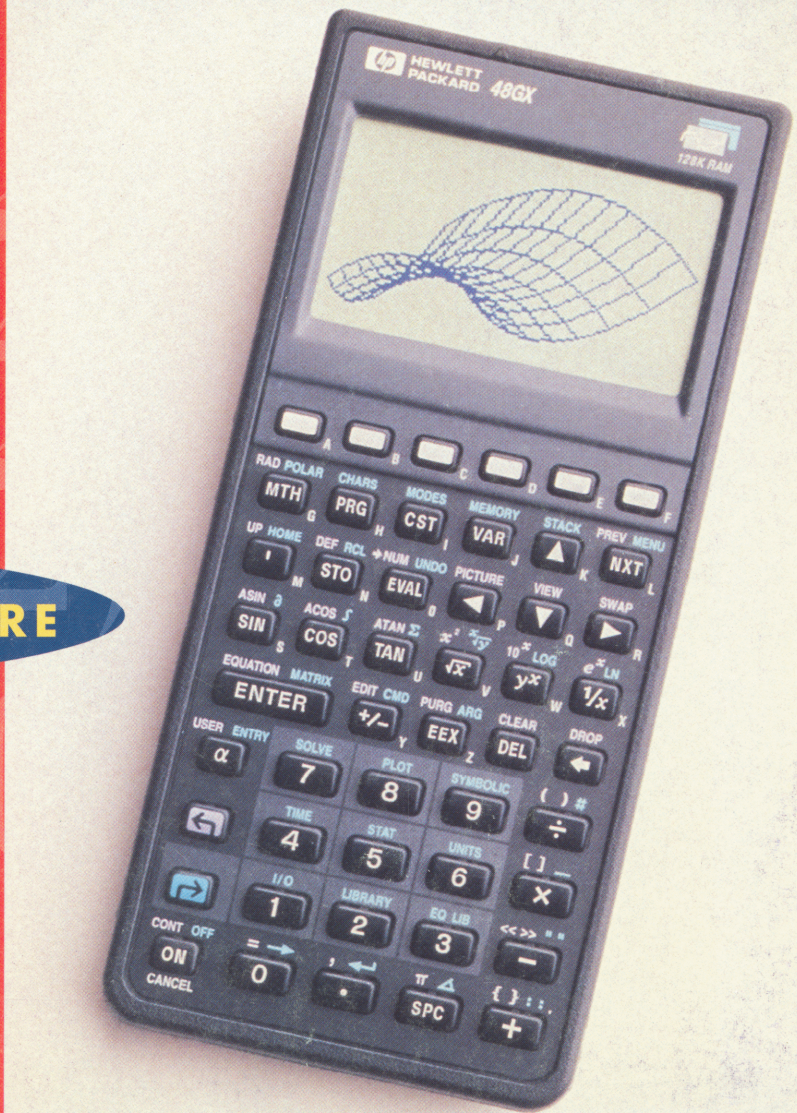
LINEAR ALGEBRA INVESTIGATIONS WITH THE HP-48G/GX

DONALD R. LATORRE



LINEAR ALGEBRA INVESTIGATIONS

with the

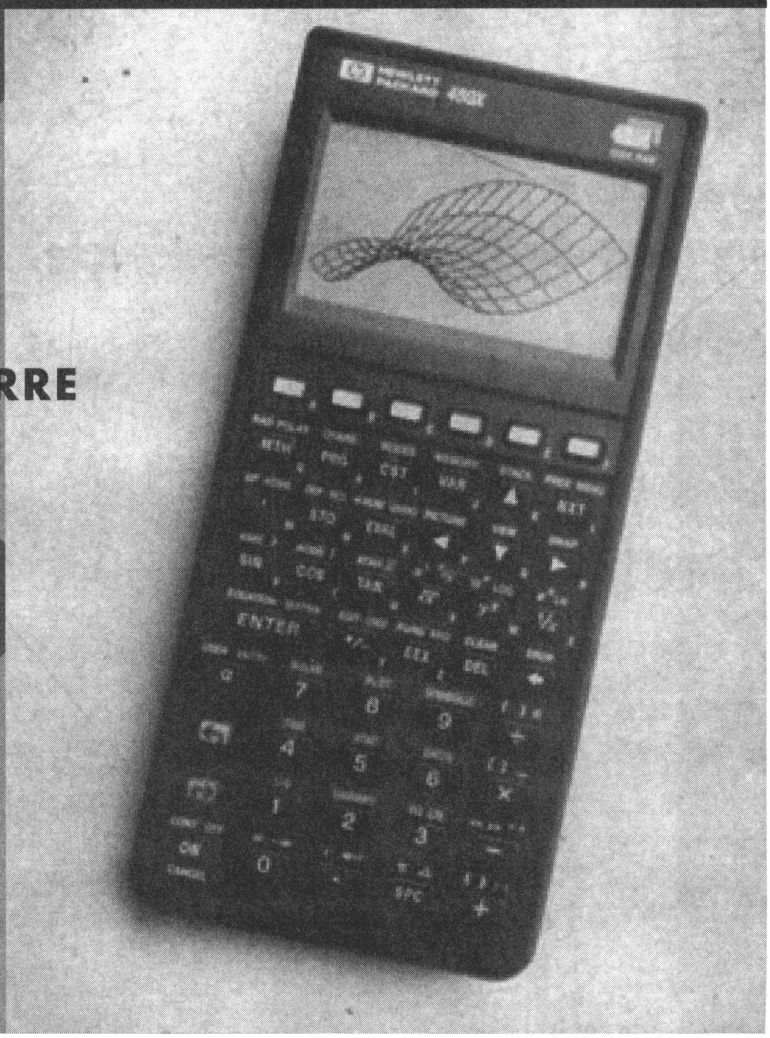
HP-48G/GX

LINEAR ALGEBRA INVESTIGATIONS

with the

HP-48G/GX

DONALD R. LATORRE
Clemson University



CHARLES RIVER MEDIA, INC.
Rockland, Massachusetts

Publisher: David F. Pallai
Production: Reuben Kantor
Cover Design: Gary Ragaglia
Printer: InterCity Press, Rockland, MA.

CHARLES RIVER MEDIA, INC.
P.O. Box 417
403 VFW Drive
Rockland, Massachusetts 02370
617-871-4184
617-871-4376 (FAX)
chrivmedia@aol.com

This book is printed on acid-free paper.

Linear Algebra Investigations with the HP-48G/GX Donald R. LaTorre

ISBN: 1-886801-20-7

Printed in the United States of America
95 96 97 98 99 7 6 5 4 3 2 First Edition

CHARLES RIVER MEDIA titles are available for bulk purchase by institutions, user groups, corporations, etc. For additional information, please contact the Special Sales Department at 617-871-4184.

CONTENTS

PREFACE

Linear Algebra is a fantastic subject! Incredibly rich and powerful in terms of its mathematical content and applicability, linear algebra is riding the crest of a nationwide resurgence of interest in mathematics at the undergraduate level.

It is not coincidental that this renewed attention to linear algebra is occurring at a time when technology is invading our classrooms. For, more than anything else, computing technology is opening up linear algebra in new and unprecedented ways. Ways that are challenging, exciting, and stimulating to students and teachers alike.

The Linear Algebra Curriculum Study Group has issued recommendations for the First Course in Linear Algebra [The College Mathematics Journal, Vol. 24, No. 1, January 1993]. The recommendations advocate a matrix theoretic approach and call for the appropriate use of technology. Not only microcomputers running state-of-the-art software such as MATLAB, but also technology in the form of the new hand-held "supercalculators" that are sweeping across the country.

This book is intended as a technology supplement for undergraduate courses in linear algebra. It presents appropriate pedagogical uses of, and teaching code for, the Hewlett Packard HP-48G/GX graphics calculators. It is designed specifically to help students and instructors incorporate these powerful hand-held devices as a computational tool for the interactive learning of linear algebra, and is independent of any particular textbook. The chapters survey the main topics of linear algebra and include activities and discovery projects that will engage students in a serious, calculator enhanced study of the material.

FEATURES

Outline of the Book

Realizing that different instructors and different textbooks approach linear algebra in their own unique ways, I have organized the material into independent chapters that follow main themes:

- Arrays
- Systems of Linear Equations
- Vector Spaces
- Orthogonality
- Eigenvalues and Eigenvectors
- Iterative Methods

These chapters are preceded by an introductory chapter on HP-48G/GX Calculator Basics and followed by two appendices: one on Vector and Matrix Norms and another on the organization of our HP-48G/GX Teaching Code.

HP-48G/GX Teaching Code

The Teaching Code itself is a collection of special-purpose HP-48G/GX programs, each one addressing a specific aspect of the course. A complete listing of the teaching code appears on the inside back cover. The code is readily available from the author for downloading to an HP-48G/GX from a microcomputer.

Pedagogy

The material is an outgrowth of my extensive classroom use of the HP-48 calculators (and before that, the HP-28 units) in teaching linear algebra every semester for the last five years. My course is at the sophomore level and is populated by students from a variety of fields: the physical and biological sciences,

mathematical and computer sciences, many engineering fields, secondary mathematics education, and an occasional social sciences student. I do not teach an abstract, proof-oriented course. Instead, I prefer to concentrate on explanations, examples, classroom discussions, and calculator activities to generate a real interest in, and enthusiasm for the learning of linear algebra. Proofs are important, and I do my share of them. But for almost all of my students, this is the only formal course in linear algebra that they will ever take, so I must be careful to not let my own predilection as a mathematician for "theorems" and "proofs" hinder the efforts of these beginning students.

Level

Although my course in linear algebra is pitched at the sophomore level, many of the students are further along in their studies. Most have completed (or will soon complete) Clemson's four semester sequence in calculus and differential equations. Thus, I am often able to nudge them beyond our normal expectations in linear algebra at the sophomore level and treat certain topics in more depth: e.g., orthogonality, least squares, and real symmetric matrices. I usually reserve the material on iterative methods and singular value decompositions for honors students as outside class projects. Of course, none of this push beyond traditional expectations would be possible without immediate access to first-rate computational capability. Violá the HP-48G/GX!

How to use this book

I use the material in this book to supplement whatever textbook I am using at the particular time. If the use of technology is to be of any significance in the learning process, it must not be used as an occasional "add-on" to the course. Rather, its use must become an integral part of the teaching and learning process. Therefore, I require my students to use their HP-48G/GX units on a regular, almost daily basis.

Whenever it is appropriate, I make homework assignments from this book; sometimes in addition to assignments from the main textbook, sometimes in lieu of such assignments. I allow free and unrestricted use of the calculators on all tests and exams. There is ample opportunity for me to assess their learning of both concepts and procedures, so the technology poses no threat. On the contrary, it has helped my students to place much of what has traditionally occupied them in courses in linear algebra (i.e., hand computation of elementary matrix algorithms) in proper perspective. I have found students to be overwhelmingly enthusiastic about the use of the calculators as a tool for effective learning.

Acknowledgments

I must express my deep appreciation to both individuals and organizations for their encouragement and support during the preparation of this book.

First, to Bill Wickes of Hewlett Packard, whose wisdom and genius in the design of the HP-48 has been a major source of inspiration. Especially also, to Paul McClellan of the software design team at Hewlett Packard for his splendid development of calculator versions of the LAPACK Fortran code for the matrix operations that are built in to the HP-48G/GX. Gil Strang of MIT has had an enormous influence on my understanding of linear algebra, on what is really important about teaching, and what good teaching can really do. To my friends on the Linear Algebra Curriculum Study Group — David Carlson, Charlie Johnson, David Lay, and Duane Porter — for accepting me into their national dialogue on the subject. To Steve Leon of the University of Massachusetts - Dartmouth, for numerous stimulating conversations on using technology in teaching linear algebra. And, with great admiration to April Haynes, who with considerable skill and patience, produced by word-processing everything that appears between the covers of this book.

Finally, I wish to gratefully acknowledge the support of the National Science Foundation (NSF USE-9153309) during the preparation of this book. I am also indebted to the Fund for the Improvement of Postsecondary Education (FIPSE) for their early support of my ventures into teaching with hand-held technology.

Clemson University *Don LaTorre*

June, 1995

1 HP-48G/GX CALCULATOR BASICS

This brief chapter is intended to provide new users with a basic introduction to the HP-48G/GX calculator and its operation. It is no substitute for the User's Guide, but should help you get started quickly.

Notation

To help you recognize calculator keystrokes and commands, we shall adopt certain notational conventions.

- With the exception of the six white keys on the top row, keys will be represented by helvetica characters in a box: [ENTER], [EVAL], [STO], etc.
- Shifted keys on the 48G/GX may occasionally have the key name in a box preceded by the appropriate shift as in [↰] [CST]. But ordinarily, we will not show the shift.
- Menu keys for commands on various menus will show the key name *in outline form* in a box, as in [RSWP] or [DET]
- Calculator operations and commands that appear in programs or in the text material will be in helvetica characters, e.g., DUP SWAP INV.

On, Off and Contrast

Press the [ON] key (bottom left of the keyboard) to turn the unit on. Press [↱] [OFF] to turn it off. The [OFF] key is the right-shifted (green) version of the

ON key. With the calculator on, hold down the ON key and press + to darken the display contrast or − to make it lighter.

Stack Display Screen

When you first turn on a factory fresh HP-48G series calculator, you will be looking at the *stack display screen*. To remove any objects from the screen that may remain from previous use, press the ON key several times then the DEL key (on the same row of keys as ENTER). Above the horizontal line near the top of the screen you will see {HOME}, indicating that you are in your HOME directory. Immediately below are levels 1-4 of the *stack*. Like lines on a piece of paper, the stack is a sequence of temporary storage locations for numbers and the other kinds of objects used by the calculator such as algebraic expressions, arrays, equations, and programs.

Just below level 1 are six menu boxes. Normally, these menu boxes will have labels in them that reflect the operation of the *six white menu keys* beneath them. If you press the MTH key near the top left of the keyboard, the labels will show that the first page of the MTH menu contains the six *submenus* VECTR, MATR, LIST, HYP, REAL, and BASE; the NXT key (same row as MTH) will turn you to the second page of the MTH menu and another NXT will cycle you back to the beginning. Return to the previous page with PREV (the left shifted NXT key). The small horizontal tabs above the labels in the MTH menu indicate that each of the boxes contains a submenu (a file, folder or *subdirectory* in HP parlance). Open the MATR (= matrix) submenu by pressing the white menu key beneath it to access the various subdirectories and special commands for working with matrices. Press MTH to return to the MTH menu at any time.

Similarly, the PRG key opens the PRG (= Program) menu where you may use the white menu keys to access the various submenus of commands for use in writing

programs. An extremely important key is the [VAR] key. It opens the VAR (= Variables) menu, *which is where you look to find the objects that you have created and stored into the memory of the machine.* For routine calculations on the stack, it does not matter which menu labels are active. Simply press [CST] to make them all blank.

Keyboard

The keyboard of an HP-48G series calculator may at first appear to be somewhat intimidating. But, like the control panel of any high-performance device, it enables you to control and to monitor a vast array of operations. The number entry keys are bordered on the right by [+], [−], [×], and [÷]; and on the left by [ON], [↱], [↰], and [α]. The right-shift key [↱] (green) and the left-shift key [↰] (purple) are color coded to many of the keyboard labels, and the [α] key is used to obtain alphabetical characters.

Adjacent to [ENTER] is [+/−] for changing signs, then [EEX] for entering exponents, [DEL] for deleting characters (and clearing the stack), and [⌫] for backspace-and-delete (and dropping objects from level 1). The [SIN], [COS], [TAN], and [√x] keys are just above, as are [y^x] (for obtaining powers) and [1/x] (for reciprocals and matrix inverses). Above the trig function keys are ['] (tick), for entering algebraic expressions, and [STO] and [EVAL] for storing and evaluating objects. The four cursor keys [Δ], [▽], [◁] and [▷] control the movement of the cursor when it is active.

Applications and Command Menus

You will notice that some keys have both purple and green labels printed above them (like the [α] key), but many have only one of the two (like the [7], [8] and [▷] keys).

The keys that have only green labels above them represent *applications*, e.g., I/O, PLOT, SOLVE, TIME, UNITS. The right-shifted version of an application key invokes a specially designed user-interface that lets you interact directly with the named application, often through the use of *input forms*, which are the HP equivalent of the familiar computer "dialogue boxes". Alternatively, the left-shifted version of an application key gives you access to the various commands on the *command menu* that is associated with the particular application. The commands may be included in programs or executed directly from the keyboard while viewing the stack display screen. In linear algebra, we almost never use any of the above named applications.

Display Settings

It is best to keep the calculator's angle mode set to radians in order to work with trigonometric functions. Press [↰] (purple) [MTH] to toggle between radian mode and degree mode. When radian mode is set, the message RAD appears at the top left of the stack display screen.

To display numbers in standard form, set your unit to STD display mode (the default setting) by pressing [↰] [MODES] (the left-shifted [CST] key), opening the FMT (= Format) menu and checking to see that the left-most menu box reads [STD☐]. The small box next to STD indicates that STD mode is active. If the menu simply reads [STD] press the associated white menu key to activate STD mode. Now press [↱] [MODES] to interact with the main MODES screen. You should see that the number format is highlighted and set to Std, and that the angle measure is set to Radians. Press the [▽] twice to highlight the coordinate system field (it should read Rectangular, by default). To see how to change such a field, press the white menu key beneath [CHOOS], use [▽] to highlight Polar and press [OK]. You have just changed to polar coordinates. Now change back to

rectangular coordinates. When the display is set to show only a fixed number of digits to the right of the decimal point, say with 3 FIX to display only three such digits, the numerical calculations are still executed internally to the full 12- or 15-digit precision of the machine. Only the display is affected. By resetting to STD mode, you will display full 12-digit precision. Unless stated otherwise, we will assume throughout this book that the display mode is set to STD and that the coordinate system is set to RECTANGULAR.

The √ by BEEP means that the beeper is turned on (to alert you of syntax errors, alarms, etc.). To activate the clock, highlight the clock field and press the [√] key. Leave the fraction mark (FM,) unchecked so that decimal points, rather than commas, will appear in decimal numbers like 123.45. Exit this screen by pressing [OK]. Notice that the time and date now appear above the horizontal line. If you wish to modify the time or date, press [↱] [TIME] (the green shifted [4] key) and proceed as above.

Symbolic Execution Mode

The HP-48G/GX is a third generation *symbolic* calculator, which means that you can apply operations and functions to symbolic expressions and obtain symbolic results. For example, you can enter the symbolic expressions for x^2 and for $sin\ x$, then press the [+] key to obtain the symbolic result $x^2 + sin\ x$. Most other calculators are numerical calculators, capable of applying functions and operations only to numerical objects to obtain numerical results.

Symbolic execution mode is controlled by a system *flag* (flag -3). In the default state, flag -3 is clear and the HP-48 is in Symbolic Execution Mode. In this mode, the symbolic constants (e, i π, MAXR, and MINR) and functions with symbolic arguments will evaluate to symbolic results. But if flag -3 is set, Numerical Results mode is

active and the symbolic constants and functions with symbolic arguments will evaluate to numbers.

We *strongly recommend* that you keep your HP-48G/GX in Symbolic Execution Mode. If you go to the MODES menu with the [↰] [CST] keys and open the [MISC] submenu, the SYM menu key should read [SYM□]. The small box that appears next to SYM indicates that Symbolic Execution Mode is active. If no box appears in this key, simply press the [SYM] key to change it to [SYM□].

Numerical Calculations

Simple numerical calculations are done on the stack. The idea is this: put inputs on the stack and then execute commands that use the inputs. To enter -12.34, begin by pressing the appropriate number keys and the decimal point key (bottom row, center), then use [+/−] to change the sign. Notice that the typing starts at the bottom left of the display screen, below level 1 of the stack, on the *command line*. Press [ENTER] to put -12.34 on level 1. Now enter 56.789; notice that [ENTER] inserts it onto level 1, moving -12.34 up to level 2. Press [+] to compute the sum. To recapture the stack before you added, press [↱] [UNDO] (the right-shifted EVAL key). Now subtract 56.789 from -12.34 with [−], then use [UNDO] and swap positions with [SWAP] (the right cursor key [▷]; no need to press [↰] now). Now subtract again to get 69.129. Take the square root with [√x], then cube the result with 3 [Y^x]. You should have 574.765129278.

To edit this result, press the [▽] (down cursor) key, use the right cursor key to move the cursor over the 9, delete the 9 with [DEL] and press 3 [ENTER]. Now use [↱] [LN] (the right-shifted [1/x] key) to obtain the natural logarithm. To multiply by π, press [↰] [π] (π is obtained with the left-shift [SPC] key) then [×]. Notice the symbolic result '6.35396147609 * π' on level 1,

enclosed in tick marks. To convert this to a numerical result, use [↰] [→NUM] (the left-shift [EVAL] key). Now drop the 19.9615586945 from level 1 with [⇦]. The [⇦] key drops objects from level 1; the adjacent key (labeled CLEAR in purple) clears the entire stack. Normally, you need not left-shift these keys; shifting is required only when the command line is active.

Data Entry

When keying a sequence of real numbers into the command line, say 1.1, 2.2 and 3.3, you must separate the numbers with spaces or commas for proper recognition, as in 1.1 2.2 3.3 or 1.1, 2.2, 3.3. *We recommend that you use spaces for ease of use.* For consistency we will show commas, but you should always interpret them as spaces. You need not insert commas or spaces between a real number and a complex number (an ordered pair), or between two complex numbers, because the calculator recognizes parentheses as object delimiters. *Unless we specify otherwise, all examples and exercises in this book assume the calculator is set to STD display mode.*

Algebraic Expressions

Algebraic expressions must be typed in beginning with a ' (tick) mark using the ['] key. Alphabetical characters are obtained by first pressing [α] and then the desired key. *Note that alphabetical characters appear in white letters to the lower right of the keys on the top four rows.* To produce, say 'S', press ['] followed by [α] [SIN] [ENTER]. Lower case characters are obtained by the sequence ['] [α] [↰], then the character key. For example, ['] [α] [↰] [D] [ENTER] puts 'd' on level 1. (Thus [α] *left-shift* will give *lower case*).

To enter the algebraic expression 'SIN(X)', press ['] [SIN] [α] [1/x] [ENTER]. Notice the location of the cursor after each keystroke; after [1/x] the cursor is still inside the right parenthesis. To move it outside, use the right cursor

key [▷]. But, pressing [ENTER] does it all for you. As a more complicated example, try 'COS(X ∧ 2)/(2 * X ∧ 3)'. The keystroke sequence is:

['] [COS] [α] [1/x] [Y^x] 2 [▷] [÷] [↰] [÷] 2
[×] [α] [1/x] [Y^x] 3 [ENTER].

Yes, it is necessary to insert the * in 2*X∧3; if you forget, when you press [ENTER], an Invalid Syntax message will appear and you can then correct your typing. If things are not going well on the command line, remember that the [⇦] key will backspace and delete. Finally, if you get desperate, press [ON] (sometimes, more than once) to cancel what is taking place and then start over.

Stack Manipulation

We often need to manipulate the stack. For example, to duplicate one or more levels, to copy an object from a higher level down to level 1, or to otherwise rearrange the stack. Complete details can be found in chapter 3 of the HP-48G series User's Guide, but we will survey the basics here. This survey should suffice for most purposes.

To make a duplicate copy of the object on level 1, simply press [ENTER]. This executes the DUP command, which duplicates level 1. We have already commented on the obvious keyboard commands DROP (the [⇦] key), CLEAR (the [DEL] key), SWAP (the [▷] key), and UNDO (the [↱] [EVAL] key). Although the DROP, CLEAR and SWAP keys are labelled in purple, it is not necessary to use the purple [↰] key except when the command line is active.

The best way to understand the other stack commands is to begin with your stack arranged like this:

4: 'S'
3: 'T'
2: 'U'
1: 'V'

Now press the [Δ] key to engage the *interactive* stack. The interactive stack is an environment that lets you interact with the stack and is active when the dark pointer ▶ appears at the left of the screen. You exit the interactive stack with [ENTER] or [ON] (either one will work). So arrange your stack as in the above illustration and then press the [Δ] key. The commands that are most often used are PICK, ROLL, ROLLD, →LIST and (on the next page) DUPN and DRPN.

Move the pointer up to level 3 and press [PICK] [ENTER]. The command PICK copies the content of level 3 to level 1. Use [⇦] to DROP the 'T' from level 1.

Now move the pointer back to level 3 and press [→LIST] [ENTER]. Notice that the contents of levels 1 – 3 were put into a list (lists use curly braces). Now restore the stack to its original state with [UNDO].

The commands DUPN and DRPN (on the next page) are almost self-evident. With the pointer situated on level N, DUPN will duplicate the first N levels of the stack while DRPN will drop the first N levels. Try using DRPN with the pointer at level 3. Press [ENTER] to exit, then use [UNDO] to restore everything.

The last two commands, ROLL and ROLLD are extremely useful. With the pointer specifying the number N of levels, ROLL will push (*roll*) the stack upward, causing the object on level N to fall down to level 1. Try using 4 ROLL to rearrange the stack:

4: 'S'		4: 'T'
3: 'T'	4 ROLL ☞	3: 'U'
2: 'U'		2: 'V'
1: 'V'		1: 'S'

(The *'S' rolled* off the top level and fell down to level 1.)

The command ROLLD (*roll down*) is just the opposite: it pulls the specified number of objects down, causing the level 1 object to move to the top level. Restore the current stack to its original state with 4 ROLLD. Now use CLEAR to clear the stack.

RPN

RPN stands for *Reverse Polish Notation*, the type of logic used by almost all Hewlett Packard calculators. The essence of RPN is this: first provide the inputs, then execute commands that operate on the inputs. When we did our earlier calculations on the stack, we were using RPN entry. Thus, to add -12.34 and 56.789 in RPN we input -12.34 and 56.789, then executed the command +. In fact, we built -12.34 using RPN: input 12.34, then press [+/−]. Notice how this differs from the *algebraic entry logic* employed by most other types of calculators. Algebraic entry requires that we type in -12.34 + 56.789 from left-to-right and then press an [ENTER] or [=] key. To produce a numerical result for $\sqrt{ln\ 2.3}$ on the HP-48 using algebraic entry we type

' √ LN 2.3 EVAL

But to obtain this using RPN, we do

2.3 LN √

RPN is an especially powerful logic for constructing the algebraic expressions that we encounter in a beginning study of calculus. Expressions such as

$$\sqrt{1 + \cos^2(x^3)} \quad \text{or} \quad (1 + x)^{2/3} + \frac{2x + 1}{\sqrt{x^2 - 4}} .$$

Consider the first of these two. Superficially, it is simply the square root of one plus the cosine squared of x^3. But it is important that we understand this expression mathematically, *from inside out*, as follows: start with x and cube it, take the cosine of x^3 and square the result, then add 1 and take the square root. RPN entry corresponds exactly to this way of thinking:

'X' 3 ^ COS SQ 1 + √ .

A more complex example is provided by the second of the above two expressions. First, try entering this expression using direct algebraic entry (remember to start with a ' (tick) mark); what did you find out? Now use RPN entry as follows: begin by putting the three main components '(1 + X) ^ (2/3)', '2*X + 1', and '√(X ^ 2 − 4)' on the stack in this order (you can use either direct algebraic entry or RPN for any of them); now press [÷] to build the quotient, then [+] to obtain the sum.

This last example clearly illustrates why RPN is the preferred method for entering complicated expressions onto the stack. Most users tend to develop their own style, often using direct algebraic entry to build simple components and then RPN to produce the more complicated final results. Of course, all programs on the HP-48 must be written in RPN. For example, the program

« SQ SWAP SQ + √ »

uses RPN logic to take two inputs from the stack, say x and y, and then returns the result $\sqrt{x^2 + y^2}$.

User-Defined Functions

A convenient way to represent a mathematical function on the HP-48 is to create a *user-defined function*. In HP-48 language, a user-defined function is a short program

that captures the essence of the formal way that we define a function by an equation like $F(x) = 2 \sin x + \sin 4x$. Here, F is the name of the function, x is the input variable, and the expression to the right of the = sign is an algebraic description of the desired output for a given input x.

The user-defined function that represents this mathematical function is the program « → X '2*SIN(X) + SIN(4*X)' » stored in the global variable F. The DEFINE command lets you create a user-defined function directly from an equation. For the example at hand, simply enter the equation 'F(X) = 2 * SIN(X) + SIN(4 * X)' onto level 1 of the stack and press [◁] [DEF]. If you access the VAR menu with the [VAR] key, you will see the label [𝔽] appearing above a white menu key; this identifies F as the name of the user-defined function. To verify that the variable named F actually contains the above program, you can recall the contents of variable F by pressing [▷] [𝔽] (the right-shift key [▷] will recall); press [DROP] when you've finished viewing the program.

To evaluate this function, enter the desired input and press the menu key [𝔽]. For example, put 'T ^ 2' on level 1 and press [𝔽] to see '2 *SIN(T ^ 2) + SIN(4 * T ^ 2)'. Likewise, press 4 [𝔽] to see $2 \sin 4 + \sin(4*4)$ evaluated as -1.80150830728. User-defined functions of two or more variables are constructed in the same way.

Memory Management

The HP-48 can manipulate and store many types of *objects*, such as real and complex numbers, algebraic expressions, vectors and matrices, lists, graphics, programs, and text. Any of these objects can be placed on the stack, but to be saved in the calculator's memory it must be given a name and stored. When you store an object, it is stored as a *variable* in *user memory* (that part of the calculator's memory that you, the user, have access to) and is accessible through the VAR menu. The

variables that you create in this way are called *global variables* to distinguish them from other kinds of variables that the HP-48 uses (e.g., *local variables* – that are created within and used entirely by a program, and *system variables* – that are used by the calculator's operating system). You can think of a global variable as a named storage location containing an object.

For example, suppose that you wish to create a variable named TRY1 containing a program that will accept numbers x and y as inputs and calculate $\sqrt{x^2 + y^2}$. Here is the program:

« SQ SWAP SQ + √ »

To build the program, press [↰] [−] to get the program delimiters « », then use [↰] [√x] [↰] [▷] [↰] [√x] [+] [√x] [ENTER]. Now put the name 'TRY1' on the stack and press the [STO] key. If you press the VAR key you will see that the leftmost menu key is labeled [TRY1]. To run the program with inputs 1 and 2, put 1 and 2 on the stack and then press [TRY1] to see the result 2.2360679775. In fact, you need not actually enter the inputs onto the stack: simply press 1 [SPC] 2, then [TRY1] to get the result. The HP-48 recognizes spaces as object separators and TRY1 will take the inputs directly from the command line. We will use this shortcut extensively with our programs that manipulate matrices.

To delete a variable from user memory, put its name on stack level 1 and execute the command PURGE. The [PURGE] key is the left-shifted [EEX] key. To purge variable TRY1, press ['] (tick), [TRY1] [ENTER], then [PURGE].

To organize the variables that you create, you can put them into files (or *directories*). Whenever you create a variable and store it, it is stored in the current directory. If you are using a factory fresh HP-48 then your current directory is the HOME directory, indicated by the list { HOME } at the top left of the stack display

screen. The name of the current directory always appears as the rightmost name in the list that begins with HOME, as above. To create a subdirectory named MTRX in which you can store any variables that you may need in a study of linear algebra, begin by putting the name 'MTRX' on stack level 1. Now press ↰ MEMORY (the left-shifted VAR key), open the DIR submenu and execute the command CRDIR (create directory). If you then open the VAR menu you will see the MTRX directory on the left. The short bar above the label is suggestive of the tab on a file folder, and reminds you that MTRX is a subdirectory. Press MTRX to open this directory and notice the list {HOME MTRX} at the top of your screen, indicating that the current directory is now MTRX. This directory is presently empty, containing no variables. To return to the parent directory HOME, you need only go up one level in the directory tree. The commands UP and HOME, executed by shifting the ' (tick) key appropriately, send you up one level or, alternatively, send you directly to HOME.

A few final comments about storing and purging variables from directories. The same variable can exist in different directories, often containing different objects. For example, whenever you use the PLOT application, copies of the reserved variables EQ (the "equation") and PPAR (the plot parameters) are stored into the current directory. Likewise, whenever you use the SOLVE application, a copy of EQ and the "unknown" variable are stored in the current directory. In this way, EQ and, say, 'X' can appear in different directories with different contents. Since the contents of EQ and PPAR are automatically updated whenever the PLOT application is used, it is usually not important to purge them. On the other hand, a variable like 'X', which is the default independent variable for graphing, should be purged from the current directory immediately after it is used. Keep in mind, also, that when you purge 'X' from a particular directory it may continue to exist in an "ancestral" directory where it may cause trouble later on. For example, suppose that MTRX is

the current directory, that no variable 'X' is stored in MTRX, but that the parent directory HOME contains the variable 'X' in which the value 2 is stored. Suppose further that you wish to take the symbolic derivative of a function f with respect to the independent variable 'X'. Because 'X' appears in the parent directory, the derivative will be automatically evaluated at the value $x = 2$. This is because the HP-48 always searches *upward* in the directory tree in search of variables; it does *not* search for variable in directories that are on the same level as, or below, the current directory. And, having found that HOME contains variable 'X' with the value 2, the derivative at $x = 2$ was returned. Had the calculator found no value for 'X', it would have treated 'X' symbolically, as was desired. Moral: purge all 'X"s.

The Matrix Commands

The HP-48G series calculators contain an impressive variety of commands for working with matrices. Many of these commands execute professional level code constructed from the new LAPACK Fortran library of matrix routines.

The MTH MATR menu is the place to look. In addition to such commands as LSQ (for obtaining least squares solutions) and EGV (for finding eigenvalues and associated eigenvectors), this menu includes five submenus, each containing commands that are thematically linked.

- The MAKE submenu, whose commands are useful for making special kinds of matrices and for manipulating matrix entries.
- The NORM submenu, whose commands produce various matrix norms, the spectral radius, an estimate of the condition number, the rank, determinant and trace.
- The FACTR submenu, containing commands for the RREF, LU, LQ, QR, and Schur factorizations, as well as the singular value decomposition (SVD).

- The ROW submenu, with commands for executing various operations with rows of a matrix.
- The COL submenu, with commands for executing various operations with columns of a matrix.

To execute one of the built-in matrix commands, say DET (determinant), for a matrix on level 1 of the stack, you open the appropriate submenu that contains the command (in this case, the second page of the MTH MATR NORM menu) and press the associated white key. But to keep from having to open several submenus, you can simply hold down the [α] key, type the name of the command DET, and then press [ENTER] to execute. We shall say more about using the built-in commands as we encounter them.

2 ARRAYS

Rectangular arrangements of real or complex numbers are recognized by the HP-48G/GX as *arrays*. Arrays can be one-dimensional (vectors) or two-dimensional (matrices) and are considered to be single objects. They can be manipulated with many of the same basic commands used in ordinary arithmetic. We shall begin by examining some ways of entering, editing, and manipulating arrays.

2.1 ENTERING ARRAYS

A one-dimensional array (vector) is represented on the calculator by enclosing a sequence of real or complex numbers in square brackets, as in [1 2 3] or [(1, 2) (3, 4) (5, 6)]. A two-dimensional array (matrix) is distinguished by an initial square bracket [, followed by each row vector, and ends with a closing square bracket] . For example, in standard display mode the 2×3 real matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \text{ will appear as } \begin{matrix} [\,[\,1\ \ 2\ \ 3\,] \\ [\,4\ \ 5\ \ 6\,]\,] \end{matrix}.$$

Similarly, the 3×2 complex matrix

$$\begin{bmatrix} 1+i & 1+2i \\ 2+i & 2+2i \\ 3+i & 3+2i \end{bmatrix} \text{ will appear as } \begin{matrix} [\,[\,(1,\,1),\,(1,\,2)\,] \\ [\,(2,\,1),\,(2,\,2)\,] \\ [\,(3,\,1),\,(3,\,2)\,]\,] \end{matrix}.$$

There are several ways to enter arrays: directly from the keyboard, with the built-in MatrixWriter, or as a dimensioned sequence of numbers. The built-in random matrix generator can also be used.

Direct Keyboard Entry

The vector [1 2 3] is entered with keystrokes ◀ [ ] 1, 2, 3 ENTER. Although we show commas to separate the three numbers, you should instead insert spaces with the SPC key. To enter a matrix, start with [[by pressing the [ ] key twice (the left-shifted × key), enter the first row and press ▶ to move the cursor beyond the innermost bracket, then continue entering the remaining entries in row order and press ENTER.

EXAMPLE: Keystrokes [ ] [ ] 1, 2, 3 ▶ 4, 5, 6, 7, 8, 9 ENTER will produce the following matrix:

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

The ▶ key simply defines the number of columns. Now press DROP to drop this matrix from the stack. (When no command line is present you need not press the ◀ to DROP.)

To put the complex number (1, 2) on the stack, use the left-shifted ÷ key to get the double parentheses ( ), and press 1 SPC 2 ENTER. When building a matrix with complex numbers, use the right cursor key ▶ to move beyond each right parenthesis. The numbers can be any mixture of real or complex numbers (ordered pairs), but if any one entry is complex then the entire array will be complex. Also, you need not insert spaces between two complex numbers or between a real and a complex number. Try entering the following matrix:

```
[ [ (1,2)  (3,4) ]
  [ (5,0)  (0,6) ] ]
```

Using the MatrixWriter

Enter the MatrixWriter application by pressing [↱] [MATRIX]. This activates a spreadsheet-type display, with a dark cursor resting in the 1-1 position. Check to see that the [GO⇨□] command is active by noting a small white box within this menu label (if the box is not present, simply press the white key beneath the [GO⇨] label to activate it.) Key in the numbers of the first row of the matrix in row order separated by spaces and then press [ENTER]. When you are ready to go to the second row press [▽]. This will define the number of columns and position the cursor at the 2-1 entry. Now key in the remaining entries of the matrix in row order (separated by spaces) and press [ENTER]. A final [ENTER] will put the matrix onto the stack.

EXAMPLE. The keystrokes [↱] [MATRIX] 1, 2, 3 [ENTER] [▽] 4, 5, 6, 7, 8, 9 [ENTER] [ENTER] will produce this matrix on the stack:

```
[[ 1  2  3 ]
 [ 4  5  6 ] .
 [ 7  8  9 ]]
```

Clearly, for entering simple matrices (say, with small integer entries) the command line is faster and easier to use than the MatrixWriter application. But the MatrixWriter has the advantage that for more complicated matrices, an entry can be calculated (using RPN syntax) on the command line within the MatrixWriter environment before it is entered into its position. As an example, construct the numerical approximation to the matrix

$$\begin{bmatrix} \sqrt{17} & \ln 3 \\ e & \pi/2 \end{bmatrix}.$$

Although the term MatrixWriter suggests that it can be used only for matrices, it is actually an environment for entering, reviewing and editing both *vectors and matrices*. To enter a vector using the MatrixWriter, say vector [1 2 3 4], clear the stack and enter the MatrixWriter environment with [↱] [MATRIX]. Note that the menu key [VEC□] appears. If you press 1 2 3 4 (separate the digits with spaces) [ENTER] [ENTER], the *vector* [1 2 3 4] will show on the stack. The presence of the white box in [VEC□] indicates that vector entry is active. If you toggle off this key to see [VEC] without the box, the keystrokes 1, 2, 3, 4 [ENTER] [ENTER] will return the *matrix* [[1 2 3 4]]. Although in mathematics we identify this matrix with its single row vector, they are different objects as far as the calculator is concerned.

Whenever you enter the MatrixWriter with [↱] [MATRIX], the vector entry mode [VEC□] is active by default. But if there is an array on level 1 and you enter the MatrixWriter by pressing [∇] to review that array, the status of [VEC] reflects the nature of the array: [VEC□] for a vector and [VEC] for a matrix. Try it for yourself with [1 2 3 4] and with [[1 2 3 4]]. Finally, note that you can quickly convert the vector [1 2 3 4] to the matrix [[1 2 3 4]] and vice-versa by starting with either one on level 1, pressing [∇] to enter the MatrixWriter, then changing the status of [VEC] and pressing [ENTER].

A final note about entering arrays using the MatrixWriter application. Array entries can be real or complex numbers, but when you use the MatrixWriter to initially enter a matrix into the calculator, the array object type (real or complex) is determined by the 1-1 entry. Thus, if the 1-1 entry is real, you cannot enter a subsequent entry as a complex number. But, if the 1-1 entry is a complex number (an

ordered pair), any subsequent entry of a real number x will be accepted and written as the complex number (x, 0).

As a Dimensioned Sequence

Enter the numbers into the command line from left-to-right in row order separated by spaces, then the dimensions as a list, {no. rows, no. columns}, and press ENTER to place all this on the stack. Then press the menu key ⇨ARR (on the PRG TYPE menu).

EXAMPLE: Keystrokes 1, 2, 3, 4, 5, 6 { } 2, 3 ENTER ⇨ARR

return the matrix

```
[[ 1  2  3 ]
 [ 4  5  6 ]] .
```

Keystrokes 1, 2, 3, 4, 5, 6 { } 6 ENTER →ARR

return the vector [1 2 3 4 5 6].

The Random Matrix Generator

The random matrix generator, activated by the command RANM on the MTH MATR MAKE submenu will generate an array with entries from the set Z_{10} = { 0, ±1, ±2, ..., ±9 }. The size of the array is specified by an appropriate input list, a singleton list { n } for a vector and a list { m n } for an $m \times n$ matrix.

The RANM command calls upon the calculator's random number generator to construct a random matrix with a random assignment of ± signs to the entries. The calculator command RAND (found on the MTH PROB menu) generates uniformly distributed pseudo-random numbers x, where each x lies in the range $0 < x < 1$. Each execution of RAND returns a value calculated from a *seed* based upon the previous RAND value, and the seed can be changed by using the command RDZ (adjacent to

RAND in the MTH PROB menu). RDZ takes a real number z as a seed for the RAND command. If z is 0, the seed is based upon the system clock. After a complete memory reset, a built-in seed is used.

For example, begin by seeding the random number generator with 2: press 2 RDZ. Then { 4 5 } RANM will return the matrix

```
[[ 4 -2  5 -8  5 ]
 [-4  7  8  0 -6 ]
 [-4  0  8 -5 -2 ]
 [ 5  6  0  2  3 ]]
```

Now use { 6 } RANM to obtain [0 4 0 -9 0 -8].

Special Arrays

To build the identity matrix of order n, use the command IDN preceded by the number n that specifies the order. Thus, 3 IDN returns

```
[[1  0  0]
 [0  1  0]
 [0  0  1]]
```

to level 1 of the stack. When a square matrix is on level 1, the command IDN by itself will replace that matrix with the identity matrix of the same order. The IDN command appears on the MTH MATR MAKE submenu, but as with most simple commands, it is easy to simply type IDN and press ENTER to execute the command.

An array whose entries are all equal to the same constant c (a real or complex number) can be built using the CON command. For example,

```
                2: { 4 }
                1:   2        CON returns [ 2  2  2  2 ]

while:          2: { 2  3 }                 [ [ 5  5  5 ]
                1:   5        CON returns   [ 5  5  5 ] ].
```

Similar to the IDN command, you can replace a matrix on level 1 with a constant matrix by specifying only the constant and executing CON. But note that if the matrix has only real number entries then the constant must also be a real number.

It is occasionally helpful to generate matrices of a special type: diagonal, tridiagonal, triangular or symmetric. Such matrices can be readily generated with the following calculator programs.

DIAG: builds a random diagonal matrix over Z_{10}

U.TRI: builds a random upper-triangular matrix over Z_{10}

L.TRI: builds a random unit lower-triangular matrix over Z_{10}

TRIDIA: builds a random tridiagonal matrix over Z_{10}

SYMM: builds a random symmetric matrix over Z_{10}

HILB: builds a Hilbert matrix

Each of these programs uses the calculator command RAND to construct a random matrix of the desired type.

For classwork, it is often convenient to begin a particular discussion, example or exercise by having everyone in the class use the same non-zero seed for their random number generator. In this event, subsequent synchronous use of the RAND command by the class members will result in a common sequence of random numbers. Such will

occur, for example, with a common non-zero seed and then synchronous use of any of the above six programs. Thus, with only a few simple keystrokes, each member of the class can generate the same random matrix. I have found this to be an effective classroom procedure. Here are the six programs with illustrations of their use. They should all be stored in a BILDR subdirectory of a user-defined MTRX directory (see the **Memory Management** section of Chapter 1).

