DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
BANGLADESH UNIVERSITY OF ENGINEERING & TECHNOLOGY
EEE 318 Control System I Laboratory

Experiment No. 1

(a): Modeling of Physical Systems and Study of Their Open Loop Response.
(b): PID Design Method for DC Motor Speed Control.

Introduction:

To understand and control physical systems, one of the initial requirements is to obtain the **mathematical model** of the system. A model is one that quantitatively describes the relationship between the input and output of a dynamic system. To model systems, we use physical laws, such as Kirchhoff's current and voltage laws for electrical networks and Newton's law for mechanical systems, along with simplifying assumptions.

Because the systems are usually dynamic in nature, one such model relating the input and output is the *linear, time invariant differential equation* as shown below:

$$a_n \frac{d^n c(t)}{dt^n} + a_{n-1} \frac{d^{n-1} c(t)}{dt^{n-1}} + \cdots + a_0 c(t) = b_m \frac{d^m r(t)}{dt^m} + b_{m-1} \frac{d^{m-1} r(t)}{dt^{n-1}} + \cdots + b_0 r(t)$$

where, the output and input are $c(t)$ and $r(t)$ respectively and a_i and b_j are system parameters.

However, the classical approach of modeling linear systems is the **transfer function** technique which is derived from the differential equation using *Laplace transform.* Transfer function yield more intuitive information than the differential equation by visualizing the effect of system parameter variations on the system response as well as it eases the modeling of interconnected subsystems by forming block diagrams.

Second-Order System: The general second order transfer function can be expressed as:

$$G(s) = \frac{b}{s^2 + as + b} \quad (1)$$

This transfer function can also be expressed in terms of its natural frequency, ω_n and damping ratio, ζ as:

$$G(s) = \frac{{\omega_n}^2}{s^2 + 2\zeta\omega_n s + {\omega_n}^2}, (2)$$

where, $b = {\omega_n}^2$ and $\zeta = {a/2}/{\omega_n}$.

The response of the system depends on the pole location, thus in effect on the values of ζ and ω_n as seen from Fig. 1.

