

ME451

PreLab 8

Introduction to LabVIEW

8.1. Introduction to LabVIEW

The objective of this lab is to get familiar with the LabVIEW programming software environment, understand its basic capabilities and main functionalities, and apply many of the concepts from the previous labs. This lab mainly consists of two parts. The first part encompasses simulating a single degree of freedom spring-mass-damper system and study the system response under both free vibration and forced vibration. The second part tackles the designing of different control strategies for the same system and then investigating the system behavior under each control methodology.

8.1.1 LabVIEW:

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a graphical programming platform and development environment that is widely used for various measurement, test, and control applications. LabVIEW is equipped with numerous libraries that each have many functions for data acquisition, signal analysis, control, statistics and many more. In addition, LabVIEW includes a text-based programming component called Math Script with additional functionality for signal processing, analysis and mathematics. Math Script can be integrated with graphical programming using "script nodes" and uses a syntax that is generally compatible with MATLAB.

LabVIEW programs are called Virtual Instrument (VIs). Each VI contains three main parts:

- Front Panel that allows the user to interact with the VI.
- Block Diagram that contains the code that controls the program.
- Icon/Connector that provides means of manipulating and editing the VI.

The Front Panel is used to interact with the user when the program is running. Users can control the program, change inputs, and see the data being updated in real time. Controls are used for inputs such as adjusting a slide control to set an alarm value, turning a switch on or off, assigning a numeric value or even stopping the program. Indicators are used as outputs such as thermometers, LEDs, and numeric indicators. Every control or indicator on the front panel has a corresponding terminal on the block diagram. When a VI is run, values from controls flow through the block diagram, where they are used in the functions on the diagram, and the results are passed into other functions or indicators.

The front panel objects appear as terminals on the block diagram. The block diagram contains the graphical source code. Additionally, the block diagram contains functions and structures from built-in LabVIEW VI libraries. Wires connect each of the nodes on the block diagram, including control and indicator terminals, functions, and structures. The programming language used in LabVIEW, also referred to as G, is a dataflow programming language. Execution is determined by the structure of a graphical block diagram (the LabVIEW-source code) on which the programmer connects different function-nodes by drawing wires. Figure 1 shows a VI (Both Front Panel and Block Diagram) to calculate the sum and the difference of two numbers a and b.

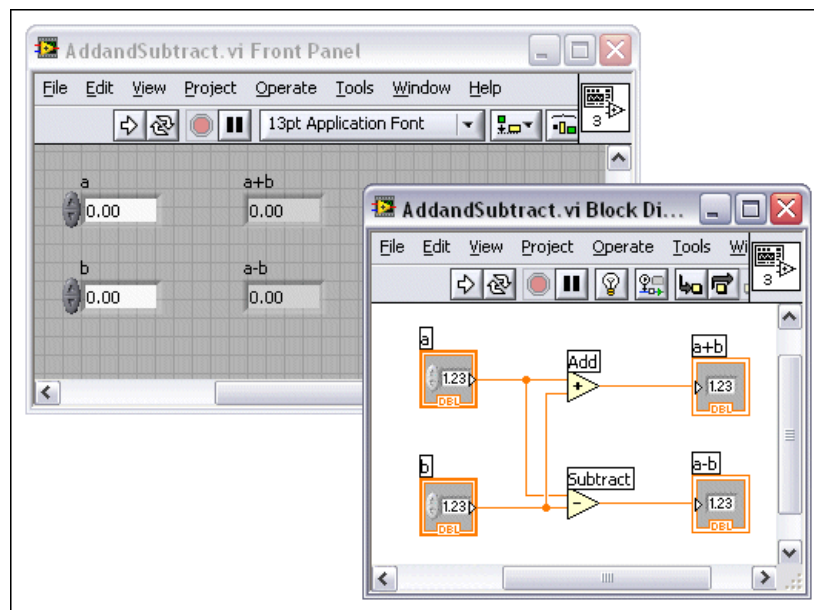


Figure 1. Typical Virtual Instrument to calculate the sum and difference of two numbers. As shown, the VI consists of both the Block Diagram and the Front Panel

- To get more familiar with Labview, [watch parts 1,2,3,5,6, and 7 from this video series](#). This 'getting started with LabVIEW' series video will show some of the basic techniques which will be used in our labs.
- You can access Labview through the remote desktop provided by DECS of MSU.
- Implement the above VI to calculate the sum and the difference of two numbers through LabVIEW

8.1.2 Second Order System transient response Characteristics/specifications:

After we have learned how to use LabVIEW, we will now try to utilize LabVIEW with our system dynamics and control theory background. The content below will review some the basic properties and terminologies of a second order system.

