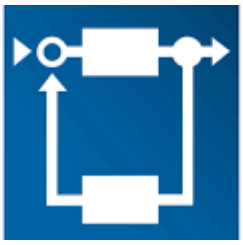
Department of Electrical Engineering
College of Engineering
University of Hail

Laboratory Manual
EE 380 – Control Engineering



PC-MATLAB PRIMER

This is intended as a guided tour through PCMATLAB. Type as you go and watch what happens.

```
>> 2*3
ans = 6
```
PCMATLAB uses several lines for the answer, but I've edited this to save space.

```
>> a=2*3
a = 6
```
Now I've saved the result in the variable "a"

```
>> a=2*3;
```
The semicolon ";" saves the result but without showing the answer.

```
>> j = sqrt (-1);
```
PCMATLAB doesn't have j stored as a variable. If you'll need it, define it this way. Now you can do complex multiplication:

```
>> (1+j) * (2 + j)
ans =   1.0000 + 3.0000i
```
and even mix multiplication and addition:

```
>> (1+j) * (2+j) + (3+j)
ans = 4.0000 + 4.0000i
```

Let's do some work with vectors. First define a vector going from 0 to 2pi (pi is stored in PCMATLAB in steps of 0.01 sec:

```
>> t = [0: 0.01: 2*pi];
```

Now let's compute cos (t) and plot t vs. cos (t):

```
>> c = cos (t);
>> plot (t, c)
```
Hit ENTER key to stop viewing plot. Type "shg" (show graphics to return. Try "grid" at the >> prompt to get an x-y grid.

What happens if you just say plot (c) like this:

```
>> plot (c)
```
The x-axis is i and y-axis is c_i.

Now let's define two vectors a and b and try several kinds of multiplication:

```
>> a = [1, 2]; b= [3, 4];
>> a.*b
ans =     3     8
```
a.*b is a vector c with $c_i = a_i\, b_i$

a*b is matrix multiplication but requires that dimensions match.

```
>> a*b
??? Error using ==> *
Inner matrix dimensions must agree.
```

So try using the transpose operator (')

```
>> a*b'
ans =    11
```

Of course! A is a row vector and b' is a column vector, so a*b' is the dot product of a and b. Same as sum (a.*b).

If you've had linear algebra, you know that a'*b (a column vector times a row vector) should produce a 2x2 matrix:

```
>> a'*b
ans =      3     4
           6     8
```

It does!

Now let's define a 2x2 matrix (; separates rows):

```
>> a= [1 2; 3 4]

a =   1     2
      3     4
```

And take its inverse:

```
>> inv (a)
ans =   -2.0000    1.0000
         1.5000   -0.5000
```

Couldn't be simpler!

And (for linear algebra devotees) find its eigenvectors and eigenvalues:

```
>> [u, v] = eig (a)
u =     -0.8246   -0.4160
         0.5658   -0.9094

 v =    -0.3723         0
              0    5.3723
```

These are the eigenvectors

The diagonal entries are the eigenvalues

```
>> eig (a)
ans = -0.3723
       5.3723
```

returns just the eigenvalues:

To list all the variables in the current workspace, together with information about their size, bytes, class, type "whos":

```
>> whos
```

Name	Size	Bytes	Class
a	2x2	32	double array
ans	2x1	16	double array
b	1x2	16	double array
c	1x629	5032	double array
j	1x1	16	double array (complex)
t	1x629	5032	double array
u	2x2	32	double array
v	2x2	32	double array

Grand total is 1275 elements using 10208 bytes

Now, let's get rid of all these variables and check:

```
>> clear
>> whos
>>
```

Let's get the t vector back and see what happens if we do:

```
>> t = [0: 0.01: 2*pi]
>> plot (sin (t), cos(t))
>> plot (sin (3*t), cos (5*t))
```

To get the axes scaled as you want, type

```
axis([-1.5, 1.5, -1, 1)
```
sets scaling for the x- and y-axes on the current plot.

```
>> axis auto
```
return to autoscaling

Now let's generate a half-wave rectified sine wave and find its DC component (average value):

```
>> w0=120*pi;                (60 Hz)
>> t =    [0: 0.0001: 1/60]; delta t chosen for 100-200 points per period.
>> whos                      check:
```

Name	Size	Total	Complex
t	1 by 167	167	No
w0	1 by 1	1	No

```
>> f = sin (t);
>> f = [f (1:84), zeros (1, 83)];     Kills negative half:
>> plot (t, f)                        Plotted as a check. Uh-oh, I forgot w0! Easy to fix:

>> f = sin (w0*t);
>> f = [f (1:84), zeros (1, 83)];
>> plot (t, f)                        Now it checks. (Note the value of checking).
```

Here's how to simulate 1/T∫ (f.dt)

```
>> T = 1/60;                 T and t are different variables.
>> dt=0.0001;
>> a0=sum (f.*dt)/T

>> a0 =0.3183
```

PCMATLAB remembers your last input line and you can edit it using the up arrow key, INS (insert), DEL (delete), and typing over. This really speeds up corrections and repetitive calculations.

```
>> quit                Leaves MATLAB
```

Try typing the following at the >> prompt:

Help.	help roots	help	help poly
demo	help Isim	help nyquist	help bode
help impulse	help step		

Suggested exercise: Generate a full-wave rectified sine wave using f = abs (sin (w_0t)) with t as above. Plot it and check that it looks right. Then calculate the DC component and see if it is twice that found above.

For most problems you'll find it a lot easier to write .m files, programs executable by PCMATLAB. Look at the sample program below to see the basic syntax. Refer to the PCMATLAB manual for details. This program calculates finite approximations to

V(t) = 4/pi [cos (w0*t) – cos(3*w0*t) /3 + cos (5*w0*t) /5 …]

Try running it and seeing what it looks like for n=1, 3, 11, 21, 101. You call the program by saying "[t, v] = sq(11).

```
function [t, v] = sq(n)
% Fourier series to n-th harmonic
titl='Fourier approximation to square wave to';
titl=[titl, num2str(n)];
titl=[titl, '-th harmonic'];
%
%
t  =[-.25 : 0.005 : 1.25] ;
w0=2*pi ;
v  =cos(w0*t) ;
factor = 1 ;
for i= 3 : 2 :n
       factor = factor*(-1) ;
       v = v + factor*(1/i) *cos(i*w0*t) ;
       end
v = (4/pi) *v ;
clg
% clg Clears (erases) Graphics memory. Shg Shows Graphics.
plot (t, v)
grid
title (titl)
shg
pause
```

To help you in understanding, note the following:

% denotes a comment line

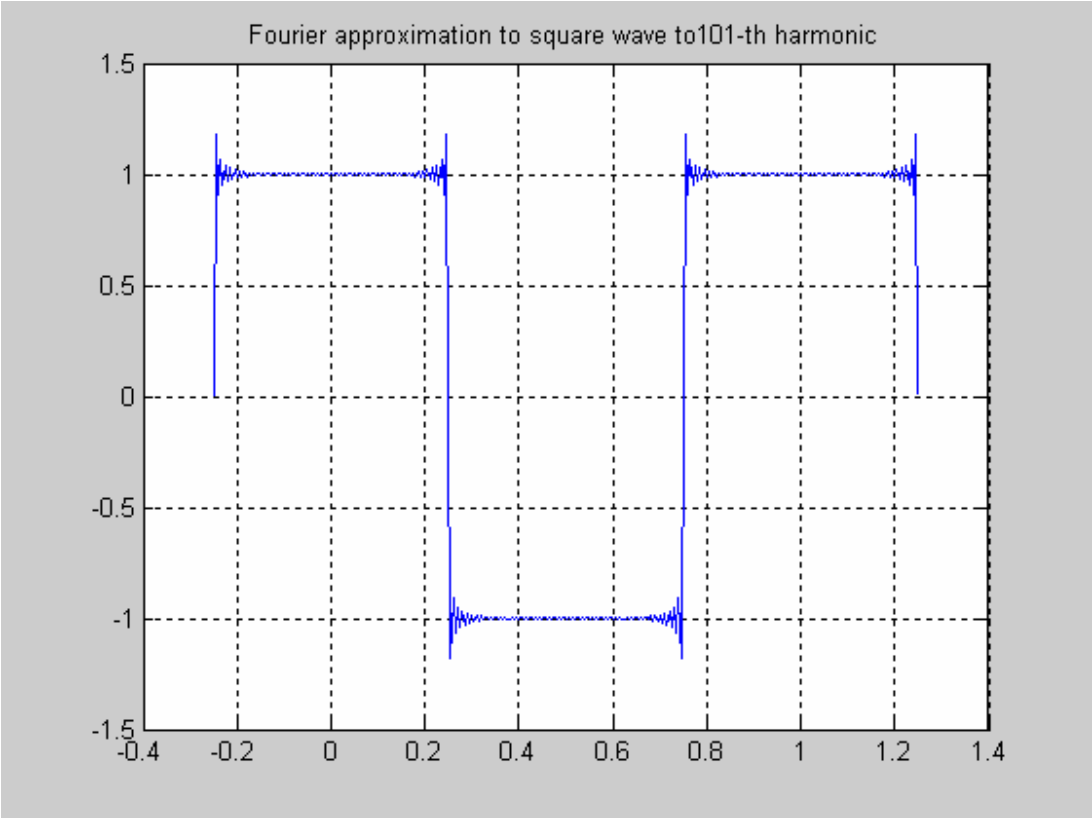
[start : increment : end] generates a vector starting at "start", ending at "end" and incrementing by "increment".
[0 : 0.25 : 1] generates the vector
(0, .25 .5, .75 1).

All of the lines involving the variable titl just generate the title for the graph. Look in reference section of PCMATLAB if you want to understand it.

v = cos (w0*t) does the following: each component of the vector t is multiplied by the scalar w0 and then the cos function is applied to each element of the resultant vector.

"for i = vector" executes the lines of code up to the "end" statement for i equal to each value of vector.

Fourier approximation to square wave to 101.000-th harmonic



Plotting Facilities

PLOT

Plot vectors or matrices. PLOT (X, Y) plots vector X versus Y. If X or Y is a matrix, then the vector is plotted versus the rows or columns of the matrix, whichever lines up. PLOT (X1, Y1, X2, Y2) is another way of producing multiple lines on the plot. PLOT (X1, Y1,':', X2, Y2, '+') uses a dotted line for the first curve and the point symbol + for the second curve. Other line and point types are:

solid	__	point	.	red	r
dashed	--	plus	+	green	g
dotted	:	star	*	blue	b
dashdot	-.	circle	o	white	w
		x-mark	x	invisible	i

PLOT (Y) plots the columns of Y versus their index. PLOT (Y) is equivalent to PLOT (real(Y), imag(Y) if Y is complex. In all other uses of PLOT, the imaginary part is ignored.

See SEMI, LOGLOG, POLAR, GRID, SHG, CLC, CLG, TITLE, XLABEL, YLABEL, AXIS, HOLD, MESH, CONTOUR, SUBPLOT.

POLAR

POLAR (THETA, RHO), makes a plot using polar coordinates of the angle THETA, in radians, versus the radius RHO.
See GRID for polar grid lines and PLOT for how to obtain multiple lines and different line-types.

XLABEL

XLABEL text, writes the text on the current plot beneath the x-axis.

YLABEL

YLABEL ('text') writes the text on the current plot beside the y-axis.

AXIS

Manual axis scaling on plots. Tying AXIS by itself freezes the current axis scaling for subsequent plots. Typing AXIS again resumes auto-scaling. AXIS returns a 4 element row vector containing the [x–min, x–max, y–min, y–max] used on the last plot.
AXIS (V) where V is a 4 element vector sets the axis scaling to the prescribed limits.
AXIS ('square') sets the plot to an approximately square ratio. In this mode, a line with slope 1 will be at a true 45 degrees, instead of skewed by the irregular shape of the screen. Thus PLOT (sin (t), cos (t)) will look like a circle instead of an oval. AXIS ('normal') sets the aspect ratio back to normal.

HOLD — Holds the current graph on the screen. Subsequent PLOT commands will add to the plot, using the already established axis limits, and retaining the previously plotted curves. HOLD ON turns on holding, HOLD OFF turns it off, and HOLD, by itself, toggles the HOLD state.

SUBPLOT — Graphing windows control. SUBPLOT(mnp), where, 'mnp' is a three digit number, breaks the graph window into an m-by-n matrix of small graph windows, and selects the p-th window for the current plot. For Example,

SUBPLOT (211), PLOT (1:3), SUBPLOT (212), PLOT (SIN (1 : 3))

Plots 1 : 3 on the top half of the screen and SIN (1 : 3) on the bottom half. After the first SUBPLOT command, subsequent plotting commands automatically advance to the next subplot window. So SUBPLOT (211), PLOT (1 : 3), PLOT (SIN (1 : 3)) achieves the same effect as above. CLG, SUBPLOT (111) or just SUBPLOT returns to the default single plot configuration.

CLC — Clear command window. See also HOME and CLG.

TITLE — TITLE ('text') writes the text at the top of the current plot as a title.

CONTOUR — CONTOUR surface. CONTOUR (Z) produces a contour plot of matrix using the values in Z as heights above a plane.
CONTOUR (Z, N) draws N contour lines, overriding the default automatic value.
CONTOUR (Z, V) draws LENGTH (V) contour lines at the locations specified in vector V.
CONTOUR (Z, N, X, Y) where X and Y are vectors, specifies the X- and Y- axes used on the plot. See also MESH.

MESH — Mesh surface. MESH (Z) produces a 3-dimensional mesh surface "picture" of matrix Z using the values in Z as heights above a plane. See MESHDOM to plot functions of two variables.

MESH (Z, M) specifies a view-point. The two-element vector M = [AS EL] contains AZ, the azimuth or horizontal rotation, and EL, the vertical elevation (both in degrees). Azimuth revolves about the z-axis, with positive values indicating counter-clockwise rotation of the view-point (clockwise rotation of the object). Positive values of elevation correspond to moving above the object; negative values move below. Here are some examples:

EL = 90 is directly overhead.
M = [0 0] looks directly up the first column of Z,
From the Z(m, l) element.
AZ = 180 is behind the matrix.
M = [-37.5 30] is the default

MESH (Z, S) and MESH (Z, M, S) control the scale factors used to set the X, Y and Z axes. The vector S is defined as S = [sx sy sz], where the three scalars, relative to each other, set the size of the object in each of the three dimensions. See also CONTOUR.

ANALYSIS AND SIMULATION OF CONTROL SYSTEMS USING MATLAB

MATLAB can be used to analyze systems described by transfer functions or state space. Since transfer functions are ratio of polynomials, let us see how MATLAB handles polynomials.

POLYNOMIALS

Polynomials in MATLAB are represented by a vector containing the coefficients in descending order.

- For example, the roots of the polynomial

$p(s) = s^3 - 5s^2 - 1$

can be found as shown ⇒

```
>> p=[1 -5 0 1];
>> r=roots(p)
r =
   4.9593
   0.4698
  -0.4292
```

- The coefficients of the polynomial can be found from the roots as shown ⇒

```
>> p=poly(r)
p =
   1.0000   -5.0000    0.0000    1.0000
```

- Product of polynomials can be accomplished easily using MATLAB. Suppose we want to expand the following polynomial

$n(s) = (3s^3 - s + 1)(s^2 + s)(s + 3)$

The MATLAB commands for this operation is shown

```
p1=[3 0 -1 1];p2=[1 1 0];p3=[1 3];
>> p1p2=conv(p1,p2)
p1p2 =
    3    3   -1    0    1    0
>> n=conv(p1p2,p3)
n =
    3   12    8   -3    1    3    0
```

OR

```
>> conv(conv([3 0 -1 1],[1 1 0]),[1 3])
ans =
    3   12    8   -3    1    3    0
```

- To evaluate any polynomial for any given value of the argument, the POLYVAL command can be used as shown

```
value=polyval(n,-4)
value =
       2244
value=polyval(n,-3)
value =
     0
```

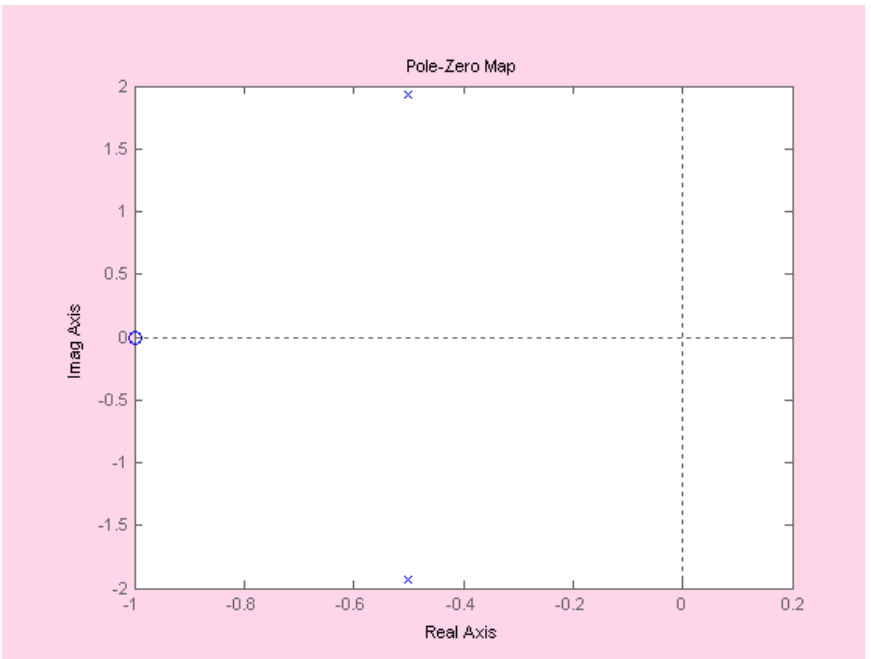
TRANSFER FUNCTIONS

- To find or plot the poles and zeros of any transfer function [TF], the PZMAP command can be used as shown. Suppose the TF is given by

$$G(s)=\frac{s+1}{s^2+s+4}$$

```
>> num=[ 1 1];den=[1 1 4];
>> [p,z]=pzmap(num,den)
p =
  -0.5000 + 1.9365i
  -0.5000 - 1.9365i
z =
   -1
```

```
pzmap([ 1 1],[1 1 4])
```



- To write a system in a transfer function form, the PRINTSYS command can be used as shown

```
num=[ 1 1];den=[1 1 4];
>> printsys(num,den)

num/den =

      s + 1
  --------------
  s^2 +  s + 4
```

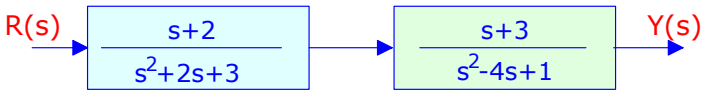
- To create a transfer function form, the TF command can be used as shown

```
num=[ 1 1];den=[1 1 4];
>> G=tf(num,den)

Transfer function:
   s + 1
--------------
s^2 + s + 4
```

BLOCK DIAGRAMS

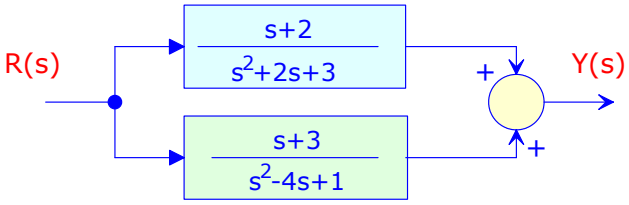
Block diagram reduction can be carried out using MATLAB commands. The following operations are examples of block diagram reduction.

