

TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ



TÀI LIỆU THỰC HÀNH
ĐIỀU KHIỂN QUÁ TRÌNH

TS. Dương Đình Tú

Nghệ An, 3/2024

MỤC LỤC

| | |
|--|-----------|
| Bài 1. Mô hình hóa phương trình/hệ phương trình vi phân, sai phân trên MATLAB/SIMULINK..... | 5 |
| 1.1. Mục tiêu..... | 5 |
| 2.2. Nội dung thực hành..... | 5 |
| 2.3. Các bước thực hành..... | 5 |
| 1.3.1 Mô hình ODEs đơn giản | 6 |
| 1.3.2 Điều kiện kết thúc mô phỏng..... | 8 |
| 1.3.3 Mô hình hóa ODEs bậc nhất..... | 9 |
| 1.3.4 Xác thực giải pháp số và cài đặt tham số điều khiển | 11 |
| 1.3.5 Phương pháp mô hình vector hóa..... | 13 |
| 1.3.6 Tạo khung chung cho ODEs bậc nhất | 14 |
| 1.3.7 Khung mô hình cho ODEs bậc nhất biến thiên theo thời gian | 16 |
| 1.3.8 Mô hình hóa ODEs tuyến tính bậc cao | 17 |
| 1.3.9 Mô hình hàm truyền cho ODEs tuyến tính không đổi | 18 |
| 1.3.10 Mô hình hóa ODEs bậc cao không tuyến tính | 20 |
| 1.3.11 Mô hình hóa ODEs ẩn | 20 |
| 1.3.12 Mô hình hóa ODEs không liên tục | 22 |
| 1.3.13 Mô hình hóa hệ phương trình ODEs bậc cao | 23 |
| 1.3.14 Mô hình hóa ODEs có trễ..... | 24 |
| 1.3.15 Mô hình hóa ODEs có trễ loại trung tính..... | 26 |
| 1.3.16 Mô hình hóa ODEs chuyển đổi | 26 |
| 1.3.17 Mô hình hóa ODEs với đầu vào ngẫu nhiên | 27 |
| 1.3.18 Mô hình hóa phương trình sai phân | 28 |
| 1.4. Câu hỏi thực hành..... | 31 |
| TÀI LIỆU THAM KHẢO | 31 |
| Bài 2. Mô hình hóa hệ thống điều khiển trên MATLAB/SIMULINK..... | 32 |
| 2.1. Mục tiêu..... | 32 |
| 2.2. Nội dung thực hành..... | 32 |
| 2.3. Các bước thực hành..... | 32 |
| 2.3.1 Mô hình hóa hệ thống tuyến tính liên tục | 32 |
| 2.3.2 Mô hình hóa hệ thống tuyến tính rời rạc | 38 |
| 2.3.3 Mô hình hóa hệ thống phi tuyến | 42 |

| | |
|---|----|
| 2.3.4 Tự động sắp xếp các khối trong SIMULINK | 45 |
| 2.3.5 Phân tích gần đúng hệ thống điều khiển phi tuyến | 46 |
| 2.4. Câu hỏi thực hành..... | 52 |
| TÀI LIỆU THAM KHẢO | 53 |
| Bài 3. Mô hình hoá lý thuyết | 54 |
| 3.1. Mục tiêu..... | 54 |
| 3.2. Nội dung thực hành..... | 54 |
| 3.3. Các bước thực hành | 54 |
| 3.3.1 Mô hình hoá hệ thống trộn | 54 |
| 3.3.2 Tuyến tính hoá mô hình phi tuyến | 60 |
| 3.3.3 Mô hình quá trình khuấy trộn chất lỏng đẳng nhiệt | 66 |
| 3.3.4 Mô hình quá trình khuấy trộn chất lỏng không đẳng nhiệt..... | 70 |
| 3.3.5 Mô hình quá trình gia nhiệt có khuấy trộn | 71 |
| 3.3.6 Mô hình quá trình của bồn phản ứng khuấy trộn liên tục..... | 72 |
| 3.3.7 Mô hình quá trình mức chất lỏng | 73 |
| 3.3.8 Mô hình quá trình trao đổi nhiệt chất lỏng | 74 |
| 3.3.9 Mô hình quá trình của bộ trao đổi nhiệt bằng hơi | 75 |
| 3.3.10 Mô hình quá trình của bộ trao đổi nhiệt có trễ | 76 |
| 3.3.11 Mô hình quá trình phản ứng 2 pha | 76 |
| 3.3.12 Mô hình quá trình khí nén | 77 |
| 3.3.13 Mô hình quá trình thuỷ lực | 77 |
| 3.4. Câu hỏi thực hành..... | 78 |
| TÀI LIỆU THAM KHẢO | 78 |
| Bài 4. Thiết kế và chỉnh định tham số bộ điều khiển PID | 79 |
| 4.1. Mục tiêu..... | 79 |
| 4.2. Nội dung thực hành..... | 79 |
| 4.3. Các bước thực hành | 79 |
| 4.3.1 Thiết kế bộ điều khiển ON-OFF | 79 |
| 4.3.2 Thiết kế bộ điều khiển PID | 80 |
| 4.3.3 Chỉnh định bộ điều khiển PID | 90 |
| 4.4. Câu hỏi thực hành..... | 98 |
| TÀI LIỆU THAM KHẢO | 98 |

| | |
|---|-----|
| Bài 5. Điều khiển quá trình sản xuất xi măng | 99 |
| 5.1. Mục tiêu..... | 99 |
| 5.2. Nội dung thực hành..... | 99 |
| 5.3. Các bước thực hành | 99 |
| 5.3.1 Điều khiển định lượng trên cân băng..... | 99 |
| 5.3.2 Điều khiển định lượng trên cân băng..... | 100 |
| 5.3.3 Điều khiển phối hợp nhiều thành phần..... | 100 |
| 5.3.4 Thiết kế bộ điều khiển PID bền vững cho quá trình nghiên | 100 |
| 5.3.5 Mô hình quá trình truyền nhiệt trong lò nung clinke | 101 |
| 5.3.6 Mô hình quá trình truyền nhiệt trong lò nung clinke | 101 |
| 5.4. Câu hỏi thực hành..... | 102 |
| TÀI LIỆU THAM KHẢO | 102 |

Bài 1. Mô hình hóa phương trình/hệ phương trình vi phân, sai phân trên MATLAB/SIMULINK

1.1. Mục tiêu

Sau khi hoàn thành bài thực hành này, sinh viên có khả năng sử dụng được các công cụ MATLAB/SIMULINK để mô hình hóa và mô phỏng các phương trình/hệ phương trình vi phân và phương trình/hệ phương trình sai phân từ đơn giản cho đến phức tạp.

2.2. Nội dung thực hành

Thực hiện mô hình hóa các loại phương trình/hệ phương trình vi phân và phương trình/hệ phương trình sai phân bằng MATLAB/SIMULINK.

2.3. Các bước thực hành

Phương trình/hệ phương trình vi phân và phương trình/hệ phương trình sai phân là nền tảng toán học của hệ động lực. Các phương pháp mô hình hóa phương trình/hệ phương trình vi phân và phương trình/hệ phương trình sai phân rất quan trọng trong việc mô phỏng hệ động lực học. Hầu hết các quá trình đều được mô tả toán học bằng các phương trình/hệ phương trình vi phân và phương trình/hệ phương trình sai phân. Để giải quyết bài toán mô hình quá trình, cần phải biết mô hình hóa và mô phỏng phương trình/hệ phương trình vi phân và phương trình/hệ phương trình sai phân.

a) Mô hình hóa và mô phỏng phương trình/hệ phương trình vi phân

Việc sử dụng SIMULINK để mô hình hóa và mô phỏng phương trình/hệ phương trình vi phân thông thường (ordinary differential equations/ODEs) cho phép chúng ta có cái nhìn trực quan hơn về mô hình. Các nguyên tắc chung khi sử dụng SIMULINK để mô hình hóa và mô phỏng ODEs bao gồm:

- Để xác định hai tín hiệu $x(t)$ và $x'(t)$, sử dụng khối  Integrator. Tín hiệu đầu ra có thể được coi là tín hiệu $x(t)$, sau đó tín hiệu đầu vào đương nhiên là tín hiệu $x'(t)$.
- Nếu hai tín hiệu bằng nhau thì chúng có thể được nối trực tiếp bằng đường tín hiệu. Trong mô hình mô phỏng, phương pháp này thường được sử dụng để đóng các vòng lặp mô phỏng.
- Nếu tín hiệu $u(t)$ đã biết, nó có thể được theo sau bởi khối  Gain (nhấp đúp vào khối để đặt giá trị của mức tăng thành k). Đầu ra của khối là $ku(t)$.

4. Ví dụ, nếu tính toán đại số phi tuyến trên tín hiệu $v(t)$ để tính $(v(t) + v^3(t)/3)\sin v(t)$, tín hiệu $v(t)$ có thể được nhập vào khối Fcn  . Nhập đúp vào khối, trong hộp thoại tham số, nhập chuỗi $(u+u^3/3)*\sin(u)$ để khai báo. Tín hiệu đầu ra của khối là hàm $(v(t) + v^3(t)/3)\sin v(t)$.

5. Nhiều tín hiệu có thể được đưa vào khối Mux  để có thể tạo thành tín hiệu

vectơ; nếu tín hiệu vectơ được đưa vào khối Demux , nó có thể bị phân tách thành tín hiệu vectơ vô hướng hoặc nhỏ hơn.

6. Khi tín hiệu $u(t)$ đã biết, nếu cần làm trễ tín hiệu đó thành $u(t - d)$, thì tín

hiệu $u(t)$ có thể được đưa vào khối Transport Delay  . Nhập đúp vào khối, tham số d được nhập trong hộp thoại tham số.

7. Để lấy thông tin của một tín hiệu nhất định, nó có thể được kết nối với khối

 Scope  hoặc liên kết với khối Out  \times_{out1}^1 để xử lý hoặc hiển thị tín hiệu.

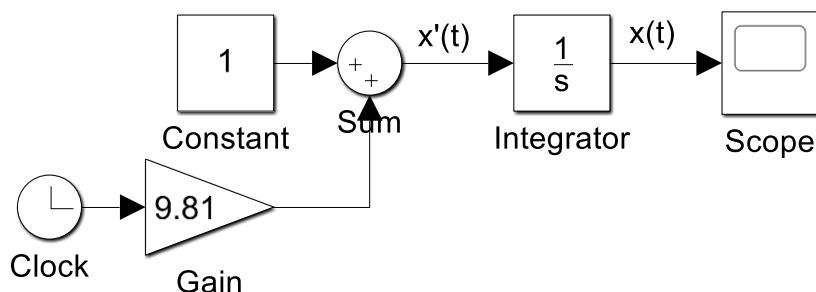
8. a) *Mô hình hóa và mô phỏng phương trình/hệ phương trình sai phân*

Tương tự như đối với mô hình hóa và mô phỏng phương trình/hệ phương trình vi phân, nhưng diễn ra trong miền rời rạc z .

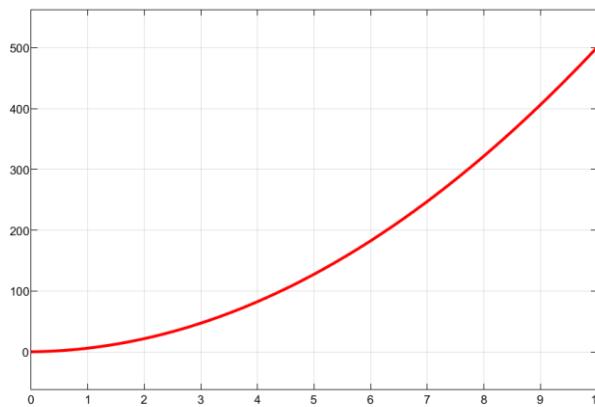
1.3.1 Mô hình ODEs đơn giản

Bài toán 1.1: Trong vật lý, phương trình của một vật rơi được mô tả như sau: $v(t) = v_0 + gt$, trong đó t là thời gian, $v(t)$ là vận tốc tức thời của vật, $v_0 = 1\text{m/s}$ là vận tốc ban đầu và $g = 9.81 \text{ m/s}^2$ là gia tốc trọng trường. Hãy mô phỏng các mối liên hệ giữa thời gian t và độ dịch chuyển $x(t)$ sử dụng SIMULINK.

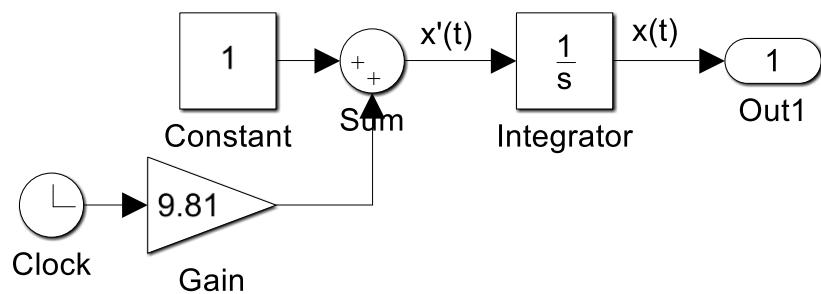
- B1: Trên SIMULINK, thiết lập mô hình như sau:



- B2: Chạy mô hình mô phỏng, nhận được kết quả như sau:



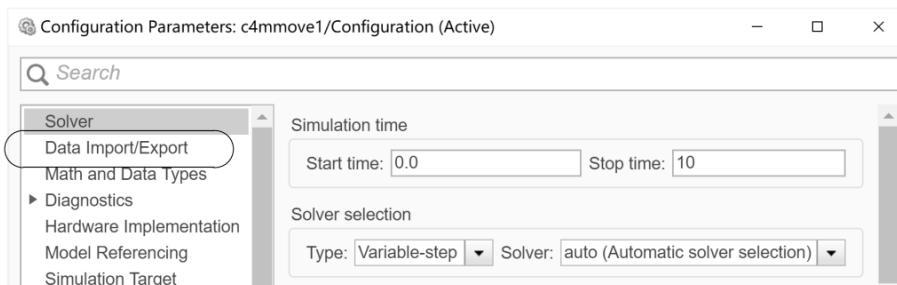
- Lưu ý: có thể sử dụng khối Out thay thế khối Scope như sau:

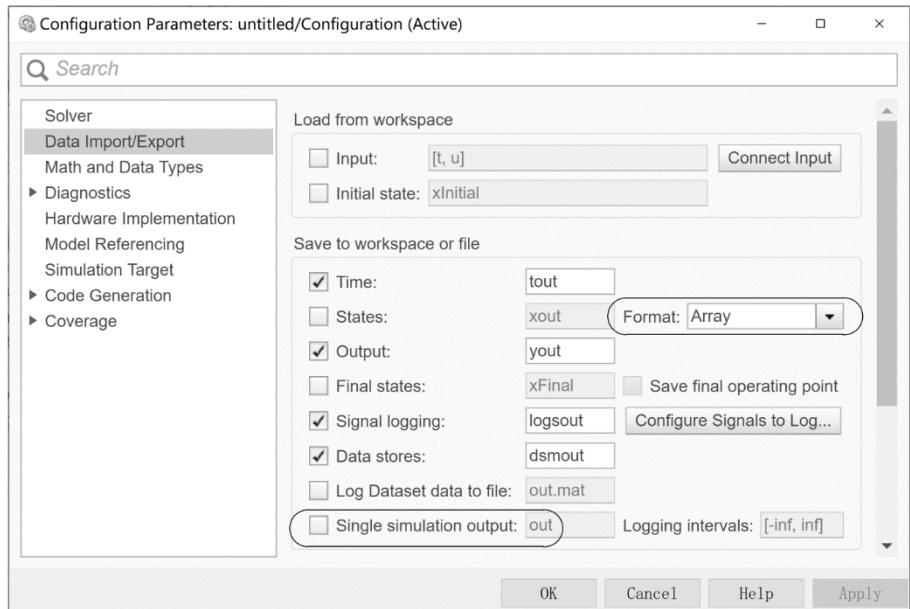


Sau đó nhập dòng lệnh sau trên của sổ Command Window của MATLAB, ta được kết quả tương tự:

```
>> plot(tout, yout{1}.Values.Data)
```

Để đơn giản hơn trong lệnh MATLAB, cần thay thế dữ liệu nhận được từ dạng Dataset thành dạng Array. Vào chức năng Model Configuration Parameters để thiết đặt lại như sau:





Lúc này trên MATLAB, chúng ta chỉ cần nhập dòng lệnh sau:

Command Window

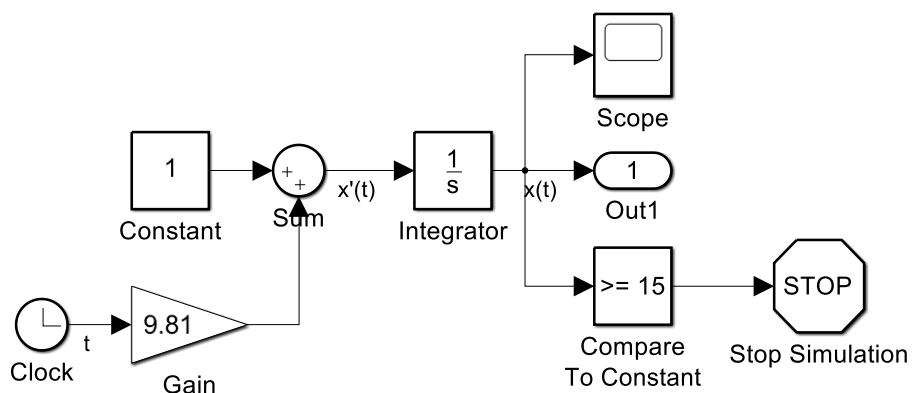
```
>> plot(tout, yout)
```

1.3.2 Điều kiện kết thúc mô phỏng

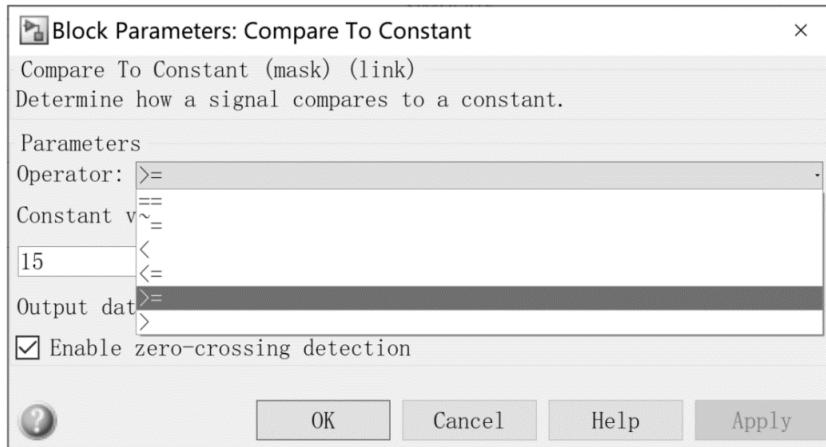
Thông thường, thời gian dừng mô phỏng có thể được xác định bằng cách điền vào vào một giá trị trong hộp chỉnh sửa Simulation stop time trên thanh công cụ. Giá trị mặc định là 10s. Tuy nhiên, nếu muốn quá trình mô phỏng tự động dừng ở điều kiện đặt trước, khói Stop Simulation trong nhóm Sinks được sử dụng.

Bài toán 1.2: Xét lại Bài toán 1, nếu điểm rơi cách mặt đất 15m thì vật rơi xuống đất trong bao lâu?

- B1: Thiết lập mô hình trên SIMULINK như sau:



Lưu ý, tại khói Compare To Constant, thiết lập như sau:



- B2: Chạy mô hình, sau đó thực hiện lệnh sau trên giao diện Command Window của MATLAB để nhận được thời gian kết thúc mô phỏng:

```
Command Window
>> tout(end)

ans =
1.6498
```

1.3.3 Mô hình hóa ODEs bậc nhất

Dạng toán học của phương trình vi phân bậc nhất được cho bởi:

$$x'(t) = f(t, x(t)),$$

trong đó, $x(t) = [x_1, x_2, \dots, x_n]^T$ được gọi là vectơ trạng thái, $f(\cdot) = [f_1(\cdot), f_2(\cdot), \dots, f_n(\cdot)]^T$ là hàm phi tuyến bất kỳ, t là thời gian. Ngoài ra, $x(t_0)$ là vectơ trạng thái ban đầu.

Bài toán 1.3: sử dụng SIMULINK để giải phương trình vi phân bậc nhất sau:

$$\begin{cases} y'(x) = -2xy(x) \ln z(x), \\ z'(x) = 2xz(x) \ln y(x), \end{cases}$$

với điều kiện ban đầu $y(0) = e$ và $z(0) = 1$. Biết rằng, nghiệm giải tích của phương trình vi phân là:

$$\begin{aligned} y(x) &= e^{\cos x^2} \\ z(x) &= e^{\sin x^2} \end{aligned}$$

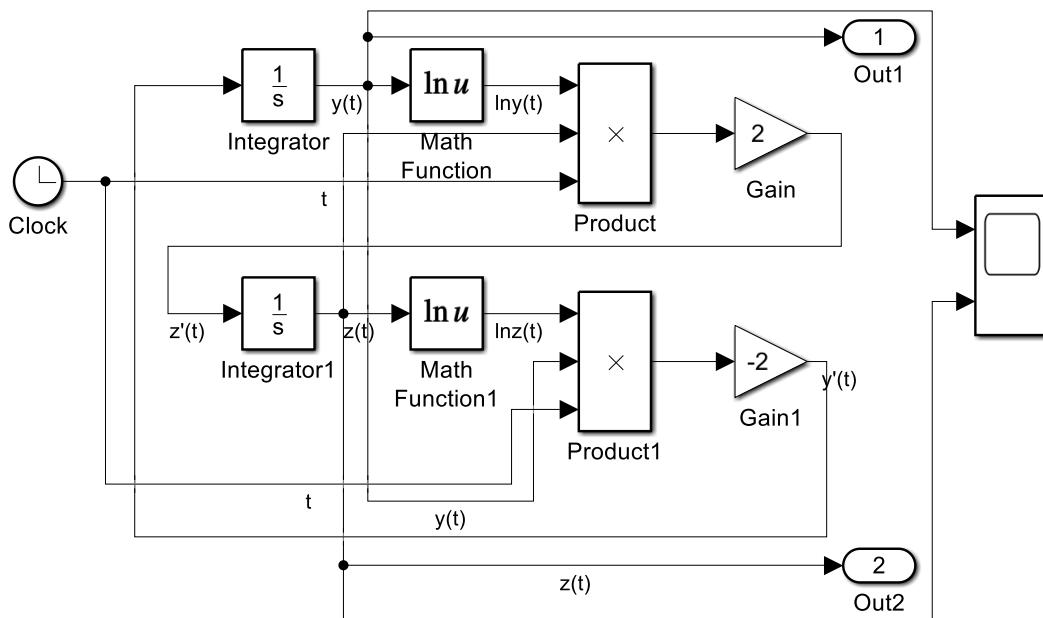
Tìm các nghiệm số.

- B1: Từ phương trình vi phân đã cho, có thể thấy biến độc lập là x , trong khi mô hình SIMULINK được điều khiển bởi thời gian, biến độc lập mặc định phải là t . Do đó, cần thay thế biến x thành t . Phương trình đã cho trở thành:

$$\begin{cases} y'(t) = -2ty(t) \ln z(t), \\ z'(t) = 2tz(t) \ln y(t), \end{cases}$$

$$\text{và } y(t) = e^{\cos t^2} \text{ và } z(t) = e^{\sin t^2}.$$

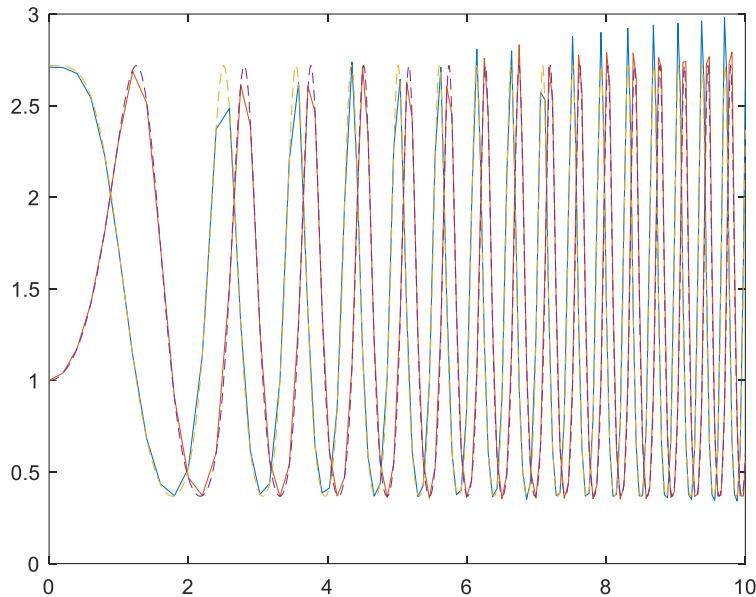
- B2: dựa vào các phương trình trên, thiết lập mô hình SIMULINK như sau:



Lưu ý rằng, với điều kiện ban đầu $y(0) = e$ và $z(0) = 1$, trên khối Integrator, đặt Initial condition là 2.71, còn trên khối Integrator1 đặt là 1.

- B3: nghiệm của phương trình vi phân được trả về biến tout và yout, sử dụng lệnh trên cửa sổ Command Window để xem kết quả. Vì có hai cổng đầu ra Out1 và Out2 trong mô hình nên yout là một ma trận có hai cột, biểu diễn các biến trạng thái tương ứng $y(t)$ và $z(t)$. Sơ đồ hàm plot () có thể được sử dụng để vẽ hai trạng thái như sau (lưu ý cần chuyển dữ liệu dạng Dataset thành dạng Array):

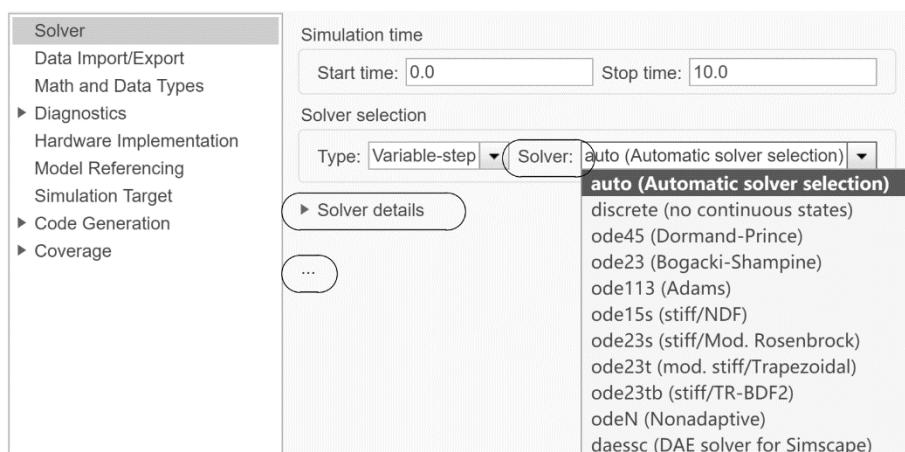
```
>> t0=0:0.001:10;
y1=exp(cos(t0.^2)); y2=exp(sin(t0.^2));
plot(tout,yout, t0,y1,'--',t0,y2,'--')
```

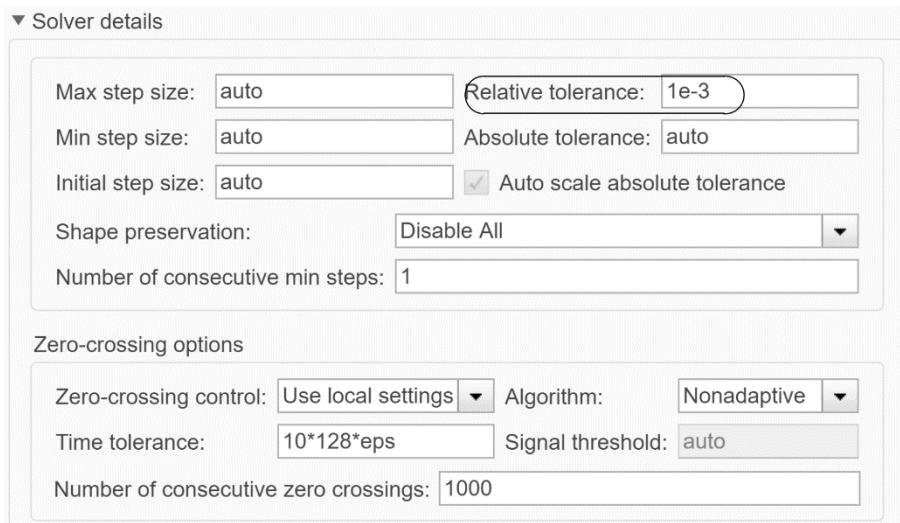


Hoặc có thể xem kết quả $y(t)$ và $z(t)$ dựa vào khối Scope của SIMULINK.

1.3.4 Xác thực giải pháp số và cài đặt tham số điều khiển

Mô hình hóa và mô phỏng SIMULINK cũng phải trải qua các quá trình xác nhận như trên dòng lệnh MATLAB. Tương tự như các hàm giải ODEs trên MATLAB như `ode45()`, các thuật toán bước thay đổi cũng được hỗ trợ trong SIMULINK. Giá trị mặc định của `relative tolerance` (khả năng chịu lỗi) tương đối được mặc định 10^{-3} . Giá trị mặc định này quá nhiều đối với các ứng dụng thông thường. Đối với hầu hết các phương trình vi phân, không thể tìm thấy kết quả mô phỏng chính xác. `absolute tolerance` (lỗi tuyệt đối) mặc định trong SIMULNIK được đặt thành 10^{-6} , cũng rất lớn. Do đó, trong các quá trình thực tế, các tham số này phải được gán giá trị nhỏ nhất có thể là 3×10^{-14} (giá trị thực nhỏ nhất được phép là $2,8422 \times 10^{-14}$). Trên SIMULINK, vào chức năng Model Configuration Parameters thực hiện như sau:





Bài toán 1.4: tìm nghiệm số chính xác của phương trình vi phân trong Bài toán 1.3. So sánh các kết quả và đánh giá độ chính xác của kết quả mô phỏng.

B1: Từ mô hình SIMULINK ở Bài toán 1.3, lần lượt thay đổi relative tolerance theo bảng sau:

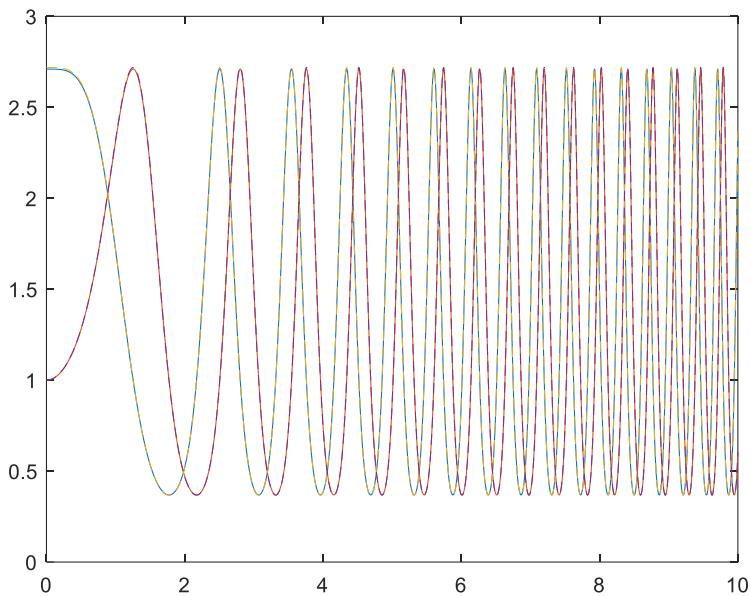
| error tolerance | 10^{-3} | 10^{-5} | 10^{-8} | 10^{-10} | 10^{-12} | 3×10^{-14} |
|-----------------|-----------|-----------|------------------------|------------------------|-------------------------|-------------------------|
| y_1 error | 0.3139 | 0.00304 | 1.193×10^{-6} | 1.184×10^{-8} | 1.236×10^{-10} | 3.921×10^{-12} |
| y_2 error | 0.3309 | 0.00314 | 1.204×10^{-6} | 1.191×10^{-8} | 1.241×10^{-10} | 3.961×10^{-12} |

- B2: thực nghiệm thấy rằng, khi relative tolerance càng giảm độ chính xác của giải pháp sẽ cao hơn. Các giải pháp. Nếu relative tolerance được đặt thành 3×10^{-14} thì sai số tối đa có thể đạt tới mức 10^{-12} . Các đường cong với relative tolerance này được thể hiện như sau. So sánh đồ thị này với đồ thị ở Bài toán 1.3, rút ra kết luận?

```
Command Window
>> t0=tout; y1=exp(cos(t0.^2)); y2=exp(sin(t0.^2));
plot(tout,yout, t0,y1, '--', t0,y2, '--')
e1=max(abs(yout(:,1)-y1)), e2=max(abs(yout(:,2)-y2))

e1 =
0.0083

e2 =
0.0083
```



1.3.5 Phương pháp mô hình vector hoá

Bài toán 1.5: giải Bài toán 1.3 bằng phương pháp điều khiển dòng lệnh (the command-line driven method) trên MATLAB. Sau đó, thiết lập mô hình SIMULINK bằng phương pháp vector hoá.

- B1: biến đổi thành mô hình vector: chọn các biến $x_1(t) = y(t)$, $x_2(t) = z(t)$, dạng chuẩn của mô hình không gian trạng thái được biểu diễn:

$$\begin{bmatrix} x'_1(t) \\ x'_2(t) \end{bmatrix} = \begin{bmatrix} -2tx_1(t) \ln x_2(t) \\ 2tx_2(t) \ln x_1(t) \end{bmatrix}$$

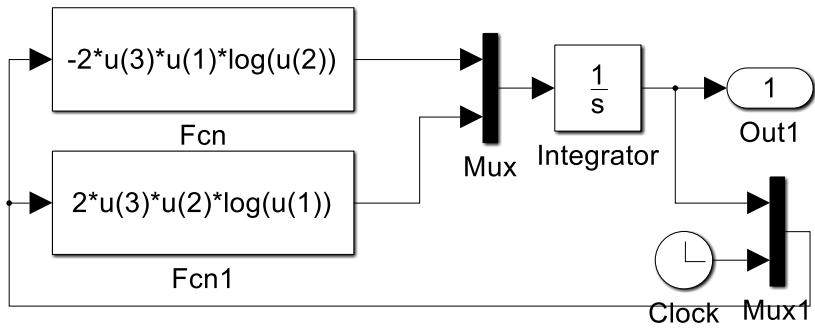
- B2: Dựa ODE ở trên, một hàm ẩn danh có thể được viết để mô tả nó, sau đó có thể tìm thấy nghiệm số một cách trực tiếp bằng đoạn code MATLAB sau đây (sử dụng hàm `ode45`):

```
>> f=@(t,x)[-2*t*x(1)*log(x(2)); 2*t*x(2)*log(x(1))];  
x0=[exp(1); 1]; [t,x]=ode45(f,[0,10],x0);
```

- B3: Xây dựng lại phương trình trong Bài toán 1.3 theo phương pháp vector hoá như sau:

$$\mathbf{y}(t) = \begin{bmatrix} -2x_3(t)x_1(t) \ln x_2(t) \\ 2x_3(t)x_2(t) \ln x_1(t) \end{bmatrix}$$

- B4: Trên SIMULINK, thiết lập mô hình như sau:



Lưu ý: xuất hiện lỗi ở khối Integrator do trạng thái khởi tạo của khối này là 0, không phù hợp. Nhập đúng vào khối này, đặt lại Initial condition là [2.71; 1].

- B5: thực hiện đoạn lệnh sau trên giao diện Command Window của MATLAB, kết quả như các giải pháp trước:

```
Command Window
>> t0=0:0.001:10;
y1=exp(cos(t0.^2)); y2=exp(sin(t0.^2));
plot(tout,yout, t0,y1, '--', t0,y2, '--')
```

1.3.6 Tạo khung chung cho ODEs bậc nhất

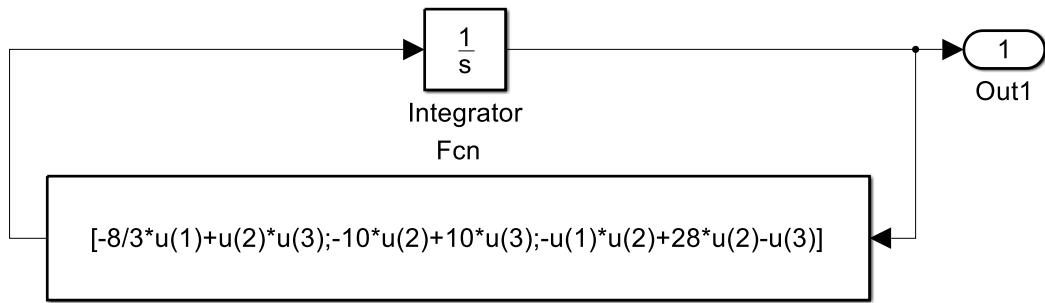
Trong các bài toán trên, khối Fcn đã được sử dụng để xây dựng các biểu thức có kích thước phù hợp. Có một điểm yếu nghiêm trọng của khối này, là chỉ có thể thu được tín hiệu vô hướng từ các cổng đầu ra của nó chứ không phải tín hiệu vectơ. Vì vậy, việc mô hình hóa trong phương pháp này khá phức tạp. Để giải quyết vấn đề này, các khối MATLAB Function hoặc MATLAB fcn có thể được sử dụng để xây dựng các khối có tín hiệu đầu vào vectơ và đầu ra vectơ. Sự khác biệt giữa hai khối là: MATLAB fcn tệp *.m được sử dụng để mô tả mối quan hệ trong các hàm, trong khi ở MATLAB Function, hàm được nhúng trong khối.

Bài toán 1.6: Thiết lập mô hình SIMULINK cho phương trình vi phân Lorenz sau:

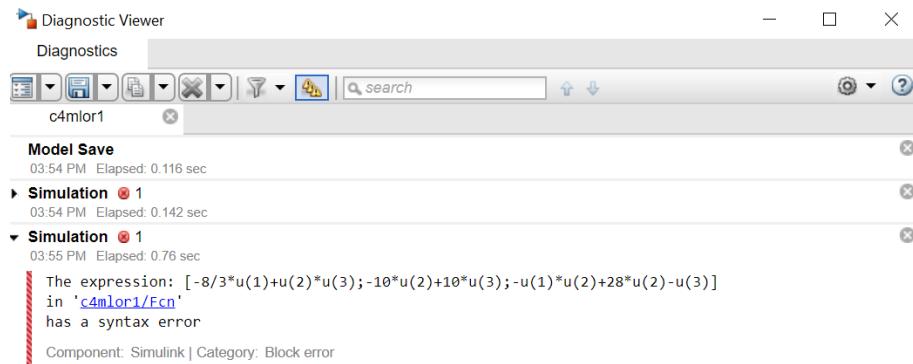
$$\begin{cases} x'_1(t) = -\beta x_1(t) + x_2(t)x_3(t), \\ x'_2(t) = -\rho x_2(t) + \rho x_3(t), \\ x'_3(t) = -x_1(t)x_2(t) + \sigma x_2(t) - x_3(t), \end{cases}$$

trong đó $\beta = 8/3$, $\rho = 10$, $\sigma = 28$ và các giá trị ban đầu là $x_1(0) = x_2(0) = 0$, $x_3(0) = \epsilon$, với ϵ giá trị nhỏ (ví dụ, $\epsilon = 10^{-10}$).

- B1: sử dụng khối Fcn để mô hình hoá:

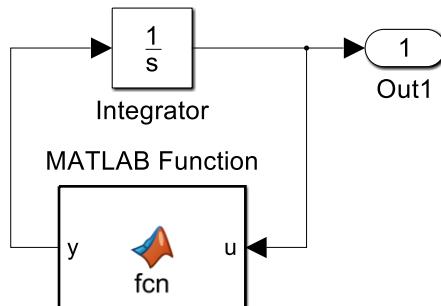


Tuy nhiên, khi thực hiện chương trình sẽ gặp lỗi sau:



Nguyên nhân là do đầu ra khối Fcn là tín hiệu vô hướng.

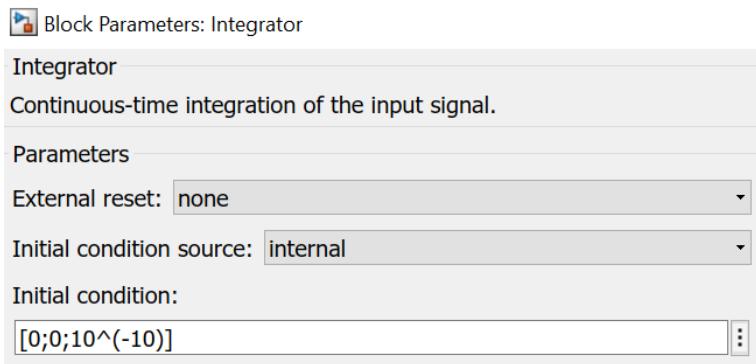
- B2: trên SIMULINK thiết lập mô hình như sau:



Trong khối MATLAB Function, thực hiện hàm như sau:

```
Command Window
MATLAB Function* + 
1 function y = fcn(u)
2
3 - y=[-8/3*u(1)+u(2)*u(3); -10*u(2)+10*u(3); ...
4 - u(1)*u(2)+28*u(2)-u(3)];
```

Lưu ý, từ điều kiện ban đầu của bài toán, cần thiết lập trong khối Integrator như sau:

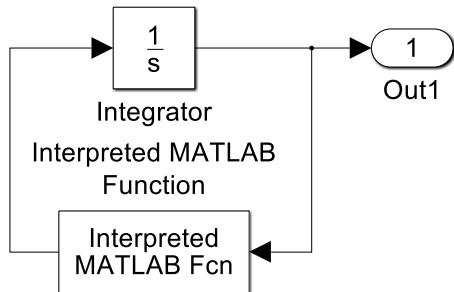


- Bước 3: kết quả được đưa ra bởi giá trị yout. Thực hiện lệnh sau trên giao diện Command Window của MATLAB để xem kết quả:

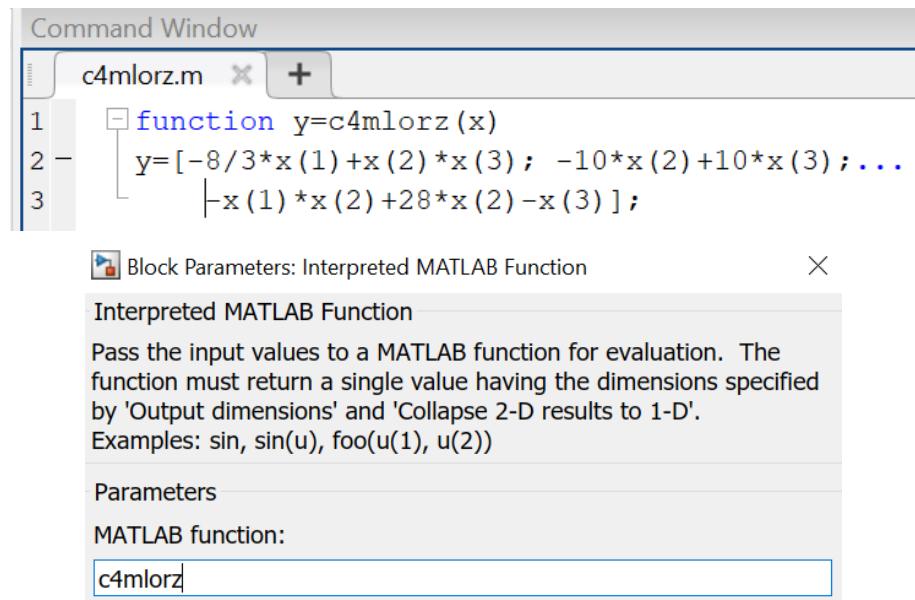
Command Window

```
>> plot3(yout(:,1),yout(:,2),yout(:,3)), grid
```

- Bước 4: sử dụng khối Interpreted MATLAB Function để thực hiện, kết quả hoàn toàn tương tự.



Lưu ý: cần tạo hàm `c4mlorz.m`, sau đó khai báo vào khối Interpreted MATLAB Function:



1.3.7 Khung mô hình cho ODEs bậc nhất biến thiên theo thời gian

Trong mô hình Simulink của các phương trình vi phân thay đổi theo thời gian, do vế phải chúa thời gian t , nhưng các khối MATLAB Fcn và MATLAB Function

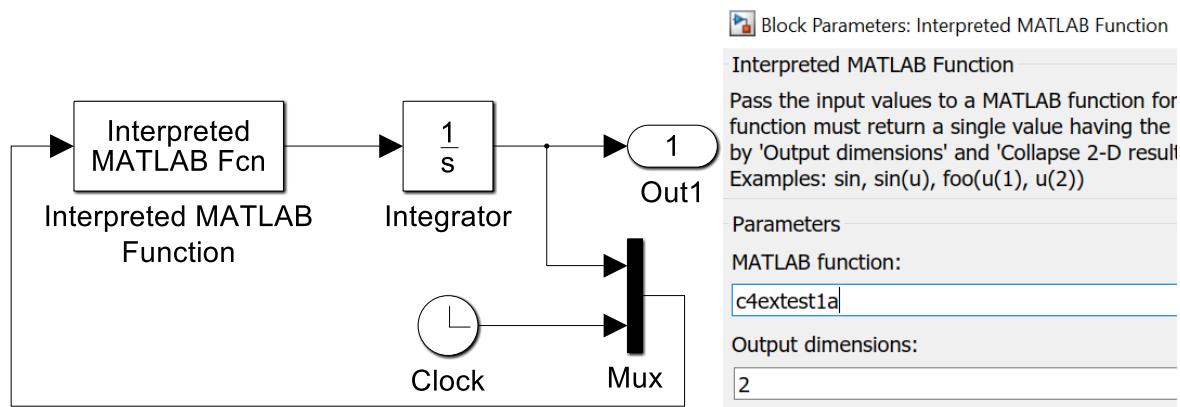
chỉ chấp nhận một tín hiệu đầu vào, nên cần có khối Mux để tăng t thành x_{n+1} , sao cho trạng thái vectơ tín hiệu được tăng cường làm đầu vào của khối. Do đó, cần có một khung mô hình hóa SIMULINK tổng quát hơn để mô tả các phương trình vi phân bậc nhất biến thiên theo thời gian.

