

**TRƯỜNG ĐẠI HỌC BÁCH KHOA TP HỒ CHÍ MINH**

**BÀI TẬP LỚN**  
**LẬP TRÌNH MÔ PHỎNG VÀ GIẢI QUYẾT**  
**CÁC BÀI TOÁN CỦA ROBOT SCARA**

**NGUYỄN GIA KHIÊM**

1810236

**KỸ THUẬT ROBOT**

**Giảng viên hướng dẫn:** TS. Nguyễn Hoàng Giáp

**Bộ môn:** Kỹ thuật Điều khiển và Tự động hoá

**TP HỒ CHÍ MINH, 06/2021**

# MỤC LỤC

<b>CHƯƠNG 1. MỞ ĐẦU .....</b>	<b>1</b>
1.1 Tổng quan.....	1
1.2 Giới thiệu các công cụ sử dụng.....	1
1.2.1 Lập trình tính toán sử dụng MATLAB .....	1
1.2.2 Lập trình giao diện sử dụng GUIDE.....	1
1.2.3 Mô phỏng mô hình robot sử dụng công cụ Simscape.....	2
1.3 Giới thiệu về SCARA Robot.....	2
1.3.1 Tổng quan về SCARA Robot .....	2
1.3.2 Thông số của SCARA Robot trong đề tài.....	3
<b>CHƯƠNG 2. NỘI DUNG THỰC HIỆN .....</b>	<b>4</b>
2.1 Mô hình và cấu trúc lập trình của SCARA Robot .....	4
2.1.1 Mô hình Solidwork .....	4
2.1.2 Mô hình Simulink sử dụng Simscape .....	4
2.1.3 Cấu trúc lập trình SCARA Robot .....	5
2.1.4 Cấu trúc thư mục của đề tài .....	7
2.2 Động học thuận (Forward Kinematics).....	8
2.3 Động học ngược (Inverse Kinematics) .....	10
2.4 Hoạch định quỹ đạo đường đi (Path planning and Trajectory planning). 11	
2.4.1 Hoạch định vận tốc hình thang .....	11
2.4.2 Hoạch định đường thẳng.....	12
2.4.3 Hoạch định đường tròn 2D .....	12
2.5 Động học vi sai (Diffirential Kinematics).....	14
2.6 Động lực học (Dynamics) .....	15
2.7 Kết quả .....	16
<b>CHƯƠNG 3. ĐÁNH GIÁ KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN.....</b>	<b>22</b>
3.1 Đánh giá kết quả.....	22
3.2 Hướng phát triển .....	22
<b>CHƯƠNG 4. TÀI LIỆU THAM KHẢO .....</b>	<b>23</b>

## DANH MỤC HÌNH ẢNH

Hình 1.1 Model Robot SCARA T6 của EPSON .....	3
Hình 1.2 Thông số kỹ thuật của Model Robot SCARA T6 .....	3
Hình 2.1 Mô hình SCARA Robot T6 trong SolidWorks .....	4
Hình 2.2 Mô hình của SCARA Robot T6 trong MATLAB Simulink .....	4
Hình 2.3 Hình ảnh của robot trong Mechanics Explorers .....	5
Hình 2.4 Cấu trúc thư mục của đề tài .....	7
Hình 2.5 Xác định các joints của robot .....	8
Hình 2.6 Xác định các links của robot .....	8
Hình 2.7 Xác định các frames của robot .....	9
Hình 2.8 Cấu trúc của two-link planar arm .....	10
Hình 2.9 Đồ thị vị trí, vận tốc và gia tốc của hoạch định vận tốc hình thang .....	11
Hình 2.10 Đường tròn đi qua 3 điểm trong mặt phẳng .....	13
Hình 2.11 Mô tả hoạch định đường tròn trong hệ trục tọa độ .....	13
Hình 2.12 Thông số mô phỏng của động cơ .....	15
Hình 2.13 Bộ điều khiển PID cho các joints .....	16
Hình 2.14 Thông số PID cho động cơ .....	16
Hình 2.15 Tổng quan giao diện mô phỏng robot .....	16
Hình 2.16 Điều khiển các khớp robot bằng thanh trượt .....	17
Hình 2.17 Bài toán Động học thuận của robot .....	17
Hình 2.18 Bài toán Động học thuận của robot .....	18
Hình 2.19 Bài toán Động học ngược của robot .....	18
Hình 2.20 Bài toán Động học ngược của robot .....	19
Hình 2.21 Bài toán Hoạch định quỹ đạo đường đi cho robot .....	19
Hình 2.22 Đồ thị giá trị, vận tốc và gia tốc của các khớp xoay theo thời gian khi di chuyển theo quỹ đạo đã hoạch định trước. ....	20
Hình 2.23 Đồ thị quãng đường, vận tốc, gia tốc của đầu công tác theo thời gian .....	20
Hình 2.24 Đồ thị giá trị các khớp xoay khi có áp dụng động lực học (mô phỏng động cơ và bộ điều khiển PID) .....	21
Hình 2.25 Hoạch định quỹ đạo có sử dụng mô phỏng động lực học .....	21

## **DANH MỤC BẢNG**

Bảng 2.1 Bảng thông số DH của Robot SCARA.....	9
Bảng 2.2 Bảng tầm giá trị của các joint variables.....	9

# CHƯƠNG 1. MỞ ĐẦU

## 1.1 Tổng quan

Trong thời đại công nghiệp hoá, hiện đại hoá của Việt Nam và sự phát triển, thay đổi nhanh chóng trên toàn thế giới, việc áp dụng các công nghệ mới và máy móc vào các dây chuyền sản xuất là một phần rất quan trọng để có thể cắt giảm chi phí về lâu dài, nâng cao năng suất lao động và cải thiện chất lượng sản phẩm. Các robot công nghiệp đã được sử dụng nhiều trong nhà máy với các ứng dụng khác nhau như gấp hàng, lắp ráp linh kiện điện tử, hàn và cắt kim loại, lắp ráp ô tô,...

Để có thể sử dụng, lập trình một robot công nghiệp cần phải hiểu được những bài toán cơ bản trong robot như động học thuận, động học nghịch, hoạch định quỹ đạo, động lực học,... Vì lý do đó, mục đích của bài tập lớn là tìm hiểu kiến thức về robot, sử dụng các phần mềm hỗ trợ và thực hành mô phỏng để kiểm chứng.

## 1.2 Giới thiệu các công cụ sử dụng

### 1.2.1 Lập trình tính toán sử dụng MATLAB

MATLAB (viết tắt của " matrix laboratory ") là một ngôn ngữ lập trình riêng biệt và là một môi trường tính toán số được phát triển bởi MathWorks. MATLAB cho phép các thao tác ma trận, các chức năng và dữ liệu, thực hiện các thuật toán, tạo giao diện người dùng và giao tiếp với các chương trình được viết bằng ngôn ngữ khác. (<https://en.wikipedia.org/wiki/MATLAB>)

Mặc dù MATLAB được thiết kế chủ yếu cho tính toán số, một công cụ (tool-box) tùy chọn sử dụng công cụ biểu tượng MuPAD cho phép truy cập vào các khả năng tính toán tượng trưng. Đặc biệt, Simulink, một trình mô phỏng đa miền đồ họa và thiết kế dựa trên mô hình cho các hệ thống động và nhúng.

Tính đến năm 2020, MATLAB có hơn 4 triệu người dùng trên toàn thế giới. Người dùng MATLAB đến từ nhiều nền tảng khác nhau về kỹ thuật, khoa học và kinh tế.

### 1.2.2 Lập trình giao diện sử dụng GUIDE

GUIDE là một môi trường phát triển cung cấp một bộ công cụ để tạo giao diện người dùng (UI). Công cụ này đơn giản hóa quá trình thiết lập và lập trình giao diện người dùng.

GUIDE Layout Editor giúp ta có thể tạo giao diện người dùng bằng cách nhấp và kéo các thành phần giao diện người dùng chẳng hạn như hệ trục (axes), bảng điều khiển (panel), nút (button), văn bản (text), thanh trượt (slider),... Không những thế, ta có thể định kích thước giao diện người dùng, sửa đổi giao diện thành

phần, căn chỉnh các thành phần, đặt thứ tự tab, xem danh sách phân cấp của các đối tượng thành phần và đặt các tùy chọn giao diện người dùng.

GUIDE tự động tạo một file chương trình chứa các hàm MATLAB điều khiển hoạt động của giao diện người dùng. Các hàm callback là các hàm thực thi khi người dùng tương tác với một thành phần giao diện người dùng.

### **1.2.3 Mô phỏng mô hình robot sử dụng công cụ Simscape**

Simscape cho phép ta nhanh chóng tạo các mô hình hệ thống trong môi trường Simulink. Với Simscape, ta xây dựng các mô hình thành phần vật lý dựa trên các kết nối vật lý tích hợp trực tiếp với sơ đồ khối và các mô hình mô hình hóa khác.

Simscape giúp ta phát triển hệ thống điều khiển và kiểm tra hiệu suất cấp hệ thống. Ta có thể tạo các mô hình thành phần tùy chỉnh bằng cách sử dụng ngôn ngữ Simscape dựa trên MATLAB, cho phép tác giả dựa trên văn bản của các thành phần. Ta có thể tham số hóa mô hình của mình bằng cách sử dụng các biến và biểu thức MATLAB, đồng thời thiết kế hệ thống điều khiển cho hệ thống vật lý của bạn trong Simulink.

