CONTROL SYSTEM LABORATORY EXPERIMENTS

ECE 486

Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

Table of Contents

Lab Session 0: Introduction to Matlab and Equipment **3**

Lab Session 1: Simulation Using the Analog Computer **8**

Lab Session 2: Digital Simulation **12**

Lab Session 3: Digital Simulation of a Closed Loop System **18**

Lab Session 4: Introduction to the DC Motor **21**

Lab Session 5: PD Control: Analog Computer & C2000 DAQ Board **33**

Lab Session 6: Lead Controller Design **45**

Appendix A: The Analog Computer **56**

Appendix B: The GP-6 Analog Computer **70**
B.1 The GP-6 Analog Computer: Characteristics and Operating Instructions . . 70
B.2 Analog Computer's Manual - GP-6: Patch Panel 71
B.3 Patch Panel Operations 73
B.4 Scaling Review 84
B.5 Full GP-6 Patch Panel Scan 86

Appendix C: Sections from Feedback Control of Dynamic Systems **87**
C.1 Obtaining Models from Experimental Data 90
C.1.1 Models from Transient-Response Data 91
C.1.2 Models from Other Data 95
C.2 A Case Study of Speed Control 96
C.2.1 Disturbance Rejection 98
C.2.2 Sensitivity of System Gain to Parameter Changes 101

Appendix D: College Control Laboratory Instrumentation **102**
D.1 Laboratory Instrumentation Overview 102
D.2 Digital Multimeter 104
D.3 HP 54600 Digital Two-Channel Oscilloscope 109
D.4 Function Generator 112
D.5 Dynamic Spectrum Analyzer 117

Lab Session 0: Introduction to Matlab and Equipment

This exercise is intended to give you a brief introduction to MATLAB, the computation software package you will be using throughout the semester in homework and laboratory exercises, and also an introduction to the laboratory equipment.

Laboratory Policies and Safety

Follow these policies whenever you work in the Control Systems Laboratory.

- **No food or drink is permitted in the laboratory.** Food and drinks must be left outside the laboratory work area.
- **Do not remove laboratory equipment or manuals.** All equipment, cables, accessories, and manuals must remain in the laboratory.
- **Do not hot-wire equipment.** Before connecting, disconnecting, or changing any wiring, turn off the affected equipment. After verifying the wiring, turn on only the equipment needed for the experiment.
- **Handle banana-plug cables by their connectors.** Do not remove a banana plug by pulling on its wire. Grip the connector and gently pull it from the jack.
- **Report equipment or software problems.** If you find damaged equipment, malfunctioning equipment, or a software problem, stop using the affected item and contact a TA.
- **Use the buddy system outside scheduled laboratory hours.** At least two people must be present in the laboratory when working outside scheduled laboratory sessions.

Station Shutdown Procedure

Before leaving the laboratory, complete the following steps:

1. Turn off all equipment used during the laboratory session.
2. Disconnect all wires and cables used during the session.
3. Store all wires, cables, and accessories neatly in their designated locations.
4. Log off the laboratory computer.

Introduction to Matlab

These exercises are just to get you familiar with MATLAB; other commands will be introduced as needed. The documents section of the course website has links to several tutorials: `http://courses.engr.illinois.edu/ece486/`. There are helpful control system tutorials for MATLAB at `http://www.engin.umich.edu/class/ctms/`.

As needed, use the resources available, including MATLAB's help facility (try either `helpwin` or `help <command>` or `doc <command>`), for any **boldfaced** words in this exercise.

1. First, create a folder on the **C:** drive of the computer to use to save your files and data throughout the semester. Navigate to `C:\matlab\ECE486` (create folders as necessary) and make a folder for your bench. Name it your NetIDs separated by an underscore, e.g. `NetID1_NetID2`. Within this folder, you should make folders for each lab you perform this semester. Make a `Lab0` folder now to save any files you use in this lab.

2. Using **tf**, construct the transfer function:

$$\frac{1}{(s+1)(s^2+0.5s+3)} \tag{0.1}$$

   Plot the **step** response, add a **grid**, and use the zoom tool in the plot GUI to find the steady-state value. Then zoom out to the original axis scale.

3. Type **edit** to create a new script file (**.m file*). M-files allow you to save a sequence of MATLAB commands to run later. The F5 key or  icon (in MATLAB R2013a or newer) can be used to run these files. In addition, a **function** may be defined in its own .m files (you may explore differences between script file vs. function file in the documentation); this function may be used just like the built-in functions. The commands **cd** ("change directory") together with **pwd** ("print working directory") and **path** (or graphically click  under HOME tab in MATLAB toolbar to check current "path") may be needed to tell MATLAB where your functions are.

4. At the top of this file, add a comment (a line starting with `%`) with your name and this lab number. Now **clear** all previously defined variables, clear the command window (**clc**), and **close all** previous plots.

5. Add another comment with the steady-state value found in the first part.

6. Re-create the transfer function in the m-file. This time, put the **step** response results into the variables y and t. Compute the peak value of the step response using **max**. **Plot** y and t.

7. Use **colon** to generate a set of times t_2 covering the range of the step response, with 0.1s spacing. Put a semicolon at the end of the line to suppress the printing of this array.

8. Compute `z = 0.33 * (1 - exp(-0.5 * t2))`. Set **hold on** for the previous plot and overlay a plot of z versus t_2, this time using the dotted (':') line style.

9. Label the traces with **legend**, and the axes with **xlabel** and **ylabel**. Put your name as the **title** of the figure. Also put the Greek characters α, ζ, π, and ω in the title (use LaTeX format, i.e. `` `\alpha',`\zeta', ... ``). **Print** the figure in landscape **orient**ation.

10. Print out your m-file and turn it in, along with the last plot (one per group). **TA check:** Please stop and wait for the TA to check your work.

Laboratory Exercise

Brief descriptions of the equipment used in this lab are given, followed by a simple exercise for you to work with them.

I. Equipment Descriptions

- Comdyna GP-6 [Analog Computer] (on table):
  Described in Appendices A and B.
- Patch panel:
  Allows for easy access to:
  - computer DAQ boards
  - HP power supply (described below)
  - internal voltage sources
  - internal power amplifier for DC motor
- DC motor (on table):
  Basic brushed DC motor with an internal *tachometer* [tach] (measures angular velocity) attached, and you can also attach a *potentiometer* [pot] (measures angular position). There are 5 banana jacks integrated with the motor mount: (see Figure 0.1)
  - Orange: Tach +
  - Gray: Tach -
  - White: Motor chassis
  - Black: Motor -
  - Red: Motor +
- Computer:
  Besides MATLAB, the computer has hardware and software for collecting and transmitting analog input and output data by using an internal Data Acquisition [DAQ] board. The DAQ is connected via a ribbon cable to the patch panel described above.
- Hewlett-Packard DC power supply [HP power supply]:
  A well-regulated 100W power supply, internally connected to patch panel.
- Agilent oscilloscope [scope]:
  A 2-channel oscilloscope that can display and perform measurements on DC (steady) and AC (time-varying) signals. It has lots of functionality; we will only use some of it. Connections are made using coaxial cables with a BNC connector (looks like your cable TV hookup) on one end.

- HP digital multimeter [DMM or DVM]:
  This works basically the same as handheld multimeters. Using two cables with banana connector ends, you can measure the voltage, current, continuity, frequency or resistance in circuits.
- HP function/arbitrary waveform generator [signal generator]:
  Generates periodic signals in a variety of shapes: sinusoids, square waves, triangular waves, sawtooth waves. You must specify the frequency, amplitude (peak-to-peak), and DC offset (how much the signal will be shifted up).

II. Hardware Exercise: Open-loop Motor Velocity Control

Wire up the system as follows:

1. Connect the signal generator output to the amplifier input on the patch panel.
2. Connect the "Ref -" (black) and "Sig Gnd" (white) jacks to each other on the amplifier input on the patch panel.
3. Connect the amplifier output to the motor (use the gray 3-plug cable and follow color codes).
4. Connect the pot to the motor by loosening the table-mount thumbscrews and pushing the joints together. Then re-tighten the thumbscrews.
5. Apply power to the pot by connecting it to the $+5V$ supply on the patch panel. Connect **HI** to the bottom jack (black) on the pot, and **LO** to the top jack (black also) on the pot.
6. Connect the pot outputs to the scope Channel 1. To do this, connect the red plug to the middle (white) jack and the black plug to the top (black) jack.
7. Connect the tach outputs to the scope Channel 2. To do this, **Tach** + (orange, on the motor mount) should receive the red plug, and **Tach** - (grey, also on the motor mount) should receive the black plug.

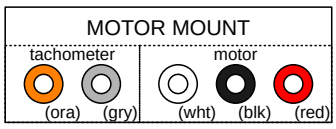


Figure 0.1: Banana jacks on motor mount.

Now you have a reference input (signal generator), plant (amplifier and motor), sensors (pot and tach), and viewable output (scope screen). This setup is called an "open-loop" configuration, since the pot and tach data is not used to modify the input to the motor.

1. First, power on the patch panel (flip the switch) and the scope. Rotate the motor by hand and adjust the scope display settings until you get a rolling view of the pot data (Channel 1). (To get the scope into **ROLL** mode, press the Horizontal *Main/Delayed*

button and select *Roll*). Also put the oscilloscope in `High Resolution` acquire mode by clicking the **Acquire** and changing the mode accordingly.

2. Describe what you see. What is the output when the motor is still? When it is spinning? How does it change when the pot is powered by the $+10V$ supply instead? Try it. **TA check:** Please stop and wait for the TA to check your work.

3. Now we will apply power to the motor. Turn on the signal generator and pick a signal type, amplitude, and offset. **Always keep the voltage below $\pm$ 5V to avoid burning out the motor**. Keep the frequency below 10 Hz or you will not be able to notice its effect. Power on the motor (flip the switch in the Amplifier part of the patch panel), and press the black button to run the motor. Look at both the pot data (Channel 1) and the tach data (Channel 2) on the scope in "roll" mode. Note that you can shift and scale the channels independently of each other.

4. Describe what you see. What kind of output does the tach provide? Why do you see a sawtooth from the pot, even with a DC signal? On the signal generator, play with different input waveforms, amplitudes, and offsets. (If you're having trouble seeing the data on the scope, you can freeze the data by pressing the *Run/Stop* button.) **TA check:** Please stop and wait for the TA to check your work.

5. Finally, explore a major problem with open-loop velocity control. Ask the TA to connect a magnetic break to the system. What is the problem? How would you fix it?

If you wish to backup your data or use it outside of lab, you may save files to the U: drive. This drive maps to your Engineering Workstation (EWS). It is your responsibility to back this up at the end of each lab.

When you leave the lab each day, remember to *log off* from the PC and *shut down* the equipment, and *then shut off* the bench power. Also *clean up* the area and put away wires, etc.

Report

Download and complete the report template available on the course lab website, ensuring you answer all questions.

Lab Session 1: Simulation Using the Analog Computer

The objective of this experiment is to explore second order systems. We will use an analog computer to simulate a basic second-order dynamic system. A thorough understanding of the response of such systems is essential for successful control system design.

Preparation

Readings:

▷ Appendix A: The Analog Computer: Theory and Practice;

▷ Appendix B: The GP-6 Analog Computer;

▷ G. F. Franklin, *et al.*, *Feedback Control of Dynamic Systems*, 7$^{\text{th}}$ or 6$^{\text{th}}$ Ed., Sec. 3.8, 7.4.1.

▷ The PC will control the test instrument (an oscilloscope) through the software package VEE. Your TA will explain how to use VEE. The PC will also be used for data logging and processing. By default, the data will be stored on your **U:\ drive**.

Prelab:

Using the symbols and instruction given in Appendix A, prepare an analog computer block diagram for the transfer function:

$$H(s) = \frac{Y(s)}{U(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}. \tag{1.1}$$

We will simulate this on the analog computer using *normalized time*. This is a common technique for analog simulations; it allows us to speed up extremely slow processes or slow down those which complete in a flash. This allows us to perform useful analysis in the lab.

Steps:

1. Write the differential equation for the transfer function.
2. Convert the differential equation to normalized time using the transformation: $\frac{\tau}{\omega_n} = t$ as shown in the Appendix and Franklin, *et al.*, Sec 3.8.2, 7$^{\text{th}}$ Ed. (6$^{\text{th}}$ Ed., Sec 3.8.2; 5$^{\text{th}}$ Ed., Sec 3.6.23; 3$^{\text{rd}}$ or 4$^{\text{th}}$ Ed., Sec 2.6.4).
   - Express $\frac{\mathrm{d}y}{\mathrm{d}t}$ and $\frac{\mathrm{d}^2y}{\mathrm{d}t^2}$ in terms of $\frac{\mathrm{d}y}{\mathrm{d}\tau}$ and $\frac{\mathrm{d}^2y}{\mathrm{d}\tau^2}$.
   - Now convert your differential equation to one in terms of τ.
3. Solve for the highest derivative and integrate to solve for $y(\tau)$.

4. Use the "solving for the highest derivative" method on page vii of Appendix A to create an all-integrator block diagram for the differential equation. Assume zero initial conditions and that $u(\tau)$ is a 2V unit step.

   Make sure your integrators have only gains of 1, 10, or 0.1, since these are the only possible values on the Comdyna GP-6. Also be sure to consider the *signs* of the gains. Arrange for easy adjustment of ζ from 0 to 2, using only one potentiometer (the GP-6 potentiometers have a range of 0 to 1). Using the Comdyna's "+" reference (10V), generate a 2V step for the input signal.

5. Use one of the analog computer diagram layouts at the very end of this lab manual or on the lab website to show the connections needed to implement your circuit. Note that the "-" reference terminals are actually -10V; use the two ground terminals if 0V is needed.

▷ *Do not* use the current multiplier in Figure 1.1 for multiplying voltages.

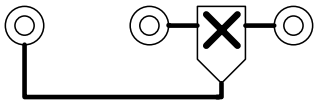


Figure 1.1: Current multiplier; do not use.

▷ Tie the output of each unused op-amp back into one of its input resistors. This causes negative feedback that reduces any random noise in the analog computer.

⚜ Laboratory Exercise ⚜

Do not turn on the analog computer until instructed to do so. Never "hot wire" the system.

First make the following connections:

1. Start VEE and load the file `N:\labs\ECE486\486Lab1.vxe` (*Note:* by default only **.vee* files are shown, you need to choose "all files" from drop-down list to see **.vxe*). When the program loads in the VEE environment the "Program Explorer" and "Properties" windows may also open. Close both of these windows to allow you to see the entire VEE program.

2. Connect wires for a second order system on the analog computer using your wiring diagram. *The appropriate wires are in the long drawer under your bench.*

3. Connect Channel 1 of the oscilloscope to y, the output of the analog computer. Connect the black lead to the analog computer ground.

4. Connect Channel 2 of the oscilloscope to any **OP** jack on the analog computer (this will be the trigger input). Connect the black lead to the analog computer ground.

   Turn on power to the Oscilloscope.

5. In Agilent VEE, verify that the Scope parameters are set to the following:

   Channel1 { V/div → 1V; Offset → 0V; Coupling → DC; Band-Width Limit → ON }
   Channel2 { V/div → 5V; Offset → 0V; Coupling → DC; Band-Width Limit → ON }

   Trigger { Level → -6.0V; Source → CHANNEL2; Slope → NEGATIVE; Noise Reject → ON; High Freq. Reject → ON }
   Time { Base → 2 S/div; Reference → LEFT }

6. Send these parameters to the scope by pressing the **Send New Scope Parameters** button in Agilent VEE. Whenever you make a change to the scope parameters you will need to press this button to send the information to the scope.

   Note: To change the path and filename of your data file in VEE, click on the *To File:* button in the To File box, type in the new path and click Ok.

7. For your first run set ζ to 2.0 by adjusting the appropriate potentiometer. To run the simulation on the analog computer first push the **IC** (initial condition) button to put the integrators in their initial state. Then the **OP** (operate) button to start the simulation. If you do not see the response starting to display on the oscilloscope the scope missed the trigger. Try again by switching back and forth between **IC** and **OP**. For your report plots, it will look nice if your responses all start at the same

point in time. So before you press the **Collect Data from Scope** button make sure you have a response that starts one time division in from the edge.

Have the TA check your data.

TA check: Please stop and wait for the TA to check your work.

8. For $H(s)$ as in the prelab, obtain step responses (zero initial conditions) for $\zeta = 2.0$, 1.5, 1.0, 0.8, 0.7, 0.5, 0.3, 0.2.

9. In MATLAB import the data by changing into the directory where the data is stored and simply typing the name of the file (without the ".m"). Each data set will be stored as a variable in order: $y1$, $y2$, etc. Each variable has two columns—the first contains the time vector, and the second the voltage at each time instant. For example, to plot the $y1$ data versus time, type `plot(y1(:,1),y1(:,2))`.

10. Display all responses (y vs. τ) on a single plot. As check points for your simulation, verify that these are close to those given in Franklin, *et al.*, Fig. 3.19(b) (Fig. 3.18(b) in 6$^{\text{th}}$ Ed., Fig. 3.16(b) in 5$^{\text{th}}$ Ed., Fig. 3.23(b) in 4$^{\text{th}}$ Ed., Fig. 3.12(b) in 3$^{\text{rd}}$ Ed.). Note that $\tau = \omega_n t$ is the scaled (normalized) time variable. *Tip:* cyclically use the '-', ':', '-.', and '–' line styles so the different traces are identifiable. Colors do not really work on the black-and-white laser printer. Print this plot for the report, and make sure everyone has a copy of the `lab1.m` file.
TA check: Please stop and wait for the TA to check your work.

Report

Download and complete the report template available on the course lab website, ensuring you answer all questions.

Lab Session 2: Digital Simulation

Digital simulation packages are an essential part of control system design. Such packages are used routinely in control system analysis and design. This lab introduces SIMULINK, the digital simulation package sold with MATLAB. SIMULINK will be used to study the dynamic response of a second order system, in both block-diagram and state-space forms, together with the effects of an additional pole and an additional zero. The characteristic responses of these systems provide important insights into control system design.

Preparation

Readings:

▷ Appendix A: See section IV-c "Representing Systems with Zeroes";

▷ G. F. Franklin, *et al.*, *Feedback Control of Dynamic Systems*, 3rd, 4th, 5th, 6th, or 7th Ed., Sec. 3.5.

Prelab

$$H_1(s) = \frac{Y(s)}{U(s)} = \frac{25}{s^2 + 6s + 25}$$

1. Develop an all-integrator block diagram for the transfer function $H_1(s)$. Do not normalize the time. What are ζ and ω_n? What are the poles of the system? Include this work in your prelab.
2. In this step, you will simulate your all-integrator block diagram using SIMULINK:
   (a) At the MATLAB command line, type `simulink`, or you can click SIMULINK library icon  under HOME tab in MATLAB toolbar to start it graphically.
   (b) Click "Blank Model." Once a blank SIMULINK file pops up, click the icon  , this will open the SIMULINK library browser window.
   (c) Open a new model by clicking on the New Model icon or by going to the File menu, under New, and selecting Model. A model window will open.
   (d) Open the Continuous library by clicking on the *Continuous* tab in the left pane of library window.
   (e) Find the *Integrator* block in the library and drag-and-drop it into the model window using the left mouse button. Or you can search with keyword of block "integrator".
   (f) Copy the *Integrator* block either by using the Copy and Paste commands or by dragging the *Integrator* within the model window with the right mouse button.

(g) Open the Math library by clicking on the *Math Operations* icon in the library window, and drag a *Gain* block and a *Sum* block into the model window.

(h) Open the *Sources* library and drag a *Step* block into the model window.

(i) Open the *Sinks* library and drag a *To Workspace* block into the model window.

(j) Copy the blocks as needed and connect the blocks so that the model resembles your all-integrator block diagram. You can change the direction of the blocks by right-clicking on the block and selecting the Format menu. To connect the blocks, drag the output port of one block (an outward-pointing arrow) to the input port of another (an inward-pointing arrow). Change the summer inputs by double-clicking on it and changing the signs in the "List of Signs" box. Experiment with the entry in this box.

Note: You can also connect two blocks in SIMULINK by clicking on the source block, holding *Ctrl*, and clicking on the target block.

(k) Find and add a *To Workspace* block to the output of the first integrator. (Optionally, you can instead use "scope" block, the settings are similar to *To Workspace*: choose data type "array" and set variable name. The variable is two-dimensional. The first column of the variable stores time, the second column response data.)

(l) Label the output of the first integrator "*y_dot*" by double-clicking on the line representing the output. Type the text in the box that appears. Click outside the text box to save the text.

(m) Change the value of the *Gain* blocks by double-clicking on the blocks, then entering the appropriate values.

(n) Double-click on the *Step* block. Set the parameters to:

▷ Step Time: 0

▷ Initial Value: 0

▷ Final Value: 1

▷ Sample Time: 0

(o) Double-click on the *To Workspace* block connected to the output. Set the parameters to:

▷ Variable name: y

▷ Save format: Array

(p) Set the parameters on the *To Workspace* block connected to *y_dot* to:

▷ Variable name: *y_dot*

▷ Save format: Array

(q) Set the simulation parameters by clicking on the Modeling menu and clicking Model Settings. In the window that pops up, ensure you have the Solver menu selected. Change the following:

▷ Stop Time: 3 (chosen to capture the main response)

▷ Type: Variable-step

▷ Solver: ode45

▷ Max Step Size: 0.001 (smaller max step sizes give smoother responses, but take longer to compute)

▷ Click on *Data Import/Export* and uncheck the *Limit data points to last* box. (*Note:* Whenever you have to deal with data export in SIMULINK always remember doing this. This option will be encountered numerous times especially in later labs.)

(r) Run the simulation by clicking on the Simulation menu and Start.

3. In the MATLAB workspace, you will find three variables: *tout*, *y*, and *y_dot*. SIMULINK always saves the time vector as `tout`. Plot the variables *y* and *y_dot* as two subplots in one figure (see **subplot**); use *tout* as the time axis for both. Label appropriately. **Print** two copies of this graph; include one with your prelab and save the other for your lab report. In addition, **print** the SIMULINK block diagram (the model window) to turn in with your prelab.

❧ Laboratory Exercise ❧

Part I: State Space Model of $H_1(s)$

1. Open a new SIMULINK model.
2. Drag in a *State-Space* block from the Continuous library.
3. In the MATLAB workspace window, set the following variables:

$$\mathbf{A} = \begin{bmatrix} -6 & -25 \\ 1 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 25 \\ 0 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

4. Double-click on the *State-Space* block, and set the **A**, **B**, **C**, **D** parameters to "A", "B", "C", "D" respectively (without the quotes). This causes SIMULINK to look up the values in the MATLAB workspace when the model is run.
5. Open the Sources library and drag a *Step* block into the model window. Connect the *Step* block to the input of the *State-Space* block, and change the parameters to those used in the prelab.
6. Open the Signals Routing library and drag a *Demux* block into the model window. Connect the *State-Space* output to the *Demux* input.
7. Open the Sinks library to find the *To Workspace* block. Connect each output of the *Demux* to a separate *To Workspace* block. Set the variable name of the top *To Workspace* block to *y_dot*, and the bottom one to *y*. Make sure both *To Workspace* blocks are set to an Array type save format (as in the prelab).
8. Set the Simulation Parameters as in the prelab, and run the simulation.
9. Make a plot containing subplots of both *y* and *y_dot*. Label and print this for your report. Set `y1=y`, `y_dot1=y_dot`, and `tout1=tout` to save these responses for later.

Part II: Effects of an extra Zero

We will now use the information from Appendix A section IV-c to introduce a zero into the all-integrator block diagram from the prelab. Sketch the block diagram by hand before implementing it in SIMULINK. This diagram will be similar to Figure 16 as shown in the appendix.

$$H_2(s) = \frac{25\left(1 + \frac{s}{\alpha\zeta}\right)}{s^2 + 10\zeta s + 25}$$

1. Modify your all-integrator block diagram so that the zero in $H_2(s)$ is added. Remove the *y_dot* "*To Workspace*" block. In the block diagram, use variables like "a" or "zeta" for α and ζ; this way you can easily set their values in the MATLAB workspace.

2. Notice that the zero is added at $s = -\alpha\zeta$, with ζ as found in the prelab. With this ζ, simulate the system with the real part of the zero in various regions. We will rename the variables after each simulation so they are not overwritten.

   $\triangleright$ ten times further left than the poles (save as $y2a$ and $tout2a$)

   $\triangleright$ on the real axis directly between the poles (save as $y2b$ and $tout2b$)

   $\triangleright$ halfway between the poles and the origin (save as $y2c$ and $tout2c$)

   $\triangleright$ in the right-half plane, with $\alpha = -3$ (save as $y2d$ and $tout2d$)

   $\triangleright$ to see the trend, also try $\alpha = -30$ (save as $y2e$ and $tout2e$)

Table 1: Student response table for the zero location and alpha value of each Part II signal.

Signal	zero location	α
$y2a$		
$y2b$		
$y2c$		
$y2d$		
$y2e$		

3. Overlay all these responses along with $y1$ on a single graph and label appropriately. Print a copy for the report. Note that in general $tout2a \neq tout2c$; since we are using variable-step integration, MATLAB is free to choose which times it evaluates. Remember this; a common mistake is to use the wrong time data when plotting.

   TA check: Please stop and wait for the TA to check your work.

Part III: Effects of an extra Pole

$$H_3(s) = \frac{25}{\left(1 + \frac{s}{\alpha\zeta}\right)(s^2 + 10\zeta s + 25)}$$

1. Remove the zero from the model in Part II and add an extra pole instead, as in $H_3(s)$. Sketch the block diagram by hand before implementing it in SIMULINK.

2. The extra pole is added at $s = -\alpha\zeta$, with ζ as found in the prelab. With this ζ, simulate the system with the real part of the added pole:

   $\triangleright$ ten times further left than the poles (save as $y3a$ and $tout3a$)

   $\triangleright$ on the real axis directly between the poles (save as $y3b$ and $tout3b$)

   $\triangleright$ halfway between the poles and the origin (save as $y3c$ and $tout3c$)

   $\triangleright$ We do not want a pole in the right half plane because that is unstable. Try it and see what happens.

Table 2: Student response table for the pole location and alpha value of each Part III signal.

Signal	pole location	α
$y3a$		
$y3b$		
$y3c$		
$y3d$		

3. Overlay all the stable extra pole responses on a single graph and label appropriately. Be sure to include the response $y1$ from the first portion of the lab. Print.
TA check: Please stop and wait for the TA to check your work.

Report

Download and complete the report template available on the course lab website, ensuring you answer all questions.

Lab Session 3: Digital Simulation of a Closed Loop System

In this experiment SIMULINK is used to study a closed loop system for speed control of a magnetic tape drive such as a cassette player. You will compare different controllers for their effectiveness in simultaneously accomplishing disturbance rejection and a smooth step response.

Preparation

Readings:

▷ G. F. Franklin, *et al.*, *Feedback Control of Dynamic Systems*, 3rd or 4th Ed., Sec. 4.1, 4.1.1. These pages are copied for you and attached at the end of this manual. In 7th or 6th Ed., see Sec. 4.2.2 for distance rejection.

Prelab:

Modeling: A diagram of the system is shown below

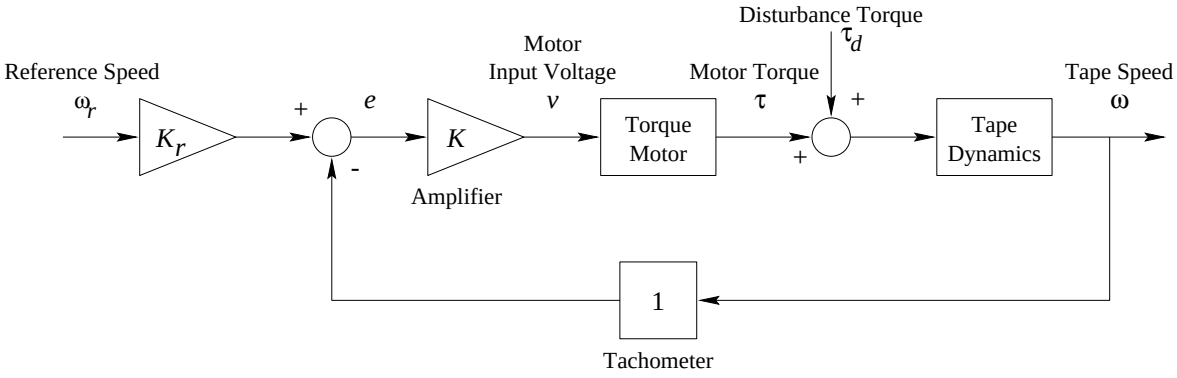


Figure 3.1: Closed-loop system with unity feedback.

The tape dynamics are modeled as a rotational mass-damper system:

$$J\dot{\omega}(t) + B\omega(t) = \tau(t) + \tau_d(t), \tag{3.1}$$

where $J = 0.25[\text{kg} \cdot \text{m}^2]$, $B = 3[\text{N} \cdot \text{m} \cdot \text{s}]$. The torque motor transfer function is determined by an open-loop test *with the motor detached from the system.* A one-volt step is applied (zero initial conditions), and the resulting motor torque is

$$\tau(t) = 5(1 - e^{-3t}), \quad t \geqslant 0. \tag{3.2}$$

The amplifier is a non-inverting gain, K. It is assumed that the tape speed sensor (tachometer) dynamics can be neglected.

Design: We want to have a closed loop system whose steady-state response from a disturbance is small, and whose step response from a reference input has low overshoot.

1. Obtain an all-integrator block diagram for the system in figure 3.1. Show the intermediate variables (τ, v etc.). Note that the motor torque is the convolution of a step input with the motor's transfer function.

2. For the closed-loop system, find the transfer functions $\Omega(s)/\Omega_r(s)$ (with $\tau_d = 0$) and $\Omega(s)/T_d(s)$ (with $\omega_r = 0$). Use numerical values where given. (*Hint*: remember to make your lead coefficient 1.)

3. For what values of K is the closed loop system stable?

4. **Controller 1:** Suppose the disturbance torque is a unit step [N · m]. Find the range of K such that the steady state output in tape speed due to τ_d (use the transfer function you get above with $\omega_r = 0$) is $\leqslant 0.01$ rad/s.

5. Express the desired value of K_r, in terms of K, such that the steady-state output speed for a unit step in ω_r is 1. ($\tau_d = 0$)

6. Using the minimum value of K that you found in (d), what are the values of the closed loop ζ and ω_n? Consider the anticipated response $\omega(t)$ to a step at ω_r ($\tau_d = 0$). What will be the expected t_s, t_r, and M_p? Use the equations provided in the Lab 1 Report doc.

7. **Controller 2:** Suppose, alternatively, that K is selected to produce a low overshoot ($\zeta \geqslant 0.75$). Find the range of K that meets this specification. If K is selected such that $\zeta = 0.75$, what will be the steady-state disturbance response to a unit step? What M_p do we achieve? Can we achieve both our desired disturbance rejection and our desired overshoot?

8. **Controller 3:** To improve the response, suppose we add *derivative feedback* (measured by an *accelerometer* in this case) as shown:

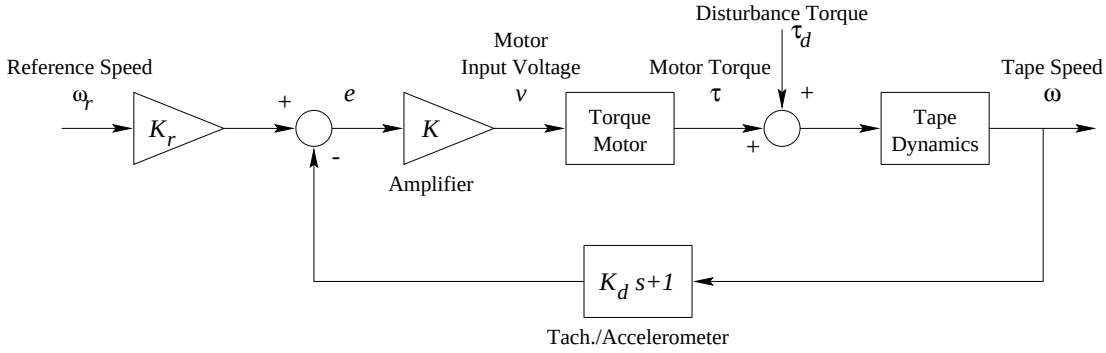


Figure 3.2: Closed-loop system with derivative feedback.

Compute again the new closed-loop transfer functions $\Omega(s)/\Omega_r(s)$ (with $\tau_d = 0$) and $\Omega(s)/T_d(s)$ (with $\omega_r = 0$). Give the ranges of K and $K \cdot K_d$ for which the closed-loop system is stable.

