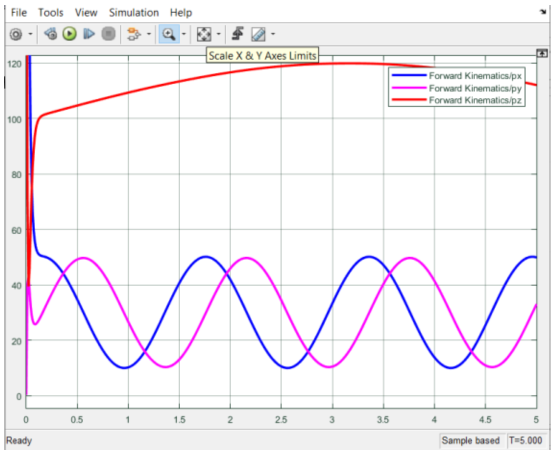
Controller 1: $k_p=0.1$; $k_i= 1986.65$

Controller 2: $k_p=0.15$; $k_i= 2501.05$

Controller 3: $k_p=0.025$; $k_i= 2618.92$

With these gains of the three PID controllers, the desired end-effect and the obtained end-effect coordinates are shown in Figure 12

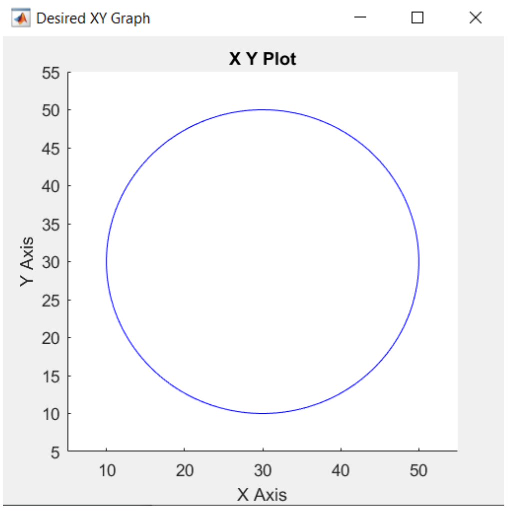


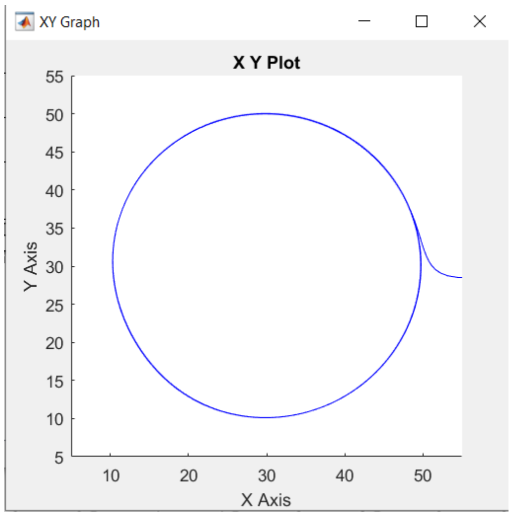


a) *Desired end-effect coordinates* b) *The obtained end-effect coordinates*

Figure 12. Desired end-effect coordinates compared to the obtained end-effect coordinates

The simulation results of desired end-effect coordinates compared to the obtained end-effect coordinates are depicted in 2D as shown in Figures 13a and 13b





a) *Desired end-effect coordinates* b) *The obtained end-effect coordinates*

Figure 13. The feedback position signal is compared with the desired signal position in 2D

The resulting desired position is depicted in 3D as shown in Figure 14.

