

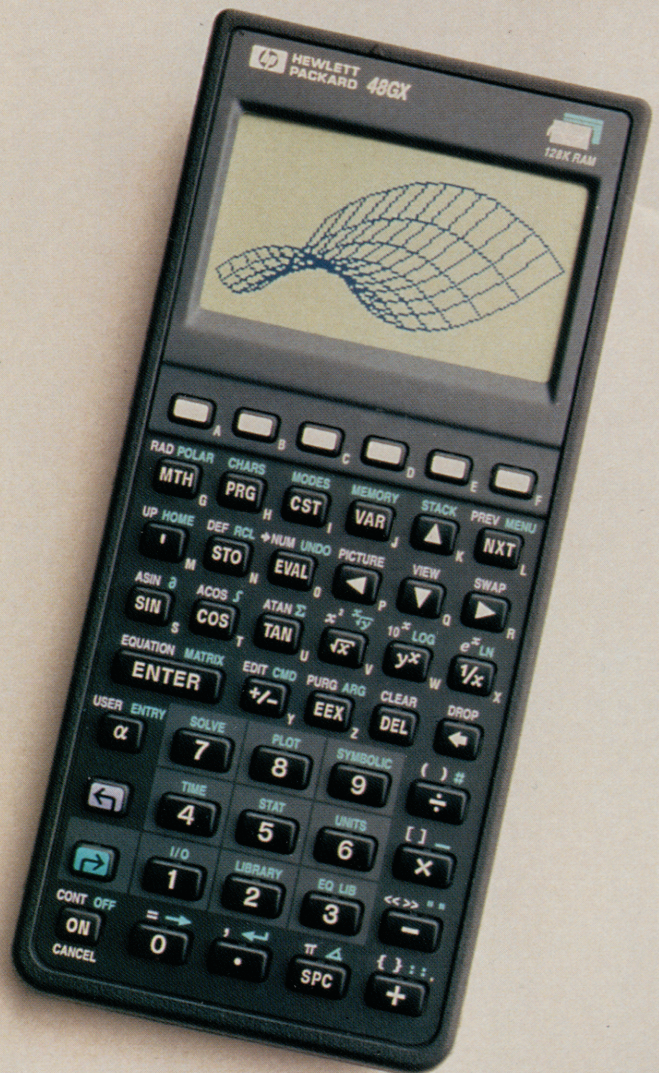
# HP-48G/GX

## INVESTIGATIONS IN MATHEMATICS

LATORRE

KREIDER

PROCTOR









**HP-48G/GX**  
**INVESTIGATIONS**  
**in**  
**Mathematics**







# HP-48G/GX INVESTIGATIONS in Mathematics

**DONALD R. LATORRE**  
*Clemson University*

**DONALD L. KREIDER**  
*Dartmouth College*

**T. G. PROCTOR**  
*Clemson University*



CHARLES RIVER MEDIA, INC.  
Rockland, Massachusetts



Copyright © 1996 by CHARLES RIVER MEDIA, INC.  
All rights reserved.

No part of this publication may be reproduced in any way, stored in a retrieval system of any type, or transmitted by any means or media, electronic or mechanical, including, but not limited to, photocopy, recording, or scanning, without prior permission in writing from the publisher.

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.

All brand names and product names mentioned in this book are trademarks or service marks of their respective companies. Any omission or misuse (of any kind) of service marks or trademarks should not be regarded as intent to infringe on the property of others. The publisher recognizes and respects all marks used by companies, manufacturers, and developers as a means to distinguish their products.

HP-48G/GX Investigations in Mathematics  
Donald R. LaTorre, Donald L. Kreider, and T. G. Proctor

ISBN: 1-886801-23-1

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

<b>PREFACE</b>	x
<b>GETTING STARTED WITH THE HP-48G/GX</b>	1
<b>PART I SINGLE VARIABLE CALCULUS</b>	15
<b>1 FUNCTIONS: EVALUATION AND GRAPHING</b>	17
1.1 FUNCTION EVALUATION	17
1.2 FUNCTION GRAPHING	23
<b>2 DERIVATIVES</b>	46
2.1 APPROXIMATING SLOPES	46
2.2 DERIVATIVES WITH THE HP-48	51
2.3 USING THE DERIVATIVE	64
<b>3 INTEGRALS</b>	105
3.1 APPROXIMATING AREA	105
3.2 INTEGRATION ON THE HP-48G/GX	129
3.3 THE FUNDAMENTAL THEOREM OF CALCULUS	138
3.4 IMPROPER INTEGRALS	147
<b>4 INFINITE SERIES</b>	152
4.1 SEQUENCES	153
4.2 SERIES	159
<b>APPENDIX FOR PART I</b>	
TEACHING CODE FOR PART I	174