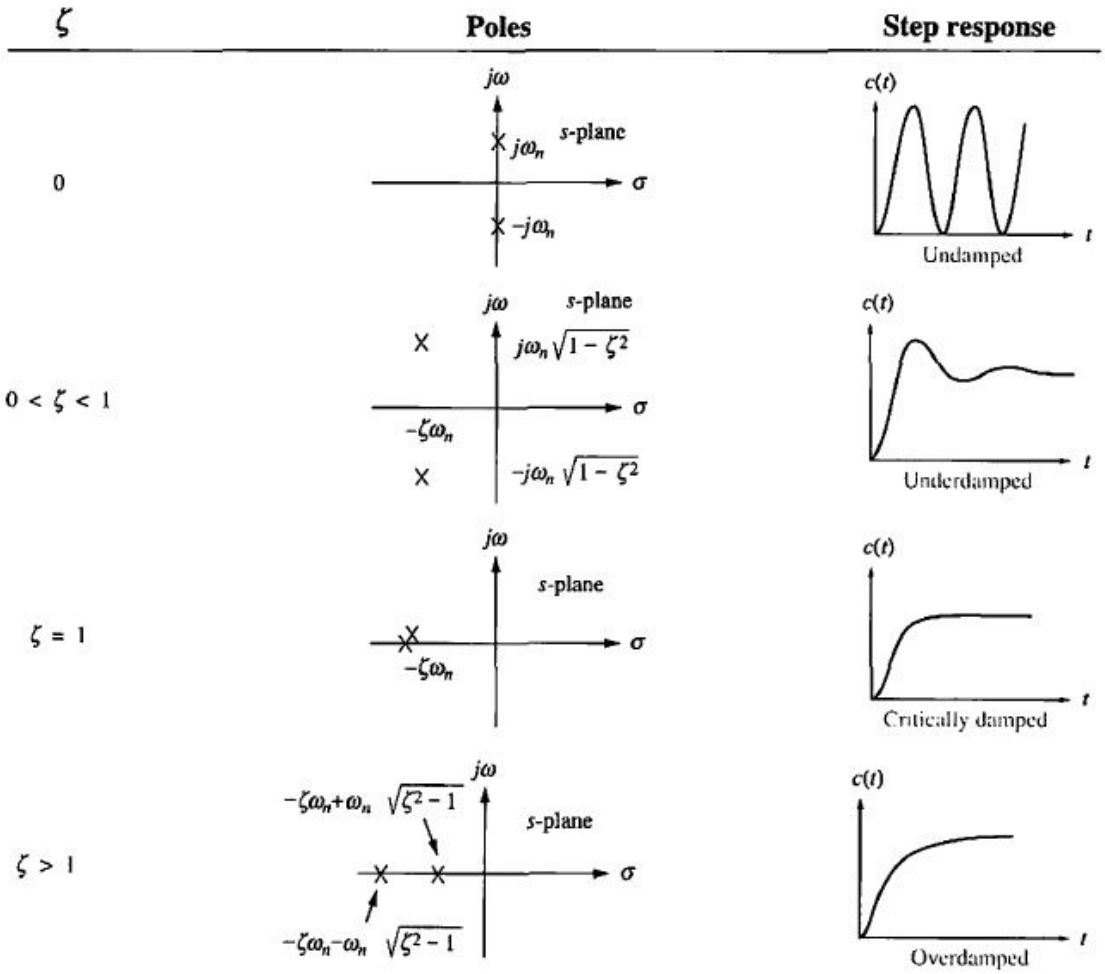


Figure 1: Second-order response as a function of damping ratio

Two significant parameters for system response are *settling time* and *percent overshoot* which can be described as follows:

Percent overshoot, %OS. The amount that the waveform overshoots the steady-state, or final value at the peak time.

$$\%\, OS = \; e^{-(\zeta\pi/\sqrt{1-\zeta^2})} \times 100 \quad (3)$$

Settling time, T_s. The time required for the transient's damped oscillations to reach and stay within ±2% of the steady-state value.

$$T_S = \frac{4}{\zeta \omega_n} \quad (4)$$

Part 1: Modeling of a dc motor

Physical setup and system equations

A common actuator in control systems is the DC motor. DC motors are widely used in numerous control applications, including robotic manipulators, tape transport mechanisms, disk drives, machine tools, and servo-valve actuators. The electric circuit diagram of the armature shown in the Fig. 2.

