University of Nebraska - Lincoln
DigitalCommons@University of Nebraska - Lincoln

P. F. (Paul Frazer) Williams Publications | Electrical & Computer Engineering, Department of

April 1993

Introduction to Electrical Engineering and the Art of Problem Solving: Volume I

P. Frazer Williams
University of Nebraska - Lincoln, pfw@moi.unl.edu

Follow this and additional works at: http://digitalcommons.unl.edu/elecengwilliams

 Part of the Electrical and Computer Engineering Commons

Williams, P. Frazer, "Introduction to Electrical Engineering and the Art of Problem Solving: Volume I" (1993). *P. F. (Paul Frazer) Williams Publications*. 2.
http://digitalcommons.unl.edu/elecengwilliams/2

This Article is brought to you for free and open access by the Electrical & Computer Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in P. F. (Paul Frazer) Williams Publications by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

INTRODUCTION TO ELECTRICAL ENGINEERING AND THE ART OF PROBLEM SOLVING

Notes for EEngr 121 and 122

Volume I

©*Frazer Williams*
University of Nebraska-Lincoln
Fall, 1993

PREFACE

These notes were written to accompany a two-semester introductory course in Electrical Engineering. The primary goal of the course is to introduce you to the art of technical problem solving. Conventional Electrical Engineering curricula, in favorable cases at least, produce engineers who are reasonably adept at operating many of the tools important to the field, such as mathematics and circuit analysis. Often, however, these engineers lack skill in choosing which tools to use and in designing a plan of attack to solve problems which are new to them, even though they have all the information and skills needed to fabricate a satisfactory solution. Problem solving is more of an art than a science, and it can only be learned through experience. I hope these notes and the homework exercises will help you to develop this important ability early in your academic career.

Problem solving is not something that can be learned by just listening to some lectures or reading some material, no matter how well prepared. I don't know of any specific set of reasoning or rules I can give you to make you proficient at it. You just have to jump in and flounder around for a while. I can give you some suggestions, however, which may keep you from drowning.

1. Make sure you understand the problem. If it's a homework or a test problem, make sure you understand what all the words mean and how they fit together. Before you start looking for "the answer," make sure you understand what the problem is.
2. Spend a little time deciding on a plan of attack. As you go along check frequently to make sure that you are following your plan. If it appears that the original plan won't work, change it; but do so intentionally, not accidentally.
3. Plan on spending some "quality" time on the homework. As a guideline, you should expect to spend on the average two to three hours outside of class for every lecture hour.
4. Your goal in doing homework is to arrive at an answer through a process you create, not just to get an answer, even if it's right.
5. If, after spending some time, you can't get anything to work out on a problem, seek help from the instructor, teaching assistant, or another student. Use such help wisely. Again, the main goal is not to get an answer, but rather to see where you went wrong or what idea you missed.
6. Try to view each homework problem as an opportunity to hone your problem solving skills. Remember that there is a good chance that problems similar, but not identical, to the homework problems will be on the tests. If you were unable to solve a problem in the relaxed atmosphere of home, it is unlikely you will be able to solve its cousin in the stressed atmosphere of an exam.
7. There are often several reasonable ways to solve a problem. I suggest trying to solve homework and example problems in more than one way. This is a good way to study for tests. If you work a problem in two different ways that both seem correct to you, and get different answers, try to figure out why. If after a little time both approaches still seem correct, talk to the instructor.

In writing the notes, I've tried to find non-trivial problems for which you already know, or I can easily tell you, everything needed to obtain a solution. The difficult part is sorting the useful information from the rest of the stuff you know, and putting it all together in a way that works. The important aspect of these discussions is the *process*, not the result. If the primary goal were to obtain the answer, often the best method of doing so would be something other than that in the notes. (In cases in which the method discussed is obviously a poor choice, I've tried to point that out, along with at least some directions pointing toward the better choice.)

The notes have two secondary goals. The first is to show you some of the ideas that you will encounter a little later in your academic career, and to solidify your grasp of some ideas you have seen already. Seeing the new ideas now and giving them time to "rattle around" in your head for a while will make them easier to master when you encounter them later in more specialized courses. For example

Chapter 8 is devoted entirely to the analysis of simple circuits in order to determine currents and voltages. The purpose here is not to make you proficient at circuit analysis, but rather to introduce you to the basic ideas, such as just what is a current or a voltage. Then, when you encounter the subject again in considerably more detail, you will have some understanding and familiarity to fall back on.

The other secondary goal is to provide you with skill in using a computer. In the first semester of the course, your interaction with a computer will be almost entirely through a spreadsheet program. This is a pretty specialized program which turns out to be surprisingly useful in engineering. It allows you to graph data and functions, and to do rather complex arithmetic calculations on large quantities of data easily. I hope that you will come to view the spreadsheet as a tool you can use later to help you understand things and to avoid some of the arithmetic drudgery. This is the easiest goal of the course to achieve. I think you should find the spreadsheet easy to learn and entertaining to use (it was designed that way). In the second semester, you will be introduced to a more general-purpose program, called the C compiler. C is a computer language somewhat similar to FORTRAN, Pascal, or BASIC which allows you to write your own programs, and thereby to tell the computer exactly what you want it to do. You will probably not emerge from the course as a skilled C programmer, but I hope you will be able to use C to solve many of the problems which you are likely to encounter later in your career. Perhaps most importantly, I hope you will have a good idea about just what can be done with C, and what can't, so that you will be able to further develop your skill at using C as the need arises.

I hope you will find these notes interesting, and at least in some parts fun. The core of the material in the notes should be accessible to all students in the class, but I've also tried to include material which even the best students will find challenging. This material usually comes at the end of the discussion on the topic.

On a personal note, I consider myself to be very fortunate that part of the job for which I get paid is something I really enjoy doing. I really like figuring out how to make something, designing it, putting it together, and finally seeing it work. The "something" can be a physical thing such as an electronic circuit or a piece of mechanical equipment, or something more ephemeral like a clever solution to a mathematical problem or a computer program. The part of the process I do *not* like is the part involving doing nearly the same thing I've done many times before, such as soldering the circuit together or laying out and drilling the holes, or doing the algebra or actually typing the program into the computer. The part I *do* like is getting the idea, and then seeing it actually work. I even liked putting these notes together! I really disliked writing them and getting everything just right, put thinking up how to present the material and seeing it go together was enjoyable.

It seems to me that many students go through their college careers choosing to see only the boring, unpleasant stuff, like the mechanics of circuit analysis. You are going to have to do your share of algebra and the like, but I hope that when given the chance, you will also look around and see the fun stuff, as well. It's all over the place. In writing these notes, I've tried to show you a few of the things I've found enjoyable.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. AN INTRODUCTION TO COMPUTERS

In EE121 you will be using computers to help you solve problems. I expect that some students in the class have had a great deal of experience with computers, and others have never used one before. The purpose of the first three chapters of these notes is to provide the general information you will need to make use of the computers in the EE121 computer lab. For some of you, this may be a review. As you will probably realize, I've left out a lot of interesting and useful material. Next semester, in EE122, you will learn much more about how computers operate, and later in this course we will cover the design of logic circuits, which are the building blocks of electronic computers. Because so much is left out, you may find parts of the discussion unsatisfying. For example, I will talk about the computer writing numbers to its memory or issuing a command to a disk drive, but I won't say anything about just how this is done. If you are really curious, much of the material is covered in considerable detail in the text for this class, and you can read about it there. I don't think the information will be needed, however, to use the computers in this class.

1.1 The Central Processing Unit (CPU)

Figure 1−1 shows a schematic block drawing of a generic computer system.

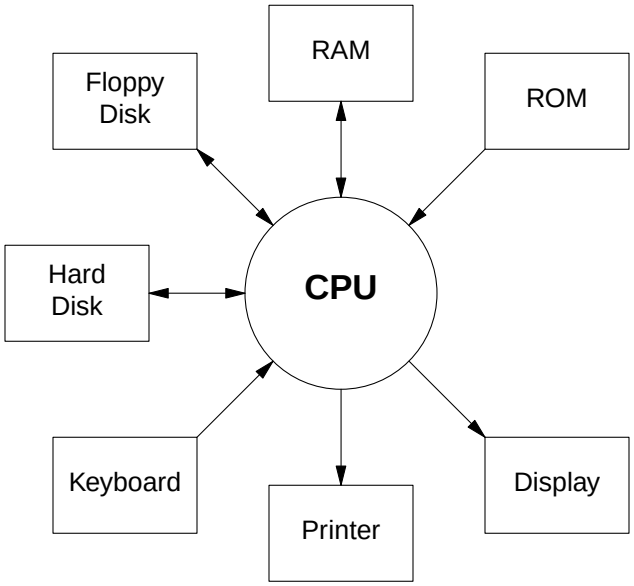

Figure 1−1. Schematic drawing of a typical computer system architecture. The arrows indicate possible directions of information flow.

The Central Processor Unit (CPU) is at the heart of the system. It is a complicated electronic circuit that operates on data. For example, the CPU can add two numbers, or it can test to see if one number is larger than another. Typically, a CPU has a repertoire of several hundred operations which it knows how to do. These operations are very basic, and many operations are just minor variations on others. The unit is very good at carrying out these operations; it can do them rapidly and (almost) never makes an error. Typically, a CPU can carry out an operation in less than one one-hundred-thousandth (10−5) of a second, and really fast CPU's can do an operation in about one one-hundred-millionth (10−8) of a second. The CPU must be told exactly what operations to perform. If you wish to use the computer to solve some problem, your task

is to figure out how the problem could be solved by using only the operations the CPU can do, and then to put them in a properly ordered list. Such a list is called a *program*. Even for a simple application, writing such a program can be a long and tedious procedure. You have to get every little detail right! Fortunately there are a number of tool programs such as spreadsheet, editor, and compiler programs to help.

1.2 Electronic Memory Units (RAM and ROM)

Most CPU's include small areas in which a few numbers and instructions can be stored. Since these areas are seldom large enough to store enough information to do anything useful, computers have auxiliary units for storing much larger volumes of information. The information stored typically includes both numbers or other data to be operated on by the CPU, as well as a list of instructions to the CPU. Several types of devices are available for storing this information, each with both advantages and disadvantages. Electronic memory circuits form one class of such devices. There are a number of different types of electronic memory circuits, but we'll discuss here only two: RAM and ROM. The first, Random Access Memory or RAM, offers fast access (both reading and writing of information) and high information storage density. Presently, a character can typically be read or written in less than one ten-millionth (10−7) of a second, and one can store a manuscript of one million (106) characters in an area of a few square centimeters (cm2). The primary disadvantage of memory of this type is that the information stored in it disappears when the electrical power is shut off. The other type of circuit, Read Only Memory or ROM, does not suffer this problem. It also allows fast reading of stored information, and provides high information storage density. Once the information is stored in such a circuit, it is permanent. You can turn off the power and then come back a year later and it will still be there. This advantage is also the primary disadvantage of the circuit. Once information is written into the circuit it cannot be erased or modified. For this reason, these circuits are commonly used to store information about a computer system which does not change, and must survive when the computer is turned off.

1.3 Disk Drives

The other main type of information storage device is called a *disk drive*. In such devices information is stored as patterns of magnetization in thin films of magnetic material coated on circular disks. The devices work similarly to a tape recorder, the main difference being that the information is stored on a thin disk which is rotated rapidly about its center, rather than on a long, thin strip (the tape) of flexible material which is drawn through the recorder at a constant speed. In some disk drives the disk containing the information can be removed, and another disk inserted. The *floppy* disk drive is the most common example of such a device. In other drives, the disk is permanently sealed inside the drive, and cannot be removed. Such drives are called *hard* disk drives. The terms, floppy and hard, refer to the flexibility of the disk, not to whether or not it can removed. Some hard disks can be removed, but these are becoming increasingly unusual.

Disk drives offer the advantage of reasonably permanent information storage, coupled with the capability to erase and rewrite information. The principal disadvantage of disk drives is that reading or writing information to them is slow in comparison to electronic memories. It typically takes a few hundredths ($\sim 2 \times 10^{-2}$) of a second to find the requested information on a hard disk, and then it takes more than one millionth (10^{-6}) of a second to read or write each character. Floppy disk drives are quite a bit slower. Hard drives typically have storage capacities of tens to hundreds of millions (107-108) of characters. Capacities of floppy disks are around one million (106) characters.

1.4 Other Devices Connected to the CPU

Besides storage devices, a computer system typically has several other auxiliary units connected to the CPU. Most systems have a keyboard and a display unit connected to provide for interaction with the human user, and many have a printer connected to allow for a permanent printout in human-readable form of the results of running a program. Some computer systems have a *mouse* which aids in pointing to a specific item or place on the display screen. Sometimes the computer is connected to others through a

networking unit, or through a data port called a *serial* or *RS-232* port, or through the telephone system using a unit called a *modem*. Although the details of how these units work can get pretty complicated, at the level we will be using them, usage is transparent, and I won't discuss them further here.

1.5 How a Computer Operates

There are two types of information typically stored in memory: data to be operated on, and a list of instructions (the program) for the CPU. The computer stores both types of information in different parts of the same storage devices. For long term storage, both the program and the data may be saved on a disk drive, but because of the slow speed of disks, the computer usually transfers both the program and the data to the high-speed electronic memory when it is ready to execute a program.

The electronic storage space available in most computer systems is quite large, and a scheme to organize the memory is required in order to be able to find a desired piece of information. For this purpose, computer designers break up storage space into small spaces, called bytes, which happen to be just a little larger than needed to store one character of information. Storing a decimal number with the standard number of digits accuracy requires 4 bytes of storage space. Most computer systems have at least half a million (5×106) bytes (also referred to as half a *megabyte*, or 500 *kilobytes*) of RAM information storage space. The computers in the EE121 computer lab have a little more than four million (4×106) bytes (4 megabytes). Each byte of storage is given a unique number, called an *address*, and the CPU can access a specific data or instruction item by simply supplying the address of the desired byte.

In a program, instructions for the CPU to execute are stored sequentially (in the order they are to be executed) in memory. When execution of a program is started, the CPU is given the address of the first instruction in the program (don't worry about how—you will cover that in EE122). It stores this address in an internal space called an instruction counter. When told to start execution, the CPU reads the instruction starting at this address and does whatever it says to do. When finished, the CPU adds the length in bytes of the instruction to the address contained in the instruction counter, and reads the information starting at the resulting address as the next instruction to be executed. The CPU continues reading and executing instructions in this manner until it encounters an instruction that tells it to stop.

1.6 Organization of Data on Disk Drives

Information is also stored on disk drive memory devices in byte-sized chunks, but the scheme for accessing a specific byte of information is somewhat more complicated in this case. (Well, actually, it's quite a bit more complicated, but I don't want to go into the details here.) Basically, the storage scheme is based on the premise that information to be stored on disks usually consists of a fairly large collection of bytes, all related, and all to be accessed at once. For example, programs for the CPU are often stored on a disk. When the program is to be run or executed, the stored information is read from the disk and written into electronic memory (RAM). The whole list of instructions is to be read in one operation, rather than one instruction now and another instruction a little later. As another example, I stored the manuscript for these notes on a disk. I used a specialized program called an *editor* to store the sequence of characters I wanted in RAM, and then when I was satisfied (or more often tired and wanted to rest and come back to it later) I told the editor to write the whole shebang to disk. Later, when I was ready to give it another shot, I told the editor to read the information from the disk, and store it back in RAM just as it had been so that I could add to or modify it.

Anyway, based on this premise, information is usually stored on disk drives as collections of bytes. Each collection is called a *file* (in analogy to storage of information in a file cabinet), and each file is given a name, usually by the user. For example, I stored this manuscript in a file I chose to call `DOS.intro`, and the file containing the program for the editor I used is stored on my hard disk in a file called simply `vi`. Each disk has a special file on it called a directory which is created and updated automatically—you need not know anything about it. The directory contains a list by name of all the files stored on the disk, the corresponding location on the disk where the information in the file is stored, and some other useful

information such as the date and time the file was created. In the directory file there would be an entry containing the name of the file, `DOS.intro`, and where on the disk it is found. When I issue an instruction to my computer to read the file named `DOS.intro` the computer first reads the directory file and searches for the name `DOS.intro`. On finding it, the computer reads the directory to get the associated disk location information and then it tells the disk drive to read the information in these locations and store it in RAM.

1.7 How a Computer Stores Integer Numbers

Consider first how people store numbers. When you write an integer such as 436, one way of thinking about what you mean is $6 \cdot 1 + 3 \cdot 10 + 4 \cdot 100$. This could also be written $6 \cdot 10^0 + 3 \cdot 10^1 + 4 \cdot 10^2$. As I'm sure you know, this scheme can be continued to give meaning to a string of digits of any length. The scheme is based on powers of ten, and is called the *decimal* system. Systems based on other numbers are possible. I understand that the Babylonians used a system based on 60, and that's the reason hours and minutes are divided up into 60 parts instead of some other more convenient division like 100.

No one knows for sure why humans came up so universally with a number system based on ten and not some other number, but it must be related to the fact that we have ten fingers. We can represent ten different digits by associating one digit with each of our fingers. (It is probably not a coincidence that the word "digit" refers both to a number between 0 and 9, and to a finger.) The circuits used in computers can conveniently represent only two digits—it's kind of like they have only two fingers. For this reason, computers universally use a number system based on powers of two to store and operate on numbers. Such a system is called a *binary* system.

A number written in binary consists of a string of 1's and 0's. The scheme is best explained with an example. Consider the number (in binary) $(10011)_2$. (To avoid confusion, I'll use a subscript to indicate the base of the number system I mean.) We decode this number just as we would a number written in decimal, starting at the right end and working to the left.

$$\begin{aligned}(10011)_2 &= 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 \\ &= 1 + 2 + 16 \\ &= (19)_{10}\end{aligned}$$

Any positive integer may be written in this way. I won't discuss it here, but it is also possible to extend the scheme a little to allow storage of negative integers, and to extend it a lot to allow storage of fractional numbers. The scheme commonly used to store fractional numbers is called the *floating point* representation.

Each digit (bigit??) in a number represented in binary can be represented in a computer by an electronic switch which is either on or off (corresponding to 1 or 0). The term for a binary digit is *bit*. A byte consists of 8 bits, and to represent a byte in a computer 8 electronic switches are required, one for each bit. The more bytes a computer uses to store an integer, the larger an integer it can deal with, but the more memory space is required to store a given number of integers. There is not universal agreement on how many bytes should be used to store an integer, but most commonly these days four bytes are used. With this much space, one can represent positive integers up to a little more than 4,000,000,000. Actually, a scheme which allows storage of signed integers is usually used. In this case, one bit is used to represent the sign (positive or negative), and the largest number that can then be stored is a little more than 2,000,000,000.

1.8 How Textual Information is Stored in Computers

Textual information is represented in a computer by assigning to each letter or other character a unique number, and storing text as a sequence of numbers (in binary of course). The most common (at least in countries which use an alphabet similar to that used by English) representation for characters is called ASCII. ASCII stands for American Standard Code for Information Interchange. The only other representation that I'm aware of is called EBCDIC (Extended Binary Coded Decimal Interchange Code I

think) and was a monstrosity promulgated by IBM. Fortunately, it has almost disappeared.

In ASCII, each character is stored in a separate byte. With eight bits, one can represent 256 different characters, many more than are needed for storing text in most occidental languages. Table 1−1 shows the ASCII representation for the printing characters.

Char.	Code	Char.	Code	Char.	Code
Space	32	@	64	‘	96
!	33	A	65	a	97
"	34	B	66	b	98
#	35	C	67	c	99
$	36	D	68	d	100
%	37	E	69	e	101
&	38	F	70	f	102
’	39	G	71	g	103
(	40	H	72	h	104
)	41	I	73	i	105
*	42	J	74	j	106
+	43	K	75	k	107
,	44	L	76	l	108
-	45	M	77	m	109
.	46	N	78	n	110
/	47	O	79	o	111
0	48	P	80	p	112
1	49	Q	81	q	113
2	50	R	82	r	114
3	51	S	83	s	115
4	52	T	84	t	116
5	53	U	85	u	117
6	54	V	86	v	118
7	55	W	87	w	119
8	56	X	88	x	120
9	57	Y	89	y	121
:	58	Z	90	z	122
;	59	[	91	{	123
<	60	\	92	|	124
=	61	]	93	}	125
>	62	^	94	~	126
?	63	_	95		

TABLE 1−1. Table of ASCII codes (in decimal) for representing textual information.

Numbers between 1 and 31 are reserved for characters associated with control of the printing device. For example 10 (decimal) means move down one line (line feed), and 13 means move all the way to the left (carriage return). For example, the text "EE 121" (without the quotes) would be represented in ASCII in decimal by 69 69 32 49 50 49, or in binary by 01000101 01000101 00100000 00110001 00110010 00110001. Note that a space is just another character.

1.9 Exercises

1. a) Write in decimal the value of the following number written in binary: 10110110.

   b) Write in binary the value of the following number written in decimal: 95.

2. Systems based on any positive integer greater than 1 are possible. Two common systems are *octal*, based on 8, and *hexadecimal*, based on 16. Numbers represented in octal consists of strings of digits between 0 and 7. For hexadecimal, we run out of digits and the letters A, B, C, D, E, and F are used to supplement the ten standard digits. Thus to count to 10 in octal we would write 1, 2, 3, 4, 5, 6, 7, 10, and in hexadecimal 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10.

   a) Write in decimal the value of the following number expressed in octal: 156.

   b) Write in decimal the value of the following number expressed in hexadecimal: 3DA.

3. The principal reason for using octal and hexadecimal number systems (as in the previous problem) is the ease of conversion to and from binary.

   a) Write the two numbers in the previous problem in binary.

   b) Write (110101101)2 expressed in binary in both octal and hexadecimal.

4. Fractional numbers can be expressed in other bases the same way as in base 10. For example, the number $(23.451)_{10}$ means

   $$2 \times 10^{1} + 3 \times 10^{0} + 4 \times 10^{-1} + 5 \times 10^{-2} + 1 \times 10^{-3}$$

   Using this idea, express as a decimal (base 10) integer the following numbers expressed in binary and octal. You can use a calculator to do the arithmetic, but if your calculator has the capability to convert automatically between bases, don't use it. Do that part by hand and show your work. An answer that simply appears will get no credit.

   a) $(110.101)_2 = (\ ?\ )_{10}$

   b) $(42.61)_8 = (\ ?\ )_{10}$

5. Encode your last and first name in ASCII. Separate the two with a comma and a space, and use lower case letters for all but the first letters of each name. Represent each character by a decimal number.

6. A standard 3½" diameter floppy disk can store about one million bytes. *About* how much area does one bit occupy on the surface of the disk?

2. AN INTRODUCTION TO MS-DOS AND THE DOS SHELL

2.1 Operating Systems and MS-DOS

All this brings me to the main point of this chapter. As you may know or have guessed, the CPU and the disk drive do not have the built-in capability to perform the complicated tasks associated with maintaining the file system. The CPU can only do much more basic operations such as reading or writing data to electronic memory, or adding two numbers. Similarly, the disk drive can only respond to basic instructions such as "find the 126,824th byte", or "read 4096 bytes of information starting at this location and write it into RAM starting at address 10624", or the like. In order to implement a file storage scheme, one needs a program for the CPU which takes care of the bookkeeping, and issues the detailed, low-level instructions to the disk drive to read or update the directory file, and then read or write information from or into the file. This is a principal task of a program called an *operating system*. MS-DOS (MS stands for Microsoft, the name of the company that wrote and sells the program, and DOS stands for Disk Operating System) is the program which is most commonly used for this purpose on IBM-like personal computers. Other operating systems you may run into include UNIX (the name came from a rather poor pun on MULTICS which was an early operating system), VMS (Virtual Memory System, it runs only on DEC VAX computers), and the operating system used on the MacIntosh family of computers.

2.2 The Command Interpreter or Shell

The task of an operating system program is to maintain the file system and other system resources. Another program called the *shell* or *command interpreter* is usually the program the user interacts with to run application programs such as spreadsheets or editors. It is common practice to refer carelessly to both the operating system and the shell just as DOS, but there really is a distinction. I'll try to maintain the distinction in this chapter, but later I'll probably succumb to convention. When I turn on one of the PC's in the computer lab, I get a bunch of stuff displayed on the screen which is associated with loading the operating system (DOS) into memory and executing it. Finally, if all goes well, DOS will be up and running, and the system is programmed to automatically load and execute the shell program stored in the file called `COMMAND.COM`. It is this program which writes the prompt, `C:\>`, to the screen. This prompt is the shell program telling me that it is ready to accept commands from the keyboard.

Most commands to the shell are simply the name of a file containing a program which can be executed, but some of the commonly used commands are "built-in" to the DOS shell. Many of the commands can be followed by some information to be passed to the program. Each chunk of additional information to be passed is called an *argument*. Arguments are separated from the command and from each other by spaces. A special type of argument used to turn on specific features is sometimes called a *switch*, and consists of a normal slash, `/`, followed by a single letter. Several examples appear in the list of commands in the next section.

For example, if in response to the prompt I type `DIR` and then press the **Enter** key, the shell program recognizes the sequence of letters `DIR` as referring to a built-in set of instructions or subprogram which reads the disk directory and prints a list of all the files in it along with some other information. The shell program executes this subprogram, causing the desired directory listing to appear on the screen. As another example, if there is a file named `STUFF.TXT` on my disk which contains some text information such as this handout, typing `PRINT STUFF.TXT` followed by pressing the **Enter** key in response to the shell prompt will cause the shell program first to search its list of built-in commands for one called `PRINT`. There is not one, so the program will then search for a file named either `PRINT.COM` or `PRINT.EXE`, and, if it finds one, load the instructions contained in it into memory and execute them. The character string `STUFF.TXT` is an argument, and it will be passed to the `PRINT` program. There is a file named `PRINT.COM` on the disk, and it contains a program which reads a file specified as an argument and prints it on the the system printer. Thus in response to my command, the computer would print the contents of the

file named STUFF.TXT.

The DOS shell pays no attention to what you have typed until you press the **Enter** key. Doing so sends the whole line to the shell program to be interpreted, and acted on if possible. If you make a mistake, you can erase it using the **Backspace** key (before pressing **Enter**!). If the whole line is so hopelessly messed up that it would be better to start over you can erase everything by holding down the Control key (labeled **Ctrl** on the keyboard) and pressing **C**. This combination is referred to as "Control C," and is sometimes written **^C**. DOS is not sensitive to case in processing file names. That means it doesn't matter whether you use capital or lower case letters. The names STUFF.TXT, stuff.txt, and StUfF.TxT all refer to the same file.

2.3 The DOS File System

DOS allows for there to be several disk drives on the system. These drives are specified to the shell by giving each a name consisting of a single letter. On the computers in the EE121 computer lab, there are two floppy drives, and one hard drive. The floppy drives use 3½" and 5¼" floppy disks, and the drives are labeled A and B, respectively. The hard drive is labeled C. A file named STUFF.TXT on the 3½" (A) drive would be specified by A:STUFF.TXT. If you don't specify a drive when naming a file, the shell assumes a default drive, called the *current* drive. The current drive can be changed by simply typing the drive letter followed by a colon at the DOS prompt. On startup the default drive is C, but this could be changed to, say, the A drive by simply typing A: at the prompt. Note that the DOS prompt changes to reflect the default drive letter.

Another feature of DOS is the hierarchical directory structure. I over simplified when I said each disk has a directory listing all the files—often it has several. The main directory is called the *root* directory. It may contain the names of both regular files and directory files. For example, the root directory on my C drive might include a file containing the manuscript of a paper I'm in the middle of writing, another file with accounting information for a research contract, and a directory named EE121 containing stuff pertaining to this class. In the EE121 directory, I might have a file containing the names of all the students in the class, another with everyone's grades, and maybe another with a calendar of events for the class such as test dates and the like. I might also have a directory named HANDOUTS containing files with the text for this and other handouts, and another directory named HOMEWORK containing files with homework assignments, and maybe solutions. Fig. 2−1 shows a schematic representation of the situation. If I were in the C root directory and issued a DIR command, I would see a list of files which would include the EE121 directory file, but would not list any of the files in the EE121 directory, or in the directories contained in the EE121 directory.

Now, if I had a file named DOSINT.TXT stored in the HANDOUTS directory of the EE121 directory of the root or \ directory of the C drive, I could refer to this file as C:\EE121\HANDOUTS\DOSINT.TXT. If the current or default drive were C, I could leave off the initial C:. Typing all this garbage every time I want to access the DOSINT.TXT file could get pretty tiring. Fortunately, just as you can specify a current drive, you can also specify a current directory. The command for doing so is either CD or CHDIR followed by the name of the directory you wish to be the default. If the argument to CD starts with a \, CD assumes the first directory in the argument is located in the root directory, otherwise it assumes the first directory is located in the current directory. For example, if the current directory were C:\EE121, I could change it to the directory containing my DOSINT.TXT file by typing CD HANDOUTS at the DOS prompt. On the other hand, if I were in the \EE121\HOMEWORK directory, typing CD HANDOUTS would result in an error message from a puzzled CD. To get to the HANDOUTS directory I would have to type either the full directory name, \EE121\HANDOUTS, or I could use a shorthand the shell knows about, ..\HANDOUTS. The shell understands two consecutive periods, .., to mean the directory just above the current one. The root directory is accessed by typing just a single backslash, \. Typing CD \ changes me to the root directory, no matter where I am. Note that the shell prompt reflects the current directory as well as the current drive. This feature can be changed, but I'll not go into that here.

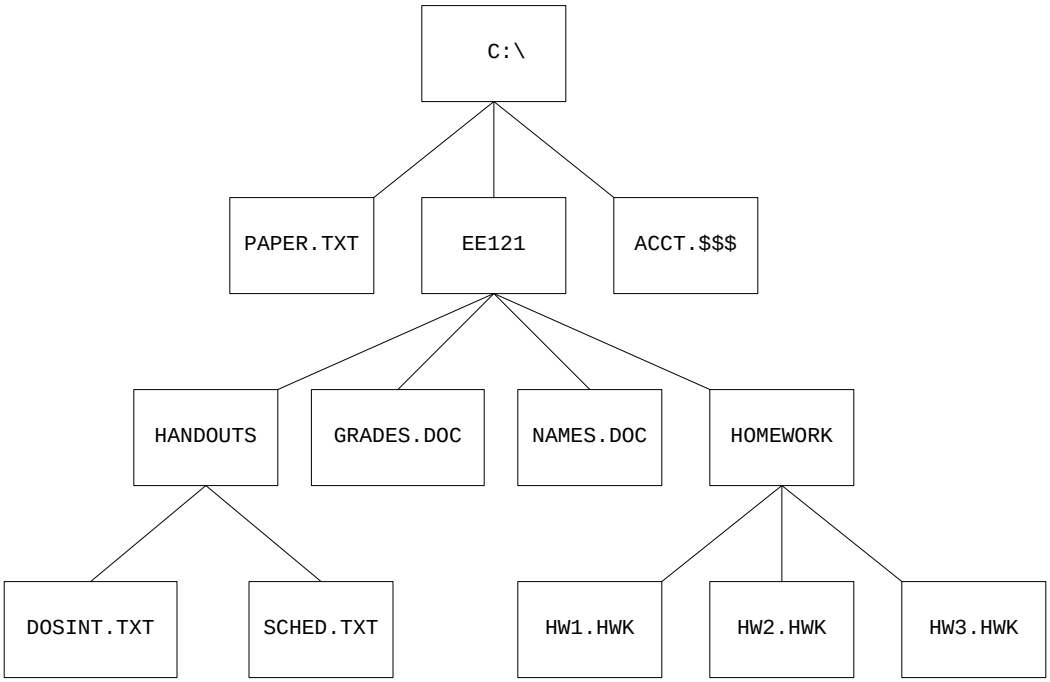

Figure 2−1. Schematic drawing of a possible file system directory structure starting with the root directory at the top, and containing several sub-directories.

On other subjects, you may have noticed that most DOS files have names of the form of some letters followed by a period followed by some more letters. This is the convention, although it is not required. The first set of letters is called the *filename*, and the last set (following the period) the *extension*. DOS allows the filename to be up to eight characters long, and the extension can be up to three. The shell has two *wildcard* characters which are often useful. A `?` character stands for any single character. For example if the current directory had the files `STUFF1.TXT`, `STUFF2.TXT`, and `STUFF26.TXT`, the command `DIR STUFF?.TXT` would list directory information about `STUFF1.TXT` and `STUFF2.TXT`, but not `STUFF26.TXT`. The `*` wildcard character means that any set of characters can occupy the remaining places in the filename or extension. For example, `DIR STUF*.TXT` would list all the `STUFF` files above. Both `DIR STUF*.*` and `DIR STUF*` would also list all the stuff files. `DIR *.*` would list all files in the current directory. Unfortunately, the wildcard feature only works for a few commands.

The shell has another useful feature, called *redirection*. Suppose I want to store the list of files in the current directory in a file called `LIST.TXT`. I could do so by entering `DIR > LIST.TXT`. Note the `>` character. It is a redirection character, and it tells the shell to put what would have normally been printed on the screen into the file following it. Another redirection character is `<`. This character directs the shell to take input for the command not from the keyboard, but rather from a file. For example, the `SORT` command takes lines of text, sorts them into alphabetical order, and prints the result on the screen. If the lines of text to be alphabetized are contained in a file, the input redirection character, `<`, is used to pass the file name to `SORT`. If I wanted the files I just put into `LIST.TXT` to be alphabetized, I could issue the command `SORT < LIST.TXT`. This would result in a list on my screen in alphabetical order. If I wanted to save it in a file called, say, `ALPHA_L.TXT`, I would just append an output redirection,

`SORT < LIST.TXT > ALPHA_L.TXT`.

There's one more redirection character, `|`, to talk about. This character is called a *pipe*. This character should be placed between two commands, the first of which normally produces output on the screen, and the second which uses input redirection to obtain input. For example, we could get a sorted directory listing on the screen directly by entering `DIR | SORT`. The name of the feature is descriptive; it establishes a connection (pipe) between the output of one command program, and the input of a second.

I can't resist an editorial comment here. This redirection feature, like the wildcard feature, is not implemented very well in the DOS shell. With some commands they work, with others they don't. The inspiration for both features was taken from the shell used with the UNIX operating system, which initially (around 1974) ran on computers much smaller and less capable than even an IBM PC. These features have worked fully on the UNIX shell since the beginning, but they still are broken under the DOS shell, 16 years later and about 10 years after the introduction of DOS. It's amazing!

2.4 Using Floppy Disks

To use a 3½" floppy disk make sure the top drive is empty and then insert it into the drive, pushing it in until it clicks into place. When you are done, just remove the disk by pushing the little button, which should cause the disk to pop out. To use a 5¼" floppy, make sure the drive is empty and the lever is rotated so that it is horizontal and then insert it into the drive and rotate the lever until it points down. To remove the floppy, rotate the lever to the horizontal position and remove the floppy. A detail is that a floppy disk must be formatted before you can use it. (You only need to format a given floppy disk once.) To format a disk, insert it in the appropriate drive, and enter either `FORMAT A:` or `FORMAT B:`. The formatting process takes several minutes, so be patient. When the process is finished the program asks you what label you want to put on the disk. You can specify any name you like up to 11 characters long. This information is put in a special place in the root directory of the disk, and it will appear every time you use `DIR` to get a directory listing of the disk. Labels can be useful when you have a number of floppies, to remind you what each contains.

WARNING: Formatting a disk removes all information on it. You can format a floppy disk which has already been formatted once, but doing so will remove all information from it. DON'T EVEN THINK ABOUT FORMATTING THE HARD DISK!

2.5 Exercises

1. When the computer is turned off, all information in RAM, including the programs for the operating system and shell, are lost. It is not a good idea to put either of these into ROM, because information put there cannot ever be changed. If a new, improved version of the operating system (OS) program came out, as frequently happens, a computer with the OS in ROM would become obsolete. Further, different customers just might want to use different operating systems. The obvious solution to this problem is to place the operating system (OS) and shell programs into disk files which are read in when the power is turned on. The problem with this approach is that on power-up, the CPU has no program at all, and doesn't know how to read a file from the disk, or to load the contents of such a file into memory or to transfer control to the program and execute it. Clearly some rudimentary program must be placed in ROM, and the computer must be wired to start up be executing this program. This program should be designed to allow the maximum flexibility regarding the operating system program to be loaded from the disk. Consider how this problem might be solved, and discuss in as much detail as you can your ideas for a good solution. What functions should the program in the ROM perform? How should the operating system program be put on the disk so that the ROM program can find it? Remember that the length of the OS program cannot be known in advance. This problem is called the *bootstrap* problem, I guess because the computer must pull itself up from complete oblivion by its bootstraps.

2. Using one of the computers in the computer lab, list on the display screen the contents of the root directory. Print such a list out on one of the printers. Turn in both the printout and a detailed list of what you did to get it.

3. Put a listing of the contents of the root directory of the `C` drive into a file located on the hard disk. Copy the file to a floppy disk, naming it on the floppy with your first name, followed by a period, followed by `LIS`. (For example if I were doing it, the file would be `FRAZER.LIS`.) Remove the file you just created on the hard disk. Turn in the floppy along with a detailed description of the command you issued to do the job. WARNING: Be careful about using wildcard characters with the `DEL` command. One slip and you can delete more than you intended.

4. A popular editor program is called EMACS. There is a version of this program on the computers in the EE121 lab. Use EMACS to create a file containing your name, and a short paragraph describing your career goals, put the file on a floppy, and turn it in. If you are not familiar with EMACS, there is a pretty good tutorial contained with it. To run EMACS just enter at the DOS prompt `EMACS`. You should get a screen giving you several options, one being the tutorial. After that, everything should be self-explanatory. I have one request, however. In the tutorial you are asked to save the tutorial file after you have modified it. Please don't do that because saving it modifies the *only* copy of the file containing the text of the tutorial, and the next student trying to run the tutorial will get your changes.