The closed loop transfer function (of the system shown in Figure 1) that relates the output $C(s)$ to the input $R(s)$ can be derived as follows:

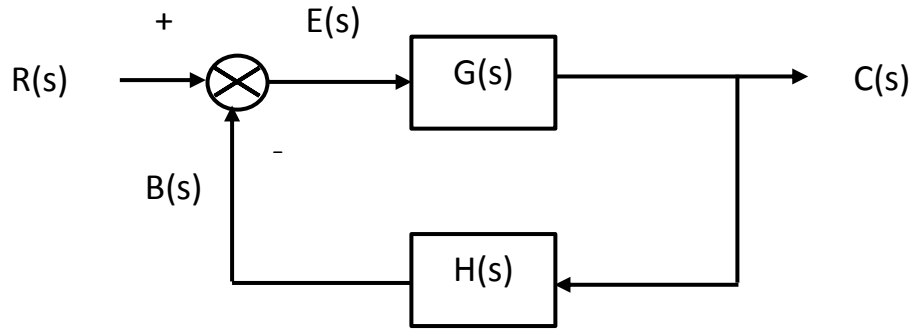


Figure 2. Block Diagram of a closed loop system with feedback element

$$C(s) = G(s) * E(s) \quad (1)$$

$$E(s) = R(s) - B(s) = R(s) - H(s) * C(s) \quad (2)$$

By eliminating $E(s)$ from Eqn. (1) gives:

$$C(s) = G(s) * [R(s) - H(s) * C(s)] \quad (3)$$

Or:

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + H(s) * G(s)} \quad (4)$$

The transfer function relating $C(s)$ to $R(s)$ is called the closed loop transfer function. Obviously, the output response of a given closed loop system is dependent on both the closed loop transfer function or in other words the system dynamics and the input to the system.

Since systems with energy storage cannot respond instantaneously to inputs, they will exhibit transient response when subjected to inputs or disturbances. Usually, a desired system performance is given in terms of transient response specifications. In most cases, performance characteristics are defined in terms of the transient response to a unit step input. Transient response characteristics include:

1. Rise Time
2. Settling time
3. Steady State Error
4. Overshoot

The **rise time** is the time required for the response to rise from 10% to 90% of the set point. The **settling time** is the time required for the response curve to reach and stay within 2% of the final value (in some cases 5% instead of 2% is used as a percentage of the final value). The **overshoot** (the maximum overshoot) is the maximum peak value of the response curve measured from unity (or the final steady state value). Figure 3 shows the transient response characteristics of a second order system subjected to a unit step input.

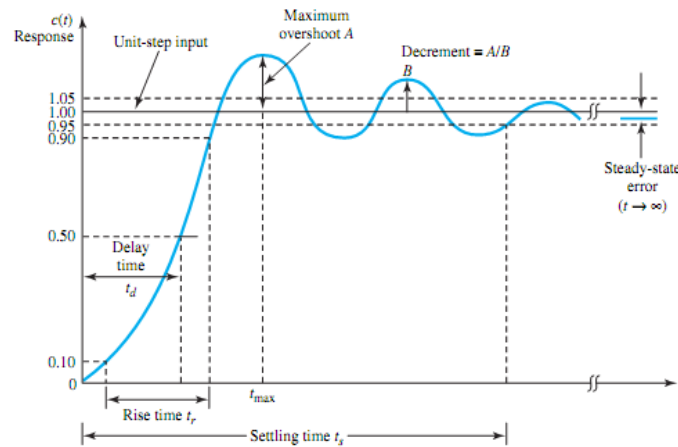


Figure 3. Typical Transient and Steady State Response Characteristics of a second order system subjected to step input

The settling time can be calculated from:

$$T_s = \frac{4}{\xi \omega_n} \quad (5)$$

Where ξ is the damping ratio and ω_n is the natural frequency.