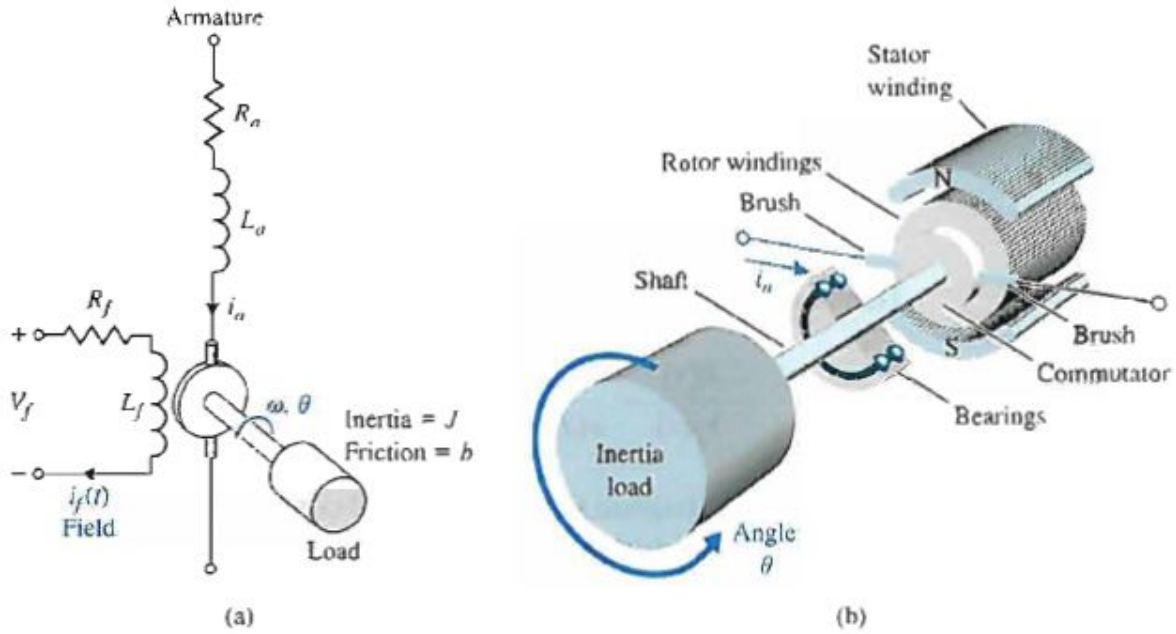


Figure 2: (a) Electrical diagram and (b) Sketch of a dc motor.

For this example, we will assume the following values for the physical parameters.

* moment of inertia of the rotor (J) = 0.01 kg.m^2/s^2
* damping ratio of the mechanical system (b) = 0.1 Nms
* electromotive force constant (K = K_e= K_t) = 0.01 Nm/Amp
* electric resistance (R) = 1 ohm
* electric inductance (L) = 0.5 H
* input (V): Source Voltage
* output (θ): position of shaft
* The rotor and shaft are assumed to be rigid

The motor torque, **T**, is related to the armature current, **i**, by a constant factor **Kt**. The back emf, **e**, is related to the rotational velocity by the following equations:

$$T = K_t \, . i$$
$$e = K_e . \dot{\theta}$$

In SI units (which we will use), **Kt** (armature constant) is equal to **Ke** (motor constant).

From the figure above we can write the following equations based on Newton's law combined with Kirchhoff's law:

$$J\ddot{\theta} + b\dot{\theta} = K.i$$
$$L\frac{di}{dt} + Ri = V - K\dot{\theta}$$

Transfer Function

Using Laplace Transforms on the above modeling equations, we can get the following open-loop transfer function where the rotational speed is the output and the voltage is the input.

$$\frac{\dot{\theta}}{V} = \frac{K}{(Js+b)(Ls+R)+K^2}$$

This is a second order system. Thus the settling time and percent overshoot can be calculated by expressing it in the standard form of Eq.(2) and using Eq.s (3) and (4).

Matlab representation

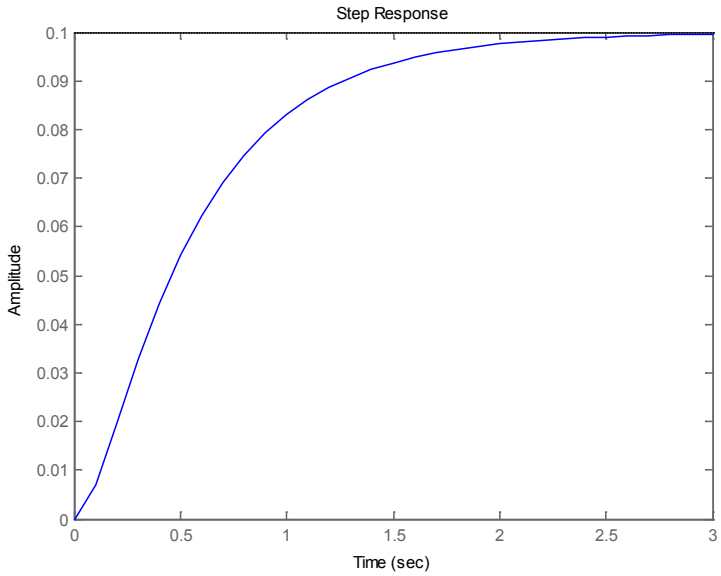
We can represent the above transfer function into Matlab. Create a new m-file and enter the following commands:

```
J=0.01;
b=0.1;
K=0.01;
R=1;
L=0.5;
num=K;
den=[(J*L)  ((J*R)+(L*b))  ((b*R)+K^2)];
open_sys = tf(num,den);
```

