CHAPTER FOUR

MATLAB PRESENTATION AND SIMULATION RESULTS

4.1 Introduction

MATLABis a useful high-level programming language which not only provides the tools for carrying out the matrix operations, but also contains several other features, such as the time-step integration of linear or nonlinear governing differential equations, which are invaluable in modern control analysis and design**.** Many people, who shied away from modern control courses because of their dread of linear algebra,began taking interest in the subject when matlab became handy. Nowadays, personal computerversions of MATLAB are commonly applied to practical problems across the board including control of aerospace vehicles, magnetically levitated trains, and even stock-market applications.

SIMULINK is a very useful Graphical Users Interface (GUI) tool for modeling control systems, and simulating their time response to specified inputs. It lets you work directly with the block-diagrams (rather than mathematical equations) for designing and analyzing control system. For this purpose, numerous linear and nonlinear blocks, input sources, and output devices are available, so that you can easily put together almost any practical control system. Another advantage of using SIMULINK is that it works seamlessly with MATLAB and can draw upon the vast programming features and function library of MATLAB. A SIMULINK block-diagram can be converted into a MATLAB program called (M-file). We will be using SIMULINK as a design and analysis tool especially in simulating the response of a control system designed with MATLAB[10].

Modern control design has heavy computing requirements. In particular one needs to:

1. manipulate symbolic algebraic expressions.
2. perform intensive numerical calculations and simulations for proto-typing and testing quickly and reliably.
3. to implement the controller at high speed in special hardware such as an embedded controller or a digital signal processing (DSP) chip perhaps using assembler.

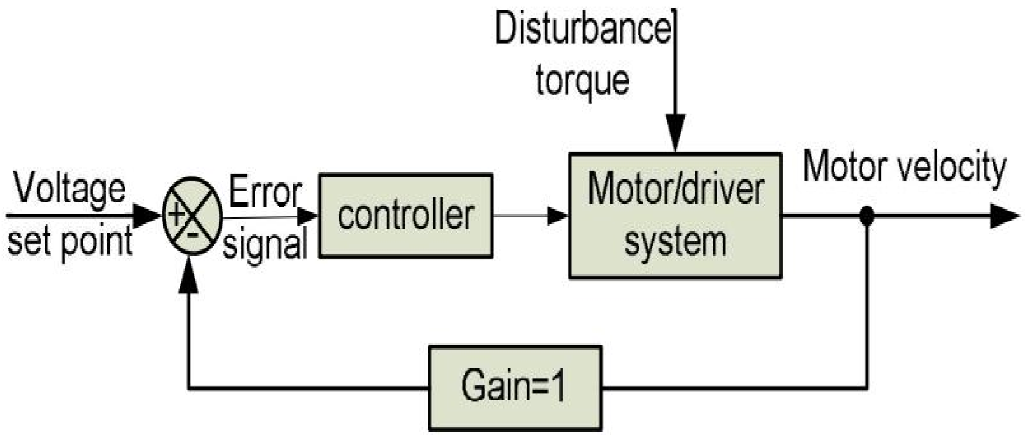

Figure 4.1: shows the block diagram of closed loop control for typical DC motor

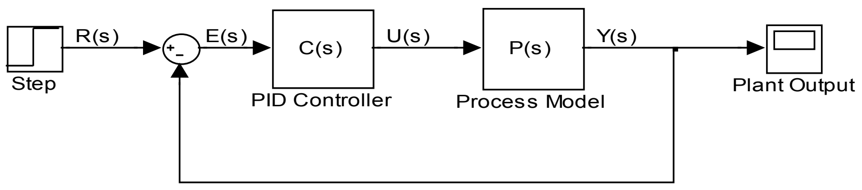

Figure 4.2: show the simple block diagram of the PID controller in MATLAB SIMULINK