The overshoot can be calculated from:

$$Mp = \frac{c(t_p) - c(t_\infty)}{c(t_\infty)} \quad (6)$$

Where $c(t_p)$ is the amplitude at the peak time and $c(t_\infty)$ is the amplitude at the steady-state.

8.2. Simulation of a Spring Mass Damper System

Consider a single spring-mass-damper system, with its FBD shown Figure 4.

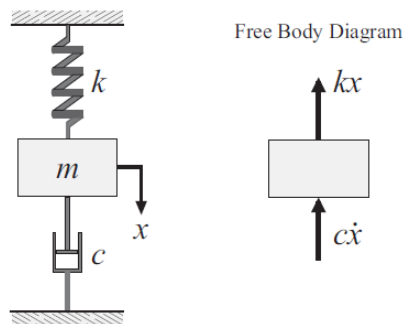


Figure 4. Mass Spring Damper System

The transfer function of the system that relates the displacement x to the input F is found as:

$$G(s) = \frac{1}{ms^2 + bs + k} \quad (7)$$

Note that during the derivation of this transfer function, we assumed zero initial conditions. However, in the lab, we will be able to give the system non-zero initial conditions.

8.3 Control of a Spring Mass Damper System

Now we can apply various controllers into the S-M-D system to achieve our desired system response. Let's first review the PID controller:

8.3.1 Proportional-Integral-Derivative controller

PID controller is a powerful control strategy for a variety of the automatic process control applications in industry. It has become a standard tool since process control took place in the 1940s. PID controllers can regulate flow, temperature, pressure, level, and many other industrial process variables. Proper utilization of PID controller requires parameters tuning that will regulate the system response characteristics. A typical block diagram for a plant with a PID controller is shown in Figure 5.

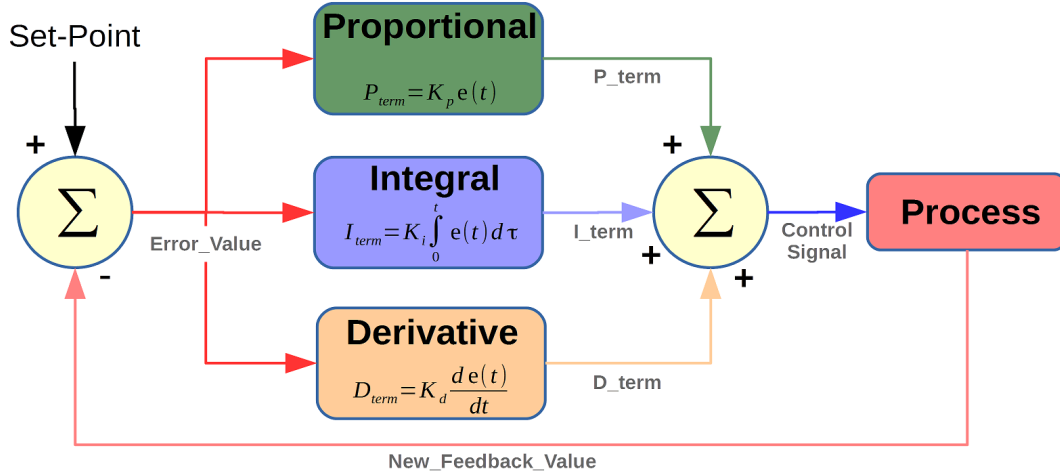


Figure 5. Block diagram of a Plant with a PID controller [Photo Courtesy to REF]

The transfer function of the PID controller can be represented as:

$$K_p + \frac{K_I}{s} + K_D s = \frac{K_p s + K_I + K_D s^2}{s} \quad (8)$$

Where:

- K_p = Proportional gain
- K_I = Integral gain
- K_D = Derivative gain

By referring to Figure 5, the error value represents the tracking error, or the difference between the desired set point and the actual output (y). This error signal is the input for the PID controller, and the controller computes proportional, the derivative and the integral of this error signal. The control action (u) is the summation of the proportional gain (K_p) times the magnitude of the error, plus the integral gain (K_I) times the integral of the error, plus the derivative gain (K_D) times the derivative of the error.

