Jawaharlal Nehru Engineering College

Laboratory Manual

CONTROL SYSTEM -I

For

Third Year Students

PREFACE

It is my great pleasure to present interactive Control system components(csc) Demonstration modules developed for third year Electrical Engineering students.

Hence ,this manual consists of a ready to use set of demonstrations, illustrating the CSC concepts ,that can be beneficial to the students . Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions.

SUBJECT INDEX

1. DOs and DON'Ts in Laboratory:

Do not handle   different kit without reading the instructions/Instruction manuals

1. Go through through the procedure and precautions given in manual.
4. Strictly observe the instructions given by the teacher/Lab Instructor.

2 Instruction for Laboratory Teachers::

1. Lab work completed during prior session ,should be corrected during the next lab session.

2. Students should be guided and helped whenever they face difficulties.

3. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

EXPERIMENT No. 1

OBJECTIVE: -To use Potentiometer as an error detector

APPARATUS REQUIRED: -

- CRO
- Connecting leads
- Experimental Kit

THEORY: --A Potentiometer is an Electromechanical Transducer which converts angular or linear displacement into proportional electrical v/g. When a reference v/g applied across the fixed terminals of the Potentiometer the o/p v/g measured at variable terminal is proportional to the input displacement.

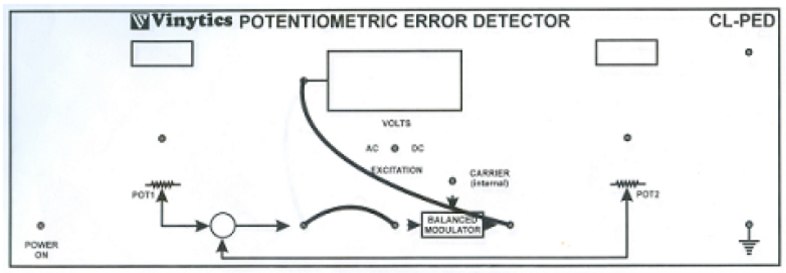


Fig.1: Connection Diagram for AC Excitation

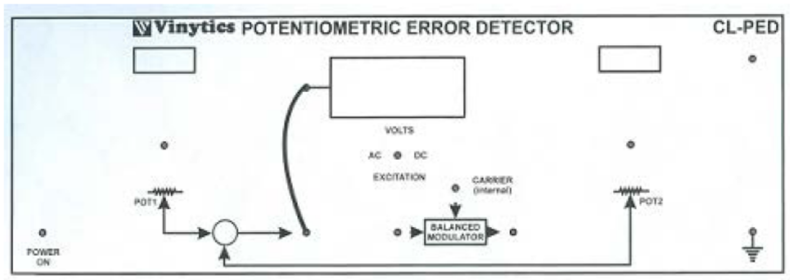


Fig.2: Connection Diagram for DC Excitation

PROCEDURE: -

For DC Excitation.

1. Connect the components as shown in figure.
2. Set the excitation switch to DC.
3. Keep pot 2 fixed at any position and do not disturb it position. Let this be θ_2.
4. Turn Pot1 in step of 20^0 (at one degree interval when there is a sudden change in v/g)
5. Plot Vo vs. θ_1.Observe Linearity and Rang
6. Repeat for another position of Pot 2.

Error Detector Coefficient:-

$$\text{Error Detector Coefficient} = \text{Slope } (Ke) = \frac{\text{Change in o/p v/g } (\Delta Vo)}{\text{Change in shaft position } (\Delta\theta_e)}$$

Where $\theta_e = \theta_1 - \theta_2$

For AC Excitation

1. Display the carrier on the CRO and measure its amplitude and frequency.
2. Switch the excitation to AC now and observe Vo on the CRO while turning either Pot 1 or Pot 2 very slowly. Use the internal carrier for external triggering of the CRO. Notice and record how phase of Vo change when $\theta_e = (\theta 1 - \theta_{2)}$ change sign.
3. Record and plot peak to peak (or r.m.s) Vo as a function of θe. Note that information about the sign of θe is lost.
4. Next connect Vo to the input the BALANCE MODULATOR and its out put to the DVM.
5. Record and plot the demodulator output V_{DEM} as a function of θe. . Note that information about the sign of θe is restored. It may be noted that a non zero DC v/g is present for $\theta_e = 0$

OBSERVATION AND CALCULATION: -

Note down the value of θ_1 & θ_2 for every position of pot 1 in degrees. Also note down the value of o/p v/g (Vo) in volts.

1.7 Observation Table: -

For DC Excitation

S.No	Pot-1 position $\theta_1$degree	Pot-2 position $\theta_2=180^0$	$\theta_e = \theta_1 - \theta_2$	Output v/g (Vo)

For AC Excitation

Pot position $\theta_2=180^0$

S.No	Pot1 position $\theta_1$degree	Vo (r m s) mv	$\theta_e = \theta_1 - \theta_2$	Output v/g (Vo)

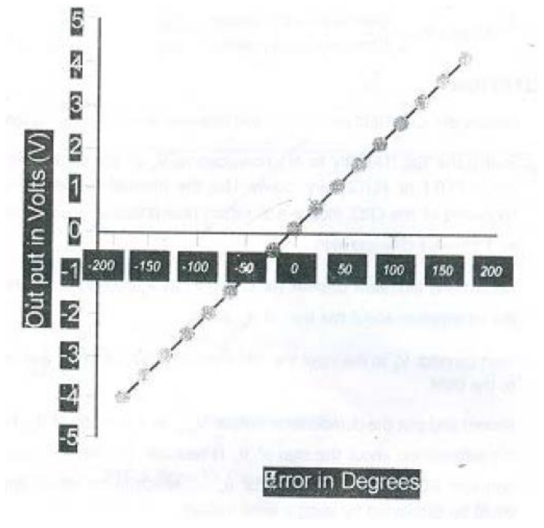


Fig.3: Error Vs output with AC excitation

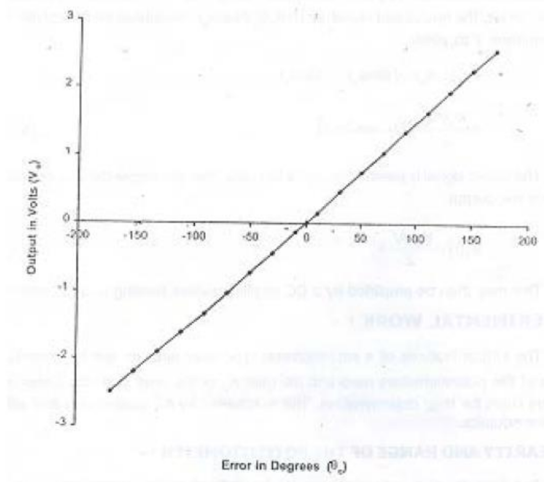


Fig.4: Error Vs. Output with DC excitation

RESULT:

Graph of input and output characteristic has been observed.

<u>POINTS FOR DISCUSSION</u>:-

1. What is Potentiometer?
2. What is Resolution?
3. What is Linearity?
4. Application of Potentiometer.
5. Type of Potentiometer.
6. Life of Potentiometer.

2.SYNCHRO CHARACTERISTICS

Aim: Study of synchro characteristics.

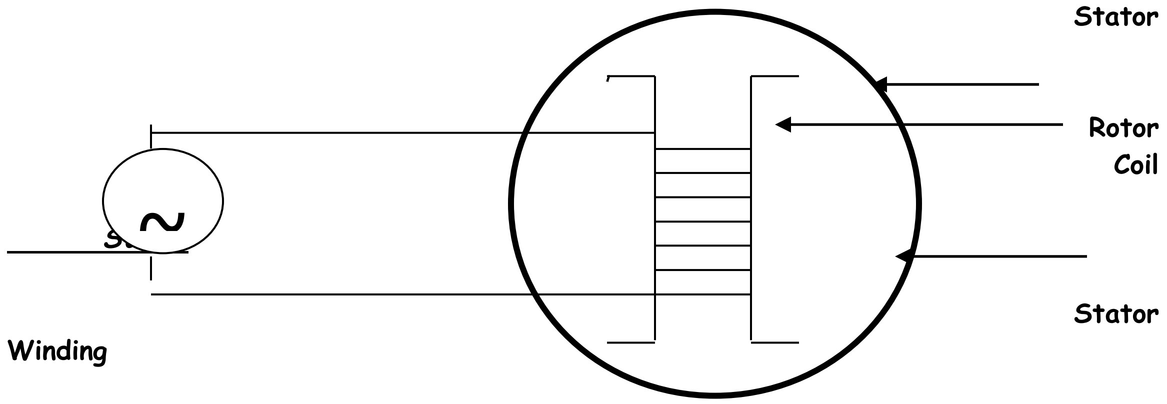
a) Study of synchro transmitter characteristics.
b) Study of synchro transmitter / receiver characteristics.

Apparatus: Synchro transmitter / receiver set-up, multimeter comnnecting wires etc.

Theory:

Principal: A synchro is an electromagnetic transducer commonly used to convert an angular position of a shaft into an electric signal.
The basic synchro unit is usually called a synchro transmitter. Its construction is similar to that of a three- phase alternator. The stator is of laminated silicon steel and is slotted to accommodate a balanced three phase winding which is usually of concentric coil type and is Y-connected. The rotor is of dumb-bell construction and is wound with a concentric coil. An a.c. voltage is applied to the rotor winding through slip rings.

Constructional features of synchro transmitter:
fig.-a



Operation : Let an a.c. voltage

$$V_r (t) = V_r \sin w_c t$$

fig-b

Be applied to the rotor coil which produces a sinusoid ally time varying flux directed along its axis and distributed nearly sinusoid ally in the air gap along the stator periphery. As the air gap flux is sinusoid ally distributed, the flux linking any stator coil is proportional to the cosine of the angle between the rotor and stator coil axes and so is the voltage induced in each stator coil.
Let Vs1n, Vs2n and Vs3n.
Be the voltage induced in the stator coil S1,S2 and S3 with respect to the neutral.

Vs1n = KVr sin wct cos (0-120)
Vs2n = = KVr sin wct cos 0
Vs3n = KVr sin wct cos / (0+240)
The three terminal voltages of the stator are.
Vs1s2 = Vs1n – Vs1n = / 3KVr sin (0+ 240) sin wct.
Vs2s3 = Vs3n – Vs3n = / 3KVr sin (0+120) sin wct.
Vs3s1 = Vs3n – Vs1n = / 3KVr sin 0 sin wct.

Procedure:

(A):

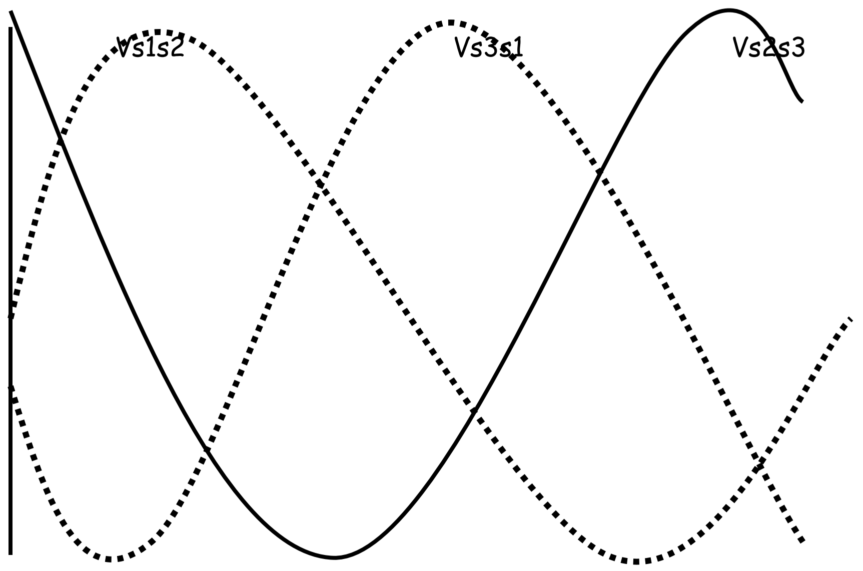
1. Connect the system in main supply.
2. Starting from zero position note down the voltage between stator terminal i.e. Vs2s1,Vs1s3 and Vs2s3 in a sequential position.
3. Plot the graph of angular position Vs voltage of three winging i.e. terminal voltage.