<b>PART II DIFFERENTIAL EQUATIONS</b>	177
<b>5 PLOTTING SOLUTIONS FOR DIFFERENTIAL EQUATIONS ON THE HP-48</b>	181
5.1 USING BUILT-IN PROGRAMS	182
5.2 ELEMENTARY USER PROGRAMS	191
<b>6 FIRST ORDER DIFFERENTIAL EQUATIONS</b>	200
6.1 POPULATION PROBLEMS	207
6.2 MOTION OF A PARTICLE IN ONE DIMENSION	211
6.3 INPUT OUTPUT PROBLEMS	216
<b>7 SECOND ORDER DIFFERENTIAL EQUATIONS</b>	225
7.1 SECOND ORDER INPUT OUTPUT PROBLEMS	230
7.2 TRAJECTORIES IN THE $y_1$ - $y_2$ PLANE	238
7.3 LINEAR VARIATIONAL SYSTEMS IN THE PLANE	244
<b>8 LINEAR SYSTEMS OF DIFFERENTIAL EQUATIONS WITH CONSTANT COEFFICIENTS</b>	254
8.1 HOMOGENEOUS SYSTEMS	254
8.2 NON-HOMOGENEOUS SYSTEMS	266
<b>9 MISCELLANEOUS SYSTEMS</b>	273
9.1 THE LORENTZ EQUATIONS	278
9.2 EARTH, MOON, SATELLITE MOTION	281
9.3 DISCRETE DYNAMICAL SYSTEMS	284
9.4 PARAMETER IDENTIFICATION PROBLEMS REVISITED	290

9.5 DIRECTION FIELDS	298
9.6 PROGRAMS FOR THREE DIMENSIONAL TRAJECTORIES	305
<b>PART III ENGINEERING MATHEMATICS</b>	<b>309</b>
<b>10 MORE ON SOLUTIONS FOR DIFFERENTIAL EQUATIONS</b>	<b>310</b>
10.1 A RUNGE KUTTA METHOD FOR TWO EQUATIONS	311
10.2 SERIES SOLUTIONS	315
<b>11 BESSEL FUNCTIONS</b>	<b>325</b>
11.1 BESSEL'S EQUATION: SOLUTIONS OF THE FIRST KIND	325
11.2 BESSEL'S EQUATION: GENERAL SOLUTIONS	331
11.3 SELECTED PROPERTIES OF BESSEL FUNCTIONS	335
11.4 POSTSCRIPT ON NUMERICAL SOLUTIONS OF DIFFERENTIAL EQUATIONS	341
11.5 BOUNDARY VALUE PROBLEMS	348
<b>12 ORTHOGONAL FUNCTIONS</b>	<b>354</b>
12.1 FOURIER SERIES	355
12.2 LEGENDRE POLYNOMIALS	358
<b>13 APPLICATIONS TO VECTOR CALCULUS</b>	<b>368</b>
13.1 ANALYSIS OF SPACE CURVES	369
13.2 LINE INTEGRALS	372
13.3 DOUBLE INTEGRALS	375