   P.S. EMACS is an excellent and very powerful editor, and best of all, it's free! The version on the EE121 computers is one adapted to run under MS-DOS, and it's called FREEMACS. If you have your own IBM PC or clone computer and would like the editor, just copy everything in the directory `\PUB\EMACS` onto a floppy disk. If you don't have a copy of the `UNZIP` utility, copy `UNZIP.EXE` from the `\PUB` directory as well. On your own computer create a directory named `EMACS` in the root directory, insert your floppy into the drive and change default to that drive. Entering `UNZIP` without any arguments will get you instructions on how to transfer the files from the floppy to your hard disk. The only files you need are in `EMACS16A.ZIP`. You might also want the spelling checker in `EMACSPEL.ZIP`.

3. SOME DOS COMMANDS

This section lists a few of the commands available with DOS. The format I will use to describe the commands is the following. The first line of the section describing each command consists of the command name followed by possible arguments. Arguments enclosed in square brackets are optional, arguments without square brackets are not optional, and must be supplied. Something printed in constant-width, block letters, for example `CD` below, should be entered verbatim as printed, whereas something printed in italics (for example *directoryname* below) should be substituted for.

`CD` [*directoryname*]

When issued with an argument this command changes the current directory to that specified as the argument.

When issued with no argument, the command prints the name of the current directory.

The command `CHDIR` is exactly equivalent.

`COPY` [*drive*:] *filename1* [*drive*:] [*filename2*]
`COPY` *filename1* `+` *filename2* [`+ ...`] *filenameN*

The first form of the command copies *filename1* to *filename2*, if supplied. If *filename2* is not supplied, the first file is copied to a file with the same name, but in the current directory on the current drive.

The second form puts all the files separated by a `+` sign together, and writes the result into the file specified by the last argument (which is not preceded by a `+`).

`DEL` [*drive*:] *filename* [`/P`]

This command deletes the named file from the directory. The optional `/P` argument (called a switch) causes the computer to first print the filename on the screen, and then ask whether or not you want it deleted. Answer '`Y`' for yes, '`N`' for no. Wildcards work with this command, but they should be used with care, because it is easy to delete files you want to keep, and it is very difficult to restore them.

The command `ERASE` is completely equivalent to `DEL`.

`DIR` [*drive*:] [*filename*] [`/P`] [`/W`]

When issued without arguments, this command lists the files in the current directory. It gives the size in bytes along with the date and time of last modification of each file. The `/P` and `/W` switches are useful when there are too many files in a directory to fit on one screen of the display. The `/P` switch causes the `DIR` command to print one screen-full at a time, pausing each time for the user to press a key. The `/W` switch causes only the filenames to be printed in up to five columns so that more file names can be put in one screen.

`FORMAT` *drive*:

This command formats a floppy disk in the drive specified by *drive*. You must use this command to format brand new floppy disks before you can use them.

WARNING: Formatting a previously formatted floppy disk destroys any previously existing data on the floppy.

`MD` [*drive*:] *directoryname*

This command creates a directory with the name given in the argument.

The command is completely equivalent to the MKDIR command.

MORE < *filename*
command | MORE

This command is used to print the contents of a file (first form) or the output of a command (second form), one screenfull at a time. After the command has printed a screenfull, pressing any key will produce a new screenfull.

PRINT *filename*

This command prints the contents of the file named in the argument on the system printer.

RD [*drive*:] *directoryname*

This command removes the named directory. It will fail if the directory is not empty.

This command is completely equivalent to RMDIR.

REN [*drive*:] *filename1 filename2*

This command changes the name of *filename1* to *filename2*.

This command is completely equivalent to RENAME.

TYPE [*drive*:] *filename*

This command prints the contents of the named file on the display screen. It does not stop when the screen is full. I use the MORE command more often than TYPE.

3.1 Exercise

1. Assume you are in the root directory, \, of drive C.

    a. Write a single DOS command line to list the files in the directory `\MONGO` of drive B, the floppy drive, sort them alphabetically, and place the result in a file on drive C called `\MONGO.LIS`. (Assume a suitable floppy is already in the drive.)

    b. The `\MONGO` directory on the floppy had a lot of files, and you now want to look at what's there. Write a single DOS command line which will print the contents of `\MONGO.LIS` on the computer screen, one screenfull at a time.

4. GETTING STARTED WITH QUATTRO PRO

4.1 Introduction

The purpose of the following four chapters is to give you the information you will need to use the Borland Quattro Pro spreadsheet program on the computers in the EE121 computer lab, and to give you several examples of how a spreadsheet program can be used in engineering. The information is specifically for Quattro Pro, version 2. Borland distributes an earlier spreadsheet program called just Quattro which is still available. The two programs are similar, but the new program has substantially more capability, and the user interface is different.

Spreadsheet programs were developed primarily for use in business, but they prove to be quite useful in engineering and science applications as well. They can be viewed as a programming language (albeit a non-conventional one!) which is easy to learn. The spreadsheet format is analogous to that of a business ledger. Entries are arranged in columns and rows, with each one being labeled by a column letter and a row number. It's a little like street intersections in parts of Lincoln. East-West running streets are named with a letter, and North-South streets with a number. The street intersections would be analogous to spreadsheet *cells*, which are the spaces in which you put numbers, formulas, or some text. The East-West streets would correspond to the columns of the spreadsheet, and the North-South to the rows. We could label the intersections by using the intersecting street letter and number. For example the intersection of C and 10th would be called C10. With this scheme, it's easy to specify any cell in the spreadsheet.

There's one more small detail. There are more than 26 columns in most spreadsheets. The 27th through the 52nd columns are labeled `AA` through `AZ`. The next 26 columns are `BA` through `BZ`, and so forth.

4.2 Executing the Quattro Pro Program

At this point, it would be a good idea to sit down at a computer with Quattro Pro installed so you can see for yourself what I am discussing. If you are using one of the computers in the EE121 computer lab, first insert a floppy disk in the floppy drive and change the default drive to the floppy by entering `A:`. To start up Quattro Pro simply enter `Q`. After a few seconds you should get a display with the Quattro Pro name in it, and a few seconds later that should disappear and the worksheet should appear on the screen. If instead you get an error message `Bad command or file name`, something is messed up on the machine. Please report the problem, and then try another machine.

4.3 How to Get Out

We'll talk about this more later, but in case you are claustrophobic, or need to stop before we get to it, here's how you get out of Quattro Pro. I think it's fool proof. First press the escape key, labeled **Esc** and located at the extreme upper left-hand corner of the keyboard several times until nothing changes when you press it. This should get the program in its initial state. Then simultaneously press the control key, labeled **Ctrl** and located at the bottom of the keyboard under either **Shift** key, and **x**. (That combination is called `control-x`.) If you have not entered any data the program will exit right away and you will get the DOS prompt. Otherwise, a little window will appear with the question "`Lose your changes and Exit?`" at the top. Type `Y`, and you should soon be back in Kansas. Try exiting now, and when you are back in DOS, type `Q` again to restart Quattro Pro so we can get on with it.

4.4 What It All Means

If all is well, you should have a rather colorful display on the screen. At the very top should be a line of text starting `File Edit` which lists groups of commands you can give the spreadsheet program. This is called the *Pull-Down Menu Bar*, and we'll talk about it in the next section. If the top line starts

`Worksheet Range`, or if it contains only `A1:`, then someone has messed up the defaults. If so, it's easy to fix, but let me first tell you what's going on.

Quattro Pro has three user interfaces which you can select. The first is the interface from the earlier Quattro program and Quattro Pro labels it `Q1`. The second is the interface designed for Quattro Pro, labeled simply `Quattro`. The third is very similar to that of a much older and more common spreadsheet program called Lotus 123, and is labeled `123`. For this class I have decided to standardize on the Quattro Pro interface because that is the interface assumed in the optional reference book for this class and because the descriptions in the Quattro Pro manual are more complete for this interface. If Quattro Pro starts up with `Worksheet Range` in the top line, the default interface choice has somehow been changed to the 123 interface. To change it back, just follow the following recipe. First press the escape key, **Esc**, several times until nothing changes (as in the above discussion about getting out of Quattro Pro), then press the following keys in order **/WGDFM**. Doing so should produce a small window with the labels of the three available interfaces, `Q1`, `Quattro`, and `123`. We want the Quattro interface, so use the down and/or up arrow keys, ↓ and ⇑, to move the highlight to the `Quattro` item, press **Enter**, and you should be all set. (The arrow keys are located at the bottom of the keyboard just to the right of the normal typewriter portion.) If the top line contains only `A1:` or something similar, do the same thing, except press **/DSM** instead of **/WGDFM** to get the little window.

Just below the pull-down menu bar is another line, called the *Input Line*, which contains `A1:` at the far left. The rest of the line is blank. The `A1` tells which cell is selected; it's the cell in column `A`, and row `1`. Just below this is a line containing the letters `A`, `B`, ..., `H`. These letters label the first eight columns of the worksheet. Note that the `A` column is highlighted. Starting just below this line, and at the far left hand side of the display is a column of numbers starting with `1`, and running through `20`. These numbers label the first twenty rows of the worksheet. Note that the `1` row is highlighted. Only a small fraction of the worksheet is shown at once. The total worksheet contains 8192 rows and 256 columns (`A-IV`).

At the very bottom of the display is the *Status Line*. This line tells you stuff you might need to know about what the program is doing. The arcane symbols at the far right of the display are for using a mouse to control the spreadsheet program. They allow you to do with the mouse much of what you can do with the keyboard. I won't be discussing mouse use, but if you really want to use the mouse, with a little experimentation, you should be able to figure out how to do it.

4.5 Keys for Klutzes

Three keys I find very useful are the *escape*, *Undo*, and *Help* keys. The escape key is labeled **Esc** on the keyboard, and is located at the upper left-hand corner. If you start doing something and then decide it wasn't such a good idea, pressing **Esc** usually puts you back to where you were just previously. The Undo key is not labeled, but is activated by holding down the **Alt** key (located on either side of the space bar) and pressing **F5**. Use this one after you have just made a change to the worksheet that was a mistake. Usually, it will restore the worksheet to what it was just before the last operation. The Help key provides information about using Quattro Pro. It is not labeled, and is activated by simply pressing the **F1** key. I suggest pressing **F1** now and browsing around a little to see what kind of information is available. Use the escape key to get out of Help once you are done. You may need to press it several times.

4.6 The Pull Down Menus

You control much of the operation of Quattro Pro by using Pull Down Menus. The names of the menus appear at the top of the screen. To access a particular menu, first activate the Menu Bar by pressing the slash key, `/`. (Note there is also a backslash key, `\`, which is different.) When you press `/` the first name in the menu bar (`File`) is highlighted and the blinking underline cursor appears under it. To select a specific menu you have two choices.

1. Use the left and right arrow keys (located at the bottom of the keyboard, between the normal typewriter part and the numeric keypad part) to move the highlight to the desired item. When you have done so, just press **Enter**.
2. Just type the highlighted letter of the name of the desired menu. Don't press **Enter** afterward.

The `File` menu contains items that let you save and retrieve worksheets to and from files, and perform other operations related to the file system. To see what you can do, and how, access the menu by either of the methods above. Covering part of the worksheet is a *menu* which lists the things you can do. Note that the top item on the menu is highlighted. To tell the program to carry out any of the operations on the menu, you have two choices, similar to those you had in choosing a menu from the Menu Bar.

1. Use the up and down arrow keys to move the highlight to the desired item, and press **Enter**.
2. Just type the highlighted letter of the name. This letter is often *but not always* the first letter. Do not then press **Enter**.

The list below describes some of the items in the menu.

`Retrieve`

This function allows you to retrieve a worksheet you created previously and saved in a file. Doing so destroys anything you have already done in the current worksheet (unless you have saved it in a file, of course). For your convenience, the names of all the likely candidates (those with extensions starting with `W`) in the current directory are listed. You can either type the name of the file you want, or you can use the arrow keys to move the highlight over the right name. In either case, press **Enter** when you are done.

`Save`
`Save As`

These functions are used to save a worksheet in a file. If you want to resave a worksheet which you previously retrieved from a file use the `Save` command. If you want to save a new worksheet, or if you want to save under a different name a worksheet you previously retrieved use the `Save As` command. If you use the `Save As` command a window will appear asking the name of the file in which you want the worksheet saved. If there is a current worksheet name (usually the last worksheet you retrieved) Quattro Pro will guess this name and put it in the window. If that's not what you wanted, just start typing the name you want, and the guessed name will disappear. You can at any time just modify, rather than completely retype, the name using the `Edit` key to be discussed a little later in Section 9 of this chapter. Press **Enter** when you are done. Saving a worksheet into an existing file causes whatever information which is contained therein to be destroyed. For this reason, if you are telling it to save to an existing file, Quattro Pro pops up a little window asking if you are sure you know what you are doing. The top line says `File already exists:`, and the remaining three offer you options. The first, `Cancel`, tells Quattro Pro to just forget the whole thing; `Replace` tells it to go ahead and overwrite the existing file; and the third, `Backup`, tells it to first rename the existing file to one with the same filename, but with the extension `.BAK`, and then to save the worksheet into the specified file.

`Directory`

This option sets the directory in which Quattro Pro looks for files to retrieve, and in which saved files are put. The current default directory is displayed in the menu to the right of the `Directory` listing in blue (if someone hasn't changed the display colors!). The default directory starts out as whatever directory you were in when you started Quattro Pro. I suggest you always keep the default directory to be one on your floppy disk so that you are guaranteed that any files you create are on the floppy, where you can save them, rather than on the hard disk, where they will probably disappear.

`Exit`

This is the same function you can invoke by pressing **Ctrl** and **x** at the same time as discussed above. If you have nothing to save, choosing this item causes execution of the Quattro Pro program to cease, and control is returned to the DOS shell. If some possibly useful information would be lost, a small window pops up to tell you. The window has `Lose your changes and exit?` on the top line, and offers three choices, `No`, `Yes`, and `Save and Exit`. The meaning of these three choices should be self-explanatory.

4.7 Moving Around the Spreadsheet

As indicated both by the input line, and by the the column and row highlighting, the selected cell when you first startup Quattro Pro is `A1`. You can change the selected cell in several ways, but the most common is to use the arrow keys located at the bottom of the keyboard, and between the normal keys (as on a typewriter) and the numeric keypad keys (as on a simple calculator or adding machine). Pressing any arrow moves the selected cell in the corresponding direction on the worksheet. For example, pressing the down arrow, ↓, three times changes the selected cell to `A4`, and then pressing the right arrow, →, twice changes it to `C4`. Try changing the selected cell using these keys, and note the changes in the display (both in the Input Line and in the highlighting) which indicate which cell is selected. Try moving to a cell which is off the screen.

There are other ways of moving around the worksheet. The keys labeled **Page Down** and **Page Up** (they're in the group just above the arrow keys) move down and up by 20 rows at once. If cell `A1` is in the upper left corner of the worksheet, for example, pressing **Page Down** will move to put the 21st row at the top. You can get the same effect in the horizontal direction in two ways. One is to hold down the control key (labeled **Ctrl**) and press either the left or right arrow keys. I'll use the caret, ^ to indicate that the control key should be held down. Control right arrow, for example, would be indicated as ^→. The **Tab** key can be used to provide the same effect. Pressing just **Tab** is the same as ^→, and holding down the **Shift** key and pressing **Tab**, **Shift-Tab** in my jargon, is the same as ^←.

For really big distances, pressing the **Home** key (just to the left of the **Page Up** key) moves you to the first cell, `A1`. The **End** key used with the arrow keys moves you to the other ends of the worksheet. For example, pressing **End** and then → moves to the right end of the worksheet. Because you seldom use the bottom rows or the rightmost columns of the worksheet, the term "end" is defined not as the farthest you can go, but rather as follows. If the current cell is not blank, then the end is the next non-blank cell which is followed by a blank cell. This would often correspond to the end of the part of the worksheet you are using. If the current cell is blank, on the other hand, the end is the position of the next non-blank cell. The combination **End** followed by **Home** moves you to the bottom right end of the non-blank area of the worksheet.

Finally, you can specify a specific cell to go to using the *GoTo* key. This is the key labeled **F5** located in the long row of keys at the top of the keyboard. After pressing **F5**, `[Enter][Esc] Enter address to go to:` appears in the Input Line near the top of the screen. Following this is the address of the currently selected cell. The program is expecting you to enter the address of the cell you wish to go to (`J57`, for example). Anything you type will appear after the colon. If you type a valid cell address and then press **Enter**, that cell will become the selected cell, and the region around that cell will be displayed. The words in the square brackets indicate how the program expects you to terminate this command. Pressing the **Enter** key tells the program to do it, and pressing the *escape* key, as usual, tells the program to forget it. All of these motion commands are summarized in Table 4−1.

4.8 Entering Information in Cells

A cell can contain three kinds of information: numbers, text, and formulas. The names of the three types are descriptive. Examples of the three are `12.567`, `"This is fascinating"`, and `12.567+B6`. Formulas and numbers are similar, and in this discussion we will not differentiate between

KEYS FOR MOVING	
Key	**Action**
←	Move left one cell
→	Move right one cell
⇑	Move up one cell
↓	Move down one cell
^→ or **Tab**	Move right one screenfull
^← or **Shift Tab**	Move left one screenfull
Page Up	Move up one screenfull
Page Down	Move down one screenfull
Home	Move to first cell, `A1`
End Home	Move to lower right end. See text for definition of "end"
End ⇑	Move to upper end
End ↓	Move to lower end
End →	Move to right end
End ←	Move to left end
F5	Move to cell you specify by address

TABLE 4−1. Keys for moving around in the worksheet

them. A text entry is often called a *label*. Information is entered into a cell by simply moving to that cell, typing it in, and pressing **Enter** when finished. When you put some information into a cell, Quattro Pro has to know which of the three types it is. If what you enter starts with any letter or punctuation mark except `/ + − $ ( @ #` or `.` then the program assumes that the information is text. A number, on the other hand, is an entry that contains mostly digits, although a number can start with a `+` or a `−` or a `$`, and it can end with a `%`. A number can contain a single decimal point, and you can use a form of scientific notation. For example, the number 5.234 × 10-12 could be entered as `5.234E-12`.

These default rules for determining information type work well for many cases, but sometimes you really want to start a bit of text with a number or one of the other forbidden symbols. For example, you might want to label a column by putting some text at the top of it that says `4th Try`, or something like that. To force the program to consider an item to be text, start it with one of the following three characters `'`, `"`, or `^`. When placed at the beginning of a text entry, these characters are not printed. They force the item to be considered as text, and control the placement of the item in the cell, causing left justification,

right justification, and centering respectively. If there is not one of these characters at the start of an entry which the program recognizes as a text item, a `'` is automatically tacked on for you, so left justification is the default.

At the other end of the problem, you may also want to enter a formula into a cell that starts with the address of some other cell, such as `C7-B6+3.2`. If you typed this as written, the program would think it was a text item, not a formula, because it starts with a letter. To force the program to interpret is as a number or formula, start it with a character which is in the number group, but which doesn't alter the value. The most common character to use for this purpose is the plus sign, `+`, in which case in the example above we would enter `+C7-B6+3.2`.

The columns of the worksheet each have a set width, nine characters by default. This width can be changed using the `Style` Pull Down Menu, but I'll delay discussing that. If you enter a character string (label in Quattro Pro jargon) into a cell, and the cell width is smaller than the length of the string, no information will be lost, and the entire string will appear on the input line of the display whenever you move the highlight over this cell. The character string will spill over into empty cells to the right if needed until it encounters a cell which is not empty. It will not spill over into cells which contain some data. This feature can be very useful for entering titles and the like for spreadsheets. As we will discuss in a later section, it is also convenient for displaying macros. The feature can lead to some confusion since it appears on the worksheet as though the long character string occupies several cells, not just the one cell at the head.

4.9 Correcting Mistakes

If you make a mistake in entering some information to be put into a cell and notice it *before* pressing the **Enter** key, you can correct it by using the **Backspace** key to erase characters until you get to the problem point, and then retype the remaining stuff. If the entry is hopelessly messed up, you can use **Esc** to start over. Quattro Pro also has an editing facility that allows you to delete and insert specific characters anywhere in the entry. You can use it to edit information as you are entering it, or you can edit information already placed in a cell. To edit the contents of a specific cell, move to that cell and press **F2**. If you want to edit what you are in the process of entering, just press **F2**. This will put you in edit mode, and the cursor (a red rectangle with a blinking underline character in it) will appear at the end of the Input Line. You can move this cursor around using the right and left arrow keys.

If you are in Insert mode (you will be unless you purposely change it), characters you type will be inserted just before the cursor. The other mode is Overwrite mode, and in this case the character under the cursor will be replaced with the character you type. You change between the two modes using the **Insert** key, located just to the left of the **Home** key. When you are in Overwrite mode, `OVR` appears in the Status Line at the bottom of the screen. In either mode, if you press **Backspace** the character just before the cursor will be erased, and if you press **Delete**, the character under the cursor will be erased. When you are satisfied with your changes, press **Enter** and the modified contents of the Input Line will be put in the selected cell. Pressing **Esc** once erases everything on the Input Line, but leaves you in edit mode; pressing it a second time exits edit mode without modifying the contents of the selected cell. This information is summarized in Table 4−2, and the function of some other edit-mode keys is discussed.

If the contents of a cell are hopelessly messed up, you can erase them completely by using the `Erase Block` item in the `Edit` menu. To use it, first move the highlight to the cell to be erased, then activate the `Edit` menu by typing `/E`. Choose the `Erase Block` item, and the menu will disappear, and `Block to be modified:` will appear in the Input Line at the top of the screen. Just press **Enter** to erase the contents of the highlighted cell. You can also erase a block of cells with this command. To do so, after choosing the `Erase` item, you must tell the computer what range of cells you want to erase. You can do so by typing the address of the upper left-hand corner of the block to be erased, followed by one or two periods, followed by the address of the cell in the lower right-hand corner of the block. Another way to do it is by pointing. If one corner of the block to be erased is highlighted, use the arrow keys to move the highlight to the opposite corner. Notice how the specified range is painted. In either case, press **Enter**

KEYS FOR EDITING	
Key	**Action**
Esc	Press once to erase the Input Line. Press twice to forget the whole thing and leave the cell contents unchanged.
Enter	Puts Input Line into selected cell, and exits edit mode.
←	Move cursor one space left.
→	Move cursor one space right.
Backspace	Delete character to left of cursor.
Ins	Toggles between *Insert* and *Overwrite* modes. Insert mode puts characters you type just to the left of the cursor, Overwrite mode replaces the character under the cursor with the one you type.
Delete	Delete character under cursor.
^\	Delete all characters from cursor to end of line.
^**Backspace**	Delete entire line.
Tab or ^→	Move cursor 5 spaces right.
Shift-Tab or ^←	Move cursor 5 spaces left.
Home	Move cursor to start of line.
End	Move cursor to end of line.

TABLE 4−2. Keys for use in edit mode.

when you are finished.

Finally, if the whole worksheet is shot, or perhaps if you are done with it and want to start another, you can erase the whole thing using the `Erase` item in the `File` menu.

4.10 An Example, Part 1

In this section I'd like to consider an example in which Quattro Pro could be used usefully. I'll continue the example through the next two chapters, using it as a vehicle to point out selected capabilities of Quattro Pro along the way. More importantly, the example illustrates a typical problem solving process. In engineering practice it is unusual that a problem appears along with a statement of how it is to be solved. Rather, the engineer is presented with a problem and an array of tools and information, some of which may be useful in solving it and some which are not. The art of engineering consists of taking these tools and applying them to solve the problem successfully. It is important that during your study of engineering you learn about and gain skill in the use of these tools, but it is more important that you develop the ability to take the tools and fashion solutions using them. Besides, just applying the same old tools the same old way

time after time gets pretty dull pretty fast. Its much more satisfying to dream up some new way to use the tools to solve some problem, especially when it works.

Anyway, here is the problem. Herbie Husker is taking EE231 lab, and his assignment is to characterize electrically a box given (well, actually lent—he has to return it at the end of the lab period) to him. The box has two terminals, and Herbie has managed to translate the lab assignment sheet into terms he thinks he understands. The sheet claims that the voltage across the terminals and the current into them should be linearly related. Herbie has translated this to mean that if V is the voltage and I the current, then they are related by a formula like

$$V = A + BI \tag{4−1}$$

where A and B are constants to be determined. It is Herbie's assignment to find the values of the two constants for his box. (In EE lingo, the constant B is called the resistance, and the symbol R is usually used for it. Herbie isn't an EE, though, so he uses B instead.)

Herbie has available to him a power supply which can deliver any voltage between 0 and 40 volts, and two meters, one of which measures voltage, and the other current. He decides to connect the terminals up to the power supply, along with the two meters, and to measure the voltage and current for several different voltages from the power supply. Herbie does so, and writes down his results in a table, which is reproduced below in Table 4−3. The equipment has not had an easy life, however, and the needles on the meters bounced around a bit. Further, Herbie has never had good eyesight, and he caught a football with his head a few years ago and ever since he sometimes sees double. Anyway, there is some noise on the data, and there may be a few readings which are just wrong.

Voltage	**Current**
2.36	−0.00029
36.8	0.0394
9.41	0.00749
15.1	0.0150
22.1	0.0106
32.4	0.0353
4.83	0.00359
21.5	0.0251
29.5	0.0323
17.7	0.0205
25.9	0.0270

TABLE 4−3. Herbie's current-voltage data.

We will analyze Herbie's results for him using a worksheet. First erase anything you may have put in the worksheet. It's usually a good idea to put titles on worksheets so that you have some chance of figuring what its about when you come back to it after several months. I frequently don't do that, and I often wish I had. I'm supposed to be encouraging good habits here, so I'll try to put a title on everything. Move to `A1` (I suggest using **Home**), and put in some informative text. I put in "Chapter 4: Herbie's voltage and current measurements." We will use the `A` column to store Herbie's voltage data, and the `B` column for the current data. Label these columns by putting `Voltage` in cell `A5` and `Current` in cell `B5`. Include the appropriate character at the start of each to cause it to be centered. Next enter Herbie's data. Just so we'll be synchronized, start in row `6` so that the first voltage, 2.36, is in cell `A6`. You can use a shortcut here. Once you have entered a number into the Input Line you could put it in the current cell by pressing **Enter**, and then you would have to press one of the arrow keys to move to the next cell. Instead, after typing the number, you can just press the appropriate arrow key. Doing so will enter the information into the current cell and then move to the next one.

The next thing we want to do is to make a graph of Herbie's data to see if it does, in fact, follow the claimed linear dependence in Eq. (4−1). (Remember, if we plot V vs. I from Eq. (4−1) we will get a straight line with slope B, and V-axis-intercept A.) That's easy to do with Quattro Pro, but we haven't talked about it yet, so save what you've done in a file so that you can play around without destroying it.

4.11 Exercises

For all problems save your finished worksheet into a file on a floppy disk, and turn the floppy in as part of the homework assignment. You should also include a clear written description of what you did. That description can either be on a separate piece of paper, or it can be as a comment in the worksheet itself. You can put more than one worksheet onto one floppy, but make sure it's obvious what file belongs to what problem.

1. Enter Herbie's data into a worksheet as in the notes. I suggest you save the worksheet twice, onto different floppies, so that you will have one to turn in and one to keep for use with the following chapters of these notes.
2. A common annoyance in engineering is the need to convert from one system of units to another. Make a worksheet that will convert from:
    1. °F to °C,
    2. inches to centimeters
    3. pounds to kilograms
    4. fluid ounces to milliliters

    Put each type of conversion in a separate row. In column A put a short text description of the conversion performed in that row (F - C for example). Use the cells in column B for entering the value to be converted, and put formulae in column C which take the value in the adjacent column B cell and perform the specified conversion. In your finished worksheet, I should be able to put a number representing a temperature in °F, for example, in the appropriate cell in column B and read off the converted result in column C.

5. MAKING GRAPHS WITH QUATTRO PRO

5.1 Introduction

Quattro Pro allows you to make all sorts of graphs. Use the `Graph` Pull Down Menu to access this capability. This menu contains options which allow you to create and annotate graphs. There are far too many possibilities to list them all here, but once you get the idea, most things are pretty self-evident. You may have noticed already that when a menu item is highlighted a short description of the item appears in the Status Line at the bottom of the screen. This feature can be helpful when browsing around to see what can be done, or when trying to figure out what !#$%!! menu contains some function you know exists, but can't remember where.

5.2 Specifying Graph Type

Back to the `Graph` menu, get this menu displayed on the screen if it isn't already. (I suggest pressing **Esc** enough times to back you out of any menus you may be in, so the cursor is back in the worksheet, and then typing `/G`.) The first item on the menu, `Graph Type`, allows you to specify the type of graph you wish to make. Selecting this item results in a sub-menu listing the types of graphs which are possible. (Remember you can select a menu item either by using the up and down arrow keys, ⇑ and ↓, to move the highlight to the desired item and then pressing **Enter**, or by typing the highlighted letter in the item name.) Most of the graph types are of little interest to engineers, but you should find `XY` and, to a lesser extent, `Line` graphs quite useful. You may also find `Bar` and `Pie` graphs useful if you have to communicate with the "front office." In our example, we want to plot Herbie's measured values of current vs. his voltage values. For this purpose, choose an `XY` plot. After doing so, note that `XY` appears in the `Graph` menu to the right of the `Graph Type` item.

5.3 Specifying What to Graph: Ranges

Our next task is to tell the computer what to graph. From the form of Eq. (4−1), we want to plot voltage on the y-axis vs. current on the x-axis, so we need to tell the computer to use the values in the voltage column, `A`, for the y-coordinates, and the values in the current column, `B`, for the x. The menu item for specifying the block of worksheet cells containing the data to plot on each axis has the unlikely name `Series`. Select it. You should now have a sub-menu with `1st Series` the top item, `2nd Series` the next, and so forth through `6th Series`. Below these are `X-Axis Series`, `Group`, and `Quit`. Quattro Pro allows you to plot up to six different sets of data on the same graph, and the first six items on this sub-menu are for specifying what data to plot for each. We only have one set of data to plot, so choose the top item, `1st Series`. Choose it and the menu disappears, and it looks like you are back in the worksheet, except the the Input Line says `Enter 1st Series Block:`, and the word `POINT` appears at the far right end of the Status Line. This is the same situation you get in with the `Erase Block` item of the `Edit` menu. The computer is asking you to specify a range of values. With the `Erase Block` item, it was the range of cells to be erased, here it is the range of cells to be associated with the y-axis coordinate of the first plot. You do so in the same way, either by typing the range or by pointing it out to the computer. To type in the range, type first the address of the cell at the top of the column, `A6` in our case, followed by either one or two periods, followed by the address of the cell at the bottom, `A16` in our example. To point out the range, move the highlight to the first cell in the range, press the period key to "anchor" the operation, then move it to the last cell, noting how the specified range gets painted. In either case, press **Enter** when you are finished. Doing so puts you back in the `Series` sub-menu.

Next we need to tell the computer what to use for the x-coordinates. The item labeled `X-Axis Series` is the one to use for this. Choose it and the same thing will happen as when you chose `1st Series` except that the Input Line will say `Enter X-axis labels block:`. This prompt is a bit misleading in our case, but the computer is asking for the range of data to be used for the x-coordinates.

Select Herbie's current values (in cells B6 through B16).

5.4 Some Other Graph Options

We are now done specifying the data to be plotted. We can also specify a lot of things about how the graph looks. For example, we can tell the computer to connect the points we've just given it with a continuous line, or we can tell it to just put some symbol like a filled-in square at each point, or we can tell it to do both. Herbie can only guess at what values of current he would have gotten for voltages he didn't use, so let's just have the computer put an x at each of his measured points. Get back to the Graph menu by either choosing the Quit item from the Series sub-menu, or pressing **Esc**. Choose the Customize Series item from the Graph menu. Doing so will produce a sub-menu with several items on it. The one we want is Markers & Lines; choose it. That will produce a sub-menu with, in order, Line Styles, Markers, Formats, and Quit. The third item allows you to specify for each possible plot on the graph whether you want only lines drawn between points, only a symbol plotted at each, or both. The first item is used for plots with lines, and it lets you choose what kind of line (solid, dotted, dashed, etc.) to use. The second is used for plots with symbols, and it lets you choose what symbol to use.

We first have to tell the program to just plot a symbol at each data point, so choose Formats. That should get you a sub-menu listing the 6 series plus an item labeled Graph. This latter item lets you specify lines only, symbols only, or both for all plots at once. We have only one plot so choose the 1st Series item. Doing so produces a small sub-menu with the three expected choices plus one which specifies that nothing be drawn. Choose Symbols. That gets you back to the previous sub-menu. We have only one series to specify, so choose Quit to return to the sub-menu previous to this one. Next we want to tell the computer what symbol to put at each point. The Marker item is the one that lets you do that, so choose it. That produces a sub-menu with items labeled 1st Series through 6th Series. These items let you choose different symbols for each of your plots. We want to specify the symbol for the first plot, so choose 1st. Yet another menu should appear listing all the symbols you can choose from. Choose the X symbol (with label E). That backs you out one sub-menu. We are finished specifying symbols, so choose Quit to back out one sub-menu further. We are also done here, so choose Quit again.

5.5 Displaying the Graph

This should have backed you out to the sub-menu with Colors at the top. This menu lets you do all sorts of things to jazz up the appearance of the graph, but before getting artistic, lets see what we have wrought so far. You can see the graph by pressing the **F10** key. Do so. You should see a fairly attractive-looking graph with yellow x's on a graph with light blue dotted grid lines, and darker blue axis labels. If not, someone has messed up the default settings. To erase the graph from the screen and return to wherever you were previously, just press any key on the keyboard. If you don't agree with the artistic merit of the graph you produced, you can change the colors of the x's from the Colors item in the Customize Series sub-menu that was on the screen just before you pressed **F10**. To change the grid style and color go back to the main, Graph, menu, choose the Overall item, and from the resulting sub-menu choose the Grid option. I suggest you play around with some of the colors and other options to see what you can do. You might want to save the worksheet to a file first, though, just in case.

5.6 Putting labels on the axes and a title on the graph.

Let's put labels on the axes. Choose the Text item from the Graph menu. From the resulting sub-menu choose X-Title, and type in the x-axis label, Current, followed by **Enter**. Next choose Y-Title, and enter Voltage. Let's also put a title on the graph, so after entering the y-axis label, choose the 1st Line item. (You can have two-line titles—that's what the 2nd Line item is for.) Enter Herbie's Data as a title.

Finally, press the **F10** key again to see the final product. It should look something like Fig. 5−1. We'll be using this graph later, so if it looks satisfactory, save the worksheet in a file so you can retrieve it later. I'm not going to try to improve on the looks of the graph further, but I suggest you spend a little time

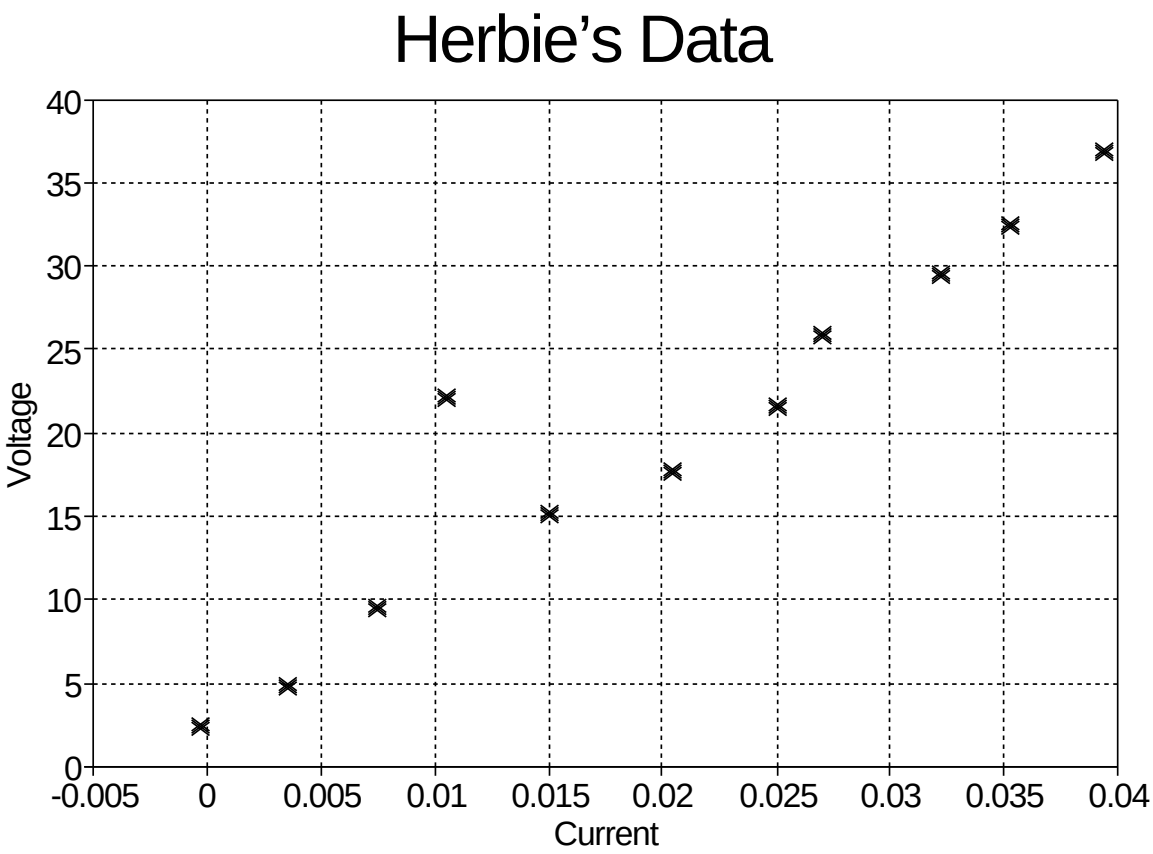

Figure 5−1. Graph of Herbie's data

exploring what you can do.