DIAG (Diagonal Matrix Generator)

Input: level 1: an integer n

Effect: returns a random n by n diagonal matrix over Z_{10} with a random assignment of ± to the entries.

```
« DUP 1 →LIST RANM SWAP DUP 2 →LIST DIAG→ »
```

EXAMPLE. Press 5 RDZ to use the seed that begins this example, then press 4 DIAG to generate

```
[[ -5  0  0  0 ]
 [  0  1  0  0 ]
 [  0  0  6  0 ]
 [  0  0  0 -7 ]]
```

U.TRI (Upper Triangular Matrix Generator

Input: level 1: an integer n

Effect: returns a random n by n upper triangular matrix over Z_{10} with a random assignment of ± to the entries.

```
« → n « 1 n FOR I 1 N FOR J IF I J > THEN 0 ELSE RAND 10
* FLOOR RAND 10 * FLOOR → X « X 5 < -1 1 IFTE » EVAL *
END NEXT NEXT { n n } →ARRY » »
```

EXAMPLE. Press 4 RDZ to use the seed that begins this example, then press 4 U.TRI to generate

$$\begin{bmatrix} -8 & 8 & 9 & -4 \\ 0 & -9 & 0 & 1 \\ 0 & 0 & 4 & -4 \\ 0 & 0 & 0 & 2 \end{bmatrix}.$$

L.TRI (Unit Lower Triangular Matrix Generator)

Input: level 1: an integer n

Effect: returns a random n by n unit lower triangular matrix over Z_{10} with a random assignment of ± to the entries.

```
« → n « 1 n FOR I 1 n FOR J IF I J ≤ THEN 0 ELSE RAND 10 *
FLOOR RAND 10 * FLOOR → X « X 5 < -1 1 IFTE » EVAL *
END NEXT NEXT { n n } →ARRY DUP IDN + » »
```

EXAMPLE. Press 3 RDZ to use the seed that begins this example, then press 4 L.TRI to generate

```
[[ 1   0   0   0 ]
 [ 1   1   0   0 ]
 [-5   2   1   0 ]
 [-1   2  -7   1 ]]
```

TRIDIA (Tridiagonal Matrix Generator)

Input: an integer *n*

Effect: returns a random *n* by *n* tridiagonal matrix over Z_{10} with a random assignment of ± to the entries

```
« → n « 1 n FOR I 1 n FOR J IF I J - ABS 1 > THEN 0 ELSE
RAND 10 * FLOOR RAND 10 * FLOOR → X « X 5 < -1 1 IFTE »
EVAL * END NEXT NEXT { n n } →ARRY » »
```

EXAMPLE. Press 3 RDZ to use the seed that begins this example, then press 5 TRIDIA to generate

```
[[ 1  -5   0   0   0 ]
 [ 2  -1   2   0   0 ]
 [ 0  -7   4   1   0 ]
 [ 0   0  -7   9   1 ]
 [ 0   0   0   3   5 ]]
```

SYMM (Symmetric Matrix Generator)

Input: level 1: an integer *n*

Effect: returns a random *n* by *n* symmetric matrix over Z_{10} with a random assignment of ± to the entries.

Required program: DIAG

```
« DUP → n « 1 n FOR I 1 n FOR J IF I J ≥ THEN 0 ELSE RAND
10 * FLOOR RAND 10 * FLOOR → X « X 5 < -1 1 IFTE » EVAL
* END NEXT NEXT { n n } →ARRY DUP TRN » 3 ROLL DIAG + +
»
```

EXAMPLE. Press 1 RDZ to use the seed which begins this example, then press 5 SYMM to generate

```
[[ -7   7  -9  -8  -5 ]
 [  7   7   8  -1   0 ]
 [ -9   8   1   5   3 ]
 [ -8  -1   5  -2  -3 ]
 [ -5   0   3  -3  -5 ]]
```

HILB (Hilbert matrix Generator)

Input: level 1: an integer *n*

Effect: returns a 12-digit approximation to the *n* by *n* Hilbert matrix.

```
« → n « 1 n FOR I 1 N FOR J I J + 1 - INV NEXT NEXT { n n }
→ARRY » »
```

EXAMPLE. Press 4 HILB to see the approximation to the 4×4 Hilbert matrix

```
[[ 1    1/2  1/3  1/4 ]
 [ 1/2  1/3  1/4  1/5 ]
 [ 1/3  1/4  1/5  1/6 ]
 [ 1/4  1/5  1/6  1/7 ]]
```

Activity Set 2.1

1. Set the number display mode in your 48G/GX to STD and enter each of the following arrays using direct keyboard entry. After each array is put on level 1 of the stack, you can see any hidden entries by pressing ∇ to view the array in the Matrix Writer environment, where the cursor keys will enable you to move to any position. The entry in the cursor position is identified on the command line.

(a)
```
[ -11  12  -13  14  -15  16  -17  18 ]
```

(b)
```
[[ 3  -1  2 ]
 [ -4  0  1 ]
 [ 2  -3  5 ]]
```

(c)
```
[[ 4   2  -1   7 ]
 [ -5  0   3   8 ]
 [ -6  9  -2   5 ]
 [ 0  -4   1  -3 ]]
```

(d)
```
[[ -9   8  -7 ]
 [  6  -5   4 ]
 [ -3   2  -1 ]
 [  0  -1   2 ]
 [ -3   4  -5 ]
 [  6  -7   8 ]]
```

(e)
```
[[ -3  -2 ]
 [ -1   0 ]
 [  1   2 ]
 [  3   3 ]
 [  5   6 ]
 [  7   8 ]
 [  9   0 ]]
```

(f)
```
[[ -9   6  -3   0  -3  -6 ]
 [  8  -5   2  -1   4   7 ]
 [ -7   4  -1   2   5   8 ]]
```

(g)
```
[[ (6, -5)  (-2,-1)  (4, -8)  ]
 [ (-3, 0)  (7, 9)   (0,-11) ]]
```

2. Practice entering each of the arrays in ACTIVITY 1 as a dimensioned sequence.

3. Enter each of the following arrays using the Matrix Writer. View any hidden entries. If part of an entry on the command line is still hidden, press the menu key EDIT and then use the right cursor key to scroll through the entry.

(a)
```
[[ -2   4   0   1 ]
 [  3  -5   2   7 ]
 [  1   3  10  -6 ]
 [ -4   5   1  -1 ]]
```

(b)
```
[[ -3  -8 ]
 [  1  -1 ]
 [ -4   6 ]
 [ -2  -7 ]
 [  5  -4 ]
 [ -8   5 ]
 [  6   0 ]]
```

(c)
```
[[ (-1, 0)  (1, -2)  (-3, 4) ]
 [ (5, -6)  (-7, 8)  (0, -9) ]]
```

(d)
```
[[ -4   3   -2   1 ]]
```

(e)
```
[[ -4 ]
 [  3 ]
 [ -2 ]
 [  1 ]]
```

(f) [[$e^{4.1}$ $\sqrt{11}$]
[$\pi/3$ $SIN\ 4$]]

(g) [[$ln(-2)$ $\sqrt{-2}$ $cos^{-1}(-2)$]]

2.2 EDITING ARRAYS

When using the HP-48G/GX, it is often necessary to edit arrays by changing some of their entries, redimensioning, separating into rows or columns, inserting new rows or columns, or applying a mathematical function to the entries of an array.

To Disassemble an Array

The calculator command OBJ→, which appears as a menu key on the PRG TYPE submenu, will disassemble an array into its component entries and indicate the dimension(s) of the array. Thus OBJ→ is the inverse command to →ARRY.

EXAMPLE. With [0 2 4 6] on level 1 the command OBJ→ will return the following stack arrangement:

5: 0

4: 2

3: 4

2: 6

1: { 1 }

With the following matrix on level 1,

```
[[ 2 4 ]
 [ 6 8 ]]
```

the command OBJ→ will return the following stack arrangement:

5: 2

4: 4

3: 6

2: 8

1: { 2 2 }

To Redimension an Array

The command used to redimension an array is RDM. A menu key for it appears on the MTH MATR MAKE submenu. Entries are taken from the original array in row order and are reassembled in that same order into a new array whose dimensions are specified by an input list. Any excess entries from the original array are discarded, and if there are too few entries in the original array then the new array is finished with zeros.

EXAMPLE. With the stack arrangement

2: [0 2 4 6 8]

1: { 3 }

the command RDM returns [0 2 4]. With

2: [0 2 4 6 8]

1: { 3 2 }

the command RDM returns the matrix

```
[[ 0 2 ]
 [ 4 6 ]  .
 [ 8 0 ]]
```

With the stack arrangement:

```
2: [[ 1  2  3 ]
    [ 4  5  6 ]
    [ 7  8  9 ]]

1: { 2  4 }
```

the command RDM returns the matrix

```
[[ 1  2  3  4 ]
 [ 5  6  7  8 ]]
```

Changing Entries

There are two ways to change entries in an array.

(i) You can copy the array from level 1 to the command line with EDIT (the ↰ +/– key), where the white cursor keys then let you move to any desired entry and change it. You can use the DEL key to delete characters, then simply key in the new characters. Return the edited matrix to level 1 with ENTER.

(ii) You can copy the array into the MatrixWriter with ∇, position the cursor over the entry to be changed, key the new entry into the command line and press ENTER to insert it at the cursor location. Return to the stack with another ENTER. This method is especially useful because you can calculate the new entry on the command line in RPN before entering it.

EXAMPLE: Begin with the following matrix on level 1

```
[[ 1  2  3 ]
 [ 4  5  6 ]]
```

Press EDIT, move the cursor over the 2 and press DEL to delete, then do 7 +/– ENTER to see

```
[[ 1 -7  3 ]
 [ 4  5  6 ]
```

Now press ∇ to view the last matrix in the Matrix Writer, position the cursor over the six and do 5 √x 2 ÷ ENTER ENTER to replace the 6 by a 12 digit approximation to $\sqrt{5}/2$.

Separating into Rows or Columns

To separate a matrix into its row or column vectors, the appropriate commands are →ROW, located on the MTH MATR ROW menu, and →COL, located on the MTH MATR COL menu. For example, with

```
[[ 0  2  4 -6 ]
 [ 5 -1  8  3 ]
 [-7  9 -4  2 ]
 [ 6 -3  5  8 ]]
```

on stack level 1, press →ROW to separate the matrix into its four row vectors. Notice that stack level 1 contains the number of rows. Press Δ five times to see the first row vector on level 5, then press ON to return to the normal stack environment.

The inverse commands to →ROW and →COL are ROW→ and COL→, located next to the →ROW and →COL commands on the appropriate menus. With four vectors on levels 1 through 4, simply press 4 COL→ to build the matrix having the four vectors as columns:

```
4: [ 0  2  4 -6 ]
3: [ 5 -1  8  3 ]
2: [-7  9 -4  2 ]
1: [ 6 -3  5  8 ]
```

4 COL→ returns

```
[[ 0  5 -7  6 ]
 [ 2 -1  9 -3 ]
 [ 4  8 -4  5 ]
 [-6  3  2  8 ]]
```

Deleting and Inserting Rows or Columns

The commands ROW– and ROW+, located on the MTH MATR ROW menu, can be used to delete and to insert rows. Analogous commands for column deletion and insertion, COL– and COL+, appear on the MTH MATR COL menu. For example, with

```
[[ 9 -7  5  0 ]
 [ 3  1  3  6 ]
 [-3  5 -6 -9 ]
 [ 4 -1 -3  5 ]
 [-8  0  2  7 ]]
```

on stack level 1, press 2 ROW– to delete row two. Notice that the diminished matrix

```
[[ 9 -7  5  0 ]
 [-3  5 -6 -9 ]
 [ 4 -1 -3  5 ]
 [-8  0  2  7 ]]
```

appears on level 2 and the deleted row [3 1 3 6] appears on level 1. Then, to insert this deleted row as the third column of the diminished matrix, press 3 COL+:

```
[[ 9 -7  3  5  0 ]
 [-3  5  1 -6 -9 ]
 [ 4 -1  3 -3  5 ]
 [-8  0  6  2  7 ]]
```

More generally, the ROW+ and COL+ commands can be used to insert all of the rows, or columns, of one matrix into another matrix at a specified position. With

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \text{ on level 2 and } B = \begin{bmatrix} 11 & 12 \\ 13 & 14 \\ 15 & 16 \end{bmatrix} \text{ on level 1,}$$

press 2 [COL+] to insert the columns of B into matrix A, starting at the column 2 position:

$$\begin{bmatrix} 1 & 11 & 12 & 2 & 3 \\ 4 & 13 & 14 & 5 & 6 \\ 7 & 15 & 16 & 8 & 9 \end{bmatrix}.$$

ROW+ works similarly.

Using the Diagonal. The HP-48G/GX units include two commands that are useful in certain special contexts.

- The →DIAG command (on the MTH MATR menu) will extract the main diagonal (as a vector) from any matrix on level 1. For example,

  1: $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ →DIAG returns the vector [1 5].

- The →DIAG command (on the MTH MATR menu) will insert a given vector as the main diagonal of a matrix of specified size, all other entries being zero. For example:

  2: [2 2 2] DIAG→ returns $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$,
  1: 3

2: [2 2 2] DIAG→ returns
1: 4

```
[[2 0 0 0]
 [0 2 0 0]
 [0 0 2 0]
 [0 0 0 0]]
```

, and

2: [2 2 2] DIAG→ returns
1: { 3 2 }

```
[[2 0]
 [0 2] .
 [0 0]]
```

Activity Set 2.2

1. Start with the following matrix on level 1:

$$A = \begin{bmatrix} 3 & 2 & -1 & 5 \\ 0 & -6 & 7 & 3 \\ 0 & 1 & 2 & -4 \\ 8 & 7 & 9 & 5 \\ 2 & -6 & 1 & 3 \end{bmatrix}.$$

   (a) Redimension A into a 4×5 matrix B that preserves row order.

   (b) Disassemble B into its entries, drop the last five entries, and reassemble the remaining entries into a 5×3 matrix C.

   (c) Change the 5 in C to -3, the 8 to 0, and the 9 to 11/8 to get a new matrix D.

   (d) Delete rows 2 and 4 from matrix D to obtain a final matrix E.

2. (a) Build a 5×4 matrix A whose (i, j)-entry is $.ij$.

   (b) Extract the submatrix B consisting of rows 2, 3, and 5.

(c) Redimension B into a 4×3 matrix C that preserves row order.

(d) Extract row 4 of C, change each 5 in this vector to a 6, and insert the result as a new row 1.

3. Enter the following matrix

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}.$$

Enlarge A by inserting an additional row on the bottom and an additional column on the right. Do this as follows:

(a) Insert a row of 4's, then a column of 5's.

(b) Now start over with A, and first insert a column of 5's, than a row of 4's.

(c) Are the results in (a) and (b) the same?

4. Enter and store A = [[1 -3 4] [2 5 0] [6 -3 4]] and B = [[7 0 -1] [5 3 2] [9 -6 0]].

(a) Use the ROW+ and COL+ commands to build the block matrices [A B] and $\begin{bmatrix} A \\ B \end{bmatrix}$.

(b) Build the block matrices $\begin{bmatrix} A & O \\ O & B \end{bmatrix}$ and $\begin{bmatrix} A & B \\ I & O \end{bmatrix}$ (*hint:* the command 3 IDN, on the MTH MATR MAKE menu, will build the identity matrix of order 3).

5. Seed your calculator's random number generator with 5 and use the RANM command to build a 3×4 matrix A. Now use RANM to generate a 3×2

matrix B. Insert B into A immediately after column 2 of A to obtain a new 3×6 matrix C.

6. Seed your calculator's random number generator with 6 and use program **SYMM** to generate a 3×3 symmetric matrix A. Then use program **TRIDIA** to generate a 3×3 tridiagonal matrix B. Insert B into A immediately after row 1 of A to obtain matrix C. Delete rows 1 and 3 of C to form matrix D. Redimension D into a 3×4 matrix E that preserves row order.

2.3 ARRAY ARITHMETIC

Addition and Subtraction

Addition and subtraction of arrays proceeds just as for real numbers. To calculate the sum $A + B$ of two arrays having the same dimension, arrange the stack like this (so, normally, A is entered first):

2: A
1: B

Now press [+]. Press [−] instead of [+] to calculate $A - B$. Note that the commands [+] and [−] add or subtract the object on level 1 to or from the object on level 2. In case A and B are stored as variables in user memory, press [𝔸] [𝔹] [+] to add.

Scalar Multiplication

To multiply an array by a scalar c, put the array and the scalar on levels 1 and 2 of the stack (in either order) and press [×]. Multiplying by -1 can be done in a single keystroke with the [+/−] key.

Dot Products and Length

The dot product of two real or complex vectors $[\ x_1\ \ x_2\ \ldots\ x_n\ ]$ and $[\ y_1\ \ y_2\ \ldots\ y_n\ ]$ is the number $\sum_{i=1}^{n} x_i y_i$. Put the two vectors on stack levels 1 and 2 and execute the command DOT. A menu key [DOT] is located on the MTH VECTR menu.

2: [4 3 1 2]
1: [-1 2 3 4]

DOT returns 13

2: [(-1, 2) (3, 4)]
1: [(1, 1) (0, 2)]

DOT returns (-11, 7)

To obtain the Hermitian product $\sum_{i=1}^{n} x_i \bar{y}_i$ or $\sum_{i=1}^{n} \bar{x}_i y_i$ of two complex vectors $[\ x_1\ \ x_2\ \ldots\ x_n\ ]$ and $[\ y_1\ \ y_2\ \ldots\ y_n\ ]$, (where the bar denotes complex conjugation) you must first conjugate the appropriate vector with CONJ (on the MTH CMPL menu) before executing DOT.

2: [(-1, 2) (3, 4)]
1: [(1, 1) (0, 2)]

CONJ DOT returns (9, -3).

Here, we conjugated the vector on level 1.

For a real or complex vector $[\ x_1\ \ x_2\ \ldots\ x_n\ ]$ the command ABS (see the MTH VECTR menu) will calculate the Euclidean length (norm)

$$\| x \| = \sqrt{|x_1|^2 + |x_2|^2 + \ldots + |x_n|^2},$$

which is the usual notion of length in R^n or C^n. In the complex case, $|x_i|^2$ is the square of the modulus of the complex number x_i.

1: [2 4 5 2] ABS returns 7.

Matrix Multiplication

To calculate a matrix product AB, proceed as in forming $A + B$ but press [×] instead of [+]. Note that in calculating AB, matrix A must be on level 2 and matrix B on level 1. The number of columns of the matrix on level 2 must equal the number of rows of the matrix on level 1.

EXAMPLE: Begin this example by seeding your random number generator with 1: 1 RDZ. Then put a matrix A on level 1 with { 2 3 } RANM:

```
[[ 7 -9 -8 ]
 [-5  8 -1 ]]
```

Put a matrix B on level 1 with { 3 4 } RANM, moving A to level 2:

```
[[ 0  5  3 -3 ]
 [-7  7  1 -2 ]
 [-5 -2  9 -7 ]]
```

Press [×] to see

$$AB = \begin{matrix}[[\ 103 & -12 & -60 & 53\ ] \\ [\ -51 & 33 & -16 & 6\ ]]\end{matrix}$$

Matrix by Vector Multiplication

As a matter of convenience, the HP-48G series calculators will let you premultiply any n-vector $x = [\ x_1\ \ x_2\ \ \ldots\ x_n\ ]$ by any $m \times n$ matrix A to obtain Ax. Thus, in this context, vector x is treated as if it were an $n \times 1$ matrix. But you should note that this treatment of x is peculiar to this context: *in all other applications, x is a vector,* not a matrix. You can not, e.g., perform a multiplication like xA, nor can you transpose x or take the determinant of a 1-vector [x]. Transposing

and finding determinants are operations to be performed on matrices and not on vectors.

Matrix Powers

Unlike the case for real or complex numbers, you cannot use the y^x key to calculate powers of a square matrix A. You can, however, obtain A^2 by using the x^2 key or by executing the command SQ. For more general powers of A, say A^k where $k = 1, 2, 3, \ldots$, you can use the following program.

A↑k (k^{th} power of a matrix)

Inputs: level 2: a square matrix
level 1: an integer k

Effect: returns the k^{th} power of the matrix

```
« → A k « A SIZE 1 GET IDN 1 k FOR I A * NEXT » »
```

EXAMPLE. Calculate B^5 for the following matrix

$$B = \begin{bmatrix} 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \end{bmatrix}.$$

Begin by entering the matrix B onto level 1. Now press 5 $A\uparrow k$ to see

$$B^5 = \begin{bmatrix} 0 & 16 & 0 & -16 \\ 16 & 0 & -16 & 0 \\ 0 & -16 & 0 & 16 \\ -16 & 0 & 16 & 0 \end{bmatrix}.$$

CAUTION: You must use caution when calculating powers of a matrix. Because your calculator only shows 12 digit mantissas, powers of even *small* matrices may lead to computational inaccuracies. For example, if

$$A = \begin{matrix}[[9 & 9 & 9 & 9] \\ [9 & 9 & 9 & 9] \\ [9 & 9 & 9 & 9] \\ [9 & 9 & 9 & 9]]\end{matrix},$$

then A^8 can be found correctly on the calculator to be the constant 4×4 matrix whose entries are $4^7 * 9^8 = 705{,}277{,}476{,}864$. But A^9 has entries $4^8 * 9^9$, a number which the calculator can only represent as 2.5389989167E13, but which is somewhat short of the actual 2.5389989167104E13.

More generally, given a square matrix A and an arbitrary polynomial $p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$, we sometimes want to find $p(A) = A^n + a_{n-1}A^{n-1} + \ldots + a_1 A + a_0 I$. The following program, P.of.A, does just that.

P.of.A (Polynomial evaluation at *A*)

Inputs: level 2: a vector [a_n a_{n-1} ... a_1 a_0] of coefficients

level 1: a square matrix *A*

Effect: returns $p(A) = a_nA^n + a_{n-1}A^{n-1} + \ldots + a_1A + a_0I$

```
« → v A « A SIZE 1 GET → k « v 1 GET  2 v SIZE
OBJ→ DROP FOR n A *  v n GET k IDN *  +
NEXT » » »
```

EXAMPLE. Find $p(A)$ for $p(x) = 1.3x^5 - .4x^4 + 2.1x^2 + 5x + 6.2$ and

$$A = \begin{matrix}[\,[\,.1 & .2 & .3 & .4\,] \\ [\,.5 & .6 & .7 & .8\,] \\ [\,.9 & .8 & .7 & .6\,] \\ [\,.5 & .4 & .3 & .2\,]\,]\end{matrix}.$$

Enter the coefficients as a vector [1.3 -.4 0 2.1 5 6.2]. Next enter matrix A. Set 3 FIX display mode and press [ℙ.𝕠𝕗.𝔸] to see

$$p(A) = \begin{matrix}[\,[\,12.455 & 6.677 & 7.099 & 7.521\,] \\ [\,16.975 & 23.597 & 17.819 & 18.241\,] \\ [\,20.545 & 20.123 & 25.901 & 19.279\,] \\ [\,9.825 & 9.403 & 8.981 & 14.759\,]\,]\end{matrix}.$$

Transpose and Trace

With a matrix on level 1, the command TRN, returns the conjugate transpose *i.e.*, the conjugate of the transpose (a menu key is located on the MTH MATR MAKE menu). Thus, if the matrix on level 1 is real, TRN returns its ordinary transpose. To obtain the ordinary transpose of a complex matrix, use TRN then CONJ. The [ℂ𝕆ℕ𝕁] key, on the MTH CMPL menu, returns the complex conjugate of its input argument.

The command TRACE will return the trace (the sum of the main diagonal entries) of a square matrix. A menu key for it appears at the end of the MTH MATR NORM menu.

EXAMPLE: Put the following matrix on level 1:

$$\begin{matrix}[\,[\,(2,3) & (7,-4) & (3,0)\,] \\ [\,(0,-1) & (2,0) & (5,1)\,]\,]\end{matrix}$$

To obtain the conjugate transpose, execute TRN:

```
[ [ (2, 3)   (0, -1) ]
  [ (7,-4)   (2,  0) ]
  [ (3, 0)   (5,  1) ]
```

Activity Set 2.3

1. Seed your calculator's random number generator with 1, and use **RANM** to build a 3×2 matrix A, then a 4×3 matrix B, and calculate BA.

2. (a) Create a 5×4 matrix $A = (a_{ij})$ where $a_{ij} = i - j$.

   (b) Extract the submatrix B consisting of rows 2, 3 and 5.

   (c) Remove col 3 from B to obtain matrix C.

   (d) Calculate C^2 and C^3.

3. Start with matrix

$$A = \begin{bmatrix} 1 & 0 & -2 & 3 \\ 2 & -3 & 0 & -1 \\ 5 & -2 & 4 & 1 \end{bmatrix}.$$

   (a) Build the matrix B whose first two rows are columns 2 and 3 of A, respectively, and calculate BA.

   (b) Now let C be the submatrix of A consisting of columns 1 and 4 of A; calculate CB.

4. (a) Generate a random 3×4 matrix A over Z_{10} and calculate A^TA; carefully observe your result.

   (b) Repeat part (a) using random 4×5 and 5×6 matrices.

   (c) Formulate a conjecture based upon your observations.

(d) Prove your conjecture.

5. (a) Seed your calculator's random number generator with 1, then generate two random 4×4 matrices A and B with the RANM command.

(b) Combine A and B into the complex matrix $A + iB$ by executing the command $R \rightarrow C$, found on the MTH CMPL menu; transpose $A + iB$.

(c) Separate your answer in (b) into its real and imaginary parts with the command $C \rightarrow R$ (also on the MTH CMPL menu), SWAP levels 1 and 2 and then recombine the two matrices into a complex matrix with $R \rightarrow C$. Now extract column 4.

6. Enter and store the following matrices:

$$A = \begin{bmatrix} 1 & -2 & 3 & 5 & 4 \\ 7 & 9 & 0 & -1 & 3 \\ -3 & 8 & 6 & 2 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 6 & 2 & -1 \\ 3 & 5 & 4 \\ -2 & 8 & 0 \\ 7 & 1 & 6 \\ 1 & -3 & 2 \end{bmatrix}.$$

(a) Get the submatrix C of B consisting of rows 2 and 4.

(b) Form the block matrix $[\, A \; C^T \,] = D$ and get the submatrix E consisting of the odd-numbered columns.

7. Enter and store the following matrices:

$$A = \begin{bmatrix} (5,1) & (2,-3) & (1,0) \\ (0,4) & (6,-1) & (3,4) \end{bmatrix}, \quad B = \begin{bmatrix} (-3,1) & (6,0) \\ (0,0) & (2,-1) \\ (4,-3) & (1,1) \end{bmatrix}.$$

(a) Find the conjugate transpose A^* of A and the transpose B^T of B.

(b) Calculate $A^* + B$, $A + B^T$, AA^*, B^TB and $(2 - 3i)A$.

8. Find $A^2 - 4A^* + 3A^T - I$ for the following matrix:

$$A = \begin{bmatrix} (2,-1) & (0,-2) & (3,-2) \\ (1,5) & (3,2) & (5,0) \\ (0,1) & (6,0) & (-1,2) \end{bmatrix}.$$

9. Consider the following matrix

$$A = \begin{bmatrix} 7 & 2 & 4 & 6 \\ -6 & -1 & -4 & -4 \\ 4 & 4 & 5 & -2 \\ -16 & -12 & -14 & -3 \end{bmatrix}$$

(a) Find $A^4 - 8A^3 + 22A^2 - 40A + 25I$

(b) Use your result from (a) to find a polynomial in A that gives A^{-1}.

(c) Calculate A^{-1} from your answer in (b).

(d) Check your result from (c).

10. For this exercise, set your calculator to 3 FIX mode. Let

$$x = [\ .1\ \ .2\ \ .3\ \ .4\ ] \text{ and } A = \begin{bmatrix} .3 & .3 & .3 & .2 \\ .4 & .3 & .2 & 0 \\ .1 & .2 & .2 & .3 \\ .2 & .2 & .3 & .5 \end{bmatrix}.$$

(a) Examine the sequence $A, A^2, A^3, \ldots$ to find $\lim_{n\to\infty} \{A^n\}$.

(b) Examine the sequence $Ax, A^2x, A^3x, \ldots$ to find $\lim_{n\to\infty} \{A^n x\}$.

(c) What is the connection between the two limits in (a) and (b)?

11. Repeat parts (b) – (c) of exercise 10 using any vector $x = [a \;\; b \;\; c \;\; d]$ of your choice where $a + b + c + d = 1$.

12. Seed your calculator's random number generator with 2 and generate two vectors in R^5. Store the first one as u and the second one as v. Now find:

    (a) $u \bullet v$ (b) $u \bullet (u + v)$ (c) $v \bullet (v - u)$

    (d) the length of $\dfrac{u - v}{\| u - v \|}$

    (e) Verify the triangle inequality: $\| u + v \| \leq \| u \| + \| v \|$.

    (f) Verify the Cauchy-Schwarz inequality:

$$|u \bullet v| \leq \| u \| \; \| v \| .$$

2.4 DETERMINANTS AND INVERSES

With a square matrix A on stack level 1, the command DET will return the determinant of A, and pressing 1/x to execute the INV command will return A^{-1} in the event that $detA \neq 0$. A menu key for DET appears on the second page of the MTH MATR NORM menu.

EXAMPLE. Put three copies of the following matrix A on the stack. Then execute DET to show $detA = 256$.

$$A = \begin{bmatrix} -4 & 4 & 8 & 8 \\ -16 & 12 & 16 & 16 \\ -8 & 4 & 12 & 8 \\ 8 & -4 & -8 & -4 \end{bmatrix}$$

Cofactor expansions tell us that a matrix having only integer entries will have an integer for its determinant. As in this example, the HP-48G and 48GX will always

return an integer for the determinant of a matrix having only integer entries if flag -54 is clear (the default case). Use DROP, then 1/x to show

$$A^{-1} = \begin{bmatrix} .75 & -.25 & -.5 & -.5 \\ 1 & -.25 & -1 & -1 \\ .5 & -.25 & -.25 & -.5 \\ -.5 & .25 & .5 & .75 \end{bmatrix} .$$

Finally, press × to check that $AA^{-1} = I$.

However, some care must be exercised with these commands in order to obtain results that are mathematically correct. To make the point, enter two copies of the matrix

$$B = \begin{bmatrix} 1 & 1 & 1 \\ 3 & 6 & 4 \\ 3 & 6 & 4 \end{bmatrix} .$$

With the default setting, its determinant is seen to be zero, and when you try to invert matrix B, you get the message INV error: Infinite Result. But now multiply B by .101031 to obtain matrix

$$C = \begin{bmatrix} .101031 & .101031 & .101031 \\ .303093 & .606186 & .404124 \\ .303093 & .606186 & .404124 \end{bmatrix} .$$

Use ENTER to put two more copies of C on the stack, then execute the DET command to obtain $detC$ = 6.1E–17. Drop the determinant, then use 1/x to get

$$C^{-1} = \begin{bmatrix} 19.7959042274 & 3.33333333333E14 & -3.33333333333E14 \\ -9.89795211371 & 1.66666666667E14 & -1.66666666667E14 \\ 0 & -5.E14 & 5.E14 \end{bmatrix} .$$

Look suspicious? Confirm your doubt by pressing SWAP, then × to show

$$C^{-1}C = \begin{bmatrix} 2 & 2 & 2 \\ -1 & -1 & -1 \\ 0 & 0 & 0 \end{bmatrix}.$$

Matrix C, like B, has two identical rows. Thus, $detC = 0$, so C has no inverse. One thing is clear: *using the calculator to calculate determinants and matrix inverses may yield incorrect results.* As in this example, *the calculator may return a non-zero value (the result of round-of error) for the determinant of a singular matrix and then a ridiculous candidate for an inverse.* The numerical calculation of matrix determinants and inverses is extremely sensitive to round-off error, scaling, and choice of numerical algorithm in a floating point environment. *Thus, our advice is to proceed with caution in a calculator environment and,* whenever possible, *avoid calculating determinants and inverses. If you think that you need to calculate a determinant or an inverse of a numerical matrix then you should think again very carefully about your problem. You can almost certainly reformulate to avoid such calculations.*

To clean up round-off error, we recommend that you round your answer to a desired number n of decimal digits, $1 \le n \le 11$. For example, to round the matrix

```
[[ 1   0   -.000000000001 ]
 [ 0   1    .000000000001 ]
 [ 0   0    .999999999999 ]]
```

to 11 decimal places, simply enter 11 RND to obtain

```
[[1   0   0]
 [0   1   0] .
 [0   0   1]]
```

Activity Set 2.4

1. Here is another example that illustrates the difficulty in numerically calculating determinants. Begin by setting flag -54 (use -54 SF) and entering the following matrix:

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ -3 & 1 & 3 & 6 & 4 & 1 \\ 0 & -1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 6 & 4 & 1 \\ 0 & 1 & 1 & 1 & 1 & 3 \\ 1 & 1 & 3 & 6 & 4 & 1 \end{bmatrix}$$

   (a) Unlike our example in the text discussion, no two rows of A are identical. Multiply A by 1000 and apply the DET command to the result. Does the result (a large integer) seem reasonable? Do you see round-off error?

   (b) Use cofactor expansions along column 1 of A to find the determinant of A. What does this tell you about *det*[1000A]?

   (c) Apply the DET command to matrix A. Use the fact that for any n×n matrix A, $det[kA] = k^n \, detA$ to explain how round-off error led to the result in (a).

   (d) Return flag -54 to its default (clear) state; use -54 CF. Now apply DET to matrix A.

   (e) Go back and read, again, the statements in italics in Section 2.4.

2.5 APPLYING FUNCTIONS TO ARRAYS

Ordinary mathematical functions of a single variable can be applied to each entry of an array. For example, when the function $f(x) = \sqrt{(x+1)^2}$ is applied to

$$A = \begin{matrix}[[\ 1\ -1\ 2\ -2\ ] \\ [\ 3\ -3\ 4\ -4\ ]]\end{matrix}$$

we obtain the matrix

$$B = \begin{matrix}[[\ 2\ 0\ 3\ 1\ ] \\ [\ 4\ 2\ 5\ 3]\ ]\end{matrix}.$$

The HP-48G/GX has no command that will apply a mathematical function to the entries of an array, but it does include a program APLY that will do this. Here is how to access program APLY. Hold down the [α] key, type TEACH and press ENTER. This procedure will load (from ROM) into your current directory a subdirectory named EXAMPLES. Open EXAMPLES, and then the PRGS subdirectory to see a menu key [APLY].

To use program APLY (not to be confused with the command APPLY, which does something else) to apply a mathematical function f to an array, arrange the stack like this:

2: array
1: "procedure for f"

Now run APLY. You have two choices for the "procedure for f":

(i) an RPN program for f

(ii) a user-defined function for f.

To work the above example on the HP-48G/GX, put matrix A on level 1 of the stack and then enter the program

« 1 + SQ √ ».

Now run program APLY to obtain matrix B. Instead of this RPN program you can put a user-defined function, say F, for f in the same directory as APLY and then simply put its name 'F' on level 1 before running program APLY. See Chapter 1 to review user-defined functions.

My recommendation is that you copy APLY into your { HOME } directory, so that you can access it from any subdirectory whatsoever by simply typing APLY. The quickest way to copy APLY into { HOME } is as follows. Use [↱] [APLY] to recall the program to the stack, press ['] [APLY] [ENTER] to put the name 'APLY' on level 1, go to { HOME } and press [STO]. Touch [VAR] to see the copy in your { HOME } directory. (At this point you may wish to purge the EXAMPLES directory.)

Activity Set 2.5

1. Apply the function $f(x) = \cos \pi x$ to the matrix

[[1 2 3]
[4 5 6]
[7 8 9]]

by writing a short RPN program and using program APLY. Note that the command → NUM (the [↰] [EVAL] key) will convert the symbolic π to a numerical value.

2. For a real number x, the command FLOOR will return the greatest integer less than or equal to x. In mathematics, we usually denote the greatest integer less than or equal to x by $\lfloor x \rfloor$. Apply the function $f(x) = \lfloor 2 + \sqrt{x} \rfloor$ to the matrix

```
[[1  2  3]
 [4  5  6] .
 [7  8  9]]
```

3. For a real number x, the command CEIL will return the least integer greater than or equal to x, usually denoted by $\lceil x \rceil$ in mathematics. Apply the function $f(x) = \left\lceil \sqrt{\frac{x+1}{x}} \right\rceil$ to the matrix

```
[[ 1 -2  3]
 [-4  5 -6] .
 [ 7 -8  9]]
```

3 SYSTEMS OF LINEAR EQUATIONS

Systems of linear equations arise in practically every field of mathematical application. Not only must we understand some of the algorithms for their solution, but also some of the surrounding theory. For it is a combination of both algorithms and theory that, when cast in matrix-theoretic terms, foreshadows many of the more sophisticated concepts that lie ahead. For brevity, we shall refer to systems of linear equations as *linear systems* and denote their matrix formulation as $Ax = b$.

The standard methods for dealing with linear systems in introductory linear algebra courses are the elimination methods, consisting of several variants of ***Gaussian elimination*** with back substitution. Many beginning courses blur the distinction between these variants in the interest of expediency. But with an eye toward a subsequent study of linear analysis or numerical methods and the use of professional elimination codes, it is important to distinguish carefully between the traditional Gaussian elimination algorithm, the back substitution process, partial pivoting and Gauss-Jordan reduction. Likewise, it is important to understand Gaussian elimination for square matrices as an algebraic process that factors a matrix A into triangular factors, $A = LU$.

3.1 GAUSSIAN ELIMINATION

In its traditional form, the Gaussian elimination algorithm for solving a square nonsingular linear system $Ax = b$ adds suitable multiples of one equation to the others with the goal of obtaining an equivalent upper triangular system $Ux = b'$. It may be necessary to interchange equations at various times for the elimination process to

continue. Back substitution then solves $Ux = b'$ systematically by solving the last equation for its single unknown, putting this value into the next-to-last equation and solving for the next-to-last unknown, and so on until all values for the unknowns have been determined. All this is usually carried out without reference to the unknowns by working with the augmented matrices $[\ A|\,b\ ]$ and $[\ U|\,b'\ ]$. Computationally, the only source of error is round-off, induced by the computational device itself. It is especially important to view the elimination as an orderly process that proceeds in a top-to-bottom, left-to-right fashion.

Once a basic understanding of Gaussian elimination has been established and several examples have been worked by hand, the calculator can be used to efficiently perform the row operations that transform $[\ A|\,b\ ]$ into $[\ U|\,b'\ ]$.

The HP-48G and 48GX units include built-in commands for row operations on the **MTH MATR ROW** menu. With a matrix A on level 1, the **RCIJ** command is used to multiply row I of matrix A by scalar c and then add the result to row J, and the **RSWP** command is used to interchange rows I and J. The **RCI** command is used to rescale row I by multiplying it by scalar c.

To solve this linear system

$$\begin{aligned} 2x_1 + 3x_2 + 2x_3 - x_4 &= 4 \\ -4x_1 - 6x_2 + x_3 + 2x_4 &= -1 \\ 4x_1 + 8x_2 + 7x_3 + 2x_4 &= -3 \\ 2x_1 + 4x_2 + x_3 - 4x_4 &= 2 \end{aligned}$$

using these commands, begin with the augmented matrix $[\ A|\,b\ ]$ on level 1:

```
[[  2   3   2  -1   4 ]
 [ -4  -6   1   2  -1 ]
 [  4   8   7   2  -3 ]
 [  2   4   1  -4   2 ]]
```

To add 2 times row 1 to row 2, press 2, 1, 2 RCIJ :

```
[[ 2   3   2  -1   4 ]
 [ 0   0   5   0   7 ]
 [ 4   8   7   2  -3 ]
 [ 2   4   1  -4   2 ]]
```

Then do -2, 1, 3 RCIJ followed by -1, 1, 4 RCIJ to finish the elimination in the first column:

```
[[ 2   3   2  -1   4 ]
 [ 0   0   5   0   7 ]
 [ 0   2   3   4  -11 ]
 [ 0   1  -1  -3  -2 ]]
```

Now interchange rows 2 and 3 with 2, 3 RSWP , then complete the elimination in the second column with -.5, 2, 4, RCIJ :

```
[[2   3    2   -1    4 ]
 [0   2    3    4   -11 ]
 [0   0    5    0    7 ]
 [0   0  -2.5   -5   3.5]]
```

A final .5, 3, 4 RCIJ produces the desired triangular system [$U|b'$]:

```
[[ 2   3   2  -1   4 ]
 [ 0   2   3   4  -11 ]
 [ 0   0   5   0   7 ]
 [ 0   0   0  -5   7 ]]
```

Back substitution by hand shows the solution vector to be [7.1 -4.8 1.4 -1.4]. To assist with the back substitution process, we can use the following program BACK.

BACK (Back substitution)

Inputs: level 2: an $n \times n$ invertible upper triangular matrix U

level 1: an n-vector b

Effect: Solves the linear system $Ux = b$ by back substitution. Solves for x_n and halts until you press [↰] [CONT], then backsolves for x_{n-1} and halts, etc. After x_n, x_{n-1}, ..., x_1 are on the stack, a final [CONT] returns x = [x_1, x_2, ..., x_n].

```
« → A b « A SIZE 1 GET → n « { n } 0 CON 'A(1, 1)' EVAL TYPE
IF THEN DUP R→C END → x « n 1 FOR J 'b(J)' EVAL 1 n FOR k
'A(j, k)' EVAL NEXT n →ARRY x DOT – 'A(j, j)' EVAL / HALT DUP x
{ j } ROT PUT 'x' STO -1 STEP n DROPN x » » » »
```

To apply BACK to the above system, start with the upper triangular system [$U|b'$], above, on level 1. Press 5 [COL-] to split off the rightmost column, then press [BACK] to see the last component of the solution vector, -1.4. Each press of [↰] [CONT] will return the next component. When all four components are on the stack, a final [↰] [CONT] shows the solution vector to be [7.1 -4.8 1.4 -1.4], as before.

As this example shows, row interchanges may be needed in order for Gaussian elimination to proceed to its natural conclusion. In so doing we are simply avoiding zero pivots. But to solve many of the linear systems that arise in science and engineering, it is just as important to avoid using pivots that are extremely small, because division by small numbers in floating point arithmetic may induce considerable error. Thus, a common pivoting strategy is to choose as the pivot element the first element on or below the pivot position whose absolute value is

maximum. The need for this so-called ***partial pivoting strategy*** is difficult to illustrate on the calculator because of its use of 12 digit mantissas. Nevertheless, I require that my students adopt partial pivoting by using the RSWP command to reinforce their understanding of this technique. It is routinely used by all professional computer codes.

We rework the above example, this time using partial pivoting throughout. With the augmented matrix on level 1:

```
[[  2   3   2  -1   4 ]
 [ -4  -6   1   2  -1 ]
 [  4   8   7   2  -3 ]
 [  2   4   1  -4   2 ]]
```

partial pivoting requires that we interchange rows 1 and 2. Thus 1, 2 RSWP gives

```
[[ -4  -6   1   2  -1 ]
 [  2   3   2  -1   4 ]
 [  4   8   7   2  -3 ]
 [  2   4   1  -4   2 ]]
```

Then, the commands .5, 1, 2 RCIJ, 1, 1, 3 RCIJ and .5, 1, 4 RCIJ complete the elimination in the first column:

```
[[ -4  -6   1    2   -1  ]
 [  0   0  2.5   0  3.5 ]
 [  0   2   8    4   -4  ]
 [  0   1  1.5  -3  1.5 ]]
```

Now interchange rows 2 and 3 with 2, 3 RSWP and then finish the elimination in the second column with -.5, 2, 4 RCIJ:

```
[[ -4 -6   1    2   -1 ]
 [  0  2   8    4   -4 ]
 [  0  0 2.5    0  3.5 ]
 [  0  0 -2.5  -5  3.5 ]]
```

A final row operation with 1, 3, 4 RCIJ does the job:

```
[[ -4 -6   1    2   -1 ]
 [  0  2   8    4   -4 ]
 [  0  0 2.5    0  3.5 ]
 [  0  0   0   -5    7 ]]
```

Notice how this matrix differs from the one we obtrained without partial pivoting. To complete the solution process, split-off column 5 with 5 COL-, then apply program BACK as before to obtain the solution vector [7.1 -4.8 1.4 -1.4].

To speed up the elimination phase without losing control over the process, we can use the following program ELIM. Program ELIM pivots on a specified entry, the pivot, to produce zeros below that entry. It will handle both real and complex matrices and can be used, more generally, to convert a matrix to row-echelon form. Notice that the program will abort and print the error message "PIVOT ENTRY IS 0" in case the intended pivot is zero. In this event, simply press ↱ UNDO to recapture the matrix before the last application of ELIM.