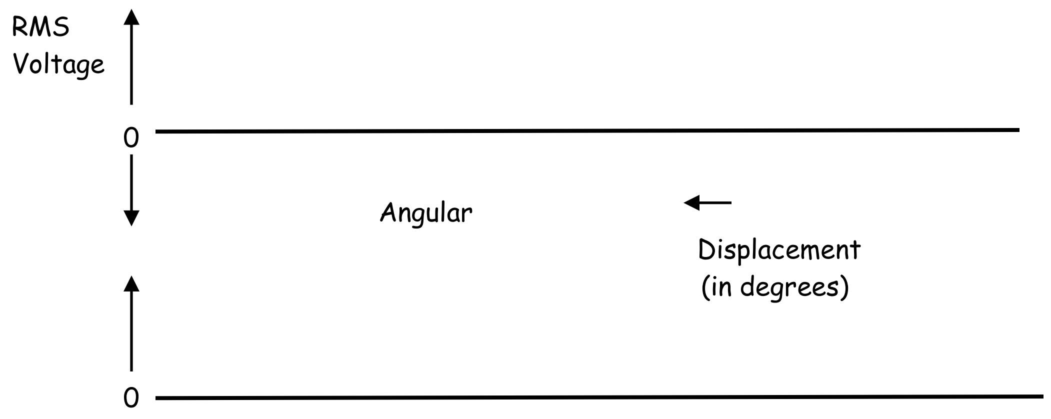
(B):

1. Connect the system to the main supply .
2. Make connections between corresponding terminals of transmitter and receiver i.e. connect S1 – S1, S2 – S2 and S3 – S3 of transmitter and receiver.
3. Switch on SW1 and SW2.
4. Move the graph of angular displacement of transmitter to angular displacement in receiver.

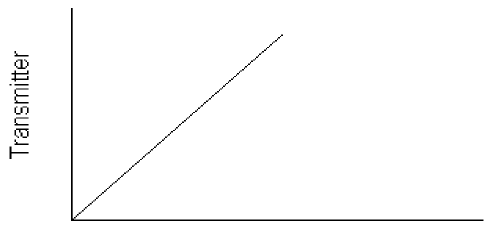
Nature of characteristics :

(A):



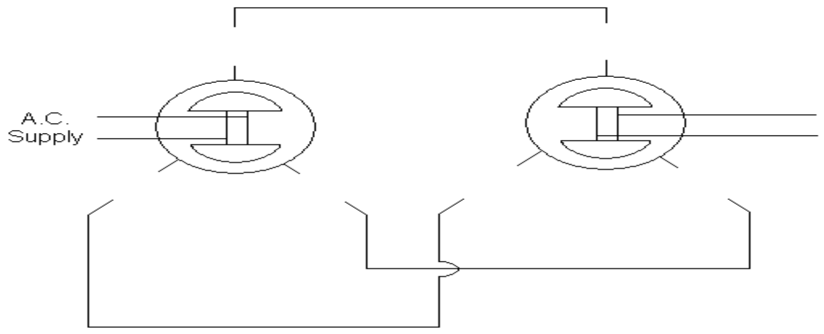


b)



NOTE: -Refer Nagrath Gopal Book

Part-b diagram.



Observation tables: -

(A)

Angular Displacement	VS1S2(V)	VS2S3(V)	VS3S1(V)

Angular Displacement	Receiver Position

(B)

Conclusion: **-(a) If angular displacement is changed depending on rotor position the voltages are induced in stator coils.**

(b)If Transmitter position is changed then receiver position is changed accordingly.

EXPERIMENT NO.

AIM:-Transfer function of armature control D.C. servomotor and A.C. servomotor

Appratus required :-

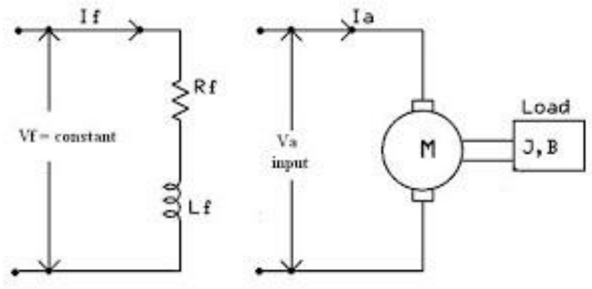
a) armature control dc servomotor.

S. No	**Description**	**Range**	**Type**	**Quantity**
1.	**DC servo motor trainer kit**	**-**		**1**
2.	**DC servo motor**			**1**
3.	**Rheostat**	**500Ω/1A**		**1**
4.	**Ammeter**	**(0-1)A**	**MC**	**1**
		(0-100) mA	**MI**	**1**
5.	**Voltmeter**	**(0–300) V**	**MC**	**1**
		(0–75) V	**MI**	**1**
6.	**Stopwatch**	**-**		**1**
7.	**Patch cords**	**-**		**As required**

THEORY:

In servo applications a DC motor is required to produce rapid accelerations from standstill. Therefore the physical requirements of such a motor are low inertia and high starting torque. Low inertia is attained with reduced armature diameter with a consequent increase in the armature length such that the desired power output is achieved. Thus, except for minor differences in constructional features a DC servomotor is essentially an ordinary DC motor. A DC servomotor is a torque transducer which converts electrical energy into mechanical energy. It is basically a separately excited type DC motor. The torque developed on the motor shaft is directly proportional to the field flux and armature current, $T_m = K_m \Phi I_a$. The back emf developed by the motor is $E_b = K_b \Phi \omega_m$.. In an armature controlled DC Servo motor, the field winding is supplied with constant current hence the flux remains constant. Therefore these motors are also called as constant magnetic flux motors. Armature control scheme is suitable for large size motors.

ARMATURE CONTROLLED DC SERVOMOTOR:



FORMULAE USED:

Transfer function of the armature controlled DC servomotor is given as
$\theta(s) / V_a(s) = K_m / [s (1+s\tau_a)(1+s\tau_m) + (K_b K_t / R_a B)]$
where

Motor gain constant, $K_m = (K_t / R_a B)$

Motor torque constant, $K_t = T / I_a$
Torque, T in Nm = $9.55 E_b I_a$
Back emf, E_b in volts = $V_a - I_a R_a$
V_a = Excitation voltage in volts

Back emf constant, $K_b = V_a / \omega$

Angular velocity ω in rad/ sec = $2\pi N / 60$

Armature time constant, $\tau_a = L_a / R_a$

Armature Inductance, L_a in H= $X_{La} / 2\pi f$
X_{La} in $\Omega = \sqrt{(Z_a^2 - Ra^2)}$
Z_a in $\Omega = V_{a2} / I_{a2}$
Armature resistance,R_a in $\Omega = V_{a1} / I_{a1}$
Mechanical time constant, $\tau_m = J / B$

Moment of inertia, J in Kg m^2 / rad = $\dfrac{W \times (60 / 2\pi)^2 \times dt/dN}{N}$

Stray loss, W in Watts = W' x [t2 / (t1-t2)]
Power absorbed, W' in watts = $V_a I_a$
t2 is time taken on load in secs
t1 is time taken on no load in secs
dt is change in time on no load in secs
dN is change in speed on no load is rpm
N is rated speed in rpm

Frictional co-efficient, B in N-m / (rad / sec) = $W'' / (2\pi N / 60)^2$
W'' = 30 % of Constant loss
Constant loss = No load i/p – Copper loss
No load I/P = $V (I_a + I_f)$
Copper loss = $I_a^2 R_a$
N is rated speed in rpm

PROCEDURE:

1. To determine the motor torque constant K_t and Back emf constant K_b:

- **Check whether the MCB is in OFF position in the DC servomotor trainer kit**
- **Press the reset button to reset the over speed.**
- **Patch the circuit as per the patching diagram.**
- **Put the selection button of the trainer kit in the armature control mode.**
- **Check the position of the potentiometer; let it initially be in minimum position.**
- **Switch ON the MCB.**
- **Vary the pot and apply rated voltage of 220 V to the armature of the servomotor.**
- **Note the values of the armature current I_a, armature voltage V_a, and speed N.**
- **Find the motor torque constant K_t and Back emf constant K_b using the above values.**

Note:
If the voltmeter and ammeter in the trainer kit is found not working external meters of suitable range can be used.

OBSERVATIONS:

S. No.	Armature Voltage, V_a (V)	Armature Current, I_a (A)	Speed, N (rpm)

b)Determination of transfer function of A.C.servomotor

Appratus required:-

S. No	Description	Range	Type	Quantity
1.	**AC servo motor trainer kit**	**-**		**1**
2.	**AC servo motor**			**1**
3.	**Ammeter**	**(0-1) A**	**MC**	**1**
		(0-100) mA	**MI**	**1**
4.	**Voltmeter**	**(0–300) V**	**MC**	**1**
		(0–75) V	**MI**	**1**
5.	**Patch cords**	**-**		**As required**

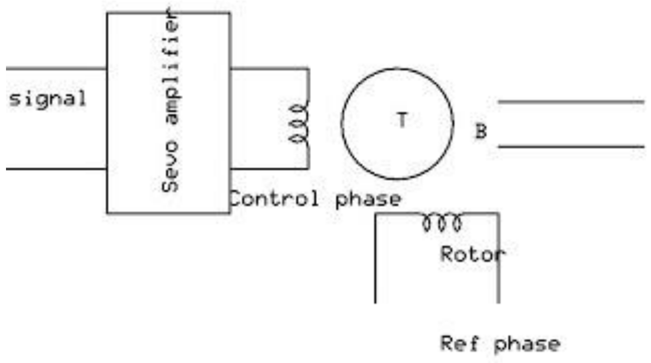
THEORY:

An AC servo motor is basically a two phase induction motor with some special design features. The stator consists of two pole pairs (A-B and C-D) mounted on the inner periphery of the stator, such that their axes are at an angle of 90^o in space. Each pole pair carries a winding, one winding is called reference winding and other is called a control winding. The exciting current in the winding should have a phase displacement of 90^o. The supply used to drive the motor is single phase and so a phase advancing capacitor is connected to one of the phase to produce a phase difference of 90^o.The rotor construction is usually squirrel cage or drag-cup type. The rotor bars are placed on the slots and short-circuited at both ends by end rings. The diameter of the rotor is kept small in order to reduce inertia and to obtain good accelerating characteristics. The drag cup construction is employed for very low inertia applications. In this type of construction the rotor will be in the form of hollow cylinder made of aluminium. The aluminium cylinder itself acts as short-circuited rotor conductors. Electrically both the types of rotor are identical.

WORKING PRINCIPLE :

The stator windings are excited by voltages of equal magnitude and 90^o phase difference. These results in exciting currents i_1 and i_2 that are phase displaced by 90^o and have equal values. These currents give rise to a rotating magnetic field of constant magnitude. The direction of rotation depends on the phase relationship of the two currents (or voltages). This rotating magnetic field sweeps over the rotor conductors. The rotor conductor experience a change in flux and so voltages are induced rotor conductors. This voltage circulates currents in the short-circuited rotor conductors and currents create rotor flux. Due to the interaction of stator & rotor flux, a mechanical force (or torque) is developed on the rotor and so the rotor starts moving in the same direction as that of rotating magnetic field.

GENERAL SCHEMATIC OF AC SERVOMOTOR:



<!--[endif]-->

FORMULAE USED:

Transfer function, $G_m(s) = K_m / (1 + s\tau_m)$

Where

Motor gain constant, $K_m = K / F_O + F$

K is $\Delta T / \Delta C$
F_O is $\Delta T / \Delta N$
Torque, T is 9.81 X R (S1 ~ S2)
R is radius of the rotor in m
Frictional co-efficient, $F = W / (2\pi N / 60)^2$
Frictional loss, W is 30 % of constant loss in Watts
Constant loss in watts = No load input – Copper loss
No load i/p = $V (I_R + I_C)$
V is supply voltage, V
I_R is current through reference winding, A
I_C is current through control winding, A
Copper loss in watts = $I_C^2 R_C$
$R_C = 174\Omega$
N is rated speed in rpm

Motor time constant, $\tau_m = J / F_O + F$

Moment of inertia J is $\pi d^4 L_R \rho / 32$
d is diameter of the rotor in m (Given d =39.5 mm)
L_R is length of the rotor in m (Given L_R =76 mm)

ρ is density = 7.8×10^2 gm / m

PROCEDURE:

1. DETERMINATION OF FRICTIONAL CO-EFFICIENT, F

1. **Check whether the MCB is in OFF position.**
2. **Patch the circuit using the patching diagram.**
3. **Switch ON the MCB**
4. **Vary the control pot to apply rated supply voltage**
5. **Note the control winding current, reference winding current, supply voltage and speed.**
6. **Find the frictional co-efficient using the above values**

OBSERVATIONS:

S. No.	Supply Voltage V (V)	Control winding Current I_c (A)	Reference Winding Current I_r (A)	Speed N (rpm)

CALCULATIONS:

Determination Of Transfer Function Of AC Servo Motor

Experiment No.

