ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Tự Sang

MÔ HÌNH HÓA ĐỘNG HỌC VÀ ĐIỀU KHIỂN CÁNH TAY ROBOT ỨNG DỤNG MẠNG NEURAL NETWORK

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY
Ngành: Kỹ thuật Robot

HÀ NỘI - 2023

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Tự Sang

MÔ HÌNH HÓA ĐỘNG HỌC VÀ ĐIỀU KHIỂN CÁNH TAY ROBOT ỨNG DỤNG MẠNG NEURAL NETWORK

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY
Ngành: Kỹ thuật Robot

Cán bộ hướng dẫn: TS. Dương Xuân Biên

HÀ NỘI - 2023

TÓM TẮT

Tóm tắt: Trong thời đại khoa học và kĩ thuật phát triển mạnh mẽ như hiện nay, hầu hết các quốc gia đang thúc đẩy việc phát triển tự động hóa trong quá trình sản xuất để nâng cao chất lượng sản phẩm cũng như chi phí sản xuất. Bên cạnh đó, robot ngày càng được cải thiện với tính linh hoạt, thông minh, chính xác và phản hồi nhanh hơn. Vì vậy, việc nghiên cứu để tận dụng các robot một cách hiệu quả là một điều vô cùng quan trọng. Trong khi đó, việc giải quyết vấn đề điều khiển cánh tay robot là điều cần thiết để điều chỉnh chúng theo quỹ đạo đã được xác định trước. Hơn nữa, triển khai và tìm kiếm một phương pháp hiệu quả để giải quyết vấn đề điều khiển cánh tay robot với nhiều bậc tự do vẫn đang là một thách thức lớn đối với các nhà nghiên cứu.

Nội dung đồ án xoay quanh việc tìm hiểu, nghiên cứu và triển khai bài toán động học ngược và điều khiển sử dụng mạng nơ-ron. Từ đó triển khai bài toán trên mô hình robot thực tế 3 bậc tự do.

Từ khóa: robot 3 bậc tự do, động học ngược, bộ điều khiển sử dụng mạng nơ-ron.

LỜI CẢM ƠN

Lời đầu tiên cho phép em xin được gửi lời cảm ơn chân thành và sâu sắc đến thầy Dương Xuân Biên – người đã hướng dẫn em nhiệt tình trong thời gian qua để hoàn thành đồ án. Sự hướng dẫn, hỗ trợ và động viên của thầy đã giúp em rất nhiều trong việc nghiên cứu, khắc phục những khó khăn và hoàn thành đồ án.

Bên cạnh đó, em xin cảm ơn các thầy cô khoa Điện tử Viễn thông, trường Đại học Công nghệ, Đại học Quốc gia Hà Nội nói chung và các thầy cô, anh chị ngành Kỹ thuật Robot nói riêng. Nhờ quá trình giảng dạy, hướng dẫn đó em đã học hỏi và trau dồi được những kiến thức quý báu để hoàn thiện Đồ án tốt nghiệp.

Do thời gian cũng như điều kiện nghiên cứu còn hạn chế nên Đồ án không tránh khỏi còn có những sai sót, em rất mong nhận được sự thông cảm và đóng góp ý kiến từ các thầy cô và các bạn đọc.

Em xin chân thành cảm ơn

Nguyễn Tự Sang

LỜI CAM ĐOAN

Em xin cam đoan Đồ án "Mô hình hóa động học và điều khiển cánh tay robot ứng dụng mạng Neural Network" hoàn toàn được thực hiện bởi bản thân dưới sự hướng dẫn của TS. Dương Xuân Biên.

Mọi thông tin tham khảo và tài liệu được sử dụng trong Đồ án sẽ được ghi rõ nguồn gốc tại phần danh mục tài liệu tham khảo. Tất cả nội dung trong Đồ án đều tuân thủ nguyên tắc không sao chép từ tài liệu hay công trình nghiên cứu của người khác.

Hà Nội, ngày 30 tháng 11 năm 2023

Tác giả

PHÊ DUYỆT CỦA CÁN BỘ HƯỚNG DẪN

Bằng văn bản này, tôi chấp thuận rằng đồ án ở dạng hiện tại đã sẵn sàng cho hội đồng thẩm định như một yêu cầu đối với bằng Kỹ sư ngành Kỹ thuật Robot tại Trường Đại học Công Nghệ.

Hà Nội, ngày 30 tháng 11 năm 2023

Cán bộ hướng dẫn

TS. Dương Xuân Biên

MỤC LỤC

MỤC LỤC HÌNH ẢNH

MỤC LỤC BẢNG BIỂU

CHƯƠNG 1. TỔNG QUAN BÀI TOÁN ĐIỀU KHIỂN CÁNH TAY ROBOT CÔNG NGHIỆP

1.1. Cánh tay robot công nghiệp và bài toán điều khiển

1.1.1. Khái niệm

Robot công nghiệp (theo như Liên đoàn Robot Quốc tế) là bộ điều khiển đa năng được điều khiển tự động, có thể lập trình lại, có thể lập trình theo ba trục trở lên [8]. Chúng được thiết kế và sử dụng trong việc thay thế con người như là chúng có thể thực hiện những tác vụ lặp lại nhiều lần và nguy hiểm với độ chính xác cao[1].

Hình 1.1. Hệ thống robot công nghiệp

Hầu hết các loại robot hiện nay được sử dụng trong ngành công nghiệp và trong các nhà máy sản xuất (Hình 1.1). Chúng có đặc điểm riêng về cấu trúc và chức năng, được chuẩn hóa và thương mại hóa rộng rãi.

1.1.2. Vai trò của cánh tay robot công nghiệp

Robot công nghiệp hiện nay đã và đang đóng vai trò như một phần quan trọng cuộc sống mỗi con người. Một số vai trò và lợi ích phổ biến của cánh tay robot công nghiệp như sau.

Nâng cao năng suất làm việc: Điều quan trọng nhất đầu tiên đối với một nhà máy sản xuất đó là sản lượng. Khi các cánh tay robot được sử dụng trong các nhà máy, họ sẽ có một cơ hội lớn để hoạt động ngày đêm mà không cần nghỉ ngơi. Hơn nữa, việc sử

[1] Industrial Robotics, "An introduction and beginner's guide (2019) RG Robotics | Advanced Automation Technology" - https://www.rg-robotics.com/industrial-robotics-an-introduction-and-beginners-guide/

dụng robot cũng mang đến một sự chính xác và bền bỉ cao hơn so với con người mặc dù điều này vẫn còn nhiều rủi ro nhưng năng suất vẫn được nâng cao rõ rệt.

Chi phí thuê người được giảm: Khi cánh tay robot công nghiệp được ứng dụng trong nhà máy, họ có thể giảm chi phí thuê công nhân một cách đáng kể. Nhà đầu tư sẽ không cần phải bỏ ra một số vốn nhất định vào thuê nhiều nhân công cho các loại công việc khác nhau mà họ chỉ cần bỏ ra một số vốn nhất định vào việc mua và bảo trì robot. Vì cánh tay robot có thể làm việc liên tục cho đến khi công việc hoàn thành với điều khiển được bảo trì và quan sát thường xuyên bởi các chuyên gia.

Cải thiện chất lượng: Việc mắc phải sai sót của mọi người trong đời sống cũng như công việc là không thể tránh khỏi. Tuy nhiên, robot đã được lập trình, thiết kế và kiểm thử rất kỹ trước khi được đưa vào sản xuất nên việc gặp phải những sai sót có thể giảm thiểu đến mức tối đa, bên cạnh nó chúng cũng được thực nghiệm rất nhiều lần trước khi được đưa vào thực tế.

1.1.3. Bài toán điều khiển

Trong ngành kỹ thuật robot, bài toán điều khiển là một trong những bìa toán không thể thiếu khi nghiên cứu lĩnh vực này. Nó quan tâm đến việc thiết kế các thuật toán và hệ thống để điều khiển robot sao cho chúng có thể thực hiện các nhiệm vụ mong muốn một cách chính xác và hiệu quả. Bài toán điều khiển thường bao gồm các yếu tố sau:

Mục tiêu điều khiển: Xác định mục tiêu hoặc nhiệm vụ cụ thể mà robot cần thực hiện. Điều này có thể là di chuyển đến vị trí cụ thể, duy trì vị trí, hoặc thực hiện các thao tác nhất định.

Mô hình hóa robot: Xây dựng mô hình toán học hoặc mô hình vật lý của robot để mô tả cấu trúc và hành vi của nó trong không gian. Điều này bao gồm việc mô tả động học, động lực học và các đặc tính khác của robot.

Xây dựng bộ điều khiển cho robot: Dựa trên việc tạo mô hình, phát triển các thuật toán và hệ thống điều khiển để quản lý các hoạt động của nó. Các phương pháp điều khiển có thể bao gồm điều khiển PID (Proportional-Integral-Derivative), điều khiển không cân bằng, điều khiển tuyến tính, điều khiển không tuyến tính, hoặc các phương pháp điều khiển thông minh như học sâu (deep learning) hoặc học tăng cường (reinforcement learning).

Cảm biến và phản hồi: Sử dụng dữ liệu từ các cảm biến để theo dõi trạng thái của robot và môi trường xung quanh. Thông tin này sau đó được sử dụng để điều chỉnh và

cải thiện việc điều khiển robot.

Tối ưu hóa hiệu suất: Bài toán điều khiển cũng liên quan đến việc tối ưu hóa hiệu suất của robot, bao gồm tối thiểu hóa thời gian thực hiện nhiệm vụ, tối thiểu hóa lỗi, hoặc tối ưu hóa tiêu chí cụ thể khác.

Điều kiện môi trường: Xác định và xử lý các yếu tố bên ngoài có thể ảnh hưởng đến việc điều khiển robot, như không gian làm việc, địa hình, và các điều kiện khí hậu.

Bài toán điều khiển trong robot yêu cầu sự kết hợp giữa kiến thức về lý thuyết điều khiển, lập trình, cơ học, và cảm biến để tạo ra các hệ thống robot thông minh và linh hoạt có khả năng hoạt động hiệu quả trong nhiều môi trường và tình huống khác nhau.

Điều khiển robot được hiểu là quá trình ra quyết định giúp robot hoàn thành được một mục tiêu đặt ra cụ thể. Nếu không có một hệ thống điều khiển tốt, một thiết bị robot sẽ hoàn toàn vô dụng. Một số thuật toán thường được sử dụng cho việc điều khiển robot như PID, mờ, trượt, thích nghi, hay mạng nơ-ron.

1.2. Các thuật toán điều khiển cánh tay robot

1.2.1. Điều khiển PID

a. Giới thiệu

Bộ điều khiển vi phân tỉ lệ (PID) là một hệ thống điều khiển phản hồi phổ biến được sử dụng rất nhiều và rộng rãi trong các ứng dụng công nghiệp. PID có thể được coi là một trong những bộ điều khiển phản hồi mà được áp dụng thường xuyên trong hệ thống điều khiển hiện nay

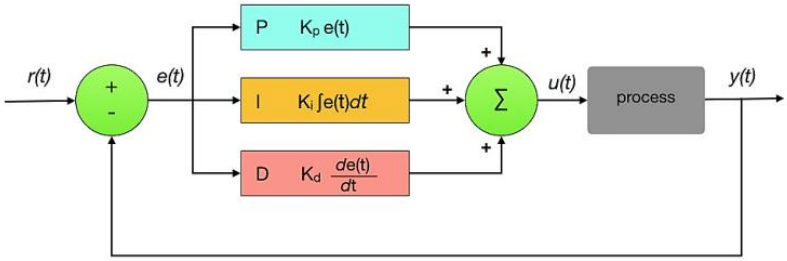
Thuật toán của bộ điều khiển PID bao gồm ba thông số đặc biệt, thường được gọi là ba yếu tố: tỷ lệ (P - Proportional), tích phân (I - Integral), và đạo hàm (D - Derivative). Tỷ lệ điều chỉnh dựa trên sai số hiện tại, tích phân điều chỉnh dựa trên tổng sai số tích luỹ từ quá khứ và đạo hàm điều chỉnh dựa trên tốc độ biến đổi của sai số..

Nhờ ba yếu tố này, chúng ta có thể hiểu rõ hơn về mối quan hệ thời gian trong hệ thống điều khiển: Yếu tố tỷ lệ (P) phụ thuộc vào sai số hiện tại, yếu tố tích phân (I) phụ thuộc vào tổng sai số tích luỹ từ quá khứ, và yếu tố đạo hàm (D) dự đoán các sai số tương lai dựa trên tốc độ biến đổi hiện tại.

b. Nguyên lý hoạt động

Tính chất khác biệt của bộ điều khiển PID là nó có khả năng sử dụng ba tham số điều khiển tỷ lệ, tích phân, đạo hàm để thay đổi đầu ra giúp việc điều khiển chính xác

và tối ưu. Nguyên lý hoạt động cơ bản của bộ điều khiển PID được thể hiện qua hình 1.6 dưới đây.



Hình 1.2. Bộ điều khiển PID[2]

Sai số $e(t) = y(t) - r(t)$ (trong đó $y(t)$ và $r(t)$ lần lượt là đầu ra phản hồi và đầu vào) được tính toán liên tục và dùng làm đầu vào cho bộ điều khiển. Nhiệm vụ khi đó của bộ PID là cố gắng tối thiểu hóa sai số này theo thời gian bằng cách điều chỉnh biến đầu ra $u(t)$ theo công thức:

$$u(t) = P + I + D \tag{1.1}$$

Thành phần P tỷ lệ thuận với giá trị lỗi hiện tại. Cụ thể nếu sai số lớn thì đầu ra điều khiển sẽ lớn tương ứng bằng cách sử dụng hệ số khuếch đại K_p. Nếu chỉ sử dụng khâu tỷ lệ sẽ dẫn đến sai số giữa điểm mong muốn và biến trạng thái thực tế phản hồi về tỷ lệ với nhau. Việc này dẫn đến khi đạt trạng thái cân bằng có thể sai số tĩnh sẽ khác 0.

Thành phần I lưu giữ các giá trị sai số e trong quá khứ và tích lũy chúng để tạo ra thành phần tích phân. Vai trò của thành phần này giúp đảm bảo sai số tĩnh trở về 0. Do trong hệ thống khi sai số e nhỏ tương ứng với phần tỷ lệ P sẽ nhỏ dẫn đến dừng hệ thống, lúc này thành phần tích phân với phần tích lũy trong quá khứ vẫn tăng và bù cho phần tỷ lệ, kéo sai số về 0. Khi sai số được loại bỏ, thành phần I sẽ dừng lại, không tăng nữa.

Thành phần D giúp ước lượng xu hướng trong tương lai của sai số e dựa trên tốc độ thay đổi hiện tại của nó. Đôi khi nó được gọi là điều khiển dự đoán do nó cố gắng giảm ảnh hưởng của sai số bằng cách sử dụng luật điều khiển tạo ra bởi tốc độ thay đổi của lỗi. Tốc độ thay đổi càng nhanh thì tác dụng điều khiển hay giảm chấn càng lớn.

[2] Proportional–integral–derivative controller. Wikipedia. Available at: https://en.wikipedia.org/wiki/Proportional–integral–derivative_controller

c. Phương trình toán học

Hàm đầu ra $u(t)$ có dạng toán hoc chính xác như sau:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \tag{1.2}$$

trong đó K_p, K_i, K_d là các số không âm, lần lượt biểu thị các hệ số cho thành phần tỷ lệ, tích phân và đạo hàm. Phương trình có thể được viết lại dưới dạng chuẩn như sau:

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right) \tag{1.3}$$

trong đó, K_i, K_d lần lượt được thay bằng $\frac{K_p}{T_i}$ và $K_d T_d$ do T_i, T_d có ý nghĩa vật lý dễ hiểu hơn - chúng đại diện cho thời gian tích phân và thời gian đạo hàm tương ứng. $K_d T_d$ là hằng số thời gian mà bộ điều khiển sẽ cố gắng tiếp cận điểm đích. $\frac{K_p}{T_i}$ xác định khoảng thời gian mà bộ điều khiển sẽ chịu được đầu ra ở trên hoặc dưới điểm đặt.

Trong miền tần số Laplace, bộ PID có hàm truyền như sau:

$$L(s) = K_p + \frac{K_i}{s} + K_d s \tag{1.4}$$

Giá trị đầu ra từ thành phần tỷ lệ sẽ tỉ lệ thuận với giá trị sai số hiện t. Khi nhân sai số với một hằng số K_p (hằng số khuếch đại tỉ lệ) thì đáp ứng tỷ lệ này có thể được điều chỉnh. Thành phần tỷ lệ được tính như sau

$$P_{out} = K_p e(t) \tag{1.5}$$

Một hằng số tỷ lệ lớn dẫn đến sự thay đổi lớn ở đầu ra với sai số đầu vào. Hệ thống sẽ bị mất ổn định trong trường hợp hệ số này quá cao. Ngược lại, hệ số nhỏ dẫn đến đáp ứng đầu ra nhỏ với sai số làm cho bộ điều khiển kém nhạy.

