Hướng dẫn đồ án thiết kế hệ thống điều khiển robot scara

Đồ án cơ điện tử (Trường Đại học Bách khoa Hà Nội)

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CƠ KHÍ

Bộ môn Cơ điện tử

SME.EDU - Mẫu 6.a	rev1

Học kỳ:

Năm học: 20 - 20

ĐỒ ÁN MÔN HỌC: THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN **Mã HP: ME4336**

Thời gian thực hiện: 15 tuần; Mã đề: VCK-…

Ngày giao nhiệm vụ: …/…/20…; Ngày hoàn thành: …/…/20..

Họ và tên sv:…… MSSV: …… Mã lớp: …… Chữ ký sv: …….

Ngày …/…/20…	Ngày …/…/20…	Ngày …/…/20…
ĐƠN VỊ CHUYÊN MÔN	**NGƯỜI RA ĐỀ**	**CB Hướng dẫn**
(ký, ghi rõ họ tên)	*(ký, ghi rõ họ tên)*	*(ký, ghi rõ họ tên)*

I. Nhiệm vụ thiết kế: Thiết kế hệ thống điều khiển cho robot SCARA

II. Số liệu cho trước:

1. Tải trọng … kg.
2. Tầm với … m.
3. Độ chính xác lặp: (x, y) = … mm, (z) = … mm.
4. Vận tốc cực đại khâu tác động cuối ….
5. Gia tốc cực đại khâu tác động cuối …..

III. Nội dung thực hiện:

1. Phân tích nguyên lý và thông số kỹ thuật

- Tổng quan về hệ thống
- Nguyên lý hoạt động
- Xác định các thành phần của hệ thống điều khiển

2. Thiết kế hệ thống điều khiển

- Mô hình hóa và xác định hàm truyền

- Đánh giá tính ổn định của hệ thống

- Mô phỏng và phân tích, đánh giá các chỉ tiêu kỹ thuật của hệ thống

- Lựa chọn các thiết bị cho hệ thống điều khiển: cảm biến, thiết bị điều khiển, cơ cấu chấp hành

- Thiết kế sơ đồ mạch điện và mạch điều khiển (1 bản A0)

3. Lập trình điều khiển

- Lập trình điều khiển robot (1 chương trình điều khiển)

- Lập trình mô phỏng chuyển động (1 chương trình mô phỏng trên Simmechanics)

HƯỚNG DẪN THỰC HIỆN ĐỒ ÁN THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN CHO ROBOT SCARA

Nhóm biên soạn: TS. Mạc Thị Thoa, TS. Nguyễn Thành Hùng

Bộ Môn: Cơ Điện Tử, Viện Cơ Khí, Đại học Bách Khoa Hà Nội

(Tài liệu lưu hành nội bộ)

Chương 1: Tổng quan về robot SCARA

Robot SCARA là robot có khớp nối ngang 4 bậc tự do. Những robot này thường được sử dụng để lắp ráp theo phương thẳng đứng và các hoạt động khác trong các mặt phẳng song song. Các robot SCARA thương mại bao gồm robot Adept One, robot IBM 7545, robot Intellingex 440 và robot Rhino SCARA.

Tài liệu này trình bày chi tiết về thiết kế và triển khai robot SCARA. Tuy nhiên, trước khi đi sâu vào chi tiết kỹ thuật, sẽ rất đáng để xem xét một số vấn đề sơ bộ.

1.1 Các mục tiêu thiết kế

Mục tiêu của đồ án này là thiết kế robot SCARA công nghiệp với sự cân bằng tối ưu về kinh tế và hiệu suất.

Sử dụng động cơ bước hoặc động cơ servo và các mạch điều khiển có sẵn.

Sử dụng phần mềm có sẵn để điều khiển robot trên máy tính cá nhân. Phần mềm này sử dụng G-code để điều khiển robot.

1.2 Các yêu cầu về hiệu suất của robot công nghiệp

Robot được thiết kế để trở thành máy móc chính xác và linh hoạt cao. Robot nói chung và robot SCARA nói riêng, được sử dụng làm vật thay thế cho người vận hành. Điều này có thể vì nhiều lý do, nhưng một tính năng quan trọng của việc sử dụng robot là chúng hầu như luôn làm công việc tốt hơn so với người vận hành. Để đạt được mục tiêu quan trọng này, robot cần tuân thủ các tiêu chuẩn hiệu suất tối thiểu nhất định. Để đưa ra ý tưởng, một số thông số kỹ thuật của robot Adept One XL SCARA được mô tả như sau.

Tầm với 800 mm

Khoảng làm việc của các khớp

- Joint 1: $\pm 150^0$
- Joint 2: $\pm 140^0$

- Joint 3: 203mm
- Joint 4: $\pm 270^0$

Tốc độ tối đa của các khớp

- Joint 1: 650^0/giây
- Joint 2: 920^0/giây
- Joint 3: 1200 mm/giây
- Joint 4: 3300^0/giây

Độ chính xác lặp

- (x, y): ±0,025 mm
- (z): ±0,038 mm
- Theta: $\pm 0{,}05^0$

Tải trọng lớn nhất 12 kg.

1.3 Hệ thống cơ khí

SCARA là một cấu hình tiêu chuẩn giữa các robot. Ở giai đoạn thiết kế sơ bộ, chúng ta có những điểm sau đây:

- Robot SCARA 3 bậc tự do.
- Tải trọng … kg.
- Tầm với … m.
- Độ chính xác lặp: (x, y) = … mm, (z) = … mm.
- Vận tốc cực đại khâu tác động cuối ….
- Gia tốc cực đại khâu tác động cuối …..
- Sử dụng bộ truyền vít me – đai ốc cho khâu tịnh tiến.

1.4 Hệ thống điện tử

Các thiết bị điện tử liên quan phụ thuộc hoàn toàn vào loại động cơ được sử dụng. Để đạt độ chính xác cao, chúng ta có thể sử dụng động cơ bước hoặc động cơ servo trong bộ truyền động chung. Do đó, phần chính của hệ thống điện tử liên quan đến việc thiết kế các mạch điều khiển động cơ bước (servo) thích hợp. Ngoài phần mềm điều khiển được sử dụng, đây là yếu tố quan trọng nhất, có thể ảnh hưởng lớn đến hiệu suất của robot. Ở đây, chúng ta chỉ tập trung vào các trình điều khiển động cơ bước (servo).

Một số lựa chọn ban đầu cần phải chú ý là:

- Loại động cơ: Đơn cực hoặc lưỡng cực.
- Nguyên lý ổ đĩa: L/R hoặc PWM
- Chế độ hoạt động: Nửa bước hoặc toàn bước.

- Đặc trưng mô-men xoắn - tốc độ.

1.5 Phần mềm và điều khiển

Phần mềm điều khiển cần thực hiện được các nhiệm vụ sau:

- Đường dẫn liên tục trong không gian.
- Chuyển động xen kẽ (Joint-interpolated motion).
- Nội suy cung tròn.
- Nội suy tuyến tính.
- Phối hợp chuyển động tuyến tính: tất cả các trục khởi động và dừng các bước chuyển động cùng một thời gian.

Chương 2: Phân tích nguyên lý và thông số kỹ thuật

Sinh viên cần tìm hiểu và trình bày các nội dung như sau:

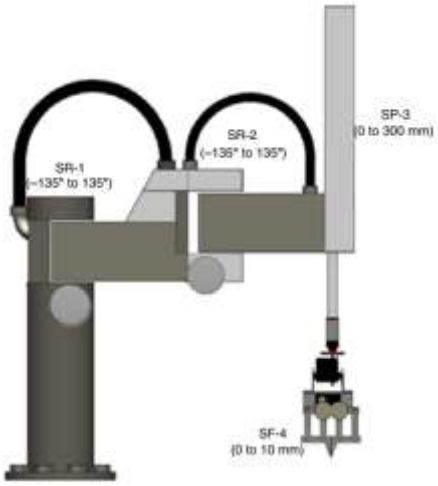
2.1. Nguyên lý hoạt động

- Phân tích nguyên lý hoạt động của hệ thống điều khiển robot SCARA

2.2. Xác định các thành phần của hệ thống điều khiển

- Động cơ: phân tích về động cơ được lựa chọn từ đồ án thiết kế hệ thống cơ khí (có thể chọn lại động cơ để đảm bảo tính năng điều khiển được tốt hơn).

- Cảm biến: cảm biến góc quay (ví dụ: SR-1, SR-2), cảm biến dịch chuyển dài (ví dụ: SP-3), cảm biến lực (ví dụ: SF-4), …

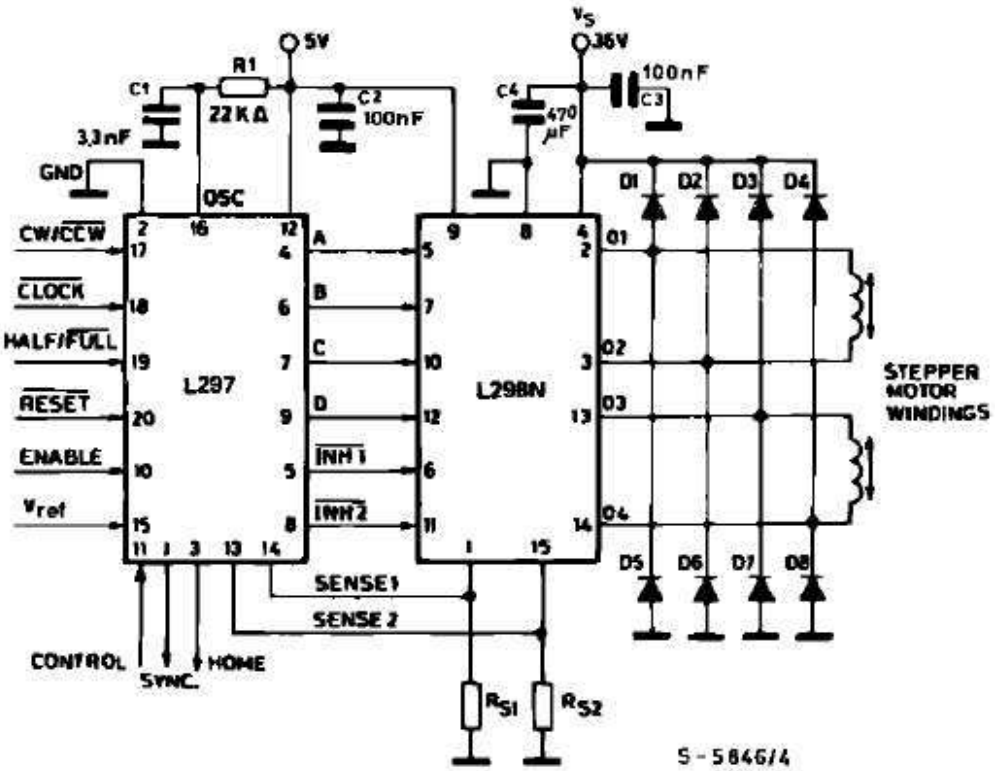


Hình 2.1 Sơ đồ bố trí cảm biến trên robot SCARA [2]

- Bộ nguồn

- Mạch dẫn động động cơ (ví dụ L298)

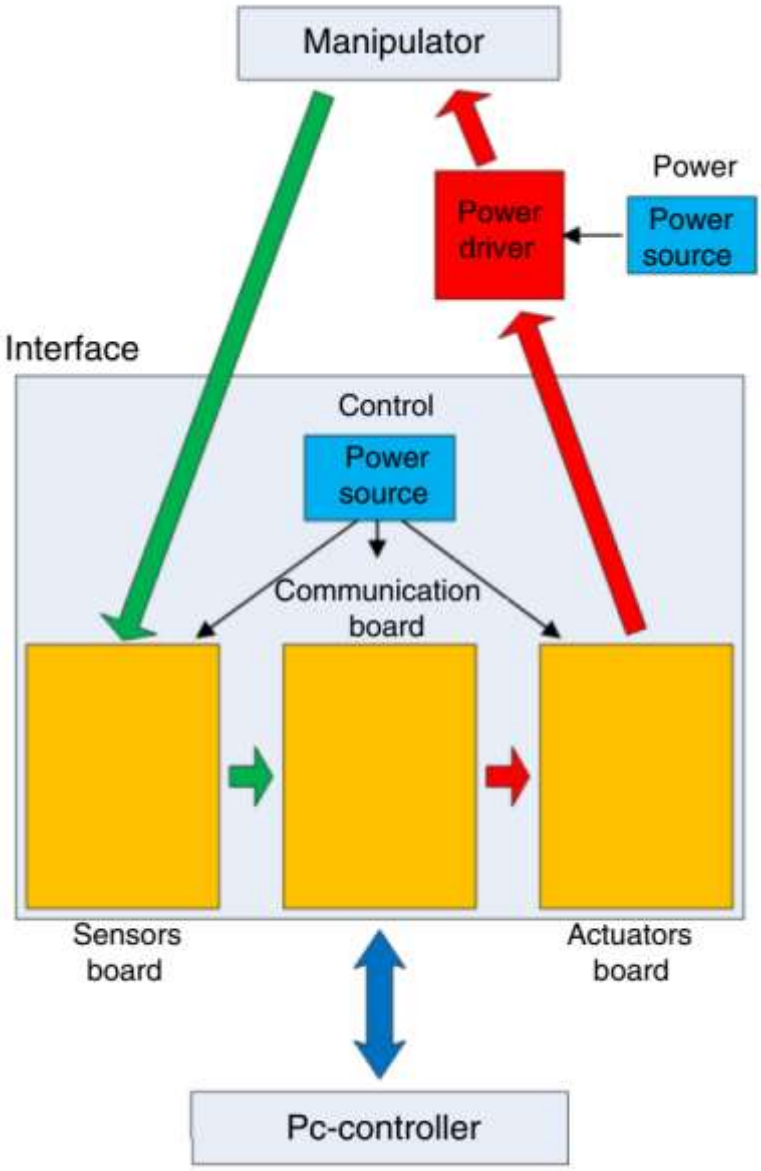
- Bộ điều khiển động cơ (ví dụ L297)



$R_{S1} = R_{S2} = 0.5\ \Omega$

D1 to D8 = 2 A Fast diodes { $V_F \leq 1.2\ V$ @ $I = 2\ A$; $trr \leq 200\ ns$

Hình 2.2 Mạch điều khiển động cơ [1]



Hình 2.3 Sơ đồ tổng thể mạch điều khiển robot SCARA [2]

- Phần mềm điều khiển robot: có thể sử dụng phần mềm có sẵn như LinuxCNC [7] hoặc viết chương trình trên phần mềm MatLab/Simulink [2].