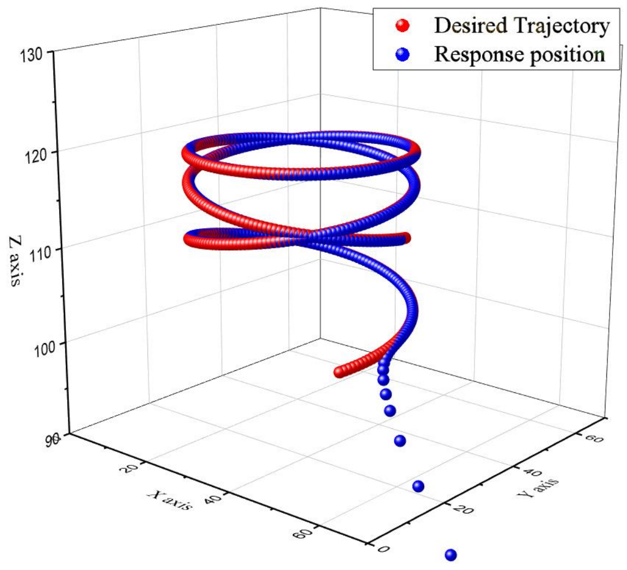
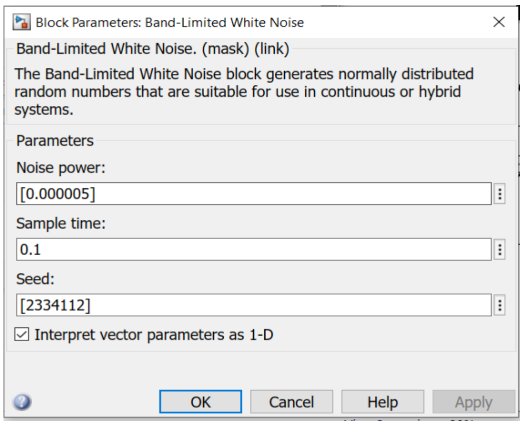


Figure 14. *The desired end-effect coordinates is compared to the obtained end-effect coordinates in 3D*

3.2.2. Simulation using PID controller for system with disturbance

Suppose that the noise of the angle sensor is given by the Matlab toolbox as shown in Figure 15.


Block Parameters: Band-Limited White Noise

Band-Limited White Noise. (mask) (link)

The Band-Limited White Noise block generates normally distributed random numbers that are suitable for use in continuous or hybrid systems.

Parameters

Noise power:

[0.000005]

Sample time:

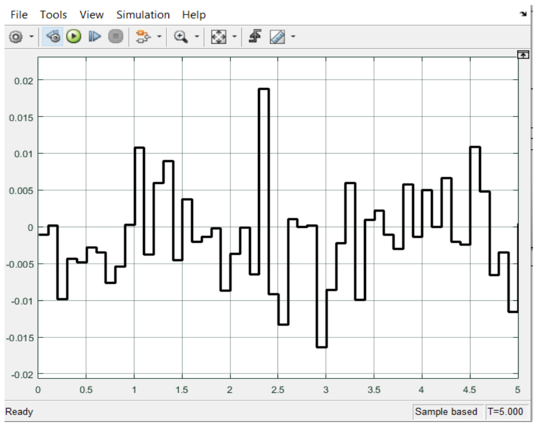
0.1

Seed:

[2334112]

☑ Interpret vector parameters as 1-D

OK Cancel Help Apply



a) White noise toolbox block *b) Signal white noise*

Figure 15. *Sensor disturbance*

Suppose that the robot system is subjected to disturbances at rotation joint 1 and 3 as shown in Figure 16.