9. Select K and K_d such that the steady state output in tape speed due to τ_d (τ_d is a unit step as before) is 0.01 rad/s and $\zeta = 0.75$. What are the resulting M_p, t_r, and t_s?

Laboratory Exercise

1. Develop an all-integrator SIMULINK block-diagram for the three closed-loop control systems. It is a good simulation practice to retain explicitly in the simulation the various subsystems - actuator, plant, amplifiers, etc. Your diagram should do so, and you should label the inputs and outputs, ω, τ, v, e, of the subsystems. It is possible to make only one diagram, and simply use MATLAB variables in the SIMULINK blocks for the different cases. Do *not* use differentiators in your diagram. Instead, route the required states around the integrators. See appendix A, section IV-c for an example of this.

2. Obtain and print the following time responses for each of the three controllers:

   - $\omega(t)$ for the disturbance response to a unit step in τ_d ($\omega_r = 0$). For comparison, overlay and label the plots for the different controllers.
   - $\omega(t)$ for the reference response to a unit step in ω_r ($\tau_d = 0$). Remember to use the appropriate value of K_r for each controller. Overlay again for comparison. You should end up with six graphs on two plots.

   In short, in this lab you end up with two figures: in figure (1), overlay *three* graphs of responses due to unit step disturbances; similarly in figure (2), overlay *three* graphs of responses due to unit step references.

   Note: Set the stop time for both of these figures to 1 second.
   TA check: Please stop and wait for the TA to check your work.

Report

Download and complete the report template available on the course lab website, ensuring you answer all questions.

Lab Session 4: Introduction to the DC Motor

Several basic features of control systems can be understood by carefully examining DC motor control as a case study. Also, the DC motor is an important component in a myriad of control systems. This DC motor experiment, the first of three, emphasizes system modeling in the time domain. The motor we will study is shown in Figure 4.1, and Figure 4.2 shows a functional block diagram. Figure 4.3 describes the motor dynamics. This diagram will be used to develop the motor transfer function.

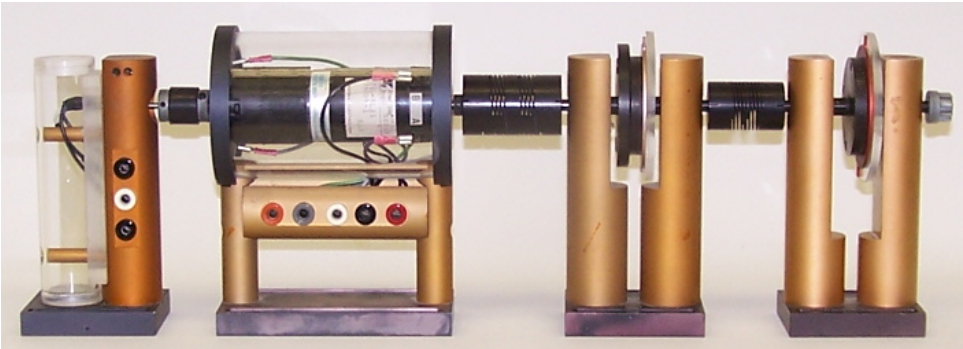


Figure 4.1: The DC motor plant, showing the potentiometer (left), motor with integrated tachometer (center), and double flywheel (right).

System parameters:

Table 3: DC motor plant parameters, symbols, and units.

Parameter	Symbol	Units
Potentiometer gain	K_{pot}	Volts/rad
Tachometer gain	K_{tach}	Volts $\cdot$ s/rad
Armature resistance	R_a	Ohms
Armature inductance	L_a	Henries
Back EMF constant	K_V	Volts $\cdot$ s/rad
Motor torque constant	K_τ	N $\cdot$ m/amp
Motor inertia	J_M	kg $\cdot$ m^2
Load (flywheel) inertia	J_L	kg $\cdot$ m^2
Total inertia ($= J_M + J_L$)	J	kg $\cdot$ m^2
Coulomb (static) friction	c	N $\cdot$ m
Viscous (sliding) friction	b	N $\cdot$ m $\cdot$ s/rad

Preparation

Readings:

A basic description of the system components may be found in B. C. Kuo, *Automatic Control Systems*, 7th edition. (A pdf file is available on the lab web site. The book is also on reserve in the engineering library.) See Sections 4.5 and 4.6, which cover potentiometers, tachometers and DC motors.

We will follow Franklin *et al.* for some basic methods in system modeling in the time domain. See Franklin *et al.* 7th or 6th Ed. Sec. 3.7 (Sec. 3.7 in 3rd Ed., Sec. 3.8 in 4th Ed.) for an introduction to system identification using transient response data. These pages are copied for you and attached at the end of this manual.

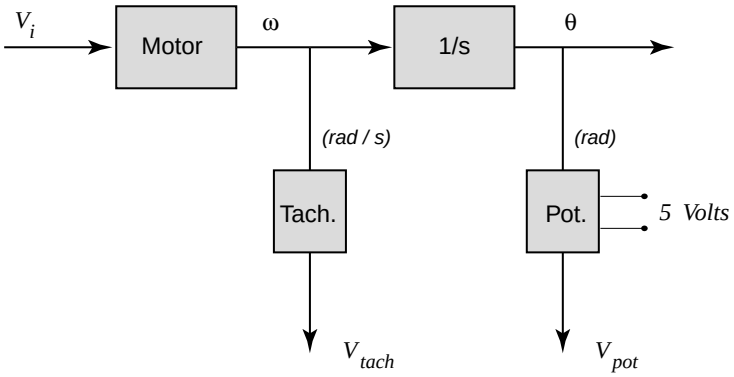


Figure 4.2: Motor block diagram.

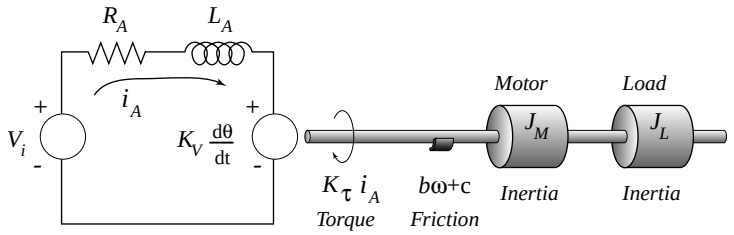


Figure 4.3: Motor schematic diagram.

Prelab:

1. The DC motor, *with the flywheel clamped* ($\omega = 0$), is described by the electrical schematic of Figure 4.4.

   The resistance R_s has been added to obtain voltage measurements, so that the current's time response can be viewed on an oscilloscope . Find $V_o(s)/V_i(s)$ in terms of R_a, R_s, and L_a. What is the *electrical time constant*, τ_e, for the motor?

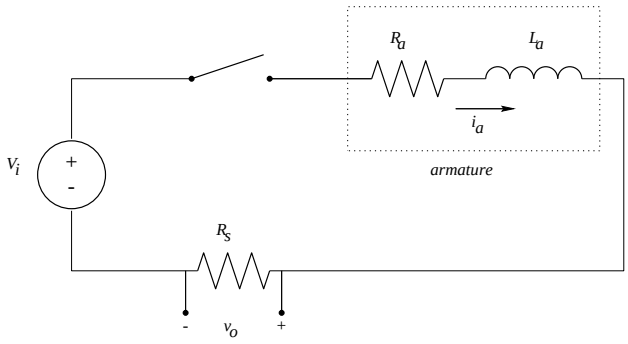


Figure 4.4: Motor electrical diagram with flywheel clamped.

Note: the final model of the motor we will develop will not have R_s. This is only used for identification.

2. The electrical parameters R_a and L_a can be determined from the step response of the motor with its flywheel clamped, assuming R_s is already known. Derive the time-domain step response of $V_o(t)$. Use this formula to show how τ_e can be obtained from a linear fit of $\ln(V_{\text{ss}} - V_o(t))$ where V_{ss} is the steady-state output voltage. This will give a reliable estimate in spite of noise.

   Load the MATLAB data in the file `prelab4.mat` from the website. The variable `Elect` ("Elect" is short for Electrical time constant data) contains the time and voltage data of a step response in columns 1 and 2, respectively (so `time=Elect(:,1)` and `voltage=Elect(:,2)`.

   There are two ways to get a linear fit on this processed data. On the command line, use `polyfit` to obtain the coefficients; for example, `coeffs=polyfit(x,y,1)` fits the model $y \approx \texttt{coeffs}(1) \cdot x + \texttt{coeffs}(2)$. To get the fit graphically, plot the data; then go to the plot's menu *Tools* → *Basic Fitting* and select the options *linear* and *show equation in figure window.*

   Regardless of the method used to fit the data, be careful to get a fit on *linear* data. If V_{ss} is wrong, the plot will be curved instead of straight; vary the value of V_{ss} until the first part of the curve is linear. Now notice that there is a sharp bend in the curve around $t = 0.5$ ms; this marks where the circuit achieved steady-state and the inductance is no longer visible. Thus you should only fit the data between $t = 0$ and $t \approx 0.5$ ms (you can pick the exact range; *Note*: if you do not know how to find the exact range, plot the *whole* range of data then you can decide which part is linear); including the other data will corrupt your results. For example, you can have Matlab only plot the first 10 data points by typing `plot(Elect(1:10,1), Elect(1:10,2))`.

   Obtain τ_e and L_a from this data, assuming that $R_s = 2.5$ and $R_a = 3.3$. Print out the log plot for your prelab. As a sanity check for your results, $L_a \approx 1$mH.

3. Now consider the DC motor with the rotor free to turn. The parameters K_V, K_τ ($= K_V$ in SI units), b, and c can be obtained from the steady-state response to a known step input V_i. Derive the steady-state equations and solve them to find these

parameters from multiple trials.

Suppose the results from two experiments were as follows:

Table 4: Steady-state armature current and angular velocity for two input voltages.

V_i	steady-state i_a	steady-state ω
5 Volts	0.249 A	77.0 rad/s
10 Volts	0.304 A	163 rad/s

Using $R_a = 3.3$ and $L_a = 1$mH, what are the values for $K_\tau = K_V$, b, and c? Use the average value obtained for K_τ as the "correct" value.

4. The mechanical time constant, and hence the total moment of inertia, J, can be determined by opening the motor circuit ($i_a = 0$) while the motor is rotating and observing the resulting decay in speed. The situation is shown in Figure 4.5.

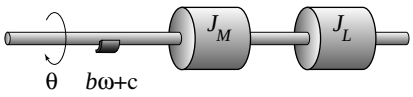


Figure 4.5: Motor mechanical diagram with open electrical circuit.

With the motor spinning only in one direction, the Coulomb friction will not change sign, and we can consider it to be a constant (a **step function** in the Laplace Domain). The dynamics are then described by the differential equation:

$$J\dot{\omega} + b\omega + c = 0. \tag{4.1}$$

Find $\omega(t)$ in terms of J, b, c, and $\omega(0)$. Explain how to find J if the other parameters are known (similar to part (b)). Illustrate your method on the inertial data saved as the variable `Iner` in the `prelab4.mat` data file. `Iner` contains the time and angular velocity data of the motor in columns 1 and 2, respectively. Use the friction values found in part (c). Print and turn in any plots needed by your method.

5. If we assume the motor has a strictly positive angular velocity, we can get a linear model by disregarding the Coulomb friction. (i.e., here you can discard c term but remember to keep b term for viscous friction. We will address this again in lab 5.) Using the numerical values above, obtain the resulting second order transfer function, $\Omega(s)/V_i(s)$ for the motor. Then compute the poles of both the second-order transfer function and its first-order approximation. Use the pole location to discuss the difference between the original transfer function and when L_a is ignored. To observe the effects of inductance, create a first-order transfer function by completely ignoring the inductance (set $L_a = 0$). Use MATLAB to produce step responses and Bode plots for both cases (overlay both cases on the same plot so they can be compared more easily). Print these plots for your prelab. What changed when the inductance was ignored? Which frequency ranges are the same? Different? Does inductance cause significant effects?

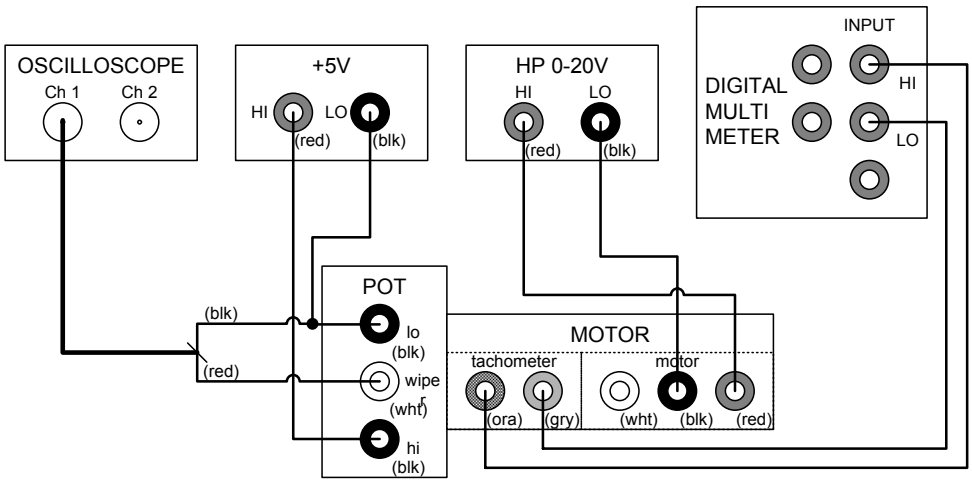
⚜ Laboratory Exercise ⚜

To identify the motor you will collect a large amount of data. In addition to the data obtained using VEE, data tables are provided in the lab text to help you collect all necessary measurements.

I. Calibration of the Tachometer

The tachometer is a generator with an output voltage proportional to the angular velocity of the motor shaft. Here we use the potentiometer to measure the angular shaft position, θ, and obtain an accurate estimate of the velocity. This allows us to calibrate the tachometer *gain*, the ratio of voltage generated by the tachometer per unit angular velocity (rad/sec). We will not need to save any waveforms to disk with VEE.

1. Turn on the PC, the scope, DMM, and the HP programmable power supply.
2. Connect the potentiometer to the motor and firmly tighten the thumb screws at the base of the potentiometer mount.
3. Start VEE and open the project file at `N:\labs\ECE486\486lab4.vxe`. You should see control panels for the Agilent scope and the HP 6632A power supply. By default, VEE will save your data in `U:\lab4.m`. You should change the **To File** field to `C:\matlab\ECE486\NETIDs` at this time. Also, if you have to restart VEE in the middle of taking data, rename the saved file to avoid overwriting it when VEE collects new data.
   *Note: Agilent VEE displays files of type *.vee by default. You must change this box to VEE RunTime Programs (*.vxe)*



4. Wire up the circuit as shown. Make sure the wires do not touch any spinning parts. Also, set the DMM to measure DC voltage.
5. Verify in VEE that the scope parameters are:

Channel1
- V/div → 1V
- Offset → 3V
- Coupling → DC
- Band-Width Limit → ON

Channel2
- V/div → 1V
- Offset → 3V
- Coupling → DC
- Band-Width Limit → ON

Trigger
- Level → 1V
- Source → CHANNEL1
- Slope → POSITIVE
- Noise Reject → ON
- High Freq. Reject → ON

Time
- Base → 20m (20 ms) S/div
- Reference → CENTER

Then click the **Send New Scope Parameters** button in VEE. Finally press the **Measure Period of Channel1** button. This will setup the scope to automatically measure the period of the saw tooth waveform that will be produced by the potentiometer when the motor is spinning. You will see cursors on the scope screen measuring the period. The measured period is displayed in a small box at the bottom left of the scope screen.

6. Turn on the 5 V power supply connected to the potentiometer using the switch on the patch panel.

7. To initialize the power supply, in VEE click the **Reset** button. Next disable the programmable power suppy's output and set **Set Voltage** to 5 V and **Set Current** to 2 A.

8. Turn the power supply's output on by using the **Output Enable** button in VEE. The motor should begin to spin and you should see a sawtooth waveform on the scope. This is the voltage output by the potentiometer as the motor spins.

9. Record the voltage output by the tachometer on the DMM in the table below. This is the voltage generated by the tachometer for the estimated angular velocity of the motor.

10. Also record the period from the scope window. Now disable the output of the power supply to save wear on the potentiometer.

11. Repeat this experiment twice more to complete the table. The ratio of the tachometer voltage and the angular velocity of the motor measured in radians/second is the tachometer gain:

$$K_{\text{tach}} = \frac{V_{\text{tach}}}{\omega}$$

Table 5: Measured tachometer calibration data and calculated angular velocity and tachometer gain.

	Measured		Calculated	
V_i	Δ_t	V_{tach}	ω	K_{tach}
5V				
10V				
15V				

Notes: Δ_t is the duration of one rotor revolution.

12. Use the average of the three observations as your estimate of the tachometer gain. Average K_{tach}: _____

II. Armature Resistance and Back-EMF

Now you will collect the data needed to find R_a and K_V.

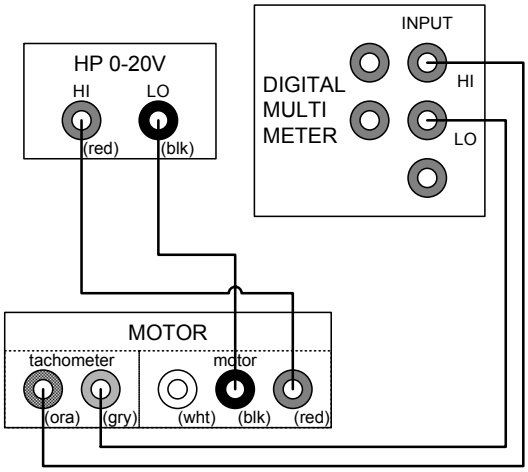


Figure 4.6: Wiring diagram for measuring V_i, i_a and V_{tach}.

1. Remove the potentiometer from the motor assembly. Adjust your wiring to match figure 4.6.
2. Within VEE adjust **Set Voltage** to 5 V, and **Set Current** to 2 A.
3. Using the **Output Enable** switch on the power supply window turn on the power to the motor. For each voltage V_i applied across the motor, record the steady-state current i_a drawn by the motor shown on the power supply and V_{tach}, the voltage generated by the tachometer as shown on the DMM.
4. Repeat this experiment 7 times to again cover the voltage range 5V–12V. Now switch leads on the *HP* 0–20V panel to get negative voltages, and repeat the 8 measurements. **Your i_a values must now be recorded as negative because the motor voltage has been reversed.** This experiment is noisy, so you will probably be able to get only 2 digits of accuracy for i_a and 3 digits for V_{tach}. There is a button in VEE that will take several measurements and report the average. Use this to get more accuracy (To make it less likely to be dysfunctional, click **Reset** *before* you **Set Voltage** each time; if the average button does not work, you will hear a beep, then you can directly read current and voltage values rather than use the "average" button).

Table 6: Measured armature current and tachometer voltage, with calculated angular velocity, for positive and negative input voltages.

	Measured		Calculated
V_i	steady-state i_a	V_{tach}	ω
5V			
6V			
7V			
8V			
9V			
10V			
11V			
12V			
-5V			
-6V			
-7V			
-8V			
-9V			
-10V			
-11V			
-12V			

Note: calculate ω using the tachometer gain found in the previous section.

5. In prelab, you solved for V_i in terms of i_a, ω, R_a and K_V. Enter the equation below.
   $V_i =$ _____

6. In MATLAB use the matrix division operator "\" (this "left divide" operator is actually not a "normal" division; it is internally using least square approximation in MATLAB if the problem is over-determined, like in this case, we have 16 equations but only two unknowns) to solve for R_a and K_V.

   ```
   x_val = ([ ia, w]\Vi)';
   ```

   R_a: _____
   K_V: _____

III. Constant and Viscous Friction Coefficients

Now you will solve for b and c. For various reasons, the frictional forces are often asymmetric; there is more resistance when the motor spins one way than the other. For this reason, we are using the friction model of Figure 4.7; the forward and reverse friction coefficients will be found independently.

1. Use two best-fit lines (one in each direction) to determine the friction coefficients.

   b_{pos}: _____ c_{pos}: _____
   b_{neg}: _____ c_{neg}: _____

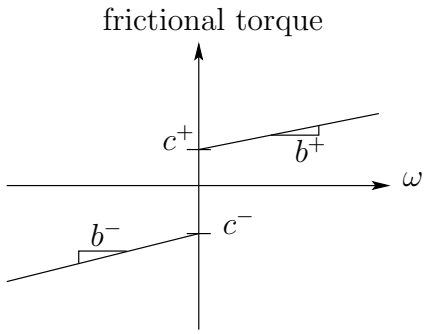


Figure 4.7: Rotary friction model.

IV. Armature Inductance

This section uses parts from the long drawer under the bench. The *Resistor Box* is a small blue box with banana jacks on the ends. **Be careful to keep the resistor box on the green rubber mat so nothing shorts on the metal tabletop.**

The armature inductance L_a is measured by observing a step-response to an input voltage while the rotor is locked. This is done by observing the voltage drop across a resistor R_s connected in series with the locked motor. The step-by-step instructions are given below:

1. In this section, we are working with very small resistances that could easily be affected by bad connectors. Use the DMM to check the resistance of all the wires you use; they should have no more than 0.05 of resistance. If they have more, then have the lab TA tighten the connectors with a screw driver.

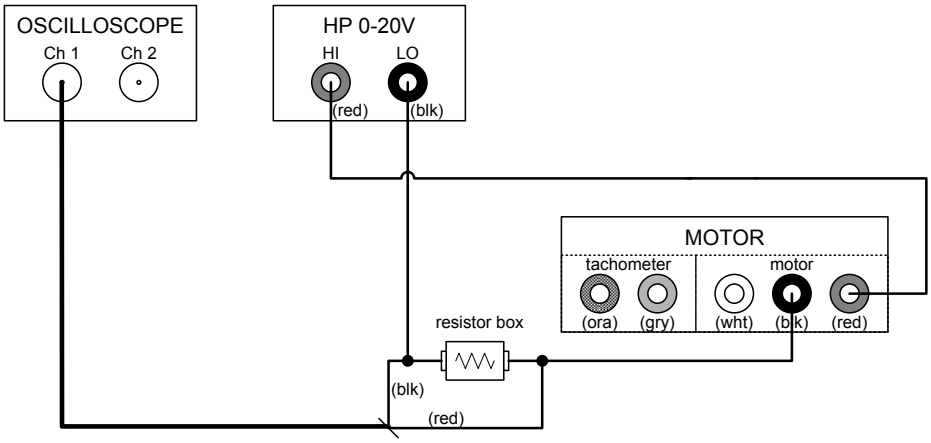
First we determine R_s:

2. Measure the resistance R_s of the resistor box using the DMM, it should be close to 2.5.

   Measured R_s: _____

To calculate L_a:

3. Disable the power supply's output by changing the **Output Enable** button to "off".
4. Attach the rotor-locking attachment to the motor, and tighten the thumb screws. There is less flex in the system if you **place the locking mechanism on the left** of the motor.
5. Wire up the circuit shown below in order to measure L_a. This corresponds to the electrical circuit of Figure 4.4.



6. Set the oscilloscope to the following:

   Channel1 { V/div → 0.5V; Offset → 1.5V; Coupling → DC; Band-Width Limit → ON }
   Channel2 { V/div → 0.5V; Offset → 3V; Coupling → DC; Band-Width Limit → ON }

   Trigger { Level → 0.5V; Source → CHANNEL1; Slope → POSITIVE; Noise Reject → ON; High Freq. Reject → ON }
   Time { Base → 0.2m (0.2 ms) S/div; Reference → CENTER }

   and then press the **Send New Scope Parameters** button to initialize the scope.

7. In Agilent VEE set the programmable power supply's voltage to 5 V and its current limit to 2 A.

8. Turn on the power supply by selecting **Output Enable** *on* in VEE. Disconnect and reconnect the (red) power supply HI voltage from the bench panel. (Make sure the scope is in the Run mode, **Run/Stop** button green.) Do this a few times until you get a smoothly rising step response. It may help to **slide the plug in and out** instead of pulling it completely out (hence we are trying to create a "manual" step function). To get the cleanest signal, it may help if you pre-stress the rubber connectors by turning the flywheel in the direction the motor applies torque; this is not always required. Do not use the Output Enable button in VEE to switch power to the motor because the time constant $\tau_{\text{powersource}} \gg \tau_{\text{motor}}$.

9. When you have a clean waveform, press the VEE **Collect Data From Scope** button. Each time that you press **Collect Data From Scope**, the data from the scope will be stored in the vector "Elect$j$", where the integer j is incremented by a counter each time that you store new data.

10. While you are not collecting data, turn off the power using the **Output Enable** switch to keep the motor from overheating.

11. Repeat this experiment five more times to cover the voltage range 5V–10V in 1 Volt increments. If the 9V or 10V signal goes off the screen, change the V/div setting for

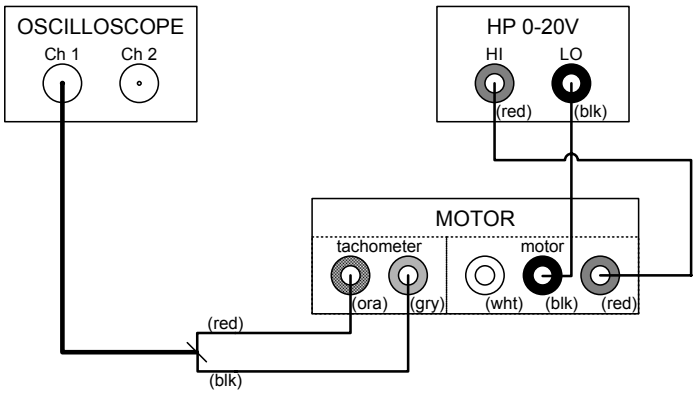
the scope in VEE; this change will not cause problems when MATLAB loads the data, but data which is not shown by VEE will be truncated. Use the technique from the prelab to get estimates of L_a, then report the average value.

Table 7: Measured armature inductance values and average armature inductance.

Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Average L_a

V. Rotor Moment of Inertia

The rotor moment of inertia can be inferred from the mechanical time constant τ_m, and the friction constants b and c. The mechanical time constant can be obtained by observing the decaying angular velocity when the current supply to the motor is cut off.



1. Remove the rotor-locking attachment and the resistor box. Change the wiring to match this diagram.
2. In VEE first click the **Clear** button to reset the counter to zero. Also in the **Data Name Formula** box change the name from "Elect" to "Iner". (Iner is short for Inertial time constant data.)
3. Set the Oscilloscope to the following:

Channel1
- V/div → 1V
- Offset → 3.5V
- Coupling → DC
- Band-Width Limit → ON

Channel2
- V/div → 1V
- Offset → 3.5V
- Coupling → DC
- Band-Width Limit → ON

Trigger
- Level → 4V
- Source → CHANNEL1
- Slope → NEGATIVE
- Noise Reject → ON
- High Freq. Reject → ON

Time
- Base → 200m (200 ms) S/div
- Reference → CENTER

Press the **Send New Scope Parameters** button to send these settings to the scope.

4. With the power supply's output disabled, adjust **Set Voltage** to 10 V in VEE. Keep **Set Current** at 2 A.
5. Using the **Output Enable** switch in the power supply's window, turn on the power to the motor. Wait till the tachometer reading is roughly constant. Pull the HI (red) power supply cable from the patch panel.
6. When you have a waveform on the Oscilloscope, make sure the scope is in the **Stop** mode and then press the **Collect Data From Scope** pushbutton. The scope waveform is now stored in a coordinate format.
7. Using the method from the prelab, calculate the moment of inertia, J, from this curve. $\triangleright$ **Remember**: The measured curve is $V_{\text{tach}}(t)$, not $\omega(t)$
8. Repeat this experiment five more times, incrementing the voltage 1 V each time to 15 V. You may need to adjust the trigger level, scale, and/or time base to keep the whole trace on the scope. You may need to switch to 2V/div for the 14V and 15V trials. **TA check:** Please stop and wait for the TA to check your work.

Table 8: Measured rotor moment of inertia values and average rotor moment of inertia.

Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Average J

Report

Download and complete the report template available on the course lab website, ensuring you answer all questions.

Lab Session 5: PD Control: Analog Computer & C2000 DAQ Board

This experiment continues the study of the DC motor begun in Lab 4. The model that you obtained in the last set of experiments was derived from transient response data. You will now design a proportional-derivative controller to do position control, and implement it on both the analog controller and digitally using Simulink and the C2000 Data Acquisition Board. In each case, you will record and analyze the step response of the closed-loop system for a set of feedback gains. Furthermore, you will explore the effects of friction compensation.

Preparation

Readings:

To prepare for the design in this lab, you should review block diagram manipulation in Franklin *et al.* Section 3.2.1. Also be comfortable with root locus design from lecture and also Chapter 5 in Franklin *et al.*, especially Section 5.2.3 for selecting gain from the root locus.

Prelab:

1. Using the results from your last experiment, obtain the time constant τ and the gain K in the motor transfer function $\Omega(s)/V_m(s) = K/(\tau s + 1)$.

2. We next consider the design of a closed-loop position control system using PD control. The structure is shown in Figure 5.1. For the sake of concreteness, assume from now on that the block MOTOR has the transfer function

$$\Omega(s)/V_m(s) = \frac{18}{1 + 0.2s}.$$

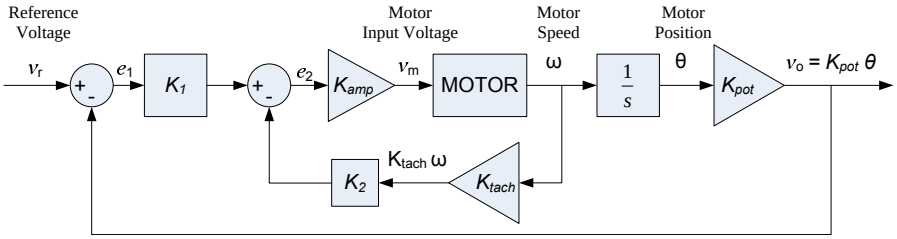


Figure 5.1: A feedback control system for the DC motor.

In the lab, we will connect the system illustrated in Figure 5.2 to the analog computer set up shown in Figure 5.3, to create the closed-loop system of Figure 5.1. We are then interested in the voltage from the potentiometer, the output of the system, and how it behaves as the reference input voltage v_r is varied. This reference input will be supplied by the HP waveform generator.

Note: In practice, the reference input would typically be an angle, not a voltage. One then must construct a device (such as a potentiometer) to translate the desired angular position to an input voltage v_r.

Suppose $K_{\text{amp}} = 2.4$, $K_{\text{pot}} = 10/2\pi \approx 1.6$ volt $\cdot$ s/rad, and $K_{\text{tach}} = 0.03$ volt $\cdot$ s/rad. Find the closed-loop transfer function $V_o(s)/V_r(s)$ in terms of the potentiometer settings P_1 $(= K_1/10)$ and P_2 $(= K_2/10)$. Given this transfer function, compute the closed-loop pole locations for the 3 sets of gains:

Table 9: Prelab parameter values for the three feedback-control cases.

Parameter	Case 1	Case 2	Case 3
P_1	0.15	0.25	0.1
P_2	0	0.35	0.5

3. For the same control system considered above, find values of P_1 and P_2 such that overshoot is approximately 15% ($M_p \approx 0.15$) and the rise time is approximately 40 ms ($t_r \approx 40$ms). Use the formulas provided in the report templates (e.g. Lab 1 or Lab 3).

4. Show the two root locus plots for your P_1 and P_2 values. Using the root locus plots, show that your values of P_1 and P_2 meet the M_p and t_r requirements from the previous question. Use the command **rlocus()**. At the ECE486 course website find

the Handbook Lectures 10 and 11 to guide you on how to isolate P_1 in the closed loop denominator for the first root locus plot and then isolate P_2 in the closed loop denominator for the second plot. (You should show two separate plots here, **not** one with both root locus plots combined).

5. Draw a wiring diagram for the analog computer setup shown in Figure 5.3 (P_1 and P_2 are variable). (This diagram implements the PD control law.)

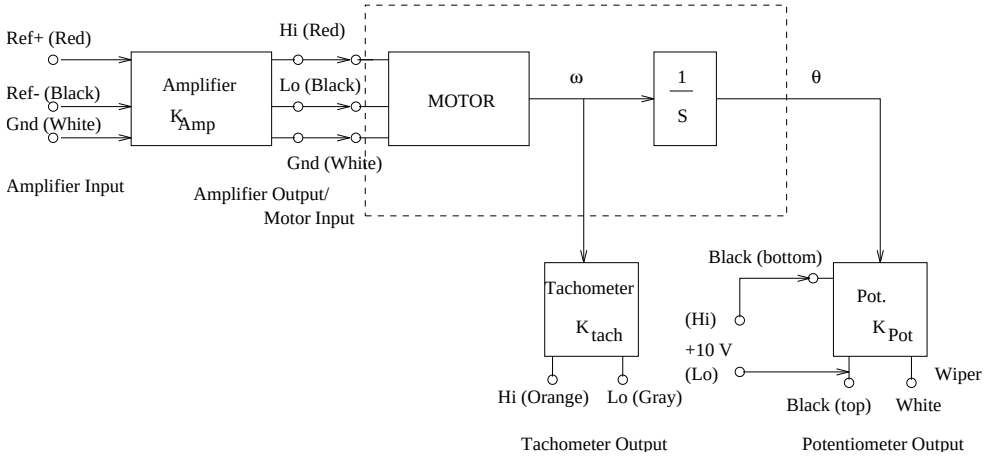


Figure 5.2: Motor block diagram.