Chương 3. Thiết kế hệ thống điều khiển

3.2 Động lực học robot SCARA

3.2.1 Động lực học thuận

Để tính toán động lực học Robot, ta sẽ đi thiết lập phương trình vi phân chuyển động của robot. Phương trình vi phân chuyển động của Robot được xây dựng theo phương trình Lagrange loại II có dạng tổng quát như sau:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right)^T - \left(\frac{\partial T}{\partial q_i}\right)^T = -\left(\frac{\partial \Pi}{\partial q_i}\right)^T + Q_i^* \tag{3.1}$$

Với: T- động năng của Robot.

$\prod$- thế năng của Robot.

Q^*- véctơ lực suy rộng không thế.

n- số bậc tự do

Phương trình vi phân chuyển động của robot có dạng:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q,\dot{q}) = \tau \tag{3.2}$$

Với $\boldsymbol{M}$ là ma trận khối lượng, $\boldsymbol{C}$ là ma trận Coriolis, $\boldsymbol{G}$ là trọng lực, và $\tau = [\tau_1 \quad \tau_2 \quad \tau_3]^t$ là lực điều khiển tại các khớp.

3.2.2 Động lực học ngược

Thế các biểu thức $\boldsymbol{q}(t),\ \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t)$ vào phương trình (3.2) để tìm các lực và ngẫu lực dẫn động tại các khâu 1, 2 và 3 của robot.

Chú ý: ngoài việc sử dụng phương trình Lagrange loại II, sinh viên có thể sử dụng phương pháp lặp Newton – Euler để giải bài toán động lực học thuận và động lực học ngược.

3.3 Phương pháp điều khiển

Giả sử đã biết trước quỹ đạo chuyển động của các khớp là $\boldsymbol{q}_d(\cdot)$. Sinh viên có thể sử dụng một trong hai phương pháp điều khiển sau để thiết kế bộ điều khiển cho robot SCARA [6].

3.3.1 Điều khiển mô men

Hãy xem xét sự tinh chỉnh sau đây của luật điều khiển vòng hở: với vị trí và vận tốc hiện tại của người thao tác, hủy bỏ tất cả các phi tuyến và áp dụng chính xác mô-men xoắn cần thiết để vượt qua quán tính của bộ truyền động,

$$\tau = M(q)\ddot{q}_d + C(q,\dot{q})\dot{q} + G(q,\dot{q}) \tag{3.3}$$

Thay thế luật điều khiển (3.3) vào phương trình chuyển động của robot (3.2) ta được:

$$M(q)\ddot{q} = M(q)\ddot{q}_d$$

Do $M(q)$ luôn nhận giá trị dương, ta có:

$$\ddot{q} = \ddot{q}_d \tag{3.4}$$

Do đó, nếu vị trí và vận tốc ban đầu của tay máy khớp vị trí và vận tốc mong muốn, tay máy sẽ đi theo quỹ đạo mong muốn. Như trước đây, luật kiểm soát này sẽ không hiệu chuẩn cho bất kỳ lỗi điều kiện ban đầu nào.

Các thuộc tính bám của luật điều khiển có thể được cải thiện bằng cách thêm phản hồi trạng thái. Độ tuyến tính của phương trình (3.4) gợi ý luật điều khiển sau:

$$\tau = M(q)\left(\ddot{q}_d - K_v\dot{e} - K_p e\right) + C(q,\dot{q})\dot{q} + G(q,\dot{q}) \tag{3.5}$$

Trong đó $e = q - q_d$, và K_v và K_p là các ma trận khuếch đại hằng số. Thay thế vào phương trình (3.2), động lực sai số (error dynamics) có thể được viết lại như sau:

$$M(q)\left(\ddot{e} + K_v\dot{e} + K_p e\right) = 0$$

Do $M(q)$ luôn nhận giá trị dương, ta có:

$$\ddot{e} + K_v\dot{e} + K_p e = 0 \tag{3.6}$$

Đây là một phương trình vi phân tuyến tính xác định lỗi giữa các quỹ đạo thực tế và mong muốn. Phương trình (3.5) được gọi là luật điều khiển mô-men.

Luật điều khiển mô-men bao gồm hai thành phần. Chúng ta có thể viết phương trình (3.5) dưới dạng:

$$\tau = \underbrace{M(q)\ddot{q}_d + C(q,\dot{q})\dot{q} + G(q,\dot{q})}_{\tau_{ff}} + \underbrace{M(q)\left(-K_v\dot{e} - K_p e\right)}_{\tau_{fb}} \tag{3.7}$$

τ_{ff} là thành phần tiếp thuận (feedforward). Nó cung cấp số lượng mô-men xoắn cần thiết để dẫn động hệ thống dọc theo quỹ đạo danh nghĩa của nó. τ_{fb} là thành phần phản hồi. Nó cung cấp các điểm chỉnh sửa để giảm bất kỳ lỗi nào trong quỹ đạo của trình thao tác.

Để hệ thống ổn định, cần chọn K_v và K_p là các ma trận chéo (xác định dương và đối xứng).

Ưu điểm của luật điều khiển mô-men là nó chuyển đổi một hệ động lực phi tuyến thành một hệ tuyến tính, cho phép sử dụng bất kỳ công cụ tổng hợp điều khiển tuyến tính nào. Kết quả thử nghiệm cho thấy bộ điều khiển mô-men có các đặc tính hiệu suất rất tốt và nó đang ngày càng trở nên phổ biến.

3.3.2 Điều khiển PD

Luật điều khiển PD có dạng như sau:

$$\tau = -K_v\dot{e} - K_p e \tag{3.8}$$

Trong đó K_v và K_p là các ma trận xác định dương và $e = q - q_d$.

Vì luật điều khiển này không có thành phần tiếp thuận (feedforward), nên nó không bao giờ có thể đạt được theo dõi chính xác cho các quỹ đạo không tầm thường. Một sửa đổi phổ biến là thêm thành phần tích phân để loại bỏ các lỗi trạng thái ổn định (steady-state errors).

Vì chúng ta chủ yếu quan tâm đến việc bám quỹ đạo, chúng ta xem xét một phiên bản sửa đổi của luật điều khiển PD:

$$\tau = M(q)\ddot{q_d} + C(q,\dot{q})\dot{q_d} + G(q,\dot{q}) - K_v\dot{e} - K_p e \tag{3.9}$$

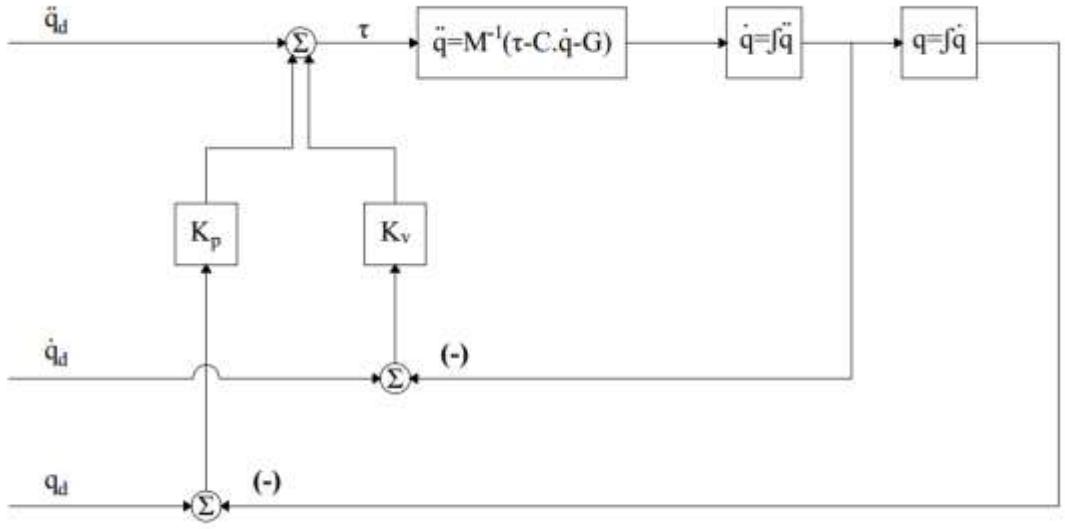
Bộ điều khiển này được gọi là bộ điều khiển PD tăng cường (augmented PD control law). Hệ thống ổn định khi K_v và K_p là các ma trận xác định dương.s

3.4 Đánh giá tính ổn định của hệ thống

Sau khi đã thiết kế được bộ điều khiển, sinh viên có thể sử dụng thuyết ổn định Lyapunov để đánh giá tính ổn định của hệ thống.

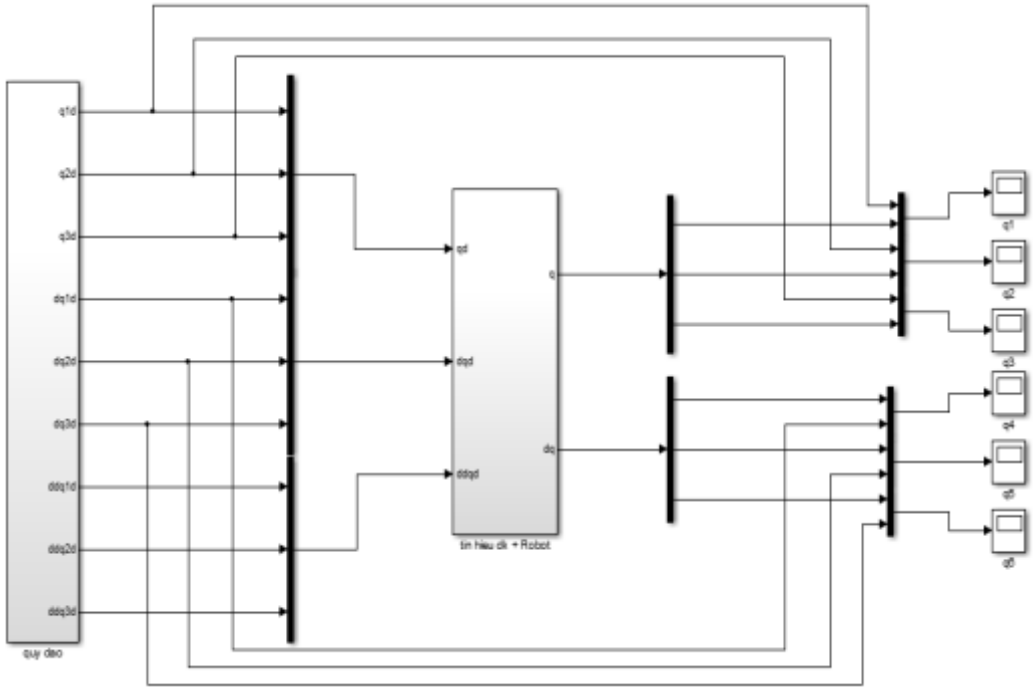
3.5 Mô phỏng và phân tích, đánh giá các chỉ tiêu kỹ thuật của hệ thống

- Xây dựng sơ đồ khối hệ thống điều khiển robot SCARA. Ví dụ:

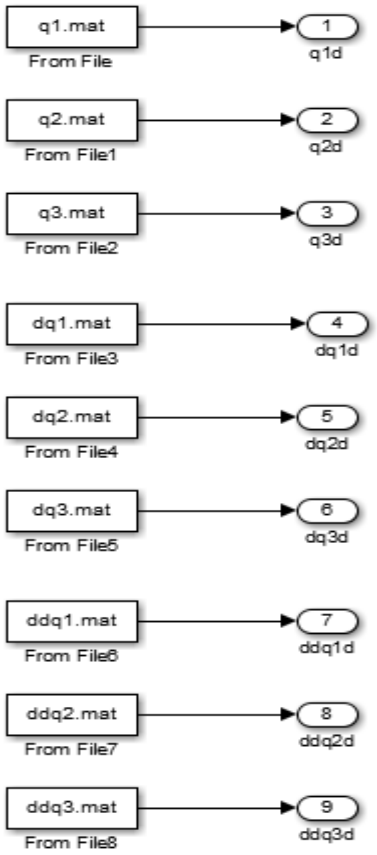


Hình 3.1 Sơ đồ hệ thống điều khiển robot Scara

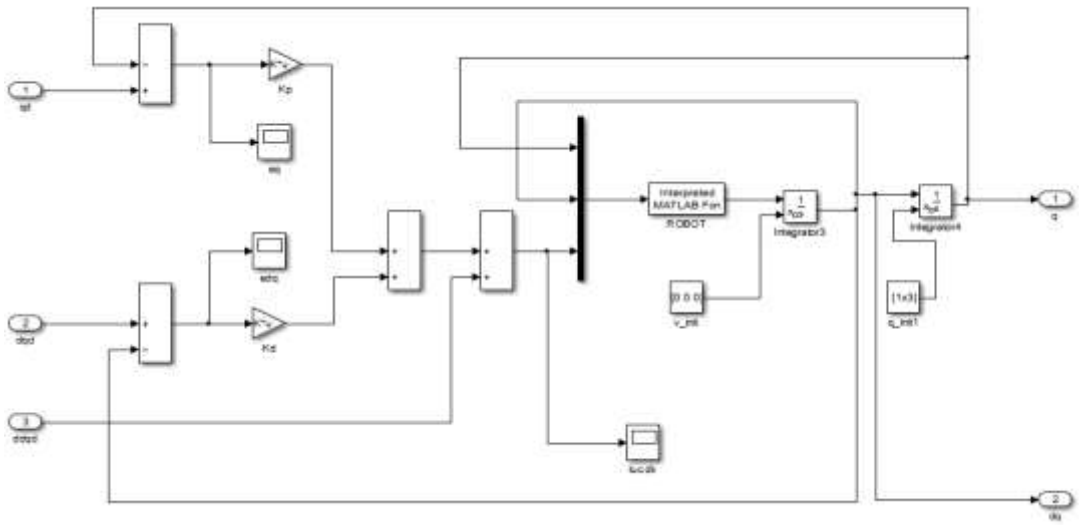
- Xây dựng chương trình mô phỏng điều khiển hệ thống (Matlab/Simulink). Ví dụ:



Hình 3.2 Sơ đồ khối hệ thống trên Matlab Simulink



Hình 3.3 Khối tín hiệu đặt



Hình 3.4 Tín hiệu điều khiển và Robot

- Mô phỏng và đánh giá các chỉ tiêu kỹ thuật của hệ thống

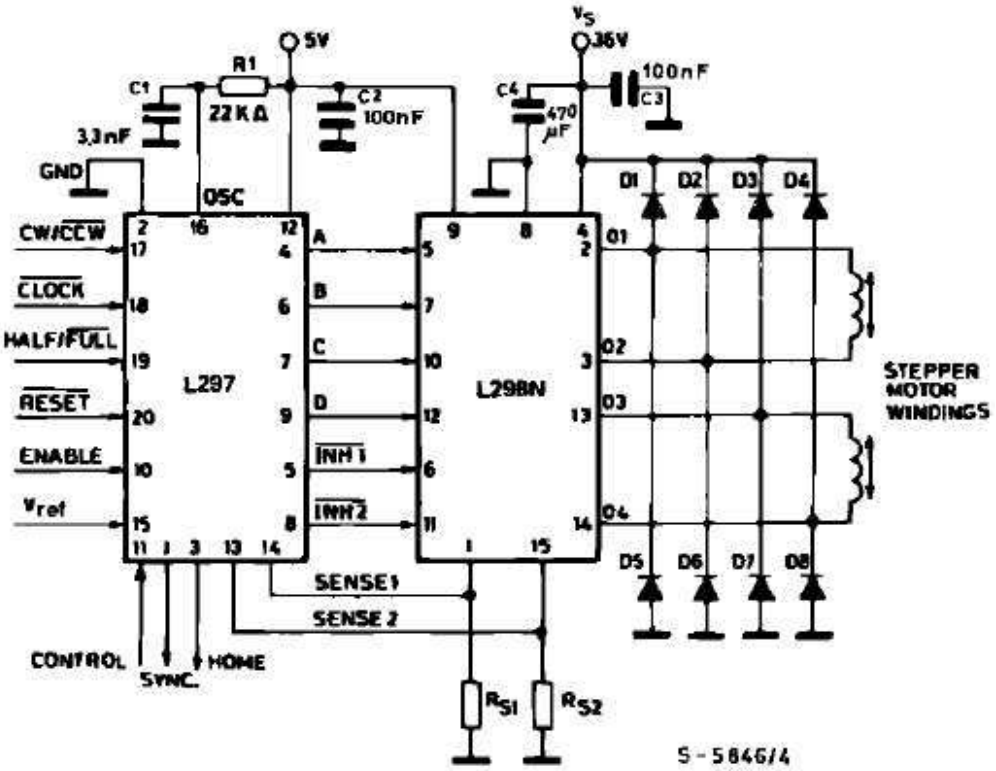
3.6 Lựa chọn các thiết bị cho hệ thống điều khiển

Lựa chọn các thiết bị cho hệ thống điều khiển như:

- Động cơ
- Cảm biến
- Bộ nguồn
- Mạch dẫn động động cơ (ví dụ L298)
- Bộ điều khiển động cơ (ví dụ L297)

3.7 Thiết kế sơ đồ mạch điện và mạch điều khiển

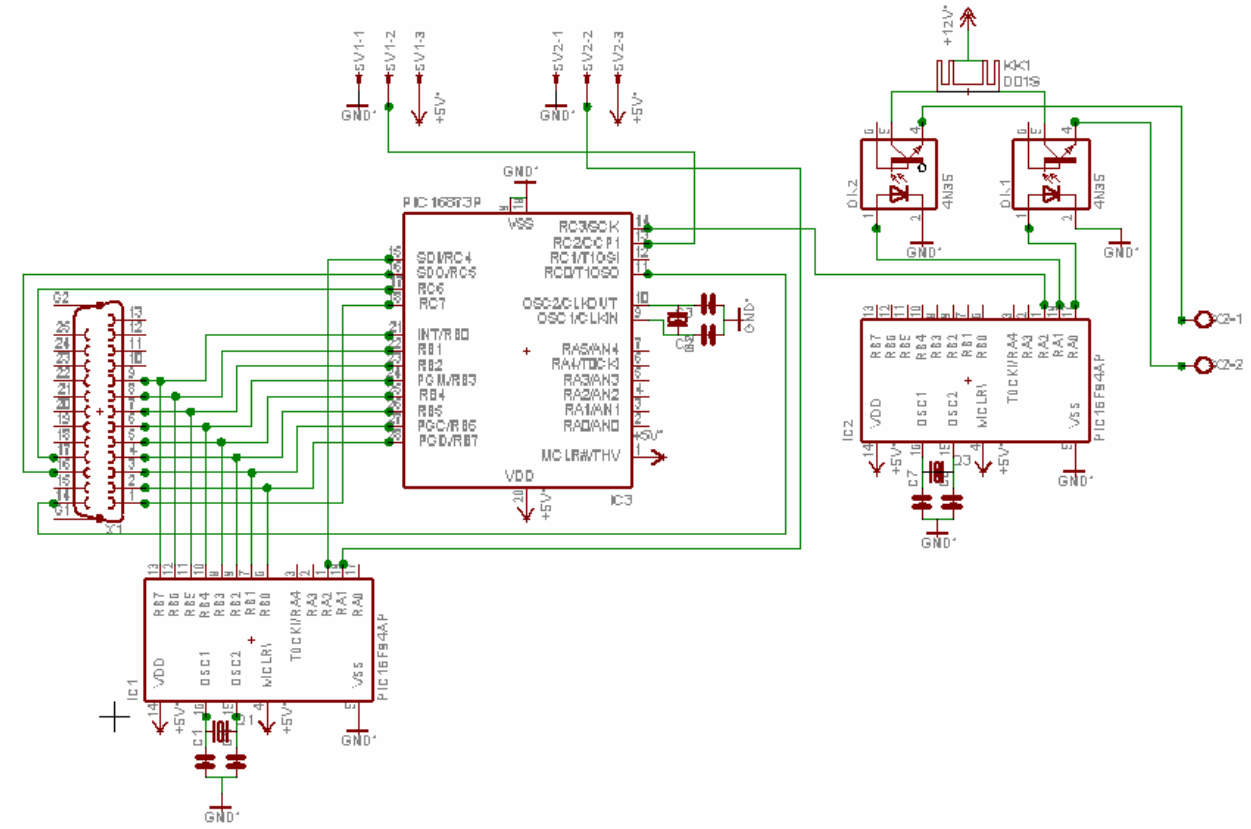
Một số ví dụ về mạch điện điều khiển robot SCARA:



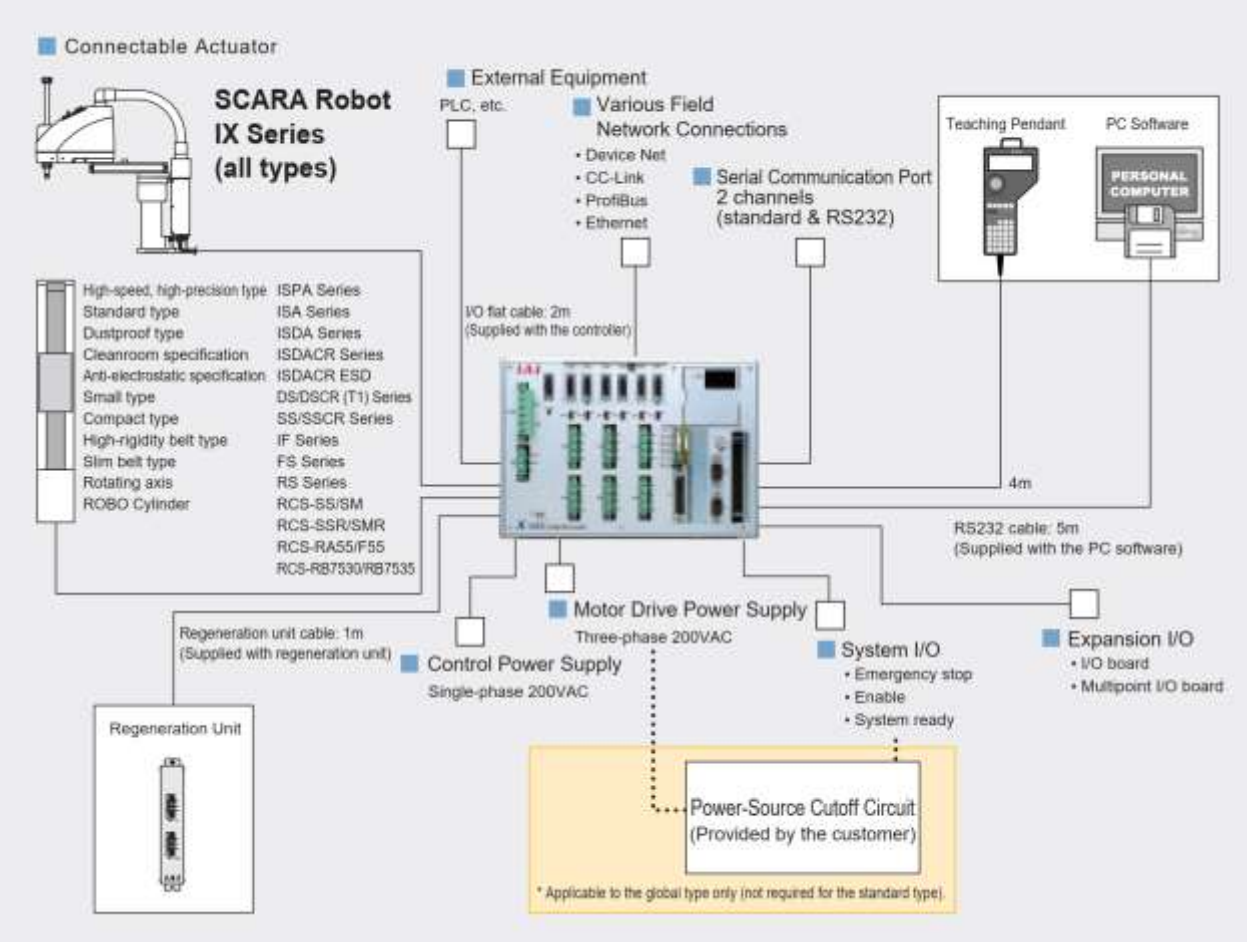
$R_{S1} = R_{S2} = 0.5\ \Omega$

D1 to D8 = 2 A Fast diodes { $V_F \leq 1.2\ V$ @ $I = 2\ A$; $trr \leq 200\ ns$

Hình 3.5 Mạch điều khiển động cơ bước sử dụng L298 và L297 [1]



Hình 3.6 Sơ đồ mạch điều khiển động cơ servo [8].



Hình 3.7 Bộ điều khiển robot SCARA của hãng IAI [9].

Chương 4. Lập trình điều khiển

4.1. Lập trình điều khiển robot

Sinh viên có thể sử dụng phần mềm có sẵn như **LinuxCNC**[7] hoặc tự viết phần mềm điều khiển sử dụng các ngôn ngữ lập trình như C++, C#, python, Java, Matlab/Simulink, LabView, … Phần mềm cần có một số tính năng như sau:

- Lập trình sử dụng G- và/hoặc M-code
- Có thể điều khiển điểm tới điểm
- Có thể điều khiển theo quỹ đạo liên tục
- Có giao diện điều khiển trên máy tính
- …

4.2. Lập trình mô phỏng hệ thống điều khiển

Tài liệu tham khảo

1. Sagar Behere: "*The Design and Implementation of a SCARA robot arm*," Bachelor thesis, University of Pune, 2002.
2. Claudio Urrea, Juan Cortés, José Pascal: "*Design, construction and control of a SCARA manipulator with 6 degrees of freedom*," Journal of Applied Research and Technology 14 (2016) 396–404.
3. F.P.M. Dullens: "*MIMO Controller Design for a SCARA Robot*," Technische Universiteit Eindhoven, 2008.
4. Claudio Urrea and John Kern: "*Modeling, Simulation and Control of a Redundant SCARA-Type Manipulator Robot*," International Journal of Advanced Robotic Systems, 2012.
5. John J. Craig: "*Introduction to Robotics: Mechanics and Control*," Pearson Education International, 2005.
6. Richard M. Murray, Zexiang Li, S. Shankar Sastry: "*A Mathematical Introduction to Robotic Manipulation*," CRC Press, 1994.
7. http://linuxcnc.org/
8. J. F. A. Diaz, M. S. Dutra, C. J. Diaz: "DESIGN AND CONSTRUCTION OF A MANIPULATOR TYPE SCARA, IMPLEMENTING A CONTROL SYSTEM," 19th International Congress of Mechanical Engineering, 2007.
9. IAI, Simultaneous Control of SCARA Robots andSingle-Axis/Cartesian Robots with One Controller, catalog.

HƯỚNG DẪN SỬ DỤNG MATLAB/ SIMULINK TRONG CÁC HỆ CƠ ĐIỆN TỬ

Nhóm biên soạn: TS. Mạc Thị Thoa, TS. Nguyễn Thành Hùng

Bộ Môn: Cơ Điện Tử, Viện Cơ Khí, Đại học Bách Khoa Hà Nội

I. Khái niệm về SIMULINK:

Là phần mềm mở rộng của Matlab dùng để mô hình hóa, mô phỏng và phân tích hệ thống động. Thường dùng thiết kế hệ thống điều khiển, DSP, hệ thống thông tin ….

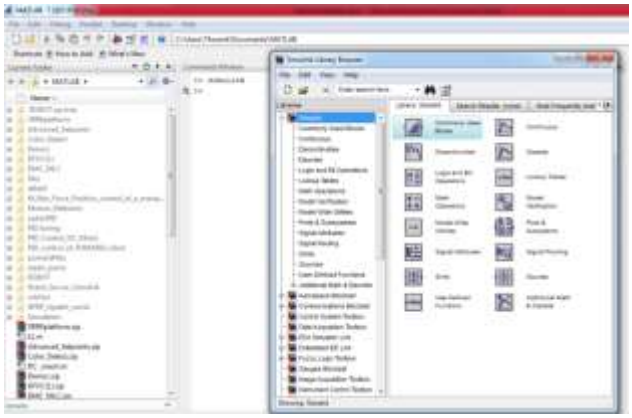
Simulink là thuật ngữ ghép bởi Simulation và Link, cho phép mô tả hệ thống tuyến tính, phi tuyến, các mô hình trong thời gian liên tục, gián đoạn, hệ gồm cả liên tục và gián đoạn. Để mô hình hóa, Simulink cung cấp một giao diện đồ họa để sử dụng và xây dựng mô hình sử dụng thao tác "nhấn và kéo" chuột do vậy có thể xây dựng mô hình trực quan hơn. Yêu cầu người sử dụng phải có kiến thức về điều khiển, xây dựng mô hình toán học của hệ thống điều khiển.