Time Domain Specifications

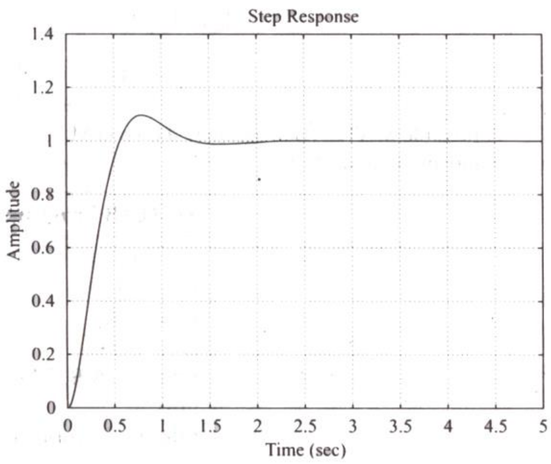
Aim : Obtaining Rise Time, Peak Time, Maximum Overshoot, & Setting Time with MATLAB.
MATLAB can conveniently be used to obtain the rise time, peak time, maximum overshoot, & setting time. Consider the system defined by

$$\frac{C(s)}{R(s)} = \frac{25}{s^2 + 6s + 25}$$

MATLAB program 1 yields rise time, peak time, maximum overshoot, & setting time. A unit-step response curve for this system is given in Figure 1 to verify the result obtained with MATLAB Program 1

MATLAB Program 1

```
%------This is a MATLAB program to find the rise time, peak time, %
maximum overshoot, & setting time  of the second-order system % &
higher order system ------
%------In this excample, we assume zeta = 0.6 and wn = 5 ------
num = [0 0 25];
den = [1 6 25];
t =0:0.005:5;
[y,x,t] = step(num,den,t);
r = 1; while y(r) < 1.0001;r = r + 1; end;
rise_time = (r – 1)*0.005
rise_time =
     0.5550
[ymax,tp] = max(y);
peak_time = (tp – 1)*0.005
peak_time =
    0.7850
max_overshoot = ymax –1
max_overshoot =
    0.0948
s = 1001; while y(s)> 0.98 & y(s) < 1.02; s = s – 1; end;
setting_time = (s-1)*0.005
setting_time =     1.1850
```



Experiment No.

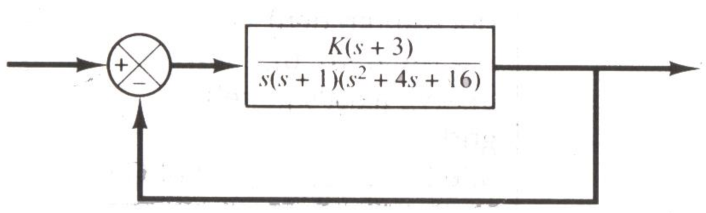
Root Locus Plot - A

Aim : Plotting root locus using MATLAB

Concider the system shown in figure 6. plot root loci with a square aspect ratio so that a line with slope 1 is a true 45 line. Choose the region of root-locus plot to be

$$-6 \le x \le 6, \qquad -6 \le x \le 6$$

where x & y are the real-axis coordinate & imaginery-axis coordinet, respectivly.



To set the given plot region on the screen to be square, enter the command

```
V = [-6   6  -6  6]; axis (v); axis('square')
```

With this command, the region of the plot is as specified & the line with slope 1 is at a true 45^0, not skewed by the irreguler shape of the screen.

For this problem, the dominator is given as a product as first and second order terms. So we mus multiply this turms to get polyomial in s. the multiplication of this terms can be done easily by use of the convolution command as shown next.

Define :

A = s (s + 1); a = [1 1 0]

B = s^2 + 4s + 16; b = [1 4 16]

Than we use the following command :

```
C = conv(a, b)
```

Note the conv(a, b) gives the product of two polynomial a & b. see the following comuter output:

```
a = [1   1   0];
b = [1   4   16];
c = conv (a, b)
c =
    1 5  20  16  0
```

The denominator polynomials is thus found to be

```
den = [1 5  20  16  0 ]
```

to find the complex-conjugaate open-loop poles (the root of s^2 + 4s. + 16 = 0), we may enter the root command as follows:

```
R = root(b)
R =
      -2.0000 + 3.464li
      -2.0000 - 3.464li
```

Thus, the system has the following open loop zero & open loop poles:

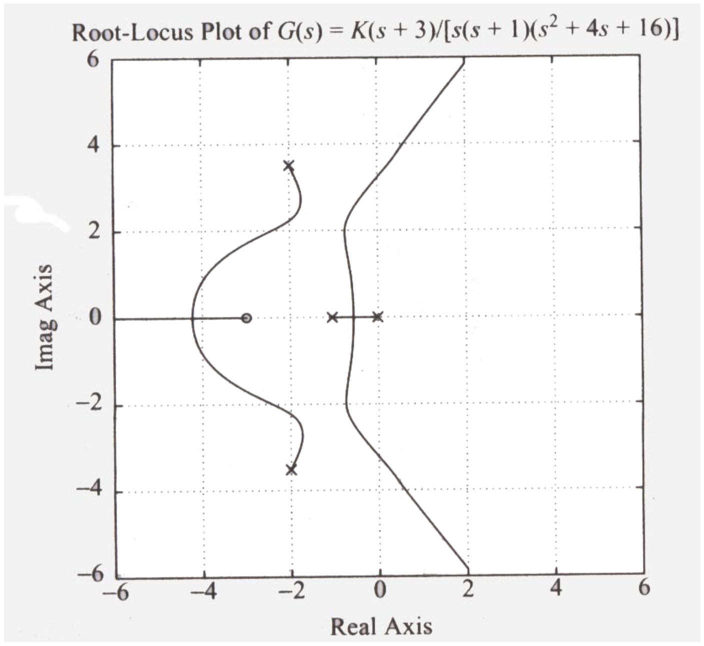
Open loop zero: $s = -3$

Open loop zero: $s = 0$, $s = -1$, $s = -2 \pm j3.4641$

MATLAB Program 6 will plot the root locus digram for this system. Yhe plot is showen in Figure 6

MATLAB Program 6

%-------- Root-locus plot -------
num = [0 0 0 1 3];
den = [1 5 20 16 0];
recous(num,den)
v = [-6 6 -6 6];
axius(v); axis('square')
grid;
title ('Root-Locus Plot of G(s) = K(s + 3)/[s(s + 1)(s^2 + 4s + 16)]')

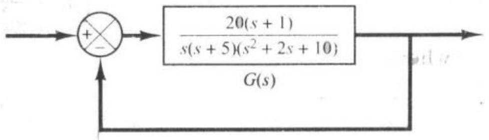


Experiment No.

BODE PLOT:

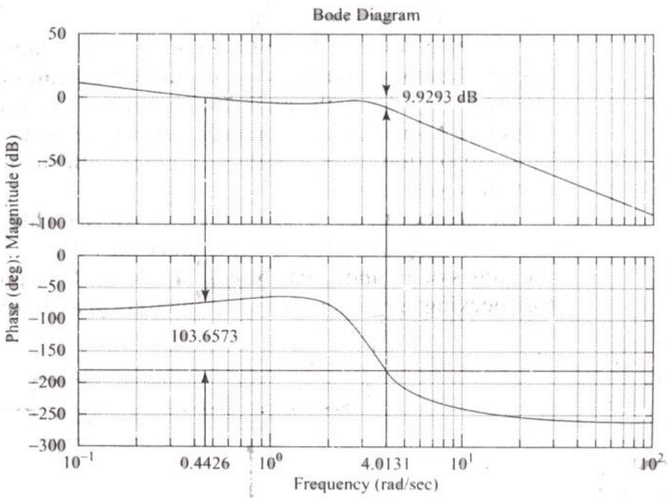
Draw a Bode diagram of the open-loop transfer function $G(s)$ of the closed-loop system shown in Figure **13** Determine the gain margin, phase margin, phase-crossover frequency, and gain-crossover frequency with MATLAB.

A MATLAB program to plot a Bode diagram and to obtain the gain margin, phase margin, phase-crossover frequency, and gain-crossover frequency is shown in MATLAB Program **13** The Bode diagram of $G(s)$ is shown in Figure



MATLAB Program 13

```
num = [0  0  0  20  20];
den = conv([1  5  0],[1  2  10]);
sys = tf(num,den);
w = logspace(-1,2,100);
bode(sys,w)
[Gm,pm,wcp,wcg] = margin(sys);
GmdB = 20*log10(Gm);
[GmdB  pm  wcp  wcg]

ans =

   9.9293  103.6573  4.0131  0.4426
```



PID Controller Design

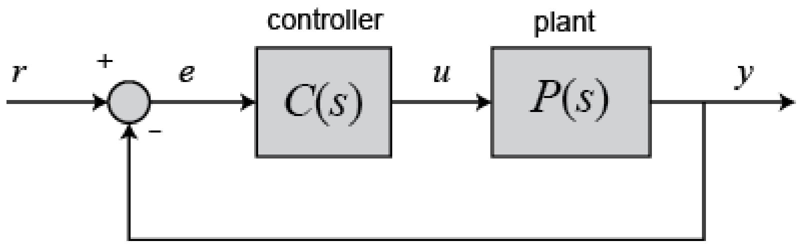
In this tutorial we will introduce a simple yet versatile feedback compensator structure, the Proportional-Integral-Derivative (PID) controller. We will discuss the effect of each of the pid parameters on the closed-loop dynamics and demonstrate how to use a PID controller to improve the system performance.

Key MATLAB commands used in this tutorial are: `tf` , `step` , `pid` , `feedback` , `pidtool` , `pidtune`

Contents

PID Overview

In this tutorial, we will consider the following unity feedback system:



The output of a PID controller, equal to the control input to the plant, in the time-domain is as follows:

(1) $$u(t) = K_p e(t) + K_i \int e(t)dt + K_p \frac{de}{dt}$$

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 tracking 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 control signal (u) to the plant is equal to 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.

This control signal (u) is sent to the plant, and the new output (y) is obtained. The new output (y) is then fed back and compared to the reference to find the new error signal (e). The controller takes this new error signal and computes its derivative and its integral again, ad infinitum.

The transfer function of a PID controller is found by taking the Laplace transform of Eq.(1).

(2) $$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

K_p= Proportional gain K_i= Integral gain K_d= Derivative gain

We can define a PID controller in MATLAB using the transfer function directly, for example:

```
Kp = 1;
Ki = 1;
Kd = 1;

s = tf('s');
C = Kp + Ki/s + Kd*s
C =

  s^2 + s + 1
  -----------
       s

Continuous-time transfer function.
```

Alternatively, we may use MATLAB's **pid controller object** to generate an equivalent contin

```
C = pid(Kp,Ki,Kd)
C =

             1
  Kp + Ki * --- + Kd * s
             s

  with Kp = 1, Ki = 1, Kd = 1

Continuous-time PID controller in parallel form.
```

Let's convert the pid object to a transfer function to see that it yields the same result as above:

```
tf(C)
ans =

  s^2 + s + 1
  -----------
       s

Continuous-time transfer function.
```

The Characteristics of P, I, and D Controllers

A proportional controller (K_p) will have the effect of reducing the rise time and will reduce but never eliminate the **steady-state error**. An integral control (K_i) will have the effect of eliminating the steady-state error for a constant or step input, but it may make the transient response slower. A derivative control (K_d) will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response.

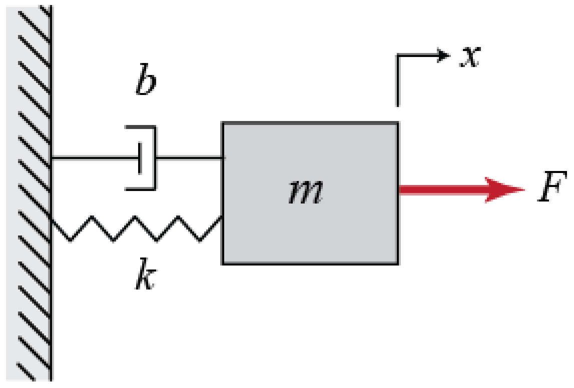
The effects of each of controller parameters, K_p, K_d, and $K_i$on a closed-loop system are summarized in the table below.

CL RESPONSE	**RISE TIME**	**OVERSHOOT**	**SETTLING TIME**	**S-S ERROR**
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	No Change

Note that these correlations may not be exactly accurate, because K_p, K_i, and $K_d$are dependent on 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.

Example Problem

Suppose we have a simple mass, spring, and damper problem.