- Series connection



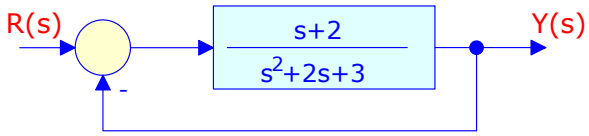

```
>> num1=[ 1 2];den1=[1 2 3 ];num2=[ 1 3];den2=[1 -4 1];
>> [num,den]=series(num1,den1,num2,den2)
num =
    0     0     1     5     6
den =
    1    -2    -4   -10     3
>> G=tf(num,den)

Transfer function:
       s^2 + 5 s + 6
-----------------------------
s^4 - 2 s^3 - 4 s^2 - 10 s + 3
```

- Parallel connection



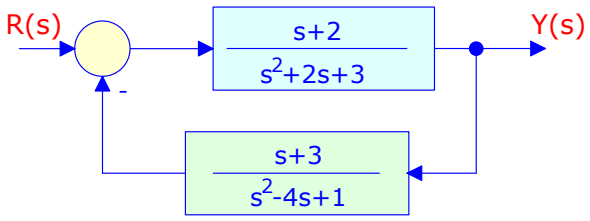

```
>> num1=[ 1 2];den1=[1 2 3 ];num2=[ 1 3];den2=[1 -4 1];
>> [num,den]=parallel(num1,den1,num2,den2)
num =
    0     2     3     2    11
den =
    1    -2    -4   -10     3
>> G=tf(num,den)

Transfer function:
  2 s^3 + 3 s^2 + 2 s + 11
------------------------------
s^4 - 2 s^3 - 4 s^2 - 10 s + 3
```

- Closed loop connection




```
>> num1=[ 1 2];den1=[1 2 3 ];
>> [num,den]=cloop(num1,den1,-1)
num =
    0     1     2
den =
    1     3     5

>> G=tf(num,den)

Transfer function:
   s + 2
-------------
s^2 + 3 s + 5
```

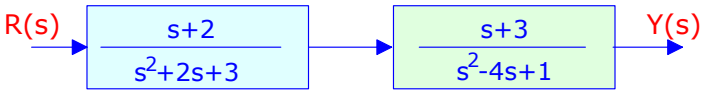
- Feedback connection




```
num1=[ 1 2];den1=[1 2 3 ];num2=[ 1 3];den2=[1 -4 1];
>> [num,den]=feedback(num1,den1,num2,den2,-1)
num =
    0     1    -2    -7     2
den =
    1    -2    -3    -5     9

>> printsys(num,den)

num/den =

     s^3 - 2 s^2 - 7 s + 2
  ----------------------------
  s^4 - 2 s^3 - 3 s^2 - 5 s + 9
```

- Connect

To derive state-space model for block diagram interconnection.

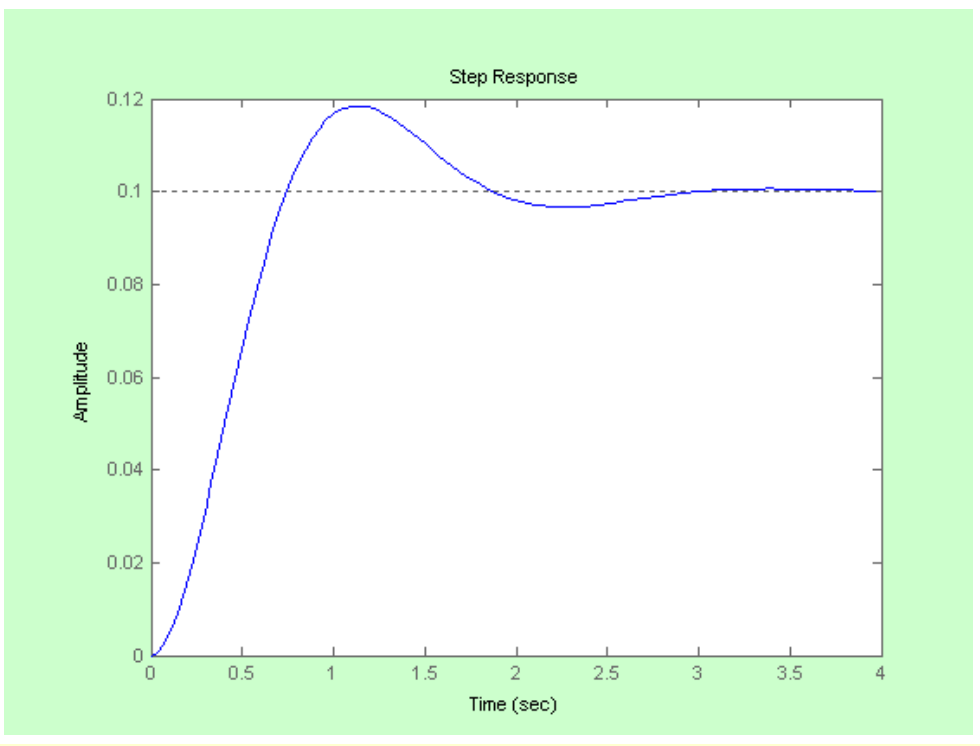



```
>> n1=[ 1 2];d1=[1 2 3 ];n2=[ 1 3];d2=[1 -4 1];nblocks=2;blkbuild;Q=[2 1];
State model [a,b,c,d] of the block diagram has 2 inputs and 2 outputs.
>> inputs=[1];outputs=[2];
>> [A,B,C,D]=connect(a,b,c,d,Q,inputs,outputs);
>> [NUM,DEN]=ss2tf(A,B,C,D,1)
NUM =
       0   -0.0000    1.0000    5.0000    6.0000
DEN =
   1.0000   -2.0000   -4.0000  -10.0000    3.0000
>> NUM=NUM(3:5);tf(NUM,DEN)

Transfer function:
      s^2 + 5 s + 6
-----------------------------
s^4 - 2 s^3 - 4 s^2 - 10 s + 3
```

STEP RESPONSE

The step response of control systems is very important. It can be simulated using the STEP command as shown

```
num=[ 1 ];den=[1 3 10 ];
>> t=[0:0.01:5];
>> step(num,den)
```



EE 380 - Control Engineering

Experiment # 2

Introduction to SIMULINK and Simulation of a Simple Speed Control System

OBJECTIVES:

1 To become familiar with the Dynamic system simulation software SIMULINK.
2 To simulate a simple first-order speed control system using SIMULINK.

INTRODUCTION

SIMULINK is a program for simulating dynamic systems. As an extension to MATLAB, SIMULINK adds many features specific to dynamic systems while retaining all of MATLAB's general purpose functionality.

SIMULINK has two phases of use: model definition and model analysis. A typical session starts by either defining a model or recalling a previously defined model, and then proceeds to analysis of that model.

To facilitate model definition, SIMULINK adds a new class of windows called block diagram windows. In these windows, models are created and edited principally by mouse driven commands. Part of mastering SIMULINK is to become familiar with the manipulation of model components within these windows.

After you define a model, you can analyze it either by choosing options from the SIMULINK menus or by entering commands in MATLAB's command window.

The progress of a simulation can be viewed while the simulation is running, and the final results can be made available in the MATLAB workspace when a simulation is complete.

TUTORIAL

To master SIMULINK, you must learn how to manipulate blocks and how to build models. You must also become familiar with the types of blocks available. Finally, you must learn how to use the analysis tools provided in SIMULINK. The construction and simulation of a simple model, described in the following section, introduces you to each of these concepts.

SIMULINK uses the metaphor of a block diagram to represent dynamic systems. Defining a system is much like drawing a block diagram. Instead of drawing the individual blocks, blocks are copied from libraries of blocks, either the standard block library supplied with SIMULINK or block libraries you build yourself.
The standard block library is organized into several subsystems, grouping blocks according to their behavior. Blocks can be copied from these or any other libraries or models into your model.

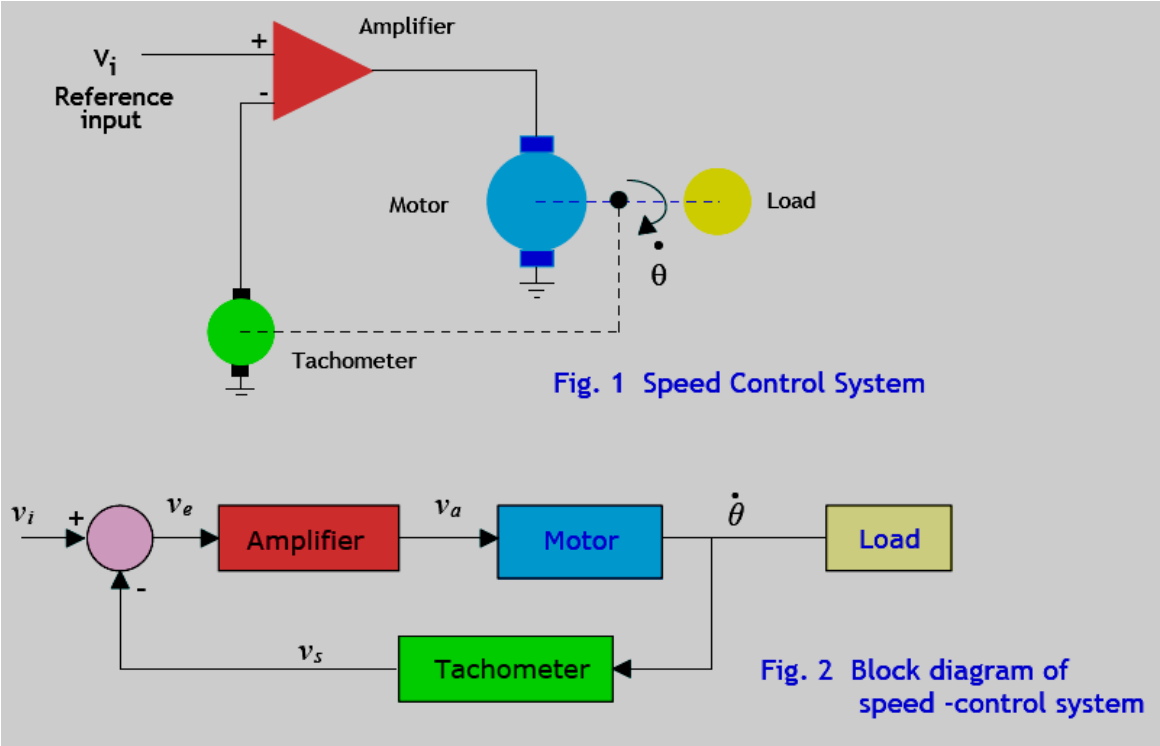
Example 1: Generate the sinusoid $4\sin(2\pi t)$and display it on the scope.

1. Double-click on the MATLAB icon to display the MATLAB command window.
2. Open the SIMULINK block library by entering the command SIMULINK, or by double-clicking on the SIMULINK icon . This command displays a new window containing icons for the subsystem blocks that make up the standard library.

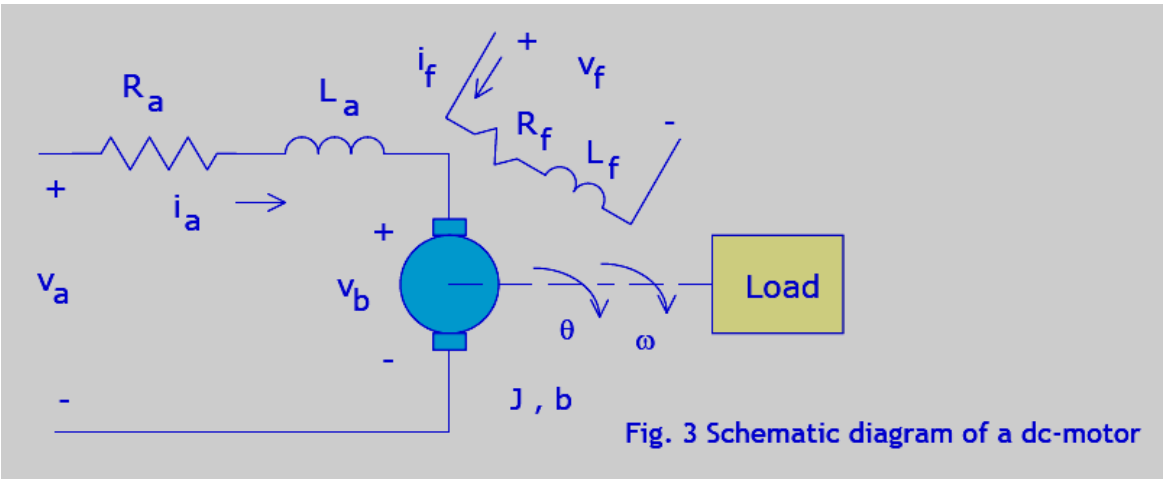
3. Open the Sources subsystem by double-cliking on the Sources block. This displays another SIMULINK window containing all the blocks that produce output but have no input.
4. Select **New** from the **File** menu to open a new system window.
5. Drag the Signal Generator block from the Sources window to the new, as yet untitled, window. [Notice that you only drag a copy of the block]
6. Open the Signal Generator Block by double-cliking on its icon. This displays a dialog box showing the controls for specifying the waveform, amplitude, and frequency of the signal generated by the block.
7. Select the text box marked **Frequency** by positioning the cursor on it, set the frequency to 1, and the **Units** to Hertz. Select the text box **Amplitude** and set the peak to 4. Click on the **OK** button to close this window.
8. Open the **Sinks library**, and drag a copy of the Scope block to the new system window.
9. Open the **Scope** to see that its window is a graphic representation of an oscilloscope. Use the pointer to select a suitable horizontal range. To set the Y-Axis Limits right-click on an axes and choose **Axes Properties**.
10. The angle bracket (>) pointing out of the **Signal Generator block's** icon represents its output port, and the angle bracket pointing into the Scope icon represents its input port. To Connect these two blocks , use the left mouse button to click on either the output or input port of one block, drag to the other block's input or output port to draw a connecting line, and then release the left mouse button. When the blocks are connected, the angle brackets disappear, and a line with an arrowhead shows the direction of the data flow.
11. Pull down the **Simulation** menu and choose **Simulation Parameters** . A dialog box opens showing all the simulation parameters that can be modified. Select the text box for the Maximum Step Size parameter, change the value to 0.01, and click on the OK button to close the simulation parameter dialog box.
12. Start the simulation by choosing Start from the **Simulation** menu. The **Signal Generator** outputs the sinusoidal waveform for each time step and the **Scope** shows its input as the trace of a sine wave.
13. The simulation stops when the stop-time in the Control Panel dialog box has been reached, or by choosing **Stop** on the **Simulation** menu.
14. Choose Save from the File menu to display a dialog box in which you can specify the filename and directory for this model.