## **1.3 Giới thiệu về SCARA Robot**

### **1.3.1 Tổng quan về SCARA Robot**

Đề tài tập trung tìm hiểu về robot SCARA (hay còn được gọi là SCARA Robot). Robot SCARA là một robot rất phổ biến trong ứng dụng sắp xếp hàng hoá. Robot SCARA ra đời vào năm 1979 tại trường đại học Yamanashi (Nhật Bản) dùng trong việc lắp ráp. SCARA gồm hai khớp quay và một khớp tịnh tiến, nhưng cả ba khớp đều có trục song song với nhau. Kết cấu này làm cho tay máy cứng vững hơn theo phương thẳng đứng nhưng kém cứng vững hơn theo phương ngang. Loại này chuyên dùng trong công việc lắp ráp với tải trọng nhỏ theo phương thẳng đứng. Từ SCARA là viết tắt của chữ “Selective Compliance Articulated Robot Actuator”. Robot SCARA luôn là chủ đề được quan tâm nhiều trong lĩnh vực tự động hóa bởi tính ứng dụng thực tiễn của nó trong xã hội hiện nay nói chung và trong ngành công nghiệp nói riêng.

### 1.3.2 Thông số của SCARA Robot trong đề tài



Hình 1.1 Model Robot SCARA T6 của EPSON

		T6-602
Mounting type		Tabletop
Arm length	Arm #1, #2	600 mm
Weight (cables not included)		22 kg
Repeatability	Joints #1, #2	±0.040 mm
	Joint #3	±0.020 mm
	Joint #4	±0.020 deg
Max. motion range	Joint #1	±132 deg
	Joint #2	±150 deg
	Joint #3	200 mm
	Joint #4	±360 deg
Payload	Rated	2 kg
	Maximum	6 kg
Standard cycle time <sup>1</sup>		0.49 sec
Joint #4 allowable moment of inertia <sup>2</sup>	Rated	0.010 kg•m <sup>2</sup>
	Maximum	0.080 kg•m <sup>2</sup>
Joint #3 downward force		83 N
Electric lines		Hand I/O: IN6/OUT4 (D-Sub 15-Pin) / User I/O: IN18/OUT12
Pneumatic lines		Φ6 mm × 2, Φ4 mm × 1
Installation environment		Standard
Available controllers		Built-in
Safety standards		CE Mark: EMC Directive, Machinery Directive, RoHS Directive ANSI/RIA R15.06-2012 NFPA 79 (2007 Edition)

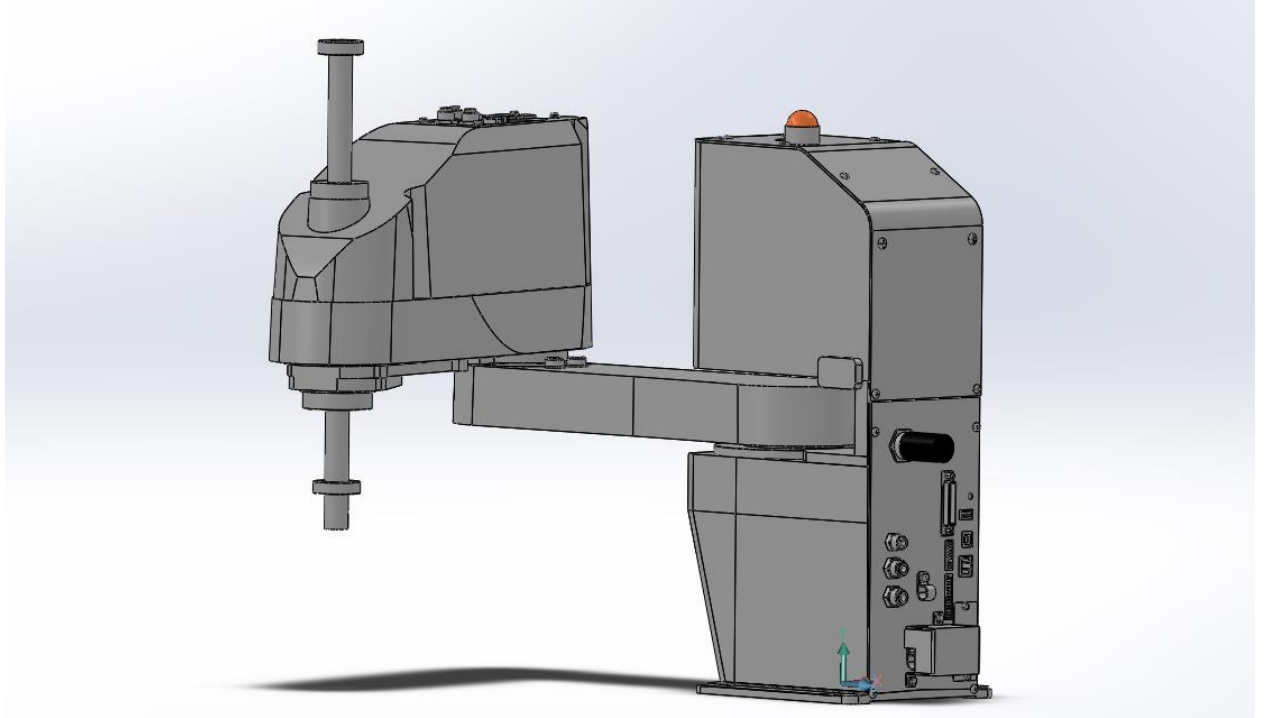
Hình 1.2 Thông số kỹ thuật của Model Robot SCARA T6

## CHƯƠNG 2. NỘI DUNG THỰC HIỆN

### 2.1 Mô hình và cấu trúc lập trình của SCARA Robot

#### 2.1.1 Mô hình Solidwork

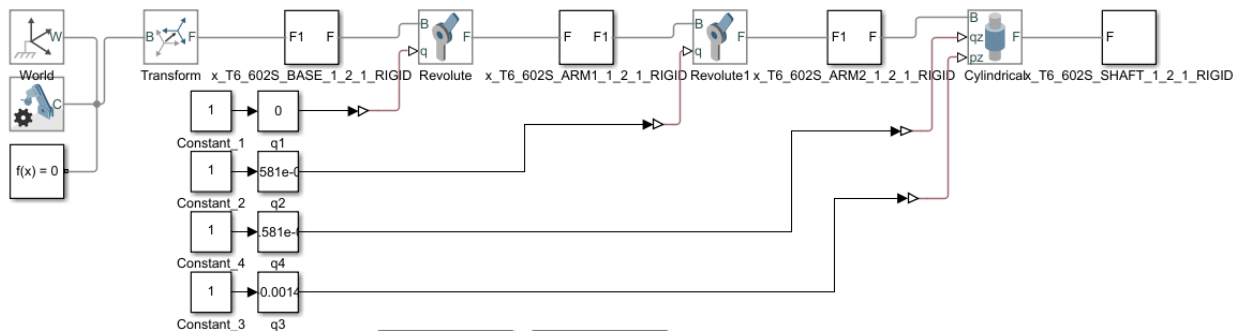
Đề tài sử dụng mô hình của Robot SCARA T6 của công ty EPSON. File SolidWorks của model này được công khai trên internet.



Hình 2.1 Mô hình SCARA Robot T6 trong SolidWorks

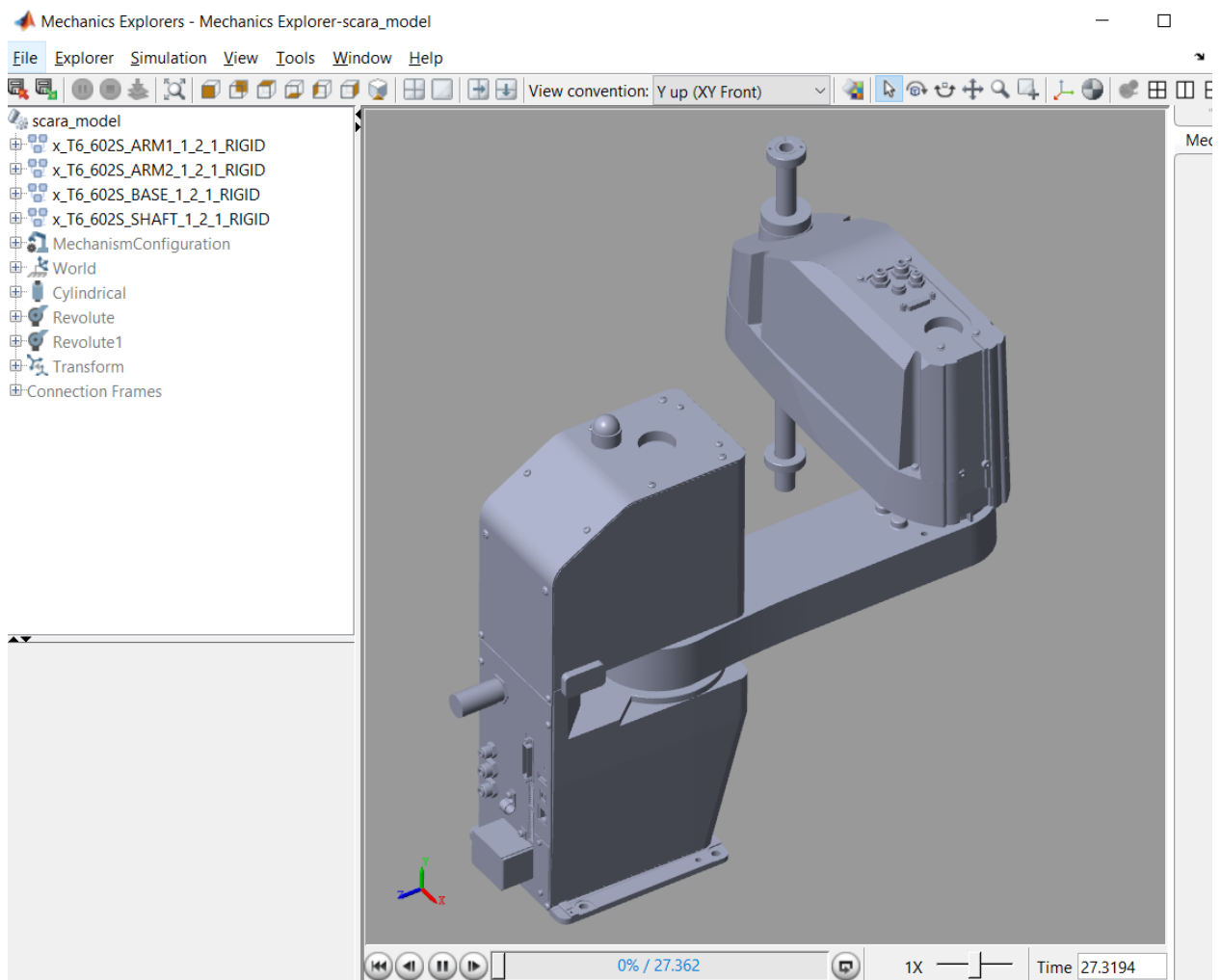
#### 2.1.2 Mô hình Simulink sử dụng Simscape

Sử dụng Simscape Multibodies ta có thể dễ dàng chuyển từ SolidWorks sang MATLAB Simulink và có thể mô phỏng được các khớp của robot.