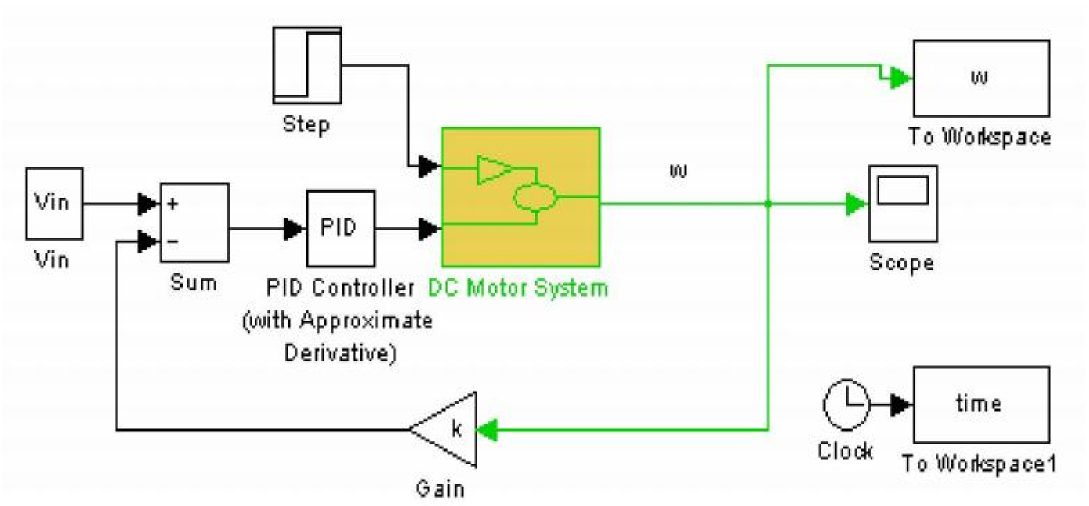

Figure 4.3: shows the simulation diagram of control DC motor by using PID controller.

4.2 Simulation

We will assume that the input of the system is the voltage source (*V*) applied to the motor's armature, while the output is the rotational speed of the shaft d(*theta*)/dt. The rotor and shaft are assumed to be rigid. We further assume a viscous friction model, that is, the friction torque is proportional to shaft angular velocity.

Table (4.1) physical parameters for the model

Moment of inertia of the rotor (J)	0.01 kg.m^2
Motor viscous friction constant (b)	0.10N.m.s
Electromotive force constant (Ke)	0.01 V/rad/sec
Motor torque constant (Kt)	0.01N.m/Amp
Electric resistance (R)	1.00 Ohm
Electric inductance (L)	0.50 H

4.3 Design requirements

First consider that our uncompensated motor rotates at 0.1 rad/sec in steady state for an input voltage of 1 Volt (this is demonstrated in the DC Motor

Speed: System Analysis page where the system's open-loop response is simulated). Since the most basic requirement of a motor is that it should rotate at the desired speed, we will require that the steady-state error of the motor speed be less than 1%. Another performance requirement for our motor is that it must accelerate to its steady-state speed as soon as it turns on. In this case, we want it to have a settling time less than 2 seconds. Also, since a speed faster than the reference may damage the equipment, we want to have a step response with overshoot of less than 10%.

In summary, for a unit step command in motor speed, the control system's output should meet the following requirements.

1. Settling time less than 2 seconds
2. Overshoot less than 10%
3. Steady-state error less than 1%

4.4 MATLAP representation

Transfer Function

We can represent the above open-loop transfer function of the motor in MATLAB by defining the parameters and transfer function as follows. Running this code in the command window produces the output shown below:

```
J = 0.01;
b = 0.1;
K = 0.01;
R = 1;
L = 0.5;
s = tf('s');
P_motor = K/((J*s+b)*(L*s+R)+K^2)
```

```
P_motor =
             0.01
  ---------------------------
  0.005 s^2 + 0.06 s + 0.1001
```