5.7 Exercises

For all problems requiring the use of Quattro Pro, when you are satisfied with your worksheet save it to a file on a floppy and turn the floppy in as part of the homework assignment. You should also include a clear written description of what you did and why. That can be handed in on a separate piece of paper, or it can be included in the worksheet as a comment. You can put more than one worksheet file on a floppy, but make sure you give each a name that makes it obvious which file goes with which problem.

1. Plot Herbie's data as described in the notes. I suggest you save the worksheet twice, to two separate floppies. That way you will have one floppy to turn in, and one to continue with the discussion in the notes.

2. In Table 5−1 the high and low temperatures in Hooterville are listed for the first ten days of October. Use Quattro Pro to plot these temperatures, using different symbols in different colors for the high and low temperatures. Label the axes appropriately, and put an appropriate title on the graph.

HIGH AND LOW TEMPERATURES

Day	High	Low
1	72	40
2	75	43
3	65	33
4	70	38
5	71	42
6	74	43
7	71	40
8	68	35
9	74	38
10	70	37

TABLE 5−1. Table of high and low temperatures in Hooterville during the first ten days of October.

6. DOING THE ANALYSIS WITH QUATTRO PRO

6.1 Recap

Let's get back to our example: Herbie's EE231 lab assignment. Recall he was handed a box with two terminals, and an assignment sheet which (after some deciphering) said that the voltage across the two terminals and the current through them are supposed to be related linearly, that is according to an equation of the form

$$V = A + BI \tag{4−1}$$

When we left our hero, he had successfully (more or less) measured values of voltage and current for several different voltages, and written the results in a table, Table 4−3, which you should have put into a worksheet, with the voltage values in column A and the current in column B, both starting in the third row. In the last chapter you had added a kinda-nifty-looking graph of Herbie's values which is shown in Fig. 5−1. If needed, please load that worksheet now, so that we can continue the discussion.

If Herbie's measurements do in fact obey Eq. (4−1), then the points should all lie on a straight line with slope of B, and V-axis intercept A. Press **F10** to get the graph on the screen. The x's do in fact seem to lie close to a straight line, except for one, the one at about 22 volts and .011 amperes. Herbie was worried that he might have misread the meter a few times, so this one could be a mistake. In fact, if the voltage had really been about 12 volts instead of 22, the point would fall pretty nicely into line. We'll leave the point in the worksheet for now, but regard it with suspicion.

Note how much easier it is to spot the non-conformist from the graph than from the table. Graphs can be very useful for detecting patterns in data.

6.2 Determining Values for the Constants A and B

We want to find values of the constants A and B in Eq. (4−1) which make the equation agree most closely with Herbie's data (except maybe for the suspicious datum). There are better ways of doing it (we'll discuss a few later), but as a first try, let's guess values of the constants, and get Quattro Pro to graph the resulting values of V and I from Eq. (4−1). If our guess is good, the plot should overlay Herbie's data points closely, if it's bad, it won't. We'll get the best values of A and B by trying different values until the agreement between Eq. (4−1) and Herbie's points appears to be the best we can get. Remember there is some random (we hope) error in Herbie's measurements, so no points are likely to agree exactly with the prediction of Eq. (4−1). Besides, maybe Eq. (4−1) does not exactly describe the box, but is only a good approximation.

Press any key to return to the worksheet. I will put the guessed values for A and B in two cells of the worksheet somewhere, and add a new column containing the prediction of Eq. (4−1) for the voltage. Each cell in the new column will contain a formula which will take Herbie's measured value for the current and evaluate Eq. (4−1) for the predicted voltage. Where should we put the new stuff? We already have Herbie's experimental data in columns A and B, so column C seems like a good place to put the new "theoretical" column. Where should we put the guessed values of A and B? We could either put them off to the side, say somewhere in column D or E, or we could put them up at the top of the worksheet. Putting them off the the side is a little easier, but later I'm going to want to put a graph there, and putting them at the top gives me an opportunity to show you some other useful features of Quattro Pro. Therefore, I choose to put A and B up at the top. Adding some appropriate labels, the worksheet will then look something like Fig. 6−1.

The problem is that I want to put some labels and the values of A and B in A4..B6, but I already have Herbie's data there. (If I could only learn to think ahead!) I need to move Herbie's data and the associated column labels down a few rows, so there is room. I could do the move by hand, but that would be

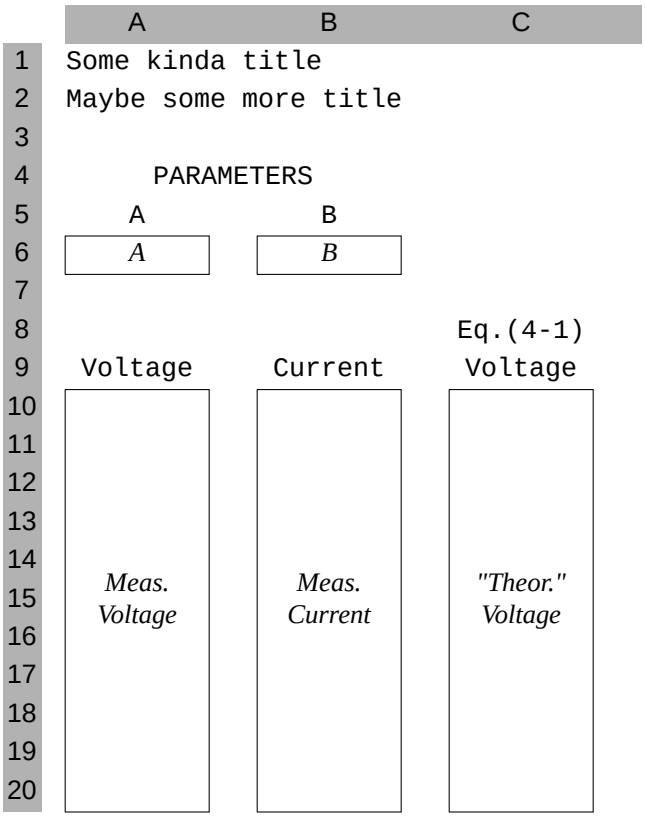

Figure 6−1. Schematic drawing of the worksheet layout

tedious and I'd probably make some mistakes. Fortunately, Quattro Pro has a `Move` command in the `Edit` menu that will do everything for me automatically, and I'll use that.

Exactly where should I move the stuff? I need space for cells containing the guessed parameters and for labels, so I decided to move the block down so that is starts in row `9`. To do that choose `Edit` from the Menu Bar, and `Move` from that menu. `Enter source block of cells:` will appear in the Input Line. Quattro Pro is asking you what block of cells is to be moved. We want to move the block `A5..B16`, so answer with that block, either by typing it in or by using the arrow keys to highlight the desired block. Press **Enter** when finished. `Destination for cells:` will then appear. We want to move the block to `A9..B20`. You can tell that to Quattro Pro either by typing it in, or by pointing the block out. Actually, you only need to tell Quattro Pro where the upper left hand corner of the block is to be put because it already knows how big the block is. You can specify either `A9.B20` or just `A9`. I suggest the latter because its easier and less prone to mistakes. In any case press **Enter**, and the block should move.

I want to point out to you one of the very useful features of spreadsheet programs. Recall that we had already set up the worksheet to graph the data in cells `A6..A16` and `B6..B16`. We have moved that data down, so to get a proper graph, the block of cells to be plotted has to be modified accordingly. Select the `Series` item from the `Graph` menu, and you will discover that Santa Claus has been here. The ranges have been modified automagically. That's the feature I wanted to point out to you. When you `Move` data around, or `Delete` or `Insert` rows or columns, or do several other operations that move the data around in your worksheet, the spreadsheet program tries to take care of all the associated housekeeping chores like adjusting ranges. One warning, though, the program is just that—a dumb program—and it

doesn't always do what seems obvious, so its usually a good idea to check to see if it did things the way you want.

With that bit of housekeeping out of the way, we are ready to add the new stuff. In row 5 I'll put labels for the guessed parameters, *A* and *B*. In A5 put ^A, and in B5 put ^B. (The initial caret, ^, tells Quattro Pro to center the label in the cell.) In row 4 I want to put the label PARAMETERS, spanning columns A and B and centered. I put ' PARAMETERS in A4, letting the contents of A4 spill over into B4. (The single quote, ', at the start tells Quattro Pro to start the label at the left side of the cell.) What should we put in cells A6 and B6 as values of *A* and *B*? I'll want to discuss this later, but doing so would interrupt the discussion here, so completely out-of-the-blue, let's just put in 1 and 1. Those are almost certainly lousy guesses, but they will get us started. Now let's put in the new column. First, I'll put in the label for the column. Put Eq.(4-1) in C8 and Voltage in C9. Next, we need to put a formula into C10 which will take the values of *A* and *B* contained in cells A6 and B6 along with the current contained in cell B10, and calculate a voltage according to Eq. (4−1). That's easy, just enter +A6+B6*B10. We have to start the thing with a plus sign so that the computer will recognize what we are entering as a formula, rather than text. The asterisk, *, is the symbol Quattro Pro uses to indicate multiplication. This formula tells the computer to take the number in A6 and add to it the product of the numbers in cells B6 and B10. Identifying A6 with *A*, B6 with *B*, and B10 with *I* in Eq. (4−1), that's just what the equation says. Notice that the number 1.0003 appears in the cell C10, whereas the formula we just entered appears on the Input Line (if the highlight is on cell C10). As a check, you can plug our guessed values for *A* and *B* along with the current from B10 in Eq. (4−1) yourself, and you should get the same number.

Now, another of the really useful features of spreadsheets: I can change my guess for *A* or *B* and immediately see the result on the calculated value. Move to A6 and change the number stored there from 1 to something else, say 10. You should immediately see the contents of C10 change accordingly. You can also try changing the guessed value of *B* stored in B6. If you had put the guessed values directly in the formula in cell C10, you would have to edit this line to change them. By putting them in a separate cell, you only have to change the contents of the cell. This feature is much more useful than it may appear here, as you will see shortly. When finished, put both constants back to 1 so we will be talking about the same thing.

6.3 Copying Stuff Between Cells

We now want to enter formulae similar to that in C10 in the remaining cells in the column. For example, in C11 we want +A6+B6*B11. We could just type this in, and type similar formulae in the remaining cells, but this could get to be quite a drag. Fortunately, spreadsheet programs have a capability to copy the contents of one cell into a range of cells. We can use this capability to copy the formula in C10 into the rest of the cells. To do so, choose the Copy item from the Edit Pull Down Menu. The line Source block of cells: should appear in the Input Line at the top of the screen. As usual with such questions, you can answer it either by typing the cell address (C10 in our case), or you can use the arrow keys to move the highlight to the desired cell. In either case press **Enter** when done. The line Destination for cells: should then appear in the Input Line. We want to answer C11 through C20, and can do so either by typing C11.C20, or by pointing as discussed before. Using either method, specify the C11..C20 range and press **Enter**.

Doing so yields what may be a surprising result—zeros appear in cells C11 through C13 and strange numbers are in the remaining cells in the column! To see what happened, move the highlight to cell C11. The formula contained in C11 will then be displayed on the Input Line. If you are unfamiliar with spreadsheet operation, what you see will probably be a surprise. After copying C10 to C11, C11 contains the formula +A7+B7*B11, not +A6+B6*B10 which was the formula in C10. In doing the copy, the program has automatically incremented the row numbers of each cell in the formula by one for us. The same thing has happened in the other rows of the copy.

Quattro Pro was trying to take care of the housekeeping chores for us, similarly to what it did with

the Move instruction. In this case Quattro Pro got part of it right, and part wrong. In each row we do want the current value used in the formula to be the one in that row, and Quattro Pro has done that for us, saving quite a bit of tedium with the edit key. That's the part it got right. On the other hand, we wanted to use the values contained in cells A6 and B6 in the formulae in all the cells in column C, but Quattro Pro erred and changed the row numbers of these cell references as well when it did the copying A kludgy way of solving the problem would be to use the editing feature of Quattro Pro to correct the formulae in C11..C20, cell-by-cell. That's a lot of trouble, and it is very error-prone. Fortunately, there is a better way.

6.4 Absolute and Relative Addresses

What we really want is that during the copy process the cell addresses A6 and B6 remain fixed while the address for B10 changes with row number. It is possible to tell Quattro Pro to do just this. Addresses that change during a copy process are called *relative* addresses, and those that remain fixed *absolute* addresses. Cell addresses in the form we used in cell C10 are recognized by Quattro Pro as relative addresses. Use the dollar sign, $, to make an absolute address. Actually, you can make an address *row-absolute* or *column-absolute*, or both. If we use the form $A6, the column letter, A, will not change during any copy but the row number, 6, can. Such an address is column-absolute. On the other hand, if we use A$6, the column letter can change, but the row number can't. That is a row-absolute address. To make an address absolutely absolute (that's not an official term), use two dollar signs, $A$6.

In our application, we want an absolutely absolute address, although a row-absolute address would work for now because we will only be copying from a cell to others in the same column. We need to change the formula in C10 to use absolute addresses. Move the highlight to C10, and use the Edit key (**F2**) to change the formula to read +$A$6+$B$6*B10. The number in the cell shouldn't change. Now use the Copy facility to copy this formula into cells C11 through C20. The numbers in column C should now be more satisfactory.

Try changing the guessed values for *A* and *B* in cells A6 and B6, and watch the values in column C change accordingly. Perhaps this is a better place to try making the point I mentioned at the end of Section 6−2. Instead of putting the values of *A* and *B* in two cells of the worksheet, and then using these cell addresses in the formulae in column C, we could have entered the values of *A* and *B* directly in the formulae in column C. If we had done so, however, to change either *A* or *B* we would have had to change all the formulae. Doing that wouldn't be too hard if we used the Copy feature intelligently, but it's still a lot easier to just change a single value in A6 or B6.

6.5 Back to Psyching Out A and B

Okey-Dokey! (How do you spell that anyway?) Now we have both Herbie's raw data, and the prediction of Eq. (4−1) in our worksheet. The next thing to do is add a plot to our graph showing the prediction. What we want is to plot the contents of column C on the same graph as the plot of column A. To differentiate the two, let's plot the "theoretical" prediction using lines to connect the points, and leave the measured plot as just symbols. Our goal will be to adjust the guessed constants, *A* and *B*, to get the "theoretical" and measured values to agree as closely as possible. Making the addition to our graph is easy, and you should know everything you need to do so, but I'll walk you through it just this one last time. I suggest you try it yourself first, without reading how I would do it. Just in case, before you do that save the worksheet in a file so you can recover from mess-ups.

Here is my suggested procedure. Get the Graph menu on the screen by choosing Graph from the Menu Bar. Choose Series, and then choose 2nd Series from the sub-menu and specify the range C7..C20. To make sure the new plot will use lines instead of symbols, choose the Customize Series item from the Graph menu. Then choose Markers & Lines, then Formats, then 2nd Series, and finally Lines. Press **F10** to see the graph. If the line color looks OK, great, otherwise change it by choosing Colors from the Customize Series sub-menu. When you are satisfied, back out of all the menus to get back to the worksheet.

Let's now get down to the business of choosing better values for *A* and *B*. To see how bad our initial blind guess was, press **F10** to put the graph up on the screen. It's not too hot, huh? To come up with better initial guesses we'll have to think a little. First, when the current, *I*, is zero, Eq. (4−1) says the voltage, *V*, is supposed to be *A*. If we look at Herbie's data, he has one point with current very nearly zero, and the corresponding voltage is 2.36 V, so a better guess for *A* would have been something like 2.36. Go back to the worksheet and make this change.

Put the graph back on the screen. That change didn't help much, the line is too horizontal; it needs more slope. We could randomly guess values for *B*, but let's think about this one too. Eq. (4−1) describes a straight line with slope of *B*. The slope is defined as the rise over the run. If we look at the extreme ends of Herbie's data, we get a rise of about 37, and a run of about .04. Dividing these gives 925. Go back to the worksheet and use this value for *B*. Put the graph back on the screen, and you'll see we are pretty close, except for the one questionable point at about 22 volts. For the moment, I'll just ignore this point.

It's now just a matter of trying several values of *A* and *B* around these and trying to choose the combination which seems to give the best fit. The continual changing between worksheet and graph gets tiring. Quattro Pro has a mode which allows you access to both the worksheet and the graph at the same time. To use it, you first have to do something Quattro Pro calls "change display mode." To do so, choose the `Options` item from the Menu Bar, and from that menu choose `Display Mode`. From the resulting sub-menu choose `Graphics Mode`. The screen should blank for a few seconds, and then the worksheet should reappear, looking a bit different. This is `Graphics Mode`. You'll find that the display changes a little more slowly in this mode than it did in the default text mode. If you want to get back, choose the item labeled `80x25` from the `Display Mode` sub-menu.

Now to get the graph on the screen at the same time as the worksheet, choose `Quit` to get back to the worksheet. Choose the `Graph` menu, and then choose the `Insert` item. Respond to the `Enter graph name` query with just **Enter**. You are then back at the worksheet, but with the line `Enter block to insert graph:` in the Input Line. The program is asking you where on the worksheet to put the graph. It will cover up, but not destroy, whatever is there. Use the standard techniques to specify a range. I suggest something like `D4..G16`. Back out of the menus, and you should see your graph lying on top of part of the worksheet. You can now change the values for *A* and *B* and see the effect on the graph (almost) instantly. If you want to remove the graph from on top of the worksheet, choose `Hide` from the `Graph` Pull Down Menu. In order to see the agreement better between the experimental data and the model you might want to increase the size of the inserted graph, but that will cover up part of the worksheet.

Figure 6−2 shows the resulting worksheet in two forms. Fig. 6−2a shows the formulae in each cell, through the eighth row, and Fig. 6−2b shows (more or less) what the worksheet looks like on the screen. For clarity of presentation I prettied it up a little by putting in some some lines and boxes. The value of the lines and boxes is questionable, but I think they do make the printed output easier to read.

I haven't discussed drawing lines on the worksheet yet, so here's the general idea. Drawing lines can help the appearance of the worksheet, but they have some undesirable side effects when the worksheet is displayed on the screen. When the worksheet is printed (as I did to make Figs. 6−2), the lines work quite a bit better, so I decided to include them on all the worksheet example figures. It's up to you whether or not to use such lines in your worksheets. If you decide to do so, choose the `Line Drawing` item from the `Style` menu. You are then asked to enter a block of cells. You can draw a box around this block, or put a line above, below, to the left, or to the right. Once you have specified the block, you are then asked which of these (or a few other) options you want. After answering that question, you are then asked what kind of line you want to draw. The only unclear choice on this menu is the `None` option. Use that to erase a line you have previously drawn. Play with the line drawing feature if you like. I don't really recommend using it regularly, but I thought I should tell you how I got the snazzy lines in the worksheets shown in the figures in these notes.

This example shows two of the important features of spreadsheet programs for engineering

a)

	A	B	C	D	E	F	G
1	Chapter 6: Comparison between Herbie's data and the						
2	predictions of the model in Eq. (4-1)						
3							
4	PARAMETERS						
5	A	B					
6	2.36	925					
7							
8			Eq.(4-1)				
9	Voltage	Current	Voltage				
10	2.36	-0.00029	+$A$6+$B$6*B10				
11	36.8	0.0394	+$A$6+$B$6*B11				

b)

	A	B	C	D	E	F	G
1	Chapter 6: Comparison between Herbie's data and the						
2	predictions of the model in Eq. (4-1)						
3							
4	PARAMETERS						
5	A	B					
6	2.36	925					
7							
8			Eq.(4-1)				
9	Voltage	Current	Voltage				
10	2.36	-0.00029	2.09175				
11	36.8	0.0394	38.805				
12	9.41	0.00749	9.28825				
13	15.1	0.015	16.235				
14	22.1	0.0106	12.165				
15	32.4	0.0353	35.0125				
16	4.83	0.00359	5.68075				
17	21.5	0.0251	25.5775				
18	29.5	0.0323	32.2375				
19	17.7	0.0205	21.3225				
20	25.9	0.027	27.335				

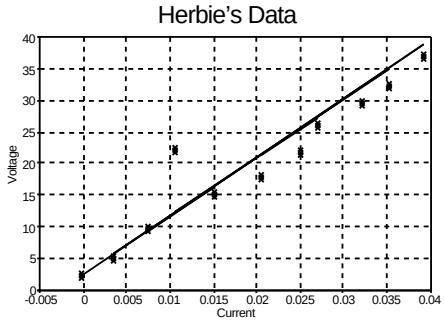

Figure 6−2. The worksheet described in the text to determine the best values of the constants *A* and *B* in Eq. (4−1) applied to Herbie's data. Part a) shows the formulas in the individual cells, and part b) shows the worksheet more or less as it appears on the computer screen.

applications.

1. Spreadsheets offer an easy method of plotting functions, both functions defined by formulae, and functions defined as tables of values.

2. By simply changing the values in either of the cells corresponding to *A* and *B*, we cause the computer to recompute 11 numbers (the predictions of Eq. (4−1)), and to then redraw the graph accordingly. Think how long it would take you to do that by hand, even with a calculator! With the

computer, you can spend more time trying to solve your problem, and less doing mundane arithmetic.

6.6 A Better Way to Determine A and B

The graphical method of determining *A* and *B* works pretty well, but it gets a bit tricky to fine tune these values. Let's develop a more quantitative method. What we want is to find values of *A* and *B* such that the "average distance" between the predicted and measured values is least. Here's an idea that almost works. Suppose we add another column, in which for each row we put the difference between the measured value and the corresponding "theoretical" value. Let's call each of these differences *errors*. We could then add up these errors and display the result as a total error in another cell, say off to the side somewhere. We could then change the values of *A* and *B* and watch the change in this total error, seeking to make it as small as possible.

Here's the flaw with this idea. Consider two measured points, one lying on one side of the corresponding "theoretical value, the other on the other side of its "theoretical" value. The sign of the error at one point is the opposite of the sign at the other. Even though these two errors may be large in magnitude, when added they tend to cancel, implying a smaller error than actually exists. A fix for this problem is to add the square of the errors, instead of the errors themselves. Doing so gives a measure of total error which is guaranteed to increase whenever any one of the individual errors increases. (We could also add the absolute values, but adding the squares is easier and more common in practice.) The modified idea, then is to put in each cell of the new column the square of the difference between the measured and "theoretical" values. We will then sum up all these squared errors to obtain a measure of how good our fit is.

As you may have guessed, I'm not the first one to have gotten this bright idea. If we were to divide this sum of the squared errors by the number of data points, we would have the mean (or average) squared error, and if we then were to take the square root of that we would have the average error without considering the sign. Such an average is called the *Root Mean Square*, or *R.M.S.* average, and it appears in all sorts of places. For example, the voltage that the power company supplies to wall plugs alternates in polarity 120 times a second. A simple time average of this voltage would give zero. The voltage is not always zero (I can attest to that from painful personal experience), but rather the voltage is positive as often as it is negative. For this reason, the voltage is usually measured as an R.M.S. average. The R.M.S. voltage at a typical wall plug is about 110 V, and the maximum voltage is about 154 V. As another example, much closer to our problem, the idea of taking an R.M.S. average of a set of errors is also not a new one, and such an average is called the *standard deviation*.

Before implementing this idea, we need to make a decision about whether or not to keep Herbie's questionable point at about 22 V and 0.011 A in the table. If I keep it in, my error estimate will be influenced strongly by it. All the points except this one lie pretty close to a straight line and will contribute only a small amount to the error for well chosen values of *A* and *B*, but this point lies quite a distance from the straight line and will make a major contribution to the error. I decide that Herbie most likely made a mistake in reading the meters, and I will remove this one point from the table. Going back to the worksheet, that's the point in row `14`. I'll move it down to row `23` instead of completely throwing it away. First, I use `Move` to move `A14..C14` to `A23..C23`. Doing that leaves me with a blank line at row `14`, in the middle of my table. To get rid of it, use the `Delete` item of the `Edit` menu. After choosing this item, Quattro Pro asks you whether you want to delete some rows or some columns. In this case you should choose `Rows`. Quattro Pro then asks you what block of rows you want to delete. For each row selected, the entire row will be deleted, so you do not need to worry about including all cells in a row. Just make sure you specify all the rows in the block that you want to delete, and no more. Specify `A14` or any range containing only cells in row `14`. Pressing **Enter** should zap the unwanted row.

Now check out the `Graph` menu to see what has happened to the ranges selected for graphing. You should find that Quattro Pro has changed the range to extend only as far as row `19`. The `Move` and `Delete` operations generally make this kind of correction automatically, but sometimes things get messed

up so it's usually a good idea to check that things got corrected correctly. At any rate, the graph should be the same as before, except that the bum point is absent.

There is also an `Insert` item in the `Edit` menu. This item works similarly to the `Delete` item, except it inserts one or more rows or columns into the worksheet.

6.7 Back to Determining A and B

We now have things cleaned up and can go back to implementing our plan. We want to put the square of the difference between the measured and "theoretical" values in column `D`. First label the column by putting `Squared` in `D8` and `Error` in `D9`. Now move to `D10` and enter the formula `+(A10-C10)^2`. Here `^2` means "caret 2", not **Ctrl** 2. (Caret is the symbol on top of the **6** key.) Quattro Pro interprets the caret, `^`, as meaning "raised to the power," so `(A10-C10)^2` is interpreted as meaning `(A10-C10)*(A10-C10)`. Copy the contents of `D10` to the rest of the rows in the column, `D11..D19`.

We now want to add up the squared errors. We could do this with a standard formula containing ten terms, `+D10+D11+` ... `+D19`, but typing that would be a drag. Quattro Pro has a number of *built-in functions*, and one of these is called `@SUM(`*Range*`)`. This function adds up the numbers in all the cells in the specified range. We will use this function, and put it in the cell `F10`. Move to `F10`, and enter `@SUM(D10..D19)`. Just so we remember what this is, enter `^Total` in `F8` and `^Error^2` in `F9`. By the way, another advantage of using the `@SUM` function over just entering a formula is that it makes it more likely that Quattro Pro will make the correct changes if we insert or delete a row from within the range we want to sum over. Figs. 6−3 show the worksheet at this point.

Everything is now ready, so we start trying different values of *A* and *B* to find the combination which produces the minimum total errors. Our best guesses for *A* and *B* were about 2.3 and 925. Put these numbers in cells `A6` and `B6` and note the total squared error in `F10`. Try changing *A* to minimize the total error. Then vary *B*, again, to minimize the error. Go back then and vary *A* again, then *B*, and continue until you are close enough to the values giving the minimum error. In a few minutes, I was able to get the total squared error down to about 10.7.

6.8 Partially Automating the Process

Manually varying *A* and *B* in an attempt to minimize the total squared error gets a bit tiring after a while. It is possible to get Quattro Pro to do a lot of the work for us automatically. To do that we will have to write and use a *macro*. Any real discussion of writing and using macros is beyond the scope of these notes. My goals in this sub-section are to show you some of the power and utility of spreadsheet programs, and to introduce or reinforce some ideas which should be useful to you. For most of the spreadsheet programming, I'll just tell you what to type in. I'll try to give you enough information so that you can see what's happening, but I won't expect you to be able to write your own macros after you are finished with this section. If you want more information, look in the manuals or in a book on Quattro Pro.

We are probably beating a dead horse here, but we want to figure out the values of *A* and *B* which give the lowest possible total squared error. We already have values which are accurate to several decimal places, and that accuracy almost certainly exceeds the accuracy of the meters Herbie used to measure *V* and *I*. I'll forge ahead anyway, because there are a couple of things I want to show you.

I want to program Quattro Pro to calculate automatically the total squared error for several different values of *A* around the value that gives the minimum error, and then to plot the these errors versus the *A* value. The plot will show how the error depends on the choice of *A*, and it will help us to find the value giving the minimum error. I'll store the starting value of *A* in cell `C5`, and the ending value in `C6`, and I'll calculate 21 points evenly spaced between the two end values. I'll store the values of *A* in cells `F22..F42` (note that's 21 cells, not 20—I mess that one up more often than not), and the corresponding values of total squared error in `E22..E42`.

One way to do all this would be to do it one value at a time. We could first put one value for *A* in `A6`

a)

	A	B	C	D	E	F	G
1	Chapter 6: Total squared error between Herbie's data and						
2	the model described by Eq. (4-1)						
3							
4	PARAMETERS						
5	A	B					
6	2.1	850					
7							
8			Eq.(4-1)	Squared		TOTAL	
9	Voltage	Current	Voltage	Error		ERROR	
10	2.36	-0.00029	+$A$6+$B$6*B10	(A10-C10)^2		@SUM(D10..D19)	
11	36.8	0.0394	+$A$6+$B$6*B11	(A11-C11)^2			

b)

	A	B	C	D	E	F	G
1	Chapter 6: Total squared error between Herbie's data and						
2	the model described by Eq. (4-1)						
3							
4	PARAMETERS						
5	A	B					
6	2.1	850					
7							
8			Eq.(4-1)	Squared		TOTAL	
9	Voltage	Current	Voltage	Error		ERROR	
10	2.36	-0.00029	1.8535	0.256542		10.6641	
11	36.8	0.0394	35.59	1.4641			
12	9.41	0.00749	8.4665	0.890192			
13	15.1	0.015	14.85	0.0625			
14	32.4	0.0353	32.105	0.087025			
15	4.83	0.00359	5.1515	0.103362			
16	21.5	0.0251	23.435	3.744225			
17	29.5	0.0323	29.555	0.003025			
18	17.7	0.0205	19.525	3.330625			
19	25.9	0.027	25.05	0.7225			
20							

Figure 6−3. My worksheet that calculates the total squared error between Herbie's empirical data and the predictions of the model described by Eq. 4−1. Part a) shows the formulae in the individual cells, and part b) shows the worksheet as it appears on the screen.

and copy the resulting error value from `F10` to `E22`. We would then increment the value of *A* in `A6` and repeat the process, copying the error this time into `E23`. Continuing until the value of *A* in `A6` reaches the upper limit would generate the table of errors in `E22..E42`. We would also want to know the values of *A* that produced each error, so every time we copy an error value to a cell in column `E` we should also copy the value of *A* stored in `A6` into the adjacent cell in column `F`.

This procedure would be tedious. Fortunately, Quattro Pro has the capability to automate such a process. To do use it, you have to write what's called a *macro*. A macro is just a sequence of commands which Quattro Pro is to execute sequentially. The command sequence is written in an arcane language and stored in cells in an otherwise unused column of the worksheet itself. To create a macro, you first have to learn the arcane language, and then you just enter the sequence of commands into unused cells in the worksheet just as you would any text. Once written, a macro is executed by choosing `Tools` from the Menu Bar, choosing `Macro` (alternately you can just press **Alt-F2** to get to this place), and then choosing `Execute`. The program then asks on the Input Line for the range to execute, and you should respond with the address of the first cell of the macro. You can give your macros names (use `Tools` | `Macro` | `Name`), and if you give a macro a two character name consisting of a backslash, `\`, followed by a letter, you can execute the macro by simply pressing the **Alt** key and the letter at the same time.

Anyway, here's the magic recipe; I'll try to provide a brief explanation of each line. First, we should put in some text to describe what the macro does. Programmers refer to this as adding comments. I've found that it is impossible to put too many comments into programs; no matter how many I put in, there is always some non-obvious part that I forget to describe, and have a tough time figuring out when I come back to the program several months after writing it. Move to cell `A21` and enter the following text

```
Macro to determine total squared
```

and move to `A22` and complete the comment by entering

```
error for a range of A values
```

We'll put the name of the macro in `A24`. This is not needed, but it seems to be customary. Its a comment, the program doesn't need it but the human might. I want to be able to execute it easily, so I'll call it `\A`. (You have to start `\A` with a character indicating this is text, or else you will get a cell-full of `A`'s.) Enter this label in `A24`.

Now we enter the macro. Move to `B24` and enter (you have to get this exactly right)

```
{GOTO}E21~
```

The purpose of this line is to position the highlight to the cell just above the first cell in the column in which we are going to store the results of our calculation (making it the current cell). The `{GOTO}` is the same as pressing **F5** (the GOTO key), and the remainder of the line is the response to the program's query about just where it should go. (There have been times when my answer would not have been `E21`!) The last character is a tilde, and is located on the key just to the left of the **1** key at the top of the keyboard. This symbol is "macro-ese" for pressing **Enter**.

Move to the next line down, `B25`, and enter

```
{FOR A6,C5,C6,F15,B27}.
```

This is the main control line which causes Quattro Pro to calculate the total squared error for the 21 values of *A* between the value stored in `C5` and that in `C6`. The statement causes the computer first to copy the value in cell `C5` into `A6`, and then to execute the block of macro code starting at `B27`. When that block is finished, the program adds the value contained in `F15` to that in `A6`, and if the result is less than or equal to the value in `C6`, it executes the block starting at `B27` again. This process continues until the value in `A6` is larger than the value in `C6`. This kind of a structure appears frequently in all sorts of programming, and it is called a *do loop*, or sometimes a *for loop*.

We need to put a block of statements in `B27` that does the following:

1. Move down one cell;
2. Copy the value of total squared error from cell `F10` to the current cell;
3. Move one cell to the right;
4. Copy the value of *A* in `A6` into this cell;

5. Move back one cell to the left to get ready for the next iteration of the for loop.

Here's the program block to do that. Move to B27 and enter

{DOWN}

This line moves the highlight down one cell to place it over the cell into which the next error result is to be placed. In B28 put

{/ Block;Values}F10~~

Note the space between the slash, /, and Block. This line copies the value (not the formula!) in cell F10 into the current cell, thereby saving the total squared error for the value of *A* during this iteration of the loop. The {/ Block;Values} part of this line is the same as choosing Edit from the Menu Bar, and then choosing Values. This item is used to copy the contents of a range of cells to another range when the source cells contain formulas, and you want to copy the values of the formulas (numbers), not the formulas themselves. If you did that yourself at the keyboard of your computer, Quattro Pro would respond by asking in the Input Line for the range to copy from. The part of the line F10~ answers this question, just as you would at the keyboard, with F10, followed by pressing **Enter**. If you were doing this manually, the Quattro Pro would then ask for the cell to copy the stuff to. The second tilde answers this query with just **Enter**, indicating the current cell.

We have now copied the total squared error into the appropriate cell of column E, and we now need to copy the value of *A* which produced it into the cell just to the right. First, we make this cell the current cell by moving one cell to the right. Move to B29 and enter

{RIGHT}

Then move to B30 and enter

{/ Block;Values}A6~~

This line is similar to the one in B28, and copies the current value of *A* in A6 into the current cell.

Finally, we need to position the highlight to get ready for the next iteration, that means we need to move one cell to the left. Therefore move to B31 and enter

{LEFT}

That finishes the macro; we now need to name it so that it's easy to execute, and to fill in the cells containing the endpoints and the increment for *A*. To name the macro choose Macro from the Tools menu; choose the Name option, choose Create; and then name the macro \A. Quattro Pro will then ask for the block containing the macro, enter B24..B25 (B24..B31 also works). Now we only need to enter the desired starting and ending values of *A* into cells C5 and C6, and the put the proper increment into F15. Move to C4, and enter the label A RANGE. Then enter appropriate values in C5 and C6 like 1.3 and 2.5. Let's also make the number of points to be calculated a parameter we can adjust. I put the labels NUMBER and POINTS in E4 and E5, and 21 into E6. While we're into labeling everything, move to A1 and put in a title for the worksheet

Finally, we want to carry out the calculation for *N* equally-spaced values of *A*, so we need to figure out what spacing to do that. We want to put this value in cell F15, where the macro we just wrote expects to find it. The desired spacing is

$$\Delta A = \frac{A_{max} - A_{min}}{N - 1}$$

Therefore enter

(C6-C5)/(E6-1)

into cell F15.

We're now ready to light the thing off. Just to make sure we get the same results, put 875 in cell B6 for *B*, and 1.3 and 2.5 in C5 and C6. Then simultaneously press **Alt** and **A**. It may look like nothing is happening, but look in the lower right-hand corner of the display, and you should see that Quattro Pro is going bananas. After a short while, the process should be finished and you should have a bunch of numbers in both columns E, and F, starting at row 22.

I want to graph our results. First let's name our existing graph (containing Herbie's raw data and a plot of the prediction of Eq. (4−1)) so that we will not lose all the settings when we create our new graph. From the Menu Bar choose `Graph` and choose `Name` from that menu. Choose `Create` from that sub-menu, and name the graph something, say `DATA`. If you ever want to retrieve this graph, use the `Display` Item on the sub-menu. With the old graph named, make a graph plotting the values in `F22..F42` on the x-axis and `E22..E42` on the y-axis. (You can erase the `2nd Series` plot by using the `Reset` option of the `Customize Series` sub-menu of the `Graph` menu.) You'll probably want to change the labels on the x and y axes, as well as the title for the graph.

When everything looks good, press **F10** to display the graph. You should see a figure that looks like a parabola, with a minimum at about 1.9. If you wanted to get this value more accurately you could go back to the worksheet and change the range of values for *A*, say to 1.8 through 2.0. Our value of 1.9 is good enough for our purposes.

This value of *A* is the one which produces the minimum error for the value of *B* in cell `B6` (`875`). But this value of *B* may not be the best. We therefore want to store our best guess of *A* (`1.9`) in `A6`, and then modify our macro so that we can get a plot of error versus *B* instead of *A*. Instead of actually modifying the macro, let's copy it to another part of the worksheet, and then modify the copy. That way we'll have two macros, one to calculate error as a function of *A*, and the other as a function of *B*.