Example 2: Speed-Control System

A speed-control system is shown in basic form in Fig. 1. The system is composed of a dc motor whose shaft speed *is to be held constant, a power amplifier that controls the motor voltage, a tachometer coupled to the motor shaft, and a circuit that detects the deviation or error between the reference input and the feedback signals. Fig. 2 is a block diagram representation of the system.



In analyzing the speed-control system, we will first determine the transfer function of the motor as shown in Fig. 3 :



The torque developed by the motor is given by:

$$T_m = K_1 \phi(t)\ i_a(t) \tag{1}$$

The air-gap flux of the motor is proportional to the field current (neglecting saturation), so that :

$$T_m = K_1 K_f\ i_f(t)\ i_a(t) \tag{2}$$

If the d.c. motor is armature- controlled, $i_f(t)$ is constant and :

$$T_m(s) = K_m I_a(s) \tag{3}$$

Also, we have :

$$V_a(s) = V_b(s) + (R_a + sL_a) I_a(s) \quad ; \quad V_b(s) = K_b \Omega(s) \quad \text{(back e.m.f.)} \tag{4}$$

The load torque for rotating inertia is given by:

$$T_l = Js^2\Theta(s) + bs\Theta(s) \tag{5}$$

Also, we have

$$T_m(s) = T_l(s) + T_d(s) \quad ; \tag{6}$$

[$T_d(s)$ is the disturbance torque and is often negligible.]

Using equations (3) to (6), we obtain

$$\frac{\Omega(s)}{V_a(s)} = \frac{K_m}{(R_a + sL_a)(Js + b) + K_b K_m}$$

However for many motors, the armature time constant La/Ra is negligible and therefore:

$$\frac{\Omega(s)}{V_a(s)} = \frac{K_m}{R_a b + K_b K_m} \frac{1}{(1 + s\frac{JR_a}{R_a b + K_b K_m})} = \frac{K_M}{1 + s\tau_m} \tag{7}$$

or in the time domain

$$\tau_m \dot{\omega}(t) + \omega(t) = K_M v_a(t)$$

for the purpose of the simulation, the following numerical values may be used:

K_A (amplifier gain)	K_M	τ_m	K_t (tachometer constant)
15 V/V	10 rad/sec/V	2.5 sec.	12 mV /rad/sec

referring to Fig. 2, the equations representing the block diagram of the speed-control system are:

$$\dot{\omega}(t) + \frac{1}{\tau_m}\omega(t) = \frac{K_m K_A}{\tau_m} v_e(t)$$

$$v_i(t) - v_s(t) = v_e(t)$$

$$v_s(t) = K_t \omega(t)$$

(9)

Note: The tachometer works essentially as a voltage generator, with the output voltage v_s proportional to the magnitude of the angular velocity ω of the input shaft.

Procedure

1. Draw a block diagram for the system.
2. For the open-loop system ($K_t = 0$), obtain a plot of ω(t). Assume v_i(t)=unit step
3. Calculate the value of τ_m from your graph and compare it with the theoretical value given.
4. How would your graph change if the inertia of the motor shaft *J* is doubled?
5. For the closed-loop case ($K_t = 12$ *mV/sec*), obtain a plot of ω(t).
6. Calculate the system time constant from your graph and compare it with the theoretical value.
7. How would your graph change if the inertia of the motor shaft *J* is doubled?
8. How would your graph change if the tachometer constant K_t is doubled?

Remark Open dcdemo.m for more information about DC motor control

Notes on Simulation

A simulation can be started from either the command line or the **Simulation** menu. All of the methods use the same arguments and menu parameters. :

Simulation from the Menu

1. A simulation can be run by selecting **Start** from the **Simulation** menu. Set the simulation parameters in the Control Panel dialog box by selecting **Simulation Parameters** from the **Simulation** menu.
2. The Control Panel dialog box has fields in which you can enter numbers or any legal MATLAB expression, for example, the variables “Start time”, Stop time”, “Min step size”, and “ Max step size”, which can be defined in the MATLAB workspace.
3. The return variables [t, x, y] are used to put the time, state, and output trajectories into the MATLAB workspace. The start and stop times for the simulation are set in the variables tstart, and tfinal. The integration parameters minstep,maxstep, and final control the relative local error, minimum step size, and maximum step size of the simulation.

Simulation from the Command Line

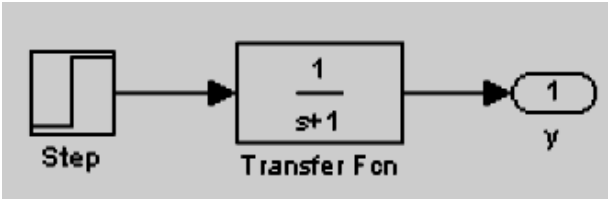
1. To configure a simulation with identical parameters as those described by the Control Panel dialog box, use the command
   [T,X,Y] = SIM('model',TIMESPAN,OPTIONS,UT)
   where model is the name of the name of a block diagram model.
   TIMESPAN is one of : Tfinal, [TStart TFinal], or [TStart OutputTimes TFinal]. OutputTimes are time points which will be returned in T, but in general T will include additional time points.
   OPTIONS : Optional simulation parameters. This is a structure created with SIMSET using name value pairs. UT : Optional extern input. Only the first parameter is required. All defaults will be

taken from the block diagram, including unspecified options. Any optional arguments specified will override the settings in the block diagram.

2. All of the integration algorithms have identical calling syntax so that a different method can be selected by simply changing the function name. [euler, rk23, rk45, sim, adams, gear].

SIMULATION STEPS

1. Open the SIMULINK block library by entering the command **simulink**
2. Open the **Continuous** subsystem by double-cliking on the Continuous block.
3. Select **New** from the **File** menu to open a new system window.
4. Build the block diagram of the system by dragging appropriate boxes.
5. Open the Sources subsystem by double-cliking on the Sources block. Drag the step function block .
6. Start the simulation.
7. Output Trajectories from SIMULINK can be plotted using Scope blocks, or Return variables and the MATLAB commands. For example:



Where the block labeled y is an output port taken from the **Sinks** list of blocks. Naming the system tfout ,

[t, x, y] = sim('tfout' , 2)

produces time histories, which can be plotted with plot(t, y)

Report

1 Obtain printouts of the block diagram. of the speed-control system.
2 Attach plots of all time responses for all cases considered. Comments on all your results.

University of Hail

Electrical Engineering Department

EE380 Control Engineering Exp. 3

Introduction to Servo Technology and CASSY Lab.

Objectives:

- To become familiar with the fundamentals of servo technology, its purposes, fundamental modules and machines.
- To identify the constants in the mathematical models of a DC motor and a tachometer experimentally.
- To become familiar with the Computer-Assisted Science-Systems (CASSY) lab.

Apparatus:

- DC-Power supply 726-88
- Servo setpoint potentiometer 734-10
- DC-Servo with tacho generator 734-44
- Power Amplifier 734-13
- Test Function Generator 734-40
- Stop-Watch
- Digital Multimeter
- CASSY Lab.

Introduction:

Servo system is a system to control mechanical instruments in compliance with variation of position or speed target value. The word “Servo” derives from the Greek “Servus (servant).” The system is called “Servo System” as it responds faithfully to command. The servo system is not the only alternative to control positioning and feed speed of mechanical facilities. Beside simple mechanical devices, however, the servo system is now the major control system to positioning and feed seed.

Usually, servos are equipped with a (small) driving machine, a DC motor or a single/multi-phase system. The major difference of servo motor compared with general use motors is that it has a detector to detect rotation speed and position.
The objective of this experiment is to introduce students with the basic equipments that they are going to use during the different lab sessions and to be able to use the CASSY lab to take measurements. A brief description of the system components is given in the appendix enclosed.

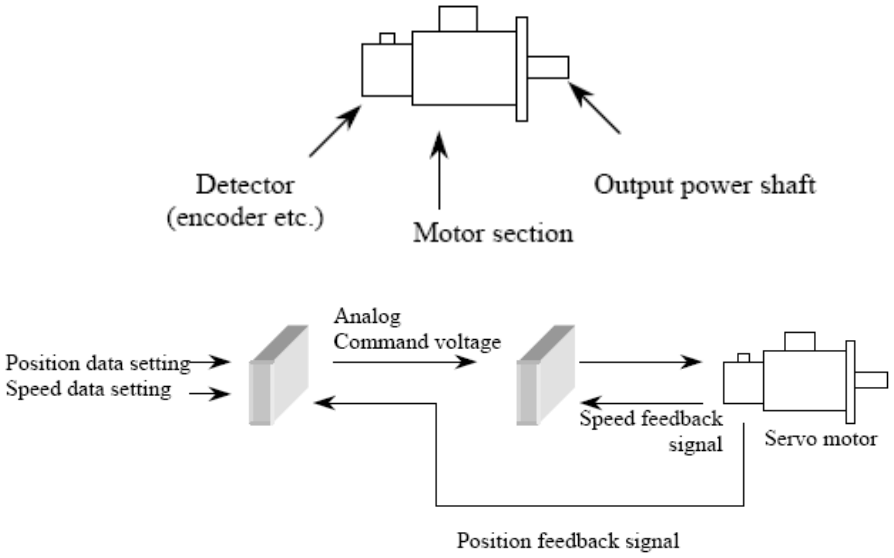


Figure 1: Diagram shows a configuration example of a servo system.

Procedure:

I. Introduction to CASSY-Lab

Refer to the enclosed start-up CASSY manual to start with the CASSY-Lab.

II. Characterization of the Setpoint Potentiometer

The setpoint potentiometer 734-10, Figure 2, consists of a potentiometer (1) with an angular scale. The potentiometer is fed by a stabilizer (+10 V) at one

terminal while the other terminal is connected to an inverter which generates (-10 V). The output (2) is buffered by an operational amplifier.
For certain control processes, it is important to make sure that the setpoint adjustment does not exceed certain rates of rise. For this reason, the output (2) is preceded by a "setpoint integrator" which makes it possible to adjust the rate of rise within ranges defined by X1 V/s up to X100 V/s (5); fine adjustments (1-10) can be made by means of V/s adjustor (6). On the ∞ setting, the setpoint integrator is not operational.
The input summing has a non-inverting input (7), to which disturbance variables or variables subordinate to the setpoint value can be feed-forwarded.

1. Connect the setpoint potentiometer to the DC-Power supply 726-88 as in Figure 3.
2. Adjust the setpoint integrator to ∞.
3. Adjust the setpoint value to 0^o (360^o), and the output voltage to 0.0V with the "zero" adjustor (3).
4. Adjust the setpoint value to 90^o, and the output voltage to 5.0V with the "scale" adjustor (4).
5. Start changing the set value clock-wise and counter clock-wise and complete table 1.
6. Draw Input-Output characteristic curve. Is it linear? Comment on you results.

	Counter Clock-wise									Clock-wise							
θ^o	175	150	120	90	60	30	20	10	0	10	20	30	60	90	120	150	175
V1				-5					0					5			

Table 1: θ is the angular displacement of the setpoint potentiometer and V1 is the resulting voltage

III. Measurement of tachometer coefficient [K_t]

The DC-Servo with tacho generator 734-44, Figure 4, consists of a geared potentiometer 30:1 (1) driven by a DC permanent-magnet motor. The motor is controlled by a proportional amplifier (Kp, min, max) (2). The amplifier is preceded by a summing point (3). This point is reached from the control input (4) via the switch (5). If the switch is actuated, the input signal is interrupted (controlling the zero-setting).
If the switch (6) is turned "on", a closed control loop is generated: motor has the desired value V_X at the potentiometer wiper output (9). Feedback with negative phase is supplied to summing point ($-V_X$), compared with V_W (input (4)) and the system deviation $V_{Xd} = V_W - V_X$ as well as the manipulated variable $V_y = V_{Xd} K_p$ are formed.

For this experiment keep switch (7) OFF. The function of this second negative feedback path will be discussed in coming experiments.

The tachometer output voltage v_t is proportional to the speed of its armature, i.e. $v_s = K_t\omega$, where ω is the angular speed. The following procedure can be used to identify the tachometer constant.

7. Assemble in accordance with the plug-in diagram shown in Figure 5 without both the 73461 unit and the feedback line U_X. The power amplifier 734-13 is used here to provide the required driving current (approximately 15-20 mA). This power amplifier has low internal impedance and little influence on the characteristics of the controlled system.
8. Set the input potentiometer on the 734-10 unit to zero and turn on the power.
9. Set the input potentiometer setting to 45°; the motor speed should change.
10. Measure the motor speed using a timer and the tachometer voltage using a voltmeter. Be sure to set the K_p to unity (by observing both the input and output voltages).
11. Calculate the value of K_t in V/rpm.
12. Repeat steps 10 to 12 for different potentiometer settings.
13. Tabulate your results and draw the input/output characteristic curve. Is it linear? Comment on your results.

IV. Measurement of the motor time constant [τ_m]

14. Assemble in accordance with the plug-in diagram shown in Figure 5.
15. Set the signal generator to provide a 0.1 Hz square wave.

16. Adjust the amplitude to obtain a steady state tachometer voltage of 5 volt.
17. Use the CASSY lab to observe the output voltage.
18. Calculate τ_m from the graph obtained in step 17. Remember that for a DC-motor with an input step, the output can be expressed as

$$\omega(t) = K_M (1 - \exp(-\frac{t}{\tau_m})$$

DISCUSSION:

1. Give a brief idea about the open and closed loop performance of motor speed control.
2. Explain how the speed of the motor could be controlled using the field circuit. Compare this method of control with the armature control method.

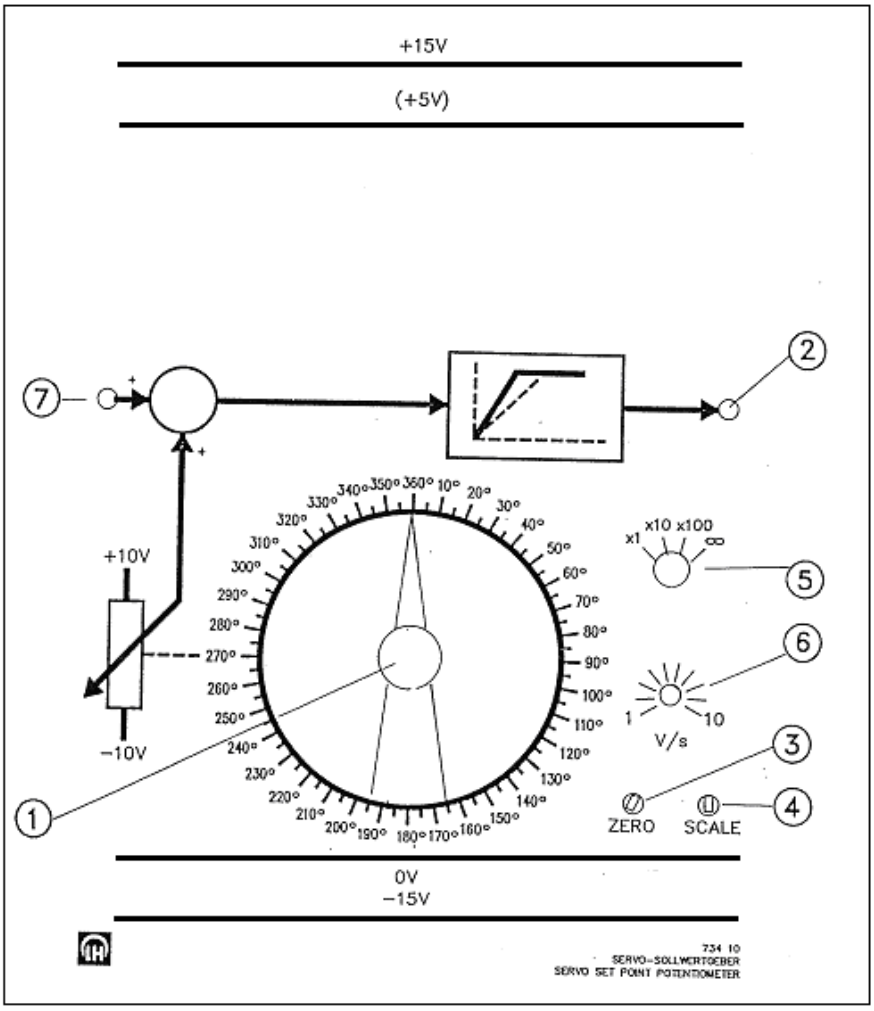


Figure 2. Schematic of set point potentiometer 734-10

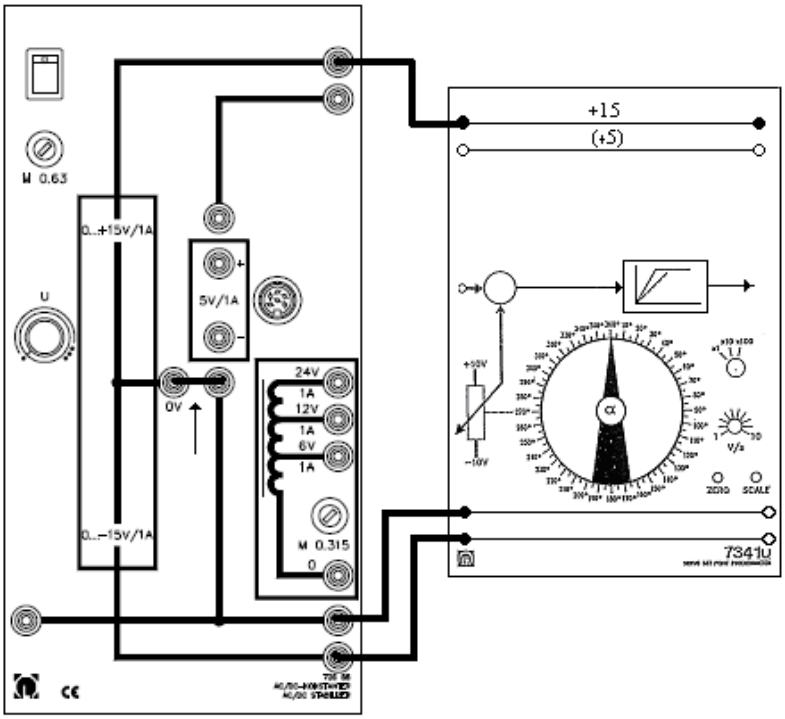


Figure 3. Wiring diagram of part II. You need to adjust the ±15V levels using the U-regulator. Also notice the Connection with the 0V level.