II. Tìm hiểu về SIMULINK và BLOCKS LIBRARY

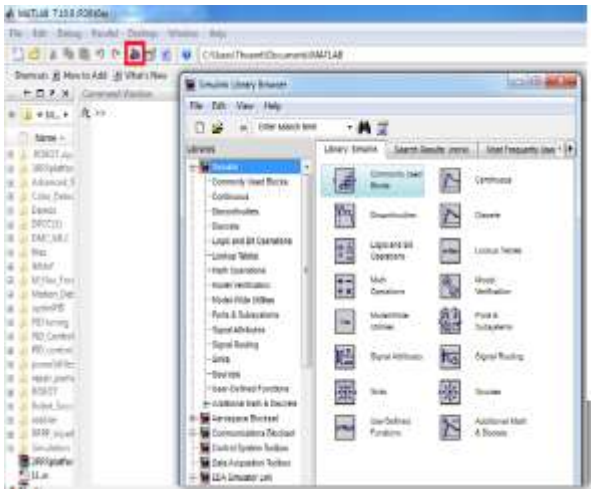
1. Cách khởi tạo SIMULINK và vẽ sơ đồ mô phỏng

Có 2 cách khởi tạo Simulink

a. Từ cửa sổ Matlab, đánh dòng lệnh simulink ↵

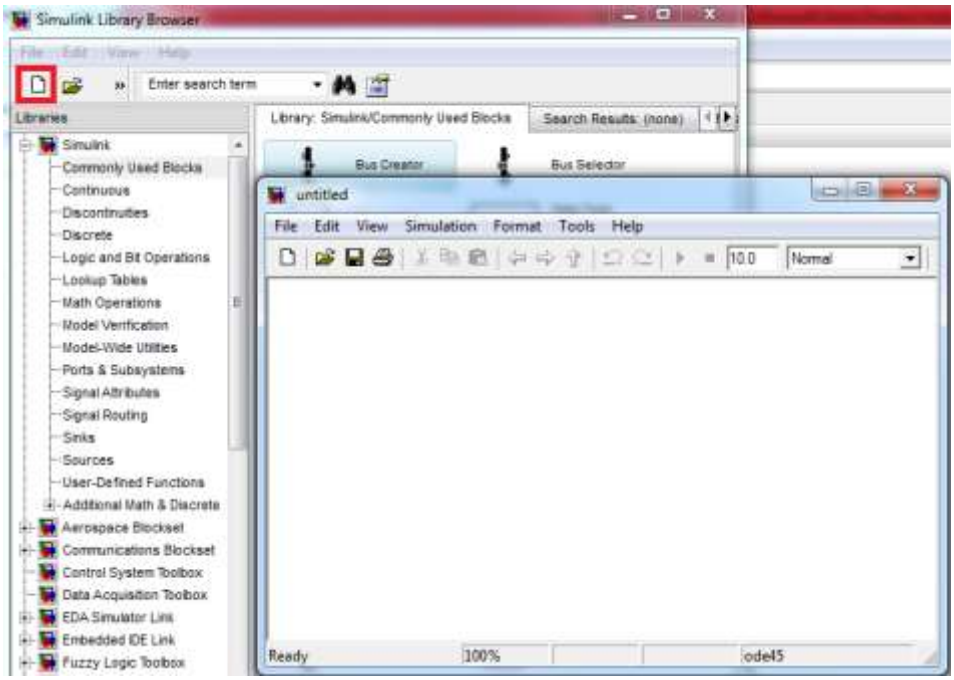


b. Click biểu tượng Simulink trên thanh công cụ vủa Matlab



Lúc này một cửa sổ xuất hiện như hình vẽ , trong đó có các thư mục chính và các thư viện con của Simulink. Các khối thư viện con gồm có: Common Used Blocks (các khối thông dụng), Continuous (Hệ thống tuyến tính & liên tục), Discrete ((Hệ thống tuyến tính & gián đoạn), Nonliear (mô hình hóa các phần tử phi tuyến như Rơle, phần tử bão hòa), Source (các khối nguồn tín hiệu), Sinks (Các khối thu nhận tín hiệu), Math (Các khối có hàm toán học tương ứng trong Matlab) ……

Để khởi tạo một file mới, nhấp chuột vào File/New Model (Ctrl+ N), màn hình cửa số Untitled được mở ra, từ đó ta xây dựng mô hình.

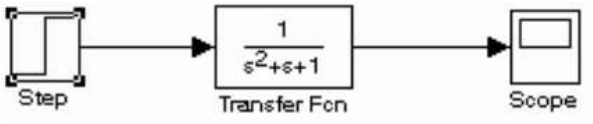


2. Tạo một sơ đồ đơn giản
   Để làm quen với Simulink ta tìm hiểu ví dụ đơn giản sau:

Bài 1: Phân tích hàm quá độ của một khâu bậc 2 có hàm truyền

$$G(p) = \frac{\omega_0^2}{p^2 + 2\xi\omega_0 p + \omega_0^2}$$

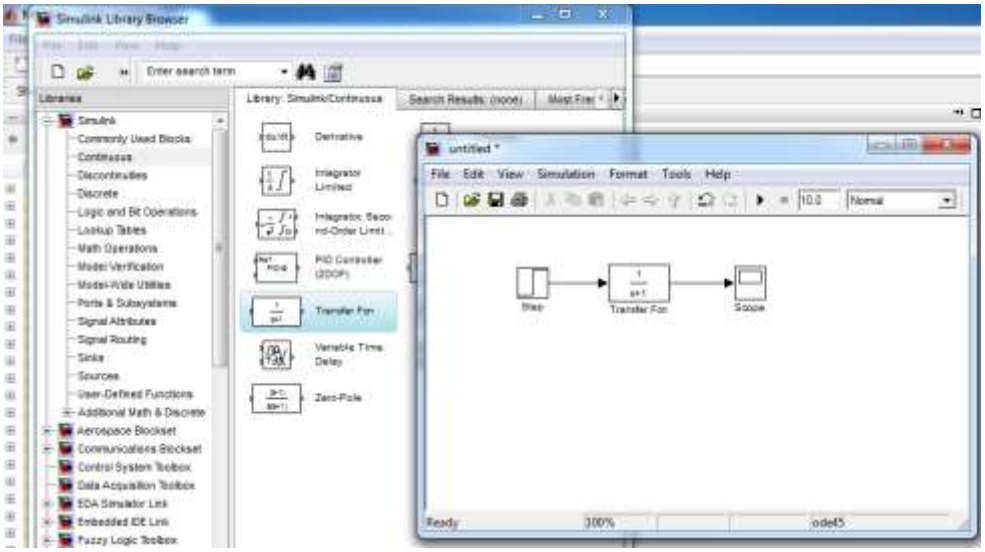
Với ω_o = 1rad/s, ξ = 0,5. Các bước để thực hiện được sơ đồ mô phỏng như hình 2.1 như sau:



Hình 2.1 Mô hình Simulink đơn giản

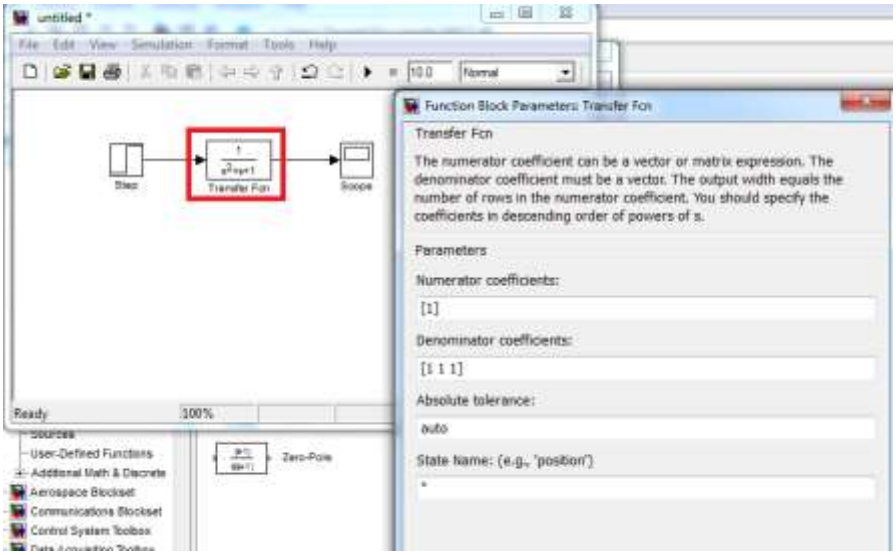
Trong sơ đồ này chọn các khối từ thư viện:

a. Source (các khối nguồn tín hiệu): Chọn khối Step
b. Sinks (Các khối thu nhận tín hiệu): Chọn khối Scope
c. Continuous (Hệ thống tuyến tính & liên tục): Chọn khối Transfer Fcn



Hình 2.2. Mô hình trên Simulink

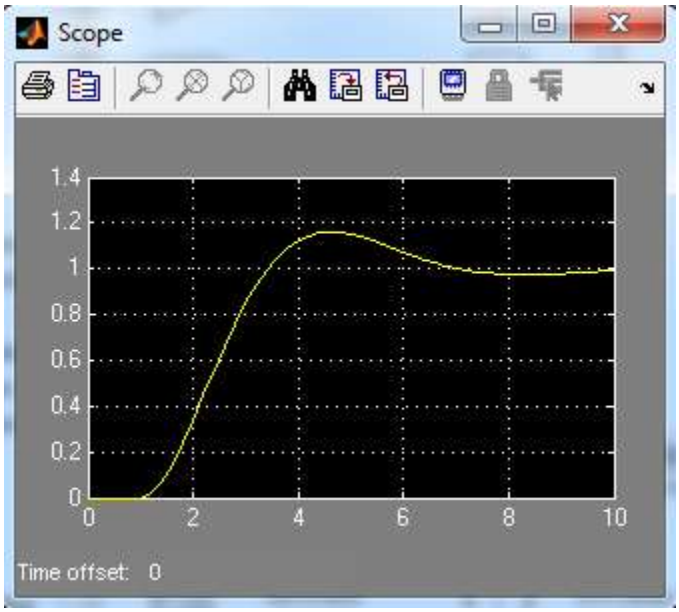
Ta thu được File như hình 2.2. Để đạt thông số cho từng khối, ta mở khối đó bàng double-click vào nó. Lúc này các thông số đạt theo hướng dẫn trên màn hình. Ví dụ ta muốn xây dựng hàm truyền G(s) = $1/s^2 + s + 1$, numerator coefficients = 1, denominator coefficients = [1 1 1]



Hình 2.3. Thay đổi thông số các khối Simulink

Đường nối giữa các khối tạo ra bàng cách dùng chuột kéo các mũi tên ở đầu (cuối) mỗi khối đến vị trí cần nối.

Khi tạo ra sơ đồ khối như hình 2.3, ta tiến hành mô phỏng (các tham số mạc định). Chọn Simulation -> Start. Xem kết quả mô phỏng bàng cách mở khối Scope



Hình 2.4. Kết quả mô phỏng

Bài 2: Vẽ quỹ đạo chuyển động theo thời gian

Giả sử phương trình quỹ đạo của một hệ được mô tả như sau:

$$s(t)=\begin{cases}\dfrac{F\tau_1^{\,2}}{2T_1}; & 0\le\tau_1\le T_1\\ F\tau_2+\dfrac{FT_1}{2}; & 0<\tau_2\le T_2\\ \dfrac{-F\tau_3^{\,2}}{2T_3}+F\tau_3+L-\dfrac{FT_3}{2}; & 0<\tau_3\le T_3\end{cases} \quad (1)$$

$$T_1=\frac{F}{A}\quad T_3=-\frac{F}{D}\quad T_2=\frac{L}{F}-\frac{(T_1+T_3)}{2}$$

Với

$A = 500\ mm/sec^2$

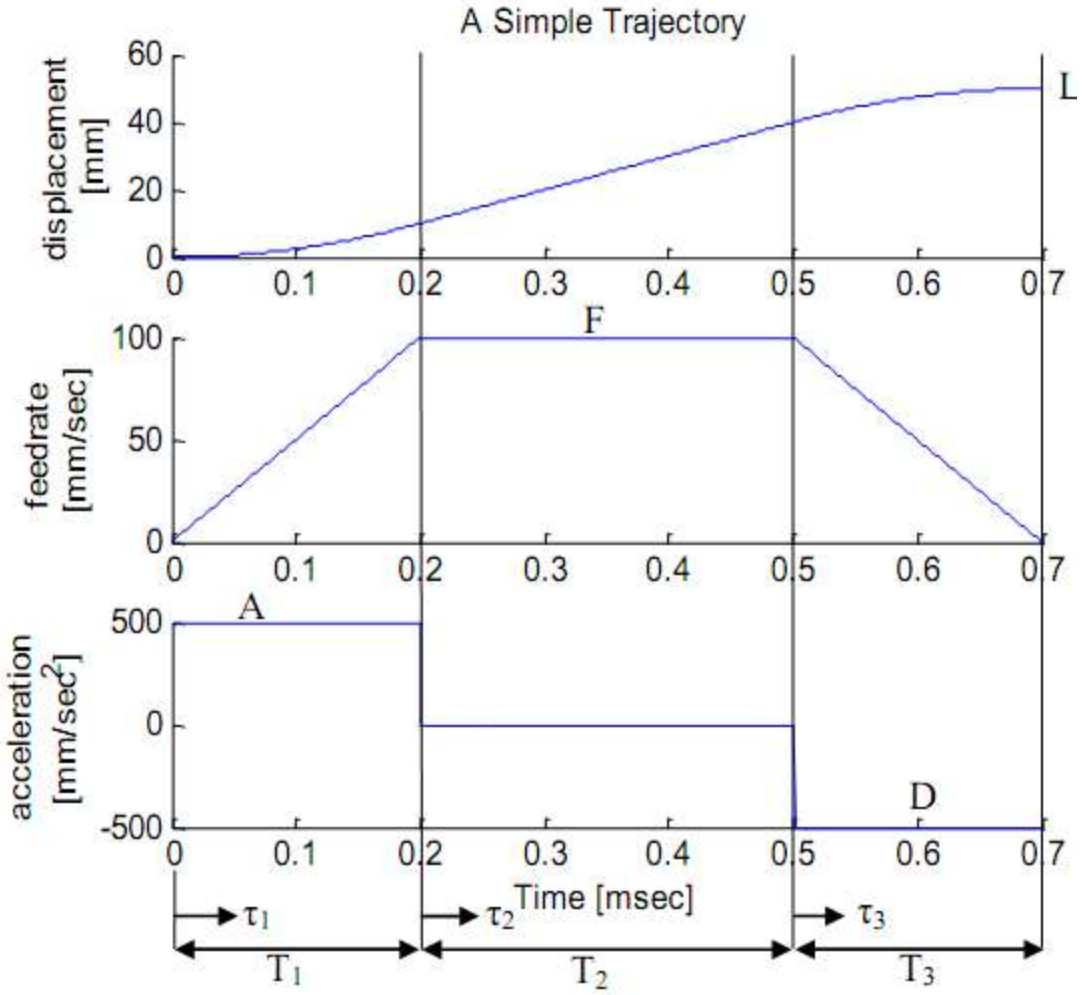
$D = -500\ mm/sec^2$

$F = 100\ mm/sec$

$L = 50\ mm$

$T_s = 0.001\ sec$

1. Tính toán quỹ đạo, vận tốc và gia tốc chuyển động
2. Vẽ đồ thị của quỹ đạo, vận tốc và gia tốc chuyển động theo thời gian như hình vẽ sau
3. Lưu các dữ liệu của quỹ đạo chuyển động theo thoig gian sang file text