Hình 2.2 Mô hình của SCARA Robot T6 trong MATLAB Simulink





Hình 2.3 Hình ảnh của robot trong Mechanics Explorers

### 2.1.3 Cấu trúc lập trình SCARA Robot

Một robot được cấu thành từ các thành phần cơ bản như các joints và các links. Tất cả các bài toán của robot đều được kèm với các hệ trục tọa độ (frames). Vì thế cần thiết phải tạo các class của những thành phần cơ bản này để có thể tạo được robot cách dễ dàng, có cấu trúc và linh hoạt khi cần chỉnh sửa, thay đổi. Mỗi class có các properties và các methods đặc trưng giúp tính toán một cách có cấu trúc. Các class này được kế thừa từ class `dlnode` (doubly linked-list) – một cấu trúc dữ liệu để có thể kết nối các object lại với nhau. Trong báo cáo chỉ để cơ bản phân khai báo các properties, chi tiết về code có thể xem trong source code đính kèm.

#### Class Link:

```
classdef C_Link < dlnode
    properties (Access = private)
        index;
        a; alpha; d; theta;
    end
    . . .
end
```

## Class Joint:

```
classdef C_Joint < dlnode
    properties (Access = private)
        index;
        type;
        parentLink C_Link;
        childLink C_Link;
        q;
        A = zeros(4);
        T = zeros(4);
    end
    methods (Access = public)
        function tformAT(obj) ...
        function update(obj, value) ...
    ...
end
```

































## Class Frame:

```
classdef C_Frame < dlnode
    properties (Access = public)
        index;
        p;
        o;
        joint C_Joint;
        link C_Link;
        frame0 C_Frame;
    end
    methods (Access = public)
        function update(obj) ...
    ...
end
```

**Class Robot:** là class chính với các thành phần cơ bản cũng như là các methods giúp giải quyết các bài toán cơ bản trong robot.

```
classdef C_Scara < handle
    properties (Access = public)
        model;
    end
    properties (Access = private)
        qr; %q response
        q;q_dot_; q_dot; q_2dot;
        q; q_next; q_mov;
        ef_p; ef_o; ef_p_mov;
        inverseParameters;
        a_max; mov_cycle; cycle; per_v;
        isInitialized; isMoving; isUsingJacobian; isPlanned; isUsingDynamics;
        s,s_mov;s_dot_mov;s_2dot_mov;%including s,s_dot,s_2dot
        link0 C_Link; link1 C_Link; link2 C_Link; link3 C_Link; link4 C_Link;
        joint1 C_Joint; joint2 C_Joint; joint3 C_Joint; joint4 C_Joint;
        frame0 C_Frame; frame1 C_Frame; frame2 C_Frame; frame3 C_Frame; frame4 C_Frame;
        timer timer; Ts;
        model_status;
        J; J_main; %Jacobian
    end
    ...
end
```

### 2.1.4 Cấu trúc thư mục của đề tài

	SCARA_robot_GUI.m	%chứa code để điều khiển GUI
	SCARA_robot_GUI.fig	%GUI layout
	Scara_DH_params.mat	%thông số DH của robot
	add_path.m	%thêm đường dẫn vào project
	Utilities	%chứa các hàm ứng dụng của robot
	trajectory_planning.m	 Jacobian_joint.m
	tform.m	 inverse_kinematics_scara.m
	r2d.m	 find_rotation_matrix.m
	plot_circle.m	 find_circle_2d.m
	path_planning_linear.m	 find_angle_2_vectors.m
	path_planning_circular_2D.m	 d2r.m
	Model	
	scara_model.slx	%mô hình Simulink của robot
	dynamics.slx	%mô hình Simulink của động lực học
	scara_model_DataFile.m	%thông số mô hình robot
	EPSON - T6 Robot	%chứa các file SolidWorks của robot
	GUI_function	%chứa các hàm ứng dụng của GUI
	resetAxes.m	
	resetAllAxes.m	
	init_GUI.m	
	Class	%chứa các class của robot
	dlnode.m	%class doubly linked-list node
	C_Scara.m	%class chính của robot
	C_Link.m	%class link
	C_Joint.m	%class joint
	C_Frame.m	%class frame

Hình 2.4 Cấu trúc thư mục của đề tài

## 2.2 Động học thuận (Forward Kinematics)

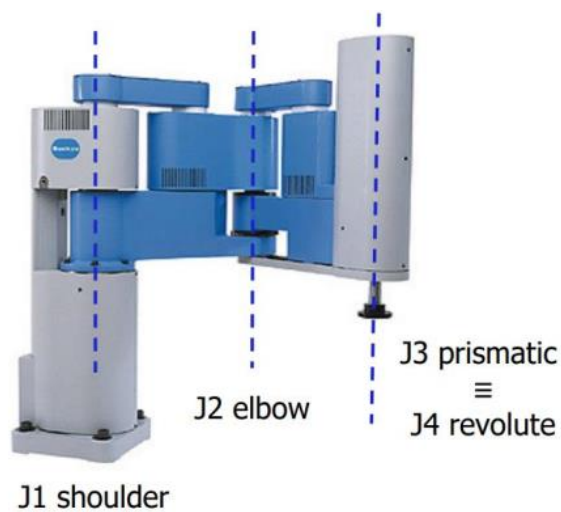
Động học thuận là bài toán đi tìm vị trí và hướng xoay của các hệ tọa độ được đặt trên các khớp của robot, và quan trọng nhất là trên đầu công tác, khi biết trước giá trị của các biến khớp.

### Phương pháp Denavit-Hartenberg (DH)

Bất kỳ một robot nào cũng có thể xem là tập hợp các khâu gắn liền với các khớp. Nếu đặt cho mỗi khâu một hệ trục tọa độ và sử dụng các phép biến đổi thì có thể mô tả vị trí và hướng tương đối giữa các hệ tọa độ.

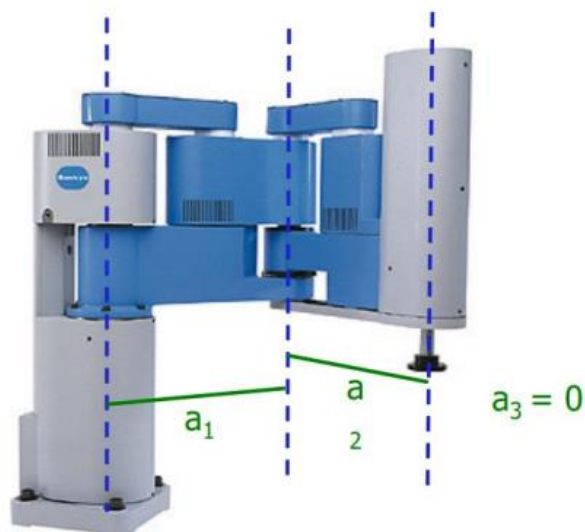
Để xác định bảng DH của SCARA Robot

Bước 1: Xác định các joints



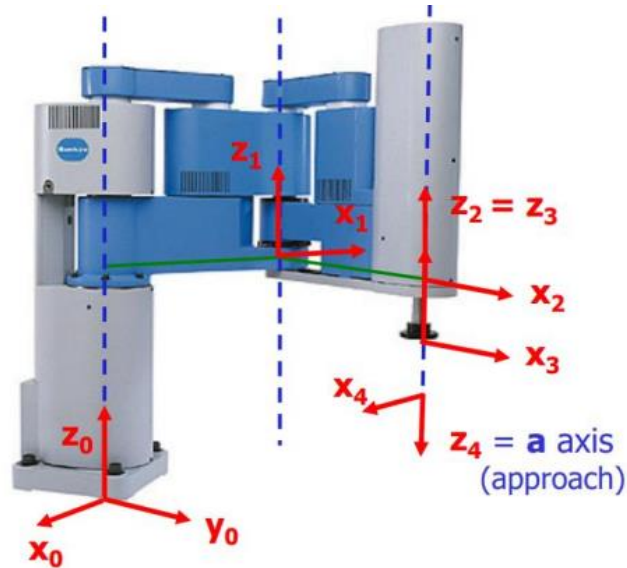
Hình 2.5 Xác định các joints của robot

Bước 2: Xác định các links



Hình 2.6 Xác định các links của robot

### Bước 3: Xác định các frames



Hình 2.7 Xác định các frames của robot

### Bước 4: Lập bảng thông số DH

Với cấu trúc của robot trong bài tập lớn, bảng DH của SCARA Robot như sau:

Link	$a_i$	$\alpha_i$	$d_i$	$\theta_i$
1	0.3250	0	0.0054	0
2	0.2750	0	0	1.5708
3	0	0	0	0
4	0	3.1416	-0.0040	0

Bảng 2.1 Bảng thông số DH của Robot SCARA

Với tầm của các joint variables:

Joint Variables		Min	Initialized Value	Max
$q_1$	$\theta_{\theta 1}$	- 132 deg	0 deg	132 deg
$q_2$	$\theta_{\theta 2}$	- 150 deg	90 deg	150 deg
$q_3$	$d_3$	200 mm ~ 0.2m		
$q_4$	$\theta_{\theta 4}$	-180 deg	0 deg	180 deg

Bảng 2.2 Bảng tầm giá trị của các joint variables

Từ đây ta sử dụng công thức tính động học thuận thông qua tính toán các ma trận chuyển vị trung gian  $T_i^{i-1}$  và công thức

$$T_n^0 = T_1^0 T_2^1 T_3^2 \dots T_n^{n-1}$$

Với ma trận chuyển vị được tính toán dựa trên các thông số DH của từng link như sau:

```
function matrix = tform(a, alpha, d, theta)
    matrix = [cos(theta) , -cos(alpha)*sin(theta) , sin(alpha)*sin(theta) , a*cos(theta) ;
              sin(theta) , cos(alpha)*cos(theta) , -sin(alpha)*cos(theta) , a*sin(theta) ;
              0 , sin(alpha) , cos(alpha) , d ;
              0 , 0 , 0 , 1 ];
```

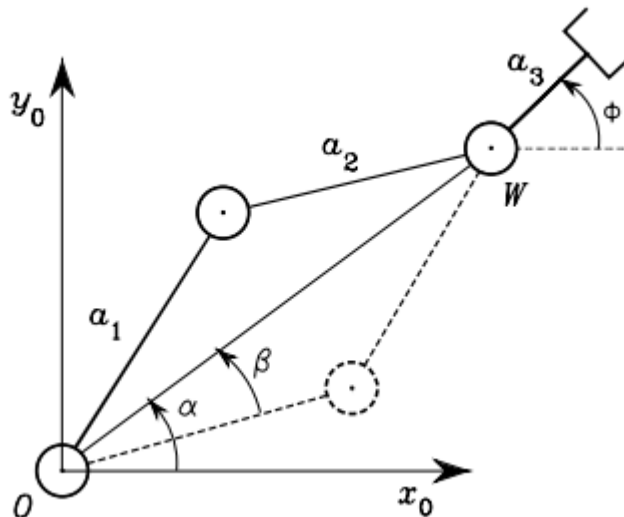
### 2.3 Động học ngược (Inverse Kinematics)

Trong việc hoạch định robot, ta hoạch định trước vị trí và hướng của những điểm robot mà sẽ đi tới, nên yêu cầu đặt ra là tìm các giá trị biến khớp tương ứng với vị trí và hướng đó. Động học ngược là bài toán đi tìm giá trị của các biến khớp khi biết trước vị trí và hướng xoay của đầu công tác. Vì góc  $\vartheta$  và  $\psi$  luôn cố định, nên ta chỉ quan tâm vào góc  $\varphi$  của robot. Giả sử ta biết trước vị trí  $x, y, z$  và góc  $\varphi$  của robot.

Theo hệ phương trình động học thuận vừa tìm được ở trên, ta dễ dàng tính được:

$$d_3^* = z - d_1 - d_4$$

Khi nhìn robot theo hướng từ trên xuống (top view), ta thấy được SCARA Robot có cấu trúc như một two-link planar arm:



Hình 2.8 Cấu trúc của two-link planar arm

Vì thế, mô hình động học thuận của robot có thể được tính toán như sau:

```
function q_next = inverse_kinematics_scara(ef_desired_pose, inverseParameters)
    q_next = zeros(4,1);

    a1 = inverseParameters(1); a2 = inverseParameters(2); d1 = inverseParameters(3); d4 = inverseParameters(4);
    Pwx = ef_desired_pose(1); Pwy = ef_desired_pose(2);

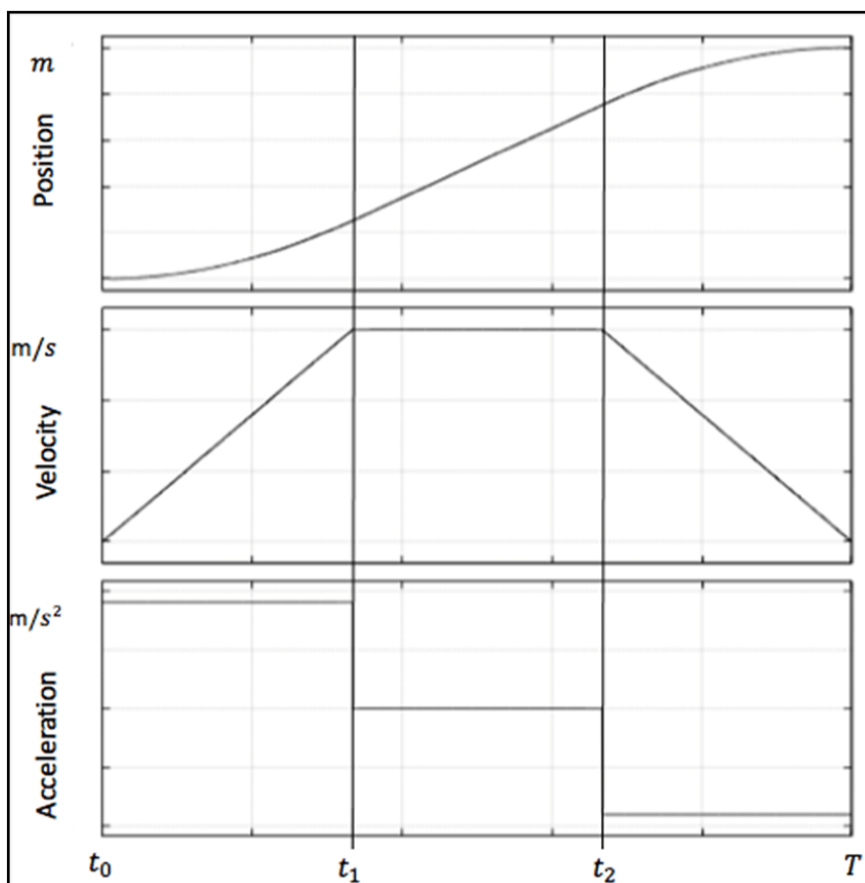
    % Calculate (sin cos) value
    c2 = (Pwx^2 + Pwy^2 - a1^2 - a2^2)/(2*a1*a2);
    q_next(2) = + acos(c2);
    alpha = atan2(Pwy, Pwx);
    beta = acos((Pwx^2 + Pwy^2 + a1^2 - a2^2)/(2*a1*sqrt(Pwx^2 + Pwy^2)));
    q_next(1) = alpha - beta;
    q_next(3) = ef_desired_pose(3) - d1 - d4;
    q_next(4) = ef_desired_pose(6) - q_next(1) - q_next(2);
end
```

## 2.4 Hoạch định quỹ đạo đường đi (Path planning and Trajectory planning)

Path planning và Trajectory planning bài toán quan trọng để có thể hoạch định và điều khiển được Robot.

Đề tài này tập trung vào hoạch định đường thẳng (Linear path), hoạch định đường tròn 2D (2D Circular path) và hoạch định vận tốc hình thang (1D trapezoidal trajectory). Các cách hoạch định khác có thể xây dựng dễ dàng dựa trên cấu trúc lập trình sẵn có của robot.

### 2.4.1 Hoạch định vận tốc hình thang



Hình 2.9 Đồ thị vị trí, vận tốc và gia tốc của hoạch định vận tốc hình thang

Ở đây ta hoạch định cho vị trí, vận tốc và gia tốc của đầu công tác. Ta chia ra làm 3 giai đoạn:  $t_0 - t_1$ ,  $t_1 - t_2$ ,  $t_2 - t_3$ . Có thể dễ dàng tính toán các  $t_i$  qua các đầu vào  $s_{\max}$ ,  $v_{\max}$ ,  $a_{\max}$  thỏa mãn  $v_{\max} \leq \sqrt{s_{\max} a_{\max}}$ .