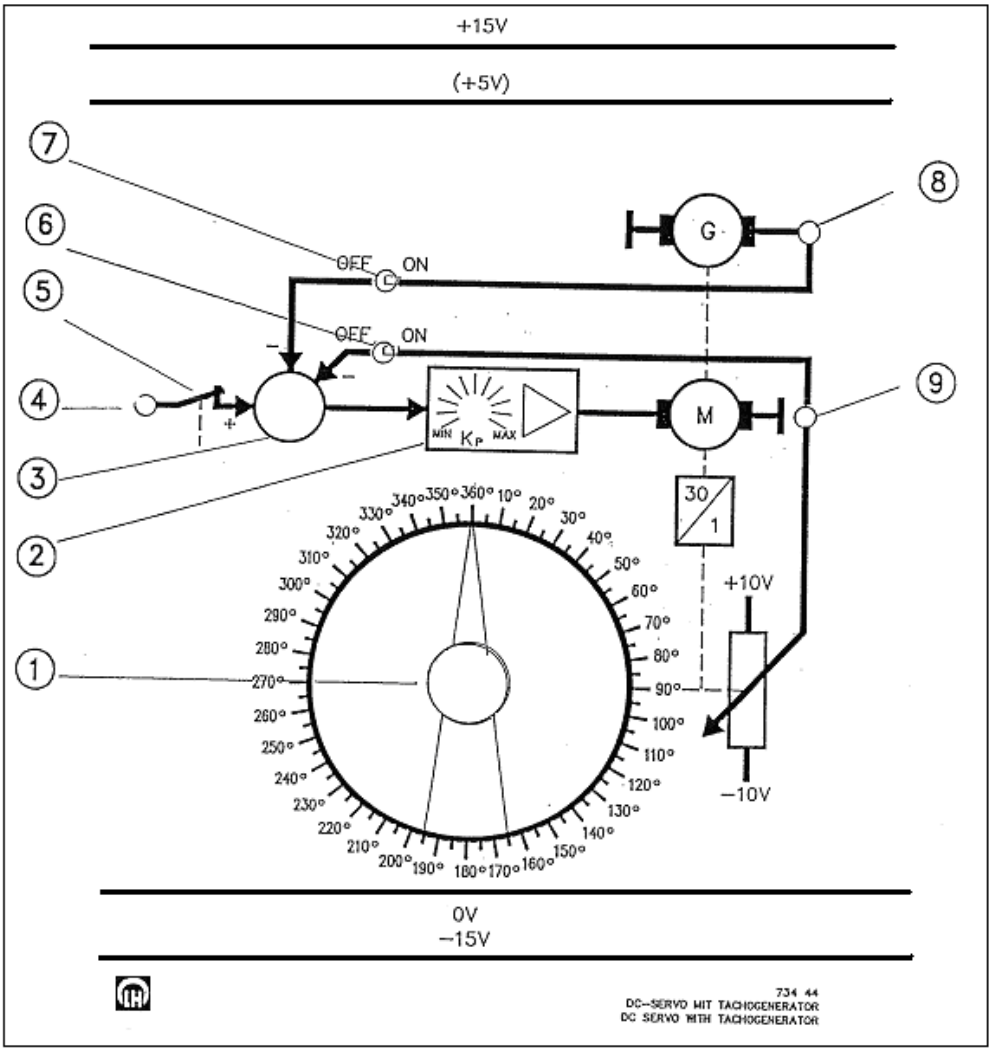


Figure 4. Schematic of set point potentiometer 734-44.

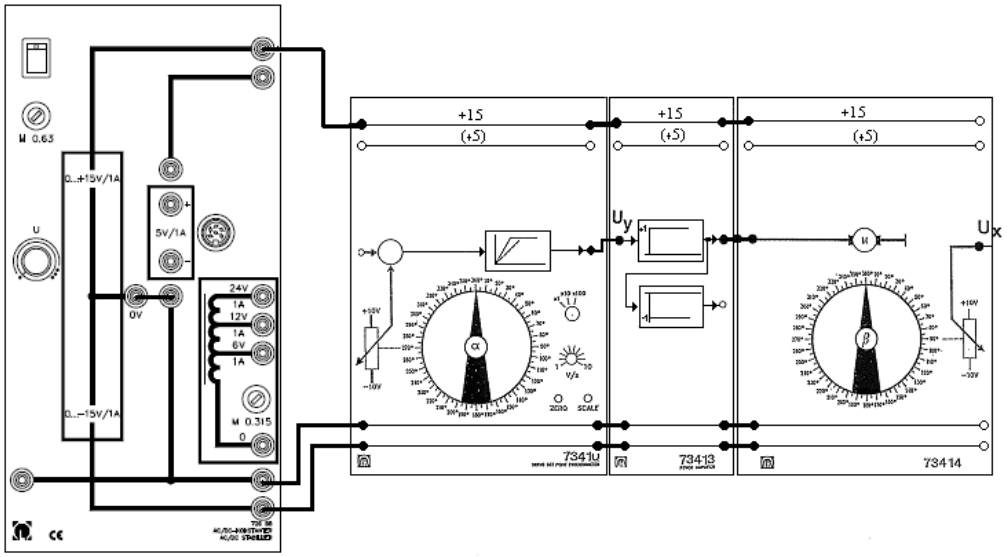


Figure 5. Wiring diagram of parts III and IV. Note that we will use the DC-servo with tacho generator 734-44 in stead of the 734-14 DC servo shown.

University of Hail

Electrical Engineering Department

EE380 Control Engineering Exp. 4

Servo Motor Position Control Using a Proportional Controller

Objectives:

- To become familiar with position control.
- To distinguish between positive and negative feedback.
- To study the effect of the gain employed on the effectiveness of the control system.

Apparatus:

- AC/DC stabilizer 726-88
- Servo setpoint potentiometer 734-10
- DC-Servo with tacho generator 734-44
- PID-Controller 734-061
- Toggle Switch, single-pole
- CASSY Lab.

Introduction:

Servos are “helpful souls” which spare human beings a lot of “adjusting work”. They are devices, usually for control purposes, which create the same conditions at location B (states, positions, speeds) as those prevailing or specified at location A. If transmission is to be as accurate as possible, the status at location B should be “fed back”, “compared” with the status at location A, and “adjusted” correspondingly (control loop).

Systems without status indication are termed “Open control loops”. Looking at the results of the adjustment (e.g. electric winding of car windows) we see that this is an example of “control with non-independent feedback.”

Such tasks are always performed by servomotors with powers ranging from approximately 20 to 20 KW.

Procedure:

1. Assemble in accordance with the plug-in diagram. Note that instead of the 726-86 and 734-14 units we will use the 726-88 and 734-44 units respectively.
2. With switch S1 ➔ OFF, Adjust the setpoint integrator to ∞.
3. Adjust the setpoint value to 0^o (360^o), and the output voltage to 0.0V with the "zero" adjustor.
4. Adjust the setpoint value to 90^o, and the output voltage to 5.0V with the "scale" adjustor.
5. Connect the output of the setpoint potentiometer directly to the input of the DC-Servo.
6. Adjust Kp associated with the DC-Servo 734-44 to unity. Note that switches (6) and (7) should be kept OFF.
7. Start changing the angular displacement of the setpoint potentiometer and write down your observations. What is the type of this system?
8. Connect the PID-controller once again and adjust the PID Kp = 0.1 with Ioff and Doff.
9. Repeat step 7 and record the values of both the input and out put angles in the table below. Explain the results.

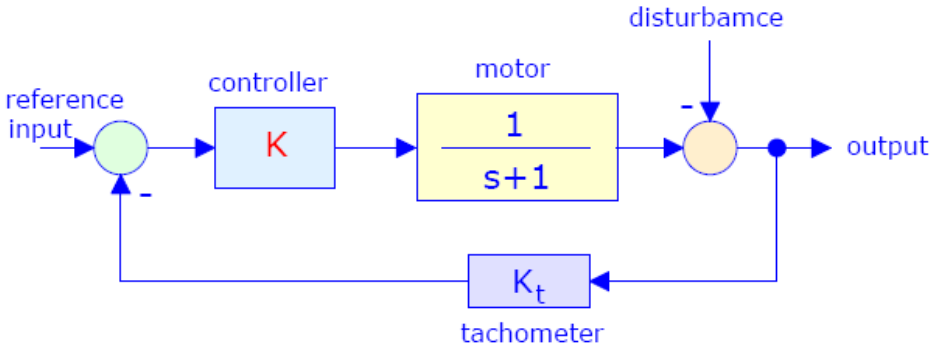
	Counter Clock-wise								Clock-wise						
$\theta_{in}{}^o$	150	120	90	60	30	20	10	0	10	20	30	60	90	120	150
$\theta_{out}{}^o$								0							

Table 1: θ_{in} is the angular displacement of the setpoint potentiometer and θ_{out} is the resulting output angle of the DC servo. *Kp=0.1*

10. Repeat step 9 for Kp = 1, 5, and 10. Comment on your results.
11. How do you explain the occurrence of controlled oscillations?

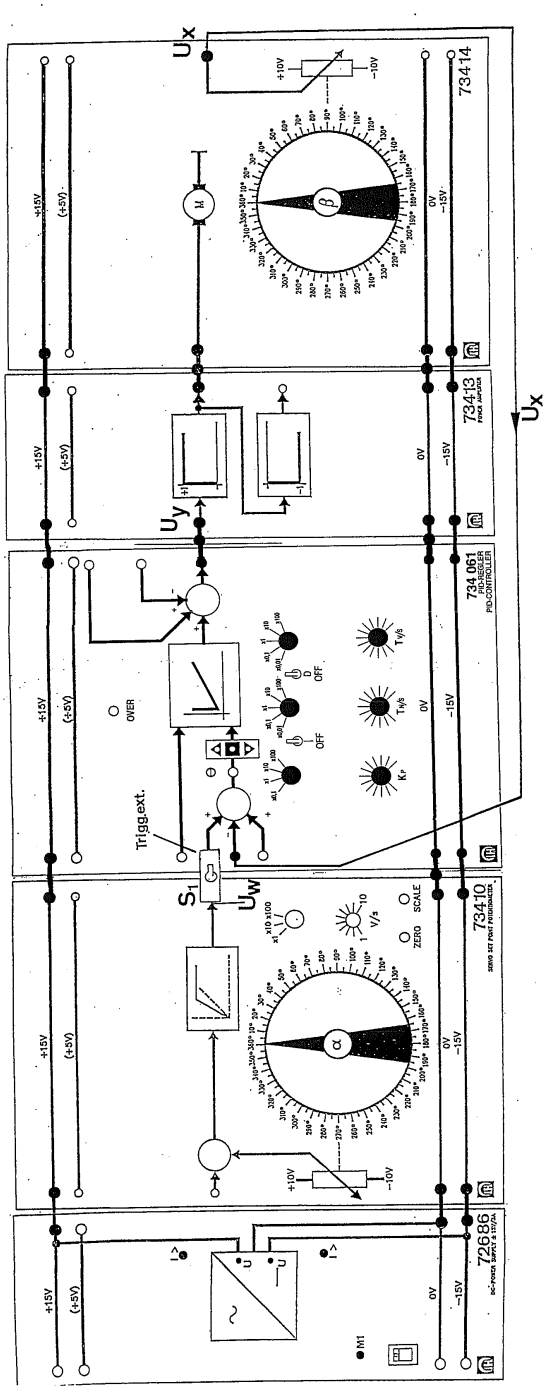
Simulink Exercise:

Use SIMULINK to simulate the behavior of the motor speed control system shown in the figure below for 10 seconds, and for different controller gains (K=1, K=10, K=20, K=50, K=80, and K=100), using a unit-step reference input and no disturbance. (Assume *Kt* = 1 volt/rad/sec).



a. Explain the effect of increasing the gain on the performance of the system.
b. Plot the steady-state error vs. K.
c. Plot the closed-loop time constant vs. K.
d. What would you recommend as being the best setting for the proportional controller *K*?

Plug-in diagram



University of Hail

Electrical Engineering Department

EE380 Control Engineering Exp. 5

Time Response of Second order Systems

Objectives:

- To develop an improved understanding of the very important class of 2nd order control systems and to become familiar with its typical characteristic response.

Apparatus:

- AC/DC stabilizer 726-88
- Test function generator 734-40
- Test element of the 2nd order
- CASSY Lab.

Introduction:

Although true second-order control systems are rare in practice, their analysis generally helps to form a basis for the understanding of analysis and design of higher-order systems.

The second-order system shown in Fig. 1 is defined as the **prototype second-order system.** The closed-loop transfer function of the system is:

$$\frac{Y(s)}{R(s)} = \frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2}$$

Where w_n is the undamped natural frequency and ξ is damping ratio.

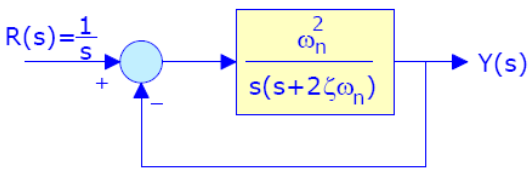


Fig. 1 Prototype second-order system

For a unit step input, the output response of the system is

$$Y(s) = \frac{w_n^2}{s\left(s^2 + 2\xi w_n s + w_n^2\right)}$$

The dynamic behavior of the system can be described in terms of the two parameters w_n and ξ.

(1) Under damped case ($0 < \xi < 1$)

In this case the unit step response can be found as

$$y(t) = 1 - \frac{1}{\beta} e^{-\zeta w_n t} \sin(w_n \beta t + \theta)$$

where

$$\beta = \sqrt{1-\zeta^2}, \quad \theta = \cos^{-1}(\zeta), \qquad \text{and } 0 \langle \zeta \langle 1.$$

(2) Critically damped case ($\xi = 1$)

In this case the unit step response can be found as

$$y(t) = 1 - e^{-w_n t}(1 + w_n t)$$

(3) Over damped case ($1 < \xi$)

In this case the unit step response can be found as

$$y(t) = 1 + \frac{1}{2\sqrt{\zeta^2 - 1}}\left(\frac{e^{-S_1 t}}{S_1} - \frac{e^{-S_2 t}}{S_2}\right)$$

where

$$S_1 = \zeta\omega_n + \omega_n\sqrt{\zeta^2 - 1}, \text{ and } \quad S_1 = \zeta\omega_n - \omega_n\sqrt{\zeta^2 - 1}$$

(4) Undamped case ($0 = \xi$)

In this case the unit step response can be found as

$$y(t) = 1 - \cos(w_n t)$$

Typical step responses of the prototype second-order system, for the four cases considered, are shown in Fig. 2. Notice that the critically damped case is the division between the overdamped cases and the underdamped cases and is the fastest response without overshoot.

