ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO THỰC HÀNH

MÔN MÔ HÌNH HÓA VÀ MÔ PHỎNG

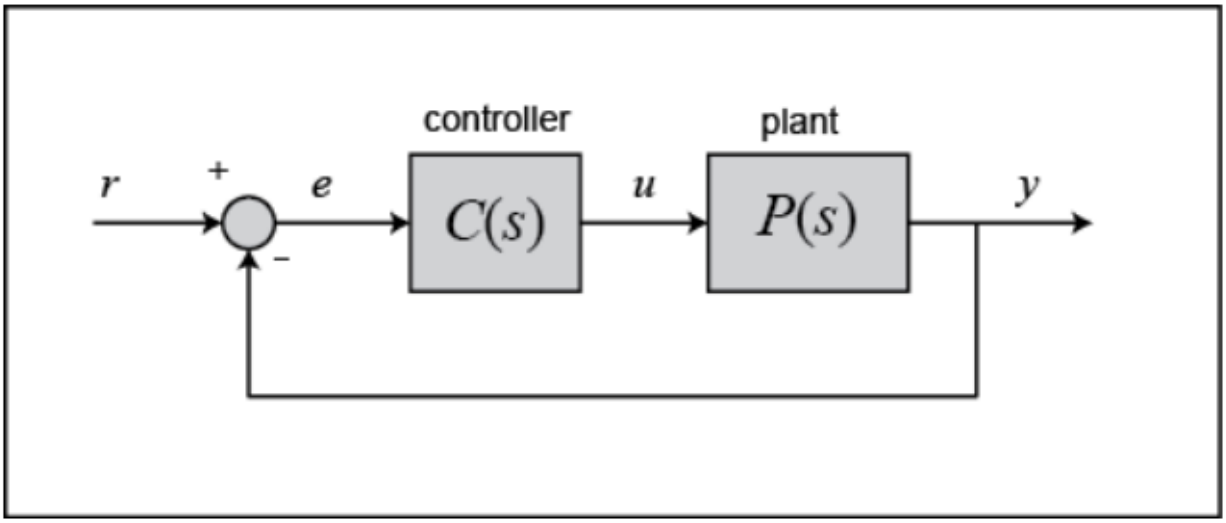
LAB 2: MODELING AND SIMULATION IN MATLAB/SIMULINK

PID CONTROL – MÔ PHỎNG CƠ CHẾ ĐIỀU KHIỂN TỰ ĐỘNG VỚI PID

Giảng viên hướng dẫn	**:**	**Nguyễn Hồng Thịnh**
Sinh viên thực hiện	**:**	**Hoàng Nhật Minh**
Mã sinh viên	**:**	**21020696**
Lớp	**:**	**ELT2031E 21**

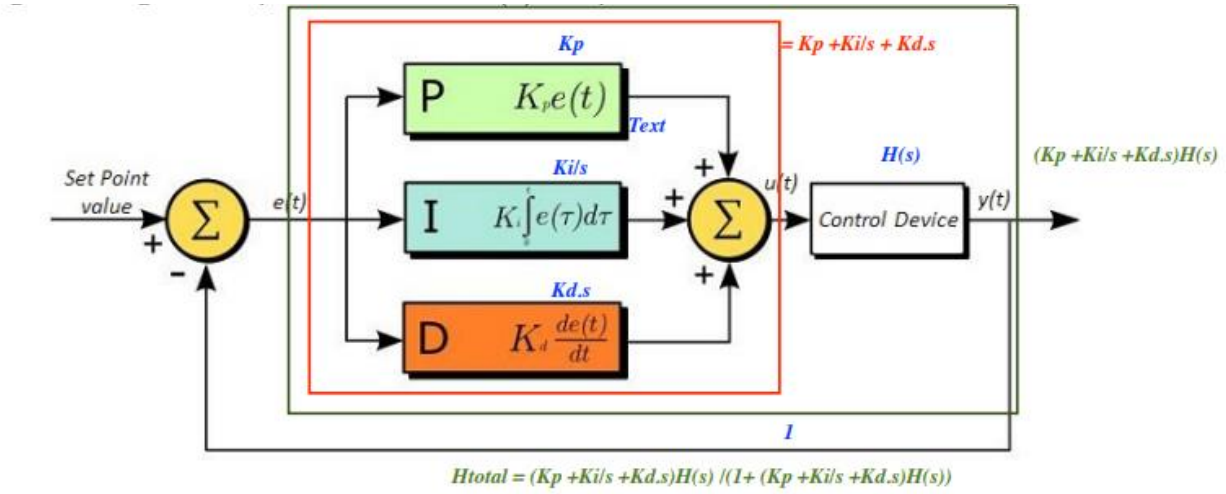
1. Pre-lab

1.1. Automatic control with feedback system



Hình 1: Unity-feedback system

P(s) là hàm truyền của đối tượng cần điều khiển. r là đầu vào của hệ thống [đầu ra mong muốn], y là đầu ra thực. Khi đó e là sai số giữa giá trị thực và giá trị mong muốn.



Hình 2: PID controler

Thông thường các bộ điều khiển C(s) được cấu thành từ 3 thành phần:

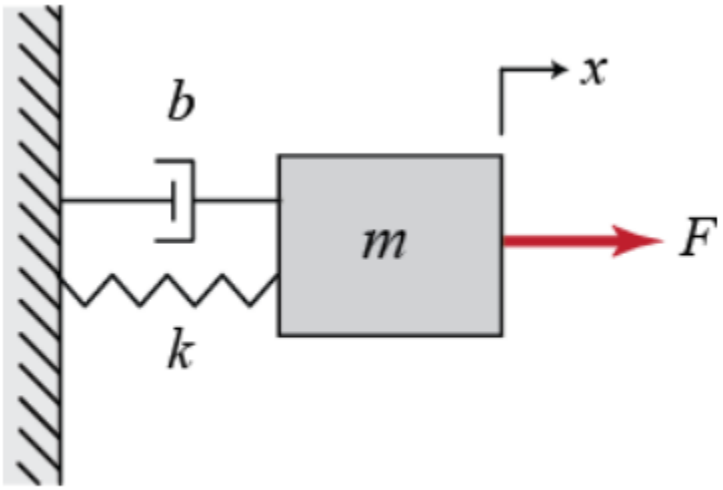
- P (Proportional controller, hệ số tỉ lệ), đánh giá tác động sai số hiện tại
- I (Integral controller, hệ số tích phân), đánh giá ảnh hưởng của tổng các sai số trong quá khứ

- D (Derivative controller, hệ số vi phân), xác định tác động của tốc độ biến đổi sai số

Phương trình mô tả hàm truyền của bộ PID nói chung là:

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

1.2. Hệ lò xo trọng lượng giảm động – Mass-spring-damper system



Hình 3: Mass-spring-damper system

Phương trình vi phân mô tả hệ thống trên là:

$$m\frac{d^2x}{dt} + b\frac{dx}{dt} + kx = F$$

Sử dụng biến đổi Laplace $\Leftrightarrow ms^2X(s) + bsX(s) + kX(s) = F(s)$

$\rightarrow \frac{X(s)}{F(s)} = \frac{1}{ms^2+bs+k} = H(s)$

- Sử dụng các giá trị tham số m = 1 kg, b = 10 N s/m, k = 20 N/m và F = 1 N, xác định hàm truyền hệ thống trên

$$H(s) = \frac{1}{ms^2 + bs + k} = \frac{1}{s^2 + 10s + 20}$$

2. Lab

2.1. Open-loop system

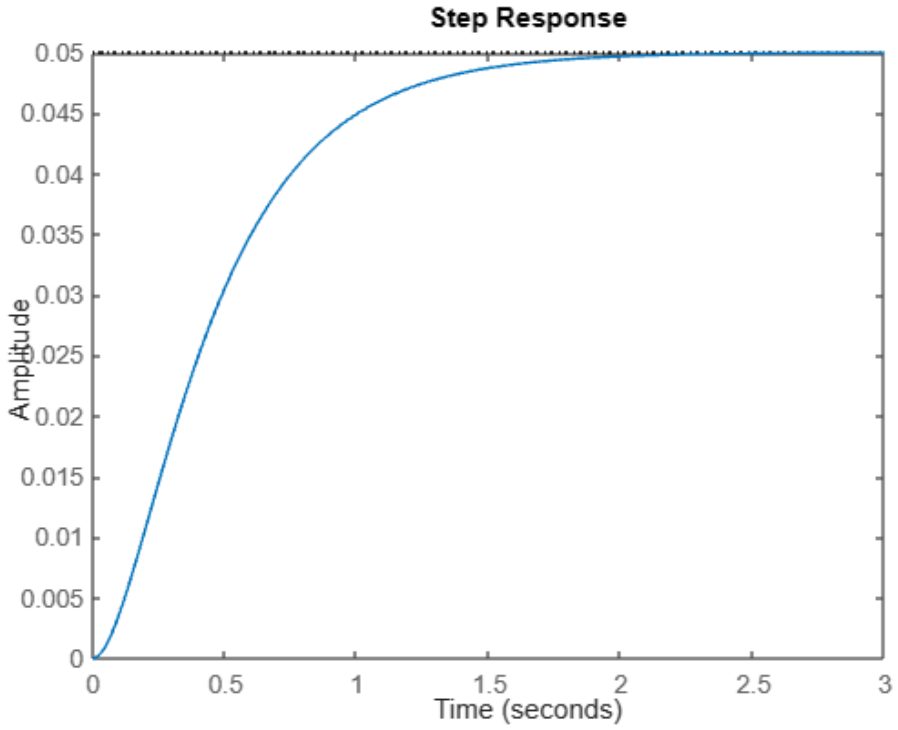
2.1.1. Tìm hiểu cách sử dụng hàm tf

- Sử dụng hàm tf để tạo các mô hình hàm truyền có giá trị thực hoặc có giá trị phức hoặc để chuyển đổi các mô hình hệ thống động sang dạng hàm truyền.
- Hàm tf là hàm biểu diễn miền tần số của các hệ thống tuyến tính bất biến theo thời gian

  $$Sys(s) = \frac{num(s)}{den(s)}$$
- Mô hình tf có thể biểu diễn các hàm truyền SISO hoặc MIMO theo thời gian liên tục hoặc rời rạc.
- Syntax: sys = tf(numerator, denominator)

2.1.2. Viết code Matlab thực hiện mô phỏng hàm truyền của hệ thống trên; vẽ đáp ứng step response của nó.

```
1. m = 1;
2. b = 10;
3. k = 20;
4. num = 1;
5. den = [m, b, k];
6. sys = tf(num,den);
7. step(sys)
```



2.1.3. Xác định các giá trị steady-state, rise time, settling time (đạt steady-state)

- Steady-state: từ 2.5s đổ đi
- Rise time: khoảng từ 0.1s đến 0.5s
- Settling time: khoảng từ 0s đến 2.5s