The modeling equation of this system is

(3) $M\ddot{x} + b\dot{x} + kx = F$

Taking the Laplace transform of the modeling equation, we get

(4) $Ms^2X(s) + bsX(s) + kX(s) = F(s)$

The transfer function between the displacement $X(s)$ and the input $F(s)$ then becomes

(5) $\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + bs + k}$

Let

```
M = 1 kg
b = 10 N s/m
k = 20 N/m
F = 1 N
```

Plug these values into the above transfer function

(6) $\frac{X(s)}{F(s)} = \frac{1}{s^2 + 10s + 20}$

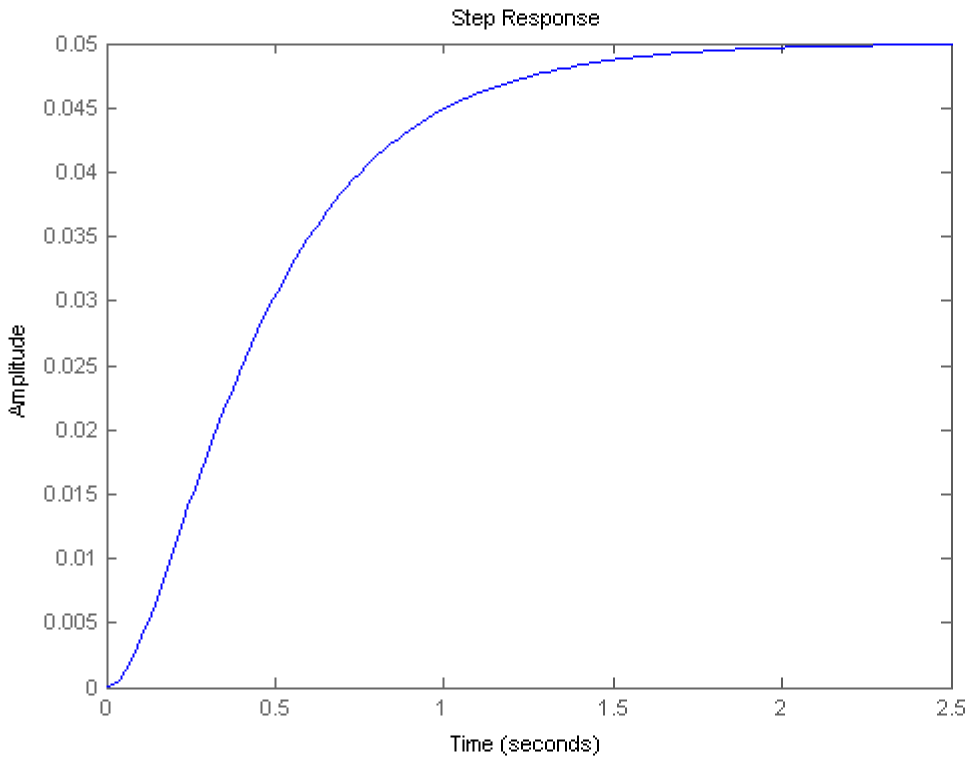
The goal of this problem is to show you how each of K_p, K_i and K_d contributes to obtain

```
Fast rise time
Minimum overshoot
No steady-state error
```

Open-Loop Step Response

Let's first view the open-loop step response. Create a new m-file and run the following code:

```
s = tf('s');
P = 1/(s^2 + 10*s + 20);
step(P)
```



The DC gain of the plant transfer function is 1/20, so 0.05 is the final value of the output to an unit step input. This corresponds to the steady-state error of 0.95, quite large indeed. Furthermore, the rise time is about one second, and the settling time is about 1.5 seconds. Let's design a controller that will reduce the rise time, reduce the settling time, and eliminate the steady-state error.

Proportional Control

From the table shown above, we see that the proportional controller (Kp) reduces the rise time, increases the overshoot, and reduces the steady-state error.

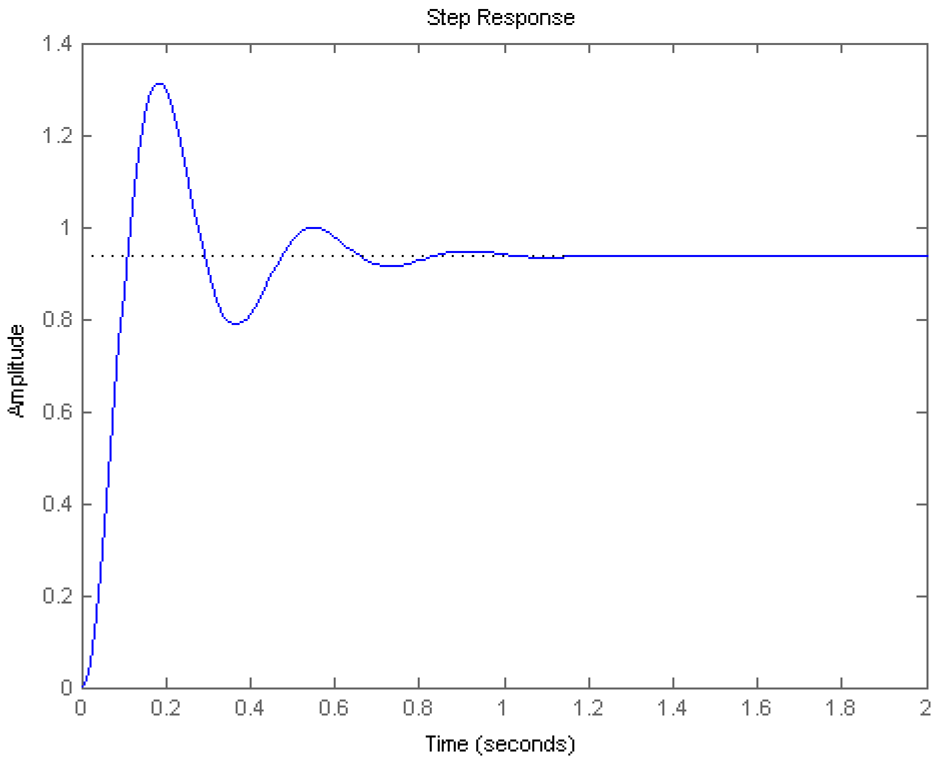
The closed-loop transfer function of the above system with a proportional controller is:

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

Let the proportional gain (K_p) equal 300 and change the m-file to the following:

```
Kp = 300;
C = pid(Kp)
T = feedback(C*P,1)

t = 0:0.01:2;
```

```
step(T,t)
C =

  Kp = 300

P-only controller.


T =

        300
  ----------------
  s^2 + 10 s + 320

Continuous-time transfer function.
```



The above plot shows that the proportional controller reduced both the rise time and the steady-state error, increased the overshoot, and decreased the settling time by small amount

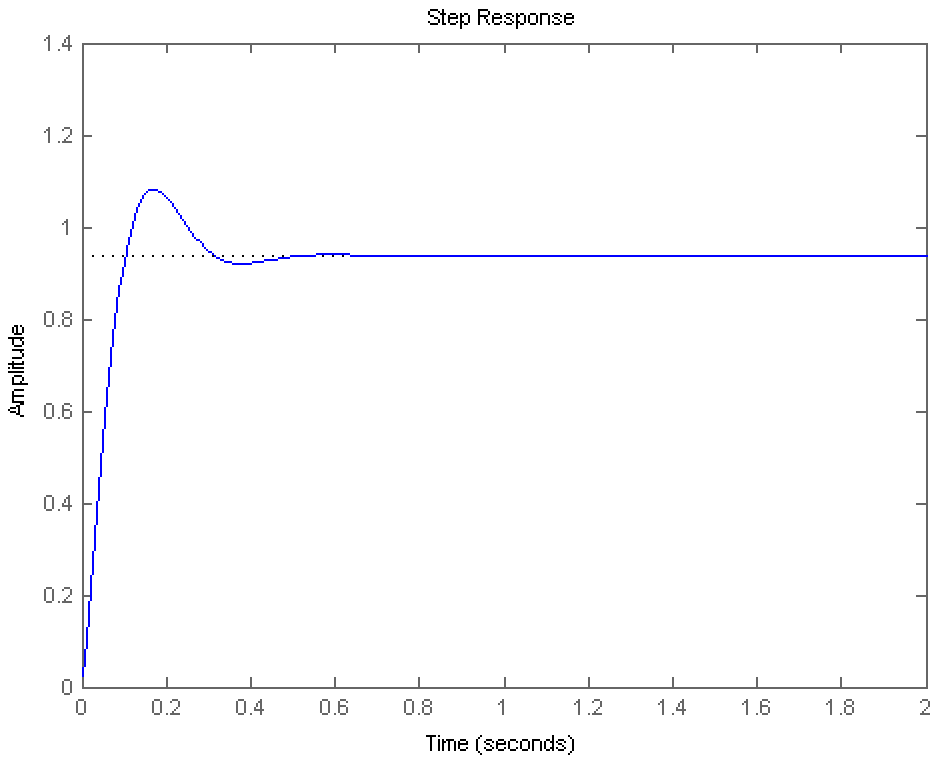
Proportional-Derivative Control

Now, let's take a look at a PD control. From the table shown above, we see that the derivative controller (Kd) reduces both the overshoot and the settling time. The closed-loop transfer function of the given system with a PD controller is:

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

Let K_p equal 300 as before and let K_d equal 10. Enter the following commands into an m-file and run it in the MATLAB command window.

```
Kp = 300;
Kd = 10;
C = pid(Kp,0,Kd)
T = feedback(C*P,1)

t = 0:0.01:2;
step(T,t)
C =


  Kp + Kd * s


  with Kp = 300, Kd = 10

Continuous-time PD controller in parallel form.


T =

     10 s + 300
  ----------------
  s^2 + 20 s + 320

Continuous-time transfer function.
```



This plot shows that the derivative controller reduced both the overshoot and the settling time, and had a small effect on the rise time and the steady-state error.

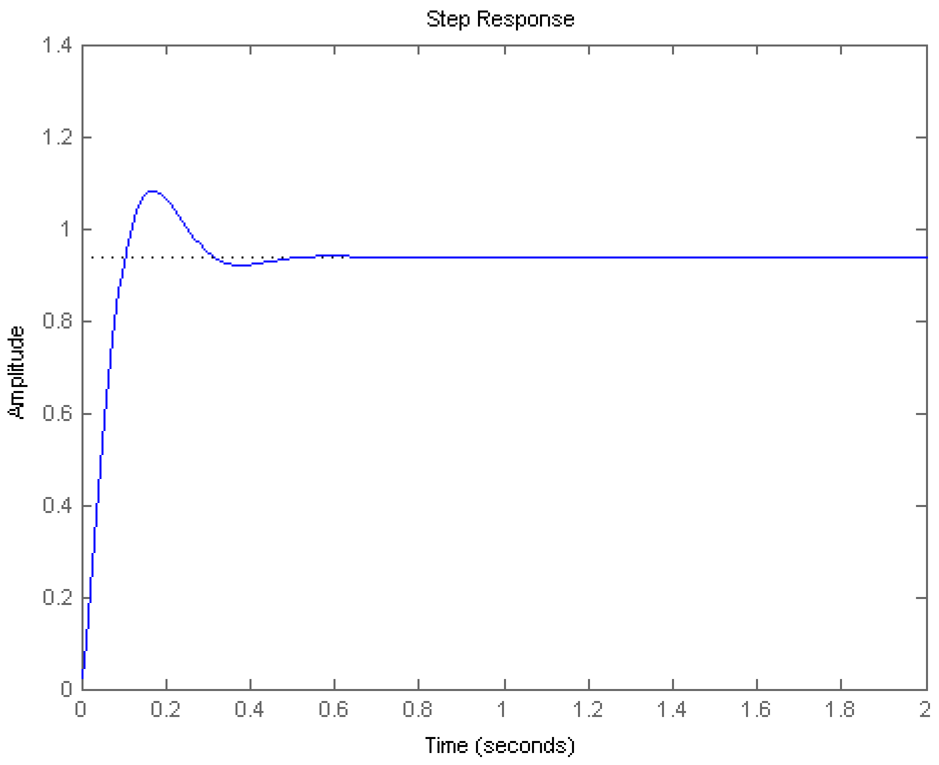
Proportional-Integral Control

Now, let's take a look at a PD control. From the table shown above, we see that the derivative controller (Kd) reduces both the overshoot and the settling time. The closed-loop transfer function of the given system with a PD controller is:

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

Let $K_p$equal 300 as before and let $K_d$equal 10. Enter the following commands into an m-file and run it in the MATLAB command window.

```
Kp = 300;
Kd = 10;
C = pid(Kp,0,Kd)
T = feedback(C*P,1)

t = 0:0.01:2;
step(T,t)
C =
  Kp + Kd * s


  with Kp = 300, Kd = 10

Continuous-time PD controller in parallel form.


T =

     10 s + 300
  ----------------
  s^2 + 20 s + 320

Continuous-time transfer function.
```



This plot shows that the derivative controller reduced both the overshoot and the settling time, and had a small effect on the rise time and the steady-state error