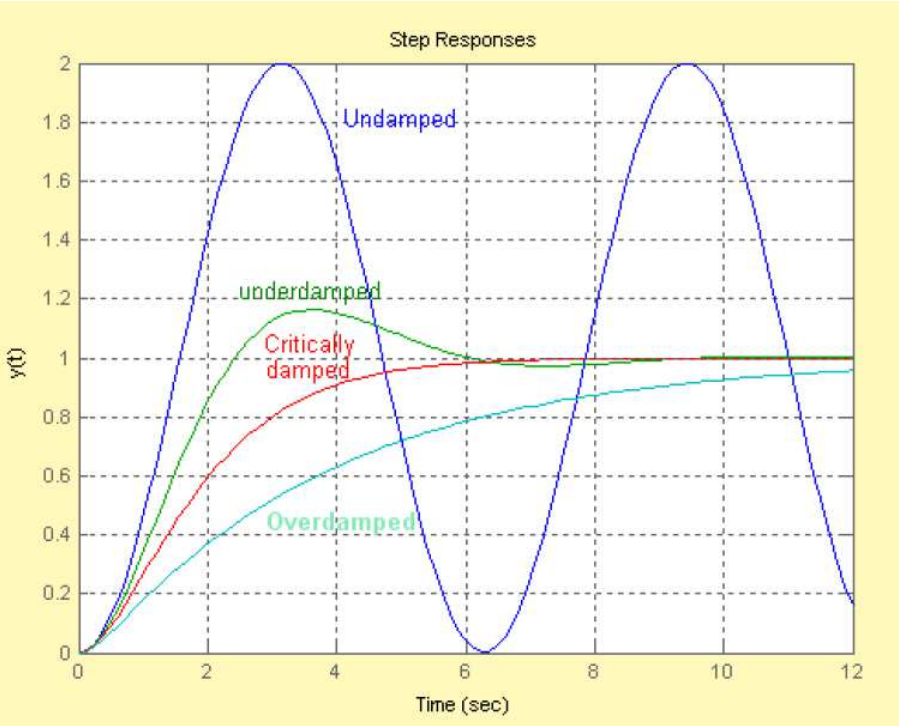


Figure 2; Typical Step responses of a prototype second order system

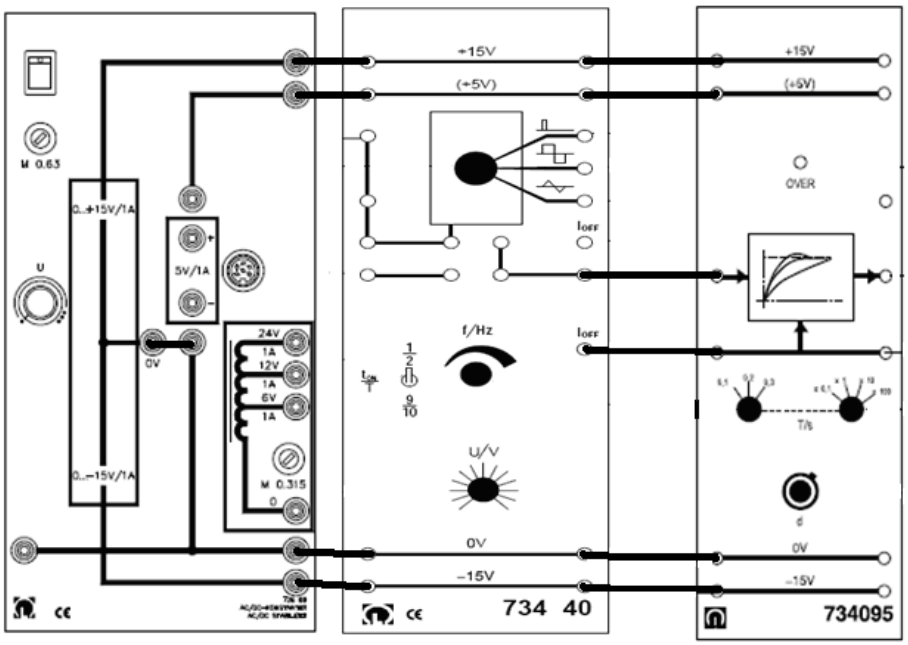
Procedure:

1. Assemble in accordance with the plug-in diagram.
2. Select the square wave form at the output of the test function generator. Set $\frac{t_{on}}{T} = \frac{1}{9}$ with minimum possible frequency. Adjust the pulse amplitude to unity.
3. Set w_n of the 2nd order element to approximately 1 rad/sec by setting the joint fine adjustment of the time constant to 0.1 sec. and the joint multiplication factor to x10.
4. Obtain plots of the input and the output for values of $d = \xi = 0$, 0.5, 0.707, 1.0, and 3.0. Complete table 1.
5. Obtain a plot of the maximum overshoot (Mp) versus the damping ratio.
6. Obtain a plot of the peak time (Tp) versus the damping ratio.
7. Obtain a plot of the rising time (Ts) versus the damping ratio.
8. Repeat steps 2 – 7 for $w_n = 0.2$ rad/sec.
9. Comment on your results.
10. Compare your results with those obtained from the Simulink exercise.

Table 1

d	Mp	Tp	Tr
0			
.5			
0.7			
1.0			
3			

Plug-in diagram



Simulink Exercise:

Use SIMULINK to simulate the behavior of the second order system assuming the following values; step input amplitude A = 1, w_n = 1 rad/sec, and zero initial conditions.

a) Obtain plots of y(t) for values of ξ = 0, 0.1, 0.5, 0.707, 1.0, 2.0 and 3.0
b) Find the **rising time** and the **maximum overshoot** obtained for all cases. Write your results in a suitable table. Comment on your results.
c) Obtain a plot of the maximum overshoot versus the damping ratio.
d) Obtain a plot of the rising versus the damping ratio.

Extra work:

Repeat the above exercise for w_n = 2 rad/sec. Comment on your results.

University of Hail
Electrical Engineering Department
EE380 Control Engineering Exp. 6

Investigation of Error Performance Indices

Objectives:

- To investigate the different error performance indices used in the design of control systems.

Apparatus:

- AC/DC stabilizer 726-88
- Test function generator 734-40
- Test element of the 2nd order
- CASSY Lab.

Introduction:

In the design of a control system, it is important that the system meets given performance specifications. Since control systems are dynamic, the performance specifications may be given in terms of the transient response behavior to specific inputs, such as step inputs, ramp inputs, etc. or the specifications may be given in terms of a performance index.

A performance index is a number which indicates the "goodness" of system performance. A control system is considered optimal if the values of the parameters are chosen so that the selected performance index is minimum or maximum. The optimal values of the parameters depend directly upon the performance index selected.
A performance index must offer selectivity; that is, an optimal adjustment of the parameters must clearly distinguish nonoptimal adjustments of the parameters. In addition, a performance index must yield a single positive number or zero, the latter being obtained if and only if the measure of the deviation is identically zero. To be useful, a performance index must be a function of the parameters of the system, and it must exhibit a maximum or a minimum. Finally, to be practical, a performance index must be easily computed, analytically or experimentally.

In control systems, consideration is given to several error criteria in which the corresponding performance indexes are integrals of some function or

weighted function of the deviation of the actual system output from the desired output. Since the values of the integrals can be obtained as functions of the system parameters, once a performance index is specified, the optimal system can be designed by adjusting the parameters to yield, say, the smallest value of the integral.

Various error performance indexes have been proposed in the literature. We shall consider the following four indexes in our study.

1	**ISE**	$J_1 = \int_0^\infty e^2\,dt$	Integral square-error criterion
2	**ITSE**	$J_2 = \int_0^\infty te^2\,dt$	Integral-of-time-multiplied square-error criterion
3	**IAE**	$J_3 = \int_0^\infty \lvert e\rvert\,dt$	Integral absolute-error criterion
4	**ITAE**	$J_4 = \int_0^\infty t\lvert e\rvert\,dt$	Integral-of-time-multiplied absolute-error criterion

Table 1; Different Error Performance Indexes.

- Characteristics of Error Performance Indexes

Integral square-error criterion (J_1)

- ✓ Easy to compute both analytically and experimentally.
- ✓ Weighs large errors heavily and small errors lightly.
- ✓ Not very selective.
- ✓ A system designed by this criterion is oscillatory and has poor relative stability.
- ✓ Has practical significance because the minimization of the performance index results in the minimization of power consumption for some systems, such as spacecraft systems.
- ✓ Figure 1 shows desired output $x(t)$, actual output $y(t)$, error $e(t)$, square error $e^2(t)$, and $e^2(t)dt$ as a function of t.

Integral-of-time-multiplied square-error criterion (J_2)

- ✓ Large initial error is weighed lightly, while errors occurring late in the transient response are penalized heavily.
- ✓ Better selectivity than the integral square-error criterion.

Integral absolute-error criterion (J_3)

- ✓ Results in a system which has reasonable damping and a satisfactory transient-response
- ✓ Selectivity is not too good

- ✓ Cannot be easily evaluated analytically
- ✓ Minimization of the integral absolute error is directly related to the minimization of fuel consumption of spacecraft systems.
- ✓ Figure 2 shows the error $e(t)$, and the absolute error $e(t)$, versus t.

Integral-of-time-multiplied absolute-error criterion (J_4)

- ✓ Large initial error is weighed lightly, while errors occurring late in the transient response are penalized heavily.
- ✓ Results in a system which has small overshoot and well damped oscillations.
- ✓ Has good selectivity and is an improvement over IAE.
- ✓ Very difficult to evaluate analytically.

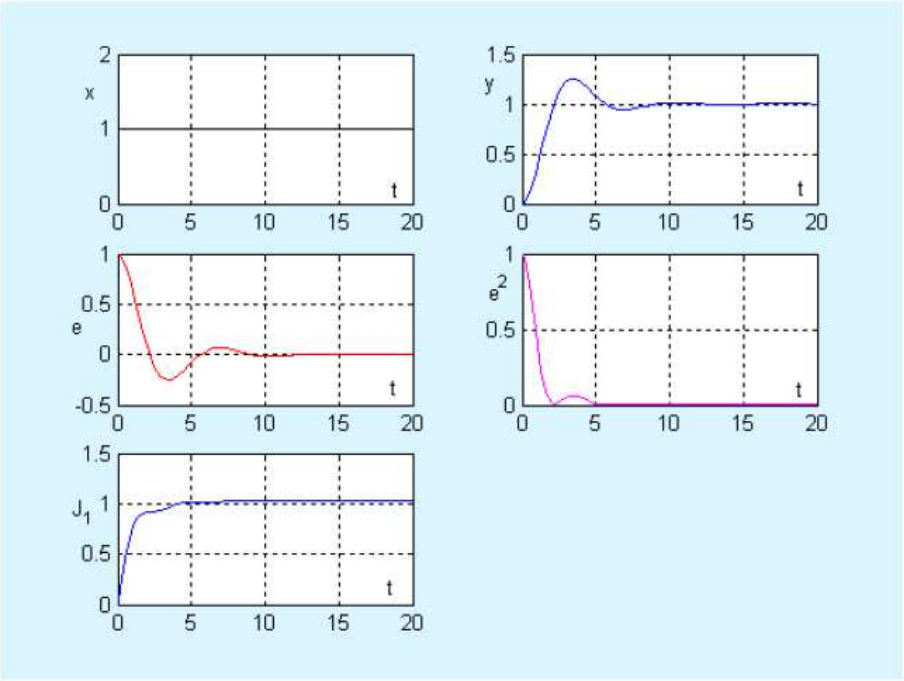


Fig. 1 Plots showing desired output $x(t)$, actual output $y(t)$, error $e(t)$, square error $e^2(t)$, and $\int e^2(t)dt$ as a function of t.

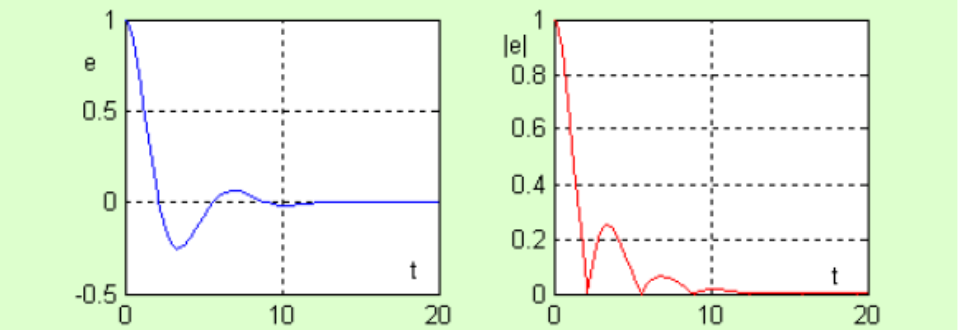


Fig. 2 Plots showing error $e(t)$, and absolute error $|e(t)|$ versus t.

Procedure:

1. Assemble in accordance with the plug-in diagram.
2. Select the square wave form at the output of the test function. generator. Set $\frac{t_{on}}{T} = \frac{9}{10}$ with minimum possible frequency. Adjust the pulse amplitude to unity.
3. Set w_n of the 2nd order element to approximately 1 rad/sec by setting the joint fine adjustment of the time constant to 0.1 sec. and the joint multiplication factor to X10.
4. Using CASSY-Lab, evaluate the four different error performance indexes for values of $d = \xi = 0.1, 0.5, 0.707, 1.0, 2.0$ and 3.0. Tabulate your results.
5. Plot the variations of each performance indexes versus the system damping ratio ξ.
6. Comment on your results.
7. Compare your results with those obtained from the Simulink exercise.

Simulink Exercise:

In this laboratory exercise, we shall solve a simple optimization problem in which we minimize the performance indexes discussed earlier. The system to be studied is shown in Fig. 1.

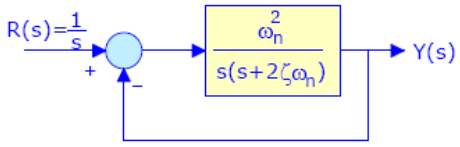


Fig. 1 Control system

$$\frac{Y(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \qquad \frac{E(s)}{R(s)} = \frac{s^2 + 2\zeta\omega_n s}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

1. Suppose that the input to the system, $R(s)$, is a unit-step applied at $t = 0$. Assume that the system is initially at rest.
2. Assume that $0.1 \le \zeta \le 2.0$, and that $\omega_n = 1.0$.
3. Using SIMULINK , develop a program to evaluate the performance indexes $J_1 \ldots J_4$ for the specified range of ζ , in appropriate steps., and the optimum value for ζ in each case.
4. Plot the variations of each of the performance indexes against the system damping ratio ζ. (preferably on a single graph).

DISCUSSION:

1. Comments on all your results. Is the optimal value of ζ the same for all performance indexes? Explain your answer.
2. Give a qualitative comparison of the selectivity of the various performance indexes considered.
3. Obtain , analytically, the optimal value of ζ that will minimize the performance index J_1 and the minimum value of J_1 [see 'Hint']. Compare these values with the corresponding values obtained from plotting your simulation results.
4. Repeat part (3) for the performance index J_2.
5. Addition of the square of error rate to the integrand of J_1 results in the performance index $J_5 = \int_0^\infty (e^2 + \dot{e}^2)\,dt$. Plot the variations of the performance index J_5 against the system damping ratio ζ. . Comments on the optimal value of ζ and the selectivity of J_5.

6. Addition of the absolute error rate to the integrand of J_4 results in the performance index $J_6 = \int_0^\infty t(|e| + |\dot{e}|)\, dt$. Plot the variations of the performance index J_6 against the system damping ratio ζ. . Comments on the optimal value of ζ and the selectivity of J_6.

HINT

$$J_1 = \int_0^\infty e^2 dt = \lim_{t\to\infty} \int_0^t e^2 dt = \lim_{s\to 0} s\{\frac{1}{s}\mathcal{L}[e^2(t)]\}$$

$$E(s) = \frac{s^2 + 2\zeta s}{s(s^2 + 2\zeta s + 1)} = \frac{s + \zeta}{s^2 + 2\zeta s + 1} + \frac{\zeta}{s^2 + 2\zeta s + 1}$$

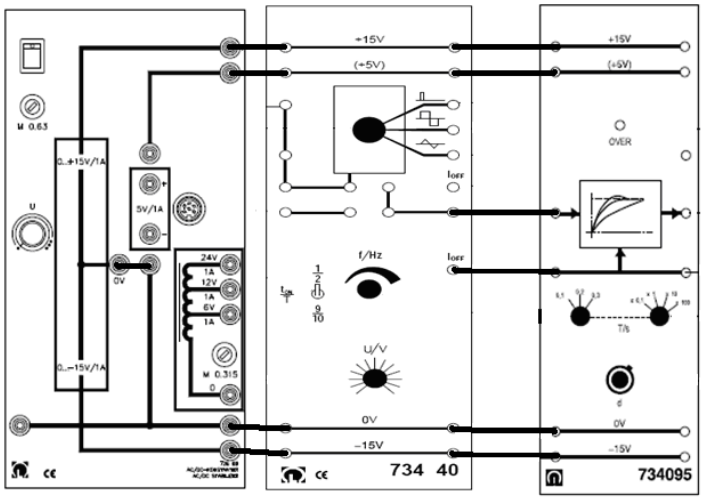
$$e(t) = \ldots$$

$$e^2(t) = \ldots$$

$$\mathcal{L}[e^2(t)]\} = \ldots$$

$\vdots$

Plug-in diagram



University of Hail
Electrical Engineering Department
EE380 Control Engineering Exp. 7

Servo Motor Speed Control with P/PI Controllers

Objectives:

- To investigate the effect of using Proportional (P) controller on the performance of closed loop systems.
- To investigate the effect of using Proportional plus Integral (PI) controller on the steady state error of closed loop systems.

Apparatus:

- AC/DC stabilizer 726-88
- Servo setpoint potentiometer 734-10
- DC-Servo with tacho generator 734-44
- PID-Controller 734-061
- Toggle Switch, single-pole
- CASSY Lab.

Introduction:

An important aspect of closed-loop control is speed control, which has many industrial applications, varying from heavy industrial, such as paper mills or steel rolling mills, to tape or video transport mechanisms.
The block diagram of the closed-loop speed control is shown in Fig. 1. The feedback signal is the output velocity signal *Vs*, normally from a tachometer,
which is compared with a reference voltage *Vr* to give an error $Ve = Vr - Vs$.
In operation the reference is set to a required value, which drives the motor to generate *Vs*, which reduces the error until the system reaches a steady speed. If the motor is loaded, the speed falls; this tends to increase the error, increasing the motor drive and thus reducing the speed fall for a given load. Note that this implies negative feedback around the loop. The speed fall with load is a very important characteristic in speed control systems.
The rotation direction can be reversed by reversing the reference voltage, though many industrial speed control systems are required to operate in one direction only.