Thành phần tích phân tác động tới hệ thống phụ thuộc cả vào độ lớn cũng như chu kỳ của sai số. Tích phân trong PID là tổng của các sai số tức thời theo thời gian và tạo ra một giá trị offset tích lũy dần. Sai số đã tích lũy sau đó được nhân với hệ số tích phân K_i và cộng vào đầu ra của bộ điều khiển. Thành phần này được tính như sau

$$I_{out} = K_i \int_0^t e(\tau) d\tau \tag{1.6}$$

Thành phần tích phân giúp tăng tốc quá trình chuyển động tới điểm đích và và hạn chế sai số tĩnh của bộ điều khiển tỷ lệ thông thường. Tuy nhiên, nó có thể làm cho giá trị hiện tại vọt lố quá giá trị đích (overshoot) do quá trình tích lũy sai số trong quá khứ.

Thành phần đạo hàm là đạo hàm của sai số được tính bằng cách xác định độ dốc của sai số theo thời gian và nhân tốc độ thay đổi này với hệ số đạo hàm K_d. Thành phần này được tính như sau

$$D_{out} = K_d \frac{de(t)}{dt} \tag{1.7}$$

Đạo hàm sẽ giúp dự đoán ứng xử của hệ thống trong tương lai và do đó cải thiện thời gian thiết lập cũng như tính ổn định của hệ thống. Ý tưởng đạo hàm không theo luật nhân quả nên khi thực hiện ta thường cần thêm 1 bộ lọc thông thấp cho thành phần này để giới hạn thành phần tần số cao và nhiễu.

Mặc dù PID có ba tham số điều khiển nhưng trong một số ứng dụng ta chỉ cần tới một hoặc hai thành phần để tạo ra điều khiển đủ tốt. Điều này đạt được bằng cách đặt các hệ số không sử dụng bằng 0 tạo ra các biến thể như PI, PD, P hoặc I. Bộ điều khiển PI thường được dùng phổ biến trong các ứng dụng mà việc đạo hàm nhạy với nhiễu đo đạc nhưng thành phần tích phân là cần thiết để sai số tĩnh bằng 0. Thành phần D ít được dùng trong thực tế hơn do tác động thay đổi của nó có thể ảnh hưởng tới sự ổn định của hệ thống.

1.2.2. Điều khiển mờ

a. Khái niệm

Logic mờ là một dạng logic nhiều giá trị trong đó các giá trị chính xác của biến có thể là bất kì số thực nào nằm trong khoảng 0 đến 1 [3]. Nó được sử dụng để xử lý khái niệm về sự thật một phần, trong đó giá trị sự thật có thể nằm trong khoảng từ hoàn toàn đúng đến hoàn toàn sai.

b. Hàm liên thuộc

Hàm thuộc đánh giá mức độ mà các phần tử x thuộc tập X. Khi hàm trả về 0, phần

[3] Fuzzy logic .Wikipedia .Available at: https://en.wikipedia.org/wiki/Fuzzy_logic

tử không thuộc tập, còn 1 thể hiện thành viên hoàn toàn trong tập. Giá trị từ 0 đến 1 biểu thị các thành viên mờ. Các dạng hàm như Gaussian, PI-shape, S-shape, Sigmoidal, Z-shape, ... trong logic mờ đo lường mức độ thuộc tập của các phần tử.

c. Biến ngôn ngữ

Biến ngôn ngữ là phần tử cốt lõi trong các hệ thống dùng logic mờ dùng để mô tả các trạng thái có thể có của đối tượng. Như vậy các biến liên tục giờ đây sẽ được chia khoảng để mô tả và đại diện bằng các biến ngôn ngữ.

d. Luật hợp thành

Mệnh đề hợp thành có cấu trúc chung là:

"*Nếu A thì B*"

trong đó A là mệnh đề điều kiện, $C = A \Rightarrow B$ là mệnh đề kết luận. Mệnh đề này được dùng để mô tả mối quan hệ đầu ra với các điều kiện đầu vào.

Định lý Mamdani: khẳng định rằng độ tin cậy của kết luận trong một hệ thống suy luận không thể lớn hơn độ tin cậy của các điều kiện đầu vào. Trong trường hợp hệ thống có nhiều đầu vào và nhiều đầu ra, mệnh đề suy luận có dạng tổng quát như sau::

"If N = n_i and M = m_i and … Then R = r_i and K = k_i and …"

e. Các phương pháp giải mờ

Giải mờ là quá trình xác định giá trị rõ ở đầu ra từ hàm thuộc $\mu_R(y)$ của tập mờ R

Phương pháp điểm cực đại:

Tư tưởng chính của phương pháp này là tìm trong tập mờ có hàm thuộc $\mu_R(y)$ một giá trị y_0 với độ phụ thuộc lớn nhất, tức là:

$$y_0 = \arg\max \mu_R(y) \tag{1.8}$$

Các bước thực hiện:

Xác định miền chứa giá trị y_0 là giá trị mà tại đó $\mu_R(y)$ đạt Max.

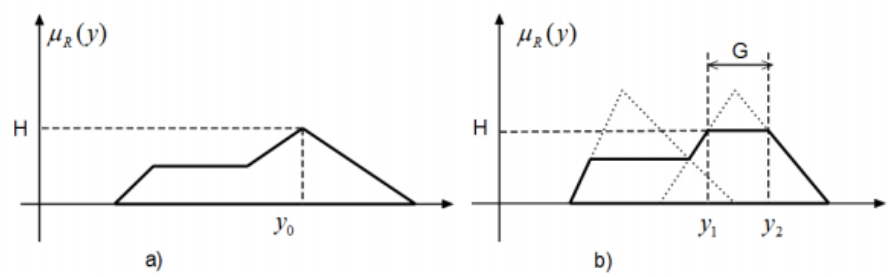
$$G = \{y \in Y \mid \mu_R(y) = H\} \tag{1.9}$$

Xác định y_0 từ G theo một trong 3 cách sau:

Nguyên lý trung bình: $y_0 = \frac{y_1 + y_2}{2}$ (1.10)

Nguyên lý cận trái: $y_0 = y_1$ (1.11)

Nguyên lý cận phải: $y_0 = y_2$ (1.12)



Hình 1.3. Phương pháp giải mờ cực đại

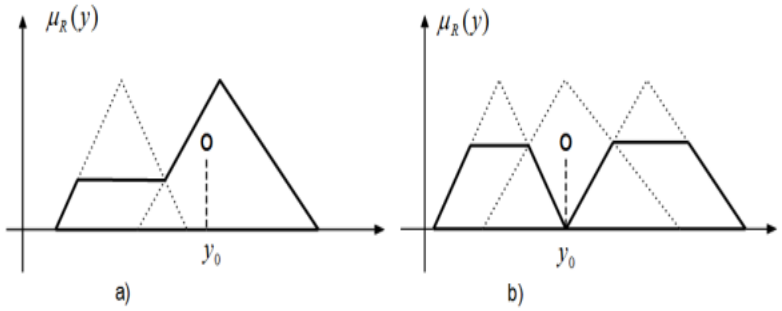
Phương pháp trọng tâm:

Phương pháp này cho kết quả y_0 là hoành độ điểm trọng tâm của miền được bao bởi trục hoành và đường $\mu_R(y)$. Công thức xác định là:

$$y_0 = \frac{\int_S y\mu_R(y)dy}{\int_S \mu_R(y)dy} \quad (1.13)$$

trong đó S là miền xác định của tập mờ R.

Đây là một trong nhiều phương pháp phổ biến. Nó cho phép xác định y_0 với sự đóng góp của toàn bộ các điểm trong tập mờ nên thu được đầu ra có thể coi là bình đẳng. Tuy nhiên phương pháp này không quan tâm tới độ thỏa mãn của mệnh đề điều khiển và thời gian tính lâu. Ngoài ra, một nhược điểm cơ bản là giá trị y_0 có thể có độ thuộc bằng 0 như hình b.

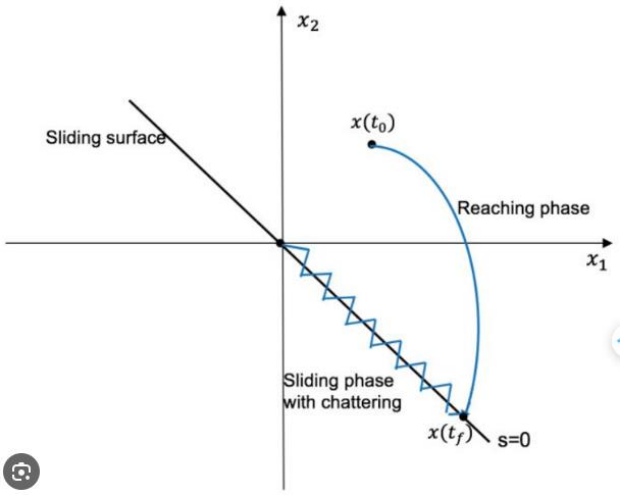


Hình 1.4. Giải mờ bằng phương pháp trọng tâm

1.2.3. Điều khiển trượt

Thuật toán điều khiển trượt (Sliding Mode Control - SMC) là một trong những phương pháp điều khiển không tuyến tính mạnh mẽ và linh hoạt được áp dụng rộng rãi trong nhiều lĩnh vực, từ robot học đến hệ thống tự động và điện tử công suất. Ý tưởng cơ bản của SMC là tạo ra một chế độ trượt (sliding mode) trong không gian trạng thái của hệ thống điều khiển để đảm bảo rằng hệ thống sẽ theo đuổi và duy trì một lối đi hoạt động ổn định dù có các nhiễu hoặc không chắc chắn trong hệ thống.

Cơ chế hoạt động của SMC bắt đầu bằng việc xác định một mục tiêu (hay một bề mặt trượt) trong không gian trạng thái mà hệ thống cần tiếp cận. Khi trạng thái của hệ thống không nằm trên bề mặt trượt, một đặc điểm của SMC là tạo ra một lực điều khiển đủ mạnh để đẩy trạng thái của hệ thống dẫn tới bề mặt trượt. Khi trạng thái hệ thống đạt đến bề mặt trượt, lực điều khiển giảm đáng kể hoặc biến mất, và hệ thống tiếp tục di chuyển trên bề mặt trượt theo một cách động học ổn định.



Hình 1.5. Hoạt động của điều khiển trượt

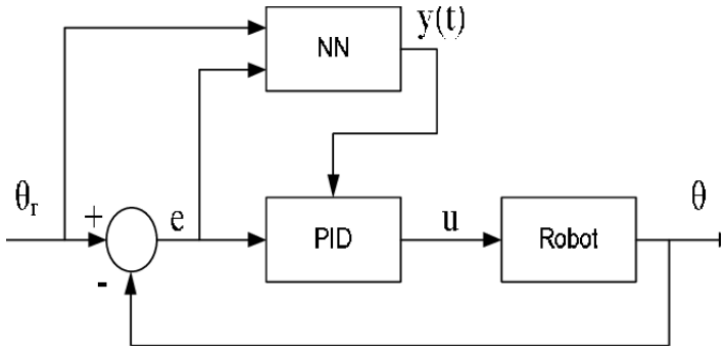
Một trong những ưu điểm lớn của SMC là khả năng chống lại nhiễu và không chắc chắn trong hệ thống, đặc biệt là trong môi trường làm việc thực tế có nhiều yếu tố không lường trước. Điều này là do chế độ trượt được thiết kế để đảm bảo rằng hệ thống sẽ di chuyển trên bề mặt trượt dù có nhiễu hay thay đổi tham số không chắc chắn.

Tuy nhiên, việc thiết kế và triển khai thuật toán điều khiển trượt không phải lúc nào cũng dễ dàng. Cần phải xác định chính xác bề mặt trượt và đặc tính chuyển động của hệ thống để có thể áp dụng SMC hiệu quả. Ngoài ra, việc chuyển đổi từ trạng thái nằm ngoài bề mặt trượt sang trạng thái trượt cũng có thể gây ra các vấn đề không mong muốn như hiện tượng động rơi hay rung lắc.

Thuật toán điều khiển trượt là một công cụ mạnh mẽ và linh hoạt cho việc điều khiển các hệ thống không tuyến tính trong môi trường không chắc chắn. Tuy nhiên, việc áp dụng nó đòi hỏi kiến thức chuyên sâu về lý thuyết điều khiển và kỹ thuật cao để đảm bảo hiệu quả và ổn định của hệ thống điều khiển.

1.2.4. Điều khiển nhờ mạng nơ-ron.

Điều khiển với mạng nơ-ron (Neural Network Control) là một trong những lý thuyết phổ biến và quan trọng trong điều khiển tự đông, trong đó đề cập đến việc sử dụng các mạng nơ-ron nhân tạo để điều khiển hệ thống. Mạng nơ-ron là một mô hình toán học lấy cảm hứng từ cấu trúc và hoạt động của não người, nó bao gồm các "nút" (neurons) kết nối với nhau để xử lý thông tin.



Hình 1.6. Sử dụng mạng nơ-ron hỗ trợ điều chỉnh thông số của PID [10]

Trong điều khiển nhờ mạng nơ-ron, mô hình mạng nơ-ron được sử dụng để ước lượng, dự đoán hoặc điều khiển hệ thống. Có hai phương pháp chính thường được áp dụng trong điều khiển nhờ mạng nơ-ron:

Mô hình dự đoán (Predictive Model): Mạng nơ-ron được sử dụng để xây dựng một mô hình dự đoán hoặc mô hình tương tác với hệ thống. Dựa trên dữ liệu đầu vào và quá trình huấn luyện, mạng nơ-ron có khả năng học và dự đoán các trạng thái hoặc xu hướng của hệ thống trong tương lai. Mô hình này sau đó có thể được sử dụng để điều khiển hệ thống.

Điều khiển trực tiếp (Direct Control): Trong phương pháp này, mạng nơ-ron được sử dụng trực tiếp để tạo ra tín hiệu điều khiển mà không cần một mô hình điều khiển truyền thống. Mạng nơ-ron có thể được thiết kế để học và ánh xạ các đầu vào từ hệ thống thành các tín hiệu điều khiển phù hợp.

Một trong những ưu điểm lớn của điều khiển nhờ mạng nơ-ron là khả năng học và tự điều chỉnh dựa trên dữ liệu thực tế, điều này có thể giúp nó thích ứng tốt hơn với các

biến đổi không gian, thời gian và điều kiện môi trường so với các phương pháp điều khiển cổ điển. Ngoài ra, các mạng nơ-ron có khả năng giải quyết các tình huống phức tạp và không gian đầu vào lớn.

Tuy nhiên, việc sử dụng mạng nơ-ron trong điều khiển cũng đặt ra một số thách thức. Khi huấn luyện mạng nơ-ron, một lượng dữ liệu lớn và đa dạng để đảm bảo được tính chính xác của hệ thống là một điều cần thiết. Ngoài ra, khả năng giải thích và kiểm soát được mạng nơ-ron cũng là một vấn đề quan trọng đối với việc triển khai trong các hệ thống yêu cầu tính tin cậy cao.

Điều khiển nhờ mạng nơ-ron đại diện cho sự hứa hẹn trong việc áp dụng công nghệ trí tuệ nhân tạo vào lĩnh vực điều khiển tự động. Sự kết hợp giữa khả năng học và linh hoạt của mạng nơ-ron có thể mở ra những cơ hội mới và cải tiến đáng kể trong hiệu suất và tự động hóa của hệ thống điều khiển.

1.3. Xác định bài toán điều khiển của đồ án

Trong đồ án này, để giải quyết bài toán điều khiển cho cánh tay robot, tôi đã sử dụng mạng nơ-ron cho việc đó. Với giá trị đầu vào của mạng nơ-ron là tọa độ các điểm theo quỹ đạo mong muốn và giá trị đầu ra là các biến khớp. Bộ dữ liệu đào tạo mạng nơ-ron được lấy từ tập hợp nhiều quỹ đạo sau khi giải quyết bài toán động học ngược với phương pháp agv

Kết luận Chương 1

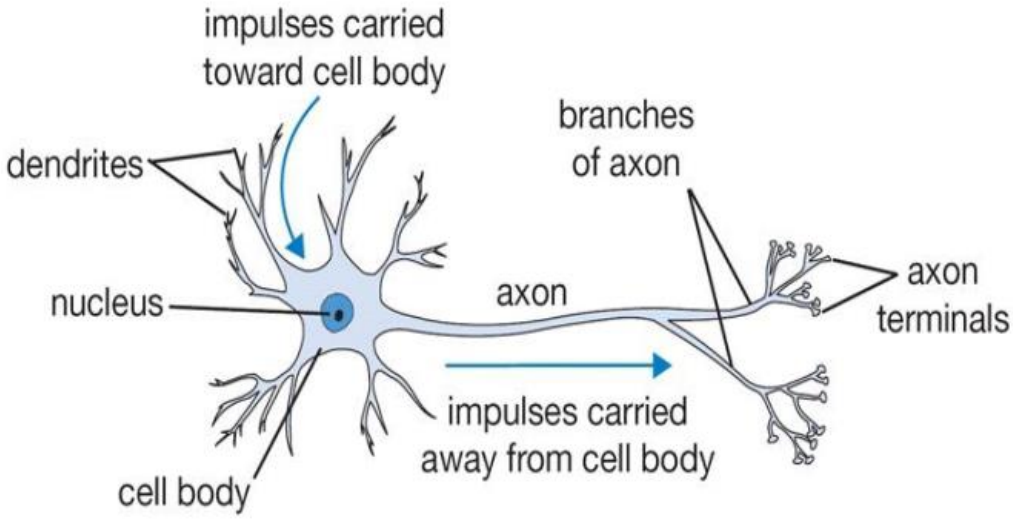
Chương này trình bày về khái niệm, cấu tạo và phân loại các loại robot công nghiệp phổ biên, cùng với đó là nêu được tổng quan các thuật toán điều khiển cho robot và xác định bài toán điều khiển cụ thể cho đồ án.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VỀ ĐIỀU KHIỂN NƠ-RON

2.1. Cơ bản về mạng nơ-ron

2.1.1. Lí thuyết về mạng nơ-ron nhân tạo

a. Hoạt động của mạng nơ-ron thần kinh



Hình 2.1. Mô hình nơ-ron thần kinh.[4]

Nơ-ron là thành phần cơ bản của hệ thống thần kinh và đóng vai trò quan trọng trong cấu trúc não. Một bộ não trung bình của con người chứa khoảng 10 triệu nơ-ron và mỗi nơ-ron này kết nối với khoảng 10.000 nơ-ron khác [1].

