MỤC LỤC

CHƯƠNG I
TỔNG QUAN VỀ HỆ THỐNG ĐIỀU KHIỂN SỐ ĐỘNG CƠ KĐB 3 PHA

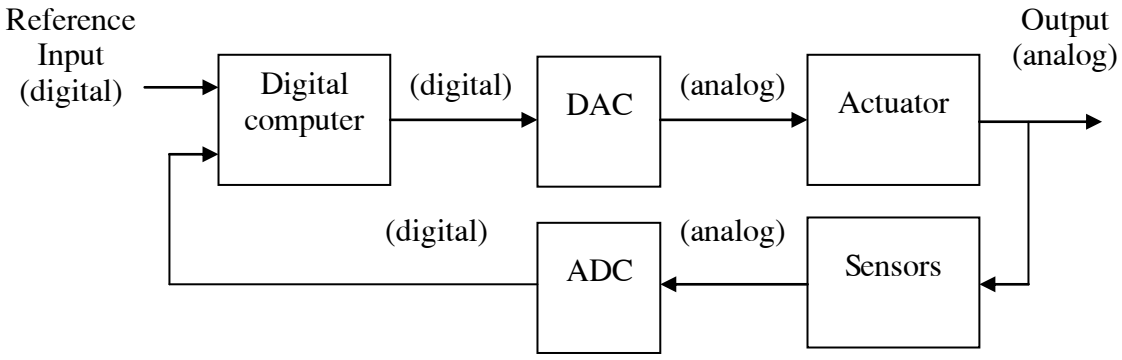
3. GIỚI THIỆU HỆ THỐNG ĐIỀU KHIỂN SỐ ĐCKĐB 3 PHA

1.1. Sơ đồ cấu trúc hệ thống điều khiển số

Các hệ thống điều khiển bằng máy tính (điều khiển số) ngày càng đ- ợc sử dụng rộng rãi trong công nghiệp. Chúng đóng một vai trò quan trọng trong việc điều khiển các quá trình công nghệ, nơi đòi hỏi sự kết hợp giữa máy tính với cơ cấu chấp hành để thực hiện một loạt các nhiệm vụ khác nhau.

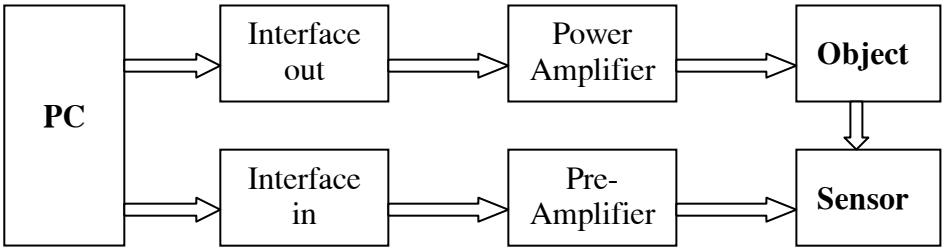
Việc sử dụng máy tính số nh- là một thiết bị bù (*compensator*) hay một thiết bị điều khiển (*controller*) đã phát triển suốt hơn hai thập kỷ qua bởi sự hiệu quả và độ tin cậy ngày càng cao của nó. Hình 1 d- ới đây là ví dụ cho sơ đồ khối của một hệ thống điều khiển số mạch đơn. Máy tính số trong hệ thống này có nhiệm vụ nhận sự sai khác giữa tín hiệu đặt với tín hiệu phản hồi về dạng số và thực hiện việc tính toán để đ- a ra tín hiệu điều khiển dạng số. Máy tính có thể đ- ợc lập trình để với đầu ra đó, chất l- ợng của hệ thống đạt đ- ợc hoặc gần đạt đ- ợc chất l- ợng mong muốn. Nhiều máy tính còn có thể nhận và thao tác với một số đầu vào, do đó một hệ thống điều khiển số th- ờng có thể là một hệ thống đa biến.

Máy tính nhận và xử lý các tín hiệu dạng số, trái ng- ợc với các tín hiệu liên tục. ***Một hệ thống điều khiển số sử dụng tín hiệu số và máy tính để điều khiển một quá trình***. Do đó số liệu đo sẽ đ- ợc chuyển đổi từ dạng t- ơng tự sang dạng số bằng bộ biến đổi t- ơng tự - số (ADC - Analog to Digital Converter) nh- đ- ợc chỉ ra trên hình 1. Sau khi xử lý các đầu vào, máy tính sẽ đ- a ra đầu ra dạng số và sau đó tín hiệu này đ- ợc chuyển đổi sang dạng t- ơng tự nhờ bộ biến đổi số - t- ơng tự (DAC - Digital to Analog Converter).



Hình 1: Ví dụ về sơ đồ khối của một hệ thống điều khiển số

Một cách tổng quát, ta có sơ đồ khối của hệ thống điều khiển số (HTĐKS) nh- hình 2.



Hình 2: Sơ đồ khối tổng quát của hệ thống điều

1.2. Các ph□ơng pháp điều khiển động cơ không đồng bộ ba pha

Động cơ không đồng bộ (ĐCKĐB) ba pha đ- ợc sử dụng rộng rãi trong công nghiệp, từ công suất nhỏ đến công suất trung bình và chiếm tỷ lệ rất lớn so với những động cơ khác. Ưu điểm của nó là dễ chế tạo, vận hành an toàn, sử dụng nguồn áp trực tiếp từ l- ới điện xoay chiều 3 pha. Tuy nhiên, tr- ớc đây, các hệ thống truyền động ĐCKĐB có điều chỉnh tốc độ lại chiếm tỷ lệ rất nhỏ do việc điều chỉnh tốc độ ĐCKĐB khó khăn hơn ĐC một chiều. Trong thời gian gần đây, do việc phát triển mạnh công nghiệp chế tạo bán dẫn công suất và kỹ thuật điện tử tin học, ĐCKĐB mới khai thác đ- ợc các - u điểm của mình và dần có xu h- ớng thay thế cho ĐC một chiều trong các hệ truyền động.

Để điều chỉnh tốc độ ĐCKĐB 3 pha, tr- ớc hết ta viết lại ph- ơng trình đặc tính cơ :

$$M = \frac{3U_1^2 R_2'}{\omega_1 s\left[\left(R_1 + \frac{R_2'}{s}\right)^2 + X_{nm}^2\right]} \qquad (1)$$

trong đó :

ω_1 _ tốc độ góc của từ tr- ờng quay

$\omega_1 = \frac{2\pi f_f}{p}$ với f_1 _ tần số của điện áp stator

p _ số đôi cực từ

U_1 _ trị số hiệu dụng của điện áp pha stator

R_1 _ điện trở của cuộn dây stator

R_2' _ điện trở rotor đã quy đổi về stator

X_{nm} _ điện kháng ngắn mạch

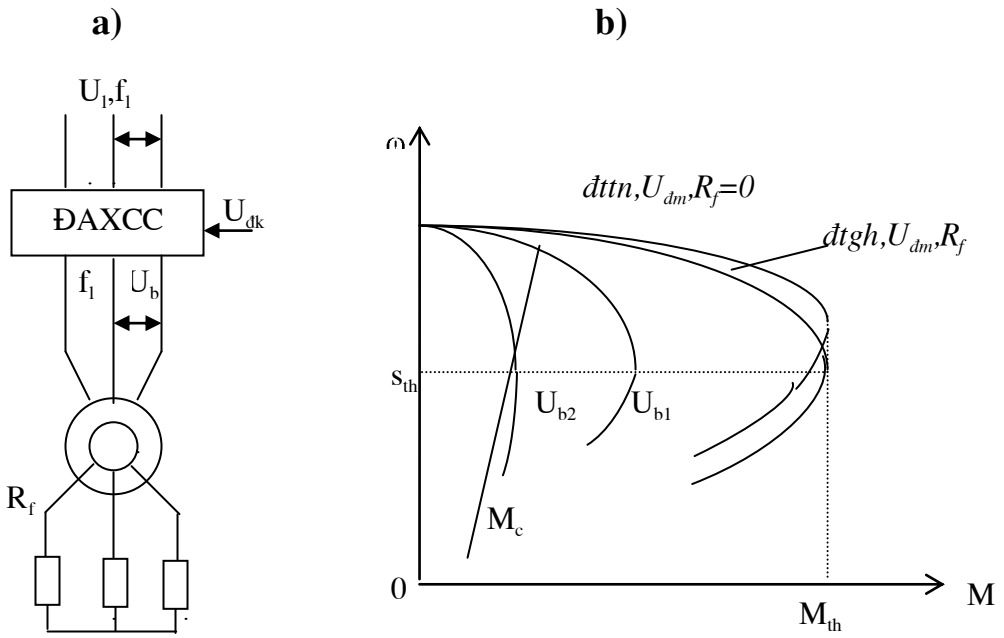
s _ hệ số tr- ợt của động cơ

$s = \frac{\omega_1 - \omega}{\omega_1}$ với ω là tốc độ góc của động cơ

Ph- ơng trình (1) cho thấy M=f(s) phụ thuộc vào các đại l- ợng U_1, ω_1, R_2'. T- ơng ứng với các đại l- ợng đó ta có ph- ơng pháp điều chỉnh điện áp động cơ, ph- ơng pháp điều chỉnh điện trở mạch rotor và ph- ơng pháp điều chỉnh tần số. Sau đây chúng ta sẽ xem xét lần l- ợt từng ph- ơng pháp và ý t- ởng thực hiện chúng trong các HTĐKS.

1.2.1. Ph□ơng pháp điều chỉnh điện áp ĐCKĐB ba pha (giữ nguyên tần số)

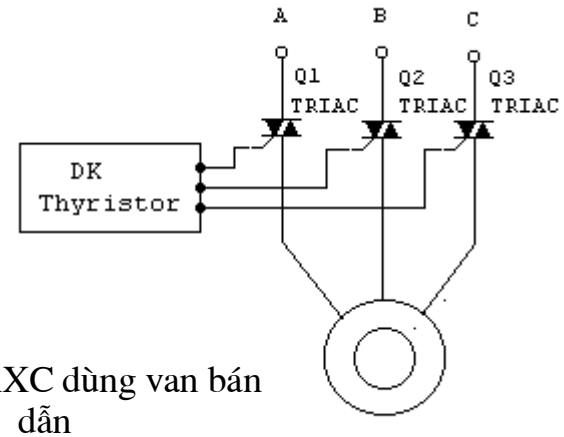
Nh- đã trình bày ở phần trên, momen của ĐCKĐB ba pha tỷ lệ với bình ph- ơng điện áp đặt lên stator. Điều đó có nghĩa là nếu thay đổi điện áp stator thì mô men của động cơ sẽ thay đổi đi bình ph- ơng lần. Dựa vào đó có thể điều khiển đ- ợc tốc độ của ĐCKĐB ba pha. Sơ đồ khối nguyên lí và đặc tính cơ điều chỉnh của ph- ơng pháp này đ- ợc chỉ ra trên hình 3.



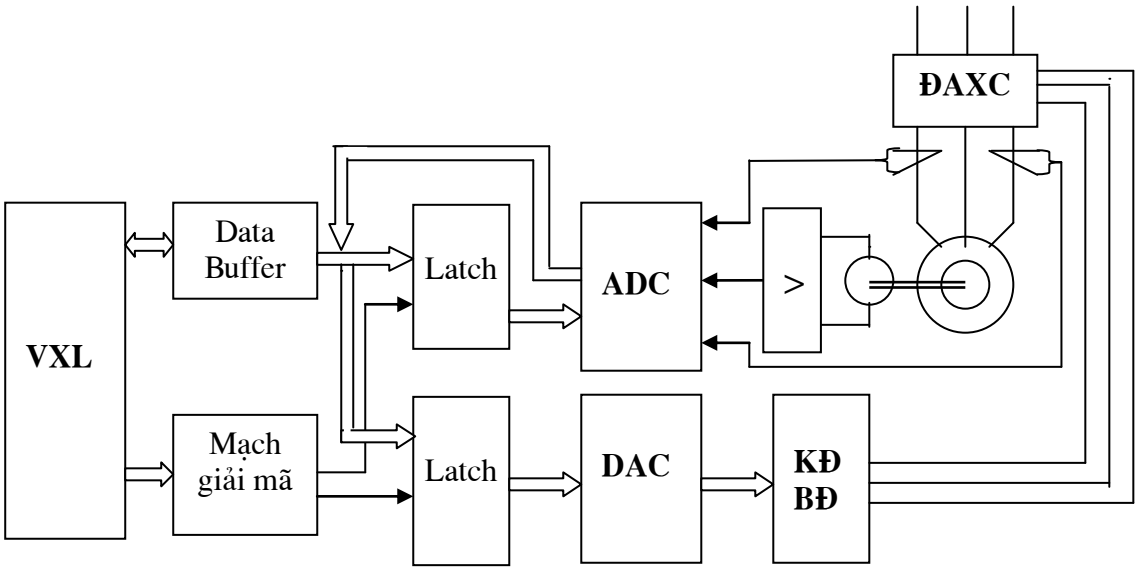
Hình 3: Điều chỉnh điện áp ĐCKĐB: a) Sơ đồ khối nguyên lý.
b) Đăc tính cơ điều chỉnh.

Để điều chỉnh điện áp ĐCKĐB, phải dùng các bộ biến đổi điện áp xoay chiều. Bộ biến đổi điện áp xoay chiều phổ biến nhất hiện nay là sử dụng van bán dẫn có cực điều khiển (hình 4). Bằng cách thay đổi các tín hiệu điều khiển đóng mở các van bán dẫn, ta có thể điều chỉnh đ- ợc điện áp stator, từ đó thay đổi đ- ợc tốc độ động cơ. Việc phát ra xung điều khiển hoàn toàn có thể thực hiện đ- ợc bằng máy tính. Tín hiệu từ vi xử lý qua các bộ biến đổi

có thể đ- a tới khối điều khiển Thyristor. Đồng thời tín hiệu phản hồi dòng và phản hồi tốc độ của động cơ đ- ợc đ- a về vi xử lý thông qua bộ biến đổi để vi xử lý tính toán đ- a ra tín hiệu điều khiển. Hình 5 sau đây sẽ minh hoạ cho các diễn giải trên.



Hình 4: ĐAXC dùng van bán dẫn

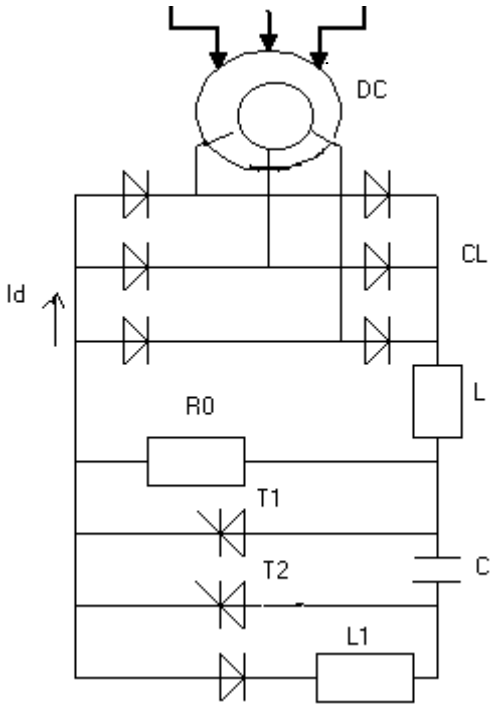


Hình 5: Sơ đồ khối của ph- ơng pháp điều chỉnh điện áp stator

Ph- ơng pháp điều chỉnh điện áp stator có nh- ợc điểm là gây ra tổn thất năng l- ợng, nhất là khi điện áp không sin sẽ sinh ra dòng Fucô làm nóng động cơ.

1.2.2. Điều chỉnh tốc độ động cơ không đồng bộ bằng điều chỉnh điện trở mạch rotor

a. Sơ đồ mạch nguyên lý

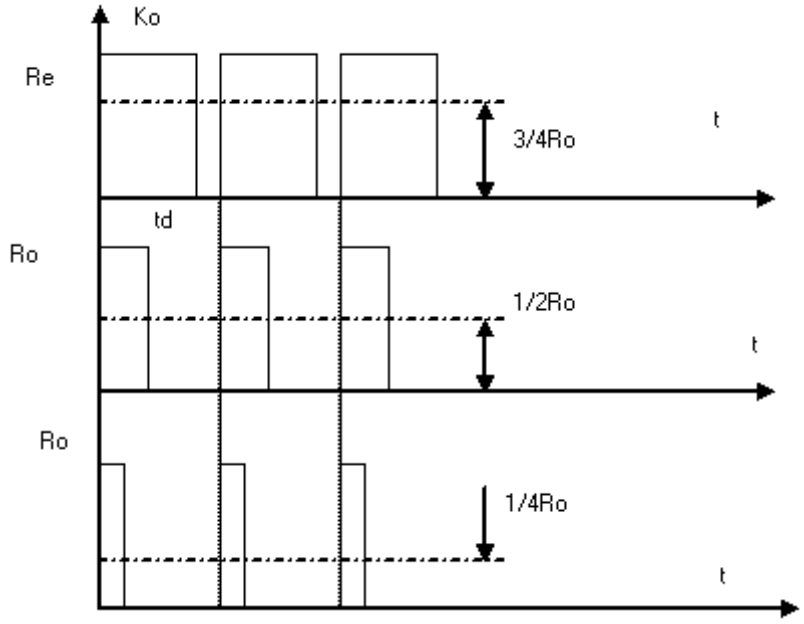


b. Nguyên lý điều chỉnh

Điều chỉnh điện trở rotor bằng ph- ơng pháp xung. Khi điện áp đ- ợc chỉnh l- u bởi cầu diode, đ- ợc cấp vào mạch điều chỉnh gồm có điện trở R_0 nối song song với một khoá bán dẫn T_1. Khoá này sẽ đ- ợc đóng cắt theo chu kỳ để điều chỉnh giá trị trung bình của điện trở toàn mạch.

c. Ph□ơng pháp điều chỉnh

Khi khoá T_1 đóng, điện trở R_0 bị loại ra khỏi mạch, dòng rotor tăng lên, khi khoá T_1 mở điện trở R_0 lại đ- ợc đ- a vào mạch, dòng điện rotor giảm. Nhờ có điện cảm L mà dòng rotor coi nh- không đổi. Với tần số đóng ngắt nhất định, ta có một giá trị điện trở t- ơng đ- ơng R_e trong mạch.



Ph- ơng pháp điều chỉnh này rõ ràng chỉ áp dụng đ- ợc với động cơ không đồng bộ rotor dây quấn, trong khi động cơ không đồng bộ rotor lồng sóc đ- ợc dùng phổ biến hơn bởi cấu tạo đơn giản, độ tin cậy cao và không cần bảo d- ỡng. Vì vậy ta không cần quan tâm đến ph- ơng pháp này lắm.

1.2.3. Ph□ơng pháp điều chỉnh tần số

Ph- ơng pháp điều chỉnh điện áp stator và điều chỉnh điện trở rotor áp dụng chủ yếu cho việc điều khiển ĐCKĐB ba pha rotor dây quấn. Việc điều khiển ĐCKĐB 3 pha rotor lồng sóc tr- ớc đây rất khó thực hiện. Ngày nay, sự phát triển mạnh mẽ của điện tử công suất lớn và kỹ thuật vi xử lý đã mở ra khả năng ứng dụng có hiệu quả ph- ơng pháp điều khiển động cơ lồng sóc bằng thiết bị biến tần. Ph- ơng pháp này cho phép điều chỉnh tốc độ động cơ trong phạm vi rộng với độ chính xác cao.