Bài toán 1.7: thực hiện lại Bài toán 1.3, sử dụng các khối hàm.

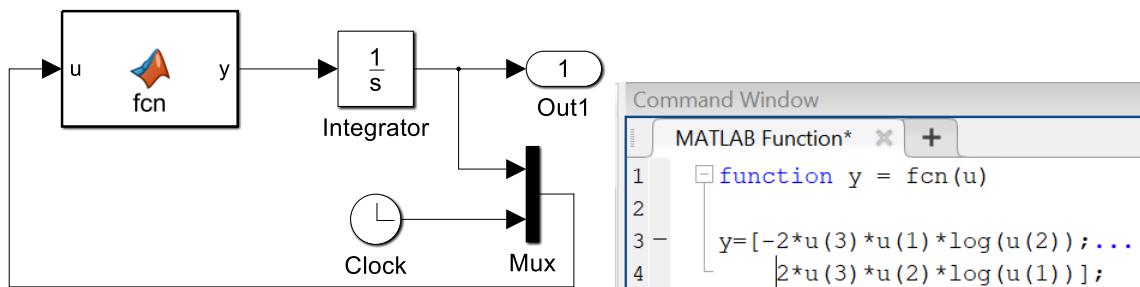
- B1: xây dựng hàm sau trên MATLAB:

```
Command Window
c4extest1a.m
function y=c4extest1a(x)
y=[-2*x(3)*x(1)*log(x(2)); 2*x(3)*x(2)*log(x(1))];
```

- B2: Thiết lập mô hình SIMULINK và thực hiện như sau:



- B3: sử dụng khối MATLAB Function và tạo hàm, thực hiện như sau:



Kết quả thực hiện là giống nhau.

1.3.8 Mô hình hóa ODEs tuyến tính bậc cao

Phương trình vi phân tuyến tính với các hệ số không đổi có dạng chung:

$$\begin{aligned} y^{(n)}(t) + a_1 y^{(n-1)}(t) + \cdots + a_{n-1} y'(t) + a_n y(t) \\ = b_1 u^{(m)}(t) + b_2 u^{(m-1)}(t) + \cdots + b_m u'(t) + b_{m+1} u(t), \end{aligned}$$

Bài toán 1.8: giải phương trình vi phân tuyến tính bất biến theo thời gian như sau:

$$y^{(4)}(t) + 5y'''(t) + 9y''(t) + 7y'(t) + 2y(t) = u'(t) + 2u(t),$$

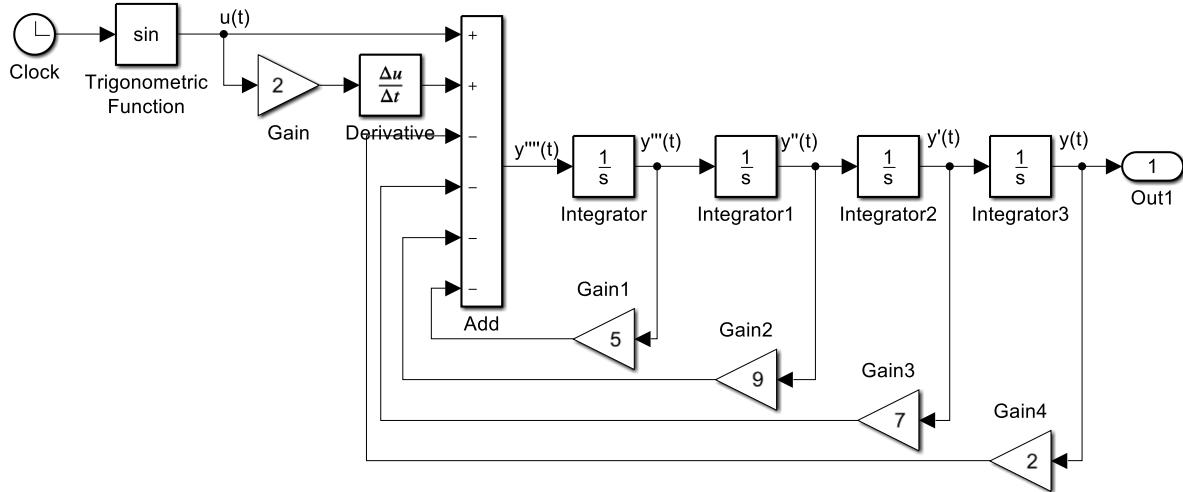
với $u(t) = \sin t$, $y(0) = 1$, $y'(0) = y''(0) = y'''(0) = 0$.

- B1: trên dòng lệnh MATLAB, có thể giải quyết như sau:

Command Window

```
>> syms t y(t); u=sin(t);
y1=diff(y); y2=diff(y1); y3=diff(y2);
y0=dsolve(diff(y3)+5*y3+9*y2+7*y1+2*y==2*diff(u),...
y(0)==1, y1(0)==0, y2(0)==0, y3(0)==0)
simplify(y0)
```

- B2: trên SIMULINK, thiết lập mô hình như sau, lưu lại tên file là c4mlin1:

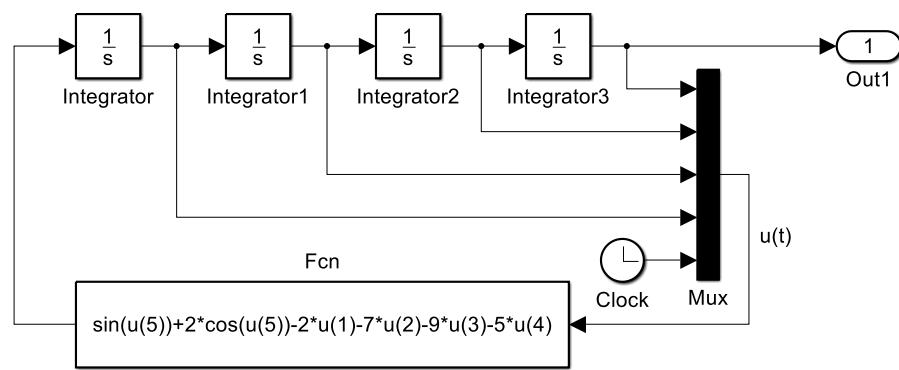


- B3: thực hiện đoạn lệnh sau trên giao diện Command Window của MATLAB.
Nhận xét về các đường cong?

```
>> tic, [t1,~,y1]=sim('c4mlin1'); toc, length(t1)
y0=7*exp(-t1)/4-2*exp(-2*t1)/5+t1.*exp(-t1)+...
3*t1.^2.*exp(-t1)/4-sqrt(2)*cos(t1-atan(1/7))/4;
max(abs(y0-y1)), plot(t1,y1,'--',t1,y0)
```

Bài toán 1.9: sử dụng phương pháp vector hoá ở mục 1.3.5 để giải Bài toán 1.8.

Thiết lập mô hình SIMULINK như sau:

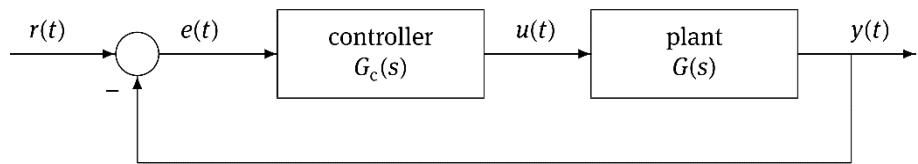


1.3.9 Mô hình hàm truyền cho ODEs tuyến tính không đổi

Mô hình hàm truyền tổng quát của các ODEs tuyến tính không đổi có dạng:

$$G(s) = \frac{b_1 s^m + b_2 s^{m-1} + b_3 s^{m-2} + \dots + b_m s + b_{m+1}}{s^n + a_1 s^{n-1} + a_2 s^{n-2} + \dots + a_{n-1} s + a_n}$$

Bài toán 1.10: Một hệ thống điều khiển phản hồi điển hình được biểu thị như sau:



trong đó quá trình $G(s)$ và bộ điều khiển $G_c(s)$ được đưa ra dưới dạng hàm truyền:

$$G(s) = \frac{e^{-s}}{(s+1)(s+2)} = \frac{e^{-s}}{s^2 + 3s + 2}, \quad G_c(s) = 1.66 + \frac{0.91}{s}$$

Hãy tìm đáp ứng của hệ thống với điều kiện ban đầu bằng 0.

Với các tính chất của phép biến đổi Laplace, mô hình hàm truyền có thể được khôi phục về mô hình phương trình vi phân. Vì mô hình $G(s)$ có trễ nên có thể thu được phương trình vi phân có trễ tương ứng. Quan sát mối quan hệ giữa các tín hiệu $u(t)$ và $y(t)$, có thể thu được phương trình vi phân có trễ sau:

$$y''(t) + 3y'(t) + 2y(t) = u(t - 1).$$

Xem xét và biến đổi $u(t)$ với $\delta(t)$ là hàm Dirac.

$$\begin{aligned} u'(t) &= 1.66(r(t) - y(t))' + 0.91(r(t) - y(t)) \\ &= 1.66\delta(t) - 1.66y'(t) + 0.91 - 0.91y(t), \end{aligned}$$

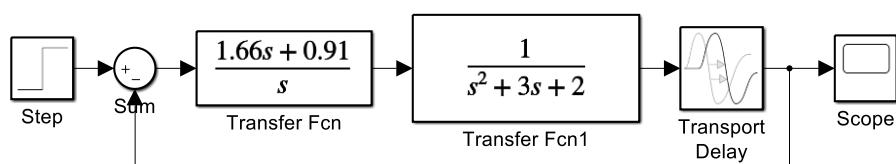
Chọn các biến trạng thái là $x_1(t) = y(t)$, $x_2(t) = y'(t)$ và $x_3(t) = u(t)$, có thể tìm thấy dạng chuẩn của phương trình vi phân:

$$\mathbf{x}'(t) = \begin{bmatrix} x_2(t) \\ x_3(t - 1) - 3x_2(t) - 2x_1(t) \\ 1.66\delta(t) - 1.66x_2(t) + 0.91 - 0.91x_1(t) \end{bmatrix}$$

- B1: Thực hiện lệnh trên giao diện Command Window của MATLAB như sau để giải bài toán:

```
Command Window
>> f=@(t,x,Z) [x(2); Z(3)-3*x(2)-2*x(1);
1.66*(1e4-1e4*heaviside(t-1e-4))-1.66*x(2)+0.91-0.91*x(1)];
ff=ddeset; ee=1e-10; ff.RelTol=ee; ff.AbsTol=ee;
tau=1; x0=zeros(3,1); % delay constants and initial values
tic, tx=dde23(f,tau,x0,[0,20],ff); toc % solve the equation
plot(tx.x,tx.y(1,:)) % draw the output curves
```

- B2: thiết lập mô hình SIMULINK như sau, sau đó so sánh với kết quả ở bước 1:



Lưu ý: chọn thời gian thực hiện là 20s. Hàm $G(s)$ có độ trễ là 1, nên tại khối Transport Delay đặt thông số Time delay là 1.

1.3.10 Mô hình hóa ODEs bậc cao không tuyến tính

Bài toán 1.11: cho phương trình ODE bậc cao không tuyến tính:

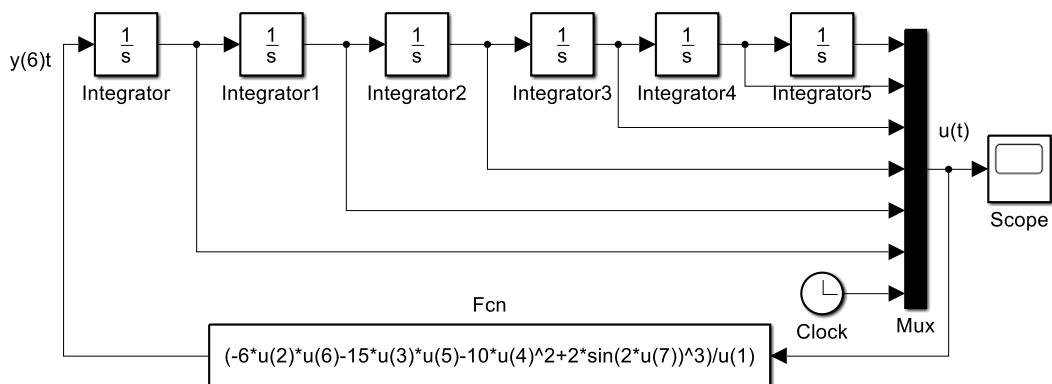
$$y(t)y^{(6)}(t) + 6y'(t)y^{(5)}(t) + 15y''(t)y^{(4)}(t) + 10(y'''(t))^2 = a \sin^m \lambda t.$$

Nếu $a=3$, $m=3$, $\lambda=2$, $y(0)=1$, $y^{(i)}(0)=0$, $i=1,2,\dots,5$, hãy thiết lập mô hình SIMULINK và vẽ đường cong các biến.

Có thể thấy rằng $y(t) = 0$ không phải là điểm kỳ dị của phương trình vi phân ban đầu. Vì phương trình giải tích của $y(t)$ chưa được biết nên về mặt lý thuyết không thể tìm ra điểm kỳ dị của các phương trình vi phân. Phương pháp số là cách duy nhất để nghiên cứu tính chất của phương trình vi phân. Số hạng đạo hàm bậc cao nhất trong phương trình ban đầu là $y^{(6)}(t)$, lúc đó:

$$y^{(6)}(t) = \frac{1}{y(t)} \left[-6y'(t)y^{(5)}(t) - 15y''(t)y^{(4)}(t) - 10(y'''(t))^2 + a \sin^m \lambda t \right],$$

Trên SIMULINK, thiết lập mô hình như sau:



Lưu ý các điều kiện ban đầu để thiết lập đúng thông số Initial condition trong các khối Integrator.

1.3.11 Mô hình hóa ODEs ẩn

Các phương trình vi phân được nghiên cứu từ Bài toán 1-11 đều là những phương trình rõ ràng. Nghĩa là, số hạng có đạo hàm bậc cao nhất được đưa ra dưới dạng một công thức rõ ràng. Nếu trong một phương trình vi phân nhất định, số hạng có bậc cao nhất không thể được viết dưới dạng một công thức rõ ràng thì phương trình vi phân được gọi là phương trình vi phân ẩn. Trong SIMULINK, sử dụng khối Algebraic Constraint để giải các ODEs ẩn.

Bài toán 1.12: giải phương trình vi phân ẩn sau đây:

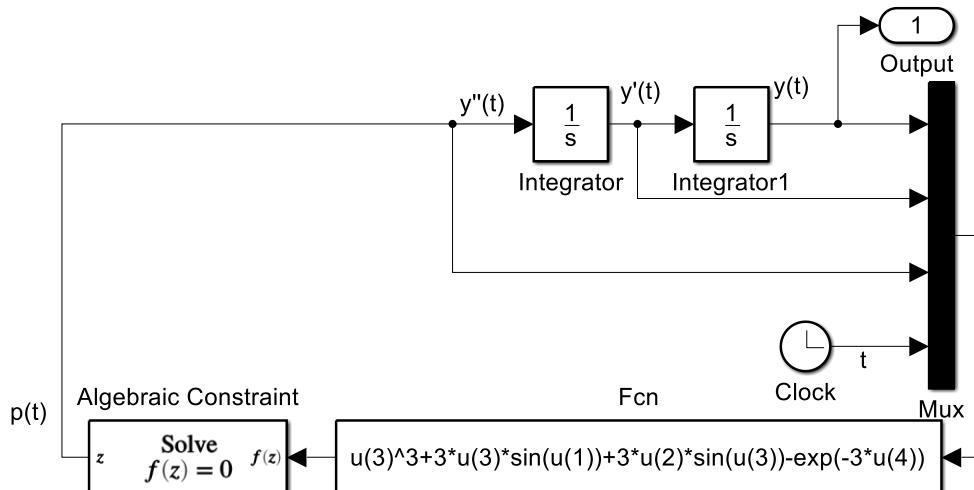
$$(y''(t))^3 + 3y''(t) \sin y(t) + 3y'(t) \sin y''(t) = e^{-3t}, \quad y(0) = 1, \quad y'(0) = -1.$$

Để kiểm tra độ chính xác của nghiệm, hãy so sánh nghiệm số thu được với nghiệm giải tích đã cho của $y(t) = e^{-t}$.

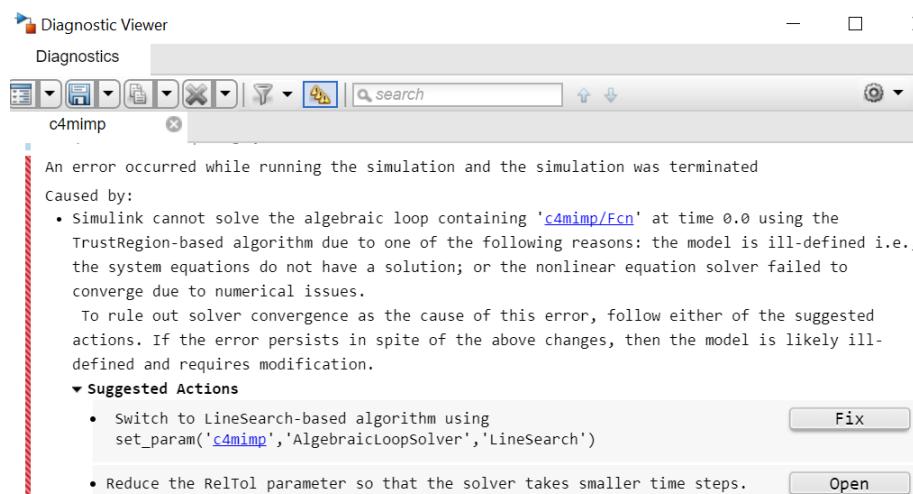
Để giải phương trình vi phân, cần có chuỗi hai bộ tích phân để xác định các tín hiệu chính $y(t)$, $y'(t)$ và $y''(t)$. Đặt $p(t) = y''(t)$, một phương trình đại số của $p(t)$ được thiết lập:

$$p^3(t) + 3p(t) \sin y(t) + 3y'(t) \sin p(t) - e^{-3t} = 0$$

- B1: trên SIMULINK, thiết lập mô hình như sau, lưu file là c4mimp:



Lưu ý các điều kiện ban đầu $y(0)=1$, $y'(0)=-1$ để thiết lập đúng thông số Initial condition trong các khối Integrator. Tuy nhiên khi thực hiện chương trình sẽ gặp lỗi như sau, điều này là do vòng lặp đại số và thuật toán vùng tin cậy mặc định không phù hợp với mô hình này. Thay vào đó, thuật toán tìm kiếm dòng được khuyến nghị. Trong hộp thoại báo lỗi, lệnh MATLAB được đề xuất sẽ được cung cấp để sửa đổi các tham số điều khiển. Nếu các tham số được sửa đổi, giải pháp có thể được tìm thấy trực tiếp, với sai số tối đa là $6.7856e^{-13}$.



- B2: Chúng ta có thể sửa lỗi bằng 2 cách: nhấp vào Fix trên thông báo, hoặc sử dụng dòng lệnh trên giao diện Command Window của MATLAB như sau:

```
Command Window
>> set_param('c4mimp','AlgebraicLoopSolver','LineSearch')
[t x y]=sim('c4mimp',5); err=max(abs(y-exp(-t)))
```

Lúc này thực hiện lại file SIMULINK một cách bình thường.

B3: Để so sánh nghiệm của phương trình trên với nghiệm của $y(t) = e^{-t}$, thực hiện lệnh sau trên giao diện Command Window của MATLAB:

```
Command Window
>> t0=0:0.001:10;
y1=exp(-t0);
plot(tout,yout, t0,y1, '--');
legend('ODE an', 'e^(-t)')
```

- B4: Lựa chọn các giá trị khác nhau cho các tham số điều chỉnh và quan sát ảnh hưởng của chúng đến kết quả mô phỏng.



Một số hạn chế:

- Khối Algebraic Constraint: chỉ có thể được sử dụng để xử lý các phương trình đại số vô hướng, không thể được sử dụng để giải các hệ phương trình vi phân ẩn.
- Do nhiều vòng lặp đại số có mặt trong mô hình mô phỏng nên ở mỗi bước mô phỏng, phương trình đại số được giải một lần. Vì vậy, tốc độ mô phỏng có thể khá chậm. Nên tránh các vòng lặp đại số nếu không cần thiết.

1.3.12 Mô hình hóa ODEs không liên tục

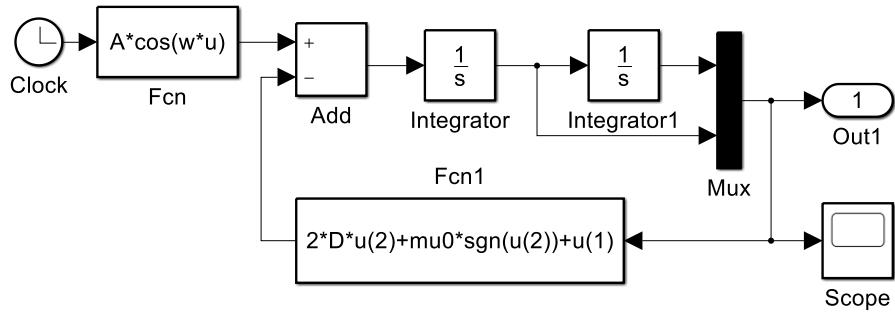
Các phương trình vi phân không liên tục là những phương trình có tín hiệu bị gián đoạn tại một số điểm nhất định hoặc xảy ra hiện tượng nhảy. Trong trường hợp thông thường, để tìm nghiệm số cho các phương trình vi phân loại này, không cần thực hiện hành động đặc biệt nào. Có thể thiết lập Relative tolerance cho các giá trị lớn hơn, chẳng hạn như 10^{-5} .

Bài toán 1.13: mô hình hóa ODE không liên tục sau đây:

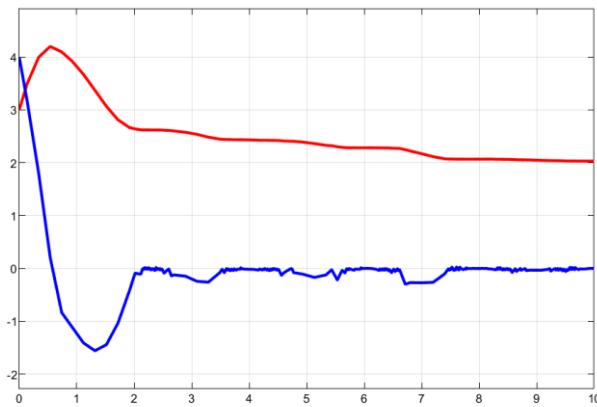
$$y''(t) + 2Dy'(t) + \mu \operatorname{sign}(y'(t)) + y(t) = A \cos \omega t,$$

với $D = 0.1$, $\mu = 4$, $A = 2$, $\omega = \pi$, các giá trị khởi tạo $y(0) = 3$, $y'(0) = 4$.

Mô hình hoá trên SIMULINK như sau:



Lưu ý: 1) có thể nhập trực tiếp các tham số A, D, μ_0, w trực tiếp trên khối Fcn hoặc từ giao diện Command Window của MATLAB; 2) chú ý các giá trị khởi tạo $y(0) = 3, y'(0) = 4$ để thiết lập đúng thông số Initial condition trong các khối Integrator. Kết quả:



Tuy nhiên, nếu giảm giá trị lỗi Relative tolerance, tốc độ tính toán sẽ chậm lại đáng kể. Nếu giảm xuống dưới giá trị 10^{-7} , kết quả có thể không còn chính xác.

1.3.13 Mô hình hóa hệ phương trình ODEs bậc cao

Bài toán 1.14: toạ độ vị trí (x, y) của phi thuyền Apollo được mô tả bằng hệ phương trình vi phân sau:

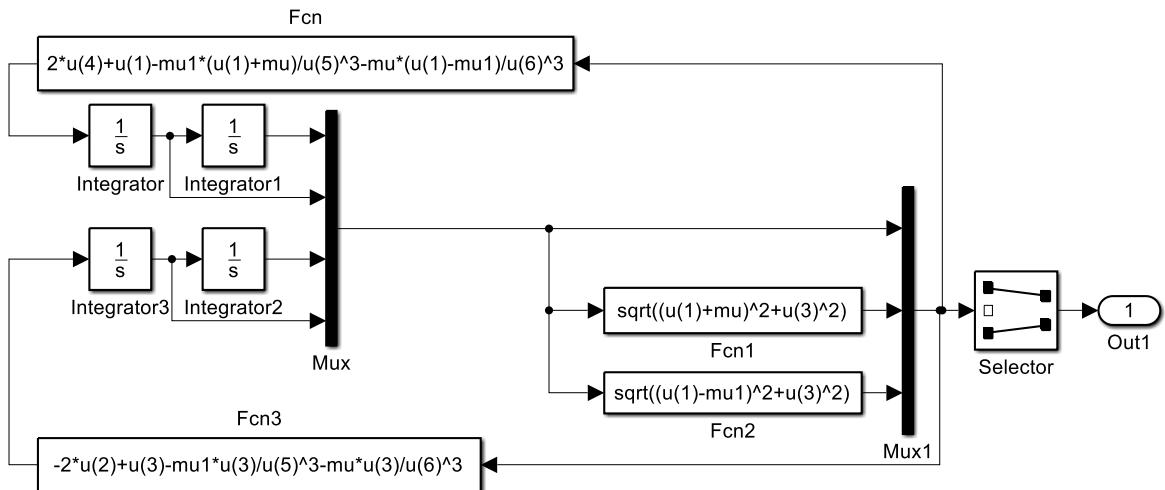
$$\begin{cases} x''(t) = 2y'(t) + x(t) - \mu^*(x(t) + \mu)/r_1^3(t) - \mu(x(t) - \mu^*)/r_2^3(t), \\ y''(t) = -2x'(t) + y(t) - \mu^*y(t)/r_1^3(t) - \mu y/r_2^3(t), \end{cases}$$

với $\mu = 1/82.45, \mu^* = 1 - \mu$ và:

$$r_1(t) = \sqrt{(x(t) + \mu)^2 + y^2(t)}, \quad r_2(t) = \sqrt{(x(t) - \mu^*)^2 + y^2(t)}.$$

Các giá trị khởi tạo: $x(0)=1.2, x'(0)=0, y(0)=0, y'(0)=-1.04935751$. Giải hệ phương trình trên và vẽ quỹ đạo (x, y) của phi thuyền Apollo.

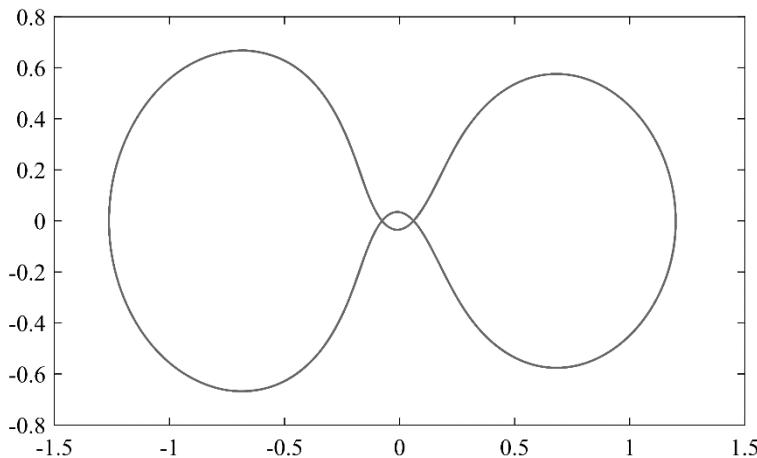
- B1: thiết lập mô hình trên SIMULINK như sau:



Lưu ý: 1) có thể nhập trực tiếp các tham số $\mu = 1/82.45$, $\mu_1 = 1 - \mu$ trực tiếp trên khối Fcn hoặc từ giao diện Command Window của MATLAB; 2) chú ý các giá trị khởi tạo $x(0)=1.2$, $x'(0)=0$, $y(0)=0$, $y'(0)=-1.04935751$ để thiết lập đúng thông số Initial condition trong các khối Integrator.

- B2: trên giao diện Commad Window của MATLAB, thực hiện lệnh sau để vẽ quỹ đạo (x,y) của phi thuyền Apollo:

```
>> plot(yout(:,1),yout(:,2))
```



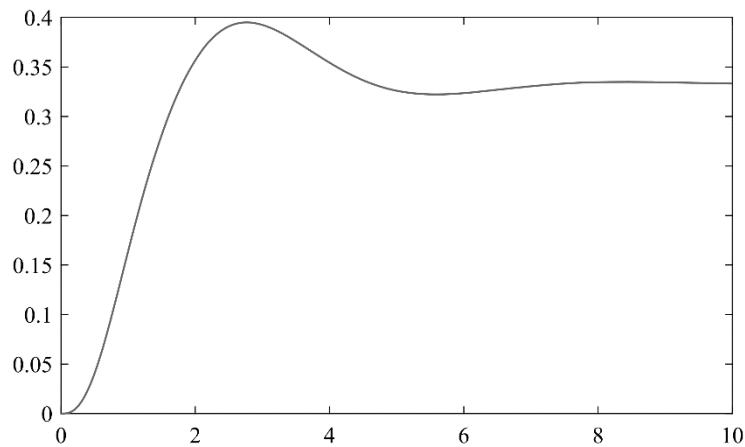
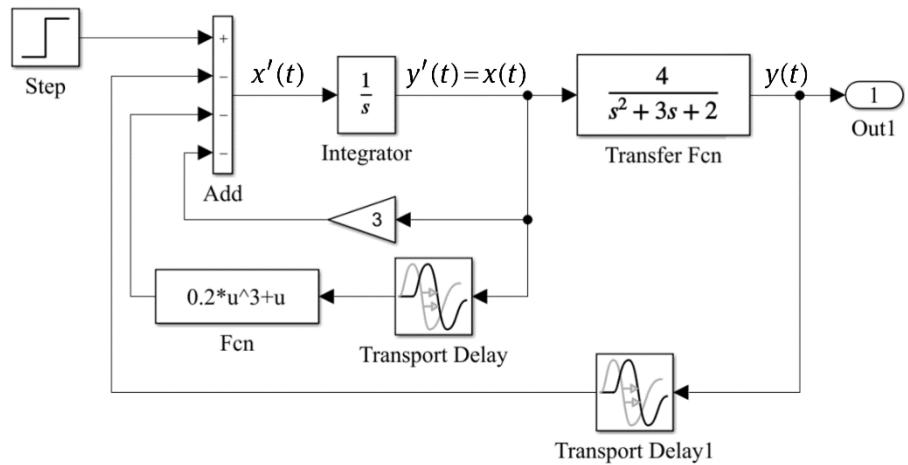
1.3.14 Mô hình hóa ODEs có trễ

Bài toán 1.15: giải hệ phương trình vi phân có trễ sau:

$$\begin{cases} x'(t) = 1 - 3x(t) - y(t-1) - 0.2x^3(t-0.5) - x(t-0.5), \\ y''(t) + 3y'(t) + 2y(t) = 4x(t), \end{cases}$$

với khi $t \leq 0$, thì $x(t) = y(t) = y'(t) = 0$.

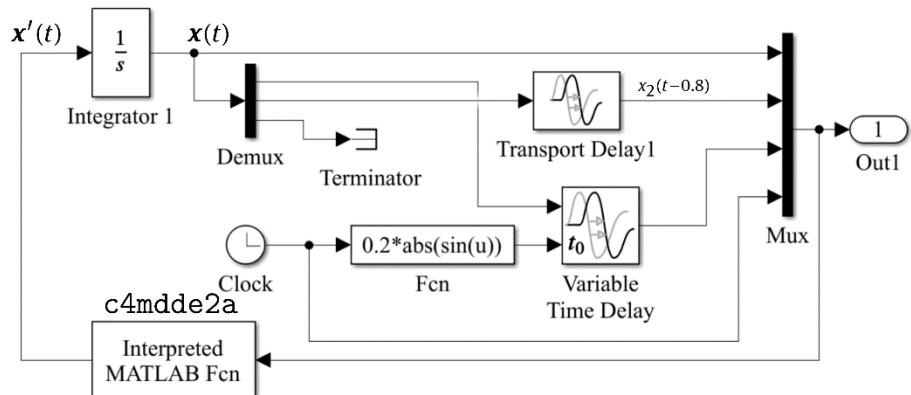
Mô hình hoá và kết quả trên SILULINK như sau:



Bài toán 1.15: giải hệ phương trình vi phân có trễ sau, giả sử rằng các hàm lich sử của các biến trạng thái đều bằng 0.

$$\begin{cases} x'_1(t) = -2x_2(t) - 3x_1(t - 0.2|\sin t|), \\ x'_2(t) = -0.05x_1(t)x_3(t) - 2x_2(t - 0.8) + 2, \\ x'_3(t) = 0.3x_1(t)x_2(t)x_3(t) + \cos(x_1(t)x_2(t)) + 2\sin 0.1t^2. \end{cases}$$

Mô hình hoá trên SILULINK như sau (c4mdde2a):



Lưu ý, hàm trong khối Interpreted MATLAB Function được viết như sau:

```

function dx=c4mdde2a(x)
dx=[-2*x(2)-3*x(5);
-0.05*x(1)*x(3)-2*x(4)+2;
0.3*x(1)*x(2)*x(3)+cos(x(1)*x(2))+2*sin(0.1*x(6)^2)];

```

1.3.15 Mô hình hóa ODEs có trễ loại trung tính

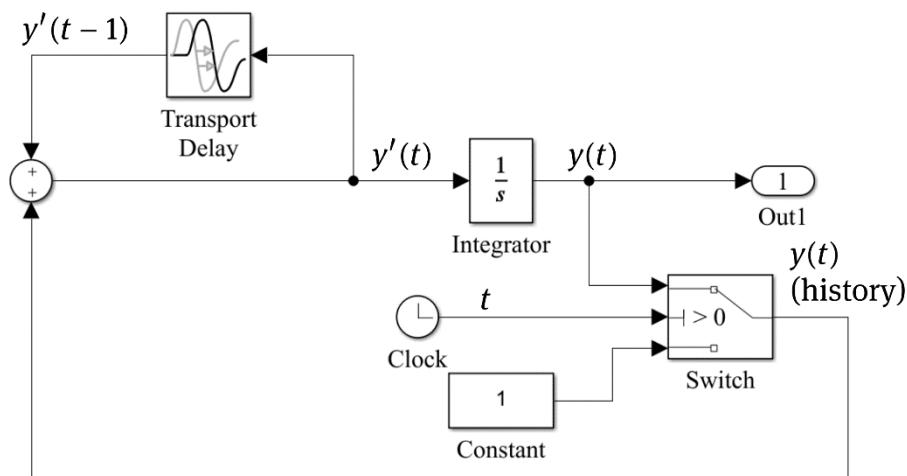
Bài toán 1.17: giải hệ phương trình vi phân có trễ sau:

$$y'(t) = y(t) + y'(t-1),$$

với khi $t \leq 0$, $y(t) = 1$ and $t \in (0, 3)$ và:

$$y(t) = \begin{cases} e^t, & 0 \leq t \leq 1, \\ e^t + (t-1)e^{t-1}, & 1 < t < 2, \\ e^t + e^{t-1} + (t-2)(t+2e)e^{t-2}/2, & 2 \leq t \leq 3. \end{cases}$$

- B1: trên SIMULINK, thiết lập mô hình như sau:



- B2: thực hiện dòng lệnh sau trên MATLAB để so sánh sai số lý thuyết:

```

>> t=tout;
z=exp(t)+(t-1).*exp(t-1).*(t>=1 & t<2)+...
(exp(t-1)+(t-2).*(t+2*exp(1)).*exp(t-2)/2).*...
(max(abs(z-yout)))

```

1.3.16 Mô hình hóa ODEs chuyển đổi

Bài toán 1.18: giả sử rằng hệ thống được đưa ra bởi phương trình:

$$\mathbf{x}'(t) = \mathbf{A}_i \mathbf{x}(t)$$

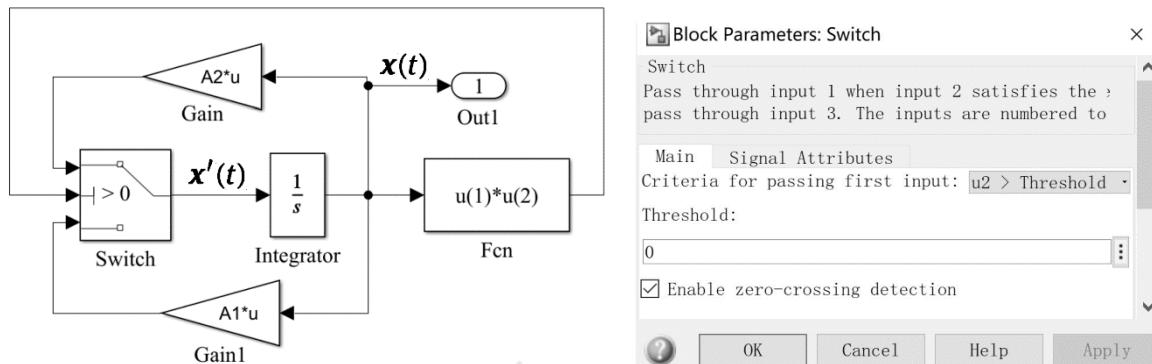
với $\mathbf{A}_1 = \begin{bmatrix} 0.1 & -1 \\ 2 & 0.1 \end{bmatrix}$, $\mathbf{A}_2 = \begin{bmatrix} 0.1 & -2 \\ 1 & 0.1 \end{bmatrix}$.

Nếu $x_1(t)x_2(t) < 0$, hệ thống chuyển sang trạng thái A1. Nếu $x_1(t)x_2(t) \geq 0$, hệ thống chuyển sang trạng thái A2. Nếu điều kiện ban đầu $x_1(0) = x_2(0) = 5$, hãy vẽ đường cong trạng thái biến trong khoảng $t \in (0, 20)$.

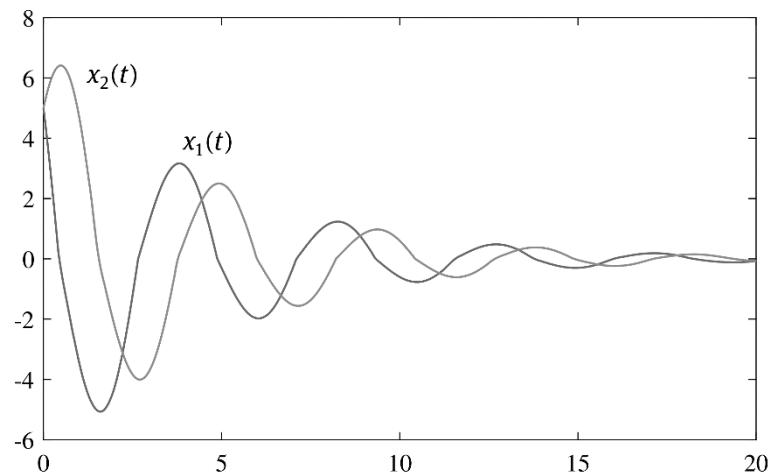
- B1: khai báo ma trận A1, A2 trên MATLAB:

```
>> A1=[0.1,-1; 2,0.1]; A2=[0.1,-2; 1,0.1]; eig(A1), eig(A2)
```

- B2: thiết lập mô hình SIMULINK như sau (lưu ý khai báo trên khôi Switch):

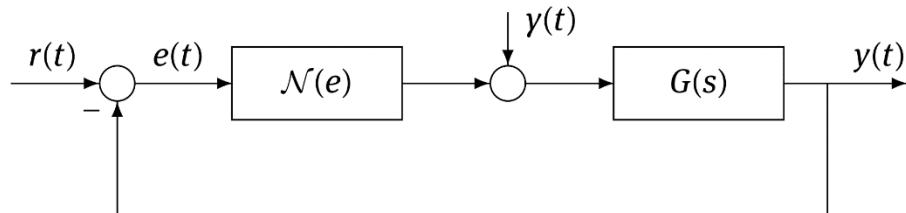


- B3: sử dụng lệnh trên MATLAB để cho ra kết quả như sau:



1.3.17 Mô hình hóa ODEs với đầu vào ngẫu nhiên

Bài toán 1.19: hệ thống điều khiển phản hồi không tuyến tính như sau:



Hàm truyền hệ thống như sau:

$$G(s) = \frac{s^3 + 7s^2 + 24s + 24}{s^4 + 10s^3 + 35s^2 + 50s + 24},$$

Đặc trưng bão hòa phi tuyến được mô tả như sau:

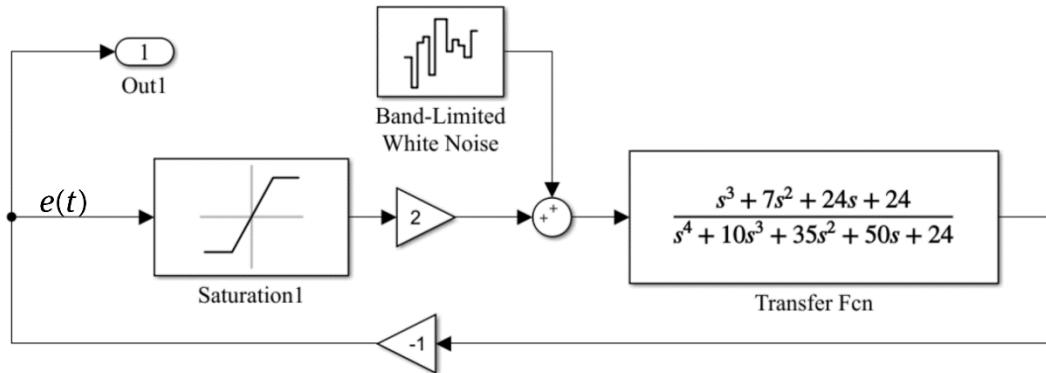
$$\mathcal{N}(e) = \begin{cases} 2e, & |e| \leq 1, \\ 2\text{sign}(e), & |e| > 1. \end{cases}$$

Nếu tín hiệu đầu vào xác định là $r(t) = 0$, phương sai của tín hiệu nhiễu trắng Gaussian $y(t)$ là $\sigma^2 = 4$. Hãy tìm hàm mật độ xác suất của tín hiệu $e(t)$. So sánh các

giải pháp số và giải tích. $e(t)$ được xác định bởi phương trình Fokker-Planck gần đúng:

$$p(e(t)) = \begin{cases} 0.2950e^{-0.3496e^2(t)}, & |e(t)| \leq 1, \\ 0.3629e^{-0.1114e^2(t)-0.4454e(t)\text{sign}(e(t))}, & |e(t)| > 1. \end{cases}$$

- B1: thiết lập mô hình trên SIMULINK như sau:

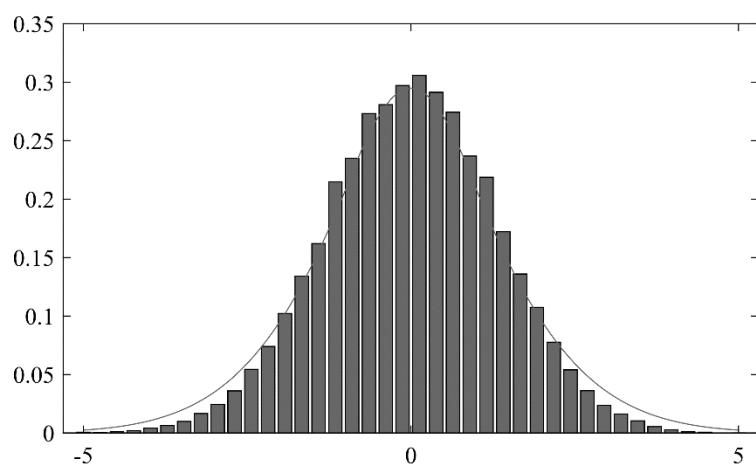


- B2: thiết lập thời gian lấy mẫu là $T=0.08$, thực hiện mô hình, ta nhận được kết quả trong biến $yout$.

- B3: thực hiện lệnh sau trên giao diện Command Window của MATLAB để vẽ hàm mật độ xác suất của tín hiệu $e(t)$.

```

>> ee=-5:0.1:5; % theoretical values
p=0.2950*exp(-0.3496*ee.^2).*abs(ee)<1)+...
    0.3629*exp(-0.1114*ee.^2-...
    0.4454*ee.*sign(ee)).*(abs(ee)>=1);
c=linspace(-5,5,40); y1=hist(yout,c);
bar(c,y1/(length(yout)*(c(2)-c(1)))), line(ee,p)
    
```



1.3.18 Mô hình hóa phương trình sai phân

Phương trình sai phân (difference equations) là một loại mô hình toán học khác để mô tả các hệ động lực. Sự khác biệt giữa phương trình vi phân và phương trình sai phân là ở chỗ, trong khi phương trình vi phân mô tả mối quan hệ giữa các tín hiệu liên tục, thì phương trình sai phân mô tả mối quan hệ giữa các tín hiệu rời rạc. Khái niệm

tín hiệu rời rạc là tín hiệu không liên tục, mà có liên quan đến các giá trị hàm tại các trường hợp thời gian cụ thể.