Copy the macro, along with associated comments, to the block starting at `A33`. Let's store the limits for the FOR loop in cells `D5` and `D6`, and use `F18` for the increment. Make the needed changes in the comment lines, and change the contents of cell `A36` from `\A` to `\B`. Modify the FOR loop line in `B37` to read

`{FOR B6,D5,D6,F18,B39}`

Change the cell reference in `B42` from `A6` to `B6`. Name the new macro `\B`, and specify the proper block containing it. Put the label `B RANGE` in `D4`, and put the values `850` and `870` in `B5` and `B6`, and enter the formula for calculating the proper increment for *B* in `F18`, `(D6-D5)/(E6-1)`.

That should do it! Fig 6−4 shows my worksheet as it appeared on my computer screen. Use the best guess for *A* by putting `1.9` in `A6` and hit the juice by pressing **Alt-B**. When it's done press **F10** to get a graph. You should have a parabola with minimum at about 858. This is the value of *B* giving the least error when $A = 1.9$. Now you need to use our new value of *B* and go back and vary *A* again. You can see-saw back and forth between varying *A* and *B* fairly easily with our setup.

Before I leave this subject, there's one mud puddle lying around I should tell you about in case you happen to fall in. Occasionally, you may find that a macro does not calculate the last value, but instead stops one step short. That can cause some strange looking graphs of error vs *A* or *B*. How can this happen? The problem has to do with the arithmetic accuracy of the computer. It's very accurate, keeping about fourteen digits, but not perfect. Fourteen digits is more than enough for most applications, but look what can happen in the `FOR` loop. Suppose that the value in `C6` is `2.0` and that the increment to be used for *A* is `0.02`. After the 20th iteration the value of *A* in `A6` may not be exactly 1.98, but rather 1.9800000000001 because of the accumulation of small arithmetic roundoff errors. Then when the `0.02` increment is added to the value in `A6`, the result is 2.0000000000001, not 2.0. That's larger than the upper limit so the computer does exactly what it was told to do, it stops because the value in `A6` is larger than the upper limit for the `FOR` loop stored in `C6`. That means it would stop without doing the last step.

This is a trivial, but remarkably annoying problem, that keeps popping up unexpectedly in diverse programming applications. In our application, one solution is to make the upper limit of the `FOR` slightly larger than the limit we really want (maybe by multiplying the value in `C3` by 1.000000001). Another solution is to just ignore the last point by reducing the range of points to be plotted by one row.

6.9 The Ultimate Solution

Although the spreadsheet eliminates a lot of the arithmetic drudgery, it still gets tiring going back and forth between minimizing the error with respect to *A* and then *B*. It happens there is a better way which I'd

Chapter 6: Total squared error vs either A or B calculated with macros. Use \A for A, \B for B dependence.

PARAMETERS		A RANGE	B RANGE	NUMBER
A	B	1.2	850	POINTS
2	875	2	870	21

Voltage	Current	Eq.(4-1 Voltage	Squared Error
2.36	-0.0003	1.74625	0.37669
36.8	0.0394	36.475	0.10562
9.41	0.00749	8.55375	0.73316
15.1	0.015	15.125	0.00063
32.4	0.0353	32.8875	0.23766
4.83	0.00359	5.14125	0.09688
21.5	0.0251	23.9625	6.06391
29.5	0.0323	30.2625	0.58141
17.7	0.0205	19.9375	5.00641
25.9	0.027	25.625	0.07562

TOTAL ERROR
13.278

DELTA A
0.0381

DELTA B
0.95238

Macro to determine total square error for a range of A values

```
\A {GOTO}E21~
   {FOR A6,C5,C6,F15,B27}

   {DOWN}
   {/ Block;Values}F10~~
   {RIGHT}
   {/ Block;Values}A6~~
   {LEFT}
```

Macro to determine total square error for a range of B values

```
\B {GOTO}E21~
   {FOR B6,D5,D6,F18,B39}

   {DOWN}
   {/ Block;Values}F10~~
   {RIGHT}
   {/ Block;Values}B6~~
   {LEFT}
```

Error	A or B
12.932	1.2
12.6582	1.2381
12.4135	1.27619
12.1977	1.31429
12.011	1.35238
11.8534	1.39048
11.7247	1.42857
11.6251	1.46667
11.5545	1.50476
11.5129	1.54286
11.5004	1.58095
11.5168	1.61905
11.5623	1.65714
11.6369	1.69524
11.7404	1.73333
11.873	1.77143
12.0346	1.80952
12.2252	1.84762
12.4449	1.88571
12.6936	1.92381
12.9713	1.9619

Figure 6−4. The worksheet as it appears on the computer screen, showing the two macros for calculating the dependence of the total square error on the value of *A* or *B*.

like to show you. Figure 6−5 shows a typical plot of the total squared error vs *A* for a fixed value of $B = 860$. A plot of total squared error vs *B* for a fixed value of *A* looks similar. These graphs look surprisingly like parabolas, and in fact they *are* parabolas. If you don't panic it's even not too hard to see that they are. Consider, for example the graph in Fig. 6−5. Recall that if this graph is to be a parabola the total

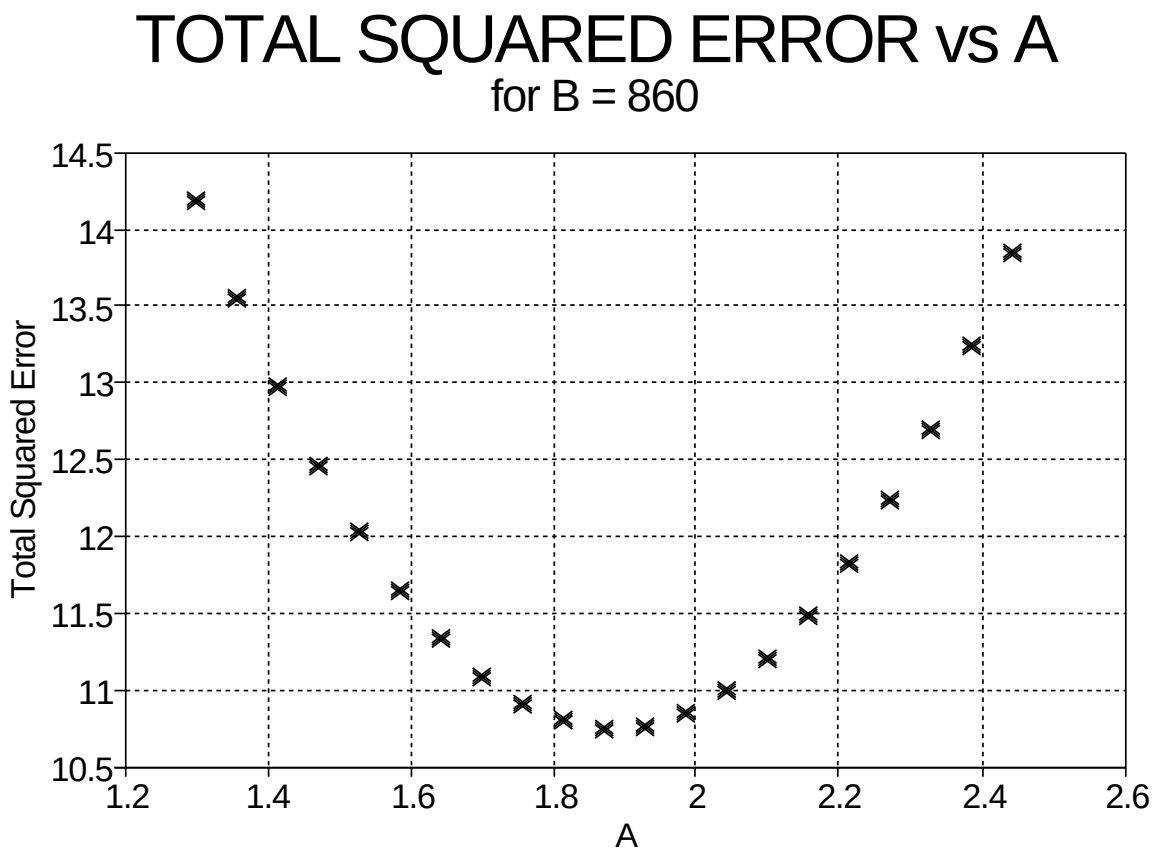

Figure 6−5. Plot showing the dependence of the total squared error on A for $B = 860$.

squared error must be given by a formula which is the sum of a term quadratic in A, one linear in A, and a constant, i.e.

$$\text{total squared error} \quad = \quad aA^2 + bA + c \tag{6−1}$$

where a, b, and c are numbers that do not depend on A. Now, the total squared error is the sum of the squares of the differences between the measured voltages and the values predicted by Eq. (4−1). The question is whether or not this total squared error can be written in a form like Eq. (6−1). If we can do so, we will get a bonus; knowing the coefficients a, b, and c, we can tell what value of A produces the minimum error! You may not remember, but you should have found in high school math that for a parabola described by Eq. (6−1), the value of A corresponding to the bottom (or top if the parabola opens downward) is

$$A_{min\ error} = -\frac{b}{2a} \tag{6−2}$$

This result can be obtained either by clever use of the quadratic formula, or by using calculus to find the extremum.

Before continuing, let's generalize a bit. Suppose we have a table of N pairs of measured values similar to Table 6−1 which we suspect has some random noise or other errors in it. Suppose also that we have reason to believe that these pairs are related by a formula of the same form as Eq. (4−1)

$$y(x) = A + Bx \tag{6−3}$$

where A and B are constants, the values of which we do not know, but want to find through comparison

x	y
x_1	y_1
x_2	y_2
3*dot*	3*dot*
x_N	y_N

TABLE 6−1. Table of measured values.

with our measured data. Because of the noise in the data, there will be some uncertainty in the values we come up with, but, just as with Herbie's data, we decide that the best values to use are those which minimize the discrepancy between the formula prediction and the measured values.

According to Eq. (6−3), when x is x_i we expect y to be

$$y(x_i) = A + Bx_i \tag{6−4}$$

Because of noise or other errors in the measurement, and maybe because Eq. (6−4) only approximately describes the relation between y and x, the values predicted by Eq. (6−4) for each x_i will probably not agree exactly with the corresponding measured values. Let's use the symbol δ_i to stand for the difference between the predicted and measured values at the i^{th} point. (People often use δ or Δ to stand for a quantity which they either want or expect to be small.) Then δ_i^2 is

$$\begin{aligned} \delta_i^2 &= \Big[y(x_i) - y_i \Big]^2 \\ &= \Big[(A + Bx_i) - y_i \Big]^2 \end{aligned} \tag{6−5b}$$

The total squared error is just the sum of these δ^2's. I'll call it Δ.

$$\begin{aligned} \Delta &= \delta_1^2 + \delta_2^2 + \cdots + \delta_N^2 \\ &= \sum_{i=1}^{N} \delta_i^2 \end{aligned} \tag{6−6b}$$

Eq. (6−6b) may be unfamiliar if you haven't encountered this notation, but it is just shorthand for Eq. (6−6a), nothing more.

Putting in Eq. (6−5b) for δ_i^2, we get

$$\Delta = \sum_{i=1}^{N} \Big[A + Bx_i - y_i \Big]^2 \tag{6−7}$$

This, when considered as a function of A, is supposed to describe a parabola. To see that it does, we just expand the square,

$$\Big[A + (Bx_i - y_i) \Big]^2 = A^2 + 2A(Bx_i - y_i) + (Bx_i - y_i)^2 \tag{6−8}$$

The total squared error, Δ, is just the sum over i of the three terms on the right hand side of Eq. (6−8), and in this form we can see that it will indeed be a parabola of the form of Eq. (6−1). The sum over the first term will supply the aA^2 term of Eq. (6−1), the sum over the second the bA term, and the sum over the third the constant c term. It only remains to work out what the three constants a, b, and c are.

The first one is easy,

$$\sum_{i=1}^{N} A^2 = A^2 + A^2 + \cdots + A^2 \qquad (6-9)$$

$$= N\,A^2$$

implying

$$a = N \qquad (6-10)$$

The next one is a little more involved.

$$\begin{aligned}\sum_{i=1}^{N} 2A(Bx_i - y_i) &= 2A(Bx_1 - y_1) + 2A(Bx_2 - y_2) + \cdots + 2A(Bx_N - y_N)\\ &= 2A\Big[(Bx_1 - y_1) + (Bx_2 - y_2) + \cdots + (Bx_N - y_N)\Big]\\ &= 2A\Big[(Bx_1 + Bx_2 + \cdots + Bx_N) - (y_1 + y_2 + \cdots + y_N)\Big]\\ &= 2A\Big[B(x_1 + x_2 + \cdots + x_N) - (y_1 + y_2 + \cdots + y_N)\Big]\\ &= 2A\left[B\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} y_i\right]\end{aligned} \qquad (6-11)$$

Thus the constant b is

$$b = 2\left[B\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} y_i\right] \qquad (6-12)$$

To figure out what value of A gives the minimum error we don't need to know the constant c, but I'll put down what it is anyway.

$$c = \sum_{i=1}^{N} (Bx_i - y_i)^2 \qquad (6-13)$$

We can now write down an expression giving the value of A which produces the minimum error. First, to save some writing, I will make the following two definitions.

$$S_x = \sum_{i=1}^{N} x_i$$

$$S_y = \sum_{i=1}^{N} y_i \qquad (6-14b)$$

Then in terms of the constants a and b, the value of A corresponding to the bottom of the parabola (and, hence to the value producing the minimum error) is

$$\begin{aligned}A_{min\ error} &= -\frac{b}{2a}\\ &= \frac{S_y - BS_x}{N}\end{aligned} \qquad (6-15)$$

From Eq. (6−15) you can see why the value of A corresponding to the bottom of the parabola changed when you changed B.

In a similar way we can find the value of B which, for a fixed value of A, produces the minimum error. To do so, we write the square just as in Eq. (6−8), except using a different ordering of the terms.

$$\Big[Bx_i + (A - y_i)\Big]^2 = B^2x_i^2 + 2Bx_i(A - y_i) + (A - y_i)^2 \tag{6−16}$$

Again, Δ is just the sum over i of the three terms on the right hand side, and we see that it also is of the form of Eq. (6−1), except with A replaced with B. Δ should then also produce a parabola when plotted vs. B with a fixed A. We obtain the coefficients a, b, and c in the same way as before.

The first term on the right side of Eq. (6−16) produces the square term,

$$\begin{aligned}\sum_{i=1}^{N} B^2x_i^2 &= B^2x_1^2 + B^2x_2^2 + \cdots + B^2x_N^2 \\ &= B^2(x_1^2 + x_2^2 + \cdots + x_N^2) \\ &= B^2\sum_{i=1}^{N} x_i^2\end{aligned} \tag{6−17}$$

Thus for the B parabola, the coefficient a is

$$a = \sum_{i=1}^{N} x_i^2 \tag{6−18}$$

The second term on the right side of Eq. (6−16) produces the linear term of Eq. (6−1).

$$\begin{aligned}\sum_{i=1}^{N} 2Bx_i(A - y_i) &= 2Bx_1(A - y_1) + 2Bx_2(A - y_2) + \cdots + 2Bx_N(A - y_N) \\ &= 2B\Big[x_1(A - y_1) + x_2(A - y_2) + \cdots + x_N(A - y_N)\Big] \\ &= 2B\Big[A(x_1 + x_2 + \cdots + x_N) - (x_1y_1 + x_2y_2 + \cdots + x_Ny_N)\Big] \\ &= 2B\left[A\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} x_iy_i\right]\end{aligned} \tag{6−19}$$

The constant b in this case is then

$$b = 2\left[A\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} x_iy_i\right] \tag{6−20}$$

This information is sufficient to write an equation giving the value of B which produces the minimum error (for a given value of A, of course). As before, I'll make a few definitions to reduce the writing. I'll number them (6−14c) and (6−14d) to go with the similar definitions in Eq. (6−14a) and (6−14b).

$$\begin{aligned}S_{xx} &= \sum_{i=1}^{N} x_i^2 \\ S_{xy} &= \sum_{i=1}^{N} x_iy_i\end{aligned} \tag{6−14d}$$

Then our sought-after value of B is

$$B_{min\ error} = \frac{S_{xy} - AS_x}{S_{xx}} \tag{6−21}$$

We now have two expressions, Eq. (6−15) and Eq. (6−21). One tells you for a given value of B what value of A produces the minimum error; the other works vice versa. If you are skeptical, these claims can be checked fairly easily with the spreadsheet. (Even if you are not, this is an exercise at the end of this section, and it may be assigned as homework.) For example, you can set B to some value, run the \A macro, and then graph the result to produce a parabola. Be sure to choose a range of A's which includes the bottom of the parabola. You can also evaluate Eq. (6−15) for this same value of B to see where it predicts the bottom of the parabola will be. (Note S_x and S_y are just the sum of the data in columns B and A, respectively.) It's all on the line here—either it works or it doesn't. You should find that it does.

What we seek, however, is the combined set of *both* A and B which give the minimum minimum error. Finding that set is pretty easy. Equations (6−15) and (6−21) provide two equations in two unknowns, A and B. If we can find the values which satisfy these two equations simultaneously, we will have found the values which produce our holy grail, the minimum minimum error. Multiplying both sides of Eq. (6−15) by N, and of Eq. (6−21) by S_{xx} and rearranging some terms gives

$$\begin{aligned} NA + S_x B &= S_y \\ S_x A + S_{xx} B &= S_{xy} \end{aligned} \tag{6−22}$$

To solve for A, multiply the top equation by S_{xx}, the bottom by S_x, and subtract, giving

$$(NS_{xx} - S_x^2)A = S_{xx}S_y - S_x S_{xy} \tag{6−23}$$

or, finally

$$A_{min\ min\ error} = \frac{S_{xx}S_y - S_x S_{xy}}{NS_{xx} - S_x^2} \tag{6−24}$$

To solve for B, just multiply the first equation in Eqs. (6−22) by S_x, and the second by N, giving

$$(S_x^2 - NS_{xx})B = S_x S_y - NS_{xy} \tag{6−25}$$

or

$$B_{min\ min\ error} = \frac{NS_{xy} - S_x S_y}{NS_{xx} - S_x^2} \tag{6−26}$$

That's it, the ultimate solution (to this problem at least)! Let's recap what happened in case you got snowed by all the symbols. Eq. (6−24) and (6−26) are supposed to give the values of A and B which cause the relationship in Eq. (6−3) to agree the most closely with a set of measured data put down in a table such as Table 6−1. The right-hand side of these equations involves only the number of points in the table, and four sums which can be calculated from the table. In fact, it is easy to program a spreadsheet to calculate the sums and to use them to calculate the optimum A, and B. Programming a worksheet to do that is pretty easy. I have a few suggestions that may make it even easier. First, you will have to evaluate two kinds of sums, {S_x and S_y} and {S_{xx}, S_{yy}, and S_{xy}}. You can use the `@SUM(`*Cell Range*`)` function that we have already discussed to do evaluate S_x and S_y easily. There is also an `@` function that will allow you to evaluate the other three sums: `@SUMPRODUCT(`*Cell Range*`,`*Cell Range*`)`. This function takes two ranges as arguments, multiplies the individual elements, term-by-term, and sums the products. Thus, to evaluate S_{xx} with the worksheet shown in Fig. 6−3 you could use the formula `@SUMPRODUCT(B10..B19,B10..B19)`. Second, you will need the number of data points in the table. You could count them by hand, but better is to use the `@COUNT(`*Cell Range*`)` function.

I won't make the worksheet for you, but I will tell you what I got for the best values of A and B: $A_{min\ min} = 2.0377$ and $B_{min\ min} = 852.63$. Those values give a total squared error of 10.65. This is supposed to be the best we can do—you should not be able to find values of A and B that give a smaller error. The error is usually reported not as a total squared value, but rather as a root mean square value. The R.M.S. error in our example is just $\sqrt{\frac{10.65}{10}} = 1.03$.

6.10 Exercises

For all problems requiring the use of Quattro Pro, when you are satisfied with your worksheet save it to a file on a floppy and turn the floppy in as part of the homework assignment. You should also include a clear written description of what you did and why. That can be handed in on a separate piece of paper, or it can be included in the worksheet as a comment. You can put more than one worksheet file on a floppy, but make sure you give each a name that makes it obvious which file goes with which problem.

1. Carry out the suggested procedures in this chapter up through Section 6−8 to analyze Herbie's data.

2. In Section 6−9 we discussed finding values of the constants A and B which resulted in the best fit of a "law" to a table of experimental data. Specifically, we considered the case where the experimental data consisted of a table of values of x and the corresponding values of y, and the "law" was $y = A + Bx$, where A and B were constants to be varied to give the best agreement between the "law" and the table of data. For the table below, what are the values of A and B giving the best agreement? Evaluate the formulae by hand, and show your work.

x	y
1	1.1
2	1.9
3	3.0

3. Make a worksheet like that shown in Fig. 6−4 that calculates the dependence of the total squared error on A and B. Add to the worksheet the evaluation of Eqs. (6−15) and (6−21) for $A_{min\ error}$ and $B_{min\ error}$. Show that as you change A, $B_{min\ error}$ changes, and for several specific values of A, run macro `\B` to verify that the bottom of the parabola occurs at the predicted value. Similarly, check that Eq. (6−15) for $A_{min\ error}$ accurately predicts the bottom of the parabola for several specific values of B. Finally, add formulae for evaluating Eqs. (6−24) and (6−26) to your worksheet and try to find values of A and B that give a smaller total squared error than that produced by the values given by these formulae.

4. Make a worksheet like that show in Fig. 6−4 that calculates the dependence of the total squared error on A. Evaluate the constants a, b, and c in Eq. (6−1) and add a plot of the corresponding parabola to the graph of the errors the macro calculates. The two plots should agree. Do they?

5. You borrow $50,000 at an interest rate of 12% per annum. If you repay the loan in monthly payments, what amount per month would you have to pay to just pay the loan off in twenty years? What would the payments be if the interest rate were 10% instead? In both cases, how much interest, total do you pay over the 20 year period? Use Quattro Pro to figure out the answer, and design the worksheet so that it is easy to change the interest rate and do the calculation over. Quattro Pro has two financial functions, `@PAYMT` and `@PMT`, which solve problems just such as this, but don't use them. Solve the problem "from scratch." Make a graph showing the amount remaining to be paid on the vertical axis and the month number on the horizontal.

   In case your financial skills are rusty, for each monthly payment you pay an amount of interest equal to one twelfth of the per annum interest rate times the amount of money still owed. Whatever amount is left over in the monthly payment after paying the interest is then applied to reduce the money you still owe. For example, at 12% per annum if the monthly payment were $700 your first payment would reduce the amount you owe by $200. The interest charged on your next payment would then be 1% of $49,800.

6. You want to use Quattro Pro to keep track of the fuel efficiency (miles per gallon) of your car. Your plan is to fill your tank every time you buy gas, and to write down on a scrap of paper the odometer reading and the number of gallons of gas you bought. When you get home, you will add these two data to a worksheet that you have set up. The idea is to put the odometer reading and the amount of gas bought into columns `A` and `B` respectively of the worksheet, and then to have the worksheet calculate the fuel efficiency in miles per gallon achieved during the time since the previous fill-up, putting the result in column `C`. Each row of the worksheet will correspond to a different fill-up. In column `D` you will put the cumulative fuel efficiency, that is the total number of miles driven since starting the procedure divided by the number of gallons of gas used.

   Set up a Quattro Pro worksheet to accomplish this goal. You should design it so that it will be as easy as possible to use. To add a new data point you should only have to start up Quattro Pro, load the worksheet file, add the two new numbers at the bottom of columns `A` and `B`, look at the result, save the worksheet back to the file, and exit Quattro Pro.

7. Consider the following function

$$f(x) = \frac{1}{e^{x/x_T} + 1}$$

   where x_T is a constant. Use Quattro Pro to graph the function for a specific value of x_T to be placed in cell `D3` of the worksheet. Plot 51 values of x ranging between −1 and +1 for the following values of x_t.

   a. 0.01

   b. 0.1

   c. 1.0

   Note: e^x in Quattro Pro is `@EXP(`*Cell address*`)`.

8. Use Quattro Pro to graph the function,

$$f(x) = \frac{\sin(20x)}{x}$$

   for about 50 values of x between –1 and +1. The function does not have a value for $x = 0$ because division by zero is strictly forbidden. (If your spreadsheet contains the value $x \approx 0$, you should be careful because of the division by zero problem.) From your graph, what value does f approach as $x \to 0$?

   Note: Quattro Pro can calculate trigonometric functions. For $\sin(x)$ use `@SIN(`*Cell address*`)`, where *Cell address* is the address of the cell containing the value of x for which you want the sine.

9. Consider the function, $f(x)$, given by

$$f(x) = (1 + x)^{\frac{1}{x}}$$

   The value of the function is not defined for $x = 0$ because the exponent involves division by x. Does the function approach a finite limit as $x \to 0$? Consider values of x both larger than and smaller than 0. What value does Quattro Pro give for $f(x)$ for $x = \pm 10^{-14}$, $\pm 10^{-15}$, $\pm 10^{-16}$, and $\pm 10^{-17}$? What's going on??

10. Consider the equation

$$e^x = x + 2$$

    Make a Quattro Pro worksheet which plots both sides of this equation on the same graph, and then use this worksheet to find the value of x which satisfies the equation accurate to four decimal places.

I suggest you set up the worksheet so that the list of x values is generated automatically by putting the low and high values of x in two cells, similar to the example worked out in Chapter 6 to plot the prediction of Eq. (4−1). That way, you can change the range of x values plotted easily. This equation is an example of a *transcendental equation*. The solution can not be expressed in terms of standard functions like powers of x, trigonometric functions, exponentials, or logarithms. The only way to solve such equations is to use a technique similar to the one you are to use here.

Note: e^x in Quattro Pro is `@EXP(`*Cell address*`)`.

11. The response of electronic circuits such as amplifiers depends on the frequency of the signal being processed. For example, audio amplifiers such as used in stereo systems are designed so that the amplification factor (the gain) is as constant as possible across the audio spectrum, but inevitably the gain falls off rapidly for signals with frequency higher than some value called the *corner frequency*. Often, the power gain, A_{power} depends on the frequency, f, according to

$$A_{power}(f) = \frac{f_0^2}{f_0^2 + f^2}$$

where f_0 is the corner frequency. Use Quattro Pro to plot $A_{power}(f)$ vs. f for the case $f_0 = 1000$ Hz, and for a range of frequencies between 10 and 100,000 Hz. Use 101 points, evenly spaced in frequency. Name this graph `Linear` so that you can generate a new graph without losing this one.

You should find that the interesting information is all scrunched up on the left side of the graph. For this reason, such data is often plotted on a *semi-logarithmic* graph. Each point plotted on a graph represents two numbers, the x value, and the y value. For the graph you just made (named `Linear`), the x value of a given point was the frequency corresponding to that point, and the y was the power gain at that frequency. On a semi-logarithmic graph of $A_{power}(f)$ vs. f, the y value is still $A_{power}(f)$, but the x value is $\log(f)$, rather than just f. Quattro Pro has the capability of generating such plots. Choose the `Mode` item from the `X-Axis` sub-menu of the `Graph` menu. If you make a semi-log plot of the gain with evenly spaced frequencies, such as you just generated, you will find a dearth of points on the left hand side. It is better to use an uneven spacing in this case. I suggest starting with the lowest frequency, 10 Hz in this case, and then making the next frequency some factor, say β, times that, and then the next β times the previous, and so forth. Choose β so that after 101 points (or however many points you decided to use) you have just gotten to the highest frequency, 100,000 Hz in this case. Using this set of frequencies, plot the response on a semi-logarithmic graph. Compare this graph with the one you generated previously using a linear (`Normal` on Quattro Pro) x-axis scale. Notice the difference in information content.

As a side note, the term *semi-logarithmic graph* refers to a plot in which the logarithm of one quantity is plotted vs the other. If we had plotted $\log(A_{power})$ vs f, instead of *vice versa*, that would also have been called a semi-logarithmic graph. Sometimes it is useful to make a fully logarithmic or *log-log* graph by plotting the logarithm of both quantities. We could have made a log-log graph of $A_{power}(f)$ by plotting $\log(A_{power})$ on the y axis, and $\log(f)$ on the x. In fact, gains are usually plotted as log-log plots, and the quantity $10\log(A_{power})$ is called a *decibel*, or simply a *db*. Specification sheets for amplifiers often give the gain variation expressed in db over some frequency band.

Note: Figuring out what value of β to use is a significant part of this problem. There are several ways to approach it.

7. OTHER QUATTRO PRO TOPICS

Quattro Pro has a large number of capabilities, and we will only discuss a few here. As you use the program throughout the semester, I encourage you to explore a little when you have some free time.

7.1 Printing

Quattro Pro allows you to print out parts of your worksheet. There are two principal modes of printing, non-graphics and graphics. Non-graphics mode lets you print out text from your worksheet. In this mode, the ASCII equivalents for every text character in the portion of the worksheet to be printed are sent to the printer. Graphs and special effects such as lines, boxes, and shading cannot be printed in this way. To print out such things graphics mode is needed. In graphics mode the page to be printed is painted onto the paper by the printer. The page is divided into a large number of small rectangles (the number depends on the resolution of the printer, but it is typically more than 10,000 per square inch), and Quattro Pro sends signals to the printer telling it for every one of these rectangles whether or not to put a small dot in it. That's a lot of information to send, and you will find that printing in graphics mode is slower than non-graphics mode.

To print a portion of a worksheet, not a graph, in either mode choose `Print` from the Pull Down Menu Bar. If the defaults haven't been messed up, the menu should show that the `Destination` is set to `Printer`, and the `Format` is set to `As Displayed`. `Printer` is the correct destination if you want to print part of the worksheet in non-graphics mode (the recommended mode for most cases). Change it to `Graphics Printer` for graphics mode (recommended only if you have some of the fancy line drawing and shading effects, or if if you have overlaid a graph over part of the worksheet and you want the printed page to match the display on the computer screen).

Specify what part of the worksheet to print using the `Block` item in the `Print` menu. The `Layout` choice of the `Print` menu allows you to adjust things like margins which you should not need to change. Two items that might be of interest on this sub-menu are the `Header` and `Footer` items. These allow you to specify things to be printed at the top and bottom of every page. If you are turning in some printout as part of an assignment, you might want to put your name, maybe the date or assignment number, and the page number (use the pound sign, #, for this) in one of them. The `Format` item of the `Print` menu allows you to specify whether for cells with formulas in them the values of the formulas (`As Displayed`) or the formulas themselves (`Cell-Formulas`) will be printed.

To do a really nifty job of printing Quattro Pro needs to know where the printer is on the page of paper. Use the `Adjust Printer` item of the `Print` menu to keep it informed. When you first start adjust the printer to the top of the page manually, and when it's ready choose the `Align` option to tell Quattro Pro that it's at the top of the page. After that, don't adjust the printer itself at all; instead use the `Skip Line` and `Form Feed` items to move the paper through the printer. The first item does just what it says, it causes the printer to skip a line. The second causes the printer to advance the paper until it gets to the top of the next sheet.

When you have finally gotten everything ready, choose the `Spreadsheet Print` item of the `Print` menu and after a short delay, the printer should start to do its thing.

If you want to print out a graph, not part of a worksheet, the procedure is different. Instead, right off the bat choose `Graph Print` from the `Print` menu. The sub-menu that appears should indicate that the `Destination` is the `Graphics Printer`. If not, change it. The `Layout` menu contains placement information which you should seldom need to worry about. If the graph you want to print is the current graph (the one that appears when you press **F10**), just select `Go` from this sub-menu. If instead you want to print out a previous graph that you named, use the `Name` item to choose it before selecting `Go`. If you forgot to name the graph before you changed the settings, tough bananas! Notice that in this mode, the

`Header` and `Footer` information is not printed.

Let's look at some of the other possible destinations for both non-graphics and graphics mode printing. In non-graphics mode there are two groups of destinations, labeled `Draft Mode Printing`, and `Final Quality Printing`. The difference between these two groups lies in the way the spreadsheet block is printed. In `Draft Mode`, to print a character Quattro Pro simply sends the ASCII equivalent of the character to the printer. `Final Quality Mode` works the same way as graphics mode. In `Final Quality Mode`, Quattro Pro "paints" each character by telling the printer what dots to print in order to form the character. Because of the quantity of data that must be sent to the printer, `Final Quality Mode` is considerably slower than `Draft Mode`, but with it you can use different fonts, and print lines and other objects.

In the `Draft Mode Printing` group the destinations are `Printer` and `File`. We have already discussed the first. It prints the specified block directly on the printer. Choosing the `File` destination saves what would have been sent to the printer in a file. This could be printed later from DOS by using either the `PRINT` command, or using `COPY/B` *filename* `PRN`, but I recommend the `COPY` form because `PRINT` may space to the top of the next page at inopportune times, whereas the `COPY` command will not cause the printer to do anything that's not in the file. The `/B` after the `COPY` tells DOS that the file is to be copied as is, with no substitutions of any kind. You can often get by without the `/B`, but occasionally, the file will have some special character in it that DOS will interpret as an end of file or some such mark. In the `Final Quality Mode` group the destinations are `Binary File`, `Graphics Printer`, and `Screen Preview`. The first works just like the `File` destination in the `Draft Mode` group. It sends the information to a file rather than the printer in a form that can be printed later by simply sending the file to the printer. The second prints the block directly on the printer. The third tries to print the spreadsheet block on the computer screen rather than the printer. It helps to save paper by letting you check to see if what will print is what you had in mind. The resolution of the screen is such that you will have difficulty reading the writing, but you should be able to recognize what's what. At the top of the screen is a menu bar similar to the one you see when you are back in the worksheet. You can use the menu bar to zoom in or out on a part of the screen, and you can use the arrow keys to move around. No matter what destination you choose, you must still select `Spreadsheet Print` from the main `Print` menu to print the block.

As discussed above, choosing the `Graph Print` option from the main `Print` menu allows you to print graphs. There are three destinations: `File`, `Graph Printer`, and `Screen Preview`. The first saves the information in a file, rather than printing it immediately, the second prints it immediately (we've already talked about this one), and the third does the same thing as the `Screen Preview` destination in the `Final Quality Mode` group as discussed above.

7.2 @ Functions

Quattro Pro has a large number of built-in functions. The names of these functions always begin with an "at" sign, `@`. I list in Table 7−1 those I expect to be most likely to be useful. Consult the Help facility or the manuals to learn about the others.

SOME USEFUL @ FUNCTIONS

Function	Returns
`@COS(`*X*`)`	Cosine of angle *X* expressed in radians
`@SIN(`*X*`)`	Sine of angle *X* expressed in radians
`@TAN(`*X*`)`	Tangent of angle *X* expressed in radians
`@ACOS(`*X*`)`	Arc cosine in radians of *X*

SOME USEFUL @ FUNCTIONS

Function	Returns
@ASIN(*X*)	Arc sine in radians of *X*
@ATAN(*X*)	Arc tangent in radians of *X*
@ATAN2(*X*,*Y*)	Arc tangent in radians of *Y*/*X*
@DEGREES(*X*)	Converts *X* radians to degrees
@RADIANS(*X*)	Converts *X* degrees to radians
@PI	Value of π
@EXP(*X*)	e^X
@LN(*X*)	Logarithm base *e* of *X*
@LOG(*X*)	Logarithm base 10 of *X*
@SQRT(*X*)	Square root of *X* (always positive)
@ABS(*X*)	Absolute value of *X*
@INT(*X*)	Integer portion of *X*
@MOD(*X*, *Y*)	Modulus; the remainder of *X*/*Y*
@ROUND(*X*,*N*)	*X* rounded to *N* digits (up to 15)
@RAND	Returns a random number between 0 and 1
@AVG(*Range*)	Average of values in *Range*
@COUNT(*Range*)	Number of non-blank cells in *Range*
@SUM(*Range*)	Sum of values in *Range*
@MAX(*Range*)	Largest value in *Range*
@MIN(*Range*)	Smallest value in *Range*
@IF(*Cond*,*Tru_Exp*,*Fls_Exp*)	Value of *Tru_Exp* if *Cond* is true; Value of *Fls_Exp* if *Cond* is false
@NOW	Current date and time
@TODAY	Current date
@HLOOKUP(*X*,*Block*,*Row*)	Find *X* in the first row of *Block*. If *X* is not found and *X* is a number then find the largest value which is still less than *X*. If *Row* is zero, returns contents of the found cell. If *Row* is greater than zero returns the contents of the cell *Row* rows below the found cell.
@VLOOKUP(*X*,*Block*,*Column*)	Same as @HLOOKUP except search first column of *Block* instead of first row.

SOME USEFUL @ FUNCTIONS	
Function	**Returns**
`@SUMPRODUCT(`*Block1*`,`*Block2*`)`	Multiplies the corresponding cells of *Block1* and *Block2* and then adds these products. The blocks must have the same size.

TABLE 7−1. Some Useful @ functions

A few of the entries in the table require further comment. As you may recall, the arc tangent function returns values with a maximum range of π. The Quattro Pro `@ATAN` function returns a value that is between $-\frac{\pi}{2}$ and $+\frac{\pi}{2}$. The `@ATAN2(`*X*`,`*Y*`)` function returns a value that ranges between $-\pi$ and $+\pi$. The value returned is the angle that a line from the origin to the point (*X*, *Y*) makes with the positive x-axis.

The `@IF` function provides conditional evaluation. The first, *Cond*, argument is some condition such as the value of some cell is less than 3, or the value of some cell is not equal to the value of some other cell. The second and third arguments are formulas, and `@IF` returns the value of the first if the condition is true, and the value of the other if it is not.

Quattro Pro uses *logical operators* to specify conditions in `@IF` functions. The operators it recognizes are shown in Table 7−2.