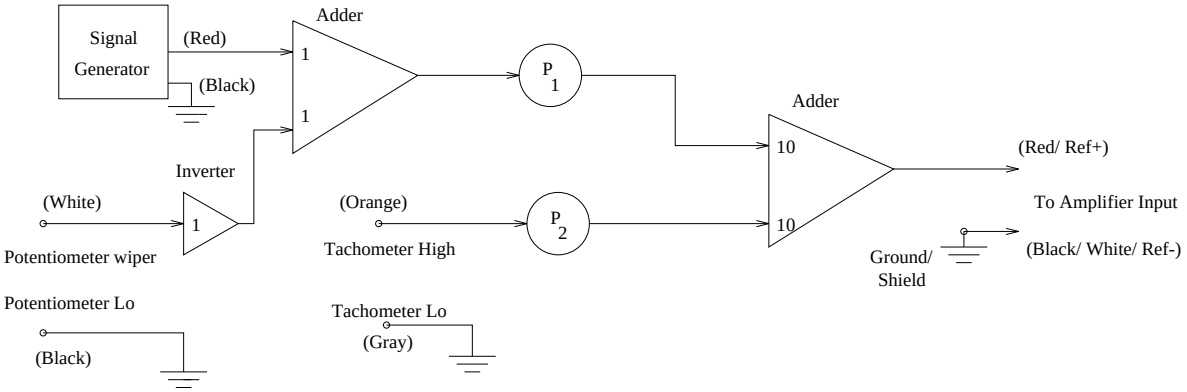


Figure 5.3: Analog computer wiring diagram.

Laboratory Exercise

I. PD Control using Analog Computer

To begin, wire up the analog computer according to your wiring diagram from prelab part (e). Then follow these steps:

1. Make sure power to the patch panel is off.
2. Connect the potentiometer to the motor assembly.
3. Use the wiring diagram (Figure 5.4) to connect the analog computer to the other components.
4. In order to collect data for analysis in MATLAB, start VEE and open `M:\labs\ECE486\486lab5.vxe`. Remember to select *Files of type:* to be **.vxe*. If desired, change the data filename with the **To File** button and the "y" in the **Data Name Formula** box.
5. Turn on the oscilloscope and function generator; wait for them to start up.
6. In VEE, verify that the scope is set to:

   Channel1 { V/div → 1V; Offset → 5V; Coupling → DC; Band-Width Limit → ON }
   Channel2 { V/div → 1V; Offset → 5V; Coupling → DC; Band-Width Limit → ON }

   Trigger { Level → 4V; Source → CHANNEL2; Slope → POSITIVE; Noise Reject → ON; High Freq. Reject → ON }
   Time { Base → 100m (100 ms) S/div; Reference → LEFT }

   Press the **Send New Scope Parameters** button to send these settings to the scope.
7. Also verify that the function generator is set to:
   - Type of Waveform → Square.
   - Volts Peak to Peak → 2V.
   - Offset → 4V.
   - Frequency in Hertz → 0.5Hz.

   Press the **Send Function Generator Settings** button to send these settings to the function generator.

   Since the scope is triggering on a 0.5 Hz signal, it will take a while to update; so be patient. If the scope never triggers, in VEE push the **Put Scope in Roll Mode for Debugging** button to switch to **ROLL** mode to see what is happening. After

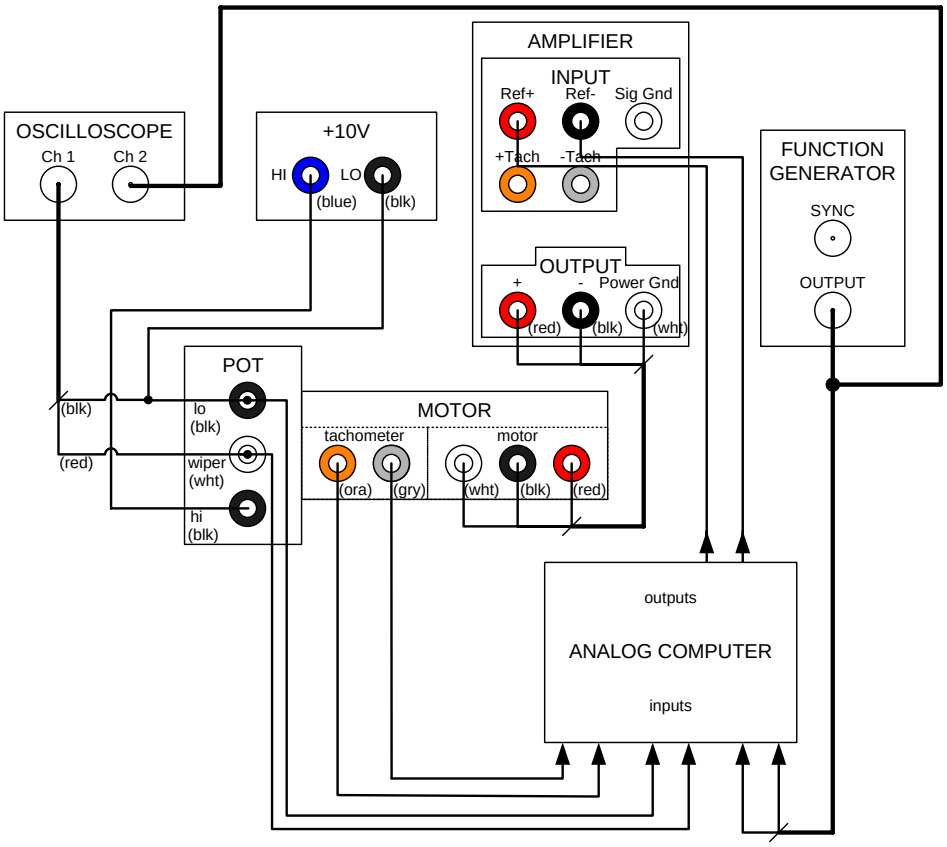


Figure 5.4: Wiring diagram for Part I.

finding your triggering problem, in VEE press the **Send New Scope Parameters** button to set the scope back up.

Have your TA check your wiring and instrument settings before you proceed.

TA check: Please stop and wait for the TA to check your work.

8. Connect the push button (in the long drawer under the bench) to the **Amp Inhibit** on the amplifier. Turn on both power switches on the patch panel.

9. Turn on the analog computer. In **Pot Set** mode, set the potentiometer levels P_1 and P_2. The values of P_1 and P_2 for which the experiment should be repeated are:

Table 10: Controller parameter values for the first three cases and the prelab part c value.

Parameter	Case 1	Case 2	Case 3	Prelab part c value
P_1	0.15	0.25	0.1	_____
P_2	0	0.35	0.5	_____

10. Switch to **OPR** mode on the analog computer. Press the push button and wait. Soon you should see a step response on the scope (it may take two changes of the input, based on the triggering settings). When you have a good trace on the scope and the scope is in the **Stop** mode, press **Collect Data from Scope** in VEE to display the waveform on the computer and save it to the file. Repeat these steps for

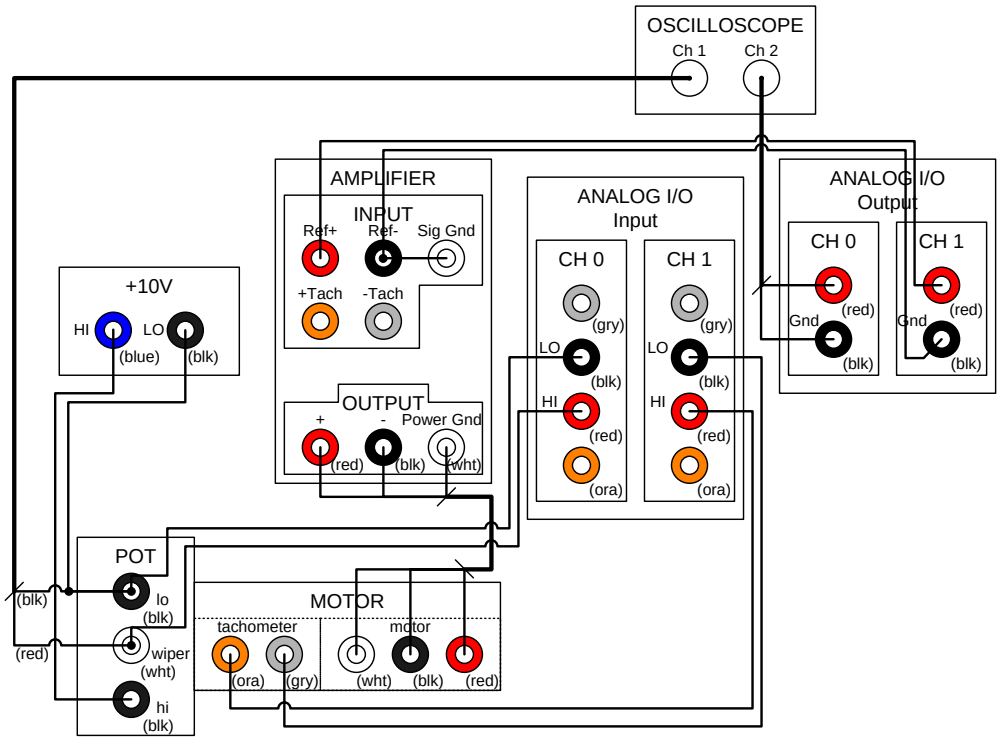


Figure 5.5: Wiring diagram for Part II.

the other values of P_1 and P_2.

11. You can adjust P_1 and P_2 while the controller is running. Find the P_1 and P_2 that meet the specifications.
    Best Analog Computer P_1: _____
    Best Analog Computer P_2: _____
    Demonstrate to your TA how to reject disturbances and avoid overshoot using these two gains. **TA check:** Please stop and wait for the TA to check your work.

12. Turn off the patch panel, the analog computer, and the function generator. Close VEE.

II. PD Control using Simulink and the C2000 DAQ Board

As already explored in previous labs, SIMULINK is a MATLAB based programming environment mainly for simulating and analyzing dynamical systems. It allows you to design controllers graphically.

In order to employ real-world signals with SIMULINK , there must be some way to receive analog signals, convert them to digital ones, and then manipulate them. The opposite is needed to send signals out, perhaps to control a peripheral like a DC motor. In ECE486, this is done using the C2000 Data Acquisition (DAQ) Board and the corresponding SIMULINK toolbox named "C2000 Microcontroller Blockset". The C2000 DAQ Board is a multipurpose microcontroller produced by Texas Instruments. In this class, we utilize two of its ADC

(analog-to-digital converter) input channels and two of its DAC (digital-to-analog converter) output channels to receive and send voltage signals. The "C2000 Microcontroller Blockset" toolbox automatically converts your SIMULINK block diagram to C code that can run on the microcontroller. This, in turn, allows the controllers you build to be run in real time and control actual systems.

In this lab, we use this setup to send analog signals out (the reference voltage and control signal) and collect data from the motor (V_{POT} and V_{TACH}). The C2000 DAQ Board is connected to the Analog I/O Input and Output sections of the patch panel.

To guard against possible network glitches, your SIMULINK files **must be saved** on the **C:** drive of your station's computer before compiling.

The following steps will walk you through setting up your SIMULINK diagram to control the motor using PD control.

1. Wire the circuit as shown in Figure 5.5.
2. Verify that the C2000 DAQBoard is powered on and it's USB cable is plugged into your bench's Windows 11 PC.
3. Copy the Lab 5 SIMULINK files provided for you. In Windows Explorer, navigate to the folder `N:\labs\ECE486`. Copy the folder **Lab5** into your C drive folder (should be located at `C:\matlab\ECE486\NETIDs`). *Note:* This copied folder contains files *Lab5_Starter.slx* for Part II and *Lab5_Friction.slx* for Part III.
4. Change the *Current Directory* (left-hand panel) in MATLAB to your group's lab 5 folder at `C:\matlab\ECE486\NETIDs\Lab5`
5. Open the *Lab5_Starter.slx* SIMULINK file by double clicking it in the left-hand panel of MATLAB. This file contains a custom block created so you can interact with the Analog I/O (input/output) section of the patch panel, and therefore the DAC and ADC channels of the C2000 DAQ Board. In *Lab5_Starter.slx* you need to select the correct COM port for the C2000 DAQBoard. In Windows 11 Device Manager check the COM port of "XDS100 Class USB Serial Port". Then in *Lab5_Starter.slx* go to "Hardware Settings" and under "Target Hardware Resources" change "External Mode" to use the COM port assigned in Device Manager and save the file "Ctrl-S".
6. To place the various blocks and gains needed in SIMULINK, you will need to look through the Library Browser by clicking the library browser icon () in SIMULINK toolbar (of the SIMULATION tab). Alternatively, you can double-click on any blank space in your diagram to then search for blocks by typing in the name.
7. To understand what portion of the control loop is accomplished in SIMULINK, refer to Figure 5.1. The analog (ADC) inputs to the C2000 DAQ Board are $V_{\text{pot}} = v_o$ and $V_{\text{tach}} = K_{\text{tach}} \cdot \omega$. These are the feedback signals. The analog (DAC) output from the C2000 DAQ Board is e_2. This is the control effort. Therefore everything between the inputs and the output will be done in software on the C2000 DAQ Board. You will also output v_r for display on the oscilloscope.

8. The starter file uses the sum of a **Constant** and a **Pulse Generator** to create the reference signal v_r, which is a square wave from 3V to 5V with a frequency of 0.5 Hz.

9. Complete your model, using blocks from the SIMULINK Library Browser. As in Figure 5.5, V_{pot} is connected to ADC channel 0, V_{tach} is connected to ADC 1, v_r is connected to DAC 0, and e_2 is connected to DAC 1. Use **Slider gain** blocks instead of a gain blocks for K_1 and K_2.

10. The starter file uses a **Mux** to observe v_r and V_{pot} simultaneously in the **Scope** plot. Set up the scope to record data by double-clicking on the scope to open it. Then click on the *parameters* button ( ). Select the *Logging* tab, click *Log data to workspace* and change the *Format* to `Array`.

11. Connect the push button to the **Amp Inhibit** on the amplifier.

    Have your TA check your wiring and SIMULINK settings before you proceed.

    TA check: Please stop and wait for the TA to check your work.

12. Run your Simulink model by selecting the *Hardware* tab and click the *Monitor & Tune* button in the top right corner of the model window to generate C code, download and run the code on the C2000 DAQBoard. This takes about 45 seconds and maybe longer for the first build. While building and loading, to see the progress, you can click *Diagnostic Viewer* on the bottom edge of your Simulink window. Please note here that when your controller is being downloaded to the C2000 DAQBoard the DAC control signal is in an unknown state. So do not press the Amp Inhibit button until your controller is running. (Nothing would be damaged, the motor would just spin.) You will know that your code is running when time starts progressing in a display at the bottom of your Simulink window. When time starts progressing, click the "Start DAC / Stop DAC" Simulink button to enable the DAC outputs and then press and hold the Amp Inhibit button. Your controller should start commanding the motor back and forth. The C2000 DAQ Board has two LEDs. The red LED blinks on and off when your controller is running and the blue LED blinks on and off when the "Start DAC / Stop DAC" Simulink button has been pressed, indicating that the DAC channels are enabled.

13. For the report, you will use MATLAB to view, plot, or manipulate the data to compute the overshoot, rise time, settling time, and steady-state error of the step response for 4 sets of gains, so collect data for each of these cases:

Table 11: Controller gain values for the first three cases and the prelab part c value.

Gain	Case 1	Case 2	Case 3	Prelab part c value
K_1	1.5	2.5	1	_____
K_2	0	3.5	5	_____

Note that these are the same values you experimented in the previous section. These are simply given in terms of K instead of P. Make sure to convert your prelab design values to K as well (they should be greater than 1, less than 10).

14. Find the K_1 and K_2 that meet the specifications. *Tip*: Set the *Stop time* to `600`s (do not go larger than this) and adjust the slider gain blocks while the system runs.
    Best SIMULINK K_1: _____
    Best SIMULINK K_2: _____
15. Close your SIMULINK model.

III. PD Control with Friction Compensation

Friction compensation is a common technique for alleviating the effects of friction and Back EMF ($K_V * \omega$). Recall from Lab 4 that the motor's first order transfer function approximation is:

$$G(s) = \frac{\omega(s)}{V(s)} = \frac{K_\tau}{JR_a s + K_\tau K_V + R_a b}.$$

To explain friction compensation it is best to look at the time domain equations for the motor with coulomb (static) friction added.

$$J\dot{\omega} = \frac{K_\tau}{R_a} V(t) - \left(\left(\frac{K_\tau K_V}{R_a} + b \right) \omega + c \right)$$

Note that coulomb friction, c, is nonlinear because it is positive for positive motor spin and negative for negative motor spin. The above equation can be simplified with $B_{Total} = \frac{K_\tau K_V}{R_a} + b$ leaving

$$J\dot{\omega} = \frac{K_\tau}{R_a} V(t) - (B_{Total}\omega + c)$$

Then to apply friction compensation
if $\omega > 0$, $V(t) = \frac{R_a}{K_\tau} (B^+_{Total}\omega + c^+)$
if $\omega < 0$, $V(t) = \frac{R_a}{K_\tau} (B^-_{Total}\omega + c^-)$
To implement this if statement, a Simulink block will be given to you.

Follow these steps to tune your motor B_{Total} and c values.

1. Close your Simulink file from the previous section.
2. Ensure the Current Directory (left-hand panel) in MATLAB is set to your group's lab 5 folder at `C:\matlab\ECE486\NETIDs\Lab5`. Open the *Lab5_Friction.slx* file by double clicking it in the left-hand panel. In *Lab5_Friction.slx* you need to select the correct COM port for the C2000 DAQBoard. In Windows 11 Device Manager check the COM port of "XDS100 Class USB Serial Port". Then in *Lab5_Friction.slx* go to "Hardware Settings" and under "Target Hardware Resources" change "External Mode" to use the COM port assigned in Device Manager and save the file "Ctrl-S".
3. Notice the manual switches: these turn on/off the controller and friction compensation. Double-click a manual switch to "switch" it. *Note:* **They can be switched while the code is running!**

4. Notice the gains K_1 and K_2. They can also be changed while the code is running.

5. Double-click on the *Asymmetric Linear Friction* block. There is a list of friction parameters. Enter your values found in Lab 4 for c^+, c^-, B^+_{Total}, and B^-_{Total} but remember B_{Total} is the b's you found in Lab 4 plus a back-emf component. See equation above with $K_\tau = K_V$ in SI units. The values of these parameters can also be changed while the code is running, and they take effect as soon as you hit **Apply**.

6. Set the manual switches to run this program in open-loop with friction compensation applied (i.e. no controller running).

7. Run your Simulink model by selecting the *Hardware* tab and click the *Monitor & Tune* button in the top right corner of the model window to generate C code, download and run the code on the C2000 DAQBoard. While building and loading, to see the progress, you can click *Diagnostic Viewer* on the bottom edge of your Simulink window. Please note here that when your controller is being downloaded to the C2000 DAQBoard the DAC control signal is in an unknown state. So do not press the Amp Inhibit button until your controller is running. (Nothing would be damaged, the motor would just spin.) You will know that your code is running when time starts progressing in a display at the bottom of your Simulink window. When time starts progressing, click the "Start DAC / Stop DAC" Simulink button to enable the DAC outputs and then press and hold the Amp Inhibit button. (Remember that the C2000 DAQ Board's red LED indicates that your controller is running and the blue LED indicates that the "Start DAC / Stop DAC" Simulink button was pressed to enable the DAC outputs.)

8. Observe the behavior when you apply an initial velocity to the motor (spin it by hand). You will adjust friction gains (double-click on the *Asymmetric Linear Friction* block) so that the motor seems to run "frictionless" in both directions.

   (h) First, remind yourself which direction is positive spin for the motor.

   (h) Tune c^+, the positive direction Coulomb friction term, by setting c^-, B^+_{Total}, and B^-_{Total} to 0. Set c^+ to a larger value than what you found in lab 4. With the motor enabled, spin the motor with your hands or it may just start spinning. You should see it continue to spin. If you do not, increase c^+ more. Now, decrease c^+ slowly until you find the value that stops the motor spinning.

   (h) Now, tune B^+_{Total} by increasing it so that the motor does not slow down too quickly when spun by hand at a good speed. The goal is not perfect compensation, just a substantial reduction in the observed friction. If B^+_{Total} is too high, the motor will speed up. So if you spin the motor and it begins accelerating, you have increased B^+_{Total} too far. Tune B^+_{Total} until the motor spins at least 5 seconds before stopping. You probably will be able to get it to spin longer than that.

   (h) Do the same process as steps (b) and (c) for c^- and B^-_{Total} respectively, enabling the motor and spinning it in the negative direction. You do not need to zero

the positive terms.

Record these gains in Table 12 below and demonstrate the running system to your TA.

Comment: the B_{Total} and c gains obtained do not compensate friction perfectly. But, it is true that employing friction compensation does help reduce the effects of friction. Because the system has friction compensation now, the MOTOR transfer function in Figure 5.1 has technically changed. Because of this, for Step 6 below you would ideally go back to Prelab 5 and re-calculate gains K_1 and K_2 based on the new transfer function. However since lab time is limited, you can simply follow what is described in Step 6 and just make sure you collect responses that are "stable" so that you can calculate specifications of time domain responses, such as t_r, t_s, and M_p.

TA check: Please stop and wait for the TA to check your work.

Table 12: Friction values from Lab 4 and Lab 5.

Friction parameter	Lab 4	Lab 5
B^+_{Total}		
c^+		
B^-_{Total}		
c^-		

9. **Final Data Run**. Run this program in closed-loop (i.e. with the controller running) with friction compensation applied. Collect step response data for the gains in the below table.

   DO NOT collect data for the $K_1 = 1.5$ and $K_2 = 0$ case, just observe what happens with the motor response. This case already has large overshoot without friction compensation. So adding friction compensation increases the overshoot tremendously. Remember that the potentiometer voltage corresponds to a single turn of the motor. So if there is too much overshoot, the potentiometer will rollover to a value that is incorrect, causing the system to go unstable. **Have your TA show you a demonstration** of using their hand to add external friction to avoid the system going unstable.

   As you collect data, experiment by turning off and on the friction compensation switch to see the effects it has on the response.

   Change the variable names in the scope parameters as needed for each set of gains.

Table 13: Controller gain values used for friction compensation cases.

Gain	Case 1	Case 2	Case 3	Prelab part c value
K_1	1.5	2.5	1	_____
K_2	0	3.5	5	_____

Report

Download and complete the report template available on the course lab website, ensuring you answer all questions.

Lab Session 6: Lead Controller Design

In Lab Session 5, motor position and angular velocity measurements were used to construct a compensator. Frequently in practice, derivative measurements are not directly available, and differentiation of position measurements may not be advisable because of measurement noise. Lead compensators are a class of controllers that can be successfully used even when derivative information is not directly available. Lead compensators have the general form

$$G_c(s) = K_c \frac{\tau_z s + 1}{\tau_p s + 1}, \tag{6.1}$$

with $\tau_p < \tau_z$. The pole provides high frequency gain roll-off that results in less amplification of high frequency measurement noise than is observed with a PD compensator. This is one significant advantage of the lead compensator. In this set of experiments, you will build upon the frequency domain viewpoint. In this lab you will directly use a Bode plot rather than a root locus to design the lead compensator G_c to meet the following objectives:

- Reasonable bandwidth
- Good transient response
- Good disturbance rejection to overcome motor friction

✠ Preparation ✠

Readings:

From Franklin *et al.*, 6th or 7th Edition,

▷ Frequency Response and Bode plots Sec. 6.1

▷ Stability Margins Sec. 6.4

▷ Lead Compensation Sec. 6.7.2

Prelab:

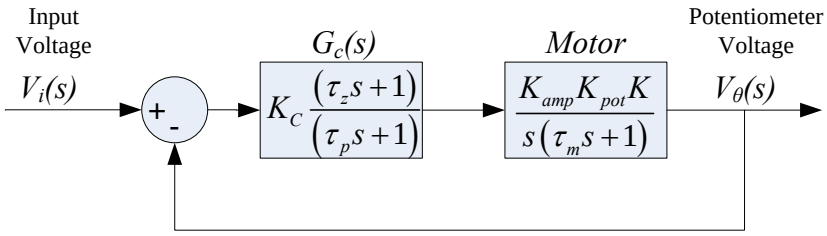


Figure 6.1: DC motor lead controller system.

1. Design an all-integrator block diagram to implement the lead compensator $G_c(s)$. Try to have one gain block per gain parameter (K_c, τ_z, and τ_p). No credit will be given for designs using derivative blocks.

2. Consider the motor controller of Figure 6.1 using the proportional control $G_c(s) = K_c$. Develop a Bode plot of the resulting loop transfer function

$$G_c(s)G_{motor}(s) = \frac{V_\theta(s)}{V_i(s)} = K_c \frac{1}{s} \frac{K_{pot}K_{amp}K}{\tau_m s + 1} \tag{6.2}$$

   Use $K_{\text{pot}} = 10/(2\pi)$, $K = 18$, $\tau_m = 2/10$, $K_{\text{amp}} = 2.4$ and $K_c = 1$.

   To obtain a short rise time, the closed loop bandwidth must be correspondingly large. A good rule of thumb for design is to approximate the closed-loop bandwidth by the open-loop crossover frequency since we typically have $\omega_c \leqslant \omega_{\text{bw}} \leqslant 2\omega_c$. We will use this heuristic to approximate the closed-loop bandwidth in our control designs. Using your Bode plot obtained above, determine the crossover frequency ω_c of the system when no compensation is used (i.e. $K_c = 1$), and find the phase margin. The MATLAB command **margin** may be used.

   Note: ω_{bw} is not necessarily the same as ω_c! The bandwidth is defined as in Figure 6.5 of Franklin. Its relationship to the damping ratio can be seen in Figure 6.3(a) of Franklin.

   Now determine the necessary value of K_c to achieve $\omega_c = 55$ rad/s. Plot the magnitude and phase response of this compensated open loop transfer function, and determine the phase margin (PM) of your design. For a second order system, as the PM approaches 0°, the damping ratio ζ is reduced and the system response becomes more oscillatory. Do you notice a decrease in the phase margin as you increase the gain?

   To improve the PM you will now design a lead compensator.

3. Obtain a Bode plot for the lead compensator with transfer function

$$G_c(s) = K_c \left(\frac{0.1s + 1}{0.01s + 1} \right) \tag{6.3}$$

You may take $K_c = 1$. Note that the phase plot has a positive bulge in the frequency range 3 rad/s $\leqslant \omega \leqslant$ 400 rad/s. When a lead compensator such as this is cascaded with the motor, their phase responses add to increase the overall phase in this frequency band. In your design, you will need to choose the size and frequency of the center of this phase bulge.

4. **High DC Gain compensator**
A lead compensator plus motor gives the open-loop transfer function:

$$G_c(s)G_{motor}(s) = K_c\left(\frac{\tau_z s + 1}{\tau_p s + 1}\right)\left(\frac{K_{\text{pot}}K_{\text{amp}}K}{s(\tau_m s + 1)}\right) \tag{6.4}$$

In this section, you will design a lead compensator with specifications: at most 15% overshoot ($M_p \leqslant 0.15$) and 30 ms rise time ($t_r \leqslant$ 30ms). Use the formulas in the Lab 1 Report template to get the necessary ζ and ω_n, which will give you a starting point for the desired pole locations. These pole location specifications for an ideal second-order system can be translated to frequency specifications on an open-loop system—see the following equations from Franklin *et al*:

$$G(s) = \frac{\omega_n^2}{s(s + 2\zeta\omega_n)} \tag{6.5}$$

$$T(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{6.6}$$

These are equations (6.29) and (6.30) in Franklin *et al* 7$^{\text{th}}$ Ed. Make sure to read pages 348 through 353 (334 through 337 in 6$^{\text{th}}$ Ed.) for context to these equations. Of the transfer functions Eq. 6.5 and Eq. 6.6, which one is the ideal 2nd order open-loop transfer function and which one is the ideal 2nd order closed-loop transfer function?

Find the resulting PM and ω_c. This will serve as a starting point for your design. Feel free to use MATLAB command **margin** for this.

Using the "Design Procedure for Lead Compensation" (see page 370 in 7$^{\text{th}}$ Ed. or page 416 in 8$^{\text{th}}$ Ed), design G_c to obtain this cross-over frequency and phase margin. To avoid poor response or control saturation, keep the DC gain in the range $2 \leqslant K_c \leqslant 4$. To prevent noise amplification keep your poles and zeros closer to the origin than 500 rad/s. As before, we are interested in the magnitude and phase characteristics of the open-loop response to determine the closed-loop response, using the feedback configuration illustrated in Fig. 6.1.

Note that higher phase margin gives lower overshoot, and higher cross-over frequency gives lower rise time, so it is fine if your design exceeds the phase margin and cross-over frequency specifications.

Give the resulting closed loop system, and plot its step response. Check that you met the time-domain specifications and adjust your design if necessary. Document the choices you made for each iteration. Also obtain a root locus plot of your final design with K_c varied from 0 to ∞, and on the root locus mark the closed loop pole locations for your design.

5. **Low DC Gain compensator**
   Consider a lead controller with transfer function

$$G_c(s) = 0.40\frac{s/2.3 + 1}{s/92 + 1} \tag{6.7}$$

   Verify that this controller meets the phase margin and bandwidth requirements. Plot the step response of the closed loop system, and verify the time domain specifications also. Note however, that the DC gain of this controller is only 0.40, which should be much smaller than the compensator that you designed. Theoretically both designs should produce a good step response, and since we have a type 1 system, we should obtain zero steady state error. However, this argument ignores the effects of disturbances: In practice we will find that a larger DC gain is required to overcome the effect of Coulomb friction.

Laboratory Exercise

I. Simulation

In the following exercises, the lead compensators studied in the prelab will be implemented using SIMULINK and the C2000 DAQ Board. Refer to the section in Lab 5 detailing this setup for more information. But before implementing the controllers, you will first study, in simulation, one of the side effects of real world implementation: saturation.

1. Create a SIMULINK simulation model of Figure 6.1; use the control block diagram and the gains in step (e) of the prelab. Use a **To Workspace** block to store the step response as "V", and set the "Save Format" to `Array`. In the simulation parameters, use **ODE45** for the best results; to smooth out the curve, set **Max step size** to `0.001`. Also, change the stop time to shortly after the transient response finishes (a couple seconds).

2. Lead compensators often produce relatively large control signals; however, the DACs used in this lab saturate at $\pm$10V. As a result, the control effort produced by SIMULINK may not match the desired signal; and correspondingly the system response may not be as expected. In order to simulate this effect of the DAC, add a **Saturation** block between the lead controller and the motor plant.

3. First simulate the system with saturation disabled; set the **Upper limit** to `inf` and the **Lower limit** to `-inf`. Set the step input to 0.5V, and run the simulation. In the MATLAB workspace, set `V05=V; time05=tout;` to save the results. Next set the step to 5V, simulate, and save with `V5=V; time5=tout;`. Finally, set the step to 50V, simulate, and save with `V50=V; time50=tout;`.

4. Now simulate the system with the saturation modeling; set the limits to $\pm$10V. Again simulate with 0.5V, 5V and 50V steps; this time, save the data as `V05sat`, `V5sat` and `V50sat`.

5. Make one plot overlaying *V05* and *V05sat*, another plot overlaying *V5* and *V5sat*, and another plot overlaying *V50* and *V50sat*. Print these plots for the report.