Bài giải:

Tạo file Matlab như sau (*.m)

```
clear all % clears the workspace

close all % closes all figure windows



% Given:

A=500; % acceleration mm/sec^2

D =-500; % deceleration mm/sec^2

F=100; % feedrate (velocity) mm/sec

L=50; % travel length mm
```

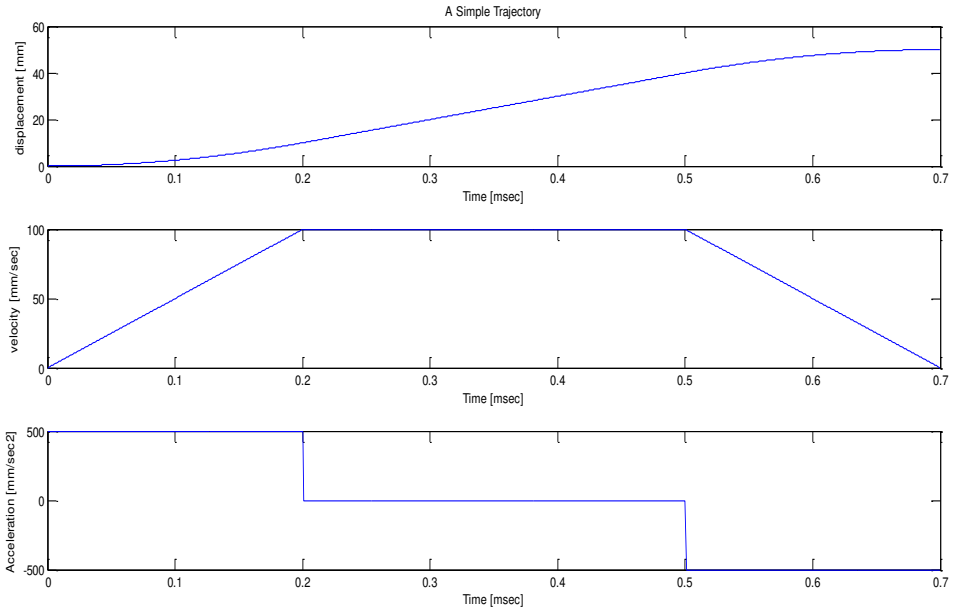
```
Ts=0.001; % sampling period sec


T1=F/A; % from equation 2

T3=-F/D; % from equation 3

T2=L/F - (T1+T3)/2; % from equation 4


if T1<0 || T2<0 || T3<0 % kinematic compatibility conditions

disp('Error: acceleration, deceleration and travel length are not kinematically
compatible.');

else

   % create row vector for time, initial time : step size : final time

   tau1=0:Ts:T1;

   tau2=0:Ts:T2;

   tau3=0:Ts:T3;

   %... (do same for tau2 and tau3)

    % preallocate array for speed

   s1=zeros(1,length(tau1));

   sd1=zeros(1,length(tau1));

   sdd1=zeros(1,length(tau1));

   %... (do same for s2 and s3)

   s2=zeros(1,length(tau2));

   sd2=zeros(1,length(tau2));

   sdd2=zeros(1,length(tau2));


   s3=zeros(1,length(tau3));
```

```
    sd3=zeros(1,length(tau3));
    sdd3=zeros(1,length(tau3));

   % from equation 1
    for index=1:length(tau1)
        s1(index)=F*tau1(index)*tau1(index)/2/T1;
        sd1(index)=F*tau1(index)/T1; % first derivative: velocity
        sdd1(index)=A; % second derivative: acceleration
    end
    for index=1:length(tau2)
        s2(index)=F*tau2(index) + F*T1/2;
        sd2(index)=F;
        sdd2(index)=0;
    end
    for index=1:length(tau3)
        s3(index)=-F*tau3(index)*tau3(index)/2/T3 + F*tau3(index) + L - F*T3/2;
        sd3(index)=-F*tau3(index)/T3 + F;
        sdd3(index)=D;
    end
end

t1=tau1'; % change tau1 from a row vector to a column vector tau1'
t2=tau2(2:end)' +T1; % shift bounds and drop first vector element
t3=tau3(2:end)'+T2+T1;
t = [t1;t2;t3]; % concatenation of time vector
```

```
s1=s1';
s2=s2(2:end)';
s3=s3(2:end)';
%
sd1=sd1';
sd2=sd2(2:end)';
sd3=sd3(2:end)';
%
sdd1=sdd1';
sdd2=sdd2(2:end)';
sdd3=sdd3(2:end)';
s = [s1;s2;s3]; % concatenation of trajectory vector
%... (do same for sd and sdd)
sd = [sd1;sd2;sd3]
sdd = [sdd1;sdd2;sdd3]

figure(1); % opens a figure window
subplot(3,1,1); % subplot(rows, columns, position)
  plot(t,s); % plots trajectory versus time
  title('A Simple Trajectory'); % creates a title for the plot
  ylabel('displacement [mm]'); % labels the y-axis
  xlabel('Time [msec]') % labels the x-axis
subplot(3,1,2);
%... (do same for sd and sdd)
```

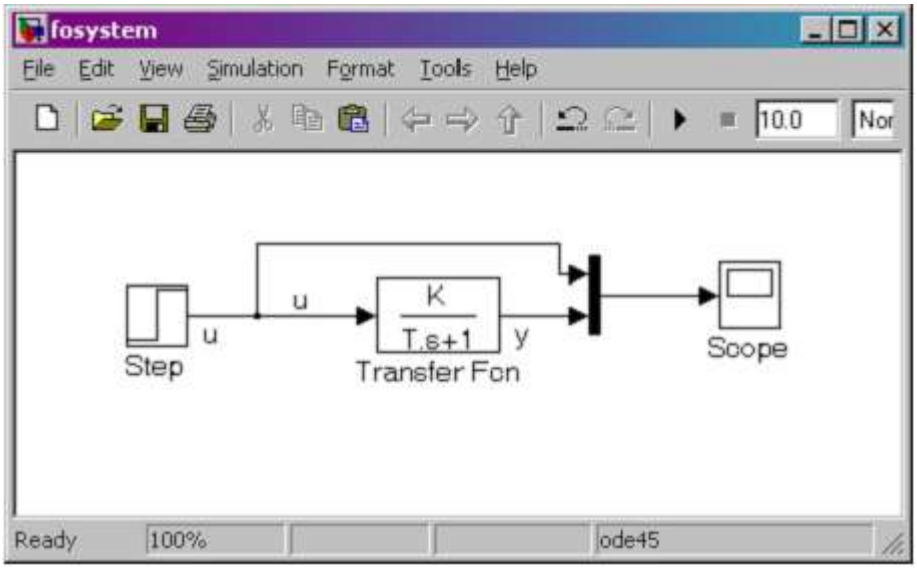
```
 plot(t,sd); % plots trajectory versus time
 %title('A Simple Trajectory'); % creates a title for the plot
 ylabel('velocity [mm/sec]'); % labels the y-axis
 xlabel('Time [msec]') % labels the x-axis
subplot(3,1,3);
  plot(t,sdd); % plots trajectory versus time
 %title('A Simple Trajectory'); % creates a title for the plot
 ylabel('Acceleration [mm/sec2]'); % labels the y-axis
 xlabel('Time [msec]') % labels the x-axis
% save data to file
data = [t s]; % t and s should be column vectors
save ex1_data.txt -ASCII -DOUBLE data
%... (save also sd and sdd to the same file)


%-- End of File --%
```

Kết quả thu được:

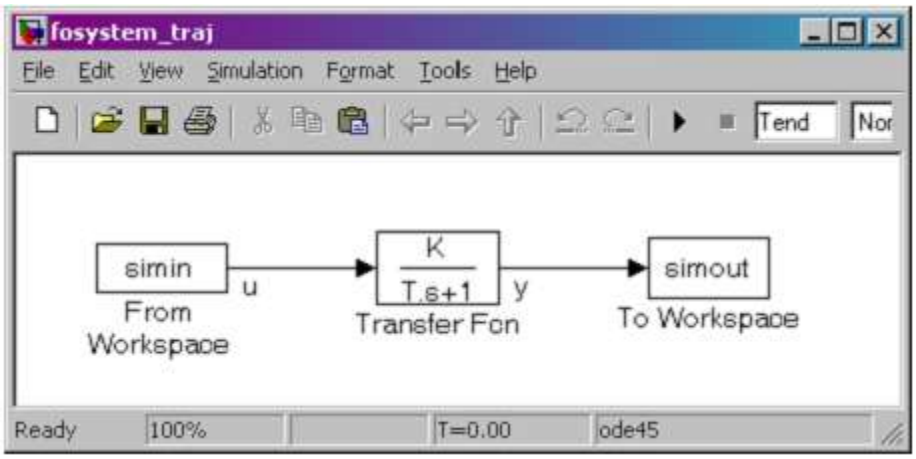


Bài 3: Mô phỏng hệ thống động lực học: Một hệ thống động lực học có đầu vào là u(t), đầu ra là y(t):

1. Thiết lập mô hình Simulink bậc nhất, hệ số K, hệ số thời gian T. Mô phỏng hệ thống khi đầu vào là hàm Step. Lưu lại dữ liệu và vẽ biểu đồ tín hiệu đầu ra.
2. Sử dụng quỹ đạo xây dựng ở bài 2 làm tín hiệu đầu vào . Lưu lại dữ liệu và vẽ biểu đồ tín hiệu đầu ra.



Hình 3.1: Hệ một bậc tự do



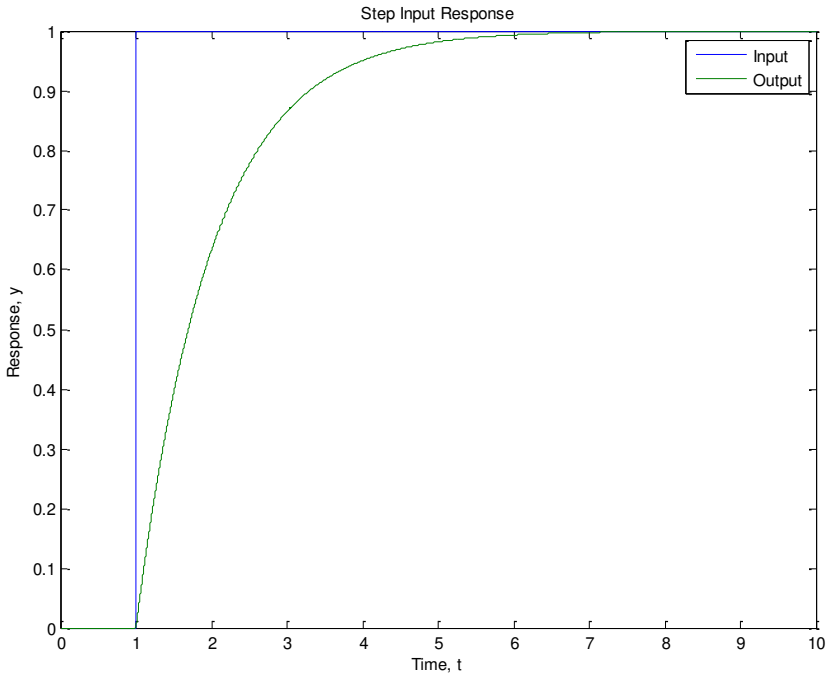
Hình 3.2 Tín hiệu đầu vào là quỹ đạo trong ví dụ 2

1. Matlab Code

```
clear all % clears the workspace

close all % closes all figure windows


K=1; % set gain in workspace

T=1; % set time constant in workspace


open('C:\Users\Thoamt\Documents\MATLAB\fosystem.mdl'); % opens the model file

sim('C:\Users\Thoamt\Documents\MATLAB\fosystem.mdl'); % runs the simulation


% extract data from Scope Data struct

t = ScopeData.time;

u = ScopeData.signals.values(:,1);

y = ScopeData.signals.values(:,2);


% plot step input and output response
```

```
figure(2);
plot(t,u,t,y);
title('Step Input Response');
ylabel('Response, y');
xlabel('Time, t');
legend('Input','Output'); % creates a legend for the plot

% save data to file
data = [t u y];
save ex2_q1_data.txt -ASCII -DOUBLE data

%-- End of File --%
```

Kết quả mô phỏng

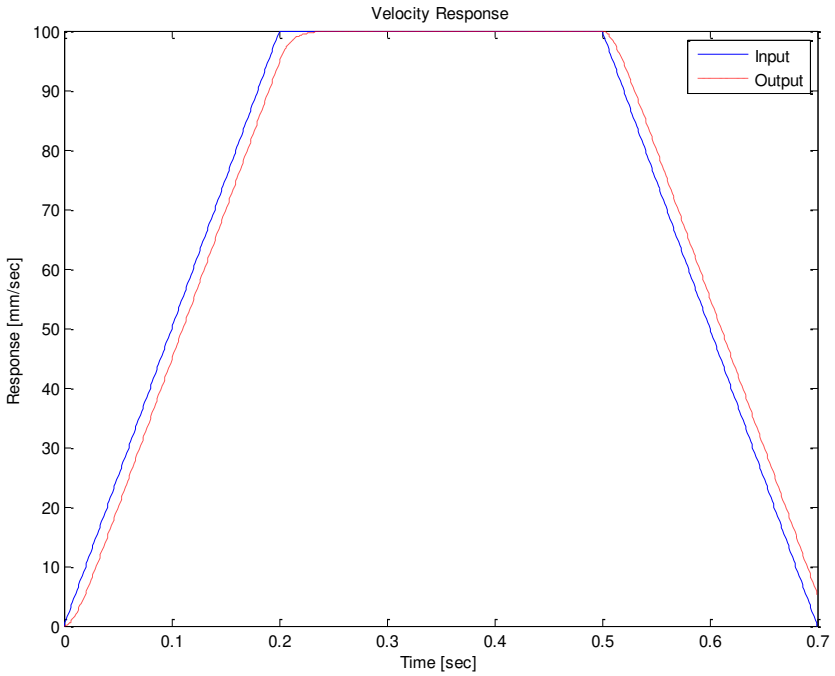


2. Matlab Code

```
clear all % clears the workspace

close all % closes all figure windows


S = load('C:\Users\Thoamt\Documents\MATLAB\ex1_data.txt'); % loads data
from file to the workspace

t = S(:,1); % extract time vector

u = S(:,3); % extract velocity profile


Tend = t(end); % get total time of simulation

Ts = 0.001; % sampling period sec


K = 1; % set gain in workspace

T = 0.01; % set time constant in workspace


simin = [t u]; % array format for 1-D input signal


open('C:\Users\Thoamt\Documents\MATLAB\fosystem_traj.mdl'); % opens the
model file

sim('C:\Users\Thoamt\Documents\MATLAB\fosystem_traj.mdl'); % runs the
simulation


time = simout.time; % if the save format of simout is "Structure with Time"

y = simout.signals.values; % output vector


% plot input and output in the same figure
```

```
figure(3);
plot(t,u,'b-'); hold on % hold on retains the current plot
plot(time,y,'r-.');
title('Velocity Response');
ylabel('Response [mm/sec]');
xlabel('Time [sec]');
legend('Input','Output'); % creates a legend for the plot

% save data to file
data = [t u y];
save ex2_q2_data.txt -ASCII -DOUBLE data

%-- End of File --%
```

Kết quả mô phỏng



Bài 4:

Tạo mô hình Simulink bậc nhất K = 1, T = 0.1. Tín hiệu đầu vào là xung vuông, biên độ 1, tần số 0.3 Hz. Thời gian lấy mẫu 0.001 giây. Quan sát tín hiệu $x_r(t)$, đầu vào u(t), vị trí thực tế x(t), qua Scope như trong Figure 4.1. Thay đổi thông số Kp quan sát sự biến đổi đầu ra. Vẽ đồ thị Figure 4.2.