LOGICAL OPERATORS	
Operator	**Meaning**
`<`	Less than
`>`	Greater than
`<=`	Less than or equal to
`>=`	Greater than or equal to
`=`	Equal to
`<>`	Not equal to
`#NOT#`	Logical negation
`#AND#`	Logical AND
`#OR#`	Logical OR

TABLE 7−2. Quattro Pro's logical operators

Here are some example of the use of `@IF`, and of the logical operators.

```
@IF(E3>F6,E17-A12,@SUM(A3..A12))
@IF(E3>F6#AND#F4=B2,26,E3)
@IF(#NOT#(E3=F6),1,2)
```

Note the use of `#NOT#` in the last example. The condition `E3#NOT#=F6` is not legal. Also note that the condition in the last example is equivalent to `E3<>F6`.

Quattro Pro stores date and time information in what Borland calls *serial number* form. The date is stored as an integer equal to the number of days since December 30, 1899. I'm not sure how the time is stored, but I guess it's the number of seconds since midnight. Both the `@NOW` and `@TODAY` functions return

a value in this form. Since that's not very useful to humans, you will probably want to change the way it's displayed. This can be done by changing the display format, and that is discussed in the next section.

7.3 Changing the Format of Your Worksheet

Use items from the `Style` Pull Down Menu to change how numbers and other data are displayed in your spreadsheet. Use the `Numeric Format` item to change the way numbers are displayed in cells. Choosing this item results in a sub-menu with a number of items on it, most of which should be self-explanatory. Of the number formats, the most useful to engineers will be the `Fixed`, `Scientific`, and `General` formats. The `General` item displays numbers that aren't too large or small in decimal notation, and the rest in scientific. You can specify how many digits are to be displayed. Quattro Pro keeps about 15 digits of accuracy, no matter what you specify for display. If a cell contains a number that represents a date or a time, and you want it displayed in a sane notation, choose the `Date` option in the `Numeric Format` sub-menu.

Sometimes the column is not wide enough to display a number. In that case the cell on the display is filled with asterisks. You can increase the width of columns using either the `Column Width` or the `Block Widths` items on the `Style` menu. You can hide an entire column with the `Hide Column` item. I think this was originally intended for financial spreadsheets containing confidential information, but I find it useful when I require one or more columns that contain intermediate values which are of no other use.

As I mentioned in Chapter 6, Quattro Pro allows you to draw lines on your worksheet and to put boxes around blocks of cells, and I've made use of this feature in many of the figures in these notes. This capability is wholly cosmetic, but it can be useful to focus attention to the main results of the worksheet, and away from uninteresting intermediate results or input data. There is also a shading facility that can be used similarly.

7.4 Entering Data from a File, Importing

Sometimes data you generate either is or can be put automatically into a file. If you want to analyze this data using a spreadsheet program, then it is necessary to enter these data into a worksheet somehow. One option is to print the file out, and then enter the data manually using the keyboard, just as you entered Herbie's data. Particularly for longer data tables, this procedure is unpleasant, and prone to errors. Most spreadsheet programs have what is termed an *Import* facility which allow you to take data from files and enter them automatically into a worksheet. Here's what Quattro Pro can do, and how you get it to do it.

Besides being able to read worksheet files produced by several other spreadsheet programs such as Lotus 123, Quattro Pro can also read files containing tables of numbers written in ASCII. If you've forgotten what ASCII is, see the discussion in Section 1.8, and Table 1−1. Suppose, for example, you have a table of data similar to Table 4−3, containing Herbie's raw voltage-current data. The computer has to know where the entry containing the voltage value ends and where the entry with the current value begins. A common technique for doing so which Quattro Pro *does not* support involves leaving space or tab characters between columns. Instead, Quattro Pro offers three methods, of which I'll discuss two.

The first method assumes that the data is saved in the file in aligned columns, with each entry in a column just below the previous entry. For example, a file might be arranged so that all entries corresponding to the first column can be found in the first 9 characters of a line, all entries corresponding to the second column in the next 9 characters, and so forth. An example of such a file is shown in Fig. 7−1. The first line is not supposed to be a part of the file; I put it there to help locate the position of each character in the line. In this case entries belonging to the first column can all be found in the first nine characters of a line, and those belonging to the second in the next nine characters.

To import such data contained in a file, first place the highlight at the head of an unused block of cells in which you wish to put the contents of the file. Then choose `Tools` from the Pull Down Menu Bar, and

```
12345678901234567890
-0.020   0.0003
 9.97     .0099
19.9      0.0203
  29.9   0.0299
39.9      .0398
```

Figure 7−1. Table of data formatted for importation as an ASCII text file.

then choose `Import` to produce a sub-menu listing the three options for importing data. For the assumed form of the data in our file, choose the first item, `ASCII Text File`. This choice will cause each line in the file to be read and placed in consecutive cells in the column pointed out by the highlight. No matter what the line contains, it is treated as a string of characters (a label in Quattro Pro jargon). A small window will appear in which the computer asks you to `Enter name of file to import:`. In response you would enter the name of the file containing the data. The contents of the file should appear just as in the file, starting at the place you pointed out with the highlight just before choosing the `Tools` menu. For example, for the file shown in Fig. 7−1, if you had initially moved the highlight to cell `A3`, cells `A3..A7` would each contain one line from the table.

At this point, the information is in the worksheet, but not in a very useful form. We need to separate out the data in the first column from that in the second, and put the two sets of data into separate columns of the worksheet. To do so, position the highlight at the top of the column containing the information just imported, and then choose the `Parse` item from the `Tools` menu. A sub-menu will appear which is used to tell the computer where the raw data is located, how to separate the data, and where the data is to be put after it has been separated out. Telling the computer where the raw data is, and where to put the processed result is easy. Choose `Input` from the `Parse` sub-menu to specify where the raw data is located in the worksheet, and choose `Output` to specify where the separated results are to be placed.

Telling Quattro Pro how to separate the raw data is a little more complicated. Such an operation is called *parsing*. You give Quattro Pro the information it needs by creating a *format line* just above the cell containing the first item or raw data to be parsed. A format line consists of a string of symbols with special meaning to Quattro Pro. These symbols are in Table 7−3.

PARSING SYMBOLS

Symbol	Meaning
`|`	The remaining characters in this cell are a format line. The `|` character must start every format line.
`V`	This is the start of a value entry
`L`	This is the start of a label entry
`T`	This is the start of a time entry
`D`	This is the start of a date entry
`>`	This continues the entry
`*`	This may be a blank space, or it may be part of an entry.
`S`	Ignore characters in this position

TABLE 7−3. Table of special symbols for format lines.

You can either create your own format line from scratch, or you can have Quattro Pro guess for you, and then patch it up using standard editing procedure. I suggest you have Quattro Pro guess first for you. To do

so, be sure the highlight covers the first item in the raw data column, and then choose `Create` from the `Parse` sub-menu.

Fig. 7−2 shows the result of telling Quattro Pro to guess a format line for the data in Fig. 7−1. (As before, the first line is something I added to help you count characters in lines.)

```
12345678901234567890
V>>>>>***V>>>>>
-0.020   0.0003
 9.97     .0099
19.9      0.0203
  29.9   0.0299
39.9      .0398
```

Figure 7−2. Format line guessed by Quattro Pro for the data in Fig. 7−1.

The format line specifies that the first nine characters of a line are to be treated as parts of the data to be placed with the first set, and the remaining characters with the second set. There must be some distinction between the > and the * characters, but I don't understand what it is. In our case, Quattro Pro's guess was a good one and no further changes are needed. If changes were required, you would use the `Edit` item from the `Parse` sub-menu. In some cases it will not be possible to create one format line which will handle all lines in your file correctly. For this reason multiple format lines are allowed. Just position the highlight on the first item for which the existing format line won't work and choose `Create` again. Quattro Pro will make a guess based on this line and insert it as an additional row just before the line. You can do that as many times as required.

Once you have suitable format lines, you should then tell Quattro Pro where the raw data (including the format lines) is located in the worksheet, and where you want to put the parsed result. Do so with the `Input` and `Output` options of the `Parse` sub-menu. When you finally have everything set and are ready to hit the juice choose `Go` from the `Parse` sub-menu. Unexpected things can sometimes happen when parsing data, so I suggest you place the parsed result in some unused portion of the worksheet, and then `Move` it back to where you want it when everything looks good.

A less general, but often easier, way to import data is to separate the column entries with commas. For example, the data in Fig. 7−1 could be represented in this format as in Fig. 7−3 (again with an ersatz first line).

```
1234567890123456789
-0.020,   0.0003
 9.97,.0099
19.9,  0.0203
  29.9,  0.0299
39.9,.0398
```

Figure 7−3. Data format using commas for column separators

In this case, we do not have to make sure that an entry is in the right block of character positions in the line. Instead, the computer uses the presence of commas to denote where one entry stops and the next one starts.

To use this option, position the highlight at the start of the block in your worksheet in which you wish to place the imported data. Then choose the `Commas Only` from the `Import` sub-menu. Quattro Pro will put the entries in the first position (before the first comma) in the column under the highlight, the entries in the second position (after the first comma and before the second if there is one) in the next column to the right, and so forth.

7.5 Recalculation

Whenever you change the value of a cell, Quattro Pro automatically recalculates the values of all cells in the worksheet which might change as a result. For very long, complicated worksheets, it may take quite a while to recalculate every such cell, however. That can be annoying when you are trying to change values in several cells and won't care what the calculated values are until after you have finished entering all your changes. Quattro Pro gives you some control over how and when recalculation occurs, as well as some other aspects of recalculation.

Choose `Recalculation` from the `Options` Pull Down Menu. Four items plus `Quit` will be on the resulting sub-menu. The first, `Mode`, lets you specify whether or not Quattro Pro will do a recalculation every time a worksheet cell value is changed. To turn off recalculation, choose `Manual` from the `Mode` sub-menu. In the manual mode when you make a change to the contents of a cell, Quattro Pro only evaluates the contents of that cell, even though the contents of cells which refer to the changed cell will then probably be incorrect. You can tell Quattro Pro to recalculate all possibly wrong cells by pressing **F9**. If the worksheet needs recalculation, the word `CALC` will appear at the bottom right of the screen in the status line.

There are two modes in which recalculation is turned on. In one, `Automatic`, Quattro Pro locks up until the recalculation is finished, whereas in the other, `Background`, you are able to enter numbers or do other worksheet operations while the recalculation is in progress. This `Background` mode is the default, and I recommend it. In this mode if you have a long worksheet don't panic if you make a change and nothing happens right away. You should panic if enough time passes to update the entire worksheet several times and still nothing happens!

7.6 Sorting Data

It is frequently desired that a set of data be sorted in some way. For example, you might want to alphabetize a list of names. A more familiar example would be Herbie's voltage-current data. Herbie wrote the data into his lab notebook in the order the data were taken, and we put them in the worksheet in the same order. You might want to sort the data into increasing order of current, for example. In fact, if when plotting the data we had chosen to connect Herbie's experimental points with lines instead of just plotting symbols, we would have encountered such a need. The plotting program draws a line from one point to the next in the order the points appear, so instead of getting a single, more-or-less straight line we would have gotten lines going all over the place. This was a mud puddle I blithely skirted by choosing to plot just symbols, but which some of the more adventurous of you may have fallen into if you tried generating a line-connected plot of the data. (Why didn't I fall into the mud puddle when I graphed the "theoretical" prediction using lines to connect the calculated points? Or did I?)

Probably the best solution would be to sort Herbie's points into an order in which the current increases monotonically. To do so use the `Sort` option from the `Database` sub-menu. The resulting sub-menu contains an item allowing you to specify what data to sort, `Block`, and items specifying how to sort it, `1st Key`, `2nd Key`, etc. When specifying the block to be sorted, be sure to include all the columns containing connected data. When Quattro Pro is moving around Herbie's current entries, for example, it should move the corresponding voltage entries around with them so that they stay connected properly. If the voltage entries are in the range `A3 ..A13`, and the current entries in `B3..B13`, we would specify the block `A3..B13` using the `Block` item.

The `1st Key` item allows you to specify what column of data to sort on. In our case, we would choose the column containing the current, and would point out the range `B3..B13`. The `2nd Key` item on the `Sort` sub-menu allows you to specify another sorting criterion in the event two items in the `1st Key` column are the same. This option seems to be intended mostly for sorting things like lists of names. In such a case, the `1st Key` might be the column containing last names, the `2nd Key` might be the column containing first names, and the `3rd Key` might be the column with middle name or initial. You are

given the option of sorting either forwards (`Ascending` order) or backwards (`Descending` order).

When everything is all set, choose `Go` from the `Sort` sub-menu and the sort should occur. I suggest saving the worksheet into a file before doing a sort. Things can go wrong.

7.7 Miscellaneous Other Capabilities

Quattro Pro offers a number of other capabilities which I have not discussed. I will mention only a few of them briefly here.

In the `Macro` sub-menu of the `Tools` menu there are a number of features related to macros. `Record` provides an easy way to create some macros. When `Record` is on whatever you do with the keyboard is recorded in macro language and saved in a block of the worksheet. If you turn `Record` mode on and then issue requests to Quattro Pro which cause it to carry out some desired task, and then turn `Record` off, you will have produced a macro which can be executed to accomplish the same task automatically. This feature seems to be most useful for applications in which it is desirable to automate some common tasks. I have not found it very useful because I usually use Quattro Pro for different purposes each time I use it. It seems that the things I usually want to do with macros cannot be done without explicitly writing the macro. I occasionally use the feature as a kind of Help facility when I want to know how I tell Quattro Pro in macro language to do something I know how to do from the keyboard.

The `Tools` menu also contains items which allow you to combine two or more worksheets, or to take part of one worksheet and insert it into another. Quattro Pro also has the capability of referring to the contents of cells which are contained in other worksheets which are not in memory, but were saved in files previously. These features seem to be useful primarily to "professional" users, perhaps maintaining a complicated accounting system.

Of more interest to engineering and science applications are two of the features listed in the `Advanced Math` item of the `Tools` menu. The `Invert` and `Multiply` items are used to invert and multiply matrices. To use them place the matrix or matrices to be operated on in blocks in the worksheet, and supply the addresses of these blocks and a block in which to place the result when requested by Quattro Pro. I have not explored the capabilities of the `Regression` and `Optimization` items. They are intended primarily for business applications, but they may be useful to engineers as well. I believe that business types and statisticians call the process of fitting a straight line to a set of data points *performing a linear regression analysis*, so I suspect we could have done the same thing we did earlier with Herbie's data using stuff in the `Regression` menu. I haven't tried it, though.

The `What-If` option allows you to do something similar to what we did with Herbie's data when we generated the total squared error for several values of the parameters, *A* and *B*. In fact, you can generate the same results using this feature as we did by writing a macro.

I have not mentioned the windowing capabilities of Quattro Pro at all. You can have two or more worksheets open at one time and move back and forth between them using the `Window` item in the Pull Down Menu Bar. You can also put two different parts of the same worksheet on the screen at the same time in different windows. To do so, choose `Options` from the `Window` menu. Use **F6** to skip back and forth between windows.

You can use the `Annotate` item in the `Graph` menu to put all sorts of special notations on graphs. You can even use it to make drawings.

7.8 Exercises

For all problems requiring the use of Quattro Pro, when you are satisfied with your worksheet save it to a file on a floppy and turn the floppy in as part of the homework assignment. You should also include a clear written description of what you did and why. That can be handed in on a separate piece of paper, or it can be included in the worksheet as a comment. You can put more than one worksheet file on a floppy, but make sure you give each a name that makes it obvious which file goes with which problem.

1. Using the worksheet you saved, print out a graph showing Herbie's experimental data plotted with X's, and the prediction of Eq. (4−1), plotted as a continuous line.

2. Create a worksheet that estimates logarithms, base 10, for numbers between 1 and 10. Here's the idea for how to do it. Back in the old'n days before calculators and computers, this is pretty much how we had to do it (only by hand, not with Quattro Pro, and after walking 3 miles through 27 foot snow drifts to get to school). Table 7−4 gives the base 10 logarithms of selected numbers between 1 and 10.

LOGARITHMS	
x	**log(x)**
1.0	0.0000
2.0	0.3010
3.0	0.4771
4.0	0.6021
5.0	0.6990
6.0	0.7782
7.0	0.8451
8.0	0.9031
9.0	0.9542

TABLE 7−4. Table of base 10 logarithms

To calculate the logarithm of a number, say x, find the largest number in the first column of the table which is smaller than x, and interpolate between this point and the next in the table. For example, if x were 4.27, then we would estimate $\log(x)$ as the value 0.27 of the way between log(4) and log(5). In your worksheet, I should be able to specify the number I want to take the logarithm of by entering it in one cell and the interpolated result should appear in another cell. Label these cells so the worksheet is convenient to use. In order to see how well the method works, use the Quattro Pro @ function, `@LOG(X)`, to produce the exact (more or less) value, and display the difference, both as a number and as a percentage, between the interpolated and exact results in other cells of the worksheet. How could you make your method work for values of x outside the range of 1 to 10?

HINT: Check out the function `VLOOKUP`.

3. You will be given a file containing in ASCII a table of values of a function, $f(x)$, vs. x. Use Quattro Pro to plot it. Connect the points by lines, and do not plot a symbol at each point.

8. VOLTAGE AND CURRENT

8.1 Introduction

Much of what electrical engineers do involves either trying to figure out how the current is flowing in some circuit, or trying to design a circuit in which the current flows in some desired way. In this chapter I will discuss what an electrical current is, and what forces influence the flow of these currents. As I'm sure most of you know, matter is composed of atoms, which consist of a bunch of protons and neutrons bound together in the nucleus by mysterious forces, and a number of electrons equal to the number of protons whizzing around the nucleus and bound by a not-so-mysterious force. The type of atom is determined by the number of protons in the nucleus. Each type of atom is called an *element*. Elements can combine either with themselves or with other elements to form all the materials we find in the world.

The force binding the electrons to the nuclei is pretty well understood. Electrons and protons have a property called *charge*. There are two types of charge, called positive and negative, and electrons have one type, protons the other. Tradition has it that the charge on a proton is of the positive variety, and the charge on an electron negative. That definition of what is positive and what is negative was made before people knew anything about atoms, and it was completely arbitrary. It doesn't matter in principle which sign of charge we associate with protons and electrons as long as they are opposite and as long as we are consistent about it. In hind sight, some confusion would have been saved if the opposite convention had been chosen, but it wasn't and we are stuck with it.

Most objects in the every-day world have very nearly equal numbers of electrons and protons, but it is possible to add or remove a significant number of electrons from most things, making them have a net negative or positive charge. It is found experimentally that objects with net positive charge repel other positively charged objects, and attract negatively charged objects; conversely, negative objects repel each other and attract those with positive charge. This force of attraction or repulsion is called the *electric* force and it is very strong. The attraction of unlike charges explains the binding of the electrons to the nucleus, but the even stronger binding of the like-charged protons together to form the nucleus is mysterious. Apparently there is some short-range force which acts between protons and which is strong enough to overcome the strong repulsion of the protons.

The best discussion of the electric and other forces I've found is by Feynman.[1]

> Consider a force like gravitation which varies predominantly inversely as the square of the distance, but which is about a *billion-billion-billion-billion* times stronger. And with another difference. There are two kinds of "matter," which we can call positive and negative. Like kinds repel and unlike kinds attract—unlike gravity where there is only attraction. What would happen?
>
> A bunch of positives would repel with an enormous force and spread out in all directions. A bunch of negatives would do the same. But an evenly mixed bunch of positives and negatives would do something completely different. The opposite pieces would be pulled together by the enormous attractions. The net result would be that the terrific forces would balance themselves out, almost perfectly, by forming tight, fine mixtures of the positive and the negative, and between two separate bunches of such mixtures there would be practically no attraction or repulsion at all.
>
> There is such a force; the electrical force. And all matter is a mixture of positive protons and negative electrons which are attracting and repelling with this great force. So perfect is the balance, however, that when you stand near someone else you don't feel any force at all. If there were even a little bit of unbalance you would know it. If you were standing at arm's length from someone and each of you had *one percent* more electrons than protons the repelling force would be incredible. How great? Enough to lift the Empire State Building? No! To lift Mount Everest? No! The

1. *The Feynman Lectures on Physics*, R.P. Feynman, R.B. Leighton, and M. Sands, (Addison-Wesley, Reading, 1964).

repulsion would be enough to lift a "weight" equal to that of the entire earth!

With such enormous forces so perfectly balanced in this intimate mixture, it is not hard to understand that matter, trying to keep its positive and negative charges in the finest balance, can have a great stiffness and strength. The Empire State Building, for example, swings only eight feet in the wind because the electrical forces hold every electron and proton more or less in its proper place. On the other hand, if we look at matter on a scale small enough that we see only a few atoms, any small piece will not, usually, have an equal number of positive and negative charges, and so there will be strong residual electrical forces. Even when there are equal numbers of both charges in two neighboring small pieces, there may still be large net electrical forces because the forces between individual charges vary inversely as the square of the distance. A net force can arise if a negative charge of one piece is closer to the positive than to the negative charges of the other piece. The attractive forces can then be larger than the repulsive ones and there can be a net attraction between two small pieces with no excess charges. The forces that hold the atoms together, and the chemical forces that hold molecules together, are really electrical forces acting in regions where the balance of charge is not perfect, or where the distances are very small.

You know, of course, that atoms are made with positive protons in the nucleus and with electrons outside. You may ask: "If this electrical force is so terrific, why don't the protons and electrons just get on top of each other? If they want to be in an intimate mixture, why isn't it still more intimate?" The answer has to do with the quantum effects. If we try to confine our electrons in a region that is very close to the protons, then according to the uncertainty principle they must have some mean square momentum which is larger the more we try to confine them. It is this motion, required by the laws of quantum mechanics, that keeps the electrical attraction from bringing the charges any closer together.

There is another question: "What holds the nucleus together"? In a nucleus there are several protons, all of which are positive. Why don't they push themselves apart? It turns out that in nuclei there are, in addition to electrical forces, nonelectrical forces, called nuclear forces, which are greater than the electrical forces and which are able to hold the protons together in spite of the electrical repulsion. The nuclear forces, however, have a short range—their force falls off much more rapidly than $1/r^2$. And this has an important consequence. If a nucleus has too many protons in it, it gets too big, and it will not stay together. An example is uranium, with 92 protons. The nuclear forces act mainly between each proton (or neutron) and its nearest neighbor, while the electrical forces act over larger distances, giving a repulsion between each proton and all of the others in the nucleus. The more protons in a nucleus, the stronger is the electrical repulsion, until, as in the case of uranium, the balance is so delicate that the nucleus is almost ready to fly apart from the repulsive electrical force. If such a nucleus is just "tapped" lightly (as can be done by sending in a slow neutron), it breaks into two pieces, each with positive charge, and these pieces fly apart by electrical repulsion. The energy which is liberated is the energy of the atomic bomb. This energy is usually called "nuclear" energy, but it is really "electrical" energy released when electrical forces have overcome the attractive nuclear forces.

We may ask, finally, what holds a negatively charged electron together (since it has no nuclear forces). If an electron is all made of one kind of substance, each part should repel the other parts. Why, then, doesn't it fly apart? But does the electron have "parts"? Perhaps we should say that the electron is just a point and that electrical forces only act between *different* point charges, so that the electron does not act upon itself. Perhaps. All we can say is that the question of what holds the electron together has produced many difficulties in the attempts to form a complete theory of electromagnetism. The question has never been answered.

You may be interested in knowing that his investigation of this question of what holds an electron together was at the base of the work for which Feynman received the Nobel prize in physics in 1965, three years after he wrote the material I just quoted.

It is the complicated interaction of electrical forces and quantum-mechanical effects which is responsible for the amazing variety of characteristics and behaviors of the different substances in the world. Here I want only to discuss a few characteristics relevant to the flow of electrical currents. In some materials,

called *conductors*, the electrons are bound only very loosely to the individual atoms, and they are easy to move around—easy that is if you don't disturb the delicate charge balance that keeps the electrical forces in check. For example, you can take some electrons off of a conductor pretty easily, but only if at the same time you put an equal number of other electrons back on it in order to maintain charge neutrality. In other materials the electrons are much more tightly bound, and it is difficult and often catastrophic to move them around. Such materials are called *insulators*. Air, for example, is a pretty good insulator. Things like lightning happen if you try too hard to move the charges around. In a typical thunderstorm the bottom of the cloud is charged negatively, and the surface of the Earth just under it positively. As a result there is an electrical force pushing the electrons in the molecules of the atmosphere toward the ground. Little happens until this force gets large enough to be irresistible, and then **KABOOM!**

8.2 Electrical Currents: Positive or Negative Carriers?

The electrical breakdown of insulators is a subject of considerable interest to me, but I'll not consider it further here. Instead, I want to discuss how charge flows in conductors. Consider two analogous situations, as shown in Fig. 8−1. In one case we have two pools of water connected by a pipe, and in the other we have two conducting spheres connected by a conductor.

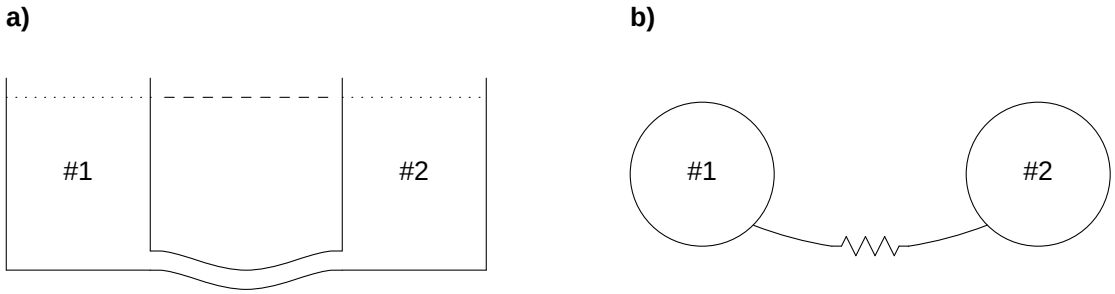

Figure 8−1. Drawing showing two analogous situations: a) two pools of water connected by a pipe, and b) two conducting spheres connected by a conductor.

If the two pools have the same amount of water in them and are at the same height, as shown in the drawing, not much happens. If on the other hand, more water is somehow put into pool #1 so that the top of the water in 1 is higher than in 2, water will flow from 1 to 2 until the top of the water is at the same height in the two pools. Similarly, if the amount of charge on the two spheres is the same and there are no other charges around, not much happens, but if I put more charge on sphere #1 than on #2 charge will flow in the conductor from #1 to #2 until the amount of charge on each one is the same.

I'll draw heavily on this analogy to explain how current flows in electrical circuits. You may get the impression, based on this discussion, that the whole subject is on a pretty shaky foundation, but that's not true. Everything can be derived with full mathematical rigor from basic physical laws governing the electric and magnetic fields, and the interaction of these fields with charged objects. I've chosen not to present the subject this way because doing so would lead me too far astray. Also, it would require some familiarity with the integral and differential calculus, and many of you may be seeing these ideas for the first time just this semester. Besides, the presentation I give here should give you more intuition about what would happen in the circuit than would a mathematically rigorous presentation, and both points of view are useful.

The analogy between the flow of water and charge is a pretty good one, except for one thing—there's only one kind of water but there are two kinds of charge, positive and negative. For example, if we put more water in pool #1 than #2 only one thing could happen—water would flow from 1 to 2. If on the other hand, sphere #1 had more positive charge than #2 two things could happen:

1. Positive charge could flow from sphere #1 to #2, or
2. Negative charge could flow from sphere #2 to #1.

In the first case, the amount of positive charge on sphere #1 would decrease, and the charge of #2 would increase. In the second case, negative charge would flow onto sphere #1, canceling some of the positive charge there and effectively reducing the charge of sphere #1. Also negative charge would flow off of #2, removing some of the negative charge that had been canceling the native positive charge on the sphere and effectively increasing the net charge of sphere #2.

Both possibilities would reduce the charge on #1 and increase it on #2, and a current of charge would flow in both cases until the net charge on both spheres was the same. It would be very difficult to determine which was occurring. In fact, the flow of charge was observed experimentally and understood surprisingly well long before the atomic nature of matter was known, and people at the time did not know whether charge flowed in conductors through the movement of positively charged particles in one direction, or negatively charged particles in the other. They guessed (Benjamin Franklin was involved in all this, by the way), but unfortunately they guessed wrong, and that wrong guess still haunts us today. They guessed that it was positive charge that moved in conductors, and current was always thought of as the flow of positively charged particles. Throughout your career in electrical engineering you will almost always conform to this wrong guess and think of electrical current as the flow of positive charge. Fortunately it is a pretty benign error because almost all observable effects cannot distinguish between the two possibilities, and calculations based on this incorrect assumption will accurately predict what actually happens (at least if you don't make any other mistakes!).

As an aside, it is possible to distinguish experimentally between the two possibilities for current flow. The most common technique for doing so uses a magnetic field, and is called the *Hall effect* after its discoverer. Just after the turn of the century, the atomic nature of matter was being uncovered, and workers realized that it made much more sense in terms of this model if current were carried by the motion of the electrons than by the protons bound tightly into the atomic nuclei. It must have been comforting when Hall effect measurements showed the sign of the charge on the current carriers to be negative. Well, it was almost always negative. There was the strange case of boron, for which the sign was clearly positive. The resolution of that puzzle was an important piece in the invention of the transistor. That's another story, though.

Back to the flow of electrical current, from now on we will always think of it as the flow of positively charged carriers. We will think of conductors as being made up of a lot of fixed negative charges and an equal number of mobile positive charges which can be made to move. That's exactly backwards, but we'll not get a wrong answer because of it. The reason is that from a charge transfer point of view a negative charge moving in one direction is equivalent to a positive charge moving in the other. We make the incorrect assumption because

1. That's what everybody else does, and it is necessary to be able to talk to other people;
2. It simplifies the arithmetic a little in that there is not an extra minus sign floating around because of the negative charge of the current carriers; and
3. Most importantly, except in rare and easy-to-recognize circumstances the error will not lead us to an incorrect analysis of current flow in circuits.

This is the last time I'll mention this point. From now on I'll toe the party line and act as if the charge carriers were positive.

We need a measure of current. In the analogy with water we would measure the flow of water in a pipe as an amount of water flowing through the pipe per unit time, measured perhaps in gallons per second. To do the same thing with current, we need an analogous measure of charge. The standard unit of charge is

the *coulomb*. The charge on an electron is approximately $-1.6 \times 10-19$ coulomb, and it would require about 6.2×1018 electrons to make up minus one coulomb of charge. The measure of electrical current is charge per second, and the standard unit is the *ampere*, which is one coulomb per second. To give you an estimate of size, currents commonly encountered range between roughly millionths of an ampere (or microamperes) and tens of amperes. The unit "ampere" is often abbreviated to just "amp".

8.3 Voltage

We also need a measure of the strength of the agent causing the current to flow. One such measure would just be the net electrical force acting on the mobile charges. If we were interested in knowing just how the current flow was distributed in a conductor (is it mostly near the outside or the center, etc.) we would use such a measure. In analysis of current flow in circuits, however, we don't usually care about such details as whether the current is mostly flowing in the top half or the bottom half of the wire; instead we just need to know the rate at which charge is flowing from one component to the next. In this case, we use a quantity called the *voltage* as a measure of the strength of the agent causing the current flow. The voltage is very closely related to an electrical potential energy, and in fact it is sometimes called the electrical potential.

This idea is best described in terms of an analogy. Consider again the two tanks of water in Fig. 8−1, and allow the two tanks to possibly have different heights and different amounts of water in them as in Fig. 8−2.

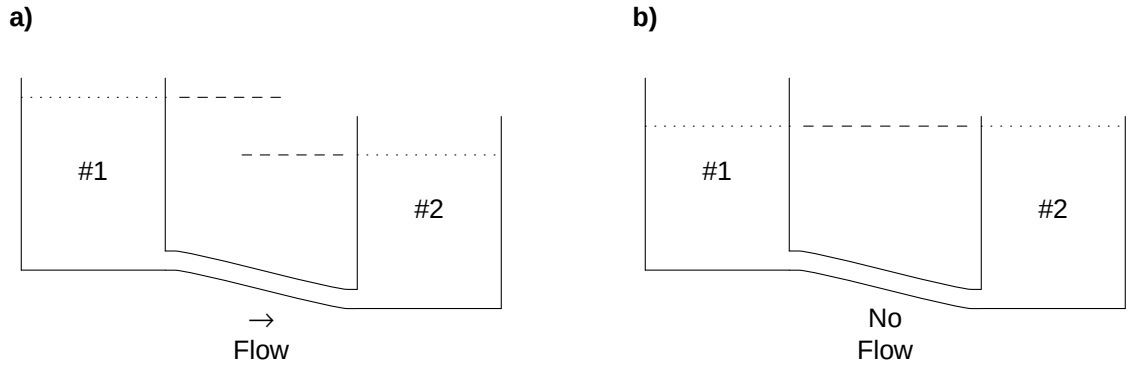

Figure 8−2. Drawing illustrating the effect of difference in water level on the flow of water. In a) and b) the tanks are at different heights, but in a) the level of the top of the water in #1 is higher than in #2, whereas in b) the levels are the same. Water flows in a) but not b).

The decisive question in determining the flow of water from one tank to the other is the difference in heights of the top of each pool of water. If a little water flows from one pool to the other, the top of one pool will lower a little and the top of the other will rise. The net result is the same as taking a thin layer of water from the top of the first pool and putting it on top of the water in the second. If the potential energy of the water at the top of pool #1 is greater than the potential energy of the water at the top of pool #2, doing so will reduce the gravitational potential energy of the system. Water will flow from #1 to #2 at a rate which depends on this difference in potential energy. On the other hand, if the gravitational potential energy of the water at the top of pool #1 is less than that of the water at the top of #2 water will flow in the opposite direction, from #2 to #1, again at a rate determined by the potential energy difference.

In an electrical circuit the voltage is analogous to the height of the top of the water in the water circuit. If the electrical potential energy of a charge carrier at the "top of the charge pool" is higher in sphere #1 than in sphere #2 current will flow from 1 to 2, and vice versa. (As an aside, the top of the charge pool is called the *Fermi level*.) The unit of voltage is the *volt*. One coulomb of charge changing voltage by one

volt gives up (or gains if the voltage change is positive) one joule of energy. Typical magnitudes of voltage differences that are commonly encountered range from millionths of a volt (or microvolts) to tens of thousands of volts (tens of kilovolts). The signal that a radio receiver picks up is in the microvolt range, and the voltage used on a color television tube is around 30 kilovolts.

The current that actually flows will be determined by the difference in potential energies or voltages of the two "pools". If the difference is positive (i.e. if $V_1 > V_2$) the current will flow from #1 to #2. A negative voltage difference would mean the current flows in the opposite direction, from #2 to #1. Note that the voltage *per se* of each sphere is not relevant; what matters is the voltage difference.

8.4 Current-Voltage Relations

If we want to be able to predict just how much current flows between the spheres, we need another bit of information—the *functional dependence* of the current in the conductor on the voltage difference of the "pools" of charge carriers at the two ends of the wire (the difference in the Fermi levels). The functional dependence of current on voltage is what Herbie Husker was required to find for the box he was given in EE231 lab in the example we considered way back in section 4.10. He could either give it as a graph of measured current vs. voltage, or he could give it as a formula, something like $V = 2.04 + 853\,I$. Actually, this is a relation giving V as a function of I. To get a relation giving I as a function of V we would just solve for I, getting, $I = \frac{1}{853}(V - 2.04)$. This functional dependence is often called the *I-V relation*, or the *I-V curve* if it is provided graphically. You will see such curves all over the place in the catalogs manufacturers of transistors distribute to describe their devices.

These I-V curves for components used in electronic circuits can be very simple or quite complicated—it depends on the component. The simplest curve is the I-V curve of a good insulator. This curve is essentially just a horizontal line passing through $I = 0$. A formula giving the functional dependence of current on voltage for an insulator is easy, $I = 0$. The next simplest curve is that of a *resistor*. Resistors are probably the most common components in electronic circuits (at least if you don't count insulators and wires). For a resistor the I-V curve is a straight line passing through the origin (that means I is zero when V is zero). This functional dependence can also be specified with a formula,

$$V = IR \tag{8−1a}$$

or, equivalently

$$I = \frac{1}{R}\,V = GV \tag{8−1b}$$

where R is a constant which characterizes the resistor. This dependence is called *Ohm's law*. Most resistors obey the law quite accurately, but many other components do not. A diode, for example, is a common component which (more or less) allows current to flow in only one direction. The I-V curve for a diode is far from a straight line.

In Eqs. (8−1) R is a constant which depends on the resistor and $G = 1/R$. R is called the resistance and G is called the conductance. The unit of resistance is volts per amp, and is called an *ohm*. The unit of conductance is amps per volt, and is called a *mho* (get it?). The symbol for an ohm is the Greek capital omega, Ω, and the symbol for a mho is an upside down capital omega. Values of resistance range over probably the largest range of any common physical quantity. A copper wire will have a resistance much less than 0.0001 Ω, whereas the resistance of even a short piece of glass is so high it is difficult to measure, but it certainly exceeds 1015 Ω.

A few materials even can become *superconductors*. These materials behave more or less normally at room temperature, but as the temperature is lowered the resistance abruptly falls to zero! I don't mean falls to some very small value, I mean ZERO. The temperature at which this transition to zero resistance takes place is called the transition temperature. Until a few years ago all known superconductors had very low transition temperatures, the highest being only about 21° above absolute zero (21 K, −252° C, or −452° F).

Recently, several materials have been found with much higher transition temperatures. The present record is about 125 K. Nitrogen liquifies at 77 K, so liquid nitrogen can be used as a refrigerant for these new "high-temperature" superconductors. That is an important advance because nitrogen is plentiful, harmless, and easy to liquify. For superconductors with transition temperatures below about 20 K, liquid helium must be used to cool them to superconductivity. Helium is much rarer, and therefore more expensive than nitrogen, a lousy refrigerant, and difficult to liquify.