Hình 2.1 thể hiện cấu trúc của nơ-ron thần kinh. Mỗi nơ-ron bao gồm phần thân (soma) chứa nhân, nơi các tín hiệu đầu vào được tiếp nhận qua các nhánh dendrites, và các tín hiệu đầu ra được gửi đi qua sợi trục (axon) giao tiếp với các nơ-ron khác. Mỗi nơ-ron nhận thông tin đầu vào qua các nhánh dendrites và truyền thông tin đầu ra qua sợi trục, kết nối với các nhánh dendrites của nơ-ron khác.

Xung điện từ các nơ-ron khác được truyền tới mỗi nơ-ron qua các nhánh dendrites. Khi các xung điện này đạt mức đủ lớn để kích hoạt nơ-ron, tín hiệu này sẽ tiếp tục đi qua sợi trục đến các nhánh dendrites của các nơ-ron khác. Điều này ám chỉ rằng, ở mỗi nơ-ron, quyết định về việc kích hoạt nó hay không được thực hiện dựa trên việc xử lý các tín hiệu này.

[4] Standford University, "CS231n Convolutional Neural Networks for Visual Recognition" – https://cs231n.github.io/neural-networks-1/

b. Mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo được lấy cảm ứng từ bộ não của con người và nó có thể được sử dụng cho bài toán về học máy và trí tuệ nhân tạo. Với những bộ mạng đó, nhiều vấn đề có thể được giải quyết với sự hỗ trợ và tính toán của máy tính.

Mạng nơ-ron chứa các thành phần gồm các nút kết nối với nhau, tạo thành các lớp. Mỗi nút biểu thị một đơn vị tính toán, và trong một mạng nơ-ron đa tầng, các nút thường được tổ chức thành các lớp liên kết với nhau. Lớp đầu tiên thường là lớp đầu vào, nhận dữ liệu đầu vào, trong khi lớp cuối cùng là lớp đầu ra, cung cấp kết quả dự đoán hoặc phân loại.

Mỗi nút trong mạng nơ-ron có thể được xem xét như một phần tử tính toán đơn giản, thường có cấu trúc tương tự với hàm toán học đa biến. Các mối quan hệ giữa các nút được biểu diễn quan trọng số liên kết giữa chúng. Các lớp giữa lớp đầu vào và lớp đầu ra được xem xét là các lớp ẩn, trong đó mỗi lớp ẩn thực hiện các phép biến đổi từ lớp trước đó sang lớp tiếp theo.

2.1.2. Cấu trúc mạng nơ-ron

a. Các lớp trong mạng nơ-ron

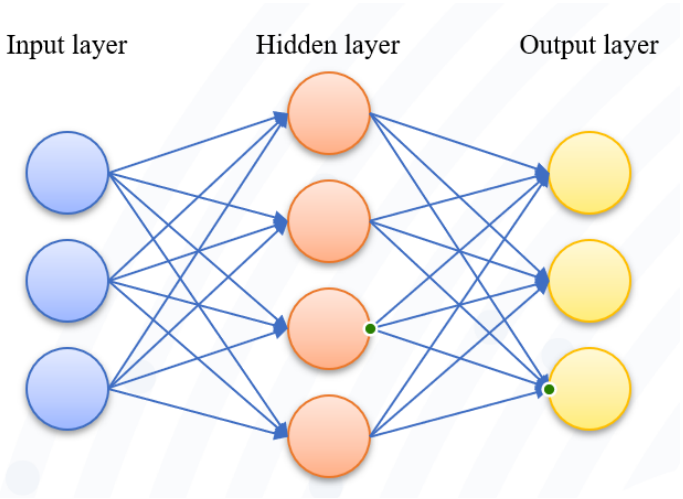
Mạng nơ-ron là được coi như là sự tổng hợp của các perception hay có thể gọi nó là perceptron đa tầng. Và mỗi một mạng nơ-ron thường bao gồm 3 kiểu lớp là [1]:

Lớp đầu vào (input layer): Lớp này là lớp đầu tiên của mạng nằm bên trái cùng của mạng, thể hiện cho các tính năng đầu vào của mạng.

Lớp đầu ra (output layer): Đây là lớp cuối cùng của mạng thể hiện cho những yêu cầu đầu ra cần thiết.

Lớp ẩn (hidden layer): Đây là một lớp vô cùng quan trọng nằm giữa 2 lớp đầu vào và lớp đầu ra thể hiện được quá trình học, suy luận logic của mạng nơ-ron. Ta cần phải sử dụng lớp ẩn vì với lớp này là một lớp không thể thiếu trong mạng nơ-ron để có thể giúp ta giải quyết và biểu diễn các mô hình phức tạp

Lưu ý: Mỗi một nơ-ron chỉ có duy nhất một lớp đầu vào và một lớp đầu ra nhưng có thể có nhiều lớp ẩn.



Hình 2.2. Mô hình cấu trúc của một mạng nơ-ron gồm 3 lớp

b. Hàm kích hoạt

Đối với một mạng nơ-ron nhân tạo, output của mỗi nơ-ron bao gồm các hàm kích hoạt – đóng vai trò là thành phần phi tuyến. Chúng ta cần phải sử dụng hàm kích hoạt phi tuyến vì nếu như không có nó thì mạng nơ-ron cũng chỉ coi như là một lớp mặc dù nó bao gồm rất nhiều lớp. Một số hàm phổ biến như sigmoid, tanh, ReLU.

c. Hàm mất mát (Loss function)

Hàm mất mát thường có kí hiệu là L – là thành phần cốt lõi trong mạng nơ-ron. Cụ thể được thể hiện trong công thức 2.5.

$$L_D(f_w) = \frac{1}{|D|} \sum_{(x,y)\in D} L(f_w(x), y) \tag{2.1}$$

Sự chênh lệch giữa dự đoán và thực tế được thể hiện qua kết quả của hàm mất mát thu (một số thực không âm). Loss function được coi như là một cách để phạt mô hình mỗi khi nó dự đoán sai và mức phạt này sẽ tỷ lệ thuận với độ lớn của sai số.

Các dạng hàm mất mát phổ biến có thể được sử dụng là:

Square loss:

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2 \tag{2.2}$$

Khi tính đạo hàm theo $\hat{y}$ ta được $\nabla L = \hat{y} - y$. Giá trị 1/2 được thêm vào công thức hàm mất mát chỉ có tác dụng để thu được đạo hàm đẹp hơn, không có hằng số phụ. Đây là dạng hàm mất mát đơn giản, trực quan và nó có đạo hàm tại mọi điểm nên rất thuận

lợi trong tính toán. Thực tế square loss có thể được dùng cho cả bài toán hồi quy và bài toán phân loại, tuy nhiên nó được dùng nhiều hơn cho bài toán hồi quy.

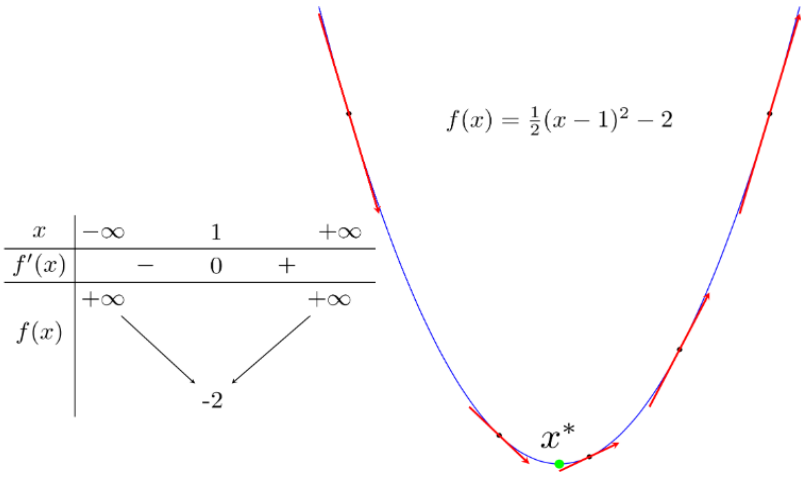
Cross entropy:

$$CE = -\sum_{i}^{C} t_i \log(s_i) \tag{2.3}$$

trong đó, t_i là giá trị chính xác và s_i là giá trị dự đoán. Dạng hàm mất mát này chủ yếu được dùng cho các bài toán phân loại. So với hàm bình phương khoảng cách thông thường, cross entropy có đặc điểm là nó nhận giá trị rất lớn (loss cao) khi giá trị dự đoán cách xa giá trị chính xác. Điều này có nghĩa hàm cross entropy sẽ cho nghiệm chính xác hơn vì những điểm ở xa bị trừng phạt rất nặng.

2.1.3. Phương pháp tối ưu hàm mất mát Gradient Descent (GD)

Đối với mạng nơ-ron, việc tìm giá trị nhỏ nhất của một hàm số xảy ra thường xuyên. Việc này nhiều khi là rất phức tạp hay thậm chí là bất khả thi trong một số mô hình học máy. Do đó, người ta sẽ tìm những điểm cực tiểu cục bộ để xem nói như là một nghiệm của bài toán. Khi giải phương trình đạo hàm bằng 0, nghiệm của chúng là các điểm cực tiểu cục bộ. Tuy nhiên, điều này có thể không được giải quyết trong một số trường hợp vì sự phức tạp của các dạng đạo hàm, số chiều của dữ liệu hay số lượng điểm dữ liệu. Do đó, có một phương pháp là bắt đầu từ một điểm (gần với nghiệm của bài toán), tiếp đó sử dụng vòng lặp để tiến đến nghiệm cần tìm cho đến khi đạo hàm bằng 0. Phương pháp này được gọi là Gradient Descent.



Hình 2.3. Gradient descent cho hàm 1 biến[5]

Ý tưởng của thuật toán tương tự như hình 2.7: x_t nằm bên phải so với x^* nếu x_t làm cho đạo hàm của hàm số tại đó lớn hơn 0 và ngược lại. Chúng ta cần di chuyển x_t về phần âm, bên trái để điểm tiếp theo x_{t+1} gần với x^* hơn, đây chính là việc di chuyển giá trị đó ngược hướng với dấu của đạo hàm

Để tổng quát, ta giả sử cần tìm cực tiểu cục bộ cho hàm $f(\theta)$, trong đó θ là một véc-tơ (biển diễn các tham số của mô hình cần tối ưu). Đạo hàm của hàm số tại một điểm θ bất kỳ được ký hiệu là $\nabla_\theta f(\theta)$. Khi đó, ta khởi tạo θ_0 bất kỳ, sau đó ở vòng lặp thứ t, quy tắc cập nhật là:

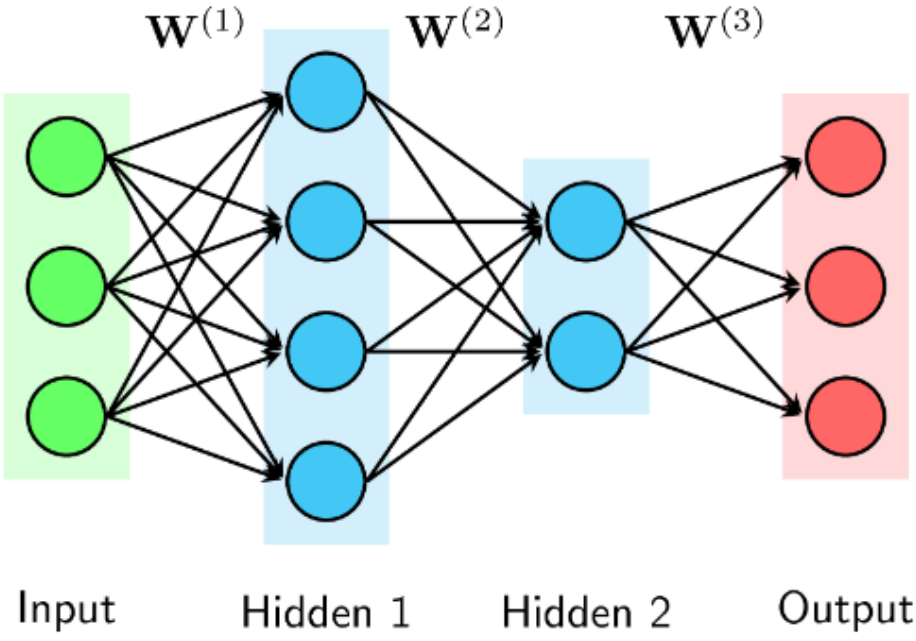
$$\theta_{t+1} = \theta_t - \eta \nabla_\theta f(\theta) \tag{2.4}$$

trong đó η gọi là tốc độ học (dấu trừ thể hiện việc đi ngược lại đạo hàm). Trong bài toán thực tế lựa chọn η cần thiết. Công việc này phụ thuộc rất nhiều vào bài toán và phải làm rất nhiều thực nghiệm để đưa ra được giá trị lựa chọn tốt nhất.

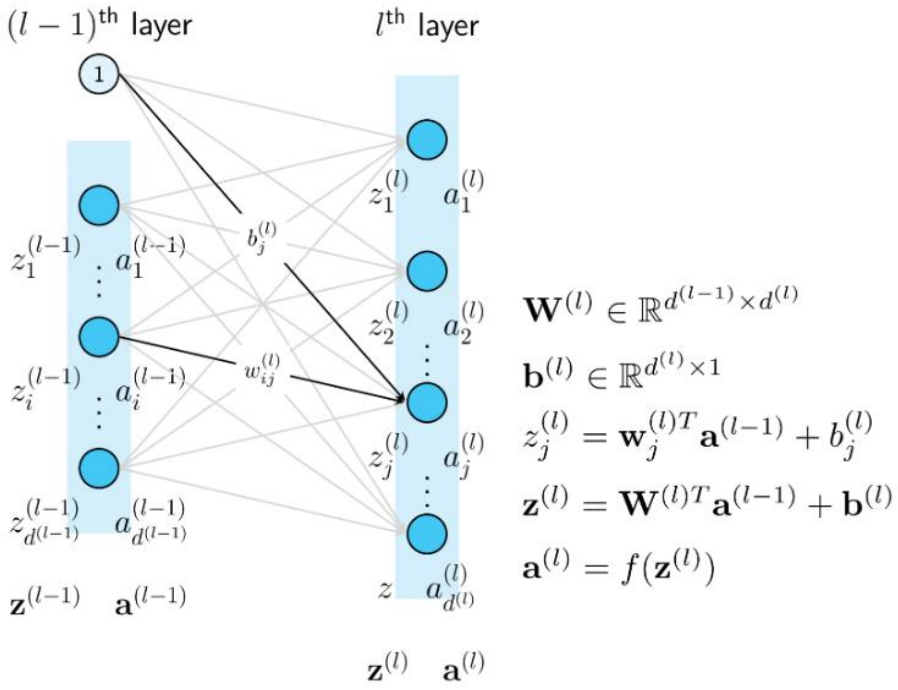
2.1.4. Thuật toán lan truyền ngược (Backpropagation)

Phương pháp tối ưu gradient descent (GD) trình bày ở phần trên chỉ sử dụng cho các mạng đơn giản không có lớp ẩn. Dựa trên GD, để có thể ứng dụng nó cho các mạng nơ-ron nhiều lớp, ta cần sử dụng đến thuật toán lan truyền ngược.

[5] Vũ Hữu Tiệp, "Bài 7: Gradient Descent" – https://machinelearningcoban.com/2017/01/12/gradientdescent/



Hình 2.4. Mạng NN với các lớp ẩn



Hình 2.5. Liên hệ giữa các lớp

Khi sử dụng GD, chúng ta cần tính được đạo hàm của hàm mất mát theo từng ma trận trọng số $W^{(l)}$ và véc-tơ bias $b^{(l)}$. Trước hết, chúng ta cần tính đầu ra dự đoán $\hat{y}$ với một input x:

$$
\begin{aligned}
&a^{(0)} = x \\
&z_i^{(l)} = \left(w_i^{(l)}\right)^T a^{(l-1)} + b_i^{(l)} \\
&a^{(l)} = f\left(z^{(l)}\right),\ l = 1,2,...,L \\
&\hat{y} = a^{(L)}
\end{aligned}
\tag{2.5}
$$

Đây này được gọi là lan truyền ngược vì cách tính toán được thực hiện từ đầu đến cuối của mạng.

Giả sử một hàm mất mát của bài toán là $J\left(W,b,X,Y\right)$, trong đó $\boldsymbol{W,b}$ là tập hợp tất cả các ma trận trọng số giữa các layer và bias của mỗi layer. X,Y là cặp dữ liệu huấn luyện.