Khi điều chỉnh tần số, để duy trì chế độ làm việc tốt nhất, phải điều chỉnh cả điện áp stator. Đối với hệ thống biến tần nguồn áp th- ờng có yêu cầu giữ cho khả năng quá tải về momen là không đổi:

$$\lambda = \frac{M_{th}}{M} = const$$

trong đó:

λ _ hệ số quá tải mô men
M_{th} _ mô men tới hạn

Với đặc tính cơ dạng gần đúng của máy sản xuất là :

$$M_c = M_{dm}\left(\frac{\omega}{\omega_{dm}}\right)^x$$

M_c _ mô men ứng với tốc độ ω
M_{dm} _ mô men ứng với tốc độ định mức ω_{dm}
x_ hệ số tuỳ thuộc vào loại máy sản xuất

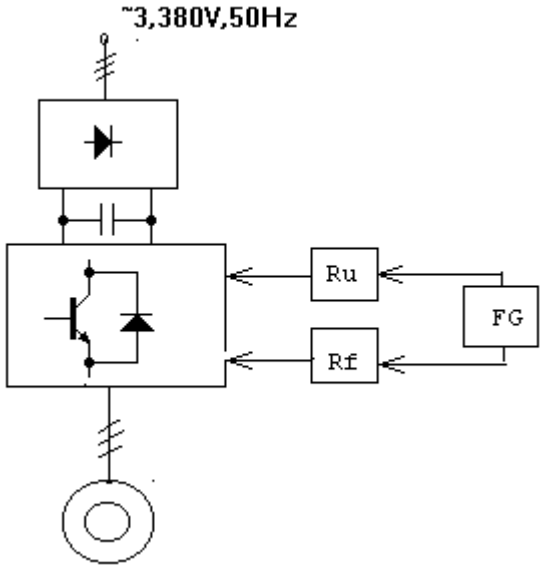
Ta có luật điều chỉnh điện áp là :

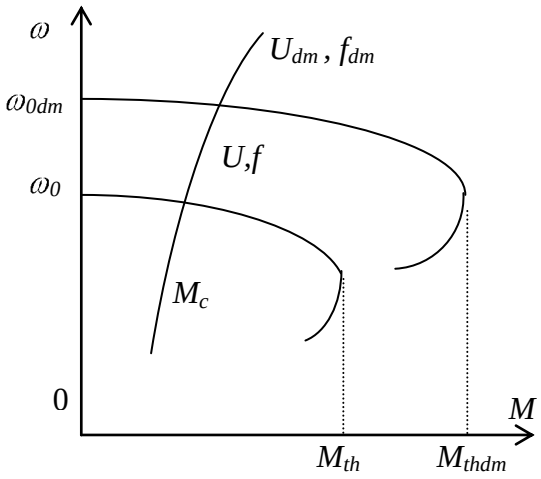
$$\frac{U_1}{U_{1dm}} = \left(\frac{f_1}{f_{1dm}}\right)^{\left(1+\frac{x}{2}\right)}$$

hay ở dạng đơn vị không tên:

$$u_1^* = f_1^{*\left(1+\frac{x}{2}\right)}$$

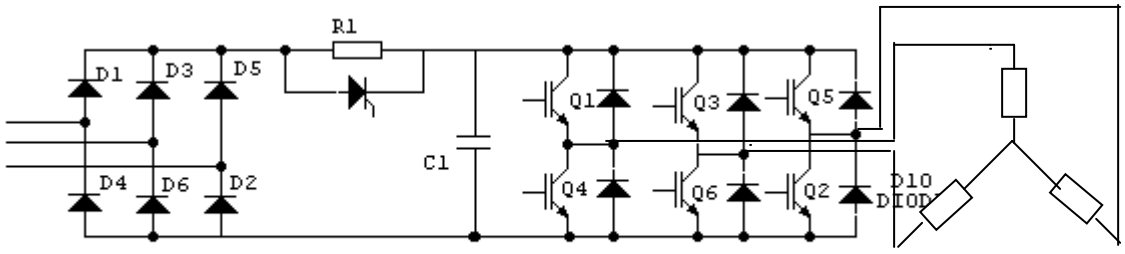
Sơ đồ khối nguyên lý và đặc tính cơ đ- ợc cho trong hình d- ới.





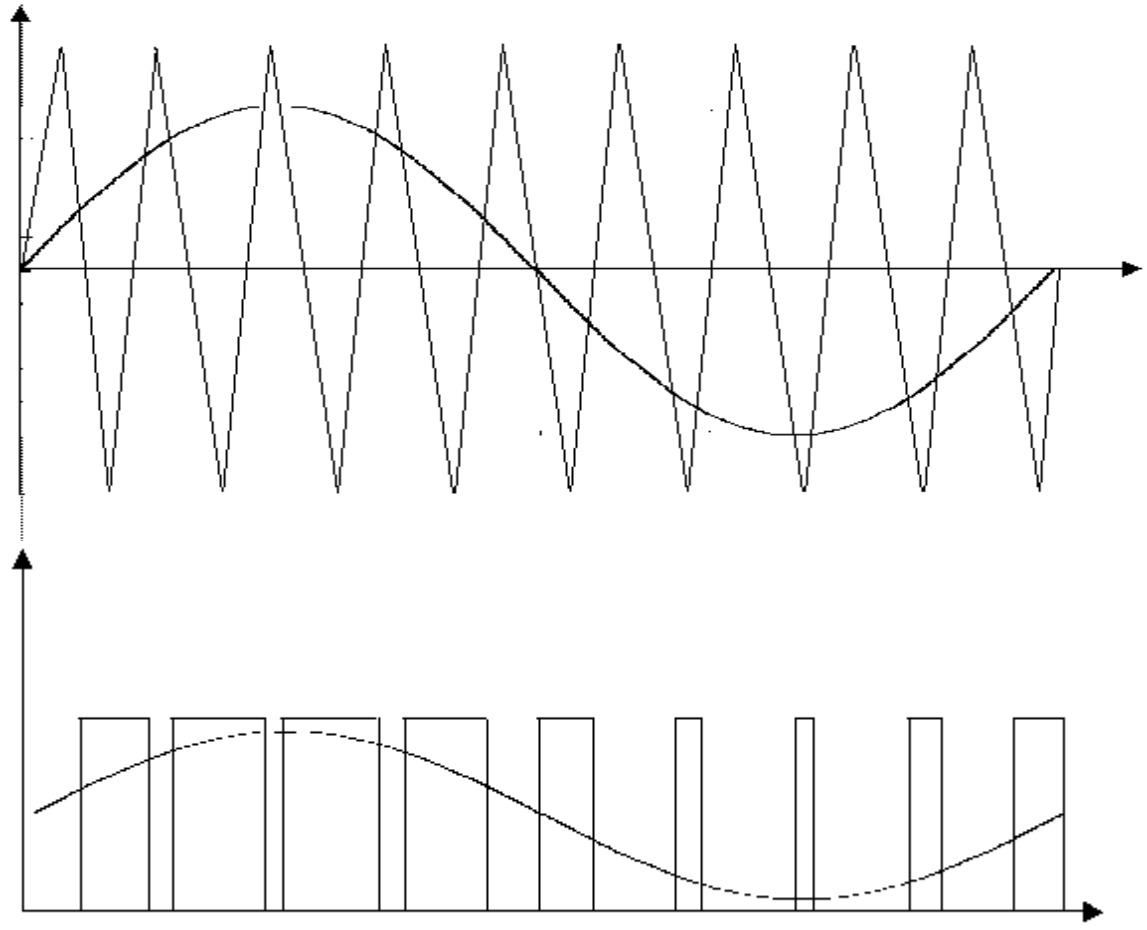
FG _ máy phát hàm
Ru _ bộ điều chỉnh điện áp
Rf _ bộ điều chỉnh tần số

Ở đây ta sẽ xét đến bộ biến tần nguồn áp làm việc theo nguyên lý điều chế độ rộng xung (PWM –Pulse Width Modulation). Bộ biến tần này cho phép điều chỉnh đồng thời cả tần số và điện áp. Mặt khác, nó còn tạo ra đ- ợc điện áp và dòng điện gần nh- hình sin (hình 6)



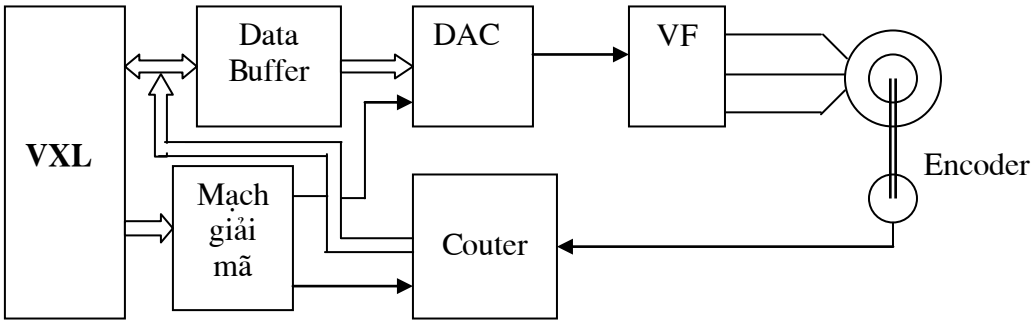
Hình 6: Sơ đồ bộ biến tần nguồn áp

Bằng ph- ơng pháp PWM ta có giản đồ điện thế và điện áp pha A nh- hình 7.



Hình 7: Ph- ơng pháp PWM

Hình 8 d- ới đây là sơ đồ khối của hệ thống điều khiển số dùng để điều khiển ĐCKĐB ba pha rotor lồng sóc sử dụng thiết bị biến tần (VF_Varied Frequency).



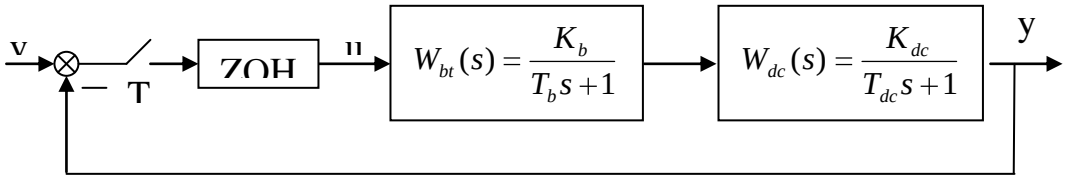
Hình 8: Mô hình hệ điều khiển số

Hệ thống điều khiển ĐCKĐB rotor lồng sóc bằng biến tần là hệ thống truyền động điện điều chỉnh có nhiều triển vọng ứng dụng. Việc nghiên cứu hệ thống này có nhiều xu h- ớng khác nhau. Nh- ợc điểm của nó là giá thành cao, phức tạp.

Theo xu h- ớng phát triển hiện nay của các hệ thống điều khiển truyền động điện và căn cứ vào yêu cầu cụ thể của đề bài, ph- ơng pháp điều khiển ĐCKĐB ba pha rotor lồng sóc dùng biến tần sẽ đ- ợc sử dụng trong bài tập này. Sơ đồ khối của hệ thống điều khiển sẽ đ- ợc xây dựng nh- trong hình 8.

4. PHÂN TÍCH HỆ THỐNG ĐIỀU KHIỂN SỐ

Khi ch- a có bộ điều khiển trong hệ thống, sơ đồ khối của hệ thống nh- sau:



Trong đó:

- $W_{bt}(s)$ là hàm truyền của biến tần, với: $K_b = 65$; $T_b = 0.02$ (s)
- $W_{dc}(s)$ là hàm truyền của động cơ, với: $K_{dc} = 6 - 10$; $T_{dc} = 0.1$ (s)

Hằng số thời gian nhỏ nhất trong đối t- ợng là $T_b = 0.02$ (s) nên chu kì lấy mẫu T phải nhỏ hơn 0.02 (s). Căn cứ vào khả năng hoạt động của máy tính và điều kiện trên, chọn chu kì lấy mẫu T = 0.005 s = 5 ms.

Cho $K_{dc} = 9$.

2.1. Hàm truyền đạt và ph□ơng trình trạng thái của đối t□ợng:

Đối t- ợng điều khiển ở đây bao gồm biến tần và động cơ. Nh- vậy hàm truyền của đối t- ợng là $W_{dt}(s) = W_{bt}.W_{dc} = \frac{65K_{dc}}{(0.02s+1)(0.1s+1)}$.

Chuyển hàm truyền của đối t- ợng sang dạng rời rạc (miền Z) với chu kì lấy mẫu T = 0.005 s = 5 ms ta có:

$$W_{dt}(z) = \frac{z-1}{z}.Z\left\{\frac{W_{dt}(s)}{s}\right\}$$

$$Z\left\{\frac{W_{dt}(s)}{s}\right\} = Z\left\{\frac{65K_{dc}}{s(0.02s+1)(0.1s+1)}\right\} = 65K_{dc}.Z\left\{\frac{1}{s} + \frac{0.25}{s+50} - \frac{1.25}{s+10}\right\}$$

$$\Rightarrow Z\left\{\frac{W_{dt}(s)}{s}\right\} = 65K_{dc}\left[\frac{z}{z-1} + \frac{0.25z}{z-e^{-50T}} - \frac{1.25z}{z-e^{-10T}}\right]$$

Suy ra:

$$W_{dt}(z) = \frac{z-1}{z}.65K_{dc}\left[\frac{z}{z-1} + \frac{0.25z}{z-e^{-50T}} - \frac{1.25z}{z-e^{-10T}}\right] = 65K_{dc}\left[1 + \frac{0.25(z-1)}{z-e^{-50T}} - \frac{1.25(z-1)}{z-e^{-10T}}\right]$$

$$= 65K_{dc}\frac{(z-e^{-50T})(z-e^{-10T}) + 0.25(z-1)(z-e^{-10T}) - 1.25(z-1)(z-e^{-50T})}{(z-e^{-50T})(z-e^{-10T})}$$

Đặt $A = e^{-50T}$, $B = e^{-10T}$ ta có:

$$W_{dt}(z) = 65K_{dc}\frac{z^2-(A+B)z+AB+(z-1)(-z+1.25A-0.25B)}{(z-A)(z-B)}$$
$$= 65K_{dc}\frac{(0.25A-1.25B+1)z+(AB-1.25A+0.25B)}{z^2-(A+B)z+AB}$$

Thay T = 0.005s có: *A = 0.7788; B = 0.9512* và do đó:

$$\boxed{W_{dt}(z) = \frac{0.3705K_{dc}.z+0.3311K_{dc}}{z^2-1.73z+0.7408}} \quad (3.1)$$

Chuyển sang ph- ơng trình trạng thái:

$$\begin{cases} x(k+1) = \begin{bmatrix} 1.73 & -0.7408 \\ 1 & 0 \end{bmatrix}.x(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix}u(k) \\ y(k) = \begin{bmatrix} 0.3705K_{dc} & 0.3311K_{dc} \end{bmatrix}\underline{x}(k) \end{cases} \quad (3.2)$$

Hai công thức trên là hàm truyền đạt và ph- ơng trình trạng thái của đối t- ợng (bao gồm biến tần và động cơ) trong miền rời rạc với chu kì lấy mẫu là 5ms hay 0.005s.

2.2. Kiểm tra tính điều khiển đ□ợc và tính quan sát đ□ợc của đối t□ợng:

Ta đã xác định đ- ợc ph- ơng trình trạng thái của đối t- ợng điều khiển (công thức 3.2). Để kiểm tra tính điều khiển đ- ợc của đối t- ợng cần tính ma trận điều khiển đ- ợc:

$$P_d = [B_d, A_d.B_d]$$
$$= \left[\begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1.73 & -0.7408 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right]$$
$$= \begin{bmatrix} 1 & 1.73 \\ 0 & 1 \end{bmatrix}$$

Có *det{P_d} = 1 ≠ 0* suy ra *rank{P_d} = 2* do đó **đối t□ợng là điều khiển đ□ợc**.

Để kiểm tra tính quan sát đ- ợc của đối t- ợng cần xác định ma trận quan sát đ- ợc của đối t- ợng. Ta có:

$$N_d = \begin{bmatrix} C_d \\ C_d.A_d \end{bmatrix}$$
$$= \begin{bmatrix} 0.3705K_{dc} & 0.3311K_{dc} \\ 0.3705K_{dc} & 0.3311K_{dc} \end{bmatrix} . \begin{bmatrix} 1.73 & -0.7408 \\ 1 & 0 \end{bmatrix}$$
$$= K_{dc} \begin{bmatrix} 0.3705 & 0.3311 \\ 0.9721 & -0.2745 \end{bmatrix}$$

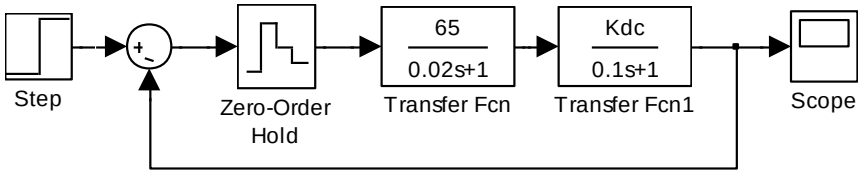
Suy ra $det\{N_d\} = -0,4235.K_{dc}^2$. Nếu $K_{dc} \neq 0$ thì $det\{N_d\} \neq 0$ và do đó $rank\{N_d\} = 2$ nên đối t- ợng quan sát đ- ợc. Thực tế thì luôn có $K_{dc} \neq 0$ nên **đối t ợng luôn quan sát đ ợc**. Tức là ta có thể khôi phục đ- ợc trạng thái của đối t- ợng thông qua quan sát đầu ra của đối t- ợng (đầu vào đ- ơng nhiên luôn biết). Đó là cơ sở để sau này có thể thiết kế đ- ợc bộ quan sát trạng thái phục vụ cho hồi tiếp trạng thái.

2.3. Xét ổn định của đối t ợng:

Phần này sẽ xét ổn định của đối t- ợng, nghĩa là xét ổn định của một hệ thống hở trong đó không có bộ điều khiển. Công thức (3.1) đã cho biết hàm truyền đạt rời rạc của đối t- ợng. Ph- ơng trình đặc tính của đối t- ợng là $z^2 - 1.73z + 0.7408 = 0$. Giải ph- ơng trình bậc hai này ta có các điểm cực của đối t- ợng là $z_1 = 0.9512$ và $z_2 = 0.7788$. Các điểm cực này đều nằm bên trong đ- ờng tròn đơn vị, do vậy đối t- ợng là ổn định, tức là hệ thống hở là ổn định. Ngoài ra, do các điểm cực này đều thực nên không có quá điều chỉnh.

2.4. Xét ổn định của hệ thống kín khi ch a có bộ điều khiển:

Xét đối t- ợng trong một hệ thống kín nh- ng ch- a có bộ điều khiển *(xem hình vẽ)*. Cần xét ổn định của hệ thống này.



Hàm truyền rời rạc của đối t- ợng đ- ợc cho trong công thức (3.1):

$$W_{dt}(z) = \frac{0.3705K_{dc}.z + 0.3311K_{dc}}{z^2 - 1.73z + 0.7408}$$

Hàm truyền đạt của hệ thống có hồi tiếp âm là:

$$W_{ht}(z) = \frac{W_{dt}(z)}{1+W_{dt}(z)}$$
$$= \frac{0.3705K_{dc}z + 0.3311K_{dc}}{z^2 - 1.73z + 0.7408 + 0.3705K_{dc}z + 0.3311K_{dc}}$$
$$= \frac{0.3705K_{dc}z + 0.3311K_{dc}}{z^2 + (0.3705K_{dc} - 1.73)z + (0.7408 + 0.3311K_{dc})}$$

Với $K_{dc} = 9$ ta có:

$$W_{ht}(z) = \frac{3.3345z + 2.9799}{z^2 + 1.6045z + 3.7207}$$

Ph- ơng trình đặc tính của hệ thống là $z^2+1.6045z+3.7207=0$

Giải ph- ơng trình trên ta có các điểm cực của hệ thống là z_1 = *-0.8023 + 1.7542i* và z_2 = *-0.8023 - 1.7542i*. Modun của hai điểm cực trên đều là $\sqrt{(-0.8023)^2 + 1.7542^2} = 1.9289 > 1$, tức là cả hai điểm cực đều nằm ngoài đ- ờng tròn đơn vị, do đó **hệ kín không ổn định**.

