Open Ended Lab 02
PID Controllers

Introduction

Proportional-Integral-Derivative (PID) control is the most common control algorithm used in industry and has been universally accepted in industrial control. The popularity of PID controllers can be attributed partly to their robust performance in a wide range of operating conditions and partly to their functional simplicity, which allows engineers to operate them in a simple, straightforward manner.

As the name suggests, PID algorithm consists of three basic coefficients; proportional, integral and derivative which are varied to get optimal response.

Tuning methods of PID Controller

Before the working of PID controller takes place, it must be tuned to suit with dynamics of the process to be controlled. Designers give the default values for P, I and D terms and these values couldn't give the desired performance and sometimes leads to instability and slow control performances. Different types of tuning methods are developed to tune the PID controllers and require much attention from the operator to select best values of proportional, integral and derivative gains. Some of these are given below.

Trial and Error Method:

It is a simple method of PID controller tuning. While system or controller is working, we can tune the controller. In this method, first we have to set K_I and K_D values to zero and increase proportional term (K_P) until system reaches to oscillating behavior. Once it is oscillating, adjust K_I (Integral term) so that oscillations stops and finally adjust D to get fast response.

Process Reaction Curve technique:

It is an open loop tuning technique. It produces response when a step input is applied to the system. Initially, we have to apply some control output to the system manually and have to record response curve.

After that we need to calculate slope, dead time, rise time of the curve and finally substitute these values in P, I and D equations to get the gain values of PID terms.

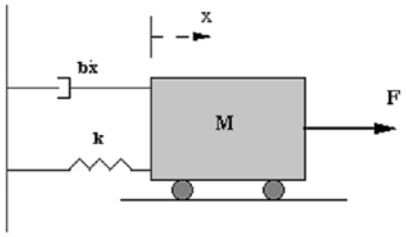
Zeigler-Nichols method:

Zeigler-Nichols proposed closed loop methods for tuning the PID controller. Those are continuous cycling method and damped oscillation method. Procedures for both methods are same but oscillation behavior is different. In this, first we have to set the p-controller constant, Kp to a particular value while Ki and Kd values are zero. Proportional gain is increased till system oscillates at constant amplitude.

Gain at which system produces constant oscillations is called ultimate gain (Ku) and period of oscillations is called ultimate period (Pc). Once it is reached, we can enter the values of P, I and D in PID controller by Zeigler-Nicohols table depends on the controller used like P, PI or PID, as shown below.

Deliverable 1

To investigate the process of tuning a PID controller we shall consider a simple mass spring system.



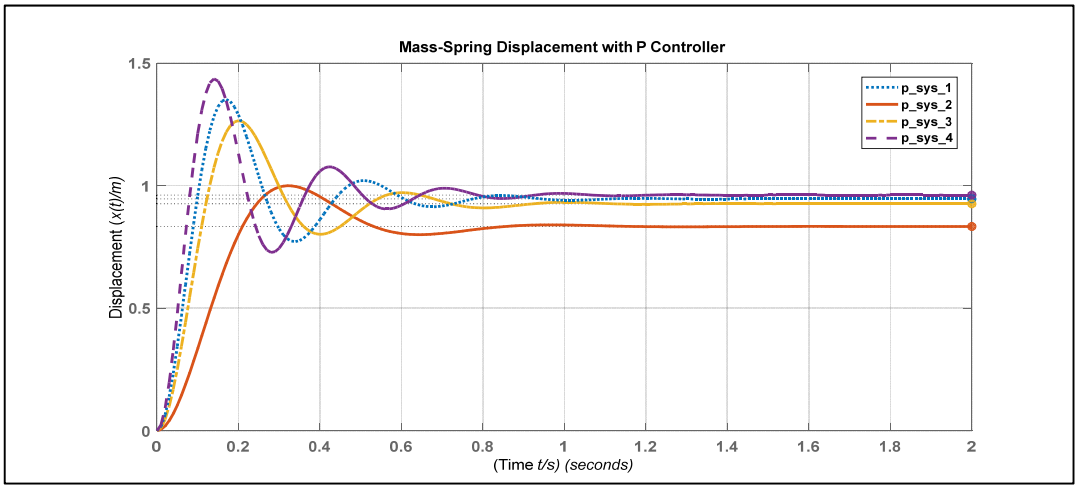
The transfer function of the system is given as

