ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Tự Sang

MÔ HÌNH HÓA ĐỘNG HỌC VÀ ĐIỀU KIỂN CÁNH TAY ROBOT ỨNG DỤNG MẠNG NƠ-RON

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 KIỂN CÁNH TAY ROBOT ỨNG DỤNG MẠNG NƠ-RON

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: Ngày nay, với sự tiến bộ vượt bậc trong lĩnh vực khoa học và công nghệ, các quốc gia trên toàn cầu đang đẩy mạnh việc tự động hóa quá trình sản xuất để nâng cao hiệu suất, chất lượng sản phẩm, cải thiện chi phí và giảm sức lao động của con người. Công nghệ robot ngày càng được cải thiện với tính linh hoạt, thông minh, chính xác và phản hồi nhanh hơn. Do đó, việc nghiên cứu để tận dụng tối đa các robot hiện có cũng như phát triển các robot mới để đáp ứng yêu cầu ngày càng cao của ngành công nghiệp hiện đại là một điều rất quan trọng. Trong quá trình này, việc giải quyết vấn đề điều khiển các cánh tay robot là điều cần thiết để điều chỉnh chúng theo quỹ đạo đã được xác định trước. Hơn nữa, tìm kiếm một phương pháp chung hiệu quả để giải quyết vấn đề điều khiển cánh tay robot với n bậc tự do đang là thách thức lớn đối với các nhà nghiên cứu trên khắp thế giới.

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

Đầu tiên, em xin gửi lời cảm ơn chân thành tới TS. Dương Xuân Biên đã trực tiếp hướng dẫn em rất tận tình trong quá trình thực hiện đồ án này. 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 toàn thể 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 tích lũy được những kiến thức quý giá cũng như chuẩn bị điều kiện tốt nhất 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 "Thiết kế, chế tạo và điều khiển mô hình cánh tay robot sử dụng mạng nơ-ron" hoàn toàn được thực hiện bởi bản thân dưới sự hướng dẫn của TS. Dương Xuân Biên.

Mọi tham khảo, trích dẫn từ các nghiên cứu và tài liệu liên quan được sử dụng trong Đồ án đều được nêu nguồn gốc một cách rõ ràng tại danh mục tài liệu tham khảo. Trong đồ án hoàn toàn không có sự sao chép tài liệu cũng như 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 được thiết kế để định vị và thực hiện các tác vụ trong môi trường sản xuất. Chúng cần có khả năng điều khiển chuyển động cũng như các lực và momen tương tác với môi trường xung quanh.

Hình 1.1. Hệ thống robot công nghiệp

Hầu hết các loại robot hiện nay được sử dụng trong ngành công nghiệp. Chúng có đặc điểm riêng về cấu trúc và chức năng, được chuẩn hóa và thương mại hóa rộng rãi. Nhóm robot này được gọi là Robot công nghiệp.

Công nghệ tự động hóa tronga lĩnh vực công nghiệp đã phát triển đến mức rất cao, không chỉ tự động hóa các quy trình vật lý mà còn cả các quy trình xử lý thông tin. Điều này bao gồm tích hợp công nghệ sản xuất, kỹ thuật điện, điện tử và kỹ thuật điều khiển tự động, trong đó có sự tự động hoá thông qua máy tính.Hiện nay, trong công nghiệp tồn tại 3 dạng tự động hoá:

Tự động hoá cứng: Xây dựng dưới dạng các thiết bị hoặc dây chuyền chuyên biệt theo từng mục tiêu cụ thể. Đây thường được áp dụng trong sản xuất hàng loạt với lượng sản phẩm cụ thể lớn.

Tự động hoá linh hoạt: Thường áp dụng trong sản xuất loạt nhỏ đến loạt vừa, đáp ứng phần lớn nhu cầu sản xuất công nghiệp. Hệ thống này sử dụng thiết bị điều khiển

đa năng, cho phép dễ dàng lập trình lại để thay đổi quy trình sản xuất các loại sản phẩm khác nhau.

Tự động hoá linh hoạt là một dạng phát triển từ tự động hoá khả trình. Nó kết hợp công nghệ sản xuất với kỹ thuật điều khiển bằng máy tính, cho phép thay đổi đối tượng sản xuất mà không cần sự can thiệp nhiều của con người. Tự động hoá linh hoạt thường thấy dưới hai dạng chính: tế bào sản xuất linh hoạt và hệ thống sản xuất linh hoạt.

Để miêu tả các đặc điểm cơ bản của robot công nghiệp, hiện nay đang áp dụng định nghĩa sau đây:

Robot công nghiệp được xem như một tay máy vạn năng, hoạt động dựa trên chương trình và có khả năng lập trình lại để thực hiện và tối ưu hoá hiệu suất cho các nhiệm vụ khác nhau trong ngành công nghiệp, từ việc vận chuyển nguyên vật liệu, chi tiết, dụng cụ đến các thiết bị chuyên dụng khác.

Robot công nghiệp có thể được cố định hoặc di động, bao gồm cả thiết bị thừa hành dạng tay máy với nhiều bậc tự do hoạt động và thiết bị điều khiển theo chương trình, có khả năng lập trình lại để thực hiện các chức năng vận động và điều khiển trong quá trình sản xuất.

Chức năng vận động của robot bao gồm các hoạt động như vận chuyển, định hướng, xếp đặt, kẹp, lắp ráp,... đối với các đối tượng khác nhau. Chức năng điều khiển chỉ ra vai trò của robot như một công cụ quản lý sản xuất, từ việc cung cấp dụng cụ và nguyên vật liệu, phân loại và phân phối sản phẩm, duy trì tốc độ sản xuất và thậm chí điều khiển các thiết bị liên quan.

Với đặc điểm có thể lập trình lại, Robot công nghiệp thiết bị tự động hóa khả trình và ngày càng trở thành bộ phận không thể thiếu được của các tế bào hoặc hệ thống sản xuất linh hoạt.

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 không thể thiếu cuộc sống mỗi người. Một số vai trò và lợi ích phổ biến của 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. Bởi vì đây là yếu tố ảnh hưởng đến thu nhập của toàn công ty. Khi sử dụng các cánh tay robot công nghiệp, các nhà máy đó sẽ có một cơ hội lớn để hoạt động liên tục mà không cần nghỉ ngơi. Hơn nữa, việc sử 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 nhân công được giảm: Khi ứng dụng robot công nghiệp trong nhà máy, họ có thể giảm chi phí nhân công 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: Con người trong công việc cũng như đời sống gần như không thể tránh khỏi sai sót. Tuy nhiên, đối với robot sai sót được giảm thiểu đến mức tối đa vì chúng đã được lập trình thiết lập rất kỹ trước khi đưa vào sản xuất, bên cạnh nó chúng cũng được thực nghiệm rất nhiều lần trước khi được đưa vào thực tế.

Nâng cao an toàn lao động: Robot công nghiệp cũng là 1 giải pháp nâng caoan toàn lao động. Những công việc đòi hỏi sức khỏe, hay nguy hiểm robot đều có thể hoàn thành thay con người. Đối với những kỹ sư điều khiển robot,họ cũng được bảo vệ bởi các tấm nhựa xung quanh. Và được bảo vệ bởi chính những robot. Các cảm biến tiệm cận, quang, hay hồng ngoại luôn được bật khi có bất thường sẽ ngắt toàn bộ thiết bị.

1.1.3. Kết cấu của tay máy công nghiệp

Tay máy là phần cơ sở, quyết định khả năng làm việc của Robot công nghiệp. Đó là thiết bị cơ khí đảm bảo cho robot khả năng chuyển động trong không gian và khả năng làm việc, như nâng hạ vật, lắp ráp,… Ý tưởng ban đầu của việc thiết kế và chế tạo tay máy là phỏng tác cấu tạo và chức năng của tay người. Về sau, đây không còn là điều bắt buộc nữa. Tay máy hiện nay rất đa dạng và nhiều loại có dáng vẻ khác rất xa với tay người. Tuy nhiên, trong kỹ thuật robot người ta vẫn dùng các thuật ngữ quen thuộc, như vai (Shoulder), cánh tay (Arm), cổ tay (Wrist), bàn tay (Hund) và các khớp (Articulations)…

Trong thiết kế và sử dụng tay máy, người ta quan tâm đến các thông số có ảnh hướng lớn đến khả năng làm việc của chúng, như:

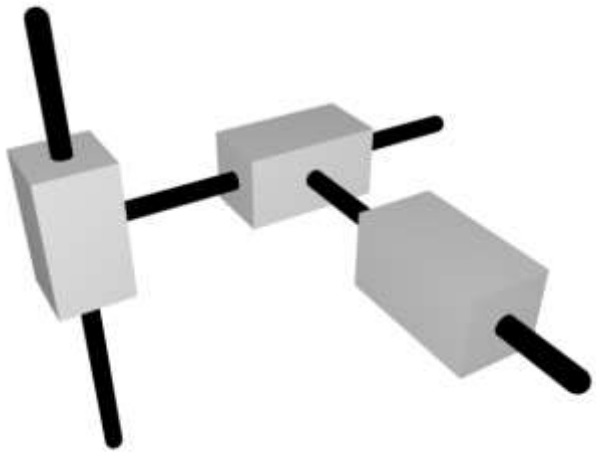
Sức nâng, độ cứng vững, lực kẹp của tay,…

Tầm với hay vùng làm việc: kích thước và hình dáng vùng mà phần công tác có thể với tới.

Sự khéo léo, nghĩa là khả năng định vị và định hướng phần công tác trong vùng làm việc. Thông số này liên quan đến số bậc tự do của phần công tác.

Để định vị và định hướng phần công tác một cách tuỳ ý trong không gian 3 chiều nó cần có 6 bậc tự do, trong đó 3 bậc tự do để định vị, 3 bậc tự do để định hướng. Một số công việc như nâng hạ, xếp dỡ,… yêu cầu số bậc tự do ít hơn 6. Robot hàn, sơn thường có 6 bậc tự do. Trong một số trường hợp cần sự khéo léo, linh hoạt hoặc cần tối ưu hoá quỹ đạo,… người ta có thể dùng robot với số bậc tự do lớn hơn 6. Các tay máy có đặc điểm chung về kết cấu là gồm có các khâu, được nối với nhau bằng các khớp để hình thành một chuỗi động học hở, tính từ thân đến phần công tác. Các khớp được dùng phổ biến là khớp trượt và khớp quay. Tuỳ theo số lượng và cách bố trí các khớp mà có thể tạo ra tay máy kiểu tọa độ đề các, tọa độ trụ, tọa độ cầu, SCARA và kiểu tay người (Anthropomorphic).

Một robot tọa độ Descartes (còn gọi là robot tuyến tính) là một robot công nghiệp có ba trục điều khiển chính là tuyến tính (tức là chúng di chuyển theo đường thẳng chứ không phải xoay) và vuông góc với nhau. Ba khớp trượt tương ứng với việc di chuyển cổ tay lên xuống, vào-ra, tới-lùi. Trong số các ưu điểm khác, cách bố trí cơ khí này đơn giản hóa giải pháp cánh tay điều khiển Robot. Các robot kiểu Descartes với thanh nằm ngang được hỗ trợ ở cả hai đầu đôi khi được gọi là rô bốt cầu trục; về cơ học, chúng giống như cầu trục, mặc dù không phải là robot thông thường. Robot cầu trục thường khá lớn. Một ứng dụng phổ biến cho loại robot này là máy tính điều khiển số (máy CNC) và in 3D. Ứng dụng đơn giản nhất được sử dụng trong máy phay và máy vẽ, nơi bút hoặc bộ định tuyến dịch qua một mặt phẳng x-y trong khi công cụ được nâng lên và hạ xuống bề mặt để tạo ra thiết kế chính xác. Các máy gắp và đặt và máy vẽ cũng dựa trên nguyên lý cơ bản của robot tọa độ Descartes.



Hình 1.2. Robot hệ tọa độ Descartes

Tay máy kiểu tọa độ trụ khác với tay máy kiểu đề các ở khớp đầu tiên: dùng khớp quay thay cho khớp trượt. Vùng làm việc của nó có dạng hình trụ rỗng. Khớp trượt nằm ngang cho phép tay máy "thò" được vào khoang rỗng nằm ngang. Độ cứng vững cơ học của tay máy trụ tốt, thích hợp với tải nặng, nhưng độ chính xác định vị góc trong mặt phẳng nằm ngang giảm khi tầm với tăng.

Tay máy kiểu tọa độ cầu khác kiểu trụ do khớp thứ hai (khớp trượt) được thay bằng khớp quay. Nếu quỹ đạo chuyển động của phần công tác được mô tả trong toạ độ cầu thì mỗi bậc tự do tương ứng với một khả năng chuyển động và vùng làm việc của nó là khối cầu rỗng. Độ cứng vững của loại tay máy này thấp hơn 2 loại trên và độ chính xác định vị phụ thuộc vào tầm với. Tuy nhiên, loại này có thể "nhặt" được cả vật dưới nền.SCARA được đề xuất lần đầu vào năm 1979 tại Trường đại học Yamanashi (Nhật bản) dùng cho công việc lắp ráp. Đó là một kiêu tay máy có cấu tạo đặc biệt, gồm 2 khớp quay và 1 khớp trượt, nhưng cả 3 khớp đều có trục song song với nhau. Kết cấu này làm tay máy cứng vững hơn theo phương thẳng đứng nhưng kém cứng vững (Compliance) theo phương được chọn (Seirective), là phương ngang. Loại này chuyên dùng cho công việc lắp ráp (Assembly) với tải trọng nhỏ, theo phương thẳng đứng. Từ SCARA là viết tắt của "Selective Compliance Assembly Robot Arm" để mô tả các đặc điểm trên. Vùng làm việc của SCARA là một phần của hình trụ rỗng.

Hình 1.3. Robot SCARA

Tay máy kiểu tay người (Anthropomorphic), có cả 3 khớp đều là các khớp quay, trong đó trục thứ nhất vuông góc với 2 trục kia. Do sự tương tự với tay người, khớp thứ hai được gọi là khớp vai (Shoulder joint), khớp thứ ba là khớp khuỷu (Elbow joint), nối cẳng tay với khuỷu tay. Với kết cấu này, không có sự tương ứng giữa khả năng chuyển động của các khâu và số bậc tự do. Tay máy làm việc rất khéo léo, nhưng độ chính xác định vị phụ thuộc vị trí của phần công tác trong vùng làm việc. Vùng làm việc của tay máy kiểu này gần giống một phần khối cầu.

1.1.4. Ứng dụng của robot công nghiệp trong các lĩnh vực.

a. Sản xuất

Cánh tay robot công nghiệp có nhiều ứng dụng quan trọng trong ngành sản xuất, bao gồm:

Dây chuyền sản xuất tự động: Các cánh tay robot được sử dụng để thực hiện các nhiệm vụ sản xuất cụ thể trên dây chuyền sản xuất tự động. Chúng có thể lập trình để thực hiện công việc như lắp ráp sản phẩm, vận chuyển vật liệu, hoặc kiểm tra chất lượng sản phẩm.

Hàn và lắp ráp: Trong ngành công nghiệp sản xuất ô tô, điện tử và hàng điện tử tiêu dùng, cánh tay robot được sử dụng để thực hiện các công đoạn hàn và lắp ráp sản phẩm. Chúng có khả năng hoạt động chính xác và liên tục, giúp tăng năng suất và chất lượng sản phẩm.

Đóng gói và đóng kiện: Trên các dây chuyền sản xuất, robot công nghiệp được sử dụng để đóng gói sản phẩm vào các bao bì hoặc thùng carton, giúp tối ưu hóa quy trình đóng gói và đóng kiện, đồng thời giảm thời gian và chi phí lao động.

Vận chuyển và xếp hàng: Cánh tay robot có thể được sử dụng để vận chuyển sản phẩm từ điểm này sang điểm khác trên dây chuyền sản xuất hoặc trong kho hàng. Chúng có thể tự động xếp hàng hoặc phân loại sản phẩm theo yêu cầu cụ thể.

Kiểm tra và kiểm soát chất lượng: Robot công nghiệp có thể được sử dụng để thực hiện các nhiệm vụ kiểm tra chất lượng sản phẩm bằng cách sử dụng cảm biến và công nghệ khác để phát hiện lỗi hoặc sự không đồng nhất trong sản phẩm.

Sản xuất linh kiện: Trong quá trình sản xuất các linh kiện nhỏ và phức tạp, cánh tay robot có thể được sử dụng để gia công, vận chuyển và kiểm tra linh kiện một cách tự động.

Nhờ vào tính linh hoạt và khả năng tự động hóa cao, cánh tay robot công nghiệp

đã trở thành một phần quan trọng trong quá trình sản xuất hiện đại, giúp tăng cường năng suất, chất lượng và hiệu quả trong ngành công nghiệp sản xuất.

Hình 1.4. Robot trong sản xuất ô tô

b. Y tế

Cánh tay robot công nghiệp được áp dụng trong lĩnh vực y tế với nhiều ứng dụng quan trọng như sau:

Phẫu thuật robot hỗ trợ: Cánh tay robot được sử dụng trong các ca phẫu thuật để hỗ trợ bác sĩ thực hiện các thao tác phức tạp và chính xác hơn. Điều này có thể giúp giảm thiểu sự xâm lấn và thời gian phục hồi sau phẫu thuật.