Sau đó ta chia các giai đoạn này theo thời gian lấy mẫu  $T_s$ , thời gian càng ngắn, việc di chuyển càng mượt nhưng khi mô phỏng với Matlab, khối lượng tính toán không tối ưu dẫn đến không thể đặt thời gian  $T_s$  quá nhỏ.

```
t1 = v_max/a_max;
t2 = (s_max - v_max*t1)/v_max + t1;
t3 = t1 + t2; %t3 = t1+(t2-t1)+(t3-t2) because (t3-t2)=t1
for i = 1:len
    %stage 1
    if t(i) < t1
        s(i) = a_max * t(i)^2 / 2;
        s_dot(i) = a_max * t(i);
        s_2dot(i) = a_max;
    %stage 2
    elseif t(i) < t2
        s(i) = a_max * t1^2 / 2 + v_max * (t(i) - t1);
        s_dot(i) = v_max;
        s_2dot(i) = 0;
    %stage 3
    else
        s(i) = a_max * t1^2 / 2 + v_max * (t2 - t1) + (v_max * (t(i) - t2) - a_max * (t(i) - t2)^2 / 2);
        s_dot(i) = v_max - a_max * (t(i) - t2);
        s_2dot(i) = - a_max;
    end
end
```

## 2.4.2 Hoạch định đường thẳng

Xét robot di chuyển thẳng từ điểm  $p_i$  đến điểm  $p_j$ . Ta có hàm tham số thể hiện đoạn vị trí, vận tốc và gia tốc của đầu công tác theo timing law  $s(t)$  là:

$$p_e(s) = p_i + \frac{s}{\|p_f - p_i\|} (p_f - p_i)$$

$$\dot{p}_e = \frac{\dot{s}}{\|p_f - p_i\|} (p_f - p_i) = \dot{s}t$$

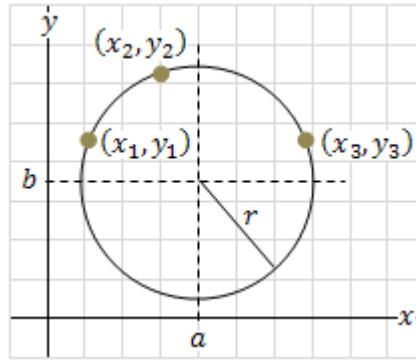
$$\ddot{p}_e = \frac{\ddot{s}}{\|p_f - p_i\|} (p_f - p_i) = \ddot{s}t$$

## 2.4.3 Hoạch định đường tròn 2D

Để robot di chuyển theo đường tròn, ta cần có tâm và bán kính định sẵn, hoặc đi qua 2 điểm chỉ định. Thực tế cho thấy việc đi qua 2 điểm chỉ định có tính ứng dụng cao hơn. Vì thế, bài tập lớn sẽ giải quyết trường hợp hoạch định đường đi tròn qua 2 điểm chỉ định trên cùng 1 mặt phẳng với vị trí hiện tại (2D).



Trước tiên ta cần tính toán được tâm và bán kính của đường tròn.



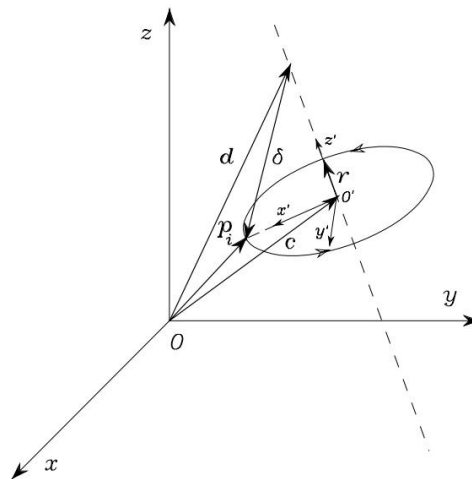
Hình 2.10 Đường tròn đi qua 3 điểm trong mặt phẳng

```
function [I,R] = find_circle_2d(point1, point2, point3)
    I = zeros(2,1); R = 0;
    x1 = point1(1); x2 = point2(1); x3 = point3(1);
    y1 = point1(2); y2 = point2(2); y3 = point3(2);

    ma = (y2-y1)/(x2-x1); mb = (y3-y2)/(x3-x2);

    I(1) = (ma*mb*(y1-y3)+mb*(x1+x2)-ma*(x2+x3))/(2*(mb-ma));
    I(2) = -1/ma*(I(1)-(x1+x2)/2)+(y1+y2)/2;

    R = sqrt((x1-I(1))^2+(y1-I(2))^2);
end
```



Hình 2.11 Mô tả hoạch định đường tròn trong hệ trục tọa độ

Xét một đường tròn  $\Gamma$  trong không gian. Trong đó:

- vector đơn vị của trục đường tròn  $r$ ,
- vector vị trí  $d$  của một điểm dọc theo trục đường tròn,
- vector vị trí  $p_i$  của một điểm trên đường tròn.

Xét hệ tọa độ  $O'-x'y'z'$ , trong đó  $O'$  trùng với tâm đường tròn, trục  $x'$  hướng dọc theo phương của vector  $p_i - c$ , trục  $z'$  được định hướng dọc theo  $r$  và trục  $y'$

được chọn sao theo quy tắc bàn tay phải. Khi được biểu thị trong hệ quy chiếu này, ta được:

$$p'(s) = \begin{bmatrix} \rho \cos(s / \rho) \\ \rho \sin(s / \rho) \\ 0 \end{bmatrix}$$

Với  $\rho = \|p_i - c\|$  là bán kính đường tròn tìm được. Chuyển về hệ trục tọa độ tham chiếu, ta được

$$p(s) = c + R p'(s)$$

Với  $R$  là ma trận xoay của frame  $O'-x'y'z'$  so với frame  $O-xyz$ .

Ta có hàm tham số thể hiện vận tốc và gia tốc của đầu công tác:

$$\dot{p}_e(s) = R \begin{bmatrix} -\dot{s} \sin(s / \rho) \\ \dot{s} \sin(s / \rho) \\ 0 \end{bmatrix}$$

$$\ddot{p}_e(s) = R \begin{bmatrix} -\dot{s}^2 \cos(s / \rho) / \rho - \ddot{s} \sin(s / \rho) \\ -\dot{s}^2 \sin(s / \rho) / \rho + \ddot{s} \cos(s / \rho) \\ 0 \end{bmatrix}$$

## 2.5 Động học vi sai (Diffirential Kinematics)

Động học vi sai mô tả mối quan hệ giữa vận tốc của các joint và vận tốc của đầu công tác. Mối quan hệ này được miêu tả bằng ma trận Jacobian. Ma trận Jacobian là một công cụ quan trọng trong việc tính những bài toán của robot như tìm điểm kỳ dị, phân tích redundancy, xác định động học ngược thông qua các thuật toán và mô tả mối quan hệ giữa lực tác dụng của đầu công tác với momen của từng joint.

Bài tập lớn này chỉ tính toán ma trận Jacobian. Các bài toán khác có thể phát triển dễ dàng dựa trên ma trận Jacobian đã tìm được.

$$J = \begin{bmatrix} J_{P1} & \dots & J_{Pn} \\ J_{O1} & & J_{On} \end{bmatrix},$$

where

$$\begin{bmatrix} J_{Pi} \\ J_{Oi} \end{bmatrix} = \begin{cases} \begin{bmatrix} z_{i-1} \\ \mathbf{0} \end{bmatrix} & \text{for a } \textit{prismatic} \text{ joint} \\ \begin{bmatrix} z_{i-1} \times (p_e - p_{i-1}) \\ z_{i-1} \end{bmatrix} & \text{for a } \textit{revolute} \text{ joint.} \end{cases}$$

Trong đề tài này, ta lập trình sử dụng công thức tổng quát để tính từng cột của ma trận.

```
function j = Jacobian_joint(robot,joint)
    framei_1 = robot.get('frame0');
    index = joint.get('index');
    while ~(framei_1.get('index') == index - 1)
        framei_1 = framei_1.Next;
    end
    %   zi_1 = framei_1.get('z');
    zi_1 = [0;0;1];
    joint_type = joint.get('type');
    switch joint_type
        case 'prismatic'
            j = [zi_1;0;0;0];
        case 'revolute'
            j = [cross(zi_1,robot.get('ef_p') - framei_1.get('p'));zi_1];
        otherwise
            warning('Something went wrong!!!');
    end
end
```

## 2.6 Động lực học (Dynamics)

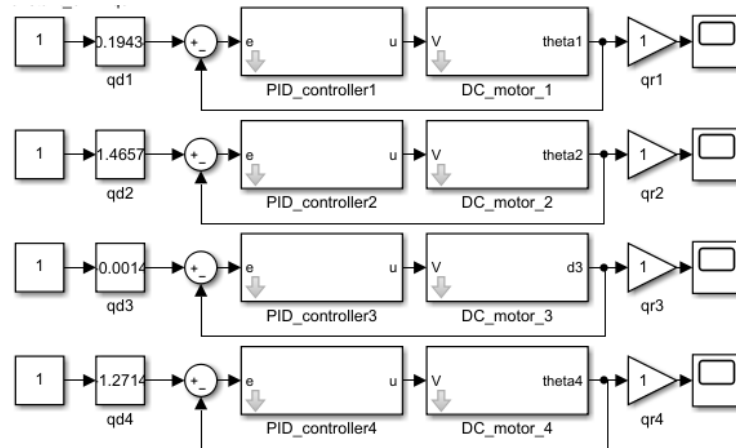
Trong các ứng dụng của robot, các hệ thống thường là tuyến tính và không thay đổi theo thời gian. Thực tế, ta điều khiển robot bằng bộ PID để các giá trị của joint bám theo giá trị đặt sau khi tính động học ngược.