$$\frac{1}{Ms^2 + Bs + K}$$

To control the response of the system a PID controller is added in series with plants transfer function. The approach to tune the controller for this task is basically a hit and trial method. System's response to different values of PID gains are plotted and their performance parameters in response to a step input are evaluated. Afterwards the gain value which gives us the best tradeoff between the performance parameters are chosen. The whole process is detailed below.

Investigation for the Gain of Proportional Controller:

***System* 1 $K_P = 350$**	
Rise Time	0.0665
Settling Time	0.7301
OS	24.83
Peak	1.3512
Peak time	0.1658
System 2 $K_P = 100$	
Rise Time	0.1423
Settling Time	0.76
OS	19.5
Peak	0.9996
Peak time	0.3224
System 3 $K_P = 250$	
Rise Time	0.0809
Settling Time	0.6828
OS	36.63
Peak	1.265
Peak time	0.2026
System 4 $K_P = 500$	
Rise Time	0.0540
Settling Time	0.7443
OS	49.24
Peak	1.4350
Peak time	0.1382



Analysis and Prediction

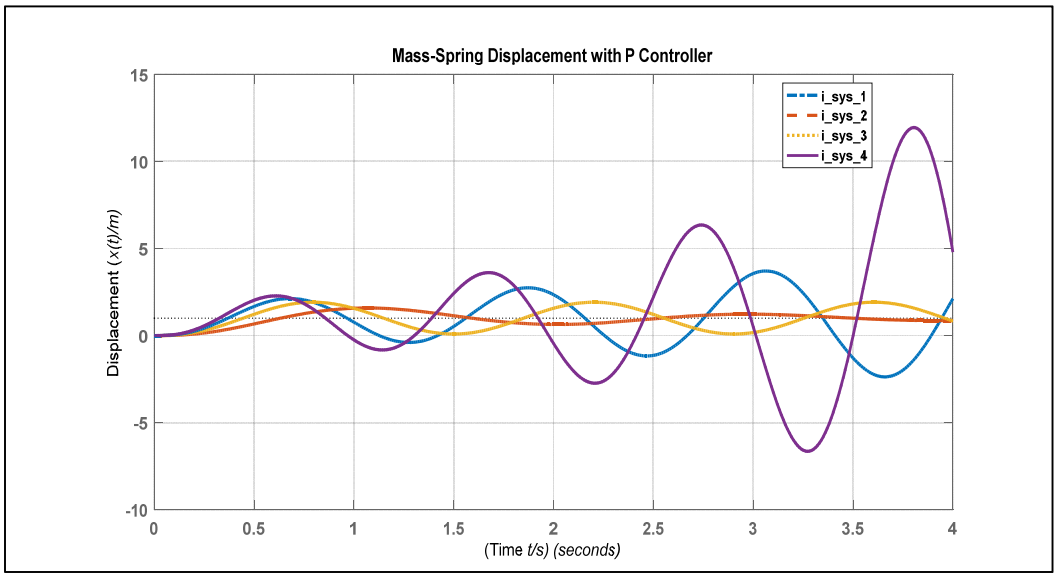
Observations tabulated above shows step response characteristics of a mass spring system for a P controller subjected to different proportional gain values.