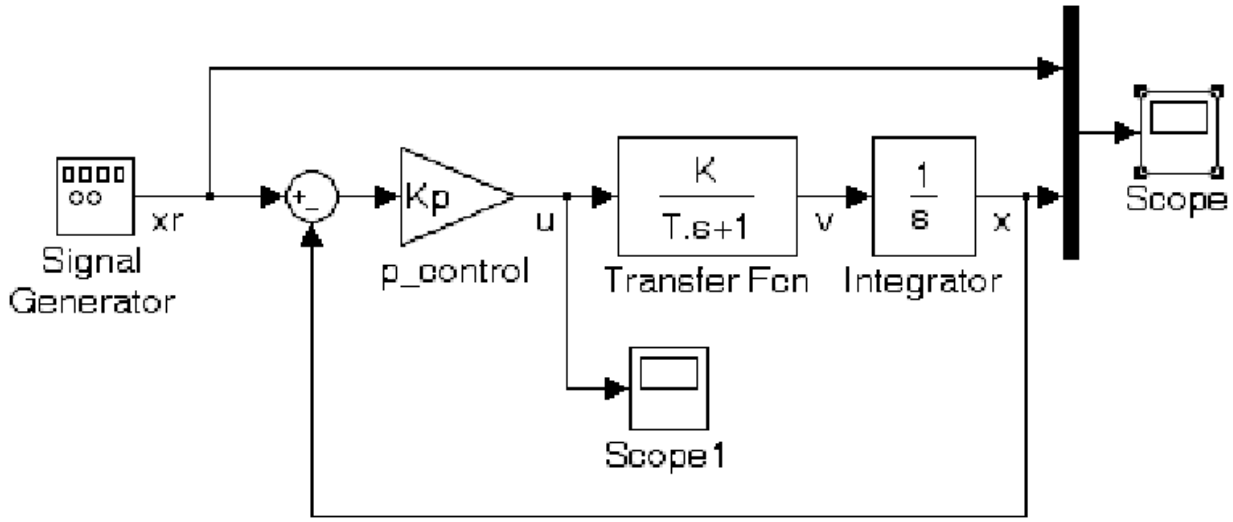


Figure 4.1 . Bộ điều khiển K_P

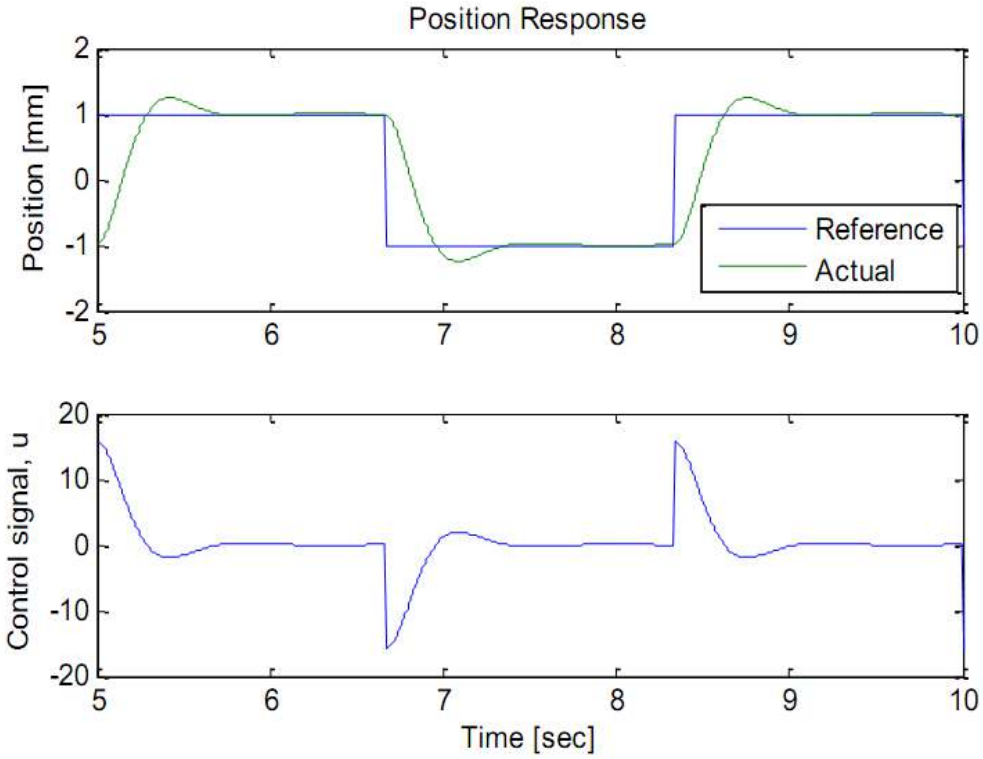


Figure 4.2 . Vị trí và tín hiệu điều khiển

Bài giải:

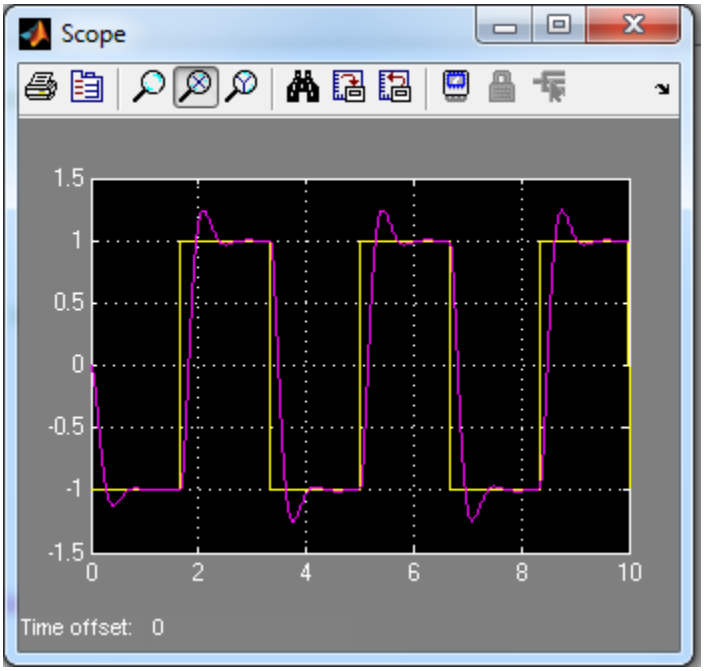
1. Tạo mô hình Simulink giống hình vẽ
2. Matlab Code

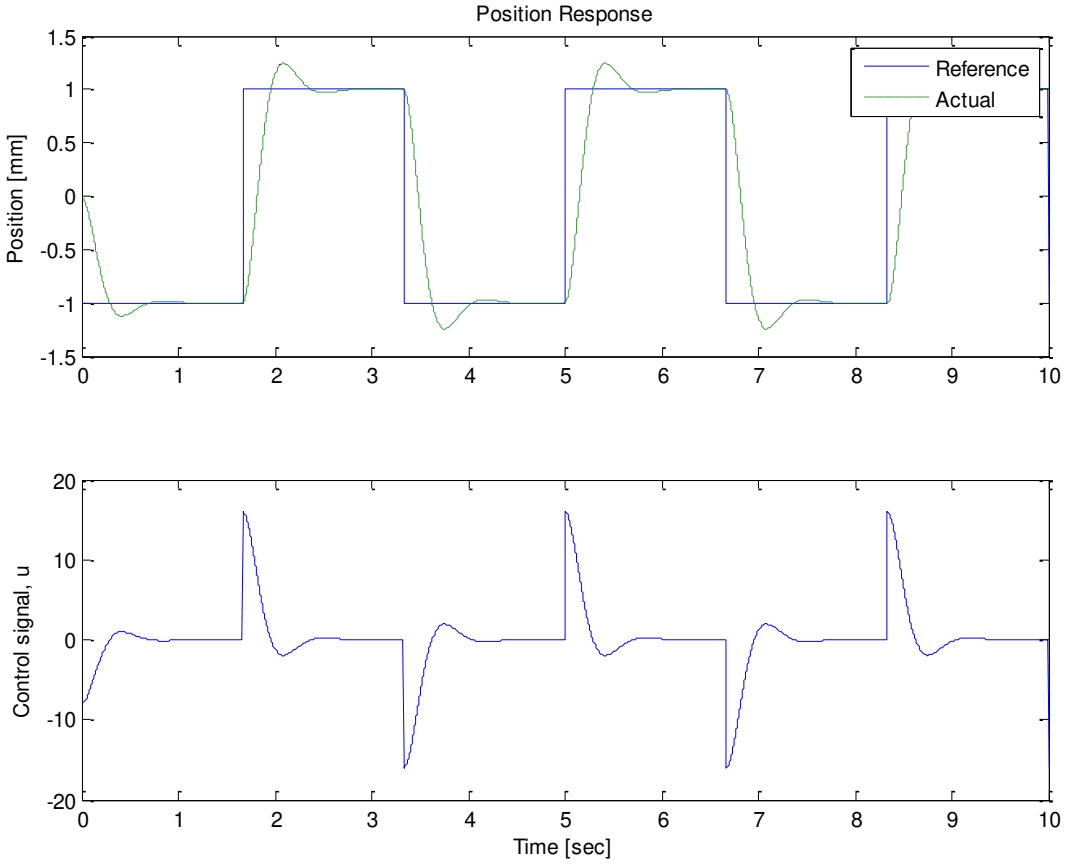
```
clear all % clears the workspace

close all % closes all figure windows



K=1; % set gain in workspace

T=0.1; % set time constant in workspace


Ts = 0.001; % sampling period sec

sq_freq=0.3; % frequency of square wave Hz
```

```
sq_amp=1; % amplitude of square wave


Kp=8; % proportional controller gain


open('C:\Users\Thoamt\Documents\MATLAB\p_controller.mdl'); % opens the
model file

sim('C:\Users\Thoamt\Documents\MATLAB\p_controller.mdl'); % runs the
simulation


% extract data from Scope Data struct

t = ScopeData.time;

xr = ScopeData.signals.values(:,1);

x = ScopeData.signals.values(:,2);

u = ScopeData1.signals.values(:,1);


% plot step input and output response

figure(4);

subplot(2,1,1);

plot(t,xr,t,x,'--');

 title('Position Response');

 ylabel('Position [mm]');

 legend('Reference','Actual'); % creates a legend for the plot

subplot(2,1,2);

plot(t,u);

 ylabel('Control signal, u');
```

```
 xlabel('Time [sec]');


% save data to file
data = [t xr u x];
save ex3_data.txt -ASCII -DOUBLE data


%-- End of File --%
```

Kết quả mô phỏng:





II. Mô phỏng hệ thống động lực học sử dụng hàm Ode/ simulink trong matlab.

1. Mô phỏng hệ thống động lực học sử dụng hàm Ode

Các hàm Ode trong matlab dùng để giải phương trình vi phân (ODE - Ordinary Differential Equations) . Trong phần này ta ví dụ hàm ode23.

Để tìm hiểu hàm ode23, nhập dòng lệnh "help ode23". Hàm ode23 ứng dụng để giải phương trình vi phân bậc nhất $\dot{y}$ = f(y,t). Ví dụ, một hệ thống được mô tả bởi phương trình vi phân sau:

$$m_1 a_1^2 \ddot{q}_1 + m_1 g a_1 \cos(q_1) = \tau \qquad (1)$$

Đây là phương trình vi phân mô tả động lực học của robot phẳng 1 bậc tự do.

q_1 : góc quay của tay robot so với phương ngang

τ : Mô men tác động lên cánh tay Robot, $\tau = 20*e^{-0.2t}\cos(2\pi t)$

a_1: chiều dài cánh tay robot

m_1: khối lượng cánh tay robot

g: gia tốc trọng trường

Do hàm ode23 chỉ giải phương trình vi phân bậc nhất, ta chọn biến trạng thái thứ 2 để biểu diễn (1) dưới dạng 2 phương trình vi phân bậc nhất. Chọn $x_1 = q_1$, $x_2 = \dot{q}1$, $u = \tau$.

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\dfrac{g}{a_1}\cos(x_1) + \dfrac{1}{m_1 a_1^{\,2}}u \end{cases}$$

Tạo hàm "robot1d.m" miêu tả hệ phương trình vi phân như sau:

```
function xdot=robot1d(t,x);
 % biendau vao x là mot vector gom x1 and x2; x1=x(1) and x2=x(2)
  g=9.8 ; a1=1 ; m1=10 ;
 %  dinh nghia cac tham so dau vao cua he thong
 u=20*exp(-t*0.2)*cos(2*pi*t);
 %  dinh nghia Momen tac dong vao he thong
 xdot= [x(2) ; -(g/a1)*cos(x(1))+u/(m1*a1^2)];
 % dinh nghia vetor xdot la dao ham cua x gom  x1_dot and x2_dot
```

Tham số thứ 2 định nghĩa trong hàm ode là khoảng thời gian hệ thống. Giả sử ta muốn quan sát hệ thống trong 10s, TSPAN = [0 10];

Điều kiện ban đầu: $x1 = \pi/2$; x 2 = 0 Hàm ode23 trả về 2 ma trận TOUT và YOUT ứng với vector thời gian, biến trạng thái của hệ thống.