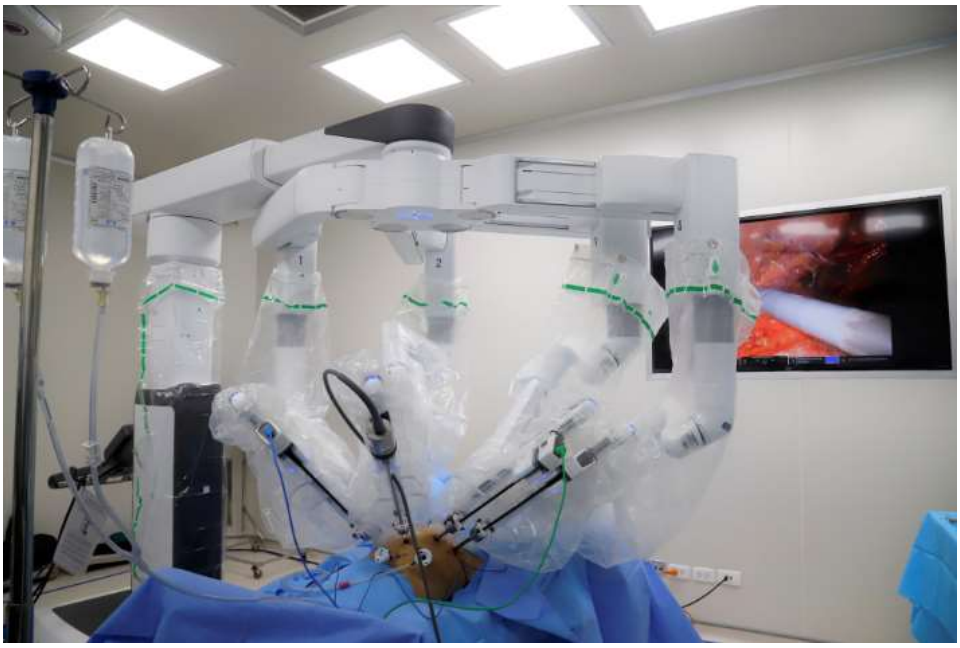
Chẩn đoán và điều trị: Cánh tay robot có thể được sử dụng để cung cấp hỗ trợ trong quá trình chẩn đoán và điều trị bằng cách mang lại thông tin chi tiết về vị trí của bệnh nhân hoặc hỗ trợ trong việc tiến hành các thủ tục y tế phức tạp như xét nghiệm và tiêm thuốc.

Dụng cụ y tế tự động: Cánh tay robot có thể được sử dụng để phát triển các dụng cụ y tế tự động như bàn tay robot hỗ trợ việc giao tiếp cho người khuyết tật hoặc các thiết bị hỗ trợ khác trong việc hồi phục chức năng.

Dịch vụ chăm sóc bệnh nhân: Cánh tay robot có thể được lập trình để cung cấp dịch vụ chăm sóc bệnh nhân, bao gồm việc vận chuyển vật phẩm y tế, phát thuốc và hỗ trợ trong việc giám sát sức khỏe của bệnh nhân.

Nghiên cứu và phát triển dược phẩm: Trong nghiên cứu dược phẩm, cánh tay robot có thể được sử dụng để tự động hóa các giai đoạn của quá trình nghiên cứu, từ phân tích dữ liệu đến thử nghiệm dược phẩm.

Các ứng dụng của cánh tay robot công nghiệp trong lĩnh vực y tế đang ngày càng mở rộng và mang lại nhiều lợi ích đáng kể trong việc cải thiện chăm sóc sức khỏe và tiến bộ trong ngành y tế.



Hình 1.5. Cánh tay robot công nghiệp được ứng dụng trong phẫu thuật

c. Nông nghiệp

Cánh tay robot công nghiệp đã có những ứng dụng đa dạng trong lĩnh vực nông nghiệp, bao gồm:

Gieo hạt và thu hoạch tự động: Các cánh tay robot có thể được sử dụng để gieo hạt và thu hoạch các loại cây trồng tự động. Chúng có khả năng chính xác trong việc đặt hạt giống và thu hoạch trái cây hoặc rau củ theo lịch trình được lập trình trước, giúp tối ưu hóa năng suất và giảm chi phí lao động.

Quản lý và chăm sóc cây trồng: Các cánh tay robot có thể được sử dụng để kiểm tra và chăm sóc cây trồng. Chúng có thể phun thuốc trừ sâu, tưới nước, cắt tỉa hoặc thu hoạch các loại cây trồng một cách tự động và chính xác.

Thu gom thông tin và giám sát: Robot có thể được trang bị cảm biến và công nghệ giám sát để thu thập dữ liệu về đất đai, thời tiết, chất lượng đất, và sức khỏe của cây

trồng. Thông tin này sau đó có thể được sử dụng để tối ưu hóa quản lý nông nghiệp và ra quyết định thông minh về việc canh tác.

Sử dụng trong vườn trồng thủy sản: Trong lĩnh vực trồng thủy sản, robot có thể được sử dụng để tưới nước, theo dõi chất lượng nước, thu hoạch sản phẩm và thậm chí là để di chuyển môi trường nuôi trồng.

Gia tăng tự động hóa trong sản xuất nông nghiệp: Robot có thể thực hiện các nhiệm vụ như cắt cỏ tự động, vận chuyển sản phẩm, hoặc thậm chí là hỗ trợ trong việc kiểm tra đất đai để xác định vùng trồng phù hợp.

Cánh tay robot công nghiệp đã có nhiều ứng dụng trong lĩnh vực nông nghiệp, giúp tăng cường năng suất, chất lượng sản phẩm và giảm thiểu sự phụ thuộc vào lao động trong môi trường nông nghiệp.

Hình 1.6. Cánh tay robot công nghiệp nhận dạng quả chín và thu hoạch

d. Hàng không vũ trụ

Cánh tay robot công nghiệp có nhiều ứng dụng tiềm năng trong ngành hàng không vũ trụ, bao gồm:

Sản xuất và lắp ráp: Cánh tay robot có thể được sử dụng trong quá trình sản xuất và lắp ráp các thành phần của tàu vũ trụ hoặc các thiết bị hàng không. Chúng có khả năng thực hiện các công việc lặp đi lặp lại một cách chính xác và hiệu quả, đảm bảo chất

lượng và độ chính xác cao trong quá trình sản xuất.

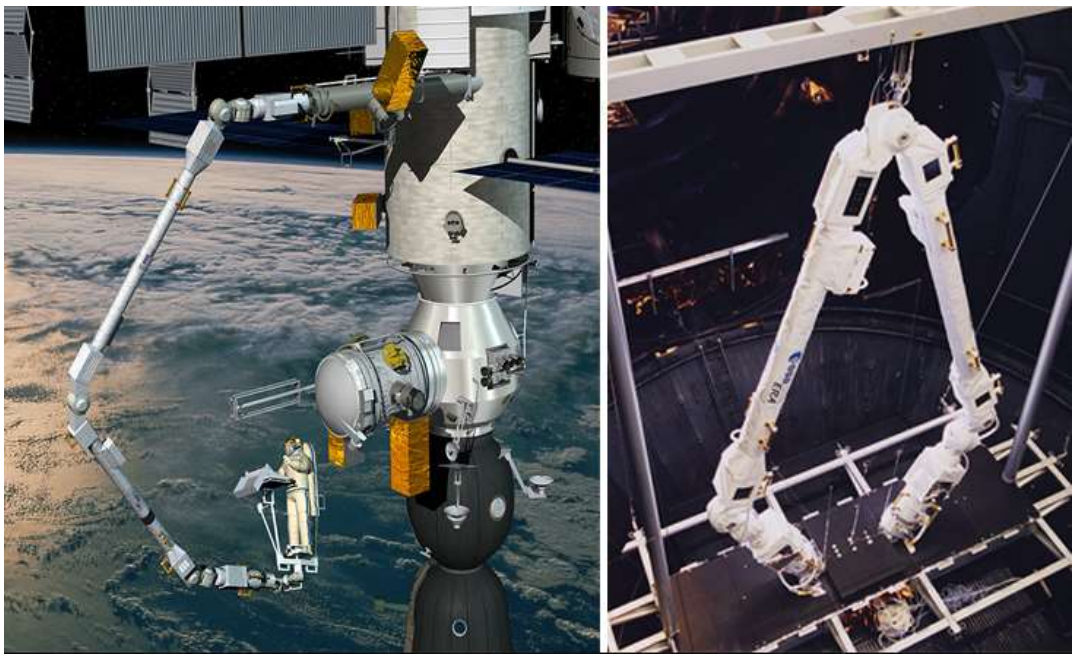
Kiểm tra và bảo dưỡng: Cánh tay robot có thể được sử dụng để kiểm tra, bảo dưỡng và sửa chữa các thành phần của tàu vũ trụ hoặc máy bay mà con người có thể gặp khó khăn hoặc không thể tiếp cận dễ dàng. Điều này giúp tăng cường an toàn và hiệu suất của các phương tiện hàng không vũ trụ.

Hỗ trợ nhiệm vụ không gian: Trong không gian, cánh tay robot có thể được sử dụng để thực hiện các nhiệm vụ như vận chuyển và lắp đặt các thiết bị, cảm biến hoặc công cụ trong không gian mà không yêu cầu sự can thiệp trực tiếp của con người.

Phục vụ du hành viên: Trong không gian, cánh tay robot có thể hỗ trợ du hành viên bằng cách cung cấp các dịch vụ như chế biến thức ăn, duy trì vệ sinh hoặc hỗ trợ trong các hoạt động hàng ngày khác.

Xây dựng cơ sở không gian: Trong tương lai, cánh tay robot có thể được sử dụng để xây dựng cơ sở không gian trên các hành tinh khác hoặc trên vũ trụ, giúp chuẩn bị cho việc định cư hoặc các hoạt động khám phá không gian dài hạn.

Các ứng dụng của cánh tay robot công nghiệp trong hàng không vũ trụ mang lại tiềm năng lớn trong việc cải thiện quy trình sản xuất, bảo dưỡng và thực hiện các nhiệm vụ không gian phức tạp, đồng thời tăng cường an toàn và hiệu suất cho ngành công nghiệp này.



Hình 1.7. Cánh tay robot trong việc sửa chữa các thiết bị ngoài không gian

1.1.5. Bài toán điều khiển

Bài toán điều khiển trong robot là một lĩnh vực quan trọng trong ngành robot học và điện tử. Nó tập trung vào 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.

Thiết kế bộ điều khiển: Dựa trên mô hình hóa, thiết kế các thuật toán và hệ thống điều khiển để điều khiển hoạt động của robot. 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

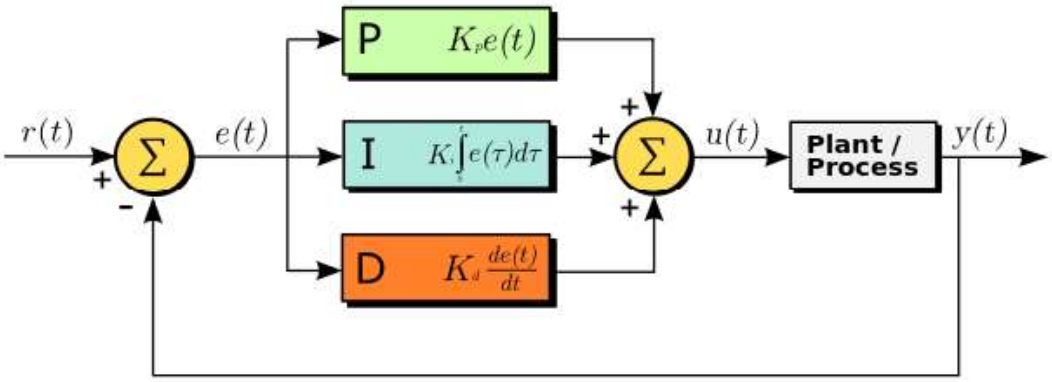
Bộ điều khiển vi phân tỉ lệ (PID - Proportional Integral Derivative) là một hệ thống điều khiển phản hồi phổ biến được sử dụng rộng rãi trong các ứng dụng công nghiệp. Bộ điều khiển PID chính là loại bộ điều khiển phản hồi được áp dụng phổ biến nhất trong các hệ thống điều khiển.

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). Giá trị tỷ lệ điều chỉnh tác động của sai số hiện tại, giá trị tích phân điều chỉnh tác động của tổng các sai số tích luỹ từ quá khứ, và giá trị đạo hàm điều chỉnh tác động của tốc độ biến đổi của sai số. Tổng hợp của ba tác động này được sử dụng để điều chỉnh quá trình thông qua một phần tử điều khiển như vị trí của van điều khiển hoặc bộ nguồn cho phần tử gia nhiệt.

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. Sơ đồ dưới đây minh họa nguyên lý hoạt động của một bộ PID.



Hình 1.8. Bộ điều khiển PID

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.

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. Ở dạng chuẩn, phương trình có thể được viết lại 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:

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

Thành phần tỷ lệ tạo ra giá trị đầu ra tỷ lệ thuận với giá trị sai số hiện tại. Đáp ứng tỷ lệ này có thể được điều chỉnh bằng cách nhân sai số với một hằng số K_p gọi là hằng số khuếch đại tỷ lệ. 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. Nếu hệ số này quá cao có thể dẫn đến hệ thống mất ổn định. 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 được phát triển từ lý thuyết tập mờ để thực hiện lập luận một cách xấp xỉ thay vì lập luận chính xác theo logic vị từ cổ điển. Logic mờ cho phép độ liên thuộc có giá trị trong khoảng đóng 0 và 1. Với mỗi thành phần ngôn ngữ x_k, nếu nó nhận được một khả năng $\mu(x_k)$ thì tập F gồm các cặp $(x,\mu(x_k))$ được gọi là tập mờ.

b. Hàm liên thuộc

Hàm thuộc sẽ lượng hóa mức độ mà các phần tử x thuộc về tập cơ sở X. Nếu hàm cho kết quả 0 đối với một phần tử thì phần tử đó không có trong tập đã cho, kết quả 1 mô tả một thành viên toàn phần của tập hợp. Các giá trị trong khoảng mở từ 0 đến 1 đặc trưng cho các thành viên mờ. Các dạng hàm thuộc trong logic mờ có rất nhiều dạng như: Gaussian, PI-shape, S-shape, Sigmoidal, Z-shape, …

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. Các luật hợp thành bao gồm: luật Max - Min, luật Max – Prod, luật Sum – Min, Luật Sum – Prod.

Định lý Mamdani: Độ phụ thuộc của kết luận không được lớn hơn độ phụ thuộc điều kiện. Nếu hệ thống có nhiều đầu vào và nhiều đầu ra thì mệnh đề suy diễn 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.

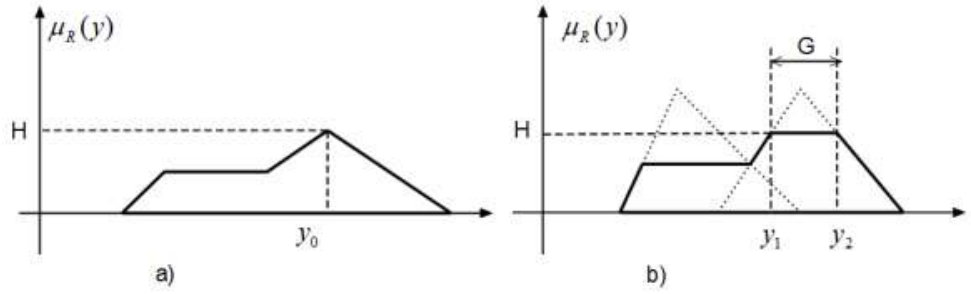
$$G = \{y \in Y \mid \mu_R(y) = H\} \tag{1.9}$$

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

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

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

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



Hình 1.9. Phương pháp giải mờ cực đại

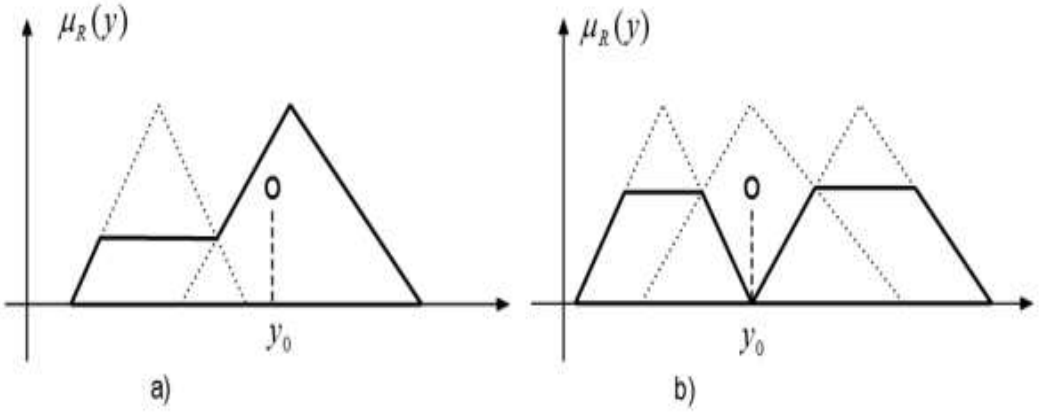
Phương pháp trọng tâm:

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:

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

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

Đây là phương pháp được sử dụng phổ biến nhất. 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 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.