Dạng chuẩn của phương trình sai phân thông thường như sau:

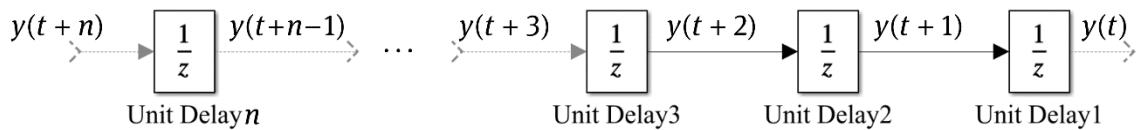
$$y((k+n)T) = f(k, y(kT), y((k+1)T), \dots, y((k+n-1)T)),$$

hoặc:

$$y(t+n) = f(t, y(t), y(t+1), \dots, y(t+n-1)).$$

với T là thời gian lấy mẫu, n là bậc của phương trình sai phân.

Nói chung phương pháp mô hình hoá phương trình sai phân tương tự như mô hình hoá phương trình vi phân, nhưng sử dụng các khối trong miền rời rạc z , điển hình khối Unit Delay.

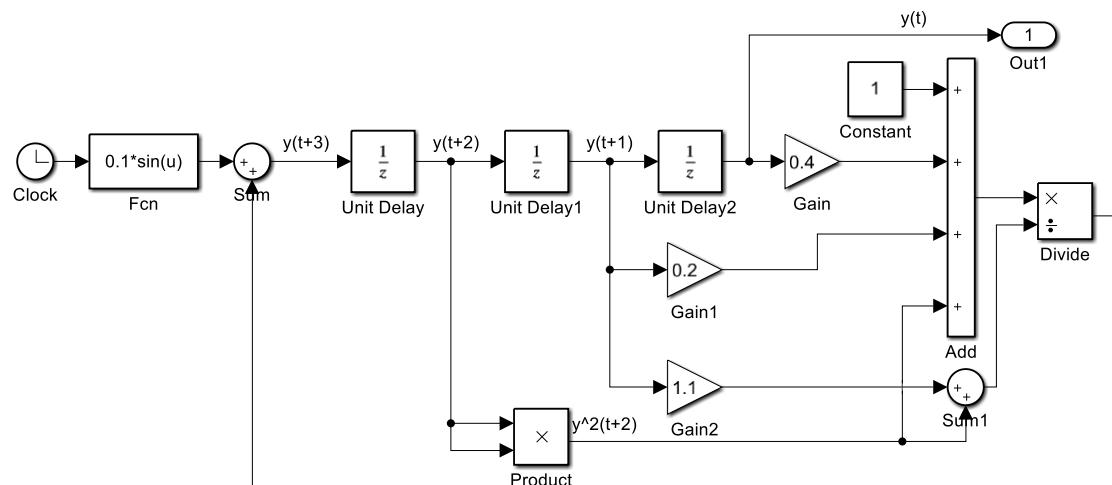


Bài toán 1.20: sử dụng SIMULINK, mô hình hoá phương trình sai phân sau:

$$y(t+3) = \frac{y^2(t+2) + 1.1y(t+1)}{1 + y^2(t+2) + 0.2y(t+1) + 0.4y(t)} + 0.1u(t),$$

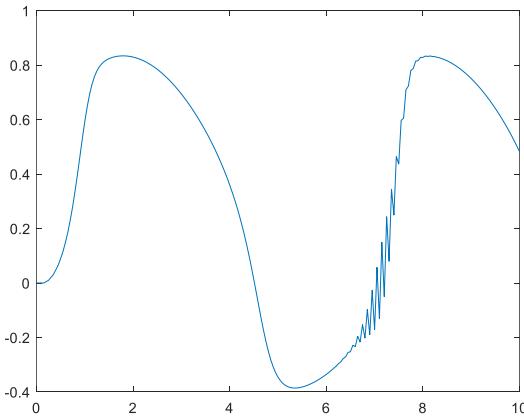
với tín hiệu đầu vào $u(t) = \sin(t)$, thời gian lấy mẫu $T=0.05s$.

- B1: thiết lập mô hình trên SIMULINK như sau:

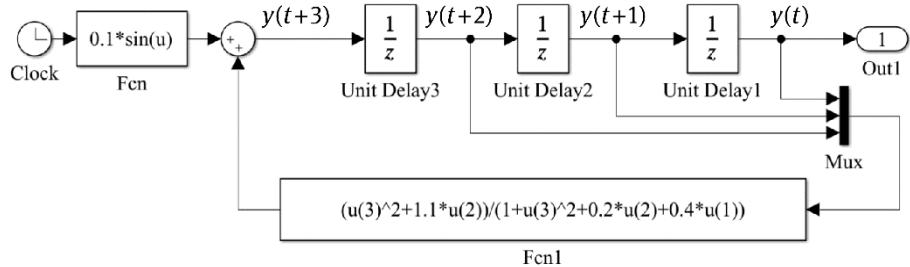


Lưu ý: để thiết lập thời gian lấy mẫu là $T=0.05s$, chỉ cần thiết lập thông số Sample time ở khối Unit Delay ngoài cùng bên trái, các khối còn lại không cần khai báo (do đã được khai báo ở khối trước đó). Tại khối Divide, thông số Number of inputs phải đặt là /*

- B2: sử dụng lệnh `plot(tout, yout)` trên giao diện Command Window của MATLAB, hoặc sử dụng khối Scope trong SIMULINK, ta được kết quả như sau:



- B3: sử dụng khối Mux để gộp các tín hiệu vector, thiết lập mô hình SIMULINK đơn giản hơn như sau:



So sánh kết quả với mô hình trước, rút ra nhận xét.

Bài toán 1.21: cho hệ phương trình sai phân sau:

$$\begin{cases} x_1(k+1) = x_1(k) + Tx_2(k), \\ x_2(k+1) = x_2(k) + Tfst(x_1(k), x_2(k), u(k), r, h), \end{cases}$$

Trong đó T là thời gian mẫu, $u(k)$ là tín hiệu đầu vào thứ k . Tham số r xác định tốc độ theo dõi, còn h là tham số cho kết quả lọc khi tín hiệu bị hỏng. Giá trị của hàm $fst(\cdot)$ có thể xác định trực tiếp bằng các công thức sau:

$$\delta = rh, \quad \delta_0 = \delta h, \quad y_0 = x_1(k) - u + hx_2(k), \quad a_0 = \sqrt{\delta^2 + 8r|y_0|},$$

$$a = \begin{cases} x_2 + y_0/h, & |y_0| \leq \delta_0, \\ x_2 + 0.5(a_0 - \delta) \text{sign}(y_0), & |y_0| > \delta_0, \end{cases}$$

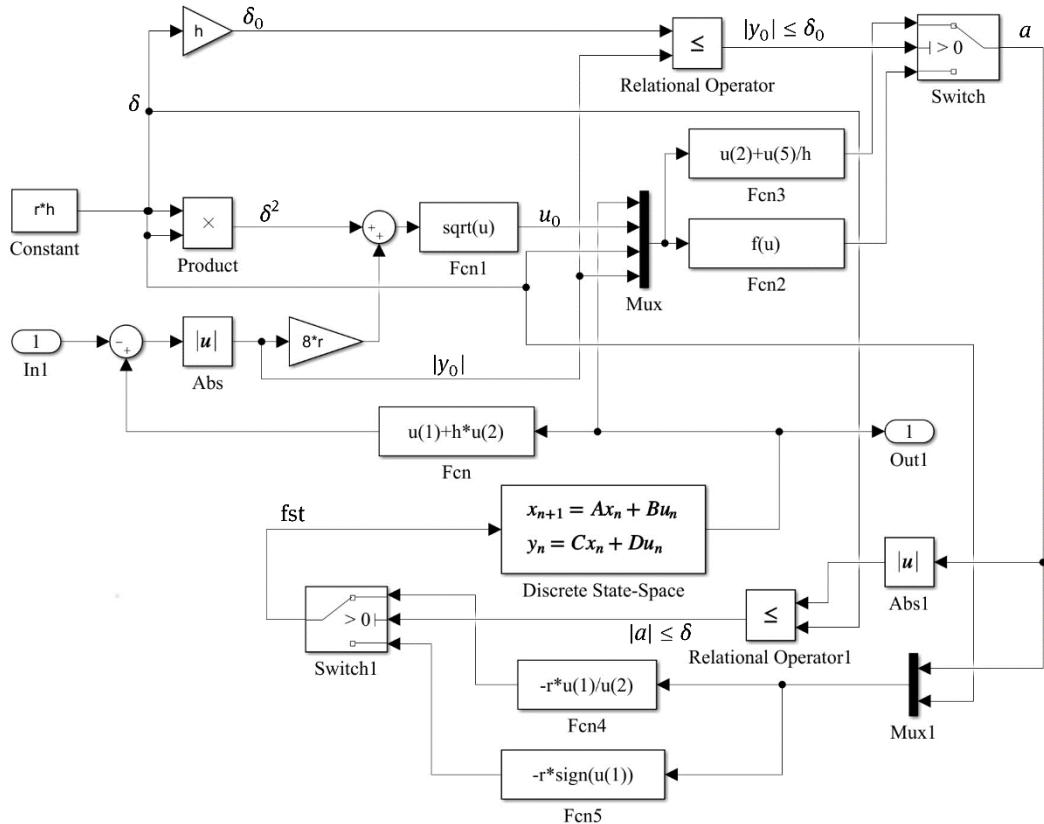
$$fst = \begin{cases} -ra/\delta, & |a| \leq \delta, \\ -r \text{sign}(a), & |a| > \delta. \end{cases}$$

Hãy xây dựng mô hình SIMULINK cho hệ phương trình trên.

Hệ phương trình sai phân đã cho là một hàm truyền rời rạc phi tuyến, trong đó $u(k)$ là tín hiệu đầu vào, $x_1(k)$ và $x_2(k)$ là tín hiệu đầu ra. Các tham số r , h và T là hằng số. Nếu hàm $fst(\cdot)$ được coi là tín hiệu đầu vào $v(t)$, thì mô hình không gian trạng thái rời rạc ở trên có thể được xem là mô hình không gian trạng thái rời rạc tuyến tính:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ T \end{bmatrix} v(k).$$

Dựa vào mô hình không gian trạng thái rời rạc tuyến tính, thiết lập mô hình SIMULINK như sau:



Lưu ý: mô hình không gian trạng thái rời rạc tuyến tính được mô tả bằng khối Discrete State-Space. Hàm trong khối Fcn2 được gán là $u(2) + 0.5 * (u(3) - u(4)) * sign(u(5))$.

1.4. Câu hỏi thực hành

Tìm hiểu Partial Differential Equation Toolbox trên phần mềm MATLAB.

TÀI LIỆU THAM KHẢO

- [1] Hoàng Minh Sơn, Cơ sở điều khiển quá trình, Nhà xuất bản Bách khoa Hà Nội, 2009.
- [2] Nguyễn Văn Chí, giáo trình điều khiển các quá trình công nghệ, Nhà xuất bản Khoa học kỹ thuật, 2016.
- [3] Nguyễn Phùng Quang, MATLAB&SIMULINK dành cho kỹ sư điều khiển tự động, Nhà xuất bản Khoa học kỹ thuật, 2003.

Bài 2. Mô hình hoá hệ thống điều khiển trên MATLAB/SIMULINK

2.1. Mục tiêu

Sau khi hoàn thành bài thực hành này, sinh viên có khả năng sử dụng được các công cụ MATLAB/SIMULINK để mô hình hoá hệ thống điều khiển.

2.2. Nội dung thực hành

Thực hiện mô hình hoá một số hệ thống điều khiển cơ bản như sau:

- Mô hình hoá hệ thống tuyến tính liên tục;
- Mô hình hoá hệ thống tuyến tính rời rạc;
- Mô hình hoá hệ thống phi tuyến.

2.3. Các bước thực hành

2.3.1 Mô hình hoá hệ thống tuyến tính liên tục

a) Mô hình hàm truyền

Hàm truyền được mô tả:

$$G(s) = \frac{b_1 s^m + b_2 s^{m-1} + \dots + b_m s + b_{m+1}}{a_1 s^n + a_2 s^{n-1} + \dots + a_n s + a_{n+1}},$$

trong đó a_i , $i = 1, 2, \dots, n+1$ và b_j , $j = 1, 2, \dots, m+1$ là các hằng số và n được gọi là bậc của mô hình. Nếu $m < n$, hệ thống hoàn toàn khả thi (có thể thực hiện được), trong khi nếu $m = n$, hệ thống khả thi. Nếu $n < m$, hệ thống không thể thực hiện được về mặt vật lý.

Trên MATLAB có thể sử dụng các lệnh sau để mô tả hàm truyền:

```
G=tf(num,den,'ioDelay',T);  
G=tf(num,den); G.ioDelay=T;  
s=tf('s'); G=tf(num,den)*exp(-T*s);
```

Bài toán 2.1: hãy khai báo trên MATLAB các quá trình có hàm truyền như sau:

$$G_1(s) = \frac{s^3 + 7s^2 + 24s + 24}{s^4 + 10s^3 + 35s^2 + 50s + 24}, \quad G_2(s) = \frac{s^2 + 4s + 3}{s^2(s + 4)((s + 1)^2 + 3)}.$$

Thực hiện đoạn code sau trên giao diện Command Window của MATLAB để khai báo:

```
>> G1=tf([1 7 24 24],[1 10 35 50 24])  
s=tf('s'); G2=(s^2+4*s+3)/s^2/(s+4)/((s+1)^2+3)
```

Trên SIMULINK, khôi Transfer Fcn trong nhóm Continuous có thể được sử dụng để mô tả hàm truyền một cách trực tiếp. Tuy nhiên, nếu hàm truyền không thể thực hiện được về mặt vật lý hoặc nếu hàm truyền có điều kiện ban đầu

khác 0 thì khối này không thể được sử dụng. Nếu đối tượng hàm truyền G đã được tạo như ở trên, lệnh $[n, d] = \text{tfdata}(G, 'v')$ có thể được sử dụng để trích xuất các vectơ hệ số tử số và mẫu số và trả về chúng trong các biến n và d . Nếu mô hình có độ trễ thời gian, có thể được trích xuất bằng thời gian trễ này bằng lệnh $T=G.\text{iDelay}$. Thông tin có thể được điền vào khối Transfer Fcn, sao cho hàm truyền có thể được biểu diễn trực tiếp trong SIMULINK.

b) Mô hình trạng thái

Mô hình toán học của hệ thống bất biến tuyến tính được mô tả như sau:

$$\begin{cases} \mathbf{x}'(t) = \mathbf{Ax}(t) + \mathbf{Bu}(t), \\ \mathbf{y}(t) = \mathbf{Cx}(t) + \mathbf{Du}(t), \end{cases}$$

trong đó $\mathbf{x}(t) \in R^{n \times 1}$ được gọi là vectơ trạng thái. Các ma trận hằng số $A \in R^{n \times n}$, $B \in R^{n \times q}$, $C \in R^{p \times n}$ và $D \in R^{p \times p}$ đã biết. Các giá trị ban đầu của trạng thái $x(t_0)$ tại thời điểm $t = t_0$ cũng đã biết, với giá trị mặc định là vectơ 0.

Trên MATLAB chúng ta sử dụng hàm ss để khai báo với cú pháp:

$$G = \text{ss}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$$

Bài toán 2.2: Hãy khai báo trên MATLAB quá trình có mô hình trạng thái như sau:

$$\begin{cases} \mathbf{x}'(t) = \begin{bmatrix} -12 & -17.2 & -16.8 & -11.9 \\ 6 & 8.6 & 8.4 & 6 \\ 6 & 8.7 & 8.4 & 6 \\ -5.9 & -8.6 & -8.3 & -6 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1.5 & 0.2 \\ 1 & 0.3 \\ 2 & 1 \\ 0 & 0.5 \end{bmatrix} \mathbf{u}(t), \\ \mathbf{y}(t) = \begin{bmatrix} 2 & 0.5 & 0 & 0.8 \\ 0.3 & 0.3 & 0.2 & 1 \end{bmatrix} \mathbf{x}(t). \end{cases}$$

Thực hiện đoạn code sau trên giao diện Command Window của MATLAB để khai báo:

```
>> A=[-12,-17.2,-16.8,-11.9; 6,8.6,8.4,6; 6,8.7,8.4,6;
      -5.9,-8.6,-8.3,-6];
B=[1.5,0.2; 1,0.3; 2,1; 0,0.5];
C=[2,0.5,0,0.8; 0.3,0.3,0.2,1]; D=zeros(2,2); %input the matrices
G=ss(A,B,C,D) %input and dispace state space modek
```

Nếu đã biết đối tượng hàm truyền G , lệnh $G1 = \text{ss}(G)$ có thể được sử dụng để chuyển G sang mô hình không gian trạng thái tương đương. Sau khi chuyển đổi, lệnh $[A, B, C, D] = \text{ssdata}(G1)$ có thể được sử dụng để trích xuất ma trận. Nếu đối tượng không gian trạng thái là $G1$, lệnh $G = \text{tf}(G1)$ có thể được sử dụng để tìm đối tượng mô hình hàm truyền G .

Trên SIMULINK, mô hình không gian trạng thái có thể được biểu diễn trực tiếp bằng khối State Space trong nhóm Continuous. Khai báo các ma trận A, B, C

và D trong hộp thoại tham số. Nếu mô hình có các điều kiện ban đầu khác không là $x(t_0)$, thì có thể được khai báo trong hộp Initial States.

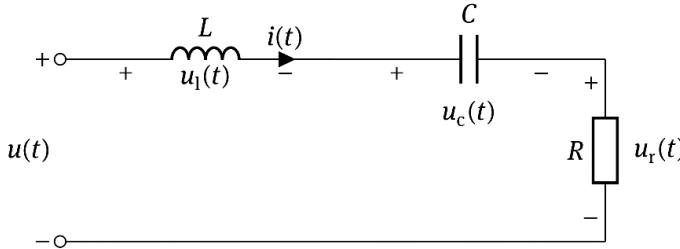
c) Mô hình không gian trạng thái tuyến tính đơn

Mô hình toán học của hệ thống tuyến tính đơn được mô tả như sau:

$$\begin{cases} \mathbf{Ex}'(t) = \mathbf{Ax}(t) + \mathbf{Bu}(t), \\ \mathbf{y}(t) = \mathbf{Cx}(t) + \mathbf{Du}(t), \end{cases}$$

trong đó $E \in R^{n \times n}$ được gọi là ma trận mô tả. Nếu mô hình không gian trạng thái đã biết, lệnh $G=dss(A, B, C, D, E)$ được sử dụng để khai báo mô hình.

Bài toán 2.3: xét mạch điện trở, tụ điện và cuộn cảm như trong hình sau:



Nếu các trạng thái được chọn là $i(t)$, $u_l(t)$, $u_c(t)$ và $u_r(t)$, mô hình không gian trạng thái của mạch được thể hiện như sau:

$$\begin{bmatrix} L & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} i'(t) \\ u'_l(t) \\ u'_c(t) \\ u'_r(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1/C & 0 & 0 & 0 \\ -R & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} i(t) \\ u_l(t) \\ u_c(t) \\ u_r(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix} u(t),$$

$$y(t) = u_c(t)$$

Nếu $R = L = C = 1$, hãy sử dụng MATLAB để vẽ đường cong đáp ứng của mạch.

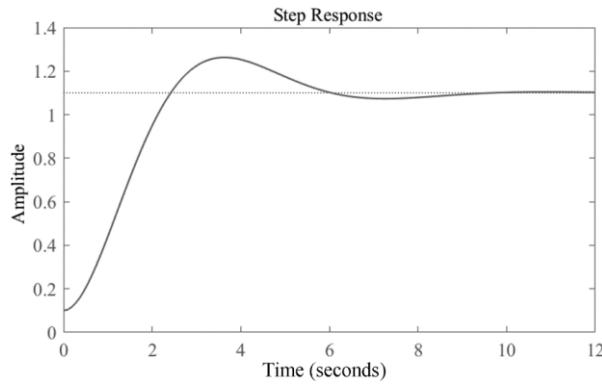
Khi $R = L = C = 1$, ta có mô hình trạng thái của mạch được viết lại:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}'(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix} u(t), \quad y(t) = x_3(t).$$

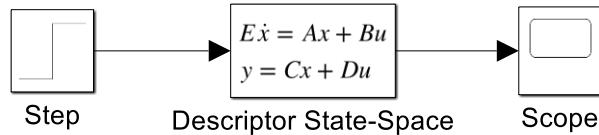
Thực hiện đoạn code sau trên giao diện Command Window của MATLAB:

```
>> E=[1,0,0,0; 0,0,1,0; 0,0,0,0; 0,0,0,0];
A=[0,1,0,0; 1,0,0,0; -1,0,0,1; 0,1,1,1];
B=[0; 0; 0; -1]; C=[0,0,1,0]; %input matrices
G=dss(A,B,C,0,E); step(G); % draw each states
G1=zpk(G) % find equivalent model
```

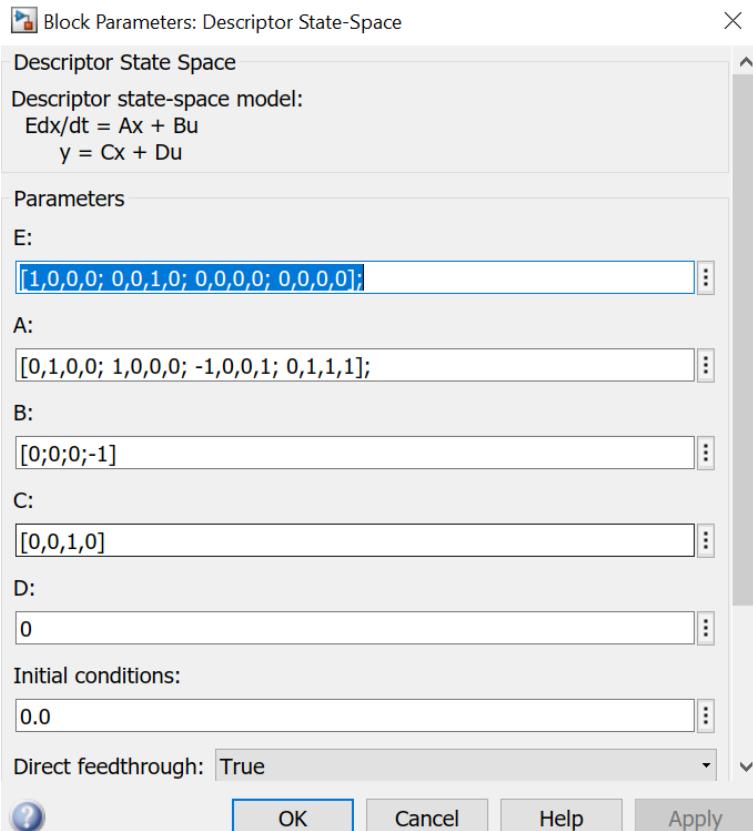
ta nhận được đường cong đáp ứng của mạch.



Trên SIMULINK, chúng ta thực hiện như sau:

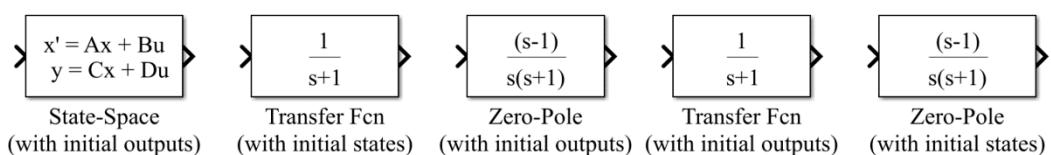


Khai báo tại khối Descriptor State-Space như sau:



d) Hàm truyền với điều kiện ban đầu khác không

Đối với các hàm truyền có các điều kiện ban đầu khác không, có thể sử dụng các khối trong nhóm thư viện Additional Linear trong Simulink Extras để khai báo.



Bài toán 2.4: xây dựng lại mô hình SIMULINK cho Bài toán 1.8 ở Bài thực hành số 1, sử dụng mô hình hàm truyền với điều kiện ban đầu khác không.

Trong Bài toán 1.8 ở Bài thực hành số 1, các khối derivative được sử dụng nên độ chính xác mô phỏng rất thấp và khá tốn thời gian. Ở bài toán này, mô hình không gian trạng thái tương đương được sử dụng và mô hình SIMULINK có thể được thiết lập để giải phương trình vi phân. Mô hình hàm truyền tương đương:

$$G(s) = \frac{2s + 1}{s^4 + 5s^3 + 9s^2 + 7s + 2}.$$

- B1: viết hàm `find_x0` trên MATLAB để xây dựng mô hình không gian trạng thái:

```
function [x0,G1,T,Y]=find_x0(G,y0,u0)
[num,den]=tfdata(G,'v'); n=length(den)-1;
d0=den(1); den=den/d0; num=num/d0; Y=y0(1); D=num(1);
y0=[y0(:); zeros(n-length(y0),1)];
u0=[u0(:); zeros(n-length(u0)-1,1)];
num=num-D*den; num=num(2:end); b=num; a=den(2:end);
A=diag(ones(1,n-1),1); A(end,:)==-den(end:-1:2);
B=[zeros(n-1,1); 1]; C=num(n:-1:1); G1=ss(A,B,C,0);
for k=2:n
    b(k,n)=-b(k-1,1)*a(n);
    for i=1:n-1, b(k,i)=b(k-1,i+1)-b(k-1,1)*a(i); end
    Y=[Y; y0(k)-b(1:k-1,1).'*u0(1:k-1)];
end
T=fliplr(b); x0=inv(T)*Y;
```

- B2: sử dụng hàm `find_x0` để tìm ra mô hình không gian trạng thái tương đương và các giá trị khởi tạo:

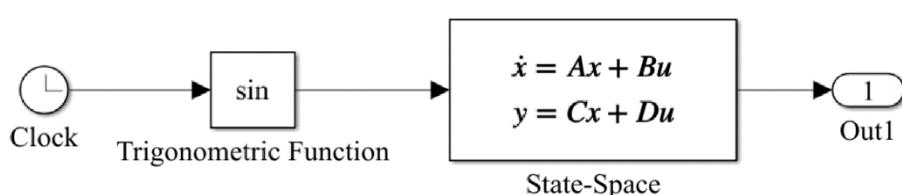
```
>> num=[2,1]; den=[1,5,9,7,2]; G=tf(num,den);
u0=[0;1;0]; y0=[1;0;0;0];
[x0,G1,T]=find_x0(G,y0,u0) %find the equivalent model
```

Mô hình không gian trạng thái tương đương và giá trị khởi tạo thu được:

$$\mathbf{x}'(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -2 & -7 & -9 & -5 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(t), \quad y(t) = [1, 2, 0, 0] \mathbf{x}(t).$$

$$\mathbf{x}_0 = [-29, 16, -8, 4]^T / 3$$

- B3: thiết lập mô hình trên SIMULINK như sau:



- B4: sử dụng lệnh trên MATLAB để mô phỏng và so sánh các kết quả:

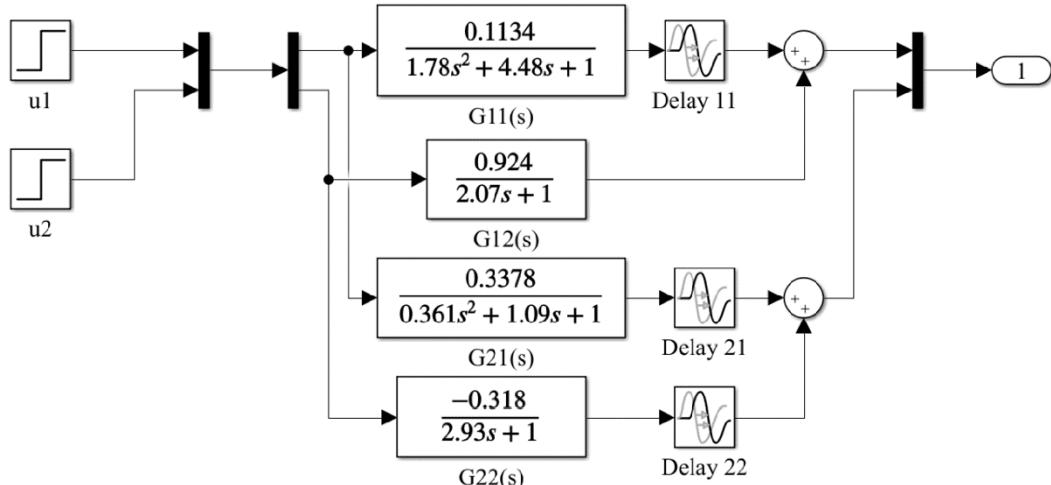
```
>> [A,B,C,D]=ssdata(G1);
tic, [t1,~,y1]=sim('c5mlin1'); toc, length(t1)
y0=7*exp(-t1)/4-2*exp(-2*t1)/5+t1.*exp(-t1)+...
3*t1.^2.*exp(-t1)/4-sqrt(2)*cos(t1-atan(1/7))/4;
max(abs(y0-y1)), plot(t1,y1,'--',t1,y0)
```

e) Mô hình hoá hàm truyền dạng ma trận

Bài toán 2.5: mô hình hoá trên SIMULINK hàm truyền có dạng ma trận như sau:

$$G(s) = \begin{bmatrix} \frac{0.1134e^{-0.72s}}{1.78s^2 + 4.48s + 1} & \frac{0.924}{2.07s + 1} \\ \frac{0.3378e^{-0.3s}}{0.361s^2 + 1.09s + 1} & \frac{-0.318e^{-1.29s}}{2.93s + 1} \end{bmatrix}$$

- B1: Thiết lập mô hình trên SIMULINK như sau, lưu ý rằng đối với thời gian trễ trong các hàm truyền thành phần, chúng sử dụng khối Transport Delay để mô hình hoá:



- B2: sử dụng khối Scope trên SIMULINK, hoặc dòng lệnh trên MATLAB để vẽ đáp ứng của hàm truyền.

Lưu ý: đối với ma trận hàm truyền hoặc các mô hình tuyến tính khác, đối tượng tuyến tính bất biến theo thời gian (LTI), có thể sử dụng các hàm trong Control System Toolbox của MATLAB để mô hình hoá. Các hàm tf() và ss() được sử dụng để biểu diễn hàm truyền và các mô hình không gian trạng thái. Đối với bài toán trên, chúng ta có thể mô hình hoá bằng dòng lệnh như sau:

```
>> g11=tf(0.1134,[1.78 4.48 1],'ioDelay',0.72);
g12=tf(0.924,[2.07 1]);
g21=tf(0.3378,[0.361 1.09 1],'ioDelay',0.3);
g22=tf(-0.318,[2.93 1],'ioDelay',1.29);
G=[g11, g12; g21, g22]; %input the transfer function matrix as a matrix
```

Hoặc sử dụng hàm `tf()` để khai báo theo cách khác như sau (đối với một hàm truyền khác):

```
>> s=tf('s'); %define s operator
G=(1+3*exp(-s)/(s+1))/(s+2); %input the composite transfer function
```

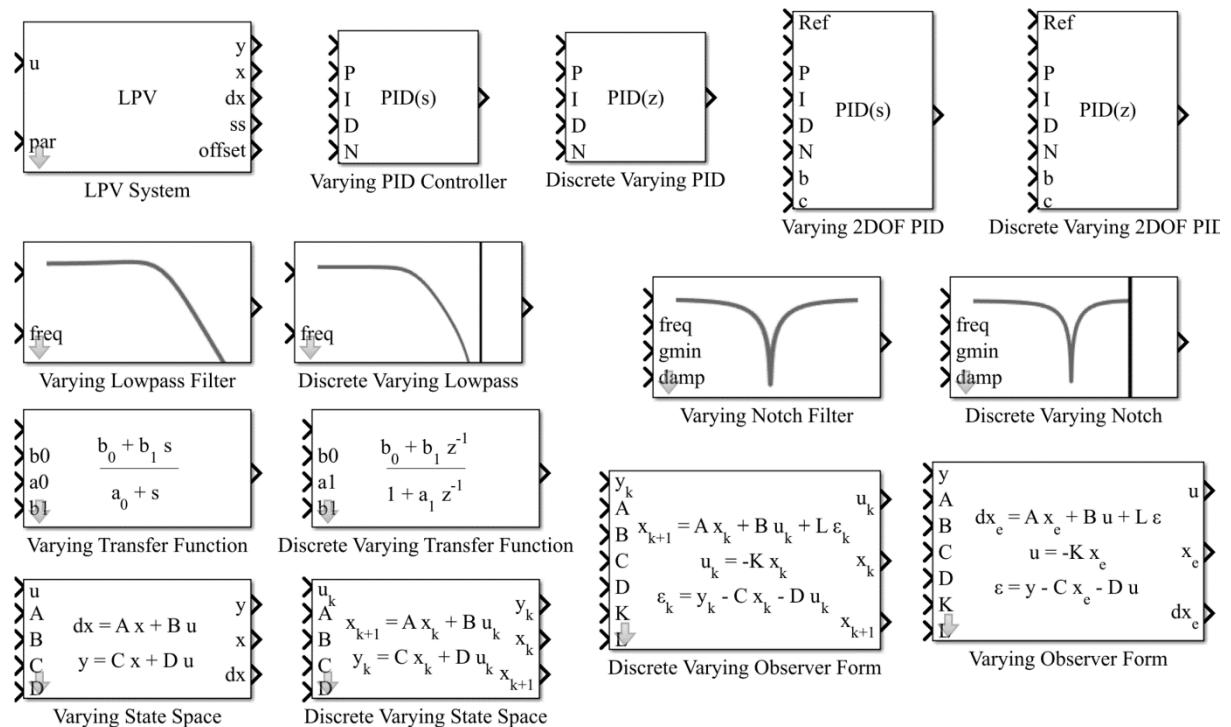
e) Mô hình hóa không gian trạng thái với các tham số khác nhau

Không gian trạng thái với các tham số khác nhau được thể hiện:

$$\begin{cases} \dot{x}(t) = A(p)x(t) + B(p)u(t), \\ y(t) = C(p)x(t) + D(p)u(t), \end{cases}$$

với $p = [p_1, p_2, \dots, p_m]$ là các tham số thay đổi.

Trên MATLAB, có thể sử dụng hàm `realp()` để mô tả. Còn trên SIMULINK, có thể sử dụng các khối trong nhóm Linear Parameter Varying.



2.3.2 Mô hình hóa hệ thống tuyến tính rời rạc

a) Mô hình toán học của hệ thống tuyến tính rời rạc

Mô hình toán học của hệ thống tuyến tính rời rạc được mô tả như sau:

$$\begin{aligned} a_1y((k+n)T) + a_2y((k+n-1)T) + \cdots + a_{n+1}y(kT) \\ = b_1u((k+n)T) + b_2u((k+n-1)T) + \cdots + b_{n+1}u(kT), \end{aligned}$$

với T là thời gian lấy mẫu của hệ thống rời rạc. Nếu đặt $t + n = (k + n)T$, mô hình được viết lại đơn giản hơn như sau:

$a_1y(t+n) + a_2y(t+n-1) + \cdots + a_{n+1}y(t) = b_1u(t+n) + b_2u(t+n-1) + \cdots + b_{n+1}u(t)$,
hoặc:

$$a_1y_{t+n} + a_2y_{t+n-1} + \cdots + a_{n+1}y_t = b_1u_{t+n} + b_2u_{t+n-1} + \cdots + b_{n+1}u_t.$$

Hàm truyền của hệ thống này được biểu diễn như sau:

$$G(z) = \frac{b_1 z^n + b_2 z^{n-1} + \cdots + b_n z + b_{n+1}}{a_1 z^n + a_2 z^{n-1} + \cdots + a_n z + a_{n+1}}$$

Không gian trạng thái của hệ thống này được biểu diễn như sau:

$$\begin{cases} \mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{G}\mathbf{u}_t, \\ \mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{D}\mathbf{u}_t. \end{cases}$$

Các hàm `tf()` and `ss()` của Control System Toolbox được sử dụng để mô hình hoá hệ thống này, với cú pháp:

```
G=tf(num,den,'Ts',T);
z=tf('z',T);
G=ss(A,B,C,D,'Ts',T);
```

Bài toán 2.6: mô hình hoá các hàm truyền sau với thời gian lấy mẫu $T=0.1$ s:

$$G_1(z) = \frac{2z + 1}{z^4 + 2.6z^3 + 2.52z^2 + 1.08z + 0.1728}, \quad G_2(z) = \frac{2z + 1}{(z + 0.6)^3(z + 0.8)}.$$

Sử dụng lệnh trên MATLAB như sau:

```
>> G1=tf([2 1],[1 2.6 2.52 1.08 0.1728],'Ts',0.1)
z=tf('z',0.1); G2=(2*z+1)/(z+0.6)^3/(z+0.8)
```

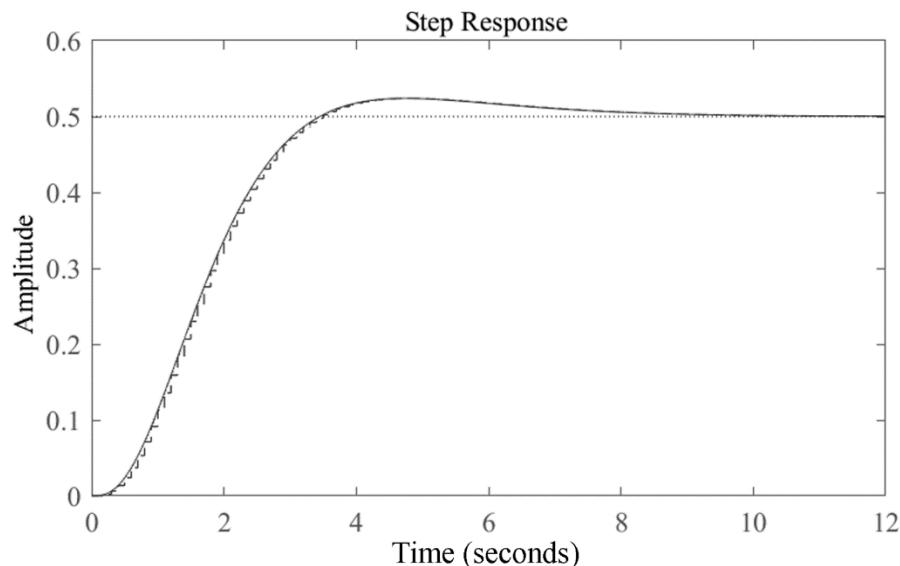
b) *Chuyển đổi giữa mô hình liên tục và mô hình rời rạc*

Trong Control System Toolbox, sử dụng các hàm `c2d()` và `d2c()`.

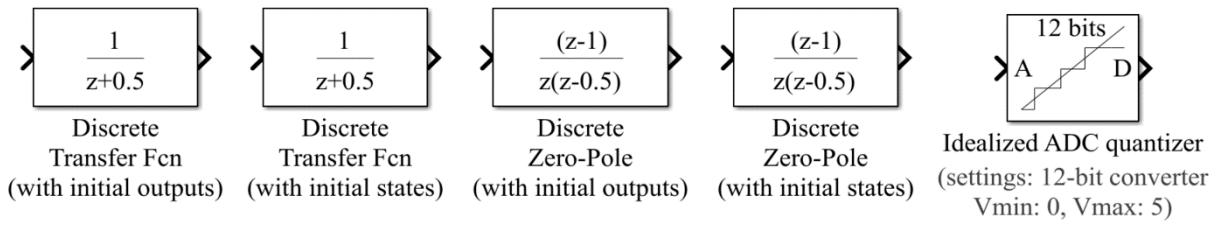
Bài toán 2.7: cho hệ thống ở bài toán 2.4. Hãy rời rạc hoá hệ thống này với các thời gian lấy mẫu khác nhau: $T=0.1$ s, $T=0.01$ s. So sánh đáp ứng bước của các mô hình này.

Thực hiện lệnh sau để thực hiện bài toán:

```
>> num=[2,1]; den=[1,5,9,7,2]; G=tf(num,den);
G1=c2d(G,0.1), G2=c2d(G,0.01), step(G,G1,'--',G2,:')
```

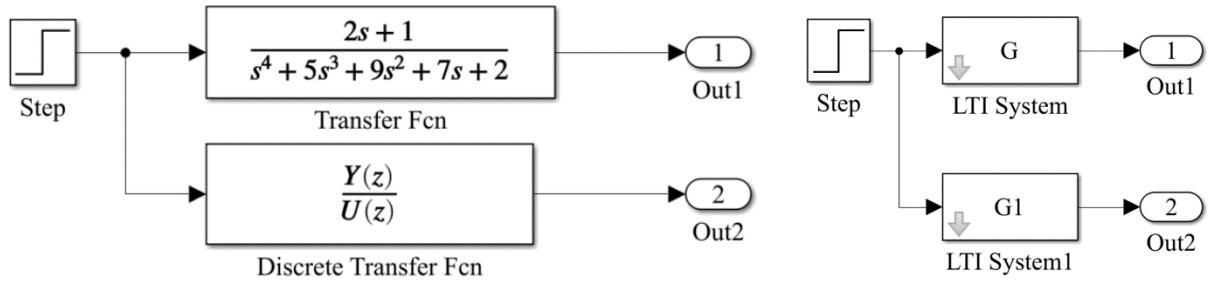


Trên SIMULINK, chúng ta sử dụng các khối trong nhóm Additional Discrete để mô tả các hệ thống rời rạc.



Bài toán 2.8: đối với hệ thống ở Bài toán 2.6, hãy mô hình hoá trên SIMULINK.

Trên SIMULINK, có thể thiết lập 2 dạng mô hình như sau:



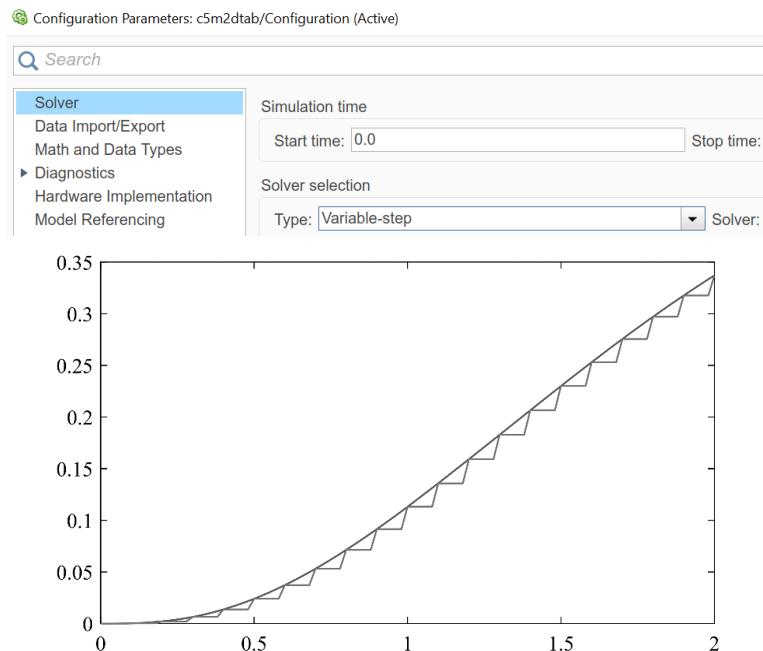
Lưu ý, các thông tin sau cần đưa vào Model Properties:

```
num=[2,1]; den=[1,5,9,7,2]; G=tf(num,den);
G1=c2d(G,0.1); [numd,dend]=tfdata(G1,'v');
```

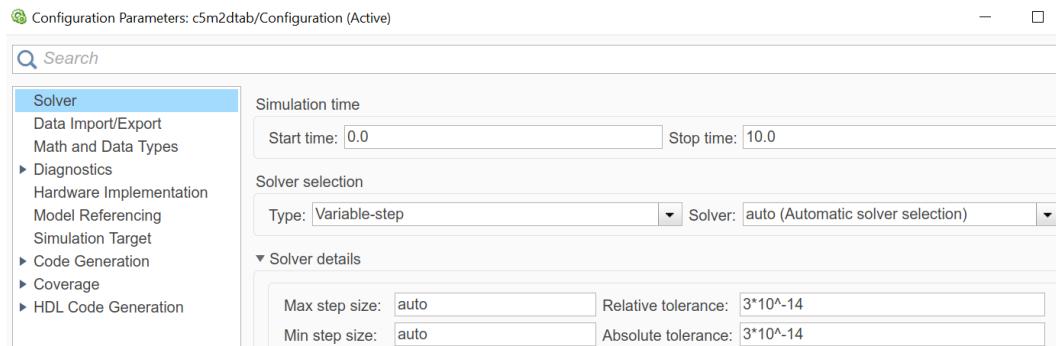
c) Tác động của các thông số điều khiển mô phỏng

Bài toán 2.9: đối với hệ thống ở Bài toán 2.7, đặt $T=0.1$ s, sau đó hãy rời rạc hoá mô hình. Phân tích ảnh hưởng của các thông số điều khiển đến kết quả mô phỏng.

Vì có cả khối liên tục và khối rời rạc nên không thể sử dụng thuật toán Fixed-step. Thay vào đó, nên áp dụng thuật toán Variable-step. Kết quả mô phỏng:



Để quan sát kết quả mô phỏng tốt hơn, Stop time được đặt thành 2s. Có thể thấy, mặc dù kết quả mô phỏng trên các mẫu rời rạc là chính xác nhưng mô hình này không giống với các hệ thống rời rạc dự kiến, vì trong trường hợp bình thường, đầu ra của các hệ thống rời rạc có dạng bậc thang. Để giải quyết vấn đề này, Relative tolerance được đặt thành các giá trị rất nhỏ, chẳng hạn như 3×10^{-14} .