Ở đây ta sử dụng hàm mất mát là hàm MSE (mean square error):

$$
J\left(W,b,X,Y\right) = \frac{1}{N}\sum_{n=1}^{N}\left\|y_n - \hat{y}_n\right\|_2^2 = \frac{1}{N}\sum_{n=1}^{N}\left\|y_n - a_n^{(L)}\right\|_2^2 \tag{2.6}
$$

với N là số cặp dữ liệu (x, y) trong tập training.

Hàm mất mát không phụ thuộc trực tiếp vào các hệ số nên phương pháp hay được sử dụng nhất có tên là backpropagation – phương pháp này giúp tính toán đạo hàm ngược từ lớp đầu. Lớp cuối cùng được tính trước đó do nó gần với đầu ra của mạng và hàm mất mát. Việc tính đạo hàm của các lớp trước được dựa trên quy tắc "chain rule".

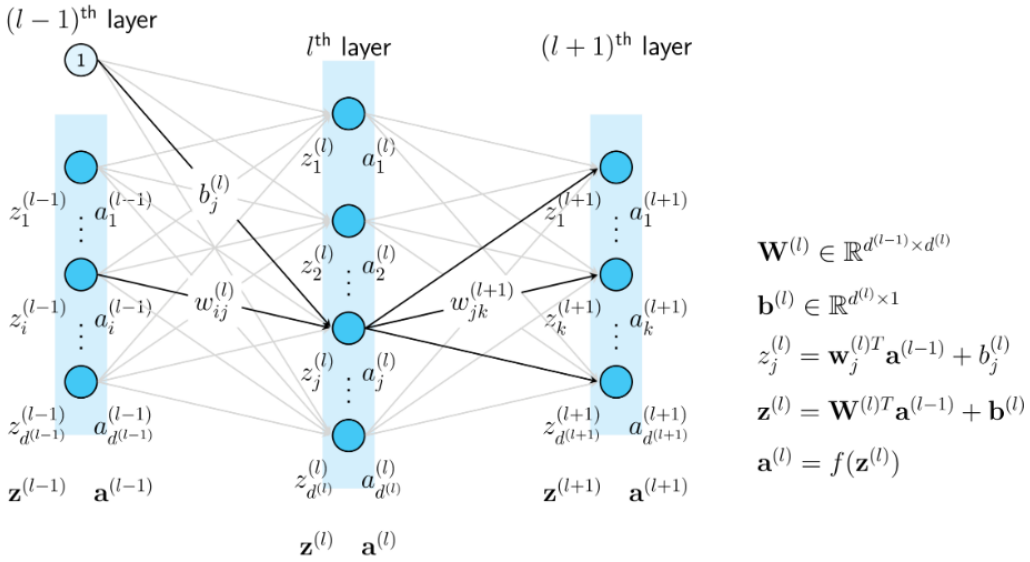
Đạo hàm của hàm mất mát theo chỉ một thành phần của ma trận trọng số của lớp cuối cùng:

$$
\frac{\partial J}{\partial w_{ij}^{(L)}} = \frac{\partial J}{\partial z_j^{(L)}} \cdot \frac{\partial z_j^{(L)}}{\partial w_{ij}^{(L)}} = e_j^{(L)} a_i^{(L-1)} \tag{2.7}
$$

trong đó $e_j^{(L)} = \dfrac{\partial J}{\partial z_j^{(L)}}$ thường dễ tính toán và $\dfrac{\partial z_j^{(L)}}{\partial w_{ij}^{(L)}} = a_i^{(L-1)}$ vì $z_j^{(L)} = \left(w_j^{(L)}\right)^T a^{(L-1)} + b_j^{(L)}$.

Tương tự, đạo hàm của hàm mất mát theo bias của layer cuối là:

$$
\frac{\partial J}{\partial b_j^{(L)}} = \frac{\partial J}{\partial z_j^{(L)}} \cdot \frac{\partial z_j^{(L)}}{\partial b_j^{(L)}} = e_j^{(L)} \tag{2.8}
$$



Hình 2.6. Mô phỏng cách tính backpropagation

Hình 2.10 mô phỏng cách tính backpropagation của một mạng gồm 3 lớp liên tiếp nhau. Sau khi đã tính được đạo hàm, công thức cập nhật trọng số được áp dụng tương tự như thuật toán gradient descent.

2.1.5. Thuật toán Levenberg-Margquardt

Có rất nhiều giải thuật có thể sử dụng cho việc dùng mạng nơ-ron vào bài toán động học ngược. Nhưng trong đồ án này sẽ sử dụng phương pháp Levenberg-Marquardt. Kenneth Levenberg và Donald Marquart là cha đẻ của thuật toán này, thể hiện một phương pháp để giải quyết việc cực tiểu hóa của một hàm không tuyến tính. Giải pháp này hội tụ rất nhanh và ổn định so với phương pháp gradient descent. Thuật toán này khá là phù hợp cho việc đào tạo và huấn luyện những mạng có kích thước nhỏ và trung bình (khoảng vài tram trọng số).

Luật cập nhật trọng số

Hàm tổng bình phương sai số ‘mse’ (mean square error) sẽ được dùng trong hàm mất mát được tính như sau:

$$E(p,w) = \frac{1}{2}\sum_{i=1}^{m}\sum_{t=1}^{n_Q} e^2 \tag{2.9}$$

trong đó:

$E(p,w)$: Hàm tổng bình phương sai số.

m: Số lượng mẫu đào tạo.

n_Q : Số khớp đầu ra cần tìm.

e: Sai số của tín hiệu ra khi duyệt n mẫu được định nghĩa:

$$(q\_pre - q\_true)^2 \tag{2.10}$$

trong đó:

q_true : Vector khớp đầu ra mong muốn.

q_pre : Vector khớp đầu ra dự đoán.

Giải thuật sử dụng luật cập nhật trọng số:

$$w_{k+1} = w_k - (J_k^T J_k \_ \mu I)^{-1} J_k e_k \tag{2.11}$$

với:

μ : là hệ số tốc độ học

k : là chỉ số lặp

J: Ma trận Jacobian có dạng:

$$J = \begin{bmatrix} \frac{\partial e_1(w)}{\partial w_1} & \frac{\partial e_1(w)}{\partial w_2} & \cdots & \frac{\partial e_1(w)}{\partial w_2} \\ \frac{\partial e_2(w)}{\partial w_1} & \frac{\partial e_2(w)}{\partial w_2} & \cdots & \frac{\partial e_2(w)}{\partial w_2} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_N(w)}{\partial w_1} & \frac{\partial e_N(w)}{\partial w_2} & \cdots & \frac{\partial e_N(w)}{\partial w_N} \end{bmatrix} \tag{2.12}$$

Tính toán ma trận Jacobian

Vector sai số có dạng cụ thể là:

$$e^T = [e_{1,1} e_{1,2} ... e_{1,n}; ....; e_{m,1} e_{m,2} ... e_{m,n_Q}] \tag{2.13}$$

Vector tham số có dạng:

$$w^T = [w_1 w_2 ... w_N] \tag{2.14}$$

trong đó:

n_i là số nơron trong lớp đầu vào.

n_1 là số nơron ở lớp thứ nhất.

n_2 là số nơron ở lớp thứ hai.

n_Q là số nơron trong lớp ra.

Chỉ số trên thể hiện cho thứ tự của lớp

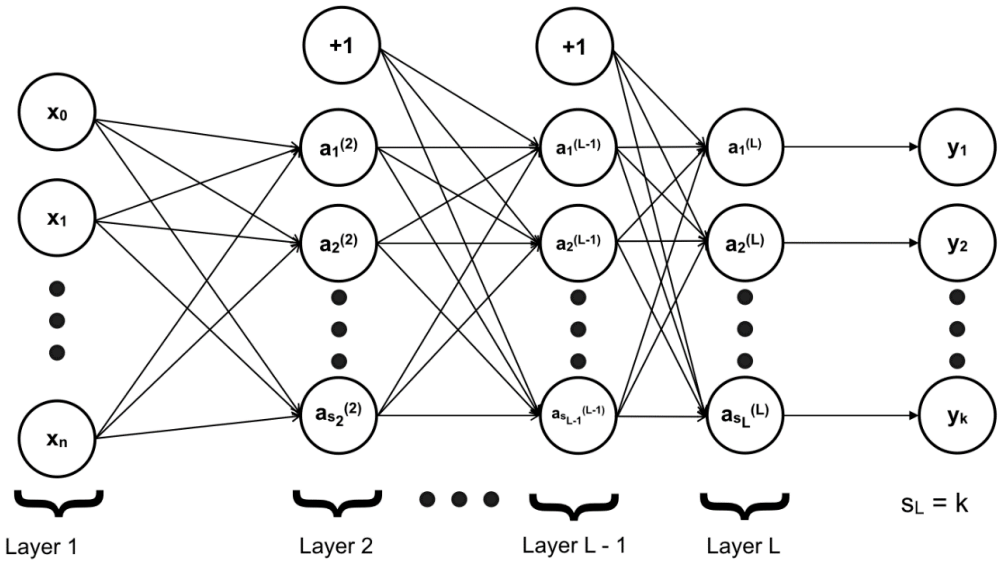
Chỉ số ‘Q’ thể hiện cho lớp ra.

Khi đó ma trận Jacobian được viết lại như sau:

$$J = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w^1_{1,1}} & \frac{\partial e_{1,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,1}}{\partial w^1_{n_1,n_i}} & \frac{\partial e_{1,1}}{\partial b^1_1} & \cdots \\ \frac{\partial e_{1,2}}{\partial w^1_{1,1}} & \frac{\partial e_{1,2}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,2}}{\partial w^1_{n_1,n_i}} & \frac{\partial e_{1,2}}{\partial b^1_1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{1,n_Q}}{\partial w^1_{1,1}} & \frac{\partial e_{1,n_Q}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,n_Q}}{\partial w^1_{n_1,n_i}} & \frac{\partial e_{1,n_Q}}{\partial b^1_1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{p,1}}{\partial w^1_{1,1}} & \frac{\partial e_{p,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{p,1}}{\partial w^1_{n_1,n_i}} & \frac{\partial e_{p,1}}{\partial b^1_1} & \cdots \\ \frac{\partial e_{p,2}}{\partial w^1_{1,1}} & \frac{\partial e_{p,2}}{\partial w^1_{1,1}} & \cdots & \frac{\partial e_{p,2}}{\partial w^1_{n_1,n_i}} & \frac{\partial e_{p,2}}{\partial b^1_1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{p,n_Q}}{\partial w^1_{1,1}} & \frac{\partial e_{p,n_Q}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{p,n_Q}}{\partial w^1_{n_1,n_i}} & \frac{\partial e_{p,n_Q}}{\partial b^1_1} & \cdots \end{bmatrix} \quad (2.15)$$

Áp dụng hàm kích hoạt cho hàm kết hợp:

$$a^{[L]} = f(z^{[L]}) = f(w^T a^{[L-1]} + b) \quad (2.16)$$



Hình 2.7. Mô hình của một mạng nơron nhiều lớp [6]

Tính đạo hàm của tổng trọng z là:

$$\frac{\partial z^{[L]}}{\partial w} = a^{[L-1]} \tag{2.17}$$

Độ dốc grad của hàm kích hoạt là:

$$grad = \frac{\partial a^{[L]}}{\partial z^{[L]}} = \frac{\partial f(z^{[L]})}{\partial z^{[L]}} \tag{2.18}$$

Sau đó dùng hàm lỗi để tính sai lệch đầu ra và lan truyền ngược lại tìm được các thông số.

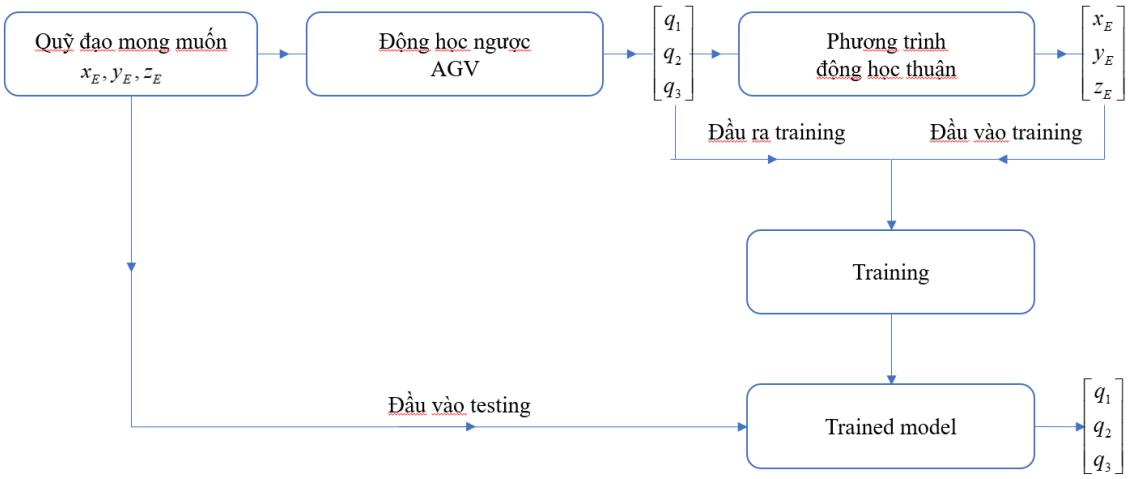
2.2. Phương pháp điều khiển bằng mạng nơ-ron

2.2.1. Mục tiêu điều khiển

Trong đồ án này, mục tiêu điều khiển cho robot là việc sử dụng bài mạng nơ-ron để đưa ra được các giá trị biến khớp với đầu vào là quỹ đạo mong muốn. Từ đó đưa được robot mô hình chạy đúng được theo quỹ đạo mong muốn. Mạng nơ-ron được sử dụng để thực hiện việc đó được đào tạo từ việc đưa các quỹ đạo ban đầu vào mạng train để lấy được các biến khớp cho trước.

2.2.2. Mô hình bài toán

[6] Trần Thế Anh, "Understanding Neural Network 1/3" - https://trantheanh.github.io/2016/10/18/ML-07/



Hình 2.8. Mô hình bài toán

a. Đầu vào

Đầu vào của quá trình đào tạo mạng nơ-ron cho việc thực hiện điều khiển robot được lấy từ bài toán việc thay các giá trị biến khớp được giải từ bài toán động học ngược với phương pháp agv sau đó thay vào phương trình động học thuận sẽ ra được tập các tọa độ điểm cuối là đầu vào của mạng

b. Đầu ra

Đầu ra của quá trình đào tạo mạng nơ-ron cho việc thực hiện điều khiển robot được lấy từ bài toán việc thay các giá trị biến khớp được giải từ bài toán động học ngược với phương pháp agv.

2.2.3. Thuật toán

Bài toán này sử dụng thuật toán Levenberg-Margquardt (đã được đề cập trong phần 2.1.5) để đào tạo mạng nơ-ron. Đây là một thuật toán được sử dụng khá rộng rãi trong việc đào tạo mạng nơ-ron với những ưu điểm sau:

Tính chất hội tụ cao: Levenberg-Margquardt thường cho kết quả hội tụ nhanh chóng hơn so với các phương pháp tối ưu hóa khác như gradient descent thông thường. Điều này là do LM kết hợp cả hai phương pháp: Gauss-Newton và gradient descent, từ đó tận dụng ưu điểm của cả hai để tối ưu hóa mô hình nhanh hơn.

Hiệu suất với bài toán phi tuyến: Khi áp dụng cho mạng nơ-ron với hàm kích hoạt phi tuyến, như hàm sigmoid hay tanh, thuật toán này thường hiệu quả hơn so với các phương pháp tối ưu hóa khác, nhất là khi mô hình có nhiều tham số.

Tự điều chỉnh tốc độ học: LM có khả năng tự điều chỉnh tốc độ học (learning rate) trong quá trình huấn luyện, giúp tránh được vấn đề về việc chọn tốc độ học cố định.

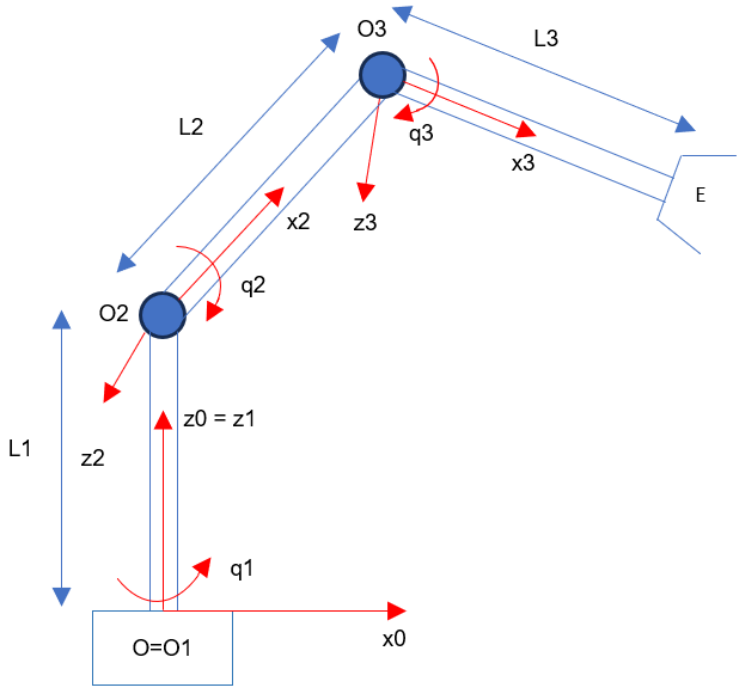
Kết luận chương 2

Chương này nói về các lý thuyết cơ bản nhất về mạng nơ-ron bao gồm cấu tạo, phương pháp giải và phương pháp điều khiển, cùng với đó cho thấy được mục tiêu mô hình và thuật toán thực hiện với mạng nơ-ron trong đồ án.