ELIM (Gaussian elimination)

Inputs: level 3: a matrix
level 2: an integer *k*
level 1: an integer *l*

Effect: pivots on the (*k*, *l*)-entry of the matrix to produce zeros below the pivot.

```
« → A k l « IF 'A(k, l)' EVAL 0 == THEN "PIVOT ENTRY IS 0" ELSE
A SIZE 1 GET → m « k 1 + m FOR i A 'A(i, l)' EVAL NEG 'A(k, l)'
EVAL / k i RCIJ 'A' STO NEXT A 10 RND » END » »
```

EXAMPLE . To use ELIM and BACK with partial pivoting to solve the linear system

$$\begin{array}{rrrrrrrrr} 5x_1 & - & 9x_2 & + & 16x_3 & + & 6x_4 & = & 48 \\ -5x_1 & + & 9x_2 & - & 16x_3 & - & 8x_4 & = & -45 \\ 10x_1 & - & 9x_2 & + & 24x_3 & + & 8x_4 & = & 72 \\ -5x_1 & - & 9x_2 & + & 8x_3 & + & 8x_4 & = & 3 \end{array},$$

begin with the augmented matrix $[A \mid b]$

```
[[ 5 -9  16  6  48 ]
 [-5  9 -16 -8 -45 ]
 [10 -9  24  8  72 ]
 [-5 -9   8  8   3 ]]
```

on level 1. The sequence of commands 1, 3 RSWP ; 1, 1 ELIM ; 2, 4 RSWP ; 2, 2 ELIM ; 3, 3 ELIM returns the equivalent upper triangular system [$U \mid b'$]

```
[[ 10    -9   24    8   72 ]
 [  0 -13.5   20   12   39 ]
 [  0     0 -2.6   -2   -1 ]
 [  0     0    0   -2    3 ]
```

Press 5, COL- to split off the last column. Then, BACK followed by four applications of CONT shows the solution as [3.00000000002 -2.00000000003 1.49999999998 -1.5]. Now clean up the obvious round off error with 10 RND to see the solution [3 -2 1.5 -1.5].

Our discussion of ELIM has been in the context of solving a nonsingular linear system, which has a unique solution. But since ELIM can be applied to convert an arbitrary non-zero matrix to a row-equivalent row echelon form, it can be applied to the augmented matrix $[A \mid b]$ of an *arbitrary* linear system $Ax = b$. If $[U|b']$ is a resulting row echelon form, the nature of the solutions to $Ax = b$ becomes apparent.

Specifically, any variable (or unknown) associated with a pivot is called a *pivot variable* while the other variables, if any, are called *free variables*. If the last non-zero row of $[U|b']$ looks like [0 0 . . . 0 *] where * is a non-zero number, the system has no solution. In any other case there is at least one solution: a unique solution if there are no free variables, but infinitely many solutions when free variables are present. It is standard practice to use back substitution to express each pivot variable in terms of the free variables, and values for the free variables may be arbitrarily (*i.e.*, freely) chosen. Here is an example.

EXAMPLE. To solve the linear system

$$\begin{array}{rcrcrcrcrcl} 4x_1 & + & x_2 & + & 3x_3 & - & 2x_4 & + & x_5 & = & 5 \\ 8x_1 & + & 2x_2 & + & x_3 & - & 5x_4 & & & = & 13 \\ 4x_1 & + & x_2 & + & 8x_3 & - & x_4 & + & 2x_5 & = & 7 \\ 8x_1 & + & 2x_2 & - & 9x_3 & - & 7x_4 & - & 2x_5 & = & 9 \end{array},$$

begin with the augmented matrix $[A \mid b]$

```
[[ 4   1   3   -2   1   5 ]
 [ 8   2   1   -5   0  13 ]
 [ 4   1   8   -1   2   7 ]
 [ 8   2  -9   -7  -2   9 ]]
```

on level 1. If we use partial pivoting, the sequence of commands 1, 2 RSWP; 1, 1 ELIM; 2, 4 RSWP; 2, 3 ELIM; and 3, 5 ELIM returns the following row echelon matrix:

```
[[ 8   2    1   -5   0   13   ]
 [ 0   0  -10   -2   -2  -4   ]
 [ 0   0    0    0   .5  -2.5 ]
 [ 0   0    0    0   0    0   ]]
```

Since the last non-zero row is well behaved,the system is consistent. Since x_1, x_3 and x_5 are pivot variables while x_2 and x_4 are free variables, there are infinitely many solutions. Back substitution (by hand . . . the HP-48 is of no help here) shows all solutions to be given by

$x_5 = -5$

$x_3 = 1.4 - .2x_4$

$x_1 = 1.45 - .25x_2 + .65x_4$

and x_2, x_4 are freely chosen.

Later, we shall give a calculator routine for the variant of Gaussian elimination known as ***Gauss-Jordan reduction***, the effect of which is to do both elimination and back substitution in one routine.

Activity Set 3.1

1. Use partial pivoting and the commands RSWP and RCIJ to convert the augmented matrix $[A|b]$ of each of the following linear systems to a row-equivalent $[U|b']$, where U is upper triangular. Record your row operations. Then use program BACK to solve the system with back substitution.

(a)
$$\begin{aligned} 4x_1 + x_2 + 3x_3 &= 6 \\ 8x_1 - 2x_2 + 4x_3 &= -8 \\ 8x_1 - 6x_2 - 2x_3 &= -36 \end{aligned}$$

(b)
$$\begin{aligned} 3x_1 + 2x_2 - 2x_3 + 2x_4 &= -5 \\ 6x_1 + 2x_2 - x_3 - 2x_4 &= -10 \\ -3x_1 + x_2 + 2.5x_3 &= 8 \\ 6x_1 + 2x_3 + 4x_4 &= 2 \end{aligned}$$

2. Use Gaussian elimination with partial pivoting to solve the following linear systems. Use program ELIM to do the pivoting. Record all your calculator commands.

(a)
$$\begin{aligned} -2x_1 + 3x_2 - x_3 + 2x_4 &= 8 \\ 8x_1 + 4x_2 + 3x_3 + x_4 &= 6 \\ 6x_1 - x_2 - 2x_3 + 3x_4 &= 22 \\ 4x_1 - 6x_2 + 2x_3 + 3x_4 &= 12 \end{aligned}$$

(b)
$$\begin{aligned} -2x_1 - 4x_2 + 5x_3 - 7x_4 &= -8 \\ x_1 + 2x_2 - x_3 + 3x_4 &= 4 \\ x_1 + 4x_2 + 6x_3 + 3x_4 &= 1 \\ 3x_1 + 8x_2 - 2x_3 + 10x_4 &= 6 \end{aligned}$$

3. Repeat Activity 2 for the following linear systems.

(a)
$$\begin{aligned} x_1 + 2x_2 + 3x_3 + 4x_4 &= 5 \\ x_1 + 3x_2 + 4x_3 + 5x_4 &= 5 \\ 3x_1 + 6x_2 + 9x_3 + 2x_4 &= -5 \\ 2x_1 + 4x_2 + 6x_3 + x_4 &= -4 \end{aligned}$$

(b)
$$\begin{aligned} -2x_1 + 3x_2 + 5x_3 - 2x_4 + 3x_5 - 9x_6 &= 6 \\ 2x_1 - 3x_2 + x_3 + 4x_4 - 7x_5 + x_6 &= 2 \\ 6x_1 - 9x_2 + 11x_4 - 19x_5 + 3x_6 &= 0 \\ 4x_1 - 6x_2 + 5x_3 + 9x_4 - 16x_5 - 2x_6 &= 8 \end{aligned}$$

3.2 LU-FACTORIZATIONS

In addition to recognizing Gaussian elimination as an orderly process for converting a square matrix to upper triangular form, it is important to understand it as a factorization process. This understanding is not only interesting from an algebraic viewpoint; it also lies at the heart of many computer codes used to handle linear systems.

When the matrix A in a linear system $Ax = b$ can be brought to upper triangular form U by Gaussian elimination without row interchanges, then $A = LU$ where L is lower triangular with 1's along its main diagonal and the entries below the diagonal are the negatives of the multipliers used in the elimination process. For example, if 3 times row 1 is added to row 2 to produce a zero in the (2, 1)-entry of U, then the (2, 1)-entry of L is -3. When row interchanges are needed to avoid zero pivots, then $A = LU$ is no longer valid; it is replaced by a factorization of the form $PA = LU$ where P is a *permutation matrix* that accounts for the various row interchanges, and the multipliers in the lower triangle of L are rearranged accordingly.

Program L.U, given below, is but a slight modification of ELIM. In addition to performing the basic elimination step L.U stores the negatives of the multipliers below the diagonal in a matrix L which initially is the identity matrix. Program →LP creates the initial L and a matrix P, also the identity matrix. If row interchanges are needed, the proper use of RSWP must be made with both P and U in order to continue, and program L.SWP will effect the necessary interchanges of the multipliers in L. At the end, the calculator shows U on the stack, and L and P as stored variables. As before, complex matrices are allowed. (Note: The • appearing in the name L.U is necessary to distinguish this teaching code from a similar, built-in command LU. More about this command later.)

L.U (Used to construct LU-factorizations)

Inputs: As stored variables: variables *L* and *P*, obtained from program →LP(below), each containing an identity matrix.
level 3: a square matrix
level 2: an integer *k*
level 1: the integer *k*

Effect: Pivots on the (*k*, *k*)-entry to return a row-equivalent matrix with zeros below the pivot; also puts the negatives of the multipliers into column *k* of *L* below the main diagonal. Press [L] to view *L*. Used iteratively to obtain an LU-factorization.

```
« → A k k « IF 'A(k, k)' EVAL 0 == THEN "PIVOT ENTRY IS 0"
ELSE A SIZE 1 GET → m « k 1 + m FOR i A 'A(i, k)' EVAL NEG
'A(k, k)' EVAL / DUP NEG 10 RND 'L(i, k)' STO k i RCIJ 'A' STO
NEXT A 10 RND » END » »
```

→ **LP** (Make *L* and *P*)

Input: level 1: a square matrix *A*

Effect: Creates variables *L* and *P*, each containing an identity matrix the same size as *A*. Used as the initial start-up to construct an LU-factorization.

```
« DUP IDN DUP 'L' STO 'P' STO »
```

L.SWP (Interchange multipliers in *L*)

Input: level 1: a square matrix *L*
level 2: an integer $i > 1$
level 3: an integer $j > i$

Effect: Interchanges the parts of rows *i* and *j* that lie to the left of the (*i*, *i*)-entry in *L*; used to update *L* by interchanging multipliers.

```
« → A i j « IF i 1 ≤ j i ≤ OR THEN "INVALID ROW INPUT" ELSE A
SIZE 2 GET → n « A 1 i 1 – FOR k 'A(i, k)' EVAL { j k } SWAP
PUT NEXT 1 i 1 – FOR m 'A(j, m)' EVAL { i m } SWAP PUT NEXT »
» »
```

EXAMPLE . Use partial pivoting to construct an LU-factorization of

$$A = \begin{bmatrix} 2 & 3 & -1 & 2 \\ -4 & -6 & 2 & 1 \\ 2 & 4 & -4 & 1 \\ 4 & 8 & 2 & 7 \end{bmatrix}.$$

Step 1: Enter A onto level 1, and press [→LP] to create appropriate starting matrices L and P. Interchange rows 1 and 2 in A with 1, 2 [RSWP], recall P to the stack and make the same row interchange, then store the updated result in P with [↰] [P]. Now press 1, 1 [L.U] to see

$$\begin{bmatrix} 4 & -6 & 2 & 1 \\ 0 & 0 & 0 & 2.5 \\ 0 & 1 & -3 & 1.5 \\ 0 & 2 & 4 & 8 \end{bmatrix}.$$

Step 2: Since the (2, 2)-entry of this last matrix is 0, we must interchange row 2 with row 4. Thus press 2, 4 [RSWP] to effect the interchange, then bring P to level 1, make the same row interchange with [RSWP] and store the result in P. Now bring L to level 1 with [L], interchange multipliers with 2, 4 [L.SWP] and store the result in L with [↰] [L].

Step 3: Now execute 2, 2 [L.U] to see $\begin{bmatrix} -4 & -6 & 2 & 1 \\ 0 & 2 & 4 & 8 \\ 0 & 0 & -5 & -2.5 \\ 0 & 0 & 0 & 2.5 \end{bmatrix}$. Store this as U.

Step 4: Get L = $\begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -.5 & .5 & 1 & 0 \\ -.5 & 0 & 0 & 1 \end{bmatrix}$ with [L], then do [U] [×] to see

$$LU = \begin{bmatrix} -4 & -6 & 2 & 1 \\ 4 & 8 & 2 & 7 \\ 2 & 4 & -4 & 1 \\ 2 & 3 & -1 & 2 \end{bmatrix}.$$

Since P is a permutation matrix, we know that $P^{-1} = P^T$. Thus $P^{-1}LU = P^TLU = A$. Recall P to level 1 and get P^T, SWAP levels with LU and then use [×] to see $P^TLU = A$.

Why are LU-factorizations important? Here are several reasons:

(i) In the case that $A = LU$, all the information regarding Gaussian eliminiation on A is stored in the factors L and U. Matrix L maintains a record of the multipliers used in the eliminiation process and U records the results of the elimination. Thus, L and U may be viewed as storehouses of information about A that can be exploited later in a variety of situations. With $PA = LU$, P records the row interchanges.

(ii) Once we have $A = LU$ we can solve $Ax = b$ for different b's by first using forward substitution to solve $Ly = b$ for y, then by using back substitution to solve $Ux = y$ for x. (In the case of $PA = LU$, we solve $Ly = Pb$ in the first step.) Indeed, this is often the preferred method built into computer codes for solving linear systems. It is a matter of economy. Assume that A is n×n and that A^{-1} as well as the factors L and U are available. Using A^{-1} to obtain $x = A^{-1}b$ requires n^2 multiplications. Solving $Ly = b$ for y by forward substitution and then solving $Ux = y$ for x by back substitution also

requires n^2 multiplications. But the difference is seen in comparing the number of multiplications required to obtain A^{-1} to the number of multiplications required to obtain the factors L and U: n^3 verses $\frac{n^3}{3}$. For large n, the savings in using L and U is substantial.

(iii) The interpretation of Gaussian elimination as a matrix factorization $PA = LU$ sets the stage for the more sophisticated matrix factorizations that are encountered in a study of numerical linear algebra; for example, the QR, Schur, and SVD factorizations.

To apply forward substitution to $Ly = Pb$ on the calculator, use the following program FWD.

FWD (Forward substitution)

Inputs: level 2: an $n \times n$ invertible lower triangular matrix L
level 1: an n-vector b

Effect: Solves the linear system $Lx = b$ by forward substitution. Solves for x_1 and halts until you press [↰] [CONT], then solves for x_2 and halts, etc. After $x_1, x_2, \ldots, x_n$ are on the stack, a final [CONT] returns $x = [x_1, x_2, \ldots, x_n]$.

```
« → A b « A SIZE 1 GET → n « { n } 0 CON 'A(1, 1)' EVAL TYPE
IF THEN DUP R→C END →  y « 1 n FOR j 'b(j)' EVAL 1 n FOR k
'A(j, k)' EVAL NEXT n →ARRY y DOT – 'A(j, j)' EVAL / HALT DUP y
{ j } ROT PUT 'y' STO NEXT n DROPN y » » » »
```

EXAMPLE. To solve

$$\begin{aligned} 2x_1 + 3x_2 - x_3 + 2x_4 &= 1 \\ -4x_1 - 6x_2 + 2x_3 + x_4 &= 2 \\ 2x_1 + 4x_2 - 4x_3 + x_4 &= 3 \\ 4x_1 + 8x_2 + 2x_3 + 7x_4 &= 4 \end{aligned}$$

by using an LU-factorization, we first obtain a $PA = LU$ factorization of the coefficient matrix

$$A = \begin{bmatrix} 2 & 3 & -1 & 2 \\ -4 & -6 & 2 & 1 \\ 2 & 4 & -4 & 1 \\ 4 & 8 & 2 & 7 \end{bmatrix}.$$

Since A is the matrix of our last example, we shall use the P, L and U obtained there:

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -.5 & .5 & 1 & 0 \\ -.5 & 0 & 0 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} -4 & -6 & 2 & 1 \\ 0 & 2 & 4 & 8 \\ 0 & 0 & -5 & -2.5 \\ 0 & 0 & 0 & 2.5 \end{bmatrix}.$$

Let b = [1 2 3 4]. To solve $Ly = Pb$ for y by forward substitution, calculate Pb = [2 4 3 1]. Then, with L on level 2 and Pb on level 1, FWD and four applications of CONT show y to be [2 6 1 2]. Then with U on level 2 and [2 6 1 2] on level 1, BACK and four applications of CONT show the solution x of $Ax = b$ to be [-2.1 1 -.6 .8].

The HP-48G and 48GX calculators include a command LU that produces an LU-factorization $PA = LU$ which differs from the one we have just described in that U has 1's along the main diagonal and the pivots appear on the main diagonal of L. To see how this can occur, imagine $PA = LU$ where L is unit lower triangular and U

has the non-zero pivots on the main diagonal. If D is the diagonal matrix whose main diagonal contains the pivots $u_{11}, u_{22}, ..., u_{nn}$ from U, then $PA = (LD)\ (D^{-1}U)$. The effect of matrix D is to multiply each column j of L by u_{jj} so that LD is lower triangular with the pivots on its main diagonal. The effect of D^{-1} is to multiply each row j of U by u_{jj}^{-1}, so that $D^{-1}U$ is unit upper triangular. The actual method used to obtain this factorization is known as the *Crout* algorithm. It employs partial pivoting throughout and is particularly well-suited to calculator use. For example, with our previous matrix

$$A = \begin{bmatrix} 2 & 3 & -1 & 2 \\ -4 & -6 & 2 & 1 \\ 2 & 4 & -4 & 1 \\ 4 & 8 & 2 & 7 \end{bmatrix}$$

on level 1 of the stack, pressing [LU] on the MTH MATR FACTR menu will return

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

to level 1,

$$U = \begin{bmatrix} 1 & 1.5 & -.5 & -.25 \\ 0 & 1 & 2 & 4 \\ 0 & 0 & 1 & .5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

to level 2, and

$$L = \begin{bmatrix} -4 & 0 & 0 & 0 \\ 4 & 2 & 0 & 0 \\ 2 & 1 & -5 & 0 \\ 2 & 0 & 0 & 2.5 \end{bmatrix}$$

to level 3.

You will recognize this U as a rescaled version of the one we obtained earlier: row i of our earlier U has been rescaled by multiplying by u_{ii}^{-1}. Likewise, L is just a rescaled version of the one we obtained earlier: column j of our earlier L has been rescaled by multiplying by u_{jj}. Rescaling our earlier U and L will not affect the final solution.

Unlike the ELIM and L.U programs, which round-off intermediate computations to 10-digit precision to clean up round off errors, the built-in matrix routines on the HP-48G and 48GX, such as the LU routine, perform all intermediate computations to 15-digit precision and then pack the computed results to the displayed 12-digits.

Finally, although our discussion has concentrated on developing an understanding of Gaussian elimination and its interpretation as an LU-factorization, you should note that the HP-48G and 48GX units enable you to solve any nonsingular linear system by applying an LU-factorization with a single keystroke. With an invertible matrix A of order n on stack level 1 and n-vector b on level 2, the command /, executed from the keyboard by pressing the [÷] key, will solve the linear system $Ax = b$ by the method cited earlier: use partial pivoting to obtain an LU-factorization $PA = LU$, then solve $Ly = Pb$ for y by forward substitution, then solve $Ux = y$ for x by back substitution. Try it with the last example. More generally, if b is replaced by an $n \times p$ matrix B, the same procedure will solve the matrix equation $AX = B$; column j in the computed X is the solution to $AX_j = B_j$, where B_j is the corresponding column in matrix B.

Activity Set 3.2

1. Use partial pivoting to construct $PA = LU$ factorizations of each of the following matrices. Then use P, L and U as in our last example to solve the linear system $Ax = b$; in each case, record the solution y to $Ly = Pb$.

(a) A =
```
[[ -2   4    5  ]
 [ -1  3.5  2.5 ]
 [  2  -1   3  ]]
```
b = [10 3.5 11]

(b) A =
```
[[ -2   6   5 ]
 [ -3   1  -.5 ]
 [ -6  -6  -3 ]]
```
b = [26 15 18]

(c) A =
```
[[  0  -2   7  -3 ]
 [  3   0   3   5 ]
 [ -3   6  -3   4 ]
 [ -9   6  -3   6 ]]
```
b = [44 -24 -24 -24]

(d) A =
```
[[ -2   0  -3   0  -5 ]
 [ -6  -3 -12  -9   3 ]
 [  6   6   6  12   6 ]
 [  0   3   3  -3   6 ]
 [ -2  -1  -4  -1   1 ]]
```
b = [-7 36 -6 27 10]

2. For each of the following matrices, find the Crout $PA = LU$ factorization by using the LU command. Then use P, L and U to solve the system $Ax = b$. Check your results using the / command.

(a) A =
```
[[ -1  -1   2 ]
 [ -4   2  -1 ]
 [ -9   6   0 ]]
```
, b = [-7 11 21]

(b) A =
```
[[  1  -5   2  -1 ]
 [  2  -7   4   1 ]
 [ -7   9   1   3 ]
 [  5  -8  -9   8 ]]
```
, b = [-8 -13 53 -76]

(c) A =
```
[[ -7  -7  -7   2  -7 ]
 [ -1   0  -5  -8   7 ]
 [  7  -1  -9   6   7 ]
 [  5   4   9   0  -9 ]
 [  9  -6  -5  -9   8 ]]
```
, b = [-16 45 -80 7 17]

3.3 GAUSS-JORDAN REDUCTION

Although Gaussian elimination with back substitution is more efficient than Gauss-Jordan reduction for dealing with linear systems in general, and is certainly the preferred method in professional computer libraries, students have traditionally used Gauss-Jordan reduction for the small-scale problems employed to learn the basic concepts. This minimizes the rational number arithmetic involved when Gaussian elimination is performed by hand on matrices with integer entries.

Gauss-Jordan reduction differs from Gaussian elimination in two ways:

(i) all pivots are converted to 1.

(ii) the basic pivot process is used to produce zero's both below *and above* the pivot element.

When applied to a non-zero matrix A, Gauss-Jordan reduction produces what is popularly called the **reduced row echelon form (RREF)** of A:

(a) any zero rows lie at the bottom;

(b) the first non-zero entry in any non-zero row (the pivot) is a 1, and lies to the right of the pivot in any preceding row;

(c) each pivot is the only non-zero entry in its column.

The reduced row echelon form of A is important because it represents the ultimate we can get from A by applying elementary row operations. As such, it is uniquely associated with A; that is, each non-zero matrix A has one and only one RREF.

When Gauss-Jordan reduction is applied to the augmented matrix $[\, A|\, b\,]$ of an arbitrary linear system $Ax = b$ we obtain an equivalent linear system $Ux = b'$ whose augmented matrix $[\, U|\, b'\,]$ is the RREF of $[\, A|\, b\,]$ and whose solutions are practically

obvious. Although impractical for extremely large linear systems that arise in practice, Gauss-Jordan reduction is in popular use as a device to solve small systems, and to further advance the theory of linear algebra. And it is easy to devise a calculator program to step through the reduction process.

The following program, PIVOT, pivots on a specified entry to convert the pivot to 1 and to produce zeros above and below the pivot. It can be used in conjunction with the command RSWP to produce the RREF matrix. The program is written to accommodate both real and complex matrices.

PIVOT (Gauss-Jordan Pivot)

Inputs: level 3: a matrix
level 2: an integer *k*
level 1: an integer *l*

Effect: converts the (*k*, *l*)-entry to 1 and then pivots on that entry to produce zeros above and below the pivot.

```
« → A k l « IF 'A(k, l)' EVAL 0 == THEN "PIVOT ENTRY IS 0" ELSE
A SIZE 1 GET → m « m IDN 'A(1, 1)' EVAL TYPE IF THEN DUP 0
CON R→C END 1 m FOR i 'A(i, l)' EVAL {i k} SWAP PUT NEXT INV
A * » 8 RND END » »
```

EXAMPLE. Solve the linear system

$$\begin{aligned} 2x_1 - 3x_2 + x_3 - 3x_4 + 2x_5 &= 6 \\ -2x_1 + 3x_2 - x_3 + 4x_4 + x_5 &= -5 \\ 6x_1 - 9x_2 + 7x_3 - 7x_4 + 5x_5 &= 20 \\ -2x_1 + 3x_2 + 3x_3 + 3x_4 - 9x_5 &= -6 \end{aligned}$$

by applying Gauss-Jordan reduction with partial pivoting to the augmented matrix. The sequence of commands 1, 3 [RSWP]; 1,1 [PIVOT]; 2,4 [RSWP]; 2,3 [PIVOT]; 3,4 [RSWP]; 3,4 [PIVOT] returns the following matrix as the reduced row-echelon form:

```
[[  1  -1.5   0   0   6.375  4.5 ]
 [  0    0    1   0  -1.75   0  ]
 [  0    0    0   1   3      1  ]
 [  0    0    0   0   0      0  ]]
```

Thus x_2 and x_5 are free variables and all solutions are given by $x = [\ 4.5 + 1.5x_2 - 6.375x_5, x_2, 1.75x_5, 1-3x_5, x_5]$.

Although the reduced row-echelon form of a matrix is not in use at the professional level to solve linear systems, it can be an effective pedagogical tool to help understand the role of pivot variables versus free variables and such vector space concepts as linear combinations, independence, bases and eigenspaces. The HP-48G and 48GX calculators provide access to the reduced row-echelon form by means of the RREF command, located on the MTH MATR FACTR menu. With a matrix on level 1 of the stack, simply press the [RREF] key (or type and enter the command RREF) to obtain the reduced row-echelon form. In order to obtain correct results, flag -54 must be clear (the default state). After some initial experiences in producing the reduced row-echelon form with program PIVOT and the command RSWP, I allow my students to call upon the RREF command thereafter. The underlying code uses partial pivoting throughout.

Activity Set 3.3

1. Perform Gauss-Jordan reduction with PIVOT to solve each of the linear systems in ACTIVITY SET 3.1. Verify your results by applying the RREF command.

4 VECTOR SPACES

Of the many concepts from linear algebra that permeate the different fields of mathematics, perhaps none is as powerful as that of a vector space. Indeed, for some, to study linear algebra is to study vector spaces and their associated notions.

Informally, a vector space V is simply a set of objects together with a way of combining any two of them under an operation called addition, and a way of multiplying any one of them by a scalar (a number). Of course, we require that these two operations obey a few basic laws. The prototype for all vector spaces is the familiar set R^n of all n-tuples of real numbers together with the usual component addition and scalar multiplication. The basic laws are the four usual properties for addition of n-tuples and the four usual properties for scalar multiplication of n-tuples. Vector spaces provide an umbrella environment that serves to both clarify and to unify a number of seemingly unrelated concepts and topics from a variety of fields.

When cast in purely abstract terms, such fundamental vector space notions as linear combinations and spanning sets, independence and dependence, bases and dimension, and change of basis appear to be somewhat removed from a study of linear systems. But exactly the opposite is true: in the historical development of linear algebra it was from a study of linear systems and their associated matrices that these vector space concepts emerged.

4.1 LINEAR COMBINATIONS AND SPANNING SETS

Recall that by a *linear combination* of vectors $v_1, v_2, \ldots, v_k$ in a vector space V (you may regard V as being R^n if it helps) we mean any vector of the form $x_1v_1 + x_2v_2 + \ldots + x_kv_k$ where the x_j's are scalars . The set of all possible linear combinations of $v_1, v_2, \ldots, v_k$ is a subspace of V, often denoted by Span $[v_1, v_2, \ldots, v_k]$, and the vectors v_i are said to *span* this subspace. To verify that Span $[v_1, v_2, \ldots, v_k]$ is a subspace of V we need only add two linear combinations of the vectors $v_1, v_2, \ldots, v_k$ to see that we obtain another one, and then multiply an arbitrary linear combination of the v_i's by a scalar to obtain still another such combination. To determine whether a given vector u lies in the subspace Span $[v_1, v_2, \ldots, v_k]$ we must determine whether u can be written as $u = x_1v_1 + x_2v_2 + \ldots + x_kv_k$ for suitable scalars x_i.

The connection to linear systems comes from the fact that for an $m \times n$ matrix A the matrix equation $Ax = b$ expresses vector b as a linear combination of the column vectors of A:

$$b = x_1A_1 + x_2A_2 + \ldots + x_nA_n$$

where A_j is column j of matrix A and x is the column vector = $[x_1, x_2, \ldots, x_n]$. The column vectors $A_1, A_2, \ldots, A_n$ of matrix A span the *column space CS(A)* of A. Vector b is a linear combination of the columns of A iff the linear system $Ax = b$ has a solution; and any solution to $Ax = b$ will serve to express b as a linear combination of these columns.

EXAMPLE 1. To investigate whether the vector u = [3 10 -2 18] is a linear combination of vectors v_1 = [1 -2 3 0], v_2 = [-1 4 2 3] and v_3 = [2 0 -1 4], we set up the linear system $Ax = u$ where A has v_1, v_2 and v_3 as its columns. Program ELIM can be used to determine whether a solution exists, but an even better choice would be

to use PIVOT because it will enable us to obtain all solutions. Applying PIVOT to the augmented matrix [$A|u$], we see that

$$\begin{bmatrix} 1 & -1 & 2 & 3 \\ -2 & 4 & 0 & 10 \\ 3 & 2 & -1 & -2 \\ 0 & 3 & 4 & 18 \end{bmatrix} \quad \text{☞} \quad \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

from which we can write $u = -v_1 + 2v_2 + 3v_3$.

EXAMPLE 2. Which of the vectors u_1 = [0 3 -6 3], u_2 = [-4 7 -4 0] and u_3 = [6 -4.5 2 2] are in the span of vectors v_1 = [4 -1 0 2] and v_2 = [0 3 -2 1] ? To answer this we investigate the three linear systems $Ax = u_i$ (i = 1, 2, 3), where A has vectors v_1 and v_2 as its two columns. Applying PIVOT to the triple augmented matrix [$A|u_1\ u_2\ u_3$] to reduce A to its RREF we find that

$$\begin{bmatrix} 4 & 0 & 0 & -4 & 6 \\ -1 & 3 & 3 & 7 & -4.5 \\ 0 & -2 & -6 & -4 & 2 \\ 2 & 1 & 3 & 0 & 2 \end{bmatrix} \quad \text{☞} \quad \begin{bmatrix} 1 & 0 & 0 & -1 & 1.5 \\ 0 & 1 & 0 & 2 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Column 3 tells us that u_1 is not a linear combination of v_1 and v_2, so not in Span [v_1, v_2]; columns 4 and 5 show the exact opposite: $u_2 = -v_1 + 2v_2$, $u_3 = 1.5v_1 - v_2$.

Activity Set 4.1

1. Which of the following vectors u_1 = [1 -5 4], u_2 = [2 11 23] and u_3 = [16 -7 3] are linear combinations of v_1 = [8 9 7] and v_2 = [3 -1 -8]? For any u_j that is a linear combination of v_1 and v_2, show such a linear combination.

2. Which of the following vectors u_1 = [-3 16 3 -15], u_2 = [-9 0 4 -3] and u_3 = [0 -5 27 14] lie in the subspace of R^4 spanned by v_1 = [-2 7 6 -5], v_2 = [5 -6 -6 4] and v_3 = [4 -8 3 9]. For any vector u_j that lies in this subspace, show how it gets there.

3. Which of the following polynomials $p(x) = -5 + 7x - 5x^2 - 13x^3$ and $q(x) = -4 - 16x^2 - 19x^3$ are linear combinations of $r(x) = 4 - 3x - 2x^2 + 3x^3$, $s(x) = 9 + 6x - 9x^2 - 5x^3$ and $t(x) = 6 + 5x + 2x^3$? For any that are, show how.

4.2 DEPENDENCE AND INDEPENDENCE

When a vector u is a linear combination of some other vectors $v_1, v_2, \ldots, v_k$ we say that u *depends* linearly upon the v_j's and that the entire set of vectors is a "linearly dependent" set. More precisely, a set of vectors $\{ v_1, v_2, \ldots, v_k \}$ ($k > 1$) is called *(linearly) dependent* if one of these vectors is a linear combination of the others. To the contrary, the set of vectors $\{ v_1, v_2, \ldots, v_k \}$ ($k > 1$) is called *(linearly) independent* if no one of these vectors is a linear combination of the others. In case we have a single vector v_1 we agree that $\{ v_1 \}$ is linearly dependent if $v_1 = 0_v$, and linearly independent if $v_1 \neq 0_v$.

To relate these notions to linear systems, recall that they may be reformulated, equivalently, as follows:

(a) the set $\{ v_1, v_2, \ldots, v_k \}$ is dependent iff there are scalars $x_1, x_2, \ldots, x_k$, *not all of which are 0*, such that $x_1v_1 + x_2v_2 + \ldots + x_kv_k = 0_v$; thus

(b) the set $\{ v_1, v_2, \ldots, v_k \}$ is independent iff whenever we have $x_1v_1 + x_2v_2 + \ldots + x_kv_k = 0_v$ then necessarily *all* $x_i = 0$.

These are the standard notions of dependence and independence found in elementary texts, but you should not lose sight of the fact that they are the mathematically equivalent reformulations of the more intuitive ideas given above.

In terms of linear systems: if matrix A has the vectors $v_1, v_2, \ldots, v_k$ as its columns then

(a) the set $\{ v_1, v_2, \ldots, v_k \}$ is dependent iff $Ax = 0$ has a non-zero solution; and

(b) the set $\{ v_1, v_2, \ldots, v_k \}$ is independent iff $Ax = 0$ has only the zero solution.

To put this to use, recall some of the conditions under which $Ax = 0$ has non-zero solutions: $Ax = 0$ has non-zero solutions iff

(i) A has fewer rows than columns,

(ii) A is row-equivalent to a row echelon matrix having fewer non-zero rows than columns; or

(iii) when A is square, A is singular.

In each of these cases, Gaussian elimination will show the existence of free variables, hence non-zero solutions.

EXAMPLE 3. Investigate the dependence/independence of the vectors $v_1 = [\,-1\ \ 2\ \ -1\ \ 3\,]$, $v_2 = [\,2\ \ -1\ \ 4\ \ 1\,]$ and $v_3 = [\,-4\ \ 5\ \ -6\ \ 5\,]$ in R^4. If the set $\{ v_1, v_2, v_3 \}$

is dependent, write an equation that expresses the dependency. We set up matrix A as

$$A = \begin{bmatrix} -1 & 2 & -4 \\ 2 & -1 & 5 \\ -1 & 4 & -6 \\ 3 & 1 & 5 \end{bmatrix}$$

and find its RREF to be

$$E = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Thus by (ii), we see that $Ax = 0$ has non-zero solutions and so $\{ v_1, v_2, v_3 \}$ is dependent. In fact, all solutions are given by $x = [-2\alpha, \alpha, \alpha]$, where α is freely chosen. Choosing $\alpha = 1$ we obtain the particular solution $x = [-2 \ \ 1 \ \ 1]$ which says that $-2v_1 + v_2 + v_3 = 0$, an equation that expresses the general dependency among these vectors.

One final observation: it is almost obvious that the non-zero rows of any row echelon matrix are independent; also the columns of any row echelon matrix that contain the pivots are independent. For example, look at the non-zero rows, and the pivot columns (columns 1, 2, and 4), of the row echelon matrix

$$\begin{bmatrix} 2 & 0 & 3 & 4 & 5 \\ 0 & 6 & 7 & 8 & 9 \\ 0 & 0 & 0 & 10 & 11 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Generally, we regard independence as being a desirable property and dependence as being undesirable. For when a set of vectors (more than one vector) is dependent, at least one of them can be written as a linear combination of the others, say $v_1 = x_2 v_2 + \ldots + x_k v_k$. Consequently, any linear combination of the vectors in

$\{ v_1, v_2, ..., v_k \}$ can be replaced by a linear combination of the vectors in the smaller set $\{ v_2, ..., v_k \}$. In terms of spanning sets, this observation is simply that $W = \text{Span} [v_1, v_2, \ldots, v_k] = \text{Span} [v_2, \ldots, v_k]$ so we have effectively deleted the vector v_1; it was redundant. And if the remaining set $\{ v_2, \ldots, v_k \}$ is dependent then we can delete still another of these vectors (say v_2, for example), so that

$$W = \text{Span} [v_1, v_2, v_3, \ldots, v_k] = \text{Span} [v_2, v_3, \ldots, v_k] = \text{Span} [v_3, \ldots, v_k] .$$

In this way we can continue deleting "redundant" vectors until we arrive at an independent spanning set for W.

EXAMPLE 4. To continue with the vectors v_1, v_2 and v_3 from **EXAMPLE 3**, recall that we found in Example 3 that the set $\{ v_1, v_2, v_3 \}$ was dependent; indeed the equation $-2v_1 + v_2 + v_3 = 0$ displays the precise way in which any one of these three vectors can be written in terms of the other two. For instance, solving for v_3 we have $v_3 = 2v_1 - v_2$. The subspace W spanned by the set $\{ v_1, v_2, v_3 \}$ is thus also spanned by the set $\{ v_1, v_2 \}$; we have deleted the redundant vector v_3. Can we delete even more? In other words, is the set $\{ v_1, v_2 \}$ dependent? The answer is no, which you can quickly verify by glancing back at the first two columns of matrix A and its reduced row echelon form (the first two columns of matrix E). Therefore $\{ v_1, v_2 \}$ is an independent spanning set for W.

Finally, notice how the columns in the reduced row echelon form convey all of the above information: the leading 1's in the first two columns of E tell us that column vectors 1 and 2 of matrix A are independent. The third column of E specifies how column 3 of A depends upon the first two columns of A, $v_3 = 2v_1 - v_2$.

Activity Set 4.2

1. Investigate the independence/dependence of each of the following sets of vectors. If dependent, write an equation that expresses the exact nature of the dependency.

   (a) The rows of

$$A = \begin{bmatrix} 3 & 6 & 10 & 7 \\ 1 & 3 & -1 & 0 \\ 1 & 0 & 4 & 2 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

   (b) The columns of the matrix A in (a).

   (c) The rows of

$$C = \begin{bmatrix} 72 & 42 & 58 & 83 & 55 \\ 60 & 24 & 90 & 20 & -34 \\ 29 & 27 & 44 & 37 & -5 \\ 12 & 45 & 82 & 4 & -111 \end{bmatrix}.$$

   (d) The columns of the matrix C in (c).

   (e) The columns of

$$E = \begin{bmatrix} 1 & 10 & 0 & -8 & -2 \\ -3 & -8 & 1 & 2 & 8 \\ 5 & 9 & -4 & 1 & -18 \\ 7 & 17 & -2 & 3 & -18 \\ 2 & -2 & 7 & 6 & 10 \end{bmatrix}.$$

2. Each of the following sets of vectors spans a subspace W of R^4. Show that the set is dependent, specify the exact nature of the dependency, and then discard vectors one at a time until you get an independent spanning set for W.

(a) $u_1 = [\,3 \quad -1 \quad -7 \quad 6\,]$, $u_2 = [\,3 \quad 6 \quad -4 \quad 7\,]$

$u_3 = [\,-2 \quad -1 \quad 3 \quad -3\,]$, $u_4 = [\,1 \quad 4 \quad 0 \quad 2\,]$

(b) $v_1 = [\,3 \quad 3 \quad -2 \quad 1\,]$, $v_2 = [\,-1 \quad 6 \quad -1 \quad 4\,]$

$v_3 = [\,-7 \quad -4 \quad 3 \quad 0\,]$, $v_4 = [\,6 \quad 7 \quad -3 \quad 2\,]$

(c) $w_1 = [\,-3 \quad 0 \quad 5 \quad -7\,]$, $w_2 = [\,2 \quad 8 \quad -1 \quad 6\,]$

$w_3 = [\,1 \quad 16 \quad 3 \quad 5\,]$, $w_4 = [\,-2 \quad -8 \quad 1 \quad -6\,]$

4.3 BASES AND DIMENSION

Spanning sets that are independent are especially desirable because no one of the spanning vectors depends linearly upon the others. This is what we mean by a basis. More formally, by a *basis* for a subspace W of a vector space V (again, you may imagine V to be R^n if it helps) we mean a collection of vectors $w_1, w_2, \ldots, w_k$ from W that

(i) is independent, and

(ii) spans W.

When you choose a basis for W, you have chosen a "well-behaved" set of vectors to use in describing or understanding W. Basis vectors are well-behaved in the sense that they are independent vectors, hence no dependency upon one another. They can be used to describe W because each vector in W is a linear combination of them.

Together, the two conditions (i), (ii) tell us that each vector in W can be written as a linear combination of the basis vectors in only one way. Here is why. Condition (ii) guarantees that each vector w in W can be written in at least one way, say $w = x_1 w_1 + x_2 w_2 + \ldots + x_k w_k$. Suppose, however, that there is another way to do this, say $w = y_1 w_1 + y_2 w_2 + \ldots + y_k w_k$. Then, easily

$$0_v = w - w = (x_1 - y_1)w_1 + (x_2 - y_2)w_2 + \ldots + (x_k - y_k)w .$$

This last equation is a linear combination of the w_j's equal to the zero vector. Therefore, because of the independence of $\{ w_1, w_2, \ldots w_k \}$, all coefficients $(x_j - y_j)$ in this combination must be zero: $(x_j - y_j) = 0$, so $x_j = y_j$ for $j = 1, \ldots, k$ and we really have only one way of expressing w.

It is important to know that whenever we have a basis for a vector space V, say $B = \{ v_1, v_2, \ldots v_n \}$, then any set in V having more than n vectors is dependent. Your textbook in linear algebra most certainly includes an argument to convince you of this fact. The impact is, of course, clear: *all bases for V contain the same total number of vectors*. This is the *dimension* of V, *dim* V. Recall that dim $R^n = n$, and *dim* $P_n[x] = n+1$ ($P_n[x]$ consists of all polynomials having degree $\leq n$).

We are interested in the four fundamental subspaces associated with a real $m \times n$ matrix A:

- the *row space* $RS(A)$: the subspace of R^n spanned by the row vectors of A;
- the *column space* $CS(A)$: the subspace of R^m spanned by the column vectors of A;
- the *null space* of A, $NS(A)$: the subspace of R^n consisting of all solutions x to the homogeneous linear system $Ax = 0$;
- the *left null space* of A, $NS(A^T)$: the subspace of R^m consisting of all solutions x to the homogeneous linear system $A^T x = 0$.

You should recall how we can produce bases for each of these subspaces. For the first three, begin by converting A to any row echelon form U by row operations; then

- A and U have the same row space, $RS(A) = RS(U)$, and the non-zero rows of U form a basis for $RS(A)$;
- although A and U do not have the same column space, $CS(A) \neq CS(U)$, the columns in A corresponding to the pivot columns in U form a basis for $CS(A)$;
- if $NS(A) = 0_v$, we have no basis. Otherwise, there will be free variables and all solutions to $Ax = 0$ can be obtained by choosing arbitrary values for the free variables. Construct special solutions as follows: assign, in turn, the value 1 to each free variable and the value 0 to the other free variables. These special solutions form a basis for $NS(A)$.
- For the left null space of A, $NS(A^T)$, simply convert A^T to row echelon form and proceed as in the case of the null space $NS(A)$.

Programs ELIM or PIVOT, or the command RREF may be used to construct bases for $RS(A)$ and $CS(A)$; but PIVOT or RREF should always be used to construct bases for the two null spaces.

EXAMPLE 5. Find bases for the row space, column space, null space, and left null space of the following matrix:

$$A = \begin{matrix}[[1 & 2 & 3 & 4 & 5] \\ [1 & 3 & 4 & 5 & 5] \\ [3 & 6 & 9 & 2 & -5] \\ [2 & 4 & 6 & 1 & -4]]\end{matrix}.$$

The reduced row echelon form of matrix A is

$$U = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Thus, the first three rows of U form a basis for the row space $RS(A)$ and columns 1, 2, and 4 *of* A form a basis for the column space $CS(A)$. Clearly x_3 and x_5 are free variables, and all solutions to $Ax = 0$ are given by

$$x = \begin{bmatrix} -x_3 - x_5 \\ -x_3 + 2x_5 \\ x_3 \\ -2x_5 \\ x_5 \end{bmatrix} = x_3 \begin{bmatrix} -1 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + x_5 \begin{bmatrix} -1 \\ 2 \\ 0 \\ -2 \\ 1 \end{bmatrix}.$$

The two vectors on the right-hand side form a basis for the null space $NS(A)$. They were obtained from the general solution by factoring out x_3 and x_5; but notice that they are the special solutions described earlier that we can obtain by setting $x_3 = 1$ and $x_5 = 0$, then $x_3 = 0$ and $x_5 = 1$.