2.5. Quá trình quá độ của hệ thống kín khi ch□a có bộ điều khiển:

Vẫn tiếp tục xét hệ kín chỉ chứa đối t- ợng mà không chứa bộ điều khiển *(mô hình trên)*. Ta đã xác định đ- ợc hàm truyền đạt rời rạc của hệ thống kín là:

$$W_{ht}(z) = \frac{0.3705K_{dc}z + 0.3311K_{dc}}{z^2 + (0.3705K_{dc} - 1.73)z + (0.7408 + 0.3311K_{dc})}$$
$$= \frac{0.3705K_{dc}z^{-1} + 0.3311K_{dc}z^{-2}}{1 + (0.3705K_{dc} - 1.73)z^{-1} + (0.7408 + 0.3311K_{dc})z^{-2}}$$
$$= \frac{Y(z)}{W(z)}$$

Chuyển sang ph- ơng trình sai phân ta có:

$K_{dc}(0.3705z^{-1} + 0.3311z^{-2}).W(z) = [1 + (0.3705K_{dc} - 1.73)z^{-1} + (0.7408 + 0.3311K_{dc})z^{-2}].Y(z)$

$\Rightarrow K_{dc}[0.3705.w(k-1) + 0.3311.w(k-2)] =$
$= y(k) + (0.3705K_{dc} - 1.73).y(k-1) + (0.7408 + 0.3311K_{dc}).y(k-2)$

$\Leftrightarrow y(k) = K_{dc}[0.3705.w(k-1) + 0.3311.w(k-2)] -$
$- (0.3705K_{dc} - 1.73).y(k-1) - (0.7408 + 0.3311K_{dc}).y(k-2)$

Với $K_{dc} = 9$, ph- ơng trình sai phân trở thành:

$$y(k) = 3,3345.w(k-1) + 2,9799.w(k-2) – 1,6045.y(k-1) – 3,7207.y(k-2)$$

Nếu tín hiệu vào w(k) là Step thì ta có:

w(k)=1 khi $k \geq 0$

& w(k)=0 khi $k < 0$

Thực hiện ph- ơng trình sai phân trên bằng ch- ơng trình C++ (viết trên Turbo C) ta vẽ đ- ợc quá trình quá độ của hệ kín khi đầu vào w(k) là Step. Ch- ơng trình C++ nh- sau:

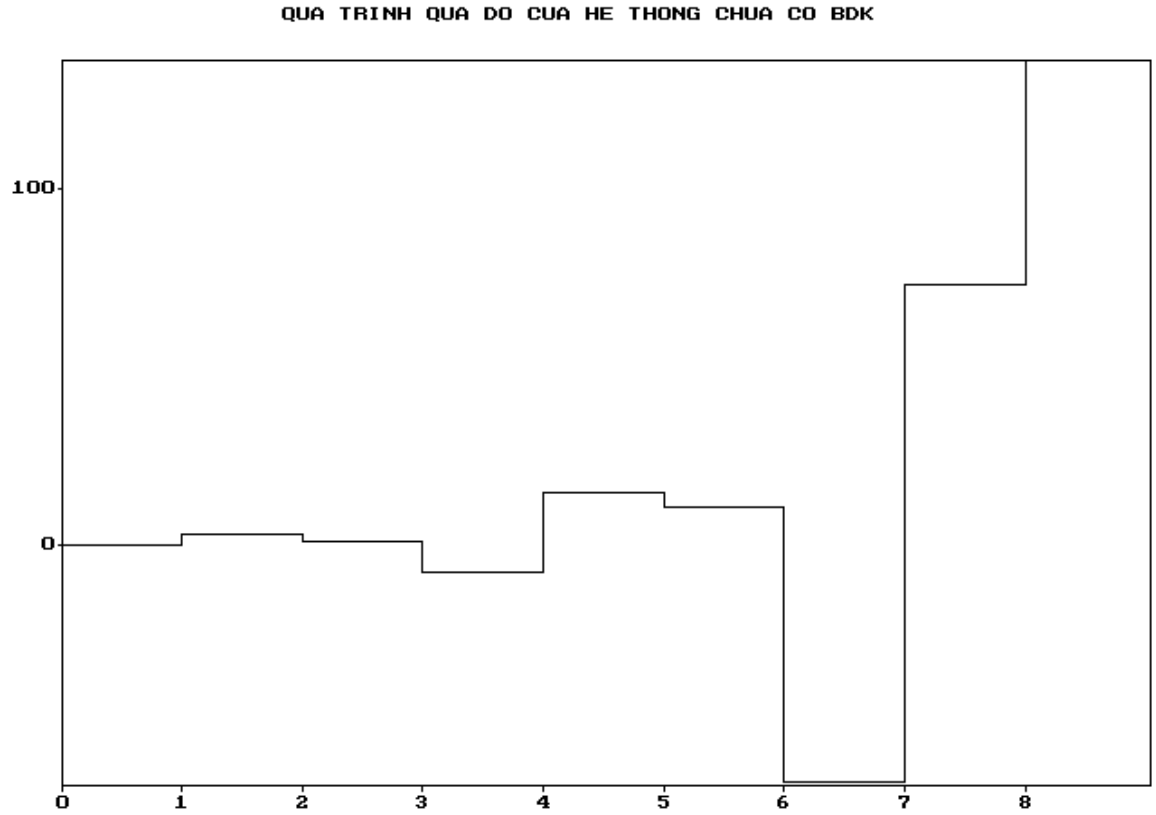
Ch□ơng trình C++ vẽ quá trình quá độ trên máy tính

```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
#include <values.h>
#include <stdlib.h>

float y_old[2] = {0.0,0.0};

inline int w(register int k) {
     return (k<0?0:1);
}

float y(register int k) {
    if (k<0) return (0);
    float tmp = 3.3345*w(k-1) + 2.9799*w(k-2) -
1.6045*y_old[0] - 3.7207*y_old[1];
    y_old[1] = y_old[0];
    y_old[0] = tmp;
    return tmp;
}

int main() {
    const int Nmax = 200;
    int n, k, tmp;
    float *yvalues, ymax = -MAXFLOAT, ymin =
MAXFLOAT;
    float scale_x, scale_y;
    char string[20];

    do {
        cout << "Tinh bao nhieu buoc ?";
        cin >> n;
    } while (n < 1 || n > Nmax);
```

```
    int gm,gd=DETECT;
    initgraph(&gd,&gm,"c:\\Borland\\TC\\BGI");
    if (graphresult() != grOk) {
        cerr << "Graphics Error !!!\n";
        return 1;
    }

    yvalues = new float[n];
    if (!yvalues) {
        cerr << "Insufficent memory.";
        closegraph();
        return 2;
    }

    scale_x = 600.0/n;
    setbkcolor(WHITE);
    cleardevice();
    setcolor(BLUE);
    rectangle(34, 49, 636, 450);
    settextjustify(CENTER_TEXT, CENTER_TEXT);
    outtextxy(319, 24, "QUA TRINH QUA DO CUA HE THONG
CHUA CO BDK");
    settextjustify(CENTER_TEXT, TOP_TEXT);
    for (k=0; k<n; ++k) {
        yvalues[k] = y(k);
        if (yvalues[k] > ymax) ymax = yvalues[k];
        if (yvalues[k] < ymin) ymin = yvalues[k];
        tmp = 34 + scale_x*k;
        line(tmp, 450, tmp, 453);
        outtextxy(tmp, 456, itoa(k, string, 10));
    }

    if (ymin > 0) ymin = 0;
    if (ymax < 0) ymax = 0;
    scale_y = 400.0/(ymax - ymin);
    k = ymax/100;
    settextjustify(RIGHT_TEXT, CENTER_TEXT);
    while (k*100.0 >= ymin) {
        tmp = (ymax - k*100.0)*scale_y + 49;
        line(32, tmp, 34, tmp);
        gcvt(k*100.0, 4, string);
        outtextxy(31, tmp, string);
        --k;
```

```
    }
    moveto(34, ymax*scale_y+49);
    for (k=0; k<n; ++k) {
        lineto(34+scale_x*k, gety());
        lineto(getx(), (ymax - yvalues[k])*scale_y +
49);
    }
    delete[] yvalues;
    getch();
    closegraph();
    return 0;
}
```

Dịch và chạy ch- ơng trình trên với số b- ớc tính là 9, ta có kết quả quá trình quá độ nh- hình sau:



2.6. So sánh kết quả với MatLab / Simulink:

Cũng hệ thống trên, mô phỏng trên MatLab ta có:

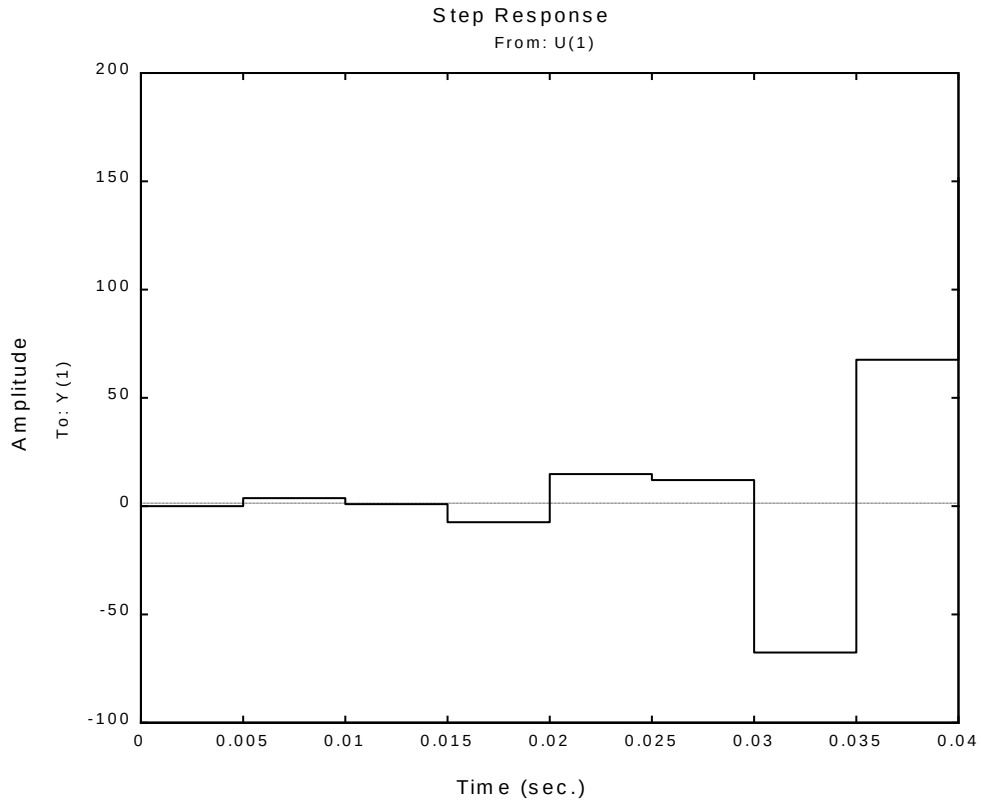
```
» Wdt = tf(65, [0.02 1])*tf(9, [0.1 1])
Transfer function:
          585
----------------------
0.002 s^2 + 0.12 s + 1
```

```
» Wdtz=c2d(Wdt, 0.005, 'zoh')
Transfer function:
   3.313 z + 2.998
---------------------
z^2 - 1.73 z + 0.7408
```

Sampling time: 0.005

```
» Wht=feedback(Wdtz, 1)
Transfer function:
   3.313 z + 2.998
---------------------
z^2 + 1.583 z + 3.739
Sampling time: 0.005

» step(Wht, 0.04)
```

Kết quả ta có quá trình quá độ nh- hình sau:



Có thể thấy quá trình quá độ đ- ợc vẽ bằng MatLab và bằng ch- ơng trình C++ là giống hệt nhau, điều đó cho thấy kết quả tính toán ở các phần tr- ớc là chính xác.

4. TỔNG HỢP HỆ THỐNG

3.1 Tổng hợp hệ thống dùng bộ điều khiển PID:

3.1.1. Bộ điều khiển PID và việc tìm các thông số cho bộ điều khiển PID:

Bộ điều khiển PID (Proportional – Integral – Derivative) là bộ điều khiển kinh điển, đ- ợc sử dụng rất nhiều khi tổng hợp hệ thống. Mặc dù hiện nay đã có các ph- ơng pháp tổng hợp hệ thống khác tốt hơn (nh- ph- ơng pháp dùng hồi tiếp trạng thái sẽ đ- ợc xét ở phần II) nh- ng bộ điều khiển PID vẫn tiếp tục đ- ợc sử dụng rộng rãi. Bộ điều khiển PID gồm ba thành phần: thành phần tỉ lệ, thành phần tích phân và thành phần vi phân. Mỗi thành phần có những ảnh h- ởng nhất định đến chất l- ợng của hệ thống, và việc lựa chọn một bộ tham số phù hợp cho ba thành phần đó sẽ đem lại cho hệ thống chất l- ợng mong muốn.

Hàm truyền liên tục của bộ điều khiển PID có thể đ- ợc viết d- ới dạng sau:

$$W_{PID}(s) = K_P + \frac{K_I}{s} + K_D s$$

Để chuyển từ bộ PID liên tục sang bộ PID số có vài cách khác nhau. Một ph- ơng pháp là chuyển gần đúng từng thành phần của bộ PID liên tục sang dạng rời rạc nh- sau:

- Thành phần tỉ lệ đ- ợc giữ nguyên.
- Thành phần tích phân đ- ợc lấy gần đúng theo Tustin: $\frac{K_I}{s} \approx \frac{K_I T(z+1)}{2(z-1)}$
- Thành phần vi phân đ- ợc lấy gần đúng theo công thức: $K_D.s \approx \frac{K_D(z-1)}{T.z}$

Nh- vậy, hàm truyền rời rạc của bộ PID số là:

$$\begin{aligned} W_{PID}(z) &= K_P + \frac{K_I T(z+1)}{2(z-1)} + \frac{K_D(z-1)}{T.z} \\ &= \frac{2.T.K_P.z.(z-1) + K_I.T^2.z.(z+1) + 2K_D(z-1)^2}{2.T.z.(z-1)} \\ &= \frac{(K_P + 0.5K_I T + \frac{K_D}{T})z^2 - (K_P - 0.5K_I T + 2\frac{K_D}{T})z + \frac{K_D}{T}}{z(z-1)} \end{aligned}$$

Có thể thấy, bộ điều khiển PID số có 2 điểm cực (0 và 1) và tối đa 2 điểm Zero.

Có nhiều ph- ơng pháp khác nhau để tổng hợp hệ thống dùng bộ điều khiển PID nói chung và bộ điều khiển PID số nói riêng. Tuy nhiên, hiện vẫn ch- a có một ph- ơng pháp tổng quát và chính xác nào để tìm đ- ợc bộ điều khiển PID tốt nhất cho một hệ thống. Các ph- ơng pháp cho đến nay vẫn chỉ cho phép xác định một cách t- ơng đối các thông số của bộ PID (đáp ứng đ- ợc phần nào chất l- ợng mong muốn). Sau đó, phải tiếp tục thay đổi các thông số (trong một lân cận xung quanh giá trị tìm đ- ợc) và "mò" cho đến khi tìm đ- ợc bộ thông số đáp ứng yêu cầu chất l- ợng đã đề ra. Nh- vậy, việc xác định các thông số cho bộ PID mang nhiều tính chất "mò mẫm". Tất nhiên "mò mẫm" cũng phải có ph- ơng pháp. Việc dò tìm các thông số cho bộ điều khiển PID phải dựa trên các nguyên tắc về ảnh h- ởng của từng thành phần trong bộ điều khiển PID đến chất l- ợng của hệ thống. Một cách chung nhất, có thể tóm tắt các nguyên tắc đó trong bảng sau:

	Rise Time	Overshoot	Settling Time	Steady State Error
K_P	Giảm	Tăng	Thay đổi ít	Giảm
K_I	Giảm	Tăng	Tăng	Triệt tiêu
K_D	Thay đổi ít	Giảm	Giảm	Thay đổi ít

Lấy một ví dụ, với sai lệch tĩnh (Steady State Error), khi tăng K_P sẽ làm giảm sai lệch tĩnh, tăng K_I sẽ có thể triệt tiêu đ- ợc sai lệch tĩnh, còn K_D ít có ảnh h- ởng. Tất nhiên, các nguyên tắc trên không đúng tuyệt đối bởi ba thông số trên có ảnh h- ởng lẫn nhau và sự thay đổi của bất kì một thông số nào cũng có thể gây ảnh h- ởng không nhỏ đến tác dụng của hai thông số còn lại.

Riêng đối với bộ PID số, có hai h- ớng chính để tổng hợp là:

- H- ớng thứ nhất là tổng hợp bộ điều khiển PID liên tục tr- ớc, sau đó chuyển bộ điều khiển tìm đ- ợc sang miền rời rạc bằng công thức gần đúng đã trình bày ở trên. H- ớng này chỉ áp dụng đ- ợc khi chu kì lấy mẫu của hệ số nhỏ hơn rất nhiều lần so với hằng số thời gian nhỏ nhất trong đối t- ợng.
- H- ớng thứ hai là tổng hợp trực tiếp trong miền rời rạc. Ph- ơng pháp này không bị hạn chế về chu kì lấy mẫu nh- ph- ơng pháp trên.

Rõ ràng, trong tr- ờng hợp cụ thể của bài tập này, do chu kì lấy mẫu (0.005s) không nhỏ hơn nhiều so với hằng số thời gian nhỏ nhất trong đối t- ợng (0.02s) nên ta phải chọn ph- ơng pháp thứ hai, tức là tổng hợp trực tiếp trong miền rời rạc.

3.1.2. Chọn thông số cho bộ điều khiển PID:

Ta đã biết rằng hệ xung số sẽ ổn định khi tất cả các điểm cực (nghiệm của ph- ơng trình đặc tính mà không trùng với điểm cực) của hệ thống nằm trong đ- ờng tròn đơn vị. Vị trí của các điểm cực cũng ảnh h- ởng đến chất l- ợng của quá trình quá độ.

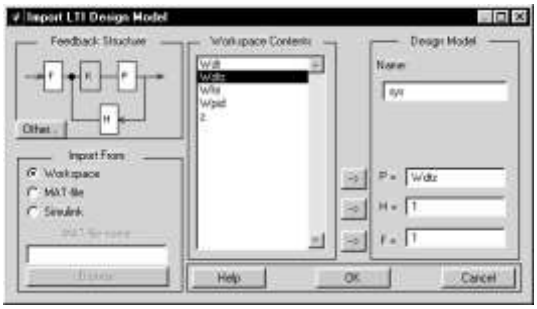
Tr- ớc hết, cần phải xác định một cách t- ơng đối bộ thông số cho bộ điều khiển PID sao cho hệ thống tr- ớc hết phải ổn định, sau nữa là đạt chất l- ợng t- ơng đối tốt. Trong tr- ờng hợp này có thể sử dụng ph- ơng pháp quĩ đạo nghiệm số để xác định các điểm cực cho hệ thống. Một công cụ rất tiện lợi và mạnh phục vụ cho mục đích này đã có sẵn trong MatLab, đó là ch- ơng trình **rltool**. Ch- ơng trình này cho phép thiết kế các bộ điều khiển trong một hệ kín bằng cách đặt vị trí của các điểm cực và điểm Zero của bộ điều khiển (hay còn gọi là bộ bù – Compensator) và của cả hệ thống. Sử dụng ch- ơng trình này để tìm sơ bộ các thông số cho bộ PID.

Tr- ớc hết phải khai báo trong MatLab hàm truyền đạt rời rạc của đối t- ợng:

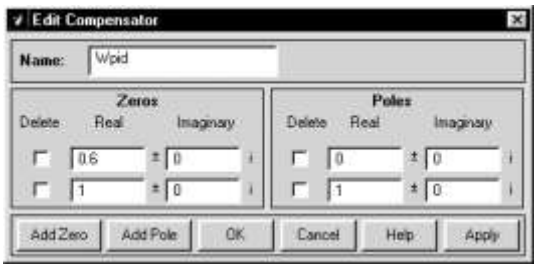
```
>> Wdt = tf(65,[0.02 1])*tf(9,[0.1 1])
Transfer function:
         585
----------------------
0.002 s^2 + 0.12 s + 1

>> T = 0.005; Wdtz = c2d(Wdt,T,'zoh')
Transfer function:
   3.313 z + 2.998
--------------------
z^2 - 1.73 z + 0.7408
Sampling time: 0.005
```

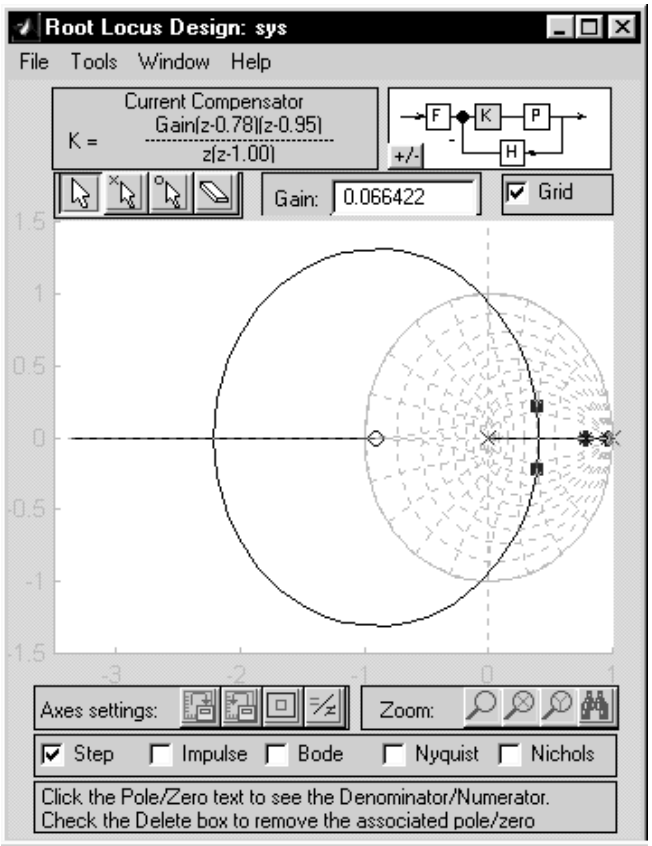
Và gọi ch- ơng trình **rltool** bằng lệnh : `>> rltool`

Sau khi ch- ơng trình **rltool** đã đ- ợc mở, nhập mô hình của đối t- ợng vào bằng cách vào menu **File\Import Model**. Một hộp thoại hiện ra cho phép nhập mô hình của đối t- ợng vào *(xem hình bên)*. Chọn biến `Wdtz` cho đối t- ợng **P** và chọn **OK**.