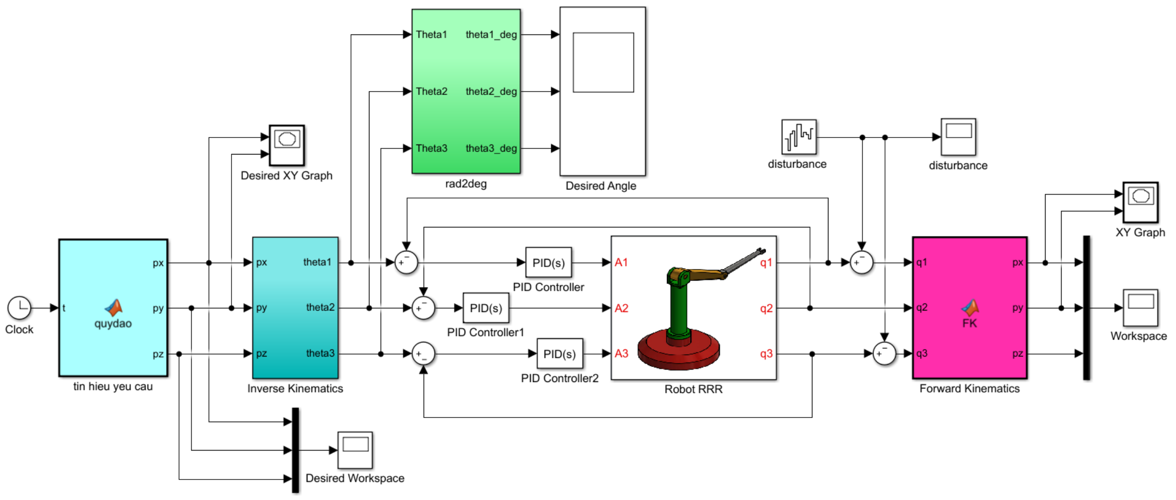
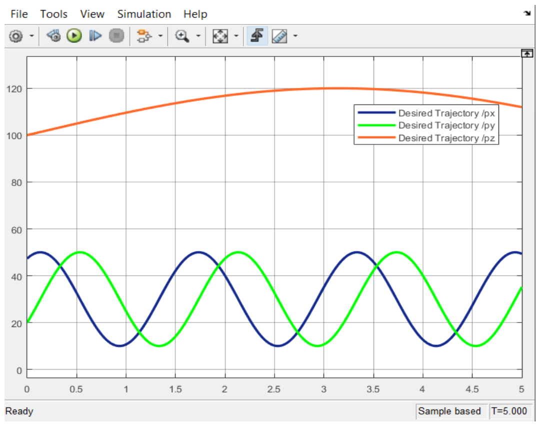
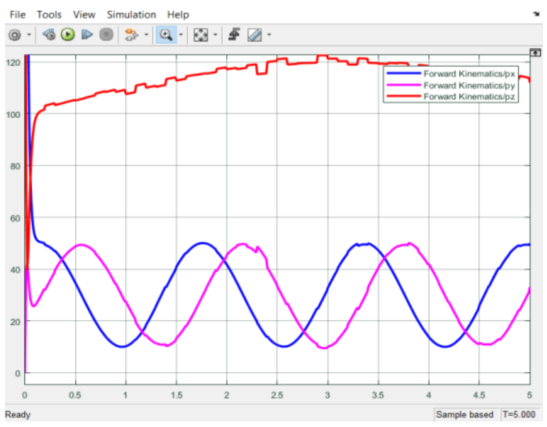


Figure 5. Robot position control diagram in Simulinks with disturbances at joints 1 and 3

With the gain values of 3 PID controllers (Controller 1: kp=0.1, ki= 1986.65; controller 2: kp=0.15, ki= 2501.05 and controller 3: kp=0.025, ki= 2618.92) then the desired end-effect coordinates is compared to the obtained end-effect coordinates in case the sensor system is disturbed as shown in Figure 16