- The values shown above suggests that the proportional gain should be set to around 350 for this system to perform best.
- The Overshoot in all cases is above 10 %. But the lowest among all of them is 19.5% for when the proportional gain is set to 100.
- Although this much overshoot isn't allowed but since this is a hit and trial approach of tuning our controller, this much error can be tolerated. More precise tuning of the controller can bring down the overshoot value.
- This minimized overshoot comes at a cost of higher settling time i.e 0.76 seconds, which is marginally higher than other systems while at 350 the settling time is slightly reduced to 0.73.
- The rise time is also slightly higher compared to other systems which is actually 0.14 seconds. This means that the response will reach its final value after 0.14 seconds while at Kp= 350 the rise time is greatly reduced to 0.06 which is less than half of the rise time at Kp = 100.
- At Kp = 100 minimum Overshoot is obtained but other parameter values are not suitable so we suggest Kp = 350 gives best performance which has an overshoot of 24.8%.

Investigation for the Gain of Integral Controller:

System 1 K_i = 300	
Rise Time	Nan
Settling Time	Nan
OS	Nan
Peak	Inf
Peak time	Inf
***System* 2 K_i = 200**	
Rise Time	0.3782
Settling Time	7.85
OS	58.28
Peak	1.582
Peak time	1.062
System 3 K_i = 100	
Rise Time	Nan
Settling Time	Nan
OS	Nan

Peak	Inf
Peak time	Inf
System 4 K_i = 400	
Rise Time	Nan
Settling Time	Nan
OS	Nan
Peak	Inf
Peak time	Inf



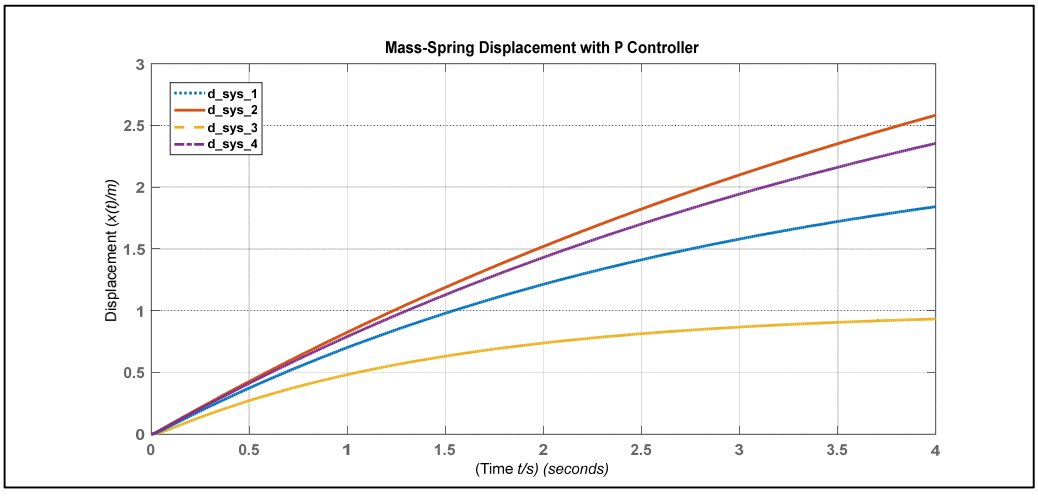
Analysis and Prediction

Observations tabulated above shows step response characteristics of a mass spring system for an Integral controller subjected to different integral gain values.

- The values shown above suggests that the integral gain should be set to around 200 for this system to perform best.
- When the value of integral gain is set to 100, 300 and 400 the response of the system becomes unstable, as the graphs given below shows divergence at these values.
- The only stable response is obtained at the value of 200 by using hit and trial approach.
- But even when the integral gain is set to 200 the overshoot is 58.2% which is very large and makes the system unstable.
- The rise time is 0.3 seconds while for other cases it's not defined.

Investigation for the Gain of Derivative Controller:

System 1 $K_d = 50$	
Rise Time	6.55
Settling Time	11.68
OS	0
Peak	2.5
Peak time	31.46
System 2 $K_d = 100$	
Rise Time	12.06
Settling Time	21.48
OS	0
Peak	5
Peak time	57.90
System 3 $K_d = 20$	
Rise Time	3.22
Settling Time	5.76
OS	0
Peak	1
Peak time	10.733
System 4 $K_d = 80$	
Rise Time	9.86
Settling Time	17.51
OS	0
Peak	4
Peak time	47.338



Analysis and Prediction

The value of derivative controller is set at different values and the parameters observed at these values are tabulated above.