6. Save this model so you have it for the report.

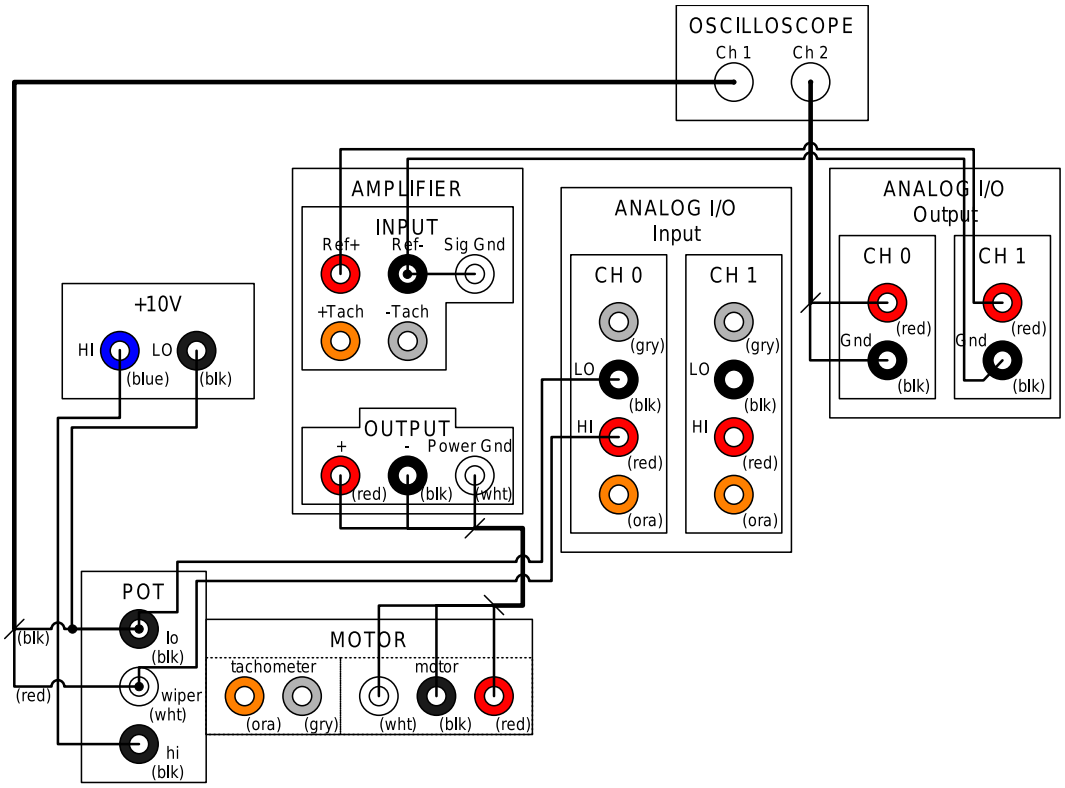
   TA check: Please stop and wait for the TA to check your work.

II. Step Response with low DC gain lead controller

The objective of this exercise is to study the step response of the motor with the low DC gain controller from Equation 6.7 in part (e) of the prelab. Since this is a type 1 system, in the absence of disturbances the steady state error will be zero. In this experiment you will see that this compensator is not able to overcome the Coulomb friction of the motor. This will result in a large steady state error in the step response.

1. Make sure power to the patch panel is off. Turn on the oscilloscope.

2. Connect the potentiometer to the motor assembly.



3. Wire up the circuit shown above.

4. Connect the push button to the **Amp Inhibit** on the patch panel.

5. Oscilloscope settings:
   (e) Channel 1 and Channel 2: Both should be On, and should be set on D.C. Coupling, 500mV/div. Place the ground lines () one division from the bottom of the screen
   (e) **Main / Delayed** button set to **Normal mode**.
   (e) **Timebase** = 50 ms.
   (e) **Trigger Mode** = Normal.
   (e) **Trigger Edge Level** = 4 V.
   (e) **Trigger Edge Source** = Ch 1
   (e) **Waveform Acquire** → Acq. Mode → High Resolution
6. Verify that the C2000 DAQBoard is powered on and it's USB cable is plugged into your bench's Windows 11 PC.
7. Copy the Lab 6 starter file provided for you. In Windows Explorer, navigate to the folder `N:\labs\ECE486`. Copy the folder **Lab6** into your C drive folder (should be located at `C:\matlab\ECE486\NETIDs`).
8. In MATLAB, change the *Current Folder* (left-hand panel) to **C:\matlab\ECE486\NETIDs\Lab6**. Open the *Lab6_Starter.slx* SIMULINK file by double clicking on it in the MATLAB panel. In *Lab6_Starter.slx* you need to select the correct COM port for the C2000 DAQBoard. In Windows 11 Device Manager check the COM port of "XDS100 Class USB Serial Port". Then in *Lab6_Starter.slx* go to "Hardware Settings" and under "Target Hardware Resources" change "External Mode" to use the COM port assigned in Device Manager and save the file "Ctrl-S".
9. In *Lab6_Starter.slx*, connect ports to corresponding signals to create a SIMULINK model that implements Figure 6.1. You may use a **Transfer Fcn** block for the controller. Set the gains to match the low-DC gain controller from part (e) of the prelab.
10. Turn on the patch panel power and amplifier.
11. When everything else is ready, navigate to the HARDWARE tab in SIMULINK and click the **Monitor & Tune** button. Wait until you see time increasing in the bottom right corner of the SIMULINK window before continuing.
12. When time starts progressing, click the "Start DAC / Stop DAC" Simulink button to enable the DAC outputs and then press and hold the Amp Inhibit button to enable the motor amplifier. (Remember that the C2000 DAQ Board's red LED indicates that your controller is running and the blue LED indicates that the "Start DAC / Stop DAC" Simulink button was pressed to enable the DAC outputs.) Explore the effects of disturbances on this controller. In between transitions of the square wave, physically rotate the motor shaft with your fingers and observe if the controller is able to bring the motor back to its previous position. Apply a disturbance torque

with your fingers. Lightly press on the motor shaft to increase its friction. Record your observations. This control is rather loose because of its low DC gain.

13. Hold the Amp Inhibit button until a good step response appears on the scope. Press the oscilloscope **Stop** button to store the display.

14. Using the oscilloscope's built-in measurement functions, measure the output overshoot, rise time. As for settling time, and the steady state error, use scope cursor to manually measure them. Does the position of the flywheel before the step make a difference?

    M_p: _____ t_r: _____ t_s: _____ e_{ss}: _____

15. Press **Run** on the oscilloscope to resume data capture.

III. Step Response with High DC gain compensation

In the previous exercise, you saw that the controller was weak and unable to overcome disturbances. We shall see in this exercise that a compensator with greater DC gain is useful in overcoming the effect of disturbances.

1. Retain all the connections that you had in the previous section.

2. Change the controller to correspond to your design from prelab part (d). (Rebuild the model by again selecting **Monitor & Tune**.

3. Explore the effects of disturbances on this controller. In between transitions of the square wave, physically rotate the motor shaft with your fingers and observe if the controller is able to bring the motor back to its previous position. Apply a disturbance torque with your fingers. Lightly press on the motor shaft to increase its friction. Record your observations. The control should feel much *tighter* than before, indicating a higher DC gain.

4. Observe the step response and measure the overshoot, rise time, settling time, and steady-state error.

    M_p: _____ t_r: _____ t_s: _____ e_{ss}: _____

IV. Frequency Response of the High DC Gain Design

One benefit of frequency design is that the frequency response can generally be found experimentally, without the need for a mathematical system model. Design may be done directly on a Bode plot, or the plot may be used to extract poles and zeros and thus create a system model. To obtain the frequency response, input a sinusoidal control signal and measure the output. In this experiment, we use the Dynamic Signal Analyzer (DSA) to automate the process. After it has collected sufficient data, the DSA will numerically fit poles and zeros to the frequency data, thus creating a system model.

Because the potentiometer wraps every 2π radians, it would be extremely difficult to use with the DSA. Thus, we will obtain the velocity transfer function (6.8) using the tachometer and convert it to find the position transfer function (6.9).

$$\frac{V_{\text{tach}}}{V_i}(s) = \frac{K_c(\tau_z s + 1)}{\tau_p s + 1} \frac{K K_{\text{tach}} K_{\text{amp}}}{\tau_m s + 1} \tag{6.8}$$

$$\frac{V_\theta}{V_i}(s) = \frac{K_c(\tau_z s + 1)}{\tau_p s + 1} \frac{K}{\tau_m s + 1} \frac{K_{\text{pot}} K_{\text{amp}}}{s} \text{ (open-loop TF)} \tag{6.9}$$

Procedure:

- Turn off the motor amplifier. Turn on the DSA.
- On the oscilloscope, change the **Timebase** to 500 ms. Switch to **ROLL** mode (**Main/Delayed** button). The other scope settings are the same as before.
- Be sure the parameter settings are correct for the high DC gain controller.
- Disconnect the potentiometer from the motor assembly (not only the feedback loop in the simulink file has to be disabled but also the potentiometer on the left of the motor has to be *physically* detached).
- In the SIMULINK model, remove the feedback loop (disconnect ADC 1 and remove the sum block). Use the high DC gain controller from prelab part (d). Set the *Stop time* to a large value less than or equal to `600`.

Wire the bench according to Figure 6.2. Do not disconnect the trigger cable attached to channel 1 of the DSA.

Have the TA check your wiring and SIMULINK model.

TA check: Please stop and wait for the TA to check your work.

We will use Keysight VEE to control the DSA. Open the file `N:\labs\ECE486\dsacontrol.vxe`. This will use the DSA to get samples of the frequency response from Channel 1 to Channel 2.
*Note: Keysight VEE displays files of type *.vee by default. You must change this box to VEE RunTime Programs (*.vxe)*

- Change the destination of your outputted data by clicking the button next to the *To File:* text (by default this button probably says `w:\fresp.m`). Navigate to your C drive folder (`C:\matlab\ECE486\NETIDs`) in the menu that opens and click **Open**.
- Press the **Initialize** button in VEE.
- Back in SIMULINK click the **Monitor & Tune** button. As before, wait until time starts increasing in the bottom right corner of the window to know your SIMULINK model is running. Also click the "Start DAC / Stop DAC" SIMULINK button to enable the DAC outputs.

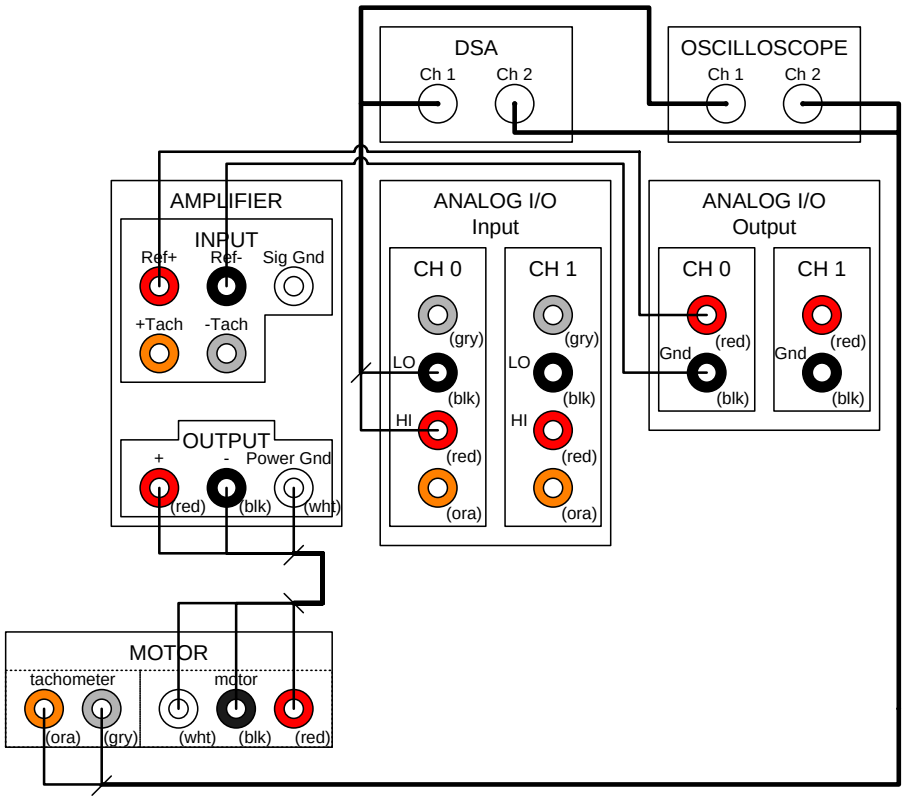


Figure 6.2: Wiring diagram for frequency response.

- Turn on the motor amplifier.
- Push the $\frac{1}{4}$ inch plug into the pushbutton jack.
- Press **Start Measurement** in VEE.
- Enter 0.5 for the start frequency, 20 for the stop frequency, 15 for the number of points, 0.2 for the (source) level, and 1 for the (source) offset voltage.

Watch the waveforms on the DSA and the scope. When the sine sweep is completed (takes about 3-5 minutes), you will save your data using VEE. The DSA displays "**Sweep Complete**" when finished.

- Pull out the $\frac{1}{4}$ inch plug from the pushbutton jack.
- Press **Collect Data**. The frequency response samples will be saved into file `fresp.m`. The first column of this file is the frequency points in rad/s and the second and third columns are the real and imaginary parts of the frequency response.
- Press **Get Fit**. Enter the frequency range for the fit—usually the same as the sweep, unless the high frequency data looks corrupted (jagged or sharp turns). Enter 2 for the number of poles and 1 for the number of zeros. The DSA will then find the coefficients for (6.10).

$$\frac{K_{\text{DSA}}(\bar{s}[\text{Hz}] - z[\text{Hz}])}{(\bar{s}[\text{Hz}] - p_1[\text{Hz}])(\bar{s}[\text{Hz}] - p_2[\text{Hz}])} \tag{6.10}$$

with K_{DSA}, z, p_1, and p_2 in Hertz as indicated.

K_{DSA}: _____ z: _____[Hz] p_1: _____[Hz] p_2: _____[Hz]

Note: Since our analysis uses s[rad/s], unit conversion is required. The easiest way to do this correctly is to convert $\bar{s}$ in-place as shown in (6.11); incorrect conversion can change the system gain by a multiple of 2π.

$$\frac{K_{\text{DSA}}(\frac{s[\text{rad/s}]}{2\pi} - z[\text{Hz}])}{(\frac{s[\text{rad/s}]}{2\pi} - p_1[\text{Hz}])(\frac{s[\text{rad/s}]}{2\pi} - p_2[\text{Hz}])} \tag{6.11}$$

In addition to converting this transfer function, you will be using the raw frequency response data in the lab report. To load in the data, you might do something like this:

```
cd U:
fresp
omega=dsa(:,1);
data=dsa(:,2)+i*dsa(:,3);
```

Making a Bode plot involves using **subplot** to separate the magnitude and phase plots. Use **abs** in MATLAB to obtain the magnitude and **phase** to obtain the phase. Do not forget to scale the magnitude to dB (see **log10**) and the phase to degrees.

Report

Download and complete the report template available on the course lab website, ensuring you answer all questions.

Appendix A

The Analog Computer - Theory and Practice

I. Introduction

Many problems encountered in scientific and engineering studies require the solution of mathematical equations or sets of equations which describe the behavior of physical systems. For complex systems, the solution of the equations may be difficult or tedious by means of pencil and paper. An analog computer may be used to overcome this difficulty by providing the engineer with an electronic model which permits the rapid solution of mathematical equations. The electronic analog computer makes possible the building of an electrical model of a physical system in which voltages will behave with time in a similar way to the actual variables of interest in the actual network or system.

The analog computer is basically a set of building blocks, each able to perform specific mathematical operations, such as addition, subtraction, integration, multiplication, etc., on voltages. By constructing an appropriately connected group of such blocks, an electronic model can be created in which the voltages at the outputs of the various blocks obey the relationships given in the mathematical description of the physical system. As our interest is primarily in dynamic systems, the describing equations are usually differential equations with time as the independent variable. By applying appropriate initial conditions and forcing functions to the electrical model, its behavior will be the same as the original physical system. By using an oscilloscope or other device, the voltages at various points in the model can be measured and compared to time or to each other.

The purpose of this tutorial is to introduce the student to a few methods of interconnecting basic computing units. This is known as analog programming. The tutorial also provides familiarity with the GP-6 analog computer. For a more extensive treatment on the use of analog computers, the student should refer to specialized texts on analog computers. Several are available in the Engineering Library.

II. Analog Computer Functions

We will be concerned with three basic operations:

1. Addition.
2. Multiplication by a constant.
3. Integration.

One computing unit which is capable of performing all three of these operations is a high-gain D.C. amplifier known as the operational amplifier. The symbol most commonly used for the operational amplifier is shown conceptually in Figure 1.

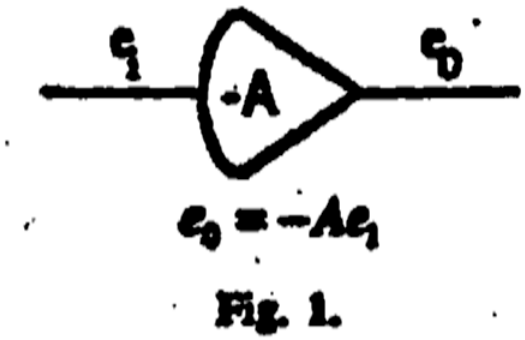


Figure A.1: Operational amplifier symbol.

Text equivalent for Figure 1. The figure shows an operational amplifier drawn as a triangle. The input voltage is labeled e_i and enters the left side of the amplifier. The output voltage is labeled e_o and leaves the right side. The gain inside the amplifier is labeled $-A$, and the input-output relation is

$$e_o = -Ae_i. \tag{A.1}$$

Typically, A is very large, and is usually assumed to approach infinity when the operational amplifier is an element of a larger circuit, as will be illustrated below.

a. Addition

Consider the operational amplifier interconnected with input and feedback resistors as shown in Figure 2. Summing the currents at the node G, assuming that the D.C. amplifier does not draw any input current, we have

$$\frac{e_1 - e_g}{R_1} + \frac{e_2 - e_g}{R_2} + \frac{e_o - e_g}{R_f} = 0. \tag{A.2}$$

Since

$$e_o = -Ae_g, \qquad -e_g = \frac{e_o}{A}, \tag{A.3}$$

we obtain

$$\frac{e_1}{R_1} + \frac{e_2}{R_2} + \frac{e_o}{R_f} - \left(\frac{e_o}{A}\right)\left(\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_f}\right) = 0. \tag{A.4}$$

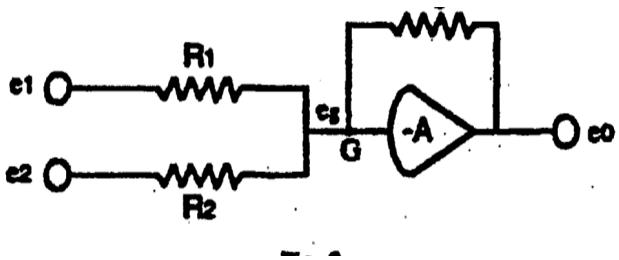


Figure A.2: Summing amplifier circuit.

Text equivalent for Figure 2. The figure shows two input voltages, e_1 and e_2, applied through input resistors R_1 and R_2 to the common summing node G. The node voltage is labeled e_g. A feedback resistor R_f connects the amplifier output back to the summing node. The amplifier block is labeled $-A$, and the output voltage is labeled e_o. The circuit is an inverting summing amplifier.

If A approaches infinity, the last term above may be dropped, and

$$\frac{e_1}{R_1} + \frac{e_2}{R_2} + \frac{e_o}{R_f} = 0. \tag{A.5}$$

Therefore,

$$e_o = -R_f \left(\frac{e_1}{R_1} + \frac{e_2}{R_2} \right) = -\frac{R_f}{R_1} e_1 - \frac{R_f}{R_2} e_2. \tag{A.6}$$

Thus, the input-output relation of this amplifier circuit is solely dependent on the ratio of feedback to input resistors.

To perform simple addition, we can let $R_1 = R_2 = R_f$. Then

$$e_o = -(e_1 + e_2). \tag{A.7}$$

Note the minus sign. If $R_1 \neq R_2$, then we perform a weighted addition. This can be generalized to more than two inputs.

b. Multiplication by a Constant

With only one input to the operational amplifier, the input-output equation is

$$e_o = -\frac{R_f}{R_1} e_i. \tag{A.8}$$

Thus, multiplication by a negative constant of any magnitude can be performed by appropriately choosing R_1 and R_f.

If the multiplicative constant is less than one, there is a simpler way to design the multiplier using a potentiometer, as shown in Figure 3. In the case of the potentiometer, there is no sign inversion.

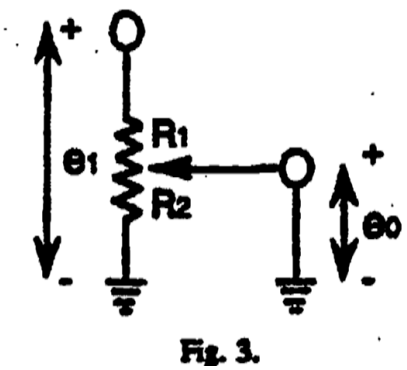


Figure A.3: Potentiometer multiplier.

Text equivalent for Figure 3. The figure shows an input voltage e_i applied across a potentiometer made of two series resistances, R_1 and R_2. The output voltage e_o is taken from the wiper between the two resistive sections. The result is

$$e_o = \frac{R_2}{R_1 + R_2} e_i = A e_i, \qquad 0 \le A \le 1. \tag{A.9}$$

c. Integration

If we connect the operational amplifier as shown in Figure 4, with a capacitor in the feedback path, the current equation at node G is

$$\frac{e_1 - e_g}{R_1} + \frac{e_2 - e_g}{R_2} + C\frac{d(e_o - e_g)}{dt} = 0. \tag{A.10}$$

Substituting $e_g = -e_o/A$ into the last equation gives

$$\frac{e_1 + e_o/A}{R_1} + \frac{e_2 + e_o/A}{R_2} + C\left(1 + \frac{1}{A}\right)\frac{de_o}{dt} = 0. \tag{A.11}$$

Since A approaches infinity, the last equation becomes

$$C\frac{de_o}{dt} = -\frac{e_1}{R_1} - \frac{e_2}{R_2}. \tag{A.12}$$

Integrating both sides with respect to t, we obtain

$$e_o(t) = -\left[\frac{1}{R_1 C}\int_0^t e_1(\tau)\,d\tau + \frac{1}{R_2 C}\int_0^t e_2(\tau)\,d\tau\right] + e_o(0). \tag{A.13}$$

Text equivalent for Figure 4. The figure shows two input voltages, e_1 and e_2, applied through input resistors R_1 and R_2 to summing node G. The amplifier output e_o is connected back to the summing node through a capacitor C instead of a feedback resistor.

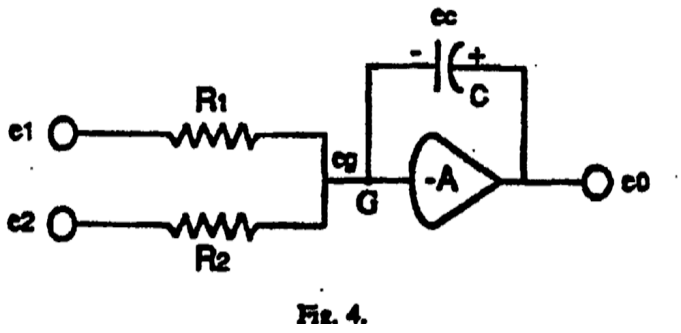


Fig. 4.

Figure A.4: Integrating amplifier circuit.

A voltage source is shown for applying the initial capacitor voltage. The circuit performs a weighted integration of the input voltages and includes an initial condition term.

Thus, we can integrate and sum using the same technique as in Figure 4. Since $e_g(0) = e_o(0) + e_c(0)$ and e_g is very small, less than approximately 1 mV, $e_o(0) = e_c(0)$. The integration constant can therefore be physically obtained by applying a voltage to the capacitor of magnitude $e_o(0)$. Because of the nature of the charging circuit for this voltage, a negative voltage must be applied to the initial-condition terminal. This convention has been adopted into the block-diagram convention described below.

It should be noted that many other input-output relations can be obtained through use of the operational amplifier and other devices, including multiplying two variables together, variable switching, nonlinear transfer functions, and other operations.

III. Analog Programming Diagrams

To make drawing analog computer diagrams easier, the circuit schematic has been replaced by an appropriate symbol.

Text equivalent for Figure 5. The figure compares circuit schematics with their analog-computer block-diagram symbols.

1. A summing amplifier has input a applied through a $1\,\mathrm{k}\Omega$ resistor and input b applied through a $2\,\mathrm{k}\Omega$ resistor. The feedback resistor is $10\,\mathrm{k}\Omega$. The output is

$$c = -(10a + 5b). \tag{A.14}$$

   The equivalent block symbol is a triangular summing block with input gains 10 and 5 and output c.

2. A potentiometer multiplier takes input a and produces output $b = 0.5a$. The equivalent block symbol is a gain block labeled 0.5.

3. An integrating amplifier has input a applied through a $10\,\mathrm{k}\Omega$ resistor and input b applied through a $100\,\mathrm{k}\Omega$ resistor. The feedback element is a $10\,\mu\mathrm{F}$ capacitor. A $5\,\mathrm{V}$

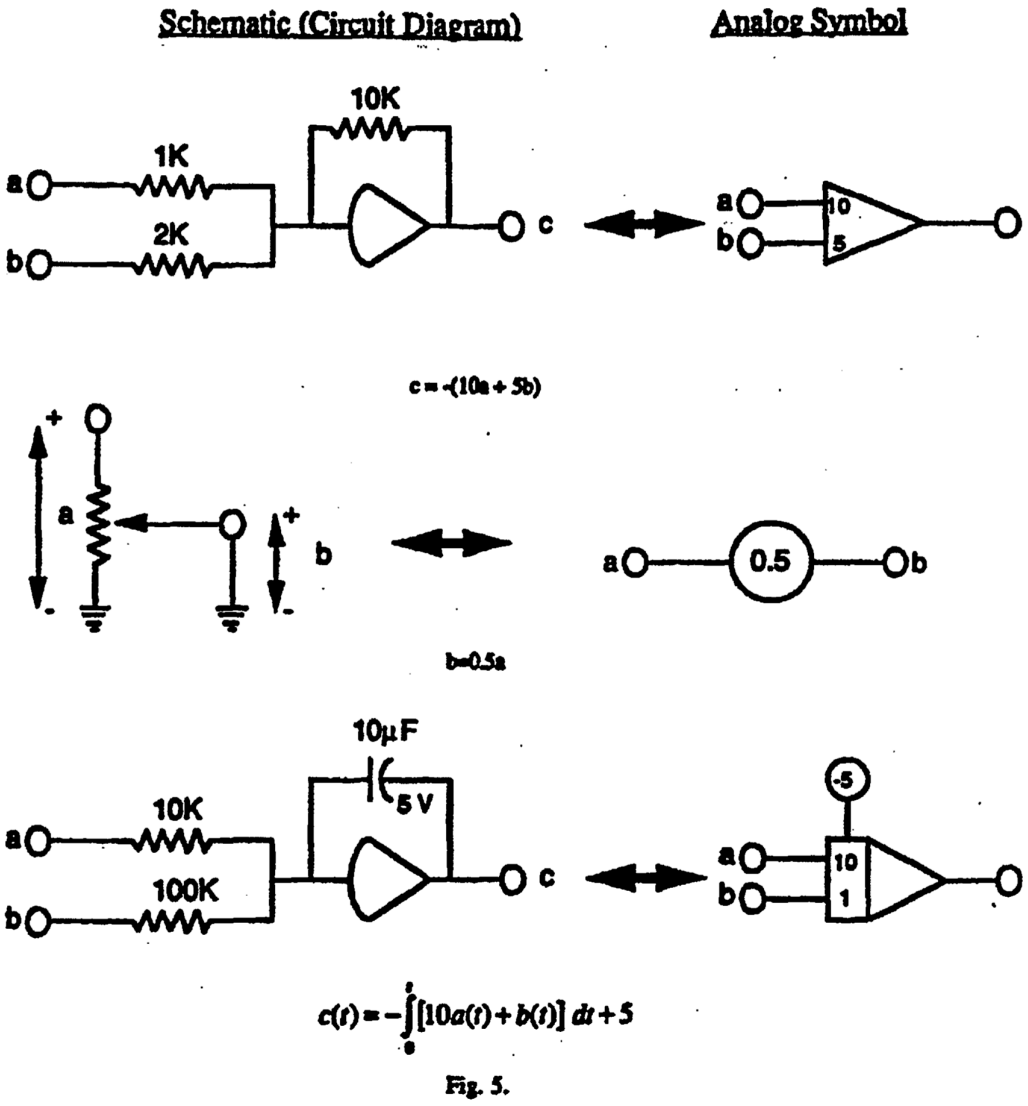


Fig. 5.

Figure A.5: Analog programming diagrams.

initial-condition source is shown. The output is

$$c(t) = -\int_0^t [10a(\tau) + b(\tau)]\, d\tau + 5. \tag{A.15}$$

The equivalent block symbol is an integrator with input gains 10 and 1 and an initial condition of 5.

IV. Going from the Equation to the Block Diagram

While there are many different methods of implementing a computer diagram from a mathematical equation, we shall concentrate on the following two methods:

1. Solving for the highest-order derivative from a single differential equation.
2. State-variable approach, which means solving several simultaneous first-order differential equations.

Part c will deal with a generalization of part b to the situation where there are derivatives of the input signal, or zeros, in the transfer function.

a. Solving for the Highest-Order Derivative

This method will be illustrated by the following problem.

Given:

$$\frac{d^2y(t)}{dt^2} - 3y(t) = u(t), \qquad y(0) = 5, \qquad \frac{dy(0)}{dt} = 4. \tag{A.16}$$

Problem: Draw a computer block diagram that will give the solution for, that is, has the output $y(t)$.

1. The first step is to solve for the highest-order derivative in the differential equation. Thus

$$\frac{d^2y(t)}{dt^2} = u(t) + 3y(t). \tag{A.17}$$

Now, applying the right-hand side of the last equation to the inputs of an integrator gives the situation shown in Figure 6. The output of the integrator will be $-dy(t)/dt$, since

$$-\int [u(t) + 3y(t)]\, dt = -\int \frac{d^2y(t)}{dt^2}\, dt = -\frac{dy(t)}{dt}. \tag{A.18}$$

Text equivalent for Figure 6. The figure shows an analog-computer integrator with two inputs, $u(t)$ and $3y(t)$. The two inputs are summed and integrated. The output is labeled

$$-\frac{dy(t)}{dt} = e_1(t). \tag{A.19}$$

An initial-condition terminal labeled 4 is connected to the integrator, corresponding to the initial derivative condition $dy(0)/dt = 4$.

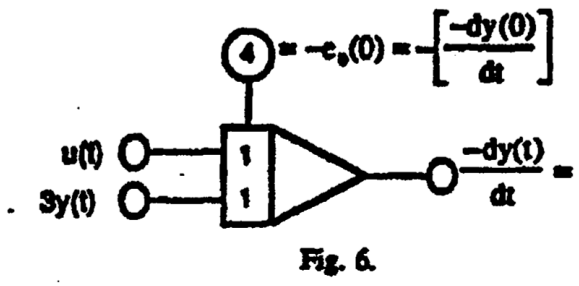


Figure A.6: First integrator block for the second-order equation.

2. Continue integrating the output of this integrator until $y(t)$ is obtained, applying initial conditions to the appropriate integrators along the way.

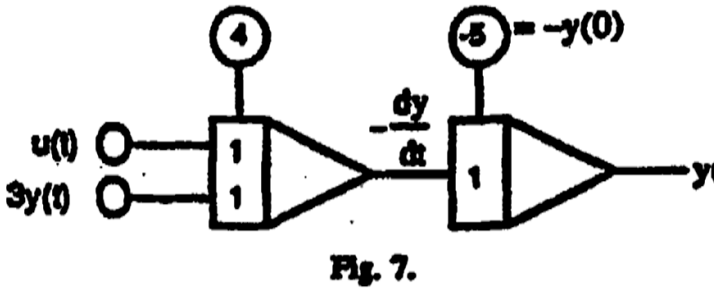


Figure A.7: Two-integrator chain used to obtain y of t.

Text equivalent for Figure 7. The figure shows the first integrator from Figure 6 followed by a second integrator. The first integrator receives $u(t)$ and $3y(t)$ and produces $-dy(t)/dt$. The second integrator receives $-dy(t)/dt$ and produces $y(t)$. The first integrator uses the derivative initial condition $dy(0)/dt = 4$, and the second integrator uses the position initial condition $y(0) = 5$.

3. Form all interconnections necessary to complete the diagram. Figure 8 gives the computer block diagram after the needed feedback connection has been made.

Text equivalent for Figure 8. The figure shows $u(t)$ applied to the first integrator. The output $y(t)$ from the second integrator is fed back through a gain of 3 and added to $u(t)$ at the input of the first integrator. The first integrator produces $-dy(t)/dt$, and the second integrator produces $y(t)$.

4. It is possible to simplify the diagram in order to obtain one with the lowest number of components. The important thing to keep in mind is that if all the loop-gain products are not changed, the same transfer function from the input to the output will result. This can be proven by Mason's rule. Figure 9 shows the simplified block diagram and the input source which supplies the unit-step input.