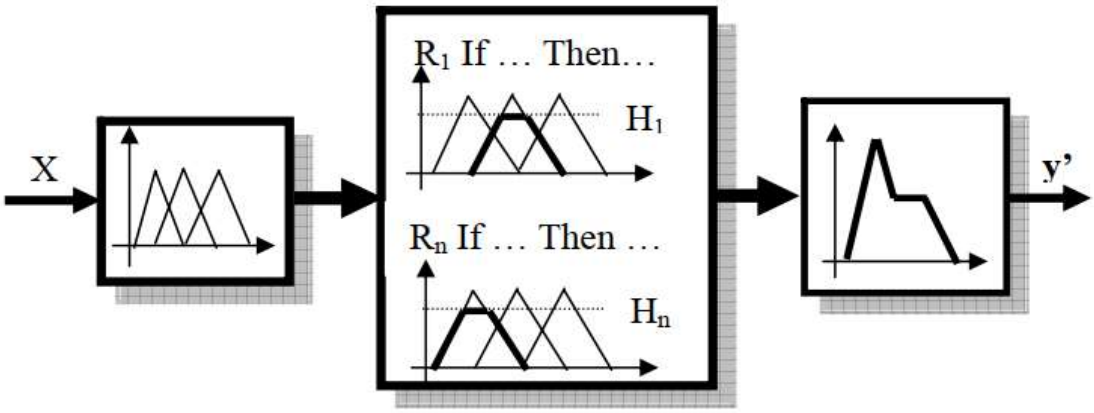


Hình 1.10. Giải mờ bằng phương pháp trọng tâm

f. Cấu trúc bộ điều khiển mờ

Một bộ điều khiển mở gồm 3 khâu cơ bản: khâu mờ hóa; thực hiện luật hợp thành; khâu giải mờ.

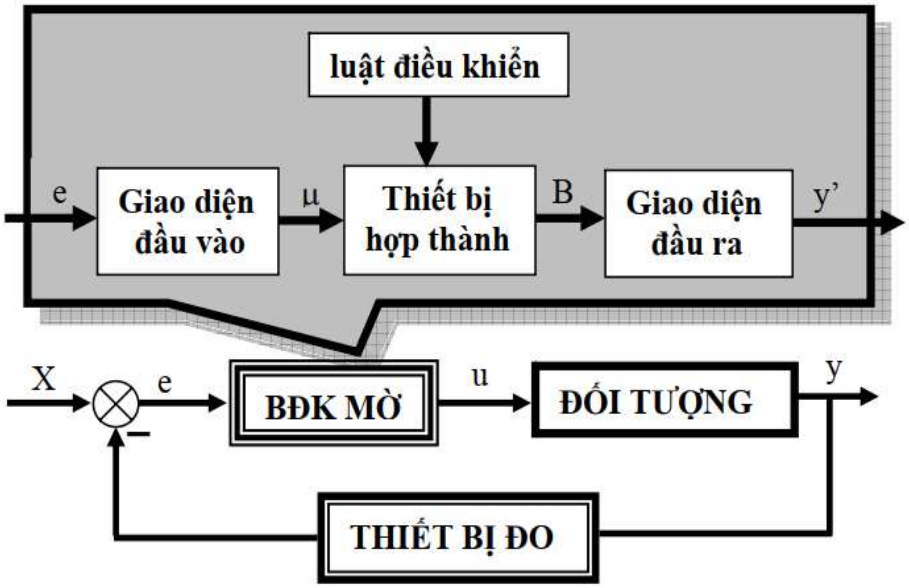
Xét bộ điều khiển mờ MISO sau, với véc-tơ đầu vào $X = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix}^T$, ta có mô hình bộ điều khiển như hình dưới đây



Hình 1.11. Bộ điều khiển mờ

g. Nguyên lý điều khiển mờ

Nguyên lý hoạt động của một bộ điều khiển mờ điển hình được minh họa như hình bên dưới.



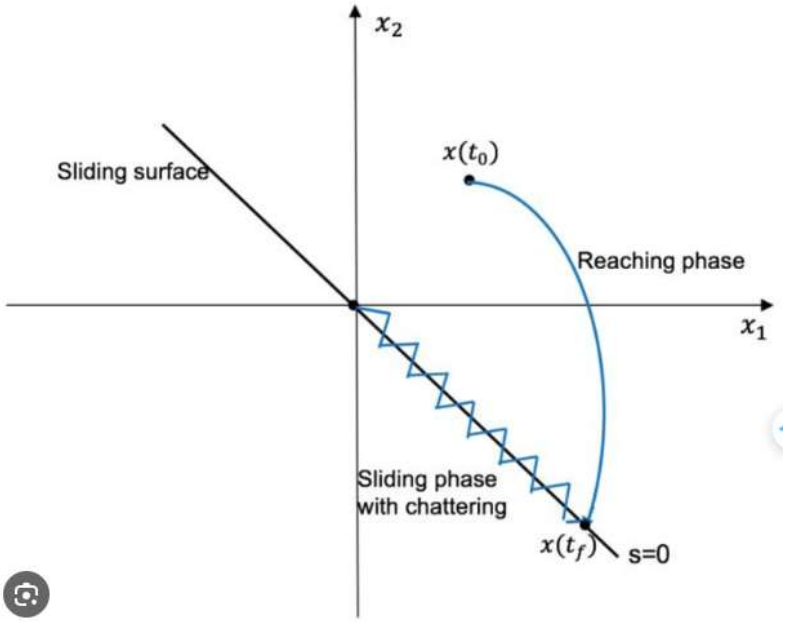
Hình 1.12. Nguyên lý điều khiển mờ

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 1.13. 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 thích nghi

a. Định nghĩa

Điều khiển thích nghi là một phương pháp sử dụng trong hệ thống điều khiển để bộ điều khiển có khả năng thích ứng với sự thay đổi trong thông số của hệ thống hoặc những điều kiện ban đầu không chắc chắn. Ví dụ, trong quá trình điều khiển một chiếc máy bay, với việc khối lượng của máy bay giảm dần do tiêu thụ nhiên liệu, cần phải áp dụng một luật điều khiển linh hoạt để điều chỉnh hoạt động của máy bay phù hợp với các điều kiện thay đổi như vậy.

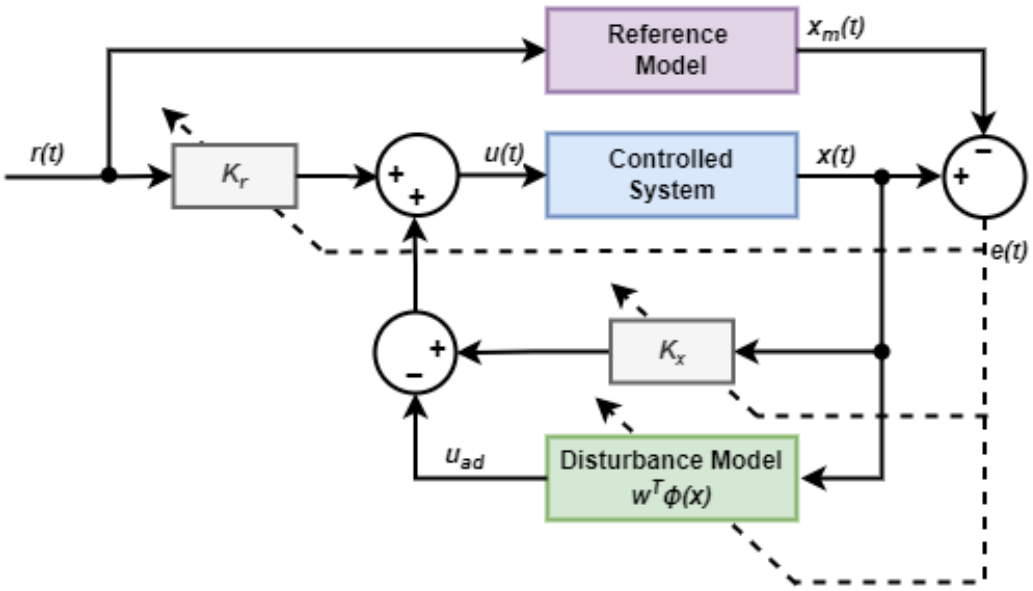
Lý thuyết điều khiển thích nghi không chỉ đem lại các lợi ích về hiệu suất và độ chính xác của hệ thống mà còn giúp hệ thống tự động thích ứng và hoạt động ổn định trong môi trường thay đổi không ngừng. Tuy nhiên, việc triển khai và áp dụng trong thực tế còn đối mặt với các thách thức liên quan đến độ tin cậy, tính khả thi, và sự ổn định của hệ thống điều khiển. Đây vẫn là một lĩnh vực nghiên cứu sôi động, đặc biệt là khi cần đối mặt với môi trường với mức độ biến đổi lớn và không chắc chắn.

b. Ước lượng tham số

Điều khiển thích nghi dựa trên việc ước lượng các tham số của hệ thống. Các phương pháp ước lượng phổ biến bao gồm phương pháp đệ qui bình phương tối thiểu và gradient descent. Cả hai phương pháp này cung cấp các quy tắc cập nhật được sử dụng để thay đổi các ước lượng theo thời gian thực, trong quá trình hoạt động của hệ thống. Ổn định Lyapunov được áp dụng để tạo ra các quy tắc cập nhật và tìm kiếm tiêu chuẩn hội tụ, thường được sử dụng để đảm bảo tính ổn định. Phép chiếu và tiêu chuẩn hóa thường được áp dụng để tăng cường tính ổn định của các thuật toán ước lượng. Điều này còn được gọi là điều khiển có khả năng điều chỉnh.

c. Phương pháp thường được áp dụng trong điều khiển thích nghi

Điều khiển thích nghi dựa trên mô hình (Model Reference Adaptive Control - MRAC): Phương pháp này xây dựng một mô hình tham chiếu và điều chỉnh hệ thống điều khiển sao cho hành vi của hệ thống thực tế xấp xỉ với mô hình tham chiếu.



Hình 1.14. Bộ điều khiển thích nghi dựa trên mô hình

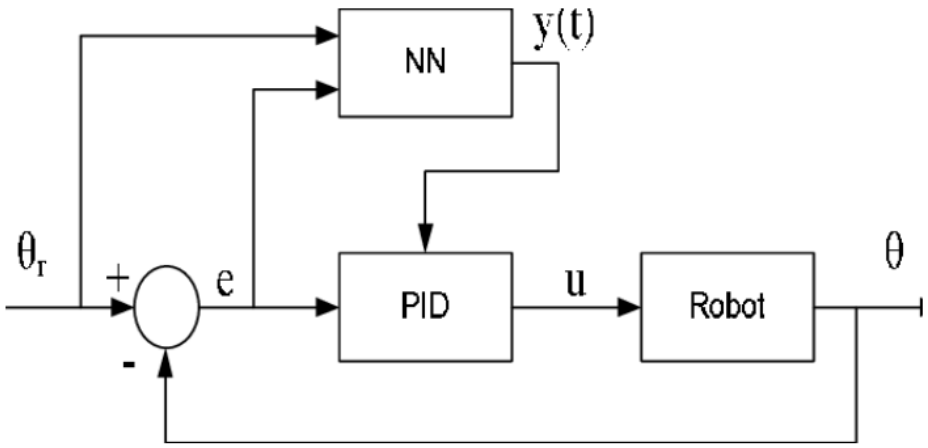
Điều khiển thích nghi dựa trên sai số (Model-Free Adaptive Control): Phương pháp này không yêu cầu một mô hình toàn diện của hệ thống, thay vào đó, nó điều chỉnh các thông số điều khiển dựa trên thông tin về sai số giữa hành vi thực tế và mong muốn.

Học tăng cường (Reinforcement Learning) trong điều khiển thích nghi: Một lĩnh vực nổi bật trong lý thuyết điều khiển thích nghi, học tăng cường là quá trình hệ thống tự học thông qua thử nghiệm và lỗi, tối ưu hóa hành vi dựa trên phần thưởng hoặc hình phạt.

Mạng nơ-ron và học sâu trong điều khiển thích nghi: Sự tiến bộ trong lĩnh vực học máy và mạng nơ-ron đã mở ra cơ hội sử dụng các phương pháp này để thiết kế các hệ thống điều khiển thích nghi với khả năng học và điều chỉnh linh hoạt.

1.2.5. Điều khiển nhờ mạng nơ-ron.

Điều khiển nhờ mạng nơ-ron (Neural Network Control) là một lĩnh vực trong lý thuyết điều khiển tự động sử dụng các mạng nơ-ron nhân tạo để thực hiện chức năng 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 1.15. Sử dụng mạng nơ-ron hỗ trợ điều chỉnh thông số của PID

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 xử lý 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. Việc huấn luyện mạng nơ-ron đòi hỏi một lượng dữ liệu lớn và đa dạng để đảm bảo tính chính xác và hiệu quả của hệ thống điều khiển. 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

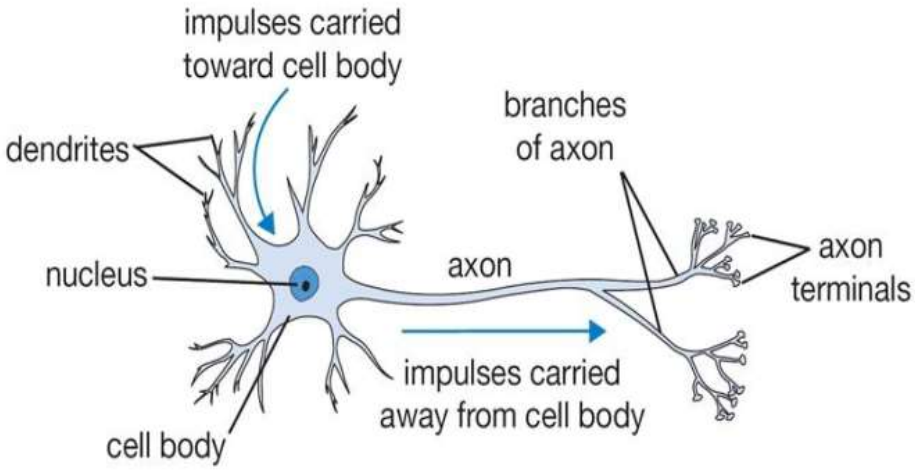
Chương này trình bày về khái niệm, cấu tạo và phân loại các loại robot công nghiệp phổ biên, cùng với đó là nêu được tổng quan các thuật toán điều khiển cho robot và xác định bài toán điều khiển cụ thể cho đồ án.

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

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.

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) kết nối với các nơ-ron khác. Đơn giản, 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. Nếu 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.

b. Mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo (Artificial Neural Network - ANN) là một mô hình xử lý thông tin được tạo ra dựa trên cách hoạt động của hệ thống thần kinh trong các sinh vật. Nó bao gồm một số lượng lớn các nơ-ron kết nối với nhau để xử lý thông tin. ANN

tương tự như bộ não của con người và có khả năng học từ kinh nghiệm thông qua quá trình huấn luyện. Nó có khả năng lưu trữ và sử dụng các kiến thức học được từ dữ liệu để dự đoán thông tin mới mà mô hình chưa từng gặp.

Mạng nơ-ron có khả năng thích ứng được với mọi thay đổi từ đầu vào. Do vậy, nó có thể đưa ra được mọi kết quả một cách tốt nhất có thể mà bạn không cần phải thiết kế lại những tiêu chí đầu ra.

Mặc dù mạng nơ-ron lấy cảm hứng từ não bộ và cách nó hoạt động.Tuy nhiên không phải bắt chước toàn bộ các chức năng của nó. Việc chính của chúng ta là dùng mô hình đấy đi giải quyết các bài toán chúng ta cần.

Mạng nơ-ron nhân tạo có thể hoạt động như mạng nơ-ron của con người. Mỗi một nơ-ron thần kinh trong nơ-ron nhân tạo là hàm toán học với chức năng thu thập và phân loại các thông tin dựa theo cấu trúc cụ thể.

Mạng nơ-ron có sự tương đồng chuẩn mạnh vối những phương pháp thống kê như đồ thị đường cong và phân tích hồi quy. Nơ-ron Network có chứa những lớp bao hàm các nút được liên kết lại với nhau. Mỗi nút lại là một tri giác có cấu tạo tương tự với hàm hồi quy đa tuyến tính. Bên trong một lớp tri giác đa lớp, chúng sẽ được sắp xếp dựa theo các lớp liên kết với nhau. Lớp đầu vào sẽ thu thập các mẫu đầu vào và lớp đầu ra sẽ thu nhận các phân loại hoặc tín hiệu đầu ra mà các mẫu đầu vào có thể phản ánh lại.

2.1.2. Cấu trúc mạng nơ-ron

a. Các lớp trong mạng nơ-ron

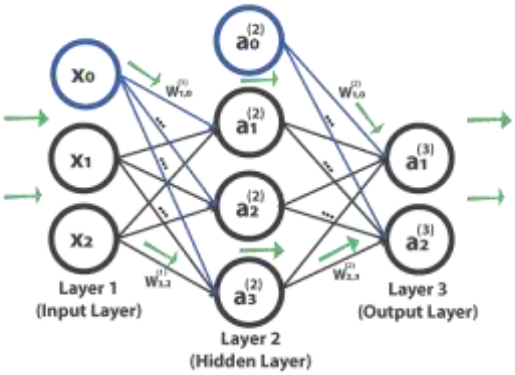
Mạng nơ-ron là sự kết hợp của nhiều tầng perceptron hay còn gọi là perceptron đa tầng. Và mỗi một mạng nơ-ron thường bao gồm 3 kiểu lớp là:

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 nhiều lớp này ta có thể xấp xỉ được những hàm số phức tạp, hay nói cách khác ta có model để biểu diễn 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.