For the left null space $NS(A^T)$ of A we begin by obtaining the reduced row echelon form of A^T:

$$\begin{bmatrix} 1 & 0 & 0 & -.1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & .7 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

This time, only x_4 is a free variable and all solutions to $A^T x = 0$ are given by

$$x = \begin{bmatrix} .1x_4 \\ 0 \\ -.7x_4 \\ x_4 \end{bmatrix} = x_4 \begin{bmatrix} .1 \\ 0 \\ -.7 \\ 1 \end{bmatrix}.$$

The single vector on the right side of the last equation forms a basis for the left null space $NS(A)$.

EXAMPLE 6. The basis for the row space of A that we obtained in EXAMPLE 5 consisted of the non-zero rows of U. Since the rows of A span $RS(A)$, we know they can be cut down to obtain a basis for $RS(A)$. How can we obtain a basis from among the *original* rows of matrix A?

From the reduced row echelon form of A^T we can choose a basis for $CS(A^T)$ consisting of columns 1, 2 and 3 of A^T. Then, since $CS(A^T) = RS(A)$, we will have a basis for $RS(A)$ that is chosen from among the original rows of A.

A few final comments are in order. When the $m \times n$ matrix A is converted to row echelon form U, it is clear that the number of non-zero rows in U is precisely the number of non-zero pivots in U. Thus

$$dim\ RS(A) = \text{the number of non-zero pivots.}$$

Similarly, it is clear that the number of pivot columns in U is the number of non-zero pivots, so that also

$$dim\ CS(A) = \text{the number of non-zero pivots}$$

This common number, $dim\ RS(A) = dim\ CS(A)$ = number of non-zero pivots is known as the *rank* of A:

$$\text{rank } A = dim\ RS(A) = dim\ CS(A)$$

Of course, an identical result holds for matrix A^T. But since the row space of A coincides with the column space of A^T, we have the result

$$\text{rank } A = \text{rank } A^T.$$

If not zero, the dimension of the null space of A is the number of "special vectors" that can be constructed from the free variables in the reduction of A to U. Thus

$$\begin{aligned} dim\ NS(\text{A}) &= \text{the number of free variables} \\ &= n - (\text{the number of non-zero pivots}) \\ &= \text{n} - \text{rank } A. \end{aligned}$$

We therefore have the fundamental result that

$$\text{rank } A + dim\ NS(A) = \text{number of columns of } A.$$

Since this results applies when we replace A with A^T, we see that

$$\text{rank } A + dim\ NS(A^T) = \text{number of rows of } A.$$

Activity Set 4.3

1. Find bases for each of the four fundamental subspaces of the following matrices.

(a) $A =$
```
[[ 3   1   1   0 ]
 [ 6   3   0   1 ]
 [ 10  -1  4   1 ]
 [ 7   0   2   1 ]]
```

(b) $B =$
```
[[ -1  2   1  -3   1 ]
 [  1  -1  0   2   1 ]
 [  2  1   3   2   6 ]
 [ -1  3   2  -3   1 ]]
```

(c) $C = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 6 & 9 & 2 & -5 \\ 1 & 3 & 4 & 5 & 5 \\ 2 & 4 & 6 & 1 & -4 \end{bmatrix}$

(d) $D = \begin{bmatrix} 4 & 1 & 8 & -6 & -3 \\ 5 & -4 & 6 & -14 & -5 \\ 8 & 4 & 6 & 2 & 6 \\ 4 & -3 & 2 & -8 & -1 \\ -8 & 3 & -2 & 8 & -3 \end{bmatrix}$

2. For each of the following matrices A find a basis for the row space from among the original rows of A.

(a) $\begin{bmatrix} 1 & 3 & 1 & 2 \\ 2 & 6 & 3 & 4 \\ 3 & 9 & 4 & 6 \\ 4 & 2 & 5 & 1 \\ 5 & -5 & 5 & -4 \end{bmatrix}$

(b) $\begin{bmatrix} -1 & 1 & 2 & -1 \\ 2 & -1 & 1 & 3 \\ 1 & 0 & 3 & 2 \\ -3 & 2 & 2 & -3 \\ 1 & 1 & 6 & 1 \end{bmatrix}$

4.4 CHANGE OF BASIS

The ability to change from one basis to another is of fundamental importance. Each linear operator on a finite-dimensional vector space can be represented in a concrete fashion by a matrix, but the matrix itself depends upon the choice of the basis. Changing to a new basis often provides us with a simpler, or more well-structured, matrix.

Although the notation used to discuss change of basis will vary from textbook to textbook, most of them follow a style somewhat like what follows. Let $B = \{ u_1, u_2, \ldots, u_n \}$ be an ordered basis for a finite-dimensional vector space W (a subspace of R^n if you wish). Each vector w in W can be written in exactly one way as a linear combination of the vectors in basis B:

$$w = x_1u_1 + x_2u_2 + \ldots + x_nu_n .$$

The column vector $[\ w\ ]_B = [\ x_1\ \ x_2\ \ \ldots\ x_n]$ is called the *coordinate matrix* of w *relative to the B-basis*. In R^n (or C^n), finding the coordinate matrix $[\ w\ ]_B$ for a given vector w and given basis B usually entails solving a linear system. But we are primarily interested in how we move from the "old" ordered basis B to a "new" ordered basis $B' = \{\ v_1, v_2, \ldots, v_n\ \}$. The theorem describing how to do this is as follows:

Let $B = \{\ u_1, u_2, \ldots, u_n\ \}$ *and* $B' = \{\ v_1, v_2, \ldots, v_n\ \}$ *be ordered bases for a vector space W. Write each of the old basis vectors in terms of the new basis B′ and consider the coordinate matrices* $[u_1]_{B'}$, $[u_2]_{B'}$,... , $[u_n]_{B'}$. *If P is the n×n matrix whose* j^{th} *column is* $[u_j]_{B'}$, *then P is invertible and is the only matrix for which* $P[w]_B = [w]_{B'}$, *for all vectors w in W. We call P the* ***change-of-basis matrix from the B-basis to the B′ basis****. (Note that P depends upon the order of the basis vectors as well as the vectors themselves.)*

EXAMPLE 7. Find the change of basis matrix P from the "old" basis B to the "new" basis B' given below, then write $w = [\ 3\ \ -2\ \ -11\ \ 17\ ]$ in terms of the new basis.

$$B = \{\ [\ 10\ \ -3\ \ -3\ \ 10\ ]^T, [\ -3\ \ 23\ \ 10\ \ -21\ ]^T, [\ -3\ \ 10\ \ 7\ -13\ ]^T, [\ 10\ \ -21\ \ -13\ \ 30\ ]^T\ \},$$

$$B' = \{\ [\ 2\ \ 1\ \ -1\ \ 2\ ]^T, [\ 1\ \ 3\ \ 2\ \ -3\ ]^T, [\ -1\ \ 2\ \ 1\ \ -1\ ]^T, [\ 2\ \ -3\ \ -1\ \ 4\ ]^T\ \}.$$

(a) To find P we must write each vector in the B-basis in terms of the B'-basis. To do this, we consider the quadruple-augmented matrix and its reduced row echelon form:

$$\left[\begin{array}{cccc:cccc} 2 & 1 & -1 & 2 & 10 & -3 & -3 & 10 \\ 1 & 3 & 2 & -3 & -3 & 23 & 10 & -21 \\ -1 & 2 & 1 & -1 & -3 & 10 & 7 & -13 \\ 2 & -3 & -1 & 4 & 10 & -21 & -13 & 30 \end{array}\right] \quad \text{☞} \quad \left[\begin{array}{cccc:cccc} 1 & 0 & 0 & 0 & 2 & 1 & -1 & 2 \\ 0 & 1 & 0 & 0 & 1 & 3 & 2 & -3 \\ 0 & 0 & 1 & 0 & -1 & 2 & 1 & -1 \\ 0 & 0 & 0 & 1 & 2 & -3 & -1 & 4 \end{array}\right].$$

Thus the change-of-basis matrix P is

$$P = \begin{bmatrix} 2 & 1 & -1 & 2 \\ 1 & 3 & 2 & -3 \\ -1 & 2 & 1 & -1 \\ 2 & -3 & -1 & 4 \end{bmatrix}.$$

(b) For $w = [\,3\ \ -2\ \ -11\ \ 17\,]^T$, we must first find its coordinate matrix $[\,w\,]_B$:

$$\begin{bmatrix} 10 & -3 & -3 & 10 & 3 \\ -3 & 23 & 10 & -21 & -2 \\ -3 & 10 & 7 & -13 & -11 \\ 10 & -21 & -13 & 30 & 17 \end{bmatrix} \overset{\text{☞}}{\sim} \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & -3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

so that $[\,w\,]_B = [\,-1\ \ 2\ \ -3\ \ 1\,]$ and therefore $P[\,w\,]_{B'} = [\,5\ \ -4\ \ 1\ \ -1\,]$.

Activity Set 4.4

1. Consider the two sets $B = \{\,u_1, u_2, u_3\,\}$ and $B' = \{\,v_1, v_2, v_3\,\}$ in R^4, where

$u_1 = [\,1\ \ 0\ \ 2\ \ 0\,]^T$, $u_2 = [\,2\ \ 0\ \ 4\ \ -3\,]^T$, $u_3 = [\,1\ \ 2\ \ 2\ \ 1\,]^T$ and

$v_1 = [\,2\ \ 1\ \ 4\ \ -1\,]^T$, $v_2 = [\,1\ \ 2\ \ 2\ \ 4\,]^T$, $v_3 = [\,0\ \ 2\ \ 0\ \ 1\,]^T$.

(a) Show that both B and B' are independent sets of vectors and that Span B = Span B'.

(b) Let W = Span B = Span B'. Show that $w = [\,1\ \ 2\ \ 2\ \ -2\,]$ is in W.

(c) Find the change-of-basis matrix P from the B-basis to the B' basis for W.

(d) Use P to express w in terms of the B'-basis. Check your result by directly expressing w in terms of the B'-basis.

2. Repeat Activity 1, this time using the sets $B = \{ u_1, u_2, u_3, u_4 \}$ and $B' = \{v_1, v_2, v_3, v_4\}$ in R^5 where

$u_1 = [\ -4 \quad -2 \quad -1 \quad 0 \quad 2\ ]$ $u_3 = [\ 2 \quad 0 \quad -1 \quad 1 \quad -2]$

$u_2 = [\ 3 \quad 1 \quad 0 \quad 0 \quad 3\ ]$ $u_4 = [\ -1 \quad -1 \quad -1 \quad 0 \quad 6]$

and

$v_1 = [\ 2 \quad 0 \quad -1 \quad 0 \quad 6\ ]$ $v_3 = [\ -2 \quad -2 \quad -2 \quad 1 \quad -6\ ]$

$v_2 = [4 \quad -2 \quad -5 \quad 0 \quad 14]$ $v_4 = [\ 0 \quad 2 \quad 3 \quad 3 \quad -7\ ]$.

Take w to be the vector $w = [\ 4 \quad -4 \quad -8 \quad -3 \quad 21\ ]$.

5 ORTHOGONALITY

Orthogonality concepts lie at the very heart of modern linear algebra. Orthogonal vectors, orthogonal projections, orthogonal bases, orthogonal subspaces, and orthogonal matrices all combine to produce a rich and elegant theory as well as powerful numerical techniques and algorithms that are widely used in numerical linear algebra.

The geometry of R^3 is the place to begin. This geometry is easily extended to R^n or to C^n by means of the standard inner product. In R^n the standard inner product of vectors $x = [\, x_1 \;\; x_2 \ldots x_n \,]$ and $y = [\, y_1 \;\; y_2 \ldots y_n \,]$ is their dot product

$$x \bullet y = x_1 y_1 \; + \; x_2 \, y_2 \; + \ldots \; + \; x_n y_n \, .$$

In C^n, where the underlying scalars are complex numbers, the standard inner product is their Hermitian product $\bar{x} \bullet y$, where $\bar{x}$ denotes the vector conjugate to x. The HP-48 command **DOT** will return the dot product of any two vectors (real or complex) on stack levels 1 and 2. To obtain the Hermitian product of two complex vectors you must first apply the **CONJ** command, then **DOT**.

For any vector $x = [\, x_1 \; x_2 \; \ldots \; x_n \,]$ the command **ABS** will return its Euclidean length (*n*orm) $\| x \|_2 = \sqrt{| x_1 |^2 + | x_2 |^2 + \ldots + | x_n |^2}$ which is the usual notion of length in R^n or C^n. When applied to an $n \times n$ matrix $A = (a_{ij})$, **ABS** will return the *Frobenius matrix norm* $\| A \|_F = \left[\sum_{i,j} | a_{ij} |^2 \right]^{1/2}$. Three other vector and matrix norms are provided on the HP-48G series calculators, but we will not use them in this

chapter. You will find a brief summary of norms in Appendix 1, as well as a discussion of their application with the HP-48.

5.1 ORTHOGONAL VECTORS AND SUBSPACES

From now on, we shall restrict our attention to R^n. Recall that two vectors x, y are called *orthogonal* if their dot product is zero: $x \bullet y = 0$. In R^2 or R^3 this amounts to saying that x and y are perpendicular vectors. A set $B = \{ v_1, v_2, ..., v_k \}$ of mutually orthogonal vectors ($v_i \bullet v_j = 0$ for $i \neq j$) is called an *orthogonal set*, and any such set of non-zero vectors is linearly independent. Consequently, B is a basis for the subspace $W = \text{Span}[v_1, v_2, ..., v_k]$. An attractive feature of such a basis is the ease with which we can obtain the coordinates of any vector w in W:

$$(*) \quad w = \frac{w \bullet v_1}{\| v_1 \|^2} v_1 + \frac{w \bullet v_2}{\| v_2 \|^2} v_2 + ... + \frac{w \bullet v_k}{\| v_k \|^2} v_k \ .$$

Even better is when each basis vector has length 1, for then (*) becomes

$$w = (w \bullet v_1)v_1 + (w \bullet v_2)v_2 + ... + (w \bullet v_k)v_k \ .$$

Vectors of length 1 are sometimes called *normal* vectors, and we can "normalize" any vector v by simply dividing by its length:

$$\frac{v}{\| v \|} \text{ has length 1.}$$

By normalizing an orthogonal basis for W we can obtain a basis of mutually orthogonal, normal vectors, an *orthonormal basis*, and it is a fundamental result that any non-zero subspace of R^n has such a basis. The proof of this is the content of the **Gram-Schmidt process**, which we shall examine later.

Look again at the criterion for the orthogonality of a set $\{ v_1, v_2, ..., v_k \}$ of vectors in R^n: $v_i \bullet v_j = 0$ for $i \neq j$. Since $v_i \bullet v_j$ is the (i, j)-entry of the $k \times k$ matrix

A^TA, where A is the $n \times k$ matrix having $v_1, v_2, \ldots, v_k$ as its columns, we see that the set of vectors $\{ v_1, v_2, \ldots, v_k \}$ is orthogonal iff A^TA is a diagonal matrix:

$$A^TA = \begin{bmatrix} v_1 \bullet v_1 & & & \\ & v_2 \bullet v_2 & & \\ & & \ddots & \\ & & & v_k \bullet v_k \end{bmatrix}.$$

And clearly $\{ v_1, v_2, \ldots, v_k \}$ is an orthonormal set iff A^TA is the $k \times k$ identity matrix.

EXAMPLE 1. To determine whether $v_1 = [\ 1\ \ 0\ \ 1\ \ -1\ ]$, $v_2 = [\ 4\ \ -6\ \ 3\ \ 7\ ]$ and $v_3 = [\ -2\ \ 3\ \ 4\ \ 2\ ]$ are orthogonal, construct the matrix A having v_1, v_2 and v_3 as its columns:

$$\begin{bmatrix} 1 & 4 & -2 \\ 0 & -6 & 3 \\ 1 & 3 & 4 \\ -1 & 7 & 2 \end{bmatrix}.$$

Then check to see that

$$A^TA = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 110 & 0 \\ 0 & 0 & 33 \end{bmatrix},$$

so the vectors are orthogonal.

More generally, since the (i, j)-entry in AB is the dot product (row of A) • (col j of B), we see that $AB = 0$ iff the rows of A are orthogonal to the columns of B.

Two subspaces W_1 and W_2 of R^n are said to be *orthogonal subspaces* if every vector in one is orthogonal to every vector in the other:

$$u \bullet v = 0 \text{ for all } u \text{ in } W_1 \text{ and } v \text{ in } W_2.$$

By the *orthogonal complement* of a subspace W of R^n we mean the subspace $W^{\perp}$ of R^n consisting of all vectors in R^n that are orthogonal to W. The four fundamental subspaces associated with a matrix are examples. Indeed,

$$\begin{aligned} x \in NS(A) &<=> Ax = 0 \\ &<=> x \text{ is orthogonal to the rows of } A \\ &<=> x \text{ is orthogonal to } RS(A). \end{aligned}$$

Thus, the nullspace $NS(A)$ and the rowspace $RS(A)$ are orthogonal complements. Since the same is true for A^T, and the row space of A^T is the column space of A, we also have:

$$x \in NS(A^T) <=> x \text{ is orthogonal to } CS(A).$$

Thus the left nullspace $NS(A^T)$ and the column space $CS(A)$ are orthogonal complements.

Activity Set 5.1

1. Let $< x, y >$ denote the standard inner product of vectors x, y in R^n or C^n. Use the following pairs of vectors to verify the *Cauchy-Schwartz Inequality*: $| < x, y > | \leq \| x \| \ \| y \|$.

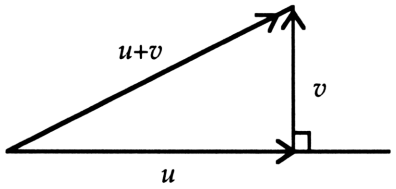
   (a) $x = [\ 1\ \ -2\ \ 3\ \ -5\ ]$, $y = [\ 6\ \ 9\ \ -7\ \ 3\ ]$ in R^4,

   (b) $x = [\ 1 + i\ \ -2 + 3i\ \ i\ ]$, $y = [\ 3\text{-}4i\ \ -i\ \ 5 + i\ ]$ in C^3,

   (c) Any two random vectors of your choosing in R^5.

2. Use the vectors in (a), (b) and (c) of Activity 1 to verify the triangle inequality: $\| x + y \| \leq \| x \| + \| y \|$

3. The Pythagorean Equality in R^n say that for any orthogonal vectors u and v, $\| u + v \|^2 = \| u \|^2 + \| v \|^2$.



Verify this equality for the following pairs of orthogonal vectors:

(a) $u = [\,-6 \;\; 4 \;\; 3 \;\; 7\,]$ and $v = [\,3 \;\; -2 \;\; 4 \;\; 2\,]$

(b) $u = [\,2 \;\; -1 \;\; -1 \;\; 1\,]$ and $v = [\,3 \;\; 2 \;\; 2 \;\; -2\,]^T$

4. Verify that each of the following sets of vectors forms an orthogonal set:

(a) $u_1 = [\,1 \;\; -1 \;\; 2 \;\; -1\,]$, $u_2 = [\,1 \;\; 2 \;\; 0 \;\; -1\,]$, $u_3 = [\,2 \;\; 0 \;\; 0 \;\; 2\,]$, $u_4 = [\,-2 \;\; 2 \;\; 3 \;\; 2\,]$

(b) $v_1 = [\,1/\sqrt{6} \;\; 1/\sqrt{6} \;\; 0 \;\; 2/\sqrt{6}\,]$, $v_2 = [\,-1/\sqrt{3} \;\; -1/\sqrt{3} \;\; 0 \;\; 1/\sqrt{3}\,]$, $v_3 = [\,-1/\sqrt{2} \;\; 1/\sqrt{2} \;\; 0 \;\; 0\,]$

(c) $w_1 = [\,-1.5 \;\; .5 \;\; .5 \;\; .5\,]$, $w_2 = [\,1 \;\; 1 \;\; 1 \;\; 1\,]$, $w_3 = [\,0 \;\; -2 \;\; 1 \;\; 1\,]$

5. For each of the matrices A given below:

(a) Find a basis for the nullspace $NS(A)$ of A and for the row space $RS(A)$ of A.

(b) Verify that $NS(A)$ and $RS(A)$ are orthogonal subspaces by checking that every basis vector for $NS(A)$ is orthogonal to every basis vector for $RS(A)$.

(c) Find a basis for the left nullspace $NS(A^T)$ of A and a basis for the column space $CS(A)$ of A.

(d) Verify that $NS(A^T)$ and $CS(A)$ are orthogonal subspaces by checking that every basis vector for $NS(A^T)$ is orthogonal to every basis vector for $CS(A)$.

(i)
$$\begin{bmatrix} 1 & 2 & 1 \\ -2 & 3 & 0 \\ -1 & -5 & -7 \end{bmatrix}$$

(ii)
$$\begin{bmatrix} 1 & 2 & -1 & 7 \\ 1 & 2 & 0 & 4 \\ -2 & -4 & 0 & -8 \end{bmatrix}$$

(iii)
$$\begin{bmatrix} 1 & -2 & 0 & 4 & 0 \\ 2 & -4 & 1 & 11 & 0 \\ -1 & 2 & 1 & -1 & 1 \\ -3 & 6 & 2 & -6 & 2 \end{bmatrix}$$

(iv)
$$\begin{bmatrix} 1 & 2 & 2 & -1 & 2 & 2 \\ 0 & 0 & 1 & -1 & 1 & 0 \\ 1 & 2 & 2 & 0 & 1 & 3 \\ 2 & 4 & 3 & -2 & 4 & 2 \\ 1 & 2 & 1 & -2 & 3 & 2 \end{bmatrix}$$

5.2 ORTHONORMAL BASES

We have already noted some of the advantages in having an orthonormal basis $v_1, v_2, \ldots, v_k$ for a subspace W of R^n. The basis vectors are mutually orthogonal and have length 1, so in this respect they are just like the standard basis vectors $e_1, e_2, \ldots, e_k$ for R^k, where e_j is column j of the identity matrix. Also, orthonormal bases are valued for the ease with which they enable us to write any vector w in W:

$$w = (w \bullet v_1)v_1 + (w \bullet v_2)v_2 + \ldots + (w \bullet v_k)v_k \ .$$

The coefficients in this equation are just dot products, and can be found by a simple matrix multiplication instead of the more involved process of solving a linear system:

$$\begin{bmatrix} \text{—} & v_1 & \text{—} \\ \text{—} & v_2 & \text{—} \\ & \vdots & \\ \text{—} & v_k & \text{—} \end{bmatrix} \begin{bmatrix} \\ w \\ \\ \end{bmatrix} = \begin{bmatrix} v_1 \bullet w \\ v_2 \bullet w \\ \vdots \\ v_k \bullet w \end{bmatrix} .$$

And finally, if vectors u and w in W are written in terms of the orthonormal basis vectors, $u = x_1v_1 + \ldots + x_kv_k$ and $w = y_1v_1 + \ldots + y_kv_k$, then it is easy to see that the dot product of u and w is given by the simple equation

$$u \bullet w = x_1y_1 + \ldots + x_ky_k \ .$$

Thus, dot products in W appear just as they do when we are using the standard basis vectors in R^n. Each non-zero subspace W of R^n has an orthonormal basis, and we shall soon consider the Gram-Schmidt process for constructing such a basis. But first, we need to review the notion of the *orthogonal projection* of one vector onto another.

The word *projection* comes from visualizing vectors in R^2 and R^3 as arrows, and pictures such as the following:

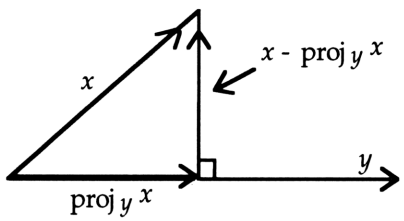


Figure 1.

As Figure 1 suggests, the orthogonal projection of vector x onto vector y , $\text{proj}_y x$ is a scalar multiple of vector y , and the vector $x - \text{proj}_y x$ is orthogonal to y:

(i) $\text{proj}_y x = ky$, for some scalar k

(ii) $x - \text{proj}_y x$ is orthogonal to y.

These two facts combine to tell us the scalar k:

$0 = (x - ky) \bullet y$ *(condition (ii))*

$0 = x \bullet y - k(y \bullet y)$

$$k = \frac{x \bullet y}{y \bullet y}$$

Thus,

$$(1) \quad \text{proj}_y x = \left(\frac{x \bullet y}{y \bullet y} \right) y$$

When working in R^n with $n > 3$, we take equation (1) as the definition of the projection vector $\text{proj}_y x$; it is then easy to verify that condition (ii) also holds.

The following program, PROJ, can be used to calculate the orthogonal projection of x onto y.

PROJ (Projection vector)

Inputs: level 2: a vector x

level 1: a vector y

Effect: Returns the orthogonal projection vector $\text{proj}_y x$

```
« → X Y « X Y DOT Y * Y Y DOT / » »
```

EXAMPLE 2. For x = [5 15 5] and y = [3 4 5] find $\text{proj}_y x$ and verify that $x - \text{proj}_y x$ is orthogonal to y.

Put two copies vector x on the stack, followed by two copies of vector y. The command 4 ROLLD will rearrange the stack to

level 4: y

3: x

2: x

1: y

Press PROJ to see $\text{proj}_y x$, then subtract to see $x - \text{proj}_y x$, then use DOT to see $y \bullet (x - \text{proj}_y x)$.

More generally, we can consider the orthogonal projection $\text{proj}_W b$ of a vector b onto a subspace W:

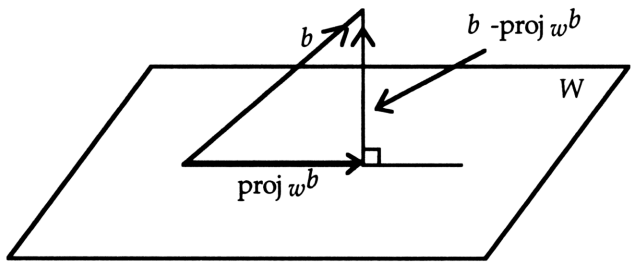


Figure 2.

Later we shall define the projection vector $\text{proj}_W b$ precisely; but for now all we need to remember is what our geometric intuition tells us: that *vector* $b - \text{proj}_W b$ *is orthogonal to each vector in* W.

The Gram-Schmidt Process

The **Gram-Schmidt** process is a procedure for building an orthonormal basis $q_1, q_2, \ldots, q_k$ from a given basis $x_1, x_2, \ldots, x_k$ for a subspace W. Here's how it works in R^n.

Let q_1 be the *normalized* version of x_1 : $q_1 = \dfrac{x_1}{\| x_1 \|}$. Then, inductively, having constructed orthonormal vectors $q_1, \ldots, q_j$, we construct q_{j+1} as follows:

(*) $q_{j+1} = x_{j+1}$ – (the sum of the projections of x_{j+1} onto $q_1, q_2, \ldots, q_j$), *normalized*.

Thus, before normalization, $q_{j+1} = x_{j+1}$ – (the projection of x_{j+1} onto the subspace spanned by $q_1, \ldots, q_j$).

Let's look at several steps:

Step 1: $q_1 = x_1$, *normalized*

Step 2: $q_2 = x_2 - \underbrace{(x_2 \bullet q_1)\, q_1}_{\text{the projection of } x_2 \text{ onto } q_1}$, *normalized*

Step 3: $q_3 = x_3 - \underbrace{(x_3 \bullet q_1)\, q_1 - (x_3 \bullet q_2)\, q_2}_{\text{the sum of the projections of } x_3 \text{ onto } q_1 \text{ and } q_2}$, *normalized*

.
.
.

etc.

This is the standard Gram-Schmidt process (there are variations). You should recall that, at each stage, Span [$x_1, \ldots, x_j$] = Span [$q_1, \ldots, q_j$], so when we're done, W = Span [$x_1, \ldots, x_k$] = Span [$q_1, \ldots, q_k$] and we have an orthonormal basis for W.

To use the HP-48G or 48GX, we begin with the basis vectors stored as variables **X1, X1, ..., XK** in user memory and execute a simple one-line program to carry out the construction at each step.

Step 1: « X1 X1 ABS / ENTER EVAL ' Q1 STO (calculates q_1 and stores it as Q1)

Step 2: « X2 X2 Q1 PROJ – DUP ABS / ENTER EVAL ' Q2 STO (calculates q_2 and stores it as Q2)

Step 3: « X3 X3 Q1 PROJ – X3 Q2 PROJ – DUP ABS / ENTER EVAL ' Q3 STO (calculates q_3 and stores it as Q3)

.
.
.

. . . and so on.

The purpose of the programs (instead of simply doing the calculations on the stack) is simple: you can review your work before execution.

EXAMPLE 3. Apply the above construction to the vectors x_1 = [2 1 0], x_2 = [0 1 1], and x_3 = [2 0 2] in R^3. You should obtain these results:

$Q1$: [.894427191 .4472135955 0]

$Q2$: [-.298142697 .596284794 .7453559925]

$Q3$: [.333333333334 -.666666666665 .666666666665] .

You may not recognize the entries, but the **Q1**, **Q2** and **Q3** you constructed in Example 3 are actually the calculator's approximations to $q_1 = \frac{1}{\sqrt{5}}$ [2 1 0], $q_2 = \frac{1}{3\sqrt{5}}$ [-2 4 5], and $q_3 = \frac{1}{6}$ [2 -4 4]. Once you have **Q1**, **Q2**, **Q3** as stored variables, you should check to see how close they are to being orthonormal by putting **Q1**, **Q2**, **Q3** on the stack (in this order), pressing 3 [COL→] to create a matrix $Q = \begin{bmatrix} | & | & | \\ Q_1 & Q_2 & Q_3 \\ | & | & | \end{bmatrix}$, then [ENTER] [TRN] [SWAP] [×] to see Q^TQ. Clean up round-off error with 11 **RND** and you should see I_3.

The Gram-Schmidt process, as we have presented it, is numerically unstable in floating point arithmetic. That is, round-off errors may conspire to produce vectors that are not, numerically, orthogonal. Although there is a variation of the Gram-Schmidt process that is more stable, it is not so geometrically obvious. In practice other methods are used: Householder reflections or Givens rotations. These are orthogonal matrices that can be used very effectively to build orthonormal vectors.

Finally, we return briefly to an idea that we met at the beginning of this section and which was central to our discussion of least squares solutions: the projection $\text{proj}_W b$ of a vector b onto a non-zero subspace W of R^n. It is not difficult to show that any vector b can be written as a sum of two vectors:

$$b = \hat{b} + z$$

where $\hat{b}$ is in W and z is in $W^{\perp}$, the orthogonal complement of W. Moreover, this decomposition of b is unique: for any given vector b, $\hat{b}$ and z are unique. Vector $\hat{b}$ is known as the *projection of* b *onto* W:

$$\hat{b} = \text{proj}_W b.$$

Since $\hat{b}$ is uniquely determined by b and W, it is independent of any particular basis that we may use to describe W. Nevertheless, the proof of the existence of $\hat{b}$ shows that we can always *describe* it in terms of any orthonormal basis $\{ v_1, v_2, \ldots, v_k \}$ for W:

$$\text{proj}_W b = (b \bullet v_1)v_1 + (b \bullet v_2)v_2 + \ldots + (b \bullet v_k) v_k.$$

Thus $\text{proj}_W b$ is the sum of the projections of vector b onto the individual basis vectors in any orthonormal basis for W.

Activity Set 5.2

1. (a) Use the **RANM** command to generate a random 4×3 matrix over Z_{10} whose columns will be called u, v and w.

   (b) Find $\text{proj}_v u$ and verify that u - $\text{proj}_v u$ is orthogonal to v.

   (c) Find $\text{proj}_w u$ and verify that u - $\text{proj}_w u$ is orthogonal to w.

2. (a) Use the **RANM** command to generate a random 4×3 matrix over Z_{10} whose columns will be called x_1, x_2 and x_3.

(b) Construct an orthonormal basis $\{ q_1, q_2 \}$ for W = Span $[x_1 \ x_2]$.

(c) Find the projection vector $\text{proj}_W x_3$ of x_3 onto W.

(d) Verify that $x_3 - \text{proj}_W x_3$ is orthogonal to W by checking that it is orthogonal to both x_1 and x_2 .

3. (a) Use the **RANM** command to generate a random 5×4 matrix over Z_{10} whose columns will be called x_1, x_2, x_3 and x_4.

(b) Construct an orthonormal basis $\{ q_1, q_2 , q_3 \}$ for W = Span $[x_1 \ x_2 \ x_4]$.

(c) Find the projection vector $\text{proj}_W x_3$ of x_3 onto W.

(d) Verify that $x_3 - \text{proj}_W x_3$ is orthogonal to W by checking that it is orthogonal to x_1, x_2 and x_4 .

4. (a) Normalize v = [1 -2 1 -3 1] to get a unit vector, then convert it to a column matrix w.

(b) Use w to build a Householder matrix $H = I - 2ww^T$.

(c) Verify that H is orthogonal.

(d) Look at H. What else is obvious?

(e) In view of your conclusion in (c), without calculating, what will H^{-1} be?

(f) Verify your result in (d) with a calculation.

5.3 ORTHOGONAL MATRICES AND QR-FACTORIZATIONS

Orthonormal bases are, in essence, rectangular coordinate systems. But what do we know about the change of basis matrix, call it Q, from one orthonormal basis B to another orthonormal basis B'? Without going into the details here, it is not hard to

see that $Q^TQ = I$, or in other words, that *the columns of Q are orthonormal vectors.* Even more is true: since Q is square, the equation $Q^TQ = I$ guarantees that also $QQ^T = I$, which says that Q *also has orthonormal rows and* $Q^{-1} = Q^T$.

In general, we shall call a square matrix Q **orthogonal** if $Q^TQ = I$. Any orthogonal matrix Q has orthonormal columns, orthonormal rows, and $Q^T = Q^{-1}$. Two other properties of such matrices are worth noting:

(i) Q preserves lengths: $\| Qx \| = \| x \|$, all x

(ii) $\det Q = \pm 1$.

If we are called upon to solve a linear system $Qx = b$ with Q orthogonal, it is easy enough: $x = Q^{-1}b = Q^Tb$.

Not only do orthogonal matrices occur when we change from one orthonormal basis to another (as for instance, when we *rotate* R^3), but they lie at the very heart of the Gram-Schmidt process. In fact, just as Guassian elimination without row interchanges on a matrix A amounts to an LU-factorization $A = LU$, the Gram-Schmidt process applied to a matrix A having independent columns amounts to a **QR-factorization $A = QR$** where Q has orthonormal columns and R is an invertible upper triangular matrix.

To see this, look back at the steps in the Gram-Schmidt process, and in each step solve for the x-vector:

Step 1: x_1 is a scalar multiple of q_1, say $x_1 = r_{11}q_1$

Step 2: x_2 is a linear combination of q_1 and q_2, say $x_2 = r_{12}q_1 + r_{22}q_2$

Step 3: x_3 is a linear combination of q_1, q_2 and q_3, say $x_3 = r_{13}q_1 + r_{23}q_2 + r_{33}q_3$

.
.
.

Step j: x_j is a linear combination of $q_1, q_2, \ldots, q_j$, say $x_j = r_{1j}q_1 + r_{2j}q_2 + \ldots + r_{jj}q_j$.

Let A be the matrix having $x_1, x_2, \ldots, x_n$ as its columns, left-to-right, and let Q be the matrix having $q_1, q_2, \ldots, q_n$ as its columns, left-to-right. Let R be the right triangular matrix determined from the coefficients r_{ij} above:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\ & r_{22} & r_{23} & \cdots & \cdot \\ & & r_{33} & & \cdot \\ & & & \ddots & \cdot \\ & & & & r_{nn} \end{bmatrix}.$$

In terms of the matrices A, Q, and R the above steps show us that

$$\begin{aligned} \text{col 1 of } A &= Q(\text{col 1 of } R) \\ \text{col 2 of } A &= Q(\text{col 2 of } R) \\ \text{col 3 of } A &= Q(\text{col 3 of } R) \\ &\vdots \\ \text{col } j \text{ of } A &= Q(\text{col } j \text{ of } R) \end{aligned}$$

so that we have $A = QR$. Moreover, since the vectors $q_1, q_2, \ldots, q_n$ are orthonormal, we know that $Q^TQ = I$ and that the i^{th} coefficient in the linear combination

$$x_j = r_{1j}q_1 + r_{2j}q_2 + \ldots + r_{jj}q_j$$

is given by $r_{ij} = q_i \bullet x_j$. Matrix R is invertible since its diagonal entries are positive: $r_{11} > 0$ because $r_{11} = \| x_1 \|$, $r_{22} > 0$ because $r_{22} = \| x_2 - (x_2 \bullet q_1)q_1 \|$, etc. Finally, using $Q^TQ = I$ and $A = QR$ we can obtain R as follows: $Q^TA = Q^TQ\,R = R$.

Thus, when the Gram-Schmidt process is applied to the columns of an m×n matrix A whose columns are independent we get a factorization $A = QR$, where Q is the same size as A and has orthonormal columns, and $R = Q^TA$ is the $n \times n$ invertible upper triangular matrix whose non-zero entries are given by $r_{ij} = q_i \bullet x_j$.

To construct the factors Q and R on the HP-48G or 48GX:

(1) Start with A and its columns X1, X2, ..., XN stored in user memory.

(2) Construct Q1, Q2, ..., Q_N from X1, X2, ..., XN by the Gram-Schmidt process.

(3) Construct and store matrix Q:

Q1 Q2 ... QN N [COL→] ['] Q [STO] .

(4) Construct and store matrix R: $R = Q^T A$.

You can verify that $A = QR$ as follows:

Press [A] [Q] [R] [×] to put A on level 2 and $Q * R$ on level 1, then use the command RND as necessary to clean-up round-off error in $Q * R$. Now press [SAME]. ([SAME] is located on the second page of the PRG TEST menu; a 1 indicates $A = QR$, and a 0 indicates $A \neq QR$. If you forget to clean up round-off error, you probably won't get $A = QR$.)

EXAMPLE 4. Continuing with the vectors from EXAMPLE 3, construct A as

$$A = \begin{bmatrix} 2 & 0 & 2 \\ 1 & 1 & 0 \\ 0 & 1 & 2 \end{bmatrix}.$$

After constructing Q you should see

$$Q = \begin{bmatrix} .894427191 & -.298142397 & .33333333334 \\ .4472135955 & .596284794 & -.666666666665 \\ 0 & .7453559925 & .666666666665 \end{bmatrix}.$$

After constructing R you should see

$$R = \begin{bmatrix} 2.2360679775 & .4472135955 & 1.788854382 \\ 0 & 1.3416407865 & .894427191 \\ .000000000003 & 0 & 2 \end{bmatrix}.$$

Verify that $A = QR$:

[A] [Q] [R] [*] 11 RND [SAME] returns 1. Now purge R, Q, A, Q3, Q2, Q1, X3, X2, X1 from user memory.

The factorization $A = QR$ of a matrix A with independent columns produced by the Gram-Schmidt process is usually the only type of QR-factorization encountered in an introductory study of linear algebra. But QR-factorizations of more general matrices, obtained by more sophisticated numerical methods, are in widespread use by the professional matrix codes that are used in science and engineering. The HP-48G and 48GX calculators incorporate such professional code for a variety of applications, including producing least squares solutions to linear systems, and for the calculation of eigenvalues and eigenvectors. Although a detailed discussion of these ideas is beyond the scope of this brief chapter, a few comments on the QR-factorization made accessible to users of the HP-48G series calculators is in order.

The command QR (located on the MTH MATR FACTR menu) will return a QR-factorization of any $m \times n$ matrix on level 1: $AP = QR$. Here, Q is an $m \times m$ orthogonal matrix, R is an $m \times n$ upper trapezoidal matrix, and P is an $n \times n$ permutation matrix. Matrix P appears on level 1, R on level 2, and Q on level 3 of the stack. Likewise, the adjacent command LQ will return an LQ-factorization of matrix A (the QR-factorization of A^T). Here, A is factored as $PA = LQ$ where P is an $m \times m$ permutation matrix, L is an $m \times n$ lower trapezoidal matrix, and Q is an $n \times n$ orthogonal matrix.

For example, with the matrix

$$A = \begin{bmatrix} -5 & -2 & 2 & 0 \\ 1 & 3 & -5 & 7 \\ -9 & 2 & 5 & 0 \end{bmatrix}$$

on level 1, executing the command QR will return the permutation matrix

```
      [ [ 1  0  0  0 ]
      [ 0  0  1  0 ]
P =   [ 0  0  0  1 ]
      [ 0  1  0  0 ] ]
```

to level 1, the upper trapezoidal matrix

```
      [ [ 10.3440804328    .676715542332    -.483368244523    -5.80041893427 ]
R =   [          0        -6.96721293451    -3.06106659926     4.46014305107 ]
      [          0                 0         -2.7196004146     -.67990010365 ] ]
```

to level 2, and the orthogonal matrix

```
      [ [ -.483368244523        -4.69488742217E-2     .874157276122 ]
Q =   [ 9.66736489046E-2        -.995316133501          -8.E-16     ]
      [ -.870062840141          -.845079735991E-2     -.485642931179 ] ]
```

to level 3. A quick calculation (using 10 RND to clean up round-off error) shows that $QR = AP$.

Activity Set 5.3

1. Consider the following matrix

```
      [ [ 1   4  -2 ]
      [ 1  -1   4 ]
A =   [ 1  -1   0 ]
      [ 1   4   2 ] ]
```

   (a) Verify that the column vectors of A are linearly independent.

   (b) Without using the built-in command QR, construct a QR-factorization of matrix A. Verify your results.

(c) Use the built-in command **QR** to obtain an $AP = QR$ factorization of matrix A. Verify your results.

(d) Verify that the Q matrix from (c) is an orthogonal matrix.

2. Repeat Activity 1 for the following matrix

$$A = \begin{bmatrix} 1 & -2 & -1 \\ 2 & 0 & 1 \\ 2 & -4 & 2 \\ 4 & 0 & 0 \end{bmatrix}.$$

3. Now that you have mastered the traditional Gram-Schmidt algorithm, you may find it convenient in the future to use the following program GS, which automatically produces the orthonormal vectors. Actually, the program implements the modified Gram-Schmidt algorithm in order to be as numerically accurate as possible.

GS (Gram-Schmidt Algorithm)

Input: levels 2 – *n*: vectors X1, . . ., XN
level 1: the number *n* of vectors

Effect: Applies the modified Gram-Schmidt Algorithm to vectors X1, . . . , XN and returns the orthonormal vectors Q1, . . ., QN

```
« → n « →LIST →L « 1 n FOR k 'L(k)' EVAL DUP ABS / 'L(k)'
STO IF k n ≠ THEN k 1 + n FOR j 'L(j)' EVAL DUP 'L(k)' EVAL
DUP 3 ROLLD DOT * – 'L(j)' STO NEXT ELSE END NEXT L OBJ→
DROP » » »
```

Rework Activities 1 and 2 using this program.

5.4 LEAST SQUARES SOLUTIONS

An important application of QR- and LQ-factorizations is to obtain least squares solutions to linear systems.

Sometimes we seek to solve a linear system $Ax = b$ for which either no solution exists, or else there are infinitely many solutions from which to choose. In either case, we may seek a vector x for which the distance $\| Ax - b \|_2$ from Ax to b is as small as possible. Here, $\| \ \|_2$ denotes the 2-norm of the included vector and such an x is called a *least squares solution* because minimizing $\| Ax - b \|_2$ is equivalent to minimizing $\| Ax - b \|_2^2$, which is a sum of squares.