**APPENDIX FOR PART III**

PROGRAMS FOR THE ADAPTIVE 4TH ORDER RUNGE-KUTTA  
METHOD 382

**PART IV LINEAR ALGEBRA 390****14 ARRAYS 392**

14.1 ENTERING ARRAYS 392

14.2 EDITING ARRAYS 405

14.3 ARRAY ARITHMETIC 413

14.4 DETERMINANTS AND INVERSES 422

14.5 APPLYING FUNCTIONS TO ARRAYS 426

**15 SYSTEMS OF LINEAR EQUATIONS 429**

15.1 GAUSSIAN ELIMINATION 429

15.2 LU-FACTORIZATIONS 439

15.3 GAUSS-JORDAN REDUCTION 449

**16 VECTOR SPACES 452**

16.1 LINEAR COMBINATIONS AND SPANNING SETS 453

16.2 DEPENDENCE AND INDEPENDENCE 455

16.3 BASES AND DIMENSION 460

16.4 CHANGE OF BASIS 466

**17 ORTHOGONALITY** 470

17.1 ORTHOGONAL VECTORS AND SUBSPACES 471

17.2 ORTHONORMAL BASES 475

17.3 ORTHOGONAL MATRICES AND QR-FACTORIZATIONS 482

17.4 LEAST SQUARES SOLUTIONS 489

**18 EIGENVALUES AND EIGENVECTORS** 495

18.1 THE CHARACTERISTIC POLYNOMIAL 495

18.2 EIGENVALUE CALCULATIONS 500

18.3 SIMILARITY 504

18.4 REAL SYMMETRIC MATRICES 510

18.5 POSITIVE DEFINITE MATRICES 513

18.6 SINGULAR VALUE DECOMPOSITIONS 520

**19 ITERATIVE METHODS** 531

19.1 THE JACOBI AND GAUSS-SEIDEL METHODS 532

19.2 THE POWER METHOD 542

**APPENDICES FOR PART IV**

A. VECTOR AND MATRIX NORMS 549

B. TEACHING CODE FOR PART IV 554

**SOLUTIONS** 557**INDEX** 631

## PREFACE

It is not too surprising that the current movement directed towards reform in the teaching and learning of mathematics is occurring at a time when hand-held technology is, literally, *invading* our classrooms. Indeed, these two movements have a strong element of casual interaction.

The so-called "calculus reform" movement is generally recognized to date from the conference *Toward a Lean and Lively Calculus*, held at Tulane University in January 1986. There was already substantial evidence of widespread dissatisfaction with the way that calculus was being taught and with the results of that teaching, but the Tulane conference was the first to legitimize that concern.

Since that time powerful programmable graphics calculators and widely available computer algebra systems have been used to enhance the teaching of elementary mathematics courses at many colleges and universities. Indeed reports of these activities are a regular feature of the annual January mathematics meetings and the annual International Conferences of Technology in Collegiate Mathematics. Such activity in calculus has spread to calculus programs in secondary schools (numerous workshops in the Technology Intensive Calculus for Advanced Placement program and at least five calculus consortiums sponsored by the National Science Foundation).

One of the most powerful and sophisticated calculators available, Hewlett-Packard's HP-48G/GX units offer students and teachers practical opportunity to bring graphical, numerical, and symbolic processing into the teaching and learning of single variable calculus, differential equations, engineering mathematics and linear



algebra. This book is a textbook supplement for undergraduate courses in such courses. It presents appropriate pedagogical uses of, and teaching code for, the Hewlett Packard HP-48G/GX graphics calculators. It is intended to help students and instructors incorporate these powerful devices as a tool for the interactive learning and is independent of any particular textbook. The chapters survey the main topics of the subject and include activities that have been carefully designed to engage students in a modern, technology enhanced study of the material.

Much of this material is an outgrowth of the extensive classroom use of the HP-48 calculators (and before that, the HP-28 units) at Clemson University in teaching single variable calculus, differential equations and linear algebra since 1987. Starting with an early pilot course taught by John Kenelly with the HP-28C in 1987, Clemson has gradually used graphics calculators in more classes and now teaches over 100 classes each year in which every student is required to have their own HP-48G/GX unit. The university is strongly oriented towards science and engineering, and our mainstream calculus is populated by students from a variety of fields: the chemical, physical and biological sciences, mathematical and computer sciences, all engineering fields, secondary mathematics education, architecture, accounting and economics, and a few liberal arts students. Our instructors concentrate on explanations, examples, classroom discussions, and calculator activities to generate interest and enthusiasm for learning mathematics. For beginning students experimentation and "convincing evidence" developed by this technology is important.

At Clemson, the material in this book is used to supplement whatever textbook is being used at the particular time. If the use of technology is to be of any real significance in the learning process, then it must not be used as an occasional "add-on" to the course. Rather, it must become an integral part of the teaching and learning process. Therefore, the students are required to use their HP-48G/GX units on a

regular, almost daily basis. The calculators seem to bring a unique, personal dimension to the use of technology.