Tiếp theo cần nhập mô hình cho bộ bù (hay bộ điều khiển). Ta đã biết là bộ điều khiển PID số có hai điểm cực là 0 và 1, và có hai điểm Zero. Do đó ta sẽ tạo ra bộ PID số bằng cách thêm 2 điểm cực và 2 điểm Zero cho bộ bù. Chọn menu **Tool\Edit Compensator** , một hộp thoại hiện ra cho phép tạo các điểm cực và điểm Zero của bộ bù. Thêm các điểm cực và điểm Zero cho bộ bù nh- mong muốn *(xem hình bên)*. Sau đó, quan sát trên đồ thị quĩ đạo nghiệm và tiến hành kéo các điểm Zero của bộ bù và điểm cực của hệ thống cho đến khi đạt đ- ợc vị trí nh- mong muốn. Sau một thời gian thử các vị trí khác nhau của các điểm Zero và điểm cực, ta tìm đ- ợc vị trí nh- trong hình d- ới đây với chất l- ợng cơ bản là tốt.



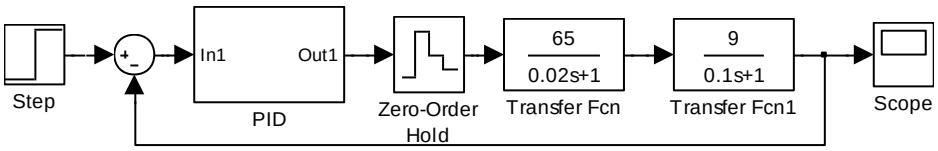


Từ bộ bù tìm đ- ợc trên **rltool** ta rút ra đ- ợc các thông số PID sau:

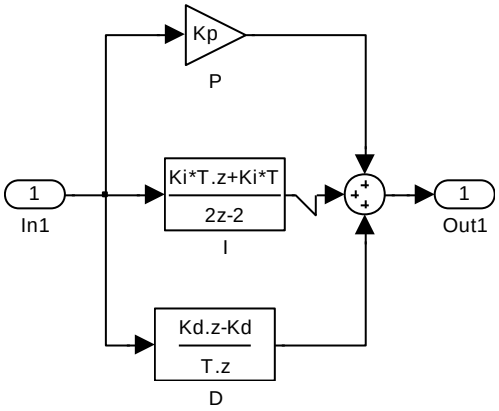
$K_P = 0.01812$
$K_I = 0.1605$
$K_D = 0.00026$

Để thử lại bộ PID đã tìm đ- ợc và cũng để tiện lợi cho việc tiếp tục tinh chỉnh các thông số PID vừa tìm đ- ợc, ta xây dựng một mô hình trên Simulink. Sơ đồ mô hình trên Simulink đ- ợc cho ở hình d- ới. Trong đó chú ý là bộ PID số đã đ- ợc đặt mặt nạ (Mask). Cấu trúc bên trong của khối PID số cũng đ- ợc cho ở hình d- ới.

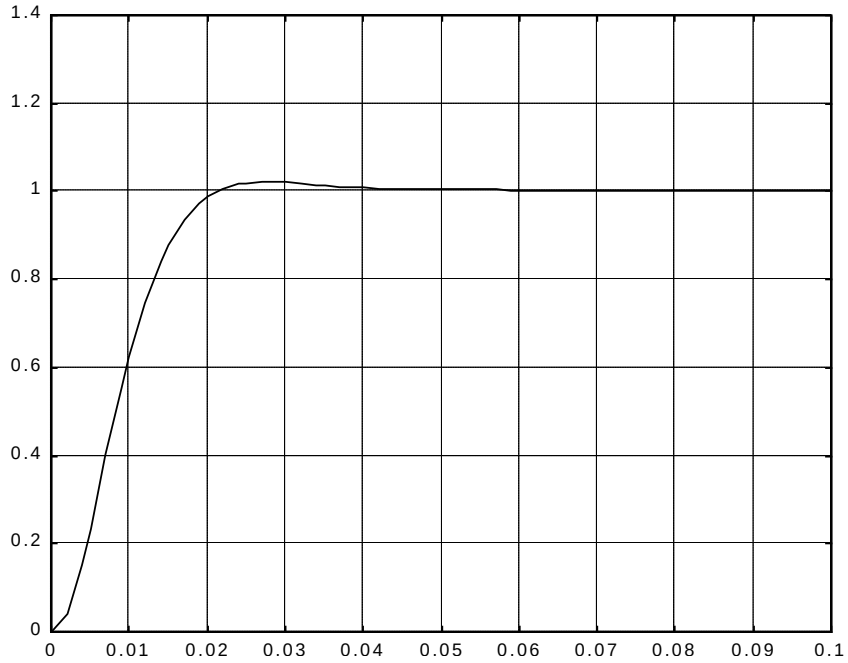
Tiến hành mô phỏng trên Simulink với bộ thông số đã tìm đ- ợc, ta có kết quả nh- hình d- ới.



Mô hình mô phỏng trên Simulink



Sơ đồ cấu trúc bên trong của khối PID số



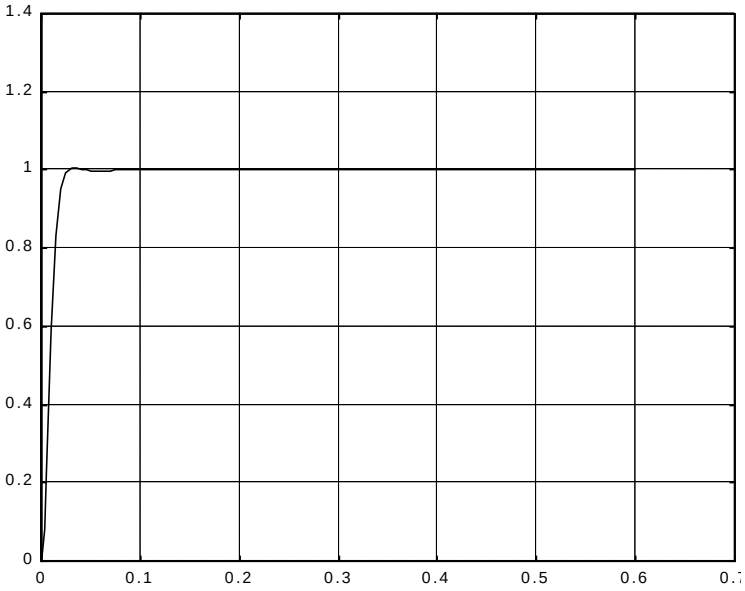
Kết quả mô phỏng với bộ số tìm đ- ợc

Qua đồ thị ta thấy hệ thống có độ quá điều chỉnh nh- ng không lớn, thời gian điều chỉnh khoảng 40ms và không có sai lệch tĩnh. Dùng Simulink để tinh chỉnh các thông số, cuối cùng ta có bộ số nh- sau:

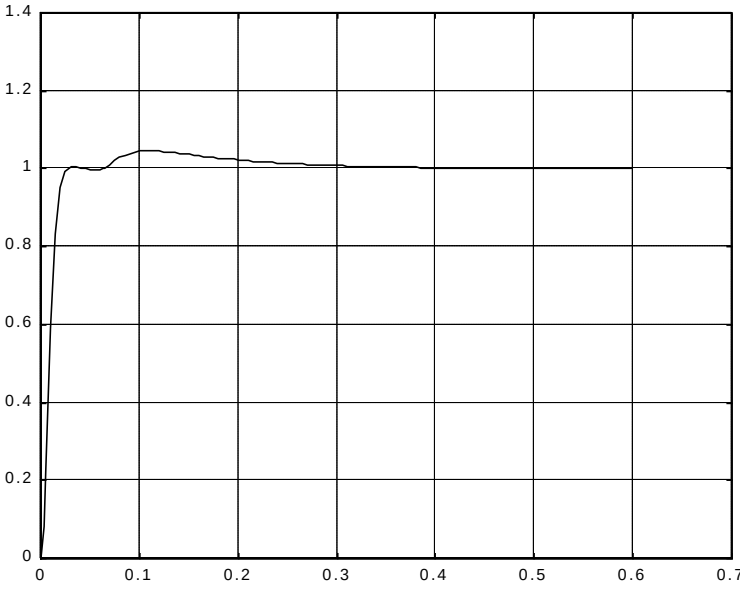
$K_P = 0.01676$
$K_I = 0.14224$
$K_D = 0.000246$

Quá trình quá độ của hệ thống với bộ thông số trên nh- sau:

Hệ thống có độ quá điều chỉnh < 1%, thời gian quá độ khoảng 30ms. Nh- vậy hệ thống đã đạt chất l- ợng rất tốt.



Để kiểm tra sự ảnh h- ởng của nhiễu đến hệ thống, ta cho một nhiễu ở đầu vào của đối t- ợng bắt đầu tại thời điểm t = 0.06s. Kết quả mô phỏng trên Simulink nh- sau:



Có thể thấy là sau khi chịu tác động của nhiễu, hệ thống lệch khỏi giá trị xác lập nh- ng sau đó lại quay trở về giá trị xác lập này (tại thời điểm 0.4s thì lại trở về đến giá trị xác lập). Nh- vậy nhiễu không ảnh h- ởng đến sai lệch tĩnh của hệ thống.

Vậy bộ PID số đ- ợc chọn ở đây có các thông số nh- sau:

$K_P = 0.01676$

$K_I = 0.14224$

$K_D = 0.000246$

và có hàm truyền đạt là:

$$W_{PID}(z) = 0.01676 + \frac{0.14224.T.(z+1)}{2(z-1)} + \frac{0.000246.(z-1)}{T.z}$$

$$= \frac{0.06632z^2 - 0.1148z + 0.0492}{z(z-1)}$$

3.2. Tổng hợp hệ thống dùng hồi tiếp trạng thái:

Mặc dù bộ điều khiển PID số đã đ- ợc sử dụng rất rộng rãi và phổ biến nh- ng nó có nhiều nh- ợc điểm khó khắc phục nh- : việc tổng hợp và thiết kế phức tạp, mang nặng tính "mò mẫm", kém chính xác, chất l- ợng khó có thể cao, chỉ áp dụng đ- ợc với từng tr- ờng hợp cụ thể mà không thể tổng quát hoá, ch- a có một ph- ơng pháp tổng quát và chính xác để thiết kế (tìm bộ thông số),... Hiện nay, đã xuất hiện nhiều ph- ơng pháp tổng hợp hệ thống hiện đại hơn, tốt hơn và khắc phục đ- ợc các nh- ợc điểm kể trên của bộ PID số. Một trong các ph- ơng pháp đó là ph- ơng pháp tổng hợp hệ thống dùng hồi tiếp trạng thái. Sử dụng ph- ơng pháp này có thể áp đặt khá chính xác các điểm cực cho hệ thống, mà ta đã biết các điểm cực sẽ quyết định đến tính chất, chất l- ợng của hệ thống. Phần này sẽ trình bày ph- ơng pháp tổng hợp hệ thống dùng hồi tiếp trạng thái áp dụng cho bài tập cụ thể này.

3.2.1. Nhắc lại về mô hình của đối t- ợng

Nh- đã trình bày ở ch- ơng 3, công thức (3.1) cho ta ph- ơng trình trạng thái của đối t- ợng (bao gồm cả bộ biến tần và động cơ):

$$\begin{cases} x(k+1) = \begin{bmatrix} 1.73 & -0.7408 \\ 1 & 0 \end{bmatrix}.x(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k) \\ y(k) = \begin{bmatrix} 0.3705K_{dc} & 0.3311K_{dc} \end{bmatrix} \underline{x}(k) \end{cases}$$

trong đó K_{dc} là một hệ số nằm trong khoảng từ 6 đến 10. Trong bài tập này ta đã chọn $K_{dc} = 9$.

Trong ch- ơng 3 cũng đã kiểm tra tính điều khiển đ- ợc và tính quan sát đ- ợc của đối t- ợng và đã khẳng định : đối t- ợng là quan sát đ- ợc khi $K_{dc} \neq 0$ mà điều này luôn thoả mãn, do đó đối t- ợng luôn quan sát đ- ợc. Đồng thời ta cũng đã chứng minh đối t- ợng luôn điều khiển đ- ợc với mọi giá trị của

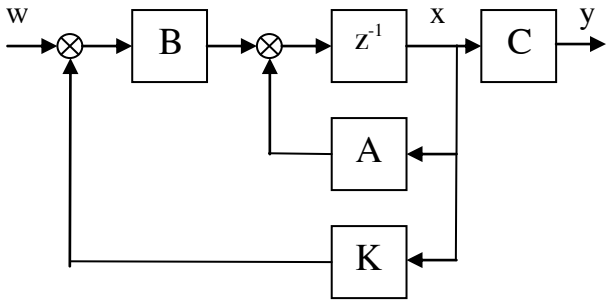
K_{dc} . Hai tính chất trên là điều kiện quan trọng để có thể tổng hợp hệ thống dùng hồi tiếp trạng thái.

3.2.2. Các ph□ơng pháp tìm bộ hồi tiếp trạng thái

Cho một đối t- ợng đ- ợc mô tả bằng ph- ơng trình trạng thái sau:

$$\begin{cases} x(k+1) = A.x(k) + B.u(k) \\ y(k) = C.x(k) \end{cases} \quad (3.2.1)$$

Các điểm cực của đối t- ợng là nghiệm của ph- ơng trình đặc tính: *det(z.I – A) = 0*. Giải ph- ơng trình đặc tính ta có đ- ợc các điểm cực của đối t- ợng : *{pc_1, pc_2, ..., pc_n}*



Vị trí của các điểm cực sẽ ảnh h- ởng đến chất l- ợng của hệ thống. Giả sử ta cần tìm một bộ hồi tiếp trạng thái cho đối t- ợng trên sao cho hệ thống (đã hồi tiếp) có các điểm cực nh- mong muốn là *{c_1, c_2, ..., c_n}*. Mô hình của hệ thống có hồi tiếp trạng thái đ- ợc cho ở hình bên.

Cho đầu vào w = 0 ta có:

$$\begin{cases} x(k+1) = A.x(k) + B.u(k) \\ u(k) = -K.x(k) \end{cases} \quad (3.2.2)$$
$$\Rightarrow x(k+1) = A.x(k) - B.K.x(k) = (A - BK).x(k)$$

Nh- vậy hệ mới sẽ có ph- ơng trình đặc tính là *det(z.I – A + B.K) = 0* và theo yêu cầu đã đề ra, ph- ơng trình đặc tính này phải có nghiệm là *{pm_1, pm_2, ..., pm_n}*. Có một số cách để xác định K thoả mãn yêu cầu trên nh- sau:

a. Cách 1

Vì ph- ơng trình đặc tính mới *det(z.I – A + B.K) = 0* phải có các nghiệm là *{pm_1, pm_2, ..., pm_n}* nên nó phải có dạng sau:

$$det(z.I - A + B.K) = (z - pm_1)(z - pm_2)...(z - pm_n) = z^n + c_1.z^{n-1} + ... + c_n$$

Trong đó K chỉ là một vector các hệ số : K = [K1, K2,..., Kn]. Khai triển đẳng thức trên và cân bằng hệ số hai vế ta tìm đ- ợc K = [K1, K2,..., Kn]

b. Cách 2

- Gọi P là ma trận điều khiển đ- ợc của đối t- ợng: P = [B, AB, ..., $A^{n-1}B$]
- Định nghĩa một ma trận W nh- sau:

$$W = \begin{bmatrix} a_{n-1} & a_{n-2} & \dots & a_1 & 1 \\ a_{n-2} & a_{n-3} & \dots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_1 & 1 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix}$$

trong đó $det(z.I - A) = z^n + a_1.z^{n-1} + ... + a_n$

- Tính ma trận T = P.W
- Tính K theo công th- c sau:
  $K = [c_n - a_n , c_{n-1} - a_{n-1} , ... , c_1 - a_1].T^{-1}$

c. Cách 3

- Tính $\Phi(A) = A^n + c_1.A^{n-1} + ... + c_n.I$
- Tính K theo công thức:
  $K = [0, 0, ... 1].[B, AB, ... , A^{n-1}B]^{-1}. \Phi(A)$

Ba cách trên tuy ph- ơng pháp khác nhau nh- ng là hoàn toàn t- ơng đ- ơng và đều cho kết quả giống nhau.

3.2.3. Ph□ơng pháp chọn điểm cực của Bessel

Khi đã có ph- ơng pháp tìm bộ hồi tiếp trạng thái thì vấn đề còn lại bây giờ là chọn điểm cực mới của hệ thống thế nào để hệ thống đạt chất l- ợng nh- mong muốn. Bessel đã xác định các điểm cực chuẩn cho các hệ thống (liên tục) bậc k sao cho với các điểm cực đó, hệ thống đạt chất l- ợng nh- sau: không có quá điều chỉnh và thời gian quá độ là T = 1s. Bessel đã tổng kết các điểm cực chuẩn đó trong bảng sau *(chú ý ở đây chỉ trình bày phần bảng cho các hệ thống bậc 1,2,3 là các bậc có dùng đến trong bài tập này).*

Bậc	Điểm cực chuẩn
1	- 4.6200
2	- 4.0530 ± 2.3400i
3	- 5.0093, -3.9668 ± 3.7845i

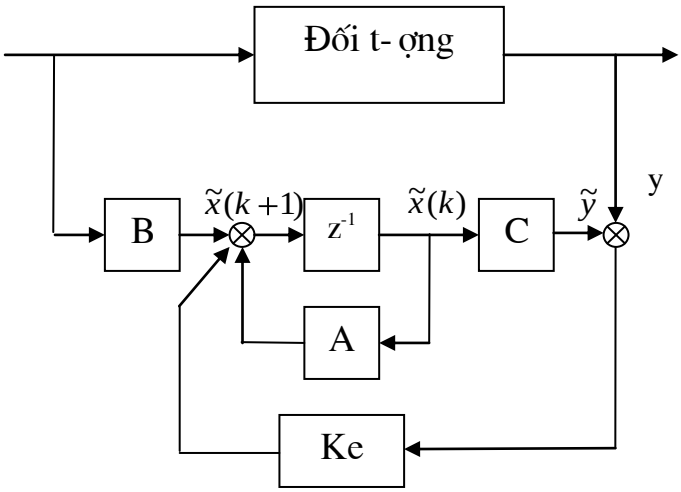
Nếu muốn hệ thống có quá trình quá độ không có quá điều chỉnh và thời gian quá độ là T bất kì (trong một giới hạn nhất định nào đó) thì hệ thống cần

có các điểm cực đ- ợc xác định bằng cách lấy các điểm cực chuẩn trong bảng của Bessel chia cho T.

Với hệ thống xung – số, để xác định các điểm cực cần thiết, tr- ớc hết ta xác định các điểm cực trong miền liên tục theo qui tắc trên (giả sử đ- ợc các điểm cực là p_1, p_2, ..., p_n), sau đó chuyển các điểm cực này sang miền Z theo công thức: $z_k = e^{T.pk}$

3.2.4. Xây dựng bộ □ớc l□ợng trạng thái (bộ quan sát trạng thái)

Một vấn đề nảy sinh khi tổng hợp hệ thống dùng hồi tiếp trạng thái là trong thực tế, hiếm khi có thể đo đ- ợc trực tiếp các biến trạng thái (x) của đối t- ợng. Vậy, cần phải làm cách nào đó xác định đ- ợc các biến trạng thái của đối t- ợng chỉ dựa trên các đại l- ợng đo đ- ợc hoặc biết đ- ợc (nh- tín hiệu điều khiển, đầu ra của đối t- ợng,...). Xây dựng đ- ợc một hệ thống nh- thế chính là xây dựng một bộ - ớc l- ợng trạng thái (hay bộ quan sát trạng thái). Bộ - ớc l- ợng trạng thái dựa vào các đại l- ợng có thể biết đ- ợc là đầu vào của đối t- ợng (chính là tín hiệu điều khiển) và đầu ra của đối t- ợng để xác định trạng thái của đối t- ợng. Chỉ làm đ- ợc điều này khi đối t- ợng là quan sát đ- ợc. Rõ ràng đối t- ợng trong bài tập này thoả mãn điều kiện đó, do vậy có thể xây dựng đ- ợc bộ - ớc l- ợng trạng thái cho đối t- ợng.