CHƯƠNG 3. MÔ HÌNH HÓA ĐỘNG HỌC VÀ ĐIỀU KHIỂN CÁNH TAY ROBOT 3DOF

3.1. Mô hình toán học cánh tay robot

3.1.1. Phân tích mô hình toán học



Hình 3.1. Mô hình robot

3.1.2. Hệ phương trình động học

Xây dựng bảng DH:

Khâu	θ_i	d_i	a_i	α_i
$O_0 \to O_1$	q_1	0	0	0
$O_1 \to O_2$	0	L_1	0	$\frac{\pi}{2}$
$O_2 \to O_3$	q_2	0	L_2	0
$O_3 \to E$	q_3	0	L_3	0

Bảng 3.1. Bảng thông số DH

Ma trận chuyển đổi thuần nhất hệ $O_0X_0Y_0Z_0$:

$$H_{01}=\begin{bmatrix}\cos(q_1) & -\sin(q_1) & 0 & 0\\ \sin(q_1) & \cos(q_1) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.1}$$

Ma trận chuyển đổi thuần nhất hệ $O_1X_1Y_1Z_1$:

$$H_{12}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 1 & 0 & L_1\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.2}$$

Ma trận chuyển đổi thuần nhất hệ $O_2X_2Y_2Z_2$:

$$H_{23}=\begin{bmatrix}\cos(q_2) & -\sin(q_2) & 0 & L_2.\cos(q_2)\\ \sin(q_2) & \cos(q_2) & 0 & L_2.\sin(q_2)\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.3}$$

Ma trận chuyển đổi thuần nhất hệ $O_3X_3Y_3Z_3$:

$$H_{3E}=\begin{bmatrix}\cos(q_3) & -\sin(q_3) & 0 & L_3.\cos(q_3)\\ \sin(q_3) & \cos(q_3) & 0 & L_3.\sin(q_3)\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.4}$$

Ma trận chuyển đổi thuần nhất khâu thao tác cuối:

$$D_{3E}=\begin{bmatrix}c_1(c_2c_3\text{-}s_2s_3) & \text{-}c_1(c_2s_3+c_3s_2) & s_1 & L_3c_1(c_2c_3\text{-}s_2s_3)+L_2c_1c_2\\ s_1(c_2c_3\text{-}s_2s_3) & \text{-}s_1(c_2s_3+c_3s_2) & -c_1 & L_3s_1(c_2c_3\text{-}s_2s_3)+L_2c_2s_1\\ c_2s_3+c_3s_2 & c_2c_3\text{-}s_2s_3 & 0 & L_3(c_2s_3+c_3s_2)+L_2s_2+L_1\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.5}$$

Tọa độ điểm thao tác:

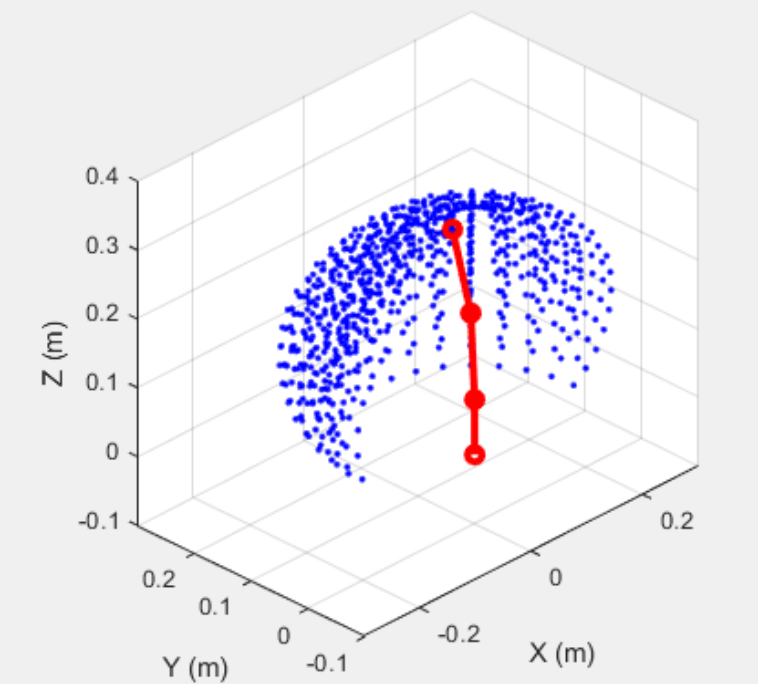
$$\begin{aligned}x_E &= \cos(q_1)(L_3\cos(q_3)\cos(q_2)\text{-}L_3\sin(q_3)\sin(q_2)+L_2\cos(q_2))\\ y_E &= \sin(q_1)(L_3\cos(q_3)\cos(q_2)\text{-}L_3\sin(q_3)\sin(q_2)+L_2\cos(q_2))\\ z_E &= L_3\sin(q_2)\cos(q_3)+L_3\cos(q_2)\sin(q_3)+L_2\sin(q_2)+L_1\end{aligned} \tag{3.6}$$

3.1.3. *Không gian làm việc*

a. Giới hạn góc khớp

$$q_1 = [0 : \pi]$$
$$q_2 = \left[\frac{\pi}{6} : \frac{\pi}{2}\right] \quad (3.7)$$
$$q_3 = \left[-\frac{\pi}{2} : 0\right]$$

b. Không gian làm việc



Hình 3.2. Không gian làm việc

Hình 3.2 mô phỏng không gian làm việc của robot trên matlab dựa trên giới hạn góc khớp được đo trên thực tế

3.2. Bài toán động học cánh tay robot

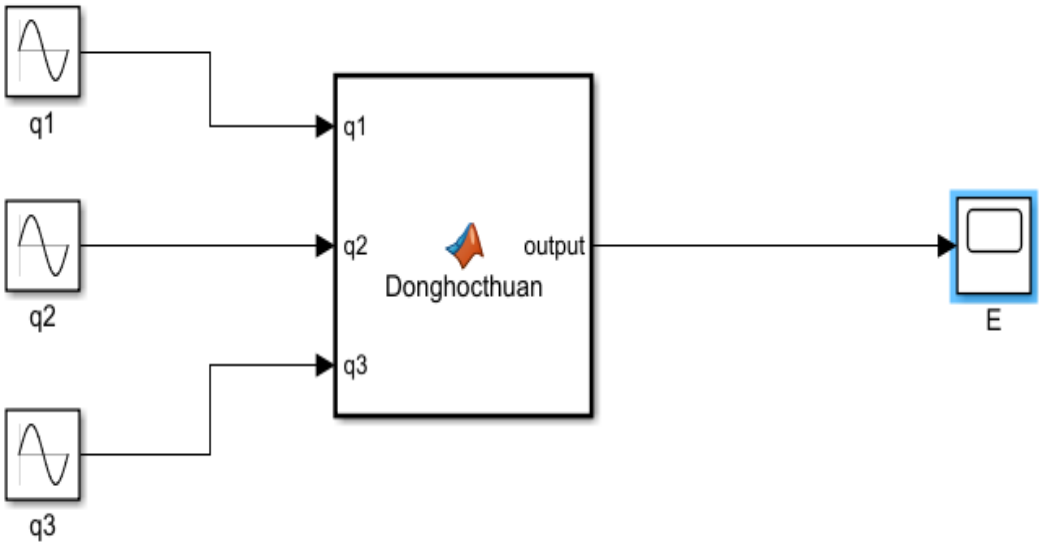
3.2.1. *Động học thuận*

a. Quỹ đạo khớp cho trước

$$q_1 = sin(t);$$
$$q_2 = -\sin\left(t - \frac{\pi}{2}\right); \quad (3.8)$$
$$q_3 = sin(t);$$

b. Mô hình simulink

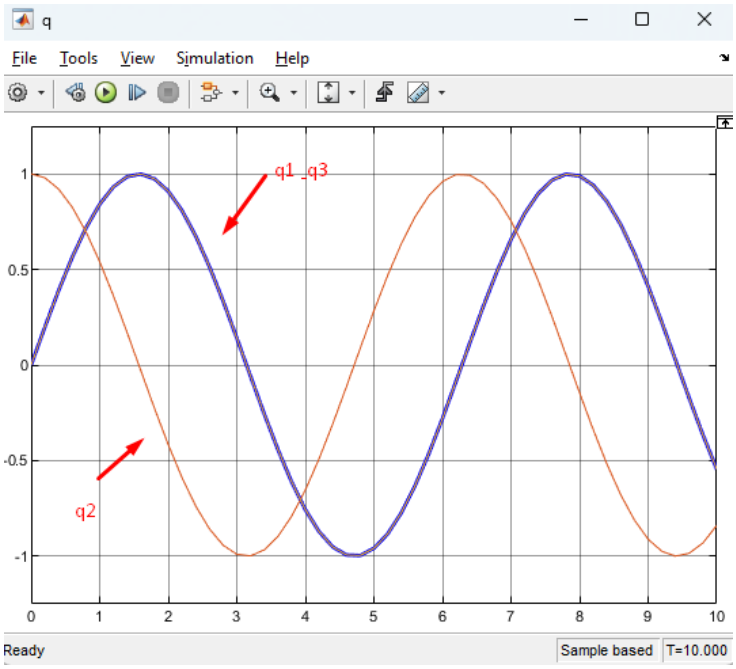
Mô hình giải bài toán động học thuận được mô tả như hình:



Hình 3.3. Simulink cho động học thuận

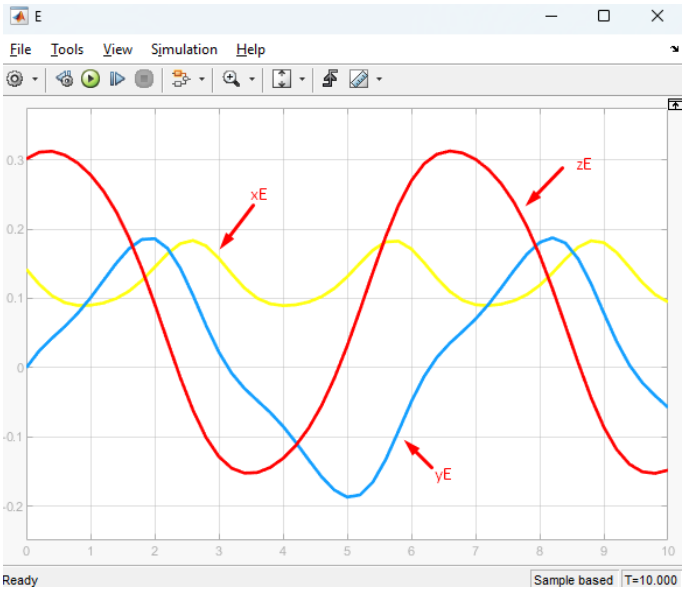
c. Kết quả mô phỏng bài toán động học thuận

Đầu vào:



Hình 3.4. Giá trị q dầu vào

Đầu ra:



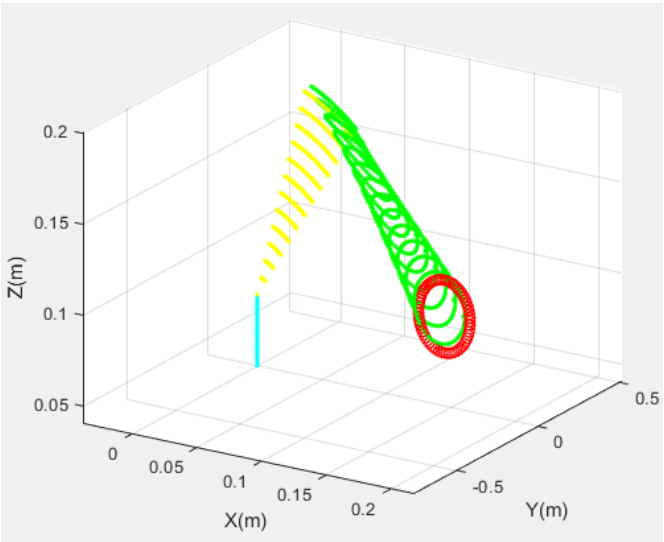
Hình 3.5. Giá trị tọa độ điểm thao tác cuối

3.2.2. Động học ngược

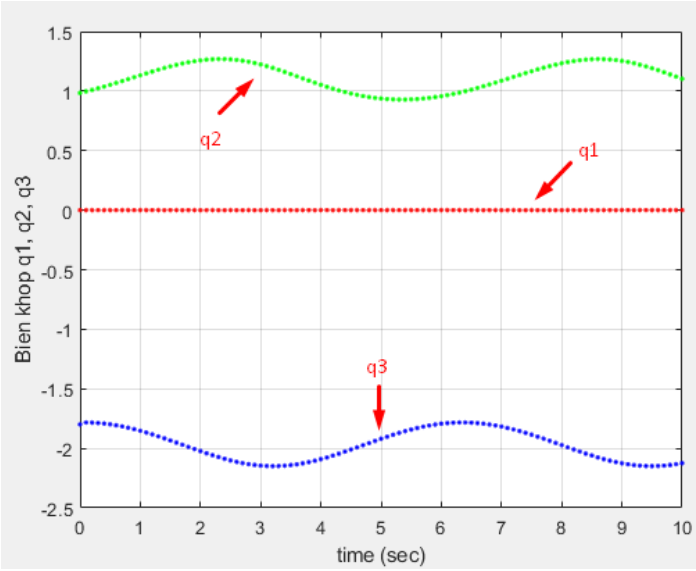
Quỹ đạo điểm tao thác cho trước:

$$\begin{aligned} x_E &= 0.145 + 0.02 * cos(t); \\ y_E &= 0; \\ z_E &= 0.09 + 0.02 * sin(t); \end{aligned} \tag{3.9}$$

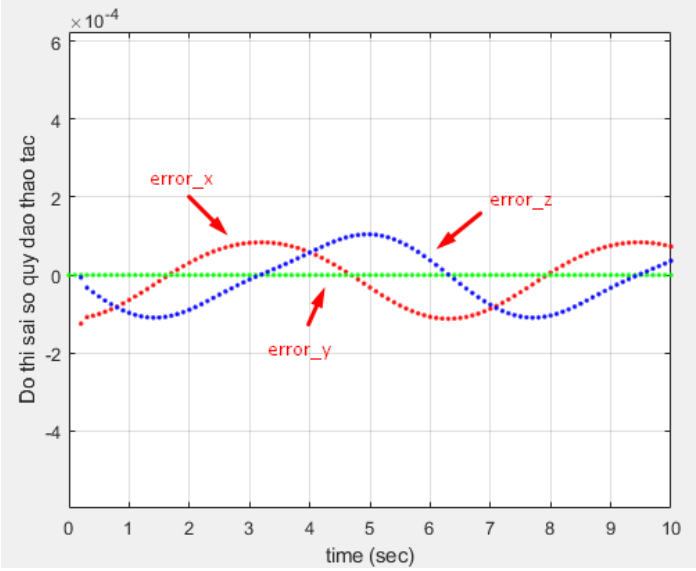
Kết quả bài toán động học ngược:



Hình 3.6. Quỹ đạo mong muốn



Hình 3.7. Giá trị q mong muốn



Hình 3.8. Sai số quỹ đạo

3.3. Bài toán điều khiển cánh tay robot 3DOF

3.3.1. Mô hình điều khiển nơ-ron

a. Khởi tạo dữ liệu mạng nơ-ron

Bộ mẫu dữ liệu để đào tạo mạng bao gồm tập hợp m giá trị của nhiều quỹ đạo của điểm thao tác cuối E.

$$((p^{(1)}, y^{(1)}), (p^{(2)}, y^{(2)}), (p^{(3)}, y^{(3)}), ..., (p^{(m)}, y^{(m)})) \quad (3.10)$$

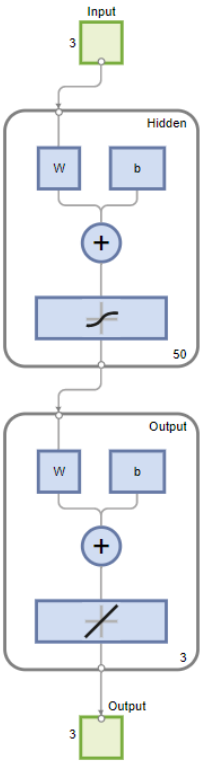
trong đó:

Đầu vào: $p = [p_x, p_y, p_z]^T$: Thể hiện tọa độ của các điểm thao tác cuối

Đầu ra: $y = [q_1, q_2, q_3]^T$: Thể hiện góc quay của mỗi khớp.

Để lấy được giá trị của giá trị đầu ra cho việc đào tạo cần phải giải quyết bài toán động học ngược (trong đồ án này đã lựa chọn phương pháp agv). Đồ án giải quyết bài toán động học ngược cho nhiều quỹ đạo mong muốn và lấy được giá trị biến khớp cho mỗi điểm thuộc mỗi quỹ đạo, sau đó tập hợp các dữ liệu đó được đưa vào mạng nơ-ron cho việc đào tạo.

b. Mô hình mạng nơ-ron



Hình 3.9. Cấu trúc mạng nơ-ron

Hình 3.8 thể hiện cấu trúc mạng nơ-ron được sử dụng, có thể thấy quá trình lan truyền thuận bao gồm có 3 đầu vào, 1 lớp ẩn gồm 50 nơ-ron, 3 đầu ra. Hàm kích hoạt sử dụng trong lớp ẩn là hàm tanh và hàm được sử dụng trong lớp ra là hàm tuyến tính.