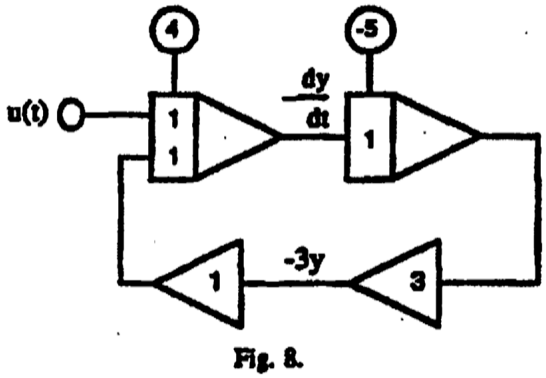


Figure A.8: Completed two-integrator block diagram.

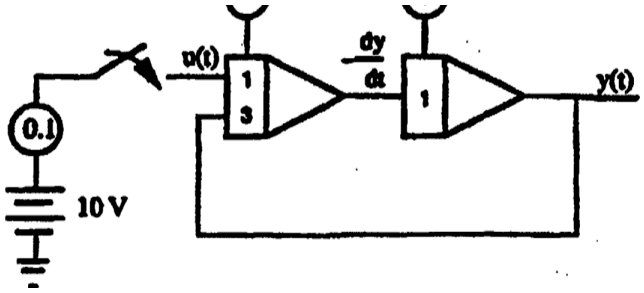


Figure A.9: Simplified block diagram with unit-step source.

Text equivalent for Figure 9. The figure shows a unit-step source generated from a voltage source and switching element. The step input $u(t)$ feeds the simplified two-integrator analog-computer realization. The feedback path from $y(t)$ returns through gain 3 to the first integrator input. The two integrators carry the initial conditions corresponding to $dy(0)/dt = 4$ and $y(0) = 5$.

As a second example, consider

$$\frac{d^2x(t)}{dt^2} + 1.5\frac{dx(t)}{dt} + 4x(t) = f(t), \qquad x(0) = 4, \qquad \frac{dx(0)}{dt} = -6. \tag{A.20}$$

Following the method outlined above, the computer diagram shown in Figure 10 may be obtained. Verify this.

Text equivalent for Figure 10. The figure shows an analog-computer realization for the equation

$$x''(t) + 1.5x'(t) + 4x(t) = f(t). \tag{A.21}$$

The input $f(t)$ is combined with feedback terms from the derivative and position states so that the first integrator input represents $x''(t)$. The first integrator produces a velocity-related signal, and the second integrator produces the position output $x(t)$. The diagram includes initial conditions corresponding to $x(0) = 4$ and $x'(0) = -6$.

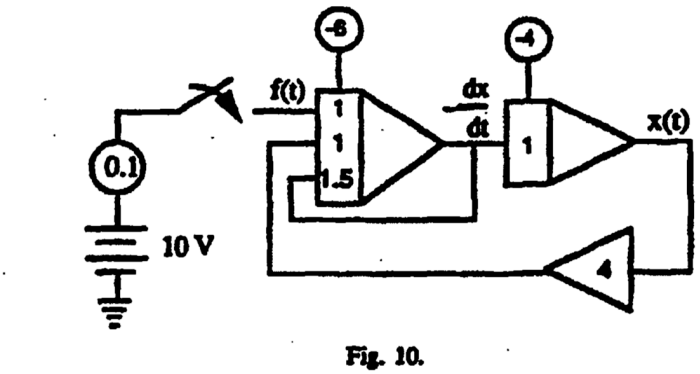


Figure A.10: Computer diagram for the second second-order example.

b. State-Variable Approach to Computer Programming

Since state equations are first-order differential equations, computer implementation may be performed for each equation using the method described in the previous section.

Consider the state equations

$$\frac{dx_1(t)}{dt} = 2x_1 - 3x_2 + f(t), \qquad x_1(0) = 2, \tag{A.22}$$

$$\frac{dx_2(t)}{dt} = 5x_1 + 2f(t), \qquad x_2(0) = 3, \tag{A.23}$$

with output equation

$$y(t) = x_1 + 10x_2. \tag{A.24}$$

Problem: Determine the output $y(t)$ when the input is a ramp function,

$$f(t) = tu(t). \tag{A.25}$$

First, applying each of the right-hand side terms of the state equations to an integrator gives the block diagram shown in Figure 11.

Text equivalent for Figure 11. The figure shows two integrators. The upper integrator corresponds to x_1 and has inputs $f(t)$ with gain 1, x_1 with gain 2, and $-x_2$ with gain 3. Its initial-condition terminal is labeled 2, and its output is labeled $-x_1(t)$. The lower integrator corresponds to x_2 and has inputs $f(t)$ with gain 2 and x_1 with gain 5. Its initial-condition terminal is labeled 3, and its output is labeled $-x_2(t)$.

Next, we make the necessary interconnections to satisfy the state equations, and the block diagram shown in Figure 12 results. Figure 13 gives the complete computer diagram with the input supply and the realization of the output equation.

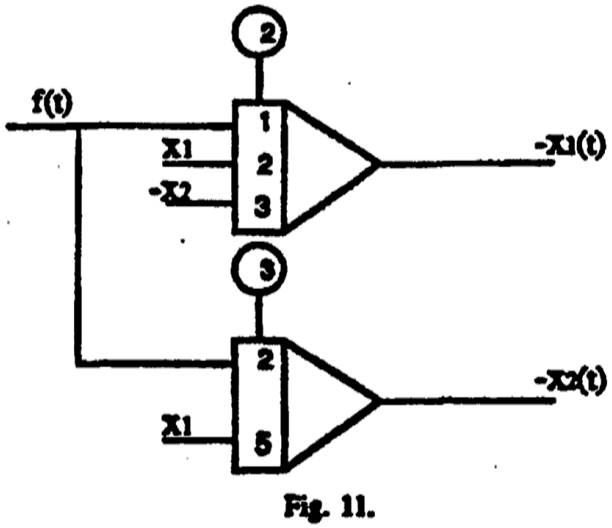


Figure A.11: Initial state-variable integrator blocks.

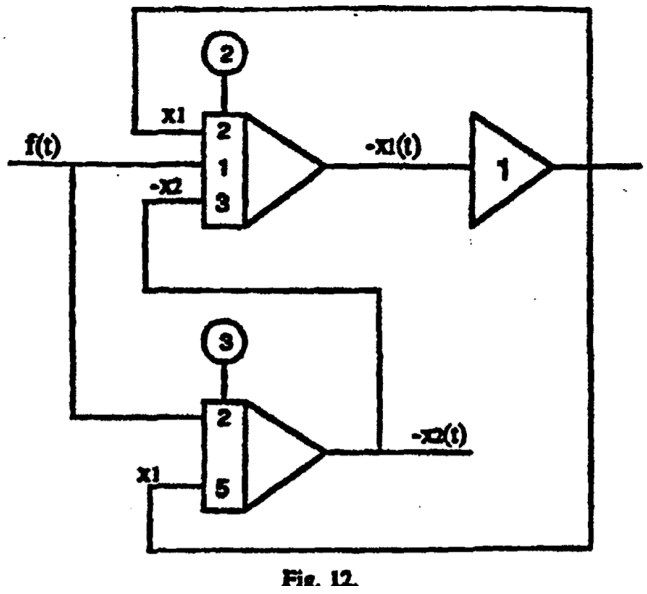


Figure A.12: State-variable interconnection diagram.

Text equivalent for Figure 12. The figure shows the two-integrator state-variable diagram after feedback interconnections have been added. The input $f(t)$ feeds both integrator input paths. The upper integrator output $-x_1(t)$ is passed through a unity inverter to generate $x_1(t)$ for use in feedback paths. The lower integrator output is $-x_2(t)$. The x_1 and $-x_2$ signals are routed back to the appropriate inputs of the upper and lower integrators so that the equations for $\dot{x}_1$ and $\dot{x}_2$ are satisfied.

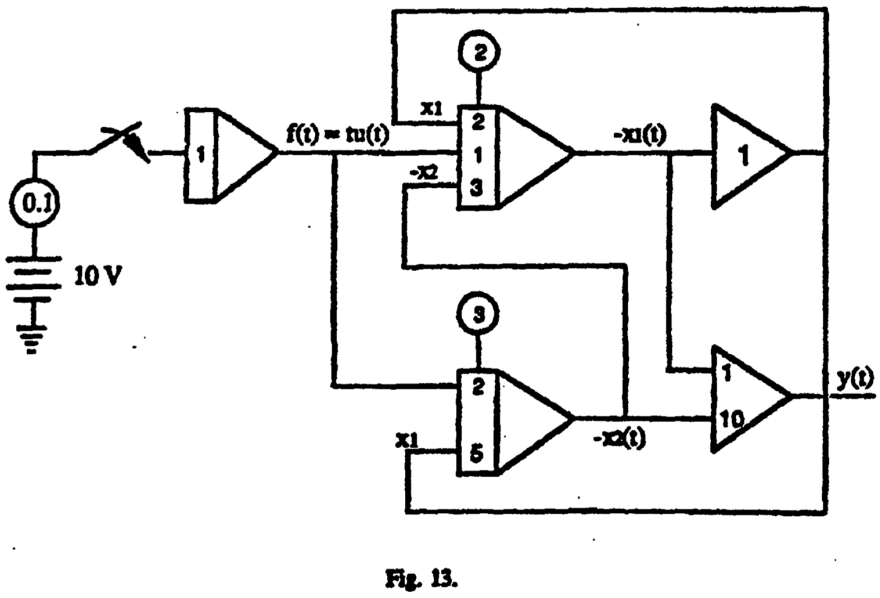


Fig. 13.

Figure A.13: Complete state-variable diagram with input supply and output realization.

Text equivalent for Figure 13. The figure adds the ramp input supply and the output realization to Figure 12. A 10 V source and potentiometer setting labeled 0.1 are used with a switch and an integrator to generate the ramp input $f(t) = tu(t)$. The ramp input enters both state integrator paths. A final summing block combines x_1 with gain 1 and x_2 with gain 10 to produce the output $y(t) = x_1 + 10x_2$.

c. Representing Systems with Zeros

When a physical system has one or more zeros, the solving-for-the-highest-derivative approach leads to difficulty when the designer actually tries to move from the equations to the block diagram. Consider a system with the transfer function

$$F(s) = \frac{Y(s)}{U(s)} = \frac{as + b}{cs + d}. \tag{A.26}$$

We can represent this as two separate transfer functions, as shown in Figure 14.

Text equivalent for Figure 14. The figure shows the input $U(s)$ passing first through a block with transfer function $1/(cs + d)$, producing an intermediate signal $Y'(s)$. The

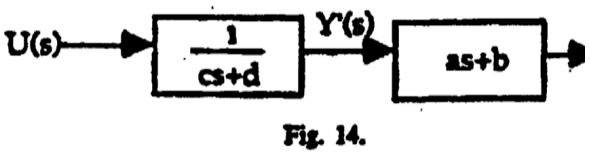


Figure A.14: Transfer-function factorization into two blocks.

signal $Y'(s)$ then passes through a second block with transfer function $as + b$, producing the final output $Y(s)$.

The transfer function from $U(s)$ to $Y'(s)$ is in standard form, and we can easily reduce this to a block diagram, as follows.

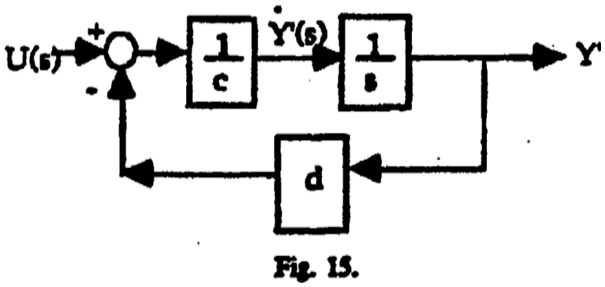


Figure A.15: Standard-form block diagram for one over c s plus d.

Text equivalent for Figure 15. The figure shows $U(s)$ entering the positive input of a summing junction. The output $Y'(s)$ is fed back through a gain block d to the negative input of the summing junction. The summing-junction output passes through a gain block $1/c$, producing $\dot{Y}'(s)$, and then through an integrator block $1/s$, producing $Y'(s)$.

Note that this is the traditional all-integrator block diagram. There is no inversion in the gain blocks or in the integrator. However, it is simple to convert the all-integrator block diagram to the analog-computer diagram.

We have labeled the derivative of $Y'(s)$. We now note that

$$Y(s) = (as + b)Y'(s) = a\dot{Y}'(s) + bY'(s). \tag{A.27}$$

Both Y' and $\dot{Y}'$ are measurable signals. We can represent the complete block diagram as shown in Figure 16.

Text equivalent for Figure 16. The figure starts with the standard realization of $Y'(s) = U(s)/(cs + d)$. The signal $\dot{Y}'(s)$ is tapped after the $1/c$ block and before the $1/s$ integrator. This derivative-related signal passes through gain a and enters the positive

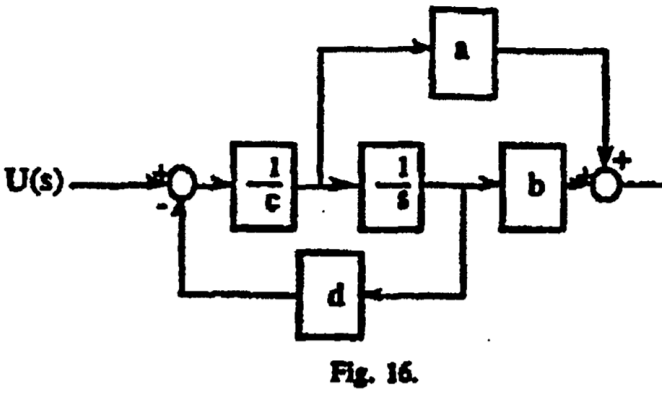


Figure A.16: Complete block diagram for a transfer function with a zero.

input of a final summing junction. The signal $Y'(s)$, taken after the integrator, passes through gain b and enters the same final summing junction. The final output is

$$Y(s) = a\dot{Y}'(s) + bY'(s). \tag{A.28}$$

The feedback path for the first part of the diagram still routes $Y'(s)$ through gain d to the negative input of the first summing junction.

This technique can be extended to systems with any number of integrators, so long as the number of zeros does not exceed the number of poles. Transfer functions that have this characteristic are known as proper transfer functions.

Appendix B

B.1 The GP-6 Analog Computer: Characteristics and Operating Instructions

The following is a transcription of the first page of Appendix B.

The GP-6 Analog Computer: Characteristics and Operating Instructions

MODE CONTROL Push Buttons:

1. **Initial Condition:** Resets integrators to initial conditions.
2. **Hold:** Stops integration at current position.
3. **Operate:** Starts integrators operating.
4. **Repetitive Operation:** Alternately resets and operates integrators.

MODE SELECTOR Switch: The MODE SELECTOR switch alternates between POT SET and OPR.

POT SET Mode.

1. The inputs of the potentiometers are set to +10 Volts.
2. The input to the digital voltmeter (DVM) is connected to the output of the potentiometers.
3. The integrators are placed in their initial condition state.

OPR Mode.

1. The inputs to the potentiometers are connected to the patch board.
2. The input to the DVM is connected to the rear terminal METER INPUT.
3. The integrators are controlled by push button selections.

Setting Coefficient Potentiometers:

1. Complete all patching so settings are made under their operating loads.
2. Position the MODE SELECTOR switch to POT SET.

3. Position the Y/POT ADDRESS to the number of the pot to be set.
4. Adjust the pot knob until the desired setting is displayed.

If an amplifier overload alarm occurs when the POT SET mode is selected, remove the condition by patching the overloaded amplifier output directly to its summing junction. (Be sure to remove the patch cord prior to running the program.)

Measurement of amplifier outputs in the OPR modes:

1. Connect the Y OUTPUT rear terminal to the METER INPUT terminal.
2. Position the MODE SELECTOR switch to the OPR position.
3. Position the Y/POT ADDRESS switch to the number of the amplifier output to be measured.

B.2 Analog Computer's Manual - GP-6: Patch Panel

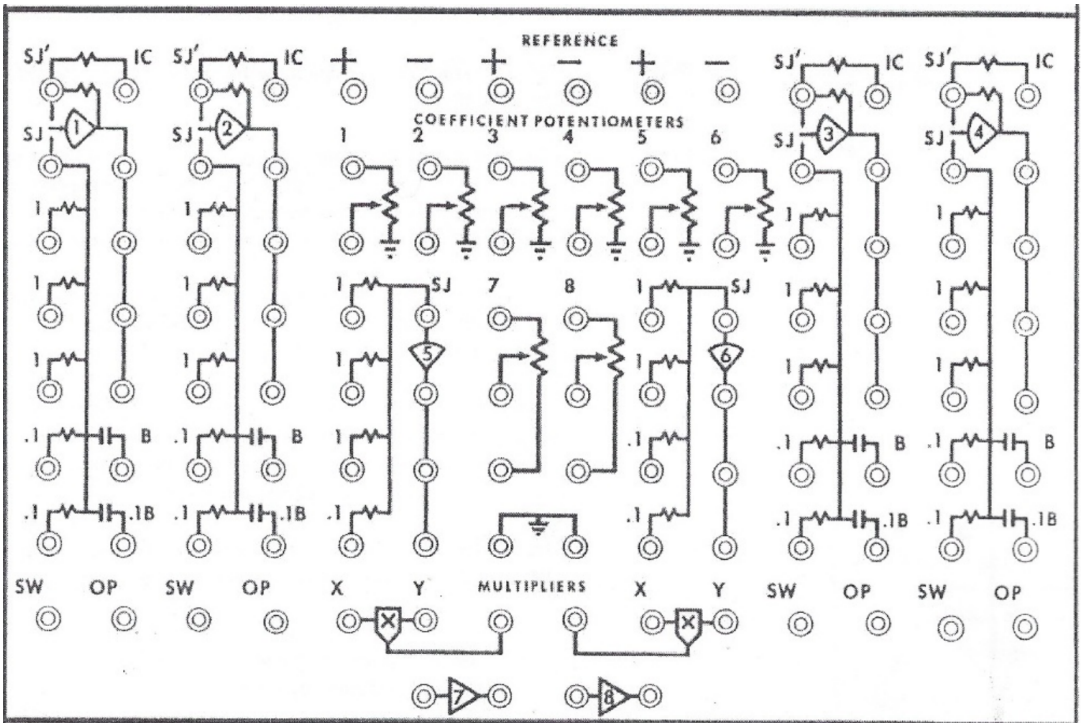


Figure B.1: GP-6 patch panel graphic from the Analog Computer's Manual - GP-6, page 7.

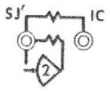
Patch panel graphics use standard analog computer programming symbols. Amplifiers 1 thru 4 are single ended, high gain amplifiers with electronic mode switches and summing resistor/integrating capacitor networks that may be programmed as summer/inverters, integrators, track/store and single pole, double throw electronic switch amplifiers. Amplifiers 4 and 5 are summer/inverters. Amplifiers 7 and 8 are inverters only. Potentiometers 1 thru 6 are grounded attenuators. Potentiometers 7 and 8 have their bottom ends open and

may be used as either voltage dividers or attenuators. Multiplier networks produce current outputs for direct connection to amplifier summing junctions, and thus may be patched as multipliers, dividers, squarers and square root extractors.

The following is an explanation of patch panel symbols:

Table B.1: Patch-panel symbols, color code, and description from the Analog Computer's Manual – GP-6.

Symbol	Color code	Description
	Red	Positive reference, considered unity (1.000) for normalized programming. (Actual amplitude is 10 volts.)
	Yellow	Negative reference.
		High gain operational amplifier.
		High gain operational amplifier with electronic switch.
		Inverter.
	Red	Amplifier output.
	Gray	The summing junctions for amplifiers 1 - 6. (Active for amplifiers 1 - 4 when a logic "1" is patched to the SW switch control jack or when there is no switch control patching.)
	Gray	Alternate summing junction for amplifiers 1 - 4. (Conducting when a logic "0" is patched to the SW switch control jack.)
	Green	Standard input summing resistor, normalized as a unity value to simplify programming. (Actual resistance is 50K ohms.)
	Green	Summing resistor input one tenth the standard value. (Actual resistance is 5K ohms.)
	Green	One end of standard integrating capacitor, normalized so that the "1" resistor and B capacitor combine to produce a one second time constant, as referred to programming time scales. (Actual capacitance is 20 ufd for the slow and .05 ufd for the fast time scales.)
	Green	Integrating capacitor that has a value one tenth the standard B capacitor. (Actual capacitance is 2 ufd for the slow and .005 ufd for the fast time scales.)

Symbol	Color code	Description
	Green	Resistor network for the SJ′ summing junction. Amplifier becomes an inverter when SJ′ is conducting. Normally used for integrator initial conditions. May also be used as the feedback and input with the SJ summing network. (See Summer patching.)
	Yellow	Attenuator: bottom end grounded; top end input and wiper output brought out to the panel – wiper indicated by the arrow.
	Yellow	Voltage divider: top and bottom end inputs and wiper brought out to the panel – wiper indicated by the arrow.
	Black	System ground.
		Multiplier network symbol.
X	Brown	One of two multiplier inputs.
Y	Brown	One of two multiplier inputs.
	Gray	Multiplier output, a current proportional to the product of “X” and “Y;” normalized so that when connected to the summing junction of an operational amplifier with a “1” resistor feedback, and with reference patched as both inputs, the amplifier output equals reference.
SW	White	Electronic switch control. Logic 0 (ground or positive voltage,) SJ′ conducts, SJ shuts off. Logic 1 (-5V or less,) SJ conducts and SJ′ shuts off. Hold logic (-2V thru -3V,) SJ shuts off and the summer resistor network is disconnected electronically from the amplifier/capacitor feedback.
OP	White	Computer’s operate bus, integrator mode logic from the central oper-push button control, patched to “SW” for normal integrator operation.

B.3 Patch Panel Operations

Each operation below follows the manual’s three-column structure: function, operation, and patching. The operation and patching columns include cropped source images from the manual page, followed by a text transcription of the formula and patching content.

Table B.2: Patch-panel operation for summer, amplifiers 1–4.

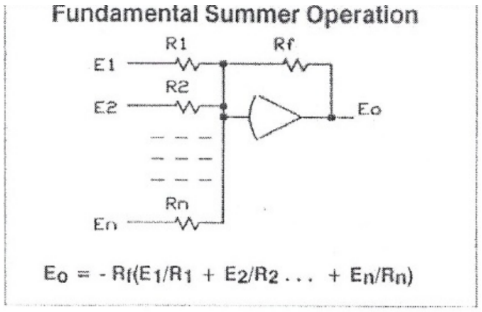
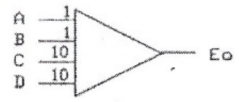
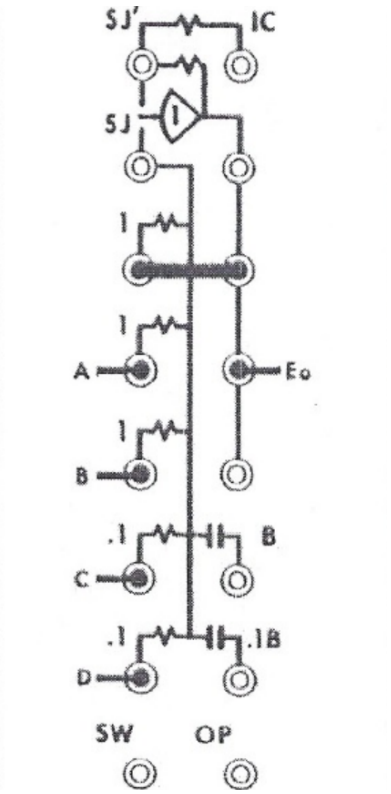
Function	Operation	Patching
SUMMER amplifiers 1–4	  Eo = - (A + B + 10C + 10D) NO PATCHING TO SWITCH CONTROL "SW" $E_o = -R_f\left(\frac{E_1}{R_1} + \frac{E_2}{R_2} + \cdots + \frac{E_n}{R_n}\right)$ $E_o = -(A + B + 10C + 10D)$ No patching to switch control "SW".	 Inputs A and B are patched to unity resistor inputs. Inputs C and D are patched to 0.1 resistor inputs. The amplifier output is labeled E_o.

Table B.3: Patch-panel operation for summer, amplifiers 1–4 using IC networks.

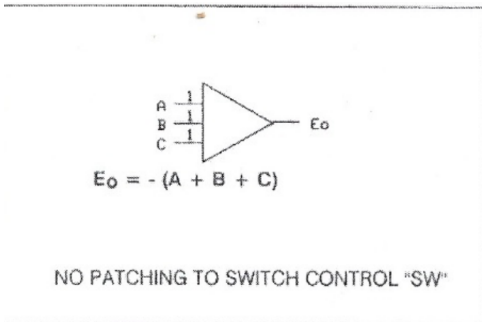
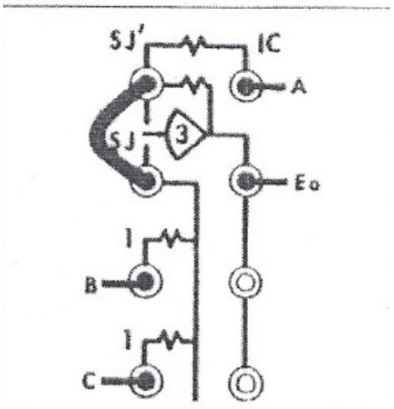
Function	Operation	Patching
SUMMER amplifiers 1–4, using IC networks	 $E_o = -(A + B + C)$ No patching to switch control "SW".	 Inputs A, B, and C are patched into the IC-network summer arrangement. The amplifier output is labeled E_o.

Table B.4: Patch-panel operation for summer, amplifiers 5 and 6.

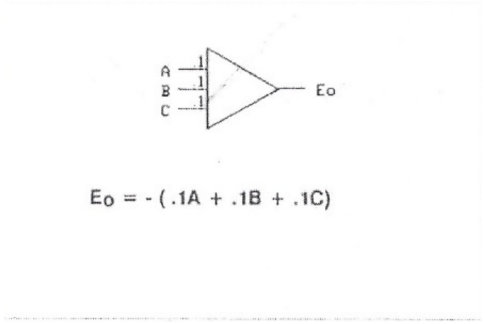
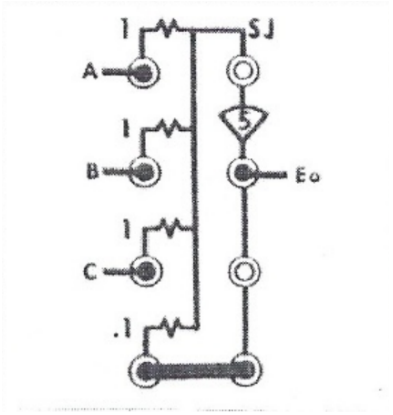
Function	Operation	Patching
SUMMER amplifiers 5 and 6	 $E_o = -(.1A + .1B + .1C)$	 Inputs A, B, and C are patched into the 0.1 inputs of the amplifier section. The amplifier output is labeled E_o.

Table B.5: Patch-panel operation for inverter, amplifiers 7 and 8.

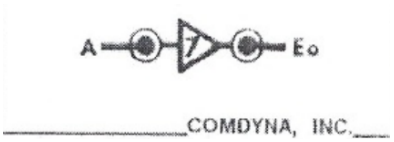
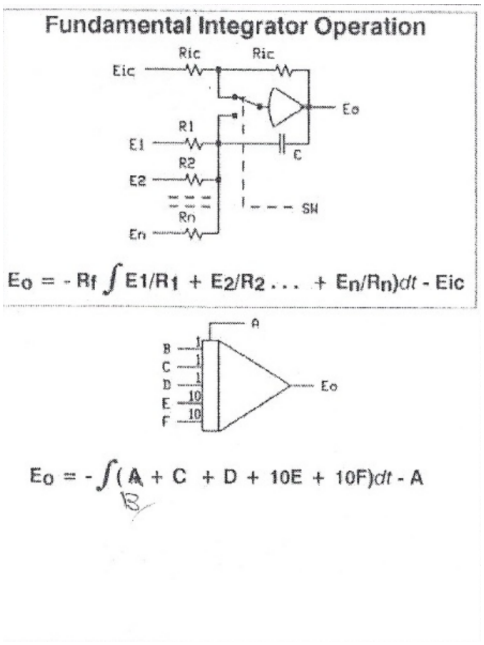
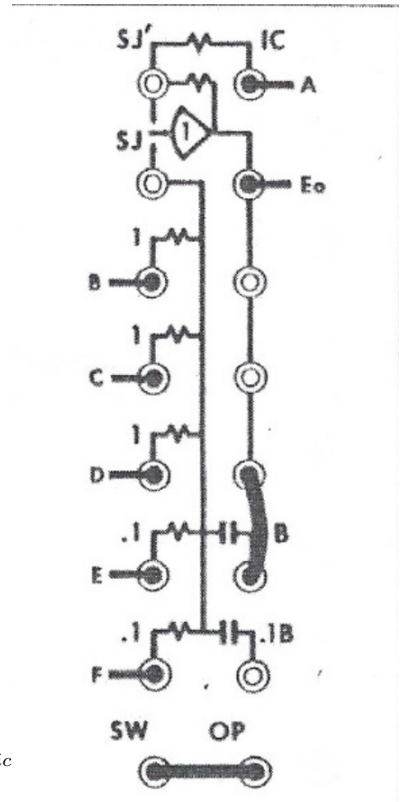
Function	Operation	Patching
INVERTER amplifiers 7 and 8	$E_o = -A$	 Input A is patched through the inverter input path. The output is labeled E_o.

Table B.6: Patch-panel operation for integrator, amplifiers 1–4.

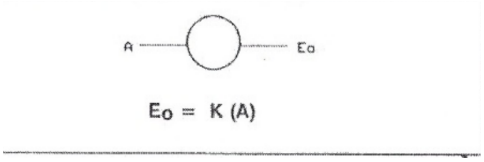
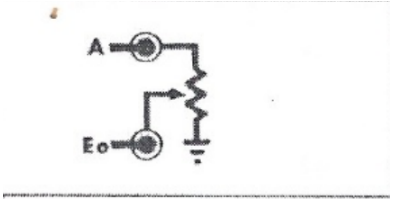
Function	Operation	Patching
INTEGRATOR amplifiers 1–4		

$$E_o = -R_f \int \left(\frac{E_1}{R_1} + \frac{E_2}{R_2} + \cdots + \frac{E_n}{R_n} \right) dt - E_{ic}$$

$$E_o = -\int (A + C + D + 10E + 10F)dt - A$$

The patch uses the IC terminal, summing inputs, B capacitor connection, and OP-to-SW connection for normal integrator operation.

Table B.7: Patch-panel operation for attenuator, potentiometers 1–6.

Function	Operation	Patching
ATTENUATOR pots 1–6		

$$E_o = K(A)$$

Input A is applied to the potentiometer. The wiper output is E_o; the bottom end is grounded.

Table B.8: Patch-panel operation for attenuator, potentiometers 7 and 8.

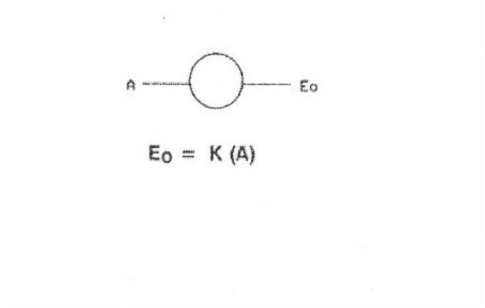
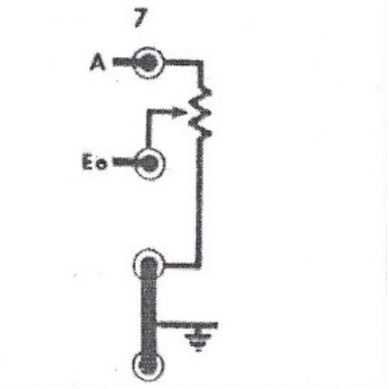
Function	Operation	Patching
ATTENUATOR pots 7 and 8	 $E_o = K(A)$	 Input A is applied to the top of the potentiometer. The bottom terminal is patched to ground. The wiper output is E_o.

Table B.9: Patch-panel operation for voltage divider, potentiometers 7 and 8.