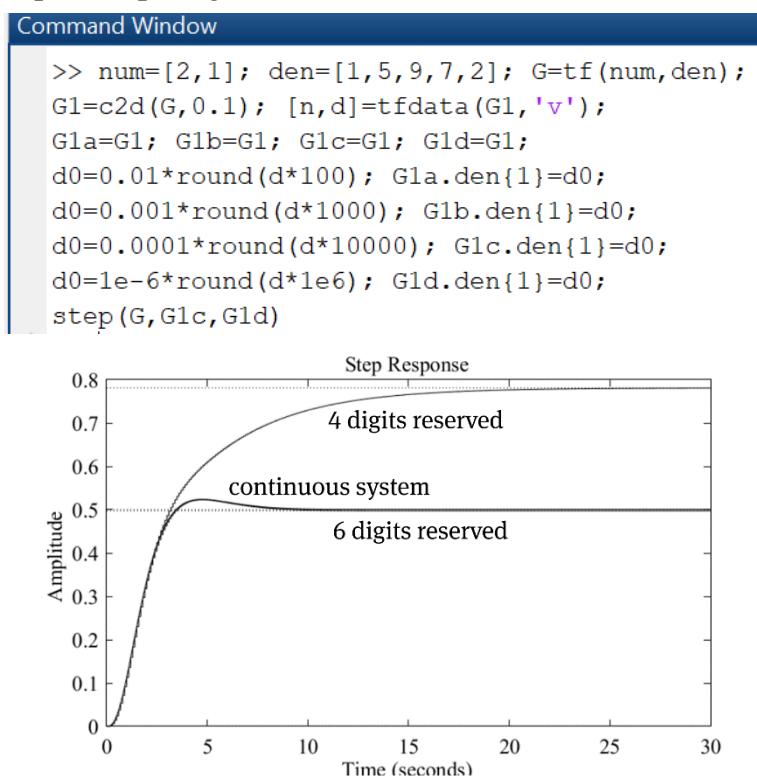


Lúc này, chúng ta nhận được đồ thị chính xác như ở Bài toán 6. Có thể thấy rằng các thông số điều khiển có thể tác động đáng kể đến kết quả mô phỏng đối với các hệ thống rời rạc. Nếu kết quả mô phỏng chính xác được mong đợi, thì nếu có thể, dung sai lỗi phải được đặt ở giá trị nhỏ nhất có thể.

d) Tác động của độ chính xác của các tham số rời rạc

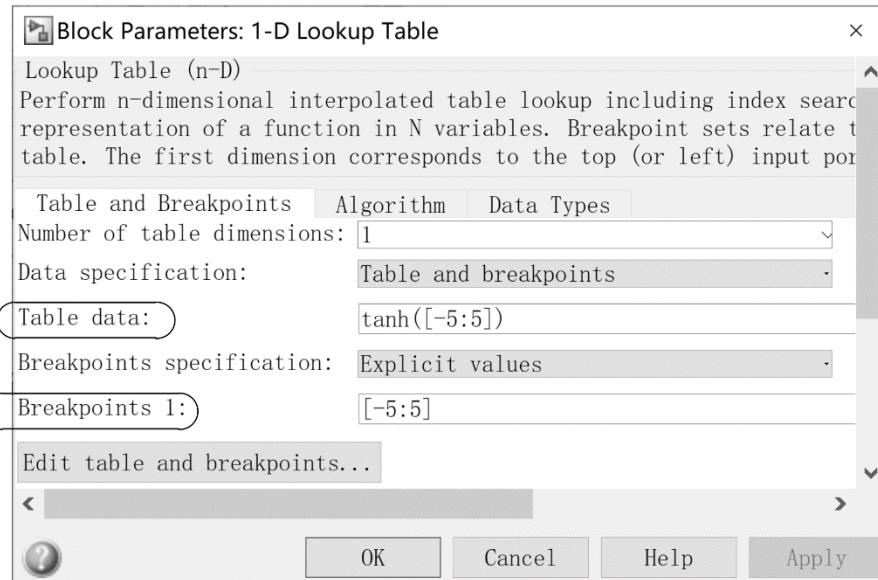
Bài toán 2.10: đối với hệ thống ở Bài toán 2.7, đặt $T=0.1$ s, sau đó hãy rời rạc hoá mô hình. Phân tích ảnh hưởng của các tham số rời rạc đến kết quả mô phỏng.

Hàm `tfdata()` có thể được sử dụng để trích xuất tử số và các hệ số. Thực hiện lệnh sau trên giao diện Command Window để biết được ảnh hưởng của các tham số rời rạc đến kết quả mô phỏng:



2.3.3 Mô hình hoá hệ thống phi tuyến

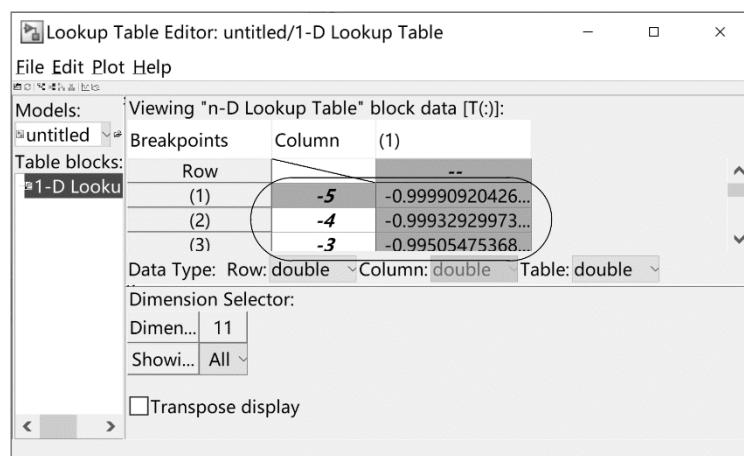
a) Khối nội suy tuyến tính 1-D Lookup Table



Thông số Number of table dimensions chỉ số chiều của bảng, mặc định là 1. Có 2 thông số quan trọng là Breakpoints 1 và Table data. Giá trị của chúng lần lượt là các vectơ ngang $u = [u_1, u_2, \dots, u_m]$ và vectơ dọc $y = [y_1, y_2, \dots, y_m]$. Chúng ta phải đặt thông số cho các vector này. Tín hiệu đầu ra của khối có thể được xác định duy nhất từ các giá trị hiện tại của tín hiệu đầu vào. Trong trường hợp bình thường, nếu tín hiệu đầu vào trong khói thỏa mãn $u_k \leq u(t) < u_{k+1}$, thì tín hiệu đầu ra $y(t)$ có thể được xác định như sau:

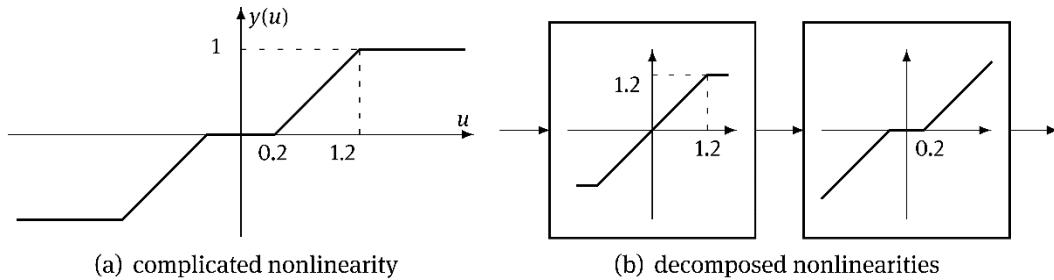
$$y(t) = \frac{y_{k+1} - y_k}{u_{k+1} - u_k}(u(t) - u_k).$$

Trong thực tế, thuật toán này được gọi là nội suy tuyến tính. Nếu muốn tính toán tín hiệu đầu ra bằng các thuật toán khác, có thể chọn tab Algorithm. Có thể chọn các thuật toán nội suy khác như Interpolation hoặc Extrapolation. Để chỉnh sửa giá trị của các điểm ngoặt, vào tab Edit table and breakpoints để thiết lập.

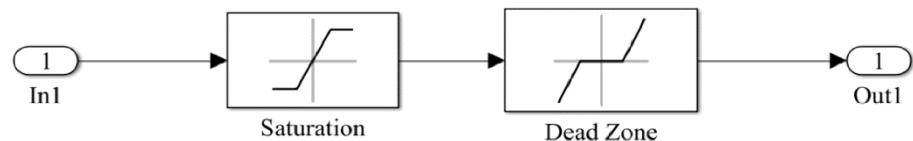


b) Xây dựng các phi tuyến có giá trị đơn

Bài toán 2.11: sử dụng khối 1-D Lookup Table để mô tả tính phi tuyến của hệ thống như sau:



- B1: xây dựng mô hình phi tuyến cấp thấp trên SIMULINK như sau, sử dụng các khối Saturation và Dead Zone, lưu ý thiết lập các thông số hệ thống cho 2 khối này:

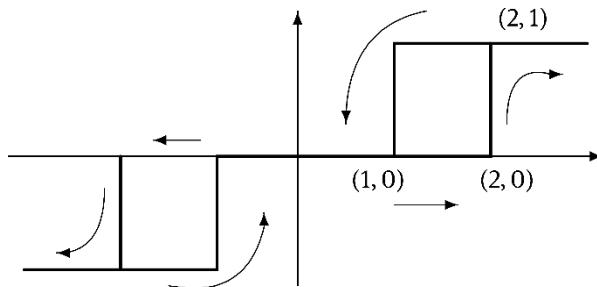


- B2: sử dụng khối 1-D Lookup Table để mô tả hệ thống. Từ các điểm phi tuyến trong hình (a), có thể thấy rằng các điểm ngoặt là $(-1.2, 1)$, $(-0.2, 0)$, $(0.2, 0)$ và $(1.2, 1)$. Nhưng sử dụng khối 1-D Lookup Table thì phải chọn thêm hai điểm. Ví dụ: có thể chọn hai điểm $(-1.5, 1)$ và $(2, 1)$, tạo ra hai vectơ $xx = [-1.5, -1.2, -0.2, 1.2, 2]$ và $yy = [-1, -1, 0, 0, 1, 1]$. Nhập 2 vector này vào giao diện Command Window của MATLAB. Sau đó, hai biến này có thể được thiết lập ở Edit boxes trong khối 1-D Lookup Table.

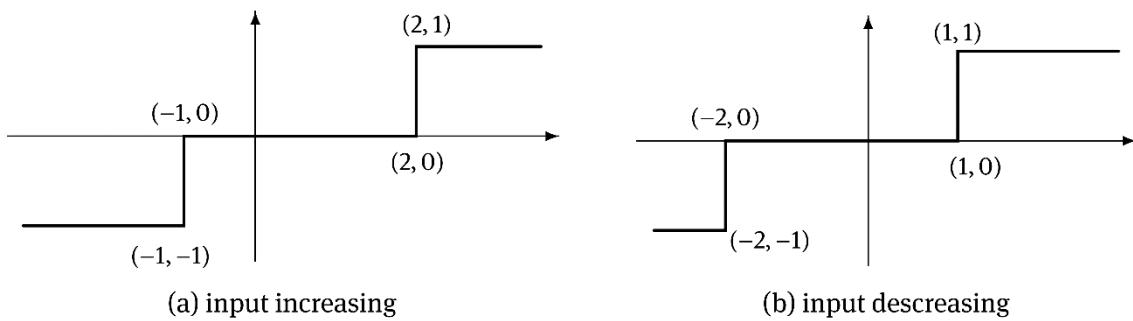
- B2: So sánh trực quan hai phương pháp mô tả trên.

c) Xây dựng các phi tuyến có giá trị đơn

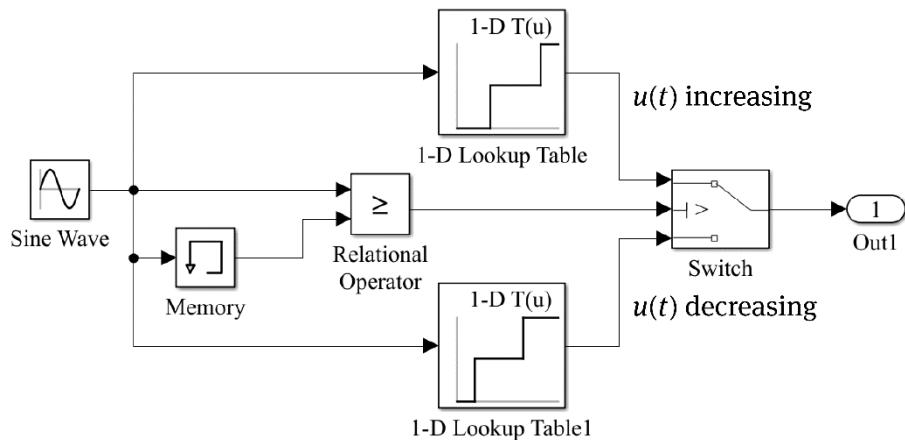
Bài toán 2.12: Xây dựng mô hình SIMULINK để mô tả tính phi tuyến có giá trị kép như sau:



Hàm phi tuyến có thể được phân tách thành hai hàm phi tuyến có giá trị đơn, như sau. Tất nhiên, hai hàm có giá trị đơn đều có điều kiện. Chúng được xác định bằng việc tín hiệu đầu vào tăng hay giảm.



Trên SIMULINK thiết lập mô hình như sau:

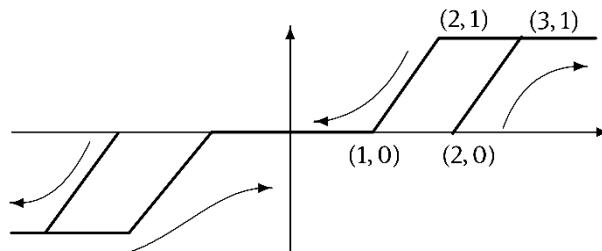


Lưu ý: thiết lập 2 vector điểm ngoặt như sau, với ϵ là giá trị rất nhỏ, ví dụ 10^{-10} .

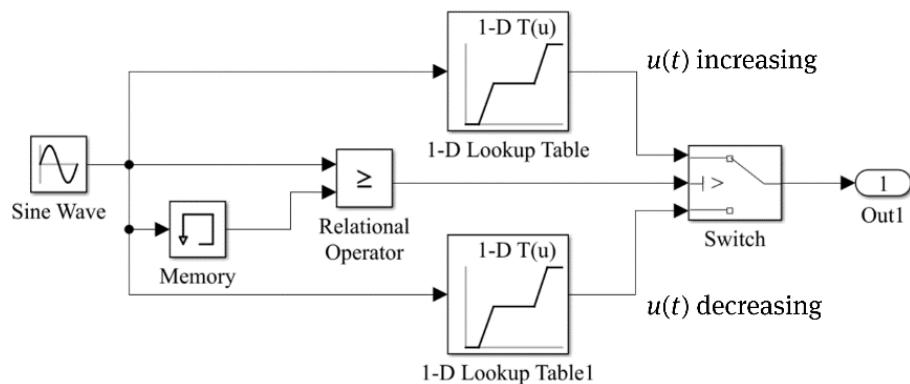
$$x_1 = [-3, -1, -1 + \epsilon, 2, 2 + \epsilon, 3], \quad y_1 = [-1, -1, 0, 0, 1, 1]$$

$$x_2 = [-3, -2, -2 + \epsilon, 1, 1 + \epsilon, 3], \quad y_2 = [-1, -1, 0, 0, 1, 1]$$

Bài toán 2.13: Xây dựng mô hình SIMULINK để mô tả tính phi tuyến có giá trị kép như sau:



Trên SIMULINK thiết lập mô hình như sau:



$$x_1 = [-3, -2, -1, 2, 3, 4], y_1 = [-1, -1, 0, 0, 1, 1]$$

$$x_2 = [-4, -3, -2, 1, 2, 3], y_2 = [-1, -1, 0, 0, 1, 1]$$

d) Khởi đà chiều 2-D Lookup Table

Bài toán 2.14: Xét hàm hai chiều như sau:

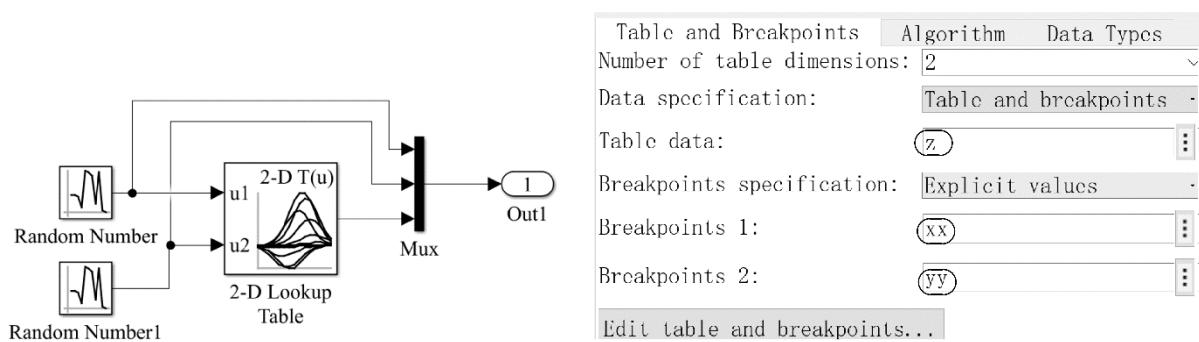
$$z = f(x, y) = (x^2 - 2x)e^{-x^2-y^2-xy}$$

Nếu có thể tạo ra các điểm lưới lưới phân bố thưa thớt trên mặt phẳng $x-y$ và giá trị hàm tại các điểm này có thể được đánh giá bằng:

```
>> xx=linspace(-3,3,15); yy=linspace(-2,2,15);
[x,y]=meshgrid(xx,yy);
z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
surf(xx,yy,z);
```

Hãy vẽ bề mặt 3 chiều của hàm bằng cách sử dụng SIMULINK.

- B1: trên SIMULINK thiết lập mô hình như sau:



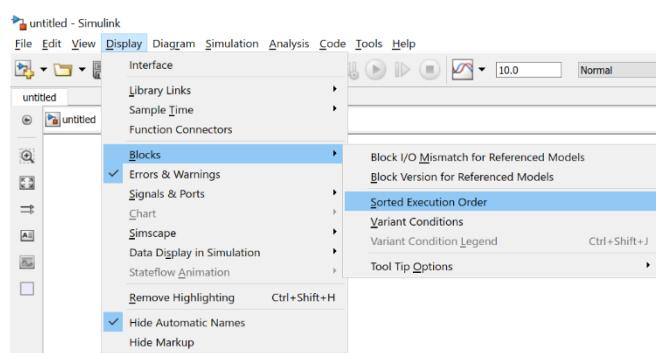
- B2: trên MATLAB sử dụng lệnh sau để vẽ bề mặt 3 chiều của hàm:

```
>> plot3(yout(:,2),yout(:,1),yout(:,3),'.')
```

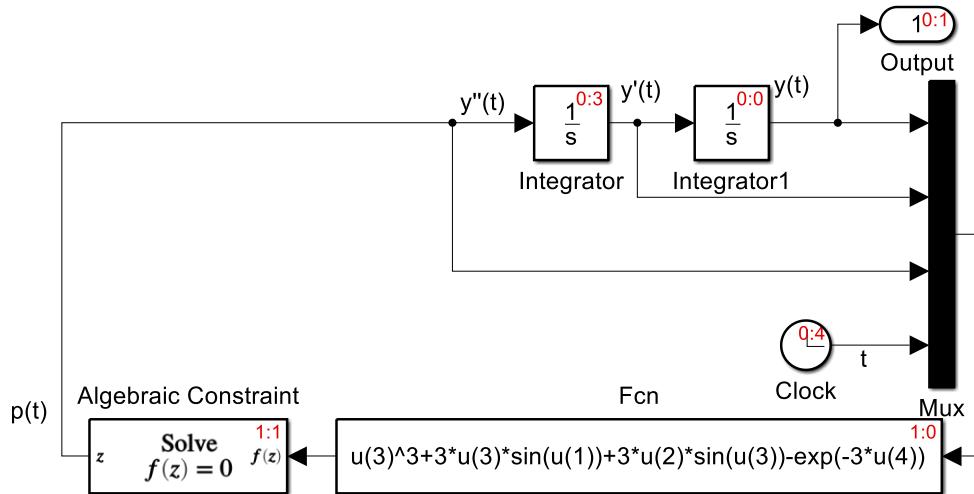
- B2: so sánh các đồ thị 3D được tạo ra bởi MATLAB và SIMULINK như trên.

2.3.4 Tự động sắp xếp các khối trong SIMULINK

Mô hình Simulink bao gồm các khối. Toàn bộ kết quả mô phỏng có thể được rút ra bằng cách thực hiện “lần lượt” các khối, có nghĩa là các khối được thực hiện theo một thứ tự nhất định. Trên cửa sổ SIMULINK, tùy chọn Display → Blocks → Sorted Execution Order có thể được chọn để hiển thị thứ tự thực hiện của các khối.



Lưu ý, chức năng này tuỳ thuộc vào các phiên bản MATLAB, hiện thị bên dưới dành cho MATLAB R2018b. Ví dụ:



2.3.5 Phân tích gần đúng hệ thống điều khiển phi tuyến

a) Xấp xỉ Padé của hệ thống trễ

Trong quá trình tuyến tính hoá các mô hình phi tuyến, phép xấp xỉ Padé thường được sử dụng. Đối với hàm truyền $f(s)$, xấp xỉ Taylor được biểu diễn như sau:

$$f(s) = c_1 + c_2 s + c_3 s^2 + \cdots + c_m s^{m-1} + \cdots$$

Phương pháp được thể hiện như sau:

$$G_{r/k}(s) = \frac{\beta_1 + \beta_2 s + \cdots + \beta_r s^{r-1} + \beta_{r+1} s^r}{\alpha_1 + \alpha_2 s + \cdots + \alpha_k s^{k-1} + \alpha_{k+1} s^k},$$

trong đó r và k lần lượt là bậc của tử số và mẫu số.

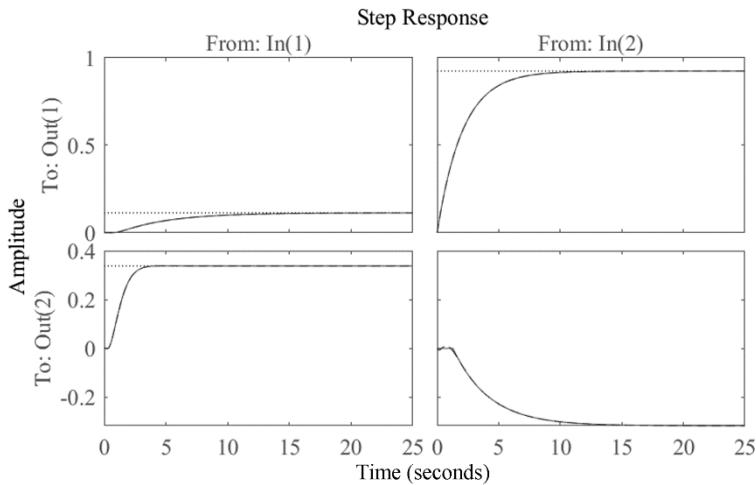
Trên MATLAB, chúng ta có thể sử dụng hàm `pade()` với cú pháp:

$$G = \text{pade}(T, r)$$

Bài toán 2.15: Xấp xỉ Padé cho ma trận hàm truyền trong Bài toán 2.5.

Trên MATLAB, chúng ta thực hiện như sau:

```
>> g11=tf(0.1134,[1.78 4.48 1], 'ioDelay', 0.72);
g12=tf(0.924,[2.07 1]);
g21=tf(0.3378,[0.361 1.09 1], 'ioDelay', 0.3);
g22=tf(-0.318,[2.93 1], 'ioDelay', 1.29);
G=[g11, g12; g21, g22];
step(G, pade(G, 2), '--') % Compare the step responses
```



b) Tuyến tính hoá các mô hình phi tuyến

Các hàm có thể được sử dụng:

- Hàm tìm các điểm cân bằng của mô hình:

$[x_0, u_0, y, \Delta x] = \text{trim}(\text{model name})$

$[x_0, u_0, y, \Delta x] = \text{trim}(\text{model name}, x_0, u_0)$ % specify the initial search points

- Tìm ma trận Jacobian: $\text{jacobian}()$

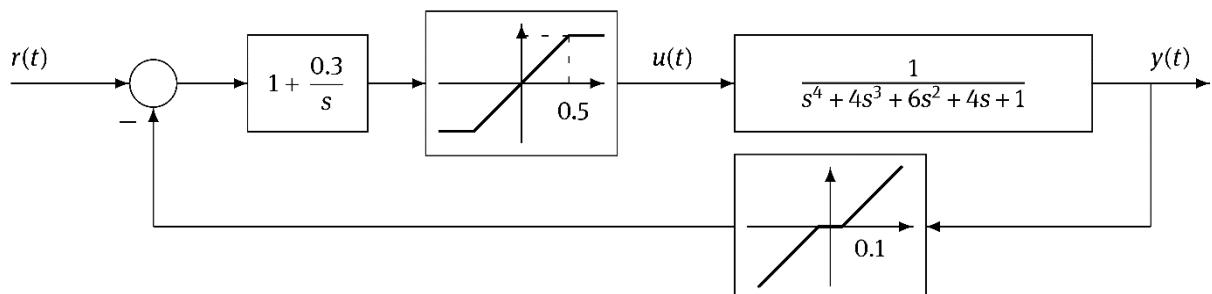
- Tuyến tính hoá mô hình phi tuyến: $G = \text{linearize}('Simulink name')$,

- Tuyến tính hoá và trích xuất các ma trận A,B,C,D sau khi tuyến tính hoá:

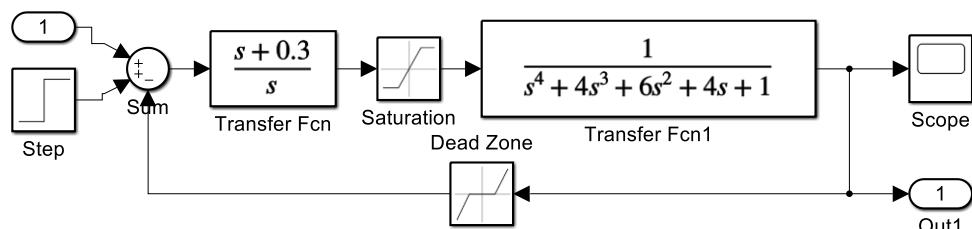
$[A, B, C, D] = \text{linmod2}('Simulink name', x_0, u_0)$

Lưu ý: đối với các mô hình SIMULINK phi tuyến liên tục thông thường, hàm $\text{linmod2}()$ được đề nghị. Nếu hệ thống phi tuyến có trễ thì nên sử dụng hàm $\text{linmod}()$. Xấp xỉ Padé có thể được sử dụng để phù hợp với trễ. Nếu có các phần tử rời rạc, nên sử dụng hàm $\text{dlinmod}()$ để tìm mô hình không gian trạng thái rời rạc.

Bài toán 2.16: Tuyến tính hoá mô hình phi tuyến sau và đánh giá chất lượng của kết quả xấp xỉ hoá.



- B1: thiết lập mô hình trên SIMULINK như sau, đặt tên c5mlinrl1:



Lưu ý thiết lập các tham số trong các khối Saturation và Dead Zone theo bài toán đặt ra.

- B2: sử dụng dòng lệnh trên MATLAB để tuyến tính hoá mô hình phi tuyến này, và nhận được hàm truyền của mô hình tuyến tính như sau:

```
Command Window
>> G=linearize('c5mlinr1'), zpk(G)
Continuous-time state-space model.

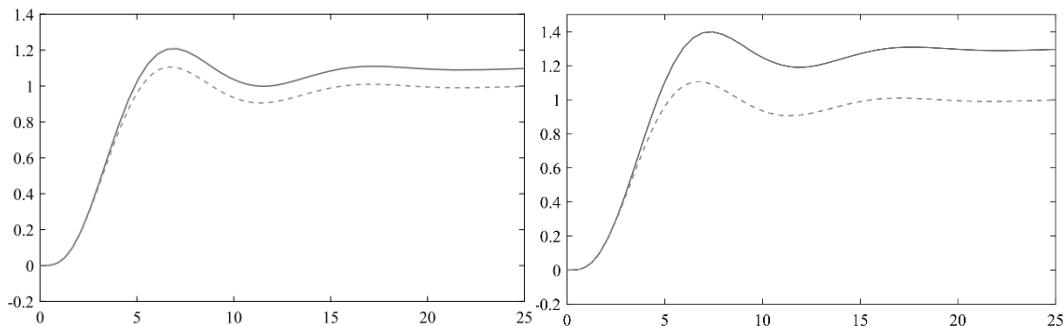
ans =
From input "In1" to output "Out1":
(s+0.3)
-----
(s+0.2185) (s^2 + 0.4498s + 0.4221) (s^2 + 3.332s + 3.253)

Continuous-time zero/pole/gain model.
```

- B3: sử dụng dòng lệnh trên MATLAB để so sánh mô hình phi tuyến và mô hình tuyến tính:

```
>> G=linearize('c5mlinr1'); G1=zpk(G)
[y1 t1]=step(G); plot(tout,yout,t1,y1,'--')
```

Nếu tăng độ rộng của Dead Zone, tức là tăng mức độ phi tuyến thì sự phù hợp của mô hình tuyến tính sẽ thấp hơn. Ví dụ Dead Zone là ± 0.2 và ± 0.3 như sau:

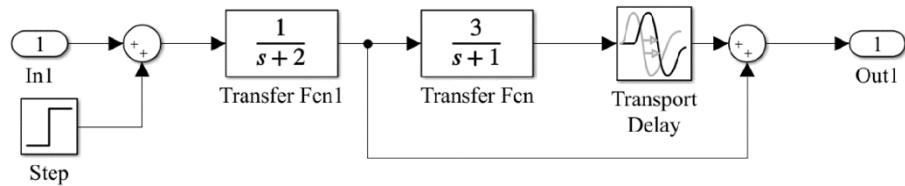


Như vậy có thể thấy rằng, việc tuyến tính hóa có thể dẫn đến sai số rất lớn. Kích thước của sai số phụ thuộc rất nhiều vào mức độ phi tuyến. Mức độ phi tuyến thì sai số càng lớn. Xét từ mô phỏng số, phải cẩn thận khi sử dụng tuyến tính hóa hoặc không sử dụng kỹ thuật tuyến tính hóa.

Bài toán 2.17: Tuyến tính hoá mô hình phi tuyến có hàm truyền như sau và và so sánh sự phù hợp của mô hình gốc và mô hình gần đúng thông qua phản hồi bước.

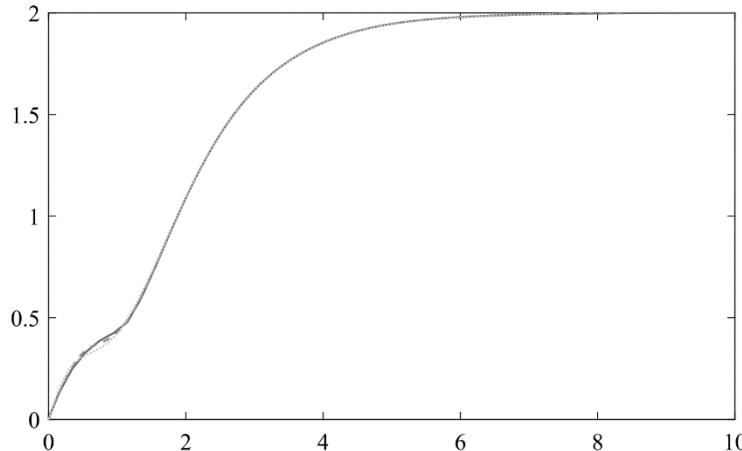
$$G(s) = \frac{1 + \frac{3e^{-s}}{s+1}}{s+2}$$

- B1: thiết lập mô hình trên SIMULINK như sau, lưu file là c5mlinr2.slx:



- B2: sử dụng lệnh sau trên MATLAB để tuyến tính hóa mô hình phi tuyến này, sau đó so sánh đáp ứng bước của 2 hệ thống.

```
>> G=linearize('c5mlinr2'); G1=zpk(G)
[y1 t1]=step(G); plot(tout,yout,t1,y1,'--')
```

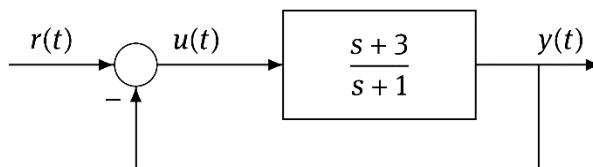


c) Hiện tượng vòng lặp đại số và cách loại trừ

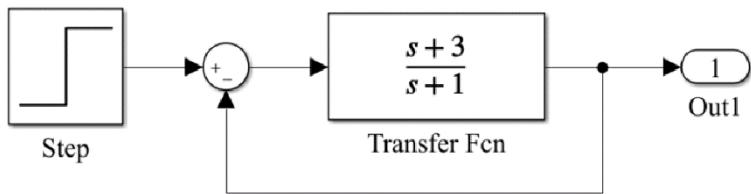
Hiện tượng vòng lặp đại số có nghĩa là trong một mô hình nhất định, tín hiệu đầu vào của một hoặc nhiều khối phụ thuộc vào tín hiệu đầu ra của chính chúng. Vì vậy, trong mỗi bước mô phỏng, các phương trình đại số cần được giải một lần. Do việc giải phương trình đại số khá tốn thời gian nên khôi lượng tính toán có thể tăng lên nếu có các vòng lặp đại số trong mô hình. Vì vậy, cần loại trừ hiện tượng này.

Trong hộp thoại Configuration Parameters, vào tab Diagnostics có ba tùy chọn, mặc định là Warning. Nếu phát hiện vòng lặp đại số, thông báo cảnh báo có thể được đưa ra nhưng quá trình mô phỏng không bị chấm dứt. Nếu chọn error, quá trình mô phỏng sẽ kết thúc nếu phát hiện thấy vòng lặp đại số. Một thông báo lỗi được hiển thị. Nếu chọn none, các vòng lặp đại số sẽ bị bỏ qua.

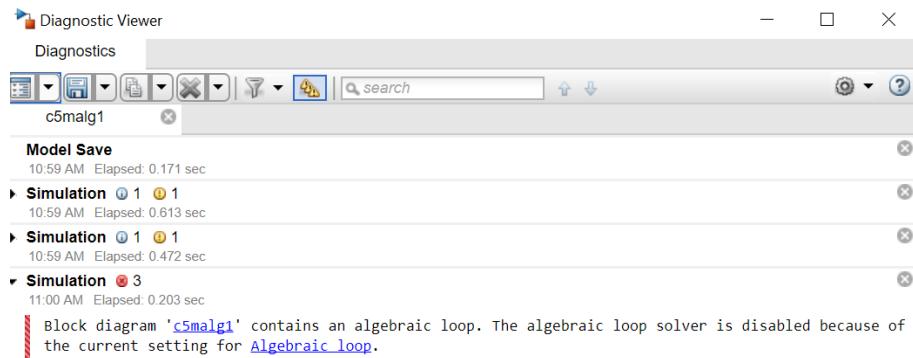
Bài toán 2.18: xem xét một hệ thống phản hồi tuyến tính đơn giản được hiển thị trong hình sau. Thiết lập mô hình SIMULINK để quan sát các vòng lặp đại số. Sử dụng phương pháp lý thuyết để loại bỏ vòng lặp đại số.



- B1: thiết lập mô hình trên SIMULINK như sau, lưu file là c5malg1.slx:



- B2: đặt tham số Diagnostics trong hộp thoại Configuration Parameters là error. Chạy chương trình, SIMULINK sẽ báo lỗi vòng lặp đại số:



Để xem chi tiết về lỗi, thực hiện lệnh sau trên MATLAB:

`Simulink.BlockDiagram.getAlgebraicLoops('c5malg1')`

Hoặc `sldebug('c5malg1')`

Để không hiển thị lỗi này, đặt tham số Diagnostics trong hộp thoại Configuration Parameters là none.

- B3: Sử dụng phương pháp lý thuyết để loại bỏ vòng lặp đại số. Trong thực tế, nếu sử dụng công thức hệ phản hồi âm để tìm mô hình tương đương thì mô hình hàm truyền tương đương có thể được suy ra trực tiếp như sau:

$$G_1(s) = \frac{G(s)}{1 + G(s)} = \frac{(s+3)/(s+1)}{1 + (s+3)/(s+1)} = \frac{s+3}{2s+4}$$

Lúc này mô hình $G_1(s)$ có thể loại bỏ hoàn toàn vòng lặp đại số. Mặc dù vậy, tính phù hợp của phương pháp này là không thể chấp nhận được, vì rất ít vấn đề có thể được đơn giản hóa theo cách này. Ví dụ, nếu có sự phi tuyến trong hệ thống thì phương pháp này không thể được sử dụng.

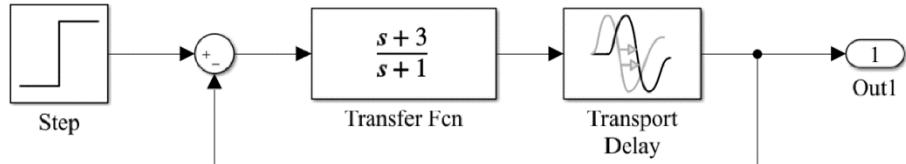
Lý do tại sao vòng lặp đại số tồn tại là vì một số tín hiệu đầu vào và đầu ra quan trọng có liên quan cùng một lúc. Nếu hai tín hiệu liên quan xảy ra ở những thời điểm khác nhau thì dự kiến sẽ tránh được vòng lặp đại số. Thực tế có 2 phương pháp:

- Phương pháp 1: Đưa vào thêm một khói Transport Delay và đưa tín hiệu đầu ra của nó vào đầu vào của khói khác. Nếu hằng số trễ đủ nhỏ thì có thể tránh được vòng lặp đại số;

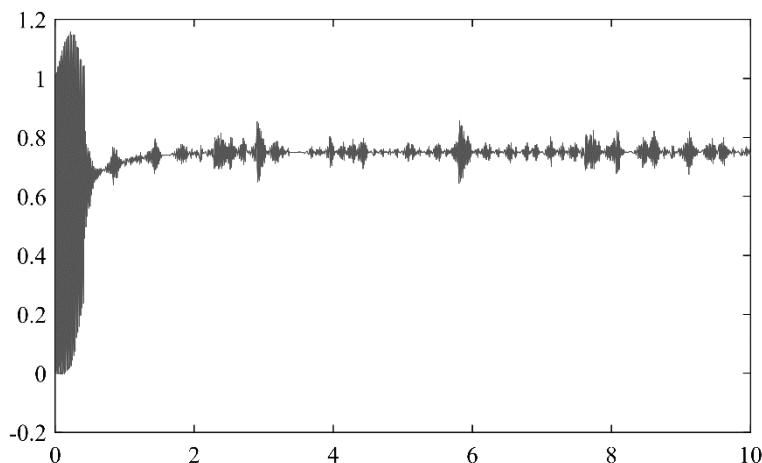
- Phương pháp 2: Đưa vào bộ lọc thông thấp, $1/(Ts + 1)$. Nếu T đủ nhỏ thì có thể tránh được các vòng lặp đại số.

Bài toán 2.19: đối với hệ thống ở Bài toán 2.18, sử dụng 2 phương pháp: đưa vào tham số trễ nhở và đưa vào bộ lọc thông thấp để loại bỏ vòng lặp đại số.

- B1: đưa vào khối Transport Delay và thiết lập mô hình SIMULINK như sau:



Thiết lập Time delay ở khối Transport Delay là 0.01s, ở chức năng Configuration Parameters đặt Solver là ode15s, Relative tolerance và Absolute tolerance là 10^{-8} . Kết quả thu được như sau:



- B2: đưa vào bộ lọc thông thấp để loại bỏ vòng lặp đại số. Từ mô hình hàm truyền đã cho, suy ra phương trình vi phân của hệ:

$$2y'(t) + 4y(t) = u'(t) + 3u(t), \quad y(0) = 0.5,$$

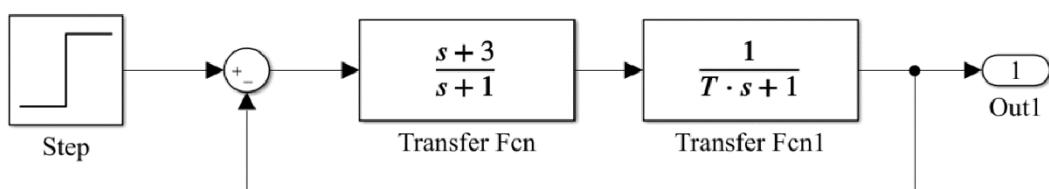
trong đó $u(t)$ là hàm step hoặc hàm Heaviside. Sử dụng các lệnh sau trên MATLAB để tìm nghiệm giải tích của phương trình vi phân:

```
>> syms t y(t); u=heaviside(t);
y0=dsolve(2*diff(y)+4*y==diff(u)+3*u, y(0)==0.5)
latex(simplify(y0))
```

thu được nghiệm:

$$y_0(t) = \frac{1}{8}e^{-2t} + \frac{3}{8}\text{sign}(t) - \frac{3}{8}e^{-2t}\text{sign}(t) + \frac{3}{8}.$$

Trên SIMULINK, thiết kế mô hình như sau, lưu file là c5malg2b.slx:

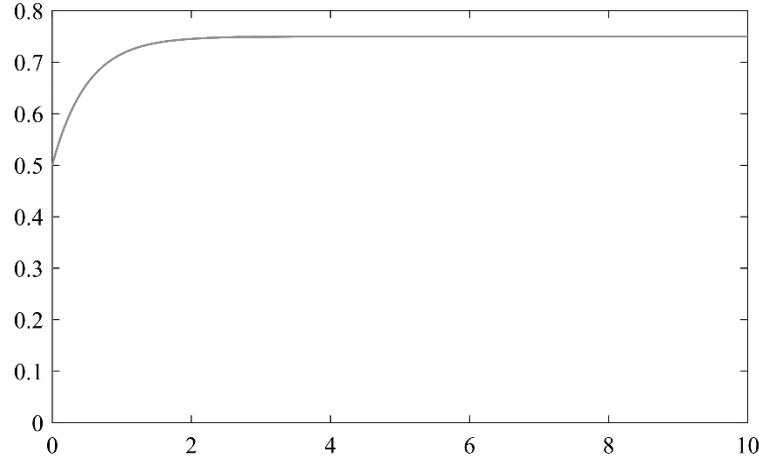


Chọn $T=0.001s$, thực hiện lệnh sau trên MATLAB để chạy chương trình:

```

>> T=0.001; tic, [t,~,y1]=sim('c5malg2b'); toc
y0=exp(-2*t)/8+3*sign(t)/8-3*exp(-2*t).*sign(t)/8+3/8;
plot(t,[y1 y0])

```



Trên thực tế, ở giai đoạn đầu, hai phương pháp có sự khác biệt nổi bật. Vì trong mô hình ban đầu có hiện tượng truyền thẳng trực tiếp nên tín hiệu đầu vào được phản xạ thành tín hiệu đầu ra ở thời điểm $t = 0$. Giá trị ban đầu của đầu ra là 0.5. Nếu sử dụng bộ lọc thông thấp, hiện tượng truyền thẳng trực tiếp sẽ bị trượt. Tín hiệu đầu ra được thay đổi từ 0 đến 0.5 trong một khoảng thời gian rất ngắn. Nếu T đủ nhỏ thì chu kì có thể được bỏ qua hoàn toàn. Ngoài khoảng thời gian ngắn, sai số tối đa là 4.6089×10^{-5} và số điểm là 6690.

```
>> ii=t>1e-5; max(abs(y(ii)-y1(ii))), length(t)
```

- B3: Giảm giá trị T , các mô phỏng được so sánh như trong bảng dưới đây.

| filter constant T | direct feedthrough | 0.001 | 0.0001 | 10^{-6} | 10^{-10} |
|---------------------------------------|---------------------------|-------------------------|-------------------------|-----------------------------|------------------------------|
| maximum error | 1.6875×10^{-14} | 4.6089×10^{-5} | 4.5995×10^{-6} | 4.5985×10^{-8} | 4.4973×10^{-12} |
| number of points | 823 | 6690 | 60675 | 1353 | 1128 |
| time elapse | 0.4133 | 0.1722 | 0.5211 | 0.1640 | 0.1961 |

Nếu không sử dụng bộ lọc thông thấp, mặc dù có vòng lặp đại số nhưng sẽ mang lại kết quả chính xác nhất và thời gian xử lý lớn hơn. Trong trường hợp bình thường, các vòng lặp đại số có thể được bỏ qua và để SIMULINK tự động giải các phương trình đại số. Trong một số trường hợp nhất định, khi giải pháp SIMULINK gặp vấn đề, các bộ lọc thông thấp có thể được đưa vào để loại bỏ các vòng lặp đại số. Có thể thấy rằng, nếu T đủ nhỏ thì kết quả xấp xỉ là khá thỏa đáng.

2.4. Câu hỏi thực hành

Tìm hiểu Control System Toolbox trên phần mềm MATLAB.