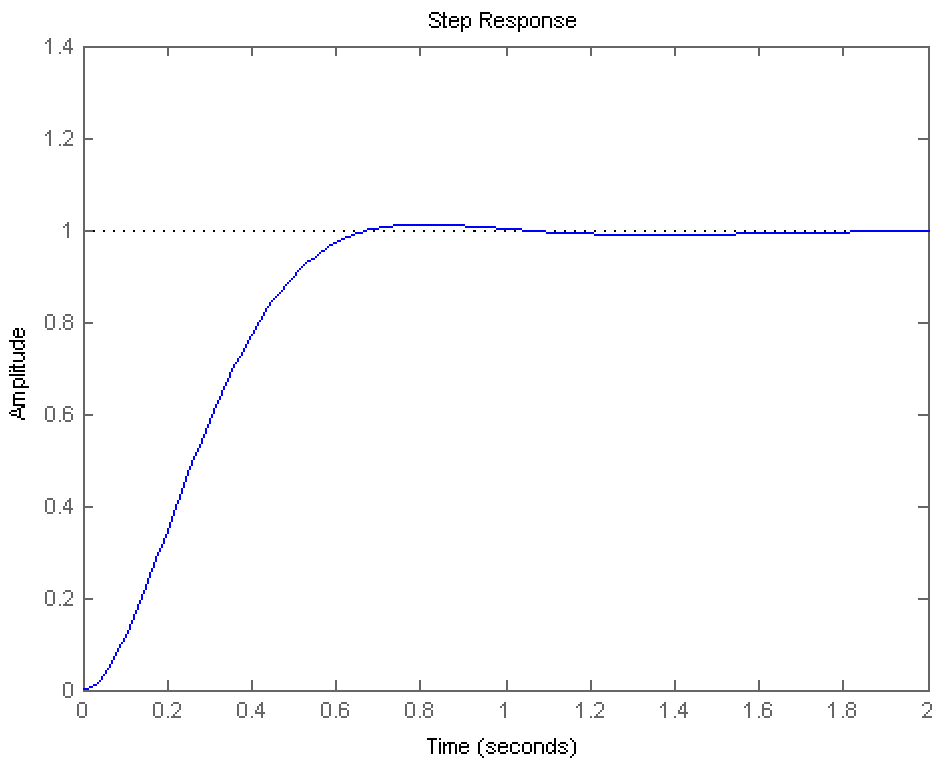
Proportional-Integral-Derivative Control

v Before going into a PID control, let's take a look at a PI control. From the table, we see that an integral controller (Ki) decreases the rise time, increases both the overshoot and the settling time, and eliminates the steady-state error. For the given system, the closed-loop transfer function with a PI control is:

(9) $\frac{X(s)}{F(s)} = \frac{K_p s + K_i}{s^3 + 10s^2 + (20 + K_p s + K_i)}$

Let's reduce the K_p to 30, and let K_i equal 70. Create an new m-file and enter the following commands.

```
Kp = 30;
Ki = 70;
C = pid(Kp,Ki)
T = feedback(C*P,1)
t = 0:0.01:2;
step(T,t)
C =

             1
  Kp + Ki * ---
             s

  with Kp = 30, Ki = 70

Continuous-time PI controller in parallel form.


T =

         30 s + 70
  ------------------------
  s^3 + 10 s^2 + 50 s + 70

Continuous-time transfer function.
```



Run this m-file in the MATLAB command window, and you should get the following plot. We have reduced the proportional gain (Kp) because the integral controller also reduces the rise time and increases the overshoot as the proportional controller does (double effect). The above response shows that the integral controller eliminated the steady-state error.

General Tips for Designing a PID Controller

When you are designing a PID controller for a given system, follow the steps shown below to obtain a desired response.

1. Obtain an open-loop response and determine what needs to be improved
2. Add a proportional control to improve the rise time
3. Add a derivative control to improve the overshoot
4. Add an integral control to eliminate the steady-state error
5. Adjust each of Kp, Ki, and Kd until you obtain a desired overall response. You can always refer to the table shown in this "PID Tutorial" page to find out which controller controls what characteristics.

Lastly, please keep in mind that you 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 you don't need to implement a derivative controller on the system. Keep the controller as simple as possible.

Automatic PID Tuning

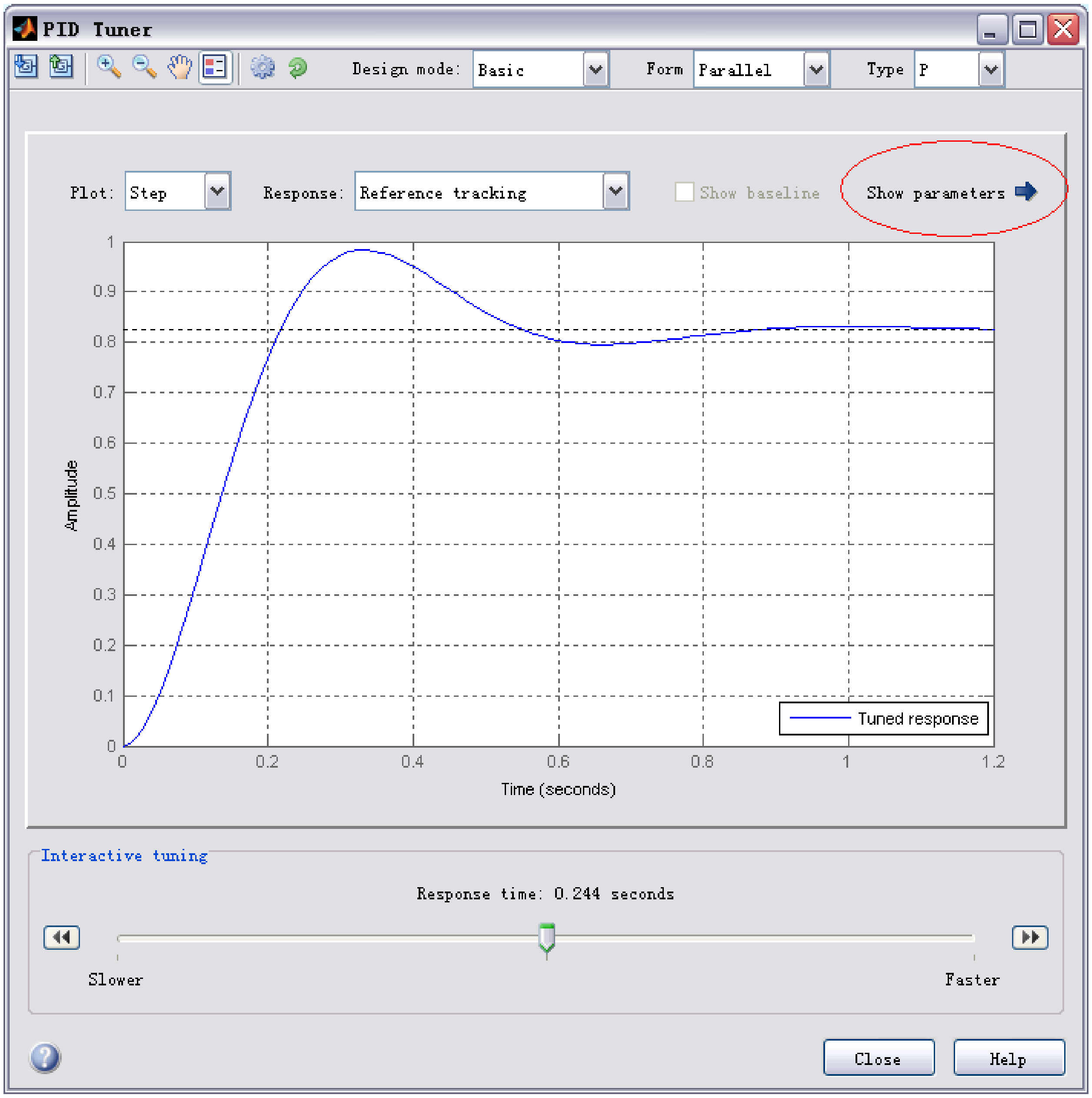
MATLAB provides tools for automatically choosing optimal PID gains which makes the trial and error process described above unnecessary. You can access the tuning algorithm directly using **pidtune** or through a nice graphical user interface (GUI) using **pidtool**.

The MATLAB automated tuning algorithm chooses PID gains to balance performance (response time, bandwidth) and robustness (stability margins). By default the algorthm designs for a 60 degree phase margin.

Let's explore these automated tools by first generating a proportional controller for the mass-spring-damper system by entering the following commands:

```
pidtool(P,'p')
```

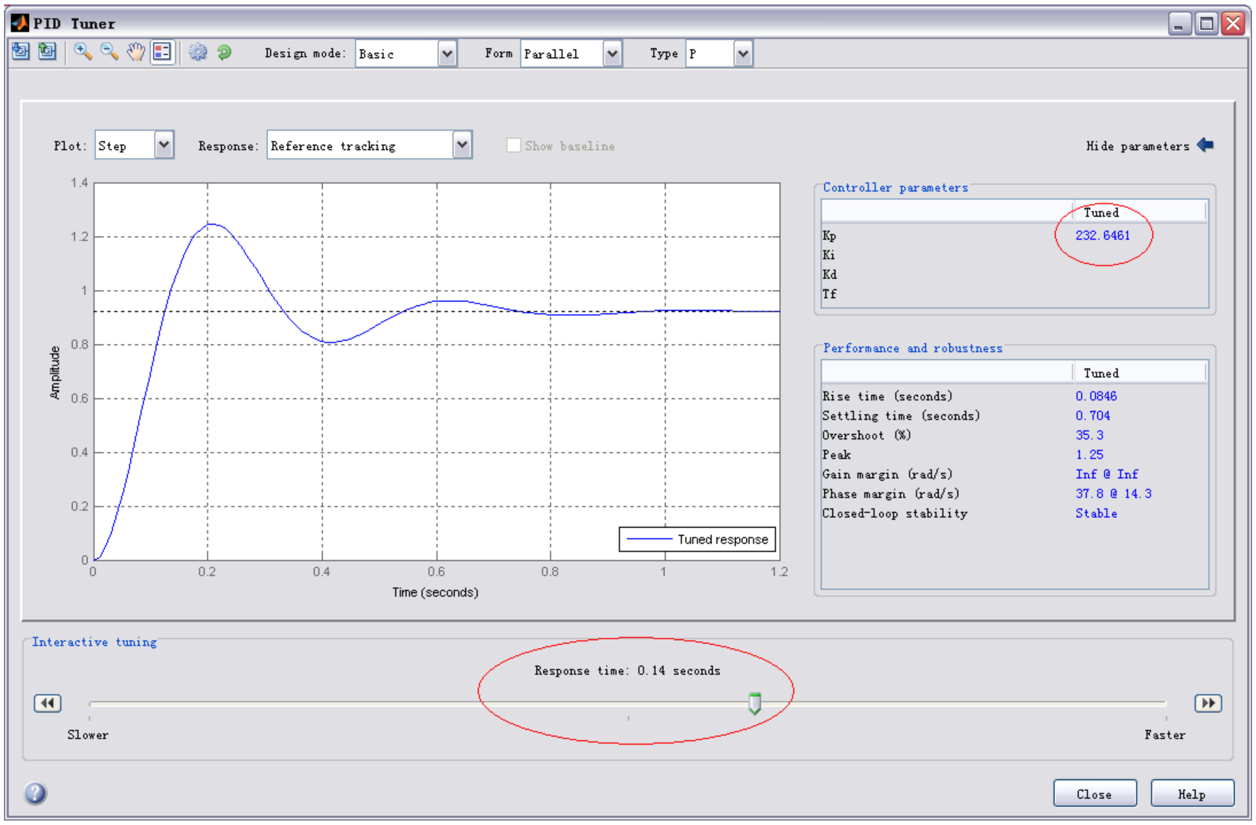
The pidtool GUI window, like that shown below, should appear.



Notice that the step response shown is slower than the proportional controller we designed by hand. Now click on the **Show Parameters** button on the top right. As expected the proportional gain constant, Kp, is lower than the one we used, Kp = 94.85 < 300.

We can now interactively tune the controller parameters and immediately see the resulting response int he GUI window. Try dragging the resposne time slider to the right to 0.14s, as shown in the figure below. The response does indeeed speed up, and we can see Kp is now closer

to the manual value. We can also see all the other performance and robustness parameters for the system. Note that the phase margin is 60 degrees, the default for pidtool and generally a good balance of robustness and performance.