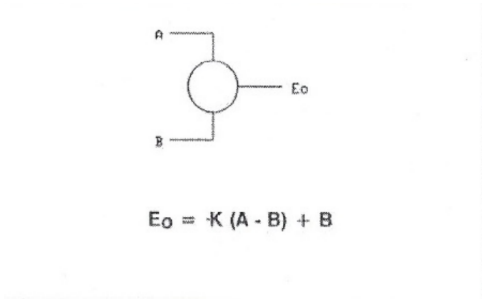
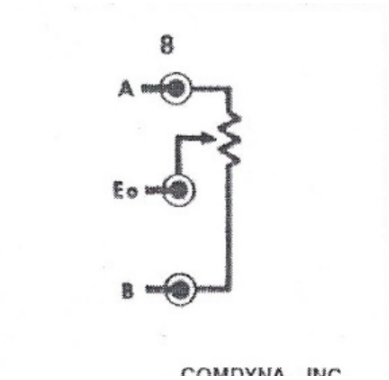
Function	Operation	Patching
VOLTAGE DIVIDER pots 7 and 8	 $E_o = K(A - B) + B$	 Input A is connected to the top end, input B is connected to the bottom end, and E_o is taken from the wiper.

Table B.10: Patch-panel operation for multiplier.

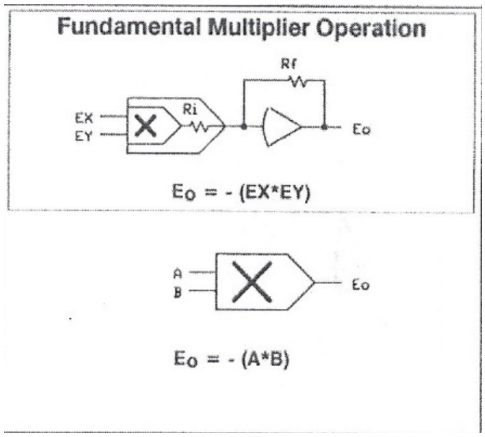
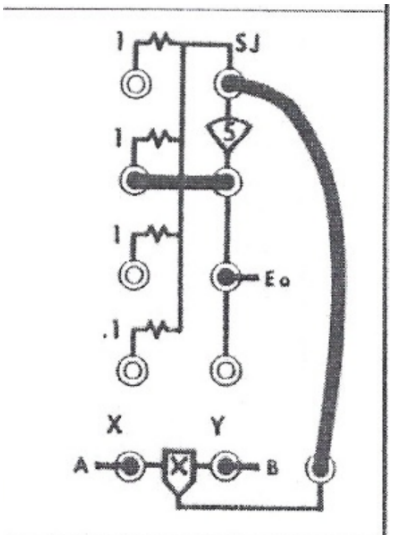
Function	Operation	Patching
MULTI-PLIER		
	$E_o = -(E_X E_Y)$	Inputs A and B patch to multiplier inputs X and Y. The multiplier output is routed to the amplifier summing junction.
	$E_o = -(AB)$	

Table B.11: Patch-panel operation for divider.

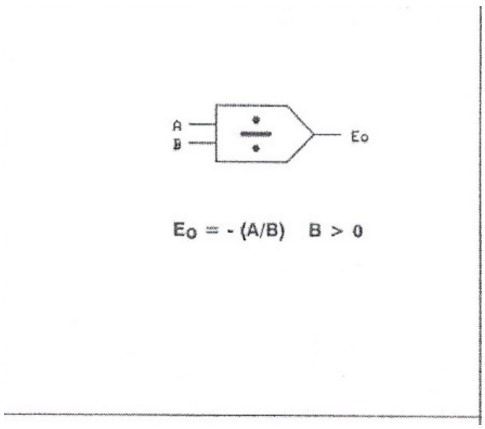
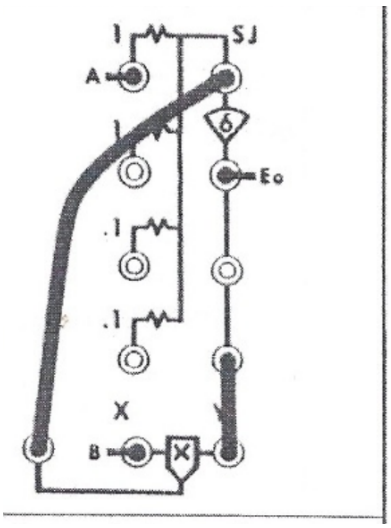
Function	Operation	Patching
DIVIDER		
	$E_o = -(A/B), \qquad B > 0$	The multiplier network is patched in a feedback arrangement to divide A by B.

Table B.12: Patch-panel operation for squarer.

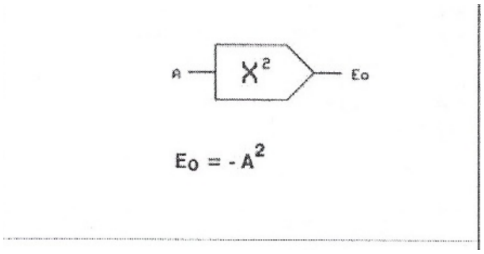
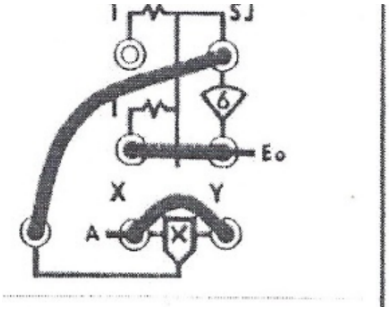
Function	Operation	Patching
SQUARER		
	$E_o = -A^2$	Input A is applied to both multiplier inputs. The product is routed through the amplifier.

Table B.13: Patch-panel operation for square root.

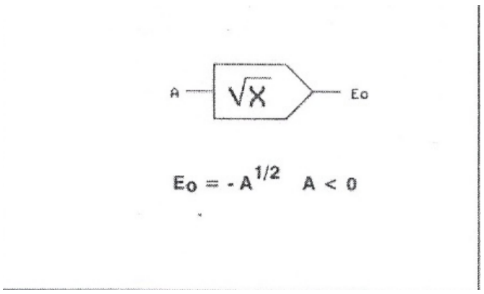
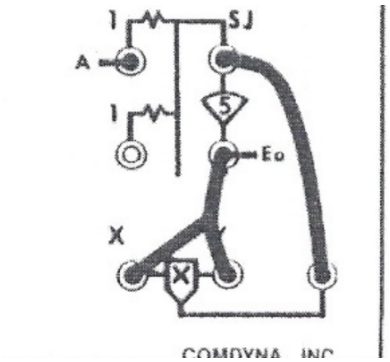
Function	Operation	Patching
SQUARE ROOT		
	$E_o = -A^{1/2}, \qquad A < 0$	The multiplier network and amplifier are patched as a square-root extractor.

Table B.14: Patch-panel operation for differentiator, amplifiers 1–4.

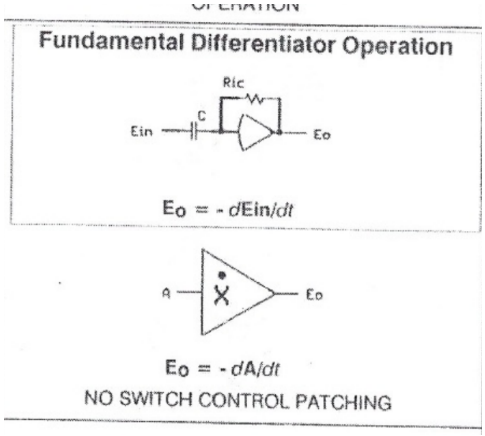
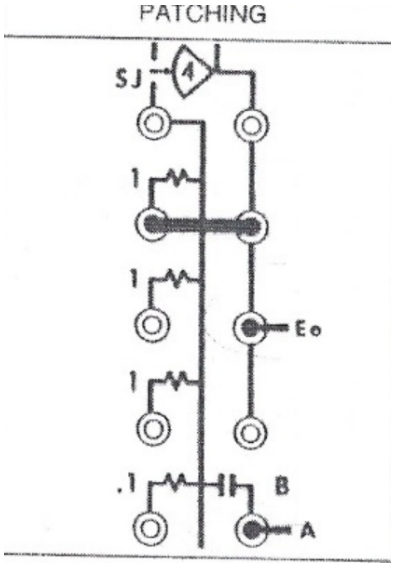
Function	Operation	Patching
DIFFERENTIATOR amplifiers 1–4	 $E_o = -dE_{in}/dt$ $E_o = -dA/dt$ No switch control patching.	 Input A is patched through the differentiator input path. Output is taken at E_o.

Table B.15: Patch-panel operation for track/store, amplifiers 1–4.

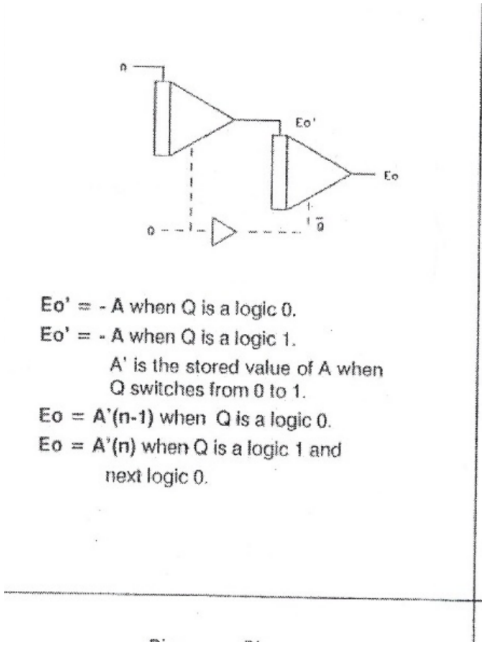
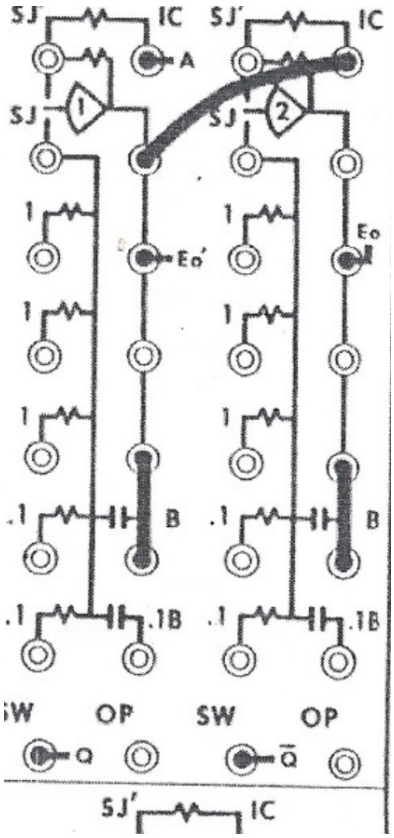
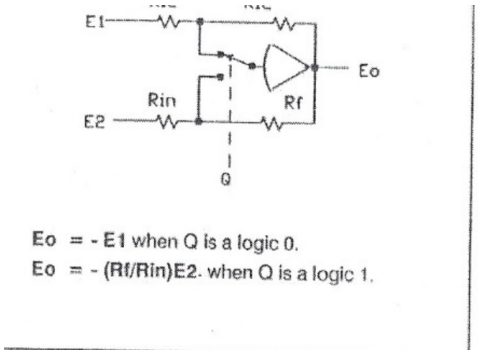
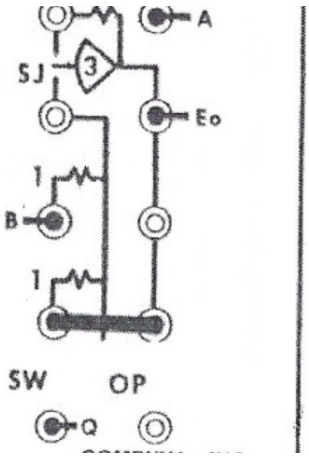
Function	Operation	Patching
TRACK/ STORE amplifiers 1–4	 $E'_o = -A$ when Q is a logic 0. $E'_o = A$ when Q is a logic 1. A' is the stored value of A when Q switches from 0 to 1. $E_o = A'(n-1)$ when Q is a logic 0. $E_o = A'(n)$ when Q is a logic 1 and next logic 0.	 The patch uses two switched amplifier sections controlled by Q and $\overline{Q}$.

Table B.16: Patch-panel operation for DPDT electronic switch, amplifiers 1–4.

Function	Operation	Patching
DPDT ELEC-TRONIC SWITCH amplifiers 1–4		

$E_o = -E_1$ when Q is a logic 0.

$E_o = -(R_f/R_{in})E_2$ when Q is a logic 1.

The switch selects the E_1 path for logic 0 and the scaled E_2 path for logic 1.

B.4 Scaling Review

The following is offered as a review of analog computer programming magnitude and time scaling.

Example Differential Equation

$$A\frac{d^2X}{dt^2} + B\frac{dX}{dt} + CX = f(t)$$

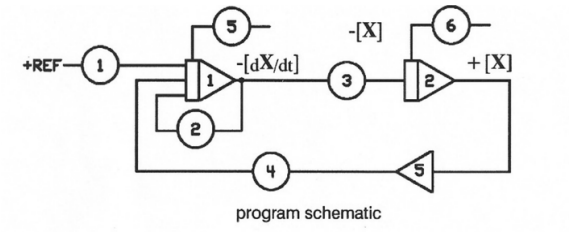


Figure B.2: Scaling-review diagram corresponding to the example differential equation from the Analog Computer's Manual – GP-6.

Accessible description. The figure shows the analog-computer patching interpretation of the preceding example differential equation. It represents the differential-equation terms as scaled signals connected through computing elements such as amplifiers, summers, and integrators.

Table B.17: Given constants and estimated maximum values for the scaling-review example.

Given constants		**Estimated maximum values**	
Quantity	**Value**	**Quantity**	**Value**
A	.0005	$X_{\max}$	.01
B	arbitrary	$(dX/dt)_{\max}$	4
C	800	X_{scaled}	$X/.01$
$f(t)$	.5	$(dX/dt)_{\text{scaled}}$	$(dX/dt)/4$

Rewritten and scaled:

$$\left[\frac{d^2X/dt^2}{4}\right] = -\frac{B}{A\beta}\left[\frac{dX/dt}{4}\right] - \frac{C}{A\beta}\frac{.01}{4}\left[\frac{X}{.01}\right] + \frac{1}{A\beta}\left[\frac{f(t)}{4}\right]$$

$$\beta = \text{time scale factor.}$$

Table B.18: Potentiometer settings from the scaling-review page.

No.	**Parameter**	**Magnitude scale factor**	**Without time scale**	**With time scale,** $\beta = 500$
1	$f(t)/A$	$1/4$	$0.25 \times 10^4/\beta$	.500
2	B/A	1	$2B \times 10^4/\beta$	$40B$
3	dX_{sf}/X_{sf}	$4/.01$	$.4 \times 10^4/\beta$	.800
4	C/A	$.01/4$	$.16 \times 10^4/\beta$	.320
5	dX_o/dt	$1/4$	$dX_o/dt/4$	$dX_o/dt/4$
6	X_o	$1/.01$	$X_o/.01$	$X_o/.01$

B.5 Full GP-6 Patch Panel Scan

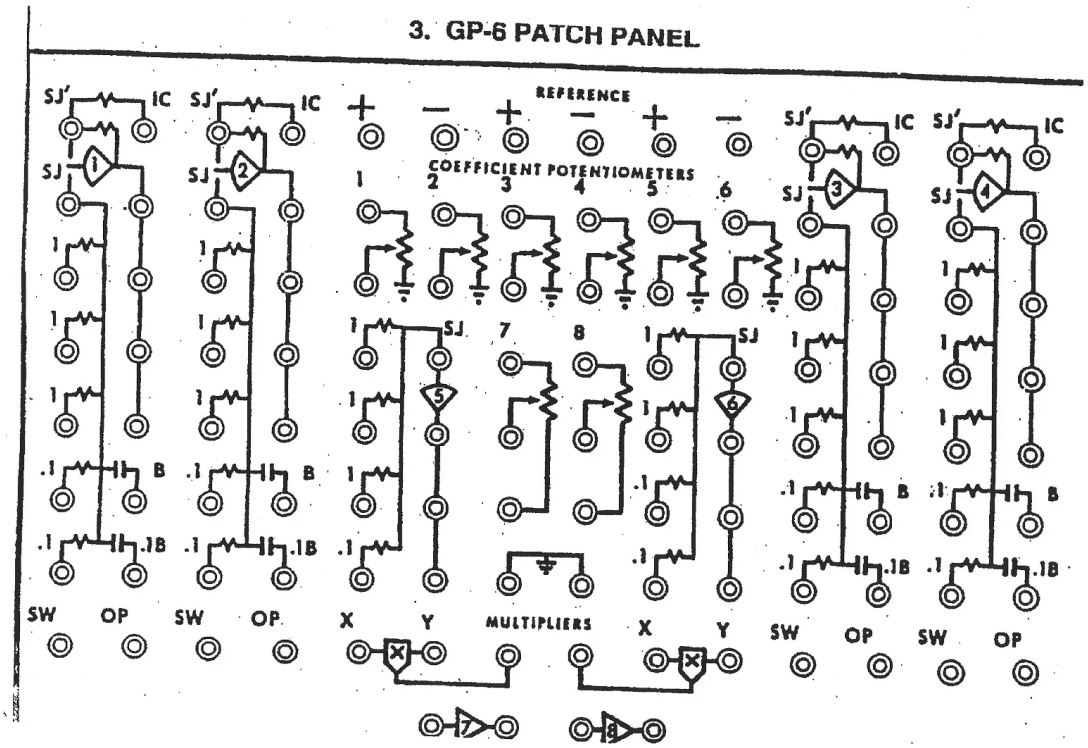


Figure B.3: Rotated and cropped scan of the full GP-6 patch panel.

The full scan shows the GP-6 patch panel as a wiring-reference layout rather than as a completed circuit. The left side contains amplifier sections 1 and 2, the center contains reference terminals, coefficient potentiometers, potentiometers 7 and 8, and multiplier networks, and the right side contains amplifier sections 3 and 4. Circular terminals indicate physical patch points. Printed component symbols identify summing junctions, input resistors, feedback paths, integrating capacitors, switch-control terminals, operate terminals, multiplier inputs, and multiplier outputs.

Appendix C

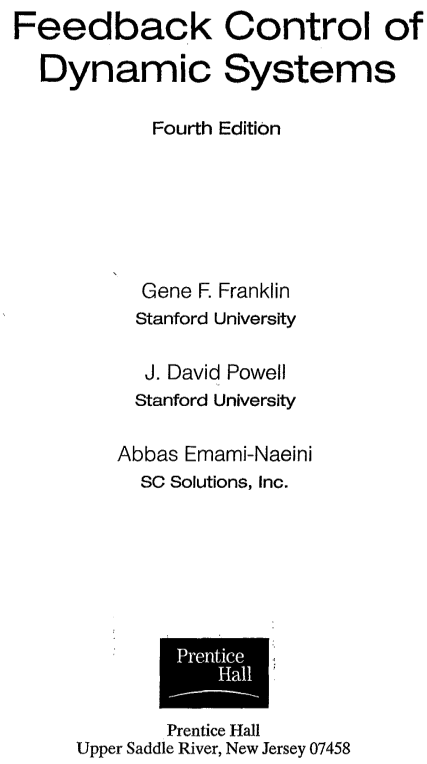


Figure C.1: Title page of *Feedback Control of Dynamic Systems*, Fourth Edition, by Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini.

Library of Congress Cataloging-in-Publication Data

Franklin, Gene F.
Feedback control of dynamic systems / Gene F. Franklin,
J. David Powell, Abbas Emami-Naeini.—4th ed.
p. cm.
Includes index.
ISBN 0-13-032393-4
1. Feedback control systems. I. Powell, J. David. II. Emami-Naeini, Abbas. III. Title.
CIP Data available.

Vice President and Editorial Director, ECS: *Marcia J. Horton*
Acquisitions Editor: *Eric Frank*
Editorial Assistant: *Jessica Romeo*
Vice President and Director of Production and Manufacturing, ESM: *David W. Riccardi*
Executive Managing Editor: *Vince O'Brien*
Managing Editor: *David A. George*
Production Editor: *Irwin Zucker*
Composition: *PreTEX, Inc.*
Director of Creative Services: *Paul Belfanti*
Creative Director: *Carole Anson*
Art Director: *Jonathan Boylan*
Assistant to the Art Director: *John Christiana*
Art Editor: *Xiaohong Zhu*
Interior Designer: *Maria Guglielmo*
Cover Designer: *Stacey Abraham*
Manufacturing Manager: *Trudy Pisciotti*
Manufacturing Buyer: *Lisa McDowell*
Marketing Manager: *Holly Stark*

About the Cover: *Photograph of computer hard drive courtesy of Corbis Images. Photograph of satellite orbiting the earth courtesy of NASA. Photograph of Boeing 747 jet aircraft courtesy of Tony Stone Images.*

Prentice Hall

© 2002, 1994, 1991, 1986 by Prentice Hall
Prentice-Hall, Inc.
Upper Saddle River, New Jersey 07458

All rights reserved. No part of this book may be reproduced, in any format or by any means, without permission in writing from the publisher.

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

MATLAB and Simulink are registered trademarks of The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA, 01760-2098.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-032393-4

Pearson Education Ltd., *London*
Pearson Education Australia Pty. Limited, *Sydney*
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., *Hong Kong*
Pearson Education Canada Inc., *Toronto*
Pearson Educatíon de Mexico, S.A. de C.V.
Pearson Education—Japan, *Tokyo*
Pearson Education Malaysia, Pte. Ltd.
Pearson Education, *Upper Saddle River, New Jersey*

Figure C.2: Copyright and publication information for *Feedback Control of Dynamic Systems*, Fourth Edition.

C.1 Obtaining Models from Experimental Data

There are several reasons for using experimental data to obtain a model of the dynamic system to be controlled. In the first place, the best theoretical model built from equations of motion is still only an approximation of reality. Sometimes, as in the case of a very rigid spacecraft, the theoretical model is extremely good. Other times, as with many chemical processes such as papermaking or metalworking, the theoretical model is very approximate. In every case, before the final control design is done, it is important and prudent to verify the theoretical model with experimental data. Secondly, in situations where the theoretical model is especially complicated or the physics of the process is poorly understood, the only reliable information on which to base the control design is the experimental data. Finally, the system is sometimes subject to on-line changes, which occur when the environment of the system changes. Examples include when an aircraft changes altitude or speed, a paper machine is given a different composition of fiber, or a nonlinear system moves to a new operating point. On these occasions we need to "retune" the controller by changing the control parameters. This requires a model for the new conditions, and experimental data are often the most effective, if not the only, information available for the new model.
There are four kinds of experimental data for generating a model.

1. Transient response, such as comes from an impulse or a step.
2. Frequency response data, which result from exciting the system with sinusoidal inputs at many frequencies.
3. Stochastic steady-state information, as might come from flying an aircraft through turbulent weather or from some other natural source of randomness.
4. Pseudorandom-noise data, as may be generated in a digital computer.

Each class of experimental data has its properties, advantages, and disadvantages.

Transient response. Transient response data are quick and relatively easy to obtain. They are also often representative of the natural signals to which the system is subjected. Thus a model derived from such data can be reliable for designing the control system. On the other hand, in order for the signal-to-noise ratio to be sufficiently high, the transient response must be highly noticeable. Thus the method is rarely suitable for normal operations, so the data must be collected as part of special tests. A second disadvantage is that the data do not come in a form suitable for standard control systems designs, and some parts of the model, such as poles and zeros, must be computed from the data.[1] This computation can be simple in special cases or complex in the general case.

Frequency response. Frequency-response data are simple to obtain but substantially more time-consuming than transient-response information. This is especially so if the time constants of the process are large, as often occurs in chemical processing industries. As with the transient-response data, it is important to have a good signal-to-noise ratio, so obtaining frequency-response data can be very expensive. On the other hand, frequency-response

[1] Original footnote 16: Ziegler and Nichols (1943), building on the earlier work of Callender et al. (1936), use the step response directly in designing the controls for certain classes of processes. See Chapter 4 for details.

data are exactly in the right form for frequency-response design methods, so once the data have been obtained, the control design can proceed immediately.

Stochastic steady-state. Normal operating records from a natural stochastic environment at first appear to be an attractive basis for modeling systems since such records are by definition nondisruptive and inexpensive to obtain. Unfortunately, the quality of such data is inconsistent, tending to be worst just when the control is best, because then the upsets are minimal and the signals are smooth. At such times some or even most of the system dynamics are hardly excited. Because they contribute little to the system output, they will not be found in the model constructed to explain the signals. The result is a model that represents only part of the system and is sometimes unsuitable for control. In some instances, as occurs when trying to model the dynamics of the electroencephalogram, or brain waves, of a sleeping or anesthetized person to locate the frequency and intensity of alpha waves, normal records are the only possibility. Usually they are the last choice for control purposes.

Pseudorandom noise (PRBS). Finally, the pseudorandom signals that can be constructed using digital logic have much appeal. Especially interesting for model making is the pseudorandom binary signal (PRBS). The PRBS takes on the value $+A$ or $-A$ according to the output, 1 or 0, of a feedback shift register. The feedback to the register is a binary sum of various states of the register that have been selected to make the output period, which must repeat itself in finite time, as long as possible. For example, with a register of 20 bits, $2^{20} - 1$, over a million, steps are produced before the pattern repeats. Analysis beyond the scope of this text has revealed that the resulting signal is almost like a broad-band random signal. Yet this signal is entirely under the control of the engineer who can set the level A and the length, bits in the register, of the signal. The data obtained from tests with a PRBS must be analyzed by computer, and both special-purpose hardware and programs for general-purpose computers have been developed to perform this analysis.

C.1.1 Models from Transient-Response Data

To obtain a model from transient data we assume that a step response is available. If the transient is a simple combination of elementary transients, then a reasonable low-order model can be estimated using hand calculations. For example, consider the step response shown in Fig. C.3. The response is monotonic and smooth. If we assume that it is given by a sum of exponentials, we can write

$$y(t) = y(\infty) + Ae^{-\alpha t} + Be^{-\beta t} + Ce^{-\gamma t} + \cdots . \tag{3.83}$$

Subtracting off the final value and assuming that $-\alpha$ is the slowest pole, we write

$$\begin{aligned} y - y(\infty) &\simeq Ae^{-\alpha t}, \\ \log_{10}[y - y(\infty)] &\simeq \log_{10} A - \alpha t \log_{10} e \\ &\simeq \log_{10} A - 0.4343\alpha t. \end{aligned} \tag{3.84}$$

This is the equation of a line whose slope determines α and intercept determines A. If we fit a line to the plot of $\log_{10}[y - y(\infty)]$, or $\log_{10}[y(\infty) - y]$ if A is negative, then we can

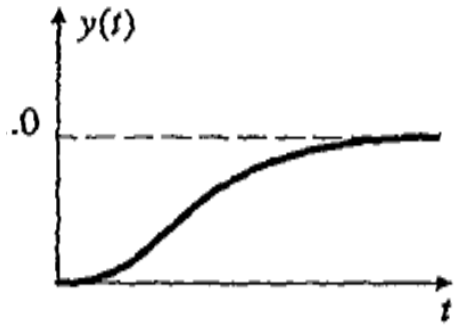


Figure C.3: A step response characteristic of many chemical processes (original Fig. 3.45).

estimate A and α. Once these are estimated, we plot $y - [y(\infty) + Ae^{-\alpha t}]$, which as a curve approximates $Be^{-\beta t}$ and on the log plot is equivalent to $\log_{10} B - 0.4343\beta t$. We repeat the process, each time removing the slowest remaining term, until the data stop being accurate. Then we plot the final model step response and compare it with data so we can assess the quality of the computed model. It is possible to get a good fit to the step response and yet be far off from the true time constants, or poles, of the system. However, the method gives a good approximation for control of processes whose step responses look like Fig. C.3.

Example 3.35: Determining the Model from Time-Response Data. Find the transfer function that generates the data given in Table 3.1 and which are plotted in Fig. C.4.

Table B.19: Step-response data.

First data pair		**Second data pair**	
t	$y(t)$	t	$y(t)$
0.0	0.000	1.0	0.510
0.1	0.005	1.5	0.700
0.2	0.034	2.0	0.817
0.3	0.085	2.5	0.890
0.4	0.140	3.0	0.932
0.5	0.215	4.0	0.975
		∞	1.000

Source: Sinha and Kuszta (1983).

Solution. Table B.19 shows and Fig. C.4 implies that the final value of the data is $y(\infty) = 1$. We know that A is negative because $y(\infty)$ is greater than $y(t)$. Therefore, the first step in the process is to plot $\log_{10}[y(\infty) - y]$, which is shown in Fig. C.5. From the line, fitted by eye, the values are

$$\log_{10} |A| = 0.125,$$

$$0.4343\alpha = \frac{1.602 - 1.167}{\Delta t} = \frac{0.435}{1} \Rightarrow \alpha \simeq 1.$$

Thus

$$A = -1.33, \qquad \alpha = 1.0.$$

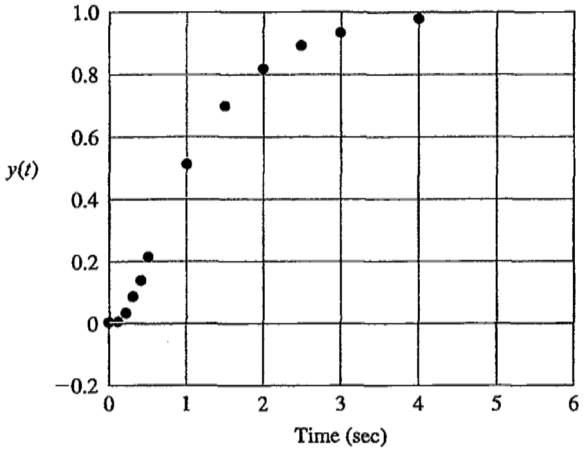


Figure C.4: Step-response data in Table 3.1 (original Fig. 3.46).

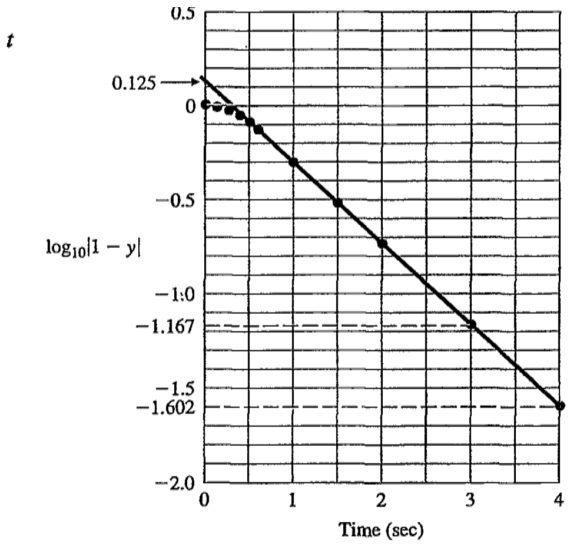


Figure C.5: Semilog plot of $\log_{10}[y(\infty) - y]$ versus t (original Fig. 3.47).

If we now subtract $1 + Ae^{-\alpha t}$ from the data and plot the log of the result, we find the plot of Fig. C.6. Here we estimate

$$\log_{10} B = -0.48,$$
$$0.4343\beta = \frac{-0.48 - (-1.7)}{0.5} = 2.5,$$
$$\beta = 5.8, \qquad B = 0.33.$$

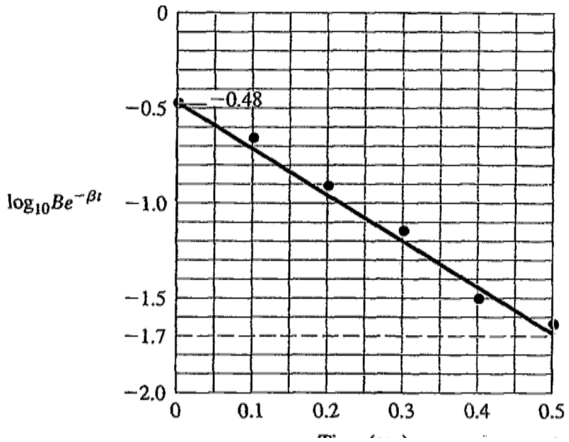


Figure C.6: Semilog residual plot of $\log_{10}[y - (1 + Ae^{-\alpha t})]$ versus t (original Fig. 3.48).