2.2. Proportional Control

2.2.1. Chứng minh lại hàm truyền mới của cả hệ là:

$$T(s) = \frac{X(s)}{R(s)} = \frac{K_p}{s^2 + 10s + (20 + K_p)}$$

*Chứng minh:

Có hàm truyền trên:

$$H(s) = \frac{1}{ms^2 + bs + k} = \frac{1}{s^2 + 10s + 20}$$

Với công thức phản hồi âm:

$$T(s) = \frac{K_p H(s)}{1 + K_p H(s)} = \frac{A}{B}$$

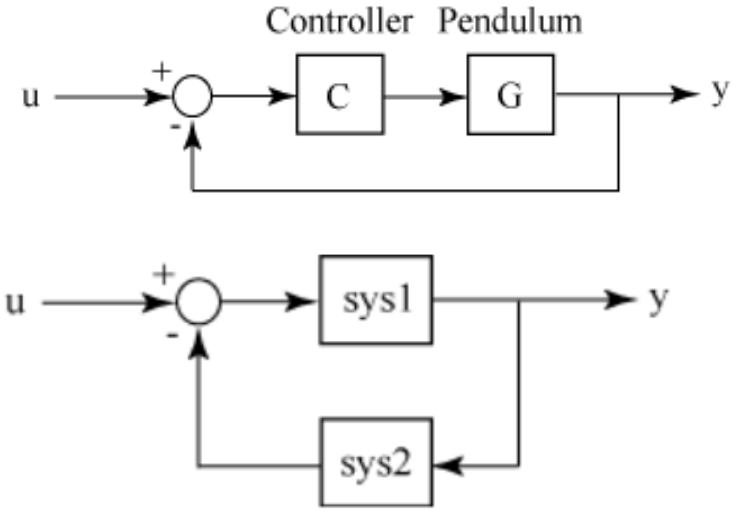
Có: $A = K_p . \frac{1}{s^2+10s+20} = \frac{K_p}{s^2+10s+20}$

Có: $B = 1 + K_p H(s) = 1 + \frac{K_p}{s^2+10s+20}$

➔ $T(s) = \frac{\frac{K_p}{s^2+10s+20}}{1+\frac{K_p}{s^2+10s+20}} = \frac{K_p}{s^2+10s+(20+K_p)}$ (dpcm)

2.2.2. Tìm hiểu cách sử dụng hàm feedback của Matlab

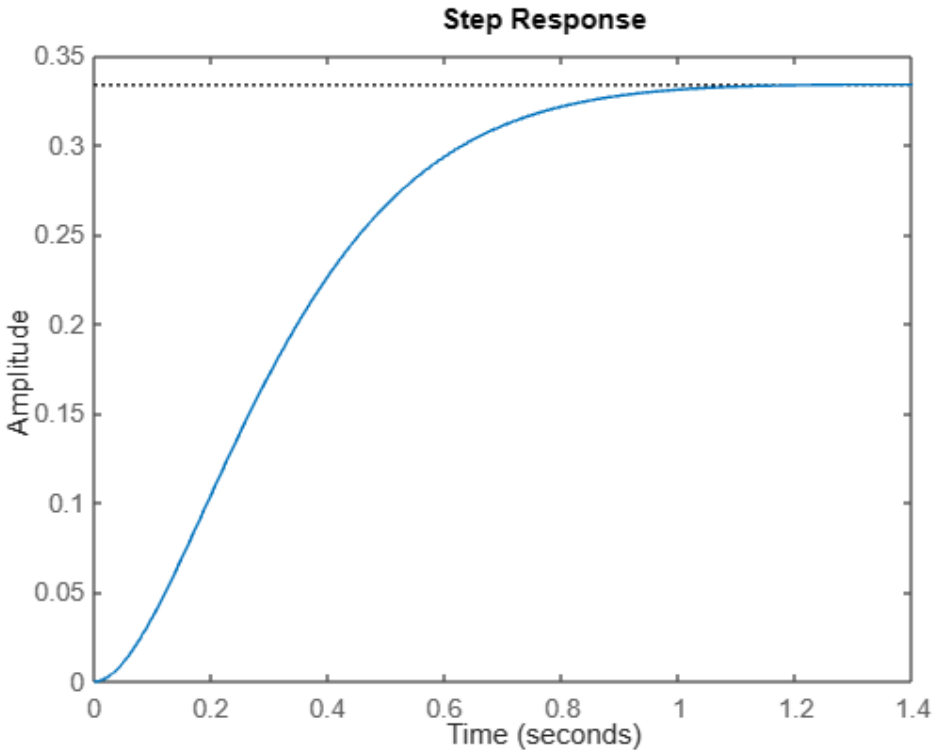
- Syntax: sys = feedback(sys1, sys2)
- Hàm feedback dùng để trả về một model object sys cho kết nối phản hồi âm của model object sys1, sys2



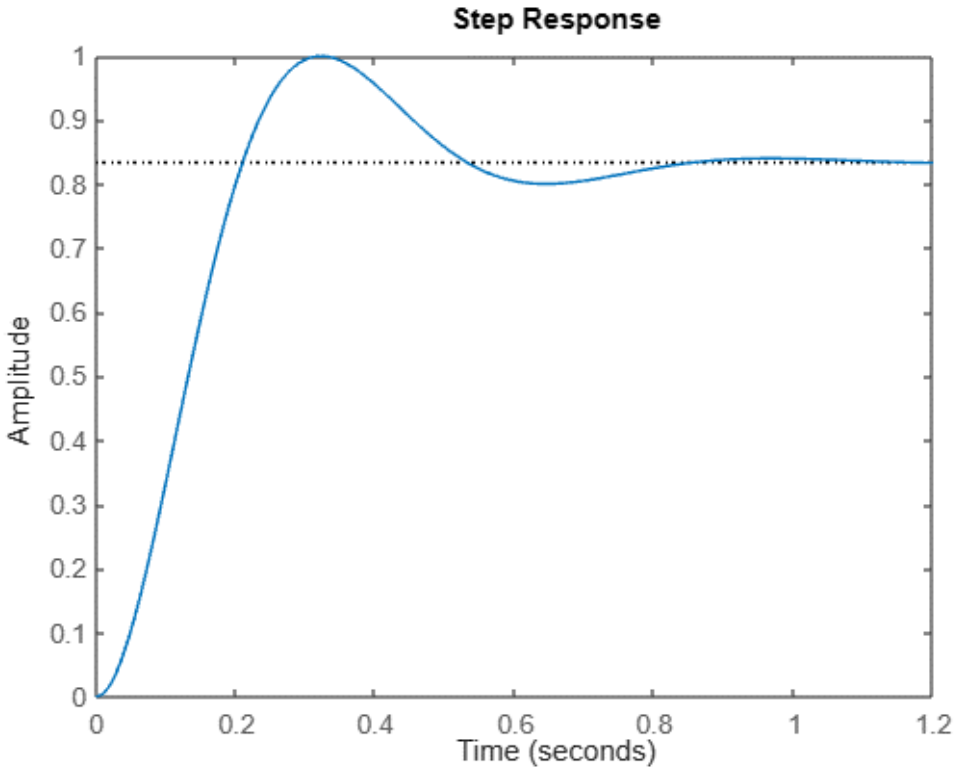
2.2.3. Lặp lại quá trình thực nghiệm như bài trước, thay đổi giá trị Kp từ 10 đến 500 và nhận xét

```
1.  m = 1;
2.  b = 10;
3.  k = 20;
4.  k_p = ;
5.  num2 = m;
6.  den2 = [m, b, k];
7.  sys1 = tf(k_p);
8.  sys2 = tf(num2, den2);
9.  sys = feedback(sys1*sys2, 1);
10. step(sys)
```

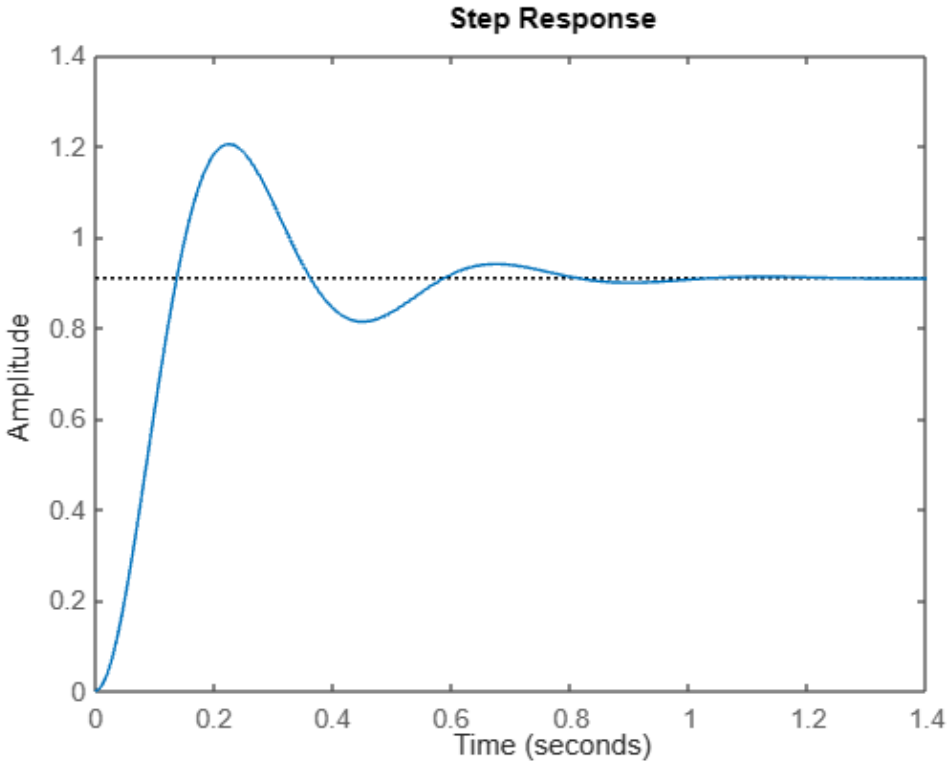
- $K_p = 10$:



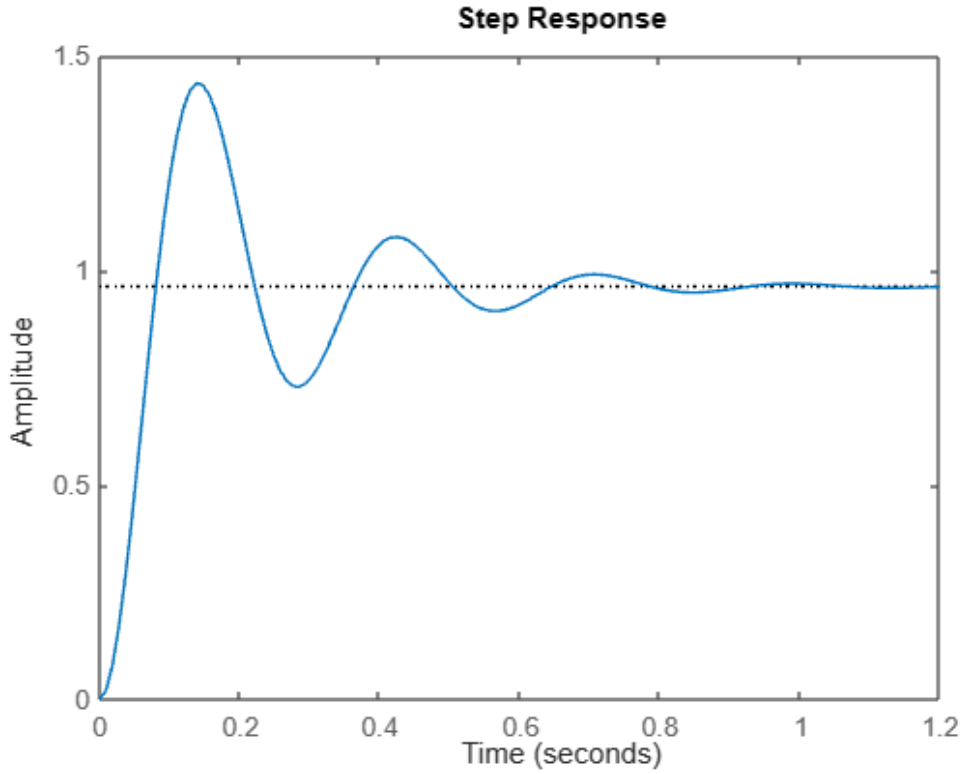
- $K_p = 100$:



- $K_p = 200$:



- $K_p = 500$:



- Nhận xét:

- Càng tăng K_p thì đáp ứng đầu ra càng mấp mô, nhiều gợn sóng

- Càng tăng K_p thì Rise time càng giảm, Settlling time càng giảm, Overshoot càng tăng, steady-state error càng tăng.

2.2.4. Kp có thể nhận giá trị âm được không? Nhận xét

Kp có thể nhận giá trị âm trong Proportional Control. Giá trị Kp âm sẽ tạo ra một tín hiệu điều khiển có dấu ngược với lỗi. Điều này có thể được sử dụng để điều khiển hệ thống theo hướng ngược lại với hướng của lỗi.

2.2.5. Đánh giá ảnh hưởng của Kp đến các tham số rise time, steady-state error, overshoot, và settling time

Càng tăng K_p thì Rise time càng giảm, Settlling time càng giảm, Overshoot càng tăng, steady-state error càng tăng

2.3. Proportional-Derivative Control (PD)

2.3.1. Chứng minh lại hàm truyền mới của cả hệ là:

$$T(s) = \frac{X(s)}{R(s)} = \frac{K_d s + K_p}{s^2 + (10 + K_d)s + (20 + K_p)}$$

*Chứng minh:

Có:

- $K_p.e(t) \rightarrow K_p$
- $K_d.\frac{de(t)}{dt} \rightarrow K_d.s$

➔ Hàm truyền bộ PD: $C(s) = K_p + K_d.s$

Với công thức phản hồi âm:

$$T(s) = \frac{(K_p + K_d.s)H(s)}{1 + (K_p + K_d.s)H(s)} = \frac{A}{B}$$

Có: $A = (K_p + K_d.s).\frac{1}{s^2+10s+20} = \frac{(K_p+K_d.s)}{s^2+10s+20}$

Có: $B = 1 + (K_p + K_d.s)H(s) = 1 + \frac{(K_p+K_d.s)}{s^2+10s+20}$

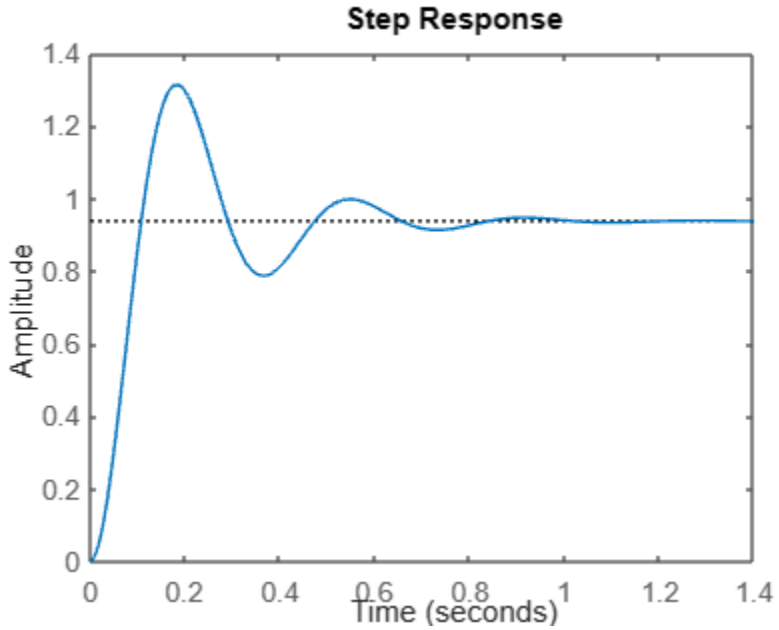
➔ $T(s) = \frac{\frac{(K_p+K_d.s)}{s^2+10s+20}}{1+\frac{(K_p+K_d.s)}{s^2+10s+20}} = \frac{(K_p+K_d.s)}{s^2+(10+K_d)s+(20+K_p)}$ (dpcm)

2.3.2. Lặp lại quá trình thực nghiệm như bài trước, với Kp=300 và Kd thay đổi

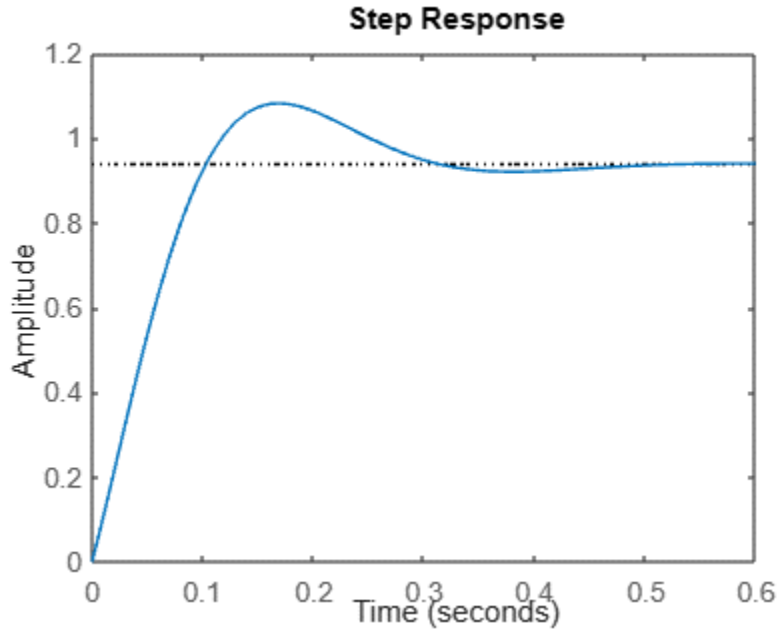
```
1. m = 1;
2. b = 10;
3. k = 20;
4. k_p = 300; k_d= ;
```

```
num2 = m;
den2 = [m, b, k];
sys1 = tf([k_d k_p]);
sys2 = tf(num2, den2);
sys = feedback(sys1*sys2, 1);
step(sys)
```

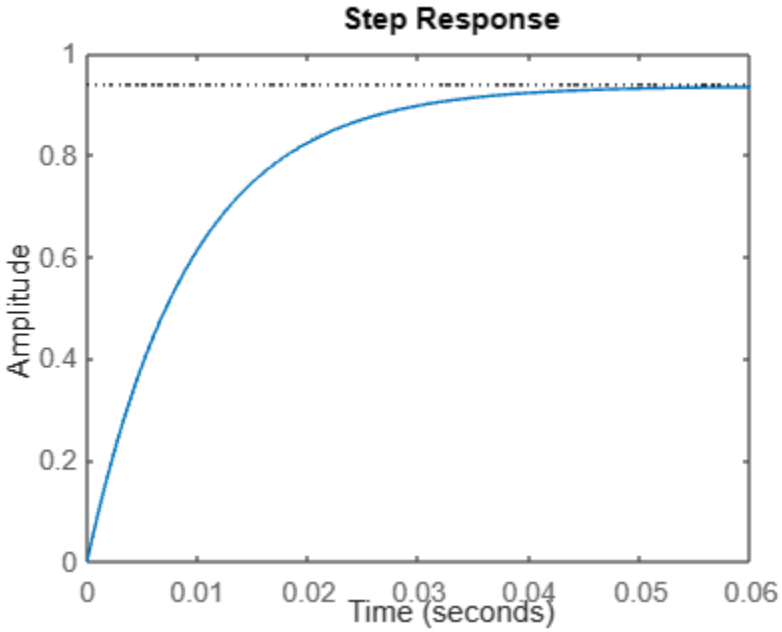
- $K_d = 0$:



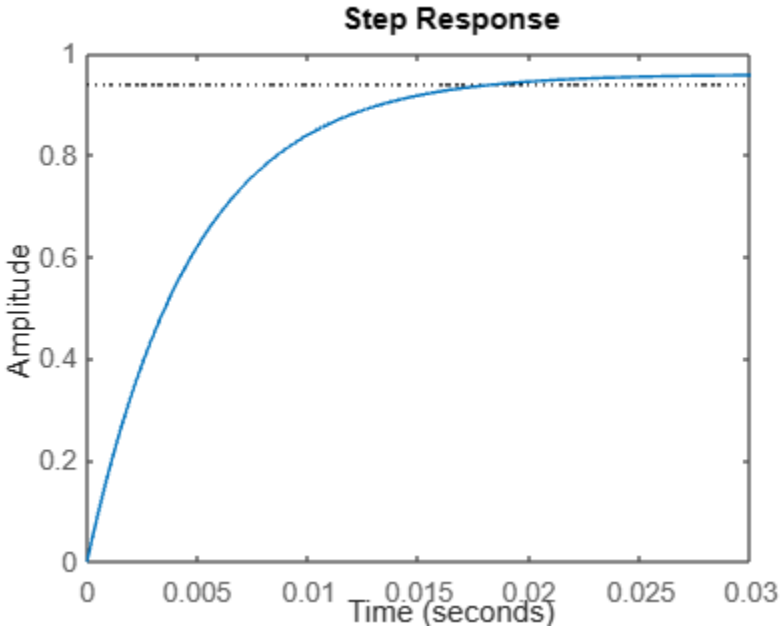
- $K_d = 10$:



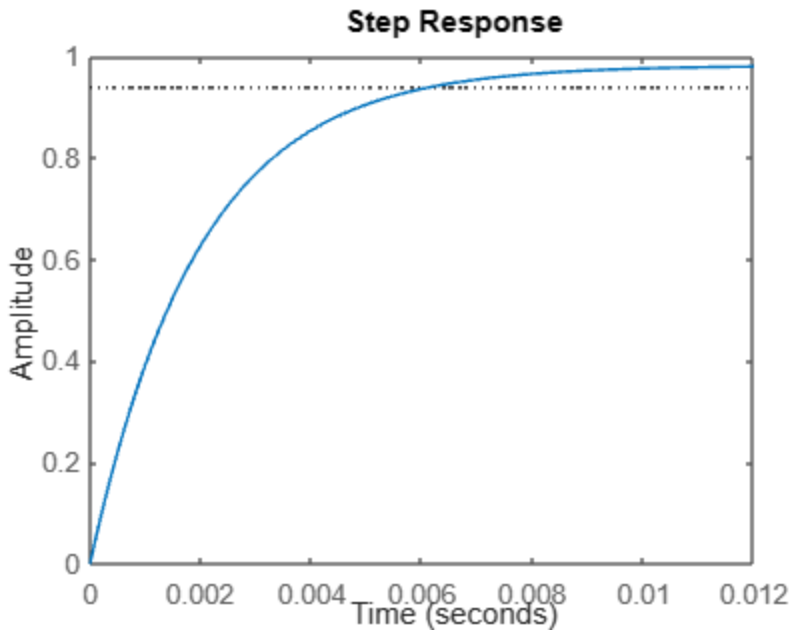
- $K_d = 100$:



- $K_d = 200$:



- $K_d = 500$:



2.3.3. So sánh kết quả khi sử dụng thêm Kd

- Khi K_d nhỏ hơn hoặc bằng 10 thì Rise time, Settling time khá ngắn
- Khi K_d từ 100 đổ đi thì Rise time, Settling time giảm rất nhiều
- So với chỉ dùng mỗi K_p (hoạt động tốt khi tăng K_p), thì khi thêm K_d sẽ hoạt động tốt hơn ở mức K_d lớn.

2.4. Proportional-Integral Control (PI)

2.4.1. Chứng minh lại hàm truyền mới của cả hệ là:

$$T(s) = \frac{X(s)}{R(s)} = \frac{K_d s + K_i}{s^3 + 10s^2 + (20 + K_p)s + K_i}$$

*Chứng minh:

Có:

- $K_p . e(t) \rightarrow K_p$
- $K_i . \int_0^1 e(\tau) d\tau \rightarrow \frac{K_i}{s}$

➔ Hàm truyền bộ PI: $C(s) = K_p + \frac{K_i}{s}$

Với công thức phản hồi âm:

$$T(s) = \frac{(K_p + \frac{K_i}{s})H(s)}{1 + (K_p + \frac{K_i}{s})H(s)} = \frac{A}{B}$$

Có: $A = (K_p + \frac{K_i}{s}).\frac{1}{s^2+10s+20} = \frac{(K_p+\frac{K_i}{s})}{s^2+10s+20}$

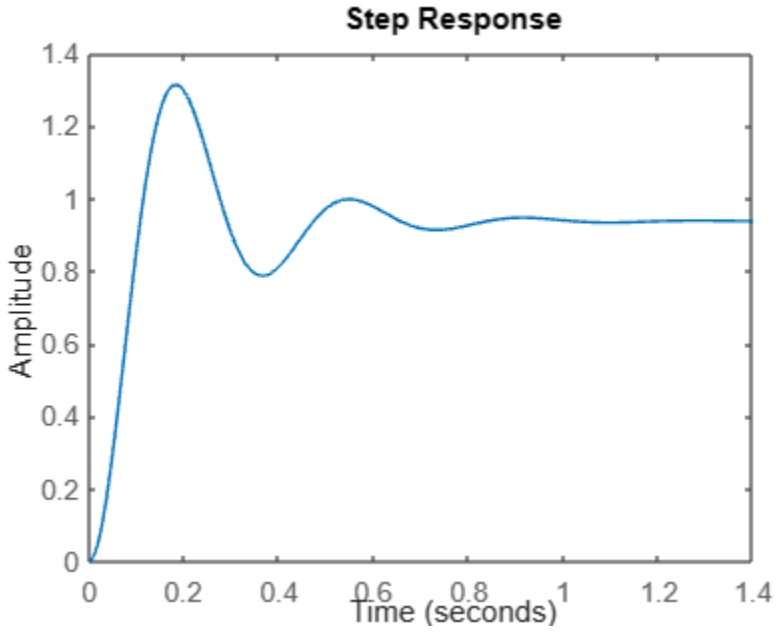
Có: $B = 1 + (K_p + \frac{K_i}{s})H(s) = 1 + \frac{(K_p+\frac{K_i}{s})}{s^2+10s+20}$

➔ $T(s) = \frac{\frac{(K_p+\frac{K_i}{s})}{s^2+10s+20}}{1+\frac{(K_p+\frac{K_i}{s})}{s^2+10s+20}} = \frac{(K_p+\frac{K_i}{s})}{s^3+10s^2+(20+K_p)s+K_i}$ (dpcm)

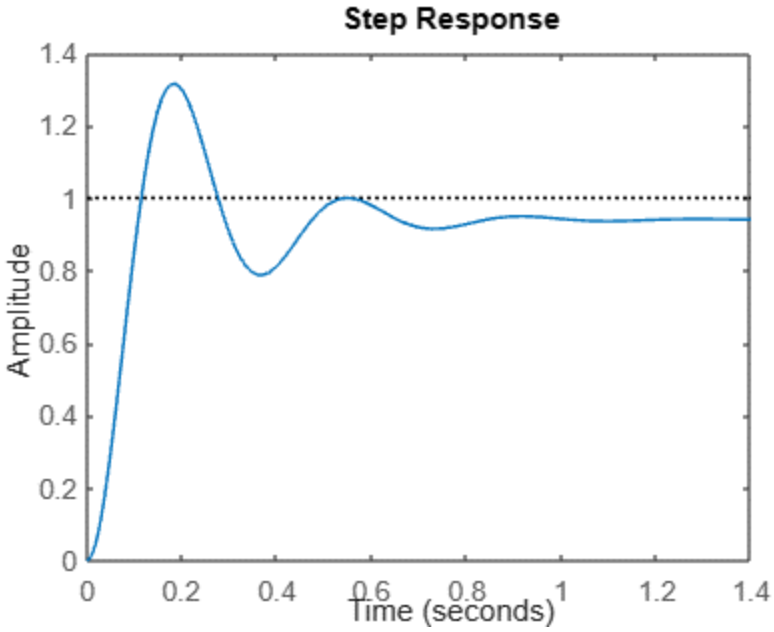
2.4.2. Lặp lại quá trình thực nghiệm như bài trước, cố định Kp=300, thay đổi giá trị Ki và nhận xét

```
1.  m = 1;
2.  b = 10;
3.  k = 20;
4.  k_p = 300; k_i = ;
5.  num2 = m;
6.  den2 = [m, b, k];
7.  sys1 = tf([k_p k_i], [1 0]);
8.  sys2 = tf(num2, den2);
9.  sys = feedback(sys1*sys2, 1);
10. step(sys)
```

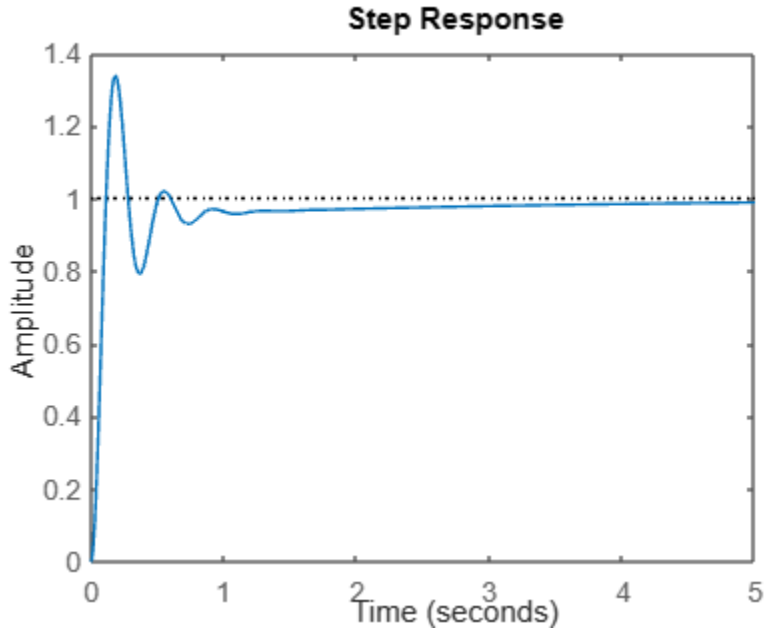
- $K_i = 0$:



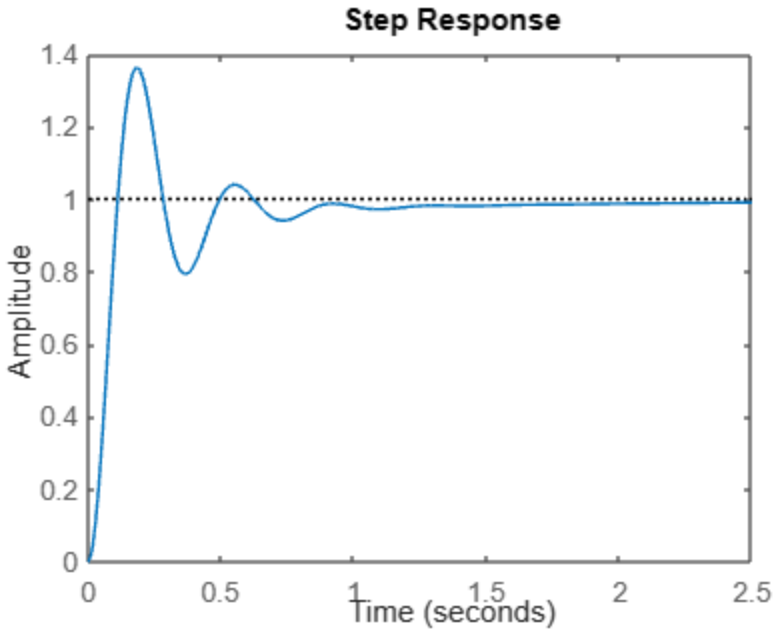
- $K_i = 10$:


Step Response
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
1.4

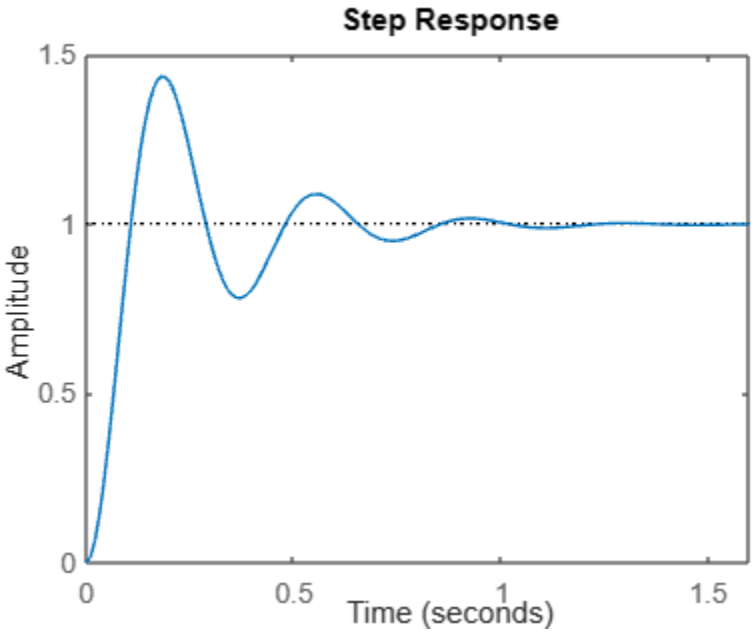
- $K_i = 100$:


Step Response
Amplitude
Time (seconds)
0
1
2
3
4
5
0.2
0.4
0.6
0.8
1
1.2
1.4

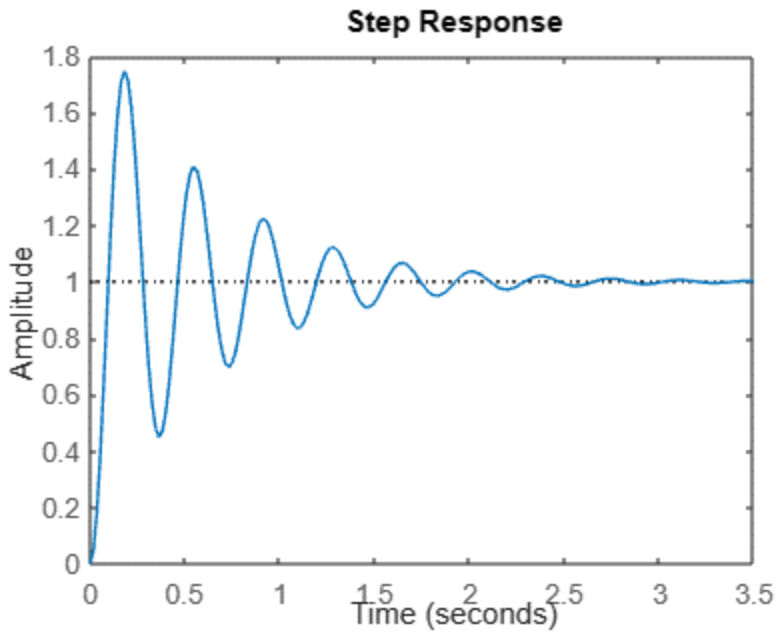
- $K_i = 200$:



- $K_i = 500$:



- $K_i = 2000$:



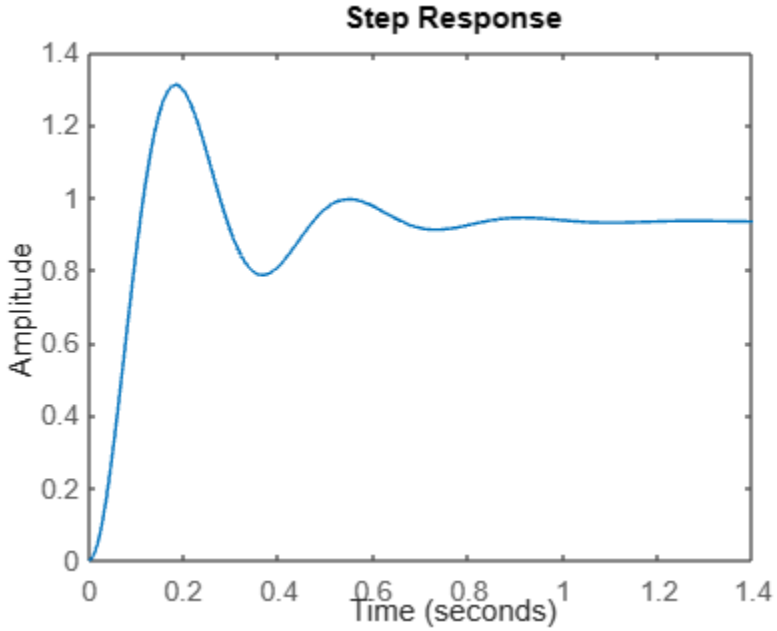
- Nhận xét:

- Khi K_i nhỏ hơn hoặc bằng 10 thì Rise time và Settling time khá ngắn.
- Khi K_i tăng thì Rise time giảm nhưng Overshoot và Settling time lại tăng lên khá nhiều.
- Khi K_i tăng rất nhiều thì Rise time giảm cực nhanh nhưng Overshoot và Settling time lại tăng lên cực nhiều và steady-state error không còn tồn tại.

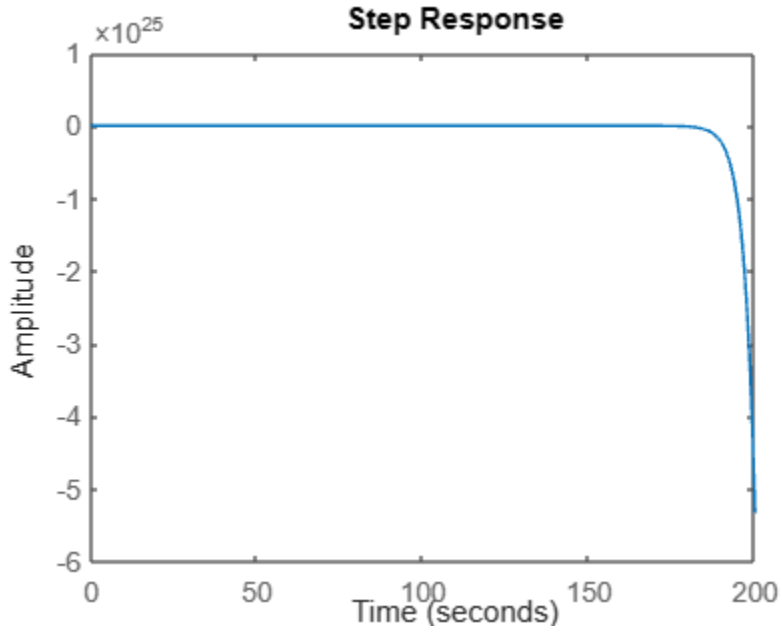
2.4.3. Ki có thể nhận giá trị âm được không? Nhận xét

K_i có thể nhận giá trị âm

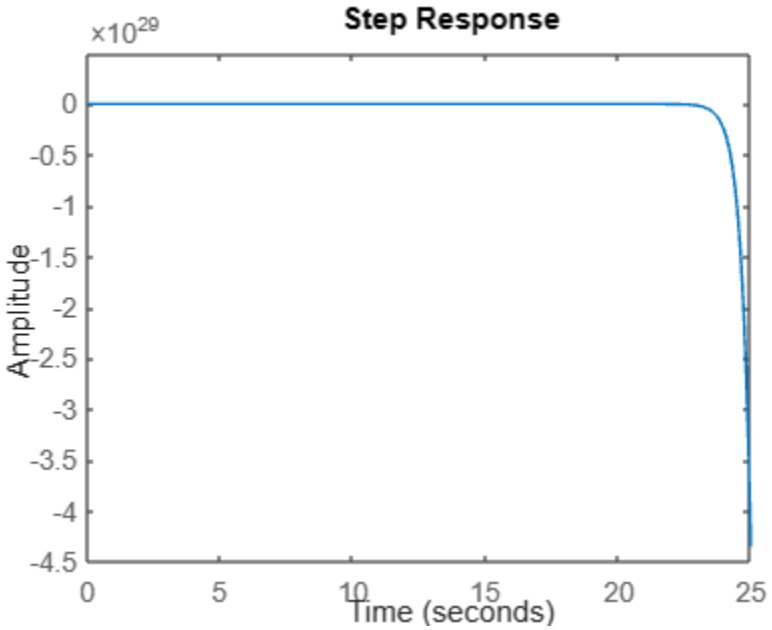
- $K_i = -10$:



- $K_i = -100$:



- $K_i = -1000$:



2.4.4. Đánh giá ảnh hưởng của Ki đến các tham số rise time, steady-state error, overshoot, và settling time

- K_i tăng thì Rise time giảm nhưng Overshoot và Settling time tăng, steady-state error có thể không tồn tại nếu tăng quá mức.

2.5. Proportional-Integral-Derivative Control (PID)

2.5.1. Chứng minh lại hàm truyền mới của cả hệ là:

$$T(s) = \frac{X(s)}{R(s)} = \frac{K_d s^2 + K_p s + K_i}{s^3 + (10 + K_d)s^2 + (20 + K_p)s + K_i}$$

*Chứng minh:

Có:

- $K_p . e(t) \rightarrow K_p$
- $K_i . \int_0^1 e(\tau) d\tau \rightarrow \frac{K_i}{s}$
- $K_d . \frac{de(t)}{dt} \rightarrow K_d . s$

➔ Hàm truyền bộ PID: $C(s) = K_p + K_d s + \frac{K_i}{s}$

Với công thức phản hồi âm: $T(s) = \frac{(K_p + K_d s + \frac{K_i}{s})H(s)}{1 + (K_p + K_d s + \frac{K_i}{s})H(s)} = \frac{A}{B}$

Có: $A = (K_p + K_d s + \frac{K_i}{s}) . \frac{1}{s^2 + 10s + 20} = \frac{(K_p + K_d s + \frac{K_i}{s})}{s^2 + 10s + 20}$