TÀI LIỆU THAM KHẢO

- [1] Hoàng Minh Sơn, Cơ sở điều khiển quá trình, Nhà xuất bản Bách khoa Hà Nội, 2009.
- [2] Nguyễn Văn Chí, giáo trình điều khiển các quá trình công nghệ, Nhà xuất bản Khoa học kỹ thuật, 2016.
- [3] Nguyễn Phùng Quang, MATLAB&SIMULINK dành cho kỹ sư điều khiển tự động, Nhà xuất bản Khoa học kỹ thuật, 2003.

Bài 3. Mô hình hoá lý thuyết

3.1. Mục tiêu

Sau khi hoàn thành bài thực hành này, sinh viên có khả năng sử dụng được các công cụ trong phần mềm MATLAB để mô hình hoá lý thuyết các quá trình cơ bản.

3.2. Nội dung thực hành

Thực hiện mô hình hoá lý thuyết một số quá trình cơ bản như sau:

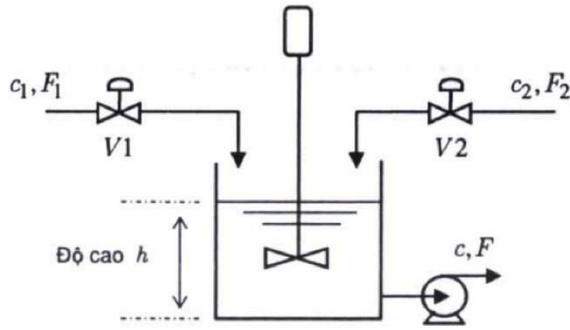
- Mô hình hoá hệ thống trộn;
- Tuyến tính hoá mô hình phi tuyến;
- Mô hình quá trình khuấy trộn chất lỏng đẳng nhiệt;
- Mô hình quá trình khuấy trộn chất lỏng không đẳng nhiệt;
- Mô hình quá trình gia nhiệt có khuấy trộn;
- Mô hình quá trình của bồn phản ứng khuấy trộn liên tục;
- Mô hình quá trình mức chất lỏng;
- Mô hình quá trình trao đổi nhiệt chất lỏng;
- Mô hình quá trình của bộ trao đổi nhiệt bằng hơi;
- Mô hình quá trình của bộ trao đổi nhiệt có trễ;
- Mô hình quá trình phản ứng 2 pha;
- Mô hình quá trình khí nén;
- Mô hình quá trình thuỷ lực.

3.3. Các bước thực hành

3.3.1 Mô hình hoá hệ thống trộn

Bài toán 3.1

a) Cho một quá trình trộn như hình vẽ sau, hãy xác định mô hình của quá trình, xác định bậc tự do của hệ thống, số biến điều khiển và số biến nhiễu. Mô phỏng hệ thống trên phần mềm MATLAB.



Hình 3.1. Mô hình quá trình trộn

Biến quá trình gồm có 2 biến ra (h, c) và 5 biến vào gồm (F, F_1, F_2, c_1, c_2). Giả thiết bỏ qua trễ vận chuyển và quan tính của quá trình khuấy trộn và coi khối lượng riêng bằng hằng số, ta có:

Phương trình cân bằng khối lượng:

$$\frac{d(\rho V)}{dt} = F_1 + F_2 - F \quad (3.1)$$

Phương trình cân bằng thành phần:

$$\frac{d(\rho V_c)}{dt} = F_1 c_1 + F_2 c_2 - F c \quad (3.2)$$

$$\rho V \frac{dc}{dt} + \rho c \frac{dV}{dt} = F_1 c_1 + F_2 c_2 - F c \quad (3.3)$$

Thay thế $V=Ah$, biến đổi ta có:

$$\begin{aligned} \frac{dh}{dt} &= \frac{1}{\rho A} (F_1 + F_2 - F) \\ \frac{dc}{dt} &= \frac{1}{\rho Ah} (F_1 c_1 + F_2 c_2 - (F_1 + F_2)c) \end{aligned}$$

Như vậy số phương trình là 2, số biến quá trình là 7 nên số bậc tự do = 7-2 = 5. Số biến điều khiển là 2 (F_1, F_2), số biến nhiễu là 3 (tỉ lệ khối lượng thành phần A vào (c_1, c_2) và lượng ra F). Đặt vector đầu ra của hệ:

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} h \\ c \end{bmatrix}$$

Hệ phương trình vi phân mô tả quá trình chuyển thành:

$$\frac{dy}{dt} = \frac{d}{dt} \begin{bmatrix} h \\ c \end{bmatrix} = f(y, F_1, F_2, F, c_1, c_2) = \begin{bmatrix} \frac{1}{\rho A} (F_1 + F_2 - F) \\ \frac{1}{\rho A y_1} (F_1 c_1 + F_2 c_2 - (F_1 + F_2)c) \end{bmatrix} \quad (3.4)$$

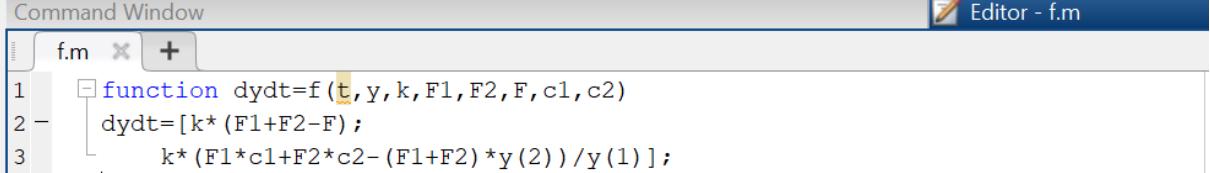
Giả thiết tại điểm làm việc cân bằng có: $\bar{F}_2 = 220$ kg/phút, $\bar{c}_1 = 0.75$, $\bar{c}_2 = 0.75$, $\bar{c} = 0.4$, $\bar{h} = 1$ m, $\frac{1}{\rho A} = 0.001$ (m/kg). Từ đây xác định được giá trị \bar{F} , \bar{F}_1 tại điểm làm việc cân bằng thông qua giải phương trình sau:

$$\begin{cases} \bar{F}_1 + \bar{F}_2 - \bar{F} = 0 \\ \bar{F}_1 \bar{c}_1 + \bar{F}_2 \bar{c}_2 - (\bar{F}_1 + \bar{F}_2) \bar{c} = 0 \end{cases} \quad (3.5)$$

Ta nhận được: $\bar{F} = 314.2857$ kg/phút, $\bar{F}_1 = 94.2857$ kg/phút.

Để mô phỏng quá trình trên MATLAB, cần thực hiện các bước sau:

- B1: vào cửa sổ Editor của MATLAB, tạo hàm f.m như sau:

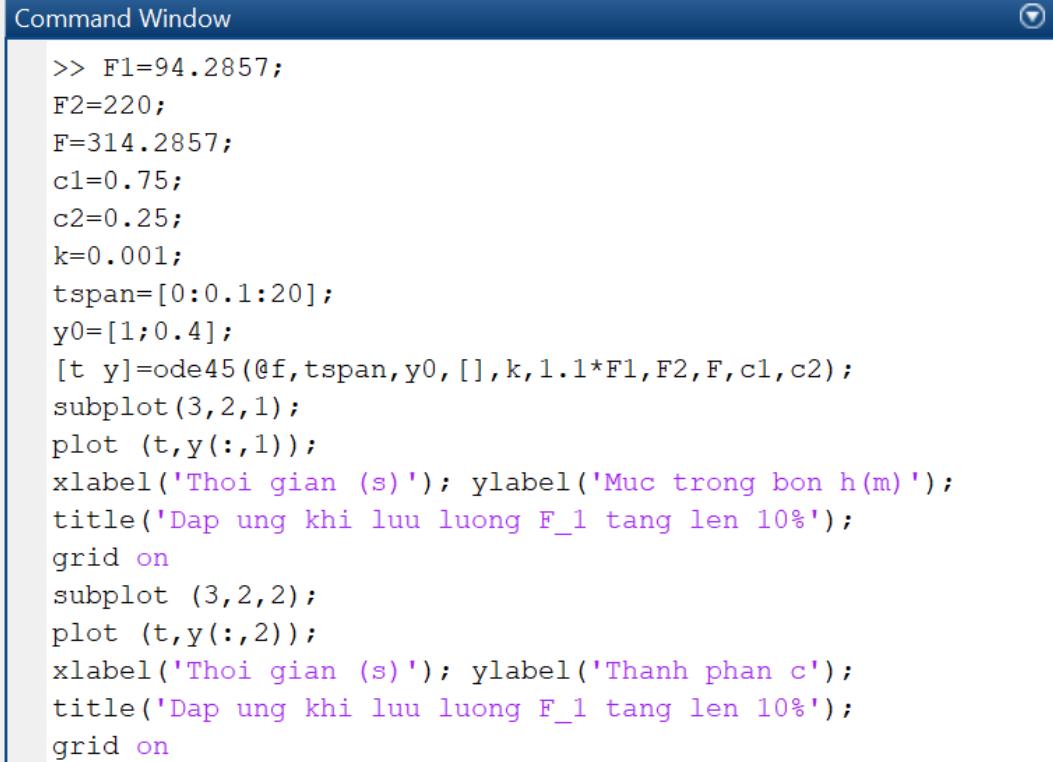


```

Command Window
Editor - f.m
f.m
1 function dydt=f(t,y,k,F1,F2,F,c1,c2)
2 dydt=[k*(F1+F2-F);
3 . . . k*(F1*c1+F2*c2-(F1+F2)*y(2))/y(1)];

```

- B2: Trên cửa sổ Command Window của MATLAB, thực hiện đoạn code sau:



```

Command Window
>> F1=94.2857;
F2=220;
F=314.2857;
c1=0.75;
c2=0.25;
k=0.001;
tspan=[0:0.1:20];
y0=[1;0.4];
[t y]=ode45(@(f,tspan,y0,[],k,1.1*F1,F2,F,c1,c2));
subplot(3,2,1);
plot (t,y(:,1));
xlabel('Thoi gian (s)'); ylabel('Muc trong bon h(m)');
title('Dap ung khi luu luong F_1 tang len 10%');
grid on
subplot (3,2,2);
plot (t,y(:,2));
xlabel('Thoi gian (s)'); ylabel('Thanh phan c');
title('Dap ung khi luu luong F_1 tang len 10%');
grid on

```

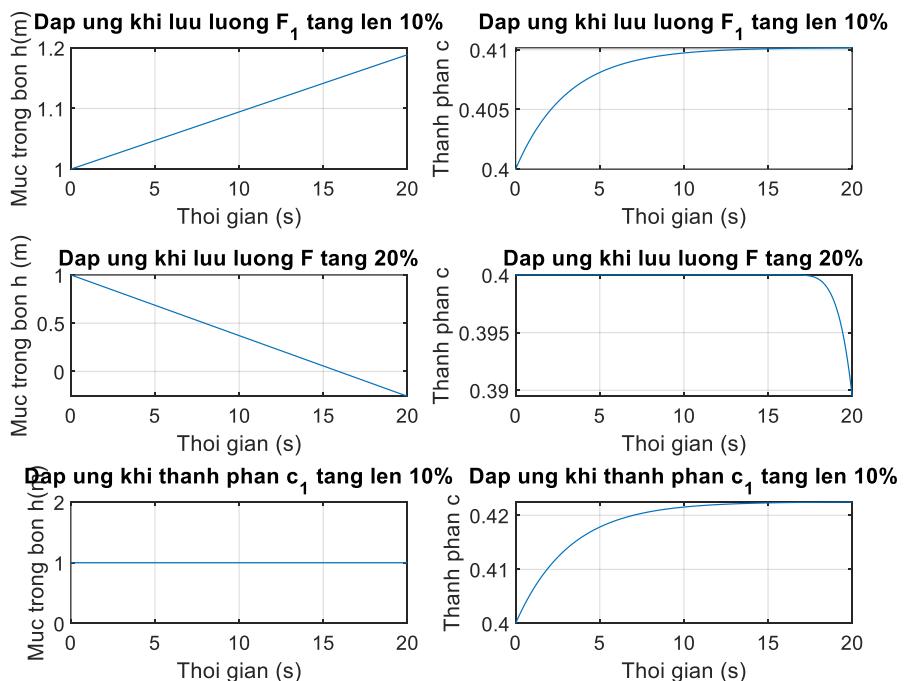
```

[t y]=ode45(@f,tspan,y0,[],k,F1,F2,1.2*F,c1,c2);
subplot(3,2,3);
plot(t,y(:,1));
xlabel('Thoi gian (s)'); ylabel('Muc trong bon h (m)');
title('Dap ung khi luu luong F tang 20%');
grid on
subplot (3,2,4);

plot(t,y(:,2));
xlabel('Thoi gian (s)'); ylabel('Thanh phan c');
title('Dap ung khi luu luong F tang 20%');
grid on
[t y]=ode45(@f,tspan,y0,[],k,F1,F2,F,1.1*c1,c2);
subplot(3,2,5);
plot(t,y(:,1));
xlabel('Thoi gian (s)'); ylabel('Muc trong bon h(m)');
title('Dap ung khi thanh phan c_1 tang len 10%');
grid on
subplot (3,2,6);
plot(t,y(:,2));
xlabel('Thoi gian (s)'); ylabel('Thanh phan c');
title('Dap ung khi thanh phan c_1 tang len 10%');
grid on

```

Kết quả:



*) Câu hỏi: khi thay đổi thành phần F_1 , ảnh hưởng đến mức trong bồn h và thành phần c như thế nào? khi thay đổi thành phần F , ảnh hưởng đến mức trong bồn h và thành phần c như thế nào? khi thay đổi thành phần c_1 , ảnh hưởng đến mức trong bồn h và thành phần c như thế nào?

b) Với mô hình quá trình trộn như trên, xây dựng mô hình hàm truyền cho quá trình. Mô phỏng quá trình thông qua hàm truyền trên MATLAB.

Từ phương trình thứ nhất của hệ (3.4), biểu diễn lại dưới dạng các chênh lệch so với điểm làm việc cân bằng của 2 biến trạng thái h và C như sau:

$$\Delta \dot{h} = \frac{1}{\rho A} (\Delta F_1 + \Delta F_2 - \Delta F)$$

Biến đổi Laplace ta được hàm truyền biểu diễn thay đổi của h theo F_1 , F_2 và F :

$$\Delta H(s) = \frac{1}{\rho A s} (\Delta F_1(s) + \Delta F_2(s) - \Delta F(s))$$

Từ phương trình thứ 2 trong hệ phương trình (2.4), khai triển Taylor ta có:

$$\Delta \dot{C} = \frac{1}{\rho A h} (-\bar{F} \Delta C + (\bar{C}_1 - \bar{C}) \Delta F_1 + (\bar{C}_2 - \bar{C}) \Delta F_2 + \bar{F}_1 \Delta C_1 + \bar{F}_2 \Delta C_2)$$

Biến đổi Laplace hai vế ta được:

$$\begin{aligned} s \Delta C(s) &= \frac{1}{\rho A h} (-\bar{F} \Delta C(s) + (\bar{C}_1 - \bar{C}) \Delta F_1(s) + (\bar{C}_2 - \bar{C}) \Delta F_2(s) + \bar{F}_1 \Delta C_1(s) \\ &\quad + \bar{F}_2 \Delta C_2(s)) \\ \left(\frac{\rho A \bar{h}}{\bar{F}} s + 1 \right) \Delta C(s) &= \frac{(\bar{C}_1 - \bar{C})}{\bar{F}} \Delta F_1(s) + \frac{(\bar{C}_2 - \bar{C})}{\bar{F}} \Delta F_2(s) + \frac{\bar{F}_1}{\bar{F}} \Delta C_1(s) + \frac{\bar{F}_2}{\bar{F}} \Delta C_2(s) \end{aligned}$$

Đặt các tham số:

$$\tau = \frac{\rho A \bar{h}}{\bar{F}}, k_{f1} = \frac{(\bar{C}_1 - \bar{C})}{\bar{F}}, k_{f2} = \frac{(\bar{C}_2 - \bar{C})}{\bar{F}}, k_{c1} = \frac{\bar{F}_1}{\bar{F}}, k_{c2} = \frac{\bar{F}_2}{\bar{F}}$$

Ta được mô hình hàm truyền cho đáp ứng đầu ra thành phần C của hệ:

$$\Delta C(s) = \frac{k_{f1}}{\tau s + 1} \Delta F_1(s) + \frac{k_{f2}}{\tau s + 1} \Delta F_2(s) + \frac{k_{c1}}{\tau s + 1} \Delta C_1(s) + \frac{k_{c2}}{\tau s + 1} \Delta C_2(s)$$

Từ các biến đổi trên, ta xác định được các hàm truyền đạt của hệ từ các đầu vào đến các đầu ra của hệ là:

$$\begin{aligned} G_{f1h}(s) &= \frac{\Delta H(s)}{\Delta F_1(s)} = \frac{1}{\rho A s} = \frac{k}{s} = \frac{0.001}{s} \text{ (kg/phút)} \\ G_{f2h}(s) &= \frac{\Delta H(s)}{\Delta F_2(s)} = G_{f1h}(s), G_{fh}(s) = \frac{\Delta H(s)}{\Delta F(s)} = -G_{f1h}(s) \\ G_{c1h}(s) &= \frac{\Delta H(s)}{\Delta C_1(s)} = 0, G_{c2h}(s) = \frac{\Delta H(s)}{\Delta C_2(s)} = 0 \\ G_{f1c}(s) &= \frac{\Delta C(s)}{\Delta F_1(s)} = \frac{k_{f1}}{\tau s + 1}, G_{f2c}(s) = \frac{\Delta C(s)}{\Delta F_2(s)} = \frac{k_{f2}}{\tau s + 1}, G_{fc}(s) = \frac{\Delta C(s)}{\Delta F(s)} \\ G_{c1c}(s) &= \frac{\Delta C(s)}{\Delta C_1(s)} = \frac{k_{c1}}{\tau s + 1}, G_{c2c}(s) = \frac{\Delta C(s)}{\Delta C_2(s)} = \frac{k_{c2}}{\tau s + 1} \end{aligned}$$

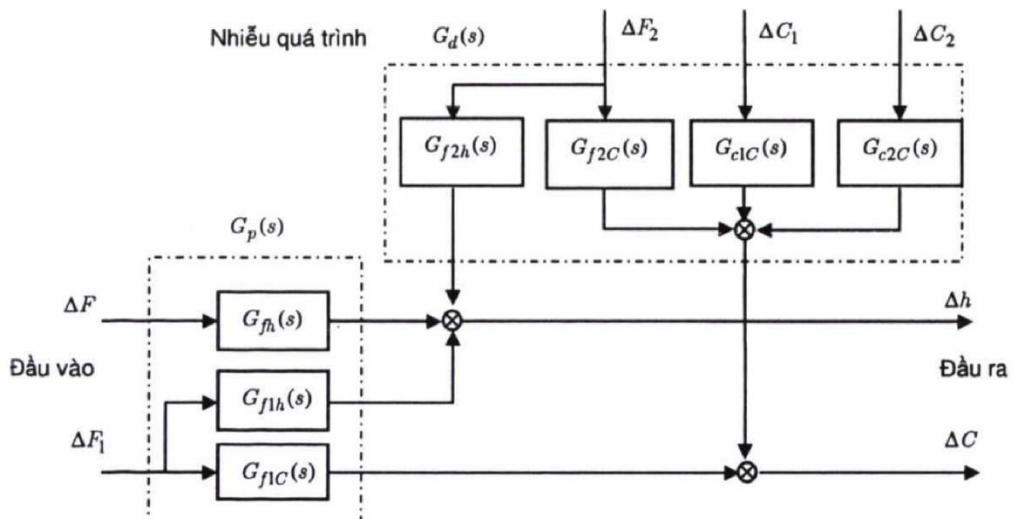
Biểu diễn dưới dạng ma trận hàm truyền đạt, ta có:

$$\underbrace{\begin{bmatrix} \Delta H(s) \\ \Delta C(s) \end{bmatrix}}_{Y(s)} = \underbrace{\begin{bmatrix} G_{f1h}(s) & G_{f2h}(s) & G_{fh}(s) & G_{c1h}(s) & G_{c2h}(s) \\ G_{f1C}(s) & G_{f2C}(s) & G_{fC}(s) & G_{c1C}(s) & G_{c2C}(s) \end{bmatrix}}_{Y(s)} \begin{bmatrix} \Delta F_1(s) \\ \Delta F_2(s) \\ \Delta F(s) \\ \Delta C_1(s) \\ \Delta C_2(s) \end{bmatrix}$$

Nếu coi các đầu vào ΔF_2 , ΔC_1 , ΔC_2 là các đầu vào nhiễu; ΔF , ΔF_1 là các biến điều khiển, ta có:

$$\underbrace{\begin{bmatrix} \Delta H(s) \\ \Delta C(s) \end{bmatrix}}_{Y(s)} = \underbrace{\begin{bmatrix} G_{fh}(s) & G_{f1h}(s) \\ G_{fC}(s) & G_{f1C}(s) \end{bmatrix}}_{G_p(s)} \underbrace{\begin{bmatrix} \Delta F(s) \\ \Delta F_1(s) \end{bmatrix}}_{U(s)} + \underbrace{\begin{bmatrix} G_{f2h}(s) & G_{c1h}(s) & G_{c2h}(s) \\ G_{f2C}(s) & G_{c1C}(s) & G_{c2C}(s) \end{bmatrix}}_{G_d(s)} \underbrace{\begin{bmatrix} \Delta F_2(s) \\ \Delta C_1(s) \\ \Delta C_2(s) \end{bmatrix}}_{d(s)}$$

$$Y(s) = G_p(s)U(s) + G_d(s)d(s) \quad (3.6)$$



Hình 3.2. Sơ đồ khối của mô hình quá trình trộn

Thực hiện đoạn code trên giao diện Command Window của MATLAB để mô phỏng quá trình trộn theo hàm truyền như sau:

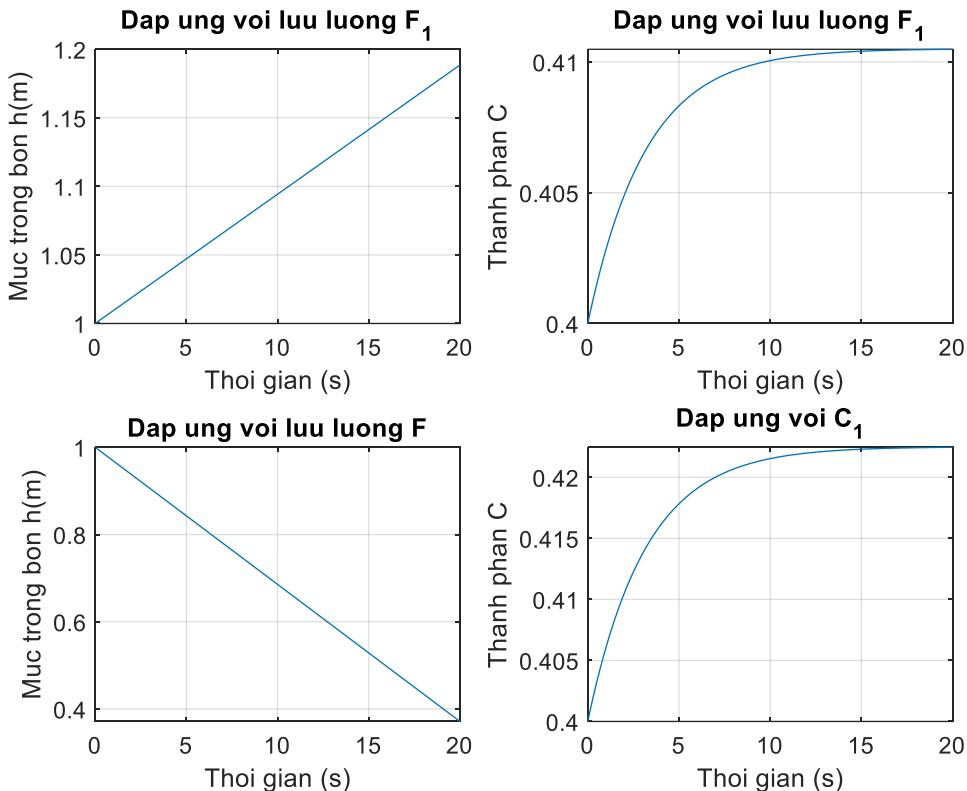
```
Command Window
>> F1=94.2857;F2=220;F=314.2857;
C1=0.75;C2=0.25;C=0.4;h=1;k=0.001;
tau=(1/k)*h/F; kf1=(C1-C)/F; kf2=(C2-C)/F;
kc1=F1/F; kc2=F2/F;s=tf('s');
G_f1h=k/s;
G_f2h=k/s;
G_fh=-k/s;
G_f1C=kc1/(tau*s+1);
G_f2C=kc2/(tau*s+1);
G_c1C=kc1/(tau*s+1);
G_c2C=kc2/(tau*s+1);
t=[0:0.1:20];
y0=[1;0.4];
y=step([G_f1h G_f1C],t)*F1*0.1;
```

```

    subplot(2,2,1);
    plot(t,y(:,1)+y0(1));
    xlabel('Thoi gian (s)');ylabel('Muc trong bon h(m)');
    title('Dap ung voi luu luong F_1'); grid on
    subplot(2,2,2);
    plot(t,y(:,2)+y0(2));
    xlabel('Thoi gian (s)');ylabel('Thanh phan C');
    title('Dap ung voi luu luong F_1'); grid on
    y=step(G_fh,t)*F*0.1;
    subplot(2,2,3);
    plot(t,y(:,1)+y0(1));
    xlabel('Thoi gian (s)');ylabel('Muc trong bon h(m)');
    title('Dap ung voi luu luong F'); grid on
    y=step(G_c1C,t) *C1*0.1;
    subplot(2,2,4);
    plot(t,y(:,1)+y0(2));
    xlabel('Thoi gian (s)');ylabel('Thanh phan C');
    title('Dap ung voi C_1'); grid on

```

Kết quả:



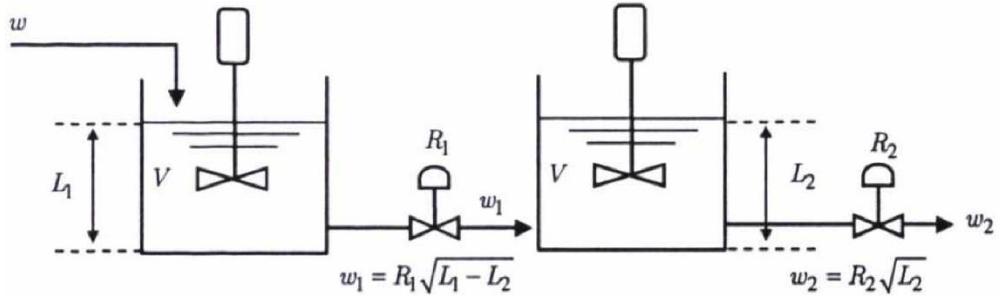
3.3.2 Tuyến tính hoá mô hình phi tuyến

Tóm tắt: sử dụng các hàm tuyến tính hoá mô hình phi tuyến đã giới thiệu ở Bài thực hành số 2. Ví dụ, sử dụng hàm linmod, cú pháp:

`[A, B, C, D]=linmod ('SIMULINKfilename', x_op, u0)`

Bài toán 3.2

Cho quá trình gồm 2 bồn chứa tương tác như sau:



Hình 3.3. Quá trình 2 bồn chứa tương tác

Mô hình trạng thái biểu diễn động học của hệ thống:

$$\frac{dL_1}{dt} = \frac{\omega}{A_1} - \frac{R_1}{A_1} \sqrt{L_1 - L_2} = f_1(L_1, L_2, \omega) \quad (3.7)$$

$$\frac{dL_2}{dt} = \frac{R_1}{A_2} \sqrt{L_1 - L_2} - \frac{R_2}{A_2} \sqrt{L_2} = f_2(L_1, L_2, \omega) \quad (3.8)$$

Gọi đầu ra là mức ở bồn chứa số 2, đặt biến trạng thái $L = [L_1 \ L_2]^T$, đầu ra $y = L_2$, đầu vào $u = \omega$. Giả sử điểm làm việc $\underline{L}^0 = [L_1^0 \ L_2^0]^T$, $u^0 = \omega^0$. Mô hình quá trình tuyến tính hoá xung quanh điểm làm việc $\underline{L}^0 = [L_1^0 \ L_2^0]^T$, $u^0 = \omega^0$ là:

$$\begin{cases} \Delta \dot{L} = A \Delta L + B \Delta u \\ \Delta y = C \Delta L \end{cases} \quad (3.9)$$

Cho các tham số: $A_1 = 2.5 \text{ m}^2$, $A_2 = 2.3 \text{ m}^2$, $R_1 = 0.6$, $R_2 = 0.5$, $L_2^0 = 1.2$, để xác định ω^0 , L_1^0 ta giải hệ phương trình xác lập sau:

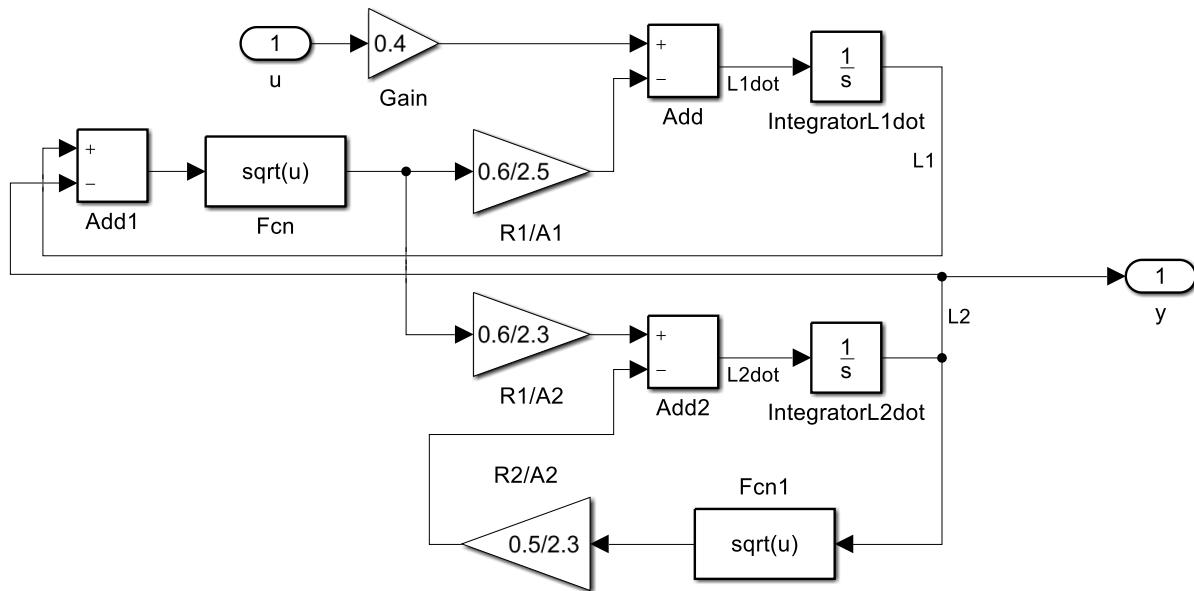
$$\begin{cases} 0 = \frac{\omega^0}{A_1} - \frac{R_1}{A_1} \sqrt{L_1^0 - L_2^0} \\ 0 = \frac{R_1}{A_2} \sqrt{L_1^0 - L_2^0} - \frac{R_2}{A_2} \sqrt{L_2^0} \end{cases}$$

Nhận được:

$$\begin{aligned} \omega^0 &= 0.5477 \frac{m^3}{h}, L_1^0 = 2.0333 \text{ m}. Vậy L^0 = [L_1^0 \ L_2^0]^T = [2.0333 \ 1.2], u^0 = \omega^0 \\ &= 0.5477. \end{aligned}$$

Để tuyến tính hóa quá trình phi tuyến này, chúng ta thực hiện các bước sau:

- B1: mô phỏng SIMULINK cho hệ phi tuyến trên, ghi lại, đặt tên file là twotank:



- B2: Thực hiện đoạn code trên giao diện Command Window của MATLAB để xem MATLAB định nghĩa các thông số của quá trình, thứ tự các biến trạng thái:

```
Command Window
>> [size,x0,states]=twotank([],[],[],0)

size =
2
0
1
1
0
1
1

x0 =
0
0

states =
2×1 cell array
{'twotank/IntegratorL2dot'}
{'twotank/IntegratorL1dot'}
```

Biến `sizes` chỉ ra rằng theo thứ tự từ trên xuống: số biến trạng thái của mô hình là 2, số biến trạng thái rác là 0 (vì mô phỏng cho hệ liên tục), số đầu ra là 1, số đầu vào là 1, hai giá trị cuối luôn nhận giá trị 1. Biến `x0` chỉ ra rằng vectơ giá trị đầu của các biến trạng thái là 0 và 0. Biến `states` chỉ ra thứ tự biến trạng thái trong mô hình được sắp xếp là: L2 L1, như vậy thứ tự sắp xếp biến trạng thái của MATLAB khác với thứ tự sắp xếp biến trạng thái của mô hình từ đầu, chúng ta phải chú ý điều này để nhập vectơ điểm trạng thái làm việc cho đúng thứ tự và sau khi nhận được các

ma trận A, B, C, D do MATLAB tính ta cần phải hiệu chỉnh lại theo thứ tự biến trạng thái chúng ta mong muốn.

- B3: Dùng hàm `linmod` theo cú pháp sau đây để xác định các ma trận A, B, C, D: (hãy thử dùng các hàm khác đã giới thiệu ở Bài thực hành số 2)

```

Command Window
>> u0=0.5477;
L0=[1.2 2.0333]';
[A,B,C,D]=linmod('twotank',L0,u0)

A =
-0.2421    0.1429
 0.1315   -0.1315

B =
 0
 0.4000

C =
 1    0

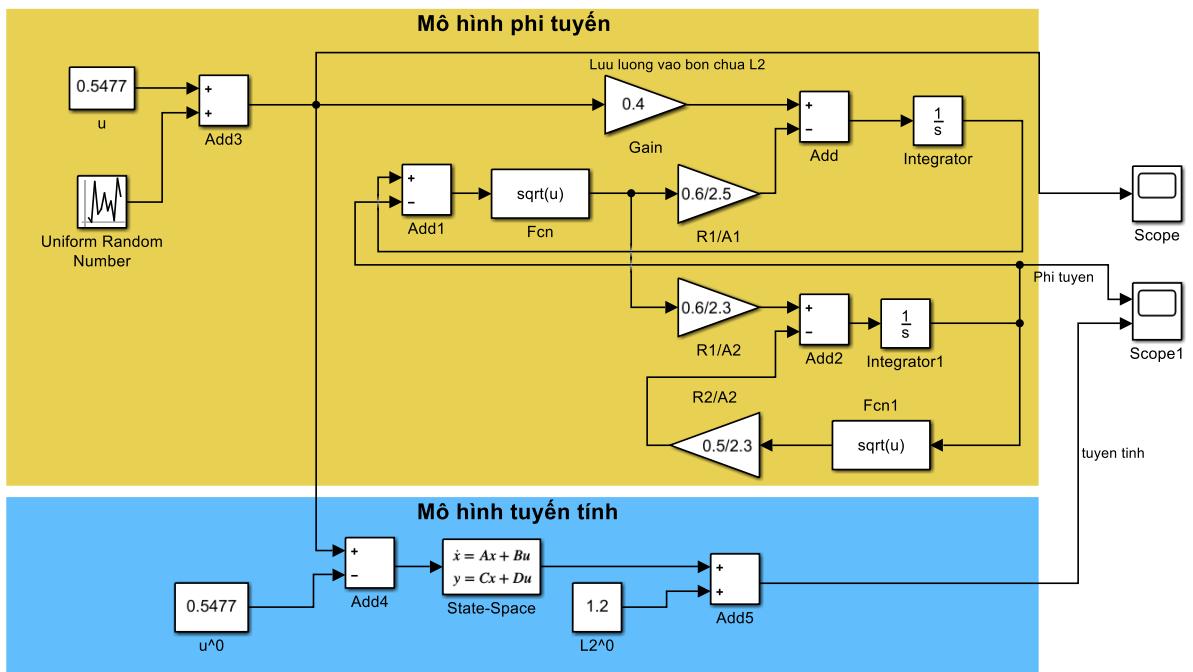
D =
 0

```

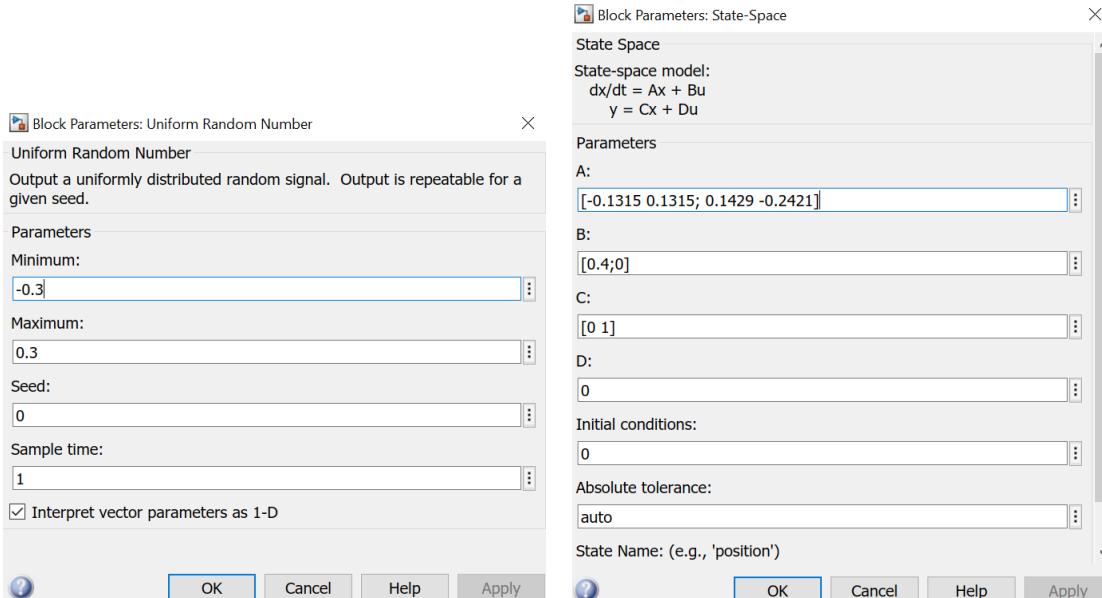
Do thứ tự biến trạng thái do Matlab định nghĩa là L2 L1 khác với ở mô hình đưa vào là L1 L2, cho nên ta đổi lại thứ tự hàng cột các ma trận vừa nhận được và thu được các ma trận A, B, C, D cần xác định như sau:

$$A = \begin{bmatrix} -0.1315 & 0.1315 \\ 0.1429 & -0.2421 \end{bmatrix}, B = \begin{bmatrix} 0.4 \\ 0 \end{bmatrix}, C = [0 \quad 1], D = 0$$

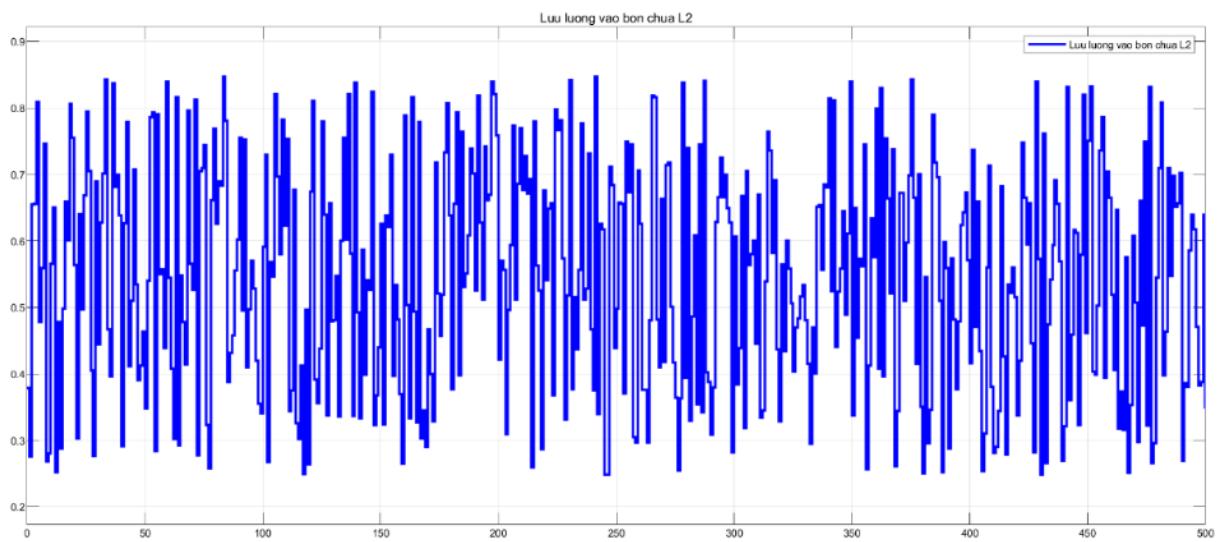
- B4: mô phỏng quá trình phi tuyến và tuyến tính của hệ để thuận tiện so sánh được thực hiện trên SIMULINK như sau:



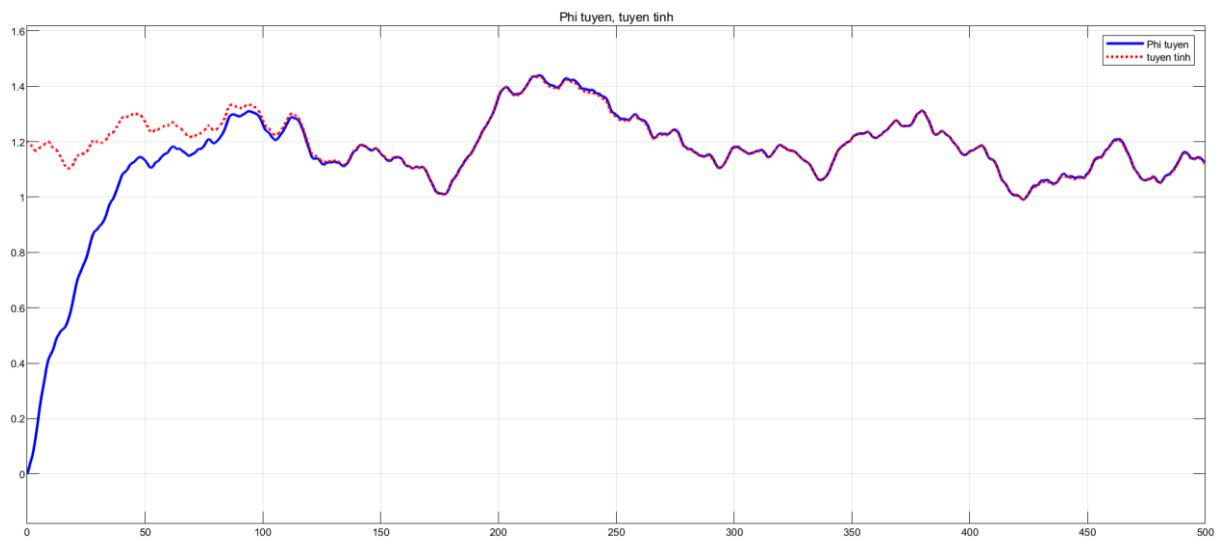
Lưu ý thiết lập tại khối Uniform Random Number và khối State-Space như sau:



Đặt thời gian mô phỏng là 500s. Kết quả trên Scope thể hiện lưu lượng vào bồn chứa 2 (L2):



Kết quả trên Scope1 thể hiện sự so sánh đáp ứng mức trong bồn chứa 2 với 2 mô hình tuyến tính và phi tuyến:



*) Câu hỏi: Tại sao trong khoảng 100s đầu, có sự khác biệt giữa 2 mô hình, sau đó thì hoàn toàn trùng khớp nhau?

Lưu ý: Ta cũng có thể chuyển mô hình tuyến tính trên không gian trạng thái sang mô hình tuyến tính dưới dạng hàm truyền bằng cách sử dụng lệnh ss2tf, cú pháp như sau:

```

Command Window
>> [num,den]=ss2tf(A,B,C,D,1)

num =
0         0      0.0572

den =
1.0000    0.3736   0.0130

```

Như vậy, hàm truyền là:

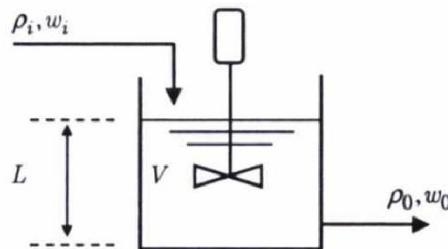
$$G(s) = \frac{\Delta L_2(s)}{\Delta \omega(s)} = \frac{0.0572}{s^2 + 0.3744s + 0.0131}$$

*) *Câu hỏi: Sử dụng hàm truyền này mô phỏng mô hình tuyến tính trên SIMULINK trong đó có so sánh với mô hình phi tuyến về đáp ứng mức trong bồn chứa 2.*

3.3.3 Mô hình quá trình khuấy trộn chất lỏng đang nhiệt

Bài toán 3.3

Cho mô hình quá trình khuấy trộn chất lỏng đang nhiệt như hình vẽ:



Hình 3.4. Quá trình khuấy trộn chất lỏng đang nhiệt

Chất lỏng đầu vào với lưu lượng $\omega_i(m^3/s)$ với khối lượng riêng ρ_i , chất lỏng đầu ra có lưu lượng $\omega_0(m^3/s)$ với khối lượng riêng ρ_0 , nhiệt độ trong bồn chứa xem như không đổi, quá trình khuấy là lí tưởng. Để xây dựng mô hình quá trình cần xuất bắt đầu từ các phương trình cân bằng. Tổng khối lượng đi vào là $\rho_i \omega_i$, tổng khối lượng đi ra là $\rho_0 \omega_0$, lượng vật chất được tích luỹ trong bồn chứa là $\frac{dm}{dt} = d(\rho V)/dt$. Do đó ta có phương trình cân bằng khối lượng:

$$\rho_i \omega_i = \rho_0 \omega_0 + \frac{d(\rho V)}{dt}$$

Quá trình khuấy là lí tưởng, nên giả thiết khối lượng riêng là đồng nhất: $\rho_0 = \rho_i = \rho$. Do đó:

$$A \frac{dL}{dt} = \omega_i - \omega_0 \quad (3.10)$$

với $V = AL$, A là diện tích cắt ngang của bồn chứa (m^2).