Bộ - ớc l- ợng trạng thái đ- ợc xây dựng theo mô hình bên.

Ta có:

$$x(k+1) = A.x(k) + B.u(k)$$

$$\begin{aligned}\tilde{x}(k+1) &= A.\tilde{x}(k) + B.u(k) + Ke.(y(k) - \tilde{y}(k)) \\ &= A.\tilde{x}(k) + B.u(k) + Ke.(C.x(k) - C.\tilde{x}(k)) \\ &= A.\tilde{x}(k) + B.u(k) + Ke.C(x(k) - \tilde{x}(k)) \\ \Rightarrow x(k+1) - \tilde{x}(k+1) &= A.(x(k) - \tilde{x}(k)) - Ke.C.(x(k) - \tilde{x}(k)) \\ \Rightarrow &\begin{cases} e(k+1) = (A - Ke.C).e(k) \\ e(k) = x(k+1) - \tilde{x}(k+1) \end{cases}\end{aligned}$$

Mục đích của bộ - ớc l- ợng trạng thái là phải làm sao cho *e(k)* tiến đến 0. Nhận thấy hệ ph- ơng trình trên có dạng giống nh- hệ (3.2.2). Vì vậy ta có thể áp dụng ph- ơng pháp đặt điểm cực để tìm Ke.

Với đối t- ợng cụ thể của bài tập này, ta chọn điểm cực theo Bessel cho hệ bậc hai sao cho thời gian quá độ là T = 0.005s, suy ra điểm cực cần thiết là:

$p_{1,2}$ = (- 4.0530 ± 2.3400i) / 0.005

và trong miền Z thì điểm cực là:

$zp_{1,2} = \exp(0.005.p_{1,2})$ = - 0.0121 ± 0.0125i

Suy ra ph- ơng trình đặc tính mới phải là:

$$\det(z.I - A + Ke.C) = (z - zp_1)(z - zp_2)$$
$$= (z + 0.0121 + 0.0125i)(z + 0.0121 - 0.0125i)$$
$$= z^2 + 0.024164.z + 0.000302$$

Ta lại có:

$$\det(z.I - A + Ke.C)$$
$$= \det(z.\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - \begin{bmatrix}1.73 & -0.7408\\ 1 & 0\end{bmatrix} + \begin{bmatrix}Ke1\\ Ke2\end{bmatrix}\begin{bmatrix}0.3705K_{dc} & 0.3311K_{dc}\end{bmatrix})$$
$$= \det(\begin{bmatrix}z - 1.73 + 0.3705.Ke1.K_{dc} & 0.7408 + 0.3311Ke1.K_{dc}\\ -1 + 0.3705.Ke2.K_{dc} & z + 0.3311Ke2.K_{dc}\end{bmatrix})$$
$$= z^2 + (0.3705.Ke1.K_{dc} + 0.3311Ke2.K_{dc} - 1.73).z + (0.7408 + 0.3311Ke1.K_{dc} - 0.8473Ke2.K_{dc})$$

Cân bằng hai vế của ph- ơng trình đặc tính ta có hệ ph- ơng trình sau:

$$\begin{cases}0.3705.Ke1.K_{dc} + 0.3311Ke2.K_{dc} - 1.73 = 0.024164\\ 0.7408 + 0.3311Ke1.K_{dc} - 0.8473Ke2.K_{dc} = 0.000302\end{cases}$$

Giải ra ta có nghiệm:

$Ke1 = 2.9302/K_{dc}$

$Ke2 = 2.0191/K_{dc}$

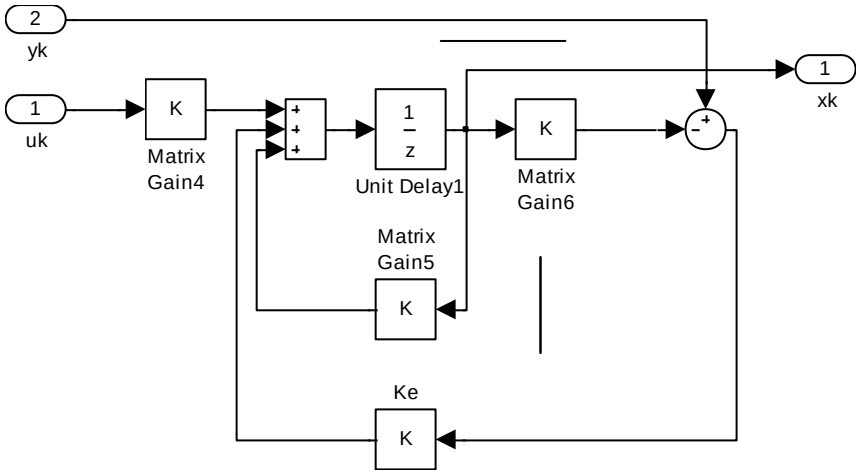
hay:

$$Ke = \frac{1}{K_{dc}}\begin{bmatrix}2.9302\\ 2.0191\end{bmatrix}$$

Đó chính là dạng tổng quát của Ke áp dụng cho bộ - ớc l- ợng trạng thái. Thay số K_{dc} = 9 ta có:

$$Ke = \begin{bmatrix}0.32557\\ 0.22434\end{bmatrix}$$

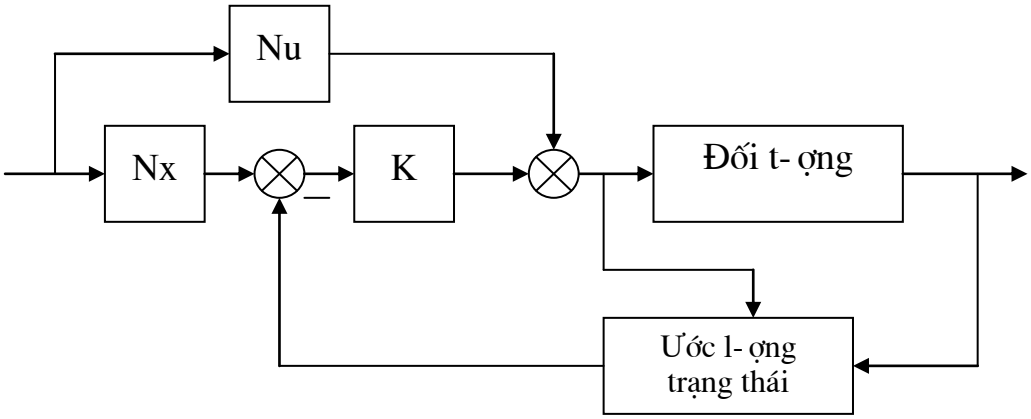
Để tiện lợi khi khảo sát và tổng hợp hệ thống sau này, ta sẽ xây dựng một khối con (sub-system) trên Simulink chứa bộ - ớc l- ợng trạng thái và đặt mặt nạ (Mask) cho nó rồi đ- a vào th- viện. Sơ đồ cấu trúc của bộ - ớc l- ợng trạng thái trên Simulink nh- hình sau:



3.2.5. Tổng hợp hệ thống dùng hồi tiếp trạng thái:

Sơ đồ khối của hệ thống có hồi tiếp trạng thái đã đ- ợc trình bày trong mục 2. Một thành phần đ- ợc thêm vào sơ đồ trên là bộ - ớc l- ợng trạng thái. Tuy nhiên, nếu thuần tuý chỉ hồi tiếp trạng thái qua bộ hồi tiếp K nh- vậy thì hệ thống sẽ có sai lệch tĩnh rất lớn. Điều này đ- ợc giải thích do trong đối t- ợng và cả trong hệ kín đều không có khâu tích phân, ngoài ra đối t- ợng có hệ số khuếch đại rất lớn (lên đến hơn 500). Không những thế, khi có nhiễu tác động vào hệ thống (chủ yếu là nhiễu ở đối t- ợng) thì hệ thống sẽ bị ảnh h- ởng rất lớn và sai lệch tĩnh sẽ có thể rất lớn. Để khắc phục đ- ợc vấn đề này, có hai giải pháp có thể đ- ợc sử dụng:

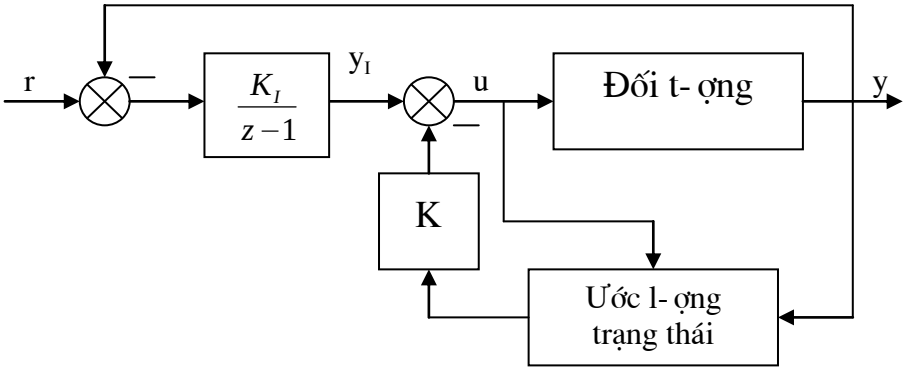
- Dùng các khâu bù đầu vào và bù song song để giảm sai lệch tĩnh của hệ thống và hạn chế ảnh h- ởng của nhiễu nh- hình d- ới. Thực tế, ph- ơng pháp này không thể triệt tiêu hẳn đ- ợc sai lệch tĩnh và vẫn chịu ảnh h- ởng không nhỏ của nhiễu. Sở dĩ nh- vậy là vì khi thiết kế bộ bù, ta không thể tính chính xác đ- ợc các thông số mà luôn có sự làm tròn, và bản thân các thông số của đối t- ợng cũng sẽ bị thay đổi trong quá trình hoạt động.



- Đ- a thêm khâu tích phân vào vị trí thích hợp trong hệ thống *(xem hình d- ới)*. Khâu tích phân có đặc điểm là nó có khả năng triệt tiêu hoàn toàn đ- ợc sai lệch tĩnh và giảm ảnh h- ởng của nhiễu đến hệ thống. Với hệ xung – số khâu tích phân đ- ợc lựa chọn có hàm truyền đạt số là:

$$W_I(z) = \frac{K_I}{z-1}$$

Sơ đồ khối của hệ thống đã bù bằng khâu tích phân nh- sau:



Sau khi điểm qua hai ph- ơng pháp trên, ta thấy ph- ơng pháp thêm khâu tích phân (ph- ơng pháp 2) có nhiều - u điểm hơn, bởi vậy trong bài tập này sẽ sử dụng ph- ơng pháp thêm khâu tích phân.

Tr- ớc hết cần tìm ph- ơng trình trạng thái của khâu tích phân. Theo sơ đồ trên ta có:

$$\frac{Y_I(z)}{U_I(z)} = \frac{K_I}{z-1}$$

$$\Rightarrow \begin{cases} x_I(k+1) = x_I(k) + u_I(k) \\ y_I(k) = K_I.x_I(k) \end{cases}$$

Ta thấy khâu tích phân này chỉ có một biến trạng thái duy nhất và đầu ra của khâu tích phân tỉ lệ với biến trạng thái này.

Đặt $X(k)=\begin{bmatrix} x(k) \\ x_I(k) \end{bmatrix}$ ta có:

$$u(k)=K_I.x_I(k)-K.x(k)=-\begin{bmatrix} K & -K_I \end{bmatrix}.\underline{X}(k)$$

$$u_I(k)=-C.x(k)$$

$$\begin{cases} x(k+1)=A.x(k)+B.u(k) \\ x_I(k+1)=x_I(k)+u_I(k)=-C.x(k)+x_I(k) \end{cases}$$

$$\Rightarrow \begin{cases} X(k+1)=\begin{bmatrix} A & 0 \\ -C & 1 \end{bmatrix}X(k)+\begin{bmatrix} B \\ 0 \end{bmatrix}u(k) \\ u(k)=-\begin{bmatrix} K & -K_I \end{bmatrix}.\underline{X}(k) \end{cases}$$

$$\Rightarrow X(k+1)=\left(\begin{bmatrix} A & 0 \\ -C & 1 \end{bmatrix}-\begin{bmatrix} B \\ 0 \end{bmatrix}.\begin{bmatrix} K & -K_I \end{bmatrix}\right)X(k)$$

Đặt $e(k)=X(k)-X(\infty)$ là sai lệch tĩnh ta có:

$$e(k+1)=\left(\begin{bmatrix} A & 0 \\ -C & 1 \end{bmatrix}-\begin{bmatrix} B \\ 0 \end{bmatrix}.\begin{bmatrix} K & -K_I \end{bmatrix}\right).e(k)$$

Mục đích của ta là phải làm cho e(k) tiến đến 0 càng nhanh càng tốt để triệt tiêu đ- ợc sai lệch tĩnh. Đến đây ta lại thấy dạng quen thuộc của (3.2.2) và do đó có thể áp dụng ph- ơng pháp tìm bộ hồi tiếp trạng thái (đã trình bày ở phần 2) để tìm $[K\ -K_I]$.

Hệ mới bây giờ có bậc là 3 chứ không còn là 2 nữa. Sử dụng bảng điểm cực chuẩn của Bessel cho hệ bậc 3 và chọn thời gian quá độ là T = 20ms = 0.02s ta có điểm cực mới là:

$$p_1=\frac{-5.0093}{0.02}=-250.46;\ p_{2,3}=\frac{-3.9668\pm 3.7845i}{0.02}=-198.34\pm 189.23i$$

$$\Rightarrow zp_1=e^{0.005.p1}=e^{0.005(-250.46)}=0.2858;\ zp_{2,3}=e^{0.005.p2,3}=e^{0.005(-198.34\pm 189.23i)}=0.2169\pm 0.3009i$$

Ph- ơng trình đặc tính mới là:

$$\det\left(z.I - \begin{bmatrix} A & 0 \\ -C & 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix}. K \quad -K_I \right)$$

$$= \det\left(\begin{bmatrix} z & 0 & 0 \\ 0 & z & 0 \\ 0 & 0 & z \end{bmatrix} - \begin{bmatrix} 1.73 & -0.7408 & 0 \\ 1 & 0 & 0 \\ -0.3705K_{dc} & -0.3311K_{dc} & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. K1 \quad K2 \quad -K_I \right)$$

$$= \det\begin{pmatrix} z-1.73+K1 & 0.7408+K2 & -K_I \\ -1 & z & 0 \\ 0.3705K_{dc} & 0.3311K_{dc} & z-1 \end{pmatrix}$$

$$= z^3 + (-2.73 + K1)z^2 + (2.4708 - K1 + 0.3705K_I.K_{dc} + K2)z + 0.3311.K_I.K_{dc} - 0.7408 - K2$$

Mặt khác ta lại có:

$$\det\left(z.I - \begin{bmatrix} A & 0 \\ -C & 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix}. K \quad -K_I \right)$$

$$= (z - zp_1)(z - zp_2)(z - zp_3)$$

$$= (z - 0.2858)(z - 0.2169 + 0.3009i)(z - 0.2169 - 0.3009i)$$

$$= z^3 - 0.7196.z^2 + 0.2616z - 0.0393$$

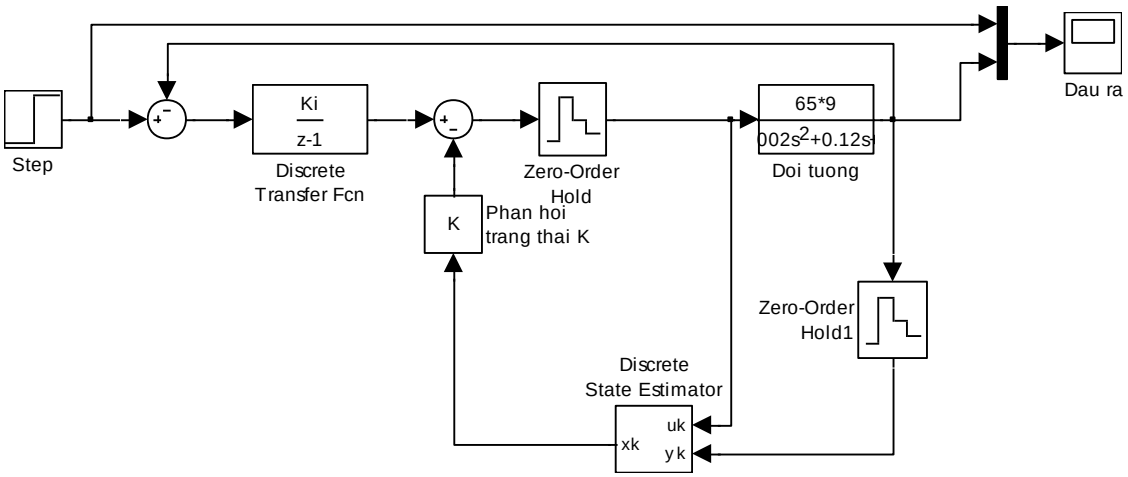
Cân bằng hai vế của hai ph- ơng trình trên ta có:

$$\begin{cases} -2.73 + K1 = -0.7196 \\ 2.4708 - K1 + 0.3705K_I.K_{dc} + K2 = 0.2616 \\ 0.3311.K_I.K_{dc} - 0.7408 - K2 = -0.0393 \end{cases}$$

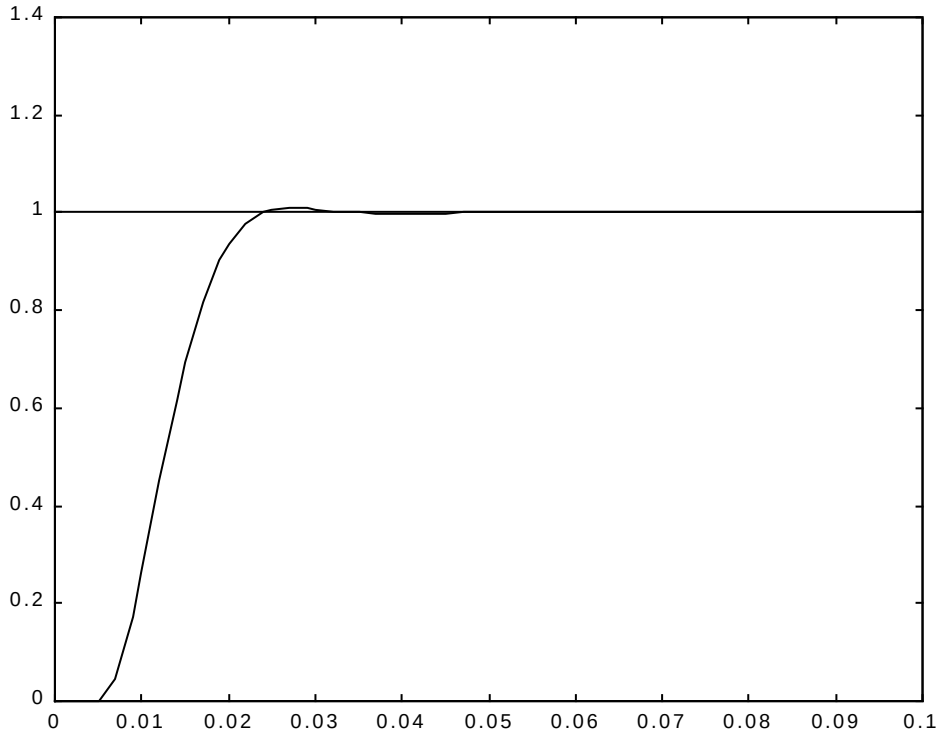
$$\Rightarrow \begin{cases} K1 = 2.0103 \\ K2 = -0.4643 \\ K_I = \dfrac{0.7163}{K_{dc}} \end{cases}$$

Vậy ta đã tìm đ- ợc các hệ số cần thiết một cách tổng quát. Nhận thấy giá trị của K1, K2 không phụ thuộc vào K_{dc} nghĩa là các giá trị đó phù hợp với mọi K_{dc}. Trong bài tập này, chọn $K_{dc} = 9$ ta có: $K_I = 0.0796$.