We thus seek x so that vector Ax, which lies in the column space W of A, is closest to b. Looking back at Figure 2 we see that Ax must be the projection of vector b onto the column space W, in which case $b - Ax$ is orthogonal to each vector in $CS(A)$. Remembering that the vectors that are orthogonal to $CS(A)$ are precisely the vectors in $NS(A^T)$, we are practically forced into $A^T[b - Ax] = 0$, or equivalently

$$A^TAx = A^Tb.$$

The linear system $A^TAx = A^Tb$ is referred to as the system of *normal equations*; thus, vector x is a *least squares* solution to $Ax = b$ iff it is a solution to the associated system of normal equations. In general, the normal equations will have more than one solution. But in the special case that A has full column rank, *i.e.*, rank $A = n$, we know that A^TA is invertible, so the system of normal equations $A^TAx = A^Tb$ has a *unique* solution x. Since, in general, there may be more than one least squares solution, we desire one having minimum norm; that is, a least squares solution x for which $\| x \|_2$ is minimal among all such solutions.

More generally, with an array B on level 2 and a matrix A on level 1 of the stack, the command LSQ (a menu key is on the MTH MATR menu) will return a

minimum norm least squares solution of the generalized system $AX = B$. If B is a vector then the solution X has the minimum norm $\| X \|_2$ over all vectors X that minimize $\| AX - B \|_2$. If B is a matrix, then each column X_j of X is a minimum norm least squares solution of $AX_j = B_j$. The **LSQ** command constructs the solution X by computing a "complete orthogonal factorization" of the coefficient matrix A using either, or both, of the QR- and LQ-factorizations of A. Complete orthogonal factorizations are somewhat beyond the scope of introductory linear algebra.

For example, to obtain a minimum norm least squares solution to the linear system

$$\begin{aligned} 2x_1 - 3x_2 + x_3 - 3x_4 + 2x_5 &= 6 \\ -2x_1 + 3x_2 - x_3 + 4x_4 + x_5 &= -5 \\ 6x_1 - 9x_2 + 7x_3 - 7x_4 + 5x_5 &= 20 \\ -2x_1 + 3x_2 + 3x_3 + 3x_4 - 9x_5 &= -6 \end{aligned} ,$$

enter vector [6 -5 20 -6] onto level 2, then the coefficient matrix

```
[[ 2 -3  1 -3  2]
 [-2  3 -1  4  1]
 [ 6 -9  7 -7  5]
 [-2  3  3  3 -9]]
```

onto level 1 and execute the **LSQ** command. The solution that is returned is

x = [.47724708537 -.715870628056 .809514855209 -.38779751786 .462579917262].

Fitting Curves to Data

Least squares solutions arise in curve fitting problems. Suppose we have n data points (x_1, y_1), (x_2, y_2), ... , (x_n, y_n) where all the x_j's are distinct. Consider the problem of finding a polynomial $P(t) = c_0 + c_1 t + ... + c_m t^m$ of degree $\leq m$ that passes

through these data points, *i.e. fits* the data. We know from elementary mathematics that we can fit a line to any two data points, and a quadratic polynomial to any three. Therefore, we shall require that the number n of data points exceed the degree m of the polynomial: $n \geq m + 1$. Our requirements are $P(x_i) = y_i$ for $i = 1, \ldots, n$ or

$$\begin{aligned} c_0 + c_1 x_1 + \ldots + c_m x_1^m &= y_1 \\ c_0 + c_1 x_2 + \ldots + c_m x_2^m &= y_2 \\ &\vdots \\ c_0 + c_1 x_n + \ldots + c_m x_n^m &= y_n \end{aligned} .$$

This linear system has n equations and $(m + 1)$-unknowns (the coefficients of the polynomial $P(t)$).

In terms of matrices, the system is $Ac = y$, where

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^m \\ 1 & x_2 & x_2^2 & \ldots & x_2^m \\ & & \vdots & & \\ 1 & x_n & x_n^2 & \ldots & x_n^m \end{bmatrix},$$

$$\text{and } c = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_m \end{bmatrix} \text{ and } y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

Since there are at least as many equations as unknowns, the system will, in general, be overdetermined and we naturally seek a least squares solution. However, A has independent columns, for if $Ac = 0$ were to have a non-zero solution, this would mean that there is a non-zero polynomial $P(t)$ of degree $\leq m$ having $m + 1$ roots . . .

an impossibility. Then, since A has independent columns, there is a *unique least squares solution*, given by the unique solution to the normal equations

$$(A^TA)c = A^Ty.$$

It is tempting to obtain the least squares solution by calculating $x = (A^TA)^{-1}A^Ty$ or by applying Gaussian elimination via the / command. But the coefficient matrix A^TA is likely to be *ill-conditioned*, in the sense that solutions to $(A^TA)x = A^Ty$ are somewhat sensitive to perturbations caused by round-off errors. This is especially the case with large data sets where the x-values are equally spaced. Thus, good computational practice suggests that the above two approaches to solving the normal equations be abandoned in favor of the more sophisticated one provided by the calculators built-in LSQ command. We shall return to this conditioning question in the Activities.

The following program, P.FIT, will create the coefficient matrix A.

P.FIT (Polynomial Fit Matrix)

Input: level 2: an integer m
level 1: a list $\{x_1, x_2, ..., x_n\}$

Effect: Returns the matrix

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ & & \vdots & & \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix}$$

```
« DUP SIZE → m lst n « 1 n FOR j lst j GET → x « 1 1 m FOR
i x i ^ NEXT » NEXT n m 1 + 2 → LIST → ARRY » »
```

EXAMPLE 5. Find the least squares cubic polynomial that fits the data: (1, .6), (2, 1.2), (3, 2), (4, 2.8), (5, 4.1).

Key in the number 3, then the list { 1 2 3 4 5 } of the x-coordinates of the data and press P.FIT to see

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \\ 1 & 5 & 25 & 125 \end{bmatrix} .$$

Now put y = [.6 1.2 2 2.8 4.1] on the stack, SWAP levels 1 and 2, and execute the LSQ command to see the least squares solution c = [-.16 .85 -.125 .025]. Thus the least squares cubic polynomial fit is $P_3(t) = -.16 + .85t - .125t^2 + .025t^3$.

Activity Set 5.4

1. Consider the overdetermined linear system

$$\begin{aligned} 2x_1 - x_2 + x_3 &= 0 \\ 2x_1 + 3x_2 - x_3 &= 1 \\ 3x_1 - 3x_2 + 3x_3 &= 8 \\ 3x_1 + x_2 + x_3 &= 6 \end{aligned} .$$

   (a) Show that this system is inconsistent (i.e., has no solution in the usual sense) but that the coefficient matrix A has full column rank.

   (b) Obtain the unique least squares solution by applying Gaussian elimination to the associated system of normal equations.

   (c) Obtain the unique least squares solution by applying the LSQ command.

2. (a) Fill-in the following table of values for $f(x) = (x + 2)^2 e^{-x}$ (round to 3 decimal places).

x	-2.2	-1	.5	1.5	3
$y = (x+2)^2 e^{-x}$					

(b) Plot the 5 data points.

(c) Find the least squares cubic polynomial $P_3(x)$ for this data; overlay your data plot with the graph of $P_3(x)$.

(d) Find the least squares polynomial $P_4(x)$ of degree 4 for this data; overlay your data plot with the graph of $P_4(x)$.

3. Augment the data in our last example with a sixth data point (6, 5.8), so that the data becomes (1, .6), (2, 1.2), (3, 2), (4, 2.8), (5, 4.1), (6, 5.8) and consider the problem of fitting a cubic polynomial to this data.

   (a) Build the coefficient matrix A^TA of the system of normal equations and obtain an approximation to its condition number with the command COND on the MTH MATR NORM menu. This large condition number ($\approx 3 \times 10^6$) indicates that A^TA is ill-conditioned so that using Gaussian elimination or matrix inversion to solve the normal equations may produce inaccurate results.

   (b) Find the least squares cubic polynomial by using the LSQ command, then with the / command; note that the two solutions agree to 12 decimal places.

   (c) Now find the least squares cubic polynomial by applying the RREF command to the augmented matrix, then by inverting the coefficient matrix. Compare the accuracy of the two solutions with those obtained above in (b).

6 EIGENVALUES AND EIGENVECTORS

Eigenvalue-eigenvector considerations are of paramount importance in many of the applications of linear algebra to science and engineering, especially in those applications involving systems of linear differential equations. The HP-48G/GX calculators can assist students in developing conceptual understandings by removing the computational burden associated with hand calculation of characteristic polynomials, eigenvalues and associated eigenvectors, and the construction of diagonalizing matrices. We have already seen how to use the calculators to solve linear systems (useful for finding eigenvectors) and to construct orthonormal bases. What remains is to see how they might be reasonably used in eigenvalue-eigenvector investigations.

6.1 THE CHARACTERISTIC POLYNOMIAL

Given a square matrix A of order n, any real or complex number λ for which there is a non-zero vector x such that $Ax = \lambda x$ is called an *eigenvalue* of A, and the vector x is called an associated *eigenvector*. If matrix A is regarded as an operator acting on vector x, then $Ax = \lambda x$ simply says that A sends x onto a scalar multiple of itself. To find such pairs (λ, x) we consider the equation $Ax = \lambda x$, which is clearly equivalent to $(A - \lambda I)x = 0$. For a given eigenvalue λ, x is an associated eigenvector iff x is a non-zero solution to the homogeneous linear system with coefficient matrix $A - \lambda I$. So the eigenvectors of A associated with λ are just the non-zero vectors in the nullspace of $A - \lambda I$. We often call this nullspace the *eigenspace* of A determined by λ.

Although it is no easy task to find the eigenvalues of matrix A, theoretically things are simple. For we know that $(A - \lambda I)x = 0$ has a non-zero solution iff $A - \lambda I$ is singular, which happens precisely when $det(A - \lambda I) = 0$. The left-hand side, $det(A - \lambda I)$, is a polynomial of degree n in λ, called the *characteristic polynomial* of matrix A. Some writers prefer to use $det(\lambda I - A)$ instead, but the difference is minor since these two polynomials differ only by a factor of $(-1)^n$. What really counts is that the eigenvalues of A are the roots of either of these polynomials, and for any such root λ the associated eigenvectors are the non-zero solutions to the homogeneous linear system $(A - \lambda I)x = 0$.

All this is rather elegant from a purely algebraic viewpoint, but it can be a computational nightmare. In the first place, the defining equation for the characteristic polynomial, $det(A - \lambda I)$, is computationally impractical for all but small, highly-specialized matrices. And secondly, it is no easy task to determine the roots of a polynomial.

Given an $n \times n$ matrix A, the calculator program **CHAR** will calculate the coefficients of the monic polynomial $det(\lambda I - A) = \lambda^n + c_{n-1}\lambda^{n-1} + ... + c_1\lambda + c_0$, which is the characteristic polynomial of A, or $(-1)^n$ times the characteristic polynomial of A, depending upon your point of view. The program implements the SOURIAU-FRAME method, which uses traces of the first n powers of A.

CHAR (Characteristic polynomial)

Input: level 1: an $n \times n$ matrix A

Effect: returns a vector [1 c_{n-1} ... c_1 c_0] of coefficients of $det(\lambda I - A) = \lambda^n + c_{n-1}\lambda^{n-1} + ... + c_1\lambda + c_0$

```
« DUP SIZE 1 GET {1} → Mtx n Poly « Mtx 1 n FOR j 0 1 n FOR
k OVER { k k } GET + NEXT j NEG / 'Poly' OVER STO+ Mtx DUP
ROT * SWAP ROT * + NEXT DROP Poly OBJ→ →ARRY » »
```

EXAMPLE 1. Enter the following matrix onto level 1:

```
[[ 4 -8  2  5 ]
 [ 0  1 -6 -2 ]
 [-9  0  7  1 ]
 [ 7  3 -8  9 ]]
```

Execute CHAR to see the vector of coefficients [1 -21 144 -421 -4623]. Thus $det(\lambda I - A) = \lambda^4 - 21\lambda^3 + 144\lambda^2 - 421\lambda - 4623$. Retrieve the matrix with UNDO and then execute the TRACE command (on the MTH MATR NORM menu) to see 21 for the trace.

Activity Set 6.1

1. (a) Generate and store random 3×3 matrices A and B over Z_{10}.

   (b) Compare trace A, trace B and trace $(A + B)$. What do you observe?

   (c) Repeat (a) - (b) using random 4×4 matrices.

   (d) Formulate a conjecture on the basis of your observations. Prove your conjecture.

2. (a) Generate a random 3×4 matrix A and a random 4×3 matrix B, both over Z_{10}.

   (b) Compare trace (AB) and trace (BA); what do you observe?

   (c) Repeat (a) - (b) for random 3×5 and 5×3 matrices over Z_{10}.

   (d) Formulate a conjecture on the basis of your observations. Prove your conjecture.

3. (a) Generate a random 3×3 matrix A over Z_{10}.

   (b) Calculate the characteristic polynomials of A and A^T. What do you observe?

   (c) Repeat (a) - (b) for random 4×4 and 5×5 matrices over Z_{10}.

   (d) Formulate a conjecture based upon your observations.

   (e) Prove your conjecture.

4. Generate and store random 3×3, 4×4, and 5×5 matrices over Z_{10}.

   (a) For each of these matrices A, calculate $detA$, $traceA$, and the polynomial $det(\lambda I - A)$. What do you observe?

   (b) Formulate your observations into conjectures; then discuss your conjectures with your instructor.

5. Let $A_n(1)$ denote the $n \times n$ matrix of all 1's.

   (a) Find $det[\lambda I - A_n(1)]$ for $n = 2, 3, 4, 5$.

   (b) For arbitrary n what will $det[\lambda I - A_n(1)]$ be?

   (c) What are the eigenvalues of $A_n(1)$?

6. Given the polynomial $p(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + \ldots + c_1\lambda + c_0$ its *companion matrix* is

$$C = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ -c_0 & -c_1 & -c_2 & \ldots & -c_{n-1} \end{bmatrix}.$$

   (a) For each of the following polynomials find its companion matrix and the characteristic polynomial $det[\lambda I - C]$ of the companion matrix:

   (i) $p(\lambda) = \lambda^3 + 5\lambda^2 - 3\lambda + 2$

   (ii) $p(\lambda) = \lambda^4 - 6\lambda^3 + 2\lambda^2 - 5\lambda + 7$

   (iii) $p(\lambda) = \lambda^5 + 5\lambda^4 + 4\lambda^3 + 3\lambda^2 + 2\lambda + 1$

   (b) What is the characteristic polynomial of the companion matrix for $p(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + \ldots + c_1\lambda + c_0$?

7. (a) Generate two random 3×3 matrices A and B over Z_{10} and calculate the characteristic polynomials of AB and BA. What do you observe?

   (b) Repeat (a) for random 4×4 and 5×5 matrices over Z_{10}.

   (c) Repeat (a) – (b) for random 3×3, 4×4 and 5×5 *complex* matrices over Z_{10}.

   (d) Formulate a conjecture based upon your observations. Discuss your conjecture and its implications with your instructor.

8. (a) Generate a random 3×3 matrix A over Z_{10} and put two copies of A on the stack.

   (b) Find the characteristic polynomial $p(x)$ of A and use program P.of.A to evaluate $p(A)$.

(c) Repeat (a) – (b) using random 4×4 and 5×5 matrices over Z_{10}.

(d) Repeat (a) – (c) using random 3×3, 4×4 and 5×5 *complex* matrices over Z_{10}.

(e) Formulate a conjecture based upon your observations. Discuss your conjecture with your instructor.

6.2 EIGENVALUE CALCULATIONS

Although low order matrices having integer entries are not typical of the matrices encountered in scientific and engineering applications, they serve us well in the learning process. But even with such matrices, finding the eigenvalues by hand as the roots of the characteristic polynomial is a difficult, if not impossible, task unless the matrices are highly contrived. To avoid such contrivance, we may use the polynomial root-finding routine PROOT on the HP-48G/GX calculators. PROOT will find all roots of an arbitrary real or complex polynomial $a_n x^n + a_{n-1}x^{n-1} + \ldots a_1 x + a_0$. It requires as input the vector [a_n $a_{n-1} \ldots a_1$ a_0] of coefficients.

EXAMPLE 2. Put three copies of the following matrix on the stack:

$$A = \begin{bmatrix} 1 & -3 & 1 & -1 \\ -6 & -2 & 5 & 1 \\ -3 & -3 & 4 & 0 \\ -3 & -3 & 1 & 3 \end{bmatrix}.$$

CHAR returns [1 -6 3 26 -24], so the characteristic polynomial is $\lambda^4 - 6\lambda^3 + 3\lambda^2 + 26\lambda - 24$. The command PROOT returns the vector [1 3 -2 4], showing that A has four distinct eigenvalues: λ = -2, 1, 3 and 4. To find the eigenspace determined by $\lambda = 3$ we proceed as follows. With A on level 2 and [1 3 -2 4] on level 1 extract 3 from the vector with the command 2 GET and build $A - 3I$ with the

commands 4 IDN [×] [−]. To obtain a basis for the nullspace of $A - 3I$, apply the RREF command. The result

```
[[ 1   0   0  -1 ]
 [ 0   1   0   1 ]
 [ 0   0   1   0 ]
 [ 0   0   0   0 ]]
```

shows that x_4 is a free variable and that the eigenvectors associated with $\lambda = 3$ are scalar multiples of x = [1 -1 0 1]. DROP the RREF matrix and enter [1 -1 0 1]. Since A still appears on level 2, a simple multiplication will confirm that $Ax = 3x$.

EXAMPLE 3. Put two copies of the following matrix on the stack:

$$A = \begin{bmatrix} 7 & 2 & 4 & 6 \\ -6 & -1 & -4 & -4 \\ 4 & 4 & 5 & -2 \\ -16 & -12 & -14 & -3 \end{bmatrix}.$$

CHAR returns [1 -8 22 -40 25], so the characteristic polynomial is $p(\lambda) = \lambda^4 - 8\lambda^3 + 22\lambda^2 - 40\lambda + 25$. PROOT returns the vector [(1, 0) (1, 2) (1, -2) (5, 0)] containing the roots. Thus the eigenvalues of A are $\lambda = 1, 5$ and $1 \pm 2i$. To find the eigenspace associated with $\lambda = 1–2i$ we proceed as follows. With A on level 2, extract (1, -2) from the vector on level 1 with the command 3 GET and build A -(1, -2)I with the commands 4 IDN [×] [−]. Apply RREF. Use [↰] EDIT to view the last column. Clean up round off error with 11 RND and see that [-1+i 1 -i 1] spans the eigenspace.

Cofactor expansions tell us that the characteristic polynomial of a matrix A having only integer entries will have only integer coefficients. Since CHAR uses traces of powers of A, it will accurately return the coefficients of the characteristic polynomial of any such A. But finding eigenvalues as the roots of the characteristic

polynomial is seldom done in computational practice because even sophisticated polynomial root finding routines are often limited in their ability to obtain multiple roots with a high degree of accuracy. For example, the roots of $x^4 - 8\lambda^3 + 10\lambda^2 + 48\lambda - 99$ are $\lambda = 3,3$ and $1 \pm 2\sqrt{3}$. Although PROOT returns decimal approximations to $1 \pm 2\sqrt{3}$ that are accurate to twelve places, it returns 2.99999907027 and 3.00000092973 for the other two values.

The numerical computation of eigenvalues is much more complicated than the numerical solution of linear systems and any discussion of the appropriate procedures is well beyond the scope of this brief chapter. But the HP-48G and 48GX calculators include code for finding eigenvalues and eigenvectors that is based upon advanced numerical techniques that use the *Schur factorization* of a matrix. (You can obtain a 12-digit version of the Schur factorization via the command SCHUR.)

EXAMPLE 4. We use the following matrix A

```
[ [ -14  -16  -26   -9 ]
  [  16   19   28   12 ]
  [  -7   -8  -11   -7 ]
  [  13   14   24   14 ] ]
```

Make another copy with ENTER. CHAR returns [1 -8 10 48 -99], and we have seen that PROOT returns only two of the four eigenvalues with 12-digit accuracy. Drop this vector. Then with A on level 1, the command EGVL (on the MTH MATR menu) returns the vector [4.46410161514 -2.46410161514 3 3] of eigenvalues accurate to 12-digits.

Activity Set 6.2

1. Adding a multiple of a row to another row will not change the determinant of a square matrix A. Will this change the eigenvalues? The characteristic

polynomial? Use your calculator to investigate these questions by experimenting with random 3×3 and 4×4 matrices over Z_{10}.

2. Consider the following matrix

$$A = \begin{bmatrix} 4 & -8 & 2 & 5 \\ 0 & 1 & -6 & -2 \\ -9 & 0 & 7 & 1 \\ 7 & 3 & -8 & 9 \end{bmatrix}$$

(a) Find the characteristic polynomial of A, then use PROOT to obtain the vector of eigenvalues as the roots of this polynomial. Go to the PRG TYPE menu and use the OBJ→ command to separate the vector into its component eigenvalues (use DROP to remove the list { 4 } that indicates the size of this vector).

(b) With the roots of the characteristic polynomial still on the stack, use the EGVL command to obtain the vector of eigenvalues of matrix A. As in part (a), separate this vector into its component entries.

(c) Use the interactive stack (e.g., the [Δ] and the ROLLD command) and the command SAME (on the PRG TEST menu) to compare the accuracy of the eigenvalues obtained by the two methods.

3. (a) Calculate and store a random 3×3 matrix A over Z_{10}. Calculate $detA$ and $traceA$.

(b) Compare $traceA$ with the sum of the eigenvalues of A; what do you observe?

(c) Compare $detA$ with the product of the eigenvalues of A; what do you observe?

(d) Repeat the above for random 4×4 and 5×5 matrices over Z_{10}.

(e) Formulate your observations as a conjecture.

(f) Discuss your conjecture with your instructor.

4. For each of the following matrices, find:

(a) all eigenvalues

(b) for the eigenvalue λ of maximum absolute value, the associated eigenspace.

$$A = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 2 & 1 & 4 \\ 2 & 1 & 4 & 3 \\ 1 & 4 & 3 & 2 \end{bmatrix}$$

$$B = \begin{bmatrix} -2 & -8 & -1 & 6 & 5 \\ 1 & 7 & 1 & -2 & -2 \\ -11 & -16 & 0 & 10 & 5 \\ -7 & -8 & -1 & 10 & 4 \\ 7 & 8 & 1 & -6 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 8 & 6 & 8 & 5 & -3 \\ -9 & -8 & -10 & -9 & 1 \\ -3 & -2 & -3 & -2 & 1 \\ 11 & 11 & 12 & 12 & 1 \\ -8 & -9 & -8 & -8 & -1 \end{bmatrix}$$

$$D = \begin{bmatrix} 54 & 22 & -4 & -2 & -12 & -6 \\ -91 & -34 & 9 & 2 & 22 & 8 \\ 170 & 70 & -12 & -7 & -37 & -20 \\ 92 & 47 & 0 & -8 & -15 & -18 \\ -27 & -7 & 5 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

6.3 SIMILARITY

Given that we earlier made a case for having *independent* sets of vectors - sets in which no one vector depends linearly upon the others – we now ask "independence questions" about the eigenvectors of A. In particular, how large can a set of independent eigenvectors of A be? Certainly no larger than n, because the eigenvectors lie in R^n and any set of more than n vectors in R^n is dependent. And the

case where A has a set of n independent eigenvectors, say $\{ x_1, x_2, \ldots, x_n \}$, is especially nice. For then we have

$$P^{-1}AP = D = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix},$$

where λ_i is the eigenvalue associated with x_i and P is the matrix having $x_1, x_2, \ldots, x_n$ as its columns:

$$P = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix}.$$

In fact, the equation $P^{-1}AP = D$ is equivalent to saying that A has n independent eigenvectors. This equation is just a rearrangement of $AP = PD$ which, when read column-by-column, simply says $Ax_j = \lambda_j x_j$. The set of x_j's is independent because the x_j's are the columns of the invertible matrix P. We are thus led to focus on the case where the $n \times n$ matrix A has n independent eigenvectors as the desirable one, and we call any $n \times n$ matrix A having fewer than n independent eigenvectors a *defective* matrix.

For a non-defective matrix A the equation $P^{-1}AP = D$ has important implications. In general, we call matrices A and B *similar* provided that $P^{-1}AP = B$ for some invertible matrix P. The term "similar" probably derives from the elementary result that *similar matrices have the same characteristic polynomial, the same eigenvalues, determinant, trace and rank.* When A is similar to a diagonal matrix D we say that A is a **diagonalizable** matrix, and the above discussion may be summarized as follows:

A is diagonalizable iff A has n independent eigenvectors.

Of fundamental help in deciding whether A is diagonalizable is the result that

eigenvectors associated with distinct eigenvalues are independent.

Consequently, if A has n distinct eigenvalues, then A has n independent eigenvectors, one associated with each eigenvalue, so A is diagonalizable. But it is also possible for A to be diagonalizable even when it has fewer than n distinct eigenvalues. There are two keys to understanding how this may happen:

(1) For any eigenvalue λ of A, the dimension of the eigenspace determined by λ, *dim* $NS(A - \lambda I)$, does not exceed the multiplicity of λ as a root of the characteristic polynomial;

(2) If $\lambda_1, \lambda_2, ..., \lambda_k$ are the *distinct* eigenvalues of A and $B_1, B_2, ..., B_k$ are bases for the associated eigenspaces then the union of these bases is an independent set of eigenvectors of A.

Think about the characteristic polynomial of A in factored form:

$$det(\lambda I - A) = (\lambda - \lambda_1)^{m_1}(\lambda - \lambda_2)^{m_2} \ldots (\lambda - \lambda_k)^{m_k}$$

where $\lambda_1, ..., \lambda_k$ are the *distinct* eigenvalues and $m_1, ..., m_k$ are their respective multiplicities. Since $det(\lambda I - A)$ is a polynomial of degree n, we have $n = m_1 + m_2 + ... + m_k$. According to (1), we have *dim* $NS(A - \lambda_j I) \leq m_j$, for each $j = 1, ..., k$. Thus, in the case where equality holds for every j, the bases in (2) will contain exactly m_j vectors and their union will produce n independent eigenvectors for A. But in the case that we have *dim* $NS(A - \lambda_j I) < m_j$ for even *one* j, the union of the bases in (2) will fail to produce n independent eigenvectors and A will be a defective matrix.

A is defective iff for some eigenvalue λ there are not enough independent eigenvectors associated with λ.

EXAMPLE 5. Consider the following matrix:

$$A = \begin{matrix}[[\ -9 & 2 & -6 & 0\ ] \\ [\ 5 & 1 & 5 & 7\ ] \\ [\ 6 & 0 & 3 & 1\ ] \\ [\ 3 & -2 & 3 & -1\ ]]\end{matrix}.$$

Program CHAR returns [1 6 9 0 0]. Thus the characteristic polynomial of A is $\lambda^4 + 6\lambda^3 + 9\lambda^2 = \lambda^2(\lambda^2 + 6\lambda + 9) = \lambda^2(\lambda + 3)^2$ and the distinct eignevalues are 0 and -3, each having multiplicity 2. A quick application of the RREF command to $A - 0I$ and to $A + 3I$ shows that $NS(A - 0I)$ and $NS(A + 3I)$ each have dimension 1, so A is a (*doubly*) defective matrix.

EXAMPLE 6. Consider the following matrix

$$A = \begin{matrix}[[\ 2 & .5 & -1 & -2 & 3\ ] \\ [\ 0 & 1 & 2 & 4 & -6\ ] \\ [\ 0 & 0 & 2 & 1 & -2\ ] \\ [\ -4 & -2 & 2 & 3 & 2\ ] \\ [\ -4 & -2 & 2 & 1 & 4\ ]]\end{matrix}.$$

The EGVL command shows the eigenvalues to be $\lambda = 1, 2, 2, 3, 4$. Since $\lambda = 2$ is the only repeated root, to settle the question whether A is diagonalizable or defective we must determine *dim* $NS[A - 2I]$. RREF shows two free variables, so dim $NS[A - 2I] = 2$, the multiplicity of 2 as a root of the characteristic polynomial. Thus A is diagonalizable. In fact, a basis for the eigenspace associated with $\lambda = 2$ consists of the vectors [-1 4 2 0 0] and [0 2 0 2 1]. The eigenspaces associated with $\lambda = 4$, 3 and 1 have [1 -2 -1 2 2], [1 -2 -1 1 1] and [-1 2 0 0 0] as bases, respectively. Using these basis vectors as the columns of matrix

$$P = \begin{bmatrix} 1 & 1 & -1 & 0 & -1 \\ -2 & -2 & 4 & 2 & 2 \\ -1 & -1 & 2 & 0 & 0 \\ 2 & 1 & 0 & 2 & 0 \\ 2 & 1 & 0 & 1 & 0 \end{bmatrix},$$

you can verify that

$$P^{-1}AP = D = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

It is important that students understand how to construct a diagonalizing matrix P for a diagonalizable matrix A by finding bases for the different eigenspaces of A. But you should also note that the HP-48G/GX calculators will produce such a P with a single keystroke. With a square matrix A on level 1, the command **EGV** (on the **MTH MATR** menu) will return to level 1 a vector containing the eigenvalues, and to level 2 a matrix P whose columns are corresponding eigenvectors. In case A is diagonalizable, the columns of P are independent so that $P^{-1}AP$ is diagonal. You should try this with the matrix A of our last example. **EGV** returns [4 3 2 2 1] as the vector of eigenvalues and you will notice that columns 1, 2, 3, and 5 of the matrix P that is returned to level 2 are rescaled versions of the vectors we constructed in the example. If you extract columns 3 and 4 and assemble them as the columns in a new matrix, the **RREF** command will show them to be independent.

Activity Set 6.3

1. (a) Find matrix $B = P^{-1}AP$ given the following matrices P and A:

$$A = \begin{bmatrix} 7 & 2 & 4 & 6 \\ -6 & -1 & -4 & -4 \\ 4 & 4 & 5 & -2 \\ -16 & -12 & -14 & -3 \end{bmatrix} \qquad P = \begin{bmatrix} 0 & -1 & -1 & -1 \\ 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

   (b) Use program CHAR to verify that A and B have the characteristic polynomial.

   (c) Verify that A and B have the same eigenvalues, trace, determinant, and rank.

2. Determine whether the following matrices A are defective or diagonalizable. For each one that is diagonalizable, find an invertible P and a diagonal D for which $P^{-1}AP = D$.

(a)
$$\begin{bmatrix} 0 & 1 & -2 & 0 \\ -2 & 3 & -4 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & -1 & 0 & -1 \end{bmatrix}$$

(b)
$$\begin{bmatrix} 10 & -5 & 12 & 0 \\ 9 & -4 & 12 & 0 \\ -5 & 3 & -5 & 0 \\ -7 & 2 & -9 & -2 \end{bmatrix}$$

(c)
$$\begin{bmatrix} 6 & -2 & -4 & -2 \\ -2 & 3 & 2 & 1 \\ 2 & 1 & 0 & -3 \\ -2 & -1 & 2 & 5 \end{bmatrix}$$

(d)
$$\begin{bmatrix} 8 & -3 & 8 & 0 & 5 \\ 9 & -4 & 12 & 0 & 9 \\ -5 & 3 & -5 & 0 & -5 \\ -7 & 2 & -9 & 0 & -5 \\ 2 & -2 & 4 & 0 & 5 \end{bmatrix}$$

(e)
$$\begin{bmatrix} 1 & 0 & 3 & 0 & -3 & 3 \\ -1 & 2 & 2 & -1 & -2 & 3 \\ 0 & 0 & 0 & -2 & 2 & 2 \\ -2 & 2 & 0 & 4 & 0 & 0 \\ -2 & 2 & -3 & -1 & 5 & 2 \\ 0 & 0 & -1 & -1 & 1 & 4 \end{bmatrix}$$

6.4 REAL SYMMETRIC MATRICES

The most important matrices for the physical sciences are the real symmetric matrices, real matrices A satisfying $A^T = A$. Such matrices play a significant role in a number of applications.

We have seen that for a general matrix, repeated eigenvalues may lead to a defective matrix. But this never happens with a real symmetric matrix:

every real symmetric matrix is diagonalizable.

This is because for each eigenvalue λ there are "enough" independent eigenvectors. The important features that surround any real symmetric matrix A of order n are as follows:

(i) The eigenvalues of A are real numbers (no complex eigenvalues occur).

(ii) The eigenspaces of A are orthogonal subspaces (eigenvectors associated with distinct eigenvalues are orthogonal).

(iii) If an eigenvalue λ of A is a root of multiplicity k of the characteristic polynomial then the associated eigenspace has dimension k.

(iv) A has n orthogonal (hence independent) eigenvectors.

If n orthonormal eigenvectors of A are used as the columns of an orthogonal matrix Q, then Q is an orthogonal diagonalizing matrix for A:

$$Q^{-1}AQ = D$$

where D is the diagonal matrix of eigenvalues of A, the eigenvalues corresponding in order to the eigenvector columns of matrix Q.

The steps to be followed in constructing an orthogonal diagonalizing matrix Q for a real symmetric matrix A should be clear:

Step 1: Find the eigenvalues of A.

Step 2: For each eigenspace, construct an orthonormal basis (perhaps by using the Gram-Schmidt process).

Step 3: The union of the orthonormal bases constructed in step 2 will be an orthonormal basis for R^n; use these basis vectors as the columns of Q.

EXAMPLE 7. Consider the real symmetric matrix

$$A = \begin{bmatrix} 2 & -1 & 0 & 1 \\ -1 & 2 & 0 & -1 \\ 0 & 0 & 2 & 0 \\ 1 & -1 & 0 & 2 \end{bmatrix}.$$

The EGVL command shows that the eigenvalues of A are $\lambda = 1, 1, 2, 4$. Applying the RREF command to both $(A - 2I)$ and to $(A - 4I)$ we obtain [0 0 1 0] and [1 -1 0 1] as bases for the associated eigenspaces, respectively. Notice that these two eigenvectors, which are associated with different eigenvalues, are orthogonal. Normalize [1 -1 0 1] to get [.577350269189 -.577350269189 0 .577350269189]. Now apply RREF to $(A - I)$ to get the basis { [1 1 0 0] , [-1 0 0 1] } for the third eigenspace. Since these vectors are not orthogonal we apply the Gram-Schmidt process to get the orthonormal basis vectors [.707106781188 .707106781188 0 0] and [-.408248290463 .408248290466 0 .816496580929]. Now put these vectors as the columns of a matrix

$$Q = \begin{bmatrix} 0 & .577350269189 & .707106781188 & -.408248290463 \\ 0 & -.577350269189 & .707106781188 & .408248290466 \\ 1 & 0 & 0 & 0 \\ 0 & .577350269189 & 0 & .816496580929 \end{bmatrix}.$$

Since Q is orthogonal, $Q^{-1} = Q^T$ and a quick check will verify that

$$Q^TAQ = Diag\,[\,2\ \ 4\ \ 1\ \ 1\,].$$

As in this example, $Q^{-1} = Q^T$ changes the similarity equation $Q^{-1}AQ = D$ to $Q^TAQ = D$. More important is when we solve for A:

$$A = QDQ^T.$$

If matrix Q has the orthonormal column vectors $q_1, q_2, \ldots, q_n$ as its columns

$$Q = [\,q_1\ \ q_2\ \ \cdots\ \ q_n\,]$$

then we have

$$A = QDQ^T = [\,q_1 \cdots\ \ q_n\,]\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}\begin{bmatrix} q_1^T \\ \vdots \\ q_n^T \end{bmatrix}$$

$$= \lambda_1 q_1 q_1^T + \lambda_2 q_2 q_2^T + \ldots \lambda_n q_n q_n^T .$$

This is a *spectral decomposition* of matrix A, so-called because the set $\{\,\lambda_1, \lambda_2, \ldots, \lambda_n\,\}$ of eigenvalues is sometimes called the *spectrum* of A. Each $\lambda_j q_j q_j^T$ in the decomposition is an n×n matrix of rank 1 (each column being a multiple of q_j). Notice that any particular spectral decomposition of A depends upon the choice of the orthonormal vectors $q_1, \ldots, q_n$.

EXAMPLE 8. To compute a spectral decomposition of the matrix A in **EXAMPLE 7** using the particular Q matrix above, we have:

$$2q_1q_1^T + 4q_2q_2^T + 1q_3q_3^T + 1q_4q_4^T$$

$$= 2\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 4\begin{bmatrix} .\overline{3} & -.\overline{3} & 0 & .\overline{3} \\ -.\overline{3} & .\overline{3} & 0 & -.\overline{3} \\ 0 & 0 & 0 & 0 \\ .\overline{3} & -.\overline{3} & 0 & -.\overline{3} \end{bmatrix} + 1\begin{bmatrix} .5 & .5 & 0 & 0 \\ .5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$+ 1\begin{bmatrix} .1\overline{6} & -.1\overline{6} & 0 & -.\overline{3} \\ -.1\overline{6} & .1\overline{6} & 0 & .\overline{3} \\ 0 & 0 & 0 & 0 \\ -.\overline{3} & .\overline{3} & 0 & .\overline{6} \end{bmatrix} = \begin{bmatrix} 2 & -1 & 0 & 1 \\ -1 & 2 & 0 & -1 \\ 0 & 0 & 2 & 0 \\ 1 & -1 & 0 & 2 \end{bmatrix} \text{ (after 10 RND).}$$

Activity Set 6.4

1. Find an orthogonal matrix Q and a diagonal matrix D such that $Q^{-1}AQ = D$.

(a)
$$\begin{bmatrix} 4 & -1 & -1 & -1 \\ -1 & 4 & -1 & -1 \\ -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & 4 \end{bmatrix}$$

(b)
$$\begin{bmatrix} 3 & -1 & 1 & 0 \\ -1 & 3 & 1 & 0 \\ 1 & 1 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

(c)
$$\begin{bmatrix} 1 & -2 & -3 & 2 \\ -2 & 1 & -2 & 3 \\ -3 & -2 & 1 & 2 \\ 2 & 3 & 2 & 1 \end{bmatrix}$$

(d)
$$\begin{bmatrix} 2 & 2 & -2 & 2 & 2 \\ 2 & 1 & -3 & -1 & -1 \\ -2 & -3 & 1 & 1 & 1 \\ 2 & -1 & 1 & 5 & -1 \\ 2 & -1 & 1 & -1 & 5 \end{bmatrix}$$

2. Find a spectral decomposition for each of the matrices in Activity 1(a) - 1(b).

6.5 POSITIVE DEFINITE MATRICES

A real symmetric matrix A of order n is called *positive definite* provided that

$$x^TAx > 0 \text{ for each non-zero } x \text{ in } R^n.$$

In the expression x^TAx, we are viewing vector x in R^n as a column matrix; thus x^TAx is a 1×1 matrix, a single real number.

Positive definite matrices are the most important of the real symmetric matrices and appear often in applications such as electrical circuit analysis, elastic stress studies, and least squares problems.

It is easy to see that any positive definite matrix A is nonsingular (for if $Ax = 0$ had a non-zero solution x then $x^TAx = 0$, contrary to $x^TAx > 0$). More importantly, any such matrix A has only positive eigenvalues (if $Ax = \lambda x$ for some non-zero x then $x^TAx = x^T\lambda x = \lambda \| x \|^2$, so that $\lambda = x^TAx / \| x \|^2 > 0$). The converse of this last result is also true, and provides us with the first of a number of different criteria for positive definiteness. These are summarized in the following theorem.

Theorem. Given a real symmetric matrix A of order n, the following are equivalent conditions:

(i) A is positive definite ($x^TAx > 0$ for all non-zero x in R^n).

(ii) All eigenvalues of A are positive.

(iii) All upper left sub-determinants of A are positive.

(iv) A can be converted to an upper triangular matrix without row interchanges and all pivots will be positive.

(v) A can be factored as $A = LL^T$, where L is a lower triangular matrix having positive diagonal entries.

(vi) A can be factored as $A = M^TM$ for some nonsingular matrix M.

These results are impressive because they embrace such a wide range of basic concepts in linear algebra: Gaussian elimination, pivots, eigenvalues, matrix factorizations, and determinants. In keeping with the spirit of this book we will not provide a proof of this theorem. That should be done by your instructor or your

linear algebra textbook. But it is appropriate to comment briefly upon several of these equivalent conditions.

Condition (iii) connects determinants to pivots. It asserts that if A_k denotes the upper left $k \times k$ submatrix then $det\ A_k > 0$ for all $k = 1, 2, \ldots, n$. For $k = 1$ this simply says that pivot $a_{11} > 0$. Then

$$A_2 = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \longrightarrow \begin{bmatrix} a_{11} & * \\ 0 & a_{22}' \end{bmatrix}$$

and pivot $a_{22}' > 0$ because $det\ A_2 = a_{11}a_{22}' > 0$ and $a_{11} > 0$.

Then we have

$$A_3 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \longrightarrow \begin{bmatrix} a_{11} & * & * \\ 0 & a_{22}' & * \\ 0 & * & * \end{bmatrix} \longrightarrow \begin{bmatrix} a_{11} & * & * \\ 0 & a_{22}' & * \\ 0 & 0 & a_{33}' \end{bmatrix}$$

and pivot $a_{33}' > 0$ because $det\ A_3 = a_{11}a_{22}'a_{33}' > 0$ and both $a_{11}, a_{22}' > 0$. Notice that with positive pivots we never have to interchange rows. In this way, condition (iv) is true.

Condition (v) comes from the LU-factorization of A. Here's how. Although LU-factorizations are not, in general, unique, there is a related factorization for certain invertible matrices that is unique. Suppose that an invertible matrix A can be brought to upper triangular form U without row interchanges. Then $A = L_1U$ where L_1 is lower triangular with 1's on its diagonal and U is upper triangular with non-zero diagonal entries $u_{11}, u_{22}, \ldots u_{nn}$. If D is the diagonal matrix $D = diag\ [\ u_{11}\ u_{22} \ldots u_{nn}\ ]$ then $D^{-1} = diag\left[u_{11}^{-1}\ u_{22}^{-1} \ldots u_{nn}^{-1}\right]$ and $A = L_1DD^{-1}U = L_1DU_1$, where the upper triangular matrix $U_1 = D^{-1}U$ also has 1's on its diagonal. This is the

LDU-factorization of A and it is easy to see that it is unique. If matrix A is also symmetric then, in addition to $A = L_1DU_1$, we also have $A = A^T = (L_1DU_1)^T = U_1^TD^TL_1^T = U_1^T D L_1^T$ and the uniqueness tells us that $L_1^T = U_1$. Thus, the LDU-factorization in this case has the form $A = L_1DL_1^T$. Finally, suppose that A is positive definite. Then D has only positive diagonal entries and can be split into the product of two "square root" matrices:

$$D = \begin{bmatrix} u_{11} & & \\ & \ddots & \\ & & u_{nn} \end{bmatrix} = \begin{bmatrix} \sqrt{u_{11}} & & \\ & \ddots & \\ & & \sqrt{u_{nn}} \end{bmatrix} \begin{bmatrix} \sqrt{u_{11}} & & \\ & \ddots & \\ & & \sqrt{u_{nn}} \end{bmatrix}$$

$$= \sqrt{D}\,\sqrt{D}\,.$$

Then $A = L_1DL_1^T = (L_1\sqrt{D}\,)\,(L_1\sqrt{D}\,)^T = LL^T$, where $L = L_1\sqrt{D}$. This factorization can be shown to be unique and is called the *Cholesky factorization of A*.

EXAMPLE 8. Construct the Cholesky factorization of

$$A = \begin{bmatrix} 2 & 4 & -4 \\ 4 & 12 & -20 \\ -4 & -20 & 50 \end{bmatrix}.$$

Elementary calculations show that A has the following LU-factorization without row interchanges:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -2 & -3 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -4 \\ 0 & 4 & -12 \\ 0 & 0 & 6 \end{bmatrix}.$$

Factoring out the diagonal entries from U we see that A has the following LDL^T factorization:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -2 & -3 & 1 \end{bmatrix} \begin{bmatrix} 2 & & \\ & 4 & \\ & & 6 \end{bmatrix} \begin{bmatrix} 1 & 2 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}.$$

Finally, accounting for square roots we see the Cholesky factorization $A = LL^T$:

$$A = \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 2\sqrt{2} & 2 & 0 \\ -2\sqrt{2} & -6 & \sqrt{6} \end{bmatrix} \begin{bmatrix} \sqrt{2} & 2\sqrt{2} & -2\sqrt{2} \\ 0 & 2 & -6 \\ 0 & 0 & \sqrt{6} \end{bmatrix} = LL^T.$$

Of all the criteria in the Theorem for determining positive definiteness, the Cholesky factorization is the one that is most practical for numerical computations. For we should avoid the computation of eigenvalues and determinants whenever possible. There is a fairly straightforward algorithm for constructing the Cholesky factorization. Most importantly, if the algorithm is applied to a real symmetric matrix A that is *not* positive definite then the algorithm *will fail*: at some point it will attempt to take the square root of a number that is not positive. Because of this, the algorithm can be applied to any real symmetric matrix as a test for positive definiteness. If the algorithm does not fail, it succeeds in calculating the Cholesky factor L in $A = LL^T$. The following code implements the algorithm.