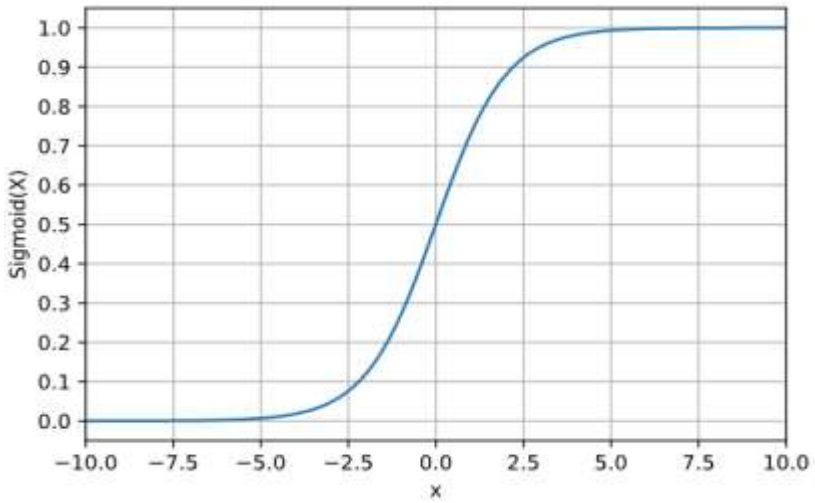
Hình 2.2. Mô hình cấu trúc của một mạng nơ-ron gồm 3 lớp

b. Hàm kích hoạt

Hàm kích hoạt mô phỏng tỷ lệ truyền xung qua axon của một neuron thần kinh. Trong một mạng nơ-ron nhân tạo, hàm kích hoạt đóng vai trò là thành phần phi tuyến tại output của các nơ-ron. Chúng ta cần phải sử dụng hàm kích hoạt phi tuyến vì nếu như không có nó thì mạng nơ-ron dù có nhiều lớp cũng chỉ có có hiệu quả như một lớp tuyến tính mà thôi. Một số hàm kích hoạt thường được sử dụng trong mạng nơ-ron.

Hàm sigmoid:

$$f(z) = \frac{1}{1+e^{-z}} \tag{2.1}$$

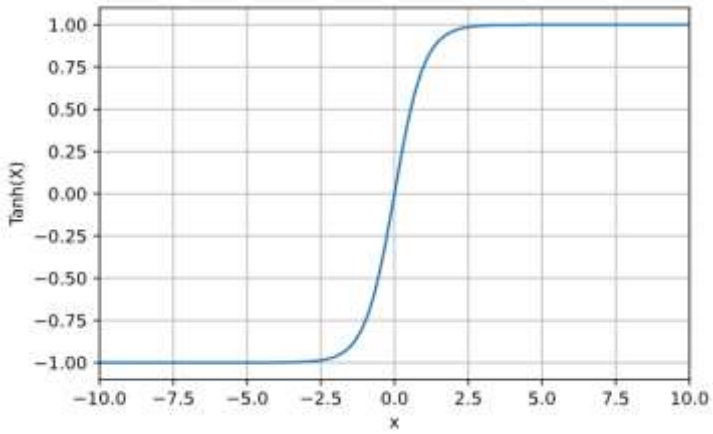


Hình 2.3. Đồ thị hàm sigmoid

Hàm Sigmoid nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (0;1). Đầu vào là số thực âm rất nhỏ sẽ cho đầu ra tiệm cận với 0, ngược lại, nếu đầu vào là một số thực dương lớn sẽ cho đầu ra là một số tiệm cận với 1.

Hàm tanh:

$$f(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.2}$$

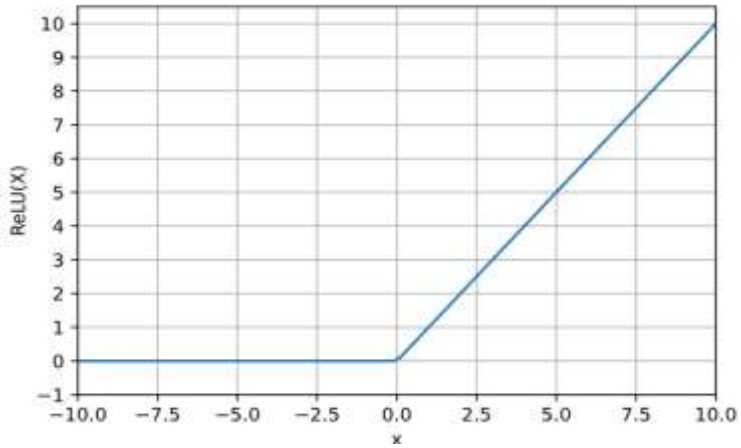


Hình 2.4. Đồ thị hàm tanh

Hàm tanh có đầu vào là một số thực và chuyển thành một giá trị trong khoảng (-1; 1). Cũng như Sigmoid, hàm tanh bị bão hoà ở 2 đầu (đạo hàm thay đổi rất ít ở 2 đầu). Tuy nhiên hàm Tanh lại đối xứng qua 0 nên khắc phục được một nhược điểm của Sigmoid.

Hàm ReLU:

$$f(z) = \max(0, z) \tag{2.3}$$

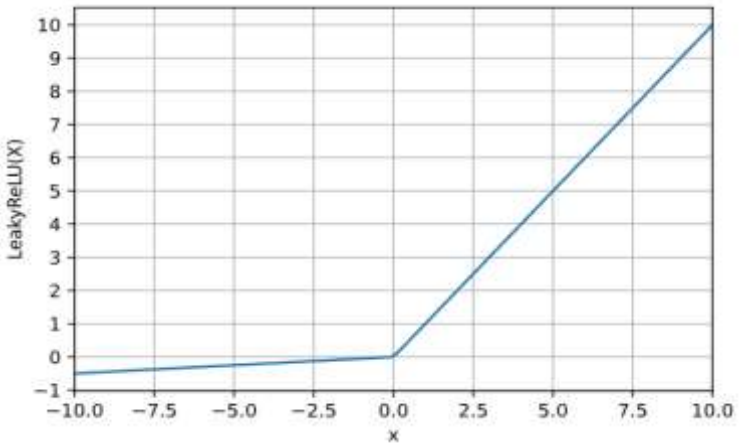


Hình 2.5. Đồ thị hàm ReLU

Hàm ReLU đang được sử dụng khá nhiều trong những năm gần đây khi huấn luyện các mạng neuron. ReLU đơn giản lọc các giá trị < 0. Nhìn vào công thức chúng ta dễ dàng hiểu được cách hoạt động của nó. Một số ưu điểm khá vượt trội của nó so với Sigmoid và Tanh như tốc độ hội tụ nhanh hơn, tính toán nhanh hơn. Tuy nhiên ReLU cũng có một nhược điểm: với các nút có giá trị nhỏ hơn 0, qua ReLU activation sẽ thành 0, hiện tượng đấy gọi là "Dying ReLU". Nếu các node bị chuyển thành 0 thì sẽ không có ý nghĩa với bước linear activation ở lớp tiếp theo và các hệ số tương ứng từ nút đấy cũng không được cập nhật với gradient descent. Nên hàm Leaky ReLU được tìm ra.

Hàm Leaky ReLU:

$$f(z) = \max(0.1z, z) \tag{2.4}$$



Hình 2.6. Đồ thị hàm Leaky ReLU

Leaky ReLU là một cố gắng trong việc loại bỏ "dying ReLU". Thay vì trả về giá trị 0 với các đầu vào nhỏ hơn 0 thì Leaky ReLU tạo ra một đường xiên có độ dốc nhỏ (xem đồ thị). Có nhiều báo cáo về việc hiệu Leaky ReLU có hiệu quả tốt hơn ReLU, nhưng hiệu quả này vẫn chưa rõ ràng và nhất quán.

c. Hàm mất mát (Loss function)

Loss function kí hiệu là L, là thành phần cốt lõi của evaluation function và objective function. Cụ thể, trong công thức thường gặp

$$L_D(f_w) = \frac{1}{|D|} \sum_{(x,y)\in D} L(f_w(x), y) \tag{2.5}$$

Loss function trả về một số thực không âm thể hiện sự chênh lệch giữa hai đại lượng $\hat{y}$ dự đoán và y đúng. Loss function giống như một hình thức để bắt mô hình

đóng phạt mỗi lần dự đoán sai và số mức phạt tỷ lệ thuận với độ lớn của sai số. Trong mọi bài toán học có giám sát, mục tiêu của ta luôn là giảm thiểu tổng mức phạt phải đóng. Trong trường hợp lý tưởng $\hat{y} = y$, hàm mất mát sẽ trả về giá trị cực tiểu bằng 0.

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

Square loss:

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2 \tag{2.6}$$

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:

$$CE = -\sum_{i}^{C} t_i \log(s_i) \tag{2.7}$$

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.

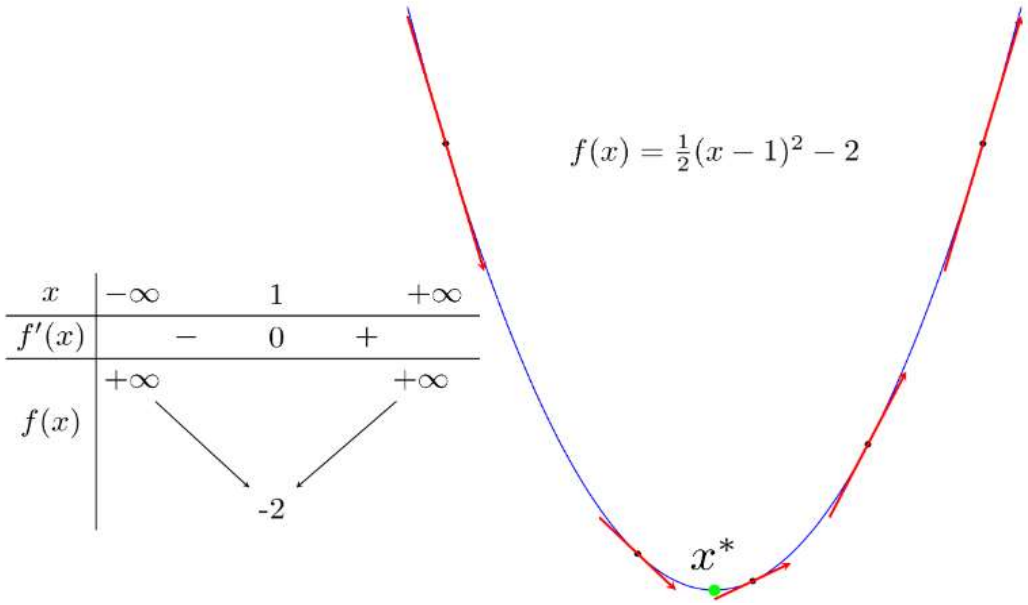
2.1.3. Phương pháp tối ưu hàm mất mát Gradient Descent (GD)

Trong quá trình làm việc với NN, chúng ta thường xuyên phải tìm giá trị nhỏ nhất của một hàm số nào đó. Nhìn chung việc tìm global minimum của các hàm mất mát trong học máy là rất phức tạp, thậm chí là bất khả thi. Thay vào đó, người ta thường cố gắng tìm các điểm local minimum, và ở một mức độ nào đó, coi đó là nghiệm cần tìm của bài toán.

Các điểm cực tiểu cục bộ là nghiệm của phương trình đạo hàm bằng 0. Nếu có khả năng xác định được toàn bộ các điểm cực tiểu này, chúng ta có thể thay từng điểm cực tiểu này vào hàm số và tìm điểm làm cho hàm có giá trị nhỏ nhất. Tuy nhiên, trong hầu hết các trường hợp, việc giải phương trình đạo hàm bằng 0 là không thể. Nguyên nhân có thể đến từ sự phức tạp của dạng của đạo hàm, số chiều lớn của dữ liệu, hoặc có quá

nhiều điểm dữ liệu.

Phương pháp tiếp cận phổ biến nhất là bắt đầu từ một điểm gần với nghiệm của bài toán, sau đó sử dụng các phép toán lặp để tiến gần tới điểm cần tìm, đến khi đạo hàm gần với 0. Gradient Descent (viết tắt là GD) và các biến thể của nó là một trong những phương pháp phổ biến được sử dụng rộng rãi.



Hình 2.7. Gradient descent cho hàm 1 biến

Ý tưởng của thuật toán như sau: nếu đạo hàm của hàm số tại x_t lớn hơn 0 thì x_t nằm bên phải so với x^* (và ngược lại). Để điểm tiếp theo x_{t+1} gần với x^* hơn, chúng ta cần di chuyển x_t về phía bên trái, tức về phía phần âm. Nói cách khác, chúng ta cần di chuyển ngược dấu với đạo hàm.

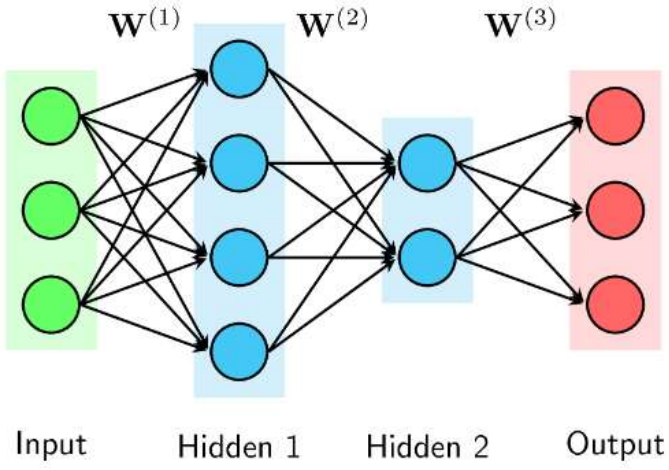
Để tổng quát, ta giả sử cần tìm global minimum cho hàm $f(\theta)$, trong đó θ là một véc-tơ (biển diễn các tham số của mô hình cần tối ưu). Đạo hàm của hàm số tại một điểm θ bất kỳ được ký hiệu là $\nabla_\theta f(\theta)$. Khi đó, ta khởi tạo θ_0 bất kỳ, sau đó ở vòng lặp thứ t, quy tắc cập nhật là:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta f(\theta) \tag{2.8}$$

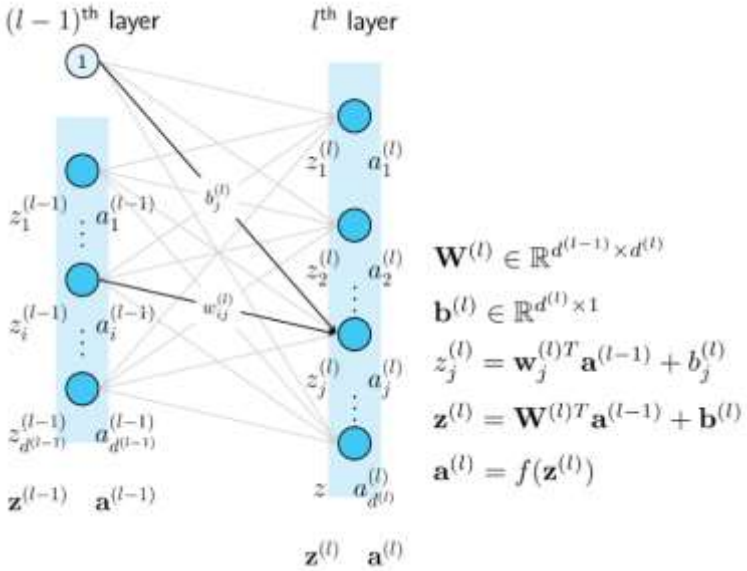
trong đó η gọi là tốc độ học (dấu trừ thể hiện việc đi ngược lại đạo hàm). Việc lựa chọn η rất quan trọng trong bài toán thực tế. Việc lựa chọn giá trị phụ thuộc nhiều vào bài toán và phải làm một vài thí nghiệm để chọn ra giá trị tốt nhất.

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 NN nhiều lớp, ta cần sử dụng đến thuật toán lan truyền ngược.



Hình 2.8. Mạng NN với các lớp ẩn



Hình 2.9. Liên hệ giữa các lớp

Để áp dụng GD, chúng ta cần tính được gradient 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:

$$
\begin{aligned}
&a^{(0)} = x \\
&z_i^{(l)} = \left(w_i^{(l)}\right)^T a^{(l-1)} + b_i^{(l)} \\
&a^{(l)} = f\left(z^{(l)}\right),\ l = 1, 2, ..., L \\
&\hat{y} = a^{(L)}
\end{aligned}
\tag{2.9}
$$

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

Giả sử $J\left(W, b, X, Y\right)$ là một hàm mất mát của bài toán, 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 với mỗi cột tương ứng với một điểm dữ liệu.