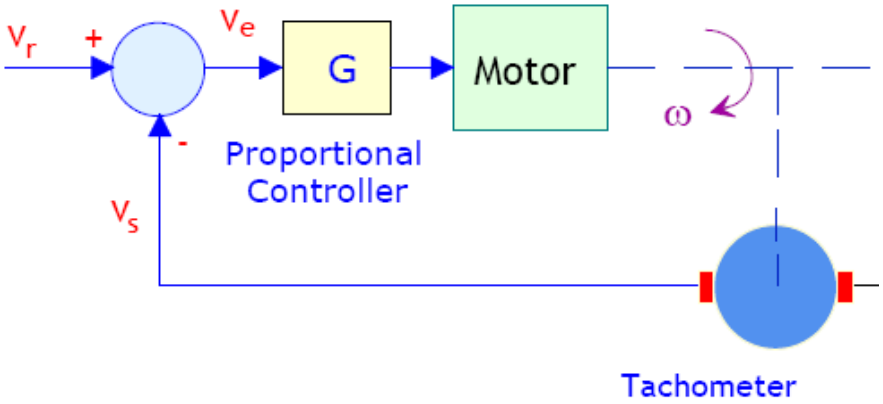
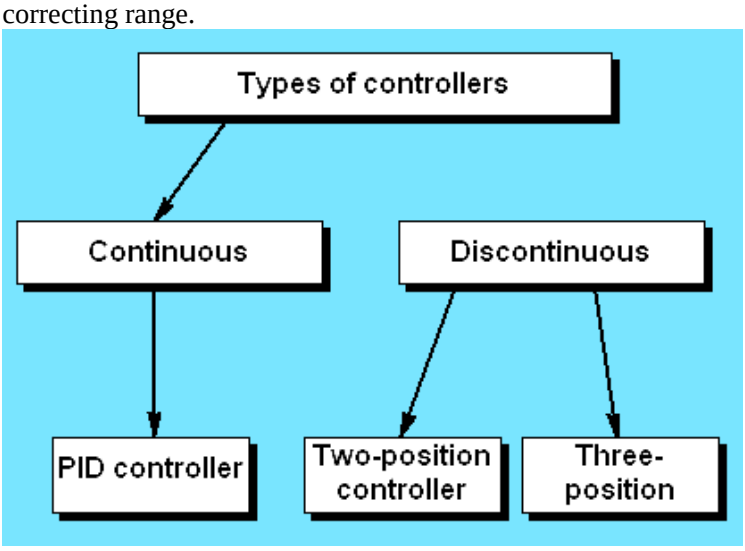


Figure 1: Closed loop speed control system.

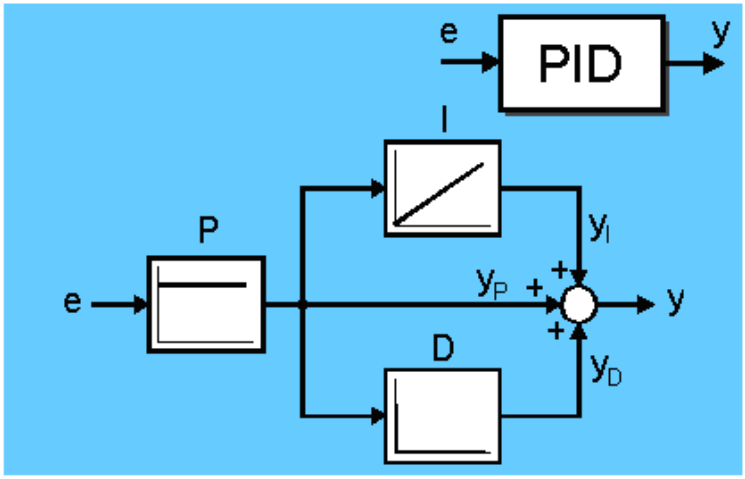
Increasing the forward path gain G will reduce the error *Ve* between the reference voltage *Vref* and the speed voltage *Vs* . A possible problem associated with increased gain is the saturation of the power amplifier. This problem can be mitigated by the use of a PI controller. Provided that there is an integrator in the forward path, the system will always adjust so that *Ve* is zero, and hence the speed voltage *Vs* always equals *Vref*.

Controller types

Classic controllers can basically be broken down into two main groups: continuous and discontinuous controllers. In the case of continuous controllers like the PID controller; the manipulated variable can assume any given value within a correcting range. Whereas with discontinuous controllers so-called switching controllers like two and three position controllers – the manipulated variable can only assume a few (discrete) values within the correcting range.

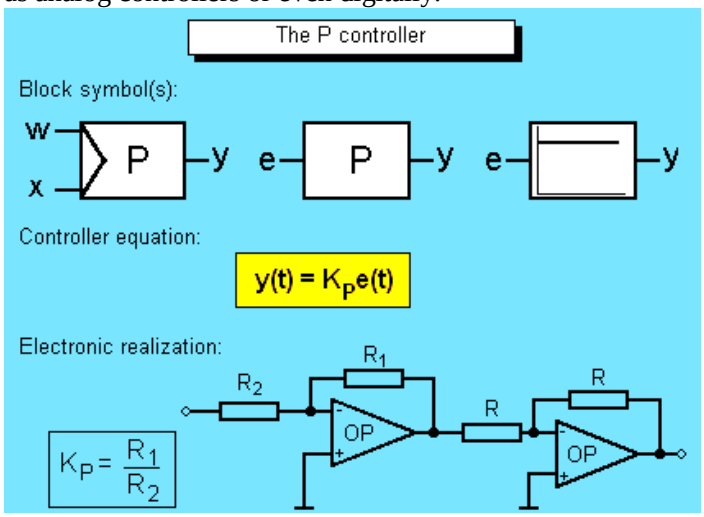


The **PID** controller is the most popular continuous controller in use. It is comprised of the parallel configuration of **P**roportional, **I**ntegral and **D**erivative elements, whereby all of the components can be given mutually independent parameters. This allows the PID controller to be optimally adapted to virtually every kind of controlled system. By connecting or disconnecting individual controller components, controller subtypes can be configured like P, PI or PD controllers.



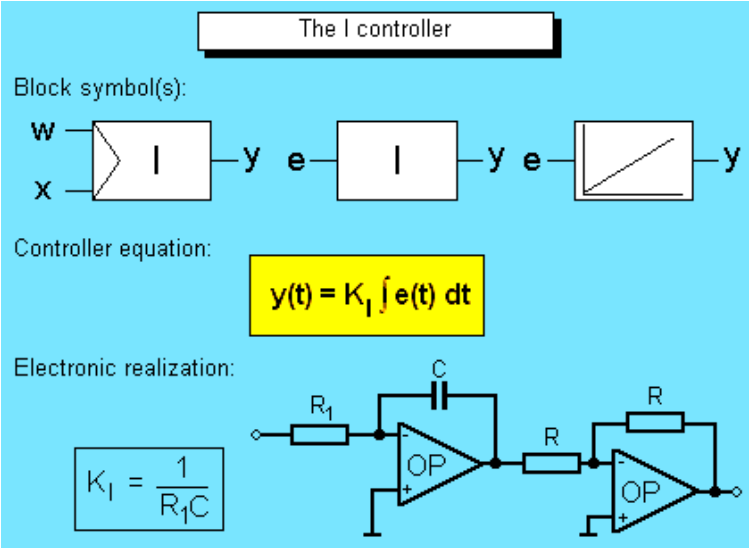
Proportional-action control
P controllers

The P controller corresponds to a conventional P element with the proportional-action coefficient K_P. It produces a manipulated variable y(t) which is proportional to the error signal e(t): the bigger the error signal the greater the control action of the P controller. As proportional-action controls generally entail residual system deviation, they are only used for simple process control operations. P controllers are easy to design electronically as analog controllers or even digitally.



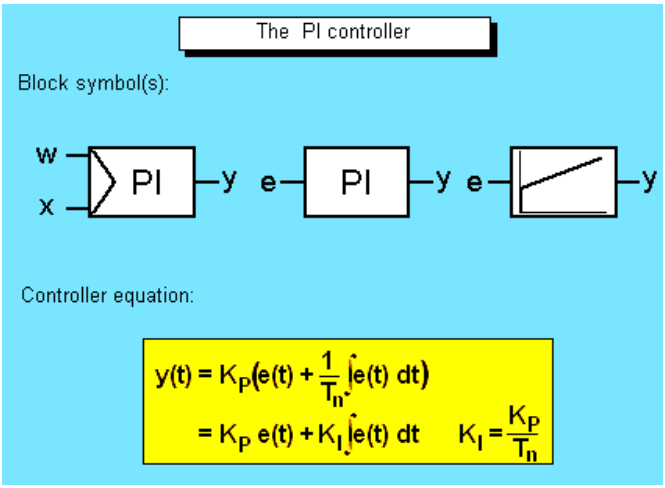
Integral-action control
***I*- controllers**

If only the ***I*** branch of the PID controller is enabled, you obtain an ***I*** controller. This corresponds to a conventional I-element or integrator. The primary advantage of the ***I*** controller is that it generally succeeds in getting the remaining error signal to almost disappear. Through the upward integration of the error signal an increasing manipulated variable arises even for low values of e(t). This ultimately makes the error signal correspond exactly to zero.



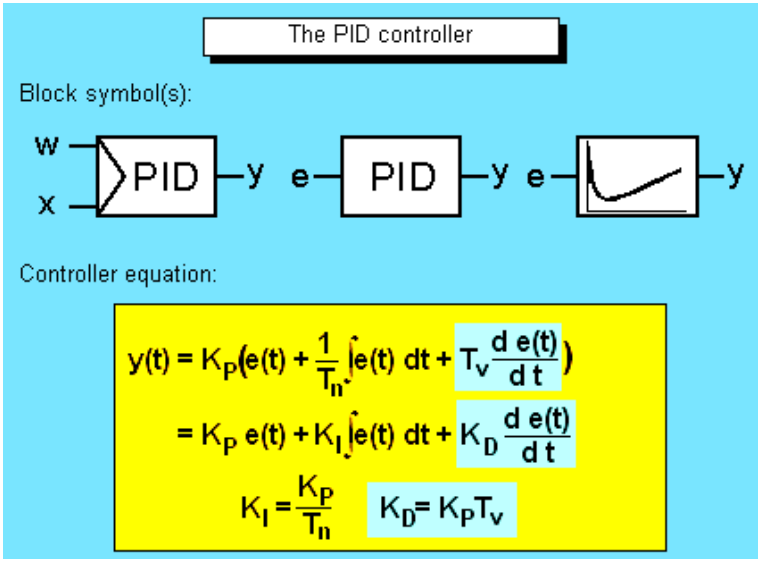
PI control
PI controllers

The parallel connection of a P and I element constitutes a PI controller. It combines the advantages of the P controller (speed) and the I controller (accuracy). The two parameters of this controller are the proportional-action coefficient KP and the reset time Tn or integral-action coefficient KI.



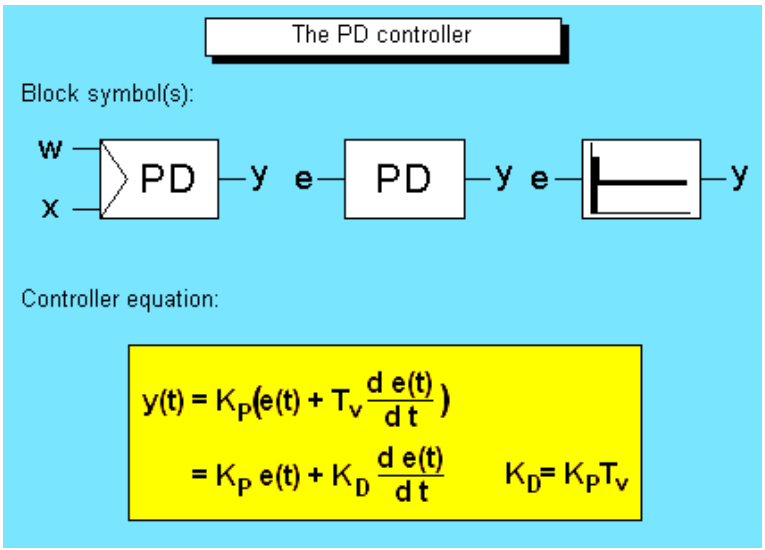
PID and PD control
PID controllers

To increase the speed of the PI controller a derivative-action element (D element) can be added; this produces a PID controller. The D element generates a manipulated variable component yD, which is proportional to the dynamic change of the error signal and is thus particularly effective for changes in the reference variable. The strength of the D component is set via the rate time Tv or the derivative-action coefficient KD.



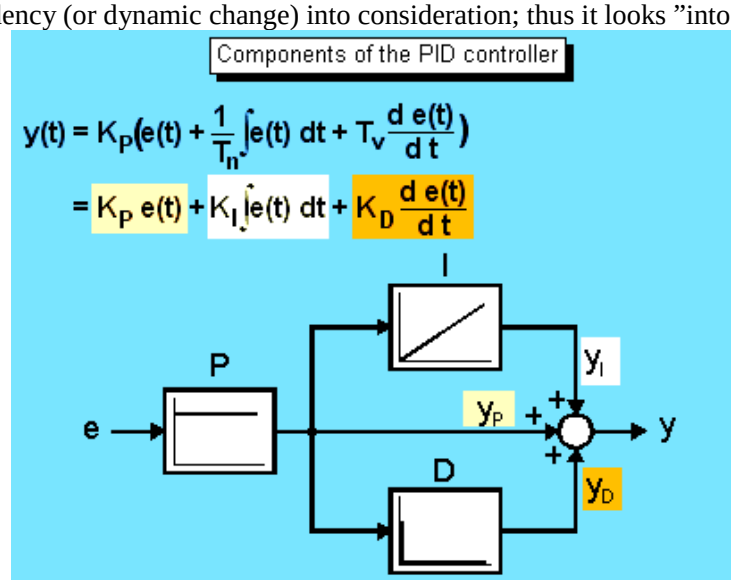
PD controller

In some cases the I-component of the PID controller can be omitted; this in effect produces the PD controller. For example, this is the case when the elimination of the remaining error signal is not necessarily required or the controlled system itself is already equipped with an integral-action component (I-element), say an integrating actuator (like a stepping motor). In the PD controller the D-element is normally designed with time lag.



Summary

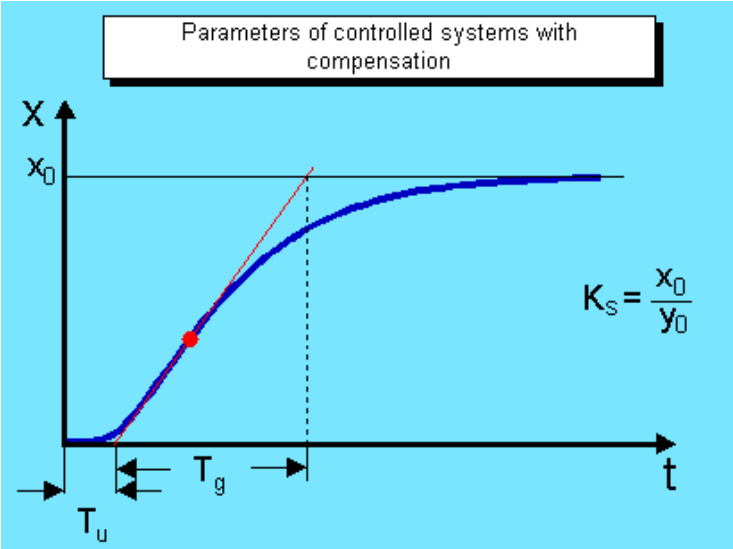
The PID controller is a universal controller with three individually connectable components configured in accordance with the controlled system. The P component evaluates the actual error signal (it ”perceives” the present) and thus provides for generally good control response. The I component evaluates the characteristic lagging behind in time (i.e. it perceives the ”past”), finally, the D component takes the error signal’s tendency (or dynamic change) into consideration; thus it looks ”into the future".



Controlled systems with compensation

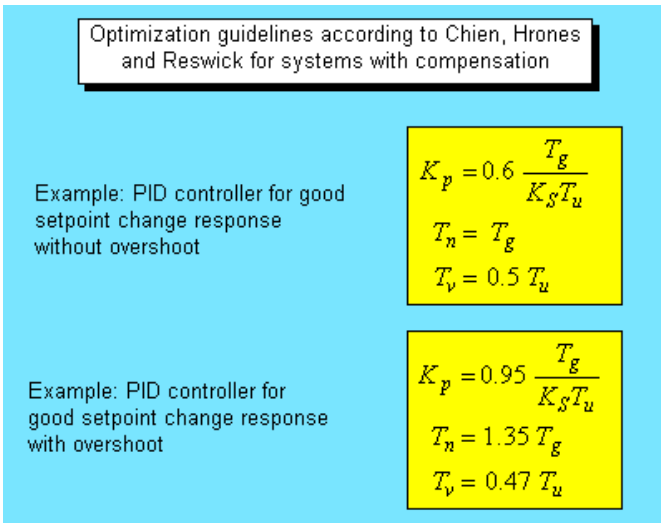
Suitable controller parameters K_P, Tn and Tv depend on the characteristic of the controlled system (system dynamics). Also, in particular for systems with compensation, there exists a whole series of adjustment or optimization procedures, which normally

refer to the controlled system parameters K_S (proportional-action coefficient of the system), Tu (delay time) and Tg (compensation time). The most famous optimization guidelines were developed by Chien, Hrones and Reswick.