Combining these results, we arrive at the y estimate

$$\hat{y}(t) = 1 - 1.33e^{-t} + 0.33e^{-5.8t}. \tag{3.85}$$

Equation (3.85) is plotted as the colored line in Fig. C.7 and shows a reasonable fit to the data, although some error is noticeable near $t = 0$.
From $\hat{y}(t)$ we compute

$$\begin{aligned} Y(s) &= \frac{1}{s} - \frac{1.33}{s+1} + \frac{0.33}{s+5.8} \\ &= \frac{(s+1)(s+5.8) - 1.33s(s+5.8) + 0.33s(s+1)}{s(s+1)(s+5.8)} \\ &= \frac{-0.58s + 5.8}{s(s+1)(s+5.8)}. \end{aligned}$$

The resulting transfer function is

$$G(s) = sY(s) = \frac{-0.58s + 5.8}{(s+1)(s+5.8)} = \frac{-0.58(s-10)}{(s+1)(s+5.8)}.$$

Notice that this method has given us a system with a zero in the RHP, even though the data showed no values of y that were negative. Very small differences in the estimated value for A, all of which approximately fit the data, can cause values of β to range from 4 to 6. This illustrates the sensitivity of pole locations to the quality of the data and emphasizes the need for a good signal-to-noise ratio.
By using a computer to perform the plotting, we are better able to iterate the four parameters to achieve the best overall fit. The data presentation in Figs. C.5 and C.6 can

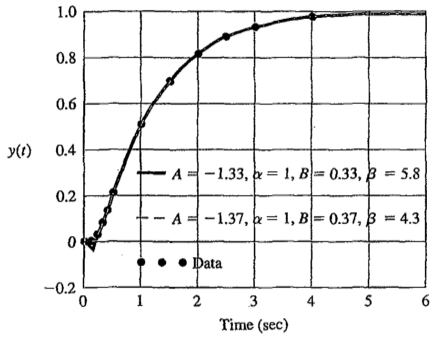


Figure C.7: Model fits to the experimental data (original Fig. 3.49).

be obtained directly by using a semilog plot. This eliminates having to calculate $\log_{10}$ and the exponential expression to find the values of the parameters. The equations of the lines to be fit to the data are $y(t) = Ae^{-\alpha t}$ and $y(t) = Be^{-\beta t}$, which are straight lines on a semilog plot. The parameters A and α, or B and β, are iteratively selected so that the straight line comes as close as possible to passing through the data. This process produces the improved fit shown by the dashed black line in Fig. C.7. The revised parameters, $A = -1.37$, $B = 0.37$, and $\beta = 4.3$ result in the transfer function

$$G(s) = \frac{-0.22s + 4.3}{(s+1)(s+4.3)}.$$

The RHP zero is still present, but it is now located at $s = +20$ and has no noticeable effect on the time response.

This set of data was fitted quite well by a second-order model. In many cases a higher-order model is required to explain the data and the modes may not be as well-separated.

If the transient response has oscillatory modes, then these can sometimes be estimated by comparing them with the standard plots of Fig. 3.23. The period will give the frequency ω_d, and the decay from one period to the next will afford an estimate of the damping ratio. If the response has a mixture of modes not well separated in frequency, then more sophisticated methods need to be used. One such method is least-squares identification, in which a numerical optimization routine selects the best combination of parameters so as to minimize the fit error. The fit error is defined to be a scalar cost function

$$J = \sum_i \left(y_{\text{data},i} - y_{\text{model},i}\right)^2, \qquad i = 1, 2, 3, \ldots,$$

for each data point, so that fit errors at all data points are taken into account in determining the best value for the parameters.

C.1.2 Models from Other Data

As mentioned early in Section 3.8, we can also generate a model using frequency-response data, which are obtained by exciting the system with a set of sinusoids and plotting $H(j\omega)$. In Chapter 6 we will show how such plots can be used directly for design. Alternatively,

we can use the frequency response to estimate the poles and zeros of a transfer function using straight-line asymptotes on a logarithmic plot.
The construction of dynamic models from normal stochastic operating records or from the response to a PRBS can be based either on the concept of cross-correlation or on the least-squares fit of a discrete equivalent model, both topics in the field of system identification. They require substantial presentation and background that are beyond the scope of this text. An introduction to system identification can be found in Chapter 8 of Franklin et al. (1998), and a comprehensive treatment is given in Ljung (1999). Based largely on the work of Professor Ljung, the MATLAB Toolbox on Identification provides substantial software to perform system identification and to verify the quality of the proposed models.

Summary

The Laplace transform is the primary tool used to determine the behavior of linear systems. The Laplace transform of a time function $f(t)$ is given by

$$\mathcal{L}[f(t)] = F(s) = \int_0^\infty f(t)e^{-st}\,dt. \tag{3.86}$$

This relationship leads to the key property of Laplace transforms, namely,

$$\mathcal{L}[\dot{f}(t)] = sF(s) - f(0). \tag{3.87}$$

This property allows us to find the transfer function of a linear ODE. Given the transfer function $G(s)$ of a system and the input $u(t)$, with transform $U(s)$, the system output transform is

$$Y(s) = G(s)U(s).$$

C.2 A Case Study of Speed Control

It is very important to maintain constant speed of the spindle in high-performance hard-disk drives used for computer data storage and, in fact, speed control with an electric motor is a generic problem calling for control. As derived in Chapter 2, Eqs. (4.1) and (4.2) below describe the dynamics of a DC motor with negligible armature inductance and including a term, $\tau_\ell(t)$, for load-torque disturbance.

$$J_m\dot{\omega}_m + b\omega_m = K_t i_a + \tau_\ell, \tag{4.1}$$

$$K_e\omega_m + R_a i_a = v_a. \tag{4.2}$$

If we take the Laplace transform of Eqs. (4.1) and (4.2) and let the velocity $\dot{\theta}_m = \omega_m$ with transform $\Omega_m(s) = s\Theta(s)$ and the transform of the load torque be $T_\ell(s)$, we obtain the transformed equations of speed control as

$$sJ_m\Omega_m(s) + b\Omega_m(s) = K_t I_a(s) + T_\ell(s), \tag{4.3}$$

$$K_e\Omega_m(s) + R_a I_a(s) = V_a(s). \tag{4.4}$$

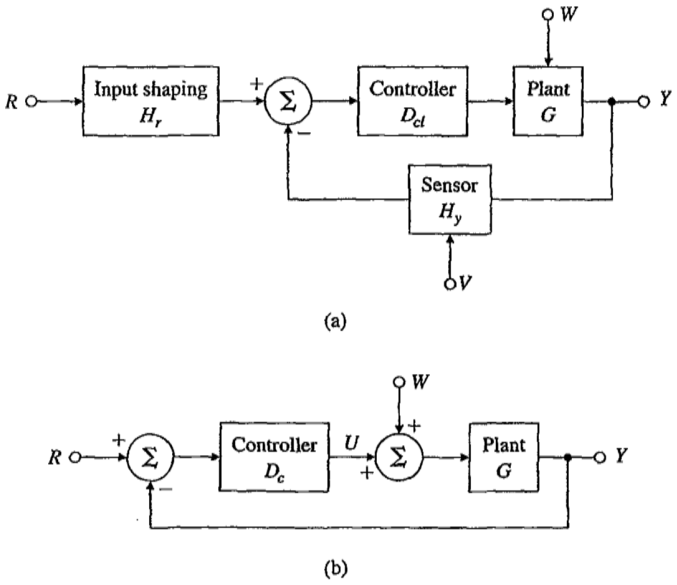


Figure C.8: Feedback control system (original Fig. 4.2).

After solving for I_a from Eq. (4.4) and substituting it into Eq. (4.3), the equation for motor speed becomes

$$(J_m R_a s + bR_a + K_t K_e)\,\Omega_m(s) = K_t V_a(s) + R_a T_\ell(s). \tag{4.5}$$

Or,

$$\left(\frac{J_m R_a}{bR_a + K_t K_e} s + 1\right) \Omega_m(s) = \frac{K_t}{bR_a + K_t K_e} V_a(s) + \frac{R_a}{bR_a + K_t K_e} T_\ell(s). \tag{4.6}$$

We can simplify the coefficients in this equation by defining new parameters as

$$(\tau s + 1)\Omega_m(s) = AV_a(s) + BT_\ell(s), \tag{4.7}$$

where the time constant and gains are given by

$$\begin{aligned} \tau &= \frac{J_m R_a}{bR_a + K_t K_e} && \text{(time constant, sec)}, \\ A &= \frac{K_t}{bR_a + K_t K_e} && \text{(voltage gain, rad/volt-sec)}, \\ B &= \frac{R_a}{bR_a + K_t K_e} && \text{(torque gain, rad/N-m-sec)}. \end{aligned} \tag{4.8}$$

We may also rewrite Eq. (4.7) in transfer-function form:

$$\Omega_m(s) = \frac{A}{\tau s + 1} V_a(s) + \frac{B}{\tau s + 1} T_\ell(s). \tag{4.9}$$

If the inputs are constants, say $v_a = a$ and $\tau_\ell = b$, then the steady-state solution of these equations is[2]

$$\omega_{ss} = Aa + Bb. \tag{4.10}$$

[2]From the final value theorem, if $Y(s) = G(s)U(s)$ and $U(s) = 1/s$, the final value of $y(t)$ is $y_{ss} = G(0)$. Thus we just have to set $s = 0$ in a stable transfer function to find the final or steady-state value to a unit step input.

The transfer functions from Eq. (4.9) are shown in Fig. C.9 for open-loop control with a controller having transfer function $D_{ol}(s)$. For this configuration we will refer to the transfer function from Ω_{ref} to Ω_m as the open-loop transfer function,

$$T_{ol}(s) = H_r D_{ol} \frac{A}{\tau s + 1}.$$

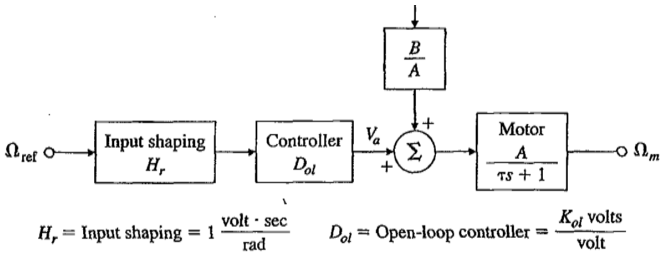


Figure C.9: Open-loop speed-control system (original Fig. 4.3).

In contrast, the block diagram in Fig. C.10 shows a closed-loop system which requires an output sensor. In this case the sensor is a tachometer, which is usually a small permanent-magnet DC machine which produces a voltage proportional to the shaft speed ω_m. In this case, the closed-loop transfer function is

$$T_{cl}(s) = \frac{\Omega_m}{\Omega_{\text{ref}}} = \frac{H_r D_{cl} \dfrac{A}{\tau s + 1}}{1 + D_{cl} H_y \dfrac{A}{\tau s + 1}}.$$

We will often refer to the transfer function of the elements around a closed loop as the loop gain, which in this case is

$$D_{cl} H_y \frac{A}{\tau s + 1}.$$

In these figures and in the analysis below, the gain or scale factor of the sensor is taken to be $H_y = 1$ volt-sec/rad. Techniques for treating more practical values will be described shortly. The input-shaping filter, H_r, is a units-conversion device also taken here to have gain 1 volt-sec/rad. The controllers in the figures are taken to be electronic circuits with transfer functions having dimensions volts/volt.

C.2.1 Disturbance Rejection

Now let us compare open-loop control with feedback control with respect to how well each system maintains a constant steady-state reference speed in the face of load or disturbance torques. For the open-loop controller, Fig. C.9, the amplifier voltage is taken to be

$$v_a = D_{ol}\omega_{\text{ref}} = K_{ol}\omega_{\text{ref}}. \tag{4.11}$$

The constant gain $D_{ol}(s) = K_{ol}$ is determined so that in steady-state the output speed, $\omega_m = \omega_{ss}$, equals the constant reference speed ω_{ref} when the load torque τ_ℓ is zero. From Eq. (4.10) this value is found to be

$$K_{ol} = \frac{1}{A}.$$

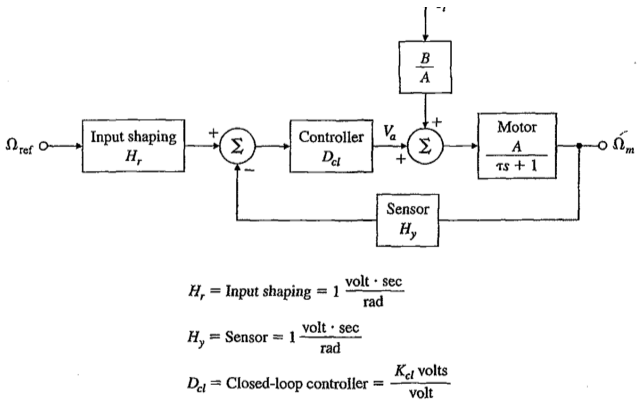


Figure C.10: Feedback speed-control system (original Fig. 4.4).

In the steady state, $s = 0$, and with no load torque, $\tau_\ell = 0$, Eq. (4.10) gives the output speed as

$$\omega_m = \omega_{ss} = Aa = A\frac{1}{A}\omega_{\text{ref}} = \omega_{\text{ref}}.$$

With the feedback controller, Fig. C.10, the controller transfer function is taken to be $D_{cl} = K_{cl}$ so that

$$v_a = K_{cl}(\omega_{\text{ref}} - \omega_m). \tag{4.12}$$

Combining the motor model given by Eq. (4.7) with the transform of the feedback controller equation described by Eq. (4.12), the closed-loop system equations are

$$[(\tau s + 1) + AK_{cl}]\Omega_m(s) = AK_{cl}\Omega_{\text{ref}}(s) + BT_\ell(s). \tag{4.13}$$

For no load torque, $\tau_\ell = 0$, and in the steady state, $s = 0$, the speed is given by

$$\omega_{ss} = \frac{AK_{cl}}{1 + AK_{cl}}\omega_{\text{ref}}. \tag{4.14}$$

If the gain K_{cl} is selected large so that $AK_{cl} \gg 1$, then, although some error remains, $\omega_{ss} \simeq \omega_{\text{ref}}$.

With these controls, both approaches provide the desired result of good tracking, $\omega_{ss} \simeq \omega_{\text{ref}}$, although on the face of it the open-loop seems to be more accurate. However, the gains have been set with zero load torque; now let us compute what effect a nonzero torque will have on the steady-state speed in the two cases. Equation (4.10) shows that in the open-loop system the steady-state speed is

$$\omega_{ss} = AK_{ol}\omega_{\text{ref}} + B\tau_\ell. \tag{4.15}$$

Here, with $K_{ol} = 1/A$, the speed with load torque for the open-loop case is

$$\omega_{ss} = \omega_{\text{ref}} + B\tau_\ell,$$

and the variation in speed caused by the load torque for the open-loop case is $\omega_{ss} - \omega_{\text{ref}} = \delta\omega$, where

$$\delta\omega = B\tau_\ell.$$

So we see that the speed error is proportional to the disturbing load torque, and the control designer has no influence over the parameter B which determines the size of the error.
For the feedback case, Eq. (4.13), the steady-state speed for constant values of ω_m and τ_ℓ becomes

$$\omega_{ss} = \frac{AK_{cl}}{1 + AK_{cl}}\omega_{\text{ref}} + \frac{B}{1 + AK_{cl}}\tau_\ell. \tag{4.16}$$

If it is possible for the designer to pick values of K_{cl} so that $AK_{cl} \gg 1$ and $AK_{cl} \gg B$, a very small error will result with or without a disturbing load torque. In fact, as long as $AK_{cl} > 0$, comparison of Eqs. (4.15) and (4.16) reveals that the feedback-controller speed errors due to a load torque will be less than the open-loop errors by exactly $1 + AK_{cl}$. This is our first result on the advantages of feedback:

> **Advantage of feedback.** System errors to constant disturbances can be made smaller with feedback than they are in open-loop systems by a factor of $1 + AK_{cl}$, where AK_{cl} is the loop gain at $s = 0$.

Example 4.1: Steady-State Error to Disturbance, Open-Loop Case. For a certain small servomotor and load, it is determined that $\tau = \frac{1}{100}$ sec, $A = 10$ rad/V-sec, and $B = 50$ rad/N-m-sec. The reference speed is $\omega_{\text{ref}} = 100$ rad/sec. Find the steady-state open-loop armature voltage needed to get this speed with zero load torque. What is the steady-state speed error with this load voltage if the load torque is $\tau_\ell = -0.1$ N-m?
Solution. For the open-loop system with an armature voltage input v_a and a load torque τ_ℓ, the steady-state output is given by Eq. (4.10) with $K_{ol} = \frac{1}{10}$ and thus $v_a = \omega_{\text{ref}}/10$.

$$\omega_{ss} = 10v_a + 50\tau_\ell.$$

Assuming $\tau_\ell = 0$, the output is

$$\omega_{ss} = 100 \text{ rad/sec} = \omega_{\text{ref}}.$$

Now suppose a constant load torque of $\tau_\ell = -0.1$ N-m is applied. The steady-state value of the output with the disturbance is

$$\omega_{ss} = 100 + 50(-0.1) = 95 \text{ rad/sec}.$$

Thus the error is 5 rad/sec or 5%.
Now consider the same example using proportional feedback of the output.

Example 4.2: Steady-State Error to Disturbance, Closed-Loop Case. Consider the feedback control structure shown in Fig. C.10, where tachometer feedback and proportional control are employed. You are to improve the ability of the system to reject steady-state disturbances by a factor of at least 100 compared with the open-loop system. The parameter values are the same as those given in Example 4.1.
Solution. From Eq. (4.16) we see that to get an improvement factor of 100 we require $1 + AK_{cl} = 100$, from which $K_{cl} = 9.9$. With this value of K_{cl}, the steady-state value of

the tracking output assuming that the load torque $T_\ell = 0$ is[3]

$$\omega_{ss} = \frac{AK_{cl}}{1 + AK_{cl}}\omega_{\text{ref}} = \frac{99}{100}100 = 99 \text{ rad/sec.}$$

The steady-state output due to the input of a constant disturbance torque is, from Eq. (4.16),

$$\omega_{ss} = \frac{B\tau_\ell}{1 + K_{cl}A} = \frac{50\tau_\ell}{1 + 10K_{cl}}.$$

The output with both reference and disturbance is $\omega_{ss} = 99 - 0.05 = 98.95$ rad/sec, which is a change of only 0.051% from the speed without disturbance. Thus the feedback has improved the disturbance rejection by a factor of about 100 over the open-loop system.

C.2.2 Sensitivity of System Gain to Parameter Changes

As another comparison of open- and closed-loop control, we consider the sensitivity of the steady-state gain or transfer function at $s = 0$ to parameter changes. The change might come about because of external effects such as temperature changes or might simply be due to an error in the value of the parameter from the start. Suppose that the motor gain in operation differs from its original design value of A to be $A + \delta A$. In the open-loop case the nominal gain is[4] $T_{ol} = K_{ol}A$, and the new overall system gain would be

$$T_{ol} + \delta T_{ol} = K_{ol}(A + \delta A) = K_{ol}A + K_{ol}\delta A = T_{ol} + K_{ol}\delta A,$$

where T_{ol} is the open-loop transfer function at $s = 0$. Using the fact that $K_{ol} = 1/A$, the normalized error in the gain is

$$\frac{\delta A}{A}.$$

[3] In process control it is common practice to note that if the set point were reset to $\omega'_{\text{ref}} = 100/99\,\omega_{\text{ref}}$, then the output will be equal to the reference value.

[4] We use T_{ol} and T_{cl} for the open-loop and closed-loop transfer functions, respectively. These are not to be confused with the transform of the disturbance torque, T_ℓ, used earlier.

Appendix D

D.1 Laboratory Instrumentation Overview

This appendix transcribes the laboratory instrumentation reference sheets used for the college control laboratory. The original pages are reproduced as figures for visual reference. The tables below provide accessible text equivalents for the figure labels, front-panel controls, menus, and soft-key maps.

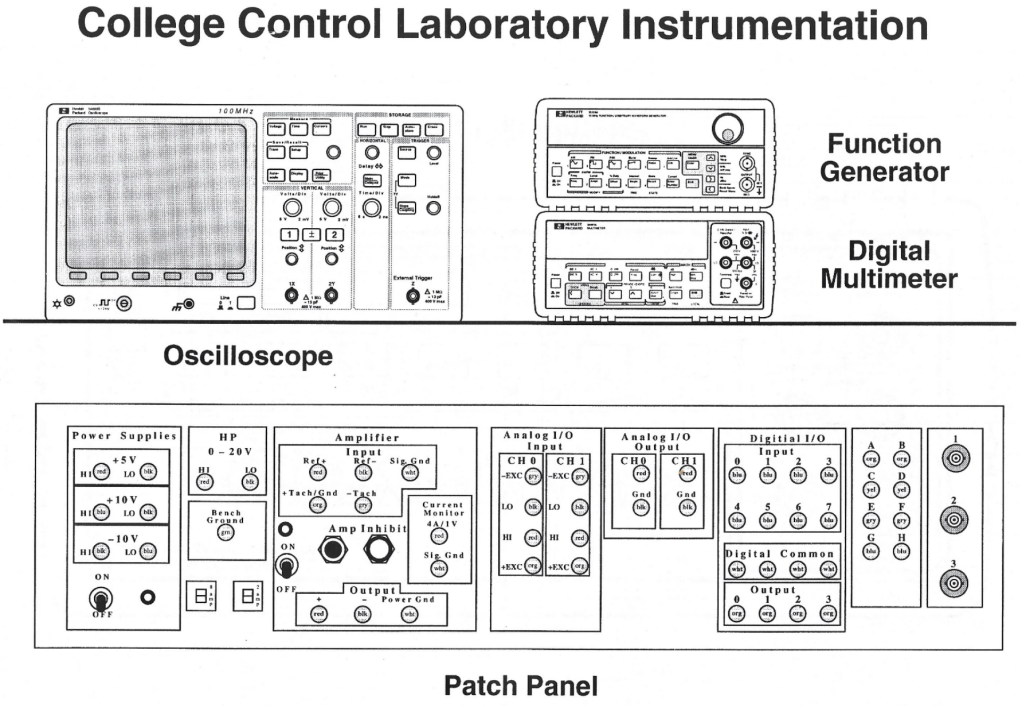
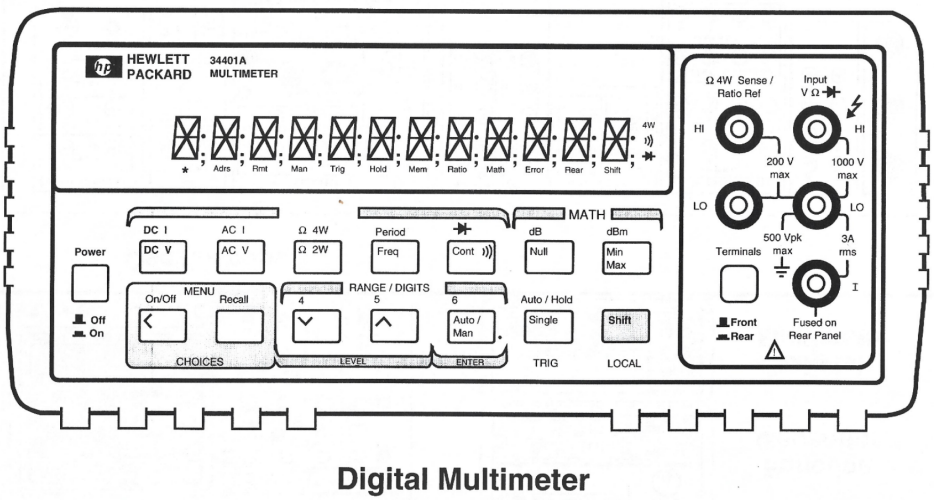


Figure D.1: College control laboratory instrumentation overview.

Table D.1: Accessible overview of the college control laboratory instrumentation.

Instrument or panel section	**Accessible description**
Oscilloscope	Digital two-channel oscilloscope used to display and measure time-domain signals. The front panel contains a display area, vertical controls for channels 1 and 2, horizontal controls, trigger controls, storage controls, measurement controls, and input connectors.
Function generator	Waveform source used to generate test inputs. The front panel contains function and modulation keys, frequency, amplitude, and offset controls, menu navigation keys, and output and sync connectors.
Digital multimeter	Measurement instrument used for voltage, current, resistance, continuity, frequency, period, ratio, math, trigger, and communication functions. The front panel contains a numeric display, measurement keys, menu keys, and input terminals.
Patch panel	Physical interface panel for connecting the control-laboratory plant and instruments. It includes power supplies, HP 0–20 V source terminals, amplifier input and output terminals, analog I/O, digital I/O, digital common terminals, lettered terminals A–H, and three numbered coaxial connectors.

D.2 Digital Multimeter



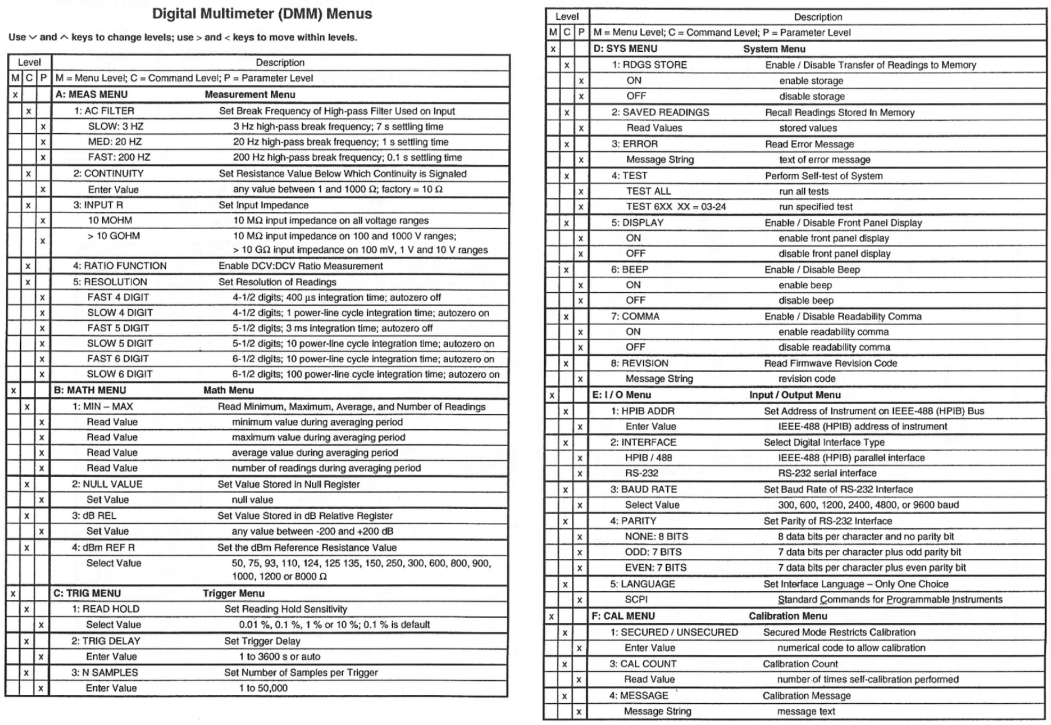
Digital Multimeter

Figure D.2: Digital multimeter front panel.

Use the down and up keys to change levels. Use the right and left keys to move within levels. In the original table, **M** denotes menu level, **C** denotes command level, and **P** denotes parameter level.

Table D.2: Accessible transcription of the digital multimeter front-panel areas.

Front-panel area	**Controls, connectors, or labels shown**
Display and indicators	Large numeric display with annunciators for address, remote, trigger, hold, memory, ratio, math, error, rear, and shift; four-wire and diode/continuity indicators also appear near the display.
Power and measurement keys	Power switch; DC voltage, AC voltage, four-wire resistance, two-wire resistance, period, frequency, continuity, diode, dB, and dBm keys.
Menu and range keys	On/off menu, recall, left and right arrow choices, range up and down, digit up and down, auto/manual, single, and shift keys.
Math and trigger keys	Null, min/max, auto/hold, and math-related key labels for computed measurements and triggered readings.
Input terminals	Four-wire sense high and low terminals, voltage/resistance input high and low terminals, current input terminal, and common low terminals. Printed limits include 200 V maximum for sense ratio reference, 1000 V maximum for voltage/resistance input, 500 V peak maximum terminal rating, and 3 A RMS fused current input.
Front/rear selection	Front and rear terminal selection labels are printed at the lower right of the terminal block.



Digital Multimeter (DMM) Menus

Use ∨ and ∧ keys to change levels; use > and < keys to move within levels.

Level			Description	
M	C	P	M = Menu Level; C = Command Level; P = Parameter Level	
x			**A: MEAS MENU**	**Measurement Menu**
	x		1: AC FILTER	Set Break Frequency of High-pass Filter Used on Input
		x	SLOW: 3 HZ	3 Hz high-pass break frequency; 7 s settling time
		x	MED: 20 HZ	20 Hz high-pass break frequency; 1 s settling time
		x	FAST: 200 HZ	200 Hz high-pass break frequency; 0.1 s settling time
	x		2: CONTINUITY	Set Resistance Value Below Which Continuity is Signaled
		x	Enter Value	any value between 1 and 1000 Ω; factory = 10 Ω
	x		3: INPUT R	Set Input Impedance
		x	10 MOHM	10 MΩ input impedance on all voltage ranges
		x	> 10 GOHM	10 MΩ input impedance on 100 and 1000 V ranges; > 10 GΩ input impedance on 100 mV, 1 V and 10 V ranges
	x		4: RATIO FUNCTION	Enable DCV:DCV Ratio Measurement
	x		5: RESOLUTION	Set Resolution of Readings
		x	FAST 4 DIGIT	4-1/2 digits; 400 µs integration time; autozero off
		x	SLOW 4 DIGIT	4-1/2 digits; 1 power-line cycle integration time; autozero on
		x	FAST 5 DIGIT	5-1/2 digits; 3 ms integration time; autozero off
		x	SLOW 5 DIGIT	5-1/2 digits; 10 power-line cycle integration time; autozero on
		x	FAST 6 DIGIT	6-1/2 digits; 10 power-line cycle integration time; autozero on
		x	SLOW 6 DIGIT	6-1/2 digits; 100 power-line cycle integration time; autozero on
x			**B: MATH MENU**	**Math Menu**
	x		1: MIN – MAX	Read Minimum, Maximum, Average, and Number of Readings
		x	Read Value	minimum value during averaging period
		x	Read Value	maximum value during averaging period
		x	Read Value	average value during averaging period
		x	Read Value	number of readings during averaging period
	x		2: NULL VALUE	Set Value Stored In Null Register
		x	Set Value	null value
	x		3: dB REL	Set Value Stored in dB Relative Register
		x	Set Value	any value between -200 and +200 dB
	x		4: dBm REF R	Set the dBm Reference Resistance Value
			Select Value	50, 75, 93, 110, 124, 125 135, 150, 250, 300, 600, 800, 900, 1000, 1200 or 8000 Ω
x			**C: TRIG MENU**	**Trigger Menu**
	x		1: READ HOLD	Set Reading Hold Sensitivity
		x	Select Value	0.01 %, 0.1 %, 1 % or 10 %; 0.1 % is default
	x		2: TRIG DELAY	Set Trigger Delay
		x	Enter Value	1 to 3600 s or auto
	x		3: N SAMPLES	Set Number of Samples per Trigger
		x	Enter Value	1 to 50,000

Level			Description	
M	C	P	M = Menu Level; C = Command Level; P = Parameter Level	
x			**D: SYS MENU**	**System Menu**
	x		1: RDGS STORE	Enable / Disable Transfer of Readings to Memory
		x	ON	enable storage
		x	OFF	disable storage
	x		2: SAVED READINGS	Recall Readings Stored In Memory
		x	Read Values	stored values
	x		3: ERROR	Read Error Message
		x	Message String	text of error message
	x		4: TEST	Perform Self-test of System
		x	TEST ALL	run all tests
		x	TEST 6XX XX = 03-24	run specified test
	x		5: DISPLAY	Enable / Disable Front Panel Display
		x	ON	enable front panel display
		x	OFF	disable front panel display
	x		6: BEEP	Enable / Disable Beep
		x	ON	enable beep
		x	OFF	disable beep
	x		7: COMMA	Enable / Disable Readability Comma
		x	ON	enable readability comma
		x	OFF	disable readability comma
	x		8: REVISION	Read Firmwave Revision Code
		x	Message String	revision code
x			**E: I / O Menu**	**Input / Output Menu**
	x		1: HPIB ADDR	Set Address of Instrument on IEEE-488 (HPIB) Bus
		x	Enter Value	IEEE-488 (HPIB) address of instrument
	x		2: INTERFACE	Select Digital Interface Type
		x	HPIB / 488	IEEE-488 (HPIB) parallel interface
		x	RS-232	RS-232 serial interface
	x		3: BAUD RATE	Set Baud Rate of RS-232 Interface
		x	Select Value	300, 600, 1200, 2400, 4800, or 9600 baud
	x		4: PARITY	Set Parity of RS-232 Interface
		x	NONE: 8 BITS	8 data bits per character and no parity bit
		x	ODD: 7 BITS	7 data bits per character plus odd parity bit
		x	EVEN: 7 BITS	7 data bits per character plus even parity bit
	x		5: LANGUAGE	Set Interface Language – Only One Choice
		x	SCPI	Standard Commands for Programmable Instruments
x			**F: CAL MENU**	**Calibration Menu**
	x		1: SECURED / UNSECURED	Secured Mode Restricts Calibration
		x	Enter Value	numerical code to allow calibration
	x		3: CAL COUNT	Calibration Count
		x	Read Value	number of times self-calibration performed
	x		4: MESSAGE	Calibration Message
		x	Message String	message text

Figure D.3: Digital multimeter menu reference sheet.