Tạo file robot1dmain.m mô phỏng như sau:

```
clear all; % clears the workspace
close all; % closes all open figures
clc; % clears the command window
```

```
% these three commands are not required for your program to work


tint= [0 10] ; % defines the time interval [t0 tf], we will simulate

% the system for 10 seconds


x0= [pi/2 0]' ; % initial conditions


[t,x]= ode23('robot1d', tint, x0);



plot(t,x);

grid

% plots the system states versus time

a1= 1;

x1=cos(x(:,1))*a1;

y1=sin(x(:,1))*a1;


figure(3);

for i=1:length(x1),

figure(3);

plot(0,0,'o',x1(i),y1(i),'o',[0 x1(i)],[0 y1(i)],'-r');

axis([-1.2 1.2 -1.2 1.2]);

%pause(t(i+1)-t(i));

end
```

Hướng dẫn kết nối Simmechanics Matlab

với phần mềm CAD

I)Giới thiệu về Simmechanics

Simmechanics là 1 toolbox hỗ trợ việc thiết kế,tính toán,mô phỏng các hệ cơ học (có kết hợp với Simulink của Matlab)

II)Hướng dẫn cài đặt

1)Yêu cầu về phần mềm

+)1 trong các phần mềm có chứa Cad:Solidworks,Inventor,catia,autocad..(Ở đây,trong phạm vi bài hướng dẫn sẽ sử dụng Solidworks)

Phần mềm Cad phục vụ mục đích thiết kế cơ hệ cần khảo sát ở dạng 3D

+)Matlab

+)Phần mềm Simmechanics Link (được download trực tiếp từ Mathworks.com),thực hiện việc đưa đối tượng 3D từ Cad vào Simmechanics trong Matlab.

Chú ý quan trọng:

i)Tùy thuộc hệ điều hành Win 7 32 bit hay 64 bit mà ta sẽ cài đặt Matlab và Solidworks ,Simmechanics Link đồng thời tương ứng.

ii)Ví dụ:Đã test thử thành công với các bộ phần mềm:

*)Solidworks 2012 SP0,Matlab R2012a,Simmechanics link 2012a với hệ điều hành Win 7 64 bit(Khuyến khích sử dụng bộ này vì từ Matlab 2012a,Simmechanics toolbox có thêm thế hệ 2 phục vụ cho việc mô phỏng trực quan hơn)

*)Solidworks 2011 SP0,Matlab R2011a,Simmechanics link R2011a(Hệ điều hành Win 7 64bit)

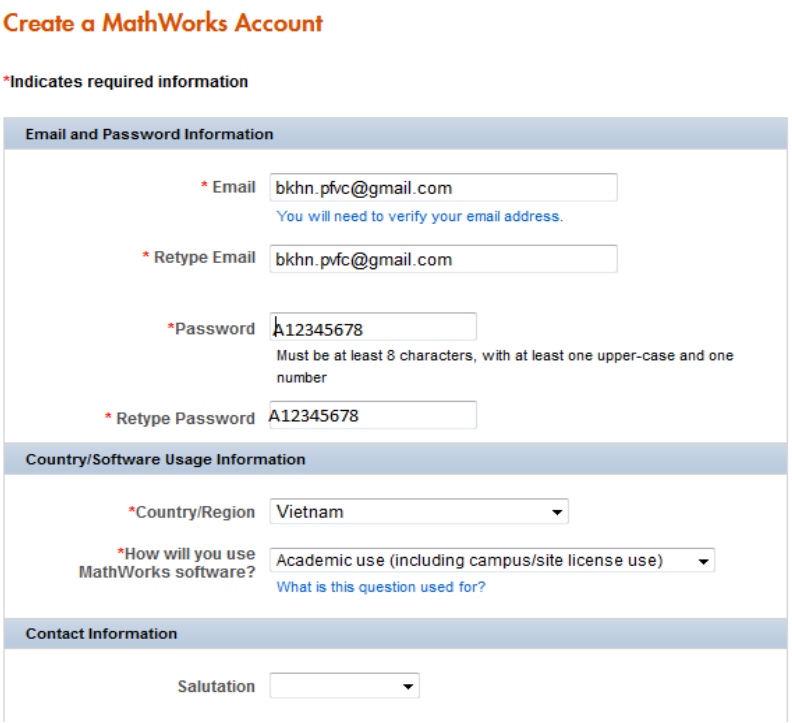
*)Soliworks 2010 SP0,Matlab R2009b,Simmechacnics link R2009b(Hệ điề hành Win 7 32 bit)

2)Hướng dẫn cài đặt

Bước 1.Download Simmechacnics link

a)Đăng ký tài khoản

Vào trang www.mathworks.com, kích chuột trái vào Create Account ở bên trên góc phải ,Sau đó nhập địa chỉ gmail,mật khẩu(Chú ý quan trọng là mật khẩu gồm ít nhất 8 kí tự,trong đó bắt buộc phải có ít nhất 1 chữ viết hoa và ít nhất 1 số),các phần khác có thể lựa chọn tùy ý



Các phần có * là bắt buộc nhập vào,còn không thì có thể để trống

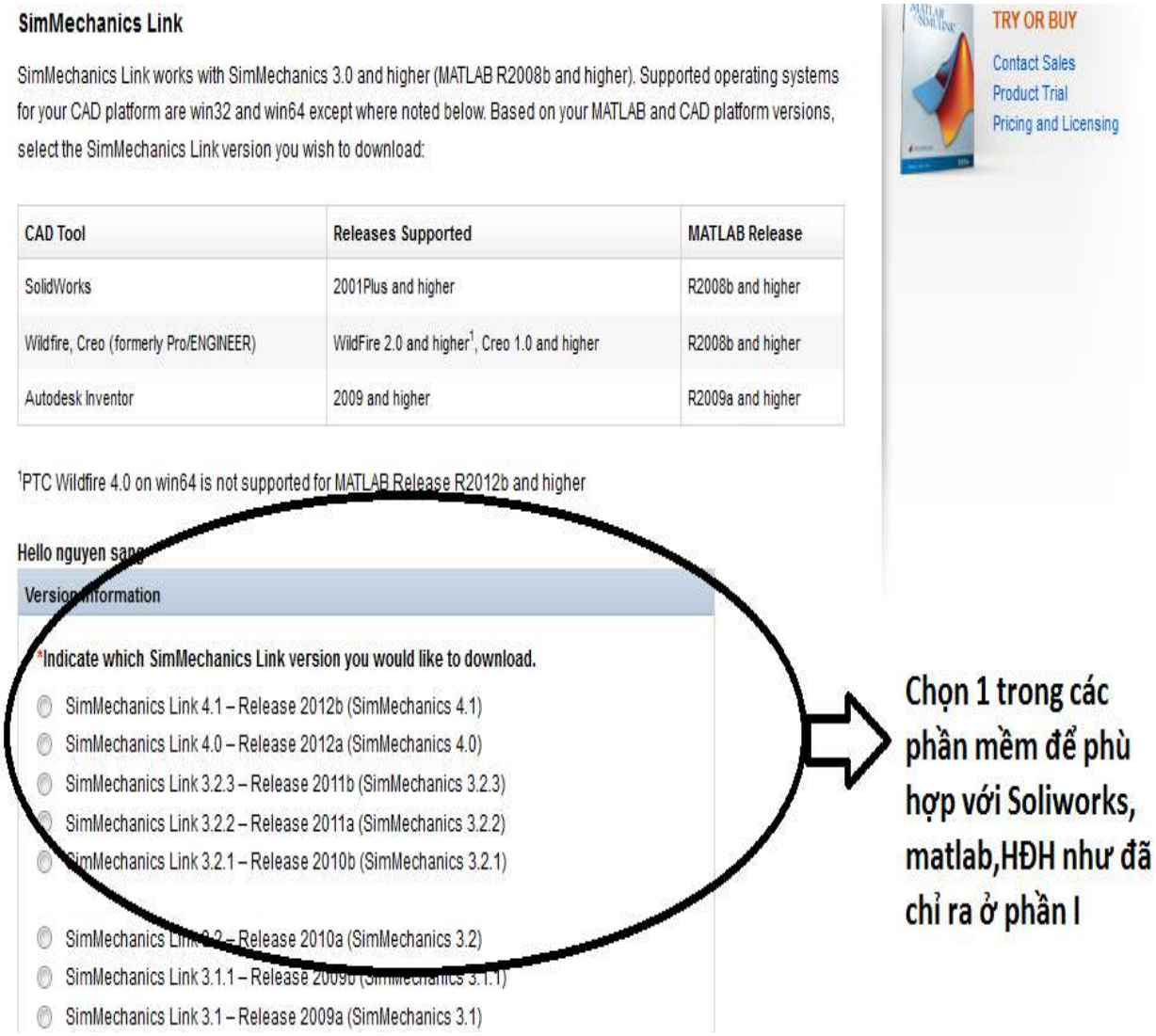
b)Download Simmechanics link

i)Sau khi đăng ký thành công tài khoản,sẽ có 1 email từ mathworks gửi vào hòm thư của bạn,cần phải kick vào link đó để xác nhận lại thông tin.

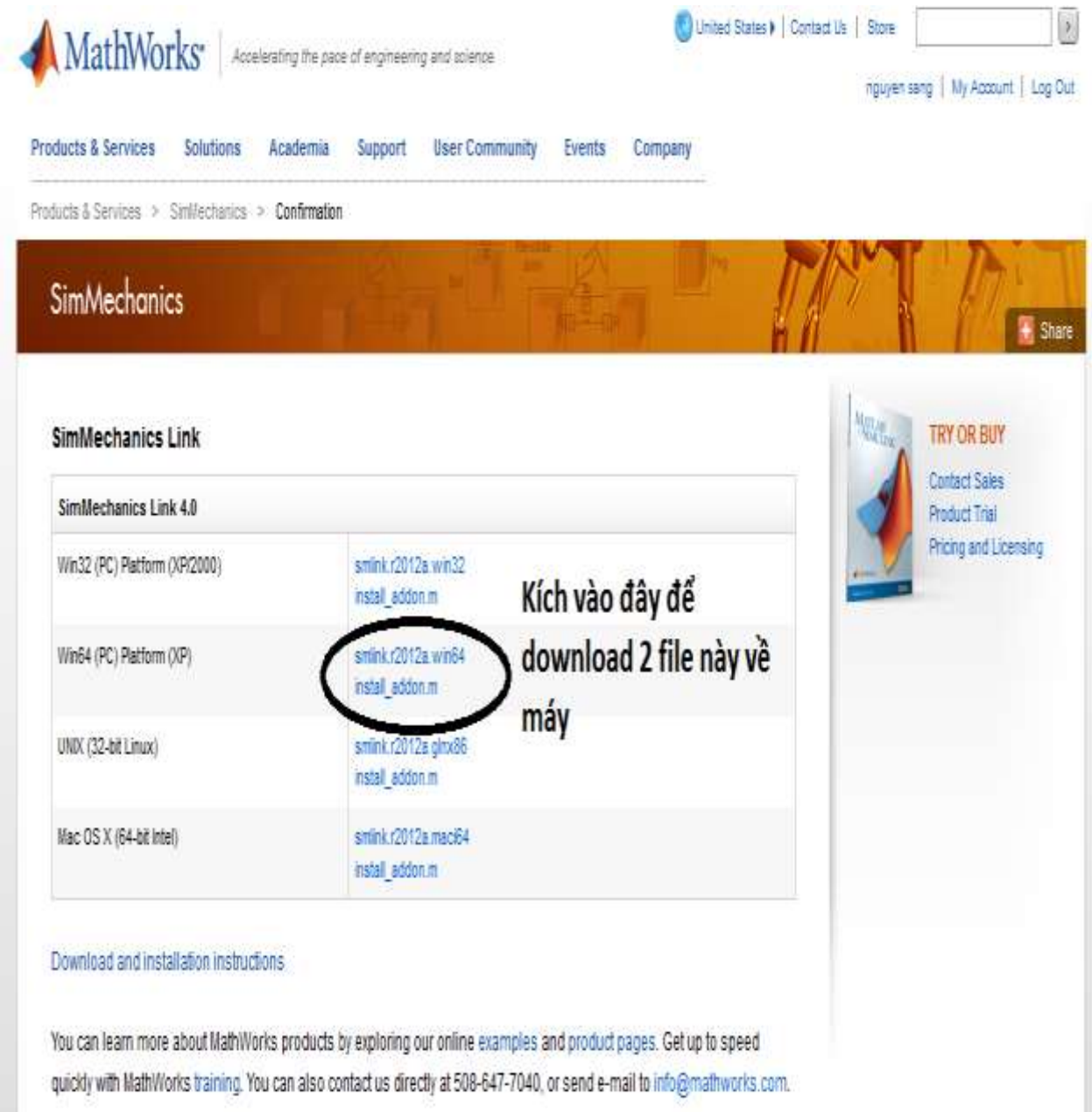
ii)Tiếp theo,khi đã có tài khoản,ta đăng nhập và vào

http://www.mathworks.com/products/simmechanics/download_smlink.html

Ở đây,ta kick chọn



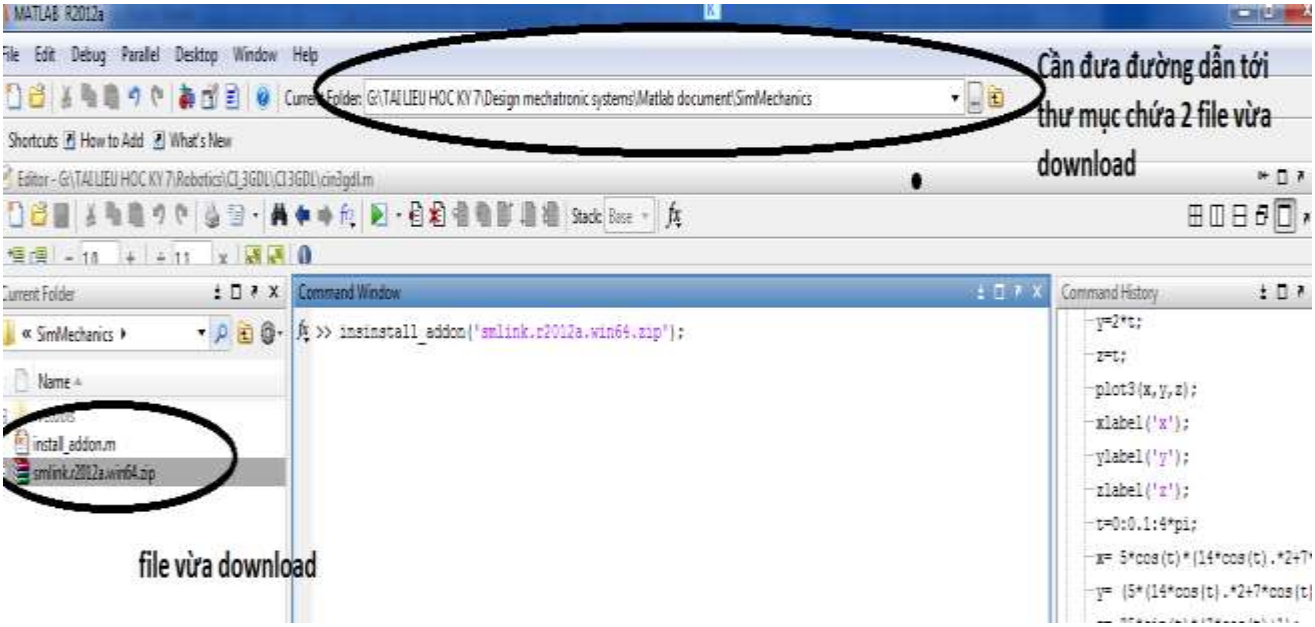
Sau khi tích vào phần cần chọn,ta sẽ nhấn nút submit ở dưới cùng trang Web.



c)Cài đặt Simmechanics Link

i)Chạy Matlab
ii)Tại dòng lệnh của matlab nhập
>>install_addon('<tên file Simmechanics vừa down>.zip');

Cụ thể ở đây là lệnh: >>install_addon('smlink.r2012a.win64.zip');Sau đó nhấn enter,một thông báo hiện ra cài đặt thành công.