Optimization guidelines according to Chien, Hrones and Reswick

When using the optimization guidelines according to Chien, Hrones and Reswick there is the possibility to make adjustments to set the desired control-loop response. For example you can emphasize good response to setpoint change or good response to disturbances. It is also possible to preset the desired swing tendency of the control loop (with/without overshoot of the controlled variable).



Procedure:

1. Assemble in accordance with the plug-in diagram.
2. Adjust the setpoint integrator of the Servo setpoint potentiometer (734-10) to ∞, S_1 ➔ OFF.
3. Adjust the setpoint value to 0° (360°), and the output voltage to 0.0V with the "zero" adjustor.
4. Set the setpoint value to 90°, and the output voltage to 5.0V with the "scale" adjustor.
5. Adjust Kp associated with the DC-Servo (734-44) to unity by comparing the input signal to the output signal.
6. Measure and record Vr and Vs For the open loop system.
7. Close the loop using a P controller with a value of unity. Measure Vr, Vs, and Ve (as defined in figure 1). Explain the results.
8. Start increasing the gain and observing its effect on Ve. Comment on your results.
9. Now switch on the Integral element (switch 5) with P controller set to unity. Observe the effect of different values of the correction time T_N on Ve (Instead we can take the effect on Vo for a sudden change in Viin --- how fast the response will be. Note Tg in previous figure). Comment on your results.

Simulink Exercise:

Use SIMULINK to simulate the behavior of the motor speed control system shown in Fig. 2, with a PI controller. (Assume K_t = 1 volt/rad/sec). [set K_p = 0, 1, 4; and let K_I = 0, 1, 2, 5,8]. Describe the effect of the integral control action. [using a unit-step reference input and no disturbance].

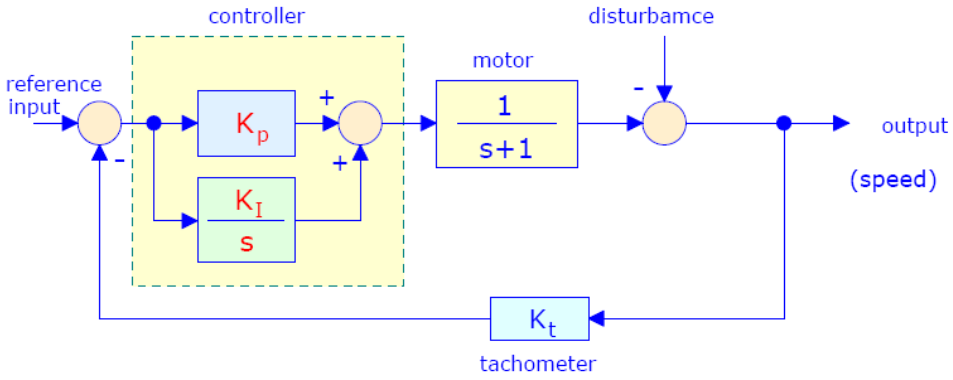
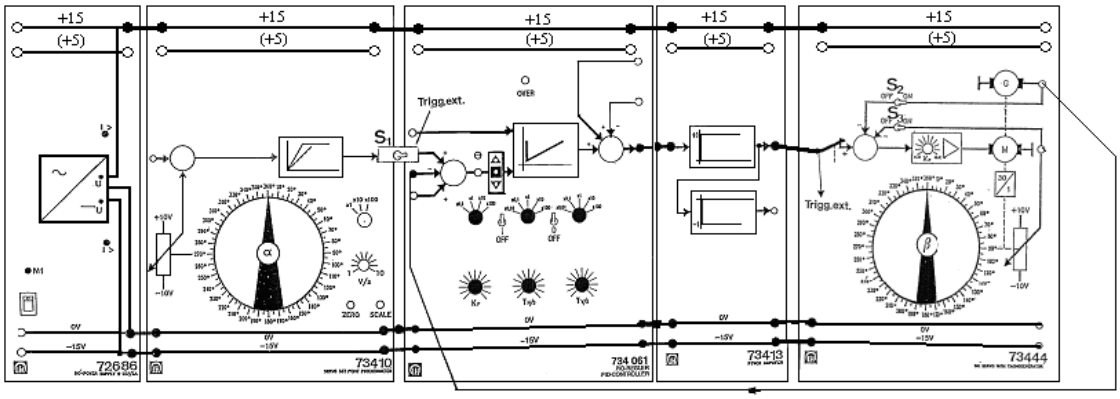


Figure 2: Speed control system with PI controller

Plug-in diagram



University of Hail

Electrical Engineering Department

EE380 Control Engineering Exp. 8

Controller Design Using Root-Locus Method

Objectives:

- To develop an understanding of the controller design using the Root-Locus method.

Apparatus:

- MATLAB

Introduction:

The root-locus method is, as generally known, a graphical method, which is used to analyze the position of the closed-loop poles. This method offers the possibility to combine in the complex ***s***-plane the desired dominant pair of poles with the root locus of the fixed part of the loop and to deform the root locus by adding poles and zeros such that two of the branches traverse through the desired dominant pair of poles at a certain gain K_o. Figure 1 shows, how the root locus can be deformed to the right by adding a pole and to the left by adding a zero.

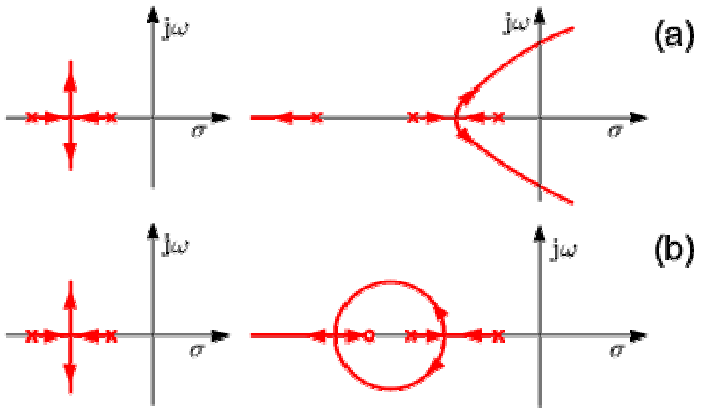


Figure 1: Deforming of the root locus (a) to the right by an additional pole, (b) to the left by an additional zero in the open loop

The principal strategy during the controller design by the root-locus method can be summarized for Phase-lead controller (compensator). A compensator is an additional

component or circuit that is inserted into a control system to compensate for a deficient performance.

Consider the first-order compensator with the transfer function

$$G_C(s) = k_C \frac{\frac{s}{z}+1}{\frac{s}{p}+1} = \frac{p}{z} k_C \frac{s+z}{s+p} = k_C' \frac{s+z}{s+p}$$

The design problem then becomes the selection of ***z***, ***p***, and $\boldsymbol{K_c}$ in order to provide a suitable performance. When $|z| < |p|$, the network is called a phase-lead network.

The following steps show the procedure for phase-lead design using the Root-Locus;

1. First locate the desired dominant root locations so that the dominant roots satisfy the specifications in terms of ξ and ω_n.
2. Sketch the root locus of the uncompensated system and determine wither the desired root locations can be realized with an uncompensated system.
3. If a compensator is necessary, place the zero of the phase-lead compensator directly below the desired location (to the left of the first two real poles).
4. Determine the pole location so that the total angle at the desired root location is 180^o and therefore is on the compensated root locus.
5. Evaluate the total system gain at the desired root location and then calculate the error constant.
6. Repeat the steps if the error constant is not satisfactory.

For more about this topic and how to design a Phase-Lag compensator, refer to **chapter 10 of your textbook**.

Design Example:

Given is a plant described by the transfer function

$$G_P(s) = \frac{1}{s(s+3)(s+5)} \quad (1)$$

For this plant a controller must be designed, such that the step response *y(t)* of the closed loop shows the following properties:

$$Mp = 16\% \qquad \text{and} \qquad t_{r,50} = 0.6\,\text{sec}$$

Where $t_{r,50}$ is defined as shown in figure 7. First, these specifications will be transformed into ξ and ω_n ($\omega_n \equiv \omega_o$) from Figures 2 and 3 below

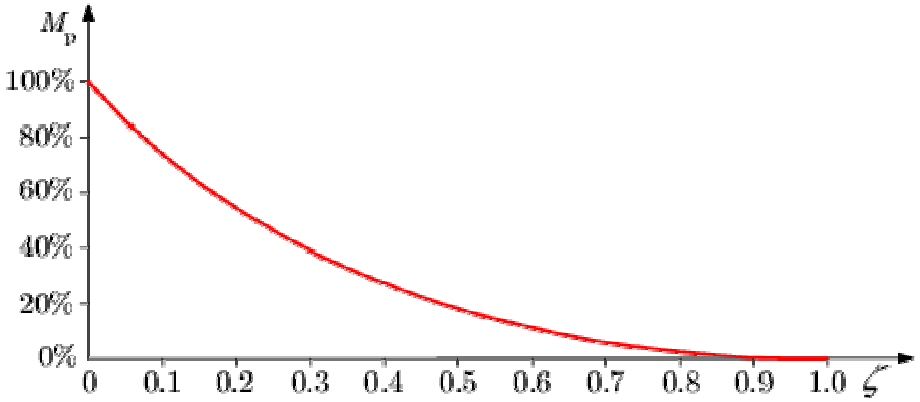


Figure 2: Maximum overshoot (in %) as function of the damping ratio.

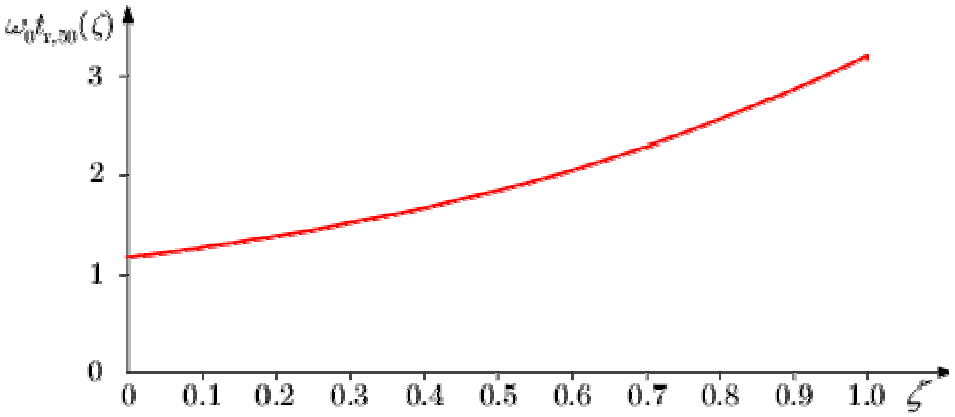


Figure 3: The product $\omega_n t_{r,50}$ (normalized rise time) as a function of the damping ratio

$$\zeta \geq 0.5 \qquad\qquad \omega_0 \geq \frac{1.85}{0.6s} = 3.1 s^{-1}.$$

In order to have a geometrical interpretation one should consider Figure 4, where a pair of complex poles

$$s_{a,b} = -\zeta\omega_0 \pm j\omega_0\sqrt{1-\zeta^2}$$

is shown. The distance d^* of both poles $s_{a,b}$ from the origin is

$$d^* = \sqrt{\omega_0^2\zeta^2 + (1-\zeta^2)\omega_0^2} = \omega_0 .$$

The angle α is

$$\cos\alpha = \frac{\omega_0\zeta}{\omega_0} = \zeta$$

where for the current case of $\xi \geq 0.5$ the condition $\alpha \leq \cos^{-1}(0.5) = 60^{o}$ is met. The damping ratio ξ

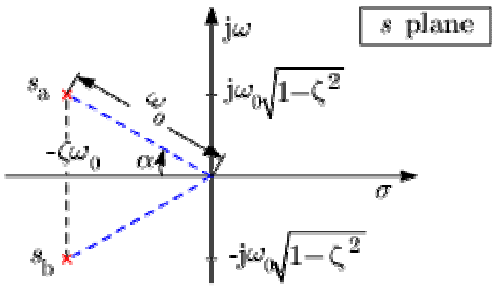


Figure 4: Pair of conjugate complex poles in the s plane

During the design one will try to increase the damping ratio not unnecessarily high as this step causes an increase in the rise time for a given natural frequency (Figure 3). Increasing the natural frequency implies an increase of the speed of the control loop. But this parameter should not be unnecessarily increased, otherwise the dominance of the pair of poles may be lost. Figure 5 shows the root locus of the closed loop using a P controller.

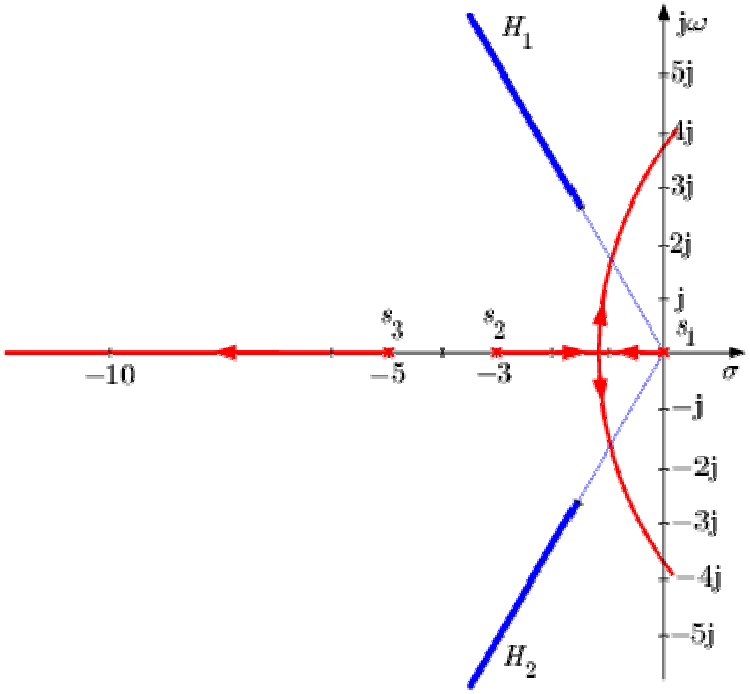


Figure 5: Root locus of plant with P controller and potential positions of the dominant pairs of poles (thick lines)

Potential positions for the dominant pair of poles are drawn by the two thick lines H_1 and H_2. It is obvious that the design using a pure P controller (changes in the gain K_o) does not lead to the goal, as the root locus does not traverse the two lines H_1 and H_2. Equally it is clear which steps have to be taken, such that the two branches of the root locus under consideration traverse the two lines H_1 and H_2. If the two poles s_2 = -3 and s_3 = -5 are shifted further left, σ_b will move left and with it the total curve without changing the structure of the system. One possibility is to perform this shift by a simple lead element which compensates the pole s_2 = -3 by a zero and a pole s_4 = -10. The resulting controller transfer function is

$$G_C(s) = K_C \frac{1 + s/3}{1 + s/10} = 3.33 K_C \frac{s+3}{s+10} \tag{2}$$

and the transfer function of the modified open loop

$$G_o(s) = 3.33 K_C \frac{1}{s(s+10)(s+5)} \,. \tag{3}$$

The root locus of the closed loop is shown in Figure 6.

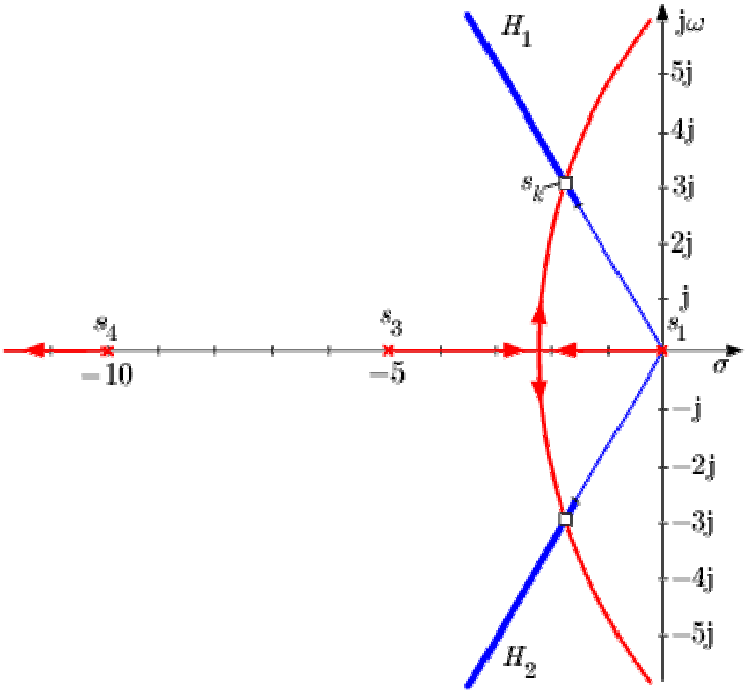


Figure 6: Root locus of the compensated system with modified controller according to Eq. (3)

It traverses the line H_1at s_k. The associated gain at this point can be determined via the magnitude requirement.