Whenever it is appropriate, homework from this book is assigned; sometimes in addition to assignments from the main textbook, sometimes in lieu of such assignments. Most of the Clemson faculty allow free and unrestricted use of the calculators on all tests and exams. There is ample opportunity to assess students learning of both concepts and procedures, so the technology poses no threat. On the contrary, it has helped the students to place in proper perspective much of what has traditionally occupied their predecessors in courses in single variable calculus: excessive attention to routine, algebraically intensive procedures for finding derivatives and antiderivatives. Many of the students are quite enthusiastic about the use of the calculators as a tool to help them learn.

At Dartmouth the emphasis on computing technology in the educational environment has been on the use of computers rather than calculators. From 1964 to 1984 all students were introduced to programming in BASIC on a campus time sharing system, and they made extensive use of it in some 600 Dartmouth courses, more than half outside the sciences. For the past decade students have used personal computers, and they are now required to purchase one. Most have Macintosh computers. The use of graphing calculators was not widespread until recently when students began arriving with a graphing calculator in their possession. It has been instructive to observe the interaction between computer and calculator technology. Students are now commonly expected to use Mathematica in their first calculus course, and both subsequent physics and engineering courses capitalize on the students' skills in using computer algebra systems. Graphing calculators are not required, but their use is encouraged, and many students express a preference for the portability and ease of use of calculators. Groups of students, sitting under an elm tree working together on their mathematics homework, are most likely not using a computer. They, perhaps

more than their faculty instructors, have come to recognize computers and calculators as complementary, non-competing technologies. An earlier version of this book has been used by many Dartmouth students who wanted to learn to make more effective use of their graphing calculators. They have found it to be accessible and most interesting.

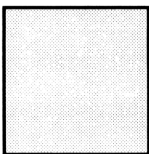
Professor Don Krieder and I want to acknowledge the contributions of several pioneers in the development of calculators in the teaching of elementary mathematics and to this book. To John Kenelly who foresaw the potential at Clemson and at other institutions with the HP units in calculus. To Tom Tucker (of Colgate University) for numerical integration routines, which date back to the early days of the HP-28C. To Bill Wickes, Charlie Patton and Paul McClellan of Hewlett Packard. (Bill Wickes improved the code for the program FTC dealing with the Fundamental Theorem of Calculus, Charlie Patton wrote the program TAYLAT to speed up Taylor polynomial calculations and Paul McCellan developed calculator versions of LAPACK code for the matrix operations that are built in to the HP-48G/GX.) To Robert Simms of Clemson for his very creative programs GRECT, GSEQ and GPS that enable students to visualize rectangle approximations to integrals and the graphs of infinite sequences. To Gloria Orr and April Haynes, who with considerable skill and patience, produced by word-processing everything that appears between the covers of this book. Finally to our co-author, Professor Don LaTorre the inspiration for this publication and many other developments in the use of calculators. Even though this past summer Don suffered a disabling stroke, he maintains a keen interest in this project. We wish him the best of luck in the future.

*Clemson University*

*Gil Proctor  
October, 1995*







# GETTING STARTED WITH THE HP-48G/GX

This section 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 `ROOT` or `TANL`.
- 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 HYP (= hyperbolic) submenu by pressing the white menu key beneath it to access the special commands for working with hyperbolic functions. 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<sup>x</sup>** (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, SYMBOLIC. 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.

## Display Settings

It is best to keep the calculator's angle mode set to radians in order to work with trigonometric functions. Press  (purple)  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   (the left-shifted  key), opening the FMT (= Format) menu and checking to see that the left-most menu box reads . The small box next to STD indicates that STD mode is active. If the menu simply reads  press the associated white menu key to activate STD mode. Now press   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 , use  to highlight Polar and press . 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  $\sqrt{\phantom{x}}$  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  $\sqrt{\phantom{x}}$  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  $\text{OK}$ . Notice that the time and date now appear above the horizontal line. If you wish to modify the time or date, press  $\text{TIME}$  (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 ( $e$ ,  $i$ ,  $\pi$ , 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  $\leftarrow$  CST keys and open the MISC submenu, the SYM menu key should read SYM  $\square$ . 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  $\square$ .



## 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  $\pm/\mp$  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  $\rightarrow$  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  $\rightarrow$ ; no need to press  $\leftarrow$  now). Now subtract again to get 69.129. Take the square root with  $\sqrt{x}$ , then cube the result with 3  $Y^X$ . You should have 574.765129278.