Thực hiện mô hình hệ thống trên trong Simulink ta có sơ đồ sau:

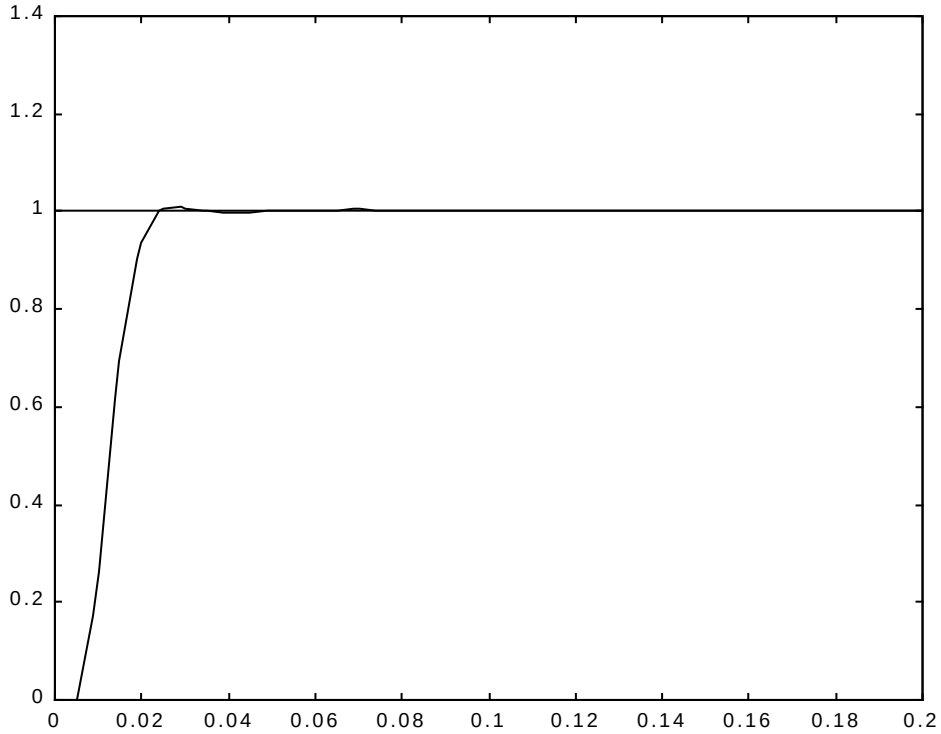


Với các giá trị K1, K2, K_I đã tính ở trên, thay vào mô hình ta có quá trình quá độ nh- sau:



Nhận xét thấy quá trình quá độ rất tốt, thời gian quá độ khoảng 20ms, độ quá điều chỉnh rất nhỏ và không có sai lệch tĩnh.

Xét ảnh h- ởng của một nhiễu ở đầu vào của đối t- ợng ta có:



Có thể thấy ngay là hệ thống rất ít bị ảnh h- ởng của nhiễu.

3.2.6. So sánh hai bộ điều khiển tìm đ□ợc:

Qua so sánh kết quả mô phỏng với hai bộ điều khiển: bộ điều khiển PID số và bộ hồi tiếp trạng thái ta có nhận xét:

- Hệ thống với bộ hồi tiếp trạng thái có thời gian quá độ nhanh hơn, tuy nhiên hơi có quá điều chỉnh nh- ng rất nhỏ.
- Hệ thống với bộ hồi tiếp trạng thái ít chịu ảnh h- ởng của nhiễu hơn, tức là nó bền vững hơn.

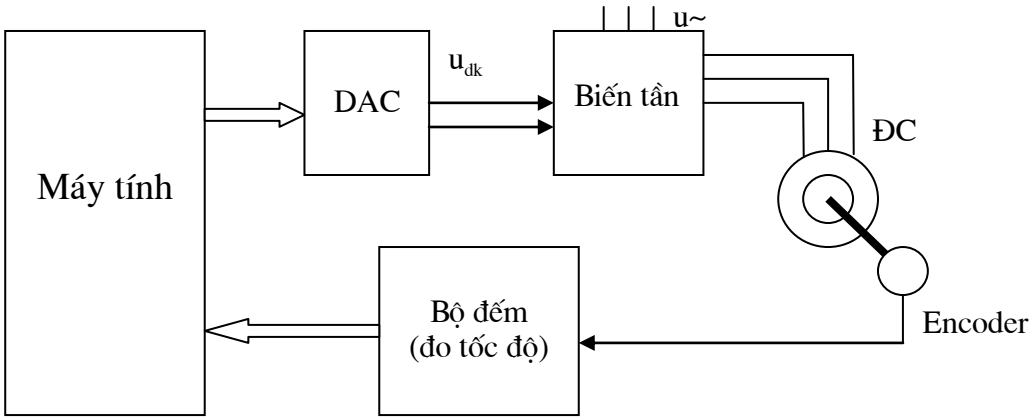
Qua các kết luận trên, em quyết định chọn bộ hồi tiếp trạng thái (có khâu tích phân) để xây dựng hệ thống điều khiển số cho động cơ không đồng bộ ba pha.

CHƯƠNG III
THIẾT KẾ PHẦN CỨNG HỆ THỐNG ĐIỀU KHIỂN SỐ ĐỘNG CƠ KĐB 3 PHA

4. SƠ ĐỒ KHỐI VÀ Ý TƯỞNG THIẾT KẾ

Cho đến nay, có nhiều ph- ơng pháp khác nhau để điều khiển động cơ không đồng bộ pha (nh- đã trình bày ở trên). Mỗi ph- ơng pháp có những - u, nh- ợc điểm riêng và tuỳ từng ứng dụng cụ thể mà cần lựa chọn một ph- ơng pháp thích hợp. Hiện nay, ph- ơng pháp tốt nhất và đ- ợc ứng dụng nhiều nhất là ph- ơng pháp điều chỉnh tần số nguồn cung cấp cho động cơ bằng các bộ biến tần. Ph- ơng pháp này có những - u điểm so với các ph- ơng pháp khác nh- cho phép điều chỉnh trong một dải tốc độ rộng, hiệu suất cao, cho phép thực hiện các ph- ơng pháp điều chế khác nhau với độ chính xác (trong điều khiển) và chất l- ợng cao,... Tuy nhiên, so với các ph- ơng pháp khác, ph- ơng pháp này có nh- ợc điểm là phức tạp, phần cứng đắt tiền, đòi hỏi hệ thống phải có tốc độ nhanh và năng lực xử lý cao. Tuy nhiên, với xu thế phát triển hiện nay của kĩ thuật và sự xuất hiện của nhiều loại biến tần chế tạo sẵn của các hãng khác nhau, việc sử dụng ph- ơng pháp điều chỉnh tần số nguồn cung cấp cho động cơ đã trở thành một lựa chọn tất yếu. Vì lí do đó, trong bài tập này, em quyết định chọn ph- ơng pháp điều chỉnh tần số nguồn cung cấp để điều khiển động cơ không đồng bộ ba pha.

Sơ đồ khối cơ bản của hệ thống điều khiển động cơ không đồng bộ ba pha dùng máy tính bằng ph- ơng pháp điều chỉnh tần số nguồn cung cấp nh- sau:



Nguyên lý hoạt động của sơ đồ khối trên nh sau:

Encoder đo tốc độ quay của động cơ (ĐC) d- ới dạng các xung gửi về bộ đếm. Bộ đếm đếm số xung, từ đó xác định đ- ợc tốc độ quay của động cơ. Tốc độ này đ- ợc hồi tiếp về bộ điều khiển mềm (soft - controller, d- ới dạng ch- ơng trình điều khiển trong máy tính). Bộ điều khiển mềm tính

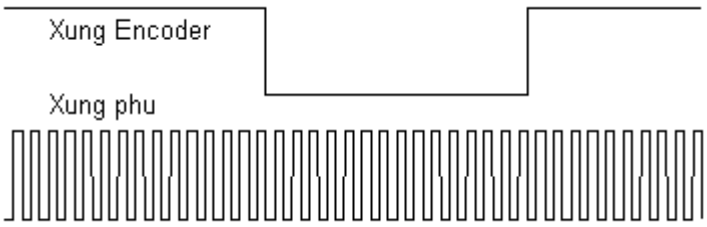
toán theo một thuật toán xác định (sẽ đ- ợc trình bày cụ thể ở ch- ơng IV và V) dựa trên đầu vào là tốc độ đặt (do ng- ời sử dụng đặt) và tốc độ thực của động cơ hồi tiếp về, từ đó có đ- ợc tín hiệu điều khiển thích hợp. Tín hiệu điều khiển này đ- ợc máy tính đ- a qua bộ biến đổi Số - T- ơng tự (DAC), biến đổi thành điện áp điều khiển (u_{dk}) để điều khiển biến tần. Căn cứ theo điện áp điều khiển, biến tần cung cấp cho động cơ một điện áp xoay chiều ba pha có tần số nh- mong muốn. Do tần số điện áp cung cấp đ- ợc thay đổi, đặc tính cơ của động cơ cũng thay đổi theo (nh- đã trình bày ở ch- ơng I), làm thay đổi tốc độ của động cơ với momen tải xác định.

Nh- vậy, để điều khiển động cơ không đồng bộ ba pha bằng máy tính, vấn đề chủ yếu còn lại là xây dựng mạch ghép nối vào/ra với máy tính. Mạch ghép nối đ- ợc thiết kế trên cơ sở ý t- ởng sau:

Để đo tốc độ quay của động cơ, ta dùng Encoder. Nguyên tắc hoạt động của Encoder là dùng một đĩa tròn có nhiều lỗ bố trí theo mép đĩa gắn với trục quay của động cơ. Số lỗ có thể thay đổi, tuỳ thuộc vào độ phân giải cần thiết. Hai bên đĩa có lắp LED và cảm biến quang học đối diện nhau. Khi có một lỗ đi qua chỗ gắn cảm biến thì cảm biến quang học sẽ nhận đ- ợc ánh sáng từ LED và tạo ra một xung. Căn cứ vào xung đó và số lỗ (hay số xung, hay độ phân giải) trong một vòng quay của Encoder, có thể tính đ- ợc tốc độ quay của động cơ. Có hai cách để xác định tốc độ quay từ Encoder:

- Cách 1: đếm số xung do Encoder gửi về trong một khoảng thời gian T xác định. Giả sử Encoder có N xung/một vòng quay và trong thời gian T đếm đ- ợc k xung phát ra từ Encoder. Khi đó, tốc độ quay của động cơ đ- ợc tính theo công thức sau: $n = \frac{60.k}{T.N}$ *(vòng/phút)*. Sai số tốc độ quay là: $\Delta n = \frac{60}{T.N}\Delta k$. Với sai số l- ợng tử của k thì $\Delta n = \frac{60}{T.N}$ là hằng số và chỉ phụ thuộc vào thời gian đếm xung T và độ phân giải N của Encoder.

- Cách 2: đếm số xung phụ tần số cao trong một chu kì xung của Encoder. Đồ thị sau giúp thể hiện rõ hơn cách này.



Giả sử xung phụ có tần số f, và số xung phụ đếm đ- ợc trong một chu kì xung của Encoder là q. Khi đó, tốc độ quay của động cơ đ- ợc tính theo

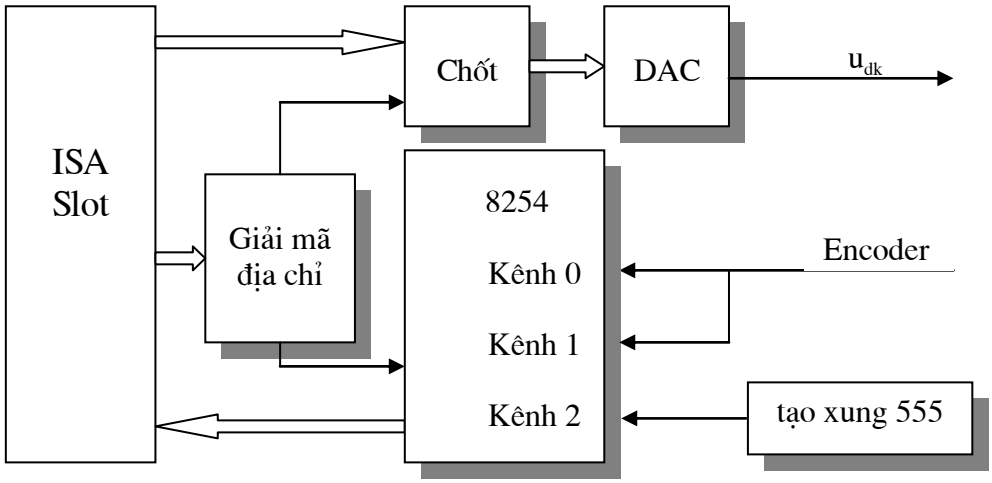
công thức sau: $n = \frac{60.f}{N.q}$ *(vòng/phút)*. Sai số tốc độ quay ứng với sai số l- ợng tử của q là: $\Delta n = \frac{60.f}{N}\frac{1}{q(q+1)}$. Rõ ràng sai số tốc độ quay trong tr- ờng hợp này không cố định mà thay đổi theo q. Khi q càng lớn, nghĩa là tốc độ quay càng chậm, thì sai số càng nhỏ và ng- ợc lại. Bởi vậy cách này chỉ thích hợp khi đo tốc độ quay chậm.

Trong bài tập này, căn cứ theo yêu cầu cụ thể của việc điều chỉnh tốc độ động cơ, em chọn cách đo tốc độ quay thứ nhất. Nh- sẽ trình bày sau này, thời gian trích mẫu (chu kì điều khiển) đ- ợc chọn là T_S = 5ms = 0,005s. Encoder đ- ợc chọn có N = 2000 *(xung/vòng)* (xem phần sau). Do đó, sai số đo tốc độ quay sẽ là $\Delta n = 6$ *vòng*. Sai số này không nhỏ, nh- ng là chấp nhận đ- ợc với bài tập. Để giảm sai số đo tốc độ quay, có thể chọn loại Encoder khác có N lớn hơn (có thể lên tới 4096, 8000 hay hơn nữa), khi đó sai số sẽ đ- ợc giảm đi đáng kể. Một giải pháp khác là xây dựng mạch đo theo cả hai ph- ơng pháp trên, và lựa chọn cách đếm một cách linh hoạt căn cứ theo tốc độ hiện thời là nhanh hay chậm: khi tốc độ cao thì dùng cách 1, khi tốc độ thấp thì dùng cách 2. Tuy nhiên, trong bài tập này không cần thiết phải xây dựng mạch đo nh- vậy.

Để đếm số xung từ Encoder gửi về, bộ đếm đ- ợc sử dụng là vi mạch 8254. Đây là vi mạch đếm - thời gian lập trình đ- ợc với khả năng rất mạnh và linh hoạt. Vi mạch có ba kênh (đ- ợc đánh số từ 0 đến 2) có thể hoạt động hoàn toàn độc lập với nhau. Tần số hoạt động tối đa là 4MHz, hoàn toàn đáp ứng đ- ợc yêu cầu của bài tập. Để tạo ra thời gian đếm chuẩn T, một vi mạch thời gian LM555 đ- ợc sử dụng để tạo ra xung đồng hồ chuẩn tần số 10Khz. Xung này đ- ợc chia tần bởi một kênh của 8254, tạo ra một xung khác có chu kì là 2*T. Xung này đ- ợc đ- a vào chân Gate của kênh đếm xung Encoder (có xung Encoder đ- a vào chân Clock) để tạo ra thời gian đếm chuẩn T. Giải pháp đơn giản là chỉ dùng một kênh để đếm xung Encoder. Tuy nhiên, giải pháp này có nh- ợc điểm là mỗi khi cần đo tốc độ, ch- ơng trình cần phát tín hiệu bắt đầu đếm và phải chờ đến khi đếm xong mới đọc đ- ợc tốc độ quay. Một giải pháp khác đ- ợc lựa chọn là dùng cả hai kênh còn lại của 8254 để đếm xung Encoder nh- ng đếm lệch nhau, nghĩa là khi kênh này đếm thì kênh kia ngừng và chốt số đếm, và ng- ợc lại. Nh- vậy, khi cần đọc tốc độ quay, chỉ việc xác định kênh nào đang đếm và đọc số đếm từ kênh kia. Việc xác định kênh đang đếm có thể đ- ợc thực hiện dễ dàng nhờ khả năng đọc ng- ợc trạng thái (Readback command) của 8254. Vì đầu ra OUT của bộ đếm chia tần của 8254 có hai giá trị là 0 và 1 t- ơng ứng với hai tr- ờng hợp hoạt động khác nhau của hai bộ đếm còn lại, do đó, để thuận tiện khi lập trình xác định địa chỉ cổng của bộ đếm đang chốt số đếm (bộ đếm đang không hoạt động), ta sẽ dùng bộ đếm 2 để chia tần và hai bộ đếm 0 và 1 để đếm xung Encoder. Khi đầu ra OUT2 = 0 thì bộ đếm 0 ngừng đếm (chốt số đếm) còn bộ đếm 1 sẽ đ- ợc phép đếm, và ng- ợc lại. Nh- vậy việc xác định địa chỉ cổng của bộ đếm đang chốt số đếm (để đọc số đếm) sẽ đơn giản chỉ là vài lệnh dịch bit và OR số học.

Để phát ra điện áp điều khiển biến tần, một bộ biến đổi số - t- ơng tự (DAC) đ- ợc sử dụng. Một nguyên tắc cần đ- ợc đảm bảo là điện áp đầu ra của DAC phải luôn đ- ợc giữ ổn định (không đổi) trong suốt chu kì điều khiển cho đến khi có tín hiệu mới xuất ra. Nh- ng dữ liệu trên bus dữ liệu lại luôn luôn thay đổi. Bởi vậy, cần phải có một bộ chốt ở tr- ớc DAC để chốt dữ liệu cũ lại cho DAC cho đến khi máy tính ghi một dữ liệu mới ra đó. Nh- vậy, ch- ơng trình sẽ không thao tác trực tiếp với DAC mà thông qua một bộ chốt. Bộ chốt đ- ợc sử dụng ở đây là vi mạch 74LS373.

Có thể tóm tắt lại ý t- ởng trên qua sơ đồ sau:



Phần trên đã trình bày ý t- ởng thiết kế cơ bản của mạch điều khiển động cơ không đồng bộ ba pha. Phần sau đây sẽ trình bày sơ đồ mạch cụ thể để hiện thực hoá ý t- ởng trên.

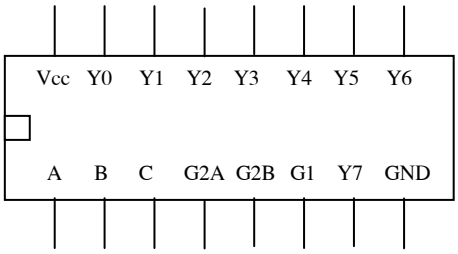
5. SƠ ĐỒ MẠCH GHÉP NỐI VÀO RA

Sơ đồ mạch điện (*xem cuối ch- ơng)* trình bày sơ đồ nguyên lý của mạch ghép nối vào/ra với máy tính thông qua slot ISA của máy tính. Ngoại trừ biến tần và Encoder, toàn bộ các thành phần còn lại của hệ thống điều khiển động cơ không đồng bộ ba pha đều đ- ợc đặt trên card ghép nối này. Giữa card và các thiết bị ngoài (bộ biến tần và Encoder) đ- ợc nối với nhau thông qua một jack cắm loại DB-9. Mục 3 sau đây sẽ trình bày cụ thể hơn về các linh kiện, các trị số đ- ợc dùng trong mạch.

3. GIẢI THÍCH SƠ ĐỒ MẠCH NGUYÊN LÝ

Sơ đồ mạch nguyên lý có các thành phần cơ bản sau đây:

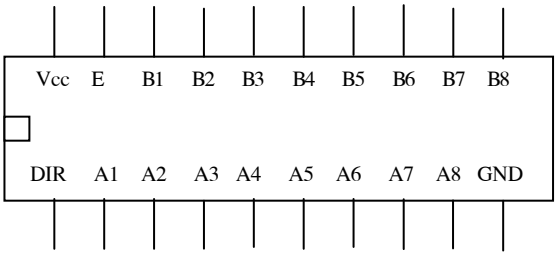
- ✓ Vi mạch giải mã địa chỉ 74LS138: đây là vi mạch giải mã 3-8 với ba đầu vào lựa chọn A,B,C. Tám đầu ra có mức tích cực thấp: có mức điện áp thấp khi tích cực và ng- ợc lại. Có ba đầu vào cho phép, trong đó có hai đầu tích cực thấp ($\overline{G2A}, \overline{G2B}$) và một đầu tích cực cao (G1). Sơ đồ chân và bảng chân lý của vi mạch giải mã đ- ợc trình bày ở hình sau:



Vào			Ra							
EN	C	B	7	6	5	4	3			
	A		2	1	0					
0	x	x	1	1	1	1	1			
	x		1	1	1					
1	0	0	1	1	1	1	1			
	0		1	1	0					
1	0	0	1	1	1	1	1			
	1		1	0	1					
1	0	1	1	1	1	1	1			
	0		0	1	1					
1	0	1	1	1	1	1	0			
	1		1	1	1					
1	1	0	1	1	1	0	1			
	0		1	1	1					
1	1	0	1	1	0	1	1			
	1		1	1	1					
1	1	1	1	0	1	1	1			
	0		1	1	1					
1	1	1	0	1	1	1	1			
	1		1	1	1					

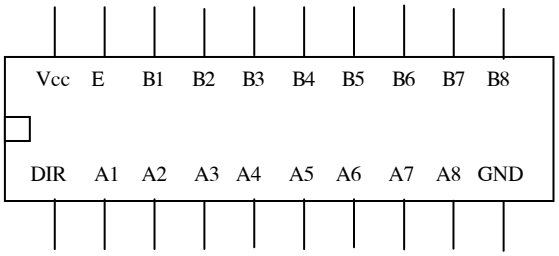
$$EN = \overline{G2A} + \overline{G2B} + G1$$

- ✓ Bộ đệm 74LS245 *(Bus Transceiver, Octal Bus, 3 states)*: vi mạch này gồm tám bộ đệm ba trạng thái hai chiều, th- ờng đ- ợc dùng làm đệm hai chiều cho Bus dữ liệu. Các chân A0-A7 và B0-B7 là các chân vào/ra hai chiều. Chân DIR xác định chiều đi của dữ liệu: DIR=1 thì chiều từ A sang B, DIR=0 thì chiều từ B sang A. Chân E tích cực thấp sẽ cho phép hay không cho phép đ- a dữ liệu qua. Khi E=0, các đầu dữ liệu ra sẽ giống các đầu dữ liệu vào t- ơng ứng (nh- ng có khả năng chịu tải tăng lên, tức là công suất tăng lên). Khi E=1, các đầu ra đều bị treo ở trạng thái cao trở. Sơ đồ chân của vi mạch này đ- ợc trình bày ở hình sau.