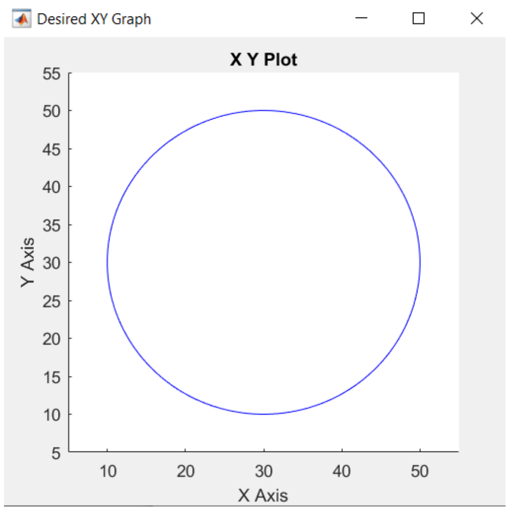




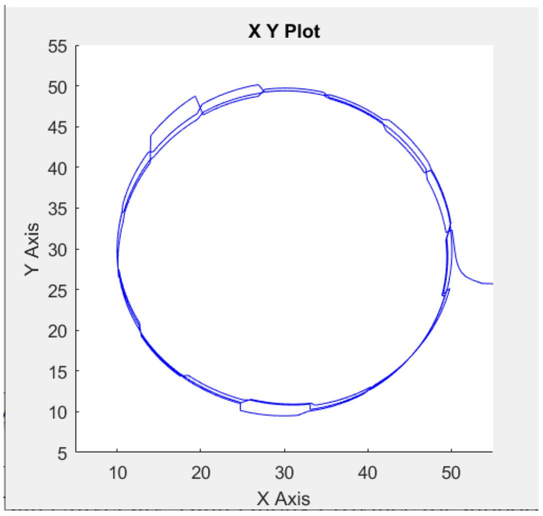
a) *Desired end-effect coordinates* b) *The obtained end-effect coordinates*

Hình 16. *Desired end-effect coordinates compared to the obtained end-effect coordinates with* disturbance case

The simulation results of desired end-effect coordinates compared to the obtained end-effect coordinates are depicted in 2D as shown in Figures 17a and 17b



a) *Desired end-effect coordinates*



b) *The obtained end-effect coordinates*

Figure 17. *Desired end-effect coordinates compared to the obtained end-effect coordinates in 2D when joint 1 and 3 are disturbance*

The resulting desired position is depicted in 3D as shown in Figure 18.

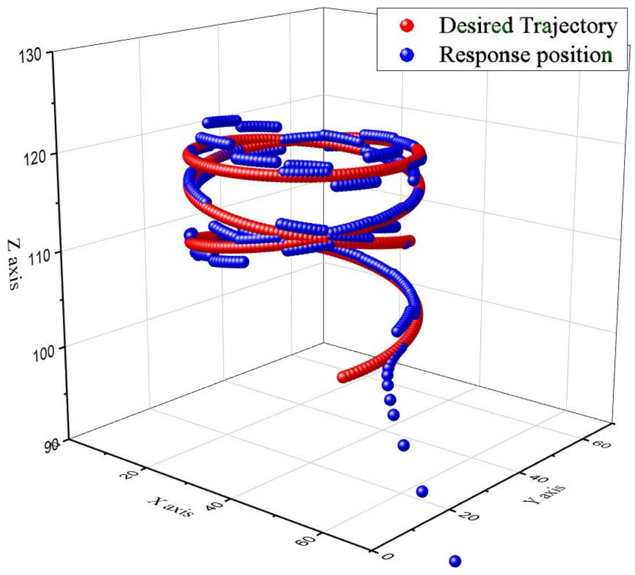


Figure 18. *The desired end-effect coordinates is compared to the obtained end-effect coordinates in 3D when joint 1 and 3 are disturbance*

4. Conclusion

Figures 9, 10, 11 show that the working system does not reach the desired value because the gain coefficients of the 3 PID sets are small. We see that with the Gain values of 3 PID controllers (Controller 1: *kp=0.1, ki= 1986.65*; controller 2: *kp=0.15, ki= 2501.05* and controller 3: *kp=0.025 , ki= 2618.92*), the obtained end-effect coordinates approach desired end-effect coordinates, which proves that the inverse kinematics unit works correctly. Figures 16, 17, 18 show that the system is affected by noise at joints 1 and 3 that also affects the control signal of joint 2. With the impact of this noise, the control quality is affected. end-effect coordinates is deviating from the desired signal. Therefore, to improve the control quality, we will implement control algorithms or error compensation algorithms to reduce the impact of noise in the next article.

Acknowledgements

Thanks to Thu Dau University, Binh Duong Province for supporting the project under grant number NNC.21.2.014.