Để mô tả chính xác hệ thống, phương trình thứ 2 được xác định, lưu lượng ra là một hàm của mức trong bồn chứa:

$$\omega_0 = \alpha\sqrt{L} \quad (3.11)$$

Phân tích bậc tự do của mô hình:

- Tổng số biến mô tả quá trình là 3: ω_i , L , ω_0 ;
- Tham số của quá trình là A , V , ρ ;
- Số phương trình mô tả quá trình là 2 (3.10 và 3.11).

Vậy, bậc tự do của quá trình là $3-2=1$. Do đó chỉ cần dùng 1 mạch vòng điều khiển để ổn định mức L trong bồn chứa.

Với mô hình phi tuyến, chúng ta phải tuyến tính hóa mô hình quá trình tại điểm làm việc. Quá trình tuyến tính hóa thu được mô hình của hệ thống với mô tả như sau:

$$A \frac{d\Delta L}{dt} = \Delta\omega_i - \Delta\omega_0 \quad (3.12)$$

$$\Delta\omega_0 \approx \left(\frac{d}{dL} (\alpha\sqrt{L}) \Big|_{L=\bar{L}} \right) \Delta L = \frac{1}{2\sqrt{\bar{L}}} \alpha L \quad (3.13)$$

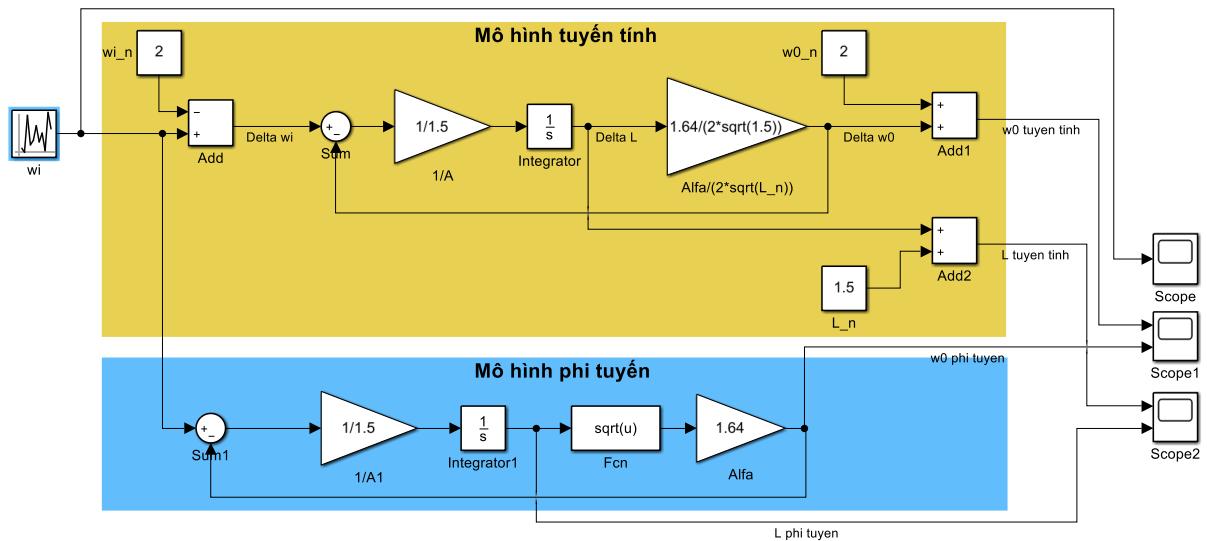
Hoặc chuyển sang dạng toán tử Laplace:

$$\Delta L(s) = \frac{1}{As} (\Delta\omega_i(s) - \Delta\omega_0(s)) \quad (3.14)$$

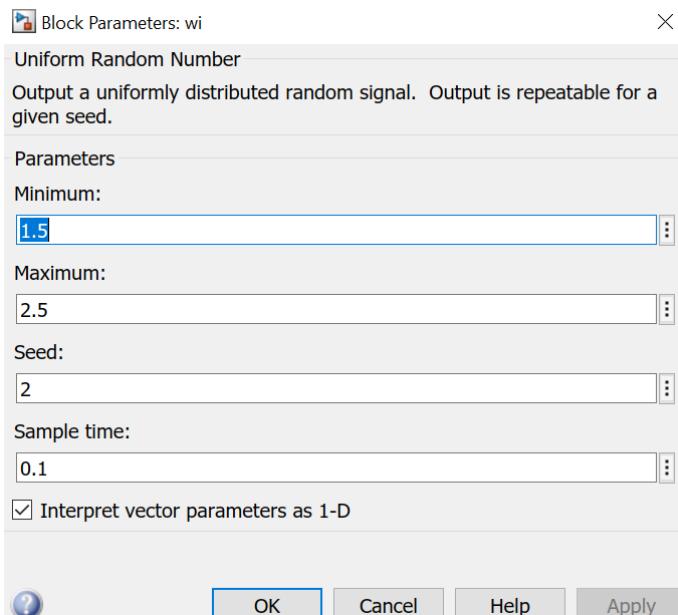
$$\Delta\omega(s) = \frac{1}{2\sqrt{\bar{L}}} \alpha \Delta L(s) \quad (3.15)$$

Với các tham số đầu vào như sau: $A = 1.5 m^2$, quá trình mức trong bồn chứa có giá trị tại điểm làm việc $\bar{L} = 1.5 m$, $\bar{\omega}_i = 2 m^3/s$, hệ số xả $\alpha = 1.64$. Tại điểm làm việc có $\bar{\omega}_i = \bar{\omega}_0$. Mô phỏng mô hình tuyến tính và phi tuyến của quá trình này trên SIMULINK để so sánh đáp ứng của hệ thống trong 2 trường hợp đó.

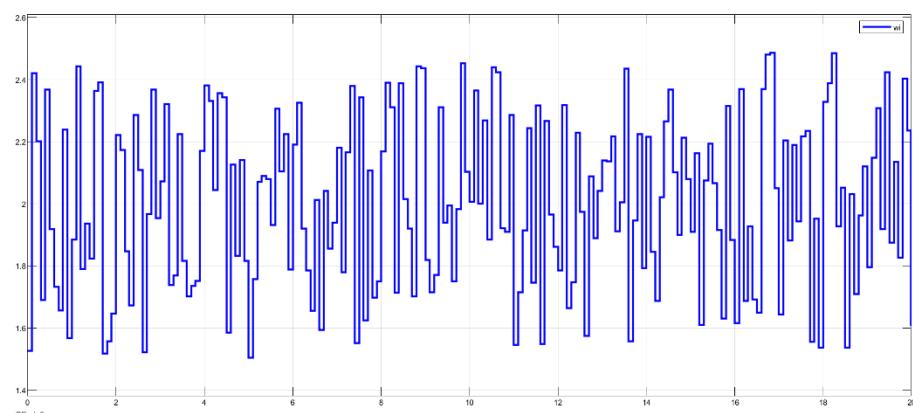
- B1: Xây dựng các mô hình trên SIMULINK như sau:



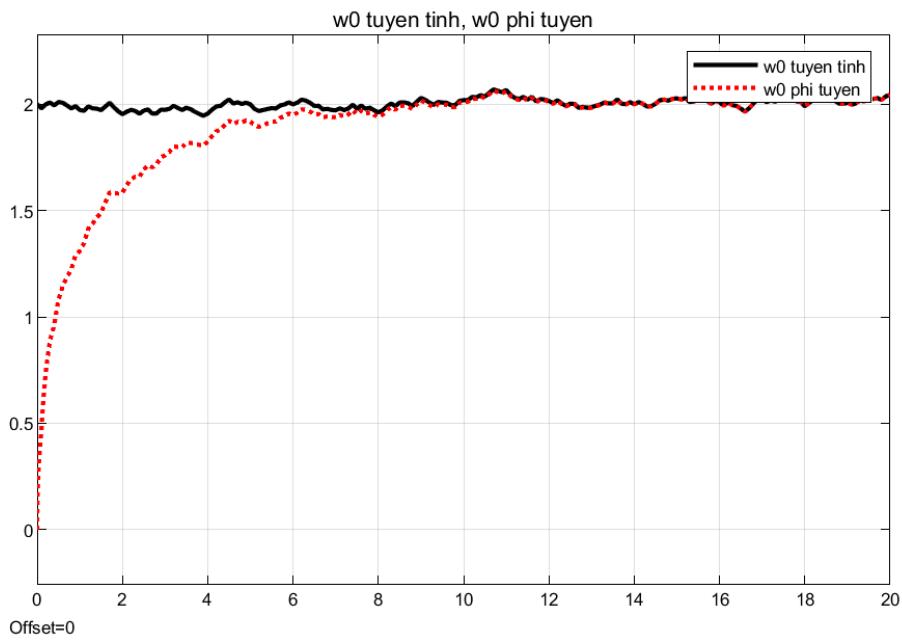
Đặt các thông số trong khối Uniform Random Number như sau:



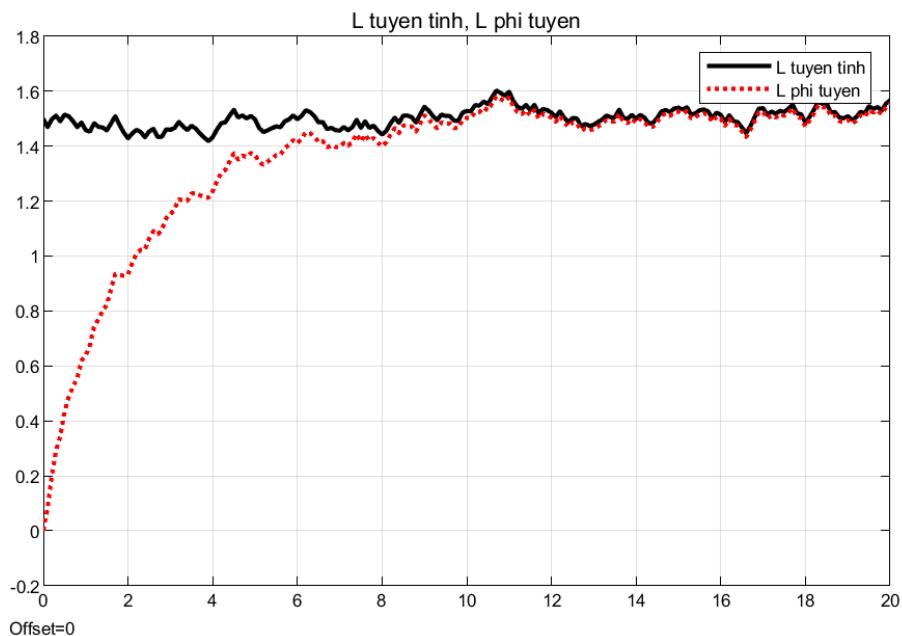
- B2: Đặt thời gian mô phỏng là 20s. Kết quả trên Scope cho thấy đáp ứng của lưu lượng vào $\omega_i (m^3/s)$:



Kết quả trên Scope 1 cho thấy đáp ứng lưu lượng ra $\omega_0 (m^3/s)$ trong 2 trường hợp: mô hình phi tuyến và tuyến tính:



Kết quả trên Scope 2 cho thấy đáp ứng mức trong bồn L trong 2 trường hợp: mô hình phi tuyến và tuyến tính:

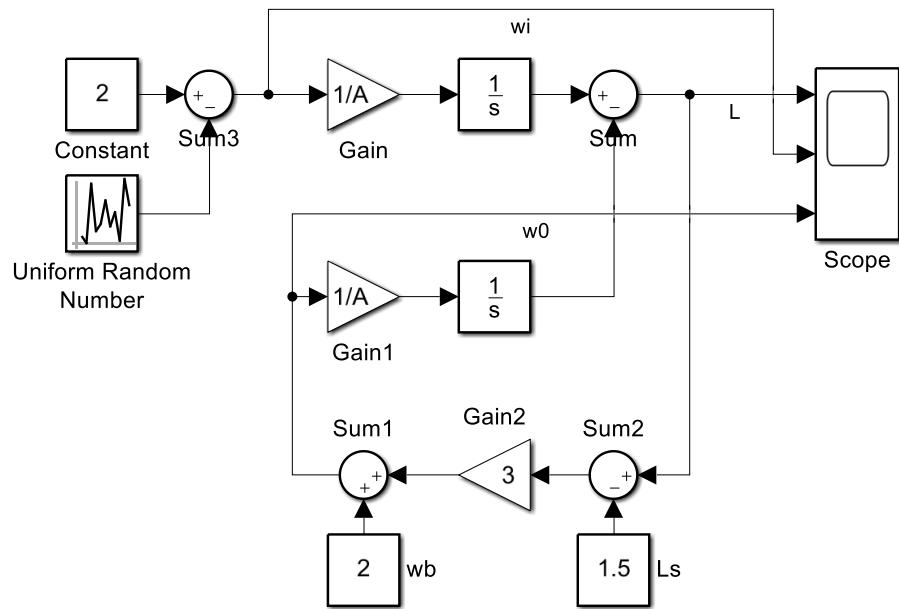


*) Câu hỏi: đối với đáp ứng lưu lượng ra ω_0 và mức trong bồn L , tại sao trong giai đoạn đầu (0-10s) là không trùng khớp nhau? Sau đó lại trùng khớp hoàn toàn?

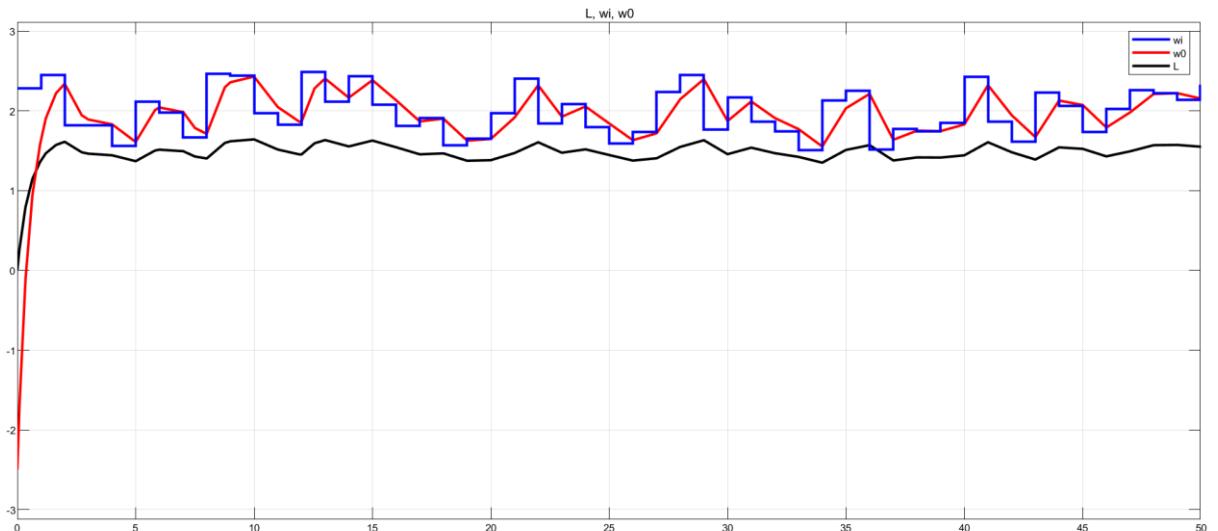
- B3: để điều khiển vòng kín, có thể sử dụng bộ điều khiển tỉ lệ, giả thiết giá trị lưu lượng mong muốn là L_s , biến điều khiển là lưu lượng ra có thể viết như sau:

$$\omega_0 = K_c(L - L_s) + \omega_b$$

với K_c là hệ số tỉ lệ, ω_b là lưu lượng nền có giá trị bằng ω_0 khi $L = L_s$. Sử dụng các thông số đã cho, ta có $K_c = 3$, $\omega_b = 2$. Thiết lập mô hình trên SIMULINK như sau:

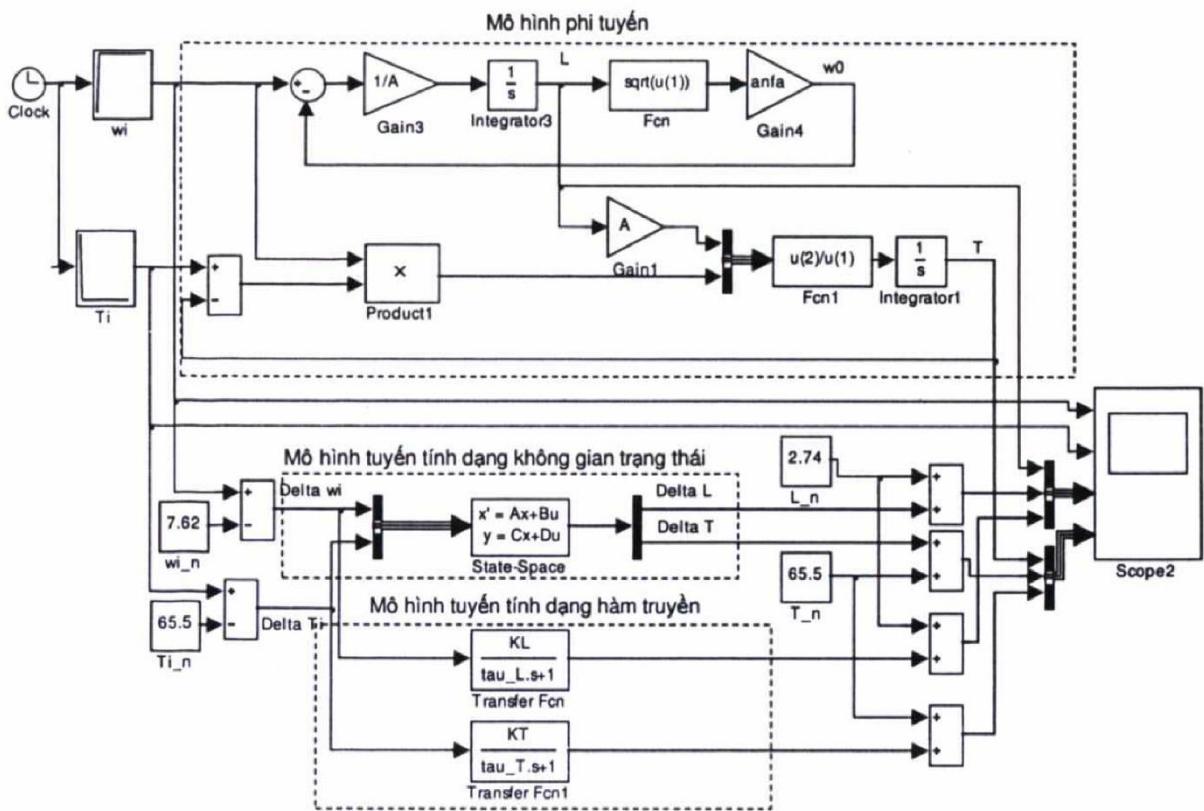


Chúng ta nhận được các đáp ứng như sau:

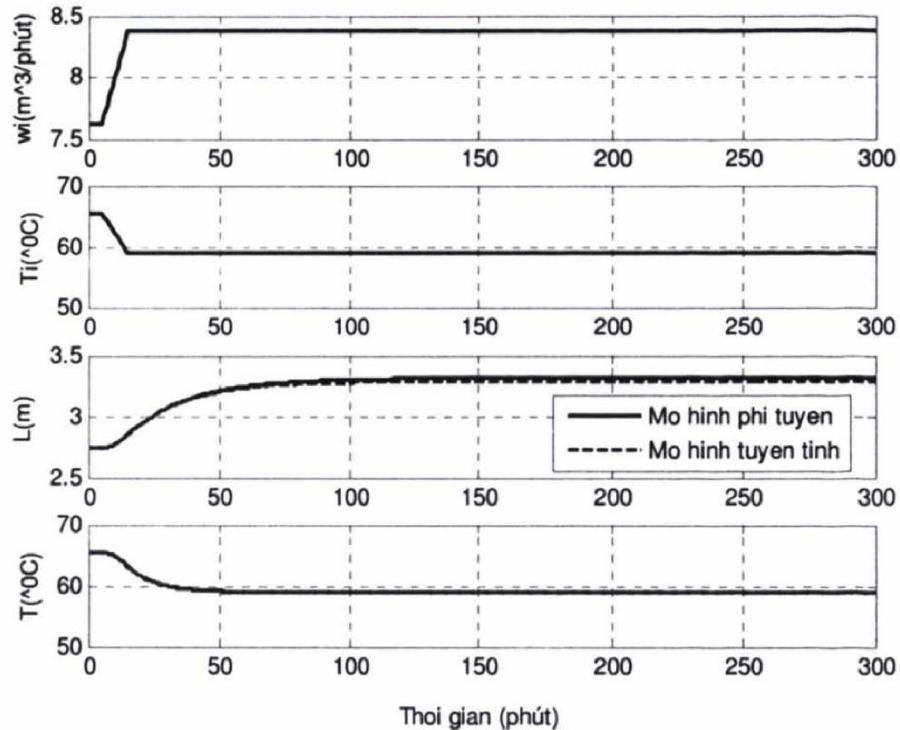


3.3.4 Mô hình quá trình khuấy trộn chất lỏng không đambia nhiệt

Bài toán 3.4: hãy mô hình hoá quá trình khuấy trộn không đambia nhiệt (mục 2.9.2, trang 73-76, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:

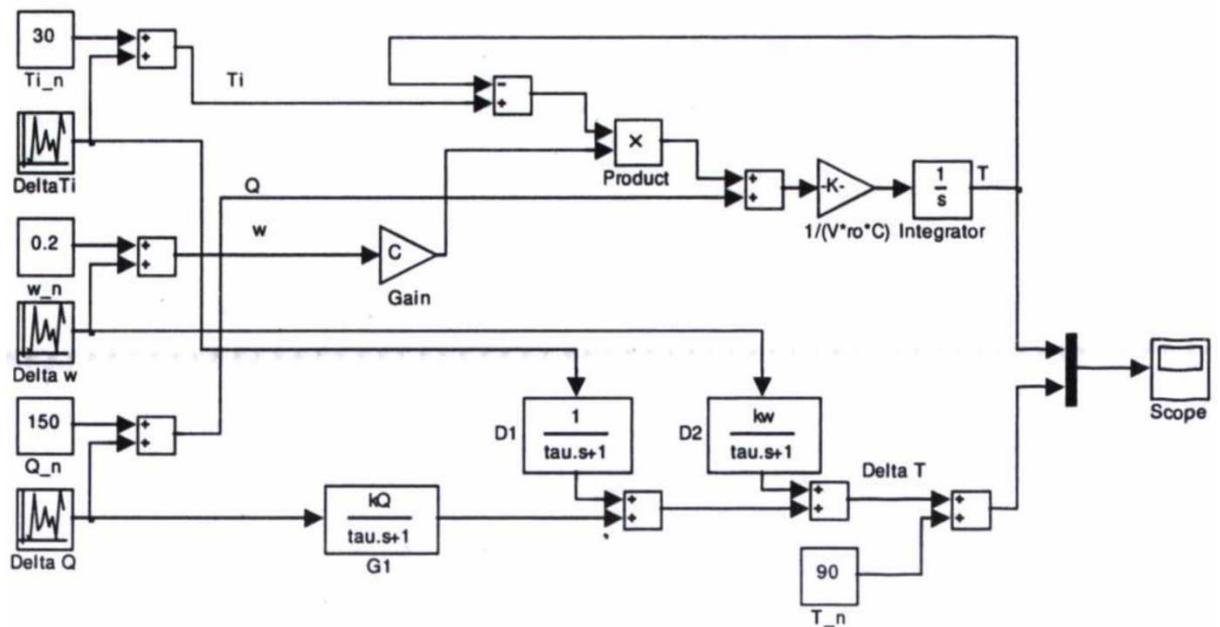


Kết quả, đáp ứng mô phỏng đối với mô hình tuyến tính và phi tuyến như sau:

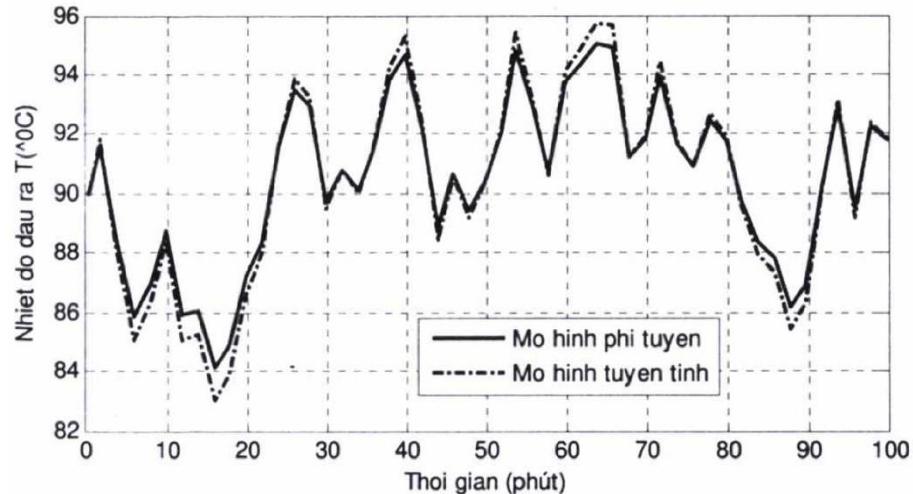


3.3.5 Mô hình quá trình gia nhiệt có khuấy trộn

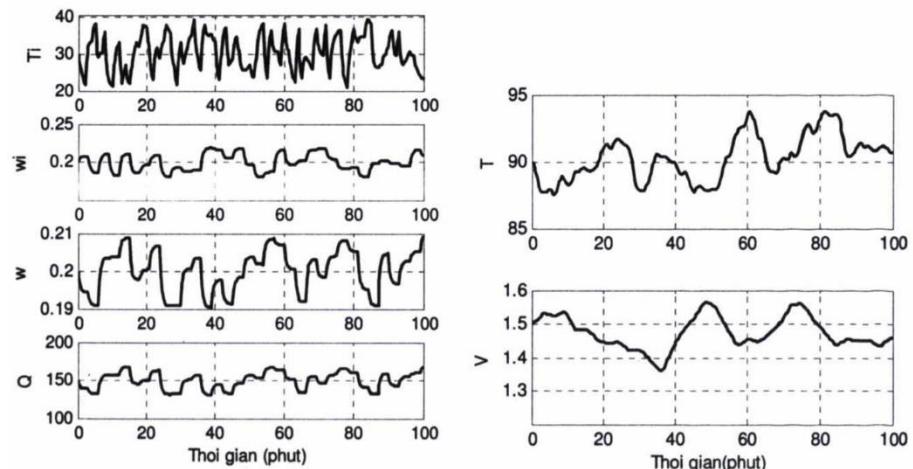
Bài toán 3.5: hãy mô hình hoá quá trình gia nhiệt có khuấy trộn (mục 2.9.3, trang 76-83, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:



Kết quả so sánh đáp ứng của mô hình tuyến tính và phi tuyến như sau:

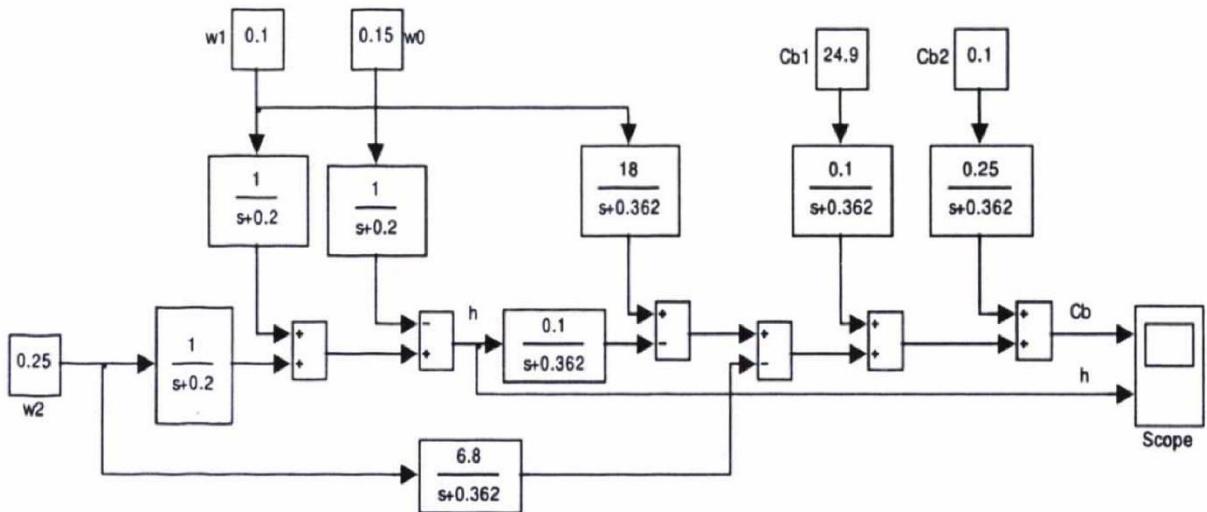


Minh họa sự phụ thuộc của V và T vào các đầu vào T_i , ω_i , ω , Q :

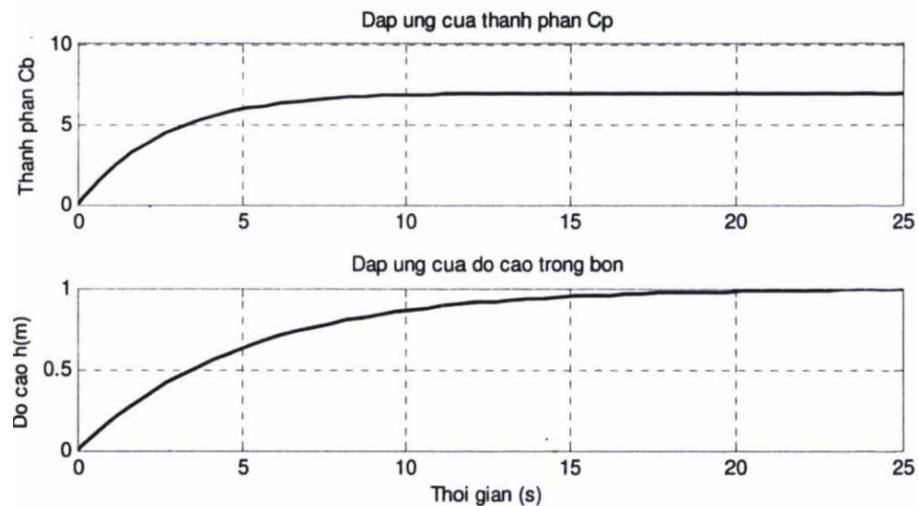


3.3.6 Mô hình quá trình của bồn phản ứng khuấy trộn liên tục

Bài toán 3.6: hãy mô hình hoá quá trình của bồn phản ứng khuấy trộn liên tục (mục 2.9.4, trang 83-86, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:

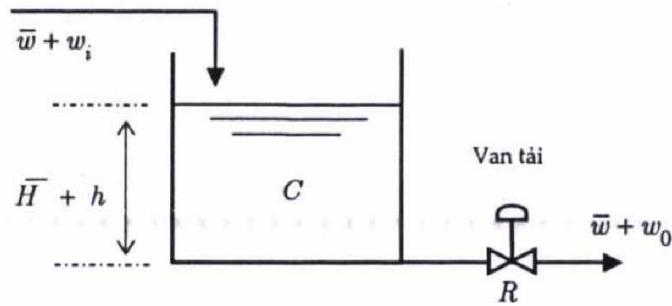


Đáp ứng của thành phần C_b và độ cao h của quá trình khuấy trộn liên tục với các thông số tại điểm làm việc:



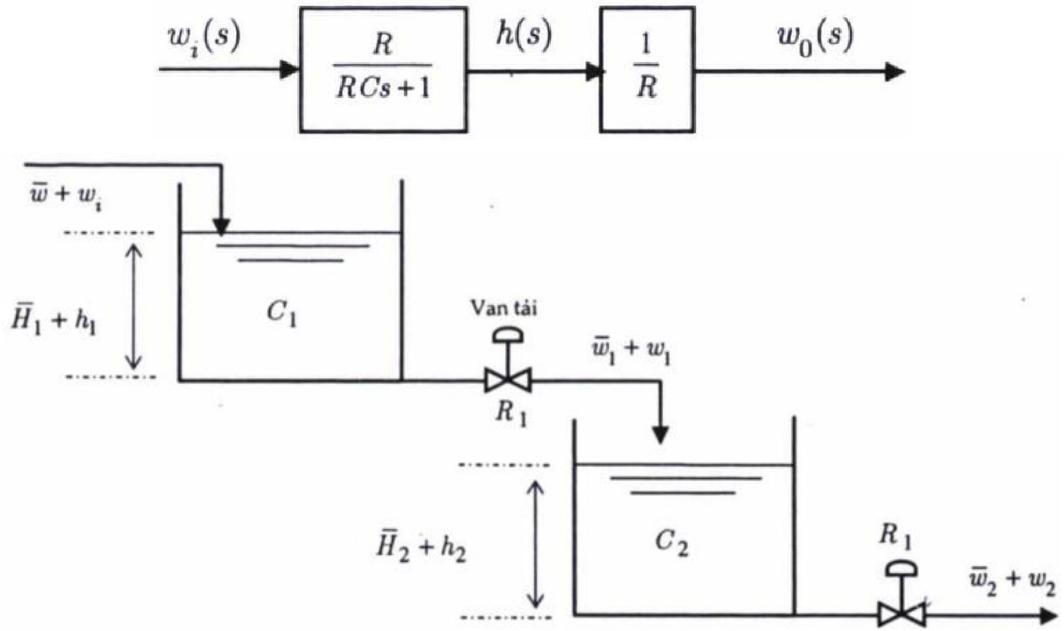
3.3.7 Mô hình quá trình mức chất lỏng

Bài toán 3.7: hãy mô hình hoá quá trình mức chất lỏng 1 bồn chứa và 2 bồn chứa (mục 2.9.5, trang 86-88, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:



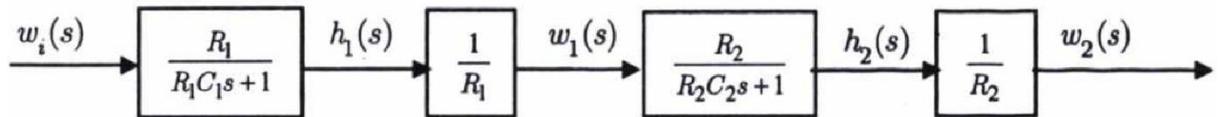
Hình 3.5. Quá trình mức chất lỏng 1 bồn chứa

$$G(s) = \frac{w_0(s)}{w_i(s)} = \frac{R}{RCs + 1} \frac{1}{R}$$



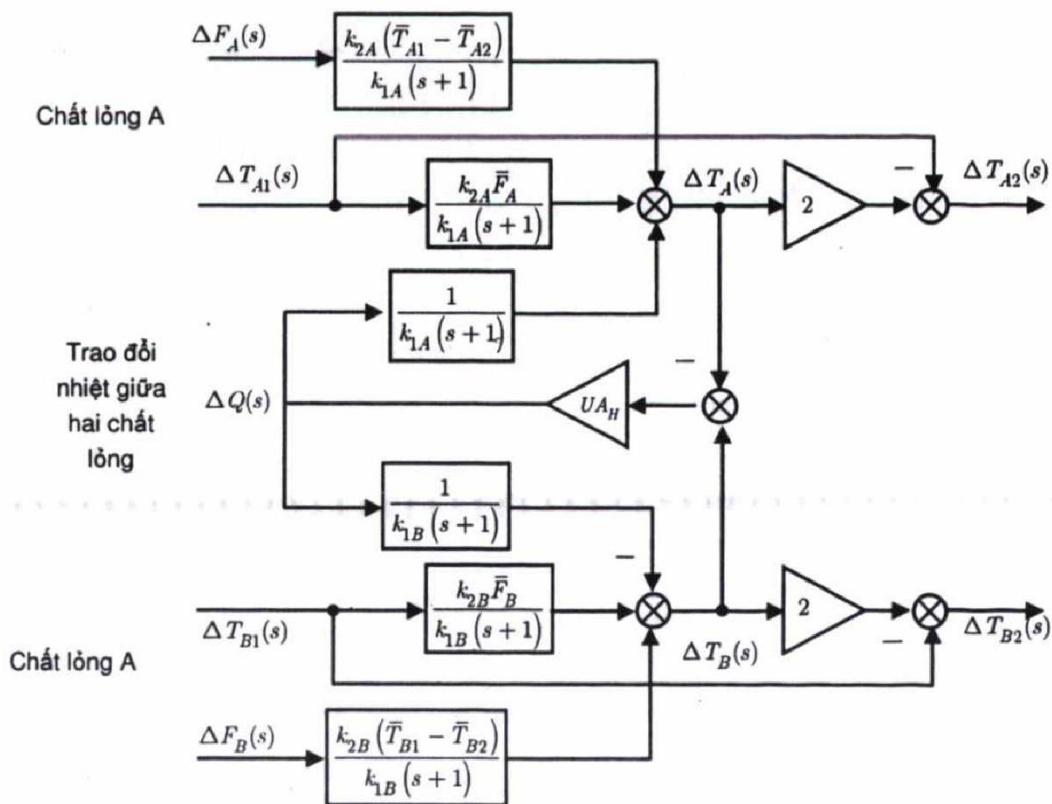
Hình 3.6. Quá trình mức chất lỏng 2 bồn chứa không tương tác

$$G(s) = \frac{w_2(s)}{w_i(s)} = G_1(s)G_2(s) = \frac{1}{(R_1C_1s + 1)(R_2C_2s + 1)}$$

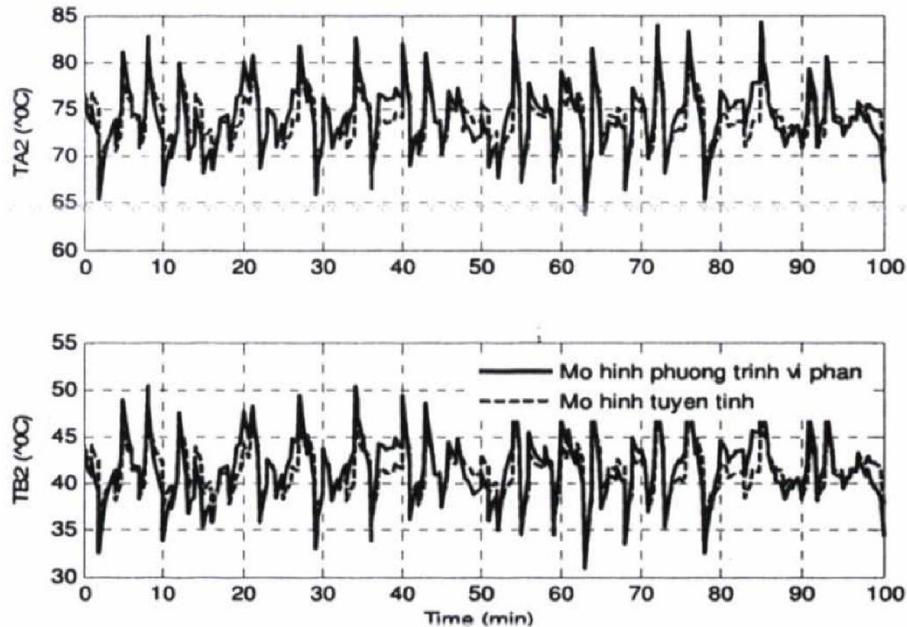


3.3.8 Mô hình quá trình trao đổi nhiệt chất lỏng

Bài toán 3.8: hãy mô hình hoá quá trình trao đổi nhiệt chất lỏng (mục 2.9.6, trang 88-92, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:

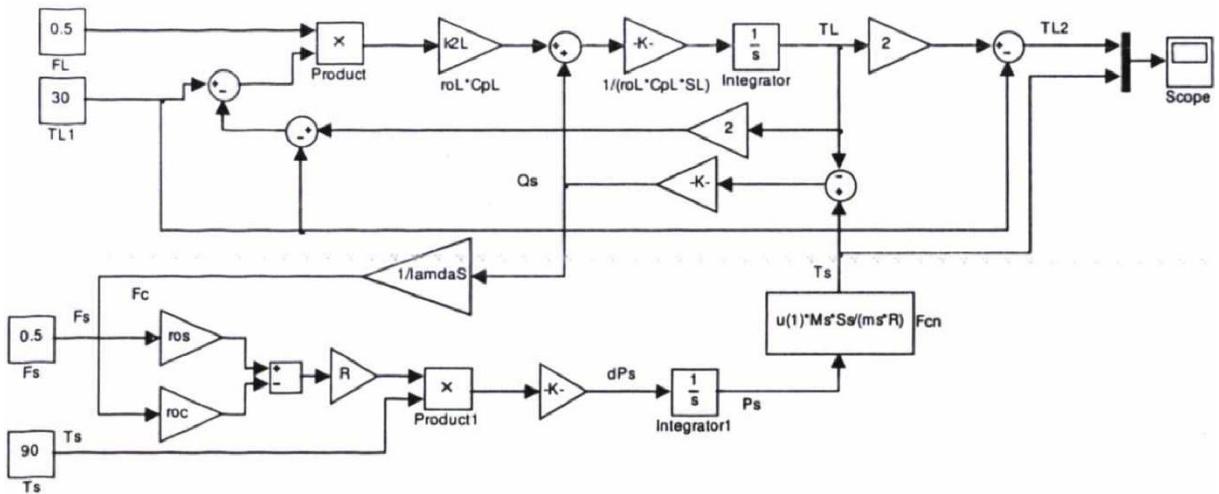


So sánh đáp ứng nhiệt độ đầu ra của chất lỏng A, B khi các đầu vào thay đổi xung quanh điểm làm việc đối với mô hình phi tuyến và mô hình tuyến tính

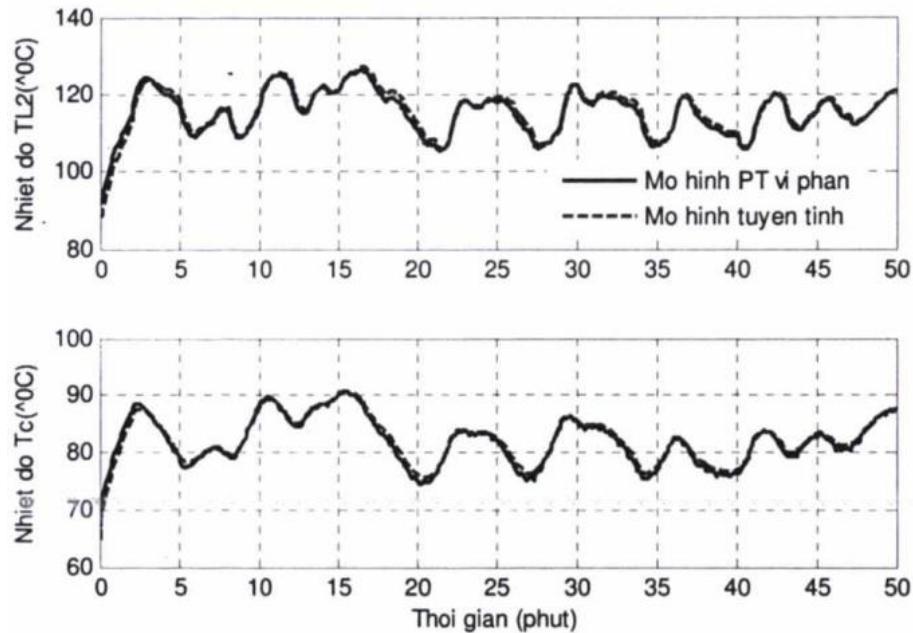


3.3.9 Mô hình quá trình của bộ trao đổi nhiệt bằng hơi

Bài toán 3.9: hãy mô hình hoá quá trình trao đổi nhiệt bằng hơi (mục 2.9.7, trang 92-97, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:

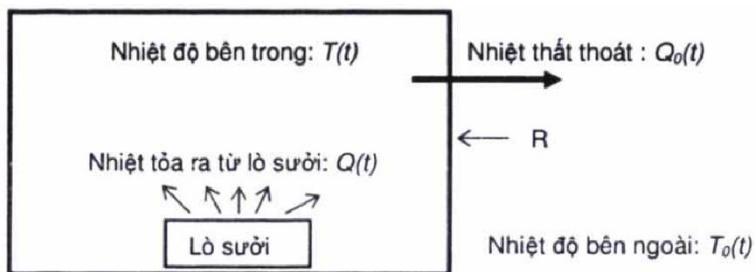


So sánh đáp ứng nhiệt độ đầu ra đối với mô hình phi tuyến và mô hình tuyến tính:



3.3.10 Mô hình quá trình của bộ trao đổi nhiệt có trễ

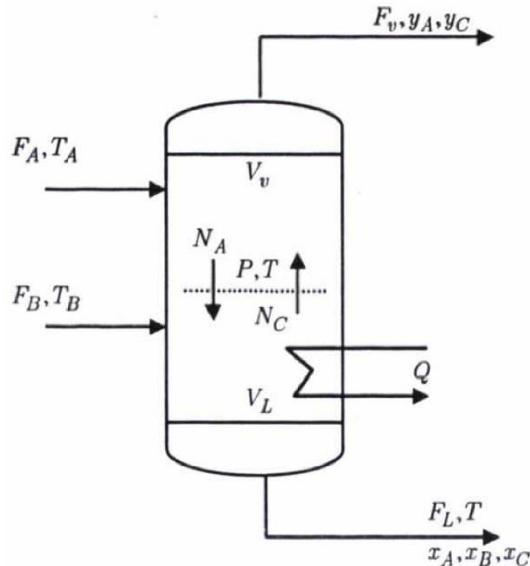
Bài toán 3.10: hãy mô hình hóa quá trình trao đổi nhiệt có trễ (mục 2.9.8, trang 97-101, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:



Hình 3.7. Quá trình trao đổi nhiệt có trễ

3.3.11 Mô hình quá trình phản ứng 2 pha

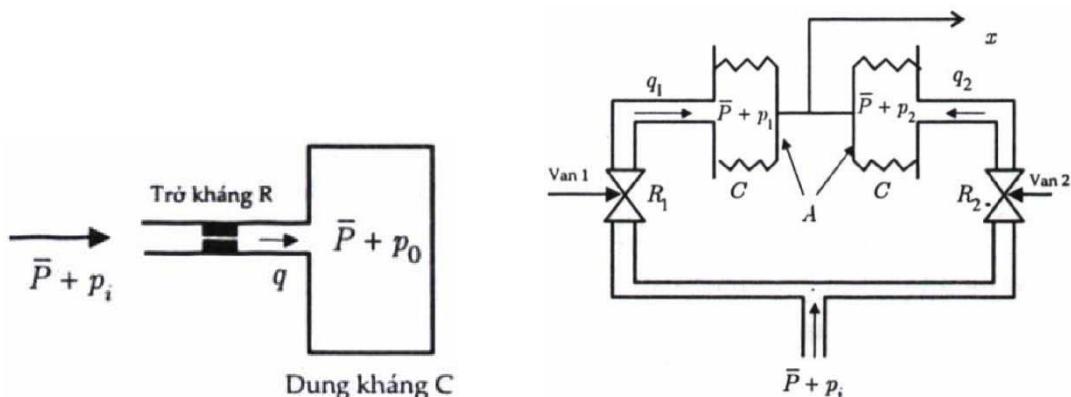
Bài toán 3.11: hãy mô hình hoá quá trình phản ứng 2 pha (mục 2.9.9, trang 101-103, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:



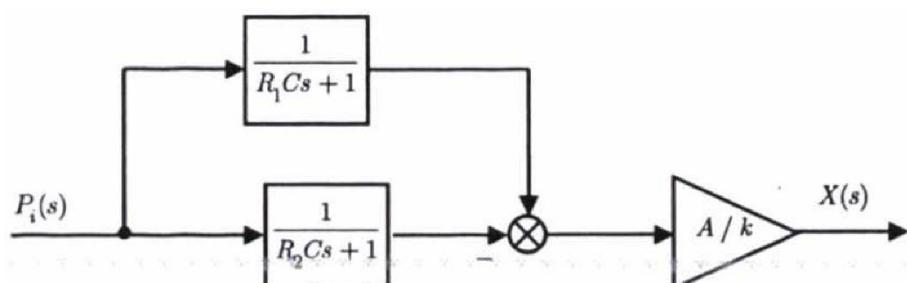
Hình 3.8. Quá trình phản ứng 2 pha

3.3.12 Mô hình quá trình khí nén

Bài toán 3.12: hãy mô hình hoá quá trình khí nén (mục 2.9.10, trang 103-105, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:

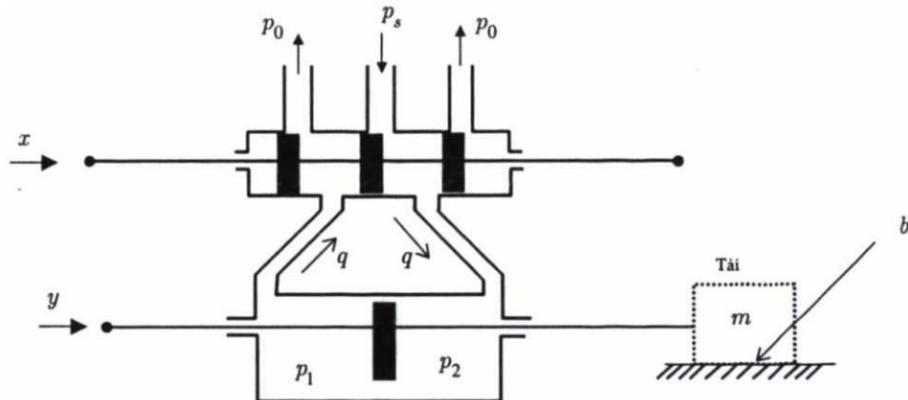


Hình 3.9. Quá trình khí nén và dịch chuyển bằng khí nén



3.3.13 Mô hình quá trình thuỷ lực

Bài toán 3.13: hãy mô hình hoá quá trình thuỷ lực (mục 2.9.11, trang 105-106, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:



Hình 3.9. Quá trình dịch chuyển bằng thuỷ lực

3.4. Câu hỏi thực hành

Thực hiện mô hình hoá quá trình trong các bài tập 4-10, mục 2.10, chương 2, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí.