Two other common circuit components are *capacitors* and *inductors*. The current through a capacitor, I, is proportional to the time derivative of the voltage, V, across it.

$$I = C\,\frac{dV}{dt} \tag{8−2}$$

The constant of proportionality, C, is called the *capacitance*, and the unit of capacitance is the *farad*. Common values of capacitance range from a few picofarads (10^{-12} F) to a few millifarads (10^{-3} F). The voltage, V, across an inductor is proportional to the time derivative of the current, I, through it.

$$V = L\,\frac{dI}{dt} \tag{8−3}$$

The constant of proportionality, L, is called the *inductance*, and the unit of inductance is the *henry*. Common values of inductance range from nanohenries (10^{-9} H) to a few henries. Because of the time derivatives in the current-voltage relationships for these devices, we will not consider them in this course. In the following course, EEngr 122, we will consider circuits with inductors and capacitors in them.

8.5 Voltage Sources

Back to our analogy with two pools of water, suppose for some reason I want to keep a certain current of water flowing through the pipe connecting them. If I just have the two pools, with the water level in one starting out higher than in the other, then as the current flows in the pipe water will be transferred from one pool to the other. The levels will equalize, causing the potentials of the two pools to become the same, and current to cease. To keep current flowing I would need some kind of a pump as in Fig. 8−3, to pump the water continuously back from pool #2 to pool #1.

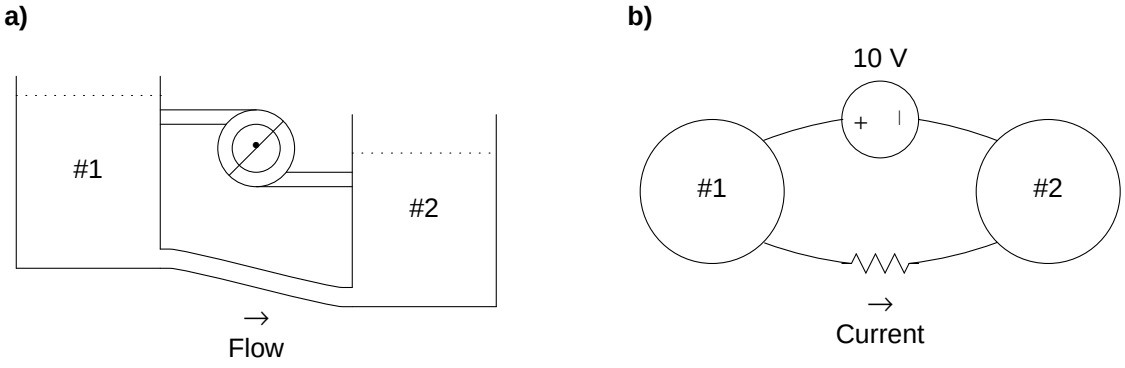

Figure 8−3. Drawing illustrating the operation of a voltage source. In a) a pump maintains the level of water in tank #1 higher than in #2 and water flows continuously in the connecting pipe. In b) a voltage source maintains the voltage of sphere #1 higher than that of #2, and current flows continuously in the connecting resistor.

The pump would transfer water from pool #2 to #1, causing the level of the water in #1 to rise above that of #2. The higher level in 1 than 2 would cause water to flow, as desired, in the pipe connecting the two pools. As you have probably guessed, there are electrical analogs to pumps. They are called *sources*, and ideally

they come in two models: *voltage sources* which maintain a set voltage difference between two conductors (analogous to a pump which maintains a certain water height difference, or head), and *current sources* which maintain a set current flow in a conductor (analogous to a pump which pumps a certain amount of water per unit time).

A battery is an everyday example of a voltage source. It maintains the voltage across its terminals at nearly the advertised value as long as you don't try to draw too much current from it. Everyday examples of current sources are much more scarce, but they occur commonly in transistor circuits. When you take a circuit analysis course you will encounter both types of sources frequently, but here we'll only consider voltage sources. Unless we specify otherwise, we will always assume that the currents never get so large that sources are strained, allowing us to assume that the voltages maintained by our voltage sources are always the advertised values. A mythical source which maintains its current or voltage at the specified value no matter what you do to it is called an *ideal* source. A snazzier way of saying what I just said would have been "We'll assume all our voltage sources are ideal".

In describing a pump it is necessary to specify both the "strength" of the pump, and the direction in which it moves water. The "strength" would be specified by giving either the difference in water levels or the water flow it can maintain. The direction would be specified most likely by an arrow, or by marking on the inlet and outlet pipes. Similarly for a voltage source, one must specify the voltage difference it maintains between its two terminals, and which terminal has the higher voltage (the polarity). In Fig. 8−3, the source maintains a voltage difference of 10 V, and the left hand terminal has the higher voltage, as implied by the + and − markings.

There is one difference between common practice in specifying pumps and sources, however. With a pump, the head or water flow is always specified as a positive number, and the water always flows in the direction of the arrow on the pump or as indicated by inlet and outlet markings. With an electrical source, the specified voltage or current can be a negative number. In this case, the current direction or the voltage polarity is just the opposite of the markings. This convention may seem unnecessarily complicated, but it turns out to be very useful, as I will point out later.

8.6 Schematic Diagrams and Polarity Conventions

I now want to discuss how to figure out what the currents and voltages in circuits will be. First we need to find a way to specify what the circuit is. The common method is to use a *schematic diagram*. We use agreed-upon symbols for sources, resistors, and other electronic components. The common symbols for voltage and current sources as well as resistors, capacitors, and inductors are shown in Fig. 8−4.

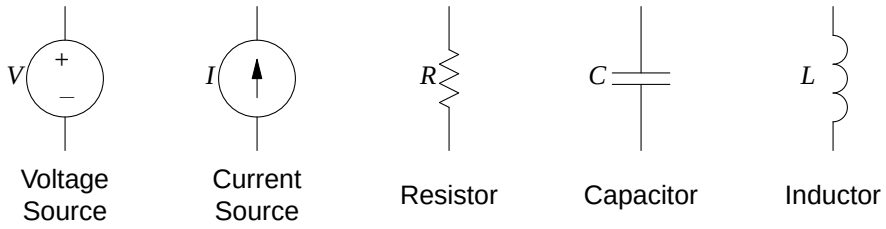

Figure 8−4. Symbols used in schematic diagrams for voltage and current sources and for resistors, capacitors, and inductors.

The resistance of a resistor is shown next to the corresponding symbol. For a voltage source, the voltage difference it maintains is shown next to it, and the "+" and "−" symbols are used to indicate which terminal has the higher voltage as follows. If the voltage given is a positive number, the terminal labeled "+" has the higher voltage; if the voltage given is a negative number, the "−" terminal is the higher. Perhaps a clearer

way of looking at this is that the voltage difference given is the voltage of the terminal labeled "+" minus the voltage of the terminal labeled "−". For a current source, an arrow is provided inside the circle as shown, and the current the source delivers is shown next to it. If the specified current is positive, current flows in the direction of the arrow, if it is negative, it flows in the opposite direction.

In a real circuit components are connected together with wires; in a schematic diagram they are connected with lines. In a real circuit we can usually neglect the resistance of these wires; in a schematic diagram we always take the lines to have zero resistance. With this assumption, then, all points on a wire must have the same voltage. The point where two or more components connect together is called a *node*. A node is sometimes shown explicitly on a schematic diagram by putting a dot over the connection.

8.7 The Current Flowing in a Simple Circuit

Enough preliminaries, how do you figure out how current flows in a circuit? To do so, it is necessary that we know the current-voltage relations of all the components of the circuit, and how the components are connected. Consider the circuit shown in Fig. 8−5 consisting of simply a 3 V voltage source and a 100 Ω resistor.

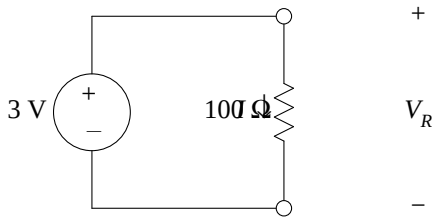

Figure 8−5. Schematic diagram of a simple circuit

How much current flows in the resistor, and in what direction? Let's call the current I. Then by Ohm's law, the voltage drop across the resistor must be $V_R = 100\,I$. The voltage across the voltage source is 3 V, and the terminals of the voltage source are connected with ideal wires to the terminals of the resistor, so the voltage drops across the source and the resistor must be the same.

$$100\,I = 3$$

or

$$I = \tfrac{3}{100} = .03 \text{ A}$$
$$= 30 \text{ mA}$$

Here "A" is an abbreviation for "ampere", and "mA" is an abbreviation for "milliampere", or thousandth of an ampere.

In what direction is the current flowing through the resistor? In passive components (those without charge pumps inside them) current flows from higher voltage to lower, so if the voltage at the top of the resistor is higher than at the bottom, current will flow from the top down; whereas if the voltage on the top is lower the current will flow from the bottom up. In our case the top of the voltage source has the "+" sign, and the voltage is given as a positive value, so the top has the higher voltage, and the current flows from the top of the resistor to the bottom. Note that this current must come from somewhere, and it can only come from the voltage source, so the source must be supplying 30 mA. Note also that the current must flow into the lower voltage terminal of the voltage source and out of the higher voltage terminal, opposite to the direction in a passive component. This can happen because some agent besides direct electrical force causes the current to flow inside the battery. Inside the source the electric force pushes positive charges from the positive terminal to the negative and some other agent, such as a chemical reaction, must be active to pull the charges "uphill" against the electric force so that the source can supply positive charge from the

positive terminal, and accept it at the negative. (I like to think of the agent as being some poor little horse, sweating like crazy, inside the battery pulling the positive charges against the electrical forces, from the negative terminal up to the positive.)

8.8 Conventions for Specifying Current Direction and Voltage Polarity

Notice the way in which I indicated the current through the resistor and the voltage across it, I and V_R in Fig. 8−5. To specify either quantity two pieces of information were needed: a number giving the size of the quantity (number of amperes or volts), and some directional symbol which allows us to say in which direction the current is flowing, or in which direction the voltage decreases. The convention is the same as used in labeling voltage and current sources. For currents, if the number given (I in this case) is positive, current flows in the direction pointed by the arrow; if negative, in the opposite direction. Similarly, for voltages, if the number given (V_R in this case) is positive, the node with the higher voltage is that labeled with the plus sign; if negative, the higher voltage node is that with the minus sign.

You may be wondering why we connect the two symbols. For currents, for example, why not give a number which is always positive, and use the label to specify directly the direction of current flow? The reason is that in more complicated circuits, it is often not obvious which direction the current in a component flows at the start of a calculation. One of the reasons for doing the calculation is to figure out just that! Part of the power of algebraic analysis of circuits is that if we are consistent with our conventions, the algebra will figure out directions of current flow and voltage polarities for us through the sign of the answer.

For every component you are free to choose whatever sign convention you like. If I had chosen to draw the arrow associated with the current through the resistor in Fig. 8−5 pointing in the other direction, everything would have been OK, but I would have finally gotten $I = -30$ mA, instead of +30 mA. The current in both cases flows in the same direction (the actual circuit can't possibly know how I'm going to decide to draw an arrow on a piece of paper!), but in one case I specify this direction using a positive number for I, and in the other I must use a negative number.

Some choices of sign conventions make life easier than others, however. I strongly recommend that when you specify both a voltage across some component and the current through it, as we did for the resistor in Fig. 8−5, you choose the direction of the arrow to point from the side labeled with a plus sign for the voltage drop to the side with a minus sign. If you do this, then Ohm's law, $V = IR$, will be correct *including the sign*. With this convention, if V is positive, both Ohm's law and physical sense say that I is also positive; and if V is negative, both agree that the current flows in the opposite direction. If you stick to the convention, the algebra will figure out the current direction as well as magnitude for you. If you were to choose the opposite convention, then it would still work, but you would have to write Ohm's law as $V = -IR$.

For example, in Fig. 8−5, V_R must be the same as the voltage drop from the top of the voltage source to the bottom, +3 V. Therefore, by Ohm's law, $I = V_R/100 = +3/100$ A. I is positive, so current flows in the direction of the arrow, from the top of the resistor to the bottom. That's just what it should do. On the other hand, if the voltage source had been −3 V instead of +3 V, the voltage drop across the source, and hence V_R, would have been −3 V, and I would have been $I = V_R/100 = -3/100$ A. I would have been negative, implying that current flowed opposite to the arrow, from the bottom of the resistor to the top. Again, that's exactly right. The current in a passive component always flows from higher voltage to lower.

8.9 Circuits with Two Resistors

Let's consider a slightly more complicated circuit, shown in Fig. 8−6. In this case, two resistors, one 100 Ω and the other 150 Ω, are connected to a 3 V source. Call I_1 the current through the 100 Ω resistor, and I_2 that through the 150 Ω. Perhaps I should label the voltage drop across each resistor separately, calling them maybe V_{R_1} and V_{R_2}, but the two voltage drops are the same because the ends of the two resistors are connected by wires, so I'll just stick with one label, V_R. This voltage drop is just equal to that across the voltage source, or +3 V, because the two nodes are connected to the terminals of the voltage source with

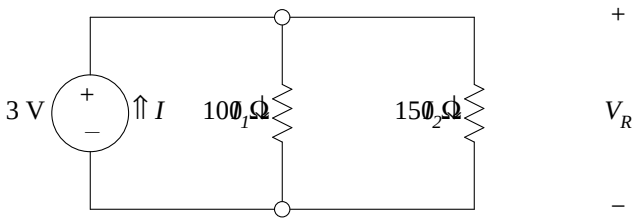

Figure 8−6. A circuit containing two resistors. The resistors are said to be connected in *parallel*.

wires, so it's easy to figure out the two currents.

$$I_1 = \frac{V_R}{100} = \frac{+3}{100}\ \text{A} = +30\ \text{mA}$$
$$I_2 = \frac{V_R}{150} = \frac{+3}{150}\ \text{A} = +20\ \text{mA}$$

Note that both currents are positive, implying that they flow in the direction of the arrows in Fig. 8−6. This current must come from somewhere, and the only possibility is the voltage source, so it must be supplying $I = I_1 + I_2 = 50$ mA.

Just to demonstrate again how you can use the algebra to determine the direction of current flow, I'll work the problem, but this time using the opposite sign conventions for I_1 and I_2. Fig. 8−7 shows the circuit with these changed conventions. I want to emphasize that *the physical circuit is the same*, the only thing that has changed is the mind of the crazy guy trying to figure out what the circuit does.

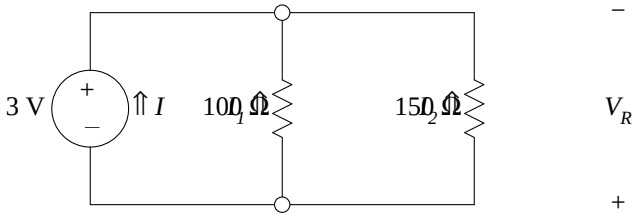

Figure 8−7. The same circuit as in Fig. 8−6, but with reversed sign conventions for I_1, I_2, and V_R.

As before, we write

$$I_1 = \frac{V_R}{100}$$
$$I_2 = \frac{V_R}{150}$$

This time, however, V_R is the voltage drop across either resistor, starting at the bottom and going up (because of the location of the + and − signs associated with V_R). This voltage drop is the same as the voltage drop across the source, going in the same direction. The upper terminal of the source is higher in voltage than the lower, so there is a voltage rise, not drop, of 3 V. The voltage drop is therefore −3 V, and $V_R = -3$ V. Putting this value into the equations for I_1 and I_2 gives

$$I_1 = -30\ \text{mA}$$
$$I_2 = -20\ \text{mA}$$

This time both I's are negative, implying that current flows in the opposite direction to the arrows, or down, just as before, and just as it ought to be.

We consider one more circuit with two resistors, shown in Fig. 8−8. In this case, it is clear that the

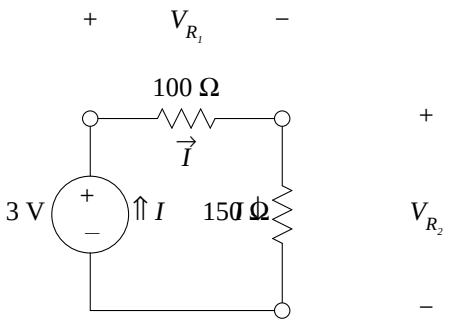

Figure 8−8. A different circuit involving two resistors. The resistors in this case are said to be connected in *series*.

currents through the voltage source, and through the two resistors must all be the same because there's nowhere else for it to go. The voltage drops across each of the two resistors and across the voltage source are not the same, however. Instead, the total voltage drop across both the resistors must equal the voltage drop across the source. The total drop is the drop across both the 100 Ω and 150 Ω resistors, and is just the sum of the individual voltage drops, $V = V_{R_1} + V_{R_2}$. But by Ohm's law, each of the individual drops is just I times the corresponding resistance,

$$V = 100\,I + 150\,I = (100 + 150)\,I = 250\,I \qquad (8-4)$$

Finally, equating this total drop to the drop across the source gives

$$I = \frac{+3}{250}\ \text{A} = 12\ \text{mA}$$

The sign of I is positive, so the current flows in the direction of the arrows.

8.10 A Non-Trivial Circuit

So far, we have looked only at circuits for which it was easy to tell *a priori* the direction of current flow in all components. Those of you with a little experience in the area probably found the analysis of the circuits pretty simple. We now will analyze a more complicated circuit, for which the direction of current flow in at least one resistor is not at all obvious at the start. Consider the circuit shown in Fig. 8−9 a).

a) **b)**

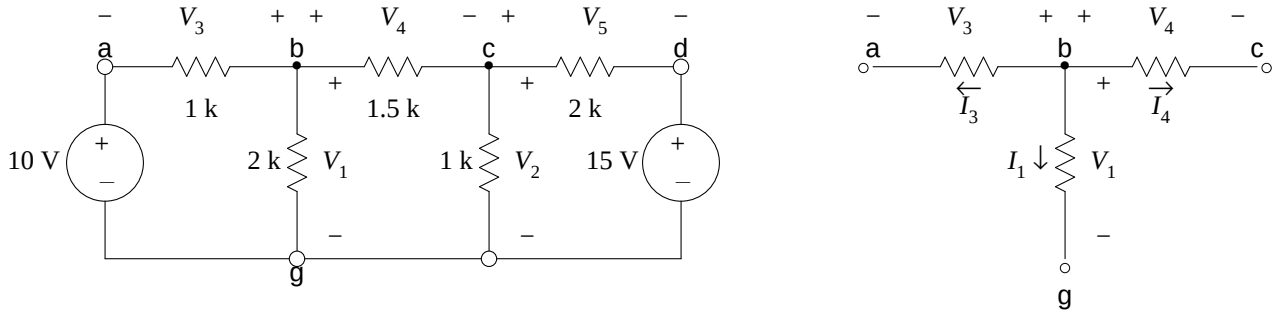

Figure 8−9. a) More complex circuit for analysis. The k in the resistor values is a standard shorthand for kΩ, or 1000's of ohms. Voltage drops across all resistors have been defined and nodes labeled with letters a-d and g. b) Sub-circuit taken from a) for easier analysis. The same node labels are used, and the three resistors are the left-most resistors in a), with values 1 k, 2 k, and 1.5 k.

Resistor values have been left off for clarity, and currents through each defined. Note the relative directions of the arrows relative to the + and − signs on the voltage definitions.

In Fig. 8−9, I have labeled the nodes with letters a-d and g, and defined voltage drops across all resistors as V_1, V_2, ..., V_5. The polarities have been arbitrarily assigned as shown on the diagram. These polarity definitions do not mean that the voltage on the side labeled with a + sign is higher than or positive with respect to the voltage on the side labeled with a − sign. For some of the resistors, particularly the 1.5 k resistor, *I don't know* at this point which side has the higher voltage. Rather, the + and − signs simply mean that if the corresponding voltage (V_4 for example) turns out to be positive, the voltage of the + side is higher; otherwise, the voltage of the − side is higher.

At first glance, this seems like a formidable problem algebraically in that there are five unknowns. Fortunately, several of the unknowns are simply related to others. To make this relationship easier to see, let's agree to measure all voltages with respect to the bottom-most node, labeled g in the figure. (If you find that statement confusing, think of using a voltmeter to measure the voltages at all the top nodes in the circuit. You would connect one terminal of the voltmeter to the bottom node, and then sequentially connect the other to each of the top nodes, noting the reading each time.) Then the voltage of node a is just 10 V, and the voltage of node b is just V_1. But V_3 is the voltage of the node marked with a + minus the voltage of the node with a −. These two voltages are V_1 and 10, respectively, so V_3 is just

$$V_3 = V_b - V_a = V_1 - 10 \tag{8−5a}$$

In a similar manner, we can relate V_4 and V_5 to V_1, V_2, and the voltage of the right hand source, 15 V. Notice how the + and − signs associated with each voltage definition in the figure determine which voltage is subtracted from which.

$$\begin{aligned} V_4 &= V_b - V_c = V_1 - V_2 \\ V_5 &= V_c - V_d = V_2 - 15 \end{aligned} \tag{8−5c}$$

To analyze this circuit we will make use of the fact that no net current can either enter or leave a node. If it did, charge (of one sign or the other) would continuously build up on the node until something broke down. The situation is analogous to a tee in a plumbing system. Water can flow into and out of the pipes connected to the tee, but whatever flows in must flow out, and *vice versa*. If, say, more water flowed in than flowed out, the amount of water in the tee would increase. Water, like charge, is not very compressible, so the pressure in the tee would rise rapidly until it blew up.

To analyze the situation more carefully, consider the node labeled b in Fig. 8−9 a). For clarity, I've separated out just this node and the components connected directly to it in Fig. 8−9 b). I've left off the values of the resistors, and I've defined currents, I_1, I_3, and I_4 through each component. Notice that in defining the polarity of these currents I have adhered to the convention that the arrow in the definition of the current flow through a component points from the + to the − signs in the definition of the voltage across the component. Applying our no-net-current rule to this node requires that

$$I_3 + I_1 + I_4 = 0 \tag{8−6}$$

Note that this equation implies that at least one of the I's must be negative, meaning that the current in at least one of the resistors flows in the direction opposite to the arrow on the definition of the corresponding current.

Now these currents are related to the corresponding voltages by Ohm's law,

$$I_3 = \frac{V_3}{1000}$$
$$I_1 = \frac{V_1}{2000} \tag{8−7c}$$
$$I_4 = \frac{V_4}{1500}$$

Putting these relations in Eq. (8−6), and using Eqs. (8−5) to express V_3 and V_4 in terms of V_1, V_2, and the values of the voltage sources, we obtain

$$\frac{V_1 - 10}{1000} + \frac{V_1}{2000} + \frac{V_1 - V_2}{1500} = 0 \tag{8−8a}$$

We apply the same analysis to node c, encountering only one small difficulty; with the polarities in the figure the current through the 1.5 k resistor has its arrow pointing into the node, not away from it. The difficulty is minor because the current leaving the node through this component is just the negative of this value. Thus we obtain

$$-\frac{V_1 - V_2}{1500} + \frac{V_2}{1000} + \frac{V_2 - 15}{2000} = 0 \tag{8−8b}$$

I don't like the denominators in Eqs. (8−8), so I will multiply both sides of both equations by 6000, and collect terms, getting

$$\begin{aligned} 13\,V_1 - 4\,V_2 &= 60 \\ -4\,V_1 + 13\,V_2 &= 45 \end{aligned} \tag{8−9b}$$

These equations can be easily solved for V_1 and V_2, and the results used in Eqs. (8−5) and (8−7) to determine all the voltages and currents,

$$\begin{aligned} V_1 &= \tfrac{320}{51}\ \text{V} & I_1 &= \tfrac{160}{51}\ \text{mA} \\ V_2 &= \tfrac{275}{51}\ \text{V} & I_2 &= \tfrac{275}{51}\ \text{mA} \\ V_3 &= -\tfrac{190}{51}\ \text{V} & I_3 &= -\tfrac{190}{51}\ \text{mA} \\ V_4 &= \tfrac{15}{17}\ \text{V} & I_4 &= \tfrac{10}{17}\ \text{mA} \\ V_5 &= -\tfrac{490}{51}\ \text{V} & I_5 &= -\tfrac{245}{51}\ \text{mA} \end{aligned} \tag{8−10}$$

Notice that V_3 is negative. That is not bad, it just means that the voltage of node a is higher than that of node b. Also I_3 is negative. That only means that current flows opposite to the arrow in Fig. 8−9 b), into node b, and away from node a. Similar observations apply to V_5 and I_5.

The final summary is as follows.

1. The 10 V source supplies $\frac{190}{51}$ mA of current. This current flows out the top terminal and through the 1 kΩ resistor. The current then splits, with $\frac{160}{51}$ mA, or about 84% of it flowing through the vertical 2 kΩ resistor, and the remaining $\frac{10}{17}$ mA, or 16%, flowing through the 1.5 kΩ resistor.
2. The 15 V source supplies $\frac{245}{51}$ mA of current. This current flows out the top terminal and through the 2 kΩ resistor. $\frac{275}{51}$ mA flows through the vertical 1 kΩ resistor. Of this, $\frac{245}{51}$ mA, or about 89%, comes from the 15 V source via the 2 kΩ resistor, and the remaining $\frac{10}{17}$ mA, or 11%, from the 10 V

supply via the 1.5 kΩ resistor.

8.11 Exercises

1. Investigate the charge imbalance expected on nodes in circuits. In circuit analysis it is commonly assumed that any current flowing into a node is *exactly* balanced by an equal current flowing out of it. In fact, this assumption is so revered that it's called a law: Kirchoff's Current Law, to be exact. If the voltage of the node is changing, this law is not strictly true, because the way the circuit changes the voltage of a node is by piling up or removing charge from it. If the voltage is changing, the charge on the node must change also, and there must be unequal currents flowing into and out of the node. The claim is that it takes so little charge to change the voltage that this current imbalance is small, and can be neglected. In this exercise, you are to check the validity of such a claim. Suppose that the voltage on the node changes from +5 V to −5 V and estimate the change in the charge on the node associated with this voltage change. Dividing the change in the charge by the time over which it changed, call that T, will give an estimate of the current imbalance.

   You will need an equation giving the charge on the node as a function of the node voltage. Unfortunately, that relation depends on the shape and size of the wires making up the node, and on the location of other components in the circuit. It is very complicated. As a crude approximation, use the relation giving the voltage on a conducting sphere of radius r, relative to infinity,

$$V(q) = \frac{q}{4\pi\varepsilon_0 r} \approx 10^{10}\ \frac{q}{r} \qquad (8-11)$$

   valid for q in coulombs, r in meters, and V in volts. Most nodes do not have a spherical shape, and the voltage we talk about in circuits is not relative to infinity, but rather relative to some nearby ground plane, so Eq. (8−11) will probably be in error by more than a factor of two, but it should be in the right ball park. (If you have not encountered electrostatics before, and consequently don't know what the heck I'm talking about, don't worry about it, just take Eq. (8−11) to describe the relationship between the voltage of a node and the charge on it.) Typical circuits have dimensions of the order of a few centimeters, so take r to be 3 cm.

   a. What is the change in charge on the node when the voltage changes from +5 to −5 V?

   b. Rather arbitrarily, let's consider Kirchoff's law correct if it leads to an error of no more than 0.1%. Typical currents in circuits are of the order of 1 mA (= .001 A). How fast would the voltage have to change from +5 to −5 V in order to produce a current imbalance at the node larger than our limit of 0.1% of this current?

   c. If the voltage is changing periodically between + and − 5 V, taking T seconds to make each transition, then the voltage has frequency $1/(2T)$. With the assumptions in part b), starting at about what frequency would you expect the inexact balancing of currents into and out of a node to become significant? (In other words, at what frequencies does Kirchoff's Current Law become suspect?)

   Before I get accused of heresy, even for frequencies so high that the current imbalance is not negligible, Kirchoff is kept sacrosanct by describing the error as being due to an effect called *parasitic capacitance*. A capacitor is a component we discussed only briefly which can store charge. The charge building up due to the imbalance in current is thought of as flowing onto a fictitious parasitic capacitor. The procedure is not as hokey as it sounds.

2. Consider the circuit shown in Fig. 8−8. We found, in Eq. 8−4 that the voltage across both resistors, V, was related to the current through them, I, by

$$V = (100 + 150)I = 250I$$

   This is the same voltage-current relationship expected from just a single resistor with resistance equal to the sum of the resistances of two individual resistors. If our task had been to calculate the current drawn from the voltage source, we could have instead calculated the current supplied by the

voltage source in a single resistor circuit like that of Fig. 8−5, except with a 250 Ω resistor instead of 100 Ω. They are two different problems, but the answers to both are the same, and the second problem is easier.

Show that this observation can be generalized to any two resistors connected in series as in Fig. 8−8. Let the two resistors have resistances R_1 and R_2, and show that the current-voltage relationship seen by the voltage source is the same as for a single resistor with resistance $R = R_1 + R_2$.

3. Consider the circuit of Fig. 8−6. We found that for a 3 V source, a total of 50 mA of current was drawn. This current is the same as would have been drawn by a single 60 Ω resistor. In this case, the connection between the 60 Ω equivalent resistance and the resistance of the individual resistors is not as obvious as in the previous problem. Show that for any two resistors of resistance R_1 and R_2 connected in parallel as in Fig. 8−6 the equivalent resistance, R, is given by

$$R = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2}}$$

Note that this equation can be written

$$R = \frac{R_1 R_2}{R_1 + R_2}$$

4. What are the voltages, V_1 and V_2, in Fig. 8−10?

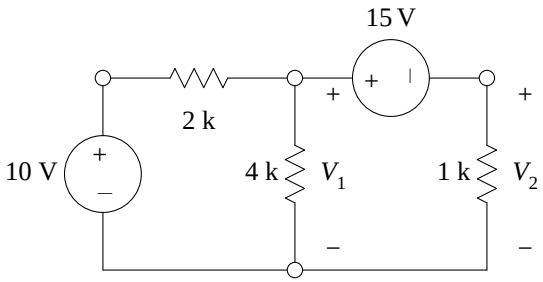

Figure 8−10. Circuit for exercise 8−4.

HINT: The current through the 15 V source is the same as the current through the 1 kΩ resistor.

5. Consider the circuit shown in Fig. 8−11.

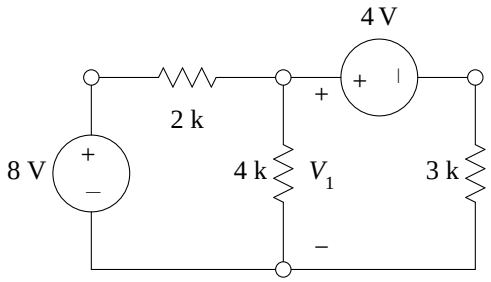

Figure 8−11. Circuit for exercise 8−5.

a. What is V_1?

b. Suppose that the left-hand voltage source (the one labeled 8 V in the figure) were variable, so that the output voltage could be adjusted. To what voltage would it have to be adjusted in order to make V_1 zero?

6. Find the voltages v_1 and v_2 in the circuit in Fig. 8−12.

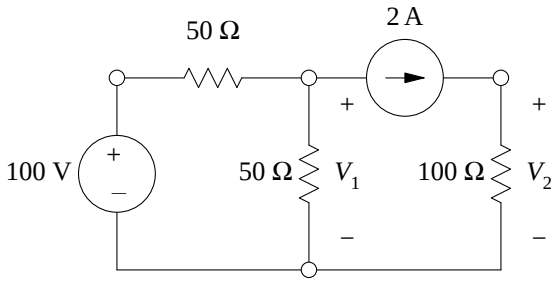

Figure 8−12. Circuit for exercise 8−6.

9. MATRICES AND SETS OF LINEAR ALGEBRAIC EQUATIONS

9.1 Introduction

A common problem in electrical engineering is the solution of a set of N linear algebraic equations in N unknowns. For example, in the last problem of Chapter 8 we encountered such a set with $N = 2$. If N is not very large, say less than about five, the solution of such a set by analytic means (using algebra) is feasible, although the probability of making an arithmetic or algebraic error increases rapidly with N. For larger values of N, manual methods become increasingly impractical, and one is forced to use some automated method, usually involving a computer. The purpose of this chapter is to introduce you to some of the computer methods available for solving such problems, and to present some elementary material on matrices. This latter part may be mostly review for you.

Consider a set of N linear algebraic equations in N unknowns. I want to be general and allow for any number of unknowns, so instead of labeling the unknowns x, y, and z, or some such scheme, I'll label them with subscripts, x_1, x_2, ..., x_N. Similarly, instead of using a, b, ... $zq26$, for the coefficients of the x's in each equation, I'll use two subscripts, both running from 1 to N. The first subscript will label which equation the coefficient is associated with, and the second will label which variable.

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,N}x_N &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,N}x_N &= b_2 \\ 3dot & \\ a_{N,1}x_1 + a_{N,2}x_2 + \cdots + a_{N,N}x_N &= b_N \end{aligned} \tag{9-1}$$

The standard problem is: given such a set of equations with all the a's and b's known numbers, find the set of x's, $\{x_1, x_2, \cdots, x_N\}$ such that all N of the equations are satisfied at once.

9.2 Matrices

Writing all the junk in Eq. (9−1) every time we want to talk about the set of equations can get tiring rapidly. A very useful shorthand notation has been developed to deal with (among other things) such sets of equations. Actually its more than just a shorthand notation, it's a minor branch of mathematics, but for now just consider it as shorthand. We write all the coefficients of the unknowns in Eq. (9−1) in an ordered, rectangular array called a *matrix*, and call the matrix something descriptive, such as A. In these notes I'll try to conform to the convention of using bold Helvetica font, $\mathbf{A}$, for variables referring to matrices. I would then write

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ 3dot & 3dot & 3dot & 3dot \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N} \end{pmatrix} \tag{9-2}$$

The first subscript labels the row number, the second the column. The *dimension* of a matrix is given as $N_{rows} \times N_{columns}$. In this example, $\mathbf{A}$ has dimension $N \times N$. The constants $\{b_1, b_2, \cdots, b_N\}$ can also be gathered into a matrix, in this case of dimension $N \times 1$. Such a matrix (with one dimension equal to 1) is called a *vector*, and in these notes I will use bold Roman font, **b**, to denote vectors. The unknown x's can also be grouped together into a vector, **x**.

$$\mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ 3dot \\ b_N \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ 3dot \\ x_N \end{pmatrix} \tag{9−3}$$

9.3 Matrix Addition and Multiplication

The addition of matrices is defined if the dimensions of the two matrices are the same. If **A** is defined as in Eq. (9−2), and **C** is defined similarly, then

$$\mathbf{A} + \mathbf{C} = \begin{pmatrix} a_{1,1} + c_{1,1} & a_{1,2} + c_{1,2} & \cdots & a_{1,N} + c_{1,N} \\ a_{2,1} + c_{2,1} & a_{2,2} + c_{2,2} & \cdots & a_{2,N} + c_{2,N} \\ 3dot & 3dot & 3dot & 3dot \\ a_{N,1} + c_{N,1} & a_{N,2} + c_{N,2} & \cdots & a_{N,N} + c_{N,N} \end{pmatrix} \tag{9−4a}$$

The multiplication of an ordinary number and a matrix is defined as you might expect.

$$f\mathbf{A} = \begin{pmatrix} fa_{1,1} & fa_{1,2} & \cdots & fa_{1,N} \\ fa_{2,1} & fa_{2,2} & \cdots & fa_{2,N} \\ 3dot & 3dot & 3dot & 3dot \\ fa_{N,1} & fa_{N,2} & \cdots & fa_{N,N} \end{pmatrix} \tag{9−5a}$$

Writing all this out gets quite cumbersome, and it gets even worse when two matrices are multiplied. We can make use of a shorthand notation in which we give a formula involving symbols, such as i and j, to denote the row and column numbers and which is valid for all allowed values of i and j. For example, the definition of the addition of two matrices can be written

$$(\mathbf{A} + \mathbf{C})_{i,j} = \mathbf{A}_{i,j} + \mathbf{C}_{i,j} = a_{i,j} + c_{i,j} \tag{9−4b}$$

and the multiplication of a matrix by an ordinary constant is

$$\left(f\mathbf{A} \right)_{i,j} = f\mathbf{A}_{i,j} = fa_{i,j} \tag{9−5b}$$

With this notation we can define the multiplication of two matrices. The definition may seem unduly complicated, but the motivation for the definition is to make it as useful as possible. I'll show the utility of it just as soon as I define it. Consider two matrices, **A** and **C**, with the first having dimension $M \times P$, and the second $P \times N$. Notice that the number or columns of the first must equal the number of rows of the second for multiplication to be defined. The the product of the two is defined as a matrix of dimension $M \times N$ as follows

$$\begin{aligned} (\mathbf{AC})_{i,j} &= a_{i,1}c_{1,j} + a_{i,2}c_{2,j} + \cdots + a_{i,P}c_{P,j} \\ &= \sum_{k=1}^{P} a_{i,k}c_{k,j} \end{aligned} \tag{9−6b}$$

For example,

$$\begin{pmatrix} 1 & 2 & 0 \\ 3 & -3 & -1 \\ -2 & 1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix} = \begin{pmatrix} (1\cdot 1) + (2\cdot 2) + (0\cdot -1) \\ (3\cdot 1) + (-3\cdot 2) + (-1\cdot -1) \\ (-2\cdot 1) + (1\cdot 2) + (1\cdot -1) \end{pmatrix} = \begin{pmatrix} 5 \\ -2 \\ -1 \end{pmatrix} \tag{9−7}$$

The left matrix has dimension 3×3 and the right matrix 3×1, so multiplication of the two is defined, and the result is a 3×1 matrix. Notice that matrix multiplication is not *commutative*. That means that the order of the matrices to be multiplied makes a difference. In our example, the multiplication of the two matrices in the opposite order is not only not the same, it's not even defined!