Table D.3: Digital multimeter measurement menu.

Command or parameter	**Description**
A: MEAS MENU	Measurement menu.
1: AC FILTER	Set break frequency of the high-pass filter used on the input.
SLOW: 3 HZ	3 Hz high-pass break frequency; 7 s settling time.
MED: 20 HZ	20 Hz high-pass break frequency; 1 s settling time.
FAST: 200 HZ	200 Hz high-pass break frequency; 0.1 s settling time.
2: CONTINUITY	Set resistance value below which continuity is signaled.
Enter Value	Any value between 1 and 1000 Ω; factory value is 10 Ω.
3: INPUT R	Set input impedance.
10 MOHM	10 MΩ input impedance on all voltage ranges.
> 10 GOHM	10 MΩ input impedance on 100 V and 1000 V ranges; greater than 10 GΩ input impedance on 100 mV, 1 V, and 10 V ranges.
4: RATIO FUNCTION	Enable DCV:DCV ratio measurement.
5: RESOLUTION	Set resolution of readings.
FAST 4 DIGIT	4-1/2 digits; 400 μs integration time; autozero off.
SLOW 4 DIGIT	4-1/2 digits; 1 power-line cycle integration time; autozero on.
FAST 5 DIGIT	5-1/2 digits; 3 ms integration time; autozero off.
SLOW 5 DIGIT	5-1/2 digits; 10 power-line cycle integration time; autozero on.
FAST 6 DIGIT	6-1/2 digits; 10 power-line cycle integration time; autozero on.
SLOW 6 DIGIT	6-1/2 digits; 100 power-line cycle integration time; autozero on.

Table D.4: Digital multimeter math and trigger menus.

Command or parameter	**Description**
B: MATH MENU	Math menu.
1: MIN-MAX	Read minimum, maximum, average, and number of readings.
Read Value	Minimum value during averaging period.
Read Value	Maximum value during averaging period.
Read Value	Average value during averaging period.
Read Value	Number of readings during averaging period.
2: NULL VALUE	Set value stored in null register.
Set Value	Null value.
3: dB REL	Set value stored in dB relative register.
Set Value	Any value between -200 and +200 dB.
4: dBm REF R	Set the dBm reference resistance value.
Select Value	50, 75, 93, 110, 124, 125, 135, 150, 250, 300, 600, 800, 900, 1000, 1200, or 8000 Ω.
C: TRIG MENU	Trigger menu.
1: READ HOLD	Set reading hold sensitivity.
Select Value	0.01%, 0.1%, 1%, or 10%; 0.1% is default.
2: TRIG DELAY	Set trigger delay.
Enter Value	1 to 3600 s or auto.
3: N SAMPLES	Set number of samples per trigger.
Enter Value	1 to 50,000.

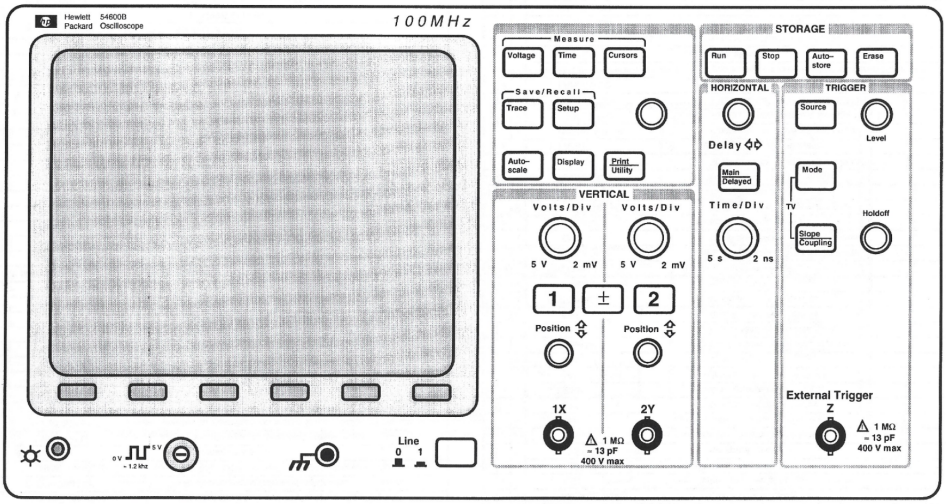
Table D.5: Digital multimeter system menu.

Command or parameter	**Description**
D: SYS MENU	System menu.
1: RDGS STORE	Enable or disable transfer of readings to memory.
ON	Enable storage.
OFF	Disable storage.
2: SAVED READINGS	Recall readings stored in memory.
Read Values	Stored values.
3: ERROR	Read error message.
Message String	Text of error message.
4: TEST	Perform self-test of system.
TEST ALL	Run all tests.
TEST 6XX, XX = 03–24	Run specified test.
5: DISPLAY	Enable or disable front-panel display.
ON	Enable front-panel display.
OFF	Disable front-panel display.
6: BEEP	Enable or disable beep.
ON	Enable beep.
OFF	Disable beep.
7: COMMA	Enable or disable readability comma.
ON	Enable readability comma.
OFF	Disable readability comma.
8: REVISION	Read firmware revision code.
Message String	Revision code.

Table D.6: Digital multimeter input/output and calibration menus.

Command or parameter	**Description**
E: I/O Menu	Input/output menu.
1: HPIB ADDR	Set address of instrument on IEEE-488 (HPIB) bus.
Enter Value	IEEE-488 (HPIB) address of instrument.
2: INTERFACE	Select digital interface type.
HPIB / 488	IEEE-488 (HPIB) parallel interface.
RS-232	RS-232 serial interface.
3: BAUD RATE	Set baud rate of RS-232 interface.
Select Value	300, 600, 1200, 2400, 4800, or 9600 baud.
4: PARITY	Set parity of RS-232 interface.
NONE: 8 BITS	8 data bits per character and no parity bit.
ODD: 7 BITS	7 data bits per character plus odd parity bit.
EVEN: 7 BITS	7 data bits per character plus even parity bit.
5: LANGUAGE	Set interface language - only one choice.
SCPI	Standard Commands for Programmable Instruments.
F: CAL MENU	Calibration menu.
1: SECURED / UNSECURED	Secured mode restricts calibration.
Enter Value	Numerical code to allow calibration.
3: CAL COUNT	Calibration count.
Read Value	Number of times self-calibration performed.
4: MESSAGE	Calibration message.
Message String	Message text.

D.3 HP 54600 Digital Two-Channel Oscilloscope



Hewlett Packard 54600 Digital Two-channel Oscilloscope

Figure D.4: Hewlett Packard 54600 digital two-channel oscilloscope front panel.

Table D.7: Accessible transcription of the HP 54600 oscilloscope front-panel areas.

Front-panel area	**Controls, connectors, or labels shown**
Display and soft keys	Large graticule display with six unlabeled soft keys directly below the screen.
Measurement section	Voltage, time, and cursors keys; save/recall trace and setup keys; auto-scale, display, and print/utility keys.
Vertical section	Channel 1 and channel 2 volts-per-division knobs; channel 1 and channel 2 select keys; position controls; channel input connectors labeled 1X and 2Y.
Horizontal section	Delay control, main/delayed key, and time-per-division knob.
Storage section	Run, stop, auto-store, and erase keys.
Trigger section	Source, level, mode, slope/coupling, and holdoff controls; external trigger connector labeled Z.
Input ratings	Channel inputs are labeled 1 MΩ, 13 pF, and 400 V maximum. The external trigger input is also labeled 1 MΩ, 13 pF, and 400 V maximum.

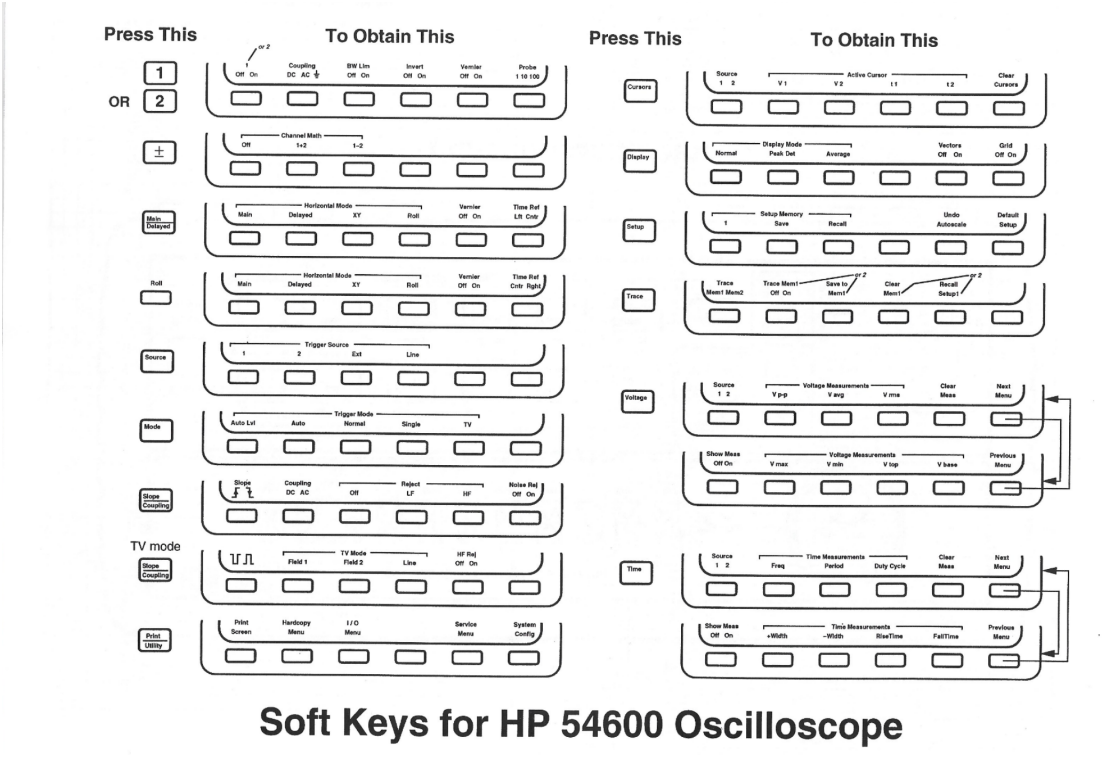


Figure D.5: Soft keys for the HP 54600 oscilloscope.

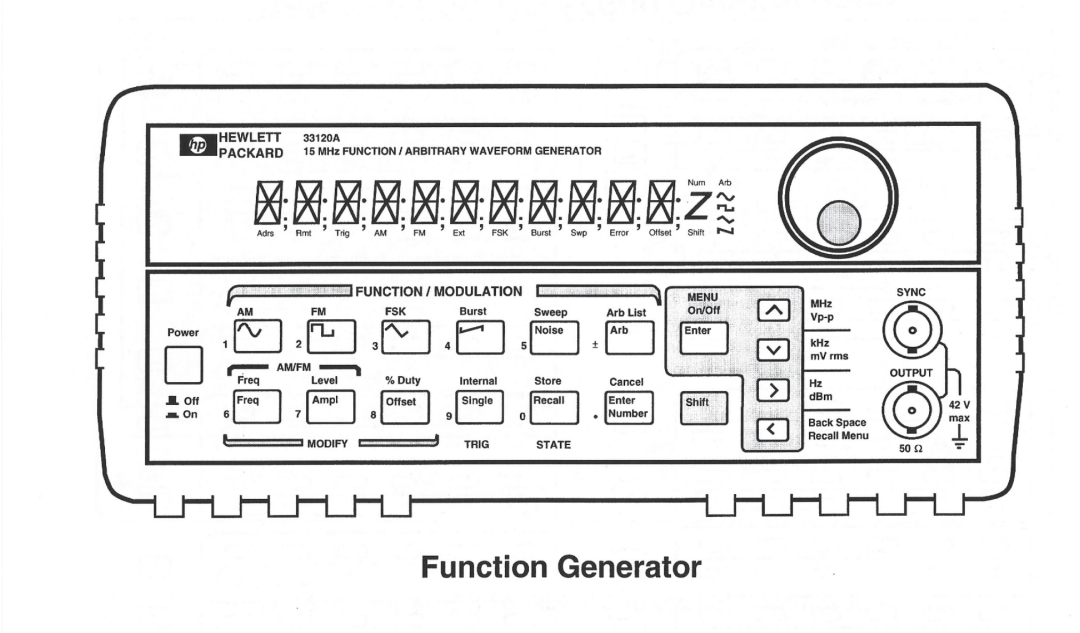
Table D.8: HP 54600 oscilloscope soft-key map, left side.

Press this	**Soft-key group**	**Options shown**
1 or 2	Channel setup	Off/On; coupling DC, AC, or ground; bandwidth limit Off/On; invert Off/On; vernier Off/On; probe 1, 10, or 100.
±	Channel math	Off; 1+2; 1-2.
Main/Delayed	Horizontal mode	Main; Delayed; XY; Roll; vernier Off/On; time reference left or center.
Roll	Horizontal mode	Main; Delayed; XY; Roll; vernier Off/On; time reference center or right.
Source	Trigger source	Channel 1; channel 2; external; line.
Mode	Trigger mode	Auto level; auto; normal; single; TV.
Slope/Coupling	Trigger coupling and slope	Slope rising or falling; coupling DC or AC; off; reject LF or HF; noise reject Off/On.
Slope/Coupling in TV mode	TV trigger mode	TV polarity or line-field selection; field 1; field 2; line; HF reject Off/On.
Print/Utility	Utility menus	Print screen; hardcopy menu; I/O menu; service menu; system configuration.

Table D.9: HP 54600 oscilloscope soft-key map, right side.

Press this	**Soft-key group**	**Options shown**
Cursors	Cursor menu	Source 1 or 2; active cursor V1, V2, t1, or t2; clear cursors.
Display	Display mode	Normal; peak detect; average; vectors Off/On; grid Off/On.
Setup	Setup memory	Setup memory 1; save; recall; undo autoscale; default setup.
Trace	Trace memory	Trace memory 1 or 2; trace memory Off/On; save to memory 1 or 2; clear memory 1 or 2; recall setup 1 or 2.
Voltage	Voltage measurements, first menu	Source 1 or 2; peak-to-peak voltage; average voltage; RMS voltage; clear measurement; next menu.
Voltage	Voltage measurements, second menu	Show measurement Off/On; maximum voltage; minimum voltage; top voltage; base voltage; previous menu.
Time	Time measurements, first menu	Source 1 or 2; frequency; period; duty cycle; clear measurement; next menu.
Time	Time measurements, second menu	Show measurement Off/On; positive width; negative width; rise time; fall time; previous menu.

D.4 Function Generator



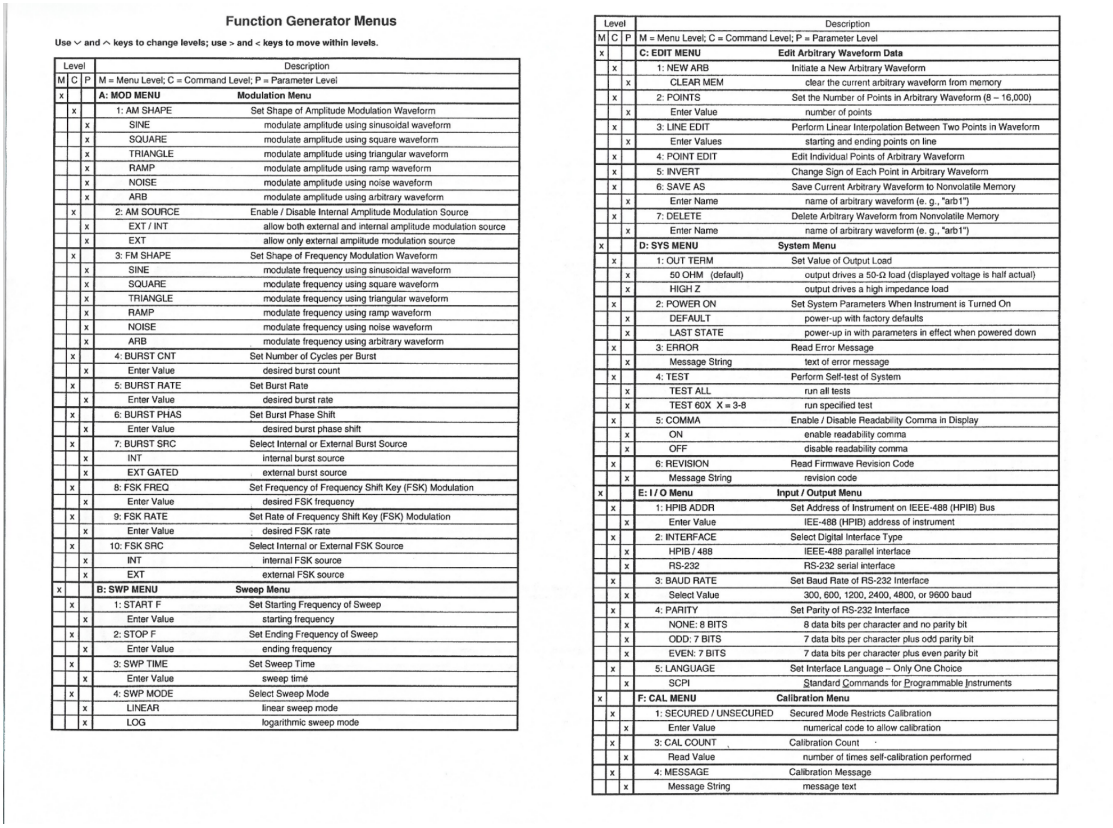
Function Generator

Figure D.6: Function generator front panel.

Use the down and up keys to change levels. Use the right and left keys to move within levels. In the original table, **M** denotes menu level, **C** denotes command level, and **P** denotes parameter level.

Table D.10: Accessible transcription of the function generator front-panel areas.

Front-panel area	**Controls, connectors, or labels shown**
Display and indicators	Long numeric display with annunciators including address, remote, trigger, AM, FM, external, FSK, burst, sweep, error, offset, and shift; display also marks numeric and arbitrary waveform status.
Function and modulation keys	AM, FM, FSK, burst, sweep, noise, and arbitrary waveform keys.
Modify keys	Frequency, amplitude, and offset keys; percent duty and level functions are also indicated.
Trigger and state keys	Internal, single, store, recall, cancel, enter number, and shift keys.
Menu navigation	Menu on/off and enter area; up, down, left, and right arrow keys; large rotary knob for changing selected values.
Units and connectors	Unit labels include MHz, V peak-to-peak, kHz, mV RMS, Hz, and dBm. The right side contains a sync connector and an output connector labeled 50 Ω and 42 V maximum.



Function Generator Menus

Use ∨ and ∧ keys to change levels; use > and < keys to move within levels.

Level			Description	
M	C	P	M = Menu Level; C = Command Level; P = Parameter Level	
x			**A: MOD MENU**	**Modulation Menu**
	x		1: AM SHAPE	Set Shape of Amplitude Modulation Waveform
		x	SINE	modulate amplitude using sinusoidal waveform
		x	SQUARE	modulate amplitude using square waveform
		x	TRIANGLE	modulate amplitude using triangular waveform
		x	RAMP	modulate amplitude using ramp waveform
		x	NOISE	modulate amplitude using noise waveform
		x	ARB	modulate amplitude using arbitrary waveform
	x		2: AM SOURCE	Enable / Disable Internal Amplitude Modulation Source
		x	EXT / INT	allow both external and internal amplitude modulation source
		x	EXT	allow only external amplitude modulation source
	x		3: FM SHAPE	Set Shape of Frequency Modulation Waveform
		x	SINE	modulate frequency using sinusoidal waveform
		x	SQUARE	modulate frequency using square waveform
		x	TRIANGLE	modulate frequency using triangular waveform
		x	RAMP	modulate frequency using ramp waveform
		x	NOISE	modulate frequency using noise waveform
		x	ARB	modulate frequency using arbitrary waveform
	x		4: BURST CNT	Set Number of Cycles per Burst
		x	Enter Value	desired burst count
	x		5: BURST RATE	Set Burst Rate
		x	Enter Value	desired burst rate
	x		6: BURST PHAS	Set Burst Phase Shift
		x	Enter Value	desired burst phase shift
	x		7: BURST SRC	Select Internal or External Burst Source
		x	INT	internal burst source
		x	EXT GATED	external burst source
	x		8: FSK FREQ	Set Frequency of Frequency Shift Key (FSK) Modulation
		x	Enter Value	desired FSK frequency
	x		9: FSK RATE	Set Rate of Frequency Shift Key (FSK) Modulation
		x	Enter Value	desired FSK rate
	x		10: FSK SRC	Select Internal or External FSK Source
		x	INT	internal FSK source
		x	EXT	external FSK source
x			**B: SWP MENU**	**Sweep Menu**
	x		1: START F	Set Starting Frequency of Sweep
		x	Enter Value	starting frequency
	x		2: STOP F	Set Ending Frequency of Sweep
		x	Enter Value	ending frequency
	x		3: SWP TIME	Set Sweep Time
		x	Enter Value	sweep time
	x		4: SWP MODE	Select Sweep Mode
		x	LINEAR	linear sweep mode
		x	LOG	logarithmic sweep mode

Level			Description	
M	C	P	M = Menu Level; C = Command Level; P = Parameter Level	
x			**C: EDIT MENU**	**Edit Arbitrary Waveform Data**
	x		1: NEW ARB	Initiate a New Arbitrary Waveform
		x	CLEAR MEM	clear the current arbitrary waveform from memory
	x		2: POINTS	Set the Number of Points in Arbitrary Waveform (8 – 16,000)
		x	Enter Value	number of points
	x		3: LINE EDIT	Perform Linear Interpolation Between Two Points in Waveform
		x	Enter Values	starting and ending points on line
	x		4: POINT EDIT	Edit Individual Points of Arbitrary Waveform
	x		5: INVERT	Change Sign of Each Point in Arbitrary Waveform
	x		6: SAVE AS	Save Current Arbitrary Waveform to Nonvolatile Memory
		x	Enter Name	name of arbitrary waveform (e. g., "arb1")
	x		7: DELETE	Delete Arbitrary Waveform from Nonvolatile Memory
		x	Enter Name	name of arbitrary waveform (e. g., "arb1")
x			**D: SYS MENU**	**System Menu**
	x		1: OUT TERM	Set Value of Output Load
		x	50 OHM (default)	output drives a 50-Ω load (displayed voltage is half actual)
		x	HIGH Z	output drives a high impedance load
	x		2: POWER ON	Set System Parameters When Instrument is Turned On
		x	DEFAULT	power-up with factory defaults
		x	LAST STATE	power-up in with parameters in effect when powered down
	x		3: ERROR	Read Error Message
		x	Message String	text of error message
	x		4: TEST	Perform Self-test of System
		x	TEST ALL	run all tests
		x	TEST 60X X = 3-8	run specified test
	x		5: COMMA	Enable / Disable Readability Comma in Display
		x	ON	enable readability comma
		x	OFF	disable readability comma
	x		6: REVISION	Read Firmwave Revision Code
		x	Message String	revision code
x			**E: I / O Menu**	**Input / Output Menu**
	x		1: HPIB ADDR	Set Address of Instrument on IEEE-488 (HPIB) Bus
		x	Enter Value	IEE-488 (HPIB) address of instrument
	x		2: INTERFACE	Select Digital Interface Type
		x	HPIB / 488	IEEE-488 parallel interface
		x	RS-232	RS-232 serial interface
	x		3: BAUD RATE	Set Baud Rate of RS-232 Interface
		x	Select Value	300, 600, 1200, 2400, 4800, or 9600 baud
	x		4: PARITY	Set Parity of RS-232 Interface
		x	NONE: 8 BITS	8 data bits per character and no parity bit
		x	ODD: 7 BITS	7 data bits per character plus odd parity bit
		x	EVEN: 7 BITS	7 data bits per character plus even parity bit
	x		5: LANGUAGE	Set Interface Language – Only One Choice
		x	SCPI	Standard Commands for Programmable Instruments
x			**F: CAL MENU**	**Calibration Menu**
	x		1: SECURED / UNSECURED	Secured Mode Restricts Calibration
		x	Enter Value	numerical code to allow calibration
	x		3: CAL COUNT	Calibration Count
		x	Read Value	number of times self-calibration performed
	x		4: MESSAGE	Calibration Message
		x	Message String	message text

Figure D.7: Function generator menu reference sheet.

Table D.11: Function generator modulation and sweep menus.

Command or parameter	**Description**
A: MOD MENU	Modulation menu.
1: AM SHAPE	Set shape of amplitude modulation waveform.
SINE	Modulate amplitude using sinusoidal waveform.
SQUARE	Modulate amplitude using square waveform.
TRIANGLE	Modulate amplitude using triangular waveform.
RAMP	Modulate amplitude using ramp waveform.
NOISE	Modulate amplitude using noise waveform.
ARB	Modulate amplitude using arbitrary waveform.
2: AM SOURCE	Enable or disable internal amplitude modulation source.
EXT / INT	Allow both external and internal amplitude modulation sources.
EXT	Allow only external amplitude modulation source.
3: FM SHAPE	Set shape of frequency modulation waveform.
SINE	Modulate frequency using sinusoidal waveform.
SQUARE	Modulate frequency using square waveform.
TRIANGLE	Modulate frequency using triangular waveform.
RAMP	Modulate frequency using ramp waveform.
NOISE	Modulate frequency using noise waveform.
ARB	Modulate frequency using arbitrary waveform.
4: BURST CNT	Set number of cycles per burst.
Enter Value	Desired burst count.
5: BURST RATE	Set burst rate.
Enter Value	Desired burst rate.
6: BURST PHAS	Set burst phase shift.
Enter Value	Desired burst phase shift.
7: BURST SRC	Select internal or external burst source.
INT	Internal burst source.
EXT GATED	External burst source.
8: FSK FREQ	Set frequency of frequency-shift-keying modulation.
Enter Value	Desired FSK frequency.
9: FSK RATE	Set rate of frequency-shift-keying modulation.
Enter Value	Desired FSK rate.
10: FSK SRC	Select internal or external FSK source.
INT	Internal FSK source.
EXT	External FSK source.
B: SWP MENU	Sweep menu.
1: START F	Set starting frequency of sweep.
Enter Value	Starting frequency.
2: STOP F	Set ending frequency of sweep.
Enter Value	Ending frequency.
3: SWP TIME	Set sweep time.
Enter Value	Sweep time.
4: SWP MODE	Select sweep mode.
LINEAR	Linear sweep mode.
LOG	Logarithmic sweep mode.

Table D.12: Function generator arbitrary waveform edit and system menus.

Command or parameter	**Description**
C: EDIT MENU	Edit arbitrary waveform data.
1: NEW ARB	Initiate a new arbitrary waveform.
CLEAR MEM	Clear the current arbitrary waveform from memory.
2: POINTS	Set the number of points in arbitrary waveform, from 8 to 16,000.
Enter Value	Number of points.
3: LINE EDIT	Perform linear interpolation between two points in waveform.
Enter Values	Starting and ending points on line.
4: POINT EDIT	Edit individual points of arbitrary waveform.
5: INVERT	Change sign of each point in arbitrary waveform.
6: SAVE AS	Save current arbitrary waveform to nonvolatile memory.
Enter Name	Name of arbitrary waveform, for example “arb1”.
7: DELETE	Delete arbitrary waveform from nonvolatile memory.
Enter Name	Name of arbitrary waveform, for example “arb1”.
D: SYS MENU	System menu.
1: OUT TERM	Set value of output load.
50 OHM (default)	Output drives a 50-Ω load; displayed voltage is half actual.
HIGH Z	Output drives a high-impedance load.
2: POWER ON	Set system parameters when instrument is turned on.
DEFAULT	Power up with factory defaults.
LAST STATE	Power up with parameters in effect when powered down.
3: ERROR	Read error message.
Message String	Text of error message.
4: TEST	Perform self-test of system.
TEST ALL	Run all tests.
TEST 60X, X = 3–8	Run specified test.
5: COMMA	Enable or disable readability comma in display.
ON	Enable readability comma.
OFF	Disable readability comma.
6: REVISION	Read firmware revision code.
Message String	Revision code.

Table D.13: Function generator input/output and calibration menus.

Command or parameter	**Description**
E: I/O Menu	Input/output menu.
1: HPIB ADDR	Set address of instrument on IEEE-488 (HPIB) bus.
Enter Value	IEEE-488 (HPIB) address of instrument.
2: INTERFACE	Select digital interface type.
HPIB / 488	IEEE-488 parallel interface.
RS-232	RS-232 serial interface.
3: BAUD RATE	Set baud rate of RS-232 interface.
Select Value	300, 600, 1200, 2400, 4800, or 9600 baud.
4: PARITY	Set parity of RS-232 interface.
NONE: 8 BITS	8 data bits per character and no parity bit.
ODD: 7 BITS	7 data bits per character plus odd parity bit.
EVEN: 7 BITS	7 data bits per character plus even parity bit.
5: LANGUAGE	Set interface language - only one choice.
SCPI	Standard Commands for Programmable Instruments.
F: CAL MENU	Calibration menu.
1: SECURED / UNSECURED	Secured mode restricts calibration.
Enter Value	Numerical code to allow calibration.
3: CAL COUNT	Calibration count.
Read Value	Number of times self-calibration performed.
4: MESSAGE	Calibration message.
Message String	Message text.

D.5 Dynamic Spectrum Analyzer

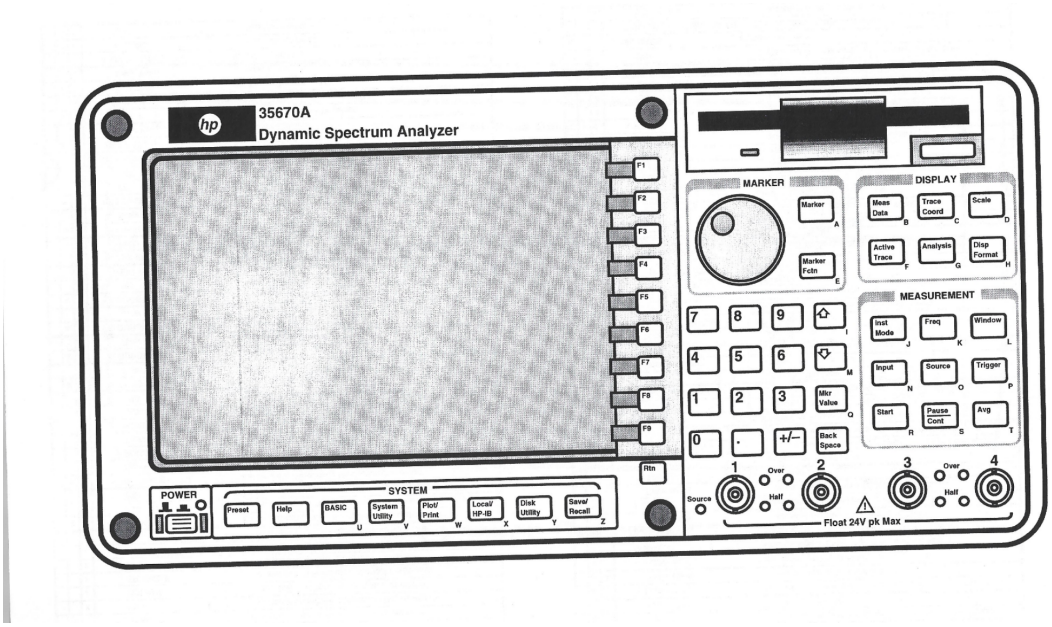


Figure D.8: Dynamic spectrum analyzer front panel.

Table D.14: Accessible transcription of the dynamic spectrum analyzer front-panel areas.

Front-panel area	**Controls, connectors, or labels shown**
Display and soft keys	Large display with vertical function keys F1 through F9 and a Return key on the right side.
System keys	Power switch; preset; help; BASIC; system utility; plot/print; local HP-IB; disk utility; and save/recall.
Marker controls	Marker knob, Marker key, and Marker Function key.
Display controls	Measurement data, trace coordinate, scale, active trace, analysis, and display format keys.
Measurement controls	Instrument mode, frequency, window, input, source, trigger, start, pause/continue, and average keys.
Numeric and editing keys	Numeric keypad 0 through 9, decimal point, plus/minus key, arrow keys, marker value key, and backspace key.
Connectors	Four lower connectors labeled 1, 2, 3, and 4; source is marked near connector 1. Printed input warning indicates floating 24 V peak maximum.