P.DEF (Test for Positive Definiteness)

Input: level 1: a real symmetric matrix

Effect: If the input matrix is positive definite, the Cholesky factor L is returned to level 1 and the input matrix to level 2. If not positive definite, the message "NOT POS DEFINITE" appears.

```
« DUP → A « A SIZE 1 GET → n « 1 n FOR j IF j 1 ≠ THEN 1 j
1 – FOR k 'A(j, j)' EVAL 'A(j, k)' EVAL SQ – 'A(j, j)' STO NEXT END
IF 'A(j, j)' EVAL 0 ≤ THEN "NOT POS DEFINITE" KILL END 'A(j, j)'
EVAL √ 'A(j, j)' STO IF j n ≠ THEN j 1 + n FOR i IF j 1 ≠ THEN 1
j 1 – FOR k 'A(i, j)' EVAL 'A(i, k)' EVAL 'A(j, k)' EVAL * – 'A(i, j)'
STO NEXT END 'A(i, j)' EVAL 'A(j, j)' EVAL / 'A(i, j)' STO NEXT END
1 n FOR i 1 n FOR j IF j i > THEN 0 'A(i, j)' STO END NEXT NEXT
» A » »
```

Given the Cholesky factorization $A = LL^T$ of a positive definite matrix A, we can use the two factors to solve the linear system $Ax = b$ in two easy steps:

(i) solve $Ly = b$ for y by forward substitution

(ii) then solve $L^Tx = y$ for x by back substitution.

EXAMPLE 9. Use program P.DEF to check the following matrices A for positive definiteness; where possible, use the Cholesky factorization to solve $Ax = [\ 1\ \ 1\ \ 1\ ]$.

(a) $A = \begin{bmatrix} 6 & -1 & -4 \\ -1 & 0 & -6 \\ -4 & -6 & 0 \end{bmatrix}$

(b) $A = \begin{bmatrix} 2 & 4 & -4 \\ 4 & 12 & -20 \\ -4 & -20 & 50 \end{bmatrix}$

With the matrix A in (a) on level 1, executing P.DEF returns the message "NOT POS DEFINITE". The matrix A in (b) is the matrix of EXAMPLE 8, so is already known to be positive definite. With this matrix on level 1, P.DEF returns the input matrix A to level 2 and the following Cholesky factor L in $A = LL^T$:

$$L = \begin{bmatrix} 1.41421356237 & 0 & 0 \\ 2.8284712475 & 1.99999999999 & 0 \\ -2.8284712475 & -6.00000000003 & 2.4494897427 \end{bmatrix}.$$

To check this last result, press the ENTER key to duplicate L, transpose with TRN, multiply and then clean up round-off error with 10 RND.

To solve $Ax = b$ using the Cholesky factors L and L^T we first solve $Ly = [\ 1\ 1\ 1\ ]$ for y using forward substitution (program FWD) to obtain [.707106781188 -.5 0] (after cleaning up some obvious round-off error). Then we use back substitution (program BACK) to solve $L^Tx = y$ for $x = [\ 1\ -.25\ 0\ ]$.

Activity Set 6.5

1. Which of the following symmetric matrices are positive definite?

(a) $A = \begin{bmatrix} 3 & 3 & 4 \\ 3 & -3 & 7 \\ 4 & 7 & 2 \end{bmatrix}$

(b) $A = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & 1 \\ -1 & 1 & 2 \end{bmatrix}$

(c) $A = \begin{bmatrix} -5 & -7 & -2 & 5 \\ -7 & 9 & -4 & -5 \\ -2 & -4 & 5 & 8 \\ 5 & -5 & 8 & -8 \end{bmatrix}$

(d) $A = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 1 \\ -1 & -1 & 2 & -1 \\ -1 & 1 & -1 & 2 \end{bmatrix}$

$$\text{(e)}\quad A = \begin{bmatrix} 5 & 8 & 2 & -9 & -9 \\ 8 & -8 & -7 & -1 & 4 \\ 2 & -7 & 0 & -8 & -2 \\ -9 & -1 & -8 & 9 & 3 \\ -3 & 4 & -2 & 3 & 7 \end{bmatrix} \qquad \text{(f)}\quad A = \begin{bmatrix} 4 & -10 & -4 & 3 & 3 \\ -10 & 32 & 12 & -2 & -4 \\ -4 & 12 & 6 & -3 & -3 \\ 3 & -2 & -3 & 9 & 7 \\ 3 & -4 & -3 & 7 & 6 \end{bmatrix}$$

2. For any of y the matrices A in Activity 1 that are positive definite, use the Cholesky factors in $A = LL^T$ to solve the linear system $Ax = b$, where $b = [\,1 \;\; 2 \;\; 3\,]$, $b = [\,1 \;\; 2 \;\; 3 \;\; 4\,]$, or $b = [\,1 \;\; 2 \;\; 3 \;\; 4 \;\; 5\,]$ as appropriate.

3. Test the following symmetric matrix A for positive definiteness. If positive definite, use the Cholesky factors in $A = LL^T$ to solve the linear system $Ax = b$.

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \qquad b = [\,1 \;\; 1 \;\; 1 \;\; 1 \;\; 1\,]$$

6.6 SINGULAR VALUE DECOMPOSITIONS

Now that we have some experience with matrix factorizations like $A = LU$, $A = QR$, $A = LL^T$ $A = PDP^{-1}$, and $A = QDQ^{-1}$ we turn finally to *singular value decompositions*, factorizations like

(1) $A = U\Sigma V^T$.

As we might expect from the case $A = PDP^{-1}$, the middle term Σ in (1) will still be a diagonal matrix: all off-diagonal entries are zero. And just as in the more elegant factorization $A = QDQ^T$, both of the outside matrices U and V will be orthogonal. What is different about the singular value decomposition (1) is that because U and V are different orthogonal matrices, Σ will no longer contain the

eigenvalues of A; instead it will contain the *singular values* of A, non-negative numbers $\sigma_1 \geq \sigma_2 \geq \ldots$. In fact, A itself need not be a square matrix. We can obtain singular value decompositions $A = U\Sigma V^T$ for an arbitrary rectangular matrix A.

Without concerning ourselves with the details of the construction of singular value decompositions (they can be considerable), what can we find out about the factors U, Σ, and V from the factorization itself? Since U and V are orthogonal, we expect that symmetric matrices are somehow involved. Indeed they are: both A^TA and AA^T are symmetric.

We begin with A^TA. From $A = U\Sigma V^T$ we find that

$$A^TA = (V\Sigma^TU^T)(U\Sigma V^T)$$

$$= V(\Sigma^T\Sigma)V^T. \tag{2}$$

Since Σ has entries σ_1, σ_2 , ... along the main diagonal, $\Sigma^T\Sigma$ will have diagonal entries σ_1^2, σ_2^2, ... Thus, equation (2) is an orthogonal diagonalization of the real symmetric matrix A^TA, so the σ_1^2, σ_2^2, ... *are the eigenvalues of* A^TA and *the column vectors of* V *are orthonormal eigenvectors of* A^TA.

Similarly, we find that

$$AA^T = (U\Sigma V^T)(V\Sigma^TU^T)$$

$$= U(\Sigma\Sigma^T)U^T. \tag{3}$$

Thus, the σ_1^2, σ_2^2, ... *are also the eigenvalues of* AA^T and *the column vectors of* U *are orthonormal eigenvectors of* AA^T.

If A is a 5×2 matrix, then A^TA is 2×2 while AA^T is 5×5. The difference in their eigenvalues is this: they have the same non-zero eigenvalues (including multiplicities), but any eigenvalue 0 will have different multiplicities for the two matrices. For example, consider the matrix

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \\ 9 & 0 \end{bmatrix} .$$

Then A^TA is the 2×2 matrix

$$A^TA = \begin{bmatrix} 165 & 100 \\ 100 & 120 \end{bmatrix}$$

whose eigenvalues are $\lambda = 245, 40$. But AA^T is the 5×5 matrix

$$AA^T = \begin{bmatrix} 5 & 11 & 17 & 23 & 9 \\ 11 & 25 & 39 & 53 & 27 \\ 17 & 39 & 61 & 83 & 45 \\ 23 & 53 & 83 & 113 & 63 \\ 9 & 27 & 45 & 63 & 81 \end{bmatrix}$$

whose eigenvalues are $\lambda = 245, 40, 0, 0, 0$.

In the general case, consider any eigenvalue λ of A^TA, say $(A^TA)x = \lambda x$ for some non-zero x. Then λ is a real number and we have

$$\begin{aligned} \|Ax\|^2 = Ax \bullet Ax &= (Ax)^T(Ax) \\ &= x^T(A^TAx) \\ &= x^T\lambda x \\ &= \lambda \| x \|^2 \end{aligned}$$

so that $\| x \| \neq 0$ implies $\lambda \geq 0$. Then, since the non-zero entries $\sigma_1^2, \sigma_2^2, \ldots$ of $\Sigma^T\Sigma$ are just the non-zero eigenvalues of A^TA, each σ_j is the (positive) square root of a λ_j: $\sigma_j = \sqrt{\lambda_j}$, $j = 1, \ldots, r$. (Parenthetical note: notice that in the case where matrix A has full

column rank, A^TA is nonsingular, so that $A^TAx \neq 0_v$. Thus each eigenvalue λ of A^TA will satisfy $\lambda > 0$, so that A^TA will be positive definite.)

How many non-zero λ_j's and σ_j's are there? The notion of rank is the key:

$$\begin{aligned}\text{rank } A &= \text{rank } (U\Sigma V^T)\\ &= \text{rank } (\Sigma V^T), \text{ since } U \text{ is nonsingular}\\ &= \text{rank } \Sigma, \text{ since } V \text{ is nonsingular}\\ &= \text{the number } r \text{ of non-zero diagonal entries: } \sigma_1, \sigma_2, \ldots, \sigma_r.\end{aligned}$$

Normally, we assume that the σ_j's are arranged on the diagonal of Σ in decreasing order:

$$\Sigma^{m\times n} = \begin{bmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \\ \hline 0 & & & 0 \end{bmatrix}, \text{ where } \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r.$$

Using equation (2), we see in a similar way that

$$\begin{aligned}\text{rank } A^TA &= \text{rank } (\Sigma^T\Sigma)\\ &= \text{the number } r \text{ of non-zero diagonal entries: } \sigma_1^2, \sigma_2^2, \ldots, \sigma_r^2.\end{aligned}$$

Thus rank A = rank A^TA. Replacing A in this last result with A^T we have rank A^T = rank AA^T. Thus, all four of these matrices have the same rank:

$$\text{rank } A = \text{rank } A^T = \text{rank } A^TA = \text{rank } AA^T.$$

One final comment: although the singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$ are uniquely determined by A (the σ_j^2 are the non-zero eignevalues of A^TA and AA^T), the matrices

U and V in $A = U\Sigma V^T$ are *not* unique. The columns of U are orthonormal eigenvectors of AA^T and the columns of V are orthonormal eigenvectors of A^TA, and different choices for these columns can be made.

The HP-48G/GX units include the command SVD (on the MTH MATR FACTR menu) for computing a singular value decomposition of a matrix. The command implements a version of the LAPACK routine xGESVD in producing the decomposition. There are also several related commands. The command SVL (also on the MTH MATR FACTR menu) computes only the singular values, and RANK (on the MTH MATR NORM menu) returns a value for the rank of matrix A determined as the number of non-zero singular values of A. If Flag -54 is clear (the default state), RANK automatically treats any computed singular value as zero if it is less than 10^{-14} times the size of the largest computed singular value. The command SNRM (also on the MTH MATR NORM menu) returns the *spectral norm* of a matrix, which is defined as the largest singular value.

EXAMPLE 10. Begin with the following matrix on level 1:

$$A = \begin{bmatrix} 1 & -1 & 1 \\ -3 & -9 & -15 \\ 0 & -9 & -9 \\ -7 & 2 & -12 \end{bmatrix}.$$

(a) The command RANK returns the value 2 for the rank of A. You can verify this by applying the command RREF to A, obtaining

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Thus column 3 of A is 2 times column 1 plus column 2.

(b) With A on level 1, the command SVL returns the following vector S of singular values:

[23.9965775198 10.0580449062 5.06289164242E-14].

The third component shows why the RANK command returned 2 for the rank. Replace the third component with 0 and then square the components in vector S by applying the procedure « SQ » APLY. You will obtain the following vector S^2:

[575.835732664 101.164267335 0].

(c) With the 3×3 matrix A^TA on level 1, the command EGVL will show the vector of eigenvalues to be

[5.38034799593E-13 575.835732666 101.164267334].

Compare this with the components of vector S^2.

(d) With the 4×4 matrix AA^T on level 1, the command EGVL will show the vector of eigenvalues to be

[2.80338767559E-14 575.835732666 101.164267334 -1.41891530649E-12].

(e) With A on level 1, use the command SVD to obtain a singular value decomposition. The stack will be arranged as follows:

3: matrix U

2: matrix V^T

1: vector S of computed singular values

```
      [ [ -2.88503909637E-2   -.157851068131   -.434161002673    .88642818039     ]
      [    .734808458395      -.200855325398    .586417975786    .275368163386    ]
U =   [    .486192964128      -.505806397342   -.607810084379   -.371945145644    ]
      [    .472059939039       .823953922836   -.313344990004   8.61767003328E-3 ] ]
```

```
      [ [ -.230770214413    -.417394627437    -.878935056262 ]
V =   [ -.529224377248     .811859013819    -.246589740676 ]
      [ -.816496580928    -.408248290464     .408248290464 ] ]
```

```
S = [  23.9965775198      10.0580449062      5.06289164242E-14 ]
         σ1                 σ2                  σ3
```

(f) You can verify that column 1 of U is an eigenvector of AA^T corresponding to the eigenvalue $\sigma_1^2 = 575.835732664$ as follows: move U to level 1 and use 1 COL- to get column 1 of U, make a duplicate copy and then compare

AA^T * (*col* 1 of U) = [-16.6130860183 423.128967009 279.967281716 271.828980859]

with the vector

σ_1^2 * (*col* 1 of U) = [-16.6130860182 423.128967008 279.967281715 271.828980858]

by subtracting them. Notice that the components agree up to the twelfth digit.

(g) Likewise, you should verify that column 2 of V is an eigenvector of A^TA corresponding to the eigenvalue $\sigma_2^2 = 101.164267335$: compare the vector

A^TA * (*col* 2 of V) = [-53.5385963797 82.1311223117 -24.9460704477]

with the vector

$\sigma_2^2 * (col\ 2\ of\ V)$ = [-53.5384963801 82.1311223123 -24.9460704478].

Application to Least Squares Solutions

An important application of singular value decompositions is to produce least squares solutions of arbitrary linear systems $Ax = b$. Recall that by a least squares solution to Ax = b we mean a vector x for which $\| Ax - b \|_2$ is as small as possible. And because there may be more than one such solution, we seek one having minimum norm: $\| x \|_2$ is minimal among all least squares solutions.

We begin with a singular value decomposition of the $m \times n$ coefficient matrix A in the linear system $Ax = b$: $A = U\Sigma V^T$. Here U and V are $m \times m$ and $n \times n$ orthogonal matrices, respectively, and Σ is the m×n diagonal matrix

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix},$$

where, Σ_1 is the r×r ($r = rank A$) diagonal matrix containing the singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$. Recall that orthogonal matrices preserve length, i.e., $\| Qz \|_2 = \| z \|_2$ for all orthogonal matrices Q. Applying this with the orthogonal matrix U^T and the vector $z = Ax - b$, we have

$$\begin{aligned} \| Ax - b \|_2^2 &= \| U^TAx - U^Tb \|_2^2 \\ &= \| \Sigma V^Tx - U^Tb \|_2^2 \\ &= \| \Sigma y - c \|_2^2 \text{, with } y = V^Tx \text{ and } c = U^Tb. \end{aligned}$$

Thus $\| Ax - b \|$ will be minimized if $\| \Sigma y - c \|_2$ is minimized. Exploiting the special structure of Σ we have

$$(*) \quad \| Ax - b \|_2^2 = \| \Sigma y - c \|_2^2 = \sum_{j=1}^{r} | \sigma_j y_j - c_j |^2 + \sum_{j=r+1}^{m} | c_j |^2 .$$

Certainly we can make (*) as small as possible by choosing y_j's that will cause the first term to become 0: $y_j = c_j/\sigma_j$ for $1 \le j \le r$. Since $x = Vy$, we can minimize $\| x \|_2$ by minimizing $\| y \|_2$. And since (*) places no restriction upon the y_j for $j = r + 1, \ldots, m$, we can minimize $\| y \|_2$ by choosing $y_{r+1} = \ldots = y_m = 0$.

Reformulating these results in terms of matrix multiplication, we see that the minimum norm least squares solution x can be obtained as follows: calculate, in tern,

(i) $c = U^T b = \begin{bmatrix} \overline{c} \\ d \end{bmatrix}$ ($\overline{c}$ is an r-vector)

(ii) use $\overline{c}$ to build $\overline{y} = \Sigma_1^{-1}\,\overline{c}$ ($\overline{y}$ is also an r-vector)

(iii) use $\overline{y}$ to build $x = V\begin{bmatrix} \overline{y} \\ 0 \end{bmatrix}$ (an n-vector)

EXAMPLE 11. We return to the matrix A of our previous example and use b = [1 -2 0 -1]. Since the rank of the augmented matrix $[A|\, b]$ is 3 and the rank of A is 2, $Ax = b$ has no solution in the usual sense. Use SVD to build a singular value decomposition $A = U\Sigma V^T$ for A. Then delete the last two entries in $c = U^T b$ to obtain

$$\overline{c} = [\ \text{-1.97052724679} \quad \text{-.580094340171}\ ].$$

Now delete the last entry in the vector of singular values and use the command 2 DIAG→ to construct the 2 × 2 matrix

$$\Sigma_1 = \begin{matrix} [\ [\ 23.9965775198 & 0\ ] \\ [\ 0 & 10.0580449062\ ]\ ] \end{matrix}.$$

Then calculate

$$\overline{y} = \Sigma_1^{-1}\,\overline{c} = [\ \text{-8.21170121099E-2} \quad \text{-5.76746619826E-2}\ ].$$

Finally, build the 3-vector y = [$\bar{y}$ 0] and calculate the minimum norm least squares solution as

$$x = Vy = [\ 4.94729975623E\text{-}2 \quad -1.25484945237E\text{-}2 \quad 8.63975006007E\text{-}2\ ].$$

If you compare this solution with the one obtained by the LSQ command, you will notice that the components agree until the twelfth decimal digit.

A final comment. In the most frequent applications of least squares solutions to linear systems $Ax = b$, the $m \times n$ coefficient matrix A has many more rows than columns, so that m is very much larger than n. In the singular value decomposition $A = U\Sigma V^T$, matrix U is m × m and V is $n \times n$, so U will, in general, be very much larger than V. Although our simple examples do not show it, computing a very large U can be expensive. However, as the equations (i) – (iii) that precede **EXAMPLE 11** show, U is needed only to modify vector b to obtain the r-vector $\bar{c}$. Fortunately, there are less expensive ways to obtain $\bar{c}$ than by computing matrix U and these ways are often exploited in professional computer code.

Activity Set 6.6

1. (a) Obtain a singular value decomposition $A = U\Sigma V^T$ for the following matrix

$$A = \begin{bmatrix} 0 & -7 & -5 & -1 \\ -7 & 0 & 8 & -4 \\ 9 & 9 & 3 & 3 \\ 2 & 9 & 1 & 4 \\ 6 & 6 & -12 & 9 \end{bmatrix}.$$

   (b) Compare the non-zero eigenvalues of A^TA with the squares of the singular values of A. Repeat, using the non-zero eigenvalues of AA^T.

   (c) Verify that column 1 of U is an eigenvector of AA^T associated with the eigenvalue σ_1^2, where σ_1 is the singular value of greatest magnitude.

(d) Verify that column 3 of V is an eigenvector of A^TA associated with the eigenvalue σ_3^2, where σ_1 is the singular value of least magnitude.

2. (a) Solve the linear system $Ax = b$ where

$$A = \begin{bmatrix} 4 & 8 & -6 \\ -6 & 1 & -4 \\ -3 & -7 & 7 \end{bmatrix} \quad \text{and} \quad b = [\,-16 \quad -2 \quad 17\,].$$

(b) Use a singular value decomposition of A to obtain a minimum norm least squares solution to $Ax = b$. Compare your results to those from (a).

(c) Use the built-in LSQ command to obtain a minimum norm least squares solution to $Ax = b$. Compare your results to those from (a) and (b).

3. (a) Use a singular value decomposition to obtain a minimum norm least squares solution to $Ax = b$, where A is the matrix of Activity 1 and $b = [\,1 \quad 0 \quad -2 \quad 1 \quad 2\,]$.

(b) Use the built-in LSQ command to obtain a minimum norm least squares solution to the linear system in (b). Compare your results to those in (b).

4. When the coefficent matrix A in a least squares problem $Ax = b$ has full column rank, the matrix A^TA appearing in the system of normal equations $A^TAx = A^Tb$ is positive definite (see the parenthetical note on page 147 of Section 6.4). Using a Cholesky factorization $A^TA = LL^T$, the normal equations are $LL^Tx = A^Tb$ and can therefore be solved by forward and back substitutions: solve $Ly = A^Tb$ by forward substitution, then $L^Tx = y$ by back substitution. Apply this approach and solve the least squares problem associated with fitting a cubic polynomial to the data (1.1, -1), (2.2, 2), (3.3, -3), (4.4, 4), (5.5, -5), (6.6, 6). Compare your answer to the one obtained by using the built-in professional code LSQ.

7 ITERATIVE METHODS

For large linear systems, Gaussian elimination can be costly. The number of multiplications/divisions required to solve $Ax = b$, where A is $n \times n$, is

$$\frac{n^3 - n}{3} \text{ multiplications/divisions for the elimination phase,}$$

followed by an additional

$$n^2 \text{ multiplications/divisions for the back-substitution phase}$$

for a total of

$$\frac{n^3 - n}{3} + n^2 = \frac{n^3}{3} + n^2 - \frac{n}{3} \text{ multiplications/divisions.}$$

When n is large, the $\frac{n^3}{3}$ term dominates; thus, if n is doubled the number of multiplications/divisions is increased by a factor of 8.

Fortunately, many of the large linear systems that arise in practice have coefficient matrices A that are *sparse*, *i.e.*, all but a small fraction of the entries are zero. And there are versions of Gaussian elimination for sparse matrices that capitalize upon the sparseness. But iterative techniques also use sparseness to good advantage by generating a sequence of increasingly better approximations to a solution. By way of introduction to iterative techniques for linear systems, we shall briefly discuss two simple approaches: the *Jacobi* and *Gauss-Seidel iterations*.

We will also consider the elementary iterative process known as the *power method* for approximating the dominant eigenvalue and an associated eigenvector of

certain matrices. Though somewhat limited in its applicability, the power method sets the stage for a subsequent study of the preferred iterative technique for finding eigenvalues, the **QR-algorithm**. However, the QR-algorithm is beyond the scope of this book.

7.1 THE JACOBI AND GAUSS-SEIDEL METHODS

The Jacobi and Gauss-Seidel methods for solving a linear system $Ax = b$ are based upon splitting the matrix A into a difference $A = M - N$, and then rewriting $Ax = b$ as $Mx = Nx + b$. Starting with an initial estimate x_0 for x, we generate a sequence of successive approximations $\{ x_k \}$ where

$$(1) \qquad Mx_k = Nx_{k-1} + b \quad (k \geq 1).$$

With suitable choices for M and N the sequence $\{ x_k \}$ will converge to a solution x. In particular, M should be invertible in order that x_k be uniquely defined:

$$x_k = M^{-1}(Nx_{k-1} + b) = (M^{-1}N)x_{k-1} + M^{-1}b$$

The matrix $M^{-1}N$ is called the *iteration matrix* and is the key to convergence.

- **The Jacobi iteration** takes M to be the diagonal part of A, so $N = M - A$ is the negative of the off-diagonal part of A.
- **The Gauss-Seidel** iteration takes M to be the lower triangular part of A, so $N = M - A$ is the negative of the strictly upper triangular part of A.

Convergence of $\{ x_k \}$ to x is usually defined in terms of the vector max norm:

$$\{ x_k \} \to x \quad \text{if} \quad \lim_{k \to \infty} \| x_k - x \|_\infty = 0,$$

where for $v = [v_1 \ \ v_2 \ \ \ldots \ \ v_n]$ we define $\| v \|_\infty = \max_i | v_i |$

This is equivalent to requiring that each component of $\{ x_k \}$ converge to the corresponding component of x.

More advanced work shows that for an arbitrary initial estimate x_0, $\{ x_k \} \rightarrow x$ iff $| \lambda | < 1$ for each eigenvalue λ of the iteration matrix $M^{-1}N$; in equivalent terms, iff the spectral radius $\rho(M^{-1}N) = max \{ | \lambda | \}$ is less than 1.

To see how the Jacobi and Gauss-Seidel methods differ, write iteration equation (1) in detail (the components of x_k will be displayed as $x_k = [x_1^{(k)} x_2^{(k)} \ldots x_n^{(k)}]$). For the Jacobi iteration, we have

$$
\begin{aligned}
a_{11}x_1^{(k)} &= &&- a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - \quad \ldots \quad - a_{1n}x_n^{(k-1)} &&+ b_1 \\
a_{22}x_2^{(k)} &= -a_{21}x_1^{(k-1)} &&- a_{23}x_3^{(k-1)} - \quad \ldots \quad - a_{2n}x_n^{(k-1)} &&+ b_2 \\
&\vdots \\
a_{nn}x_n^{(k)} &= -a_{n1}x_1^{(k-1)} - a_{n2}x_2^{(k-1)} - \quad \ldots \quad - a_{n,n-1}x_{n-1}^{(k-1)} &&&&+ b_n
\end{aligned}
$$

Thus, to obtain the components of x_k on the left-hand side *we clearly need to have all* $a_{ii} \neq 0$. It is also apparent that the Jacobi method uses the components of the vector x_{k-1} calculated during the $(k-1)^{st}$ iteration (on the right-hand side) to obtain the components of x_k (on the left-hand side) during the k^{th} iteration.

When equation (1) is written in detail for the Gauss-Seidel iteration and the diagonal terms are isolated on the left, we see a difference:

$$
\begin{aligned}
a_{11}x_1^{(k)} &= &&- a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - \quad \ldots \quad - a_{1n}x_n^{(k-1)} &&+ b_1 \\
a_{22}x_2^{(k)} &= -a_{21}x_1^{(k)} &&- a_{23}x_3^{(k-1)} - \quad \ldots \quad - a_{1n}x_n^{(k-1)} &&+ b_2 \\
&\vdots \\
a_{nn}x_n^{(k)} &= -a_{n1}x_1^{(k-1)} - a_{n2}x_2^{(k)} - \quad \ldots \quad - a_{n,n-1}x_{n-1}^{(k)} &&&&+ b_n
\end{aligned}
$$

We calculate $x_1^{(k)}$ from the first equation and immediately use it in the second equation to calculate $x_2^{(k)}$; then we use both $x_1^{(k)}$ and $x_2^{(k)}$ in the third equation to calculate $x_3^{(k)}$, etc. Thus, in the Gauss-Seidel iteration, components calculated early

in the k^{th} iteration are used as soon as they are available to calculate other components in that iteration. This continual updating of components often causes the Gauss-Seidel process to converge faster than the Jacobi process. But there are matrices A for which *only one* of these two processes will converge. Thus, we need both methods.

We previously noted that a necessary and sufficient condition for convergence of either iterative method is that the spectral radius of the iteration matrix $M^{-1}N$ be less than 1: $\rho(M^{-1}N) < 1$. Since for any square matrix B, $\rho(B) \leq \| B \|$ for any matrix norm, estimates on $\rho(M^{-1}N)$ are usually expressed in terms of matrix norms; thus a *sufficient* condition for convergence is that $\| M^{-1}N \| < 1$, for any matrix norm. Because the row-sum norm $\| \bullet \|_\infty$ and the column-sum norm $\| \bullet \|_1$ are so easy to calculate, they are often used:

$$\| A \|_\infty = \max_i \sum_j | a_{ij} | \quad \text{and} \quad \| A \|_1 = \max_j \sum_i | a_{ij} | .$$

Another criterion that is *sufficient* for the convergence of either process is that the coefficient matrix A be ***diagonally dominant***:

(i) row diagonally dominant, $| a_{ii} | > \sum_{\substack{j=1 \\ j \neq i}}^{n} | a_{ij} |$ for $i \leq n$

or

(ii) column diagonally dominant, $| a_{jj} | > \sum_{\substack{i=1 \\ i \neq j}}^{n} | a_{ij} |$ for $i \leq n$.

The basis for this criterion is the following result:

THEOREM

(a) A is diagonally dominant iff the Jacobi iteration matrix has row-sum norm < 1.

(b) If A is diagonally dominant then the Gauss-Seidel iteration matrix has row-sum norm < 1. (The converse is false; see Activity 2.)

We shall use the row-sum and column-sum norms in two calculator programs to test the iteration matrices for convergence. But you should remember that these tests are only sufficient for convergence, not necessary. Thus, it is possible for the tests to fail and still have convergence.

Here are some HP-48G/GX calculator programs that can be used to implement the Jacobi and Gauss-Seidel iterations.

- **D.DOM:** tests the coefficient matrix for diagonal dominance.
- **JTEST:** tests the Jacobi iteration matrix to see if its row-sum norm $\| \bullet \|_\infty$ or column-sum norm $\| \bullet \|_1$ is less than 1.
- **JACOBI:** performs the Jacobi iterative process.
- **STEST:** tests the Gauss-Seidel iteration matrix to see if its row-sum norm $\| \bullet \|_\infty$ or column-sum norm $\| \bullet \|_1$ is less than 1.
- **SEIDL**: performs the Gauss-Seidel iterative process.

The stopping criterion used in both the JACOBI and SEIDL programs is the usual vector max norm measure of relative error:

$$\frac{\| x_k - x_{k-1} \|_\infty}{\| x_k \|_\infty} < \epsilon$$

where ϵ is the error tolerance specified by the user, e.g., $\epsilon = 5 \times 10^{-8}$ for approximately 8 significant digits.

D.DOM (Diagonal dominance)

Input: level 1: an $n \times n$ matrix A

Effect: tests to see if A is diagonally dominant. Returns one of the messages "ROW DIAG DOMINANT", "COL DIAG DOMINANT", or "NOT DIAG DOMINANT".

```
« → A A SIZE 1 GET → n « 1 n FOR i 1 n FOR j 'A(i, j)' EVAL
NEXT n →ARRY DUP { i } GET ABS 2 * SWAP CNRM IF ≤ THEN A
TRN 'A' STO 1 n FOR i 1 n FOR j 'A(i, j)' EVAL NEXT n →ARRY
DUP { i } GET ABS 2 * SWAP CNRM IF ≤ THEN MAXR →NUM 'i'
STO "NOT DIAG DOMINANT" END IF i n == THEN "COL DIAG
DOMINANT" END NEXT KILL END IF i n == THEN "ROW DIAG
DOMINANT" END NEXT » » »
```

JTEST (Test Jacobi iteration matrix)

Input: level 1: an $n \times n$ matrix A

Effect: tests the Jacobi iteration matrix for $Ax = b$ to see if its row-sum norm or column-sum norm is less than 1. Returns an appropriate message.

```
« → A « A SIZE 1 GET → n « n IDN DUP 1 n FOR i 'A(i, i)' EVAL
{ i i } SWAP PUT NEXT A SWAP / – → itmtrx « IF itmtrx RNRM 1 <
THEN "RNRM < 1" ELSE IF itmtrx CNRM 1 < THEN "CNRM < 1"
ELSE "RNRM, CNRM ≥ 1" END END » » » »
```

JACOBI (Jacobi iteration)

Inputs: level 3: an $n \times n$ matrix A

level 2: an n-vector b

level 1: an accuracy level ϵ in the form .00005

Effect: returns, at timed intervals, the successive terms of the Jacobi iteration for $Ax = b$ starting with $x_0 = 0$; terminates when the relative error is less than ϵ or when 60 iterations have occurred; display is set to n FIX where n is the number of digits in ϵ.

```
« → A b ε « A SIZE 1 GET → n « n IDN 1 n FOR i 'A(i, i)' EVAL
{ i i } SWAP PUT NEXT DUP A − → M K « 0 'ct' STO ε XPON NEG
FIX { n } 0 CON 'xn' STO DO xn 'xo' STO K xo * b + M / 'xn' STO
xn CLLCD 3 DISP .5 WAIT 1 'ct' STO+ UNTIL xn xo − RNRM xn
RNRM 10^-12 + / ε < ct 60 == OR END IF ct 60 < THEN ct
'iterations' →TAG ELSE 60 'iterations' →TAG END xn { ct xo xn }
PURGE » » » »
```

EXAMPLE 1. Consider the linear system $Ax = b$ where

$$A = \begin{bmatrix} 4 & 2 & -2 \\ 1 & 6 & -2 \\ -2 & -2 & 10 \end{bmatrix}$$

and b = [2 10 -3]. A is column diagonally dominant, hence invertible, so $Ax = b$ has a unique solution. With A on level 1, JTEST returns the message "CNRM<1". To apply Jacobi iteration to determine the solution x to approximately 5 decimal place accuracy, enter A, b and .00005. Press JACOB to see the iterations converge to [-0.37498 1.71876 -0.03126] after 19 iterations. After changing back to STD display mode you should verify that the exact solution is given by x = [-.375 1.71875 -.03125].

STEST (Test Gauss-Seidel iteration matrix)

Input: level 1: an $n \times n$ matrix A

Effect: tests the Gauss-Seidel iteration matrix for $Ax = b$ to see if its row-sum norm or column-sum norm is less than 1. Returns an appropriate message.

```
« → A « A SIZE 1 GET → n « n IDN DUP 1 n FOR i 1 i FOR j
'A(i, j)' EVAL { i j } SWAP PUT NEXT NEXT A SWAP / – → itmtrx « IF
itmtrx RNRM 1 < THEN "RNRM<1" ELSE IF itmtrx CNRM 1 < THEN
"CNRM<1" ELSE "RNRM, CNRM≥1" END END » » » »
```

SEIDL (Gauss-Seidel iteration)

Inputs: level 3: an $n \times n$ matrix A

level 2: an n vector b

level 1: an accuracy level ϵ in the form .00005.

Effect: returns, at timed intervals, the successive terms of the Gauss-Seidel iteration for $Ax = b$ starting with $x_0 = 0$; terminates when the relative error is less than ϵ or when 60 iterations have occurred; display is set to n FIX where n is the number of digits in ϵ.

```
« → A b ε « A SIZE 1 GET → n « n IDN 1 n FOR i 1 i FOR j
'A(i, j)' EVAL { i j } SWAP PUT NEXT NEXT DUP A − → M K « 0 'ct'
STO ε XPON NEG FIX { n } 0 CON 'xn' STO DO xn 'xo' STO K xo
*  b + M / 'xn' STO xn CLLCD 3 DISP .5 WAIT 1 'ct' STO+ UNTIL
xn xo − RNRM xn RNRM 10^12 + / ε < ct 60 == OR END IF ct 60
< THEN ct 'iterations' →TAG ELSE 60 'iterations' →TAG END xn { ct
xo xn } PURGE » » » »
```

EXAMPLE 2. Use the linear system of EXAMPLE 1. With ϵ = .00005, SEIDL shows the iterations converging to [-0.37499 1.71875 -0.03125] after only 8 iterations.

Activity Set 7.1

1. Consider the following linear system:

$$\begin{aligned}
-3x &+ 9z + 3w + 2v &&= -29\\
-2x + y &+ 2w - 6v &&= 29\\
10x + 2y - 4z &+ 2w - v &&= -26\\
x - 2y + 2z &+ 7w - v &&= 6\\
-2x + 8y + 4z &- w &&= 24
\end{aligned}$$

   (a) Is the coefficient matrix diagonally dominant?

   (b) Apply program JTEST. What is the problem here? Swap rows four and five and try again.

   (c) Apply program STEST to the last matrix you had in (b). What is your conclusion?

   (d) Rearrange the equations to get an equivalent system with a diagonally dominant coefficient matrix.

   (e) Solve the rearranged system by Gauss-Seidel iteration, accurate to approximately 6 decimal places.

   (f) Write a paragraph explaining your observations and what you have learned by this activity.

2. (a) Use matrix

$$A = \begin{bmatrix} 9 & 2 & -3 \\ -1 & 4 & 2 \\ 2 & -3 & 5 \end{bmatrix}$$

   to show that the converse of part (b) of the above Theorem is false.

(b) Test the Jacobi iteration matrix for $Ax = b$ for convergence.

3. Consider the linear system $Ax = b$ where

$$A = \begin{bmatrix} 6 & 1 & -2 & 0 & 0 \\ 0 & 4 & 0 & -1 & 0 \\ 1 & -1 & 5 & 2 & 1 \\ -1 & 0 & 1 & 5 & 0 \\ 0 & 1 & 0 & -1 & 3 \end{bmatrix}$$

and $b = [\ -2.2 \quad -4.4 \quad 46.5 \quad -19.8 \quad 18.7\ ]$.

(a) Is A diagonally dominant?

(b) Apply the Jacobi iteration to obtain a solution that is accurate to approximately 8 decimal digits; how many iterations were required?

(c) Now apply the Gauss-Seidel iteration using an accuracy factor of 5×10^{-8}; how many iterations were required?

(d) What is the exact solution?

4. Consider this tridiagonal system:

$$\begin{aligned} 2x_1 + x_2 &= 5 \\ x_1 + 2x_2 + x_3 &= -2 \\ x_2 + 2x_3 + x_4 &= -13 \\ x_3 + 2x_4 + x_5 &= -10 \\ x_4 + 2x_5 &= 8 \end{aligned}$$

(a) Apply D.DOM, JTEST and STEST as tests for convergence.

(b) On the basis of your results in (a), apply an iterative method to solve the system to approximately 8 decimal place accuracy.

5. Consider the tridiagonal system

$$\begin{aligned}
x_1 - x_2 &= -4 \\
-x_1 + 3x_2 - x_3 &= 11 \\
- x_2 - 3x_3 - x_4 &= -1 \\
- x_3 + 3x_4 - x_5 &= 3 \\
- x_4 + 3x_5 - x_6 &= -2 \\
- x_5 + x_6 &= -3
\end{aligned}$$

(a) Apply all our tests for convergence. What can you conclude?

(b) Remember, these tests are only *sufficient* conditions for convergence. Thus, ignore the test results and try for an iterative solution anyway, accurate to approximately 6 decimal places.

(c) What is the actual solution?

7.2 THE POWER METHOD

The power method is a simple iterative technique for finding an approximation to the dominant eigenvalue and an associated eigenvector of a matrix A. By a *dominant* eigenvalue we mean an eigenvalue λ_1 satisfying

$$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$$

where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are all the eigenvalues of A. Since a matrix may fail to have a dominant eigenvalue, the power method is not a general purpose technique. However, it forms the basis for other, more powerful, iterative methods; in particular, the QR-algorithm.

The power method is based upon the following assumptions about the matrix A:

(a) A is real and diagonalizable;

(b) A has a dominant eigenvalue λ_1.

We start with an arbitrary vector y_0 written in terms of independent eigenvectors $x_1, x_2, \ldots, x_n$ as $y_0 = a_1x_1 + \ldots + a_nx_na$ where $a_1 \neq 0$, and form the sequence of unit vectors $\{ y_k \}$ as $y_k = \dfrac{Ay_{k-1}}{\| Ay_{k-1} \|}$, for $k = 1, 2, \ldots$. Here, we are using the usual Euclidean vector norm (or length). Thus $y_1 = \dfrac{Ay_0}{\| Ay_0 \|}$, $y_2 = \dfrac{Ay_1}{\| Ay_1 \|}$, $y_3 = \dfrac{Ay_2}{\| Ay_2 \|}$, etc. Under the two stated assumptions and that $a_1 \neq 0$, the power method asserts that

(i) the sequence $\{ y_k \}$ converges to a unit eigenvector v associated with λ_1;

and

(ii) the sequence $\{ Ay_k \bullet y_k \}$ converges to the dominant eigenvalue λ_1.

To see why, consider the sequence $z_0 = y_0$, $z_k = Az_{k-1}$ (k = 1, 2, ...) without normalization:

$$z_1 = Ay_0, z_2 = Az_1 = A^2y_0, \ldots, z_k = Az_{k-1} = A^ky_0.$$

Using $A^kx_j = \lambda^kx_j$ and our expression for y_0 we have

$$z_k = A^ky_0 = a_1\lambda_1^kx_1 + \ldots + a_n\lambda_n^kx_n, \text{ or}$$

$$(2)\quad z_k = \lambda_1^k\left[a_1x_1 + a_2\left(\frac{\lambda_2}{\lambda_1}\right)^k x_2 + \ldots + a_n\left(\frac{\lambda_n}{\lambda_1}\right)^k x_n \right].$$

$$\text{Now } y_1 = \frac{Ay_0}{\| Ay_0 \|} = \frac{z_1}{\| z_1 \|},\quad y_2 = \frac{Ay_1}{\| Ay_1 \|} = \frac{A\left(\dfrac{z_1}{\| z_1 \|}\right)}{\left\| A\left(\dfrac{z_1}{\| z_1 \|}\right)\right\|} = \frac{Az_1}{\| Az_1 \|} = \frac{z_2}{\| z_2 \|},$$

and in general, $y_k = \dfrac{z_k}{\| z_k \|}$. Looking back at (2), we see that

$$(3)\quad y_k = \frac{\lambda_1^k \left[a_1 x_1 + a_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k \right] x_2 + \ldots + a_n \left(\frac{\lambda_n}{\lambda_1}\right)^k x_n}{|\lambda_1^k| \left\| a_1 x_1 + a_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k x_2 + \ldots + a_n \left(\frac{\lambda_n}{\lambda_1}\right)^k x_n \right\|}$$

Since $|\lambda_1| > |\lambda_j|$ for $j = 2, \ldots, n$, we have $\left| \frac{\lambda_j}{\lambda_1} \right| < 1$ for $j = 2, \ldots, m$. Thus from (3),

$$\text{as } k \to \infty, \quad y_k \to \pm \frac{a_1 x_1}{\| a_1 x_1 \|} = v,$$

which is a unit eigenvector associated with λ_1.

Finally, since $y_k \to v$, $Ay_k \to Av = \lambda_1 v$, so that $Ay_k \bullet y_k \to \lambda_1 v \bullet v = \lambda_1 (v \bullet v) = \lambda_1 \| v \|^2 = \lambda_1$. This completes the argument.

In summary, under our three assumptions, the power method gives us two sequences:

- a sequence of vectors $\{ y_k \}$ converging to an eigenvector v associated with the dominant eigenvalue λ_1 ;
- an associated sequence of numbers $\{ Ay_k \bullet y_k \}$ converging to the dominant eigenvalue λ_1 .

In practice, we calculate the sequences together, term-by-term. Because the convergence of $\{ y_k \}$ to a vector v amounts to the convergence of the components of y_k to the corresponding components of v, we base our stopping criterion in terms of the relative error $\frac{\| y_k - y_{k-1} \|_\infty}{\| y_k \|_\infty}$ on the sequence $\{ y_k \}$. The sequence of numbers $\{ Ay_k \bullet y_k \}$ often converges more rapidly.

To implement the entire process on the HP-48G/GX requires a program to calculate both sequences, term-by-term, and to apply the desired stopping criterion. Program POWER does just that.