To edit this result, press the  $\nabla$  (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  $\rightarrow$  LN (the right-shifted  $1/x$  key) to obtain the natural logarithm. To multiply by  $\pi$ , press  $\leftarrow$   $\pi$  ( $\pi$  is obtained with the left-shift SPC key) then  $\times$ . Notice the symbolic result '6.35396147609 \*  $\pi$ ' on level 1, enclosed in tick marks. To convert this to a numerical result, use  $\leftarrow$   $\rightarrow$ NUM (the left-shift EVAL key). Now drop the 19.9615586945 from level 1 with  $\leftarrow$ . 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.

## **$n^{\text{th}}$ Roots**
















With a real or complex number on stack level 1, the  key will return its square root. If the number is real and negative, say -3, then the  key will return a complex number whose real part is zero: (0, 1.73205080757) for the square root of -3.









To take the  $n^{\text{th}}$  root of a real number  $x$  for  $n > 2$  we can calculate  $x^{1/n}$ :  $2^{1/3}$  is 1.25992104989. But when  $n$  is odd and  $x$  is negative, this procedure will always return a complex number:  $(-8)^{1/3}$  is (1, 1.73205080757). This result is the *principal* cube root of -8, certainly not the *real* cube root that we expected. To obtain the real  $n^{\text{th}}$  root of a negative number for an odd value of  $n$ , use the XROOT key , which is the right-shifted  key. For example, to obtain the real cube root of -8, simply enter -8 and then 3 (the desired root); press  to obtain -2.













## **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  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   . Lower case characters are obtained by the sequence   , then the character key. For example,      puts 'd' on level 1. (Thus  *left-shift* will give *lower case*).

To enter the algebraic expression 'SIN(X)', press     . Notice the location of the cursor after each keystroke; after  the cursor is still inside the right parenthesis. To move it outside, use the right cursor key . But, pressing  does it all for you. As a more complicated example, try 'COS( X ^ 2)/( 2 \* X ^ 3)'. The keystroke sequence is:

     2  ÷  ÷ 2  
    3 .

Yes, it is necessary to insert the \* in  $2 * X \wedge 3$ ; if you forget, when you press , 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  (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  $\boxed{+/-}$ . 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  $\boxed{\text{ENTER}}$  or  $\boxed{=}$  key. To produce a numerical result for  $\sqrt{\ln 2.3}$  on the HP-48 using algebraic entry we type

'  $\sqrt{\phantom{x}}$  LN 2.3 EVAL

But to obtain this using RPN, we do

2.3 LN  $\sqrt{\phantom{x}}$

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  $\wedge$  COS SQ 1 +  $\sqrt{\phantom{x}}$ .

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)  $\wedge$  (2/3)', '2 \* X + 1', and ' $\sqrt{\phantom{x}}$ (X  $\wedge$  2 - 4)' on the stack in this order (you can use either direct algebraic entry or RPN for any of them); now press  $\boxed{\div}$  to build the quotient, then  $\boxed{+}$  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  $\sqrt{x^2 + y^2}$ .

## 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  $\boxed{\leftarrow}$   $\boxed{-}$  to get the program delimiters « », then use  $\boxed{\leftarrow}$   $\boxed{\sqrt{x}}$   $\boxed{\leftarrow}$   $\boxed{\rightarrow}$   $\boxed{\leftarrow}$   $\boxed{\sqrt{x}}$   $\boxed{+}$   $\boxed{\sqrt{x}}$   $\boxed{\text{ENTER}}$ . Now put the name 'TRY1' on the stack and press the  $\boxed{\text{STO}}$  key. If you press the VAR key you will see that the leftmost menu key is labeled  $\boxed{\text{TRY1}}$ . To run the program with inputs 1 and 2, put 1 and 2 on the stack and then press  $\boxed{\text{TRY1}}$  to see the result 2.2360679775. In fact, you need not actually enter the inputs onto the stack: simply press 1  $\boxed{\text{SPC}}$  2, then  $\boxed{\text{TRY1}}$  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 often use this shortcut with our programs.