Open loop response: Add the following commands onto the end of the m-file and run it in the Matlab command window:

```
step (open_sys,0:0.1:3);
title('Step Response for the Open Loop System');
```

You should get the following plot:



Class Work:

Find the poles of the system $G(s) = \frac{1}{s^2-6s+25}$ by using the command *pole()*. Move the poles (a) vertically in the s-plane by keeping their real part constant (b) horizontally by keeping their imaginary part constant and (c) by keeping θ constant where $\cos\theta = \zeta$. Plot the step response of the system for all three cases and comment on settling time, peak time and percent overshoot change.

Report:

1. Determine the maximum speed that the motor can achieve and time to reach that speed when 1 volt is applied.
2. How steady state speed and settling time of the output response vary with the variation of R and input voltage?
3. Can you suggest any modification in the system if we want to increase the steady state speed and reduce the settling time simultaneously?

Part 2: PID Design Method for DC Motor Speed Control

Control systems analysis and design focuses on three primary objectives:

1. Producing the desired transient response
2. Reducing steady state errors
3. Achieving stability

A control system that provides an optimum performance without necessary adjustments is rare. Usually we find it necessary to compromise among many conflicting and demanding specifications and to adjust the system parameters to provide a suitable and acceptable performance. However, we often find that it is not sufficient to adjust a system parameter and thus obtain the desired response. Rather we are required to reconsider the control system design and insert additional components in the structure of the system.

This additional component or device that equalizes or compensates for the performance deficiency is called *compensator* or *controller*.

The schematic diagram of a unity feedback control system looks like:

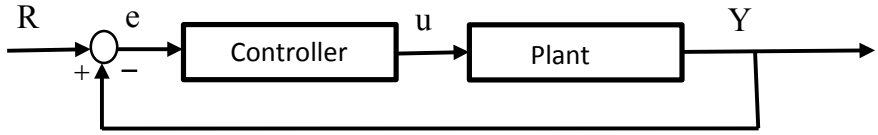


Figure 3: A unity feedback control system

Plant: A system to be controlled.
Controller: Provides the excitation for the plant; Designed to control the overall system behavior.

There are several techniques available to the control systems engineer to design a suitable controller. One of controller widely used is the **proportional plus integral plus derivative (PID) controller**, which has a transfer function:

$$K_P + \frac{K_I}{s} + K_d s$$

- K_P = Proportional gain
- K_I = Integral gain
- K_d = Derivative gain

First, let's take a look at how the PID controller works in a closed-loop system using the schematic shown above. The variable (e) represents the error, the difference between the desired input value (R) and the actual output (Y). This error signal (e) will be sent to the PID controller, and the controller computes both the derivative and the integral of this error signal. The signal (u) just past the controller is now equal to the proportional gain (Kp) times the magnitude of the error plus the integral gain (Ki) times the integral of the error plus the derivative gain (Kd) times the derivative of the error.
This signal (u) will be sent to the plant, and the new output (Y) will be obtained. This new output (Y) will be sent back to the sensor again to find the new error signal (e). The controller takes this new error signal and computes its derivative and it's integral again.
This process goes on and on.

** We should keep in mind that we do not need to implement all three controllers (proportional, derivative, and integral) into a single system, if not necessary. For example, if a PI controller gives a good enough response (like the above example), then we don't need to implement derivative controller to the system. We must keep the controller as simple as possible.