TÀI LIỆU THAM KHẢO

- [1] Hoàng Minh Sơn, Cơ sở điều khiển quá trình, Nhà xuất bản Bách khoa Hà Nội, 2009.
- [2] Nguyễn Văn Chí, giáo trình điều khiển các quá trình công nghệ, Nhà xuất bản Khoa học kỹ thuật, 2016.
- [3] Nguyễn Phùng Quang, MATLAB&SIMULINK dành cho kỹ sư điều khiển tự động, Nhà xuất bản Khoa học kỹ thuật, 2003.

Bài 4. Thiết kế và chỉnh định tham số bộ điều khiển PID

4.1. Mục tiêu

Sau khi hoàn thành bài thực hành này, sinh viên có khả năng sử dụng được các công cụ trong phần mềm MATLAB để thiết kế và chỉnh định tham số bộ điều khiển PID cho một số quá trình cơ bản.

4.2. Nội dung thực hành

Thực hiện thiết kế bộ điều khiển phản hồi, chỉnh định tham số bộ điều khiển PID cho một số quá trình cơ bản.

4.3. Các bước thực hành

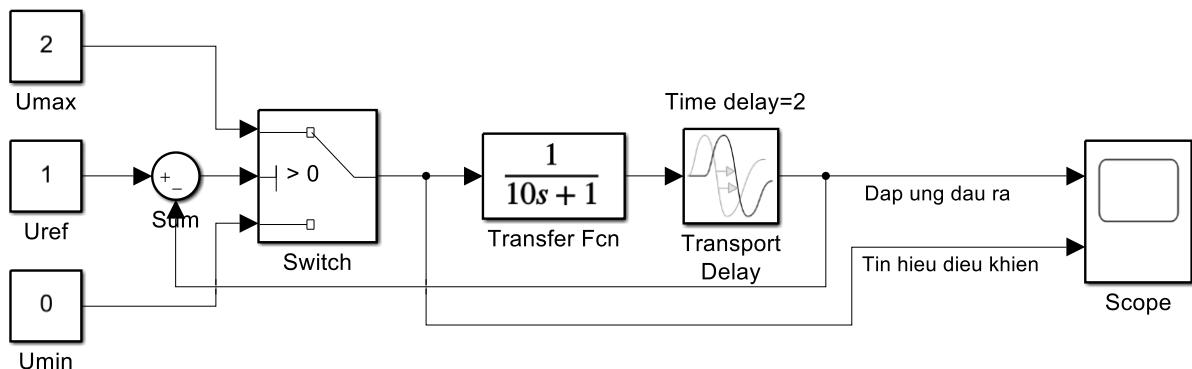
4.3.1 Thiết kế bộ điều khiển ON-OFF

Bài toán 4.1. Cho một quá trình có trễ với hàm truyền:

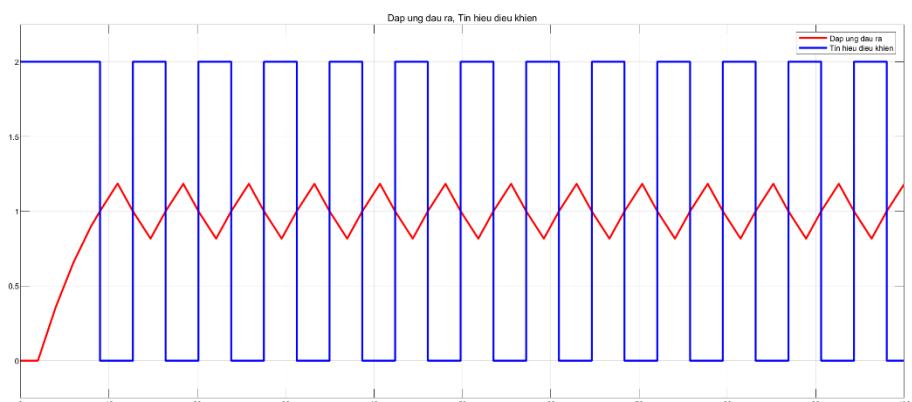
$$P(s) = \frac{1}{10s + 1} e^{-2s}$$

Thiết kế bộ điều khiển ON-OFF trên SIMULINK với các tham số $u_{max}=2V$, $u_{min}=0 V$, giá trị đặt là 1. Xuất dạng tín hiệu điều khiển thể hiện tính rung trong các trường hợp: lý tưởng và có khâu ro-le.

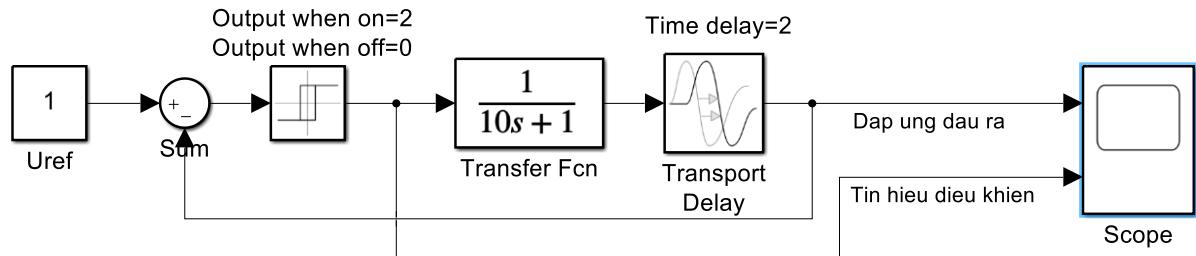
- B1: thiết kế hệ thống trên SIMULINK trong trường hợp lý tưởng như sau:



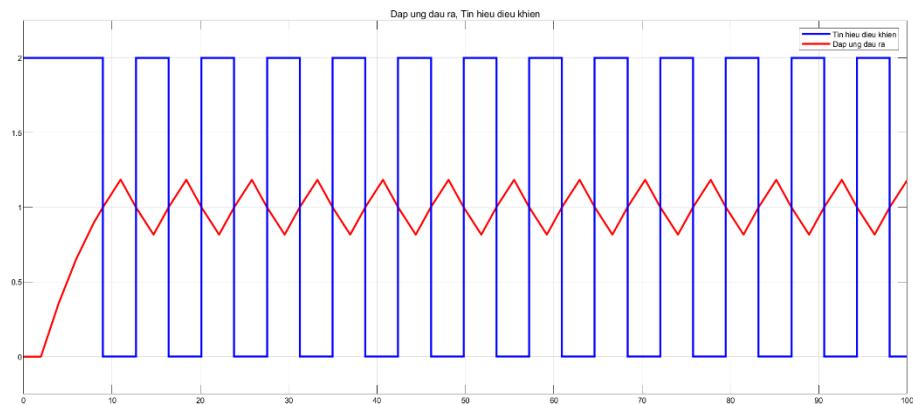
Đặt thời gian mô phỏng là 100s. Đáp ứng đầu ra và tín hiệu điều khiển được thể hiện:



- B2: thiết kế hệ thống trên SIMULINK trong trường sử dụng khâu rơ-le như sau:



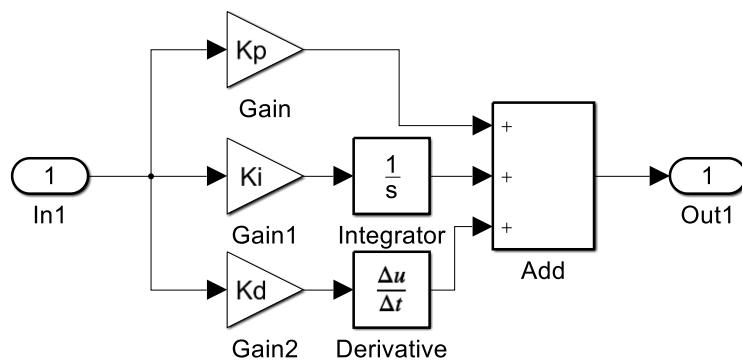
Đặt thời gian mô phỏng là 100s. Dáp ứng đầu ra và tín hiệu điều khiển được thể hiện:



*) Câu hỏi: nhận xét về bộ điều khiển ON-OFF?

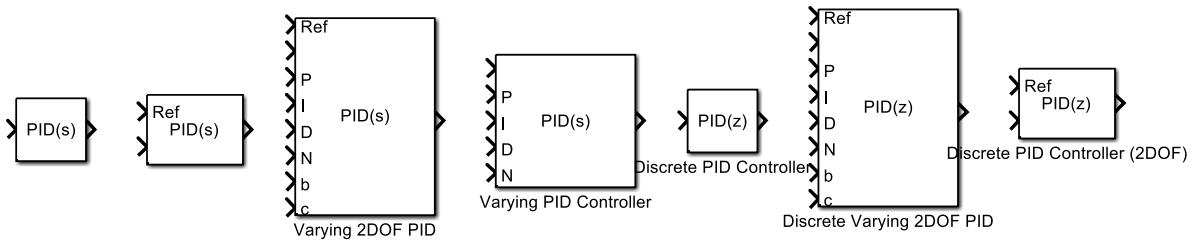
4.3.2 Thiết kế bộ điều khiển PID

Có thể dễ dàng thiết kế bộ điều khiển PID bằng dòng lệnh trên MATLAB hoặc bằng các khối trên SIMULINK. Trên SIMULINK, mô hình tổng quát của bộ điều khiển PID như sau:



Hình 4.1. Mô hình SIMULINK tổng quát bộ điều khiển PID

Hoặc sử dụng các khối có sẵn của SIMULINK:



Hình 4.2. Các khối PID trong SIMULINK

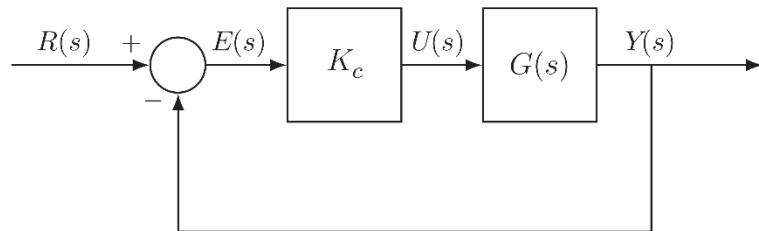
Tuy nhiên, tuỳ thuộc vào yêu cầu bài toán mà có thể thiết kế các dạng bộ điều khiển PID theo các mô hình khác nhau.

Lưu ý về mặt kí hiệu:

- Bộ điều khiển P được mô tả như sau:

$$u(t) = K_c e(t)$$

với K_c chính là hệ số tỉ lệ K_p . Trong hệ thống điều khiển phản hồi, bộ điều khiển P được thiết kế như sau:



Hình 4.3. Kiến trúc bộ điều khiển P trong vòng phản hồi

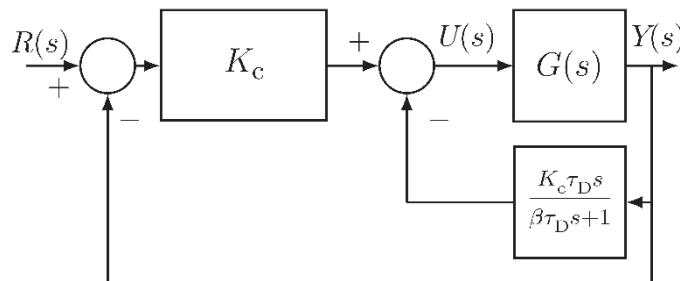
- Bộ điều khiển PD được mô tả như sau:

$$u(t) = K_c e(t) + K_c \tau_D \dot{e}(t)$$

Dưới dạng hàm truyền:

$$\frac{U(s)}{E(s)} = K_c + K_c \tau_D s.$$

với K_c chính là hệ số tỉ lệ K_p . Hệ số vi phân $K_d = K_c \tau_D$, trong đó τ_D là tham số vi phân. Trong hệ thống điều khiển phản hồi, bộ điều khiển PD được thiết kế như sau:



Hình 4.4. Kiến trúc bộ điều khiển PD trong vòng phản hồi

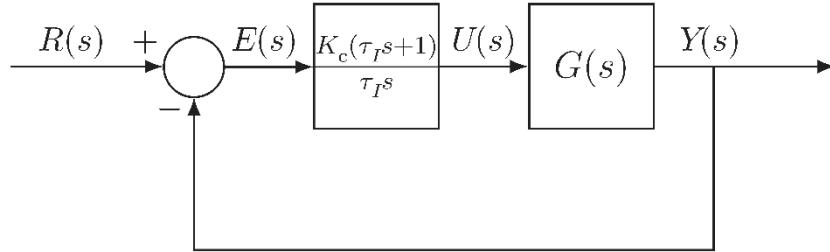
- Bộ điều khiển PI được mô tả như sau:

$$u(t) = K_c e(t) + \frac{K_c}{\tau_I} \int_0^t e(\tau) d\tau$$

Dưới dạng hàm truyền:

$$U(s) = K_c E(s) + \frac{K_c}{\tau_I s} E(s)$$

với K_c chính là hệ số tỉ lệ K_p . Hệ số tích phân $K_i = \frac{K_c}{\tau_i}$, trong đó τ_i là tham số tích phân. Trong hệ thống điều khiển phản hồi, bộ điều khiển PI được thiết kế như sau:



Hình 4.5. Kiến trúc bộ điều khiển PI trong vòng phản hồi

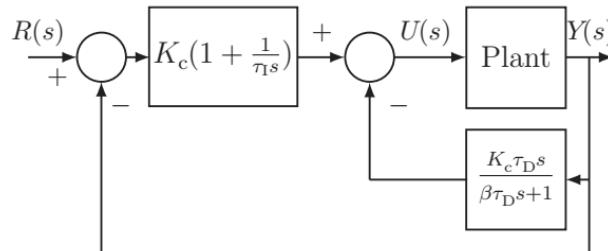
- Bộ điều khiển PID được mô tả như sau:

$$u(t) = K_c e(t) + \frac{K_c}{\tau_I} \int_0^t e(\tau) d\tau + K_c \tau_D \frac{de(t)}{dt}$$

Dưới dạng hàm truyền:

$$\frac{U(s)}{E(s)} = K_c \left(1 + \frac{1}{\tau_I s} + \tau_D s \right)$$

với K_c chính là hệ số tỉ lệ K_p . Hệ số tích phân $K_i = \frac{K_c}{\tau_i}$, trong đó τ_i là tham số tích phân. Hệ số vi phân $K_d = K_c \tau_D$, trong đó τ_D là tham số vi phân. Trong hệ thống điều khiển phản hồi, bộ điều khiển PID được thiết kế như sau:



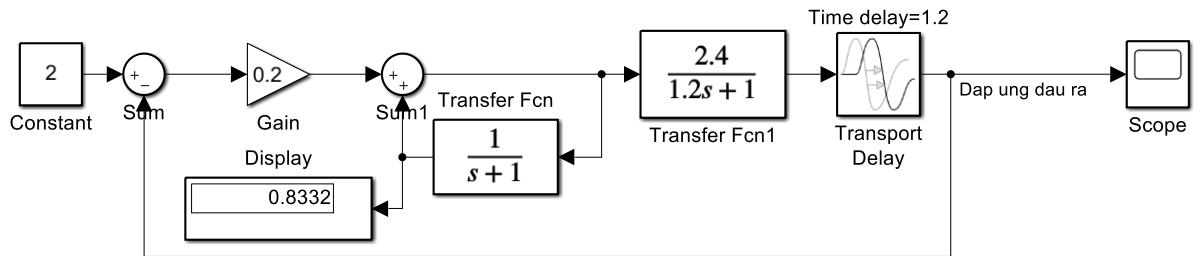
Hình 4.6. Kiến trúc bộ điều khiển PID trong vòng phản hồi

Bài toán 4.2. Cho một quá trình có trễ với hàm truyền:

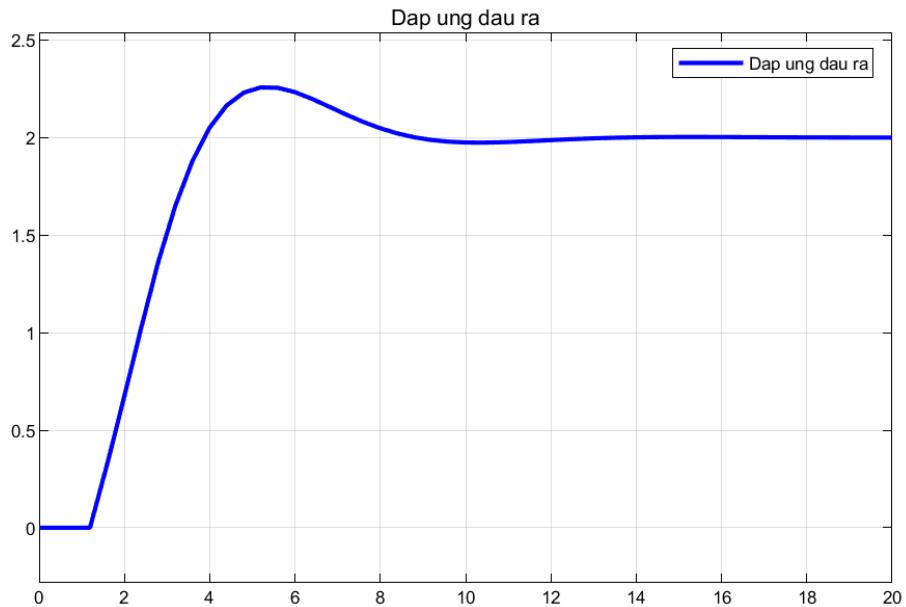
$$P(s) = \frac{2.4}{1.2s + 1} e^{-1.2s}$$

Thiết kế bộ điều khiển PI trên SIMULINK với cấu hình reset tự động $1/(s + 1)$ đóng vai trò là tín hiệu bù cho thành phần tỉ lệ $K_p = 0.2$, tín hiệu đặt là 2.

Thiết kế hệ thống trên SIMULINK trong trường hợp lý tưởng như sau:



Đáp ứng đầu ra của hệ thống:



*) Câu hỏi: thay thế bộ điều khiển PI bằng bộ điều khiển ON-OFF như trong Bài toán 1. So sánh chất lượng bộ điều khiển PI so với bộ điều khiển ON-OFF?

Bài toán 4.3. Cho một quá trình bậc 3 với hàm truyền:

$$P(s) = \frac{1}{(s + 1)^3}$$

Phân tích chất lượng của bộ điều khiển P với các giá trị khác nhau của K_p , K_p từ 0.1 đến 1 (step = 0.1). Phân tích chất lượng của bộ điều khiển PI với $K_p = 1$, K_i từ 1 đến 1.5 (step = 0.1). Phân tích chất lượng của bộ điều khiển PID với $K_p = 1$, $K_i = 1$, K_d từ 0.1 đến 2 (step = 0.1).

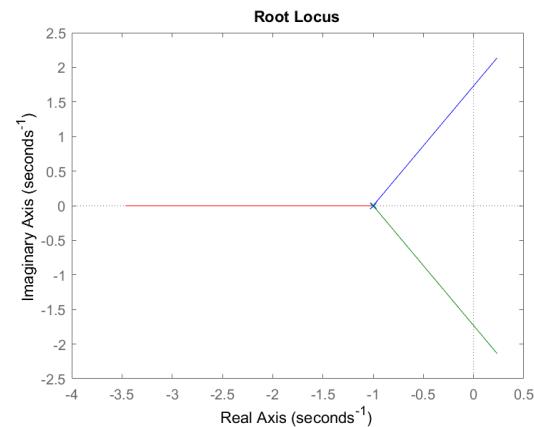
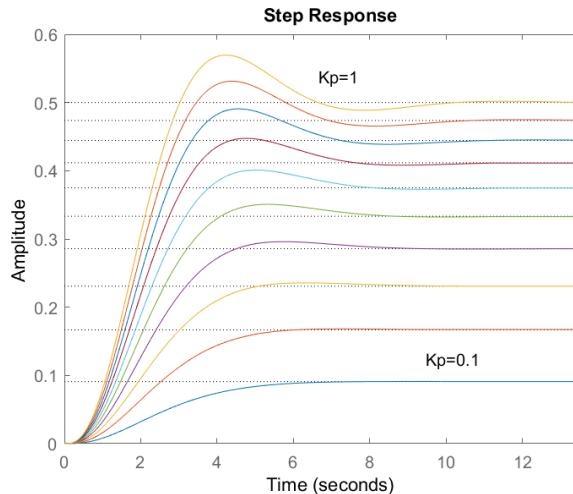
Đối với việc phân tích sự thay đổi của quá trình trong một dải giá trị của tham số điều khiển, sử dụng dòng lệnh MATLAB sẽ thuận lợi hơn so với việc sử dụng SIMLINK.

- B1: thực hiện dòng lệnh sau trên giao diện Command Window của MATLAB để phân tích chất lượng của bộ điều khiển P với các giá trị khác nhau của K_p , K_p từ 0.1 đến 1 (step = 0.1).

Command Window

```
>> P=tf(1, [1, 3, 3, 1]);
for Kp=[0.1:0.1:1]
    Gk=feedback(Kp*P, 1); step(Gk); hold on;
end
figure
rlocus(P, [0, 15])
```

Đáp ứng đầu ra và quỹ đạo nghiệm số của quá trình nhận được như sau:



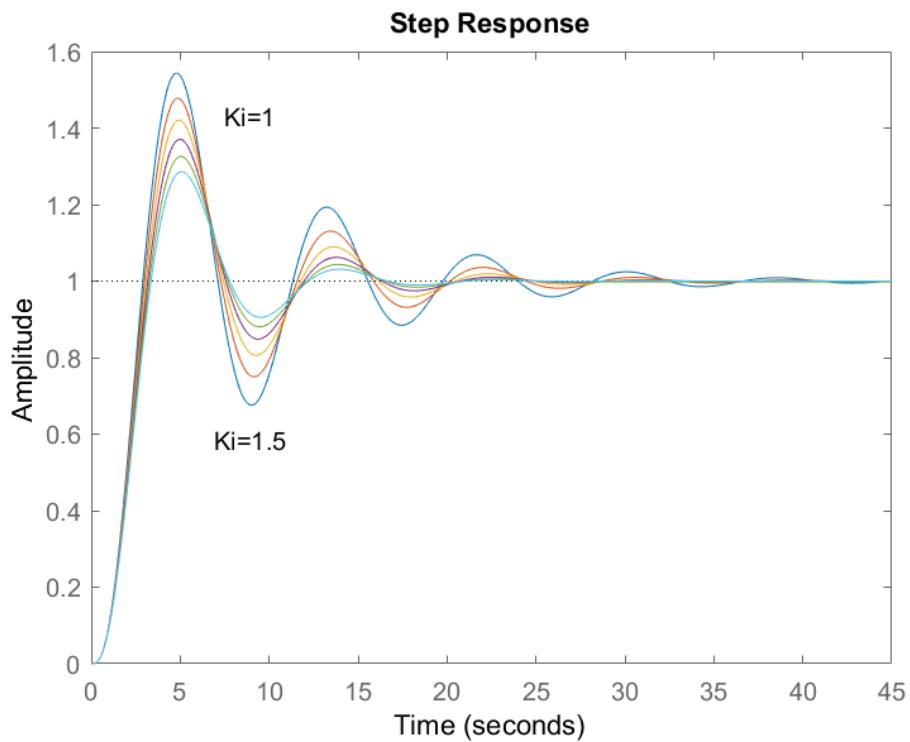
Từ đáp ứng của hệ kín cho thấy rằng khi K_p tăng, tốc độ đáp ứng của hệ tăng lên, độ quá điệu chỉnh cũng sẽ tăng và giảm sai lệch tĩnh, tuy nhiên khi K_p đủ lớn thì hệ kín mất ổn định, từ quỹ đạo nghiệm số cho thấy rằng nếu K_p nằm ngoài vùng từ 0 đến 8 thì hệ kín sẽ mất ổn định.

- B2: thực hiện dòng lệnh sau trên giao diện Command Window của MATLAB để phân tích chất lượng của bộ điều khiển PI với $K_p = 1$, K_i từ 1 đến 1.5 (step = 0.1).

Command Window

```
>> s=tf('s');
for Ki=[1:0.1:1.5]
    C=1+1/(Ki*s); Gk=feedback(C*P, 1); step(Gk); hold on
end
```

Đáp ứng đầu ra của quá trình nhận được như sau:

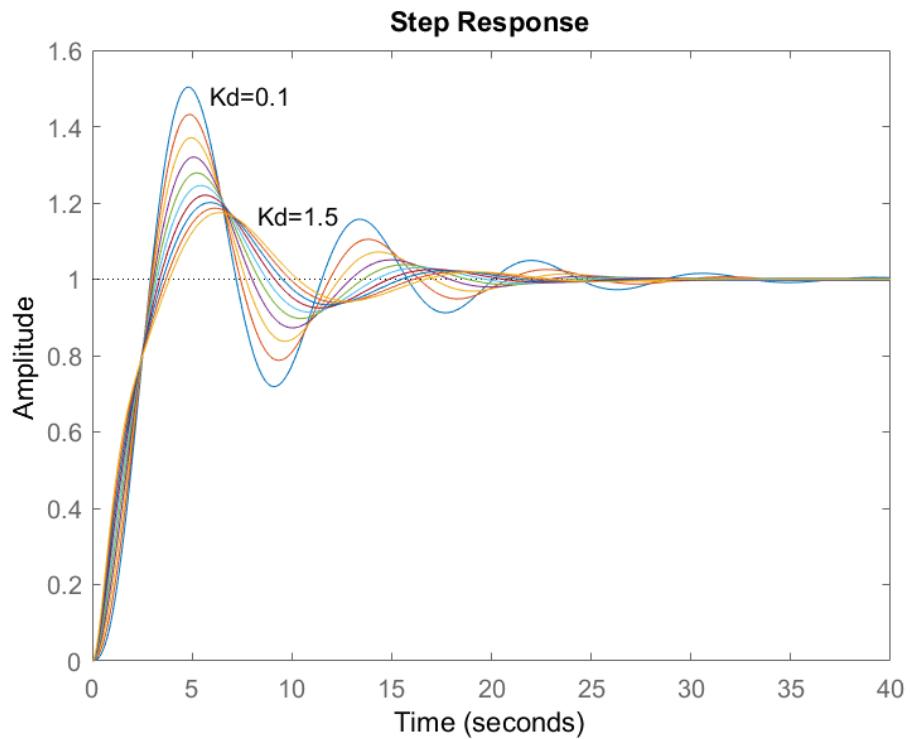


Từ đáp ứng cho thấy rằng khi hằng số thời gian tích phân càng lớn độ điều chỉnh càng giảm, dao động cũng giảm. Tuy nhiên khi có thành phần điều khiển I, hệ thống không còn sai lệch tĩnh, nhưng nếu K_i quá nhỏ hệ kín sẽ mất ổn định ($K_i < 0.6$)

- B3: thực hiện dòng lệnh sau trên giao diện Command Window của MATLAB để phân tích chất lượng của bộ điều khiển PID với $K_p = 1$, $K_i = 1$, K_d từ 0.1 đến 2 (step = 0.1).

```
Command Window
>> Kp=1;Ki=1;s=tf('s');
for Kd=[0.1:0.2:2]
    C=Kp*(1+1/(Ki*s)+Kd*s); Gk=feedback(C*P,1);
    step(Gk); hold on;
end
```

Đáp ứng đầu ra của quá trình nhận được như sau:



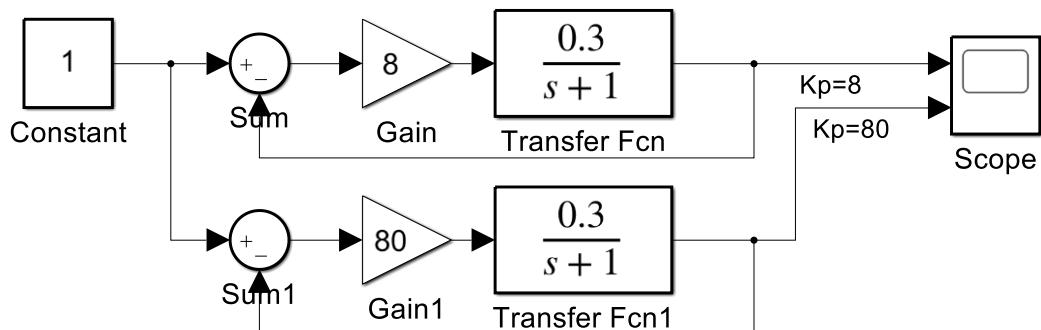
Từ đáp ứng cho thấy, khi K_d nhỏ độ quá điều chỉnh sẽ tăng, khi K_d giảm độ quá điều chỉnh sẽ giảm, tuy nhiên thời gian tăng (rise time) sẽ dài hơn.

Bài toán 4.4. Cho một quá trình có hàm truyền:

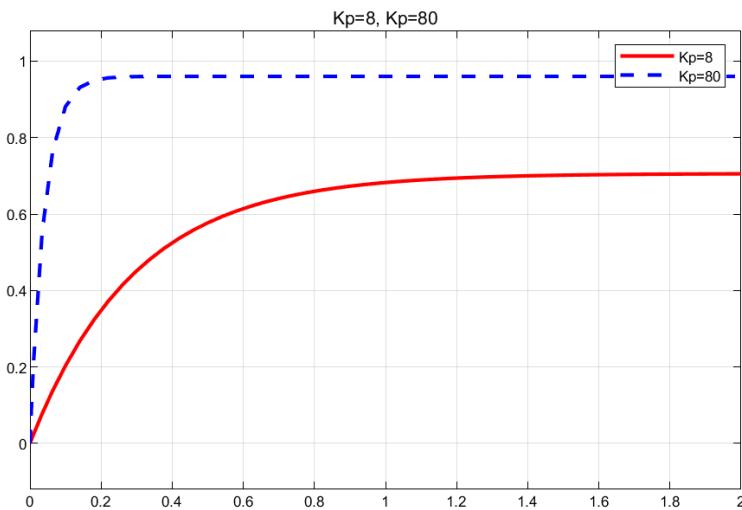
$$G(s) = \frac{0.3}{s + 1}$$

Thiết kế bộ điều khiển tỉ lệ P cho hệ thống này với $K_p = 8$ và $K_p = 80$. So sánh đáp ứng đầu ra. Rút ra nhận xét về bộ điều khiển P.

- B1: thiết kế mô hình trên SIMULINK như sau:



- Đặt thời gian mô phỏng là 2s, đáp ứng đầu ra của 2 trường hợp $K_p = 8$ và $K_p = 80$ như sau:

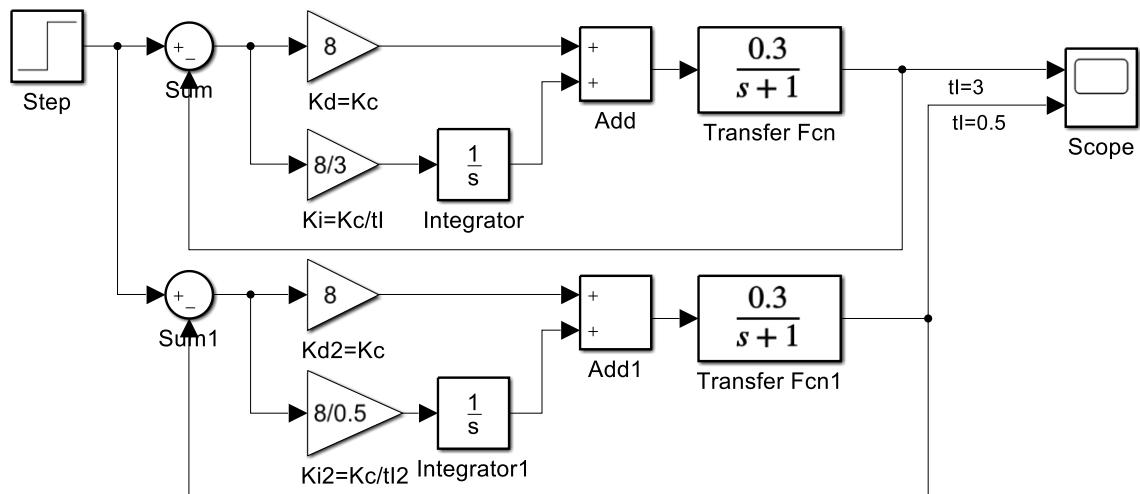


Bài toán 4.5. Cho một quá trình có hàm truyền:

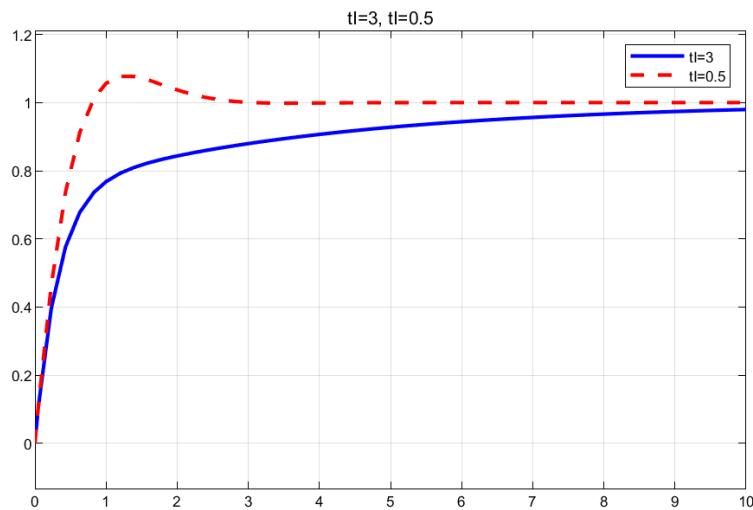
$$G(s) = \frac{0.3}{s + 1}$$

Thiết kế bộ điều khiển tỉ lệ PI cho hệ thống này với $K_c = 8$, $\tau_I = 3$ và $\tau_I = 0.5$. Hãy thiết kế theo 3 dạng mô hình: dạng mô hình tổng quát (Hình 4.1), dạng mô hình khói SIMULINK có sẵn (Hình 4.2) và dạng theo hàm truyền (Hình 4.5). So sánh đáp ứng đầu ra khi $\tau_I = 3$ và $\tau_I = 0.5$, rút ra nhận xét.

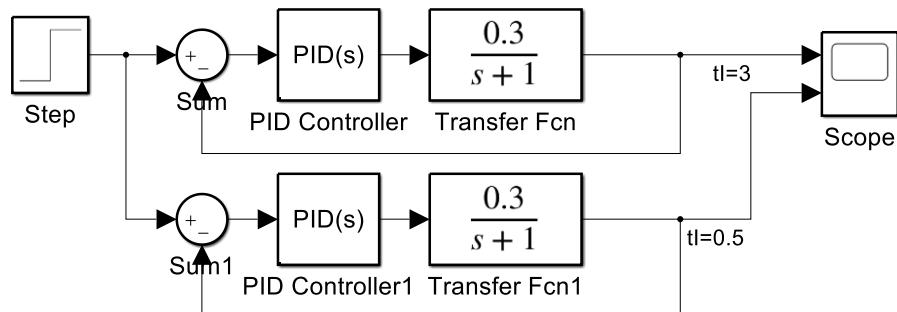
- B1: thiết kế mô hình trên SIMULINK theo dạng mô hình tổng quát (Hình 4.1) như sau:



Đặt Initial value của khối Step là 1, thời gian mô phỏng là 10s. Kết quả mô phỏng như sau:

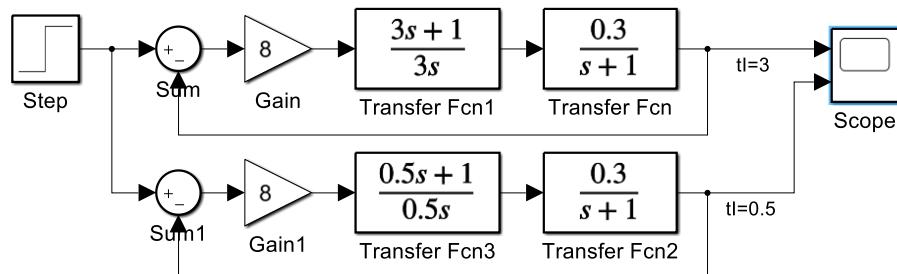


- B2: thiết kế mô hình trên SIMULINK theo dạng mô hình khối có sẵn trên SIMULINK (Hình 4.2) như sau:



Lưu ý: ở khối PID controller, đặt giá trị các tham số như sau: $P=8$, $I=8/3$, $D=0$, ở khối PID controller1, đặt giá trị các tham số như sau: $P=8$, $I=8/0.5$, $D=0$. Kết quả mô phỏng tương tự B1.

- B3: thiết kế mô hình trên SIMULINK theo dạng mô hình hàm truyền (Hình 4.5) như sau:



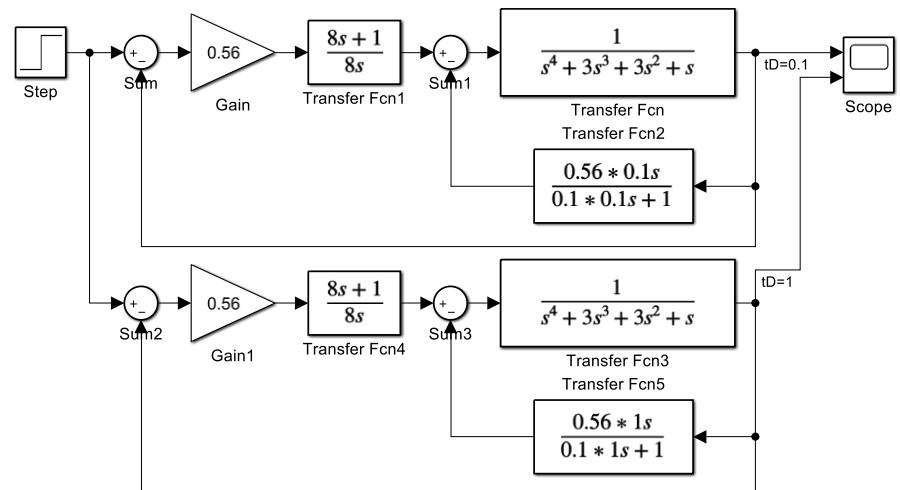
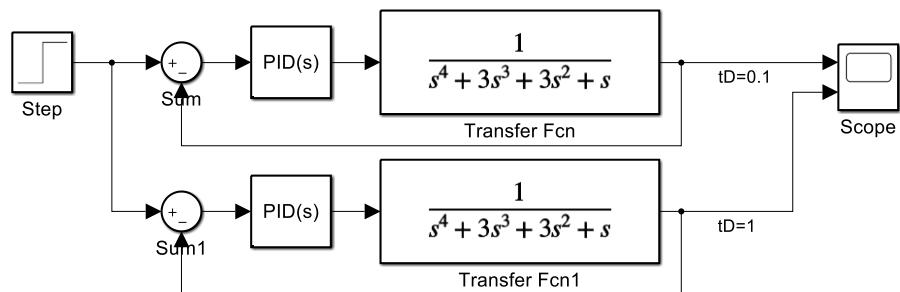
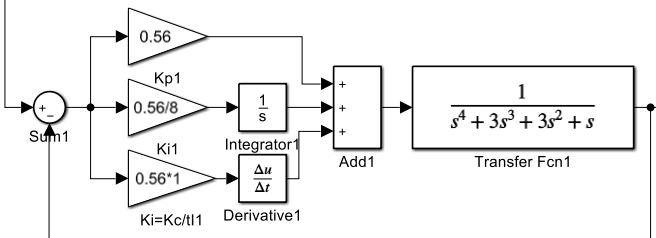
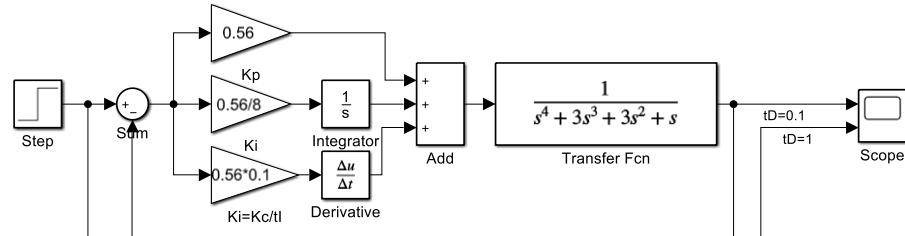
Kết quả mô phỏng tương tự B1. Nhận xét về việc thay đổi tham số τ_I .