Design mode: Basic Form Parallel Type P

Plot: Step Response: Reference tracking

Controller parameters

	Tuned
Kp	232.6461
Ki	
Kd	
Tf	

Performance and robustness

	Tuned
Rise time (seconds)	0.0846
Settling time (seconds)	0.704
Overshoot (%)	35.3
Peak	1.25
Gain margin (rad/s)	Inf @ Inf
Phase margin (rad/s)	37.8 @ 14.3
Closed-loop stability	Stable

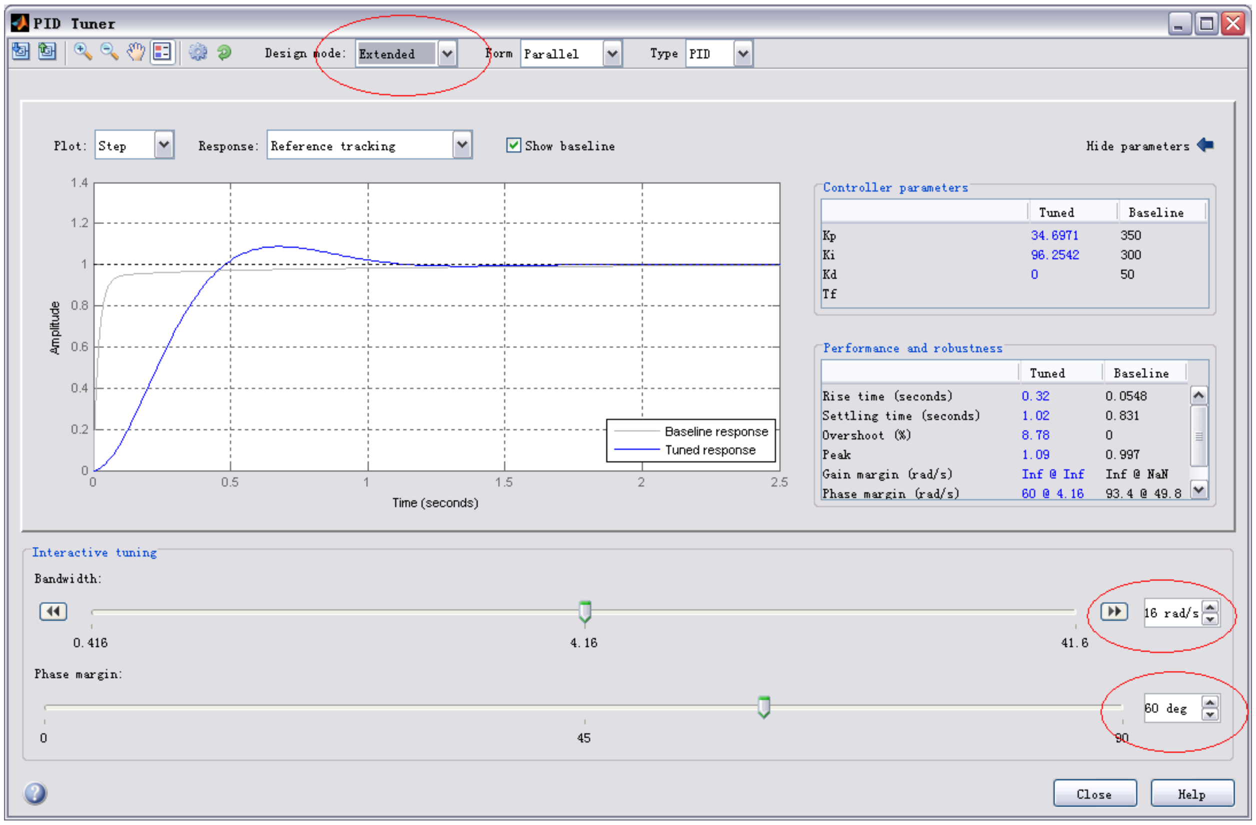
Response time: 0.14 seconds

(baseline) Now let's try designing a PID controller for our system. By specifying the previously designed or controller, C, as the second parameter, pidtool will design another PID controller

(instead of P or PI) and will compare the response of the system with the automated controller with that of the baseline.

```
        pidtool(P,C)
```

We see in the output window that the automated controller responds slower and exhibits more overshoot than the baseline. Now choose the **Design Mode: Extended** option at the top, which reveals more tuning parameters.



Controller parameters

	Tuned	Baseline
Kp	34.6971	350
Ki	96.2542	300
Kd	0	50
Tf		

Performance and robustness

	Tuned	Baseline
Rise time (seconds)	0.32	0.0548
Settling time (seconds)	1.02	0.831
Overshoot (%)	8.78	0
Peak	1.09	0.997
Gain margin (rad/s)	Inf @ Inf	Inf @ NaN
Phase margin (rad/s)	60 @ 4.16	93.4 @ 49.8

Now type in **Bandwidth: 32 rad/s** and **Phase Margin: 90 deg** to generate a controller similar in performance to the baseline. Keep in mind that a higher bandwidth (0 dB crossover of the open-loop) results in a faster rise time, and a higher phase margin reduces the overshoot and improves the system stability.

Finally we note that we can generate the same controller using the command line tool **pidtune** instead of the pidtool GUI

```
opts = pidtuneOptions('CrossoverFrequency',32,'PhaseMargin',90);
[C, info] = pidtune(P, 'pid', opts)
C =

             1
  Kp + Ki * --- + Kd * s
             s

  with Kp = 320, Ki = 169, Kd = 31.5

Continuous-time PID controller in parallel form.

info =
                Stable: 1
    CrossoverFrequency: 32
```

PhaseMargin: 90

(*g*) Environmental test-chamber temperature control system
(*h*) An automatic positioning system for a missile launcher
(*i*) An automatic speed control for a field-controlled dc motor
(*j*) The attitude control system of a typical space vehicle
(*k*) Automatic position-control system of a high speed automated train system
(*l*) Human heart using a pacemaker
(*m*) An elevator-position control system used in high-rise multilevel buildings.

1.3 CONTROL SYSTEM CONFIGURATIONS

There are two control system configurations: open-loop control system and closed-loop control system.

(*a*) ***Block.*** A block is a set of elements that can be grouped together, with overall characteristics described by an input/output relationship as shown in Fig. 1.3. A block diagram is a simplified pictorial representation of the cause-and-effect relationship between the input(s) and output(s) of a physical system.

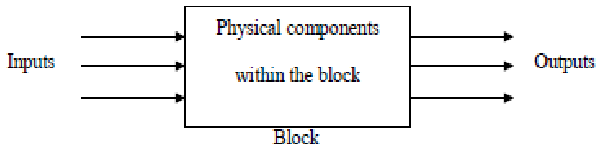


Fig. 1.3 Block diagram

The simplest form of the block diagram is the single block as shown in Fig. 1.3. The input and output characteristics of entire groups of elements within the block can be described by an appropriate mathematical expressions as shown in Fig. 1.4.

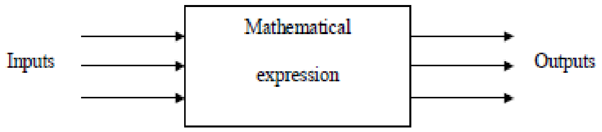


Fig. 1.4 Block representation

(*b*) ***Transfer Function.*** The transfer function is a property of the system elements only, and is not dependent on the excitation and initial conditions. The transfer function of a system (or a block) is defined as the ratio of output to input as shown in Fig.1.5.

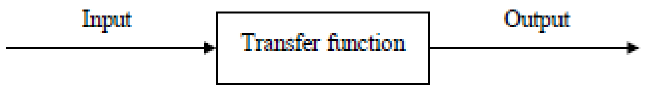


Fig. 1.5 Transfer function

$$\text{Transfer function} = \frac{\text{Output}}{\text{Input}}$$

(*c*) ***Open-loop Control System.*** Open-loop control systems represent the simplest form of controlling devices. A general block diagram of open-loop system is shown in Fig. 1.6.

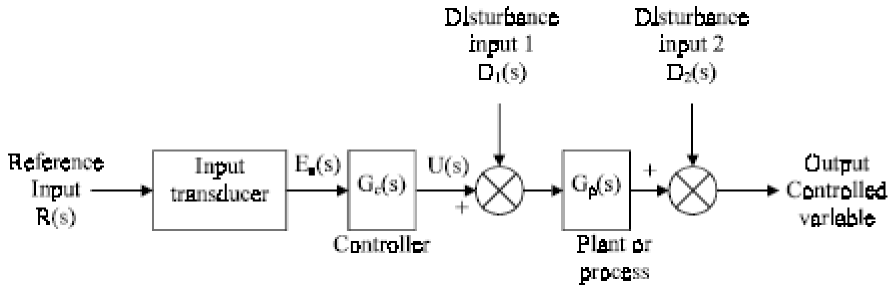


Fig. 1.6 General block diagram of open-loop control system

(*d*) ***Closed-loop (Feedback Control) System.*** Closed-loop control systems derive their valuable accurate reproduction of the input from feedback comparison. The general architecture of a closed-loop control system is shown in Fig. 1.7. A system with one or more feedback paths is called a **closed-loop system.**

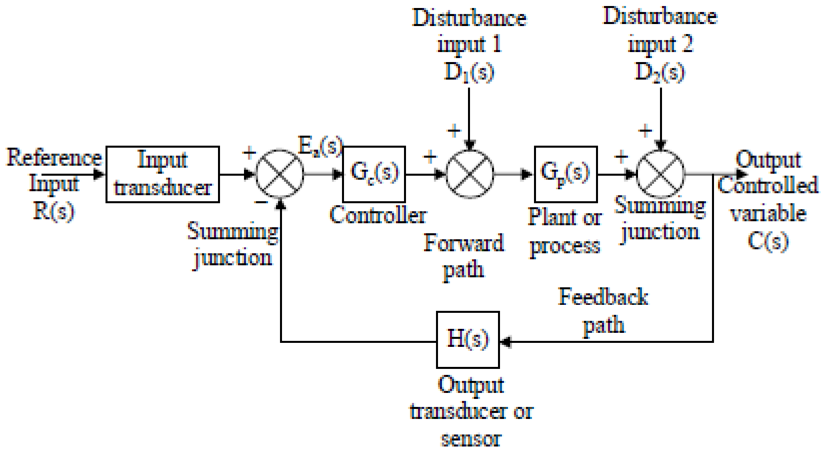


Fig. 1.7 General block diagram of closed-loop control system

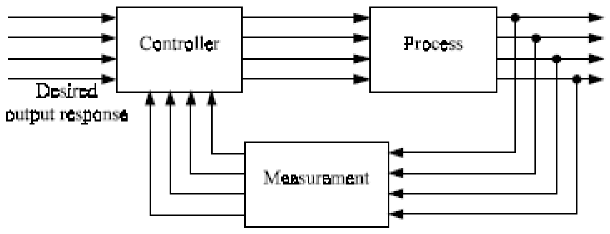


Fig. 1.12 Multivariable control system

1.6 FEEDBACK SYSTEMS

Feedback is the property of a closed-loop system, which allows the output to be compared with the input to the system such that the appropriate control action may be formed as some function of the input and output.

For more accurate and more adaptive control, a link or feedback must be provided from output to the input of an open-loop control system. So the controlled signal should be fed back and compared with the reference input, and an actuating signal proportional to the difference of input and output must be sent through the system to correct the error. In general, feedback is said to exist in a system when a closed sequence of cause-and-effect relations exists between system variables. A closed-loop idle-speed control system is shown in Fig. 1.13. The reference input N_r sets the desired idle-speed. The engine idle speed N should agree with the reference value N_r and any difference such as the load-torque T is sensed by the speed-transducer and the error detector. The controller will operate on the difference and provide a signal to adjust the throttle angle to correct the error.

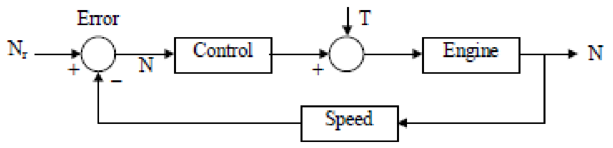


Fig. 1.13 Closed-loop idle-speed control system

CONTROL SYSTEM ANALYSIS AND DESIGN OBJECTIVES

Control systems engineering consists of *analysis* and *design* of control systems configurations. Control systems are *dynamic*, in that they respond to an input by first undergoing a transient response before attaining a steady-state response which corresponds to the input. There are three main objectives of control systems analysis and design. They are:

1. Producing the response to a transient disturbance which is acceptable

2. Minimizing the steady-state errors: Here, the concern is about the accuracy of the steady-state response

3. *Achieving stability:* Control systems must be designed to be stable. Their natural response should decay to a zero values as time approaches infinity, or oscillate.

System analysis means the investigation, under specified condition, of the performance of a system whose mathematical model is known. *Analysis* is investigation of the properties and performance of an existing control system.

By *synthesis* we mean using an explicit procedure to find a system that will perform in a specified way. *System design* refers to the process of finding a system that accomplishes a given task. *Design* is the selection and arrangement of the control system components to perform a prescribed task. The design of control systems is accomplished in two ways : *design by analysis* in which the characteristics of an existing or standard system configuration are modified, and *design by synthesis*, in which the form of the control system is obtained directly from its specifications.

INPUT/OUTPUT IN MATLAB

In this section, we present some of the many available commands in MATLAB for reading data from an external file into a MATLAB matrix, or writing the numbers computed in MATLAB into such an external file.

2.18.1 The fopen Statement

To have the MATLAB read or write a separate data file of numerical values, we need to *connect* the file to the executing MATLAB program. The MATLAB functions used are summarized in Table 2.32.