Division of two matrices is not defined, but the inverse of a matrix is for most square matrices. (Square means that the number or rows equals the number of columns.) If $\mathbf{A}$ is an $N \times N$ square matrix, then the inverse of $\mathbf{A}$ is written as $\mathbf{A}$-1, and is defined as that matrix which when multiplied on either side by $\mathbf{A}$ gives the identity matrix. The *identity* matrix is a matrix with 1's on the diagonal and 0's elsewhere.

$$\mathbf{A}^{-1}\,\mathbf{A} = \mathbf{A}\,\mathbf{A}^{-1} = \mathbf{I} \tag{9−8}$$

where

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 3dot & 3dot & 3dot & 3dot & 3dot \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \tag{9−9}$$

Note that $\mathbf{A}$-1 must also be an $N \times N$ square matrix. Given $\mathbf{A}$, it is usually not trivial to find $\mathbf{A}$-1. In fact, $\mathbf{A}$-1 may not even exist!

All this is very similar to arithmetic operations on ordinary numbers. The inverse of a number, x, is written as x^{-1}, and is defined as that number which when multiplied by x gives 1. Given the definition of inverses, and a way to calculate them, one need not define division. If one wants to calculate the quotient of two numbers, say x / y, one could simply multiply x by the inverse of y. The ordinary number 0 does not have an arithmetic inverse. Similarly, there are matrices which do not have an inverse. A difference between arithmetic and matrix inverses is that there is only one number without an inverse whereas there are a lot of matrices that do not have inverses.

9.4 Matrices and Sets of Linear Equations

Back to the set of N linear algebraic equations in N unknowns, and the definitions of $\mathbf{A}$, $\mathbf{x}$, and $\mathbf{b}$, perhaps you can see the method to the madness of the definition of matrix multiplication. We can write Eqs. (9−1) very simply as

$$\mathbf{A}\,\mathbf{x} = \mathbf{b} \tag{9−10}$$

If that's not evident to you, write out the ith row of Eq. (9−10),

$$a_{i,1}x_1 + a_{i,2}x_2 + \cdots + a_{i,N}x_N = b_i$$

This is just the ith equation in Eqs. (9−1).

If matrix division were defined, we could solve Eq. (9−10) formally by just dividing by $\mathbf{A}$. Division is not defined, but we can accomplish the same end by multiplying both sides on the left by $\mathbf{A}$-1. Noting that the identity matrix times any other matrix (at least any matrix for which such multiplication is defined) gives just that matrix back, Eq. (9−10) becomes

$$\mathbf{x} \;=\; \mathbf{A}^{-1}\,\mathbf{b} \tag{9−11}$$

This may seem like a real break through, but it's not because it is more difficult to calculate **A**-1 than it is to solve the set of equations directly.

9.5 Using Quattro Pro to Solve Sets of Linear Algebraic Equations—I

We will look at two methods for solving sets of linear algebraic equations using a spreadsheet. The first method is pretty transparent and kind of fun to mess around with, but it doesn't work for all sets of equations. (Actually, it can be made to work for all sets of equations which have a unique solution, but doing so can get kind of messy.) Development of the method will give you some practice using matrix notation and algebra, and explaining to Quattro Pro how to carry it out will illustrate some other spreadsheet programming techniques. The name of the method is *Gauss-Seidel iteration*, and it was developed by in the early 1800's.

Suppose we have a set of equations such as Eq. (9−10) to solve for the vector **x**. Written out, the equations look like Eq. (9−1). We start by guessing a solution, let's call it $\mathbf{x}^{(0)}$. This guess will almost certainly be incorrect, so when we multiply $\mathbf{x}^{(0)}$ on the left by **A** we will get not **b**, as required by Eq. (9−10), but some other vector, say $\mathbf{b}^{(0)}$. The idea now is to dream up some method for generating a new guess, call it $\mathbf{x}^{(1)}$, which is closer to the true solution in that $\mathbf{b}^{(1)}$, which is generated by multiplying $\mathbf{x}^{(1)}$ on the left by **A**, is closer to **b** than was $\mathbf{b}^{(0)}$. Once we have such a method, we can then treat $\mathbf{x}^{(1)}$ as a guess to be improved and use our method on it to generate $\mathbf{x}^{(2)}$. We can continue this process, generating in turn $\mathbf{x}^{(3)}$, $\mathbf{x}^{(4)}$, ... until we obtain a guess, $\mathbf{x}^{(n)}$, for which the difference between $\mathbf{b}^{(n)}$ and **b** is sufficiently small.

Here's what Gauss and Seidel came up with. It doesn't always work; sometimes the "improved" guess is worse than the original. In each of the separate equations in Eqs. (9−1) (or represented by Eq. (9−10)) solve for a different unknown. Let's solve the first for x_1, the second for x_2, and so forth. The ith equation would then give

$$\begin{aligned} x_i &= \frac{1}{a_{i,i}}\left(b_i - a_{i,1}x_1 - a_{i,2}x_2 - \cdots - a_{i,i-1}x_{i-1} - a_{i,i+1,}x_{i+1} - \cdots - a_{i,N}x_N\right) \\ &= \frac{1}{a_{i,i}}\left(b_i - \sum_{\substack{k=1 \\ k\neq i}}^{N} a_{i,k}x_k\right) \end{aligned} \tag{9−12}$$

Gauss-Seidel iteration uses these equations to generate the "improved" guesses by using the old guesses to evaluate the right hand sides of the equations, and calling the left hand side the "improved" guess.

$$x_i^{(n+1)} = \frac{1}{a_{i,i}}\left(b_i - \sum_{\substack{k=1 \\ k\neq i}}^{N} a_{i,k}x_k^{(n)}\right) \tag{9−13}$$

Note the difference in the superscripts of the x's on the two sides of the equal sign.

That's it, we only need to clean things up a little before implementing the method on the spreadsheet. The requirement in the summation of Eq. (9−13) that $k \neq i$ is clumsy to program, so we add and subtract $\frac{a_{i,i}}{a_{i,i}}x_i^{(n)}$ to the right hand side, getting

$$x_i^{(n+1)} = x_i^{(n)} + \frac{1}{a_{i,i}}\left(b_i - \sum_{k=1}^{N} a_{i,k}x_k^{(n)}\right) \tag{9−14}$$

To use Quattro Pro to solve this set of equations, we consider a specific example. Let **A** and **b** be given by

$$\mathbf{A} = \begin{pmatrix} 4 & 2 & 1 \\ 1 & 5 & -1 \\ 1 & 1 & 8 \end{pmatrix} \qquad \text{and} \qquad \mathbf{b} = \begin{pmatrix} 14 \\ 10 \\ 20 \end{pmatrix} \tag{9−15}$$

Enter the matrix **A** in the block A6..C8, and the vector **b** in the block D6..D8, and label these blocks by putting A in B5, and b in D5. We'll use column E of the spreadsheet to hold the old guesses for x. Let's start with all 1's, so put a 1 in E6..E8, and label the column with x in E4.

In Eq. (9−14) on the right hand side, $x_i^{(n)}$ is the old guess, and the rest of the stuff is the correction to the old guess to make the new "improved" guess. The size of this correction is a measure of how far off we are. This might be useful information, so I'll put the correction by itself in one column, and generate the new guess by adding the old guess to the correction in another. I'll put the correction in column F, and the new, "improved" guess in E, both below the rows containing **A** and **b**. In cell F10 enter Delta x and in E10 enter New x. In cell F11 enter the formula

```
(+D6-@SUMPRODUCT(A6..C6,$E6..$E8))/A6
```

and copy this into F12..F12. Note that I used absolute addresses for the cells containing the x's because I don't want these to change when doing the copy, and relative addresses for the cells containing the elements of **A** because I do want these to change. But all is not bliss because the formula in F12 divides by A7, which corresponds to $a_{2,1}$, rather than B7, which corresponds to $a_{2,2}$. Similarly, the formula in F13 should divide by D8 rather than A8. I don't know any clever way of fixing this, except using the **Edit** key (**F2**), so use it to correct the formulae in F12..F13. Finally, the cells containing the new guesses are easy. In E11 just enter +E6+F11, and copy that into E12..E13.

If all is well, you should have a set of three numbers in E11..E13 which are closer to the real solution than your original guess in column E6..E8, but still not very accurate. To get the next guess you could copy the new values for **x** in E11..E13 into E6..E8 by hand and then let the spreadsheet recalculate. Better would be to use the copying capability of the spreadsheet, but be careful. If you just Copy you will copy the formulae, not the numbers they evaluate to. To copy just the numbers, use the Values item in the Edit menu. Do this procedure several times and you should see the new values coming closer to the true solution, as indicated by smaller corrections in the F column.

This copying procedure gets to be a drag, and it's prone to catastrophic error if you copy into the wrong column accidentally. Is there some way to perform the iteration automatically? Yes, there is. Suppose in the block E6..E8 we put not numbers, but formulae similar to those in column E11..E13. Then we would keep the stuff in column F the same, but change the contents of E6, for example, to +E6+F11. Doing that makes the formula in E6 is *circular*. That means that the value of the cell depends on the value of the cell. Consider the job of the spreadsheet in trying to keep the values of all cells accurately equal to the values of the formulae in them. To evaluate cell E6, it would take whatever value was there, add the value of F11, and then put that new value back in E6. But now the value of E6 would be changed, so the contents would be inaccurate. The spreadsheet would then have to recalculate the value, thereby changing it and again invalidating its just-completed calculation. Unless F11 contains zero, it's a hopeless situation and the conscientious spreadsheet is likely to undergo a nervous breakdown. Actually, the situation is even worse; the formula in E6 is doubly circular. Besides the direct circular dependence of E6 on itself, it also depends on F11, which in turn depends on several cells, including E6.

Fortunately, Quattro Pro is designed to deal with such situations. From the Options menu choose the Recalculation item, and from the resulting sub-menu choose Mode. Doing so provides you with three choices, Automatic, Manual, and Background. With both Automatic and Background, Quattro recalculates the worksheet every time a value is changed. If we are going to put in a circular formula, perhaps these might not be the best choices. (Actually, they are OK, but we'll get to that.) Choose Manual, and then check on the Recalculation sub-menu that the number of iterations is set equal to

1. The set value is that to the right of the `Iteration` item in the sub-menu. If it's not 1, choose this item and set it to 1. When that's finished, back out of the menus to get back to the worksheet. In this mode, Quattro Pro will only automatically recalculate the value of a cell you change from the keyboard. If you want it to recalculate the entire worksheet, press **F9**. Doing so causes Quattro Pro to recalculate the values of every cell in the worksheet once. Setting the number of iterations (via the `Recalculation` sub-menu) to some value greater than 1, say *n*, causes Quattro Pro to make *n* sweeps across the worksheet, recalculating cell values as it goes.

Now we're ready to put in the dreaded circular formula. In `E6` enter `+E6+F11`, and copy that into `E7..E8`. Press **F9** a few times, and watch the values of **x** in `E6..E8` converge, and the corrections in column `F` go to zero. Note that Quattro Pro detects the presence of a circular formula and puts the word `CIRC` at the bottom of the screen near the center. (In case you had not intended to create a circular formula, Quattro Pro will tell you which cell contains the culprit. Look at the item just above `Quit` in the `Recalculation` sub-menu.) Also, in `Manual` mode Quattro Pro tells you when something has changed so that the worksheet needs to be recalculated by displaying the word `CALC` in the status line at the bottom of the screen.

After playing with this worksheet a little, several improvements come to mind. First, it would be nice to be able to specify the initial guesses to be used. As things now stand, Quattro Pro starts with whatever values happen to be in cells `E6..E8`. I've got a bright idea for changing the worksheet to allow specification of an initial guess. Before I describe my idea, you should spend some time thinking about how you would solve the problem. There are probably several solutions, and the one I thought up may not be the best. I suggest that you pause here to see what solution you can come up with before your mind is polluted with my idea. Save your worksheet first so we can talk about my solution when you're done.

Here's the way I thought of to do it. Since I'm writing this I'm afraid you're stuck with my idea, no matter how good your ideas may have been. If you think you have an idea which is as good or better, bring it up in class for discussion.

We'll put the initial values in an unused block, and we'll use another unused cell to indicate to Quattro Pro whether we want to load the initial values, or to improve the previous values. With these changes, I decided to move things around a little. I'll put the initial guesses for **x** in the block `E6..E8`, and I'll put the "on switch" in `F6`. I'll put the current best estimate for **x** in `B11..B13`, and the change in **x**, `Delta x` in `B17..B19)`. In `E6..E8` enter whatever you like for initial guesses, and label the column by putting `Initial` in `E4`. We'll use a `0` in the switch cell, `F6`, to tell Quattro Pro that we want to load initial values, and a `1` to tell it to hit the juice. To implement this idea, in `B11` put `@IF($F$6=0,+E6,+B11+B17)`, and copy that into `B12..B13`.

While I'm at it, I decide to put in some improvements that help show how the solution is going. For my own edification, I'll put the product of **A** and this value of **x** in `D11.D13` so that I can see how well the current **x** comes to solving the equation. Again for my own edification, I decide to put the difference between **Ax** and **b** in `C17..C19`, and I'll keep track of the number of iterations in `D17`. In cell `D11` put the formula `@SUMPRODUCT(A6..C6,$B$11..$B$13)`, and copy it downward into `D12.D13`. That should put the product **Ax** using the current approximation to **x** into this block. In `C17..C19` I want to put the difference between **Ax** and **b**, which is a measure of how close I am to the correct solution. The cells `D11.D13` have the value of **Ax** in them, so enter the formula `+D6-D11` in `C17` and copy it into `C18.C19`. Label this block by putting `Error` in `C16`.

I still have to fill in the block containing the formulae for Delta x. I notice that the term inside the parentheses on the right hand side of Eq. 9−14 is just this error, so in `B17` I can just use `+C17/A6`. The formulae in `B18` and `B19` are then `+C18/B7` and `+C19/C8`. Label this column with `Delta x` in `B16`. For the iteration counter put the formula `@IF($F$6=0,0,+D17+1)` in `D17`. Fig. 9−1 shows the worksheet after these modifications. Part a) shows the formulae in the individual cells, and I had to expand the width of some columns and squeeze others in order to get enough space for some of the longer formula.

Part b) shows the worksheet as it actually appears on the computer screen.

a)

	A	B	C	D	E	F	G
1	Chapter 9: Gauss Seidel solution of a set						
2	of three equations in three unknowns.						
3							
4					Initial	Reset/	
5	A			b	x	Go	
6	4	2	1	14	1	0	
7	1	5	-1	10	1		
8	1	1	8	20	1		
9							
10		x					
11		@IF($F$6=0,+E6,+B11+B17)					
12	A	@IF($F$6=0,+E7,+B12+B18)	=	@SUMPRODUCT(A6..C6,$B$11..$B$13)			
13		@IF($F$6=0,+E8,+B13+B19)		@SUMPRODUCT(A7..C7,$B$11..$B$13)			
14				@SUMPRODUCT(A8..C8,$B$11..$B$13)			
15		Status Info.					
16		Delta x	Error	N. Its.			
17		+C17/A6	+D6-D11	@IF($F$6=0,0,+D17+1)			
18		+C18/B7	+D7-D12				
19		+C19/C8	+D8-D13				
20							

b)

	A	B	C	D	E	F	G
1	Chapter 9: Gauss Seidel solution of a set						
2	of three equations in three unknowns.						
3							
4					Initial	Reset/	
5	A			b	x	Go	
6	4	2	1	14	1	0	
7	1	5	-1	10	1		
8	1	1	8	20	1		
9							
10		x					
11		1		7			
12	A	1	=	5			
13		1		10			
14							
15		Status Info.					
16		Delta x	Error	N. Its.			
17		1.75	7	0			
18		1	5				
19		1.25	10				
20							

Figure 9−1. My Gauss-Seidel worksheet. Part a) shows the formulae in the individual cells, and part b) shows the worksheet as it appears on my computer screen. The column widths in a) are squeezed and expanded a little in order to have space for the longer formulae.

Put the value `0` in `F6` and press `F9`. Doing so should load your initial guesses from `E6..E8` into **x** in `B11..B13`. Now change the value in `F6` to 1 and press `F9` several more times. With each recalculation, the values of **x** in `B11..B13` should converge toward the correct solution, and the corrections in `B17..B19` and errors in `C17..C19` should approach zero. Once you are satisfied with how the

worksheet works, you might want to increase the number of iterations Quattro Pro performs when told to recalculate. I suggest a value between 5 and 10. Use the `Iteration` item of the `Recalculation` sub-menu.

Unfortunately, Gauss-Seidel iteration does not always work—for some matrices the new guesses are farther off than the previous ones. To illustrate the point change the values of **A** and **b** to

$$\mathbf{A} = \begin{pmatrix} -1 & 1 & 2 \\ 3 & -1 & 1 \\ -1 & 3 & 4 \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} 2 \\ 6 \\ 4 \end{pmatrix} \tag{9−16}$$

Load your initial conditions, and then press **F9** several times. It would probably be a good idea to set the number of iterations in the `Recalculation` menu back to 1. You should find that the corrections increase with each iteration. If you are concerned that you may have made an error in programming the worksheet, you can load as an initial guess the true solution of this set of equations, (1, −1, 2). With this initial guess, the corrections should all be zero, and subsequent recalculations should cause no change. Change the initial guess a little from the correct solution, however, and on each iteration the corrections take the new values farther and farther from the true solution.

It is possible to understand why Gauss-Seidel iteration is unstable for this matrix and stable for the previous matrix, but it ain't easy! Crudely, the problem is related to the relative magnitudes of the elements on the diagonal of **A** and those off the diagonal. For stability, one wants large diagonal elements and small off-diagonal elements. It in fact is possible to rewrite the equations represented by Eq. (9−16) in a form which is stable to Gauss-Seidel. To do so, we need to reorder both the rows and the columns of **A**. Changing the order of the rows of **A** is the same as changing the order in which we write the equations such as Eqs. (9−1) for which Eq. (9−10) stands (providing we move around the elements of **b** in the same way). Such a change should have no effect on the solution (except maybe for making the method stable or unstable). Interchanging columns is equivalent to changing the order in which the unknown elements in the vector **x** are written. For example, to make the present problem tractable with Gauss-Seidel, we need to interchange the first and second columns of **A**. Doing so will mean that the top element in **x** will now be x_2, and the second will be x_1. Then interchange the top and the bottom rows of **A**, and **b**. We now have **A′** and **b′** given by

$$\mathbf{A}' = \begin{pmatrix} 3 & -1 & 4 \\ -1 & 3 & 1 \\ 1 & -1 & 2 \end{pmatrix} \quad \text{and} \quad \mathbf{b}' = \begin{pmatrix} 4 \\ 6 \\ 2 \end{pmatrix} \tag{9−17}$$

Putting these matrices into our worksheet, and reloading some reasonable guess, we find that after 20-30 iterations the method converges to values for **x** = (–1, 1, 2). That is correct because interchanging the first two columns of **A** causes the first two elements of **x** to be interchanged.

Once you think you have the solution of the problem fairly well in hand, you might want to set the recalculation mode to `Background`. In this case the whole worksheet is updated automatically every time a value in a cell is changed. Quattro Pro will recalculate the worksheet, check to see if the resulting changes have made some cell values invalid (because of a circular reference), and if so recalculate again. This procedure will continue over and over until either nothing changes or the number of iterations exceeds the limit specified by the `Iteration` item of the `Recalculation` sub-menu. Change the recalculation mode to `Background`, and set the maximum number of iterations to `10`. Return to the worksheet and enter `0` into cell `F6`. The initial values should be immediately loaded into `B11..B13`. Now enter `1` in `F6`, and watch the values change. To initiate 10 more iterations you can either enter `1` in `F6` again, or press **F9**.

Before continuing, I have to correct one small detail—the method we started programming was, in fact the Gauss-Seidel method, but when we made the worksheet modification involving the circular reference, the method became the Jacobi method. This really isn't a very important point, but I'm trying to be accurate here. Consider evaluating Eq. (9−12) for $i = 1$, then $i = 2$, then 3, and so forth. For $i = 1$, in

evaluating the right-hand side, we would use the values of x obtained from the last iteration (or the guessed values if this is the first time through). For $i = 2$, however, we have two reasonable alternatives for x_1 on the right side.

1. We could use the old value from the previous iteration, or
2. We could use the value we just calculated.

Similarly, when calculating the new x_3 we could use old values for x_1 and x_2, or the ones just calculated, and so forth. If we always use the values from the previous iteration in evaluating the right hand side of Eq. (9−12), the method is called Gauss-Seidel. If, on the other hand, we use the most up-to-date value we have, it is called the Jacobi method.

If you think about how the first program worked, the method was Gauss-Seidel, as advertised, because we used values of the x's in one column to generate new values, which we put in another column. Only when we were all done, did we move the new values into the column used for the old values. On the other hand, when we modified the worksheet to update the old values automatically (and thereby created circular references), the worksheet was always using the most recent values it had for the x's, and the method became Jacobi. The characteristics of the two methods are similar, but the Jacobi method usually converges a little faster, and is often easier to program.

9.6 Using Quattro Pro to Solve Sets of Linear Algebraic Equations—II

There are better ways such as *Gaussian elimination* and *LU decomposition* to solve sets of linear algebraic equations than Gauss-Seidel iteration. These methods are generally faster, and work in almost all cases. They are difficult to program, however, with a spreadsheet program such as Quattro Pro. More conventional programming languages are much more suitable for these cases. Quattro Pro does have the built-in capability, though, to invert square matrices. There is also a built-in matrix multiplication capability. To use either capability, choose the `Advanced Math` item from the `Tools` menu. From this point on, it should be pretty clear how to proceed. You can use the `Invert` item to calculate an inverse of **A**, and then use the `Multiply` item to multiply the inverse times the vector **b**. In general, this is the recommended way to solve a set of linear algebraic equations such as Eq. (9−1) using Quattro Pro.

9.7 Exercises

1. Let

$$\mathbf{A} = \begin{pmatrix} 1 & -2 & -3 \\ -1 & 2 & 1 \\ 2 & 3 & 2 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 3 & -2 & -3 \\ -1 & 1 & 2 \\ 2 & -1 & -1 \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix}$$

   Calculate by hand

   a. $\mathbf{A} + \mathbf{B}$

   b. $2\,\mathbf{A}$

   c. $\mathbf{A}\,\mathbf{x}$ (Show your arithmetic)

   d. $\mathbf{A}\,\mathbf{B}$ (Show your arithmetic for at least three elements.)

2. Make a worksheet which adds two 3×3 matrices. You should be able to put the two matrices to be added in two 3×3 blocks of cells in the worksheet, and have the result appear in a third 3×3 block.

3. Consider the following set of three simultaneous algebraic equations in three unknowns.

$$\begin{aligned} 5r + 3s - 3t &= -1 \\ 3r + 2s - 2t &= -1 \\ 2r - s + 2t &= 8 \end{aligned}$$

   a. This set of equations can be written in matrix form as

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

   where

$$\mathbf{x} = \begin{pmatrix} r \\ s \\ t \end{pmatrix}$$

   What are $\mathbf{A}$ and $\mathbf{b}$?

   b. The inverse of $\mathbf{A}$ above is

$$\mathbf{A}^{-1} = \begin{pmatrix} 2 & -3 & 0 \\ -10 & 16 & 1 \\ -7 & 11 & 1 \end{pmatrix}$$

   Making use of this inverse, find the solution to the set of equations for r, s, and t. Do the arithmetic by hand, and show your work.

4. A *permutation matrix* is a square matrix which can be used to move the components of a vector matrix around. A row of a permutation matrix consists of elements which are all zeroes, except for a single element which is a one. The position of the single one is different in each row.

   a. Consider the permutation matrix, $\mathbf{P}$, below and show that for any vector matrix consisting of 4 elements, the effect of multiplying the vector by $\mathbf{P}$ is just to permute the components of the vector. How does $\mathbf{P}$ permute the components of the vector? (I.e. into what positions are the first, second, third, and fourth components put?)

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

b. Write a permutation matrix for 4-component vectors which accomplishes the following permutation

$$1 \rightarrow 4$$
$$2 \rightarrow 3$$
$$3 \rightarrow 1$$
$$4 \rightarrow 2$$

In other words, the first element of a vector is moved to the fourth position, etc.

5. Permutations can be used to encrypt text. The idea is the following. Both the sender and receiver of the message must have agreed previously on a permutation matrix. Say the matrix has dimension $N \times N$. Then the message text is broken down into consecutive sub-strings, each of length N characters, and the permutation is applied to each of these sub-strings separately. The result is then sent as the encrypted message. At the receiving end, the same procedure is applied, except the reverse permutation is used.

A spreadsheet can be used to carry out most of the drudgery of the encryption and de-encryption. Put the permutation matrix, $\mathbf{P}$, in a block of a worksheet, and put the message to be encoded into a separate column, with one character in each cell. You can't just multiply $\mathbf{P}$ by vectors made up of characters because you can't do arithmetic on text. Instead, you can use the `@CODE()` built-in function to obtain the ASCII code for each character, and put the result in a separate column. You can then use the matrix multiplication capability of Quattro Pro to multiply the permutation matrix by parts of this column. Put the results in still another column, and use the `@CHAR()` built-in function to convert the ASCII codes back to characters.

To decode the message follow the same procedure, except use a permutation matrix that just undoes what $\mathbf{P}$ does. This matrix is just the inverse of $\mathbf{P}$, $\mathbf{P}^{-1}$. You can use the matrix inversion capability of Quattro Pro to generate that matrix. Actually, it is easy to determine the inverse of $\mathbf{P}$ without using Quattro Pro. $\mathbf{P}^{-1}$ must undo what $\mathbf{P}$ does, so given $\mathbf{P}$, what matrix causes the reverse permutation?

Use this procedure to encode the following message (without the quotes) "My name is *your name*." where you are to replace *your name* with your name. (For example, if I were doing it, the message would be "My name is Frazer Williams.") When that's done, decode the message to show that the technique works. Use the following permutation matrix.

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Carry out the decoding process on the encrypted message to show that the whole thing works.

6. Three Hooterville grocery stores have different prices for five produce items as shown in Table 9−1. You want to buy 1.5 pounds of apples, 2.5 of oranges, 3 of tomatoes, 1.25 of cucumbers, and 4.5 of cabbage, and you want to buy it all at one store. Use the matrix multiplication capability of Quattro Pro to determine the total cost at each store so that you can decide which store would be the cheapest. Enter these price data as a 3×5 matrix into a worksheet, and the quantities of produce into a 5×1 vector. Use matrix multiplication to obtain the answer as a 3×1 vector, with elements being the total cost at each store.

PRODUCE PRICES (Per Pound)					
Store	**Apples**	**Oranges**	**Tomatoes**	**Cucumbers**	**Cabbage**
Acme	0.23	0.25	0.49	0.38	0.29
Billy Bob's	0.25	0.21	0.41	0.41	0.26
Joe's Market	0.21	0.28	0.47	0.45	0.33

TABLE 9−1. Prices per pound for selected produce items at three Hooterville grocery stores.

10. SIMULATIONS AND MODELING

10.1 Introduction

Computer programs are often used to simulate real or imaginary situations. Simulations are used when it is inconvenient to test a new idea or design experimentally. For example an engineer might resort to a simulation to test the design of a new bridge. Building the bridge and then seeing whether or not it falls down is just not practical. Electrical engineers often simulate circuits before building them. Once the circuit is simulated, it is easy to vary the values of components and note the change in performance. Further, if the actual circuit doesn't function properly it is difficult to determine if the reason is a flaw in the basic design, an error in wiring, or some other effect such as parasitic capacitance not considered in making the design. In the simulation there are no wiring errors, although mathematical and programming errors can be troublesome, and there are no active effects other than those put into the simulation. Comparison of the behavior of the actual and the simulated circuits can often help determine where the problem lies.

To make a simulation, one must know the basic laws governing the situation in question. The fidelity of the simulation to reality can be no better than the accuracy of these laws in describing the behavior of the components involved. If the situation is not very complicated and the mathematics simple enough algebraic manipulation, ingenuity, and common sense are adequate to determine what the laws predict. In this case an *analytic solution* is said to be available. A computer may be helpful in evaluating the resulting algebraic formulae, but it is not really a part of the solution process. On the other hand, if the situation under study is complex or the mathematics required to solve the governing equations difficult it may not be possible to derive an analytic solution. In this case, the consequences of the laws are often determined by programming a computer to simulate the operation of the laws. Such a solution is called a *numerical solution*.

The calculations of current and voltage in Chapter 8 were all simulations of what the corresponding real circuit would do. (Actually, they are more commonly called a *model* of the actual circuit. The distinction between a simulation and a model is a fine one, and I'll not discuss it further here primarily because I don't understand it very well myself.) They were based on a "law" called Ohm's law and on some common sense about the net current into a node. Assuming we make no algebra or arithmetic mistakes, our predictions will describe the behavior of a real circuit just as well as the laws and common sense we used describe the behavior of the individual components. For example, if the voltage source in Fig. 8−5 corresponds in the real circuit to a partly discharged battery, the voltage it delivers could be quite a bit less than the 3 V shown in the diagram and, even worse, it might decrease in time as the battery discharges. None of this behavior was put into the laws supposedly describing the source, and our simulation would therefore fail to describe what happened in the actual circuit.

Besides errors arising from the use of an inadequate set of laws, errors can arise in working out what the consequences of these laws are. As problems become more complex or the mathematical manipulations required more difficult, it is often necessary to make approximations. These approximations will introduce some level of error into the simulation, and it is quite an art to find a set of approximations which make the problem tractable, yet introduce a suitably small error. Such approximations may involve leaving out part of the problem which is thought to contribute only slightly to the outcome, or using a form of one or more laws which is easier to handle, but less accurate than the correct one.

When using a computer to help with the simulation other types of errors arise. Often laws are known in the form of differential equations—think of Newton's law for example. A computer is very good at doing arithmetic, but very bad at solving differential equations directly, so that when faced with trying to solve such equations on a computer one usually tries to approximate them by a set of algebraic equations which the computer can solve more readily. Such approximations can be quite tricky, and are frequently the source of real difficulties.

In this chapter we will look at a different type of simulation. We will consider several situations in which the outcome is determined by the laws of chance. We will look specifically at three problems. All can be figured out without simulating them on a computer. As we discuss the three problems, notice the differences in utility between the simulation results and the analytic solution. The first problem is pretty simple, and for it I discuss simulation only to introduce the ideas we will use. The last two are not so easy, and it might make sense to run a computer simulation just to check our reasoning in developing the analytic solution.

At the base of all simulations is the generation of random numbers, a task at which a computer is notoriously ill-suited. Given the same instructions it does the same thing each time unless some component fails, in which case it freaks out completely. Quattro Pro has a built-in function which uses a complicated set of arithmetic operations to produce numbers claimed to be random. It works pretty well, but the numbers are not truly random, and the lack of perfect randomness may be a major source of error in such simulations.

Another, expected, source of error results from the random nature of the simulation. Every time we run the simulation, we can expect to get a different result. Which result should we use? Because of this question, it is usually necessary to run such simulations a large number of times and average the results.

10.2 Simulating Flipping a Coin

We consider first simulating flipping a coin. This example is sufficiently easy that it would make little sense to use a computer to simulate it if our only concern were the answer. The example illustrates, however, several ideas I will want to use later in more involved problems. This is the problem:

> If you flip an un-biased coin three times, what is the probability that you will get all heads?

You probably already know the answer to the question, $P(3) = (½)^3 = (2)^{-3} = 0.125$, but let's simulate the situation using Quattro Pro anyway.

10.2.1 The Plan of Attack.

We need a plan of action, a sequence of operations which Quattro Pro knows how to execute which will provide an answer to our question. Here's my plan. We'll put the result of each simulated coin flip in separate cells of a worksheet. The coin flip will be simulated by dreaming up a formula which evaluates to 0 half the time and 1 the other half, and associating 1 with heads and 0 with tails. We will put the formula in three adjacent cells in one row, to simulate three consecutive coin flips. In a fourth cell adjacent to these three we can evaluate how many heads we got by using the `@SUM` built-in function to add up the values in the three coin flip cells. We will copy these four cells many times, going down the spreadsheet, and we will determine how many times three heads appeared by counting the number of 3's in the fourth column. Dividing by the number of such "experiments" will give us the probability we seek.

First we need a formula to simulate a coin flip, a formula which gives either 0 or 1 with equal probability. We will base our simulation on the `@RAND` built-in function. This function returns a number between 0 and 1, with all values being (it is claimed) equally likely. We want either 0 or 1, and nothing in between, so we use the formula `@IF(@RAND<.5,0,1)`. If `RAND` returns a value less than 0.5, the formula has value 0, otherwise 1. Assuming the values `RAND` returns are uniformly distributed over the range [0, 1], and that the value returned is truly random, this should provide a good simulation of a coin flip. (As an aside, we could have also used the formula `@INT(2*@RAND)`. The built-in function `@INT()` returns the integer part of it's argument, and `2*@RAND` returns a number between 0 and 2.)

10.2.2 The Implementation.

To implement the simulator, put the formula simulating a coin flip in cell A15, and copy this cell into B15..C15. Label what you've done by entering Flip in A13, and #1 in A16, and put similar labels at the tops of columns B and C. In column D we will count the number of heads (1's). Enter the formula @SUM(A15.C15) in D15, and label the column with No. Heads at the top. In column E we'll put a 1 if we got three heads, and 0 otherwise. In E15 put the formula @IF(D15=3,1,0). This row of formulas simulates three coin flips, and Quattro Pro places a 1 in E15 if all three were heads, and a 0 otherwise. This is a simulation of just one set of three flips. We need to simulate a large number of such flips to get decent statistics. How many? I choose 500, so copy the block A15..E15 into A16..E514.

We need to put a report of the results somewhere. I purposely left room above the coin flip rows, so I'll put the report there. In A11 I'll put the total number of experiments (each consisting of three flips), in B11) I'll put the total number of experiments which yielded three heads, and in C11 I'll put the percentage of the experiments that yielded three heads. First put in the labels. In A9 and A10 put No. and Expts., in B9 and B10 put No. All and Heads, and in C9 and C10 put % All and Heads. We'll use the @COUNT function to count the number of experiments so in A11 put @COUNT(A15..A900). Put the number of all-heads experiments in B11 by putting the formula @SUM(E15..E900) there, and the percentage all heads in C11 with +B11/A11. If you really want to be fancy, you could change the format of this cell (via the Numeric Format item of the Style menu) to display as a percent.

Pressing the Calc key (**F9**) will make Quattro Pro redo the whole experiment. Doing so several times, I get answers ranging between about 10% and 14% for the percentage of experiments for which all three flips gave heads. Apparently we need to run more experiments to get better statistics. It would be nice if we could just press **F9** to make Quattro Pro recalculate the worksheet, and keep a running total of the number of experiments and the number of flips. My previous experience with the Gauss-Seidel worksheet gives me an idea about how to do that. I'll add three more cells in the report area (row 11) containing the cumulative results. I'll use a Reset/Go switch like in the Gauss-Seidel worksheet to tell Quattro Pro to either reset the cumulative totals to zero, or to accumulate. The addition is pretty straight-forward, and the complete worksheet is shown in Fig. 10−1. Part b) of the figure shows the cumulative result of running the simulation 140 times, corresponding to 70,000 individual experiments. The cumulative percentage of all heads is 12.66%, pretty close to the expected 12.5%. We should not expect to get exactly 12.5% because of statistical variations. It is difficult to tell whether this value is within the likely range of these variations or not, but this result would be close enough for most applications.

10.2.3 Discussion of Errors.

Before continuing, I'd like to discuss again the kinds of errors in this simulation.

1. We have assumed that in each flip of the coin heads and tails are equally likely. This may not be true, in which case even if the computer carries out its part perfectly, our simulation will not correctly predict what really happens. In this case the error would be due to the inaccuracy of the laws we assumed govern the coin flip.

2. By it's very nature, the process is a statistical one, and there will be considerable variation from experiment to experiment. Each experiment yields either a 0 or a 1 in column E, never 0.125. Only after a large number of experiments should the percentage of three-heads results approach 12.5%. Error due to this source would be called statistical error. It can be reduced by increasing the number of experiments.

3. The random number generator Quattro Pro uses may not be completely random. The computer cannot do what we want—generate a number randomly. Instead, each time the worksheet calls the @RAND function, Quattro Pro runs an internal subroutine that predictably (i.e. not randomly) generates a number which is supposed to appear random. It is possible that the technique used in @RAND generates numbers which in our application are not sufficiently random. In this case our results will

a)

	A	B	C	D	E	F	G
1	Chapter 10: Simulation of three coin flips.						
2							
3							
4			Reset/				
5			Go				
6			1				
7							
8		THIS TIME			CUMULATIVE		
9		No.	No. All	% All	No.	No. All	% All
10		Flips	Heads	Heads	Flips	Heads	Heads
11		@COUNT(A15..A900)	@SUM(E15..E900)	+B11/A11	@IF($C$6=0,0,D11+A11)	@IF($C$6=0,0,E11+B11)	+E11/D11
12							
13		Flip	Flip	Flip	No.	All	
14		#1	#2	#3	Heads	Heads?	
15		@IF(@RAND<0.5,0,1)	@IF(@RAND<0.5,0,1)	@IF(@RAND<0.5,0,1)	@SUM(A15..C15)	@IF(D15=3,1,0)	
16		@IF(@RAND<0.5,0,1)	@IF(@RAND<0.5,0,1)	@IF(@RAND<0.5,0,1)	@SUM(A16..C16)	@IF(D16=3,1,0)	

b)

	A	B	C	D	E	F	G
1	Chapter 10: Simulation of three coin flips.						
2							
3							
4			Reset/				
5			Go				
6			1				
7							
8		THIS TIME			CUMULATIVE		
9		No.	No. All	% All	No.	No. All	% All
10		Flips	Heads	Heads	Flips	Heads	Heads
11		500	66	13.20%	70000	8860	12.66%
12							
13		Flip	Flip	Flip	No.	All	
14		#1	#2	#3	Heads	Heads?	
15		0	1	0	1	0	
16		0	0	1	1	0	
17		1	0	1	2	0	
18		1	0	1	2	0	

Figure 10−1. Worksheet for simulating a set of three coin flips. Part a) shows the formulae in the individual cells, and b) shows the worksheet as it appears on the screen.

be in error due to an inadequacy of the random number generator sub-program of Quattro Pro. This is a type of numerical error.