- The values show above suggests that the derivative gain should be set to around 20 for this system to perform best.
- The Overshoot is all above cases is 0%.
- The minimum value of rise time is obtained at derivative gain equal to 20, which is 3.22. This means that the response will reach its final value after 3.22 seconds.
- The minimum settling time is obtained is 5.76 when the derivative gain is 20.
- Minimum peak is also obtained at derivative gain equal to 20 at the peak time of 10.73 seconds.

Deliverable 2

The term PID corresponds to the proportional, integral and derivative terms. It aims to control the response of a system by adjusting the manipulated variable. Mathematically the sum of the three terms PID constitutes the controller output or the manipulated variable.

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

Where;

$$K_p \textit{ is the proportional gain, } K_i \textit{ is the integral gain, \& } K_d \textit{ is the derivative gain}$$

The PID gains are applied on the error term resulting the difference in the set point value and the current value.

Proportional Term

Proportional or P- controller gives output which is proportional to current error e (t). It compares desired or set point with actual value or feedback process value. The resulting error is multiplied with proportional constant to get the output. If the error value is zero, then this controller output is zero. The proportional term is given by

$$P_{out} = K_p\ e(t)$$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances. In general, increasing the proportional gain will increase the speed of the control system response. However, if the proportional gain is too large, the process variable will begin to oscillate. If K_c is increased further, the oscillations will become larger and the system will become unstable and may even oscillate out of control.

Integral Term

Due to limitation of p-controller where there always exists an offset between the process variable and set point, I-controller is needed, which provides necessary action to eliminate the steady state error. It integrates the error over a period of time until error value reaches to zero. It holds the value to final control device at which error becomes zero.

Integral control decreases its output when negative error takes place. It limits the speed of response and affects stability of the system. Speed of the response is increased by decreasing integral gain Ki. A pure "I" controller could bring the error to zero, but it would be both slow reacting at the start (because action would be small at the beginning, needing time to get significant) and brutal (the action increases as long as the error is positive, even if the error has started to approach zero). However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the set point value

The integral term is given as

$$I_{out} = K_i \int_0^t e(\tau) d\tau$$

Derivative Term

The derivative component causes the output to decrease if the process variable is increasing rapidly. The derivative response is proportional to the rate of change of the process variable. Increasing the *derivative time (T_d)* parameter will cause the control system to react more strongly to changes in the error term and will increase the speed of the overall control system response. Most practical control systems use very small derivative time (T_d), because the Derivative Response is highly sensitive to noise in the process variable signal. I-controller doesn't have the capability to predict the future behavior of error. So it reacts normally once the set point is changed. D-controller overcomes this problem by anticipating future behavior of the error. Its output depends on rate of change of error with respect to time, multiplied by derivative constant.

The derivative term is given as

$$D_{out} = K_d \frac{de(t)}{dt}$$

Deliverable 3

Plot the responses of a simple mass, spring, and damper system with a P, I, D, PI, PD, and PID controller with rise time, settling times, overshoot, and steady state value for each response. Also tabulate these characteristics along with the steady state error and overshoot time.

Relevant parameters are as follows:

Table 1: Investigation Parameters

Mechanical Parameters		**Controller Parameters**	
M	1 kg	K_P	350
B	10 N.s/m	K_I	300
K	20 N/m	K_D	50

Solution

Code 01 is a MATLAB program that investigates the effect of P, I, D, PI, PD, and PID controllers on the step response of a simple damped mass spring mechanical system. It first initializes both mechanical and controller parameters as defined in Table 1, and uses them to define the following transfer functions.