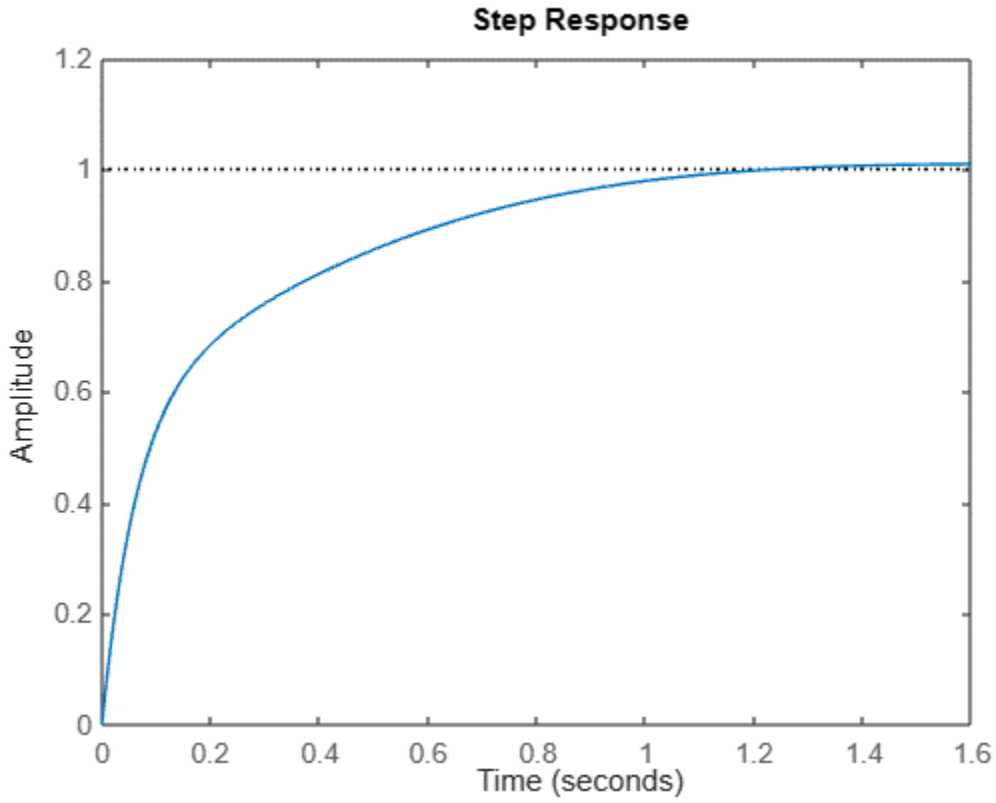
Có: $B = 1 + (K_p + K_d s + \frac{K_i}{s})H(s) = 1 + \frac{(K_p + K_d s + \frac{K_i}{s})}{s^2 + 10s + 20}$

➔ $T(s) = \frac{\frac{(K_p + K_d s + \frac{K_i}{s})}{s^2 + 10s + 20}}{1 + \frac{(K_p + K_d s + \frac{K_i}{s})}{s^2 + 10s + 20}} = \frac{(K_p + K_d s + \frac{K_i}{s})}{s^3 + (10 + K_d)s^2 + (20 + K_p)s + K_i}$ (dpcm)

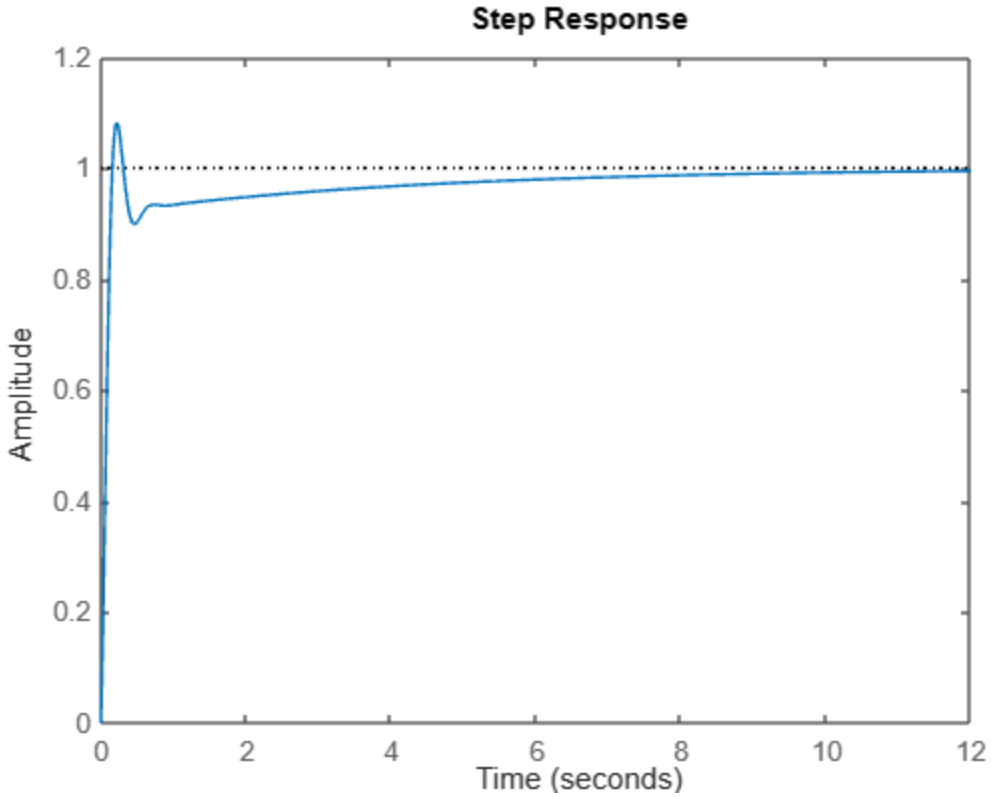
2.5.2. Lặp lại quá trình thực nghiệm như bài trước, thay đổi Kp, Kd và Ki và so sánh step-response trong các trường hợp. Xác định bộ giá trị PID em nhận xét là tốt. Giải thích

```
1.  m = 1;
2.  b = 10;
3.  k = 20;
4.  k_p = ; k_i = ; k_d = ;
5.  num2 = m;
6.  den2 = [m, b, k];
7.  sys1 = tf([k_d k_p k_i], [1 0]);
8.  sys2 = tf(num2, den2);
9.  sys = feedback(sys1*sys2, 1);
10. step(sys)
```

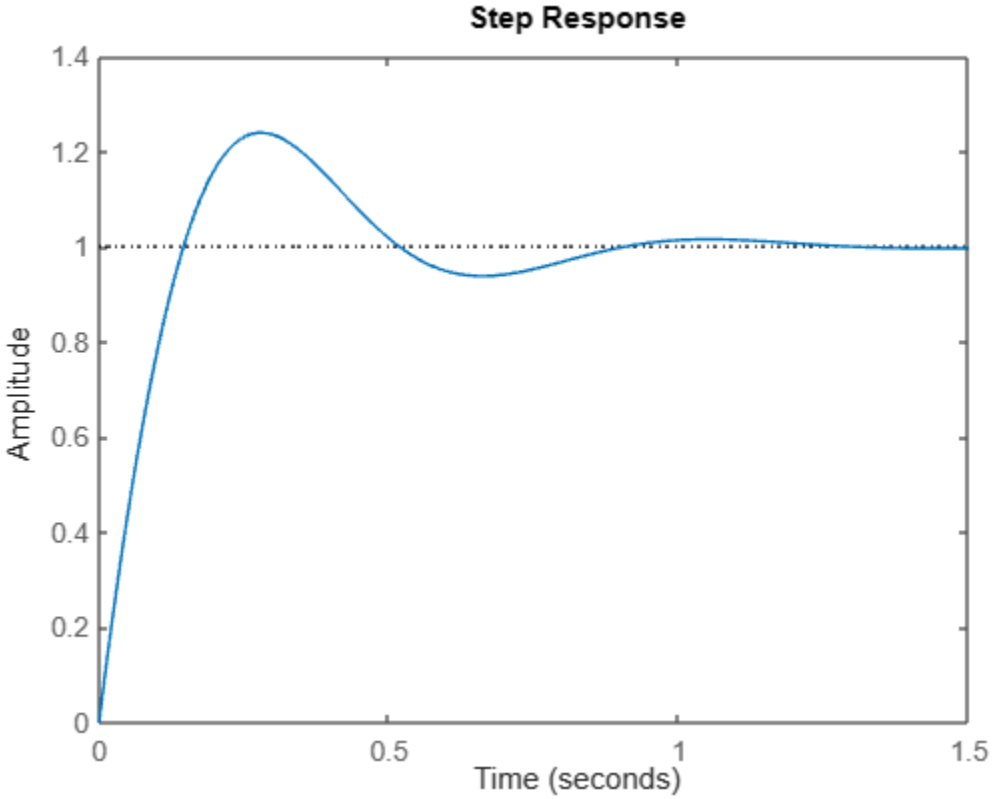
- $K_p = 50;\ K_i = 100;\ K_d = 10$:



- $K_p = 200;\ K_i = 50;\ K_d = 5$:



- $K_p = 150;\ K_i = 1000;\ K_d = 10$:



- Nhận xét:

- Bộ PID đầu tiên có Rise time lớn nhất, Overshoot~0, Settling time>1
- Bộ PID thứ 2 có Rise time khá ổn, Overshoot<1, Settling time khá lớn
- Bộ PID cuối cùng có Rise time nhỏ nhất, Overshoot>1, Settling time<1