Đề tài mô phỏng các thông số mô hình động cơ như sau:

Ki	<input type="text" value="0.043"/>
Kb	<input type="text" value="0.04297"/>
L	<input type="text" value="0.0001"/>
R	<input type="text" value="1.025"/>
Jeff	<input type="text" value="0.000056+(1e-4)*5"/>
Beff	<input type="text" value="0.00008092"/>
N	<input type="text" value="100"/>

Hình 2.12 Thông số mô phỏng của động cơ

Từ đó ta thiết lập các bộ PID cho từng joint với các thông số  $K_p$ ,  $K_i$ ,  $K_d$ :



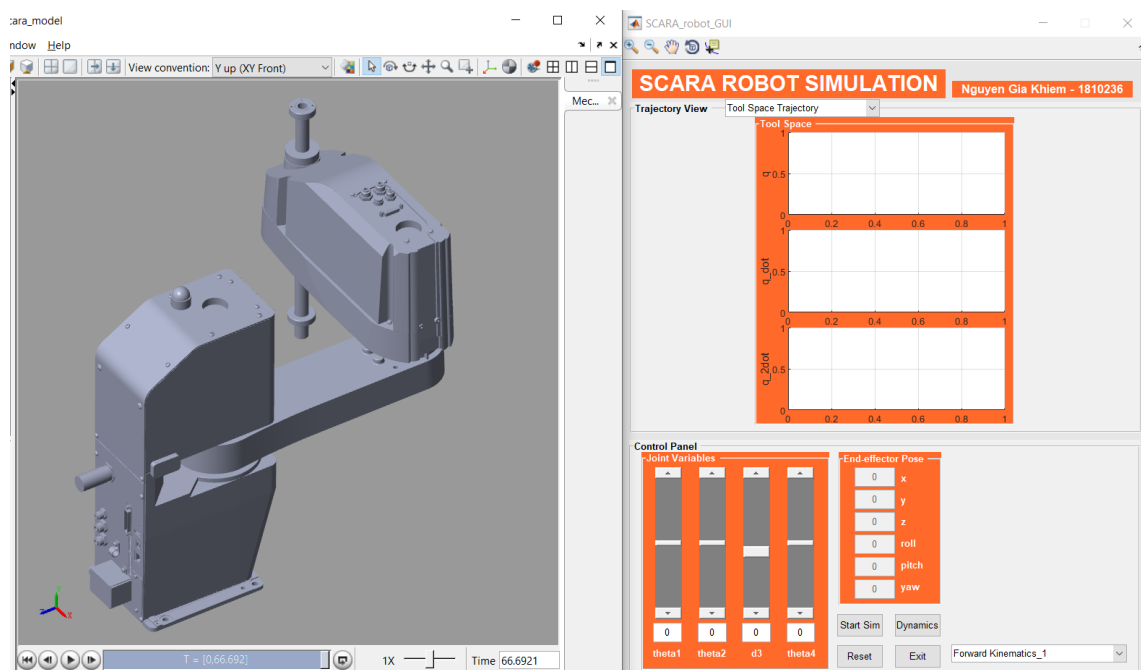
Hình 2.13 Bộ điều khiển PID cho các joints

$K_p$	350
$K_i$	0
$K_d$	50

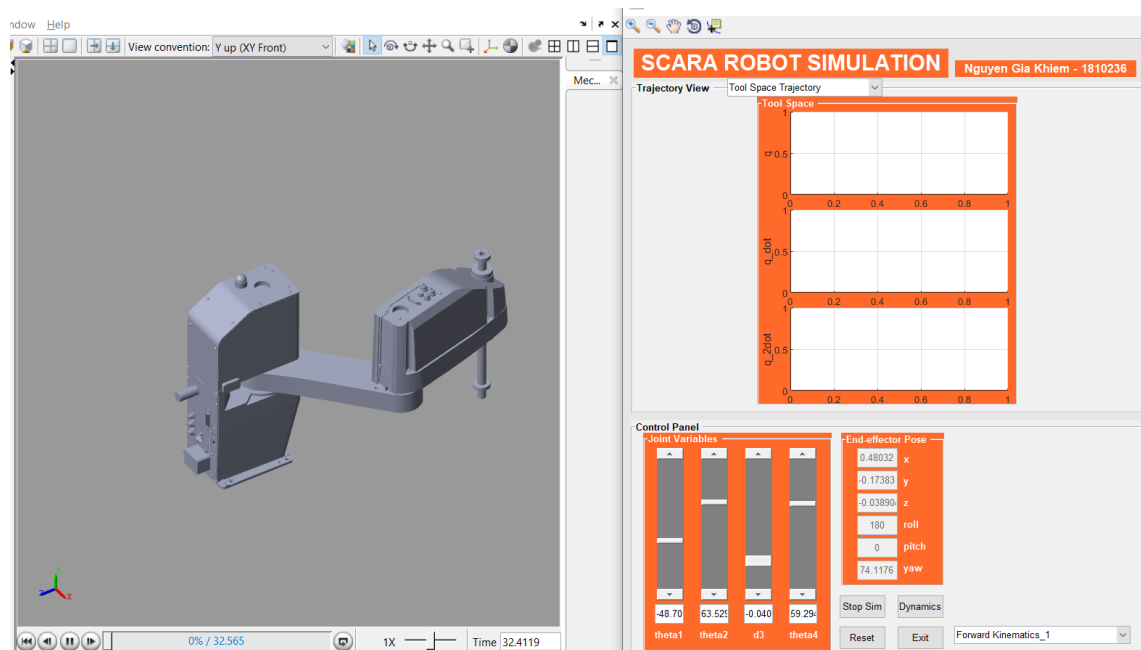
Hình 2.14 Thông số PID cho động cơ

## 2.7 Kết quả

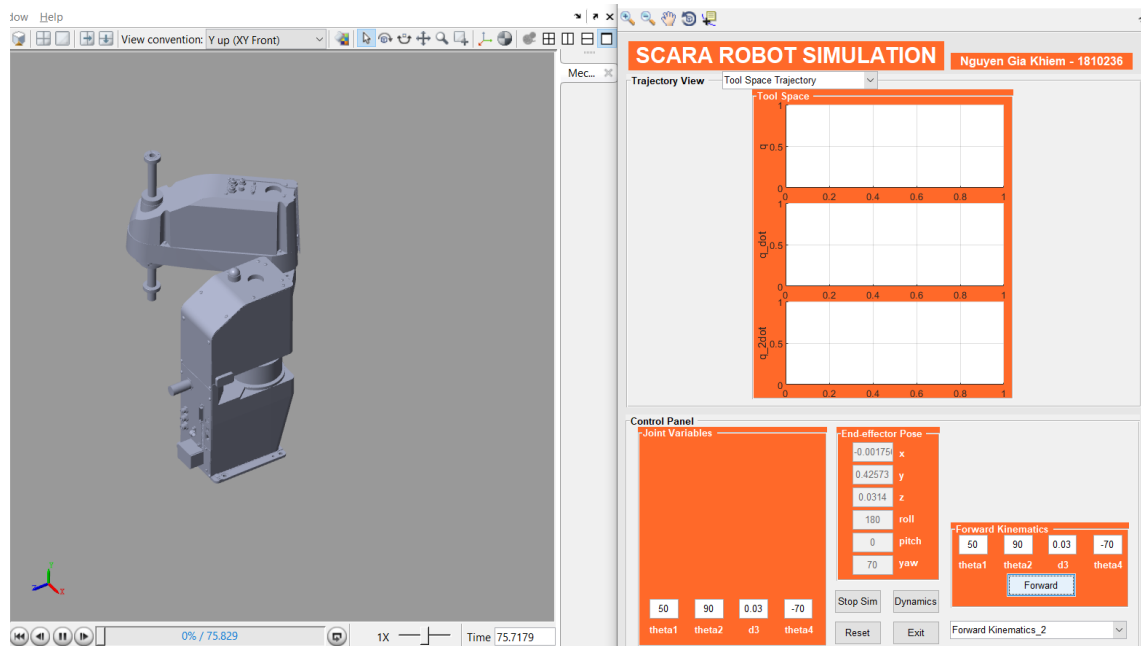
Dưới đây là một số hình ảnh trình bày sơ lược về kết quả. Video kết quả chi tiết hoạt động được gửi đính kèm với file báo cáo này.



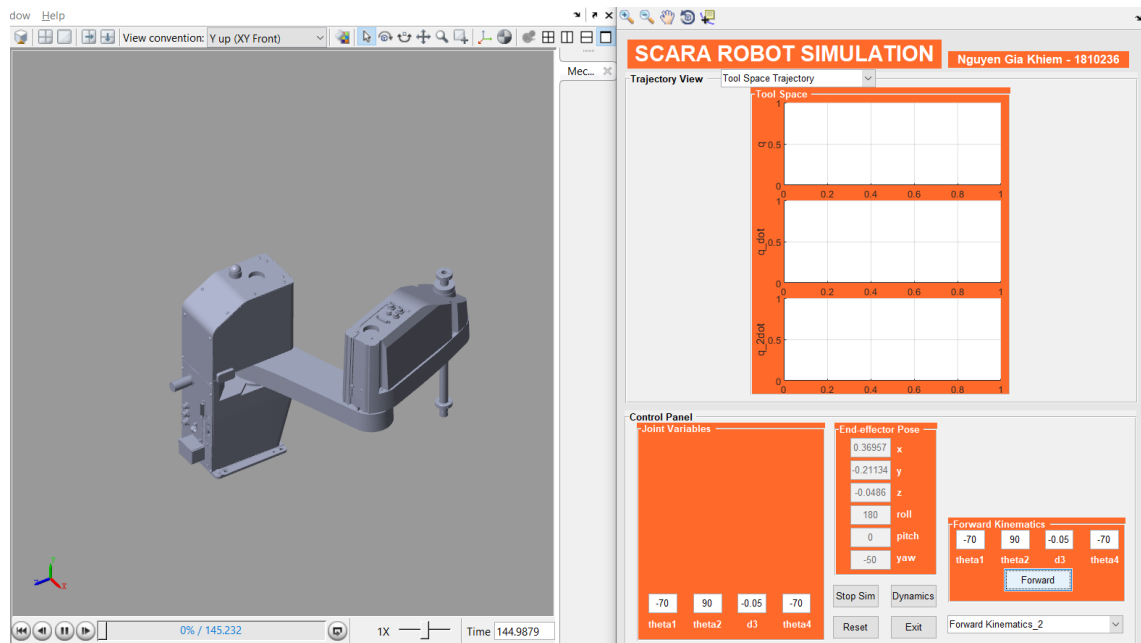
Hình 2.15 Tổng quan giao diện mô phỏng robot