Table 2: Transfer Functions

S. No	**System Name**	**Transfer Function**
1	Plant	$\frac{1}{Ms^2+Bs+K}$
2	Potential Controller Plant	$\frac{k_P}{Ms^2+Bs+K+K_p}$
3	Integral Controller Plant	$\frac{K_I}{Ms^3+Bs^2+Ks+K_I}$
4	Derivative Controller Plant	$\frac{K_D s}{Ms^2+(B+K_D)s+K}$
5	Potential-Derivative Controller Plant	$\frac{K_D s+\ K_P}{Ms^2+(B+K_D)s+(K+K_P)}$
6	Potential-Integral Controller Plant	$\frac{K_P s+K_I}{Ms^3+Bs^2+(K+K_P)s+K_I}$
7	Potential-Integral-Derivative Controller Plant	$\frac{K_D s^2+K_P s+K_I}{Ms^3+(B+K_D)s^2+(K+K_P)s+K_I}$

Code 01: Investigating 2nd Order System Characteristics with PID Controllers

```
%%  FCS Open Ended Lab 02 - PID Controller and 2nd Order System Characteristics
%   Investigating the use of a P, I, and D controllers on the time response
%   characteristics of a 2nd order mass-spring system

%%  Initialization
%   prepare workspace
clear all; close all; clc;

%   Define mechanical constants and transfer fcn for mass-spring system
M = 1; B = 10; k = 20;
plant_tf = tf(1, [M, B, k]);   % echo for confirmation
%%  Plant-Controller Setup
%   Define coefficients for each controller
k_p = 350; k_i = 300; k_d = 50;

%   Set up transfer function for each plant
p_sys = tf([0 0 k_p], [M, B, k + k_p]);
i_sys = tf([0 0 k_i], [M, B, k, k_p]);
d_sys = tf([0 k_d 0], [M, (B + k_d), k]);

%%   For more complicated combinations, using MATLAB's built-in functions
%   PD Controller
pd_controller = tf([k_d, k_p], 1);
pd_sys = feedback(pd_controller * plant_tf, 1);

%   PI Controller
pi_controller = tf([k_p, k_i], [1, 0]);
pi_sys = feedback(pi_controller * plant_tf, 1);

%   PID Controller
pid_controller = tf([k_d, k_p, k_i], [1, 0]);
pid_sys = feedback(pid_controller * plant_tf, 1);
%%  Plotting each system's response
t = 0:0.01:2;           % time domain for all plots

%   P controller
figure(); step(p_sys, t, 'k'); xlabel('(Time \it{t/s})');
ylabel('Displacement (\it{x(t)/m})'); grid on;
title('Mass-Spring Displacement with P Controller');

%   I controller
```

```
figure(); step(i_sys, t, 'k'); xlabel('(Time \it{t/s})');
ylabel('Displacement (\it{x(t)/m})'); grid on;
title('Mass-Spring Displacement with I Controller');

%   D controller
figure(); step(d_sys, t, 'k'); xlabel('(Time \it{t/s})');
ylabel('Displacement (\it{x(t)/m})'); grid on;
title('Mass-Spring Displacement with D Controller');

%   PI controller
figure(); step(pi_sys, t, 'k'); xlabel('(Time \it{t/s})');
ylabel('Displacement (\it{x(t)/m})'); grid on;
title('Mass-Spring Displacement with PI Controller');

%   PD controller
figure(); step(pd_sys, t, 'k'); xlabel('(Time \it{t/s})');
ylabel('Displacement (\it{x(t)/m})'); grid on;
title('Mass-Spring Displacement with PD Controller');

%   PID controller
figure(); step(pid_sys, t, 'k'); xlabel('(Time \it{t/s})');
ylabel('Displacement (\it{x(t)/m})'); grid on;
title('Mass-Spring Displacement with PID Controller');
```

Transfer functions 1 - 4 are implemented directly in MATLAB using the built-in tf function while the more complicated transfer functions 5 - 7 are implemented using MATLAB's block diagram reduction function feedback.