The step response of the closed loop in Figure 7 shows that the specifications are achieved.

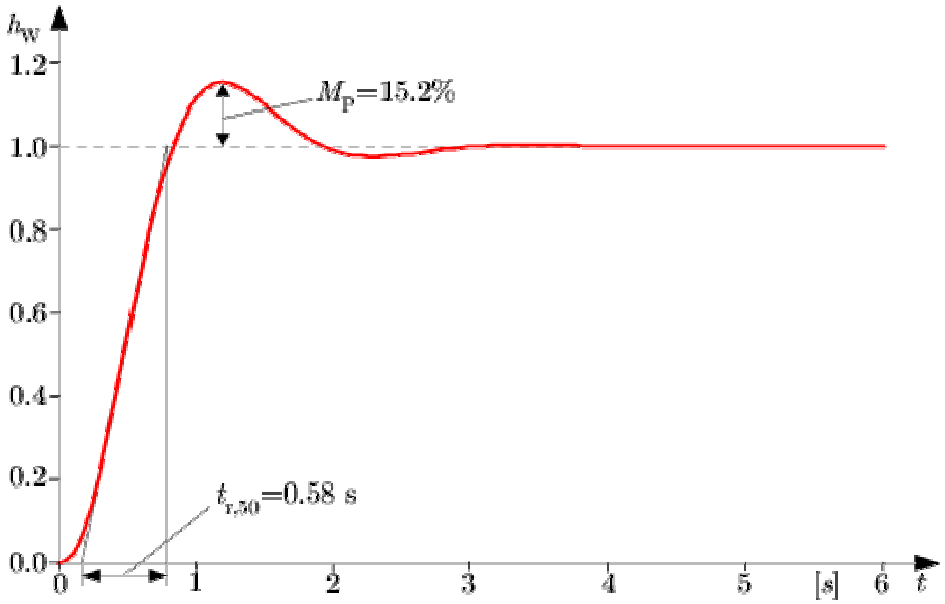


Figure 7: Closed-loop step response with modified controller according to Eq. 3

Simulink Exercise:

- Write a Matlab m-file to get the result of the example above.
- A unity feedback control system has a loop transfer function

$$L(s) = \frac{K}{s(s+2)}$$

We want the damping ratio of the dominant roots of the system to be $\zeta = 0.45$, the settling time Ts = 1 sec. and the velocity error constant to be equal to 20.

1. Find the gain (K) of the uncompensated system to satisfy the error constant requirement.
2. Find the roots of the uncompensated system. Does the system satisfy the damping ratio requirement?
3. Draw the Root-locus of the uncompensated system.
4. Design a lead compensator to satisfy the system requirements.

- Work out DP10.1 from the textbook using a lead-compensator. Show all details.

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
Electrical Engineering Department

EE 380 - CONTROL ENGINEERING **Experiment # 10**

CASCADE COMPENSATION VIA FREQUENCY DOMAIN TECHNIQUES

OBJECTIVES

To develop an understanding of the lag and lead compensators and to use the frequency domain methods in designing such compensators.

INTRODUCTION

Control systems are designed to perform specific tasks. The requirements imposed upon the control system are usually spelled out as performance specifications. They generally relate to accuracy, relative stability, and speed of response.

Setting the gain is the first step in adjusting the system for satisfactory performance. In many practical cases, however, the adjustment of the gain alone may not provide sufficient alteration of the system behavior to meet the performance specifications. In general, gain adjustment can improve the steady-state behavior, but the system will tend to be oscillatory or even unstable. It is then necessary to insert additional components in order to alter the overall behavior so that the system will behave as desired. An additional device inserted into the system for such purpose is called a **COMPENSATOR**. This device compensates for deficient performance of the original system. The four commonly employed methods of compensation are (a) cascade or series compensation, (b) feedback compensation, (c) output or load compensation, and (d) input compensation. These schemes are illustrated in Fig. 1.

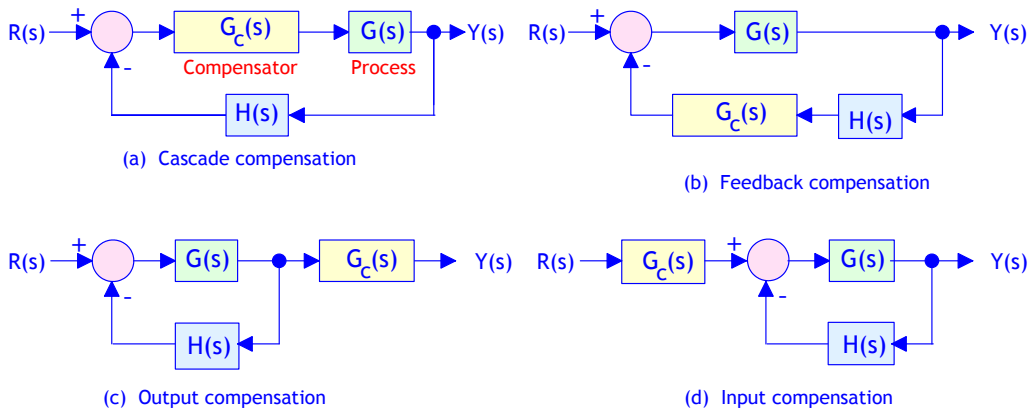


Fig. 1 Types of compensation

As indicated in Fig. 1a, a cascaded compensation consists of placing elements in series with the forward loop transfer function. such compensation may be classified into the following categories:

(1) **Phase-lag** compensation
(2) **Phase-lead** compensation
(3) Lag-lead compensation.
(4) Compensation by cancellation.

In this experiment, the design of the phase-lag and the phase-lead compensators is considered.

THEORY:

Part I : Phase-lag compensation

When the output of an element lags the input in phase and the magnitude decreases as a function of frequency, the element is called a phase-lag element. Consider the phase-lag network of Fig. 2.

The transfer function is

$$G_c(s) = \frac{V_o(s)}{V_{in}(s)} = \frac{1+\tau s}{1+\alpha\tau s} = \frac{1}{\alpha}\frac{(s+z)}{(s+p)}$$

$$\tau = R_2 C;\ \alpha = \frac{R_1+R_2}{R2} \quad (>\mathbf{1})$$

$$z = \frac{1}{\tau},\ p = \frac{1}{\alpha\tau}$$

The bode diagram and the pole-zero diagram of the phase-lag network are shown in Figs. 3 and 4 Respectively.

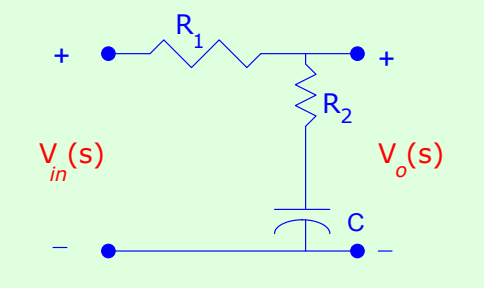


Fig. 2 Phase-lag network

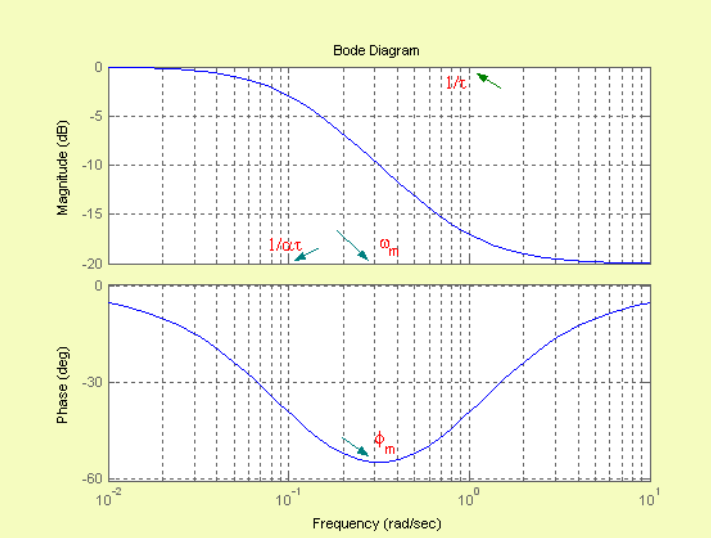


Fig. 3 Bode diagram of the Phase-lag network

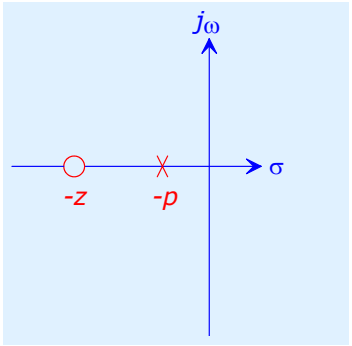


Fig. 4 Pole-zero diagram of phase-lag network.

- The lag compensator has **unity** low frequency gain and an attenuation of 'α' at high frequencies.
- By placing the compensator in cascade with the plant, the low frequency performance will be unaffected but the overall gain of the system will be reduced in the gain-crossover region to improve the stability margins. The amount of attenuation required to meet the design requirement determines the value of 'α' and the only other decision that has to be made is the actual value of the pole and zero of the compensator (i.e. τ).
- The lag compensator introduces a small phase lag at very low frequencies and at very high frequencies, and a much larger phase lag between the two corner frequencies. The obvious effect is that the **new** gain-crossover frequency is **lower** than the **original** gain-crossover frequency.
- By locating the corner frequencies of the compensator much lower than the gain cross-over frequency, the amount of phase lag introduced, **where attenuation is needed**, is small. A typical design figure is for the compensator to introduce less than 5° phase lag at the design point. This figure translates approximately into placing the upper corner frequency , $\frac{1}{\alpha\tau}$, of the compensator at **one tenth** of the new gain cross-over frequency .
-

Design steps:

1) Determine the open-loop gain such that the requirement on the particular error coefficient is satisfied.

2) Using the gain thus determined, draw the Bode diagram of the uncompensated system and determine the phase and gain margin. [MATLAB is very handy here].

3) If the specifications are not satisfied, then find the frequency point where the phase angle of the open-loop transfer function is $-180^\circ +$ phase margin $+5^\circ$. Choose this frequency as the **new gain cross-over frequency.**

4) Determine the attenuation of the system at the new cross-over frequency . This attenuation is equal to $--20\log_{10}\alpha$. Calculate α.

5) Choose the corner frequency $\frac{1}{\tau}$ (corresponding to the zero of the lag network) one decade below the new gain cross-over frequency. Thus the other corner frequency $\frac{1}{\alpha\tau}$ (corresponding to the pole of the lag network) is determined.

Part II : Phase-lead compensation

When the output of an element leads the input in phase and the magnitude increases as a function of frequency, the element is called a phase-lead element. Consider the phase-lead network of Fig. 5. The transfer function is:

$$G_c(s) = \frac{1}{\alpha}\,\frac{1+\alpha\tau s}{1+\tau s} = \frac{(s+z)}{(s+p)},\ \tau = \frac{R_1R_2}{R_1+R_2}C;\ \ \alpha = \frac{R_1+R_2}{R_2};\ (>\mathbf{1})$$

Note that the gain of forward loop transfer function is increased to offset the effect of α.

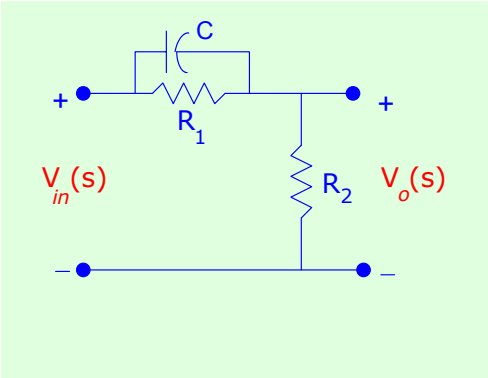


Fig. 5 Phase-lead network

The bode diagram and the pole-zero diagram of the phase-lead network are shown in Figs. 6 and 7 Respectively.

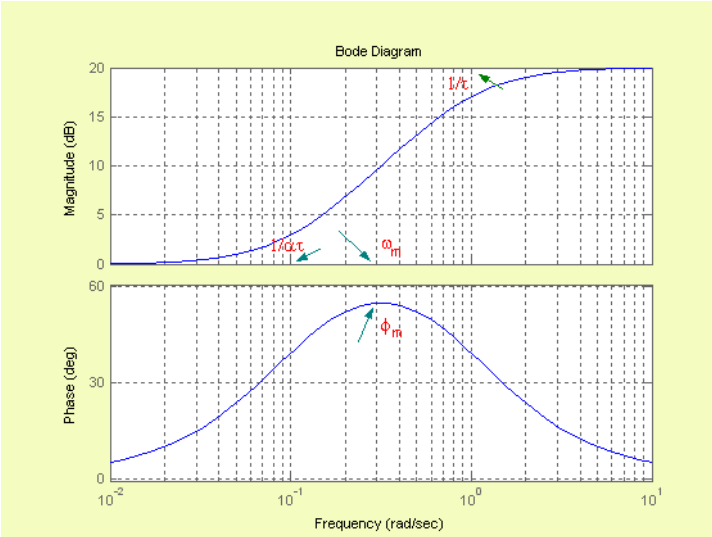


Fig. 6 Bode diagram of the Phase-lead network

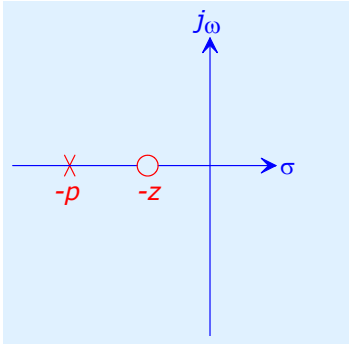


Fig. 7 Pole-zero diagram of phase-lead network

The maximum phase lead ϕ_m occurs at ω_m. It may be shown that :

$$\omega_m = \sqrt{zp} = \frac{1}{\tau\sqrt{\alpha}}\ ;\quad \sin(\varphi_m) = \frac{\alpha - 1}{\alpha + 1}\ ;\quad [\ \text{Also}\quad \alpha = \frac{1+\sin(\varphi_m)}{1-\sin(\varphi_m)}\ ]$$

Note that the amount of modification in the magnitude curve at $\omega_m = \frac{1}{\sqrt{\alpha}\,\tau}$ is $\sqrt{\alpha}$.

In a lag compensator, the **amount of attenuation** it offers is of primary interest and the phase lag it produces is discounted by centering the compensator well away from the design point. With a lead compensator,

however, its **phase characteristic** is what the designer is after and its **magnitude characteristic** is a nuisance that he cannot wish away.

Design steps:

1. Determine the open-loop gain such that the requirement on the particular error coefficient is satisfied.
2. Using the gain thus determined, draw the Bode diagram of the uncompensated system and determine the phase margin.
3. Determine the necessary phase lead angle ϕ to be added to the system. Allow for a small amount of safety. Then determine φ_m.
4. Determine the value of $\alpha = \dfrac{1+\sin(\phi_m)}{1-\sin(\phi_m)}$.
5. Determine the frequency where the magnitude of the uncompensated system is equal to $20\log_{10}\sqrt{\alpha}$. Select this frequency as the new gain cross-over frequency. This frequency corresponds to $\omega_m = \dfrac{1}{\sqrt{\alpha}\,\tau}$ and the maximum phase shift φ_m occurs at this frequency.
6. Determine the corner frequencies $\frac{1}{\alpha\tau}$ (corresponding to the pole of the lead compensator) and $\frac{1}{\tau}$ (corresponding to the zero of the lead compensator).
7. Finally , insert an amplifier with gain equal to α .

DESIGN DRILL:

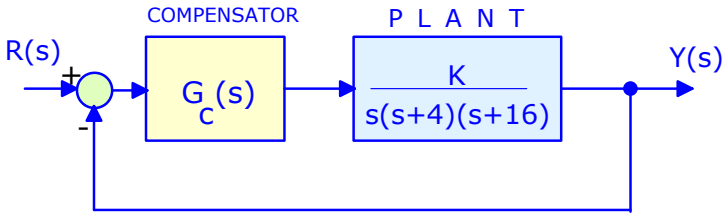
Consider a position servo whose transfer function is :

$$G_p(s) = \frac{K}{s(s+4)(s+16)}$$

Specifications:

- Velocity error coefficient $K_v = 10\,\text{sec}^{-1}$.
- Phase margin of 45°.

1. Design a **lag compensator** to satisfy the specifications.
2. Design a **lead compensator** to satisfy the specifications.



REPORT

1. Show all the design steps for the phase-lag compensator.
2. Show and comment on the bode diagram of the uncompensated system, the compensator, and the compensated systems.
3. Design a circuit to implement the phase-lag compensator.
4. Draw the root locus of the uncompensated and the compensated systems. Discuss the results.
5. Calculate the bandwidth of both the uncompensated and the compensated systems. What do you observe?
6. Provide the step-response for both the uncompensated and the compensated systems. Comment on your results.
7. Repeat steps 1 to 6 for the phase-lead compensator.