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

$$
J\left(W, b, X, Y\right) = \frac{1}{N}\sum_{n=1}^{N}\left\|y_n - \hat{y}_n\right\|_2^2 = \frac{1}{N}\sum_{n=1}^{N}\left\|y_n - a_n^{(L)}\right\|_2^2 \tag{2.10}
$$

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

Do hàm mất mát ở đây không phụ thuộc trực tiếp vào các hệ số nên phương pháp phổ biến nhất được dùng có tên là Backpropagation giúp tính gradient ngược từ layer cuối cùng đến layer đầu tiên. Layer cuối cùng được tính trước do nó gần với đầu ra dự đoán và hàm mất mát. Việc tính gradient của các layer trước được thực hiện dựa trên quy tắc "chain rule" (đạo hàm của hàm hợp).

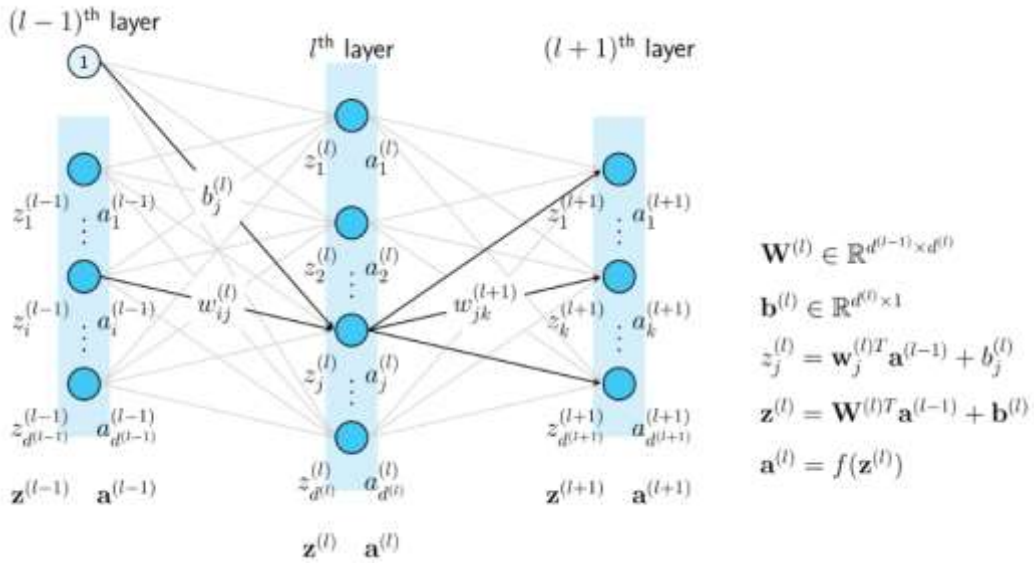
Đạo hàm của hàm mất mát theo chỉ một thành phần của ma trận trọng số của lớp cuối cùng:

$$
\frac{\partial J}{\partial w_{ij}^{(L)}} = \frac{\partial J}{\partial z_j^{(L)}} . \frac{\partial z_j^{(L)}}{\partial w_{ij}^{(L)}} = e_j^{(L)} a_i^{(L-1)} \tag{2.11}
$$

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:

$$\frac{\partial J}{\partial b_j^{(L)}} = \frac{\partial J}{\partial z_j^{(L)}} \cdot \frac{\partial z_j^{(L)}}{\partial b_j^{(L)}} = e_j^{(L)} \tag{2.12}$$



Hình 2.10. Mô phỏng cách tính backpropagation

Sau khi đã tính được đạo, 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 bải toán động học ngược. Nhưng trong đồ án này sẽ sử dụng phương pháp Levenberg-Marquardt. Giải thuật này được phát triển bởi Kenneth Levenberg và Donald Marquart, cung cấp một số giải pháp để giải quyết vấn đề cực tiểu hóa một hàm không tuyến tính. Giải pháp này có độ hội tụ nhanh và ổn định. Trong lĩnh vực mạng nơ-ron nhân tạo. thuật toán này rất phù hợp để huấn luyện những mạng có cỡ nhỏ và trung bình (vài trăm 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:

$$E(p,w) = \frac{1}{2} \sum_{i=1}^{m} \sum_{t=1}^{n_Q} e^2 \tag{2.13}$$

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:

$$(q\_pre - q\_true)^2 \tag{2.14}$$

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:

$$w_{k+1} = w_k - (J_k^T J_k \_ \mu I)^{-1} J_k e_k \tag{2.15}$$

với:

μ : là hệ số tốc độ học

k : là chỉ số lặp

J: Ma trận Jacobian có dạng:

$$J = \begin{bmatrix} \frac{\partial e_1(w)}{\partial w_1} & \frac{\partial e_1(w)}{\partial w_2} & \cdots & \frac{\partial e_1(w)}{\partial w_2} \\ \frac{\partial e_2(w)}{\partial w_1} & \frac{\partial e_2(w)}{\partial w_2} & \cdots & \frac{\partial e_2(w)}{\partial w_2} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_N(w)}{\partial w_1} & \frac{\partial e_N(w)}{\partial w_2} & \cdots & \frac{\partial e_N(w)}{\partial w_N} \end{bmatrix} \tag{2.16}$$

Tính toán ma trận Jacobian

Vector sai số có dạng cụ thể là:

$$e^T = [e_{1,1} e_{1,2} ... e_{1,n}; ....; e_{m,1} e_{m,2} ... e_{m,n_Q}] \tag{2.17}$$

Vector tham số có dạng:

$$w^T = [w_1 w_2 ... w_N] \tag{2.18}$$

trong đó:

Chỉ số trên ‘1’ thể hiện cho lớp thứ nhất.

Chỉ số ‘Q’ thể hiện cho lớp ra.

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.

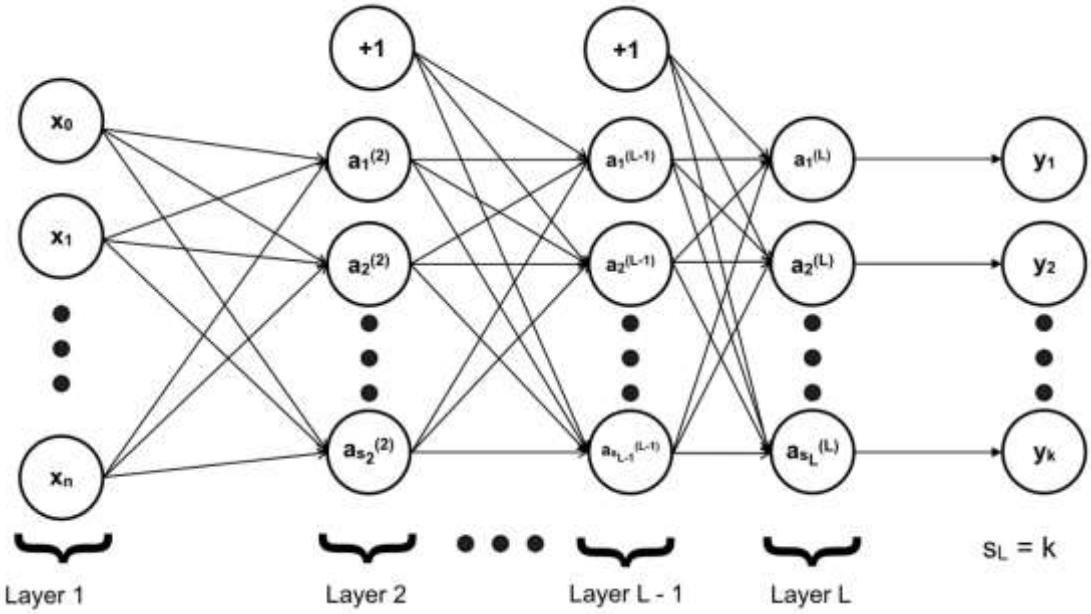
n_Q là số nơron trong lớp ra.

Khi đó ma trận Jacobian được viết lại như sau:

$$J = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w^1_{1,1}} & \frac{\partial e_{1,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,1}}{\partial w^1_{n_1,n_i}} & \frac{\partial e_{1,1}}{\partial b^1_1} & \cdots \\ \frac{\partial e_{1,2}}{\partial w^1_{1,1}} & \frac{\partial e_{1,2}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,2}}{\partial w^1_{n_1,n_i}} & \frac{\partial e_{1,2}}{\partial b^1_1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{1,n_Q}}{\partial w^1_{1,1}} & \frac{\partial e_{1,n_Q}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,n_Q}}{\partial w^1_{n_1,n_i}} & \frac{\partial e_{1,n_Q}}{\partial b^1_1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{p,1}}{\partial w^1_{1,1}} & \frac{\partial e_{p,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{p,1}}{\partial w^1_{n_1,n_i}} & \frac{\partial e_{p,1}}{\partial b^1_1} & \cdots \\ \frac{\partial e_{p,2}}{\partial w^1_{1,1}} & \frac{\partial e_{p,2}}{\partial w^1_{1,1}} & \cdots & \frac{\partial e_{p,2}}{\partial w^1_{n_1,n_i}} & \frac{\partial e_{p,2}}{\partial b^1_1} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{p,n_Q}}{\partial w^1_{1,1}} & \frac{\partial e_{p,n_Q}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{p,n_Q}}{\partial w^1_{n_1,n_i}} & \frac{\partial e_{p,n_Q}}{\partial b^1_1} & \cdots \end{bmatrix} \quad (2.19)$$

Áp dụng hàm kích hoạt cho hàm kết hợp:

$$a^{[L]} = f(z^{[L]}) = f(w^T a^{[L-1]} + b) \quad (2.20)$$



Hình 2.11. Mô hình của một mạng nơron nhiều lớp

Tính đạo hàm của tổng trọng z là:

$$\frac{\partial z^{[L]}}{\partial w} = a^{[L-1]} \tag{2.21}$$

Độ dốc grad của hàm kích hoạt là:

$$grad = \frac{\partial a^{[L]}}{\partial z^{[L]}} = \frac{\partial f(z^{[L]})}{\partial z^{[L]}} \tag{2.22}$$

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

2.1.6. Hiện tượng overfitting

Overfitting là hiện tượng trong học máy khi mô hình máy học quá tinh chỉnh cho dữ liệu huấn luyện, dẫn đến việc nó không thể tổng quát hóa tốt trên dữ liệu mới hoặc dữ liệu kiểm tra. Điều này có nghĩa là mô hình đã học quá mức chi tiết hoặc nhiễu trong dữ liệu huấn luyện, và do đó không thể dự đoán chính xác trên dữ liệu không nhìn thấy trước đó.

Có một số nguyên nhân dẫn đến overfitting:

Mô hình quá phức tạp: Khi mô hình quá phức tạp hoặc có quá nhiều thông số, nó có thể dễ dàng học các chi tiết không cần thiết hoặc nhiễu trong dữ liệu huấn luyện.

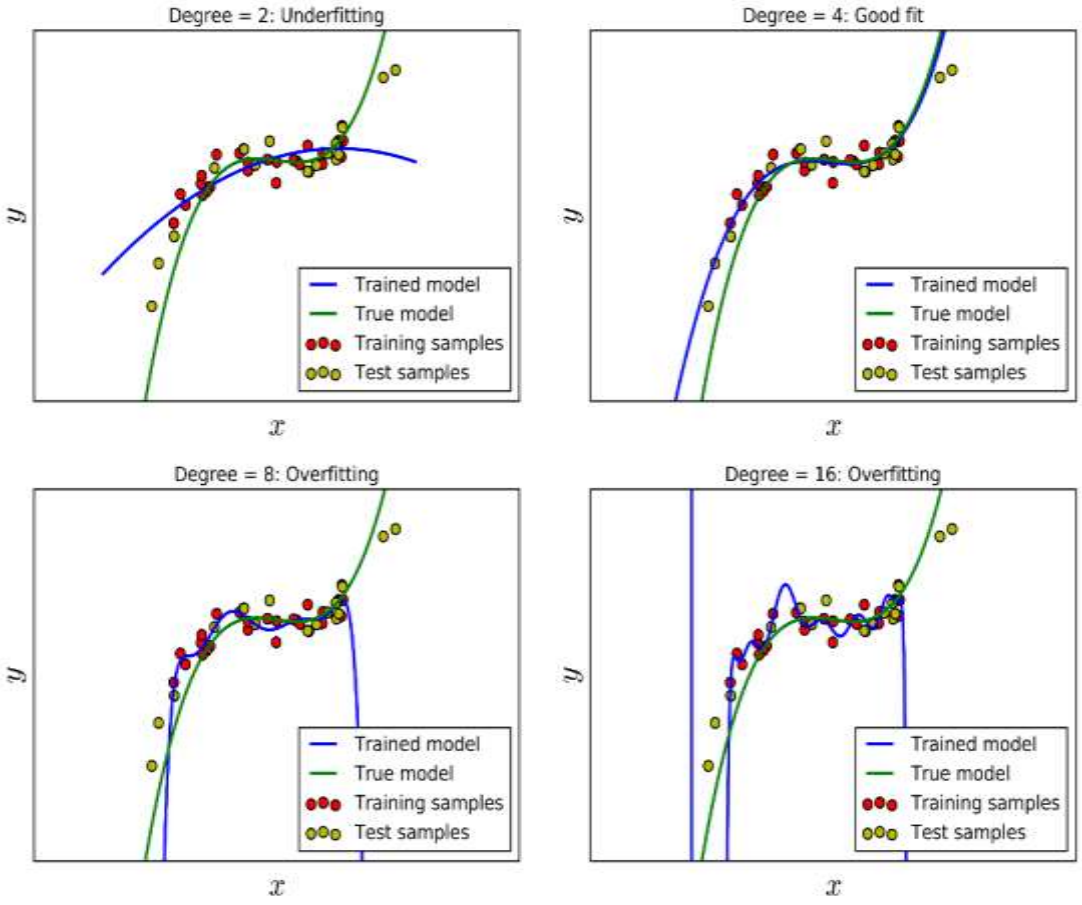
Dữ liệu huấn luyện thiếu đa dạng: Nếu dữ liệu huấn luyện không đủ đa dạng hoặc không đại diện cho dữ liệu thực tế, mô hình có thể học một cách quá mức từ các mẫu có sẵn mà không thể tổng quát hóa được.

Quá trình huấn luyện quá lâu hoặc quá ít: Nếu quá trình huấn luyện diễn ra quá lâu hoặc quá ít vòng lặp, mô hình có thể học quá mức từ dữ liệu huấn luyện mà không thể tổng quát hóa tốt trên dữ liệu mới.

Các biểu hiện của overfitting bao gồm:

Hiệu suất tốt trên dữ liệu huấn luyện nhưng kém trên dữ liệu kiểm tra: Mô hình có thể dự đoán rất chính xác trên dữ liệu huấn luyện nhưng hiệu suất trên dữ liệu kiểm tra lại thấp.

Độ chính xác cao khi dự đoán các điểm dữ liệu có sẵn, nhưng kém khi dự đoán các điểm mới hoặc thực tế: Mô hình có thể không thể tổng quát hóa tốt và không đưa ra dự đoán chính xác cho dữ liệu mới..

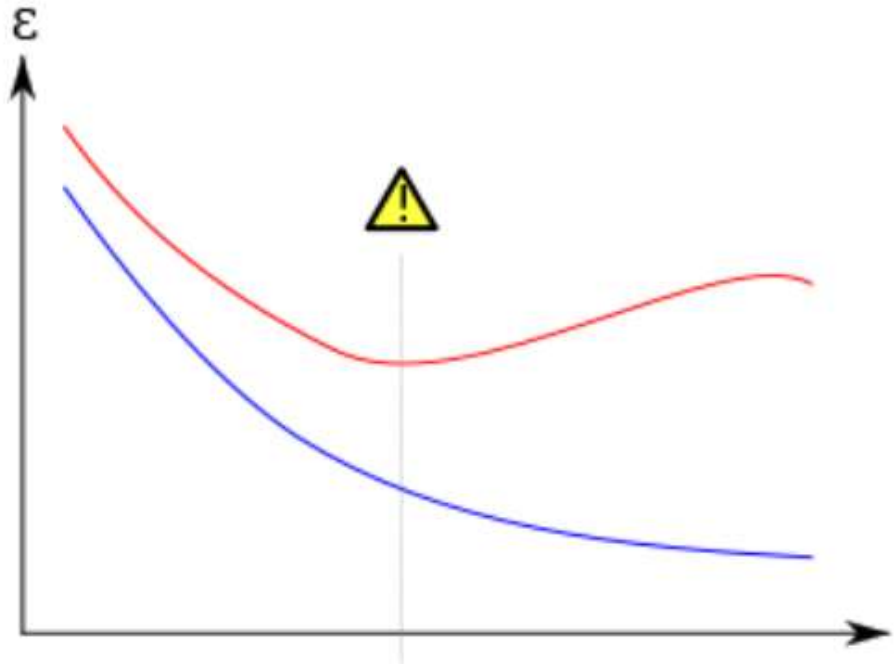


Hình 2.12. Underfitting và overfitting

2.1.7 Một số phương pháp giảm overfitting

a. Early stopping

Trong nhiều bài toán học máy, chúng ta cần sử dụng các thuật toán lặp để tìm ra nghiệm, ví dụ như Gradient Descent. Nhìn chung, hàm mất mát giảm dần khi số vòng lặp tăng lên. Early stopping tức dừng thuật toán trước khi hàm mất mát đạt giá trị quá nhỏ, giúp tránh overfitting.