8.3.1.1 Pure Proportional Controller

The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant (K_P), called the proportional gain constant. The proportional term depends on the present error, and is given by:

$$P_{out} = K_p * e(t) \quad (9)$$

Using a proportional control, the system response will always have a steady state error. The error will decrease with increasing proportional gain, but the tendency towards oscillation (and higher overshoot) will also increase. On the other hand, steady state error tends to depend inversely upon the proportional gain. The proportional gain cannot be set very high as the system can become unstable.

8.3.1.2 Pure Integral Controller

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral term is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. Thus, the Integral depends on the accumulation of past errors. The integral term is given by:

$$I_{out} = K_I * \int_0^t e(\tau) d\tau \quad (10)$$

Steady state error can be eliminated when integral action is used; however, the tendency for oscillation also increases.

8.3.1.3 Pure Derivative Controller

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain (K_D). The derivative gain can be interpreted as a prediction of future errors. The derivative term is given by:

$$D_{out} = K_D * \frac{d}{dt} e(t) \quad (11)$$

If the closed loop system response has oscillatory response, the derivative gain will increase damping at first, but decreases again when derivative gain becomes too large.

In summary, the proportional controller will have the effect of reducing the rise time and will reduce, but never eliminate, the steady-state error. An integral control will have the effect of eliminating the steady-state error, but it may make the transient response worse. A derivative control will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response. Table 1 summarizes the effects of each of controller gain parameters on a closed-loop system.

Table 1. Effect of PID Control parameters on the closed loop system response

Controller Parameter	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
K_P	Decrease	Increase	Small Change	Decrease
K_I	Decrease	Increase	Increase	Eliminate
K_D	Small Change	Decrease	Decrease	No Change

Note however that these correlations may not be exactly accurate, because (K_P) , (K_I) , and (K_D) are dependent of each other. In fact, changing one of these variables can change the effect of the other two. For this reason, the table should only be used as a reference when you are determining the values for K_i , (K_P) and (K_D) .

For the spring mass system discussed above, we will now present the closed loop transfer functions using different combinations of the PID controller to move the mass by giving a step input while having a desired performance of:

1. No steady state error
2. No overshoot
3. Settling time less than 0.25 sec.

The system parameter values are: mass=1 kg, $k=20$ N/m, $c=10$ N/(m/s), $F=1$ N.

The following are the closed loop transfer function of the spring-mass-damper system with the parameter values (mass=1 kg, k=20 N/m, c=10 N/(m/s)) using different types of controllers:

Controller	Closed Loop TF
Pure proportional	$H(s) = \frac{K_p}{s^2 + 10s + (20 + K_p)}$
Proportional-Derivative	$H(s) = \frac{K_D s + K_P}{s^2 + (10 + K_P)s + (20 + K_P)}$
Proportional-Integral	$H(s) = \frac{K_P s + K_I}{s^3 + 10s^2 + (20 + K_P)s + K_I}$
Proportional-Integral-Derivative	$H(s) = \frac{K_D s^2 + K_P s + K_I}{s^3 + (10 + K_D)s^2 + (20 + K_P)s + K_I}$

General procedure for designing/tuning a PID controller:

First, obtain an open-loop response and determine what transient response characteristics needs to be improved given the system desired response. Afterwards, you may:

1. Add a proportional control to improve the rise time
2. Add a derivative control to improve the overshoot
3. Add an integral control to eliminate the steady-state error
4. Adjust each of K_P , K_I , and K_D until you obtain the desired system response.

Remember that sometimes you might not need to implement all three controllers (proportional, derivative, and integral) into your system. If the system desired response is achieved by a PI controller, then a PID controller is not needed.