Chú ý là cần đưa đường dẫn tới thư mục chứa 2 file vừa download

Như vậy ta đã cài đặt xong Simmechanics Link

III)Cài đặt add-in trong Solidworks

a)Trong matlab

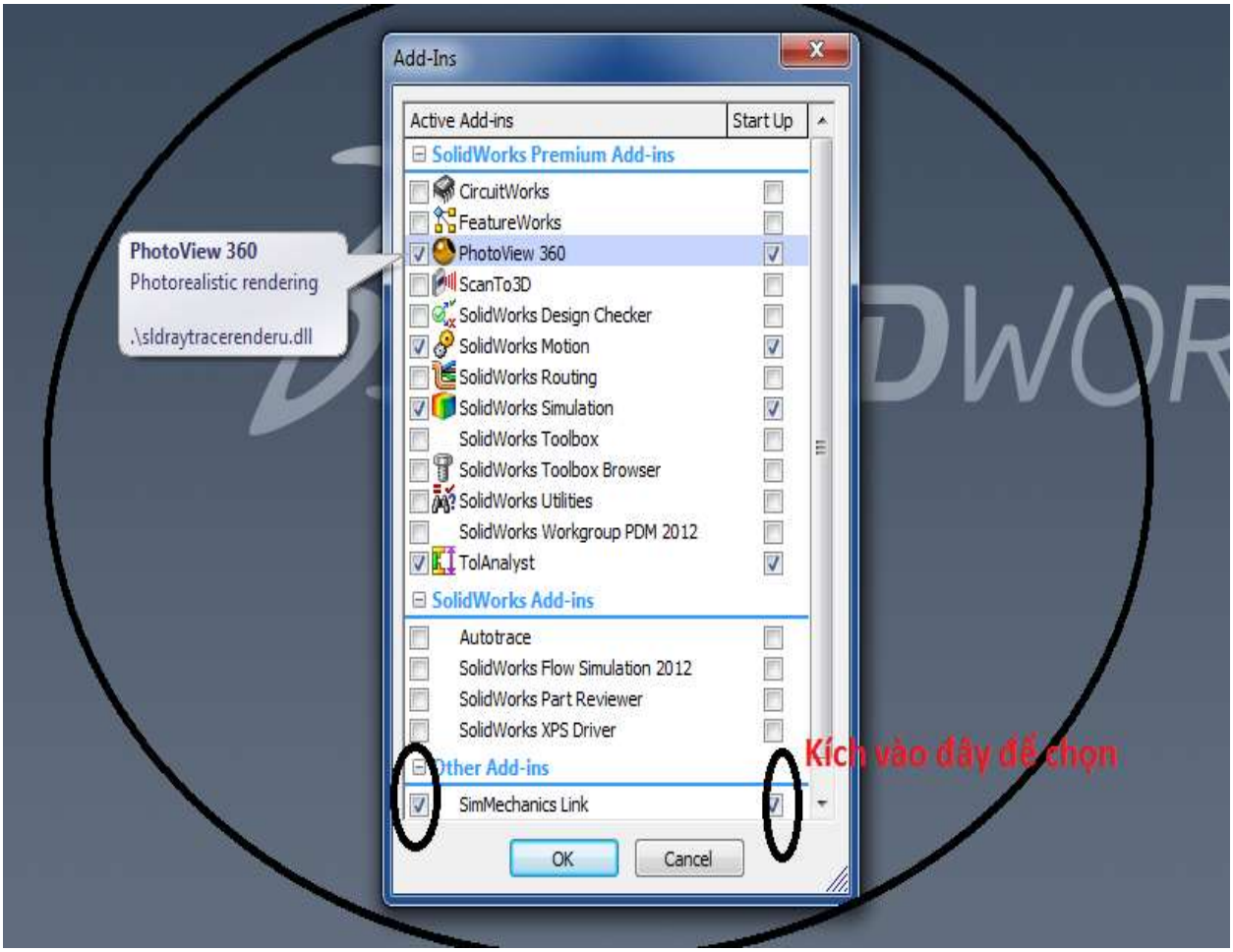
Tại dòng lệnh ta nhập lệnh

>>smlink_linksw;

Nhấn enter và đợi hiện ra thông báo Registering….

b)Trong Solidworks

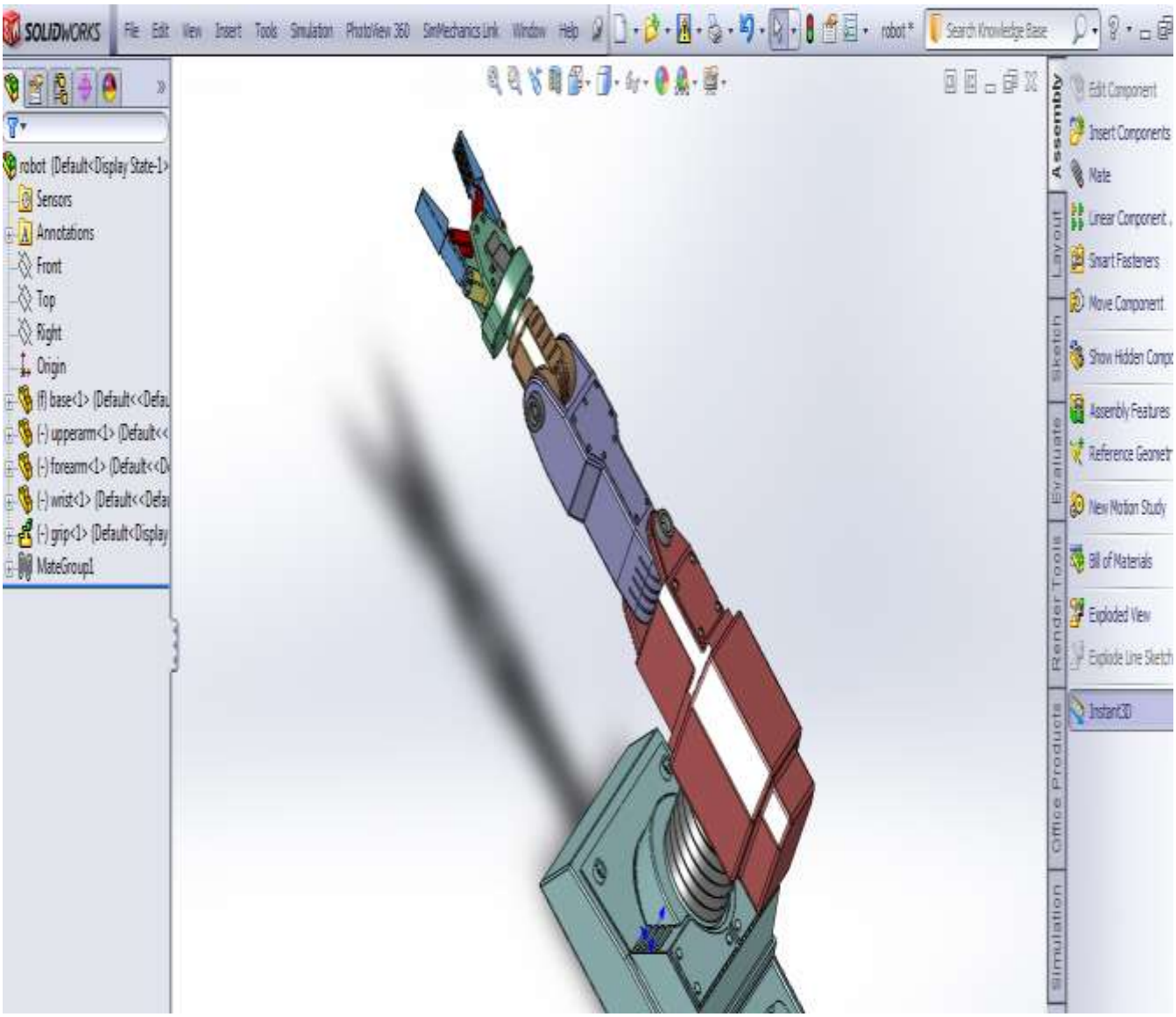
Bây giờ bắt đầu chạy Solidworks,ta vào Tool>Addins



Như vậy ta đã cài đặt thành công!

IV)Ví dụ về mô hình Robot

Bước 1:Trong solidworks,ta tiến hành vẽ 1 file assembly,sau đó chọn save as,ở ô Save as type chọn Simmechanics Link(*.xml)

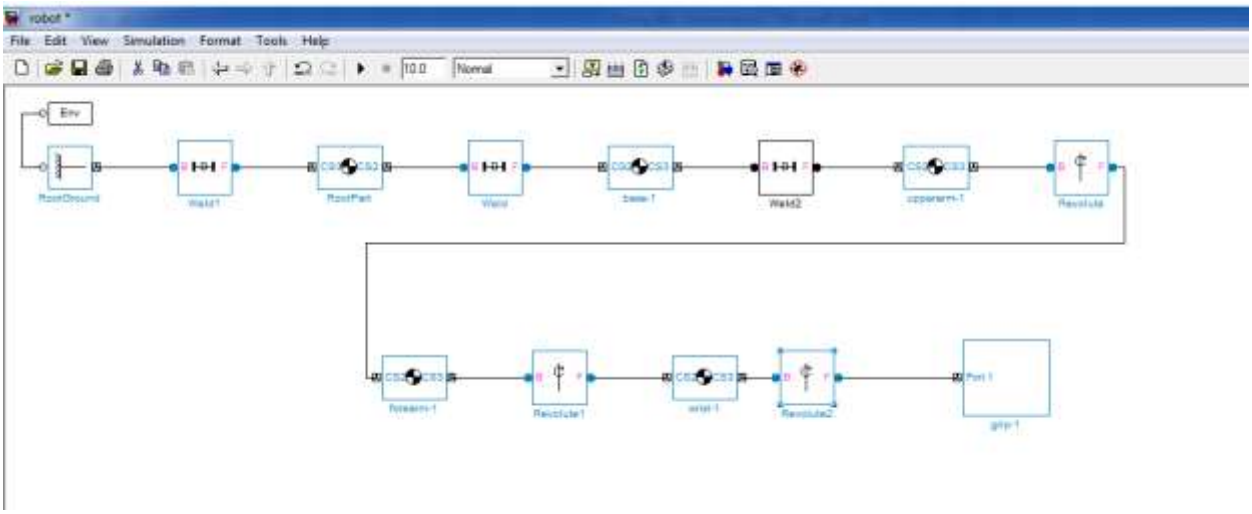


Bước 2:Trong matlab,tại dòng lệnh

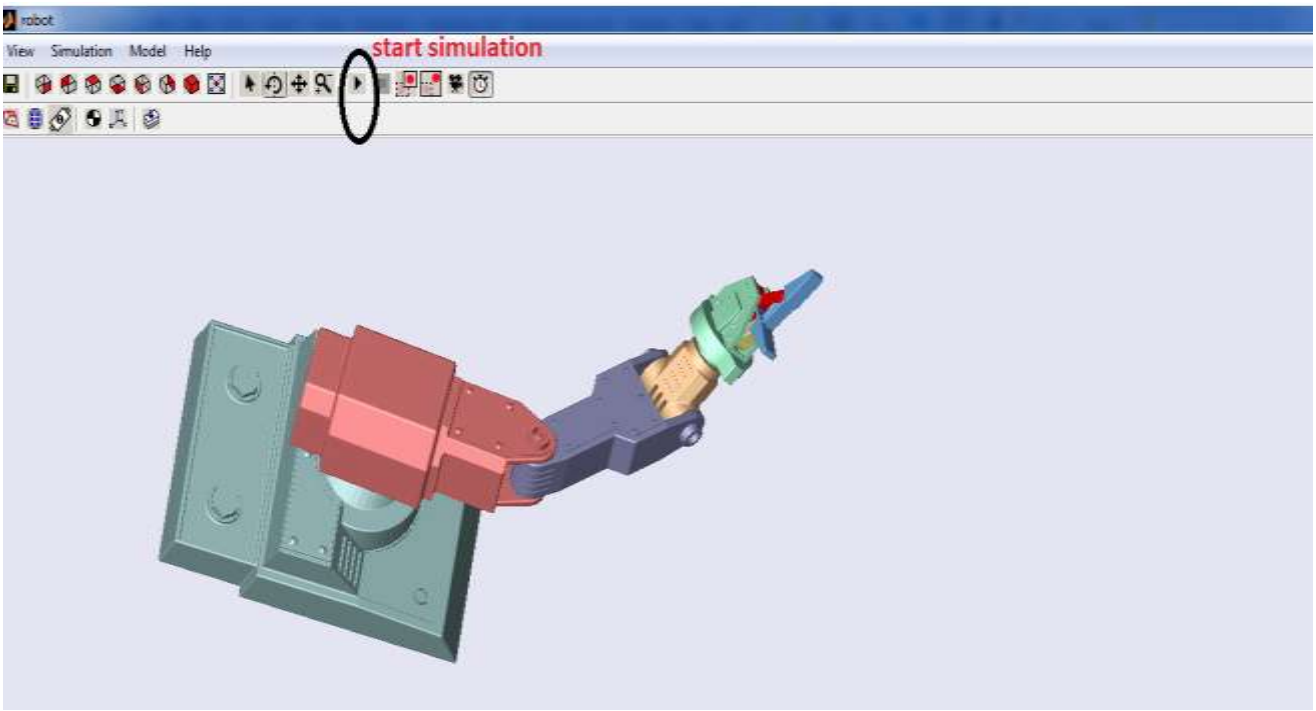
>>Mech_import(‘<tên file assembly>’);

Ví dụ :>>mech_import(‘robot’);

Sau đó enter,đợi khoảng 1 phút,ta sẽ có sơ đồ khối của file assembly vừa vẽ trong matlab(chú ý là đường dẫn matlab tới thư mục chứa file .xml vừa lưu).



Kích vào nút start simulation ta sẽ có hình ảnh của file Assembly trong màn hình đồ họa của Matlab như hình vẽ:



Việc tiếp theo là lấy các khối trong Simmechacnics và Simulink để thực hiện quá trình tính toán và mô phỏng.