Sau khi giải quyết bài toán động học ngược thu được giá trị các biến khớp theo các quỹ đạo mong muốn (trong đồ án này mạng được train với 5, 10, 15 quỹ đạo đầu vào để so sánh). Tiếp theo đó, các giá trị được tập hợp lại và đưa vào mạng nơ-ron với khoảng lần lượt là 500, 1000 và 1500 giá trị bao gồm đầu vào đào tạo là tọa độ của điểm thao tác cuối và đầu ra đào tạo là giá trị của các biến khớp.

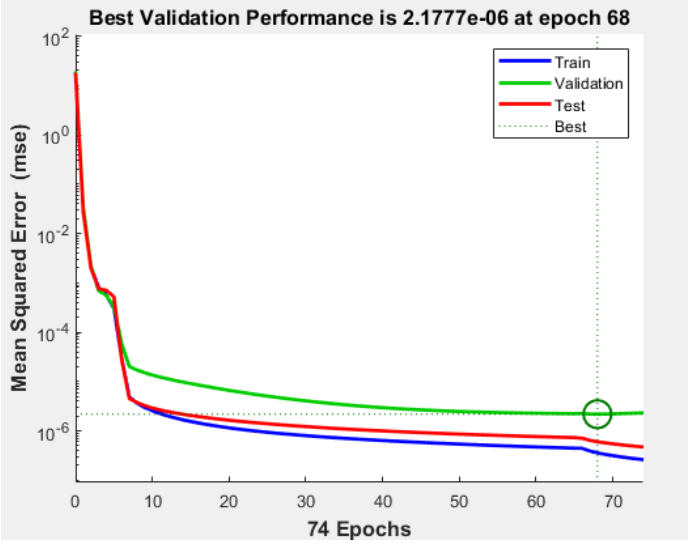
3.3.2. *Kết quả điều khiển cánh tay robot*

a. Quỹ đạo mong muốn

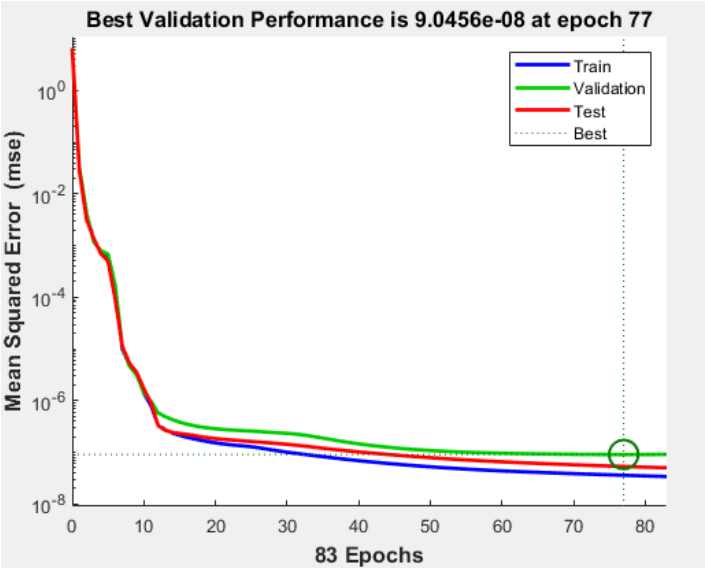
Kết quả bài toán được kiểm tra với giá trị quỹ đạo mong muốn sau:

$$\begin{aligned} x_E &= 0.145 + 0.02 * cos(t); \\ y_E &= 0; \\ z_E &= 0.09 + 0.02 * sin(t); \end{aligned} \tag{3.10}$$

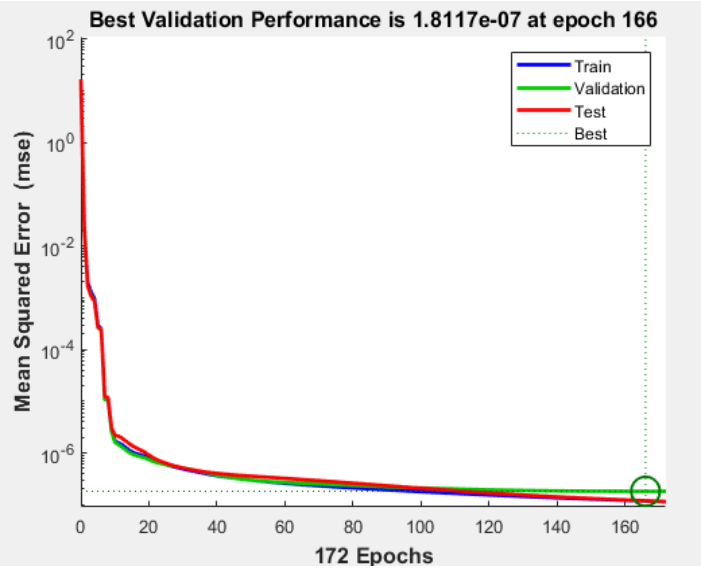
b. Kết quả quá trình đào tạo.



Hình 3.10. Kết quả đào tạo mạng (với dữ liệu đào tạo là 5 quỹ đạo)



Hình 3.11. Kết quả đào tạo mạng (với dữ liệu đào tạo là 10 quỹ đạo)

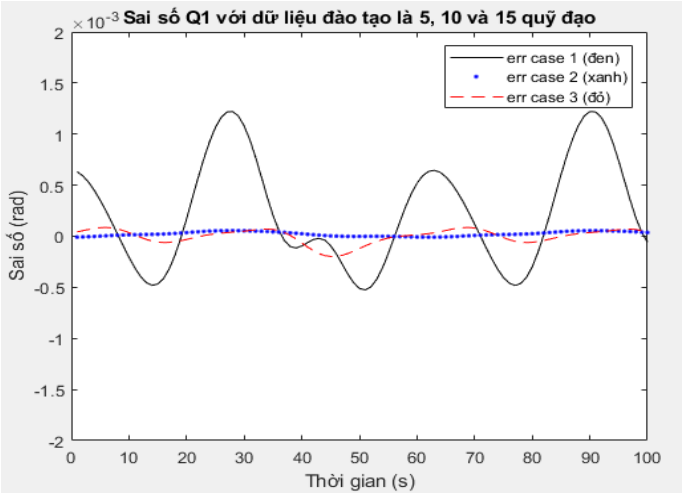


Hình 3.12. Kết quả đào tạo mạng (với dữ liệu đào tạo là 15 quỹ đạo)

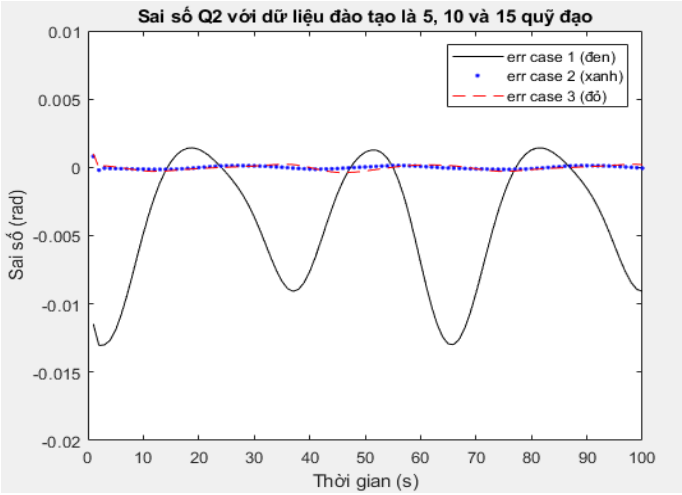
Hàm loss của 3 tập train, test và val trong cả 3 trường hợp đều giảm nhanh trong khoảng 10 epochs đầu tiên và giảm đều trong các epochs tiếp theo. Trường hợp thứ 2 có kết quả tốt hơn khi có sai số hàm loss nhỏ hơn 2 trường hợp còn lại và với số epochs để đạt được hiệu suất tối ưu trên tập val là 77 epochs

c. Kết quả mô phỏng

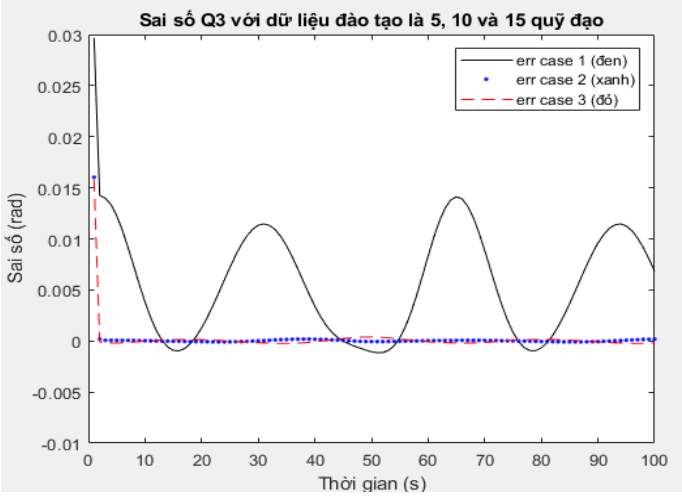
Kết quả bài toán được mô phỏng trên phần mềm matlab với 3 trường hợp đào tạo là dữ liệu đào tạo 5 quỹ đạo, 10 quỹ đạo và 15 quỹ đạo



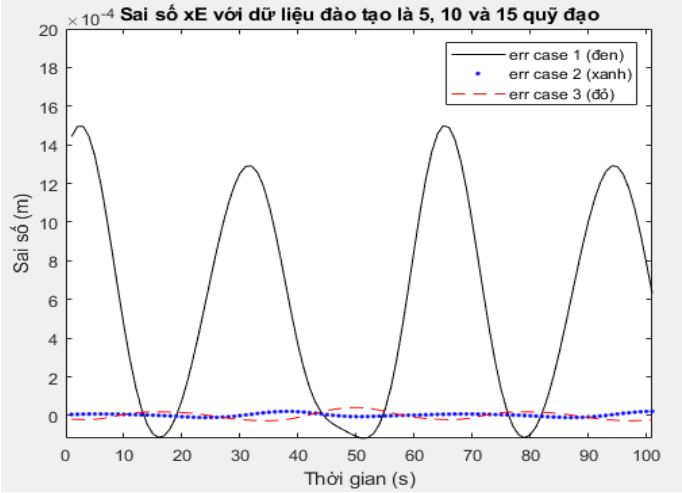
Hình 3.13. Giá trị sai số q_1 trong 3 trường hợp



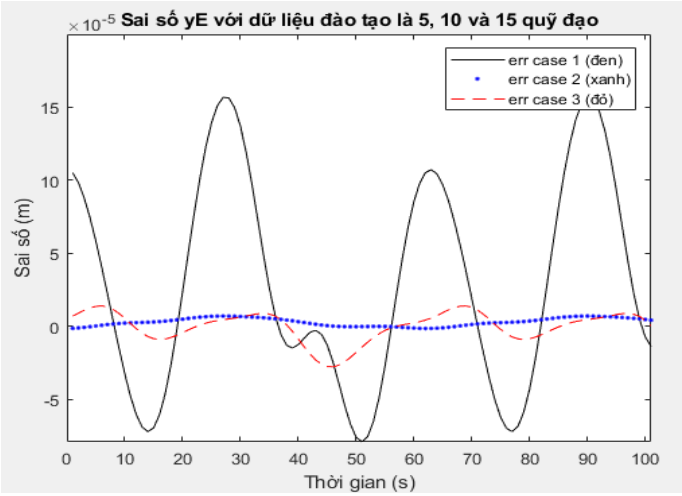
Hình 3.14. Giá trị sai số q_2 trong 3 trường hợp



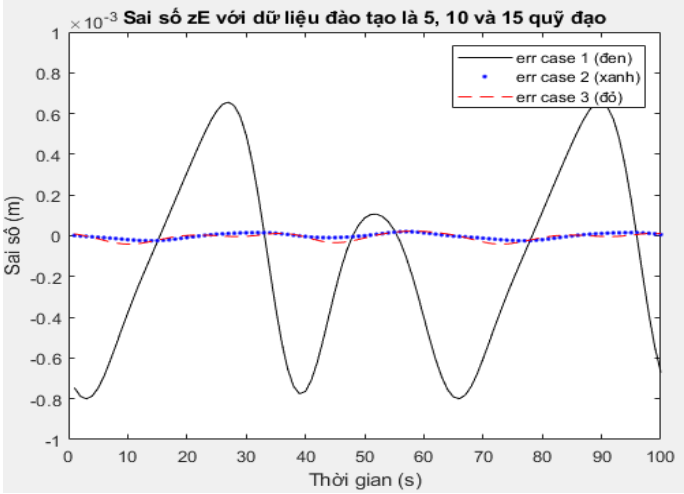
Hình 3.15. Giá trị sai số q_3 trong 3 trường hợp



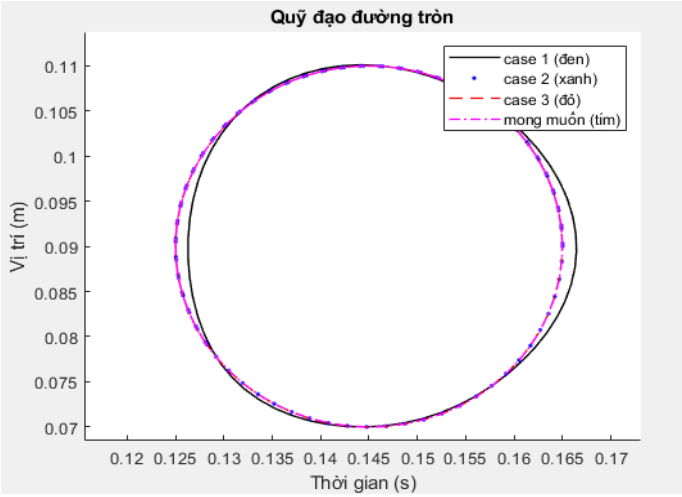
Hình 3.16. Giá trị sai số x_E trong 3 trường hợp



Hình 3.17. Giá trị sai số y_E trong 3 trường hợp



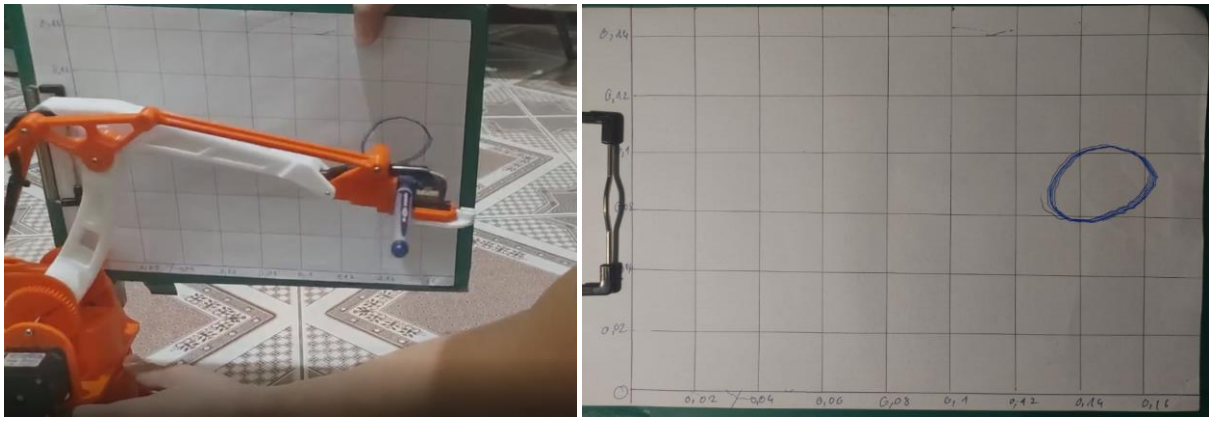
Hình 3.18. Giá trị sai số z_E trong 3 trường hợp



Hình 3.19. Quỹ đạo đường tròn theo 3 trường hợp và mong muốn

Kết quả mô phỏng bài toán trên cho thấy: với bộ quỹ đạo đào tạo là 5 quỹ đạo (case 1), quỹ đạo chưa chính xác và sai số góc khớp và tọa độ các điểm vẫn còn khá lớn. Trong khi đó, sau khi tăng số lượng quỹ đạo đào tạo lên 10 (case 2) và 15 (case 3) quỹ đạo thì thu được kết quả khá chính xác với sai số nhỏ. Với sai số quỹ đạo nhận được từ phương pháp AGV và phương pháp sử dụng mạng nơ-ron cho thấy được mạng nơ-ron cho kết quả chính xác hơn.

d. Kết quả thực nghiệm



Hình 3.20. Robot vẽ kết quả quỹ đạo thực nghiệm

Kết quả thực nghiệm được thể hiện qua video: Link video thực nghiệm

Kết luận chương 3

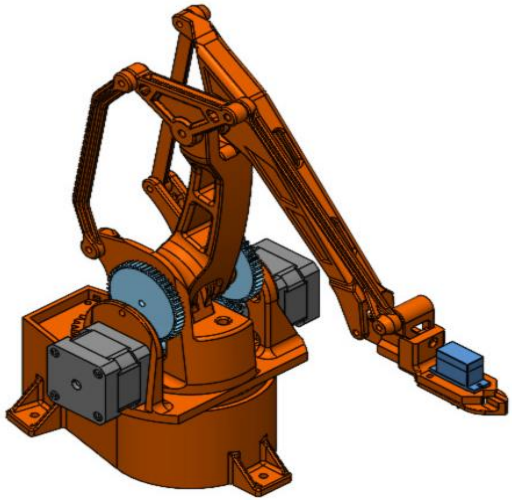
Chương này thực hiện mô hình hóa động học và điều khiển bài toán sử dụng mạng nơ-ron trên phần mềm matlab. Kết quả là đã thực hiện được điều khiển robot theo quỹ đạo cho trước ứng dụng mạng nơ-ron với sai số của quỹ đạo cũng như của góc khớp khá nhỏ. Trong quá trình đào tạo có thể thấy được rằng, với càng nhiều quỹ đạo được đưa vào để đào tạo thì độ chính xác trong quá trình kiểm nghiệm càng được nâng cao. Tuy nhiên, trong thực tế, do ảnh hưởng sai số đến từ hệ robot thực tế nên robot chạy chưa đúng được quỹ đạo đường tròn mong muốn nhưng đã khá đúng về tọa độ.