POWER (Power method)

Input: level 3: a real $n \times n$ matrix A, assumed to be diagonalizable and have a dominant eigenvalue λ

level 2: an n-vector y_0, assumed to have a non-zero component in the direction of an eigenvector associated with λ

level 1: an accuracy level ϵ in the form .00005

Effect: returns, at timed intervals, the successive terms of a sequence of vectors that approaches a dominant eigenvector, and a corresponding sequence of numbers that approaches the dominant eigenvalue; terminates the relative error in the sequence of vectors is less than ϵ or when 180 iterations have occurred. Display is set to n FIX, where n is the number of digits in ϵ.

```
« → A y0 ∈ « 0 'ct' STO ∈ XPON NEG FIX y0 'yN' STO DO yN 'yO'
STO A yO * DUP ABS / 'yN' STO A yN * yN DOT CLLCD 1
DISP yN 4 DISP .5 WAIT 1 'ct' STO+ UNTIL yN yO - RNRM yN
RNRM 10^-12 + / ∈ < ct 180 == OR END IF ct 180 < THEN ct
'iterations' →TAG ELSE 180 'iterations' →TAG END A yN * yN DOT
yN { ct yO yN } PURGE » »
```

EXAMPLE 3. Enter and store the matrix

$$A = \begin{matrix}[[\ 1 & -1 & -2 & -3\ ] \\ [\ 1 & 3 & 2 & 3\ ] \\ [\ 0 & -1 & -1 & -3\ ] \\ [-1 & 1 & 2 & 3\ ]]\end{matrix}.$$

You can verify that A has eigenvalues 3, 2, 1 and 0, hence is diagonalizable. Thus $\lambda = 3$ is the dominant eigenvalue. Let y_0 = [1 1 1 1] and proceed on the assumption that y_0 has a non-zero component in the direction of an eigenvector associated with $\lambda = 3$. Using 6 decimal place accuracy specified by $\epsilon = .000005$, POWER shows a sequence of numbers converging to $\tilde{\lambda} = 3.000009$ and a sequence of vectors converging to $\tilde{x}$ = [-.499999 .500004 -.499999 .499999] after 27 iterations. To see how close the pair $(\tilde{\lambda}, \tilde{x})$ is to being an eigenvalue-eigenvector pair for A: duplicate $\tilde{x}$ with ENTER, recall A to level 1 with A and press SWAP * to see $A\tilde{x}$ = [-1.500001 1.500013 -1.500001 1.50001]. Now do 3 ROLLD ENTER * to see $\tilde{\lambda}\,\tilde{x}$ = [-1.500000 1.500018 -1.500000 1.500000]. Compare levels 1 and 2.

COMMENT. As in this example, we usually do not know in advance whether the initial vector y_0 has a non-zero component in the direction of a dominant eigenvector. But this rarely causes difficulty in practice because the round-off errors that appear after a few iterations usually perturb the problem to the point where this is the case.

EXAMPLE 4. To see the effect of different initial vectors y_0, return to the matrix A of EXAMPLE 3 and use y_0 = [1 0 -1 0], then use y_0 = [1 2 3 4]. With 6 place accuracy, the first choice shows $\tilde{\lambda}$ = 2.999991 and $\tilde{x}$ = [.500001 -.499996 .500001 -.500001] after 28 iterations while the second choice shows $\tilde{\lambda}$ = 3.00010 and $\tilde{x}$ = [-.499998 .500005 -.499998 .499998] after 25 iterations.

Activity Set 7.2

1. Seed your random number generator with 2 and generate the following random 6×6 symmatrix matrix A

$$A = \begin{bmatrix} 5 & 4 & -2 & 5 & -8 & 5 \\ 4 & 6 & -4 & 7 & 8 & 0 \\ -2 & -4 & 0 & -6 & -4 & 0 \\ 5 & 7 & -6 & 2 & 8 & -5 \\ -8 & 8 & -4 & 8 & 3 & -2 \\ 5 & 0 & 0 & -5 & -2 & 0 \end{bmatrix}$$

over Z_{10} with program **SYMM**:

   (a) Verify that A has a dominant eigenvalue λ.

   (b) Apply the power method via program **POWER**, starting with the vector of all 1's and using 8 decimal place accuracy.

   (c) Let $\tilde{\lambda}$ and $\tilde{x}$ denote the approximations to λ and an associated eigenvector obtained in (b). Verify that $A\tilde{x} \approx \tilde{\lambda}\,\tilde{x}$.

2. Looking back at equation (2) in our derivation of the power method, you can see that the rate of convergence is governed by the factor $\left| \frac{\lambda_2}{\lambda_1} \right|$ (remember: $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$). The smaller this factor, the faster the convergence. That is, the convergence will be faster if λ_1 *strongly* dominates the next largest eigenvalue. To see this in practice, consider the following two sparse (tridiagonal) matrices A and B, that differ only in their (1,1)-entries.

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 6 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

(a) Verify that for matrix A, $\left| \frac{\lambda_2}{\lambda_1} \right| \approx .84878$ and for matrix B, $\left| \frac{\lambda_2}{\lambda_1} \right| \approx .43725$, a little more than one-half the value for matrix A.

(b) Apply the power method to matrix A using $y_0 = [\ 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1\ ]$. Aim for 12 decimal place accuracy and note the number of iterations required.

(c) Now apply the power method to matrix B using $y_0 = [\ 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1\ ]$. Again, aim for 12 decimal place accuracy and note the number of iterations required. Compare the iteration count with that in (b).

3. Seed your random number generator with 3, and use program SYMM to build the following 5×5 symmetrix matrix:

$$A = \begin{bmatrix} 1 & 1 & -5 & 2 & -1 \\ 1 & 3 & 2 & -7 & 4 \\ -5 & 2 & 5 & 1 & -7 \\ 2 & -7 & 1 & -8 & 9 \\ -1 & 4 & -7 & 9 & -9 \end{bmatrix}.$$

(a) Apply the power method using [1 1 1 1 1] and 6 decimal place accuracy. *Pay close attention to the sequence of vectors being generated.*

(b) How many iterations occurred? This is the maximum number allowed by program POWER.

To see why this many iterations occurred, notice that the stopping criteria in POWER is: $\frac{\| y_{k+1} - y_k \|}{\| y_{k+1} \| + 10^{-12}} < \epsilon$ or ct = 180 (where ct is the iteration counter). For this particular matrix, after 18 iterations, the components of y_{k+1} and y_k agree to 6 decimal places except for a sign: $y_{k+1} = -y_k$. Thus $y_{k+1} - y_k = 2y_{k+1}$, so $\| y_{k+1} - y_k \|_\infty = \| 2y_{k+1} \|_\infty = 2 \| y_{k+1} \|_\infty$ and the relative error ceases to decrease. Thus the iterations continue to the maximum allowable.

APPENDIX

A

VECTOR AND MATRIX NORMS

When a vector $v = [\, x_1, x_2, x_3 \,]$ in R^3 is interpreted geometrically, its length is given by $\| v \| = \left[x_1^2 + x_2^2 + x_3^2 \right]^{1/2}$. The well-known properties of vector lengths include:

(1) $\| v \| > 0$ if $v \neq 0$,

(2) $\| \alpha v \| = | \alpha | \, \| v \|$ for any scalar α and any vector v,

(3) $\| v + w \| \leq \| v \| + \| w \|$ for any vectors v, w.

It seems natural to adopt the corresponding notion for length R^n: for any vector $v = [\, x_1, x_2, \ldots, x_n \,]$, $\| v \| = \left[x_1^2 + x_2^2 + \ldots + x_n^2 \right]^{1/2}$. But there are situations where other scalar measures of the "size" of vectors in R^n is more meaningful. For instance, if the components of $v = [\, 6, 3, 2, 5, 9 \,]$ record the average times required to complete different components in an assembly operation, then $(6^2 + 3^2 + 2^2 + 5^2 + 9^2)^{1/2}$ is somewhat meaningless when compared to $6 + 3 + 2 + 5 + 9$ (the sum of the average assembly times) or to $\max \{ 6, 3, 2, 5, 9 \}$ (the largest average assembly time). In general, several different notions of the length, or size, of a vector may be useful.

The term *norm* is applied to any generalization of Euclidean length in R^3 as long as the above three conditions are met. The most commonly used norms for vectors in R^n are these:

- The ***Euclidean vector norm***: $\| v \|_2 = \left[\sum_i | x_i |^2 \right]^{1/2}$
- The ***vector sum norm***: $\| v \|_1 = \sum_i | x_i |$
- The ***vector max norm***: $\| v \|_\infty = \max_i | x_i |$.

All of these are true vector norms, in the sense that they satisfy the above conditions (1) – (3).

Analogous to vector norms are the *matrix norms* $\| A \|$ which are scalar measures of square matrices. To qualify as a matrix norm the number $\| A \|$ must satisfy:

(1) $\| A \| > 0$ if $A \neq 0$,

(2) $\| \alpha A \| = | \alpha | \| A \|$, for any scalar α and any matrix A,

(3) $\| A + B \| \leq \| A \| + \| B \|$, for any matrices A, B,

(4) $\| AB \| \leq \| A \| \| B \|$, for any matrices A, B.

Conditions (1) - (3) are the same as for vector norms, but (4) is new and implies that $\| A^n \| \leq \| A \|^n$. One of its more important uses is that if $\| A \| < 1$ then $\| A^n \| \to 0$ as $n \to \infty$.

When our earlier examples of vector norms are applied to square matrices, the first two are matrix norms:

- the ***Euclidean (or Frobenius) norm***: $\| A \|_F = \left[\sum_{i,j} |x_{ij}|^2 \right]^{1/2}$, and

- the ***sum norm***: $\| A \| = \sum_{i,j} | a_{ij} |$;

but the third one, $\| A \| = \max_{i,j} | a_{ij} |$ fails to be a matrix norm because condition (4) need not hold.

Although there are many ways to define matrix norms, it is especially useful to use a matrix norm that is connected to an existing vector norm. This can be done as follows: given a vector norm $\| x \|$ for vectors x in R^n, we define a matrix norm $\| A \|$ for $n \times n$ matrices by $\| A \| = \max_{\|x\|=1} \| Ax \|$.

This produces a true matrix norm (that is, (1) – (4) hold) that measures the amount by which a vector x of norm 1 is "magnified" by matrix A. We call $\| A \|$ the matrix norm *induced* by the vector norm $\| x \|$. The most important properties of induced matrix norms are these:

(5) $\| Ax \| \leq \| A \| \| x \|$ for all x, and

(6) $\| I_n \| = 1$.

When the three common vector norms are used to induce matrix norms, it can be proved[1] that:

- the vector sum norm induces the ***column-sum norm*** of A:

$$\| A \|_1 = \max_j \sum_i | a_{ij} |$$

[1]See, e.g., Section 5.6 in *Matrix Analysis*, by Horn and Johnson, Cambridge University Press, 1985.

- the vector max norm induces the ***row-sum norm*** of A:

$$\| A \|_\infty = \max_i \sum_j | a_{ij} |$$

- the Euclidean norm induces the ***spectral norm*** of A:

$$\| A \|_2 = max \{\sqrt{\lambda} : \lambda \text{ is an eigenvalue of } A^T A\}.$$

Of these three matrix norms, the column-sum and row-sum norms are used most often because they are so easy to calculate. The Frobenius norm is also easy to calculate but is not induced by a vector norm. The spectral norm, on the other hand, is much more difficult to obtain and is mainly for theoretical use.

The spectral norm is not the only connection between matrix norms and eigenvalues. For any square matrix A, its *spectral radius* $\rho(A)$ is defined by

$\rho(A) = max \{ | \lambda | : \lambda$ is an eigenvalue of $A \}$, and it can be shown that $\rho(A) \leq \| A \|$ for any matrix norm. Thus the column-sum and row-sum norms provide easy estimates of $\rho(A)$.

Norms on the HP-48G/GX

Four matrix and vector norms are provided on the MTH MATR NORM menu of the 48G/GX:

- The Euclidean (Frobenius) matrix norm $\| A \|_F$: the ABS command. Since vectors on the HP-48G/GX are one-dimensional arrays sensed as column vectors, ABS applied to a vector v returns its vector Euclidean norm $\| v \|_2$.

- The row-sum, or ∞-norm $\| A \|_\infty$: the RNRM command. For a vector v, RNRM returns its vector max norm $\| v \|_\infty$.

- The column-sum, or 1-norm $\| A \|_1$: the CNRM command. For a vector v, CNRM returns its vector sum norm $\| v \|_1$.

- The spectral norm $\| A \|_2$: the SNRM command. For a vector v, SNRM returns the vector Euclidean norm $\| v \|_2$.

EXAMPLE

(a) Consider the following matrix

$$A = \begin{bmatrix} 4 & -3 & 0 \\ 0 & 5 & -1 \\ 2 & 0 & -3 \end{bmatrix}.$$

The command ABS returns $\| A \|_F = 8$, the command RNRM returns $\| A \|_\infty = 7$, the command CNRM returns $\| A \|_1 = 8$ and the command SNRM returns $\| A \|_2 = 6.3996414623$.

(b) For v = [-1 2 5 1 -2 1], ABS returns $\| v \|_2 = 6$, RNRM returns $\| v \|_\infty = 5$, CNRM returns $\| v \|_1 = 12$, and SNRM returns $\| v \|_2 = 6$.

ACTIVITIES

1. What is the Frobenius norm for an identity matrix? Experiment with identity matrices of orders 3, 4, 5, 9 and 16 to find out.

2. How do the row-sum and column-sum norms compare for symmetric matrices? Experiment with random 3×3, 4×4 and 5×5 matrices over Z_{10} to find out.

3. Seed your random number generator with 3 and generate a random 4×4 matrix A and a random 4-vector x over Z_{10}. Use A and x to verify that for each of the four matrix and vector norms provided on the HP-48G/GX, $\| Ax \| \leq \| A \| \| x \|$.

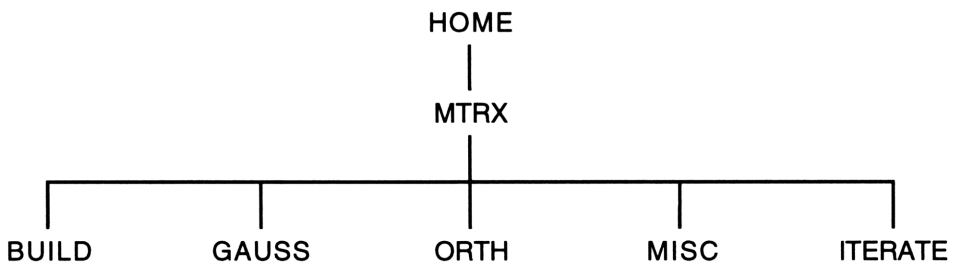
4. Consider the attempt to define a matrix norm by $\| A \| = \max_{i,j} | a_{ij} |$. Experiment with random 3×3 matrices over Z_{10} to find a pair A, B for which the inequality $\| AB \| \leq \| A \| \| B \|$ is invalid.

APPENDIX
B

TEACHING CODE: ORGANIZATION

The special-purpose HP-48G/GX programs for teaching linear algebra that are contained in this book are called *teaching code*; a listing appears on the inside back cover. The teaching code is readily available on a diskette from the author for downloading to an HP-48G/GX from a microcomputer. This appendix shows how the teaching code is organized in files, or directories.

A factory-fresh HP-48G/GX calculator contains only the built-in HOME directory, indicated by the message { HOME } at the top left of the stack display screen. The teaching code for linear algebra is stored in a directory called MTRX, and MTRX is divided into five subdirectories, each one containing teaching code related to a major topic.



- Subdirectory BUILD. Contains the teaching code used to build special types of matrices (see Chapter 2): SYMM, L.TRI, U.TRI, DIAG, TRIDIA, HILB.

- Subdirectory GAUSS. Contains the teaching code related to Gaussian elimination (see Chapter 3): ELIM, PIVOT, L.U, FWD, BACK, L.SWP, →LP.

- Subdirectory ORTH. Contains the teaching code related to orthogonality concepts (see Chapter 5): PROJ, GS, P.FIT.

- Subdirectory MISC. A subdirectory containing miscellaneous teaching code: CHAR, P.DEF, P.of.A, A↑K, APLY.

- Subdirectory ITERATE. Contains the teaching code related to iterative methods (see Chapter 7): JTEST, JACOBI, STEST, SEIDL, D.DOM, POWER.

To execute any of these programs, open the MTRX directory with the MTRX key, then open the appropriate subdirectory with its menu key. Put the necessary inputs to a particular program on the stack and then execute the name of the program by typing the name and using the ENTER key, or using the appropriate menu key.

You have access to all built-in commands from any MTRX subdirectory without exiting from that subdirectory. Simply type the command and press ENTER (be sure to first provide the necessary inputs on the stack), or use the appropriate built-in menu key. If you use a built-in menu key, you can return directly to the MTRX subdirectory with the VAR key.

To move up from a particular MTRX subdirectory to the main MTRX directory, use the UP key (the left-shifted ' key).

To rearrange any of the variables in a subdirectory of MTRX (including the MTRX directory itself), apply the command ORDER (on the ↰ MEMORY DIR

submenu) to a list that contains the names of the variables in the desired order, left-to-right.

And finally, a word of caution. With any object on stack level 1, pressing [↰] and then the menu key beneath a particular user-constructed variable (in particular, one of our teaching code programs) will overwrite the contents of that variable with the object from level 1. So be careful; in a hasty moment it is easy to destroy teaching code.

SOLUTIONS

Activity Set 2.2

1. (a) $B =$
```
[[ 3   2   -1   5   0 ]
 [ -6  7   3    0   1 ]
 [ 2  -4   8    7   9 ]
 [ 5   2  -6    1   3 ]]
```

(b) $C =$
```
[[ 3   2   -1 ]
 [ 5   0   -6 ]
 [ 7   3    0 ]
 [ 1   2   -4 ]
 [ 8   7    9 ]]
```

(c) $D =$
```
[[ 3    2     -1     ]
 [ -3   0     -6     ]
 [ 7    3      0     ]
 [ 1    2     -4     ]
 [ 0    7    1.375   ]]
```

(d) $E =$
```
[[ 3   2     -1     ]
 [ 7   3      0     ]
 [ 0   7    1.375   ]]
```

2. (a) $A =$
```
[[ .11  .12  .13  .14 ]
 [ .21  .22  .23  .24 ]
 [ .31  .32  .33  .34 ]
 [ .41  .42  .43  .44 ]
 [ .51  .52  .53  .54 ]]
```

(b) $B =$
```
[[ .21  .22  .23  .24 ]
 [ .31  .32  .33  .34 ]
 [ .51  .52  .53  .54 ]]
```

(c) $C =$
```
[[ .21  .22  .23 ]
 [ .24  .31  .32 ]
 [ .33  .34  .51 ]
 [ .52  .53  .54 ]]
```

(d)
```
[[ .62  .63  .64 ]
 [ .21  .22  .23 ]
 [ .33  .34  .51 ]
 [ .52  .53  .54 ]]
```

3. (a)
```
[[ 1  1  1  5 ]
 [ 2  2  2  5 ]
 [ 3  3  3  5 ]
 [ 4  4  4  5 ]]
```

(b)
```
[[ 1  1  1  5 ]
 [ 2  2  2  5 ]
 [ 3  3  3  5 ]
 [ 4  4  4  4 ]]
```

(c) No; notice the (4,4)-entries

4. (a)
level 2: A
level 1: B
4 COL+ returns [A B]

and

level 2: A
level 1: B
4 ROW returns $\begin{bmatrix} A \\ B \end{bmatrix}$

(b) Approaches vary

5. $A = \begin{bmatrix} -5 & 1 & 6 & -7 \\ -6 & -3 & 8 & 7 \\ -1 & 5 & 5 & -9 \end{bmatrix}$ $B = \begin{bmatrix} -9 & 4 \\ 3 & 4 \\ -7 & 9 \end{bmatrix}$ $C = \begin{bmatrix} -5 & 1 & -9 & 4 & 6 & -7 \\ -6 & -3 & 3 & 4 & 8 & 7 \\ -1 & 5 & 7 & 9 & 5 & -9 \end{bmatrix}$

6. $A = \begin{bmatrix} 0 & -2 & 4 \\ -2 & -3 & 3 \\ 4 & 3 & 6 \end{bmatrix}$ $B = \begin{bmatrix} -5 & 0 & 0 \\ -8 & 8 & 2 \\ 0 & -2 & -6 \end{bmatrix}$ $C = \begin{bmatrix} -0 & -2 & 4 \\ -5 & 0 & 0 \\ -8 & 8 & 2 \\ 0 & -2 & -6 \\ -2 & -3 & 3 \\ 4 & 3 & 6 \end{bmatrix}$

$D = \begin{bmatrix} -5 & 0 & 0 \\ 0 & -2 & -6 \\ -2 & -3 & 3 \\ 4 & 3 & 6 \end{bmatrix}$ $E = \begin{bmatrix} -5 & 0 & 0 & 0 \\ -2 & -6 & -2 & -3 \\ 3 & 4 & 3 & 6 \end{bmatrix}$

Activity Set 2.3

1. $A = \begin{bmatrix} 7 & -9 \\ -8 & -5 \\ 8 & -1 \end{bmatrix}$, $B = \begin{bmatrix} 0 & 5 & 3 \\ -3 & -7 & 7 \\ 1 & -2 & -5 \\ -2 & 9 & -7 \end{bmatrix}$ and $BA = \begin{bmatrix} -16 & -28 \\ 91 & 55 \\ -17 & 6 \\ -142 & -20 \end{bmatrix}$

2. (a) $A = \begin{bmatrix} 0 & -1 & -2 & -3 \\ 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 \\ 3 & 2 & 1 & 0 \\ 4 & 3 & 2 & 1 \end{bmatrix}$ (b) $B = \begin{bmatrix} 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 \\ 4 & 3 & 2 & 1 \end{bmatrix}$ (c) $C = \begin{bmatrix} 1 & 0 & -2 \\ 2 & 1 & -1 \\ 4 & 3 & 1 \end{bmatrix}$

(d) $C^2 = \begin{bmatrix} -7 & -6 & -4 \\ 0 & -2 & -6 \\ 14 & 6 & -10 \end{bmatrix}$ $C^3 = \begin{bmatrix} -35 & -18 & 16 \\ -28 & -20 & -4 \\ -14 & -24 & -44 \end{bmatrix}$

3. $A = \begin{bmatrix} 1 & 0 & -2 & 3 \\ 2 & -3 & 0 & -1 \\ 5 & -2 & 4 & 1 \end{bmatrix}$

(a) $B = \begin{bmatrix} 0 & -3 & -2 \\ 2 & 0 & 4 \end{bmatrix}$, $BA = \begin{bmatrix} -16 & 13 & -8 & 1 \\ 18 & -8 & 20 & -2 \end{bmatrix}$

(b) $C = \begin{bmatrix} 1 & 3 \\ 2 & -1 \\ 5 & 1 \end{bmatrix}$ and $CB = \begin{bmatrix} -6 & -3 & 10 \\ 2 & -6 & -8 \\ -2 & -15 & -6 \end{bmatrix}$

4. (a) For example, seed with 4 and generate the 3×4 matrix

$$A = \begin{bmatrix} -8 & 8 & 9 & -4 \\ -9 & 0 & 1 & 4 \\ -4 & 2 & 8 & 6 \end{bmatrix}.$$

$$\text{Then } A^TA = \begin{bmatrix} 161 & -72 & -113 & -28 \\ -72 & 68 & 88 & -20 \\ -113 & 88 & 146 & 16 \\ -28 & -20 & 16 & 68 \end{bmatrix}$$

(b) For the 4×5 matrix

$$B = \begin{bmatrix} -2 & 1 & 6 & 1 & 0 \\ 6 & 6 & -9 & -4 & 0 \\ -4 & -5 & 4 & -4 & 7 \\ -8 & -5 & 0 & 4 & -4 \end{bmatrix}$$

$$\text{we have } B^TB = \begin{bmatrix} 120 & 94 & -82 & -42 & 4 \\ 94 & 87 & -68 & -23 & -15 \\ -82 & -68 & 133 & 26 & 28 \\ -42 & -23 & 26 & 49 & -44 \\ 4 & -15 & 28 & -44 & 65 \end{bmatrix}$$

and for the 5×6 matrix

$$C = \begin{bmatrix} -6 & 9 & 4 & 8 & 1 & 2 \\ 7 & -2 & -6 & 9 & -2 & -2 \\ 5 & -3 & 1 & 4 & 4 & -8 \\ 2 & 2 & 4 & -4 & 4 & 9 \\ -1 & 5 & -7 & 5 & 5 & -4 \end{bmatrix}$$

$$\text{we have } C^TC = \begin{bmatrix} 115 & -84 & -46 & 22 & 3 & -44 \\ -84 & 123 & 18 & 59 & 34 & 44 \\ -46 & 18 & 118 & -69 & 1 & 76 \\ 22 & 59 & -69 & 202 & 15 & -90 \\ 3 & 34 & 1 & 15 & 62 & -10 \\ -44 & 44 & 76 & -90 & -10 & 169 \end{bmatrix}$$

(c) Conjecture: For any matrix A, A^TA is symmetric.

(d) Proof: $(A^TA)^T = A^T(A^T)^T = A^TA$.

5. (a) $A = \begin{bmatrix} 7 & -9 & -8 & -5 \\ 8 & -1 & 0 & 5 \\ 3 & -3 & -7 & 7 \\ 1 & -2 & -5 & -2 \end{bmatrix}$ and $B = \begin{bmatrix} 9 & -7 & 5 & 0 \\ 3 & 1 & 3 & 6 \\ -3 & 5 & -6 & -9 \\ 4 & -1 & -3 & -5 \end{bmatrix}$

(b) $A + iB =$
```
[[ (7, 9)   (-9, -7)  (-8, 5)   (-5, 0) ]
 [ (8, 3)   (-1, 1)   (6, 3)    (5, 6)  ]
 [ (3, -3)  (-3, 5)   (-7, -6)  (7, -9) ]
 [ (1, 4)   (-2, -1)  (-5, -3)  (-2, 5) ]]
```

$(A + iB)^T =$
```
[[ (7, 9)   (8, 3)   (3, -3)   (1, 4)  ]
 [ (-9, -7) (-1, 1)  (-3, 5)   (-2, -1)]
 [ (-8, 5)  (6, 3)   (-7, -6)  (-5, -3)]
 [ (-5, 0)  (5, 6)   (7, -9)   (-2, 5) ]]
```

(c) After separating, SWAPing and recombining the new matrix is

```
[[ (9, 7)  (-7, -9)  (5, -8)   (0, -5) ]
 [ (3, 8)  (1, -1)   (3, 0)    (6, 5)  ]
 [ (-3, 3) (5, -3)   (-6, -7)  (-9, 7) ]
 [ (4, 1)  (-1, -2)  (-3, -5)  (5, -2) ]]
```

6. (a) $C =$
```
[[ 3  5  4 ]
 [ 7  1  6 ]]
```

(b) $D =$
```
[[  1 -2  3  5  4  3  7 ]
 [  7  9  0 -1  3  5  1 ]
 [ -3  8  6  2  1  4  6 ]]
```

$E =$
```
[[  1  3  4  7 ]
 [  7  0  3  1 ]
 [ -3  6  1  6 ]]
```

7. (a) $A^* =$
```
[[ (5, -1)  (0, -4) ]
 [ (2, 3)   (6, 1)  ]
 [ (1, 0)   (3, -4) ]]
```

$B^T =$
```
[[ (-3, 1)  (0, 0)   (4, -3) ]
 [ (6, 0)   (2, -1)  (1, 1)  ]]
```

(b) $A^* + B =$
```
[[ (2, 0)   (6, -4) ]
 [ (2, 3)   (8, 0)  ]
 [ (5, -3)  (4, -3) ]]
```

$A + B^T =$
```
[[ (2, 2)   (2, -3)  (5, -3) ]
 [ (6, 4)   (8, -2)  (4, 5)  ]]
```

$AA^* =$
```
[[ (40, 0)   (22, -40) ]
 [ (22, 40)  (78, 0)   ]]
```

$B^TB =$
```
[[ (15, -3)  (-11, 7) ]
 [ (-11, 7)  (39, -2) ]]
```

$(2-3i)A =$
```
[[ (13, -13)  (-5, -12)  (2, -3)  ]
 [ (12, 8)    (9, -20)   (18, -1) ]]
```

8. $A^2 - 4A^* + 3A^T - I =$
```
[[ (12, -10)  (19, 13)  (5, -2)  ]
 [ (0, 17)    (41, 24)  (17, 33) ]
 [ (2, 17)    (9, 24)   (29, 13) ]]
```

9. (a) $A^4 - 8A^3 + 22A^2 - 40A + 25I = 0$

(b) $A\,[\,A^3 - 8A^2 + 22A - 40I\,] = -25I$ so $A^{-1} = \frac{-1}{25}\,[\,A^3 - 8A^2 + 22A - 40I\,]$

(c) $A^{-1} = \begin{bmatrix} -4.68 & -3.28 & -4.48 & -2 \\ .56 & .76 & .16 & 0 \\ 4.32 & 2.72 & 4.52 & 2 \\ 2.56 & 1.76 & 2.16 & 1 \end{bmatrix}$

10. (a) To three decimal places, $\lim_{n\to\infty} \{A^n\} = \begin{bmatrix} .269 & .269 & .269 & .269 \\ .212 & .212 & .212 & .212 \\ .205 & .205 & .205 & .205 \\ .315 & .315 & .315 & .315 \end{bmatrix}$

(b) To three decimal places, $\lim_{n\to\infty} \{A^n x\} = \begin{bmatrix} .269 & .212 & .205 & .315 \end{bmatrix}$

(c) Each column in $\lim_{n\to\infty} \{A^n\}$ is $\lim_{n\to\infty} \{A^n x\}$.

11. The conclusion is the same as in 10(c).

12. $u = [\,4\ \ -2\ \ 5\ \ -8\ \ 5\,]$ and $v = [\,-4\ \ 7\ \ 8\ \ 0\ \ -6\,]$

(a) $u \bullet v = -20$ (b) $u \bullet (u + v) = 114$ (c) $v \bullet (v - u) = 185$

(d) $\dfrac{u - v}{\| u - v \|} = 1$ (e) $16.093476934 < 24.4210694815$ (f) $20 < 148.69431731$

Activity Set 2.4

1. (a) $\det[\,1000A\,] = -1{,}536{,}000$

(b) Using cofactors along column 1,

$$\det A = \det \begin{bmatrix} 1 & 3 & 6 & 4 & 1 \\ -1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 4 & 1 \\ 1 & 1 & 1 & 1 & 3 \\ 1 & 3 & 6 & 4 & 1 \end{bmatrix} + 3 \det \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ -1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 4 & 1 \\ 1 & 1 & 1 & 1 & 3 \\ 1 & 3 & 6 & 4 & 1 \end{bmatrix}$$

$$- \det \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 3 & 6 & 4 & 1 \\ -1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 6 & 4 & 1 \\ 1 & 1 & 1 & 1 & 3 \end{bmatrix}$$

and since each of the three matrices on the right has two identical rows, det A = 0. Thus det [1000A] = 0

(c) When $detA$ is calculated as -1.536×10^{-12} due to round-off error,
det[1000A] = det[$10^3 A$] = $(10^3)^6 \det A = 10^{18}(-1.536 \times 10^{-12})$
= -1.536×10^6 = -1,536,000.

(d) With flag -54 clear, det A is calculated to be 0.

Activity Set 2.5

1. The RPN program is « π →NUM * COS ». The result is

$$\begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & -1 \end{bmatrix}.$$

2. The RPN program is « √ 2 + FLOOR ». The result is

$$\begin{bmatrix} 3 & 3 & 3 \\ 4 & 4 & 4 \\ 4 & 4 & 5 \end{bmatrix}.$$

3. The RPN program is « DUP 1 + SWAP / √ CEIL ». The result is

$$\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix}.$$

Activity Set 3.1

1. (a)

$$\begin{bmatrix} 4 & 1 & 3 & 6 \\ 8 & -2 & 4 & -8 \\ 8 & -6 & -2 & -36 \end{bmatrix} \xrightarrow{\substack{1,2\ \text{RSWP} \\ R_{21}(-.5) \\ R_{31}(-1)}} \begin{bmatrix} 8 & -2 & 4 & -8 \\ 0 & 2 & 1 & 10 \\ 0 & -4 & -6 & -28 \end{bmatrix}$$

$$\xrightarrow{2,3\ \text{RSWP}} \begin{bmatrix} 8 & -2 & 4 & -8 \\ 0 & -4 & -6 & -28 \\ 0 & 2 & 1 & 10 \end{bmatrix} \xrightarrow{R_{32}(.5)} \begin{bmatrix} 8 & -2 & 4 & -8 \\ 0 & -4 & -6 & -28 \\ 0 & 0 & -2 & -4 \end{bmatrix} = [\, U \mid b' \,].$$

$$\xrightarrow{\text{BACK}} \begin{bmatrix} -1 & 4 & 2 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 3 & 2 & -2 & 2 & -5 \\ 6 & 2 & -1 & -2 & -10 \\ -3 & 1 & 2.5 & 0 & 8 \\ 6 & 0 & 2 & 4 & 2 \end{bmatrix} \xrightarrow{1,2\ \text{RSWP}} \begin{bmatrix} 6 & 2 & -1 & -2 & -10 \\ 3 & 2 & -2 & 2 & -5 \\ -3 & 1 & 2.5 & 0 & 8 \\ 6 & 0 & 2 & 4 & 2 \end{bmatrix}$$

$$\xrightarrow[R_{41}(-1)]{R_{21}(-.5)\ R_{31}(.5)} \begin{bmatrix} 6 & 2 & -1 & -2 & -10 \\ 0 & 1 & -1.5 & 3 & 0 \\ 0 & 2 & 2 & -1 & 3 \\ 0 & -2 & 3 & 6 & 12 \end{bmatrix} \xrightarrow{2,3\ \text{RSWP}} \begin{bmatrix} 6 & 2 & -1 & -2 & -10 \\ 0 & 2 & -2 & 2 & -5 \\ 0 & 0 & -2.5 & 3.5 & -1.5 \\ 0 & 0 & 5 & 5 & 15 \end{bmatrix}$$

$$\xrightarrow[R_{42}(1)]{R_{32}(-.5)} \begin{bmatrix} 6 & 2 & -1 & -2 & -10 \\ 0 & 2 & -2 & 2 & -5 \\ 0 & 0 & -2.5 & 3.5 & -1.5 \\ 0 & 0 & 5 & 5 & 15 \end{bmatrix} \xrightarrow{3,4\ \text{RSWP}} \begin{bmatrix} 6 & 2 & -1 & -2 & -10 \\ 0 & 2 & -2 & 2 & -5 \\ 0 & 0 & 5 & 5 & 15 \\ 0 & 0 & -2.5 & 3.5 & -1.5 \end{bmatrix}$$

$$\xrightarrow{R_{43}(.5)} \begin{bmatrix} 6 & 2 & -1 & -2 & -10 \\ 0 & 2 & -2 & 2 & -5 \\ 0 & 0 & 5 & 5 & 15 \\ 0 & 0 & 0 & 6 & 6 \end{bmatrix} = [\,U \mid b'\,] \xrightarrow{\text{BACK}} \begin{bmatrix} -1 & 0 & 2 & 1 \end{bmatrix}$$

2. (a) Row Operations:
1, 2 RSWP
1, 1 ELIM
2, 4 RSWP
2, 2 ELIM
3, 3 ELIM
5 COL-
BACK $\rightarrow [1 \;\; 0 \;\; -2 \;\; 4]$

(b) Row Operations:
1, 4 RSWP
1, 1 ELIM
2, 3 RSWP
2, 2 ELIM
9 RND
3, 3 ELIM
5 COL-
BACK and
7RND $\rightarrow [6 \;\; -5 \;\; 1 \;\; 3]$

3. (a) Row Operations: 1, 3 RSWP; 1, 1 ELIM; 3, 4 ELIM

$$\rightarrow \begin{bmatrix} 3 & 6 & 9 & 2 & -5 \\ 0 & 1 & 1 & 4.\overline{3} & 6.\overline{6} \\ 0 & 0 & 0 & 3.\overline{3} & 6.\overline{6} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Thus x_3 is a free variable $x_4 = 2$. Also

$$\begin{aligned} x_2 &= 6.\overline{6} - x_3 - 4\overline{3}\,x_4 \\ &= 6.6 - x_3 - 8.\overline{6} \\ &= -2 - x_3\ . \end{aligned}$$

$$\begin{aligned} \text{And } 3x_1 &= -5 - 6x_2 - 9x_3 - 2x_4 \\ &= -5 - 6(-2 - x_3) - 9x_3 - 2(2) \\ &= -5 + 12 + 6x_3 - 9x_3 - 4 = 3 - 3x_3, \text{ so } x_1 = 1 - x_3\ . \end{aligned}$$

(b) 1, 3 RSWP
1, 1 ELIM
2, 3 RSWP
2, 3 ELIM
3, 4 RSWP
3, 6 ELIM

$$\rightarrow \begin{bmatrix} 6 & -9 & 0 & 11 & -19 & 3 & 0 \\ 0 & 0 & 5 & 1.\overline{6} & -3.\overline{3} & -8 & 6 \\ 0 & 0 & 0 & 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Then x_2, x_4 and x_5 are free variables and $x_6 = .5$.

$$\begin{aligned} \text{Also } 5x_3 &= 6 - 1.\overline{6}x_4 + 3.\overline{3}x_5 + 8x_6 \\ &= 6 - 1.\overline{6}x_4 + 3.\overline{3}x_5 + 4 \\ &= 10 - 1.\overline{6}x_4 + 3.\overline{3}x_5 \\ \text{so } x_3 &= 2 - .\overline{3}x_4 + .\overline{6}x_5 . \\ \text{Finally, } 6x_1 &= 9x_2 - 11x_4 + 19x_5 - 3x_6 \\ &= 9x_2 - 11x_4 + 19x_5 - 3/2 \\ \text{so that } x_1 &= 1.5x_6 - 1.83x_4 + 3.1\overline{6}x_5 - .25. \end{aligned}$$

Activity Set 3.2

1. (a) $P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ $L = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ .5 & .5 & 1 \end{bmatrix}$ $U = \begin{bmatrix} -2 & 4 & 5 \\ 0 & 3 & 8 \\ 0 & 0 & -4 \end{bmatrix}$

$y = [\ 10\ 21\ -12\ ]$ $x = [\ .5\ -1\ 3\ ]$

(b) $P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ $L = \begin{bmatrix} 1 & 0 & 0 \\ .\overline{3} & 1 & 0 \\ .5 & .5 & 1 \end{bmatrix}$ $U = \begin{bmatrix} -6 & -6 & -3 \\ 0 & 8 & 6 \\ 0 & 0 & -2 \end{bmatrix}$

$y = [\ 18\ 20\ -4\ ]$ $x = [\ -5\ 1\ 2\ ]$

(c) $P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ $L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ .\overline{3} & 1 & 0 & 0 \\ 0 & -.5 & 1 & 0 \\ -.\overline{3} & .5 & .5 & 1 \end{bmatrix}$

$U = \begin{bmatrix} -9 & 6 & -3 & 6 \\ 0 & 4 & -2 & 2 \\ 0 & 0 & 6 & -2 \\ 0 & 0 & 0 & 7 \end{bmatrix}$ $y = [\ -24\ -16\ 36\ -42\ ]$

$x = [\ -2\ 1\ 4\ -6\ ]$

(d)
$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ .\overline{3} & .\overline{3} & .\overline{3} & 1 & 0 \\ .\overline{3} & 0 & 0 & .5 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} -6 & -3 & -12 & -9 & 3 \\ 0 & 3 & -6 & 3 & 9 \\ 0 & 0 & 9 & -6 & -3 \\ 0 & 0 & 0 & 4 & -8 \\ 0 & 0 & 0 & 0 & 4 \end{bmatrix} \qquad y = \begin{bmatrix} 36 & 30 & -3 & -28 & 12 \end{bmatrix}$$

$$x = \begin{bmatrix} -4 & 2 & 0 & -1 & 3 \end{bmatrix}$$

2. (a)
$$P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad L = \begin{bmatrix} -9 & 0 & 0 \\ -1 & -1.\overline{6} & 0 \\ -4 & -.6 & -1.8 \end{bmatrix} \qquad U = \begin{bmatrix} 1 & -.\overline{6} & 0 \\ 0 & 1 & -1.2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$y = \begin{bmatrix} -2.\overline{3} & 5.6 & -3 \end{bmatrix} \qquad x = \begin{bmatrix} -1 & 2 & -3 \end{bmatrix}$$

(b)
$$P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad L = \begin{bmatrix} -7 & 0 & 0 & 0 \\ 2 & -4.42857142857 & 0 & 0 \\ 5 & -1.57142857143 & -9.8064516129 & 0 \\ 1 & -3.71428571429 & -1.45161290323 & -3.53289473684 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & -1.28571428571 & -.142857142857 & -.428571428571 \\ 0 & 1 & -.967741935484 & -.419354893871 \\ 0 & 0 & 1 & -.867105263158 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$y = \begin{bmatrix} -7.57142857143 & -.483870964452 & 3.96710526316 & -1 \end{bmatrix}$$

$$x = \begin{bmatrix} -5 & 2 & 3 & -1 \end{bmatrix}$$

(c)
$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$L = \begin{bmatrix} 9 & 0 & 0 & 0 & 0 \\ -7 & -11.\overline{6} & 0 & 0 & 0 \\ 7 & 3.\overline{6} & -8.5\overline{3} & 0 & 0 \\ -1 & -.\overline{6} & -4.9\overline{3} & -15.3214285714 & 0 \\ 5 & 7.\overline{3} & 4.9\overline{3} & 8.46428571429 & -9.41258741259 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & -.\overline{6} & -.5 & -1 & .\overline{8} \\ 0 & 1 & .9\overline{3} & .428571428571 & .0\overline{6} \\ 0 & 0 & 1 & -1.33928571429 & -.0625 \\ 0 & 0 & 0 & 1 & -4.97668997669 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$y = \begin{bmatrix} 1.\overline{8} & .238095238096 & 11.0267857143 & -6.62121212126 & .270430906347 \end{bmatrix}$$

$$x = \begin{bmatrix} -2.99577087667 & .800891530589 & 2.35624071312 & -6.48662704316 & .270430906347 \end{bmatrix}$$

Activity Set 3.3

1. The RREF's for the augmented matrices are:

   a. $\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 2 \end{bmatrix}$ b. $\begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$

2. The RREF's for the augmented matrices are:

   a. $\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 & 4 \end{bmatrix}$ b. $\begin{bmatrix} 1 & 0 & 0 & 0 & 6 \\ 0 & 1 & 0 & 0 & -5 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 3 \end{bmatrix}$

3. The RREF's for the augmented matrices are:

   a. $\begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ b. $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1.\overline{3} & .6 \\ 0 & 1 & 0 & -1.\overline{2} & 2.\overline{1} & .\overline{1} & .\overline{2} \\ 0 & 0 & 1 & .\overline{3} & -.\overline{6} & -1.\overline{3} & -1.\overline{3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

Activity Set 4.1

1. Consider the matrix $\begin{bmatrix} | & | & | & | & | \\ v_1 & v_2 & u_1 & u_2 & u_3 \\ | & | & | & | & | \end{bmatrix} = \begin{bmatrix} 8 & 3 & 1 & 2 & 16 \\ 9 & -1 & -5 & 11 & -7 \\ 7 & -8 & 4 & 23 & 3 \end{bmatrix}$

   The reduced row echelon form is

$$\begin{bmatrix} 1 & 0 & 0 & 1 & .961904761905 \\ 0 & 1 & 0 & -2 & 1.84761904762 \\ 0 & 0 & 1 & 0 & 2.7619047619 \end{bmatrix}.$$

Thus neither u_1 nor u_3 is a linear combination of v_1 and v_2; but $u_2 = v_1 - 2v_2$.

2. Consider the matrix

$$\begin{bmatrix} | & | & | & | & | & | \\ v_1 & v_2 & v_3 & u_1 & u_2 & u_3 \\ | & | & | & | & | & | \end{bmatrix} = \begin{bmatrix} -2 & 5 & 4 & -3 & -9 & 0 \\ 7 & -6 & -8 & 16 & 0 & -5 \\ 6 & -6 & 3 & 3 & 4 & 27 \\ -5 & 4 & 9 & -15 & -3 & 14 \end{bmatrix}.$$

The reduced row echelon form is

$$\begin{bmatrix} 1 & 0 & 0 & 2 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -1 & 0 & 3 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Thus $u_1 = 2v_1 + v_2 - v_3$, u_2 is not in Span $[\, v_1, v_2, v_3 \,]$, and $u_3 = v_1 - 2v_2 + 3v_3$.

3. $$\begin{bmatrix} 4 & 9 & 6 & -5 & -4 \\ -3 & 6 & 5 & 7 & 0 \\ -2 & -9 & 0 & -5 & -16 \\ 3 & -5 & 2 & -13 & -19 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & -2 & -1 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & -1 & -3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Thus $p(x) = -2\, r(x) + s(x) - t(x)$ and $q(x) = -r(x) + 2s(x) - 3t(x)$.