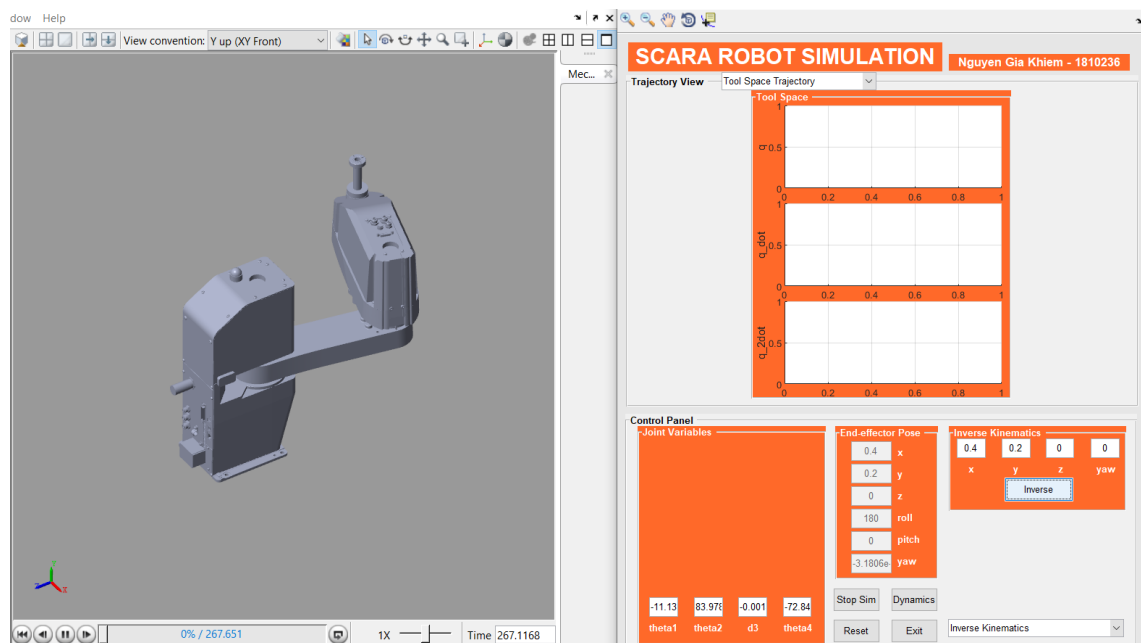
Hình 2.16 Điều khiển các khớp robot bằng thanh trượt



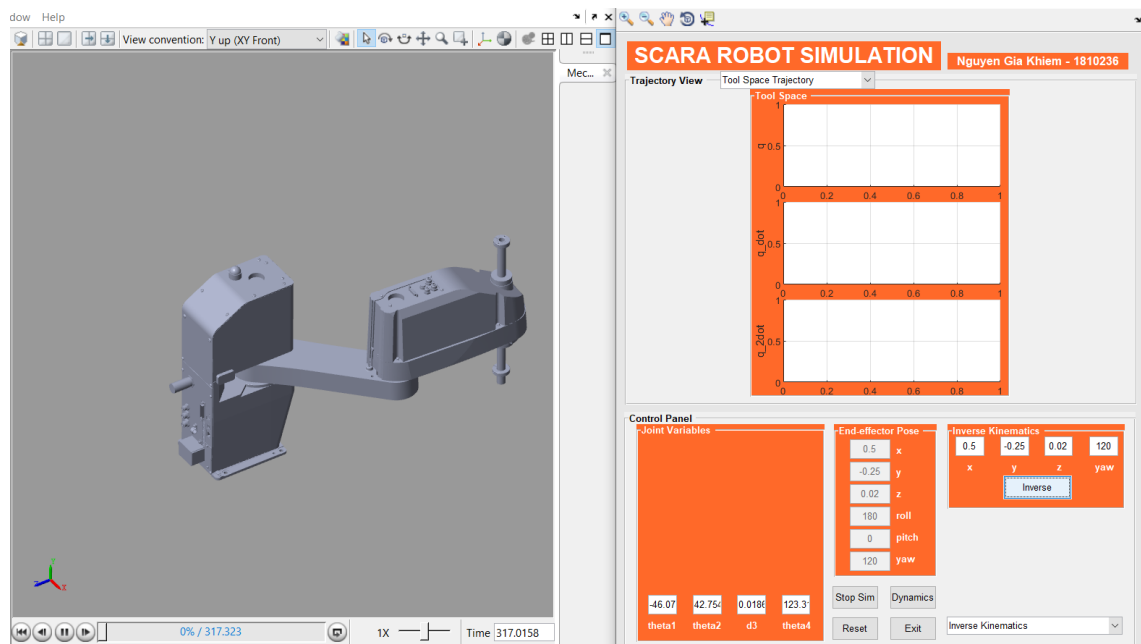
Hình 2.17 Bài toán Động học thuận của robot



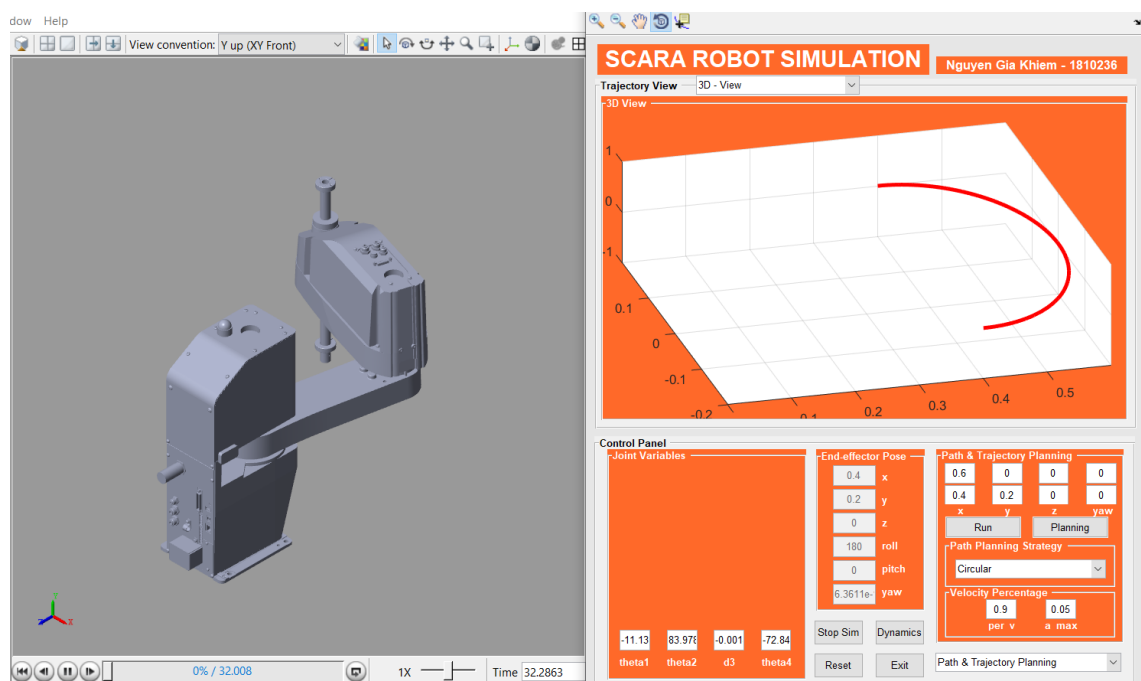
Hình 2.18 Bài toán Động học thuận của robot



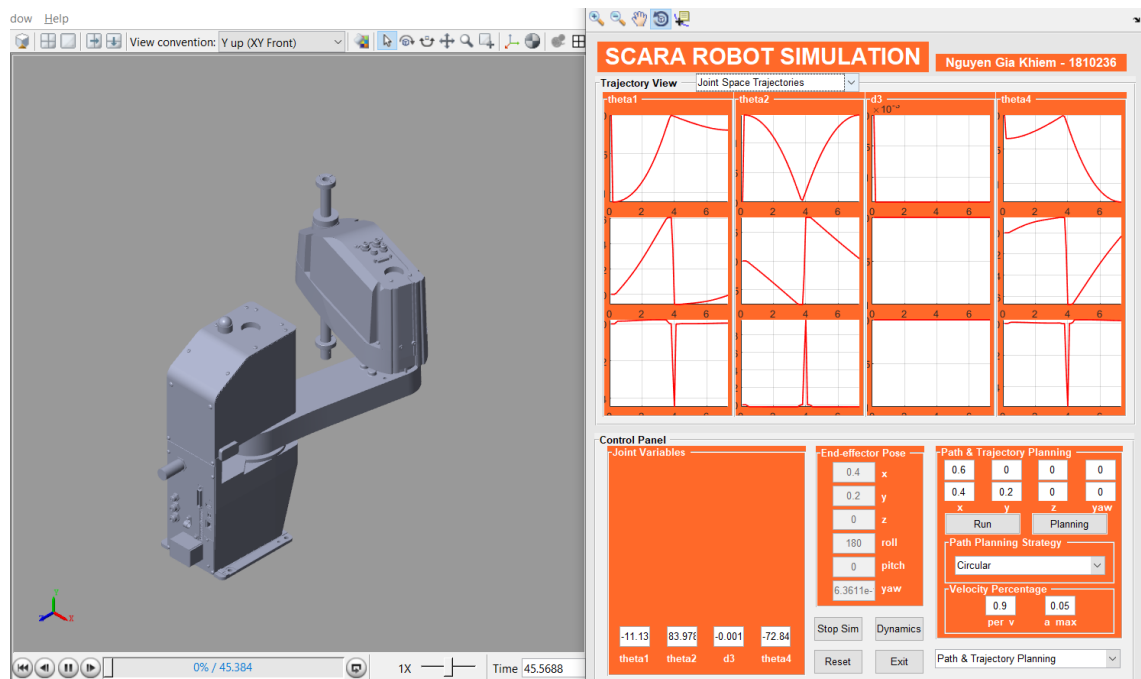
Hình 2.19 Bài toán Động học ngược của robot