Hình 2.13. Early stopping

(màu xanh: training error, màu đỏ: validation error)

Một kỹ thuật thường được sử dụng là tách từ training set ra một tập validation set. Sau một (hoặc một số, ví dụ 50) vòng lặp, ta tính cả train error và validation error, đến khi validation error có chiều hướng tăng lên thì dừng lại, và quay lại sử dụng mô hình tương ứng với điểm và validation error đạt giá trị nhỏ.

b. Thêm số hạng vào hàm mất mát

Kỹ thuật phổ biến nhất là thêm vào hàm mất mát một số hạng nữa. Số hạng này thường dùng để đánh giá độ phức tạp của mô hình. Số hạng này càng lớn, thì mô hình càng phức tạp. Hàm mất mát mới này thường được gọi là regularized loss function, thường được định nghĩa như sau:

$$J_{reg}(\theta) = J(\theta) + \lambda R(\theta) \tag{2.23}$$

trong đó, $J(\theta)$ ký hiệu cho hàm mất mát (loss function) và $R(\theta)$ là số hạng regularization, λ thường là một số dương để cân bằng giữa hai đại lượng vế phải. Tuy nhiên tham số λ thường được chọn là một số nhỏ để biểu thức regularization không làm giảm quá nhiều chất lượng của nghiệm.

Một số kỹ thuật regularization thường được sử dụng là:

***l_2 regualarization*:**

Trong kỹ thuật này:

$$R(w) = \|w\|_2^2 \tag{2.24}$$

tức norm 2 của trọng số. Hàm số này có một số đặc điểm đáng lưu ý:

Thứ nhất, $\|w\|_2^2$ là một hàm số rất mượt, tức có đạo hàm tại mọi điểm, đạo hàm của nó đơn giản là w, vì vậy đạo hàm của regularized loss function cũng rất dễ tính, chúng ta hoàn toàn có thể dùng các phương pháp dựa trên gradient để cập nhật nghiệm.

Thứ hai, việc tối thiểu $\|w\|_2^2$ đồng nghĩa với việc khiến cho các giá trị của hệ số w trở nên nhỏ gần với 0. Với mạng nơ-ron, việc các hệ số này nhỏ giúp cho nhiều hệ số trong các ma trận trọng số là nhỏ. Điều này tương ứng với việc số lượng các hidden units hoạt động (khác 0) là nhỏ, giúp tránh overfitting.

Rgualarizers for sparsity:

Trong nhiều trường hợp, ta muốn các hệ số thực sự bằng 0 chứ không phải là nhỏ gần 0 như l_2 regularization đã làm phía trên. Lúc đó, có một regularization khác được sử dụng, đó là l_0 regularization:

$$R(w) = \|w\|_0 \tag{2.26}$$

Norm 0 không phải là một norm thực sự mà là giả norm. Norm 0 của một vector là số các phần tử khác không của vector đó. Khi norm 0 nhỏ, tức rất nhiều phần tử trong vector đó bằng 0, ta nói vector đó là sparse.

Việc giải bài toán tối thiểu norm 0 nhìn chung là khó vì hàm số này không convex, không liên tục. Thay vào đó, norm 1 thường được sử dụng:

$$R(w) = \|w\|_1 = \sum_{i=0}^{d} |w_i| \tag{2.27}$$

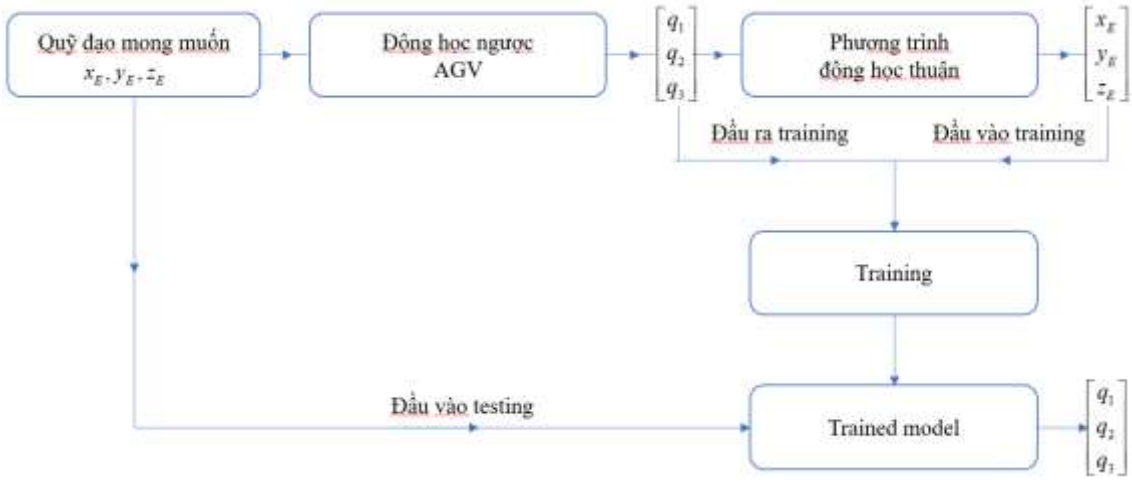
Norm 1 là tổng các trị tuyệt đối của tất cả các phần tử. Người ta đã chứng minh được rằng tối thiểu norm 1 sẽ dẫn tới nghiệm có nhiều phần tử bằng 0. Ngoài ra, vì norm 1 là một norm thực sự (proper norm) nên hàm số này là convex, và hiển nhiên là liên tục, việc giải bài toán này dễ hơn việc giải bài toán tối thiểu norm 0.

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



Hình 2.14. 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 họi 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.

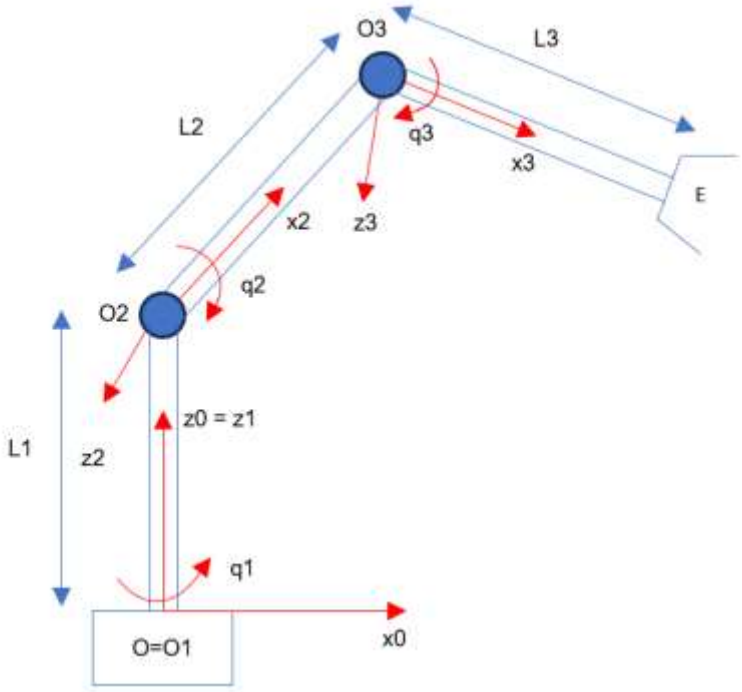
Kết luận chương 2

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:

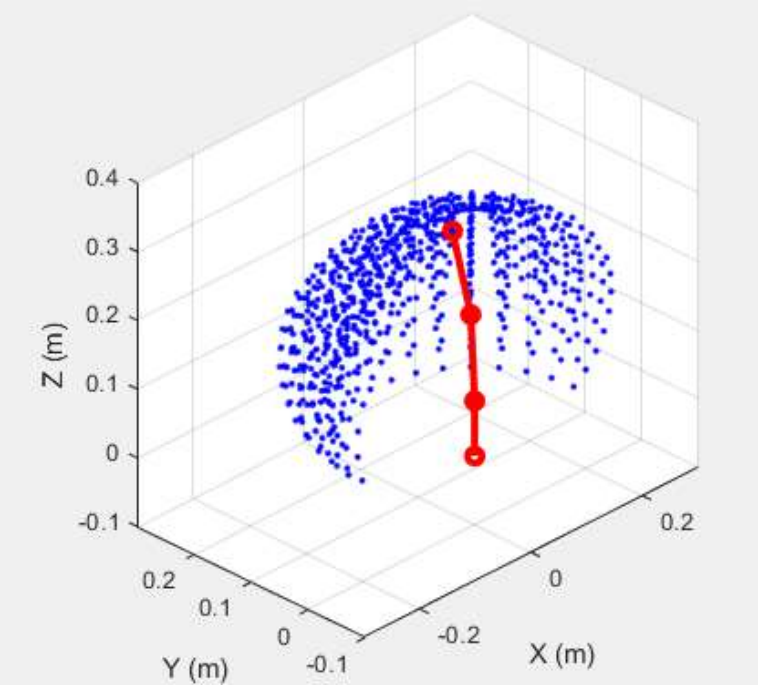
$$\begin{aligned}x_E &= \cos(q_1)(L_3\cos(q_3)\cos(q_2)\text{-}L_3\sin(q_3)\sin(q_2)+L_2\cos(q_2))\\ y_E &= \sin(q_1)(L_3\cos(q_3)\cos(q_2)\text{-}L_3\sin(q_3)\sin(q_2)+L_2\cos(q_2))\\ z_E &= L_3\sin(q_2)\cos(q_3)+L_3\cos(q_2)\sin(q_3)+L_2\sin(q_2)+L_1\end{aligned} \tag{3.6}$$

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

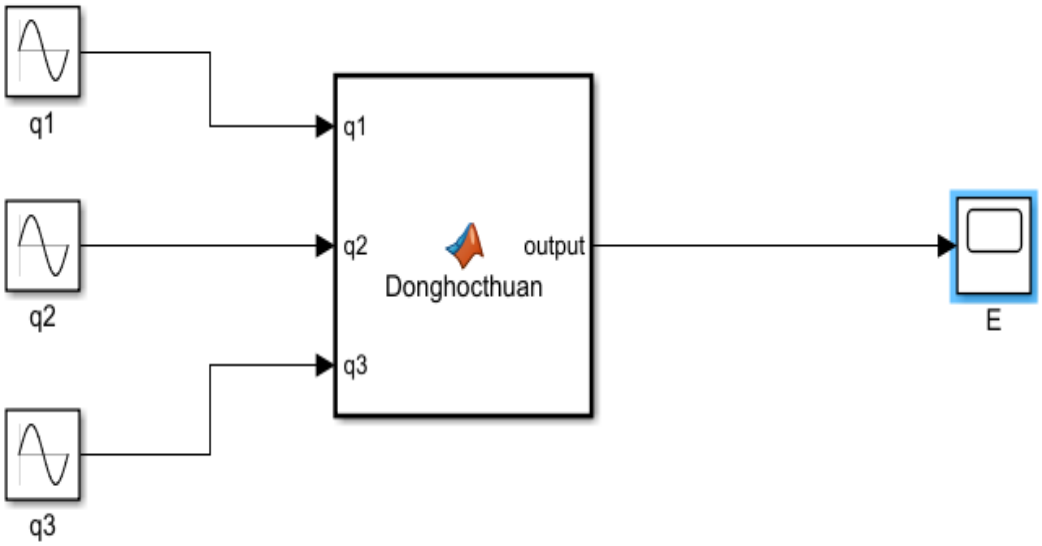
3.2.1. Động học thuận

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

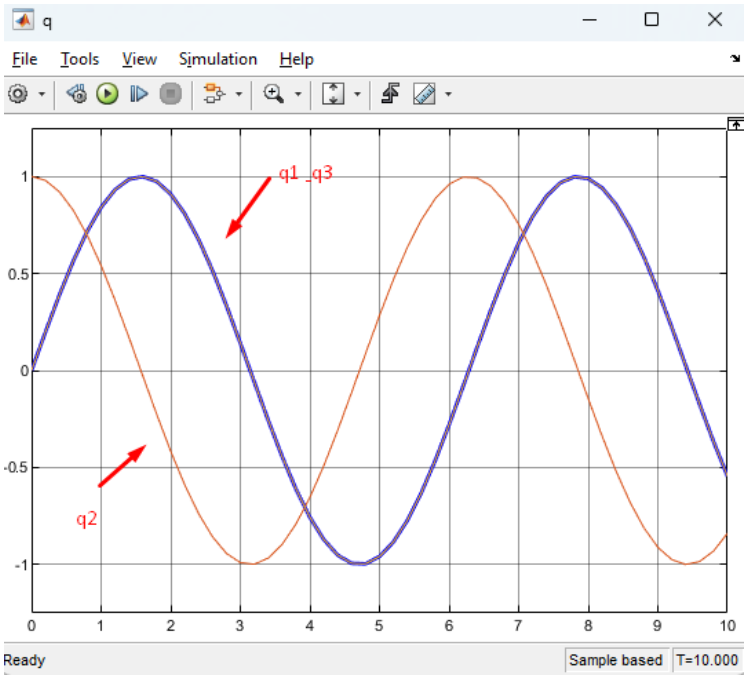
Mô hình giải bài toán động học thuận được mô tả như hình:



Hình 3.3. Simulink cho động học thuận

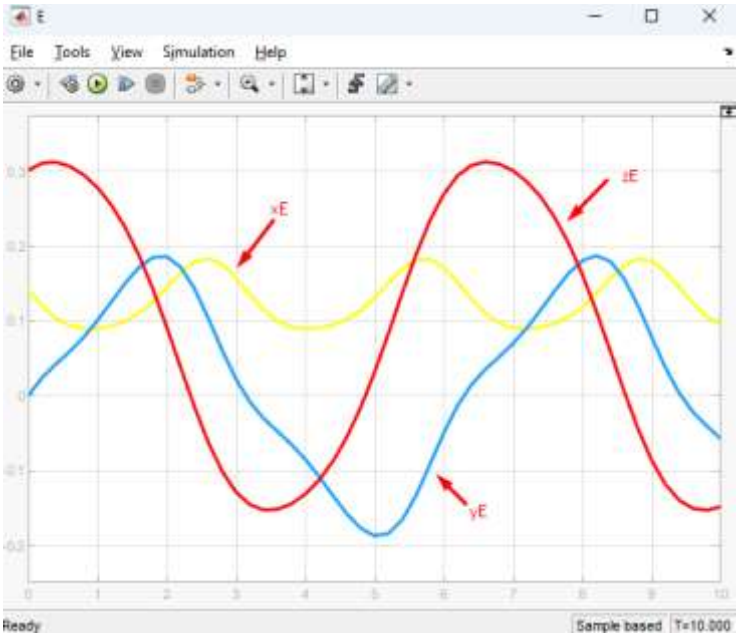
c. Kết quả mô phỏng bài toán động học thuận

Đầu vào:



Hình 3.4. Giá trị q dầu vào

Đầu ra:



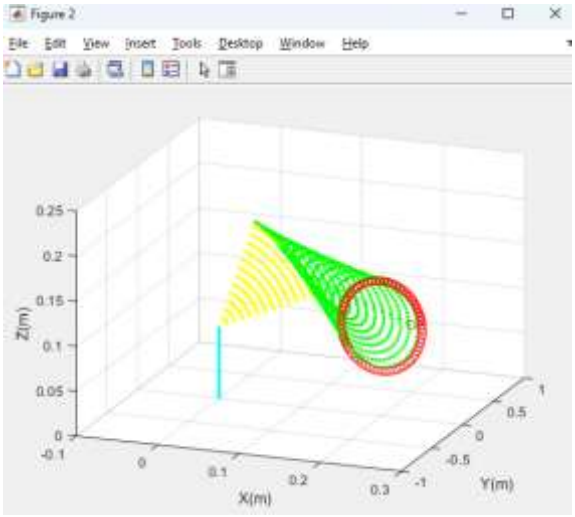
Hình 3.5. Giá trị tọa độ điểm thao tác cuối

3.2.2. Động học ngược

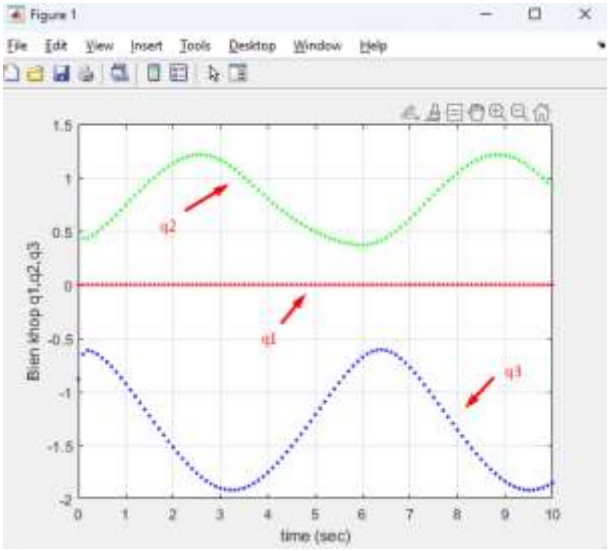
Quỹ đạo điểm tao thác cho trước:

$$\begin{aligned} x_p &= 0.2 + 0.05 * cos(t); \\ y_p &= 0 \\ z_p &= 0.1 + 0.05 * sin(t); \end{aligned} \tag{3.9}$$

Kết quả bài toán động học ngược:



Hình 3.6. Quỹ đạo mong muốn



Hình 3.7. Giá trị q mong muốn

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.