Activity Set 4.2

1. (a) Since the rows of A are the columns of A^T, we consider A^T:

The reduced echelon form of A^T is $\begin{bmatrix} 1 & 0 & 0 & 1/3 \\ 0 & 1 & 0 & -1/3 \\ 0 & 0 & 1 & -2/3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.

Since x_4 is a free variable, $A^T x = 0$ had infinitely many solutions. Thus the columns of A^T (i.e., the rows of A) are *dependent*.

All solutions to $A^T x = 0$ appear like $[\, -1/3x_4 \,,\, 1/3x_4 \,,\, 2/3x_4 \,,\, x_4 \,]$, and choosing $x_4 = 1$ we get $[\, -1/3 \,,\, 1/3 \,,\, 2/3 \,,\, 1 \,]$.

Thus the dependence among the rows of A is given by
$-\frac{1}{3}$ (row 1) + $\frac{1}{3}$ (row 2) + $\frac{2}{3}$ (row 3) + (row 4) = 0.

b. The reduced echelon form of A is

```
[[ 1  0  0  -.5 ]
 [ 0  1  0  .375 ]
 [ 0  0  1  .625 ]
 [ 0  0  0   0   ]]
```

Thus the columns of A are dependent and a general dependency is given by $A_4 = -.5A_1 + .375A_2 + .625A_3$.

c. The reduced row echelon form of C^T has a 1 in every column, so the columns of C^T (i.e., the rows of C) are independent.

d. We look to solve $Cx = 0$. The reduced row echelon form is

```
[[ 1  0  0  0  1 ]
 [ 0  1  0  0  -1 ]
 [ 0  0  1  0  -1 ]
 [ 0  0  0  1  1 ]]
```

So x_5 is a free variable and the columns of C are dependent. A general dependency relation is $A_5 = A_1 - A_2 - A_3 + A_4$.

e. The reduced row echelon form of E is

```
[[ 1  0  0  0  -2 ]
 [ 0  1  0  0  0 ]
 [ 0  0  1  0  2 ]
 [ 0  0  0  1  0 ]
 [ 0  0  0  0  0 ]]
```

So x_5 is a free variable and the columns of E are dependent. A dependency relation is $A_5 = -2A_1 + 2A_3$.

2. (a) Let A have u_1, u_2, u_3 and u_4 as its columns. The RREF of A is

```
[[ 1  0  0  -.5 ]
 [ 0  1  0  .5 ]
 [ 0  0  1  -.5 ]
 [ 0  0  0  0  ]]
```

Thus $\{ u_1, u_2, u_3, u_4 \}$ is a dependent set of vectors. In particular, $u_4 = -.5u_1 + .5u_2 - .5u_3$.

We delete u_4 and consider $\{ u_1, u_2, u_3 \}$. Let matrix B consist of the first three columns of A. The RREF of B is

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

Thus $\{ u_1, u_2, u_3 \}$ is independent and $W = \text{Span}[u_1, u_2, u_3]$.

(b) Let A have v_1, v_2, v_3 and v_4 as its columns. The reduced row echelon form of A is

$$\begin{bmatrix} 1 & 0 & 0 & -.8 \\ 0 & 1 & 0 & .7 \\ 0 & 0 & 1 & -1.3 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Thus $\{ v_1, v_2, v_3 \}$ is independent and $W = \text{Span}[v_1, v_2, v_3]$.

(c) Let A have w_1, w_2, w_3 and w_4 as its columns. The reduced row echelon form of A is

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 2 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Thus $\{ w_1, w_2 \}$ is independent and $W = \text{Span}[w_1, w_2]$.

Activity Set 4.3

1. (a) The reduced row echelon form of A is

$$E = \begin{bmatrix} 1 & 0 & 0 & .\overline{3} \\ 0 & 1 & 0 & -.\overline{3} \\ 0 & 0 & 1 & -.\overline{6} \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Rows 1, 2, 3 of E form a basis for the row space of A.

Columns 1, 2, 3 of A form a basis for the column space of A.

A basis for the null space of A: $[-.\overline{3} \quad .\overline{3} \quad \overline{6} \quad 1]$.

A basis for the left null space of A: $[.5 \quad -.375 \quad -.625 \quad 1]$.

(b) The reduced row echelon form of B is

$$E = \begin{bmatrix} 1 & 0 & 1 & 0 & 5 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Rows 1, 2, 3 of E form a basis for the row space of B.

Columns 1, 2, 4 of B form a basis for the column space of B.

A basis for the null space of B: [-1 -1 1 0 0] and [-5 0 0 2 1].

A basis for the left null space of B: [1 4 -1 1].

(c) The reduced row echelon form of C is

$$E = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Rows 1, 2, 3 of E form a basis for the row space of C.

Columns 1, 2, 4 of C form a basis for the column space of C.

A basis for the null space of C: [-1 -1 1 0 0] and [-1 2 0 -2 1].

A basis for the left null space of C: [.1 -.7 0 1].

(d) The reduced row echelon form of D is

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 & 1 \\ 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Rows 1, 2, 3 of E form a basis for the row space of D.

Columns 1, 2, 3 of D form a basis for the column space of D.

A basis for the null space of D: [0 -2 1 1 0] and [-1 -1 1 0 1].

A basis for the left null space of D:

[.586776859504 -8.59504132231 -.256198347107 1 0] and

[-1.38016528926 1.28925619835 .884297520661 0 1].

2. (a) Rows 1, 2, 4 of A form a basis for the row space of A.

   (b) Rows 1, 2, 4 of A form a basis for the row space of A.

Activity Set 4.4

1. (a) Let A have u_1, u_2 and u_3 as its columns and let C have v_1, v_2 and v_3 as its columns. Then A^T and C^T have the same reduced row echelon form

$$E = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

   Thus B and B' are independent sets of vectors and Span B = $RS(A^T)$ = $RS(C^T)$ = Span B'. Indeed, Span B = $CS(A)$ = $CS(B)$ = Span B'.

   (b) Consider the augmented matrix $[\,A|\ w\,]$. Its reduced row echelon form is

$$\begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

   which shows that $w = -2u_1 + u_2 + u_3$, so w is in W = Span B and $[\,w\,]_B = [\,-2\ \ 1\ \ 1\,]^T$.

   (c) The reduced row echelon form of the block matrix $[\,C|\ A\,]$ is

$$\begin{bmatrix} 1 & 0 & 0 & .4 & 1.2 & .4 \\ 0 & 1 & 1 & .2 & -.4 & .2 \\ 0 & 0 & 0 & -.4 & -.2 & .6 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad \text{Thus } P = \begin{bmatrix} .4 & 1.2 & .4 \\ .2 & -.4 & .2 \\ -.4 & -.2 & .6 \end{bmatrix}.$$

   (d) $[\,w\,]_{B'} = P[\,w\,]_B = [\,.8\ \ -.6\ \ 1.2\,]^T$. The reduced row echelon form of $[\,C|\ w\,]$ is

$$\begin{bmatrix} 1 & 0 & 0 & .8 \\ 0 & 1 & 0 & -.6 \\ 0 & 0 & 1 & 1.2 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

   The last column shows that $[\,w\,]_{B'} = [\,.8\ \ -.6\ \ 1.2\,]^T$.

2. (a) Let A have u_1, u_2, u_3, and u_4 as its columns and let C have v_1, v_2, v_3 and v_4 as its columns. Then A^T and C^T have the same reduced row echelon form

$$\begin{bmatrix} 1 & 0 & -.5 & 0 & 0 \\ 0 & 1 & 1.5 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Thus B and B' are independent sets of vectors and Span B = $RS(A^T)$ = $RS(C^T)$ = Span B'.

(b) The augmented matrix $[\, A|\ w\,]$ has reduced row echelon form

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 72 \\ 0 & 1 & 0 & 0 & 79 \\ 0 & 0 & 1 & 0 & -3 \\ 0 & 0 & 0 & 1 & -61 \end{bmatrix}.$$

This shows that $w = 72u_1 + 79u_2 - 3u_3 - 61u_4$; so w is in W = Span B and $[\, w\,]_B = [\, 72\ \ 79\ \ -3\ \ -61\,]^T$.

(c) The reduced row echelon form of the block matrix $[\, C|\ A\,]$ is

$$\begin{bmatrix} 1 & 0 & 0 & 0 & -136 & 57.5 & 70 & -89.5 \\ 0 & 1 & 0 & 0 & 49 & -20.5 & -25 & 32.5 \\ 0 & 0 & 1 & 0 & -36 & 15 & 19 & -24 \\ 0 & 0 & 0 & 1 & 12 & -5 & -6 & -8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Thus $$P = \begin{bmatrix} -136 & 57.5 & 70 & -89.5 \\ 49 & -20.5 & -25 & 32.5 \\ -36 & 15 & 19 & -24 \\ 12 & -5 & -6 & -8 \end{bmatrix}.$$

(d) $[\, w\,]_{B'} = P[\, w\,]_B = [\, 0\ \ 1\ \ 0\ \ -1\,]^T$. The reduced row echelon form of $[\, C|\ w\,]$ is

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The last column shows that $[\, w\,]_{B'} = [\, 0\ \ 1\ \ 0\ \ -1\,]$.

Activity Set 5.1

1. (a) $48 < (6.2449979984)(13.2287565553)$

   (b) $10.4403065089 < 4(7.21110255093)$

   (c) Answers will vary.

2. (a) $10.8627804912 < 6.2449979984 + 13.2287565553$

   (b) $7.87400787401 < 4 + 7.21110255093$

   (c) Answers will vary.

3. (a) $143 = 110 + 33$

   (b) $28 = 7 + 21$

4. (a) Let A have u_1, u_2, u_3, and u_4 as its columns, left-to-right. Then $A^TA = \text{diag}[\ 7\ \ 6\ \ 8\ \ 21\ ]$.

   (b) Let A have v_1, v_2, and v_3 as its columns, left-to-right. Then $A^TA = I_3$.

5. (i) The null space of A is the zero vector, so no basis exists.

   (ii) The reduced row echelon form of A is
   ```
   [[ 1  2  0  4 ]
    [ 0  0  1 -3 ]
    [ 0  0  0  0 ]]
   ```
   .

   Thus, a basis for $RS(A)$ is $\{[\ 1\ \ 2\ \ 0\ \ 4\ ], [\ 0\ \ 0\ \ 1\ \ -3\ ]\}$ and a basis for $NS(A)$ is $\{[\ -2\ \ 1\ \ 0\ \ 0\ ], [-4\ \ 0\ \ 3\ \ 1\ ]\}$.

   $RS(A)$ is orthogonal to $NS(A)$ because

   ```
   [[ 1  2  0  4 ]     [[ -2  -4 ]     [[ 0  0 ]
    [ 0  0  1 -3 ]] *  [  1   0 ]  =   [ 0  0 ]].
                       [  0   3 ]
                       [  0   1 ]]
   ```

   A basis for $CS(A)$ is $\{[\ 1\ \ 1\ \ -2\ ], [-1\ \ 0\ \ 0\ ]\}$

   The reduced row echelon form of A^T is

   ```
   [[ 1  0  0 ]
    [ 0  1 -2 ]
    [ 0  0  0 ]
    [ 0  0  0 ]]
   ```
   .

A basis for $NS(A^T)$ is {[0 2 1]}. $CS(A)$ is orthogonal to $NS(A^T)$ because

```
[[ 1  1 -2 ]       [[ 0 ]       [[ 0 ]
 [ -1 0  0 ]] * [ 2 ]    =  [ 0 ]] .
                [ 1 ]]
```

(iii) A basis for $RS(A)$ is {[1 -2 0 4 0], [0 0 1 3 0], [0 0 0 0 1]}.

A basis for $NS(A)$ is {[2 1 0 0 0], [-4 0 -3 1 0]}

$RS(A)$ is orthogonal to $NS(A)$ because

```
                          [[ 2 -4 ]
[[ 1 -2 0 4 0 ]   *       [ 1  0 ]     =   [[ 0 0 ]
 [ 0 0  1 3 0 ]           [ 0 -3 ]          [ 0 0 ]
 [ 0 0  0 0 1 ]]          [ 0  1 ]          [ 0 0 ]]
                          [ 0  0 ]]
```

A basis for $CS(A)$ is {[1 2 -1 -3], [0 1 1 2], [0 0 1 2]}.

A basis for $NS(A^T)$ is {[1 0 -2 1]}. $CS(A)$ is orthogonal to $NS(A^T)$ because

```
[[ 1 2 -1 -3 ]       [[ 1 ]       [[ 0 ]
 [ 0 1  1 2 ]   *    [ 0 ]    =   [ 0 ] .
 [ 0 0  1 2 ]]       [ -2 ]       [ 0 ]]
                     [ 1 ]]
```

(iv) A basis for $RS(A)$ is {[1 2 0 0 1 0], [0 0 1 0 0 0], [0 0 0 1 -1 0], [0 0 0 0 0 1]}.

A basis for $NS(A)$ is {[-2 1 0 0 0 0], [-1 0 0 1 1 0]}.

$RS(A)$ is orthogonal to $NS(A)$ because

```
                          [[ -2 -1 ]
[[ 1 2 0 0 1 0 ]          [ 1  0 ]        [[ 0 0 ]
 [ 0 0 1 0 0 0 ]    *     [ 0  0 ]    =    [ 0 0 ] .
 [ 0 0 0 1 -1 0 ]]        [ 0  1 ]         [ 0 0 ]]
                          [ 0  1 ]
                          [ 0  0 ]]
```

A basis for $CS(A)$ is {[1 0 1 2 1], [2 1 2 3 1], [-1 -1 0 -2 -2], [2 0 3 2 2]}.

A basis for $NS(A^T)$ is {[-9 3 4 2 1]}. $CS(A)$ is orthogonal to $NS(A^T)$ because

```
[[ 1  0  1  2  1 ]        [[ -9 ]        [[ 0 ]
 [ 2  1  2  3  1 ]    *    [  3 ]    =    [ 0 ]  .
 [ -1 -1  0 -2 -2 ]        [  4 ]         [ 0 ]
 [ 2  0  3  2  2 ]]        [  2 ]         [ 0 ]]
                           [  1 ]]
```

Activity Set 5.2

1. (a) Seed the random number generator with 1 (1 RDZ) to produce

```
[[  7 -9 -8 ]
 [ -5  8 -1 ]  .
 [  0  5  3 ]
 [ -3 -7  7 ]]
```

   (b) $\text{proj}_v u$ = [3.3698630137 -2.99543378995 -1.87214611872 2.62100456621]

   (c) $\text{proj}_w u$ = [4.68292682927 .585365853659 -1.75609756098 -4.09756097561]

2. (a) Seed the random number generator with 2 to produce

```
[[  4 -2  5  ]
 [ -8  5 -4  ]  .
 [  7  8  0  ]
 [ -6 -4  0  ]]
```

   (b) q_1 = [.311399577664 -.622799155328 .544949260912 -.467099366496].

   q_2 = [-.273777516016 .646186669899 .655153182431 -.279755191039].

   (c) $\text{proj}_w x_3$ = [2.34302222747 -5.07599787749 -.38417546136 -.78485938328].

3. (a) Seed the random number generator with 3 to produce

```
[[  1 -5  2 -1  ]
 [  2 -7  4  1  ]
 [ -7  9  1  3  ]  .
 [  5 -8 -9  8  ]
 [  3  3  9 -6  ]]
```

   (b) Rounding to six decimal places:

   q_1 = [.106600 .213201 -.746203 .533002 .31980]

   q_2 = [-.408126 -.486755 1.248090E-3 -.173485 .752598]

   q_1 = [-.399263 -.225475 .365122 .789454 -.180902]

(c) $\text{proj}_w x_3 =$ [1.576232 -.782662 -2.477077 -9.772820 5.504552]

4. (a)
$$\begin{bmatrix} .25 \\ -.5 \\ .25 \\ -.75 \\ .25 \end{bmatrix}$$

(b)
$$\begin{bmatrix} .875 & .25 & -.125 & .375 & -.125 \\ .25 & .5 & .25 & -.75 & .25 \\ -.125 & .25 & .875 & .375 & -.125 \\ .375 & -.75 & .375 & -.125 & .375 \\ -1.25 & .25 & -.125 & .375 & .875 \end{bmatrix}$$

(c) H is symmetric

(d) $H^{-1} = H$

Activity Set 5.3

1. (a) The RREF of A is
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

(b) $Q =$
$$\begin{bmatrix} .5 & .5 & -.5 \\ .5 & -.5 & .5 \\ .5 & -.5 & -.5 \\ .5 & .5 & .5 \end{bmatrix}$$
$R =$
$$\begin{bmatrix} 2 & 3 & 2 \\ 0 & 5 & -2 \\ 0 & 0 & 4 \end{bmatrix}$$

2. $Q =$
$$\begin{bmatrix} .2 & -.4 & -.8 \\ .4 & .2 & .4 \\ .4 & -.8 & .4 \\ .8 & .4 & .2 \end{bmatrix}$$
$R =$
$$\begin{bmatrix} 5 & -2 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & 2 \end{bmatrix}$$

Activity Set 5.4

1. (a) The RREF of the augmented matrix is I_4.

(b) $A^TA =$
$$\begin{bmatrix} 26 & -2 & 12 \\ -2 & 20 & -12 \\ 12 & -12 & 12 \end{bmatrix}$$
and $A^Tb =$ [44 -15 29] and $x =$ [$-1.\overline{6}$ $3.8\overline{3}$ $7.91\overline{6}$]

2. (a)

x	-2.2	-1	.5	1.5	3
y	.361	2.718	3.791	2.733	1.245

(b) $P_3(x) = 3.56125138045 + .168633453671x - .476477181213x^2 + .0530615957649x^3$

(c) $P_4(x) = .381750 + .347647x - .808918x^2 + .008864x^3 + .048199x^4$
(rounded to six decimal places)

3. (a) The condition number of A is 3034650.03175.

   (b) Both polynomials are $c_0 + c_1 + c_2x^2 + c_3x^3$ where

   [c_0 c_1 c_2 c_3] = [-.1$\overline{3}$ .810582010582 -.109126984127 .0231481482],

   (c) Applying the RREF command we obtain

   [-.1$\overline{3}$ .81058201058 -.109126984126 .0231481481481].

   Using $x = (A^TA)^{-1}(A^Tb)$ we obtain

   [-.133333334713 .8105820101475 -.109126983096 .0231481483408].

Activity Set 6.1

1. (a) Seed the random number generator with 1 and generate

$$A = \begin{bmatrix} .7 & -9 & -8 \\ -5 & 8 & -1 \\ 0 & 5 & 3 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} -3 & -7 & 7 \\ 1 & -2 & -5 \\ -2 & 9 & -7 \end{bmatrix}$$

   (b) Trace A = 18, Trace B = -12 and Trace $(A + B)$ = 6

   (c)
$$A = \begin{bmatrix} 5 & 0 & 3 & 1 \\ 3 & 6 & -3 & 5 \\ -6 & -9 & 4 & -1 \\ -3 & 5 & -5 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 3 & 9 & 0 & -3 \\ 6 & -3 & 5 & 0 \\ 1 & 3 & 4 & -4 \\ 0 & 5 & 3 & -9 \end{bmatrix}$$

   Trace A = 15, Trace B = -5 and Trace $(A + B)$ = 10

   (d) Trace $(A + B)$ = Trace A + Trace B

   Proof: Trace $(A + B) = \sum_{i=1}^{n} (a_{ii} + b_{ii}) = \sum_{i=1}^{n} a_{ii} + \sum_{i=1}^{n} b_{ii}$ = Trace A + Trace B

2. (a) Seed the random number generator with 2 and generate

$$A = \begin{bmatrix} 4 & -2 & 5 & -8 \\ 5 & -4 & 7 & 8 \\ -0 & -6 & -4 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 8 & -5 & -2 \\ 5 & 6 & 0 \\ 2 & 3 & 0 \\ 4 & 0 & -9 \end{bmatrix}$$

(b) Trace $(A\ B)$ = -28 = Trace $(B\ A)$

(c) A =
```
[[  0 -8  3 -2  1 ]
 [  4  0  8 -2  3 ]
 [  0  2 -7 -6  3 ]]
```
and B =
```
[[ -6  2 -3 ]
 [ -8  6  1 ]
 [ -7 -7  8 ]
 [  0 -2  8 ]
 [  6  0  8 ]]
```

Trace $(A\ B)$ = -73 = Trace $(B\ A)$

(d) Trace $(A\ B)$ = Trace $(B\ A)$

Proof: Let A be $m \times n$ and B be $n \times m$

Trace $(A\ B) = \sum_{i=1}^{m} (A\ B)_{ii} = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} A_{ij} B_{ji} \right)$ and

Trace $(B\ A) = \sum_{j=1}^{n} (B\ A)_{jj} = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} B_{ji} A_{ij} \right)$. These are clearly the same.

3. (a) Seed the random number generator with 3 and generate

A =
```
[[  1 -5  2 ]
 [ -1  2 -7 ]
 [  4  1 -7 ]]
```

$\det(\lambda I - A) = \lambda^3 + 4\lambda^2 - 25\lambda - 150 = \det(\lambda I - A^T)$

(b) A =
```
[[  9  1  3  5 ]
 [ -8 -9  8  3 ]
 [  3  9 -6 -7 ]
 [ -7 -7 -2 -7 ]]
```

$\det(\lambda I - A) = \lambda^4 + 13\lambda^3 - 70\lambda^2 - 1009\lambda + 3240 = \det(\lambda I - A^T)$

(c) A =
```
[[ -1  0 -5 -8  7 ]
 [  7 -1 -9  6  7 ]
 [  5  4  9  0 -9 ]
 [  9 -6 -5 -9  8 ]
 [  0  4  1  7 -5 ]]
```

$\det(\lambda I - A) = \lambda^5 + 7\lambda^4 + 24\lambda^3 - 1021\lambda^2 + 6257\lambda + 15082 = \det(\lambda I - A^T)$

(d) - (e) A and A^T have the same characteristic polynomial.

Proof: $\det(\lambda I - A^T) = \det(\lambda I^T - A^T) = \det(\lambda I - A)^T = \det(\lambda I - A)$

4. Seed the random number generator with 4 and generate

$$A = \begin{bmatrix} -8 & 8 & 9 \\ -4 & -9 & 0 \\ 1 & 4 & -4 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 8 & 6 & -2 \\ 1 & 6 & 1 & 0 \\ 6 & 6 & -9 & -4 \\ 0 & -4 & -5 & 4 \end{bmatrix} \text{ and } C = \begin{bmatrix} -4 & 7 & -8 & -5 & 0 \\ 4 & -4 & -6 & 9 & 4 \\ 8 & 1 & 2 & 7 & -2 \\ -6 & 9 & -2 & -2 & 5 \\ -3 & 1 & 4 & 4 & -8 \end{bmatrix}$$

(a) Det A = -479, Trace A = -21, $\det(\lambda I - A) = \lambda^3 + 21\lambda^2 + 163\lambda + 479$.
Det B = -684, Trace B = 3, $\det(\lambda I - B) = \lambda^4 - 3\lambda^3 - 134\lambda^2 + 696\lambda - 684$.
Det C = 87,902, Trace C = -16,
$\det(\lambda I - C) = \lambda^5 + 16\lambda^4 + 5\lambda^3 + 468\lambda^2 - 681\lambda - 87{,}902$.

(b) For an $n \times n$ matrix A, the constant term in $\det(\lambda I - A)$ is $(-1)^n \det A$; the coefficient of λ^{n-1} is -(Trace A).

5. (a) $\det [\lambda I - A_n(1)] = \lambda^2 - 2\lambda,\ \lambda^3 - 3\lambda^2, \lambda^4 - 4\lambda^3, \lambda^5 - 5\lambda^4$ for n = 2, 3, 4, 5.

(b) $\det [\lambda I - A_n(1)] = \lambda^{n-1}(\lambda - n)$

(c) 0 (of multiplicity $n - 1$) and n

6. (a) (i) $C = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & 3 & -5 \end{bmatrix}$, $\det [\lambda I - C] = \lambda^3 + 5\lambda^2 - 3\lambda + 2$

(ii) $C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -7 & 5 & -2 & 6 \end{bmatrix}$, $\det [\lambda I - C] = \lambda^4 - 6\lambda^3 + 2\lambda^2 - 5\lambda + 7$

(iii) $C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ -1 & -2 & -3 & -4 & -5 \end{bmatrix}$, $\det [\lambda I - C] = \lambda^5 + 5\lambda^4 + 4\lambda^3 + 3\lambda^2 + 2\lambda + 1$

(b) For $p(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + \ldots + c_1\lambda + c_0$ the characteristic polynomial of its companion matrix is $p(\lambda)$.

7. (a) Seed the random number generator with 7 and generate A and B.

$$AB = \begin{bmatrix} -30 & -81 & 94 \\ 3 & 18 & 30 \\ 7 & -13 & -36 \end{bmatrix} \quad \text{and} \quad BA = \begin{bmatrix} 28 & 22 & 106 \\ 13 & -42 & -35 \\ -13 & -15 & -34 \end{bmatrix}$$

have the same characteristic polynomial $p(\lambda) = \lambda^3 + 48\lambda^2 - 133\lambda + 33{,}528$.

(b)
$$AB = \begin{bmatrix} 96 & 63 & 57 & -124 \\ -36 & -45 & -63 & 68 \\ -52 & 96 & 16 & -27 \\ 54 & 49 & 9 & -71 \end{bmatrix} \quad \text{and} \quad BA = \begin{bmatrix} -21 & 23 & -65 & -2 \\ 77 & -5 & 63 & 74 \\ 12 & -76 & 60 & 8 \\ -24 & -84 & 74 & -38 \end{bmatrix}$$

have the same $p(\lambda) = \lambda^4 + 4\lambda^3 + 6{,}626\lambda^2 + 116{,}180\lambda + 1{,}523{,}808$.

(c)
$$AB = \begin{bmatrix} -28 & 0 & -34 & 40 & -77 \\ -72 & 46 & -104 & -70 & -38 \\ -36 & 5 & 32 & 39 & -3 \\ -30 & 2 & 69 & 68 & -59 \\ -16 & 72 & -88 & -56 & 4 \end{bmatrix} \quad \text{and} \quad BA = \begin{bmatrix} 137 & -9 & 10 & 57 & 17 \\ -5 & 26 & 78 & 23 & 17 \\ -139 & -45 & -84 & -11 & -27 \\ -98 & -74 & -118 & -2 & -10 \\ 131 & -46 & -34 & -17 & 45 \end{bmatrix}$$

have $p(\lambda) = \lambda^5 - 122\lambda^4 + 1{,}489\lambda^3 - 1{,}176{,}076\lambda^2 + 26{,}827{,}552\lambda - 294{,}808{,}320$.

(d) AB and BA have the same characteristic polynomial, thus the same determinant, trace and eigenvalues.

8. (a) Seed the random number generator with 8 and generate

$$A = \begin{bmatrix} -6 & 0 & -7 \\ 6 & -7 & 2 \\ -9 & -6 & 2 \end{bmatrix}.$$

(b) $p(x) = x^3 + 11x^2 - 35x - 705$ and $p(A) = 0$.

(c) For
$$A = \begin{bmatrix} -4 & -6 & -8 & 0 \\ 3 & 4 & -3 & -8 \\ -8 & -4 & 1 & 2 \\ 2 & 2 & 7 & 2 \end{bmatrix}$$

$p(x) = x^4 - 3x^3 - 70x^2 + 174x + 720$ and $p(A) = 0$.

(d) For
$$A = \begin{bmatrix} 6 & -5 & 0 & 3 & -2 \\ -2 & 6 & -4 & -1 & -2 \\ 0 & -9 & -4 & -5 & 4 \\ -4 & 1 & 0 & 4 & -3 \\ 5 & -9 & -6 & -2 & 0 \end{bmatrix}$$

$p(x) = x^5 - 12x^4 + 69x^3 - 24x^2 + 7x + 5288$ and $p(A) = 0$.

(e) Every $n \times n$ matrix satisfies its characteristic polynomial. This is known as the Cayley-Hamilton Theorem.

Activity Set 6.2

1. Adding a multiple of one row to another row changes the characteristic polynomial and the eigenvalues.

2. (a) CHAR returns the vector [1 -21 144 -421 -4623] of coefficients of $p(\lambda)$; PROOT and OBJ→ show the eigenvlaues to be

   4: (-3.63770276026, 0)
   3: (5.0202378832, 7.86503318779)
   2: (5.0202378832, -7.86503318779)
   1: (14.5972269939, 0)

   EGVL and OBJ→ return identical results.

3. (a) Seed the random number generator with 3 and generate

$$A = \begin{bmatrix} 1 & -5 & 2 \\ -1 & 2 & -7 \\ 4 & 1 & -7 \end{bmatrix}.$$

   det A = 150 and trace A = -4. the sum of the eigenvlaues is $-3.\overline{9}$ and the product of the eigenvlaues is 150.

   (e) The sum of the λ-values is the trace and the product of the λ-values is the determinant.

4. (a) The eigenvalues of A are 10, 2 and ± 2.82842712475. The eigenspace associated with $\lambda = 10$ is spanned by the vector [1 1 1 1].

   (b) The eigenvalues of B are 1, 2, 3, 4 and 5. The eigenspace associated with $\lambda = 5$ is spanned by the vector [1 -1 1 0 0].

   (c) The eignevalues of C are 3.73205080756, 4, ± i and .267949192431. The eigenspace associated with $\lambda = 4$ is spanned by [-2 3 1 -3 1].

   (d) The eigenvalues of D are 1, 2, 2, 3 and ± i. The eigenspace associated with $\lambda = 3$ is spanned by [-2 4 -6 -1 1 0].

Activity Set 6.3

1. (a) $B = P^{-1}AP =$
$$\begin{bmatrix} -1 & 4 & 2 & 8 \\ 0 & 1 & 0 & -6 \\ -4 & 4 & 3 & 2 \\ -0 & 0 & 0 & 5 \end{bmatrix}$$

(b) CHAR returns [1 -8 22 -40 25] for A and B.

(c) The λ-values are 1, 5 and $1 \pm 2i$; trace = 8, det = 25 and rank = 4.

2. (a) A is diagonalizable. Its eigenvalues are λ = -1, 1, 1 and 2. For D = diag [-1 1 1 2], a diagonalizing matrix P is

$$P = \begin{bmatrix} 0 & -1 & -1 & -1 \\ 0 & 1 & -1 & -2 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

(b) A is defective. Its eigenvalues are λ = -2, -1, 1 and 1, but dim $NS[A - I]$ = 1.

(c) A is diagonalizable. Its eigenvalues are λ = 2, 2, 4 and 6. For D = diag [2 2 4 6], a diagonalizing matrix P is

$$P = \begin{bmatrix} 1 & 1 & -.\overline{6} & -2 \\ 0 & 1 & .\overline{3} & 1 \\ 1 & 0 & -1 & -1 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

(d) A is defective. Its eigenvlaues are λ = -1, 0, 1, 1 and 3. Dim $NS[A - I]$ = 1.

(e) A is diagonalizable. Its eigenvalues are λ = 4, 4, 3, 2, 2 and 1. For D = diag [4 4 3 2 2 1] a diagonalizing matrix P is

$$P = \begin{bmatrix} -1 & 2 & 0 & -3 & 0 & 1 \\ -1 & 2 & -1 & -4 & 0 & 1 \\ 0 & 1 & 0 & -1 & 1 & 0 \\ 1 & -1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Activity Set 6.4

1. (a) The eigenvalues of A are $\lambda = 1, 5, 5, 5$. A basis for the eigenspace of $\lambda = 1$ is X1 = [1 1 1 1], normalized to $Q1$ = [.5 .5 .5 .5]. A basis for the eigenspace of $\lambda = 5$ consists of the vectors X2 = [-1 1 0 0], X3 = [-1 0 1 0] and X4 = [-1 0 0 1]. Convert to an orthonormal basis using the modified Gram-Schmidt algorithm via program GS to obtain

$Q2$ = [-.707106781 188 .707106781 188 0 0]
$Q3$ = [-.408248290463 -.408248290466 .816496580929 0]
$Q4$ = [-.288675134593 -.288675134595 -.288675134595 .866025403784]

Then with Q = [$Q1$ $Q2$ $Q3$ $Q4$] (in column form), Q^TAQ = diag [1 5 5 5].

(b) The eigenvalues of A are $\lambda = 1, 4, 4, 4$. A basis for the eigenspace of $\lambda = 1$ is $X1$ = [-1 -1 1 0], normalized to
$Q1$ = [-.577350269189 -.577350269189 .577350269189 0].

A basis for the eigenspace of $\lambda = 4$ consists of the vectors
$X2$ = [-1 1 0 0], $X3$ = [1 0 1 0] and $X4$ = [0 0 0 1]. Apply program GS to obtain

$Q2$ = [-.707106781188 .707106781188 0 0]
$Q3$ = [.408248290463 .408248290466 .816496580929 0]
$Q4$ = [0 0 0 1]

Then with Q = [$Q1$ $Q2$ $Q3$ $Q4$] (in column form), Q^TAQ = diag [1 4 4 4].

(c) The eigenvalues are $\lambda = -6, 2, 4, 4$. A basis for the eigenspace of $\lambda = -6$ is $X1$ = [-1 -1 -1 1], normalized to $Q1$ = [-.5 -.5 -.5 .5]. A basis for the eigenspace of $\lambda = 2$ is $X2$ = [1 -1 1 1] normalized to $Q2$ = [.5 -.5 .5 .5]. A basis for the eigenspace of $\lambda = 4$ consists of $X3$ = [-1 0 1 0] and $X4$ = [0 1 0 1], already orthogonal. Normalizing we obtain

$Q3$ = [-.707106781188 0 .707106781188 0]
$Q4$ = [0 0707106781188 0 .707106781188]

Then with Q = [$Q1$ $Q2$ $Q3$ $Q4$] (in column form) we obtain
Q^TAQ = diag [-6 2 4 4].

(d) The eigenvalues are $\lambda = -2, -2, 6, 6, 6$. A basis for the eigenspace of $\lambda = 2$ consists of the vectors $X1 =$ [0 1 1 0 0] and $X2 =$ [-2 2 0 1 1]. Apply program GS to obtain (we round to six decimal places)

$Q3 =$ [0 .707107 .707107 0 0]
$Q4 =$ [-.707107 .353553 -.353553 .353553 .353553]

A basis for the eigenspace of $\lambda = 6$ consists of the vectors $X3 =$ [-1 -1 1 0 0], $X4 =$ [1 0 0 2 0] and $X5 =$ [1 0 0 0 2]. Apply program GS to obtain

$Q3 =$ [-.577350 -.577350 .577350 0 0]
$Q4 =$ [.3086067 -.154303 .154303 .925820 0]
$Q5 =$ [.267261 -.133630 .133630 -.133630 .935414]

Then with $Q =$ [$Q1$ $Q2$ $Q3$ $Q4$ $Q5$] (in column form)
$Q^TAQ =$ diag [-2 -2 6 6 6].

2. (a) A spectral decomposition for matrix A is

$$\begin{bmatrix} .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \end{bmatrix} + 5 \begin{bmatrix} .5 & -.5 & 0 & 0 \\ -.5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 5 \begin{bmatrix} .1\overline{6} & .1\overline{6} & .\overline{3} & 0 \\ .1\overline{6} & .1\overline{6} & -.\overline{3} & 0 \\ -.\overline{3} & -.\overline{3} & .\overline{6} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$+ 5 \begin{bmatrix} .08\overline{3} & .08\overline{3} & .08\overline{3} & -.25 \\ .08\overline{3} & .08\overline{3} & .08\overline{3} & -.25 \\ -.08\overline{3} & .08\overline{3} & .08\overline{3} & -.25 \\ -.25 & -.25 & -.25 & .75 \end{bmatrix}$$

(b) A spectral decomposition for matrix A is

$$\begin{bmatrix} .\overline{3} & .\overline{3} & -.\overline{3} & 0 \\ .\overline{3} & .\overline{3} & -.\overline{3} & 0 \\ -.\overline{3} & -.\overline{3} & .\overline{3} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 4 \begin{bmatrix} .5 & -.5 & 0 & 0 \\ -.5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 4 \begin{bmatrix} .1\overline{6} & .1\overline{6} & .\overline{3} & 0 \\ .1\overline{6} & .1\overline{6} & .\overline{3} & 0 \\ .\overline{3} & .\overline{3} & .\overline{6} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$+ 4 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(c) A spectral decomposition for matrix A is

$$-6\begin{bmatrix} .25 & .25 & .25 & -.25 \\ .25 & .25 & .25 & -.25 \\ .25 & .25 & .25 & -.25 \\ -.25 & -.25 & -.25 & .25 \end{bmatrix} + 2\begin{bmatrix} .25 & -.25 & .25 & .25 \\ -.25 & .25 & -.25 & -.25 \\ .25 & -.25 & .25 & -.25 \\ .25 & -.25 & .25 & -.25 \end{bmatrix} + 4\begin{bmatrix} .5 & 0 & -.5 & 0 \\ 0 & 0 & 0 & 0 \\ -.5 & 0 & .5 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$+ 4\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & .5 & 0 & .5 \\ 0 & 0 & 0 & 0 \\ 0 & .5 & 0 & .5 \end{bmatrix}$$

(d) A spectral decomposition for matrix A (we round to six decimal digits) is

$$-2\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & .5 & .5 & 0 & 0 \\ 0 & .5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} - 2\begin{bmatrix} .5 & -.25 & .25 & -.25 & -.25 \\ -.25 & .125 & -.125 & .125 & .125 \\ .25 & -.125 & .125 & -.125 & -.125 \\ -.25 & .125 & -.125 & .125 & .125 \\ -.25 & .125 & -.125 & .125 & .125 \end{bmatrix}$$

$$+ 6\begin{bmatrix} .\overline{3} & .\overline{3} & -.\overline{3} & 0 & 0 \\ .\overline{3} & .\overline{3} & -.\overline{3} & 0 & 0 \\ -.\overline{3} & -.\overline{3} & .\overline{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$+ 6\begin{bmatrix} .095238 & -.047619 & .047619 & .285714 & 0 \\ -.047619 & .023809 & -.238095 & -.142857 & 0 \\ .0476199 & -.023809 & .023809 & .142857 & 0 \\ .285714 & -.142857 & .142857 & .857143 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$+ 6\begin{bmatrix} .071429 & -.035714 & .035714 & -.035714 & .25 \\ -.035714 & .017857 & -.017857 & .017857 & -.125 \\ .035714 & -.017857 & .017857 & -.017857 & .125 \\ -.035714 & .017857 & -.017857 & .017857 & -.125 \\ .25 & -.125 & .125 & -.125 & .875 \end{bmatrix}$$

Activity Set 6.5

1. (a) Not positive definite

   (b) Positive definite

   (c) Not positive definite

   (d) Positive definite

(e) Not positive definite

(f) Positive definite

2. (b) The solution to $Ly = b$ is y = [.707106781188 2.04124145232 2.30940107676]. The solution to $L^T x = y$ is x = [2 1 2].

(d) The solution to $Ly = b$ is y = [.577350269189 1.80739222822 6.71317113328 17.8978583435]. The solution to $L^T x = y$ is x = [37 29 50 31].

(f) The solution to $Ly = b$ is y = [.5 1.70084012854 2.27093435774 3.22711724525 1.13389341893]. The solution to $L^T x = y$ is x = [-3 -2 4 0 3].

3. The Cholesky factor is

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

The solution to $Ly = b$ is y = [1 2 3 4 5]. The solution to $L^T x = y$ is x = [15 14 12 9 5].

Activity Set 6.6

1. (a) The non-zero singular values of A are
$\sigma_1 = 22.2760011085$, $\sigma_2 = 15.655141875$, $\sigma_3 = 6.4572678035$.

Their squares are
$\sigma_1^2 = 496.220225386$, $\sigma_2^2 = 245.083467126$ $\sigma_3^2 = 41.6963074861$.

(b) The non-zero eigenvalues of $A^T A$ and AA^T are
$\lambda_1 = 496.220225387$, $\lambda_2 = 245.083467127$, $\lambda_3 = 41.6963074861$.

2. All three methods produce x = [-1 0 2].

3. Both methods produce
x = [-.276893387225 .0697403149571 -.187791269405 .128765792031].

4. Using a Cholesky factorization, we find the vector of coefficients in $p(x)$ to be
[-10.6666667605 12.7609428657 -4.14534963083 .389570640143].

Using the LSQ command we obtain
[-10.6666666667 12.7609427609 -4.1453495999 .389570637505].

The norms of these vectors agree to 6 decimal places.

Activity Set 7.1

1. (a) Not diagonally dominant.

   (b) JTEST fails to execute because the diagonal part of A is not invertible. After swapping rows four and five JTEST shows that both the row-sum norm and column-sum norm are ≥ 1.

   (c) STEST shows that both norms are ≥ 1.

   (d) Swap rows one and three, then rows two and five.

   (e) After 25 iterations, x = [-8.762155 6.560882 -8.564386 6.754029 2.265541].

2. (a) The Gauss-Seidel iteration matrix has row-sum norm < 1 but A is not diagonally dominant.

   (b) The Jacobi iteration matrix has column-sum norm < 1.

3. (a) A is column diagonally dominant

   (b) After 17 iterations,
   x = [3.10325718 -2.39543980 9.21205220 -5.18175900 5.30456021].

   (c) After 13 iterations,
   x = [3.10325731 -2.39543974 9.21205212 -5.18175896 5.30456026].

   (d) To full machine precision, the solution is
   x = [3.10325732899 -2.39543973941 9.21205211726 -5.18175895765 5.30456026059]

4. (a) The coefficient matrix is not diagonally dominant and the JTEST fails. The STEST succeeds with row-sum norm < 1.

   (b) After 55 iterations,
   x = [3.66666598 -2.33333231 -1.00000102 -8.66666590 8.33333295].

5. (a) All of the tests for convergence fail.

   (b) After 16 iterations, the Gauss-Seidel procedure converges to
   $x = [\ -.846168\ \ 3.153840\ \ -.692309\ \ -.076918\ \ -2.538454\ \ -5.538454\ ]$.

   (c) To full machine precision, the solution is
   $x = [\ -.846153846154\ \ 3.15384615385\ \ -.692307692308\ \ -.076923076923\ \ -2.53846153846\ ]$.

Activity Set 7.2

1. (a) The dominant eigenvalue is $\lambda = 22.644832095$.

   (b) After 31 iterations, the power method converges to $\tilde{\lambda} = 22.64483210$
   $\tilde{x} = [\ .03978762\ \ .56098114\ \ -.33459603\ \ .53336803\ \ .51329075\ \ -.15431704\ ]$.

2. (b) After 154 iterations, $\tilde{\lambda} = 2.94188363485$ and $\tilde{x} = [\ .55065580727\ \ -.51865369330\ \ .45650931190\ \ -.36783426864\ \ .25778203471\ \ -.13274844593\ ]$.

   (c) After 30 iterations, $\tilde{\lambda} = 6.19999998113$ and $\tilde{x} = [\ .97979591798\ \ -.19595916510\ \ .03919176386\ \ -.0078378681\ \ .005156516522\ \ -.000300099331\ ]$.

3. (a) - (b) After 180 iterations we have no convergence because the components of successive vectors agree to within a $\pm$ sign.

INDEX

HP-48G/GX TEACHING CODE

APLY	Apply procedure to matrix
A↑K	Calculate A^k
BACK	Back substitution
CHAR	Characteristic polynomial
D.DOM	Diagonal dominance
DIAG	Diagonal matrix
ELIM	Eliminate below pivot
FWD	Forward substitution
GS	Gram-Schmit procedure
HILB	Hilbert matrix
JACOBI	Jacobi iteration
JTEST	Test Jacobi iteration matrix
→LP	Build L and P
L.SWP	Swap multipliers in L
L.TRI	Unit lower triangular matrix
L.U	Eliminate below pivot; put multipliers in L
P.DEF	Cholesky factor
P.FIT	Vandermonde matrix
P.of.A	Evaluate polynomial at matrix A
PIVOT	Gauss-Jordan pivot
POWER	Power method
PROJ	Project vector x onto vector y
SEIDL	Gauss-Seidel iteration
STEST	Test Gauss-Seidel iteration matrix
SYMM	Symmetric matrix
TRIDIA	Tridiagonal matrix
U.TRI	Upper triangular matrix

ISBN 1-886801-20-7
51595
EAN
9 781886 801202