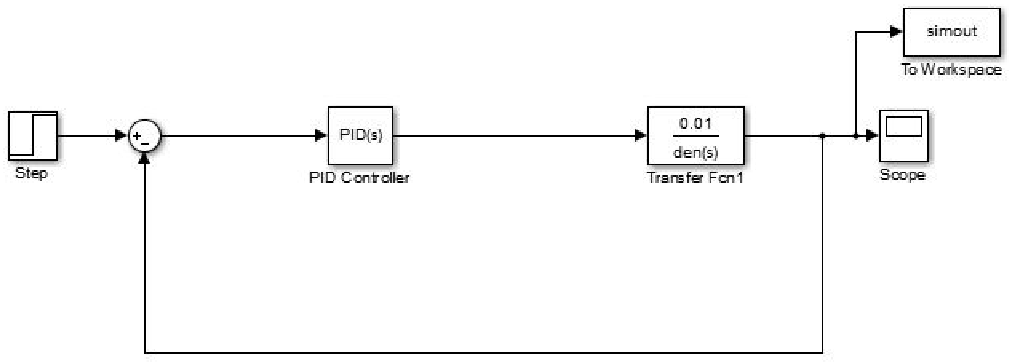

Figure 4.4: Simulink module of a DC Motor

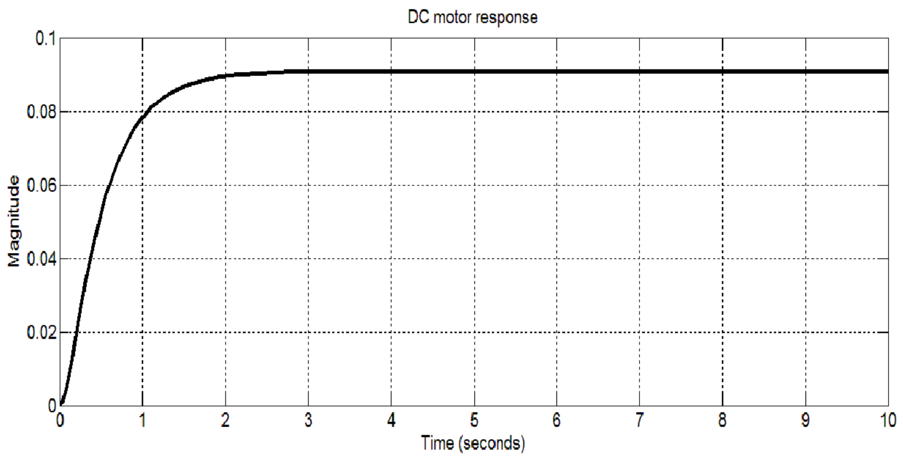

Figure 4.5: Speed response of a DC Motor without using the controller

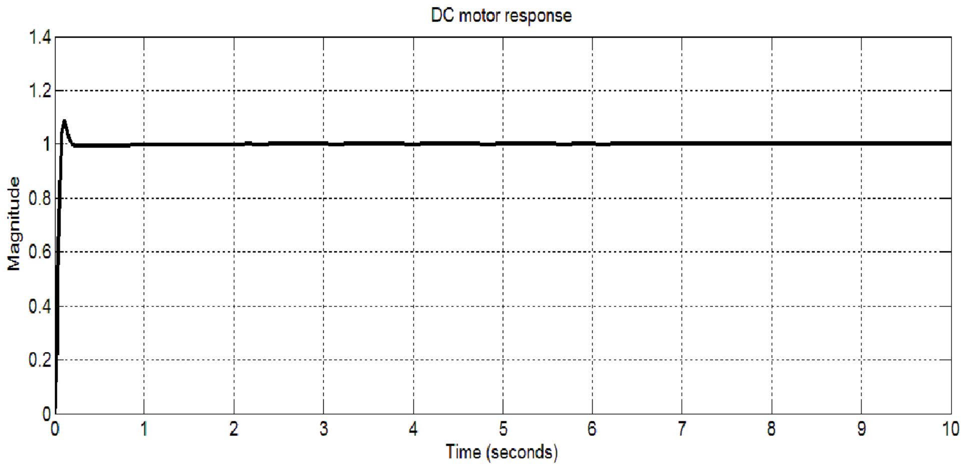

Figure 4.6: Speed response of a DC Motor after using the controller