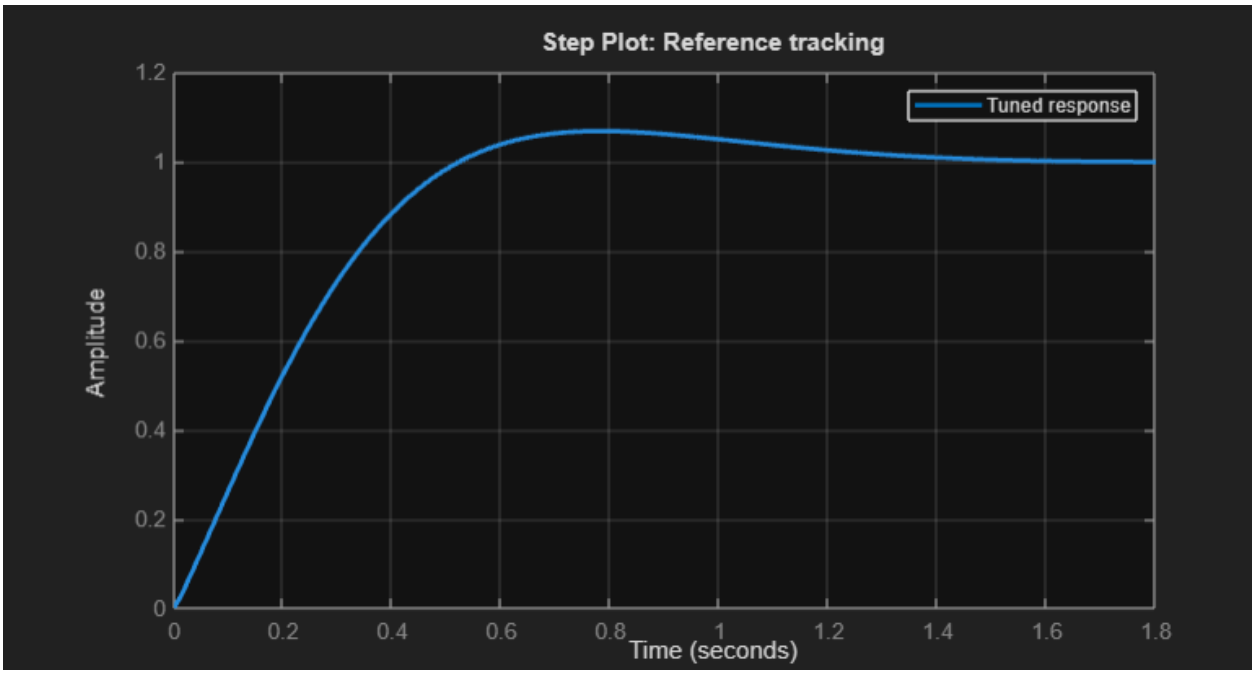
➔ Bộ PID cuối cùng sẽ là tốt nhất

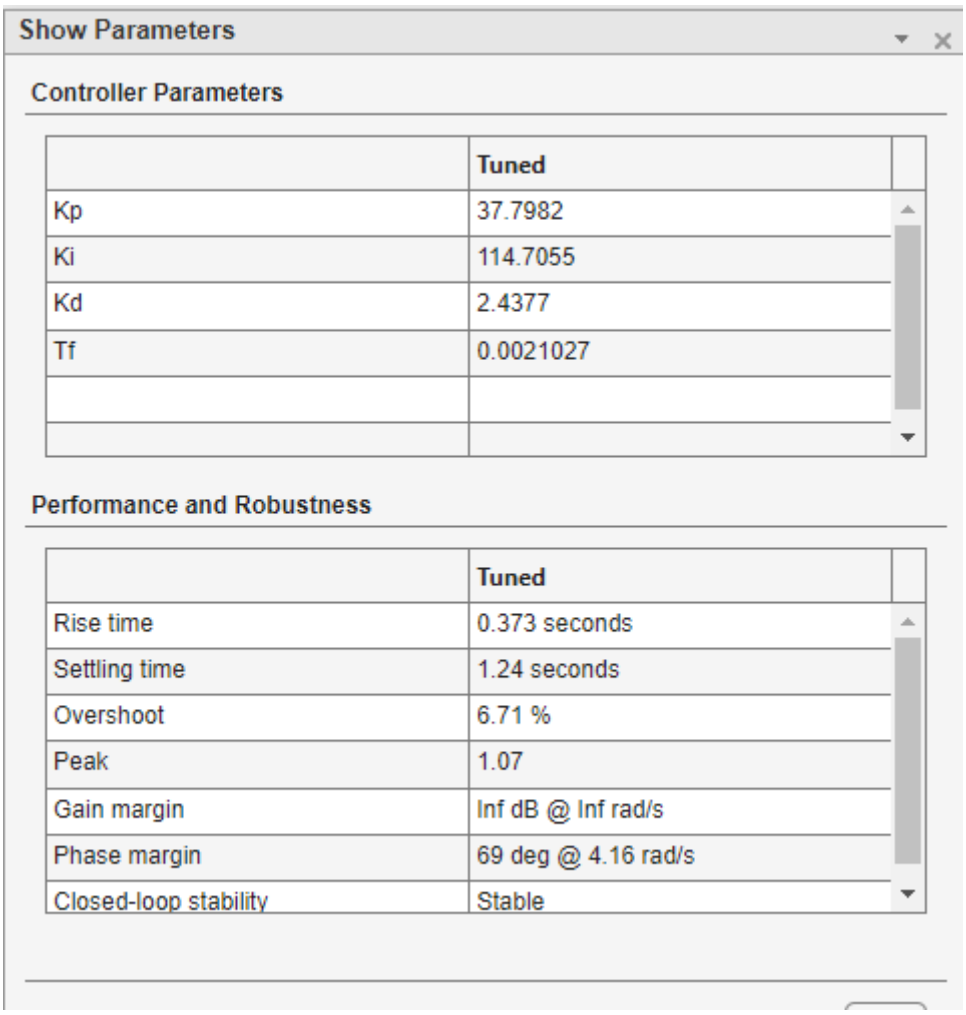
2.5.3. Lập bảng ảnh hưởng của Kp, Kd, Ki lên các đặc trưng lối ra theo mẫu:

Controller	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
Kp	Decrease	Increase	Increase	Increase
Kd	Decrease	Decrease	Decrease	Decrease
Ki	Decrease	Increase	Small Increase	Eliminate

2.6. Automatic PID Tuning

```
1. m = 1;
2. b = 10;
3. k = 20;
4. num = m;
5. den = [m, b, k];
6. sys = tf(num, den);
7. pidTuner(sys, 'PIDF')
```





- Nhận xét: kết quả được tính tự động bởi Matlab dễ dàng theo dõi, trực quan hơn so với tự nhập.

2.7. Mô phỏng mô hình sử dụng Simulink

2.7.1. Mặc dù Kd là khối đạo hàm, tuy nhiên thực tế khi thiết kế không bao giờ sử dụng trực tiếp khối đạo hàm mà thay bằng khối phản hồi với bộ tích phân. Hãy tìm hiểu tại sao?

- Khó điều chỉnh

Khối đạo hàm có thể gây ra các phản ứng không mong muốn trong hệ thống điều khiển, chẳng hạn như dao động hoặc quá điều chỉnh. Để tránh các vấn đề này, cần phải điều chỉnh rất cẩn thận hệ số Kd. Tuy nhiên, việc điều chỉnh Kd thường rất khó khăn và tốn thời gian.

- Nhạy cảm với nhiễu

Khối đạo hàm rất nhạy cảm với nhiễu. Ngay cả những nhiễu nhỏ cũng có thể gây ra các phản ứng quá mức từ bộ điều khiển. Điều này có thể dẫn đến hoạt động không ổn định của hệ thống.

Để khắc phục những vấn đề này, người ta thường sử dụng khối phản hồi với bộ tích phân thay cho khối đạo hàm. Khối tích phân có thể loại bỏ nhiễu và giúp hệ thống trở nên ổn định hơn.

Khối tích phân sẽ tích phân sai số theo thời gian. Điều này giúp loại bỏ các nhiễu nhiễu đột ngột. Đồng thời, khối tích phân cũng giúp hệ thống phản ứng nhanh hơn với các thay đổi của đầu vào.

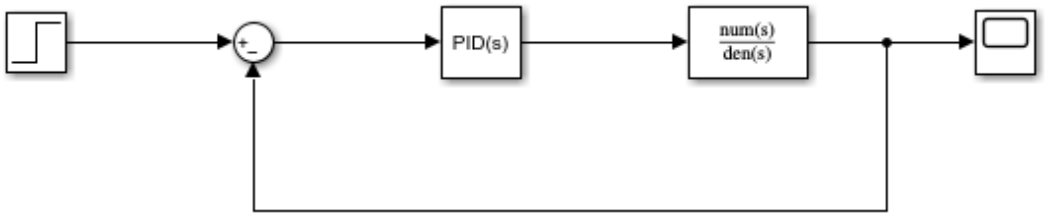
2.7.2. Trong thực tế, khối vi phân s được mô tả lại bởi phương trình

$$\frac{d}{dt} \approx \frac{N}{1+N\frac{1}{s}} = \frac{s}{\frac{1}{N}s+1} = \frac{Ns}{s+N}$$

*Xác định N trong thiết kế:

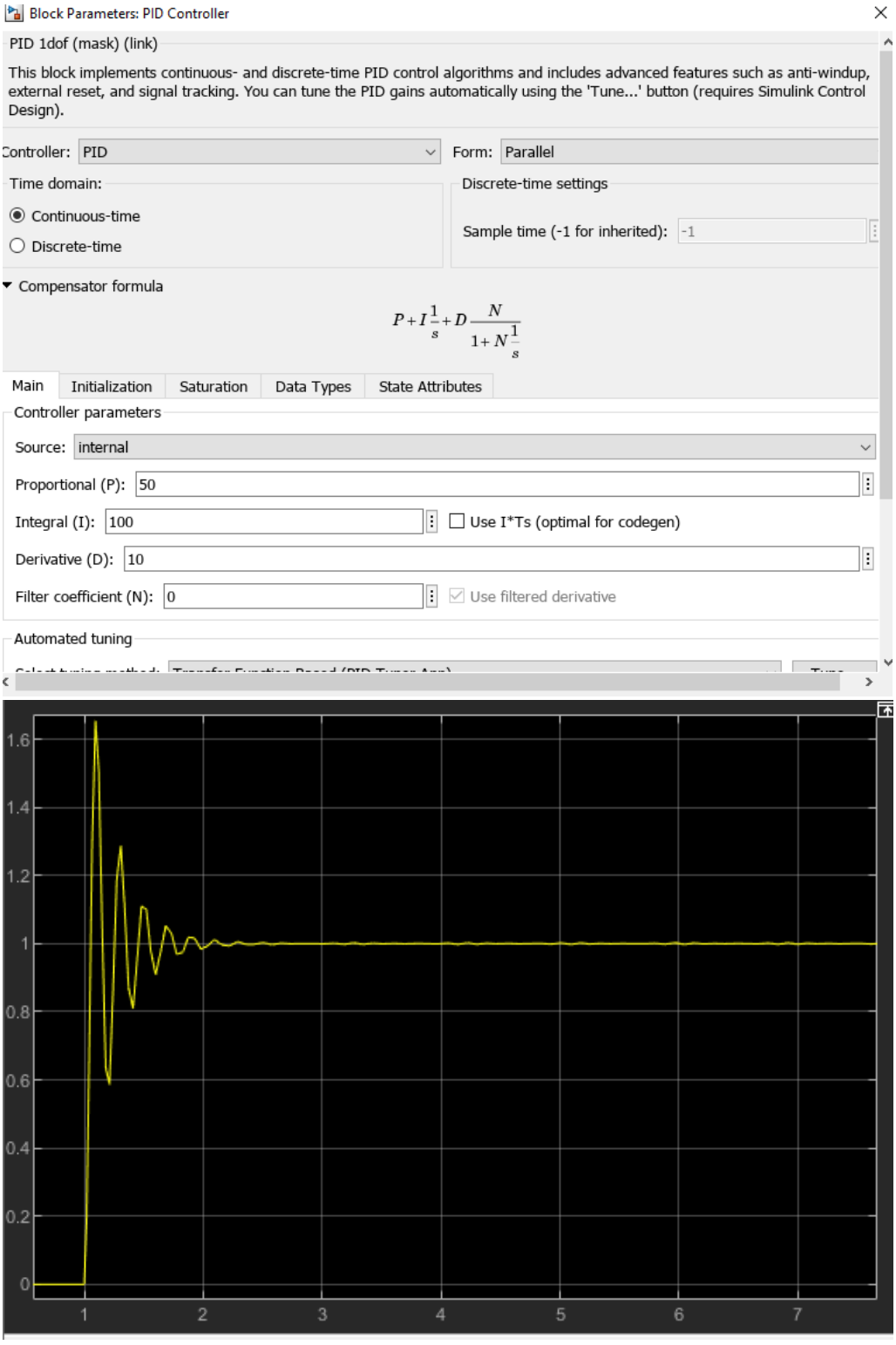
Có $\frac{Ns}{s+N}$ là bộ vi phân dùng để lọc nhiễu tần số cao. Trong đó N là hệ số bộ lọc. Nếu N càng lớn thì $\frac{Ns}{s+N} \approx s$ nhưng lại làm tăng độ nhạy của khối vi phân với nhiễu, còn nếu N càng nhỏ thì hệ thống càng ổn định nhưng lại giảm độ chính xác của khối vi phân