Table 2.32 MATLAB functions used for input/output

Function	Description
fopen	Connects an existing file to MATLAB or to create a new file from MATLAB. **fid = fopen**('*Filename*', *permission code*); where, if **fopen** is successful, **fid** will be returned as a positive integer greater than 2. When unsuccessful, a value of – 1 is returned. Both the file name and the permission code are string constants enclosed in single quotes. The permission code can be a variety of flags that specify whether or not the file can be written to, read from, appended to, or a combination of these. Some common codes are: **Code** **Meaning** **'r'** read only **'w'** write only **'r+'** read and write **'a+'** read and append The **fopen** statement positions the file at the beginning.
fclose	Disconnects a file from the operating MATLAB program. The use is **fclose(fid),** where **fid** is the *file identification number* of the file returned by **fopen.fclose('all')** will close all files.
fscanf	Reads opened files. The use is **A = fscanf(fid,** *FORMAT, SIZE*) where *FORMAT* specifies the types of numbers (integers, reals with or without exponent, character strings) and their arrangement in the data file, and optional *SIZE* determines how many quantities are to be read and how they are to be arranged into the matrix *A*. If *SIZE* is omitted, the entire file is read. The *FORMAT* field is a string (enclosed in single quotes) specifying the form of the numbers in the file. The *type* of each number is characterized by a percent sign (%), followed by a letter (***i*** or ***d*** for integers, ***e*** or ***f*** for floating-point numbers with or without exponents). Between the percent sign and the type code, one can insert an integer specifying the maximum width of the field.
fprintf	Writes files previously opened. **fprintf(fid,** *FORMAT*, **A)** where **fid** and *FORMAT* have the same meaning as for **fscanf,** with the exception that for output formats the string must end with **\n**, designating the end of a line of output.

(*Contd.*)

MATLAB can be used to obtain the partial-fraction expansion of the ratio of two polynomials, $B(s)/A(s)$ as follows:

$$\frac{B(s)}{A(s)} = \frac{\text{num}}{\text{den}} = \frac{b(1)s^n + b(2)s^{n-1} + \ldots + b(n)}{a(1)s^n + a(2)s^{n-1} + \ldots + a(n)}$$

where $a(1) \neq 0$ and num and den are row vectors. The coefficients of the numerator and denominator of $B(s)/A(s)$ are specified by the num and den vectors.

Hence num = $[b(1) \quad b(2) \; \ldots \quad b(n)]$

den = $[a(1) \quad a(2) \; \ldots \quad a(n)]$

The MATLAB command

***r*, *p*, *k* = residue(num, den)**

is used to determine the residues, poles, and direct terms of a partial-fraction expansion of the ratio of two polynomials $B(s)$ and $A(s)$ is then given by

$$\frac{B(s)}{A(s)} = k(s) + \frac{r(1)}{s - p(1)} + \frac{r(2)}{s - p(2)} + \ldots + \frac{r(n)}{s - p(n)}$$

The MATLAB command **[num, den] = residue(*r*, *p*, *k*)** where r, p, k are the output from MATLAB converts the partial fraction expansion back to the polynomial ratio $B(s)/A(s)$.

The command **printsys (num,den, 's')** prints the num/den in terms of the ratio of polynomials in s.

The command **ilaplace** will find the inverse Laplace transform of a Laplace function.

Finding Zeros and Poles of $B(s)/A(s)$

The MATLAB command **[z, p, k] = tf2zp(num,den)** is used to find the zeros, poles, gain K of $B(s)/A(s)$.

If the zeros, poles, and gain K are given, the following MATLAB command can be us find the original num/den:

[num, den] = zp2tf (z, p, k)

Example 2.7. *Consider the function*

$$H(s) = \frac{n(s)}{d(s)}$$

where $n(s) = s^4 + 6s^3 + 5s^2 + 4s + 3$

$d(s) = s^5 + 7s^4 + 6s^3 + 5s^2 + 4s + 7$

(*a*) *Find* $n(-10)$, $n(-5)$, $n(-3)$ *and* $n(-1)$

(*b*) *Find* $d(-10)$, $d(-5)$, $d(-3)$ *and* $d(-1)$

(*c*) *Find* $H(-10)$, $H(-5)$, $H(-3)$ *and* $H(-1)$

Solution:

```
(a)   >> n = [1 6 5 4 3];        % n = s ^ 4 + 6s ^ 3 + 5s ^ 2 + 4s + 3
      >> d = [1 7 6 5 4 7];      % d = s ^ 5 + 7s ^ 4 + 6s ^ 3 + 5s ^ 2 + 4s + 7
   >> n2 = polyval(n, [– 10])
       n2 = 4463
>> nn10 = polyval(n, [– 10])
     nn10 = 4463
  >> nn5 = polyval(n, [– 5])
      nn5 = – 17
 >> nn3 = polyval(n, [– 3])
     nn3 = – 45
>> nn1 = polyval(n, [– 1])
    nn1 = – 1
(b) >> dn10 = polyval(d, [– 10])
   dn10 = – 35533
>> dn5 = polyval(d, [– 5])
     dn5 = 612
>> dn3 = polyval(d, [– 3])
    dn3 = 202
>> dn1=polyval(d, [– 1])
    dn1 = 8
(c) >> Hn10 = nn10/dn10
         Hn10 = – 0.1256
      >> Hn5 = nn5/dn5
          Hn5 = – 0.0278
```

```
>> Hn3 = nn3/dn3
   Hn3 = – 0.2228
>> Hn1 = nn1/dn1
   Hn1 = – 0.1250
```

Example 2.8. *Generate a plot of*

$$y(x) = e^{-0.7x} \sin \omega x$$

where w = 15 rad/s, and $0 \le x \le 15$. Use the colon notation to generate the x vector in increments of 0.1.

Solution.

```
>> x = [0 : 0.1 : 15];
>> w = 15;
>> y = exp(– 0.7*x).*sin(w*x);
>> plot(x, y)
>> title('y(x) = e^-^0^.^7^x sin\omega x')
>> xlabel('x')
>> ylabel('y')
```

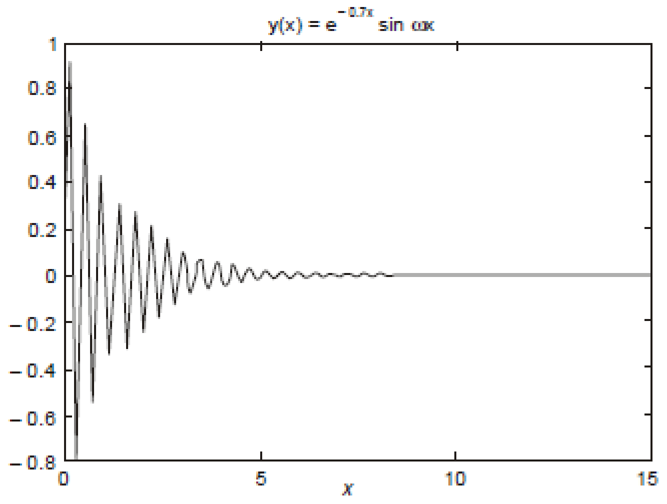


Fig. E 2.8.

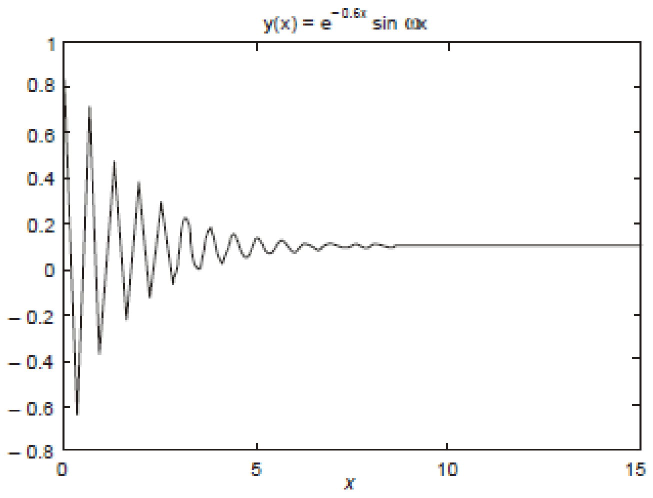
Example 2.9. *Generate a plot of*

$$y(x) = e^{-0.6x} \cos \omega x$$

where ω = 10 rad/s, and $0 \le x \le 15$. Use the colon notation to generate the x vector in increments of 0.05.

Solution.

```
>> x = [0 : 0.1 : 15];
>> w = 10;
>> y = exp(– 0.6*x).*cos(w*x);
>> plot(x, y)
>> title('y(x) = e^-^0^.^6^x cos\omega x')
>> xlabel('x')
>> ylabel('y')
```



Obtain the plot of the points for $0 \le t \le 6\pi$ when the coordinates x,y,z are given as a function of the parameter t as follows:

$$x = \sqrt{t}\sin(3t)$$
$$y = \sqrt{t}\cos(3t)$$
$$z = 0.8t$$

Solution.

```
% Line plots
>> t = [0:0.1:6*pi];
>> x = sqrt(t).*sin(3*t);
>> y = sqrt(t).*cos(3*t);
>> z = 0.8*t;
>> plot3(x, y, z, 'k', 'linewidth', 1)
>> grid on
>> xlabel ('x'); ylabel ('y') ; zlabel ('z')
```

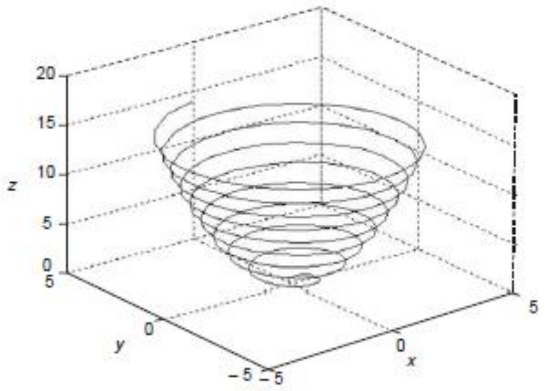


Fig. E 2.12.

Example 2.13. *Obtain the mesh and surface plots for the function* $z = \dfrac{2xy^2}{x^2 + y^2}$ *over the domain* $-2 \le x \le 6$ *and* $2 \le y \le 8$.

Solution.

```
% Mesh and surface plots
   x = – 2 : 0.1 : 6;
>> y = 2 : 0.1 : 8;
>> [x, y] = meshgrid(x, y);
>> z = 2*x.*y.^2./(x.^2 + y.^2);
>> mesh(x, y, z)
>> xlabel('x'); ylabel('y'); zlabel('z')
>> surf(x, y, z)
>> xlabel('x'); ylabel('y'); zlabel('z')
```

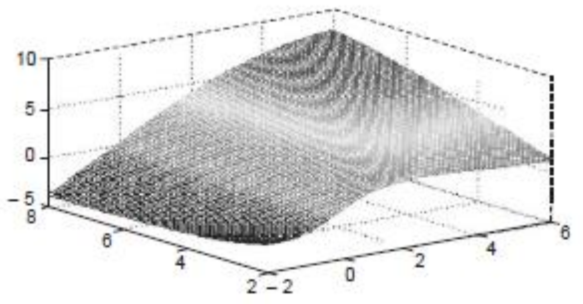


Fig. E 2.13(*a*)

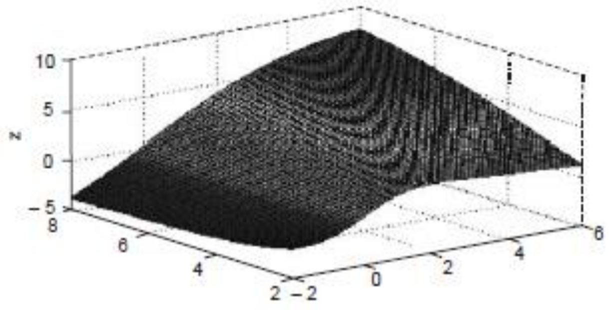


Fig. E 2.13(*b*)

ROOT LOCUS PLOTS

Locus is defined as a set of all points satisfying a set of conditions. The term *root* refers to the roots of the characteristic equation, which are the poles of the closed-loop transfer function. These poles define the time response of the system and hence the performance and stability of the system. Hence, *root-locus* defines a graph of the poles of the closed-loop transfer function as the system parameter, such as the gain is varied.

Evan's root locus method, or simply root-locus method, gives all closed-loop poles graphically, using the knowledge provided by the open-loop poles and open-loop zeros. A root-locus plot is composed of as many individual loci as there are poles. Individual loci are referred to as *branches* of the root locus.

The poles of a transfer function can be shown graphically in the s-plane by means of a pole-zero map. The root locus method is an analytical method for displaying the location of the poles of the closed-loop transfer function

$$\frac{G}{1+GH}$$

as a function of the gain factor K of the open-loop transfer function GH. The method is called the *root locus analysis*. The root locus analysis has the advantage that this method requires only the location of the poles and zeros of GH known and the factorization of the characteristic polynomial is not required. The method gives accurate time-domain response as well as frequency response information.