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

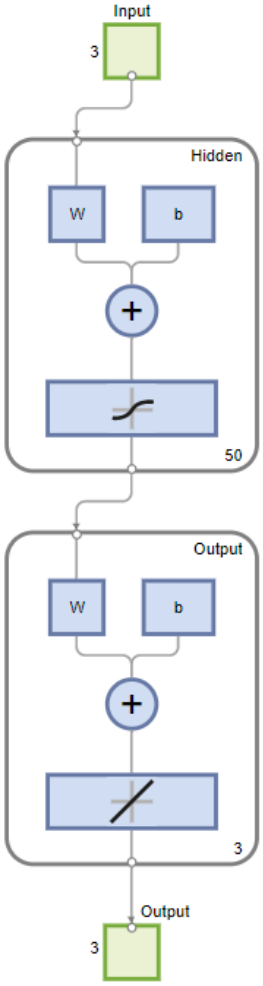
trong đó:

Đầu vào: $p = [p_x, p_y, p_z]^T$: Thể hiện tọa đô 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 3.8. 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

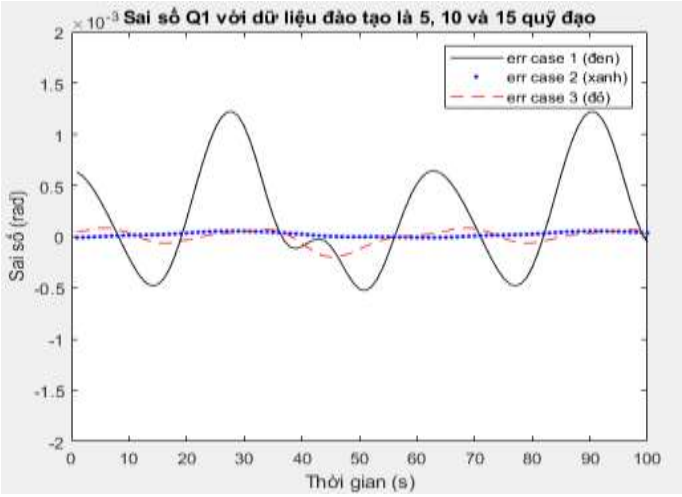
a. Quỹ đạo mong muốn

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

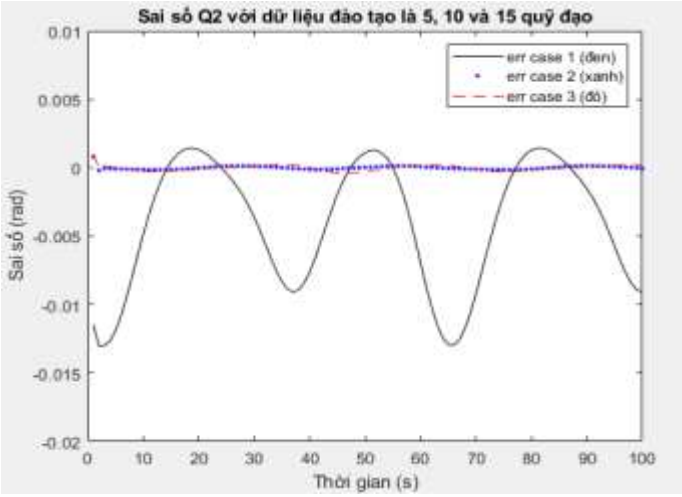
$$\begin{aligned} x_E &= 0.145 + 0.02 * cos(t); \\ y_E &= 0; \\ z_E &= 0.09 + 0.02 * sin(t); \end{aligned} \tag{3.10}$$

b. Kết quả mô phỏng

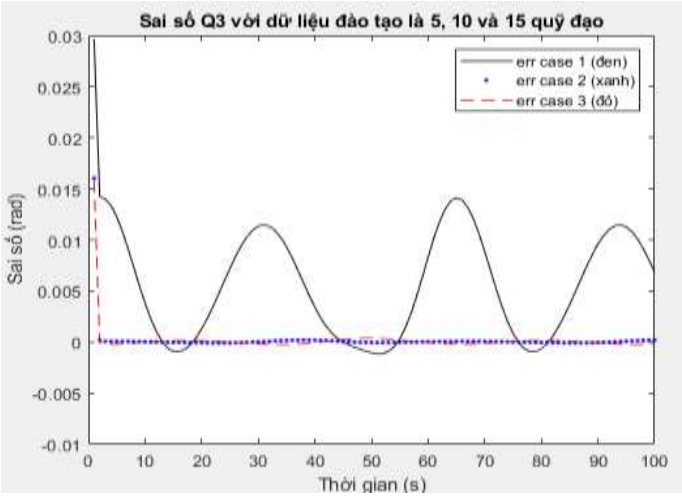
Kết quả bài toán được mô phỏng trên phần mềm matlab với 3 trường hợp đào tạo là dữ liệu đào tạo 5 quỹ đạo, 10 quỹ đạo và 15 quỹ đạo



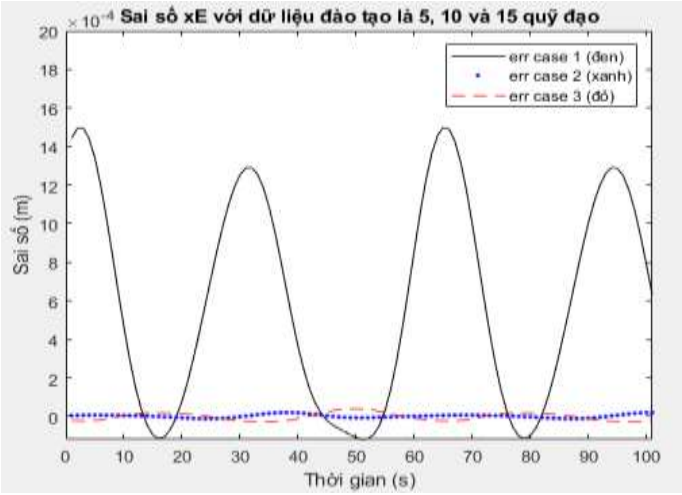
Hình 3.9. Giá trị sai số q_1 trong 3 trường hợp



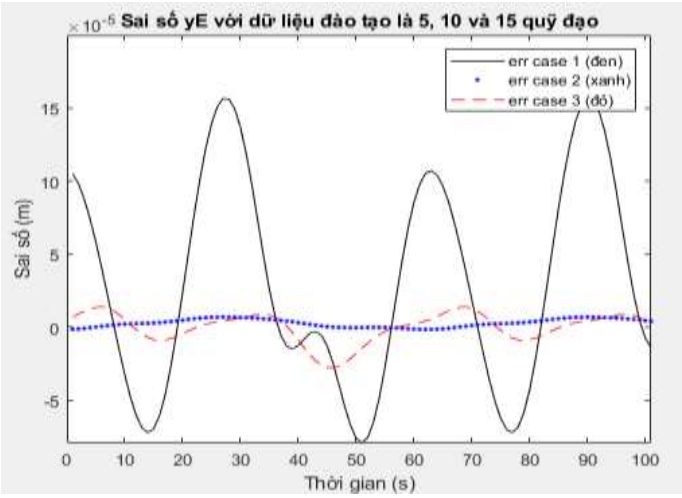
Hình 3.10. Giá trị sai số q_2 trong 3 trường hợp



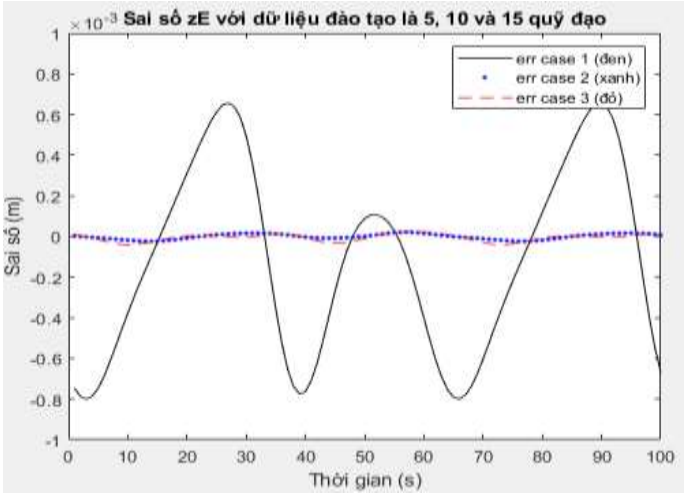
Hình 3.11. Giá trị sai số q_3 trong 3 trường hợp



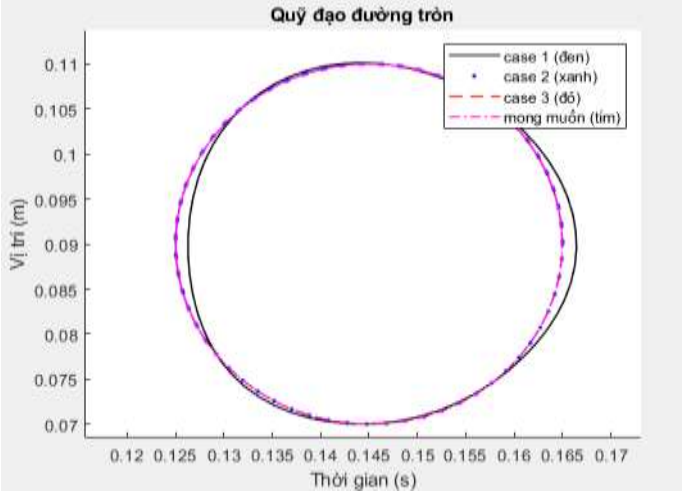
Hình 3.12. Giá trị sai số x_E trong 3 trường hợp



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

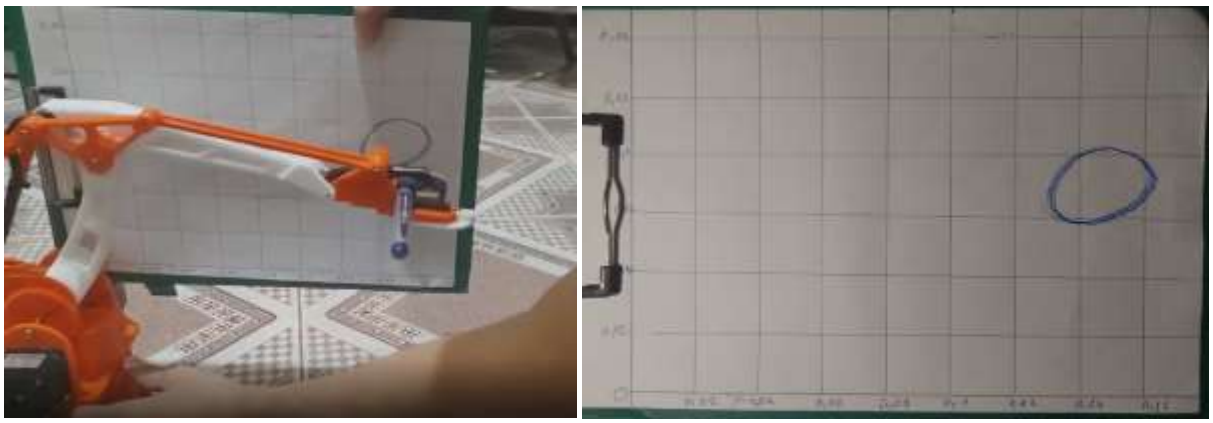


Hình 3.14. Giá trị sai số z_E trong 3 trường hợp



Hình 3.15. Quỹ đạo đường tròn theo 3 trường hợp và mong muốn

c. Kết quả thực nghiệm



Hình 3.16. 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.

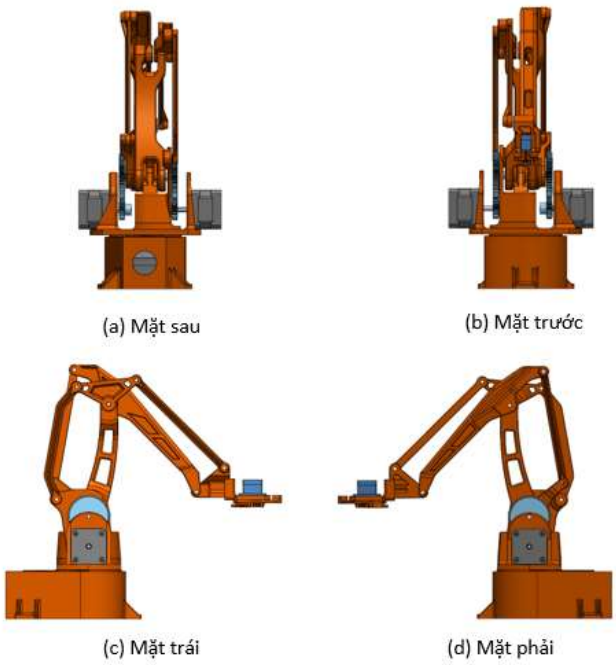
CHƯƠNG 4. ĐIỀU KHIỂN ROBOT SỬ DỤNG GRBL

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.

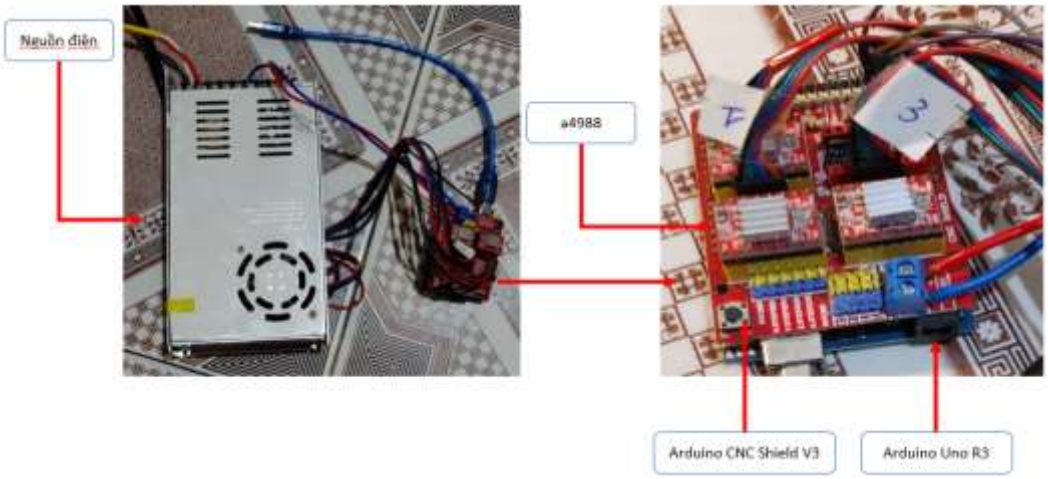
Hình 4.3. Mô hình robot thực tế

	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

Thông số kỹ thuật:

Điện áp hoạt động:5V DC (chỉ được cấp qua cổng USB)

Tần số hoạt động: 16 MHz

Dòng tiêu thụ: khoảng 30mA

Điện áp vào khuyên dung: 7-12V DC

Điện áp vào giới hạn: 6-20V DC

Số chân Digital I/O: 14 (6 chân hardware PWM)

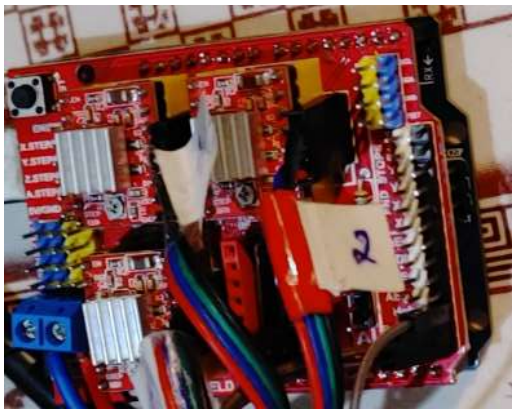
Số chân Analog: 6 (độ phân giải 10bit)

Dòng tối đa trên mỗi chân I/O: 30 mA

Dòng ra tối đa (5V): 500 mA

Dòng ra tối đa (3.3V): 50 mA

Arduino CNC Shield V3



Hình 4.6. Arduino CNC Shield V3

Thông số kỹ thuật:

Hỗ trợ firmware GRBL

Điều khiển tối đa 4 động cơ bước

Tương thích driver: A4988 hoặc DRV8825

Thêm jumper để điều khiển full step, haft step, 1/4, 1/8, 1/16

Công tắc hành trình các trục X, Y, Z, E

Điều khiển đầu khắc CNC, đầu khắc laser

Điều khiển quạt tản nhiệt

Driver

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

Hình 4.7. Driver A4988

Thông số kỹ thuật:

Điện áp cấp tối thiểu: 8 V

Điện áp cấp cực đại: 35 V

Dòng cấp liên tục cho mỗi pha: 1 A (không cần tản nhiệt, làm mát)

Dòng cấp liên tục cho mỗi pha: 2 A (khi có làm mát, tản nhiệt)

Điện áp logic 1 tối thiểu: 3 V

Điện áp logic 1 tối đa: 5.5 V

Độ phân giải: full, 1/2, 1/4, 1/8, và 1/16

Động cơ bước

Hình 4.8. Động cơ bước nema 17

Thông số kỹ thuật:

Góc quay mối bước: 1.8 ° (200 bước / vòng quay)

Dòng điện: 1.7A

Mômen Xoắn giữ: 40N.cm

Mômen xoắn: 2.2N.cm

Rotor quán tính: 54g.cm2

Độ chính xác góc bước: ± 5% (bước đầy đủ, không tải)

Khối lượng động cơ: 280g

Chiều dài thân: 40 mm

Kích thước khung: 42 x 42mm

Đường kính trục: 5mm

Chiều dài trục: 23mm

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.