Once all transfer functions for all systems have been defined, the program uses MATLAB's step function to investigate each system's response over a duration of 2 seconds. Given that this is a second order system for the differential equation

$$F_A(t) = M\frac{d^2x}{dt^2} + B\frac{dx}{dt} + K$$

the step input in this case is a force of 1 N whereas the step response is the displacement experienced by the system's mass due to this force.

The program then plots the response for each system on a separate figure. These responses, along with data cursors showing their relevant second order time response characteristics, are shown in Figures 1 - 6.

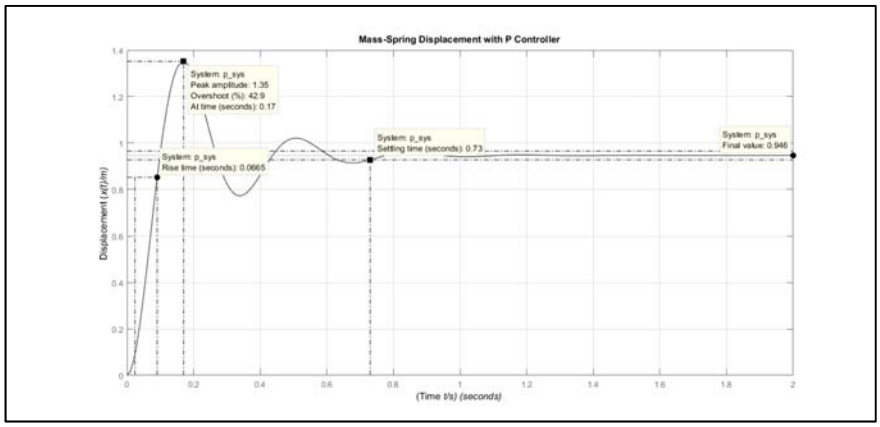


Figure 1: Response with P Controller

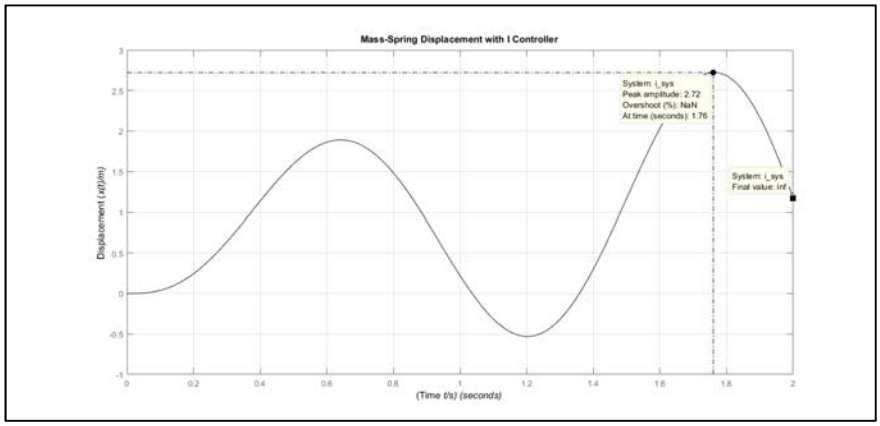


Figure 2: Response with I Controller

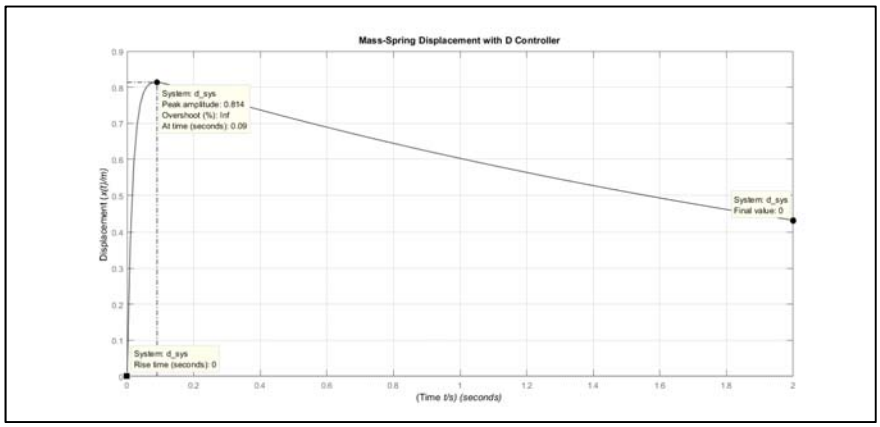


Figure 3: Response with D Controller

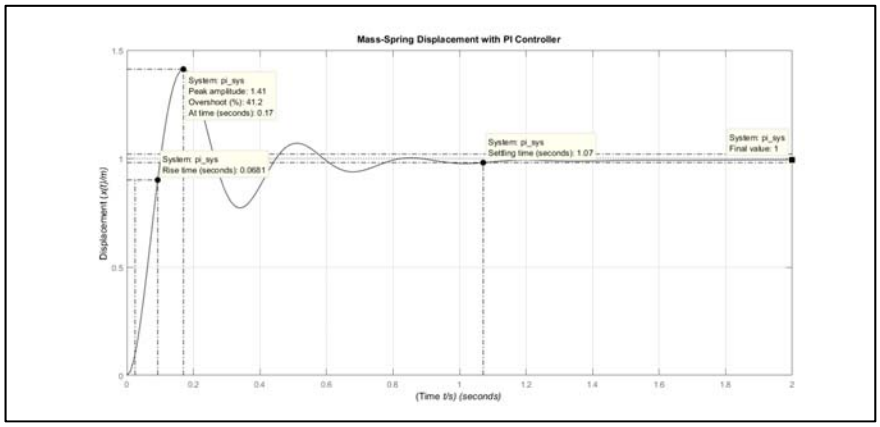


Figure 4: Response with PI Controller

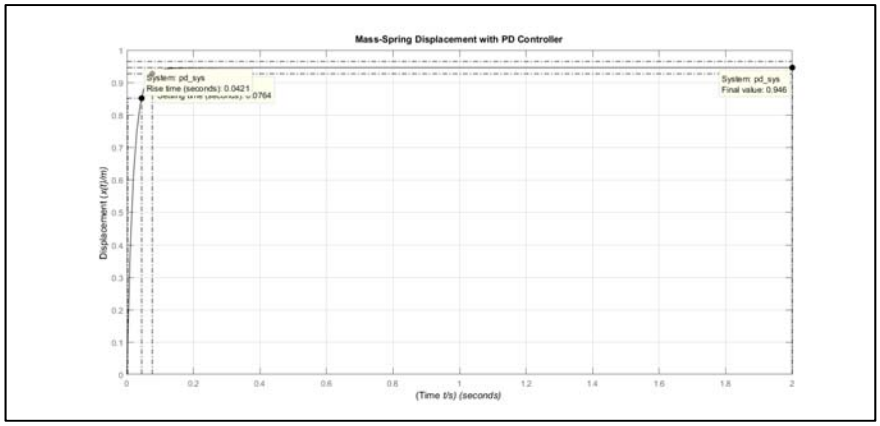


Figure 5: Response with PD Controller

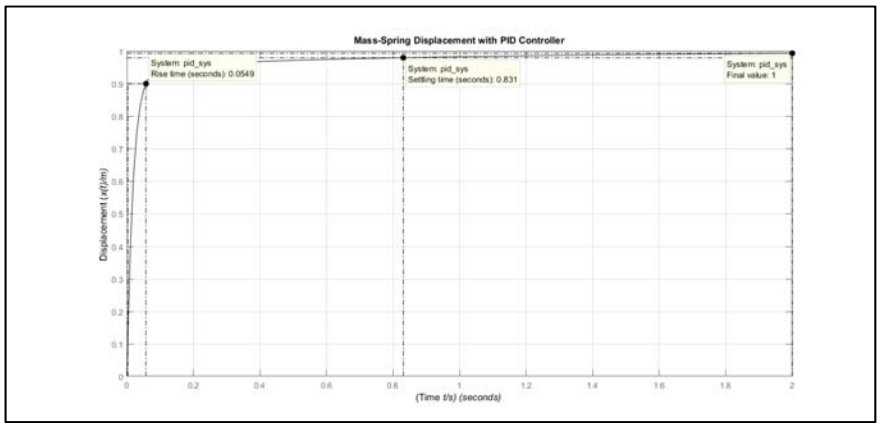


Figure 6: Response with PID Controller

Time Response Characteristics

The time response characteristics of the systems listed in Table 2 are tabulated below. All readings were obtained by plotting the step response of the system in MATLAB and using the built-in 'characteristics' feature.

Table 3: Time Response Characteristics with Different Controllers

System	Rise Time	Settling Time	Peak Value	OS Time	OS %	Steady State Value	Steady State Error
P	0.0655	0.73	1.35	0.17	42.9	0.946	0.054
I	NaN	NaN	2.72	1.76	NaN	NaN	NaN
D	0.05	NaN	0.814	0.09	NaN	0	1
PI	0.0681	1.07	1.41	0.17	41.20%	1	0
PD	0.0421	0.0764	0.95	NaN	0	0.946	0.054
PID	0.0549	0.831	1	NaN	0	1	0