Bài toán 4.6. Cho một quá trình có hàm truyền:

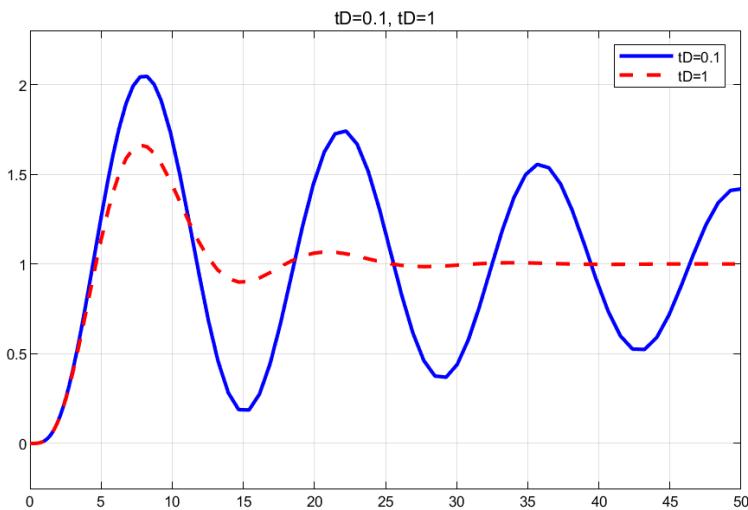
$$G(s) = \frac{1}{s(s+1)^3}$$

Thiết kế bộ điều khiển tỉ lệ PID cho hệ thống này với $K_c=0.56$, $\tau_I=8$, $\tau_D=0.5$ và $\tau_D=1$. Hãy thiết kế theo 3 dạng mô hình: dạng mô hình tổng quát (Hình 4.1), dạng mô hình khói SIMULINK có sẵn (Hình 4.2) và dạng theo hàm truyền (Hình 4.6, chọn $\beta=0.1$). So sánh đáp ứng đầu ra khi $\tau_D=0.5$ và $\tau_D=1$, rút ra nhận xét.

Thiết kế trên SIMULINK theo 3 dạng mô hình như sau:



Kết quả đáp ứng đầu ra trong 2 trường hợp khi $\tau_D=0.5$ và $\tau_D=1$ như sau:



4.3.3 *Chỉnh định bộ điều khiển PID*

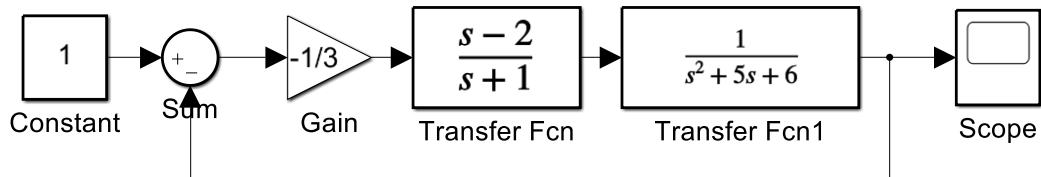
a) *Chỉnh định bộ điều khiển PID cho đối tượng không trễ*

Bài toán 4.7. Cho một quá trình có hàm truyền:

$$G(s) = \frac{s - 2}{(s + 1)(s + 2)(s + 3)}$$

Hãy chỉnh định các tham số cho bộ điều khiển PI và PID theo phương pháp Ziegler-Nichols.

- B1: thiết kế mô hình trên SIMULINK với bộ điều khiển P như sau:



- B2: bắt đầu chỉnh định bằng cách thay đổi các giá trị của K_c từ -1 đến -8, bước nhảy là 0.5. Chúng ta thấy rằng, tại $K_c = -7.5$, đáp ứng đầu ra như sau có dạng chu kỳ hình sin, vậy $K_o = K_c$. Trên Scope, chúng ta vào Tools → measurements → cursor measurements. Chọn 2 điểm đầu và cuối 1 chu kỳ, ta đo được giá trị $P_o = 3.35s$.



Lưu ý: để thuận tiện hơn trong việc thử giá trị K_c nào mà đáp ứng có dạng chu kỳ, nên sử dụng dòng lệnh MATLAB như sau:

```

end
>> P=tf([1,-2],[1,6,11,6]);
for Kp=[0:-0.5:-7.5]
    Gk=feedback(Kp*P,1);step(Gk);hold on;
end

```

- B3: dựa vào bảng tra theo phương pháp Ziegler-Nichols cho đối tượng không trễ bên dưới, chúng ta xác định được các tham số của bộ điều khiển PI và PID như sau:

| | K_c | τ_I | τ_D |
|-----|-----------|-------------------|-----------------|
| P | $0.5K_o$ | | |
| PI | $0.45K_o$ | $\frac{P_o}{1.2}$ | |
| PID | $0.60K_o$ | $\frac{P_o}{2}$ | $\frac{P_o}{8}$ |

Bảng 4.1. Quy tắc Ziegler-Nichols cho đối tượng không trễ

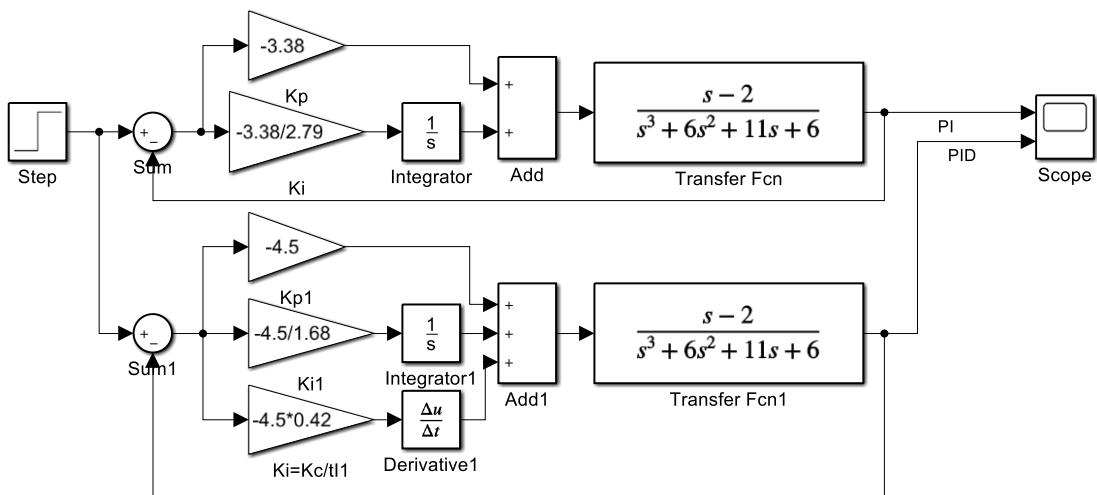
+) bộ điều khiển PI:

$$K_c = 0.45 \times (-7.5) = -3.3, 8\tau_I = \frac{3.35}{1.2} = 2.79$$

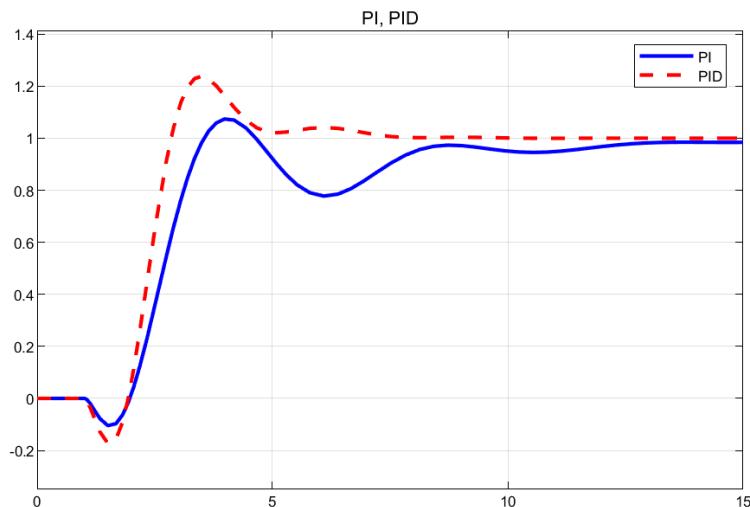
+) bộ điều khiển PID:

$$K_c = 0.6 \times (-7.5) = -4.5, \tau_I = \frac{3.35}{2} = 1.68, \tau_D = \frac{3.35}{8} = 0.42$$

- B4: thiết lập mô hình PI và PID đã chỉnh định trên SIMULINK như sau:



Kết quả như sau, hãy nhận xét về bộ điều khiển PI và PID?

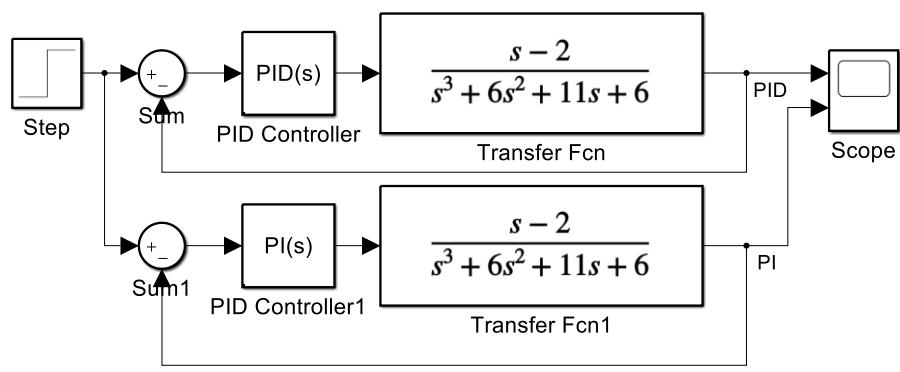


Bài toán 4.8. Cho một quá trình có hàm truyền:

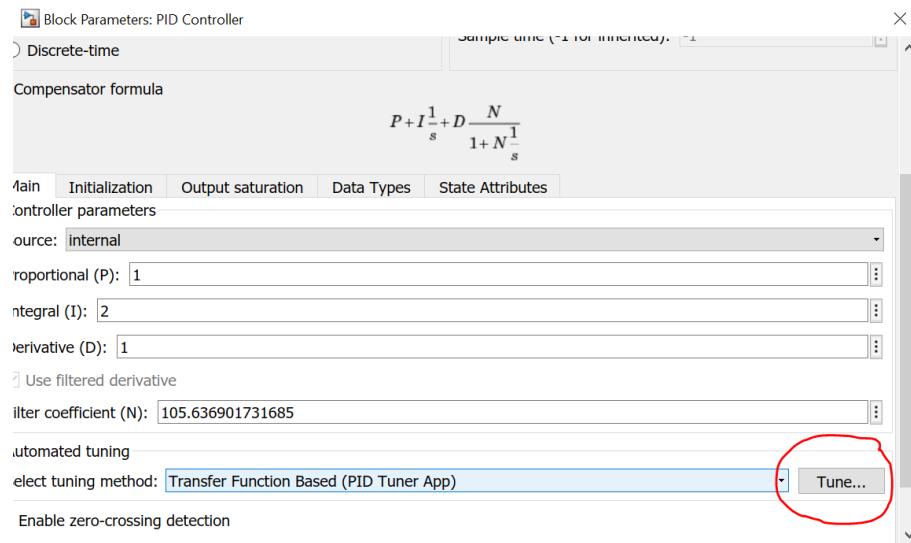
$$G(s) = \frac{s-2}{(s+1)(s+2)(s+3)}$$

Hãy chỉnh định các tham số cho bộ điều khiển PI và PID theo công cụ PID tuner của MATLAB.

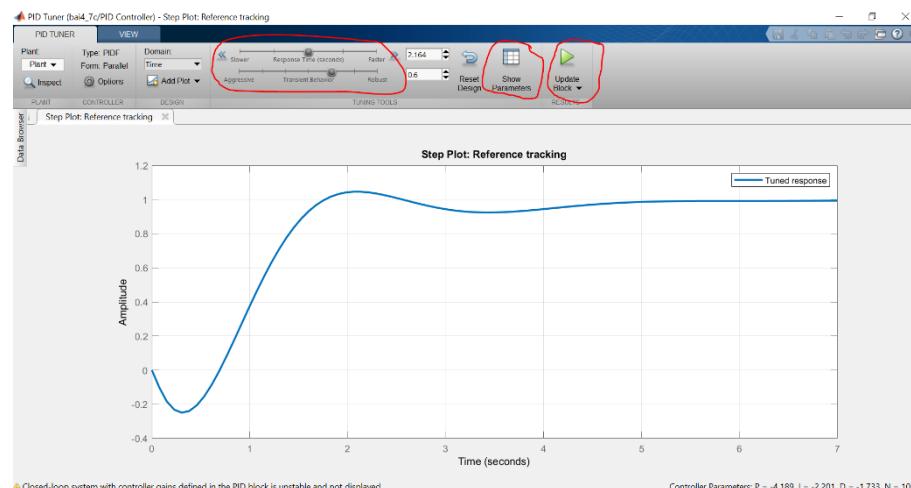
- B1: thiết lập mô hình trên SIMULINK như sau, đối với bộ điều khiển PID và PI chọn tham số bất kỳ.



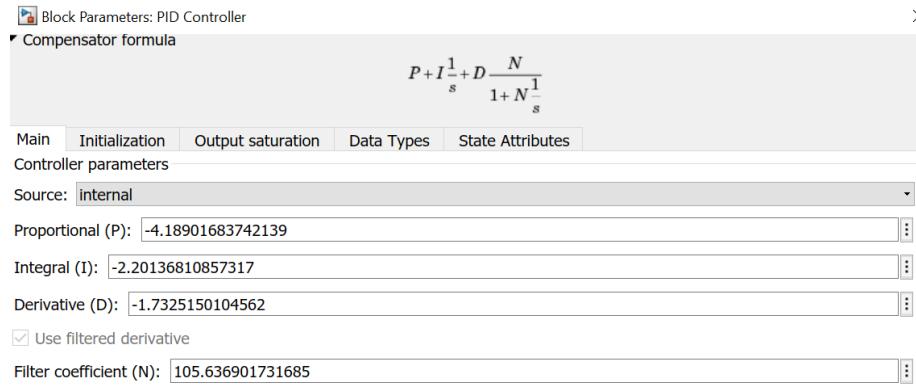
- B2: chỉnh định tham số PID. Nhấn đúp vào khối PID controller, chọn chức năng Tune:



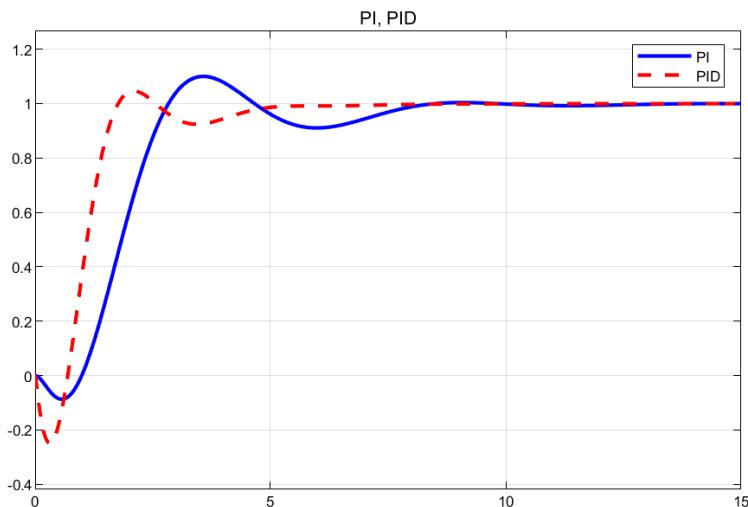
Vào giao diện của PID tuner như sau. MATLAB sẽ tự động chỉnh tham số cho bộ điều khiển PID. Tuy nhiên chúng ta có thể chỉnh định bằng tay theo tiêu chí điều khiển của mình bằng cách sử dụng các thanh kéo Response time (điều chỉnh thời gian đáp ứng nhanh hay chậm) và Transient Behavior (điều chỉnh mức độ đáp ứng mạnh hay yếu). Có thể xem các tham số sau khi chỉnh định tại chức năng Show parameters.



Sau khi lựa chọn phương án cuối cùng, chúng ta nhập vào Update block để cập nhật các tham số đã chỉnh định vào mô hình. Lúc này ở bộ điều khiển PID các tham số đã được cập nhật:



- B3: chỉnh định tham số PI. Nhấn đúp vào khối PID controller1, và thực hiện tương tự như ở bước 2.
- B4: mô phỏng, chúng ta nhận được đáp ứng của hệ thống như sau:



*) Câu hỏi: đưa ra nhận xét về 2 phương pháp chỉnh định trong Bài toán 4.7 và Bài toán 4.8?

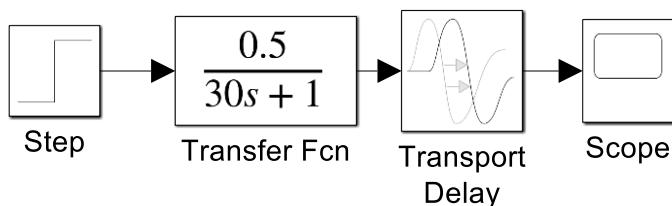
b) Chỉnh định bộ điều khiển PID cho đối tượng có trễ

Bài toán 4.9. Cho một quá trình có hàm truyền:

$$G(s) = \frac{0.5e^{-20s}}{30s + 1}$$

Hãy chỉnh định các tham số cho bộ điều khiển PI sử dụng các phương pháp: Ziegler–Nichols, Cohen–Coon, Wang–Cluett.

- B1: thiết lập mô hình $G(s)$ trên SIMULINK như sau, lưu ý thiết lập chức năng Save format trong khối To Workspace là Structure With Time:

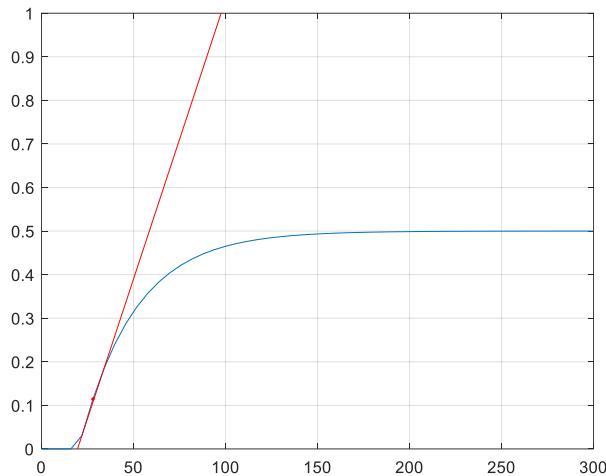


Sau đó, sử dụng lệnh sau trên MATLAB để vẽ tiếp tuyến của đáp ứng:

```

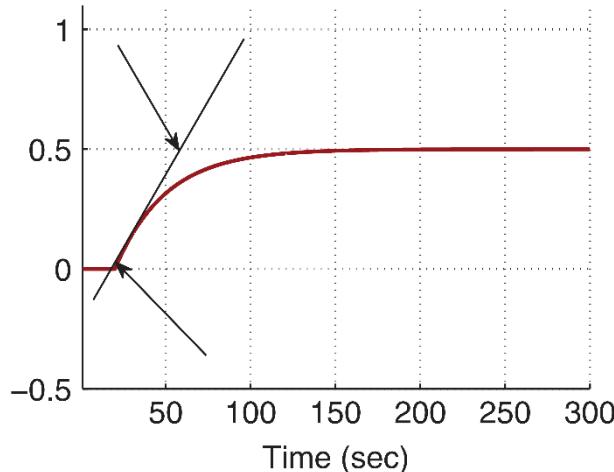
>> y=simout.signals.values;
t=simout.time;
h=mean(diff(t));
dy=gradient(y, h);
[~, idx]=max(dy);
b=[t([idx-1, idx+1]), ones(2,1)]\y([idx-1, idx+1]);
tv=[-b(2)/b(1); (1-b(2))/b(1)];
f =[tv, ones(2,1)]*b;
figure
plot(t,y)
hold on
plot(tv, f, '-r')
plot(t(idx), y(idx), '.r')
hold off
grid

```



Lưu ý: bước này cũng có thể được thực hiện hoàn toàn bằng lệnh MATLAB mà không cần sử dụng mô hình SIMULINK.

- Bước 2: sử dụng hàm ginput (2) để lấy giá trị 2 điểm cần thiết. Thiết lập lệnh trên MATLAB, sẽ xuất hiện hệ toạ độ trực (x,y). Lần lượt click vào 2 điểm 1 và 2 dưới đây để nhận giá trị trả về trên MATLAB:



```
>> ginput(2)
```

```
ans =
```

$$\begin{matrix} 19.8589 & -0.0020 \\ 58.5685 & 0.4946 \end{matrix}$$

Việc xác định cơ học như vậy có thể sai lệch không nhỏ, cho những kết quả khác nhau. Ma trận điểm (x,y) nhận được chính là ma trận:

$$\begin{bmatrix} t_1 & Y_0 \\ t_2 & Y_s \end{bmatrix}$$

- B2: xác định các tham số chỉnh định chung như sau:

$$K_{ss} = \frac{Y_s - Y_0}{U_s - U_0} \approx 0.5$$

ở đây $U_s - U_0 = 1$ vì đầu vào hệ thống là xung đơn vị (step).

$$d = t_1 = 21$$

$$\tau_M = t_2 - t_1 = 37$$

- B3: dựa vào các bảng quy tắc chỉnh định Ziegler–Nichols, Cohen–Coon, Wang–Cluett, chúng ta xác định được tham số K_c , τ_I cho bộ điều khiển PI:

| | K_c | τ_I | τ_D |
|-----|------------------------------|----------|----------|
| P | $\frac{\tau_M}{K_{ss}d}$ | | |
| PI | $0.9 \frac{\tau_M}{K_{ss}d}$ | $3d$ | |
| PID | $1.2 \frac{\tau_M}{K_{ss}d}$ | $2d$ | $0.5d$ |

Bảng 4.2. Quy tắc Ziegler–Nichols cho đổi tượng có trễ

| | K_c | τ_I | τ_D |
|-----|---|--|----------------------------------|
| P | $\frac{\tau_M}{K_{ss}d} \left(1 + \frac{d}{3\tau_M} \right)$ | | |
| PI | $\frac{\tau_M}{K_{ss}d} \left(0.9 + \frac{d}{12\tau_M} \right)$ | $\frac{d(30\tau_M + 3d)}{9\tau_M + 20d}$ | |
| PID | $\frac{\tau_M}{K_{ss}d} \left(\frac{4}{3} + \frac{d}{4\tau_M} \right)$ | $\frac{d(32\tau_M + 6d)}{13\tau_M + 8d}$ | $\frac{4d\tau_M}{11\tau_M + 2d}$ |

Bảng 4.3. Quy tắc Cohen–Coon cho đổi tượng có trễ

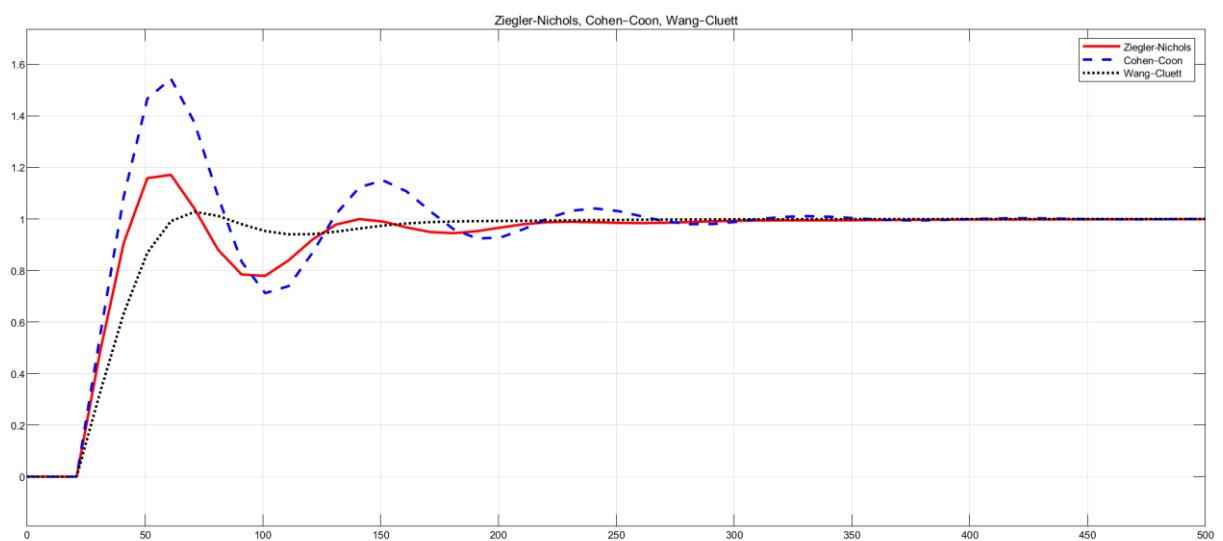
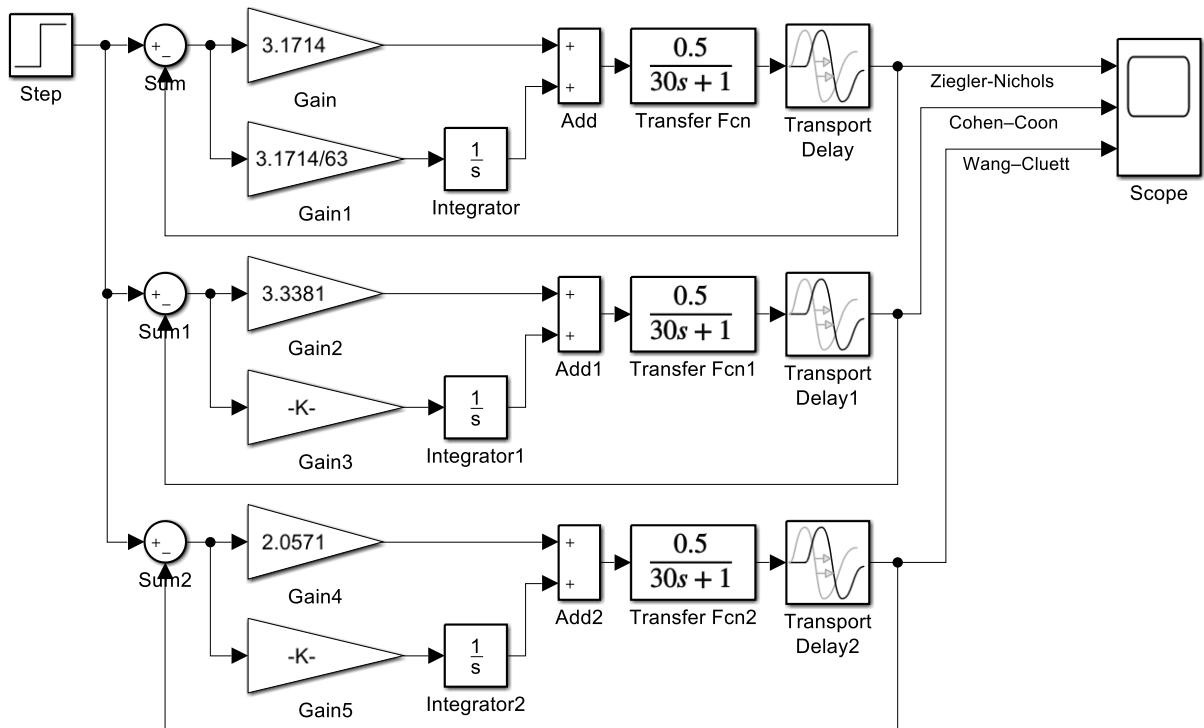
| | K_c | τ_I | τ_D |
|-----|-------------------------------|--|-------------------------------------|
| P | $\frac{0.13 + 0.51L}{K_{ss}}$ | | |
| PI | $\frac{0.13 + 0.51L}{K_{ss}}$ | $\frac{d(0.25 + 0.96L)}{0.93 + 0.03L}$ | |
| PID | $\frac{0.13 + 0.51L}{K_{ss}}$ | $\frac{d(0.25 + 0.96L)}{0.93 + 0.03L}$ | $\frac{d(-0.03 + 0.28L)}{0.25 + L}$ |

Bảng 4.4. Quy tắc Wang–Cluett cho đổi tượng có trễ với $L = \tau_M/d$

Theo các bảng quy tắc 4.2, 4.3, 4.4 ta nhận được tham số của bộ điều khiển PI như sau:

| | K_c | τ_I |
|-----------------|--------|----------|
| Ziegler–Nichols | 3.1714 | 63 |
| Cohen–Coon | 3.3381 | 32.7131 |
| Wang–Cluett | 2.0571 | 41.4811 |

- B4: thiết lập mô hình trên SIMULINK với tham số của bộ điều khiển PI như trên. Nhận xét kết quả?



4.4. Câu hỏi thực hành

Thực hiện chỉnh định tham số bộ điều khiển PID cho các quá trình ở Bài thực hành số 3-Mô hình hoá lý thuyết.

TÀI LIỆU THAM KHẢO

- [1] Hoàng Minh Sơn, Cơ sở điều khiển quá trình, Nhà xuất bản Bách khoa Hà Nội, 2009.
- [2] Nguyễn Văn Chí, giáo trình điều khiển các quá trình công nghệ, Nhà xuất bản Khoa học kỹ thuật, 2016.
- [3] Nguyễn Phùng Quang, MATLAB&SIMULINK dành cho kỹ sư điều khiển tự động, Nhà xuất bản Khoa học kỹ thuật, 2003.

Bài 5. Điều khiển quá trình sản xuất xi măng

5.1. Mục tiêu

Sau khi hoàn thành bài thực hành này, sinh viên có khả năng sử dụng được các công cụ trong phần mềm MATLAB để thiết kế, mô hình hóa và điều khiển quá trình sản xuất xi măng.

5.2. Nội dung thực hành

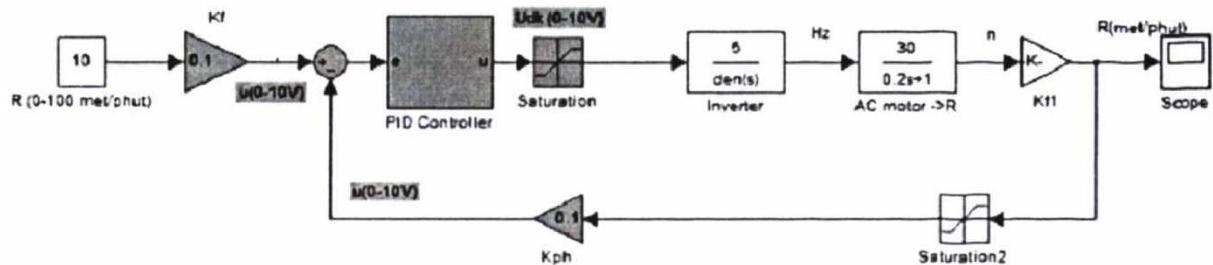
Thực hiện thiết kế, mô hình hóa và điều khiển quá trình sản xuất xi măng.

5.3. Các bước thực hành

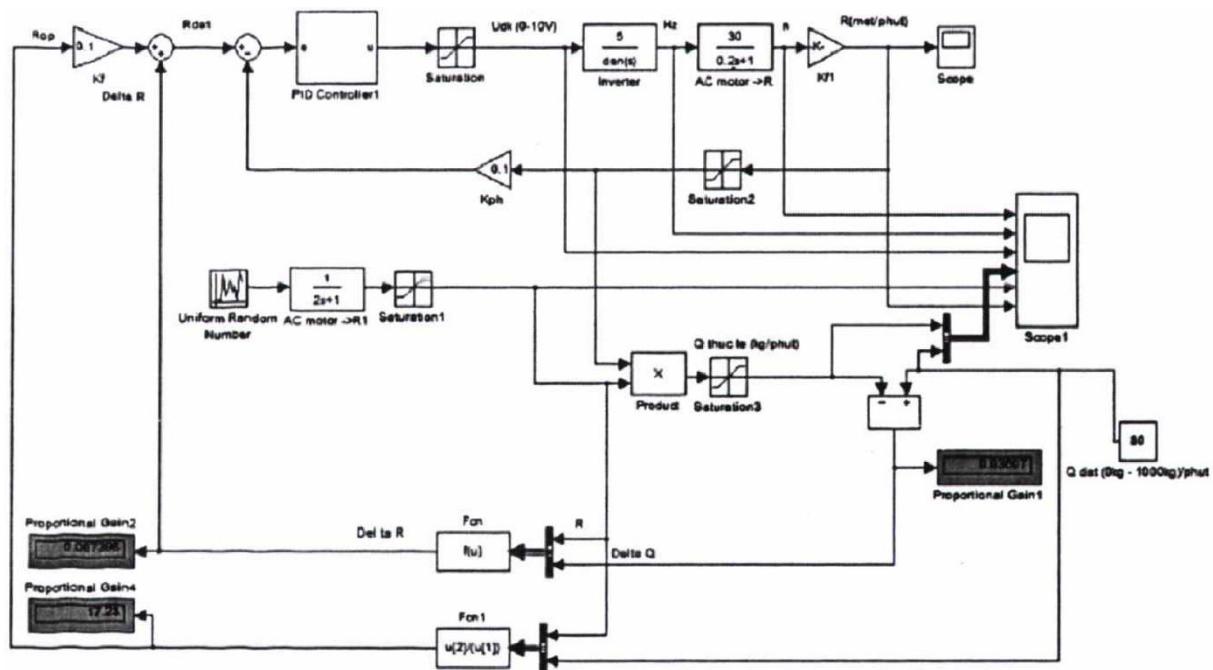
5.3.1 Điều khiển định lượng trên cân băng

Bài toán 5.1. Hãy mô hình hóa quá trình định lượng trên cân băng (mục 7.3.1, trang 358-362, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:

- Cấu trúc mạch vòng điều khiển tốc độ băng tải:

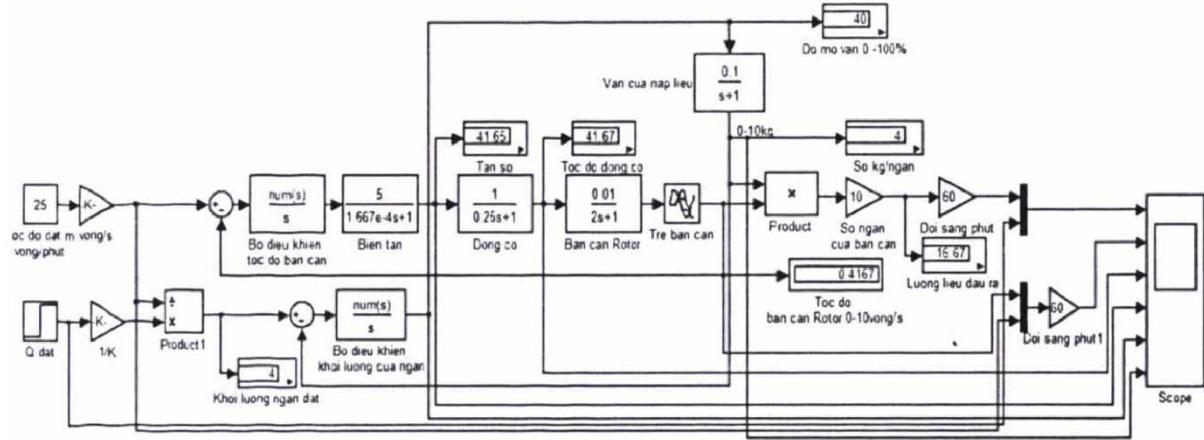


- Mô phỏng điều khiển cân băng định lượng:



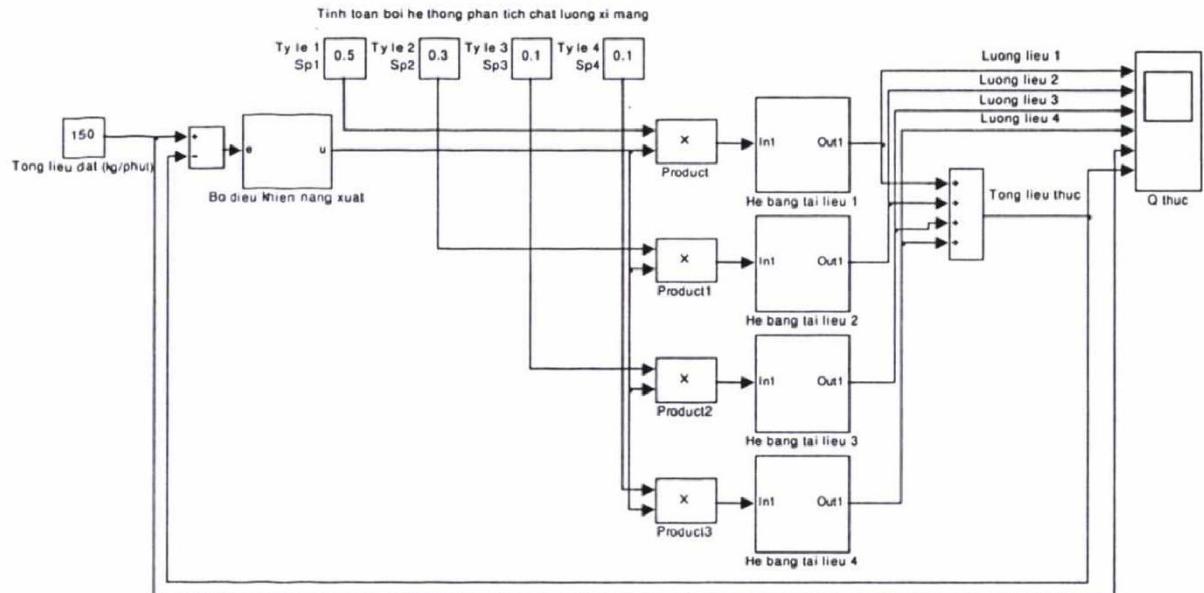
5.3.2 Điều khiển định lượng trên cân băng

Bài toán 5.2. Hãy mô hình hoá quá trình định lượng bằng cân rotor (mục 7.3.2, trang 362-365, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:



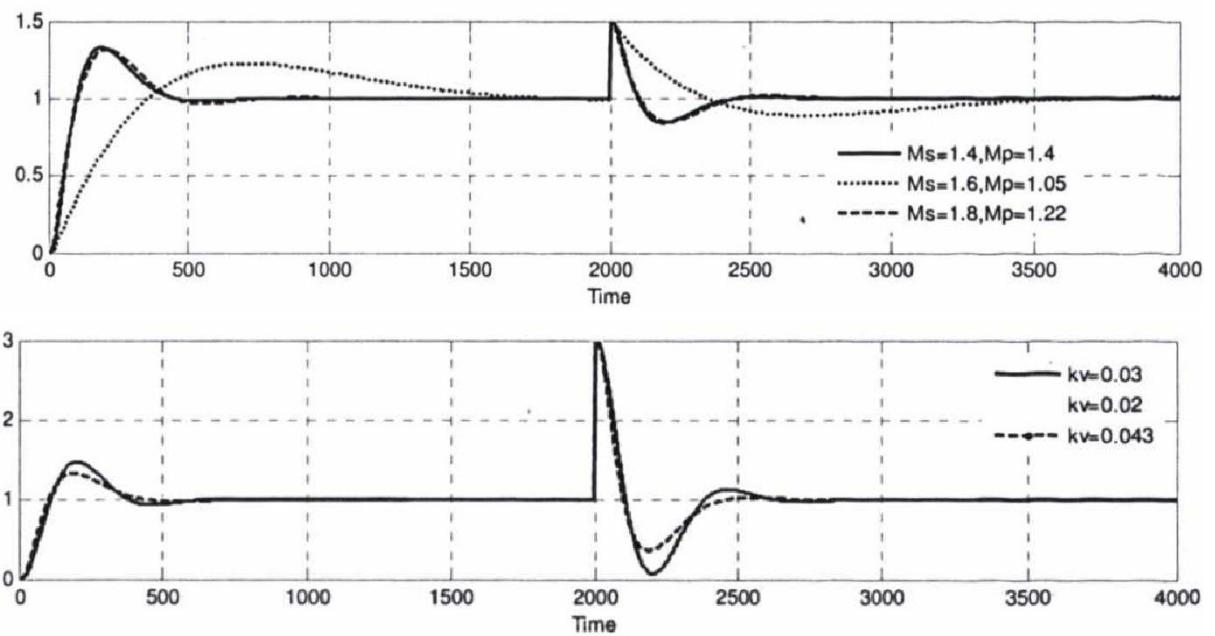
5.3.3 Điều khiển phối hợp nhiều thành phần

Bài toán 5.3. Hãy mô hình hoá quá trình phối hợp nhiều thành phần (mục 7.3.2, trang 366-368, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:



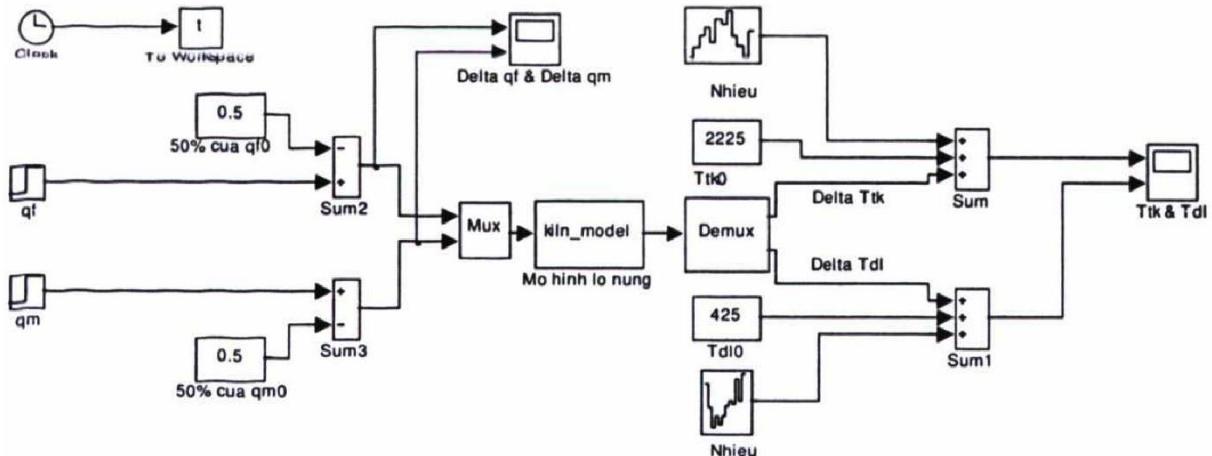
5.3.4 Thiết kế bộ điều khiển PID bền vững cho quá trình nghiên

Bài toán 5.4. Hãy thiết kế bộ điều khiển PID bền vững cho quá trình nghiên (mục 7.4.3, trang 373-378, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:



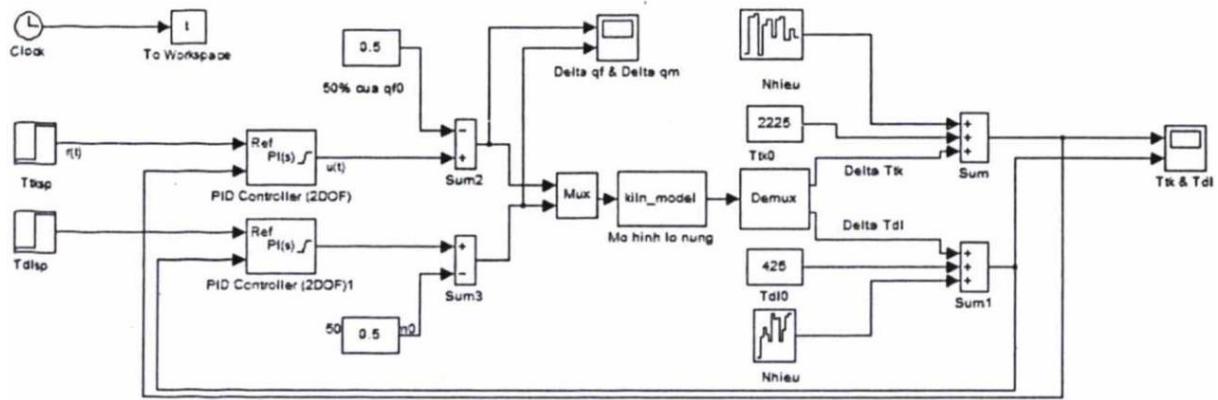
5.3.5 Mô hình quá trình truyền nhiệt trong lò nung clinke

Bài toán 5.5. Hãy mô hình hoá quá trình truyền nhiệt trong lò nung clinke (mục 7.5.3, trang 381-384, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:



5.3.6 Mô hình quá trình truyền nhiệt trong lò nung clinke

Bài toán 5.6. Hãy mô hình hoá quá trình điều khiển nhiệt độ lò nung clinke (mục 7.5.3, trang 381-384, giáo trình điều khiển các quá trình công nghệ, Nguyễn Văn Chí) trên SIMULINK như sau:



5.4. Câu hỏi thực hành

Thực hiện mô hình hoá quá trình điều khiển sản xuất xi măng (trang 351-386, giáo trình Điều khiển các quá trình công nghệ, Nguyễn Văn Chí).

TÀI LIỆU THAM KHẢO

- [1] Hoàng Minh Sơn, Cơ sở điều khiển quá trình, Nhà xuất bản Bách khoa Hà Nội, 2009.
- [2] Nguyễn Văn Chí, giáo trình điều khiển các quá trình công nghệ, Nhà xuất bản Khoa học kỹ thuật, 2016.
- [3] Nguyễn Phùng Quang, MATLAB&SIMULINK dành cho kỹ sư điều khiển tự động, Nhà xuất bản Khoa học kỹ thuật, 2003.