- ✓ Bộ chốt hexa 74LS373: vi mạch gồm tám flip-flop loại D ba trạng thái, th- ờng đ- ợc dùng để chốt dữ liệu. Khi chân LE ở mức cao, dữ liệu từ các đầu vào (Dx) đ- ợc chốt vào trong flip-flop, sẵn sàng để đ- a ra ở các đầu ra t- ơng ứng (Qx). Khi chân LE ở mức thấp, các đầu vào không ảnh

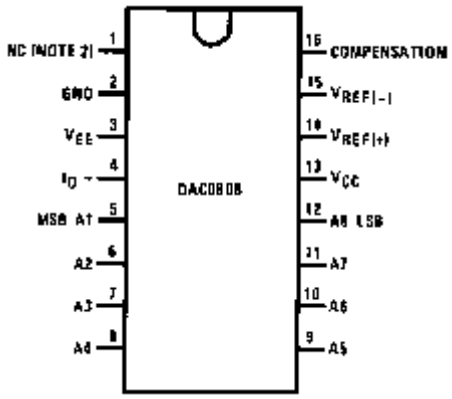
h- ởng gì đến các đầu ra. Khi chân /OE (tích cực thấp) ở mức thấp, các đầu ra đ- ợc đ- a dữ liệu ra. Khi chân /OE ở mức cao, các đầu ra đ- ợc đặt lên trạng thái cao trở, cách li khỏi bus. Sau đây là sơ đồ chân và bảng chân lý của 74LS373.



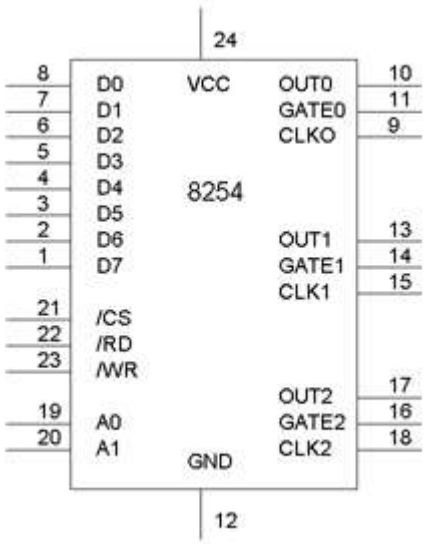
EN	LE	Vào Dx	Ra Qx
1	x	x	High Z
0	1	1	1
0	1	0	0
0	0	x	Qx

✓ Vi mạch 74LS02: đây là vi mạch gồm bốn cổng NOR hai đầu vào.

✓ Vi mạch DAC0808: là vi mạch chuyển đổi số - t- ơng tự (DAC) 8-bit. Thời gian chuyển đổi tối đa là 150 ns. Tiêu thụ công suất 33 mW với nguồn nuôi ±5V. Đầu ra là dòng điện có giá trị phụ thuộc vào mã nhị phân đ- a vào. DAC0808 t- ơng thích và giao tiếp đ- ợc với các mức logic TTL, DTL hay CMOS. Nguồn nuôi của vi mạch cho phép trong khoảng từ -18V đến +18V. Dòng vào so sánh là 2-5 mA. Nhiệt độ làm việc bình th- ờng là 0-75°C. Trong các ứng dụng thông th- ờng, chân 14 đ- ợc nối lên điện áp so sánh Vref thông qua điện trở R_{14} có giá trị sao cho Vref/ R_{14} = 2 mA. Một khuếch đại thuật toán đ- ợc sử dụng để chuyển dòng điện đầu ra (chân 4) thành điện áp ra thích hợp. Sơ đồ chân của DAC0808 đ- ợc trình bày ở hình bên.



✓ Bộ đếm - thời gian khả trình 8254: đây là vi mạch đếm - thời gian khả trình rất mạnh của Intel. Vi mạch này đ- ợc ứng dụng rộng rãi để: đếm thời gian, đếm sự kiện, chia tần số, tạo xung,... Có thể thấy vi mạch này rất hữu dụng trong mạch đếm xung Encoder. So với vi mạch 8253, vi mạch 8254 có tần số làm việc tối đa lớn hơn (đến 4 MHz), có thêm lệnh đọc ng- ợc (Read-back command) sẽ đ- ợc dùng đến trong bài tập này. Trong 8254 có ba bộ đếm độc lập với nhau, đ- ợc đánh số từ 0 đến 2. Ngoài ra trong 8254 còn có một thanh ghi từ điều khiển (CWR – Control Word Register) dùng để lập trình cho hoạt động



của vi mạch. Nh- vậy, 8254 cần 4 địa chỉ cổng để giao tiếp, bao gồm 3 địa chỉ cho 3 bộ đếm và 1 địa chỉ cho CWR. Mỗi bộ đếm có 6 chế độ làm việc. Trong bài tập này có sử dụng 2 chế độ là: chế độ 3 (tạo xung vuông) và chế độ 2. Với chế độ 3, khi đã nạp số đếm và GATE=0, bộ đếm bắt đầu đếm lùi theo b- ớc 2 (mỗi nhịp Clock giảm số đếm đi 2); khi về 0, chân OUT đảo trạng thái và số đếm đ- ợc nạp lại để đếm lại từ đầu. Nh- vậy tín hiệu ở OUT là xung vuông với tần số f_{CLK}/N. Với chế độ 2, khi đã nạp số đếm và GATE=1, bộ đếm bắt đầu đếm lùi; khi đếm hết hoặc khi GATE có s- ờn lên thì số đếm đ- ợc nạp lại và đếm lại từ đầu; nếu đang đếm mà GATE=0 thì ngừng đếm và chốt số đếm lại. Nh- vậy, chế độ 3 đ- ợc dùng với bộ đếm chia tần, còn chế độ 2 đ- ợc dùng với các bộ đếm đếm xung Encoder.

Dạng thức thanh ghi từ điều khiển (CWR):

SC1	SC0	RW1	RW0	M2	M1	M0	BCD

Trong đó:

- SC1, SC0 để chọn bộ đếm hoặc lệnh đọc ng- ợc (11)
- RW1, RW0 để chọn chế độ ghi/đọc. Chế độ 11 là chế độ đọc / ghi LSB rồi đến MSB
- M2, M1, M0 để chọn chế độ đếm: 010 cho chế độ 2 và 011 cho chế độ 3.
- BCD bằng 0 nếu đếm theo Binary, bằng 1 nếu đếm theo BCD

Dạng thức thanh ghi từ điều khiển ở lệnh đọc ng- ợc:

1	1	CNT	STS	C2	C1	C0	0

Trong đó:

- CNT = 0 nếu chốt số đếm
- STS = 0 nếu chốt trạng thái
- C2,C1,C0: để chọn bộ đếm nào đ- ợc chốt (cho bằng 1)

Khi đọc ng- ợc trạng thái, bit 7 (MSBit) là trạng thái chân OUT của bộ đếm t- ơng ứng. Hình trên là sơ đồ của 8254.

✓ Vi mạch LM555: đây là vi mạch thời gian rất hữu dụng. Sử dụng LM555 có thể tạo ra các xung vuông, xung One-shot, tạo trễ thời gian,... với độ chính xác rất cao và rất linh hoạt. Không chỉ tạo đ- ợc các xung đều đặn, LM555 còn có thể tạo ra các xung phức tạp nếu điều khiển chân CV. LM555 hoạt động với giải điện áp rộng (3V-15V). Trong bài tập này, LM555 đ- ợc dùng để tạo xung đếm chuẩn 10KHz.

- ✓ Encoder OMRON E6B2-CWZ3E: đây là loại Encoder đa mục đích (General Purpose) của OMRON. Chỉ tiêu kĩ thuật:

    - Khả năng chịu tải lớn.
    - Điện áp nguồn: từ 5V đến 24V.
    - Độ phân giải: 2000 xung/vòng.
    - Dòng tiêu thụ tối đa: 100mA.
    - Tần số làm việc tối đa: 100KHz.
    - Tốc độ quay tối đa: 6000 vòng/phút.
    - Đầu ra: điện áp trực tiếp, không cần điện trở treo.
    - Kết nối trực tiếp đ- ợc với các thiết bị TTL, LSTTL, CMOS mà không cần cách li.

<Tham khảo: "OMRON Encoder Datasheet">

<u>Giải thích sơ đồ nguyên lý:</u>

Hai vi mạch 74LS138 làm nhiệm vụ giải mã địa chỉ. Vi mạch 8254 đ- ợc chọn theo dải địa chỉ 310h-313h, với các chân địa chỉ A0, A1 đ- a thẳng vào 8254 để chọn bộ đếm và CWR. Vi mạch DAC0808 đ- ợc chọn (gián tiếp thông qua bộ đệm 74LS373) ở một trong các địa chỉ 314h-317h. Vi mạch này đ- ợc giải mã không đầy đủ, nghĩa là khi ghi vào bất kì cổng nào trong dải 314h-317h đều là giao tiếp với DAC0808. Tr- ớc DAC0808 có một bộ chốt 74LS373 để chốt dữ liệu hiện thời cho DAC, và nó chỉ cập nhập giá trị mới khi ghi (IOW=0) vào một trong các cổng 314h-317h. Một vi mạch đệm hai chiều 74LS245 đ- ợc dùng để đệm Bus dữ liệu cho cả card.

Vi mạch 8254: chân GATE2 đ- ợc nối xuống đất để bộ đếm 2 luôn đếm khi đã đ- ợc nạp số đếm. Chân CLK2 đ- ợc nối với đầu ra của vi mạch LM555, cung cấp xung đếm chuẩn. Bộ đếm 2 sẽ đ- ợc lập trình để chia xung này ra thành xung có tần số nhỏ hơn để cung cấp thời gian đếm cho hai bộ đếm còn lại. Đầu ra OUT2 đ- ợc đ- a vào chân GATE của 2 bộ đếm còn lại để cho phép hay không cho phép hai bộ đếm này đếm. Tr- ớc khi đi vào GATE1, tín hiệu này đ- ợc đảo đi để hai bộ đếm 0 và 1 luôn làm việc lệch nhau. Tín hiệu xung từ Encoder đ- ợc đ- a vào CLK0 và CLK1.

Vi mạch LM555 đ- ợc thiết lập ở chế độ tạo xung vuông với tần số đ- ợc tính $f = \frac{1.44}{(VR3 + 2VR4)C4}$. Chọn C = 0,01μF thì để có f = 10KHz phải chỉnh VR3 và VR4 sao cho VR3 + 2VR4 = 14,4KΩ. Tụ C3 = 10nF đ- ợc sử dụng để lọc nhiễu khỏi chân CV, giúp ổn định xung đầu ra.

Vi mạch DAC0808: điện áp so sánh d- ơng (chân Vref(+)) đ- ợc nối lên nguồn +12V thông qua điện trở R1 (1KΩ) và biến trở VR1 (5KΩ). Điều chỉnh

biến trở để dòng điện I_{14} = Vref(+) / (R1 + VR1) nằm trong khoảng 2-5mA. Th- ờng điều chỉnh sao cho I_{14} = 2mA. Tụ điện C2 (0,1μF) đ- ợc dùng để lọc các nhiễu tần số cao trong nguồn xuống đất, giúp ổn định điện áp so sánh. Chân Vref(-) đ- ợc nối xuống điện áp -12V. Đầu ra (dòng điện) của DAC0808 đ- ợc đ- a qua một khuếch đại thuật toán (LM358) để biến đổi dòng điện thành điện áp điều khiển đầu ra. Giả sử số đặt vào DAC0808 là n (n = 0..255) thì dòng điện đầu ra đ- ợc tính theo công thức $I_{out} = \frac{V_{ref}}{R_1 + VR_1}\frac{n}{255}$. Điện áp đầu ra đ- ợc tính theo công thức $u_{out} = u_{dk} = (R_2 + VR_2)I_{out} = \frac{R_2 + VR_2}{R_1 + VR_1}\frac{n}{255}V_{ref}$. Nếu điều chỉnh sao cho $R_2+VR_2=R_1+VR_1$ thì $u_{out} = u_{dk} = \frac{n}{255}V_{ref}$.

Cổng nối DB-9 đ- ợc dùng để kết nối giữa Card và các thiết bị bên ngoài. Các chân 1 và 2 đ- ợc dùng để đ- a điện áp Vcc ra ngoài. Chân 3 dùng để đ- a điện áp điều khiển ra biến tần. Các chân 4, 5 là các chân đ- a tín hiệu xung Encoder vào trong máy tính. Các chân 6,7,8,9 đ- ợc dùng để nối chung đất giữa mạch ngoài và trong máy tính.

Bên mạch ngoài, biến tần đ- ợc nối chung đất với mạch trong (chân 6) và đầu *Control* đ- ợc nối với điện áp điều khiển từ máy tính ra (chân 3). Encoder đ- ợc nối chung đất với mạch trong máy tính (chân 9) và đ- ợc cấp nguồn qua chân 2. Các chân tín hiệu xung ra của Encoder đ- ợc nối với chân 4 và 5 của DB-9.

CH□ ƠNG III

THIẾT KẾ PHẦN MỀM HỆ THỐNG ĐIỀU KHIỂN SỐ ĐỘNG CƠ KĐB 3 PHA

1. PHƯƠNG TRÌNH SAI PHÂN CỦA BỘ ĐIỀU KHIỂN

Nh- đã trình bày ở ch- ơng 1, phần 3, bộ điều khiển PID có hàm truyền đạt nh- sau:

$$W_{pid}(z) = \frac{Az^2 + Bz + C}{z(z-1)}$$

trong đó:

$$A = \frac{(K_i T^2 + 2K_d + 2K_p T)}{2T}$$

$$B = \frac{(K_i T^2 - 2K_p T - 4K_d)}{2T}$$

$$C = \frac{K_d}{T}$$

Từ hàm truyền đạt này, chuyển thành ph- ơng trình sai phân:

$$W_{pid}(z) = \frac{U(z)}{E(z)} = \frac{Az^2 + Bz + C}{z^2 - z} = \frac{A + B.z^{-1} + C.z^{-2}}{1 - z^{-1}}$$

$$\Rightarrow E(z).(A + B.z^{-1} + C.z^{-2}) = U(z)(1 - z^{-1})$$

$$\Rightarrow A.e(k) + B.e(k-1) + C.e(k-2) = u(k) - u(k-1)$$

$$\Rightarrow u(k) = u(k-1) + A.e(k) + B.e(k-1) + C.e(k-2)$$

Từ ph- ơng trình sai phân trên có đ- ợc thuật toán xây dựng bộ PID số nh- sau:

- Khai báo 1 biến để l- u giá trị cũ của *u* và 1 mảng để l- u các giá trị cũ của *e* :

```
float u_old;
float e_old[2];
```

  Các giá trị này đều đ- ợc khởi đầu bằng 0.

  Tính A,B,C theo Kp, Ti, Td đã chọn.

- Các b- ớc tính cho bộ PID số:

  - Đọc giá trị hồi tiếp về *y* và giá trị đặt *w*. Tính sai lệch *e* = *w* - *y*.
  - Tính giá trị tín hiệu điều khiển hiện thời:

```
u = u_old + A*e + B*e[0] + C*e[1];
```

- Xuất tín hiệu điều khiển ra.
- Cập nhập lại các biến l- u:

```
u_old = u;
e[1]=e[0];
e[0]=e;
```

- Lặp lại từ đầu.

2. PHƯƠNG ÁN XÂY DỰNG CHƯƠNG TRÌNH ĐIỀU KHIỂN.

Ch- ơng trình điều khiển và giao diện bao gồm hai phần độc lập: phần giao diện ng- ời dùng và phần thực hiện thuật toán điều khiển.

- Phần giao diện ng- ời dùng:
  - Hiển thị kết quả quá trình điều khiển, ở đây là hiển thị tốc độ của động cơ d- ới dạng số và đồ thị.
  - Nhận tốc độ đặt (tốc độ mong muốn) từ ng- ời dùng.
  - Nhận các thông số thiết lập cho hệ thống từ ng- ời dùng.
  - Yêu cầu của phần này là giao diện phải dễ dùng, thuận tiện cho ng- ời sử dụng. Giao diện phải hợp lý. Một phong cách chung trong giao diện điều khiển là mô phỏng các bàn điều khiển thiết bị (Instrument Panel).
- Phần thực hiện thuật toán điều khiển:
  - Thực hiện thuật toán điều khiển (nh- đã trình bày ở phần trên).
  - Giao tiếp với phần cứng để điều khiển đối t- ợng cũng nh- nhận các kết quả đo đạc từ các Sensor.
  - Nhận thông số hệ thống và giá trị đặt từ ng- ời dùng thông qua phần giao diện.
  - Cung cấp các số liệu cho phần giao diện (tốc độ).
  - Yêu cầu của phần này là tốc độ phải nhanh, đáp ứng đ- ợc yêu cầu về thời gian của hệ thống điều khiển. Ngoài ra, việc thực hiện thuật toán điều khiển phải chính xác.

Căn cứ vào các yêu cầu trên, để thuận tiện cho phần giao diện, ch- ơng trình sẽ đ- ợc viết trên môi tr- ờng Windows. Môi tr- ờng hệ điều hành Windows là một môi tr- ờng đồ hoạ hoàn thiện, cung cấp rất nhiều các công cụ phát triển cũng nh- các thành phần đồ hoạ cơ bản (menu, cửa sổ, hộp thoại, ...) giúp cho việc phát triển các ứng dụng đ- ợc dễ dàng và nhanh chóng. Mặc dù môi tr- ờng Windows luôn hạn chế việc truy nhập cấp thấp với phần cứng, tuy nhiên trên Windows 9x thì việc thực hiện các vào ra cơ bản với phần cứng là đ- ợc phép. Điều này hoàn toàn đáp ứng đ- ợc yêu cầu trong bài tập này.

Môi tr- ờng Windows là môi tr- ờng đa nhiệm, nghĩa là các ch- ơng trình chạy trên đó đ- ợc thực hiện gần nh- là đồng thời và chia sẻ tài nguyên với nhau. Bởi lẽ đó, khi thực hiện phần điều khiển, phần ch- ơng trình yêu cầu tốc độ nhanh và đáp ứng kịp thời, cần phải tăng mức - u tiên (priority) cho phần mã đó lên mức cao hơn bình th- ờng. Mức - u tiên *High* tỏ ra là phù hợp, tuy nhiên nếu cần thiết có thể tăng lên mức - u tiên *Real-time*. Ngoài ra, việc sử dụng Thread cho riêng phần điều khiển là cần thiết để tránh bị ảnh h- ởng bởi phần giao diện. Nh- vậy, ch- ơng trình sẽ có hai phần chính t- ơng ứng với hai Thread khác nhau của cùng một Process.

3. CHỌN CÔNG CỤ LẬP TRÌNH

Với các bài toán điều khiển, ngôn ngữ C/C++ d- ờng nh- là lựa chọn bắt buộc, bởi ngôn ngữ này cho phép viết các ch- ơng trình mạnh, nhanh, nhỏ gọn, truy nhập sâu vào phần cứng (về mặt này thì không bằng hợp ngữ). Tuy nhiên, để tối - u phần mã điều khiển, một số đoạn trình hợp ngữ sẽ đ- ợc sử dụng thêm vào đoạn mã C/C++.