CHƯƠNG 4. ĐIỀU KHIỂN ROBOT SỬ DỤNG GRBL

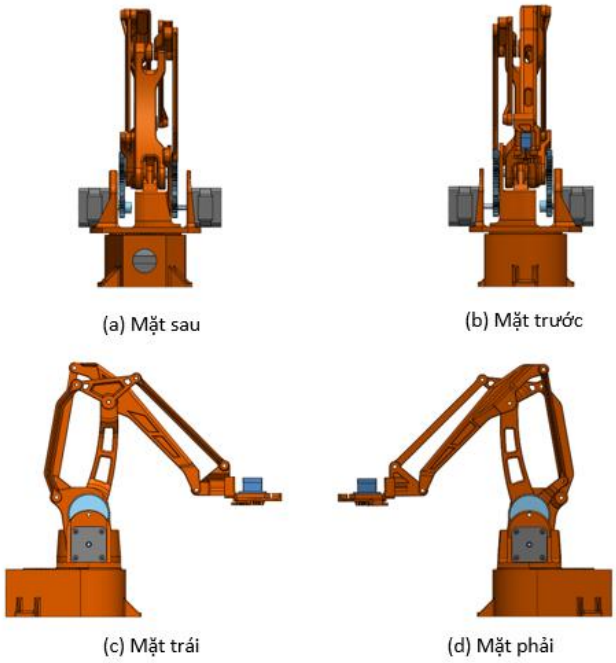
4.1. Xây dựng mô hình robot thực tế

4.1.1. Mô hình robot

Để thuận tiện trong quá trình chế tạo và bố trí các thiết bị, robot được thiết kế chi tiết trên phần mềm SolidWorks. Đây là nền tảng phần mềm hỗ trợ đắc lực cho việc thiết kế mô hình 3D, CAD, CAM, CAE. Mô hình robot sau khi thiết kế được mô tả như hình dưới đây.



Hình 4.1. Mô hình robot 4 DOF.



Hình 4.2. Các hình chiếu của robot

Sau khi hoàn thiện thiết kế, các chi tiết của robot được xây dựng thực tế sử dụng máy in 3D. Phương pháp này đảm bảo các chi tiết hoàn toàn giống với bản vẽ thiết kế.

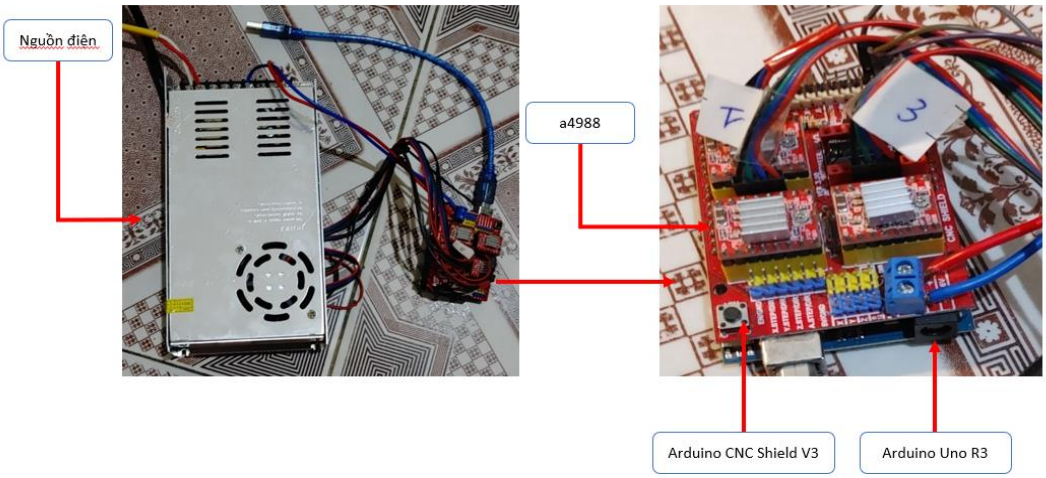
Hình 4.3. Mô hình robot thực tế

	Khâu 1	Khâu 2	Khâu 3
Chiều dài (m)	0.08	0.128	0.135

Bảng 4.1. Thông số robot

4.1.2. Sơ đồ hệ thống và thiết bị

a. Sơ đồ hệ thống



Hình 4.4. Hệ thống robot thực

b. Thiết bị được sử dụng.

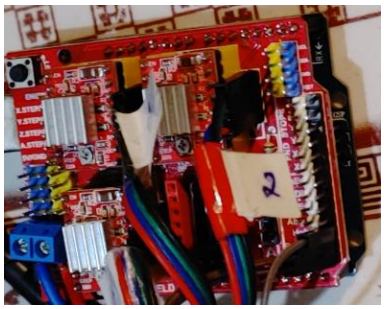
Vi điều khiển

Mô hình thí nghiệm sử dụng Arduino Uno R3 làm vi điều khiển có vai trò đọc tín hiệu encoder để cung cấp.

Hình 4.5. Arduino Uno R3

Arduino CNC Shield V3

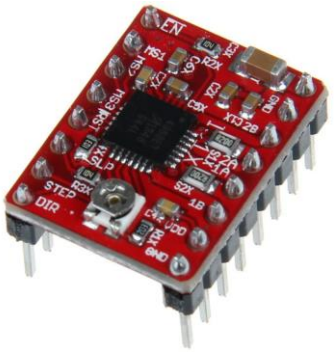
Mô hình sử dụng Arduino CNC Shield V3 để có thể kết nối được giữa Arduino với các driver mà không cần phải hàn lại.



Hình 4.6. Arduino CNC Shield V3

Driver

Mạch thực tế sử dụng driver A4988 để điều khiển cho động cơ bước.



Hình 4.7. Driver A4988

Động cơ bước

Mô hình này đã sử dụng 3 động cơ bước cho việc điều khiển mỗi khớp.

Hình 4.8. Động cơ bước nema 17

Nguồn điện.

Do đặc tính tay máy làm việc cố định nên thí nghiệm lựa chọn nguồn tổ ong làm bộ chuyển đổi trực tiếp từ nguồn AC sang nguồn DC để đảm bảo điện áp nuôi động cơ luôn ổn định.

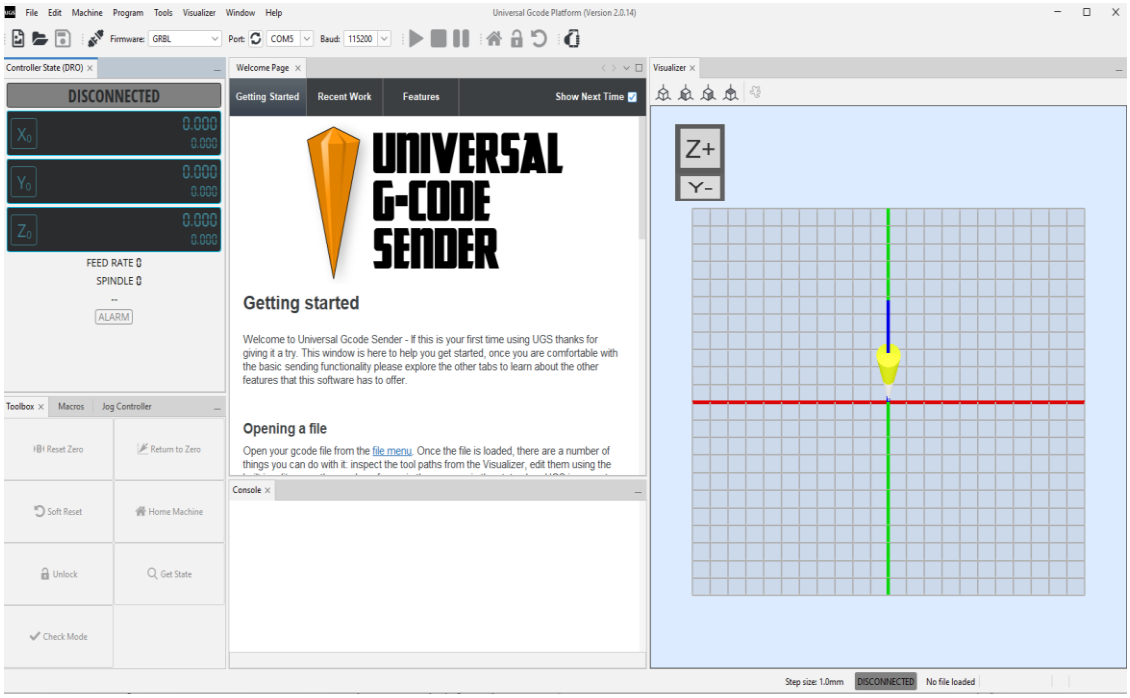
Hình 4.9. Nguồn tổ ong 12V-20A.

4.2. Lập trình giao diện GRBL

4.1.1. Giới thiệu về GRBL

GRBL là phần mềm mã nguồn mở, miễn phí dùng để điều khiển hoạt động của máy CNC. Do lợi thế hiệu suất cao, chi phí thấp mà nó có thể trở thành chuẩn công nghiệp mới. Nó chạy trên Arduino (Duemillanove / Uno) miễn là có chip Atmega 328. Bộ điều khiển được viết bằng ngôn ngữ C tối ưu hóa cao tận dụng mọi tính năng thông

minh của chip AVR để đạt được thời gian chính xác và hoạt động không đồng bộ. Nó có thể duy trì đến độ ổn định 30kHz, kiểm soát xung free jitter. Nó tuân thủ tiêu chuẩn G-code và đã được thử nghiệm với sản phẩm của một số công cụ CAM và không có vấn đề. Chuyển động vòng cung, hình tròn và chuyển động xoắn ốc được hỗ trợ đầy đủ tất cả các lệnh chính G-code khác. Các hàm,các biến, và hầu hết các chu kỳ đóng gói không được hỗ trợ, nhưng nó có thể làm việc tốt hơn nhiều nhờ chuyển chúng vào thẳng mã G bằng bất kỳ cách nào.



Hình 4.10. Phần mềm Gcode

4.1.2. Các mã lệnh điều khiển

Các lệnh Gcode CNC tiếng Việt chủ yếu được sử dụng và cài đặt trong các máy CNC tại Việt Nam, nguyên bản gốc của nó là sử dụng bằng tiếng Anh. Một số các lệnh G lập trình CNC căn bản cần biết như sau:

Nhóm lệnh G:

Tên lệnh	**Mô tả lệnh**
G00	Xác định vị trí
G01	Nội suy tuyến tính

G02	Nội suy cung tròn/xoắn vít/ xoán Ascimet cùng chiều kim đồng hồ
G03	Nội suy cung tròn/xoắn vít/ xoán Ascimet ngược chiều kim đồng hồ
G04	Dừng tịnh tiến dụng cụ/Dừng chính xác
G09	Dừng chính xác
G10	Thay đổi hệ toạ độ phôi
G11	Huỷ chế độ G10
G17	Chọn mặt mặt phẳng gia công XY
G18	Chọn mặt mặt phẳng gia công XZ
G19	Chọn mặt mặt phăng gia công ZY
G20	Đặt đơn vị làm việc theo hệ inch
G21	Đặt đơn vị làm việc theo hệ mm
G27	Quay về gốc máy
G28	Trở quay về gốc máy tự động
G29	Quay về gốc máy thứ 2, thứ 3 hoặc thứ 4
G30	Điểm O thứ hai /thứ ba/ thứ tư
G31	Bỏ qua lệnh
G33	Cắt ren
G40	Huỷ bỏ hiệu chỉnh bù bán kính
G41	Hiệu chỉnh bán kính dụng cụ cắt, dao ở bên trái công tua gia công
G42	Hiệu chỉnh bán kính dụng cụ cắt, dao ở bên phải công tua gia công
G43	Bù chiều dài dụng cụ , +

G44	Bù chiều dài dụng cụ , -
G45	Bù vị trí dụng cụ, tăng
G46	Bù vị trí dụng cụ, giảm
G47	Bù vị trí dụng cụ, tăng 2 lần
G48	Bù vị trí dụng cụ, giảm 2 lần
G49	Huỷ bù chiều dài dụng cụ
G52	Đặt hệ toạ độ địa phương
G53	Lựa chọn hệ toạ độ máy
G54	Lựa chọn hệ toạ độ máy
G55	Lựa chọn hệ toạ độ phôi thứ hai
G56	Lựa chọn hệ toạ độ phôi thứ ba
G57	Lựa chọn hệ toạ độ phôi thứ tư
G58	Lựa chọn hệ toạ độ phôi thứ năm
G59	Lựa chọn hệ toạ độ phôi thứ sáu
G60	Tiếp cận theo một hướng
G61	M lệnh dừng chính xác
G63	Chế độ Taro
G64	Chế độ cắt gọt (chế độ kiểm tra dừng chính xác)
G65	Gọi marco
G66	Gọi nhóm marco
G67	Huỷ gọi nhóm marco

G73	Gia công lỗ sâu tốc độ cao
G74	Chu trình taro
G76	Chu trình khoét lỗ
G80	Huỷ chu trình gia công lỗ
G81	Chu trình khoan lỗ nông
G82	Chu trình khoét lỗ bậc
G83	Chu trình gia công lỗ sâu
G84	Chu trình taro
G84.2	Chu trình taro cứng
G82.3	Chu trình taro cứng, ren trái
G85	Chu trình khoét lỗ
G86	Chu trình khoét lỗ
G87	Chu trình khoét lỗ, mặt sau.
G88	Chu trình khoét lỗ
G89	Chu trình khoét lỗ
G90	Đặt hệ toạ độ tuyệt đối
G91	Đặt hệ toạ độ gia số
G92	Đổi hệ toạ độ phôi/ Đặt tốc độ quay lớn nhất
G94	Đặt tốc độ tiến dao /phút
G95	Đặt tốc độ tiến dao /vòng
G96	Tốc độ bề mặt không đổi

G97	Huỷ tốc độ bề mặt không đổi
G98	Đặt kiểu rút dao, trong chu trình gia công lỗ
G99	Đặt kiểu rút dao, trong chu trình gia công lỗ

Bảng 4.2. Tập lệnh G cơ bản.

Nhóm lệnh M:

Tên lệnh	**Mô tả lệnh**
M00	Dừng chương trình
M01	Dừng lựa chọn
M02	Kết thúc chương trình
M03	Quay trục chính bên phải
M04	Quay trục chính bên phải
M05	Dừng trục chính
M06	Thay dụng cụ
M07	Kích hoạt quá trình bơm dầu trơn nguộ
M08	Phun dầu tưới nguội
M09	Tắt dung dịch trơn nguội, Tắt bơm dầu
M29	Dạng taro cứng
M30	Kết thúc chương trình
M80	Vòi phun rửa phoi ON
M81	Vòi phun rửa phoi OFF
M82	Cửa tự động ON

M83	Cửa tự động OFF
M84	Bật màn hình
M85	Tắt màn hình
M86	Điều khiển thích nghi ON
M86	Làm nguội trục chính ON
M89	Làm nguội trục chính OFF
M96	Chế độ ngắt marco
M97	Huỷ dạng ngắt marco
M98	Gọi chương trình con

Bảng 4.3. Tập lệnh M cơ bản.