Analysis of Results

The P controller shows some evidence of underdamping, as the displacement oscillates with a peak overshoot of 42.9% before reaching a steady state value of 0.946 m. However, it has a relatively low rise time and the second lowest settling time, although neither of these parameters are the lowest in all the systems investigated. This suggests that a P controller is a satisfactory, albeit not ideal, controller for this system.

The I controller seems to be the least suitable controller choices for this system. It creates an unstable system which is why MATLAB is unable to determine its rise time, settling time,

and percentage overshoot, and steady state value and error. The magnitude of the response diverges as $t \to \infty$. This means the system has an unbounded output (displacement) for a bounded input (force), and since its theoretical peak value is ∞, it is computationally intractable to determine its rise, settling, peak, and overshoot times as these values technically are undefined for a response that diverges to infinity.

The D controller is also not a suitable choice. Even though its response is stable with a finite rise time, peak time, and overshoot time, it has the highest steady state error - with the is controller, the system returns to its original position instead of maintaining a 1 m displacement. It seems to have marginally reduced the rise time, peak value, and overshoot time, but has resulted in a large %age OS and a large steady state error.

The PI controller is one of two controllers with the lowest steady state error, as its steady state value approaches 1. Its rise time, settling time, and peak time are all marginally higher than the P controller. Overall, this response is very similar to that of the P controller, suggesting that the addition of an I controller has done little to improve the response.

The PD and PID controllers seem to be amongst the best choices of controller for this system. Both of them show clear evidence of overdamping, although this is not necessarily a bad thing. The PD controller has marginally reduced the rise time but substantially reduced settling time compared to the P controller, and has also completely eliminated overshoot. It does have a 0.05 steady state error, which the PID controller eliminates entirely albeit at the cost of slightly higher rise, settling, and peak times.

Conclusion

The PID or PD controller seem to the best configurations for this tuning this system's response. Choosing between the two depends on the context or application. If the minimization of steady state error is more important for the system than speed of response, then the PID controller has a clear advantage over the PD controller. However, given that the steady state error in the PD controller is only 5% of the steady state value, this can be disregarded if the speed of the response has higher priority. In this case the PD controller is more suitable for tuning the system's response.

Overall, P controllers seem to reduce rise time and will also slightly reduce the steady state error. The I controller can potentially eliminate the steady state error entirely, but this benefit is often offset by the controller worsening the system's transient response, which can lead to a larger steady state error than before. The D controller is much better in this regard as it not only increases the stability of the system but also reduces overshoot and improves the transient response.