The root-locus method is based on the fact that the values of s that make the transfer function around the loop equal – 1 must satisfy the characteristic equation of the system. The locus of roots of the characteristic equation of the closed-loop system as the gain is varied from zero to infinity gives the root-loci plot. The root locus plot indicates the contributions of each open-loop pole or zero to the locations of the closed-loop poles.

The root locus method allows the prediction of the effects on the location of the closed-loop poles of varying the gain value or adding open-loop poles and/or open-loop zeros. Fig. 3.1 shows a canonical feedback control system whose closed-loop transfer function is given by

$$\frac{C}{R}=\frac{G}{1+GH} \tag{3.17}$$

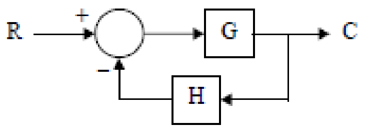


Fig. 3.1.

If the open-loop transfer function GH is represented by

$$GH=\frac{KN}{D} \tag{3.18}$$

Where D and N are finite polynomials in the complex variable s, and K is the open-loop gain factor, then the closed-loop transfer function can be written as

$$\frac{C}{R}=\frac{G}{1+KN/D}=\frac{GD}{D+DN} \tag{3.19}$$

The roots of the characteristic equation gives the closed-loop poles. That is

$$D+KN=0 \tag{3.20}$$

As the open-loop gain factor K is varied, the location of these roots in the s-plane changes. A locus of these roots plotted in the s-plane as a function of K is known as a *root locus*. We observe from Eq. (3.4) that when $K=0$, the roots of the polynomial D gives the poles of the open-loop transfer function GH. On the other hand, as K becomes very large, then the roots will become those of the polynomial N (the open-loop zeros). Hence, the loci of the closed-loop poles originate from the open-loop poles and terminate at the open-loop zeros ad K varies from zero to infinity.

Consider the system equation

$$1 + \frac{K(s + z_1)(s + z_2) \dots (s + z_n)}{(s + p_1)(s + p_2) \dots (s + p_n)} = 0 \tag{3.21}$$

Equation (3.17) can be written as

$$1 + K\frac{\text{num}}{\text{den}} = 0 \tag{3.22}$$

Where *num* is the numerator of the polynomial and *den* is the denominator polynomial, and K is the gain ($K > 0$). The vector K contains all the gain values for which the closed loop poles are to be computed.

The root loci is plotted by using the MATLAB command

rlocus (num, den)

The gain vector K is supplied by the user.

The matrix r and gain vector K are obtained by the following MATLAB commands:

$[r, k]$ = $r$locus (num, den)

$[r, k]$ = $r$locus (num, den, k)

$[r, k]$ = $r$locus (A, B, C, D)

$[r, k]$ = $r$locus (A, B, C, D, K) (3.23)

$[r, k]$ = $r$locus (sys)

In Eqns. (3.23), r has length K rows and length [den – 1] columns containing the complex root locations.

For plotting the root loci, the MATLAB command plot (r, ‘ ’) is used.

The following MATLAB command are used for plotting the root loci

with mark ‘0’ or ‘x’:

r = rlocus (num, den)

plot (r, ‘0’) or plot (r, ‘x’)

MATLAB provides its own set of gain values used to compute a root locus plot. It also uses the automatic axis scaling features of the plot command.

BODE DIAGRAMS

Polar plot is a plot of the magnitude $|G(j\omega)H(j\omega)|$ and phase angle $|G(j\omega)H(j\omega)|$ in polar coordinates for various values of frequencies ranging from 0 to ∞ then $-\infty$ to 0.

Bode plot is a plot of magnitude $|G(j\omega)H(j\omega)|$ in decibels versus $\log\omega$ and phase angle $|G(j\omega)H(j\omega)|$ versus $\log \omega$ in rectangular coordinates.

Magnitude versus phase angle plot or gain-phase plot is a plot of magnitude $|G(j\omega)H(j\omega)|$ in decibels versus phase angle $|G(j\omega)H(j\omega)|$ in rectangular coordinates with frequency as varying parameter.

In practice the frequency-functions of the system are so complex and long that the characteristic of the system cannot be determined at the desired frequency just only by inspection of the system frequency function. Hence the frequency functions of systems are plotted in graphical forms, which indicate the system characteristics. Any curve giving information regarding the gain or phase shift of the frequency function is known as the *frequency response of the* *system*. In polar-plots the amplitude of $G(j\omega)$ is plotted as the distance from origin while the phase-angle is plotted as angular displacement from right hand horizontal axis on the polar graph. These plots are simple to construct and easily provide the information regarding the magnitude and phase-angle of $G(j\omega)$ at any desired frequency as compared to rectangular-plots because polar-plot contains the ready information of both the parameters, amplitude and phase angle.

A transfer function $G(s)$ may be represented in the frequency domain as a sinusoidal transfer function by substituting $j\omega$ for s in the expression for $G(s)$. The resulting form $G(j\omega)$ is a complex function of the single variable ω. Thus it can be plotted in 2-dimensions with ω as a parameter and written in the following equivalent form:

Polar form: $G(j\omega) = |G(j\omega)| \; |\phi(\omega)|$

Euler form: $G(j\omega) = |G(j\omega)| \; (\cos \phi(\omega) + j \sin \phi(\omega))$

Here $|G(j\omega)|$ is the magnitude of complex function and $\phi(\omega)$ is the phase angle = $\arg p(\omega)$.

Bode diagrams are rectangular plots. Bode diagram are also known as logarithmic plot and consist of two graphs: the first one is a plot of the logarithmic of the magnitude of a sinusoidal transfer function, the second one is a plot of the phase angle. Both these graphs are plotted against the frequency on a logarithmic scale. Bode analysis is similar to Nyquist analysis in that here also the graphical representation of the open-loop frequency response function $G(\omega)H(\omega)$, is employed. Bode-plot consists of two graphs: the magnitude of $G(\omega)H(\omega)$, and the phase angle of $G(j\omega)H(j\omega)$, both plotted as a function of frequency ω. Logarithmic scales are used for the frequency axes and for $|G(j\omega)H(j\omega)|$. Bode plots illustrate the relative stability of a system.

The following frequency-domain specifications are used:

If the Nyquist-plot does not cross the critical point $(-1 + j0)$, the system is found to be stable. The frequency (ω_1) at which the magnitude $|G(j\omega)H(j\omega)|$ equals to one is called the gain-cross over frequency. If the plot at ω_1 is rotated through an angle ϕ in clockwise direction, the point $\omega = \omega_1$ and critical point $(-1 + j_0)$ are coincident and this indicates that the closed-loop system is *marginally stable*. Any further rotation leads to instability. The angle ϕ is known as *phase-margin*. Intersection of the plot with negative real axis corresponds to frequency $\omega = \omega_2$. The phase-angle at $\omega = \omega_2$ is

$$|G(j\omega_2)H(j\omega_2)| = -180^{\circ} \tag{3.24}$$

Hence, the closeness of Nyquist plot to critical point $(-1 + j_0)$, decides the relative stability of the systems. The closer the Nyquist plot to the critical point $(-1 + j_0)$ the system tends towards instability.

Gain margin is a factor by which the gain of a stable system is allowed to increase before the system reaches instability. Gain margin is defined as the magnitude of reciprocal of the open-loop transfer function evaluated at the frequency ω_2 at which the phase angle is -180°.

$$GM = \frac{1}{|G(j\omega_2)H(j\omega_2)|} \tag{3.25}$$

where ω_2 is the phase cross over frequency.

Phase margin of a stable system is the amount of additional phase lag required to bring the system to point of instability. It is defined as

$$PM = [180 + |G(j\omega_1)H(j\omega_1)] \tag{3.26}$$

where $|G(j\omega_1)H(j\omega_1)| = 1$ and ω_1 is called the *gain-crossover frequency*.

For a stable system both GM and PM should be positive.

Bode plot consists of two graphs in rectangular coordinates. These are (*i*) the magnitude of $G(j\omega)H(j\omega)$ in *db* versus $\log_{10}\omega$, (*ii*) the phase angle of $G(j\omega)H(j\omega)$ versus $\log_{10}\omega$.

The general open-loop transfer function of a feedback control system may be represented by

$$G(s)H(s) = \frac{k[(1+ST_1)(1+ST_2)\ldots]\,\omega_n^2}{s^N[(1+ST_a)(1+sT_b)\ldots](s^2+2\xi\omega_n s+\omega_n^2)} \tag{3.27}$$

Here the highest power of *s* in the numerator is lower than that of the denominator. Now substituting $s = j\omega$, we get

$$G(j\omega)H(j\omega) = \frac{k[(1+j\omega T_1)(1+j\omega T_2)\ldots]\,\omega_n^2}{(j\omega)^N[(1+j\omega T_a)(1+j\omega T_b)\ldots](\omega_n^2-\omega^2\ 2\xi j\omega_n\omega)} \tag{3.28}$$

Hence, the magnitude is

$$|G(j\omega)H(j\omega)| = \frac{k\,|1+j\omega T_1\,||1+j\omega T_2\,|\ldots\omega_n^2}{|(j\omega)^N\,||1+j\omega T_a\,|\ldots|\,\omega_n^2-\omega^2+2\xi j\omega_n\omega\,|} \tag{3.29}$$

and the phase is

$$|G(j\omega)H(j\omega)| = \left[\tan^{-1}\omega T_1+\tan^{-1}\omega T_2-90N-\tan^{-1}\omega T_a\ \ldots\ \tan^{-1}\frac{2\xi\omega_n\omega}{(\omega_n^2-\omega^2)}\right] \tag{3.30}$$

The magnitude can be represented in decibel form as

$$\begin{aligned}20\log_{10}|G(j\omega)H(j\omega)| &= 20\log_{10}k+20\log_{10}|1+j\omega T_1|\\ &\quad+20\log_{10}|1+j\omega T_2|-20\,N\log_{10}|j\omega|-20\log_{10}|1\\ &\quad+j\omega T_a\,|\ \ldots\ldots\ 20\log\left|\frac{\omega_n^2-\omega^2}{\omega_n^2}+\frac{2\xi\omega_n\omega}{\omega_n^2}\right|\end{aligned} \tag{3.31}$$

Equation (3.31) shows that the frequency function of an open-loop transfer function $G(j\omega)H(j\omega)$ has factors as follows:

(*a*) Constant gain factor *k*

(*b*) Poles at origin due to factor $\dfrac{1}{(j\omega)^N}$ with $N = 0, 1, 2, \ldots$

(*c*) Zeros on real axis due to $(1 + j\omega T)$

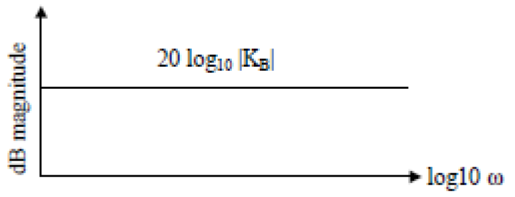
(*d*) Poles on real axis due to $\dfrac{1}{(1+j\omega T)}$ (3.32)

(*e*) Complex conjugate poles due to $\dfrac{\omega_n}{(\omega_n^2-\omega^2)+j2\xi\omega_n\omega)}$

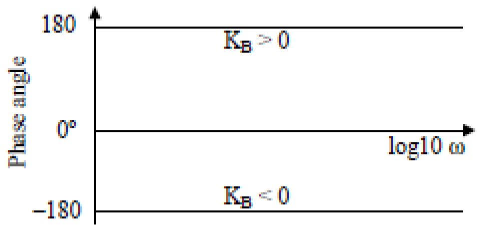
Bode plots of continuous-time frequency response functions can be constructed by summing the magnitude and phase angle contributions of each pole and zero. The asymptotic approximations of these plots are often sufficient. The asymptotic Bode plots for $G(j\omega)H(j\omega)$ are obtained by adding the graphs of each of the terms (a) to (e). For example, the Bode plot for gain term k is obtained from the expressions

$$k\,|_{db} = 20 \log_{10} k$$
$$\underline{|k} = 0 \qquad (3.33)$$

Thus, the magnitude and phase angle for the term K is independent of the frequency. Hence, the Bode plot for this term is a horizontal straight line, as shown in Fig. 3.2.



(a) Gain plot



(b) Phase plot

Fig. 3.2

From the frequency response-function for a pole of order N at origin or $\frac{1}{(j\omega)^N}$, the Bode plots are inclined straight lines. Here the magnitude in decibels (dB) = $20 \log_{10} \left| \frac{1}{(j\omega)^N} \right| = -20$ N $\log_{10}\omega$ and the phase-angle $\left| \frac{1}{(j\omega)^N} \right| = -90$ N. The Bode plots are shown in Fig. 3.3.