4.3. Kết nối giao diện GRBL và cánh tay robot

Các bước kết nối

Để kết nối GRBL với mô hình robot 3DOF, cần thực hiện các bước sau:

Bước1: Chuẩn bị các thành phần cần thiết (các thiết bị đã được thể hiện trong phần 4.1.2)

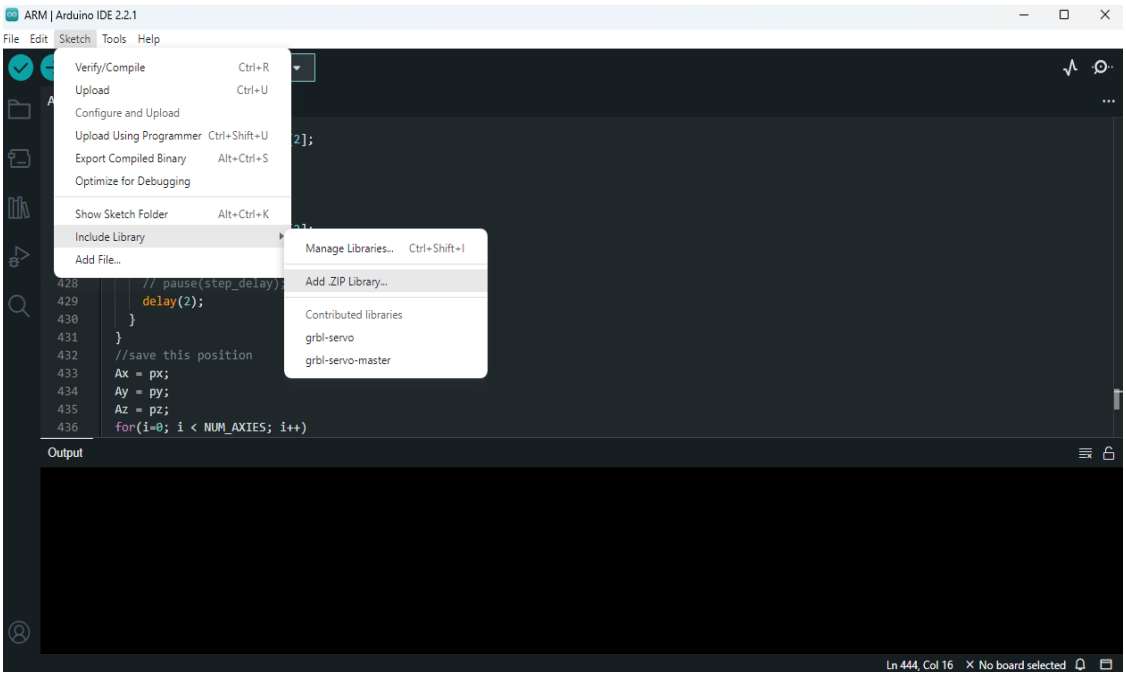
Bước 2: Kết nối các thành phần như sau:

- Kết nối Arduino Uno với máy tính thông qua cổng USB.
- Kết nối động cơ bước với biến áp Stepper.
- Kết nối biến áp Stepper với Arduino Uno.
- Kết nối các dây jumper giữa Arduino Uno và mô hình robot 3DOF.

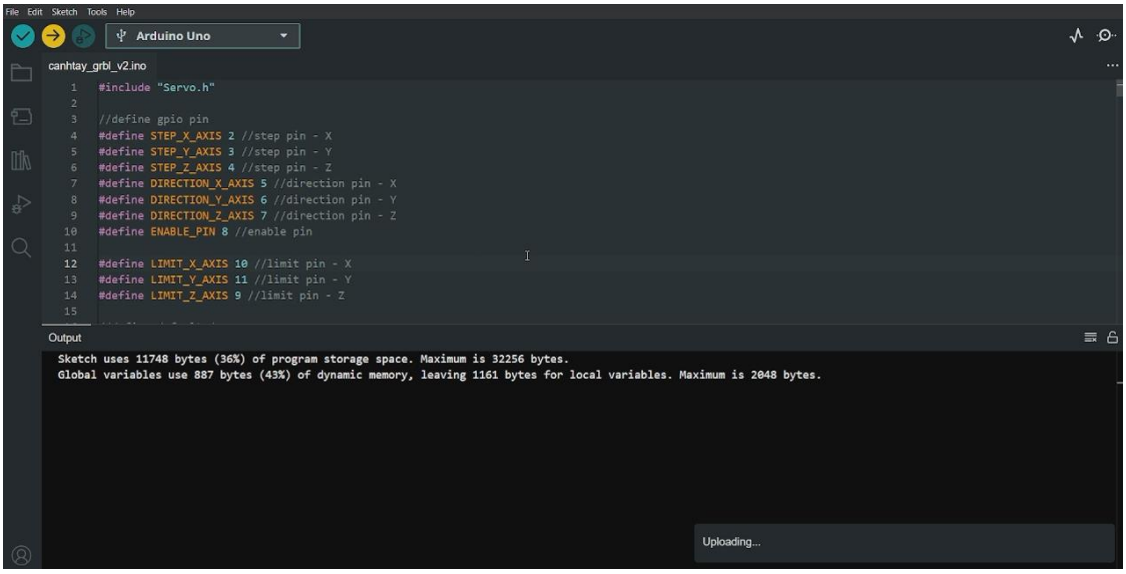
Bước 3: Cấu hình trên Arduino

Download source code của grbl trên github và mở trên Arduino:

https://github.com/trieutuanvnu/grbl-servo

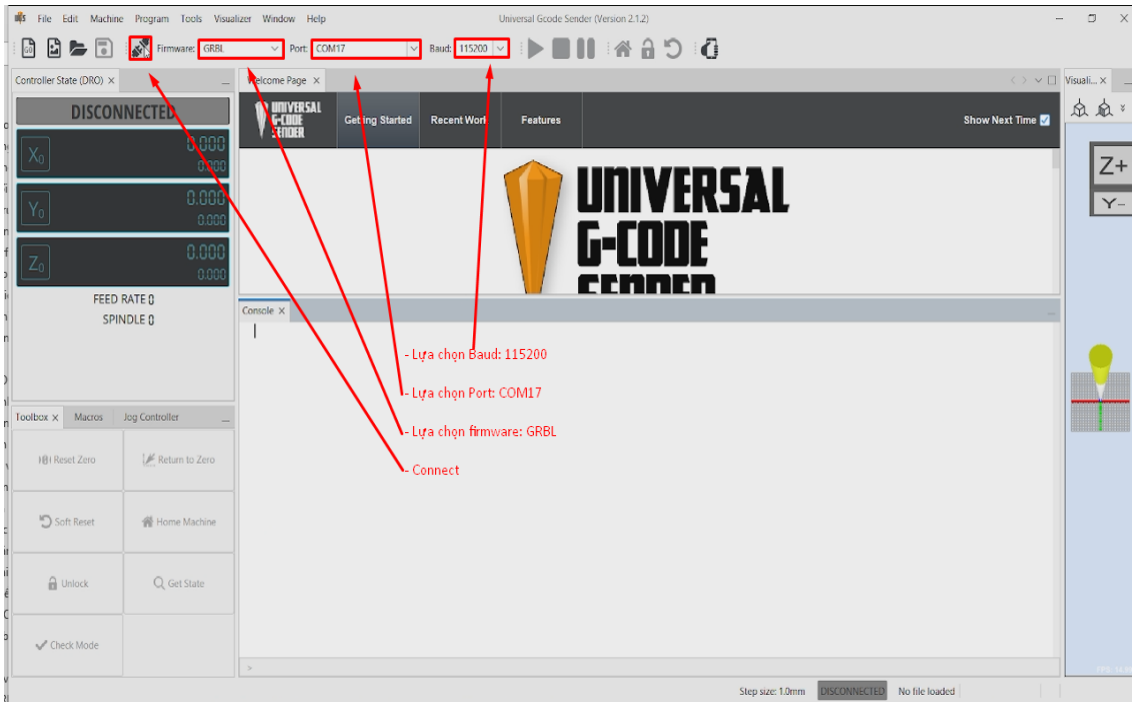


Hình 4.11. Thêm file zip của GRBL vừa download

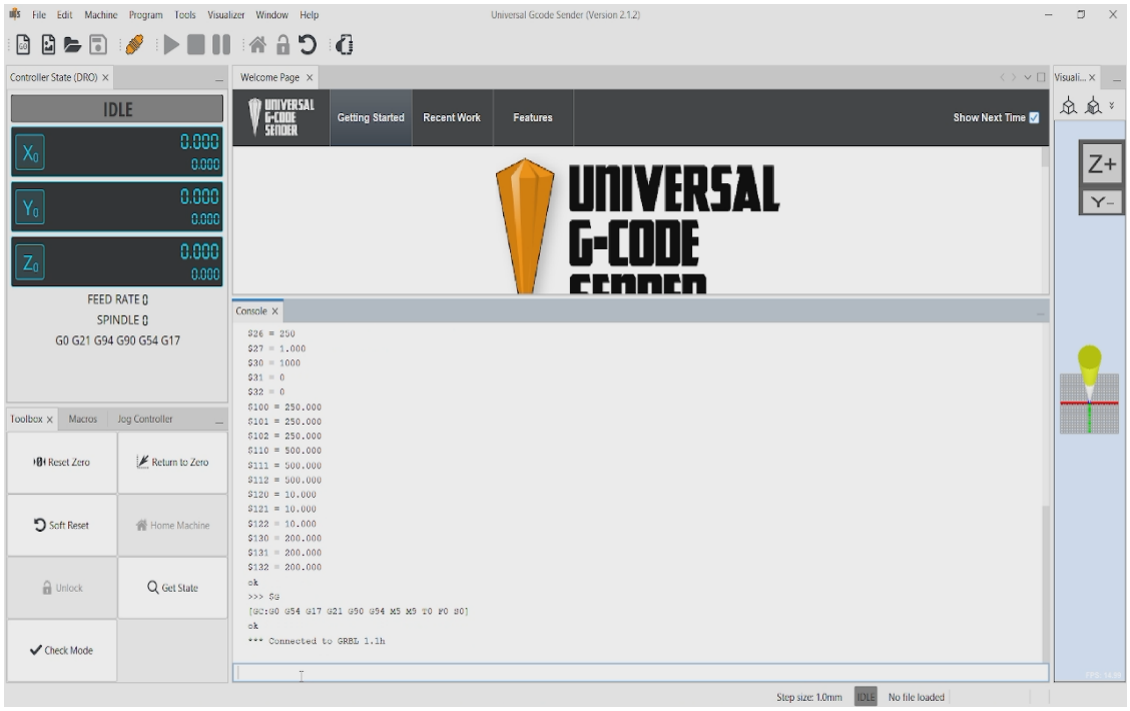


Hình 4.12. Lựa chọn board Arduino sau đó chạy code

Bước 4: Cấu hình trên phần mềm Gcode

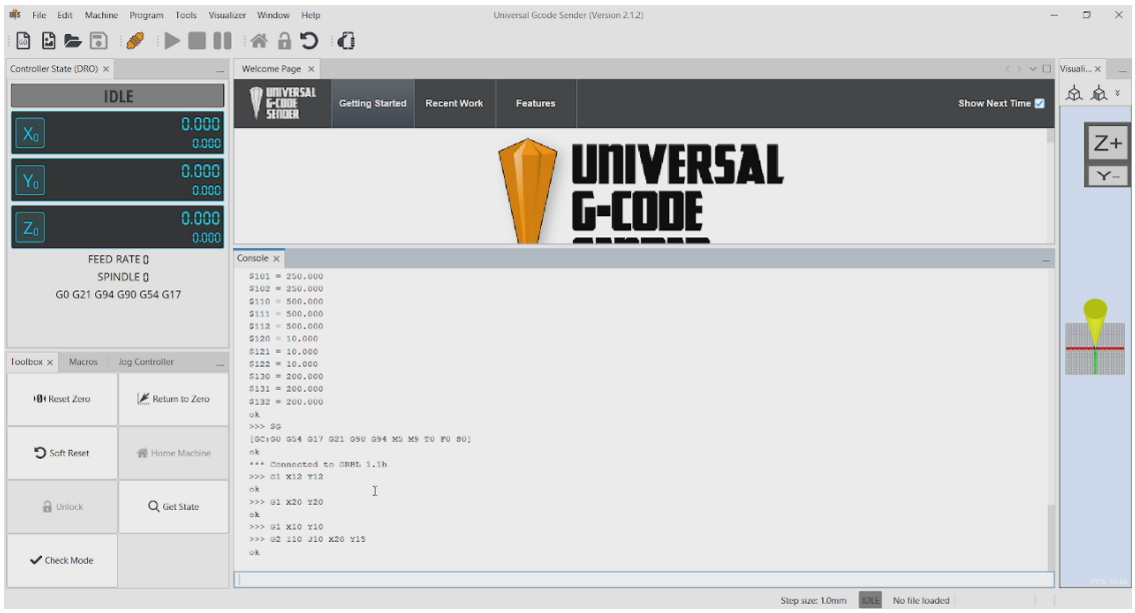


Hình 4.13. Cấu hình trước khi chạy trên GRBL



Hình 4.14. Quá trình kết nối thành công

Bước 6: Thử nghiệm kết nối.



Hình 4.15. Chạy thử với một số tập lệnh cơ bản

4.4. Kết quả thực nghiệm điều khiển

Kết quả bài toán được thực nghiệm qua việc vẽ một cung tròn trên mặt phẳng.

Kết quả chạy bài toán được thể hiện trong link drive: Link video thực nghiệm

Dưới đây là hình vẽ thể hiện tọa độ điểm đầu và điểm cuối của cung tròn.

Hình 4.16. Tọa độ điểm đầu của cung tròn

Hình 4.17. Tọa độ điểm cuối của cung tròn

Kết luận chương 4

Chương này thực hiện điều khiển mô hình robot với phần mềm Gcode, kết quả là đã thực hiện được việc điều khiển robot theo quỹ đạo mong muốn

KẾT LUẬN

Vấn đề động học ngược và điều khiển cho tay máy là một lĩnh vực được rất nhiều người quan tâm bởi vì cho đến nay vẫn chưa có một phương pháp chung nào hiệu quả nhất để giải quyết vấn đề này, đặc biệt là đối với tay máy có số bậc tự do lớn hơn 6. Do đó, ứng dụng mạng nơron nhằm đề xuất một phương pháp mới có thể ứng dụng để giải bài toán động học ngược và điều khiển cho tay máy. Trong đồ án này, em tiến hành triển khai bài toán trên cho tay máy 3-DOF bằng mạng nơ-ron.

Qua quá trình nghiên cứu, tìm hiểu và thực hiện đồ án em đã đạt được mục tiêu đề ra cho bài toán như hiểu được cách thiết kế, xây dựng và triển khai cho một robot trên thực tế thay vì mô phỏng, tiếp theo đó hiểu được về cách hoạt động của các bài toán quan trọng trong robot, cuối cùng là hiểu được về phần mềm GRBL – một phần mềm quan trọng và phổ biến trong điều khiển robot, và cách sử dụng phần mềm để điều khiển robot một cách cơ bản.

Kết quả cuối cùng đã được thực hiện một cách khá chính xác với sai số khá nhỏ khi đưa mạng nơ-ron vào học dữ liệu từ bài toán động học ngược. Trong đồ án này, dữ liệu đầu vào được lấy từ bài toán động học ngược với phương pháp agv cho 3 trường hợp là 5, 10 và 15 quỹ đạo và thấy được rằng khi số lượng quỹ đạo đào tạo tang lên thì quỹ đạo nhận được sẽ càng chính xác hơn.

Vấn đề xác định số lớp ẩn hay số nơron được sử dụng trong một lớp. Trong đồ án này em xác định các thông số này bằng việc thử nhiều giá trị khác nhau và chọn giá trị tốt nhất có thể nên khá tốn thời gian tìm nghiệm. Vấn đề khởi tạo trọng số, độ lệch ban đầu, tốc độ học. Hai thông số đó được khởi tạo một cách ngẫu nhiên trong đồ án này, ta có thể nghiên cứu thêm phương pháp để xác định thông số ban đầu phù hợp và có thể dùng thêm thuật toán tối ưu adam để thay đổi tốc độ học phù hợp sau mỗi vòng lặp.

Đối với hướng phát triển của bài toán trong tương lai, bài toán có thể phát triển thành việc kết hợp giữa điều khiển mờ hoặc điều khiển PID với mạng nơ-ron cho việc điều khiển robot trong thực tế hoặc kết hợp với việc sử dụng kĩ thuật nâng cao hơn của học máy là học tang cường cho bài toán điều khiển này.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1]. Nguyễn Đình Tuấn, "Deep learning cơ bản", 2019, tr 83-85.

[2]. Nguyễn Việt Hùng, "Ứng dụng mạng nơron để giải bài toán động học ngược tay máy", 2014, tr. 107.

[3]. Phạm Hữu Đức Dục, "Mạng nơron & ứng dụng trong điều khiển tự động", *NXB Khoa Học và Kỹ Thuật*, 2009, tr. 292.

Tiếng Anh

[4]. Adrian-Vasile Duka, "Neural network based inverse kinematics solution for trajectory tracking of a robotic arm", *The 7th International Conference Interdisciplinarity in Engineering (INTER-ENG 2013)*, pp. 20-27, 2014.

[5]. Ahmed R. J. Almusawi, L. Canan Dülger, and Sadettin Kapucu, "A New Artificial Neural Network Approach in Solving Inverse Kinematics of Robotic Arm (Denso VP6242)", pp. 3, 2016.

[6]. Bassam Daya, Shadi Khawandi and Mohamed Akoum, "Applying Neural Network Architecture for Inverse Kinematics Problem in Robotics", pp. 231-239, 2010.

[7]. Fuad Alhaj Omar, "Performance comparison of pid controller and fuzzy logic controller for water level control with applying time delay", *Department of Electric and Energy*, pp. 858-871, 2021.

[8]. IFR. 2019. "World Robotics 2019." Tech. rep. International Federation of Robotics.

[9]. Maad M. Mijwil, Adam Esen, and Aysar Alsaadi, "Overview of Neural Networks", pp. 1, 2019.

[10]. Saad Zaghlul Saeed, "Tuning PID Controller by Neural Network for Robot Manipulator Trajectory Tracking", *Al-Khwarizmi Engineering Journal*, Vol. 8, No. 1, pp. 19-28, 2013.

[11]. Yin Feng, Wang Yao-nan, Yang Yi-min, "Inverse Kinematics Solution for Robot Manipulator based on Neural Network under Joint Subspace", ISSN 1841-9836, pp. 459-472, 2014.