10.2.4 The Analytic Solution

For this problem an analytic solution is pretty easy to derive. I'll generalize a little, and determine the probability of finding all heads if I flip the coin n times. For each flip of the coin there are two possible outcomes, both equally likely. If I flip the coin twice there are $2 \cdot 2 = 2^2$ possible outcomes, each equally

likely. Each time I flip the coin, I multiply the number of possible outcomes for the set of flips by two. For n flips there would be 2^n possible outcomes. Of all these outcomes, only one corresponds to all heads, so the probability of all heads in n flips is $P(n) = 1/(2^n) = 2^{-n}$. To determine the probability of 5 heads with a worksheet simulation, we would have to modify the worksheet to simulate experiments with 5 flips. With the analytic solution we need only plug $n = 5$ into $P(n)$. Clearly the analytic solution is better in this case. The difference between the two methods becomes even greater if we want to know how the probability varies with n. With only the numerical simulation we would have to modify the worksheet several times to work for different values of n, run the simulation for each value of n enough times to get good statistics, and then try to interpolate and extrapolate to other values. With the analytic solution, the answer to the question is clear; it varies as 2^{-n}. When an analytic solution is available it is almost always preferable to a numerical solution. When the problem is sufficiently difficult that you cannot derive an analytic solution to it, it is necessary either to make enough approximations that you can, or to resort to a numerical solution. If the problem is solved by making some questionable approximations and then proceeding analytically, a numerical solution may still be useful to check on the accuracy of the approximations.

10.3 Birthday Coincidences

We next consider the following question.

> Given a class of 30 people, what is the probability that two or more people in the class will have the same birthday?

This problem also has an analytic solution, but it's not so easy to figure out. I'll derive it at the end of this section. Instead of using the analytic solution, we'll use Quattro Pro to simulate a class, and count how many students have the same birthday. Doing that several times should give us a pretty good idea what the answer is. As usual, the analytic solution is preferable to the computer simulation, but in this case it makes a little more sense to develop the simulation just to check our reasoning in deriving the analytic solution—its a little tricky.

10.3.1 The Simulation.

Getting Quattro Pro to do what we want is not so easy. Here's what I came up with. There are probably other ways to do it, and some may be significantly better. If you think you have a better way you're probably right, so bring it up in class for discussion. In column A I'll put the birthdays, given as an integer between 1 and 365. (I neglect the possibility of someone having a birthday on February 29 of a leap year.) To generate the birthdays, I'll use @RAND again. In cell A16 put the formula @INT(365*@RAND)+1, and copy that into the range A17..A45. We also need a way of identifying the students. Instead of using names, I'll just give each a number by putting 1 in cell B16, the formula +B16+1 in B17, and copying that into cells B18..B45. These numbers will then serve as "names" for the students.

We now have the birthdays in column A, and the student "names" in column B. The problem is to see if any of the birthdays are the same. Doing that by eye is difficult. Instead, I'll use the @VLOOKUP built-in function. I included this function in Table 7−1, and use of the function was the center-piece of exercise 7−2. Briefly, the function is used to find an item in a table. The function requires three arguments, @VLOOKUP(*X*,*Block*,*Column*). Here *X* is the item to be found in a table and *Block* is the range of cells containing the table. The *Column* argument specifies what VLOOKUP returns once it has found *X*. If *Column* is zero, VLOOKUP returns the contents of the cell which it has found. If *Column* is greater than zero, VLOOKUP returns the contents of the cell in the same row as the found cell, but *Column* columns to the right of it.

Here's the idea. The table I'll search is that in the range A16..B45, and contains birthdays and student "names". For each row, I'll search the whole table for the birthday of the student in that row and have VLOOKUP return the "name" it finds. If the birthday it's searching for has not occurred previously in the

table, VLOOKUP should find this student first, but if the birthday has already occurred, VLOOKUP should find the earlier occurrence. We can determine which happened from the "name" VLOOKUP returns. If the "name" returned is the same as the "name" of the student whose birthday VLOOKUP searched for, then the birthday has not yet occurred in the table and is so far unique; whereas if the "name" returned is different then the birthday has already occurred at least once and we have a match.

In putting this into the worksheet, I encountered a snag. For VLOOKUP to work the entries in the table have to be sorted into ascending order. OK, that's not big deal, I can just use the Sort item in the Database menu. Selecting the Sort item produces a new menu. From this one, you have to tell Quattro Pro the block you want sorted (Block), and the column to sort on (1st Key). In our case the range for Block is A16..B45, and for 1st Key the range is A16..A45. When you are ready, select the Go item, and the sort should occur.

Here's where I hit the snag. After moving the cells around to sort them, Quattro Pro then automatically recalculates the contents of each cell. Those in column A contain a call to the RAND function, and when these are reevaluated they return a different number, causing the birthdays to become unsorted again. ARGHH!

What to do? The solution I came up with involves copying by value (use the Value item of the Edit menu) the table into two unused columns, say C and D. Then we can sort this table without fear of the values in the cells changing (because the contents are values, not formulae) and then use the VLOOKUP function on this table.

To implement this idea, copy by value the range A16..B45 to C16..D45. Then sort the range C16..D45 by the first column into ascending order. In cell E16 enter the formula

```
@IF(@VLOOKUP(C16,$C$16..$D$45,1)=D16,0,1)
```

and copy that into E17..E45. To check if there were any birthday coincidences, enter the formula @SUM(E16..E45) into some unused cell. It would also be a good idea to put labels on the columns, and on this cell.

The worksheet made up in this way works, but we will want to carry out the calculation several times to get good statistics and the process of copying by value the birthday table and then sorting the copy is tiresome and prone to errors. This process can be automated by writing a macro. For this I suggest using the Record feature of the Macro sub-menu of the Tools menu. Choosing this item turns the feature on, and then whatever keys you press will be stored in "macro-ese", and can be replayed as a macro. Once Record is turned on, copy by value the table in A16..B45 to C16..D45 and sort the copy. Then turn Record off by again choosing the Record item of the Macro sub-menu. To save the macro you just recorded use the Paste item of the Macro sub-menu to put the macro into some convenient, unused part of the worksheet. I suggest naming the macro with a two character name consisting of a back slash, followed by a single letter, so that you can execute the macro by simultaneously pressing **Alt** and the letter. Fig. 10−2 shows the worksheet that I programmed. I changed the macro created by the Record operation a little so that it would print out better.

If you run this macro a number of times, you should find that there is usually one pair of students with the same birthdays, but often you should also find none or two. I won't carry the example any farther, but to get an estimate of the probability of one or more students with the same birthdays you would have to run the simulation many times, counting the number of times when there are one or more coincidences. You can do this either by hand, or by using a circular reference as we did in Section 9−5. Dividing by the number of times the simulation was run would give the probability of two or more students having the same birthday.

a)

	A	B	C	D	E	F	G
1	Chapter 10: Simulation of the birthdays						
2	of a class of 35 students.						
3							
4	No.						
5	Matches						
6	@SUM(E16..E45)						
7							
8	Macro to simulate one class						
9	\A {GOTO}a16~						
10	{/ Block;Values}{END}{DOWN}{RIGHT}~C16~						
11	{RIGHT 2}						
12	{/ Sort;Go}						
13							
14	FORMULA		VALUE				
15	B'day	Name	B'day	Name	Match?		
16	1+@INT(365*@RAND)	1	5	23	@IF(@VLOOKUP(C16,$C$16..$D$45,1)=D16,0,1)		
17	1+@INT(365*@RAND)	+B16+1	59	7	@IF(@VLOOKUP(C17,$C$16..$D$45,1)=D17,0,1)		

b)

	A	B	C	D	E	F	G
1	Chapter 10: Simulation of the birthdays						
2	of a class of 35 students.						
3							
4	No.						
5	Matches						
6	1						
7							
8	Macro to simulate one class						
9	\A {GOTO}a16~						
10	{/ Block;Values}{END}{DOWN}{RIGHT}~C16~						
11	{RIGHT 2}						
12	{/ Sort;Go}						
13							
14	FORMULA		VALUE				
15	B'day	Name	B'day	Name	Match?		
16	171	1	5	23	0		
17	343	2	59	7	0		
18	111	3	78	9	0		
19	101	4	89	26	0		
20	290	5	90	19	0		

Figure 10−2. Worksheet to determine the number of birthday coincidences in a class of 30 students. Part a) shows the formulae in the individual cells, and b) shows the worksheet as it appeared on the computer screen.

10.3.2 The Analytic Solution.

At the start of this section I mentioned that it is possible to solve this problem analytically, and I would like to do that here. The trick is to calculate not the probability that two or more students will have the same birthday, but the opposite, the probability that no students will have the same birthday. The probability we want then is one minus this probability. Let n be the number of students in the class. We assume that all possible sets of birthdays for the students are equally probable. I'll count the number of ways in

which these birthdays could be chosen under the constraint that no students have the same birthday, and divide that by the total number of ways in which the birthdays could be chosen.

Figuring out the total number of ways the birthdays could be chosen is the easiest, so I'll do that first. For one student we could choose any of 365 birthdays (ignoring the possibility of leap day birthdays). For two students, there are a total of $365 \times 365 = 365^2$ ways in which the birthdays could be chosen. (For every choice for the first student there are 365 choices for the second.) For three students there will be 365^3, and so forth. For n students these are 365^n possible sets of birthdays.

How many sets of birthdays are there such that no two students have the same birthday? Consider choosing the birthdays of each of the students in order, subject to the constraint. The first student can have any of 365 birthdays. The second cannot have the birthday chosen for the first, so there are only 364 possibilities. For the third there are $363 = 365 - 3 + 1$, and so forth up to the n^{th} student, for whom there are $365 - n + 1$ possibilities. The total number of ways in which these birthdays can be chosen, $N_{all\ diff}$, is therefore

$$N_{all\ diff}(n) = 365 \cdot 364 \cdot \cdots \cdot (365 - n + 2) \cdot (365 - n + 1) \tag{10−1}$$

The probability of all different birthdays is this value divided by 365^n,

$$P_{all\ diff}(n) = \frac{365 \cdot 364 \cdot \cdots \cdot (365 - n + 2) \cdot (365 - n + 1)}{365 \cdot 365 \cdot \cdots \cdot 365 \cdot 365} \tag{10−2}$$

Finally, the probability that two or more students have the same birthday is just 1 minus the probability that no students have the same birthday,

$$P(n) = 1 - P_{all\ diff}(n) \tag{10−3}$$

How does $P(n)$ depend on n? How large a class is required to have a 50-50 chance of two or more students having the same birthday? The necessary information is contained in Eqs. 10−3 and 10−2, but they are not exactly transparent. I suggest that we should use Quattro Pro to evaluate Eq. 10−3 for a range of values of n. Eq. 10−2 is not in the most convenient form for evaluation on a spreadsheet. There are n terms in both the numerator and denominator of Eq. 10−2, so we can write it as

$$P_{all\ diff}(n) = \frac{365}{365} \cdot \frac{364}{365} \cdot \cdots \frac{365 - n + 2}{365} \cdot \frac{365 - n + 1}{365} \tag{10−4}$$

That's a little better. In each row of the worksheet, we can evaluate a different term on the right hand side of Eq. 10−4, and keep a running product. To do so move to an unused part of the worksheet, say `A60`. We'll put values of n in column `A`. Enter `1` in `A60`, and `+A60+1` in `A61`. We'll put the terms on the right hand side of Eq. 10−4 in column `B`, starting in row `60`. In `B60` enter `(366-A60)/365`, and copy it into `B61`. In column `C` we will keep the cumulative product of terms in column `B` as required in Eq. (10−4). Enter `+B60` into `C60`, and `+C60*B61` into `C61`. Finally, in column `D` we will put $P(n)$ as per Eq. (10−3). Enter `1-C60` into `D60`, and copy it into `D61`.

We will calculate $P(n)$ for values of n up to 50, so copy `A61..D61` into `A62..D109`. Column `C` will now contain $P_{all\ diff}(n)$, and column `D` the values of $P(n)$ for the values of n given in column `A`. I suggest plotting both $P_{all\ diff}(n)$ and $P(n)$ using the graphing capability of Quattro Pro. Fig. 10−3 shows such a plot. At between 22 and 23 students the probability of having at least one coincidence of birthdays exceeds that of not having a coincidence. For a class of 30 students we should find one or more coincidences about 70% of the time.

10.4 Strategy for Let's Make a Deal

There was (maybe still is) a long-running television game show titled *Let's Make a Deal*, hosted (I think) by Monte Hall. The rules of the game were as follows. A contestant was shown three curtains. Behind one curtain there was a good prize, and behind the other two were bad prizes (a year's supply of stale pretzels or something). The contestant was asked to choose a curtain. After doing so, he/she was

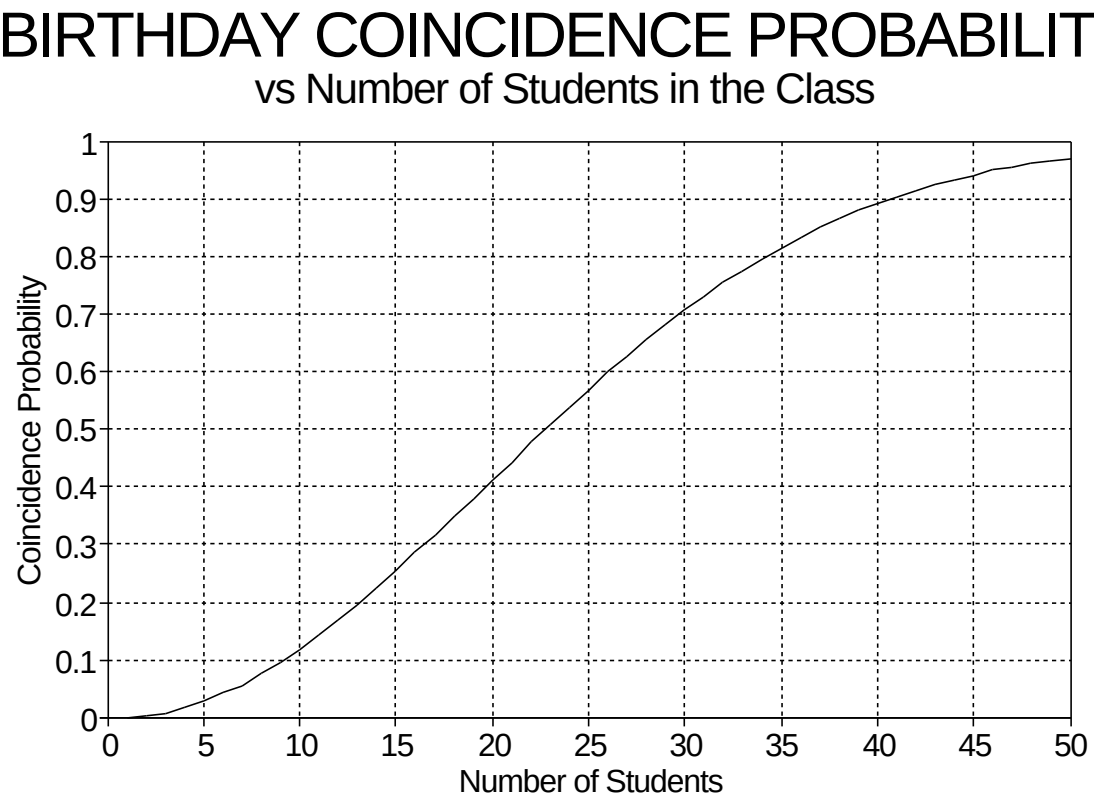

Figure 10−3. Graph showing the dependence of the probability that two or more students in a class will have the same birthday on the size of the class.

shown what was behind another curtain (one other than the one chosen). This curtain always covered a bad prize. The contestant was then given the chance to change his/her choice. Finally, after a lot of hand wringing and audience participation, the contestant won whatever was behind the curtain he/she ended up choosing. The question is:

> Is it better strategy to keep the first choice, or to change choices when given the chance? (Or, does it matter?)

10.4.1 A Plan for Answering the Question

It is possible to derive an analytic solution to this problem. Most people (including me) get the wrong answer the first time. I propose to answer the question by using Quattro Pro to simulate the game. Using the computer we will mimic playing the game many times using each strategy, and then we will see which one wins more often.

We will use the `@RAND` function to choose randomly which curtain will hide the good prize, and which curtain the contestant chooses initially. Quattro Pro will decide which curtain the contestant will be shown (possibly using `@RAND` again), and will score both strategies by checking whether the curtain finally chosen under each strategy hid the good prize or the bad one. Because there will be a strong statistical component to the answer, we'll run a lot of such simulated games and tabulate the results by adding up how many times each strategy wins.

10.4.2 Laying out the Worksheet

To implement the idea, we'll set up the spreadsheet so that each row corresponds to a different game. The first column will contain the game number, the second the number of the curtain hiding the good prize, and the third the curtain number the contestant chooses first. In another column we'll put the number of the curtain shown to the contestant, and then two more will contain the contestant's final choice for each strategy. For one strategy this will be the same curtain number initially chosen; for the other it will be the curtain not initially chosen and not yet shown. Finally, in two more columns we will compare the final choices to the curtain number hiding the good prize and enter a one if it matches, a zero if not.

10.4.3 The First Three Cells

Filling in the first three columns is easy. To fill in the first column, in each cell except the topmost we put the contents of the cell just above plus one. In the topmost row we'll just put a one in this column. Having the computer choose which curtain hides the good prize, and which curtain the contestant chooses is also pretty easy. The problem is the same one we encountered in the previous section where we had to choose a birthday for each student in the class. The formula to use is `@INT(3*@RAND)+1`, so we'll put it in the second and third columns.

10.4.4 The Curtain Number to be Shown

Generating the number of the curtain to be shown the contestant is quite a bit more complicated. This curtain must not be the one the contestant chose, and it must not be the curtain hiding the good prize. If the chosen curtain is not the one with the good prize, the curtain to be shown is uniquely determined. For example, if curtain #1 hides the good prize, and the contestant chooses curtain #2, then the curtain to be shown is #3. On the other hand, if the contestant happened to choose the curtain hiding the good prize, then either of the remaining two curtains could be shown. In this case we would have to invoke the random number generator again to decide which one. Putting this on a spreadsheet is not so easy. The first thing I thought of was a series of `@IF` statements, but I rapidly gave up on this idea. It could be done, but it gets messy fast. Try it!

Here's another idea that works pretty well. Let's first consider the problem of specifying which curtain to show the contestant for the case that the curtain chosen is not the one with the good prize. Suppose we were to try to cook up some formula which when given two different integers between one and three would evaluate to the third. If we had such a formula, then we could use it to tell the computer how to specify the curtain to be shown the contestant in this case. Specifically, we want a function of two arguments, say x, and y, such that the function produces the numbers shown in the Table 10−1.

x	y	f(x, y)
1	2	3
1	3	2
2	3	1
2	1	3
3	1	2
3	2	1

TABLE 10−1. Definition of the function $f(x, y)$ which, given two *different* integers in the range 1 to 3, returns the integer in the same range which is different from both x and y. We don't care what f returns for other values of x and y.

Let's try a function of the form

$$f(x, y) = a + bx + cy \tag{10−5}$$

where a, b, and c are constants we have to determine so that f will have the properties in Table 10−1. We have three unknowns (a, b, and c), so we need three equations to determine them. Let's use the first three lines of the table. Doing so gives

$$\begin{aligned} a + \ \ b + 2c &= 3 \\ a + \ \ b + 3c &= 2 \\ a + 2b + 3c &= 1 \end{aligned}$$

These equations are easy to solve. Subtracting the first from the second gives $c = -1$, and subtracting the second from the third $b = -1$. Finally putting these two values in any of the equations gives $a = 6$. Thus our formula is

$$f(x, y) = 6 - (x + y) \tag{10−6}$$

Checking this formula against the first three lines of Table 10−1 shows that we did the algebra correctly, and checking it against the last three lines shows that we were also very lucky. (Enjoy it; it doesn't happen very often!) The formula also works for these last lines as well, even though that was not part of the specification in determining the constants a, b, and c.

Before continuing, let me say a few words about how I dreamed up Eq. (10−5). Looking at Table 10−1, it is clear that f must depend on both x and y. Polynomials are often good choices in situations such as this one because they are easy to evaluate and to manipulate. What order polynomial should we use? Each term in the polynomial introduces another unknown constant which can be adjusted to make the polynomial agree with Table 10−1. There are six lines in Table 10−1, so requiring our polynomial to agree with the table for all six lines will result in six equations. Six equations uniquely specify six unknowns, so it looks like we need a polynomial with six terms, such as

$$Q(x, y) = a + bx + cy + dx^2 + ey^2 + fxy$$

I found six equations in six unknowns distasteful, and I had a hunch that $f(x, y)$ would turn out to be symmetric in x, and y. (Note in Table 10−1 that the last three lines are the same as the first three, except that the values of x and y are interchanged.) For this reason I decided to try a polynomial with only three terms, like Eq. (10−5). In case my hunch was wrong, and the function turned out not to be symmetric, my plan was to develop a second function like Ex. (10−5) to be used for the second half of Table 10−1. Fortunately, f turned out to be symmetric, and Eq. (10−5) can be used for all values of x and y in Table 10−1.

Back to programming Quattro Pro, if the contestant did not initially choose the curtain hiding the good prize, we can use Eq. (10−6) to determine the curtain number to be shown. If, on the other hand, the contestant happened to choose the curtain hiding the good prize, we have a choice of two curtains to show, and we will have to use the random number generator again to choose which one. We'll generate an integer which is either 0 or 1, with equal probability. For ease of discussion, call this number r. Generating r is easy, we just put the formula `@INT(2*@RAND)` in a cell. To generate the curtain number to be shown we need a formula which has as arguments this random number, r, and a number *not* to be generated (the curtain number initially chosen, which in this case is the same as the curtain number hiding the good prize). There probably are easier ways to dream up such a formula, but I'll try to find one of the form

$$g(x, y) = d(x) + e(x)y$$

where g is defined in Table 10−2. Thus $d(x)$ and $e(x)$ must be defined by Table 10−3.

We need to dream up a function for $d(x)$ that will agree with Table 10−3. I'll try a polynomial in x. There are three lines in the table. That will translate into three equations, so I'll need three unknowns. To get three unknown constants a second order polynomial will be required, so I'll try a formula for d of the form

x	y	g(x, y)
1	0	2
1	1	3
2	0	1
2	1	3
3	0	1
3	1	2

TABLE 10−2. Definition of the function $g(x, y)$, which given an integer, x, in the range $1\cdots3$ returns the smaller of the remaining two integers in the same range if y is 0, and the larger if y is 1.

x	d(x)	e(x)
1	2	1
2	1	2
3	1	1

TABLE 10−3. Definitions of the functions $d(x)$ and $e(x)$ used in calculating the function $g(x, y)$.

$$d(x) = a + bx + cx^2 \tag{10−7}$$

where the constants a, b, and c are different from those in Eq. (10−5), and are to be chosen so that $d(x)$ obeys the table. Letting x be 1, 2, and 3 successively gives

$$\begin{aligned} a + \ \ b + \ \ c &= 2 \\ a + 2b + 4c &= 1 \\ a + 3b + 9c &= 1 \end{aligned}$$

Subtracting the first equation from the second and third gives

$$\begin{aligned} b + 3c &= -1 \\ 2b + 8c &= -1 \end{aligned}$$

Subtracting twice the first equation from the second gives $c = \frac{1}{2}$, and substituting this result in either gives $b = -\frac{5}{2}$. Finally, substituting these two values in any of the original three equations yields $a = 4$. Thus our function for $d(x)$ is

$$d(x) = 4 - \tfrac{5}{2}\,x + \tfrac{1}{2}\,x^2 \tag{10−8}$$

Going through a similar procedure for $e(x)$ results in the formula

$$e(x) = -2 + 4x - x^2 \tag{10−9}$$

We can put the formulae for r and for d and e in separate cells, with the arguments for d and e being the value of the cell giving the curtain number initially chosen. Then to determine the curtain to be shown to the contestant, we use an `@IF` function. If the curtain number hiding the good prize equals the curtain number chosen, the contents of this cell will be $d + er$, where r, d, and e are gotten from the contents of the previous cells as just discussed. Otherwise, the contents of this cell will be the value of the function given in Eq. (10−6), with the curtain number hiding the good prize, and the curtain number initially chosen as arguments.

10.4.5 The Remaining Cells

We will use two cells for the curtain number finally chosen: one cell for strategy number 1 in which the contestant does not change his/her initial choice, and another for strategy number 2 in which the contestant does change. The contents of the strategy number 1 cell are easy; they are just the contents of the cell containing the initial choice. For the strategy number 2 cell, we need the curtain number which was not originally chosen, and was not shown to the contestant. For this purpose, we can use the same formula we have already developed as Eq. (10−6), using as arguments the curtain number originally chosen, and the curtain number shown to the contestant. Finally, we will use two more cells to tabulate the results of each strategy. If strategy number 1 wins the good prize (i.e. if the number finally chosen matches the curtain number hiding the good prize), we will put a 1 in the corresponding cell, and a 0 otherwise. We do the same in the second cell for strategy number 2. After I was satisfied everything was working properly I hid the columns containing the values of r, d, and e so that all the interesting columns could be displayed on the screen at once.

Once we have the entire row working properly, we can just use the `Copy` function on the spreadsheet to make lots of copies of the row. I chose to make 499 copies, giving me 500 rows, each corresponding to a different game. To score each strategy, I used the `@SUM` function to sum the number of wins for each, and then below the cells containing this result I put a formula giving the percentage of wins for each strategy. Fig. 10−4 shows the worksheet I made. Part a) shows the formulae in the individual cells, and part b) shows the worksheet as it appears on the computer screen. In part a) all columns of the worksheet are shown, and in b) columns `D..F` have been hidden. Because of the number of cells and the length of the formulae, in part a) I split the worksheet into two, with columns `A..F` just above columns `G..K`. For clarity of presentation I prettied it up a little by putting in some labels and some lines and boxes. Neither are essential, and the value of the lines and boxes is questionable, but I do recommend putting in labels.

10.4.6 An Analytic Solution

Deriving an analytic solution to this problem is a little tricky. It seemed to me when I first saw it that both strategies should be equally effective. How could changing your mind after being shown a curtain change the probability of winning the good prize? After running the simulation several times you should see that I was wrong. Strategy number 2 (in which the contestant changes his/her mind) wins about twice as often as strategy number 1.

Why is that? The crucial point is to realize that with strategy number one the contestant wins only if he/she initially chooses the curtain with the good prize. Since there are three curtains, the probability of doing that is 1/3. With strategy number 2, on the other hand, the contestant wins every time he/she chooses a curtain with a bad prize the first time. Since there are two wrong curtains, the probability of choosing one is 2/3. Thus the probability of winning with strategy number 1 is 33%, and with strategy number 2 it is 67%.

a)

Chapter 10: Let's Make a Deal Simulation

WINNING Strat 1
@SUM(J11..J900)/H6

Game No.	Initial Curtains Chosen	Prize			
1	@INT(3*@RAND)+1	@INT(3*@RAND)+1	@INT(2*@RAND)	4-2.5*B11+0.5*B11*B11	-2+4*B11-B11*B11
+A11+1	@INT(3*@RAND)+1	@INT(3*@RAND)+1	@INT(2*@RAND)	4-2.5*B12+0.5*B12*B12	-2+4*B12-B12*B12

PERCENT Strat 2	Number of Games
@SUM(K11..K900)/H6	@COUNT(A11..A900)

	Final Choice		Final Score	
Shown	Strat 1	Strat 2	Strat 1	Strat 2
@IF(B11=C11,E11+D11*F11,6-(B11+C11))	+B11	6-(B11+G11)	@IF(C11=H11,1,0)	@IF(C11=I11,1,0)
@IF(B12=C12,E12+D12*F12,6-(B12+C12))	+B12	6-(B12+G12)	@IF(C12=H12,1,0)	@IF(C12=I12,1,0)

b)

Chapter 10: Let's Make a Deal Simulation

WINNING Strat 1	PERCENT Strat 2	Number of Games
32%	68%	500

Game	Initial Curtains			Final Choice		Final Score	
No.	Chosen	Prize	Shown	Strat 1	Strat 2	Strat 1	Strat 2
1	1	3	2	1	3	0	1
2	2	3	1	2	3	0	1
3	1	3	2	1	3	0	1
4	3	1	2	3	1	0	1

Figure 10−4. Worksheet for simulating Let's Make a Deal. In part a) all columns are shown and the worksheet was split in two. In part b) columns D..F are hidden.

10.5 Exercises

1. Program Quattro Pro to carry out the birthday simulation as described in Section 10.3 and run the simulation at least 20 times, recording the number of birthday coincidences each time, for class sizes of 10, 30, and 50 students. After doing so, program Quattro Pro to evaluate the analytic answer given by Eq. (10−4) for class sizes ranging from 1 to 50 students, and plot the probability of a coincidence vs. class size for this range. Compare the results of the analytic solution with your

numerical simulation.

2. Program Quattro Pro to carry out the *Let's Make a Deal* simulation described in Section 10.4 and run the simulation several times, noting the percentage of the time both strategies win. Compare your simulation results with the predictions of the analytical result.

3. Derive a simple polynomial function, $f(x)$, which agrees with the three values listed in the table below.

x	f(x)
-1	1
0	3
1	2

4. Suppose you want to find the simplest polynomial, $p(x)$, that satisfies the table below.

   a. Without actually doing the messy algebra, outline in as much detail as you can a mathematical procedure to find $p(x)$.

   b. Use Quattro Pro to actually find the polynomial without having to do a lot of messy algebra.

x	p(x)
−2	1
−1	3
0	6
1	−3
2	2

5. Use the random number generator function of Quattro Pro to calculate π. Here's the idea. Consider a circle of radius one inscribed inside a square with sides of length 2. If you were to randomly choose a point inside the square, the probability that it would also lie inside the circle would be just the ratio of the area of the circle to that of the square. Determine the ratio of the area of the circle to that of the square by using a simulation to determine the probability that a point randomly chosen inside the square is also inside the circle, and from that and the known area of the circle determine π. Use `@RAND` to calculate the x and y coordinates of the random point with both lying in the range −1 to 1. In case you've forgotten, the x and y coordinates of the points on a circle of radius r are related according to

$$x^2 + y^2 = r^2$$

   If $x^2 + y^2 \geq r^2$, then the point lies outside the circle, whereas if $x^2 + y^2 \leq r^2$, it lies inside. Compare your value for π with the correct value. (The built-in function `@PI` returns π).

6. This problem is a variation on the previous problem. Suppose you have a closed plane figure the area of which you would like to calculate. You could inscribe the figure inside a square or other plane figure for which you know the area and then, as in the previous problem, generate points randomly inside the known plane figure. After generating a large number of points, the fraction that were also inside the original plane figure for which the area is not known would just be equal to the ratio of the areas of the two figures. This procedure is called *Monte Carlo integration*.

   Use this idea to determine the area of an ellipse with semi-major and semi-minor axes specified in two cells of the worksheet. Explain what you do. You will have to choose a geometrical figure in which to inscribe the ellipse. Discuss why you chose the figure you did.

   In case you don't remember from high school algebra, the border of an ellipse is described by

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

where a and b are the semi-major and semi-minor axis lengths of the ellipse.

It can be shown using the integral calculus that the area of an ellipse with axes a and b is πab. Compare the accuracy of your simulation with the analytic result for several values of a and b.

7. A famous problem in statistics is the *random walk* or *drunken sailor* problem. Here it is. Imagine a very drunk sailor in the middle of a narrow bridge with high guard rails so that he can't fall off. The bridge is narrow, so he can only go in two directions, forward or backward. He is so drunk that at each step, the probability of either is the same. If each step has the same length, say L_{step}, after N steps, how far from the center on the average has he gotten? Since both directions are equally probable at any step, you might think that on the average he would stay at the center of the bridge, but that's actually not true. I won't derive it for you, but it can be shown that after N steps, on the average he will have moved a distance $L_{step}\sqrt{N}$, with both forward and backward being equally probable.

   Program Quattro Pro to simulate 200 drunken sailors, each on his own bridge. Follow each sailor for 100 steps and note how far he has gotten at each step. As with the discussion of the average error between Herbie's measured data and the predictions of the "law" in Section 6−6, there is a problem in determining the average distance from the center of the bridge because the distance for some sailors will be positive (they went forward more often than back), and for others negative (they went backward more often). The solution to the dilemma is the same as it was with Herbie's data—take the root-mean-square (R.M.S.) average. The analytic result given above is for an R.M.S. average. Compare the simulation results with the analytic result given above.

   Perhaps you are wondering why this is a famous problem. The reason is that essentially the same problem appears in a wide range of subjects. The application I'm most familiar with is that of *diffusion*. Imagine a bottle of cheap perfume in one corner of a closed room with no air currents in it. If you were on the opposite side of the room from the bottle of perfume and someone were to open the bottle, at first you would not notice anything, but after a while you would begin smelling it. We would say that the molecules of perfume diffused to your nose. If you think about what is going on at a molecular level, you'll see that the problem is closely related to the random walk problem. Evaporating perfume molecules are knocked every which way by collisions with air molecules, and effectively make a random walk away from the bottle. The only differences between this problem and the one we just solved are that this one is three-dimensional whereas our problem was restricted to only one dimension, and the "step" lengths of the perfume molecules are also random. Both effects are easily included in the analytic solution to the problem, and the result is the same as we got for the inebriated sailor, except that L_{step} becomes the average step length. Including the random step length in the spreadsheet simulation is pretty easy, and converting the simulation to a three-dimensional one is not too much harder.

8. It is tempting to try to simulate a card game, such as poker, using the computer. Unfortunately, there are several difficulties in doing so. Here's one of the difficulties, along with a suggested solution.

   One of the problems with simulating a card game is simulating the deal. Each player must get a preset number of cards with all cards being equally probable, except that two players cannot get the same card. The problem is similar to the one we encountered in the *Let's Make a Deal* simulation, except that 52 cards are involved rather than 3. Deriving a formula similar to Eq. (10−5) that would work for this case is truly a formidable problem!

   A method that is feasible is the following. In one column put 52 random numbers, one below the other. In the second column, put the numbers 1−52 in any order. Each number will correspond

to a particular card. Then sort the table consisting of these two columns by the first column. Since the numbers in the column are supposed to be random, sorting them should produce a random ordering of the cards in the second column. If each player is to get *n* cards, then the first *n* rows would give the cards dealt to the first player, the second *n* rows to the second player, and so forth.

Program a worksheet to deal a deck of cards in this way for a poker game with four players. (Assume that each player gets five cards.) Use the line drawing capability of Quattro Pro (in the `Style` menu) to draw a box around each hand. If you are ambitious, instead of numbers in the second column, you could put text describing each card such as S3, or "5 of diamonds".

9. Many games of chance are based on dice ("dice" is the plural, "die" the singular). In case you may not be familiar with these things, a die is a small cube with six faces. There are spots on each face; one face has 1 spot, one has 2, another 3, and so forth up to 6. In a game one or more dice are thrown onto a table, and after they have come to a rest, the number of spots on the top faces is counted. The individual die are supposed to be made so that each face is equally likely to wind up on top. Thus with one die integers between 1 and 6 can be randomly generated. With two dice, integers between 2 and 12 are generated. The integer thus generated is said to be "rolled."

   In this exercise you are to determine the probabilities of rolling each of the possible numbers. The question can be answered either by making a numerical simulation, or by deriving an analytic solution.

   a. Program Quattro Pro to simulate rolling the dice many times, and to count the number of times each of the possible results (2→12) occurs.

   b. Derive an analytic solution and compare with the result obtained above.

10. The resistance, *R*, of the circuit in Fig. 10−5 , as seen from the terminals is

$$R = R_1 + \frac{R_2 R_3}{R_2 + R_3}$$

When you buy a resistor of some nominal value, 1000 Ω for example, the actual resistance will probably be a little different. The manufacturer generally guarantees only that the actual value will be within some percentage, called the *tolerance*, of the advertised value. Thus the actual resistance of a 5% tolerance, 1000 Ω resistor would be within ±5% of 1000 Ω (i.e. between 950 and 1050 Ω). For this reason, if you were to construct the circuit below, you would probably find that the resistance was not exactly what you expected because the values of the resistors would not be exactly what you assumed.

The question here is, if all three resistors in the circuit below have the same tolerance, how could you determine the likely range of error between the expected and actual resistance? In other words, if you were to buy a lot of resistors of each value and to construct a lot of circuits according to the diagram below, you would find that no circuit would have exactly the expected resistance, with each circuit having a different error between the expected and actual resistance. The question of interest is: "For such a set of circuits what would be the RMS (root mean square) error?".

Questions such as this one are difficult to answer analytically, but are easy to answer using a numerical simulation. Assume that the actual values of the three resistors vary randomly over the range of values specified by the tolerance, with all allowed values being equally likely. Program Quattro Pro to determine the RMS error between the expected and actual values of *R* for a given set of nominal resistor values and a given tolerance by numerically simulating the procedure described in the paragraph above. Put the nominal values of each of the three resistors in cells `A2`, `B2`, and `C2`, and the tolerance in cell `D2`. The answer should appear in another cell.

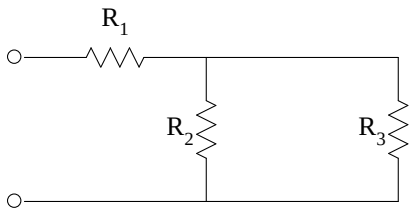

Figure 10−5. Circuit for exercise 10−10