2.7.3. Thực hiện mô phỏng hệ thống trên

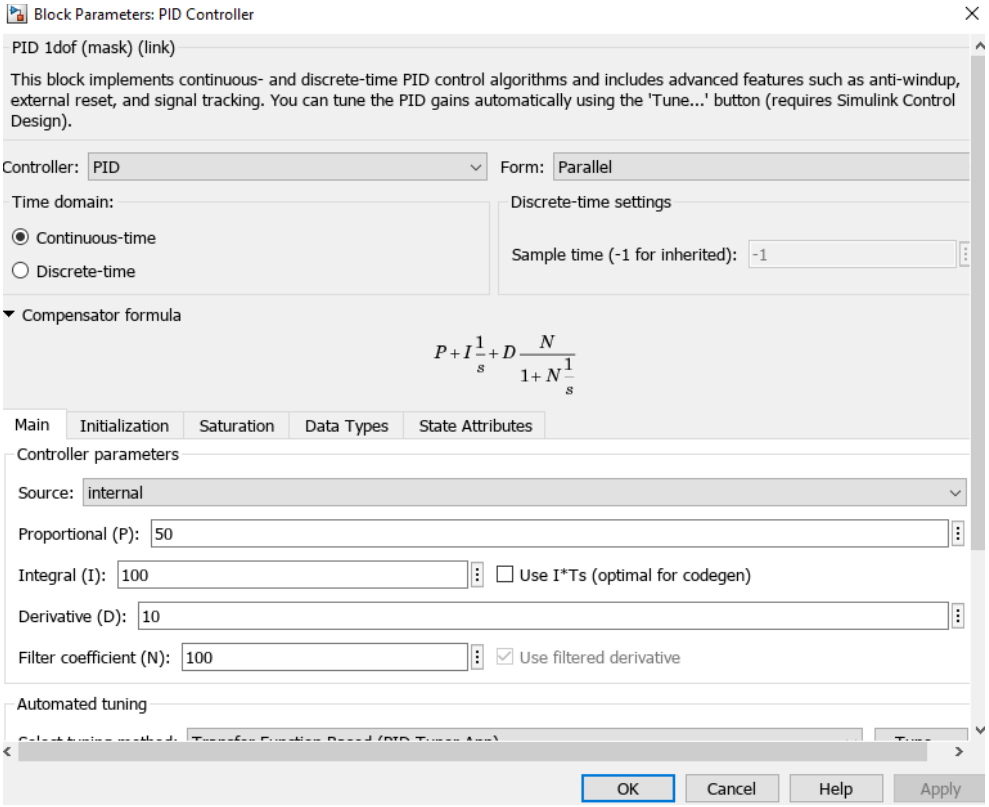


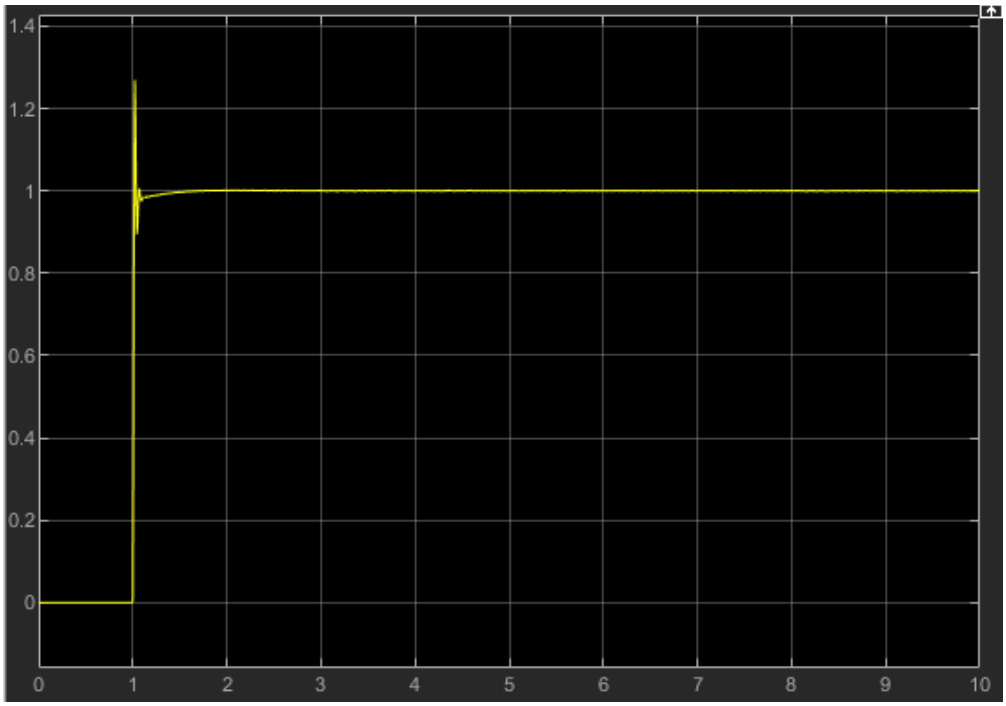
2.7.4. Thay đổi các tham số và kiểm chứng kết quả đầu ra tương ứng

*P=50, I=100, D=10, N=0

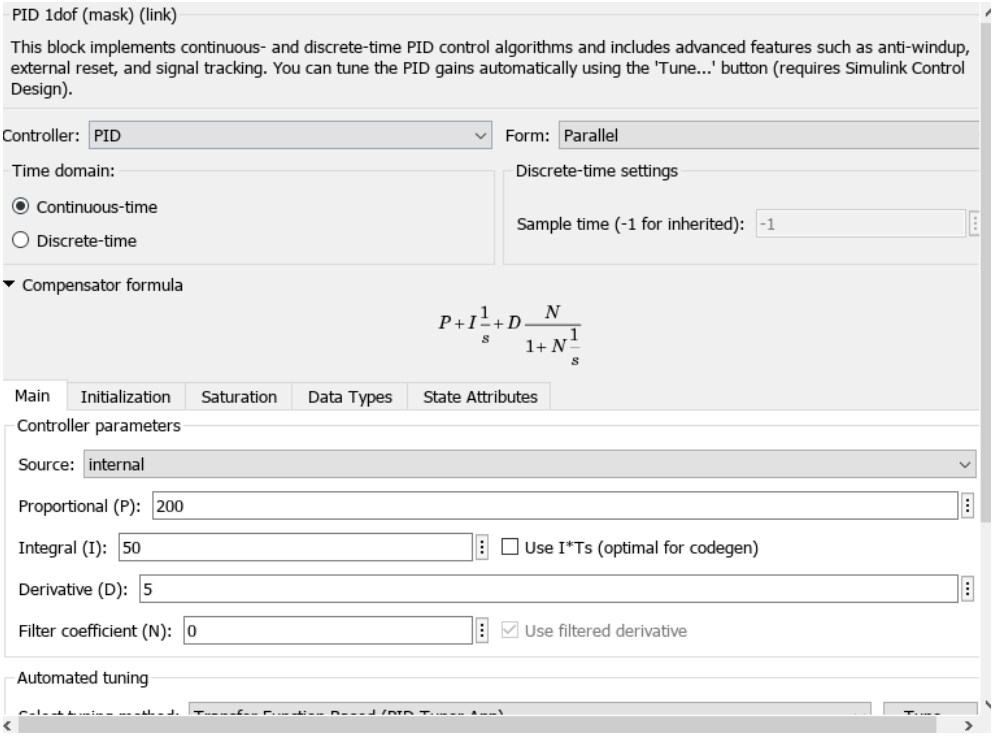

Block Parameters: PID Controller
PID 1dof (mask) (link)
This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).
Controller: PID
Form: Parallel
Time domain:
Continuous-time
Discrete-time
Discrete-time settings
Sample time (-1 for inherited): -1
Compensator formula
$P + I\frac{1}{s} + D\frac{N}{1+N\frac{1}{s}}$
Main
Initialization
Saturation
Data Types
State Attributes
Controller parameters
Source: internal
Proportional (P): 50
Integral (I): 100
Use I*Ts (optimal for codegen)
Derivative (D): 10
Filter coefficient (N): 0
Use filtered derivative
Automated tuning
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
2
3
4
5
6
7

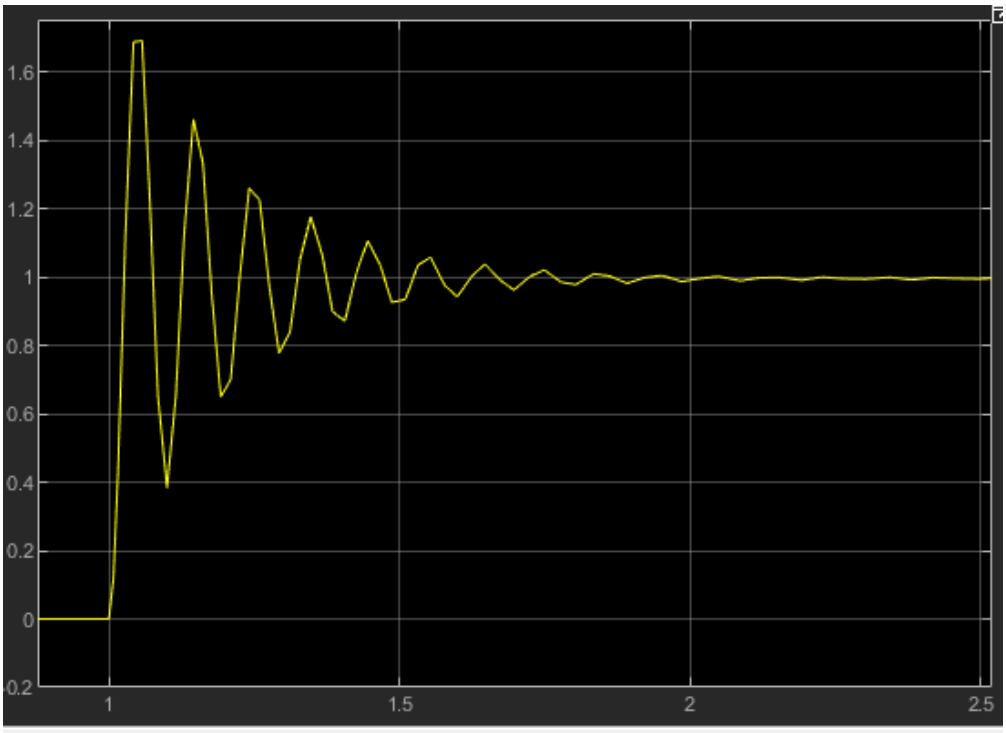
*P=50, I=100, D=10, N=100





*P=200, I=50, D=5, N=0





*P=200, I=50, D=5, N=100