Reference

Ambuja Singh, Ratna Priya Kanchan, Saakshi Singh, & IGDTUW. (2016). A Review Paper on Analysis and Simulation of Kinematics of 3R Robot with the Help of RoboAnalyzer. *International Journal of Engineering Research And*, *5*(04), IJERTV5IS040535. https://doi.org/10.17577/IJERTV5IS040535

Bahani, A., Ech-Chhibat, M. E. H., Samri, H., & Elattar, H. A. (2023). The Inverse Kinematics Evaluation of 6-DOF Robots in Cooperative Tasks Using Virtual Modeling Design and Artificial Intelligence Tools. *International Journal of Mechanical Engineering and Robotics Research*, 121–130. https://doi.org/10.18178/ijmerr.12.2.121-130

C. C., O., A. B., I., B. A., O., N. T., T., & O. R., O. (2020). Model Design of 3-Link Robotic Manipulator Using Maplesim. *International Journal of Scientific Research in Science and Technology*, 01–06. https://doi.org/10.32628/IJSRST207449

Chittawadigi, R. G., Jain, A., Shah, S., & Saha, S. (2011). *Recursive Robot Dynamics in RoboAnalyzer*. https://www.semanticscholar.org/paper/Recursive-Robot-Dynamics-in-RoboAnalyzer-Chittawadigi-Jain/e1842c710f6a7053c7371ce37e0680372e728903

Gurel, C. S. (2018). *Modeling and Robot Simulation of a 5-DOF Robot Manipulator in MapleSim and SimMechanics Environments*.

Krisbudiman, A., Hendro Nugroho, T., & Musthofa, A. (2021). Analysis Industrial Robot Arm with Matlab and RoboAnalyzer. *International Journal of Advanced Engineering, Management and Science*, *7*(3), 75–80. https://doi.org/10.22161/ijaems.73.10

Mashali, M., Addeif, M., & Embarak, M. (2020). SCARA ROBOT LINKS LENGTH OPTIMIZATION BY USING MATLAB AND VERIFICATION WITH SIMMECHANICS AND SOLIDWORKS. *INTERNATIONAL JOURNAL OF ADVANCES IN SIGNAL AND IMAGE SCIENCES*, *6*(2), Article 2. https://doi.org/10.29284/ijasis.6.2.2020.8-19

Nguyen, T.-T., Nguyen, T., Pham, H., & Bui, T. (2023). Proposing a Graphic Simulator for an Upper Limb Exoskeleton Robot. *Applied Bionics and Biomechanics*, *2023*, e2844202. https://doi.org/10.1155/2023/2844202

Othayoth, R. S., Chittawadigi, R. G., Joshi, R. P., & Saha, S. K. (2017). Robot kinematics made easy using RoboAnalyzer software. *Computer Applications in Engineering Education*, *25*(5), 669–680. https://doi.org/10.1002/cae.21828

Rajeevlochana, C. G., Jain, A., Shah, S. V., & Saha, S. K. (n.d.). *Recursive Robot Dynamics in RoboAnalyzer*.

Tan.V.N. (n.d.). Simulating 3r robot dynamics using imported cad in Maplesim. *Tạp chí khoa học Đại học Thủ Dầu Một*. Retrieved December 22, 2023, from https://ejs.tdmu.edu.vn/simulating-3r-robot-dynamics-using-imported-cad-in-maplesim-92-a25id.html

Shaogang, L., & Farah, E. (2017). Multibody Dynamics Modeling and Simulating Using Maple and Maplesim. *International Journal of Control and Automation*, *10*(5), 83–92. https://doi.org/10.14257/ijca.2017.10.5.08

Singh, A., Kanchan, R. P., Singh, S., & IGDTUW. (2016). A Review Paper on Analysis and Simulation of Kinematics of 3R Robot with the Help of RoboAnalyzer. *International Journal of Engineering Research And*, *5*(04). https://www.academia.edu/53755595/A_Review_Paper_on_Analysis_and_Simulation_of_Kinematics_of_3R_Robot_with_the_Help_of_RoboAnalyzer

Yuan Shaoqiang, Liu Zhong, & Li Xingshan. (2008). Modeling and simulation of robot based on Matlab/SimMechanics. *2008 27th Chinese Control Conference*, 161–165. https://doi.org/10.1109/CHICC.2008.4604913