Design requirements:
As seen in the previous experiment, our uncompensated motor can only rotate at 0.1 rad/sec with an input voltage of 1 Volt. Our desired speed is 1 rad/sec for the same input voltage, while steady-state error should be less than 1%. Other performance requirement is that the motor must accelerate to its steady-state speed as soon as it turns on. In this case, we want it to have a settling time of 1 second. Since a speed faster than the reference may damage the equipment, we want to have an overshoot of less than 5%.
Summarizing the above requirements, if we simulate the reference input (r) by a unit step input, then the motor speed output should have:
• Settling time less than 1 seconds
• Overshoot less than 5%
• Steady-state error less than 1%

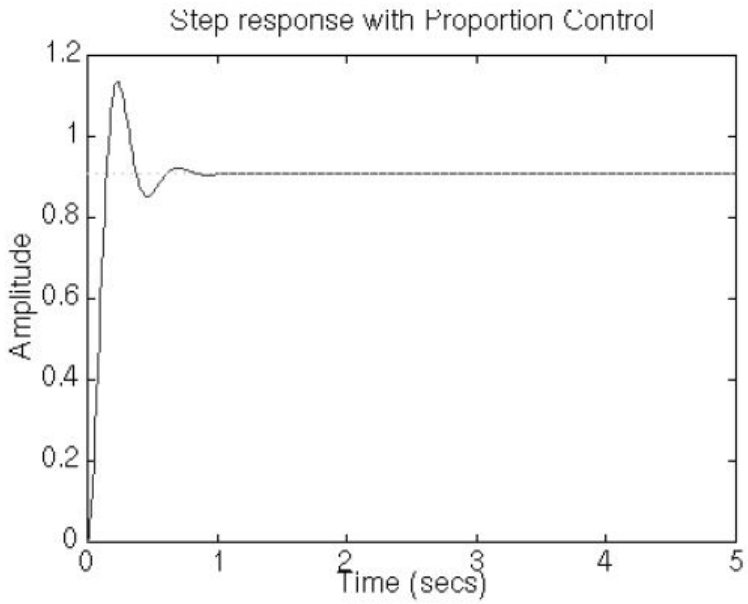
Procedures:

1. Proportional control:

In order to meet the design specifications, we need to develop a feedback system with an appropriate controller as shown in figure 3. First try using a proportional controller with a gain of kp = 100. Add the following code to the end of your m-file:

```
kp = 100;
closed_sys = feedback(open_sys*kp,1);
step (closed_sys,0:0.01:3);
title ('Step Response Proportional Control');
```

You should get the following plot:



Determine rise time, steady-state error, overshoot and settling time from the output response plot.

2. Proportional-Integral control:

From the plot above we see that both the steady-state error and the overshoot are too large. To improve the system response, let's try with a PI controller with Ki = 200 and Kp as before. Observe the effect of PI controller on rise time, overshoot, settling time and steady-state error.

3. Proportional-Derivative control:

Modify your matlab code with Kp equals 100 as before, Kd equals 10 and Ki equals 0. Run the program and note the effect of PD controller on rise time, overshoot, settling time and steady-state error.

4. Proportional-Derivative-Integral control:

Modify your matlab code with Kp equals 100, Kd equals 10 and Ki equals 200. Run the program and note the effect of PID controller on rise time, overshoot, settling time and steady-state error.

Report:

4. Give your inferences on the effects of proportional, integral and derivative control upon system performances such as rise time, overshoot, settling time and steady-state error.
5. Is it possible to redesign the above PID controller with another set of Kp, Ki and Kd ? If possible, design the controller.

References:

1. *Control Systems Engineering* by Norman S. Nise (Chapter: 4)
2. *Modern Control Systems* by Richard C. Dorf and Robert H. Bishop (Chapter: 2)

Reviewed By: Dr. Celia Shahnaz and Mrs. Fariah Hayee