Về công cụ và môi tr- ờng phát triển: hiện nay có hai công cụ phát triển rất mạnh dùng ngôn ngữ C/C++, đó là Visual C++ của Microsoft và CBuilder của Inprise (tên mới của Borland). Trong khi Visual C++ - u tiên khả năng can thiệp sâu vào hệ thống và chỉ đóng gói đơn giản các thành phần (đồ hoạ, file,...) của hệ thống thì CBuilder lại tận dụng tối đa khả năng h- ớng đối t- ợng trong C++ để đóng gói các thành phần hệ thống, giúp lập trình viên càng ít phải can thiệp chi tiết vào hệ thống càng tốt. Tất nhiên CBuilder sẽ có chút hạn chế khi lập trình viên muốn lập trình cấp thấp. Với mục đích nhanh chóng tạo ra ch- ơng trình với giao diện phù hợp, dễ sử dụng mà vẫn đảm bảo yêu cầu về tốc độ, tính hiệu quả, công cụ CBuilder đ- ợc chọn để thực hiện phần mềm cho bài tập này.

4. MÃ NGUỒN CỦA CHƯƠNG TRÌNH

Ch- ơng trình nguồn đ- ợc viết d- ới dạng một project trong CBuilder. Hệ thống module trong project đ- ợc tổ chức nh- sau:

Tên file	Chú thích
ACMotor.bpr	File Project
ACMotor.cpp, ACMotor.res	File module chính và file resource
UMain.*	Các file nguồn cho module của form chính
UAbout.*	Các file nguồn cho module của hộp thoại About
USettings.*	Các file nguồn cho module của hộp thoại Settings
common.h, common.cpp	Module chứa các đinh nghĩa biến chung
UControl.*	Module chứa Thread Class để điều khiển hệ thống

Nội dung mã nguồn cụ thể nh- sau:

a. Module chính:

File ACMotor.cpp

```
#include <vcl.h>
#pragma hdrstop
USERES("ACMotor.res");
USEFORM("UMain.cpp", frmMain);
USEUNIT("UControl.cpp");
USEFORM("UAbout.cpp", AboutBox);
USEFORM("USettings.cpp", dlgSettings);
USEUNIT("common.cpp");
USEUNIT("..\..\userlib\GraphScope.cpp");

WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
  try
  {
    Application->Initialize();
    Application->Title = "Three-phase Motor Digital
Controller";
    Application->CreateForm(__classid(TfrmMain), &frmMain);
    Application->CreateForm(__classid(TAboutBox), &AboutBox);
    Application->CreateForm(__classid(TdlgSettings),
&dlgSettings);
    Application->Run();
  }
  catch (Exception &exception)
  {
    Application->ShowException(&exception);
  }
  return 0;
}
//---------------------------------------------------------
-----
```

b. Module của form chính:

File UMain.h

```
//---------------------------------------------------------
---------------------

#ifndef UMainH
#define UMainH
//---------------------------------------------------------
```

```
----------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
#include "GraphScope.h"
#include "UControl.h"
//---------------------------------------------------
----------
class TfrmMain : public TForm
{
__published: // IDE-managed Components
  TPanel *pnlitle;
  TLabel *lblTitle;
  TPanel *pnlSystem;
  TPanel *pnlButtons;
  TButton *btnSettings;
  TButton *btnControl;
  TButton *btnExit;
  TButton *btnAbout;
  TPanel *pnlControlPanel;
  TPanel *pnlGraph;
  TTimer *timSpeedUpdate;
  TPanel *pnlParams;
  TRadioButton *PController;
  TRadioButton *PIController;
  TRadioButton *PDController;
  TRadioButton *PIDController;
  TLabel *Label1;
  TEdit *KpEdit;
  TEdit *TiEdit;
  TEdit *TdEdit;
  TLabel *Label2;
  TLabel *Label3;
  TLabel *Label4;
  TPanel *Panel1;
  TLabel *btnDefault;
  TLabel *Label5;
  TLabel *curSpeed;
  TLabel *Label7;
  TLabel *Label6;
  TLabel *desiredSpeed;
```

```
  TButton *SetNewSpeed;
  void __fastcall FormCreate(TObject *Sender);
  void __fastcall btnAboutClick(TObject *Sender);
  void __fastcall btnExitClick(TObject *Sender);
  void __fastcall FormDestroy(TObject *Sender);
  void __fastcall btnControlClick(TObject *Sender);
  void __fastcall timSpeedUpdateTimer(TObject *Sender);
  void __fastcall PControllerClick(TObject *Sender);
  void __fastcall btnDefaultClick(TObject *Sender);
  void __fastcall btnSettingsClick(TObject *Sender);
  void __fastcall SetNewSpeedClick(TObject *Sender);
private:
  TControlThread *CtrlThread;
public:      // User declarations
  TGraphScope *SpeedScope;
  __fastcall TfrmMain(TComponent* Owner);
  void __fastcall ChangeControlState();
  __fastcall ~TfrmMain();
protected:
};
//---------------------------------------------------------
----------
extern PACKAGE TfrmMain *frmMain;

#endif
```

File UMain.cpp

```
//---------------------------------------------------------
----------
#include <vcl.h>
#pragma hdrstop

#include <stdlib.h>
#include "UMain.h"
#include "UAbout.h"
#include "common.h"
#include <time.h>
#include "USettings.h"

//---------------------------------------------------------
----------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmMain *frmMain;
```

```
//---------------------------------------------------
----------
__fastcall TfrmMain::TfrmMain(TComponent* Owner)
  : TForm(Owner)
{
  ::randomize();
  ThreadProtect = new TCriticalSection;
}
//---------------------------------------------------
----------
void __fastcall TfrmMain::FormCreate(TObject *Sender)
{
  btnDefault->Caption = "D\nE\nF\nA\nU\nL\nT";
  SpeedScope = new TGraphScope(this);
  SpeedScope->Parent = pnlGraph;
  SpeedScope->Align = alClient;
}
//---------------------------------------------------
----------

void __fastcall TfrmMain::btnAboutClick(TObject *Sender)
{
  AboutBox->Show();
}
//---------------------------------------------------
----------void __fastcall TfrmMain::btnExitClick(TObject
*Sender)
{
  if (bControlling)
  {
    CtrlThread->Terminate();
    timSpeedUpdate->Enabled = false;
    lblTitle->Caption = "Terminating...";
    Cursor = crHourGlass;
    CtrlThread->WaitFor();
  }
  Close();
}
//---------------------------------------------------
----------
void __fastcall TfrmMain::FormDestroy(TObject *Sender)
{
  delete SpeedScope;
}
```

```
//---------------------------------------------------
----------

void __fastcall TfrmMain::ChangeControlState()
{
  if (!bControlling)
    btnControl->Caption = "&Control";
  else
    btnControl->Caption = "Stop &Control";
  btnControl->Enabled = true;
}
void __fastcall TfrmMain::btnControlClick(TObject
*Sender)
{
  if (bControlling)
  {
    CtrlThread->Terminate();
    timSpeedUpdate->Enabled = false;
    pnlParams->Enabled = true;
    btnSettings->Enabled = true;
    btnDefault->Enabled = true;
  }
  else
  {
    SpeedScope->NumValue = 0;
    SpeedScope->AddValue(0.0, 0.0);
    Kp = StrToFloat(KpEdit->Text);
    Ki = StrToFloat(TiEdit->Text);
    Kd = StrToFloat(TdEdit->Text);
    CtrlThread = new TControlThread(false);
    StartControlTime = time(NULL);
    timSpeedUpdate->Enabled = true;
    pnlParams->Enabled = false;
    btnSettings->Enabled = false;
    btnDefault->Enabled = false;
  }
  btnControl->Enabled = false;
}
//---------------------------------------------------
----------
void __fastcall TfrmMain::timSpeedUpdateTimer(TObject
*Sender)
{
  double myspeed;
```

```
  ThreadProtect->Acquire();
  myspeed = dSpeed;
  ThreadProtect->Release();
  SpeedScope->AddValue(difftime(time(NULL),
StartControlTime), myspeed);
  curSpeed->Caption = FloatToStr(dSpeed);
}
//---------------------------------------------------
----------

__fastcall TfrmMain::~TfrmMain()
{
  delete ThreadProtect;
}

void __fastcall TfrmMain::PControllerClick(TObject
*Sender)
{
  TiEdit->Enabled = false;
  TdEdit->Enabled = false;
  if (PIDController->Checked || PIController->Checked)
    TiEdit->Enabled = true;
  if (PIDController->Checked || PDController->Checked)
    TdEdit->Enabled = true;
}
//---------------------------------------------------
----------
void __fastcall TfrmMain::btnDefaultClick(TObject
*Sender)
{
  if (PIDController->Checked)
  {
    KpEdit->Text = "0.01767";
    TiEdit->Text = "0.15";
    TdEdit->Text = "0.00026";
  }
  else if (PIController->Checked)
  {
    KpEdit->Text = "1";
    TiEdit->Text = "0.1";
  }
  else if (PDController->Checked)
  {
    KpEdit->Text = "1";
```

```
    TdEdit->Text = "0.01";
  }
  else
    KpEdit->Text = "1";
}
//---------------------------------------------------
----------

void __fastcall TfrmMain::btnSettingsClick(TObject
*Sender)
{
  if (dlgSettings->ShowModal() == mrOk)
  {
    NumPulse = StrToInt(dlgSettings->EncNumPulse->Text);
    TSamp = dlgSettings->SampTime->Value / 1000.0;
  }
}
//---------------------------------------------------
----------

void __fastcall TfrmMain::SetNewSpeedClick(TObject
*Sender)
{
  AnsiString value;
  double newSpeed, oldSpeed;
  ThreadProtect->Acquire();
  oldSpeed = desSpeed;
  ThreadProtect->Release();
  value = FloatToStr(oldSpeed);
  if (InputQuery("Set Desired Speed", "Enter the desired
speed", value))
  {
    newSpeed = StrToFloat(value);
    if (newSpeed >= 0 && newSpeed <= 3000 && newSpeed !=
oldSpeed)
    {
      ThreadProtect->Acquire();
      desSpeed = newSpeed;
      ThreadProtect->Release();
      desiredSpeed->Caption = FloatToStr(newSpeed);
    }
  }
}
```

c. Module của hộp thoại About:

File UAbout.h

```
#ifndef UAboutH
#define UAboutH
//----------------------------------------------------
----------
#include <vcl\System.hpp>
#include <vcl\Windows.hpp>
#include <vcl\SysUtils.hpp>
#include <vcl\Classes.hpp>
#include <vcl\Graphics.hpp>
#include <vcl\Forms.hpp>
#include <vcl\Controls.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Buttons.hpp>
#include <vcl\ExtCtrls.hpp>
//----------------------------------------------------
----------
class TAboutBox : public TForm
{
__published:
    TPanel *Panel1;
    TLabel *ProductName;
    TLabel *Copyright;
    TButton *OKButton;
  TMemo *Memo1;
  TLabel *Notice;
  void __fastcall FormClose(TObject *Sender,
TCloseAction &Action);
  void __fastcall OKButtonClick(TObject *Sender);
private:
public:
    virtual __fastcall TAboutBox(TComponent* AOwner);
};
//----------------------------------------------------
----------
extern PACKAGE TAboutBox *AboutBox;
//----------------------------------------------------
----------
#endif
```

File UAbout.cpp

```
//----------------------------------------------------
```

```
----------
#include <vcl.h>
#pragma hdrstop

#include "UAbout.h"
//---------------------------------------------------------
----------
#pragma resource "*.dfm"
TAboutBox *AboutBox;
//---------------------------------------------------------
----------
__fastcall TAboutBox::TAboutBox(TComponent* AOwner)
    : TForm(AOwner)
{
}
//---------------------------------------------------------
----------

void __fastcall TAboutBox::FormClose(TObject *Sender,
TCloseAction &Action)
{
  Action = caHide;
}
//---------------------------------------------------------
----------

void __fastcall TAboutBox::OKButtonClick(TObject
*Sender)
{
  Close();
}
//---------------------------------------------------------
----------
```

d. Module chứa Thread Class điều khiển:

File UControl.h

```
//---------------------------------------------------------
----------

#ifndef UControlH
#define UControlH
//---------------------------------------------------------
```

```
----------
#include <Classes.hpp>
#include "GraphScope.h"

//---------------------------------------------------
----------
class TControlThread : public TThread
{
private:
   __int64 SmplCnt;  //Sampling Time Count Value
protected:
  void __fastcall Execute();
  void __fastcall RunWhenTerminate(TObject *Sender);
  void __fastcall ChangebControlling();
public:
   __fastcall TControlThread(bool CreateSuspended);
};
//---------------------------------------------------
----------
#endif
```

File UControl.cpp

```
//---------------------------------------------------
----------
#include <vcl.h>
#pragma hdrstop

#include <stdlib.h>
#include "UControl.h"
#include "common.h"
#include "UMain.h"

#pragma package(smart_init)
//---------------------------------------------------
----------

//   Important: Methods and properties of objects in VCL
can only //     be
//   used in a method called using Synchronize, for
example:
//
//      Synchronize(UpdateCaption);
//
```

```
//   where UpdateCaption could look like:
//
//      void __fastcall Unit1::UpdateCaption()
//      {
//        Form1->Caption = "Updated in a thread";
//      }
//---------------------------------------------------
----------

__fastcall TControlThread::TControlThread(bool
CreateSuspended)
  : TThread(CreateSuspended)
{
  Priority = tpHighest;
  FreeOnTerminate = true;
  OnTerminate = RunWhenTerminate;
  //Khoi tao phan cung
  //So dem cho 8254
  unsigned short int Count8254 = TSamp * 20000;
  asm {
    MOV   DX, 0x313
    MOV   AL, 0xB6
    OUT   DX, AL

    MOV   DX, 0x312
    MOV   AX, Count8254
    OUT   DX, AL

    XCHG  AH, AL
    OUT   DX, AL
    //Counter0: mode 2, 16bit, value=FFFF
    MOV   DX, 0x313
    MOV   AL, 0x34
    OUT   DX, AL
    MOV   DX, 0x310
    MOV   AL, 0xFF
    OUT   DX, AL
    NOP           //delay
    OUT   DX, AL
    //Counter1: mode 2, 16bit, value=FFFF
    MOV   DX, 0x313
    MOV   AL, 0x74
    OUT   DX, AL
    MOV   DX, 0x311
```

```
    MOV   AL, 0xFF
    OUT   DX, AL
    NOP           //delay
    OUT   DX, AL
  }
}
//---------------------------------------------------
----------
void __fastcall TControlThread::Execute()
{
  double u_old, e_old[2];
  double y, w, u, e;
  double A,B,C;
  Synchronize(ChangebControlling);
  u_old = 0.0;
  e_old[0] = e_old[1] = 0.0;
  A = (Ki*TSamp*TSamp + 2*Kd + 2*Kp*TSamp)/2/TSamp;
  B = (Ki*TSamp*TSamp - 2*Kp*TSamp - 4*Kd)/2/TSamp;
  C = Kd/TSamp;
  _LARGE_INTEGER freq;
  if (!(::QueryPerformanceFrequency(&freq)))
  {
    MessageDlg("No high-accuracy timer found.", mtError,
TMsgDlgButtons() << mbOK, 0);
    return;
  }

  _LARGE_INTEGER lastTime;
  ::QueryPerformanceCounter(&lastTime);
  _LARGE_INTEGER curTime = lastTime;
  SmplCnt = TSamp * freq.QuadPart;
  register unsigned short int PulsesCounted;
  double heso = 60.0/(NumPulse*TSamp);
  double heso_u = 255.0/12;
  unsigned char udk;
  while (!Terminated)
  {
    while (curTime.QuadPart - lastTime.QuadPart <
SmplCnt)
      ::QueryPerformanceCounter(&curTime);
    //Doc toc do
    asm {
      //Read-back command, latch Counter 2 status
      MOV   DX, 0x313
```

```
      MOV   AL, 0xE8
      OUT   DX, AL
      //Read status
      MOV   DX, 0x312
      IN    AL, DX
      MOV   AH, AL  //save status2 to AH
      //Read-back command, latch Counter 0 & 1
      MOV   DX, 0x313
      MOV   AL, 0xD6
      OUT   DX, AL

      MOV   AL, AH  //restore status2
      MOV   CL, 7   //Shift Right to get OUT2
      SHR   AL, CL
      XOR   AH, AH  //now, AX contains the offset of the
latch counter

      MOV   DX, 0x310
      ADD   DX, AX  //now DX contains the port addr of
the latch counter
      //And read count value now
      IN    AL, DX
      MOV   AH, AL
      IN    AL, DX
      XCHG  AH, AL
      //FFFF-AX = number of pulses counted
      NOT   AX
      MOV   PulsesCounted, AX
    }
    y = PulsesCounted*heso;
    ThreadProtect->Acquire();
    dSpeed = y;
    w = desSpeed;
    ThreadProtect->Release();
    //Thuat toan dieu khien
    e = w - y;
    u = u_old + A*e + B*e_old[0] + C*e_old[1];
    //Va xuat tin hieu dieu khien ra
    u_dk = u*heso_u;
    asm {
      MOV   DX, 0x314
      MOV   AL, u_dk
      OUT   DX, AL
    }
```

```
    u_old = u;
    e_old[1] = e_old[0];
    e_old[0] = e;
    lastTime = curTime;
  }
}
//---------------------------------------------------
----------

void __fastcall TControlThread::RunWhenTerminate(TObject
*Sender)
{
  Synchronize(ChangebControlling);
}

void __fastcall TControlThread::ChangebControlling()
{
  bControlling = !bControlling;
  frmMain->ChangeControlState();
}
```

e. Module của hộp thoại Settings:

File USettings.h

```
//---------------------------------------------------
----------
#ifndef USettingsH
#define USettingsH
//---------------------------------------------------
----------
#include <vcl\ExtCtrls.hpp>
#include <vcl\Buttons.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Controls.hpp>
#include <vcl\Forms.hpp>
#include <vcl\Graphics.hpp>
#include <vcl\Classes.hpp>
#include <vcl\SysUtils.hpp>
#include <vcl\Windows.hpp>
#include <vcl\System.hpp>
#include "CSPIN.h"
//---------------------------------------------------
----------
class TdlgSettings : public TForm
```

```
{
__published:
    TButton *OKBtn;
    TButton *CancelBtn;
  TGroupBox *GroupBox1;
  TLabel *Label1;
  TEdit *EncNumPulse;
  TButton *Button1;
  TGroupBox *GroupBox2;
  TLabel *Label4;
  TCSpinEdit *SampTime;
  TButton *Button2;
  void __fastcall Button1Click(TObject *Sender);
  void __fastcall Button2Click(TObject *Sender);
private:
public:
    virtual __fastcall TdlgSettings(TComponent* AOwner);
};
//---------------------------------------------------------
----------
extern PACKAGE TdlgSettings *dlgSettings;
//---------------------------------------------------------
----------
#endif
```

File USettings.cpp

```
//---------------------------------------------------------
----------
#include <vcl.h>
#pragma hdrstop

#include "USettings.h"
//---------------------------------------------------------
----------
#pragma link "CSPIN"
#pragma resource "*.dfm"
TdlgSettings *dlgSettings;
//---------------------------------------------------------
----------
__fastcall TdlgSettings::TdlgSettings(TComponent*
AOwner)
    : TForm(AOwner)
{
}
```

```
//---------------------------------------------------
----------

void __fastcall TdlgSettings::Button1Click(TObject
*Sender)
{
  EncNumPulse->Text = "2000";
}
//---------------------------------------------------
----------

void __fastcall TdlgSettings::Button2Click(TObject
*Sender)
{
  SampTime->Value = 5;
}
//---------------------------------------------------
----------
```

f. Module chứa các định nghĩa biến chung:

File common.h

```
#ifndef COMMONH
#define COMMONH
#include <time.h>
#include <syncobjs.hpp>

//bControlling: is the computer controlling the Motor ?
extern bool bControlling;
//dSpeed: current speed (rpm)
extern double dSpeed;
//StartControlTime: the time at which controlling starts
extern time_t StartControlTime;
//ThreadProtect: protect thread crisis
extern TCriticalSection *ThreadProtect;
//NumPulse: Number of Pulse per Rotate
extern unsigned int NumPulse;
//TSamp: Sampling Time
extern double TSamp;
//desSpeed: desired speed
extern double desSpeed;
//Kp, Ki, Kd: parameters of PID controller
extern double Kp, Ki, Kd;
#endif  COMMONH
```

File common.cpp

```
#include "common.h"
bool bControlling = false;
double dSpeed = 0.0;
double desSpeed = 0.0; //desired speed
time_t StartControlTime;
TCriticalSection *ThreadProtect;
unsigned int NumPulse = 2000;
double TSamp = 0.005;
double Kp, Ki, Kd;
```