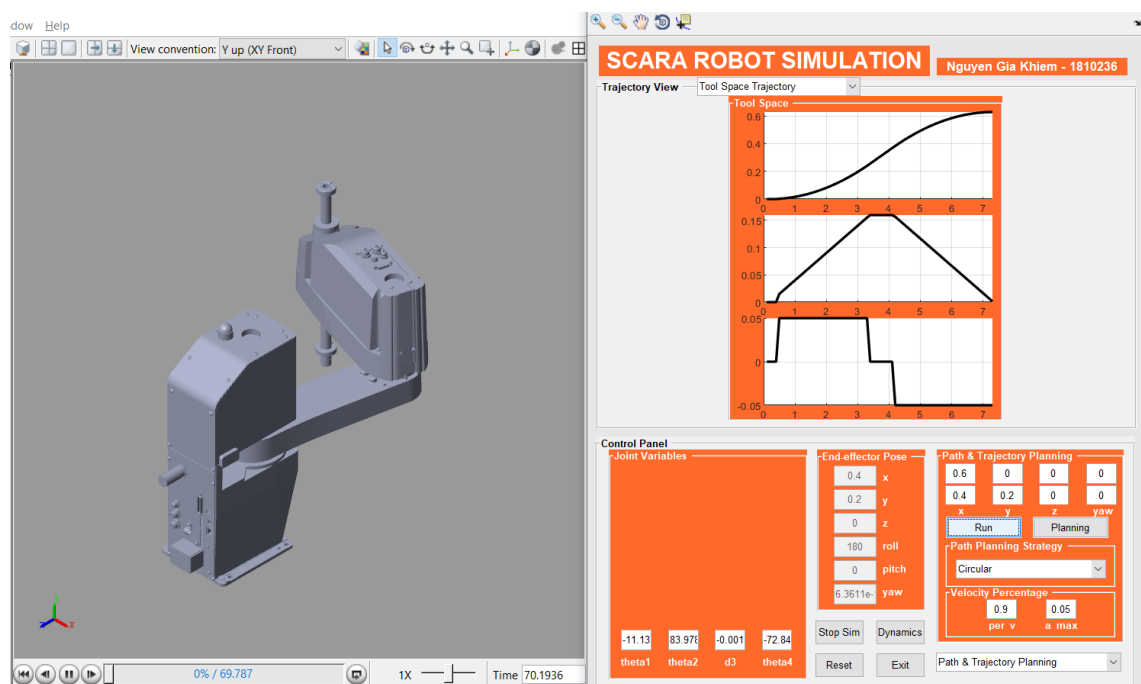
Hình 2.20 Bài toán Động học ngược của robot



Hình 2.21 Bài toán Hoạch định quỹ đạo đường đi cho robot

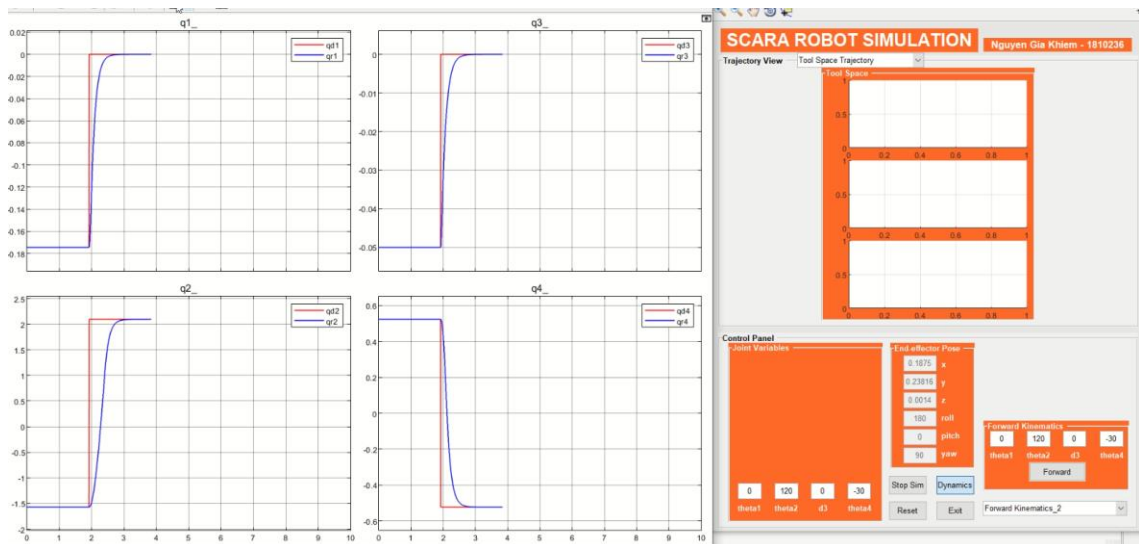


Hình 2.22 Đồ thị giá trị, vận tốc và gia tốc của các khớp xoay theo thời gian khi di chuyển theo quỹ đạo đã hoạch định trước.

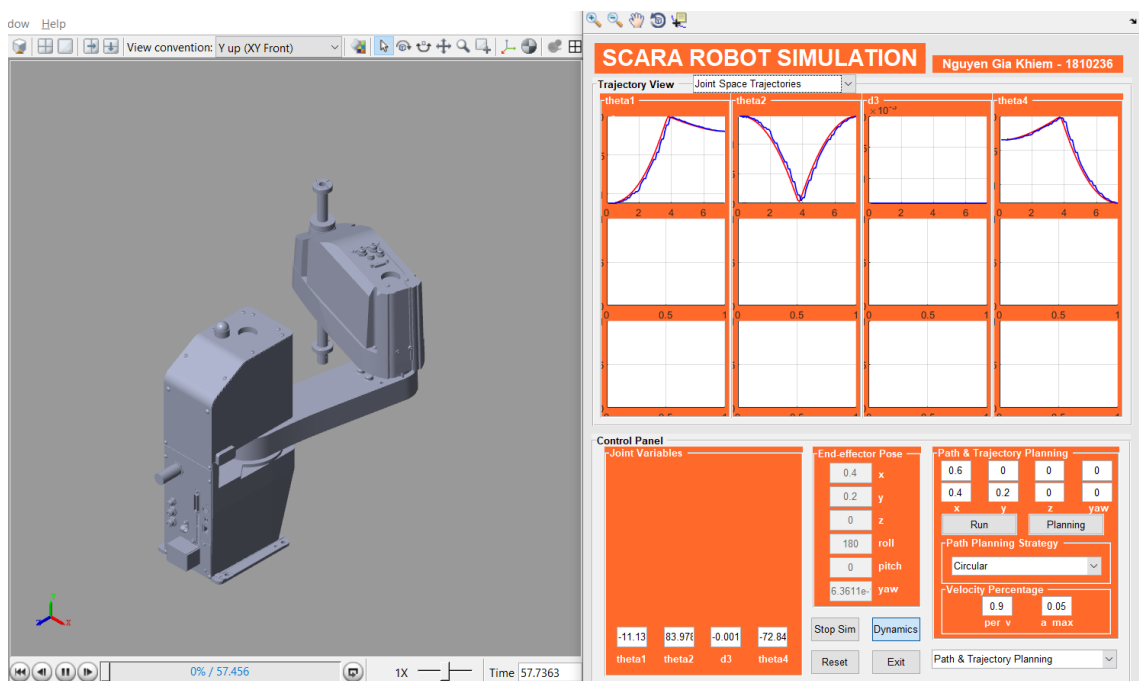


Hình 2.23 Đồ thị quỹ đạo, vận tốc, gia tốc của đầu công tác theo thời gian





Hình 2.24 Đồ thị giá trị các khớp xoay khi có áp dụng động lực học (mô phỏng động cơ và bộ điều khiển PID)



Hình 2.25 Hoạch định quỹ đạo có sử dụng mô phỏng động lực học

## **CHƯƠNG 3. ĐÁNH GIÁ KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN**

### **3.1 Đánh giá kết quả**

- Mô phỏng được SCARA Robot một cách ổn định với giao diện dễ nhìn, dễ sử dụng
- Giải quyết được hầu hết các bài toán cơ bản của SCARA Robot
- Lập trình cấu trúc cho robot dễ dàng, linh hoạt.
- Chưa tối ưu được các phép tính dẫn đến thời gian lấy mẫu còn lớn.

### **3.2 Hướng phát triển**

- Cải thiện việc lập trình tính toán cho robot để giảm thời gian lấy mẫu.
- Tham khảo các trình mô phỏng của các hãng để tối ưu
- Phân chia cấu trúc lập trình sao cho hợp lý và gọn gàng hơn để từ đó có thể phát triển những robot khác nhau.

## CHƯƠNG 4. TÀI LIỆU THAM KHẢO

- [1] Nguyễn Hoàng Giáp. *Kỹ thuật robot*. Bài giảng điện tử
- [2] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo (2008). *Robotics - Modelling, Planning and Control 2<sup>nd</sup> edition*. Springer
- [3] Epson T6 SCARA Robots | SCARA T Series [Online]:  
[https://epson.com/Support/Robots/SCARA-Robots/SCARA-T-Series/Epson-T6-SCARA-Robots/s/SPT\\_RT6-602SS](https://epson.com/Support/Robots/SCARA-Robots/SCARA-T-Series/Epson-T6-SCARA-Robots/s/SPT_RT6-602SS)
- [4] Epson T6 SCARA Robot [Online]:  
<https://grabcad.com/library/epson-t6-scara-robot-1>