Thông số kỹ thuật:

Điện áp đầu vào: 110/ 220V AC.

Điện áp đầu ra: 12V DC.

Dòng điện: 20A.

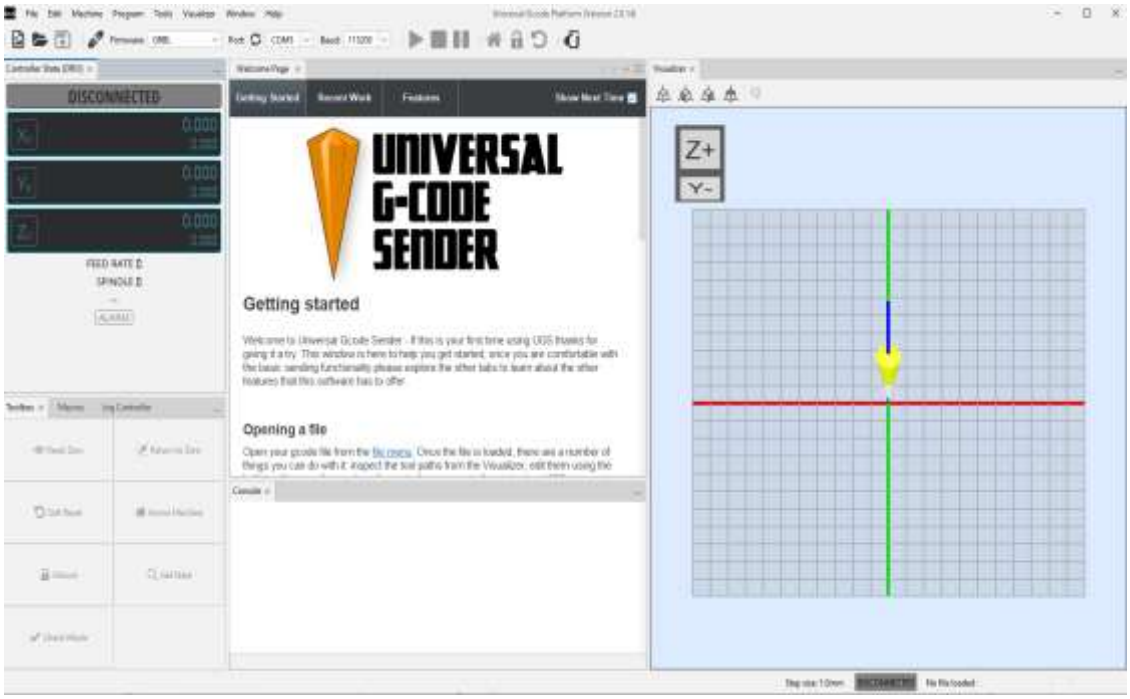
Nhiệt độ làm việc: Từ 0 – 60 độ C.

Kích thước: 198 x 98 x 42mm

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)

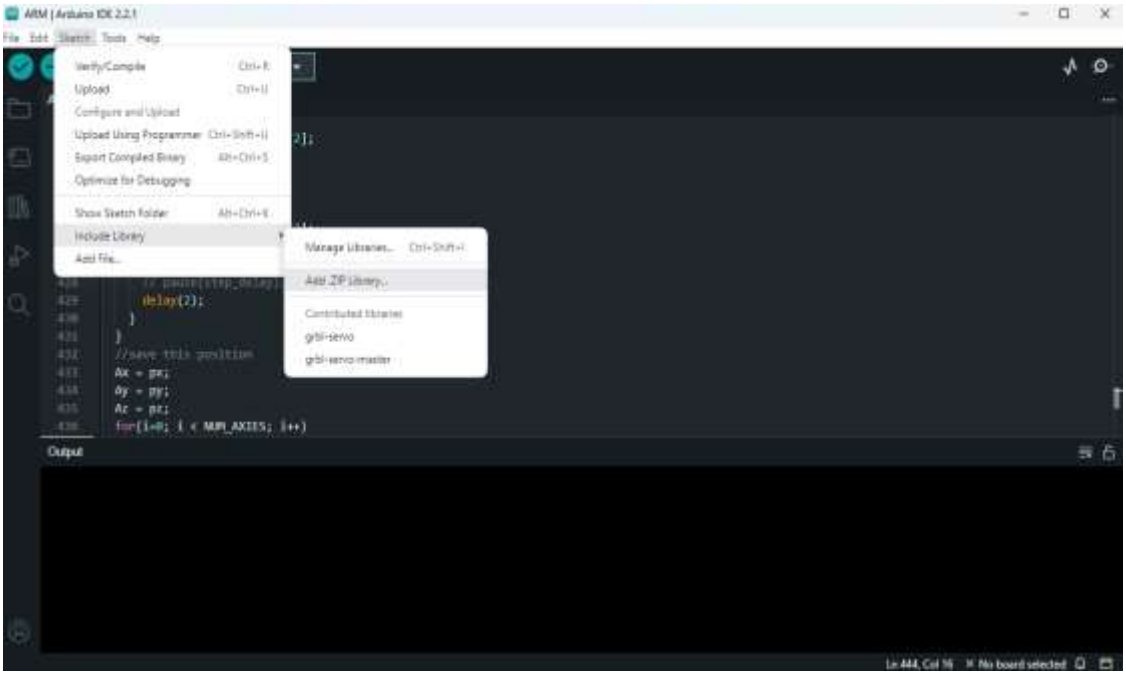
Bước 2: Kết nối các thành phần như sau:

- 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.

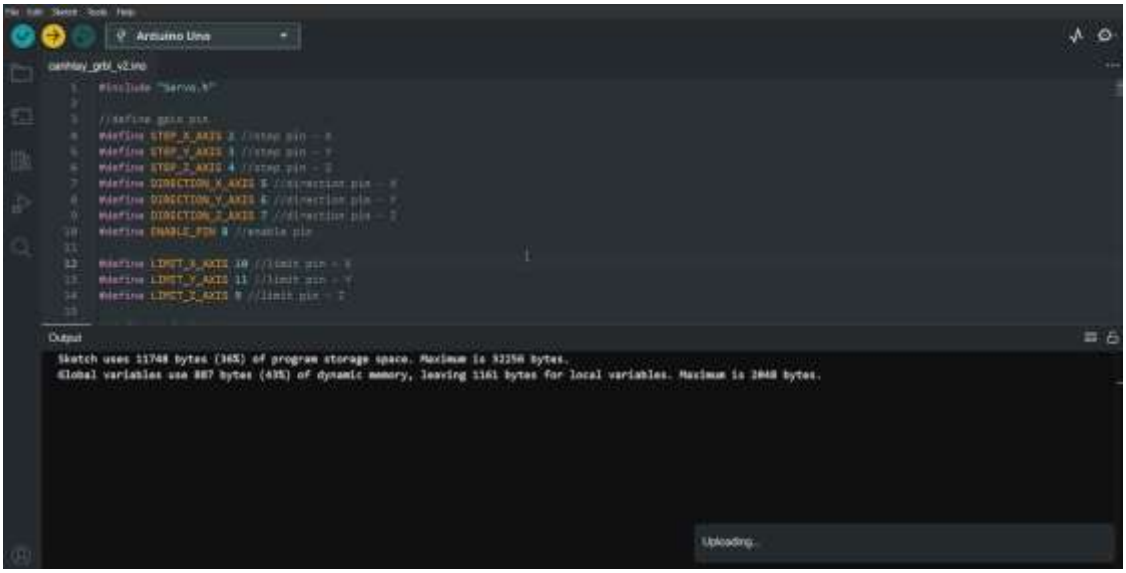
Bước 3: Cấu hình trên Arduino

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

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

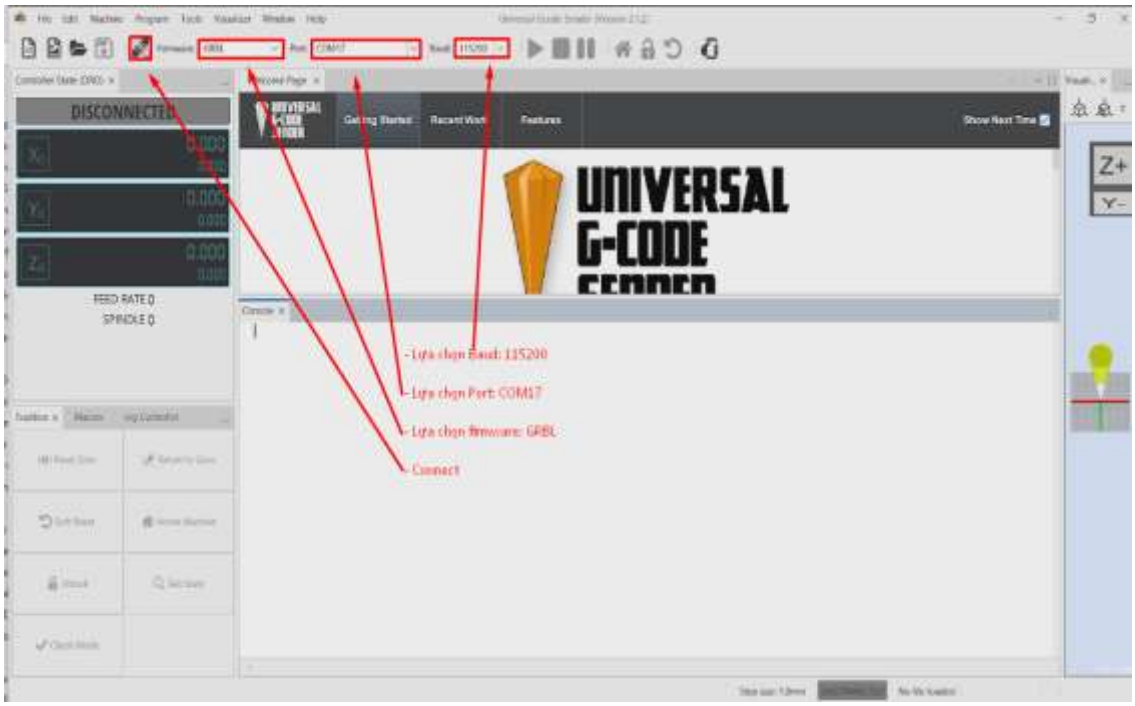


Hình 4.11. Thêm file zip của GRBL vừa download

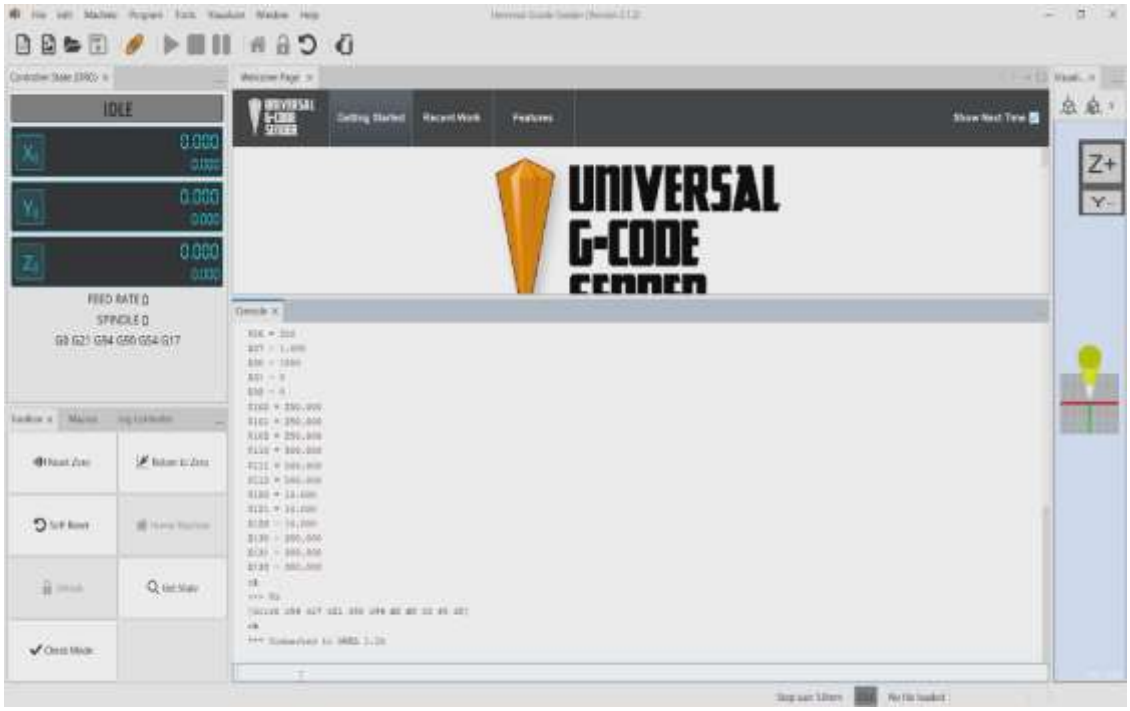


Hình 4.12. Lựa chọn board Arduino sau đó chạy code

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 4-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 sau 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 khác 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

[1]. Phạm Hữu Đức Dục, *Mạng nơron & ứng dụng trong điều khiển tự động, nhà xuất bản Khoa Học và Kỹ Thuật*, pp. 292, 2009.

[2]. Nguyễn Đình Thúc, *Trí tuệ nhân tạo - Mạng nơron phương pháp và ứng dụng, nhà xuất bản Giáo dục*, pp. 233, 2000.

[3]. Nguyễn Việt Hùng, *Ứng dụng mạng nơron để giải bài toán động học ngược tay máy*, pp. 107, 2014.

[4]. Raşit Kokör & Cemil Öz, Tarik Çakar, Hüseyin Ekiz, *A Study of Neural Network Based Inverse Kinematics Solution for a Planar Three Joint Robot*, Sakarya University, 2003.

[5]. Alavandar S. and Nigam M. J. *Neuro-Fuzzy based Approach for Inverse Kinematics Solution of Industrial Robot Manipulators*, Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. III, pp. 224-234, 2008.

[6]. Jolly Shah and S.S.Rattan and B.C.Nakra, *Kinematic Analysis of a Planer Robot Using Artificial Neural Network*, International Journal of Robotics and Automation (IJRA) Vol.1, pp. 145 – 151, 2012.

[7]. Raşit Kokör, *A neuro-genetic approach to the inverse kinematics solution of robotic manipulators*,Scientific Research and Essays Vol. 6, pp. 2784-2794, 2011.

[8]. Panchanand Jha, Novel, *Artificial Neural Network Application for Prediction of Inverse Kinematics of Manipulator*, M.S. Thesis, Department of Mechanical Engineering National Institute of Technology Rourkela, India, 2009.

[9]. Sandi Baressi Segota, Nikola And ̄elic ́, Vedran Mrzljak, Ivan Lorencin, Ivan Kuric and Zlatan Car, *Utilization of multilayer perceptron for determining the inverse kinematics of an industrial robotic manipulator*, International Journal of Advanced Robotic Systems, pp. 1-11, 2021.

[10]. Shital S, N. Ramesh Babu, *Comparison of RBF and MLP neural networks to solve inverse kinematic problem for 6R serial robot by a fusion approach*, Elsevier, pp. 1083-1092, 2010.

[11]. Aghajarian, Kourosh Kiani, *Inverse Kinematics Solution of PUMA 560 Robot Arm Using ANFIS*, Semnan University, pp. 574-587, 2011.

[12]. Hasan, Azlan, Hayder, *Neural Networks Based Inverse Kinematics Solution for Serial Robot Manipulators Passing Through Singularities*, University Technology MARA, pp. 460-477, 2011.

[13]. H. Toshani, M. Farrokhi, *Kinematic Control of a Seven DOF Robot Manipulator with Joint Limits and Obstacle Avoidance Using Neural Networks*, ICCIA, pp. 976-981, 2011.
[14]. Wesam Mohammed Jasim, *Solution of Inverse Kinematics for SCARA Manipulator Using Adaptive Neuro-Fuzzy Network*, University of Anbar, pp. 59-66, 2011.
[15]. Y. Feng, W. Yao-nan, Y. Yi-min, *Inverse Kinematics Solution for Robot Manipulator based on Neural Network under Joint Subspace*, pp. 460-472, 2012.
[16]. Koker, *A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators*, Engineering with Computers, 2013.
[17]. Mouli, Jyothi, Raju, *Design and Implementation of Robot Arm Control Using LabVIEW and ARM Controller*, IOSR-JEEE, pp. 80-84, 2013.
[18]. Adrian-Vasile Duka, *Neural network based inverse kinematics solution for trajectory tracking of a robotic arm*, INTER-ENG 2013, pp.21-27, 2014.
[19]. 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.
[20]. Pajaziti, Cana, *Robotic Arm Control with Neural Networks Using Genetic Algorithm Optimization Approach*, Open Science Index, Mechanical and Mechatronics Engineering Vol:8, pp. 1431-1435, 2014.