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

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 CALC in which you can store any variables that you may need in a study of calculus, begin by putting the name 'CALC' on stack level 1. Now press  $\boxed{\leftarrow}$   $\boxed{\text{MEMORY}}$  (the left-shifted  $\boxed{\text{VAR}}$  key), open the DIR submenu and execute the command CRDIR (create directory). If you then open the VAR menu you will see the  $\boxed{\text{CALC}}$  directory on the left. The short bar above the label is suggestive of the tab on a file folder, and reminds you that CALC is a subdirectory. Press  $\boxed{\text{CALC}}$  to open this directory and notice the list {HOME CALC} at the top of your screen, indicating that the current directory is now CALC. 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 **CALC** is the current directory, that no variable **'X'** is stored in **CALC**, 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.



# **PART I**

## **SINGLE VARIABLE CALCULUS**

Part I of this work is a textbook supplement for undergraduate courses in single variable calculus. It presents appropriate pedagogical uses of, and teaching code for, the Hewlett Packard HP-48G/GX graphics calculators. It is intended to help students and instructors incorporate these powerful devices as a tool for the interactive learning of single variable calculus, and is independent of any particular textbook. The chapters survey the main topics of the subject and include activities that have been carefully designed to engage students in a modern, technology enhanced study of the material.

No two instructors and no two textbooks approach single variable calculus alike. Therefore, the material has been organized into independent chapters that address main topics:

- Functions and Graphs
- Derivatives
- Integrals
- Series

The Teaching Code given in the APPENDIX FOR PART I is a collection of special-purpose HP-48G/GX programs, each one addressing a specific aspect of the course. The appendix contains a complete listing of the teaching code appears on the inside back cover. The code is readily available from the editor for downloading to an HP-48G/GX from a microcomputer.

Calculus students are well served by classroom and homework exercises which reveal the graphs as well as the analytical representations of the elementary functions. A sense

of size and geometric pattern of these functions can be acquired through participation by the student beginning in class and directed by the course instructor. This use of calculators gives some immediate feedback concerning the student's grasp of the concepts. Questions about the steps required in deriving the solutions of problems can be resolved in class. Implications to be derived from graphs or tables can be discussed and illustrated through class exercises in which part of the solution has been worked out by the student. Small group learning can be initiated in class.

Use of the HP-48G/GX calculator requires some learning time just as the use of other high level mathematics software. However the instruction set required by the calculator has much in common with the instruction set of this software. The greater resolution in the graphs presented and the speed of computation on larger computers do have advantages. However the student will be able to transfer between these environments quickly, and the portability of the calculator will serve the student throughout his career. A particular advantage of the HP-48-GX is that the memory allows the student to arrange work into directories and store program and results for many subjects.

The material in this book can be used to supplement a traditional or a reform calculus textbook. Highlights include comparisons between rectangular, trapezoidal and quadratic method for approximating the integral and the associated errors, the speed toward convergence of a series such as the  $p$  series, the graphs associated with various Taylor polynomials of a function and the graphical presentations of a function and its derivative in a common picture. We hope you will find the enhancement of your calculus course with a calculator as much fun as we did.

# 1

## FUNCTIONS: EVALUATION AND GRAPHING

Beginning calculus is a study of the behavior of functions: their variation, rates of change, limiting behaviors. We shall thus begin with a brief look at how functions can be represented, evaluated and graphed on the HP-48G/GX calculators.

### 1.1 FUNCTION EVALUATION

#### Evaluating with SOLVR

The basic idea of a function  $F$  of a single variable  $x$  is simple enough: for each value of the input variable  $x$  we obtain exactly one output value  $F(x)$ . The HP-48G/GX units have a built-in environment that is ideal for the evaluation of functions, the SOLVR. Although the SOLVR is designed to solve equations, the format of its menu makes it convenient for evaluating functions. With the function on level 1, press  $\leftarrow$  **SOLVE** to access the SOLVE application, open the ROOT subdirectory and then load the function on level 1 into EQ by pressing  $\leftarrow$  **EQ** (remember: left-shift will *load*). Now press the **SOLVR** key. To evaluate the function stored in EQ at a number (or variable), simply key in the number (or variable), press **X** then **EXPR=**. For example, to numerically investigate the behavior of the function  $f(x) = \frac{x+2}{2x+1}$  as  $x$  approaches 0 we can proceed as follows.

Put '(X + 2) / (2 \* X + 1)' on level 1 and press  $\leftarrow$  **SOLVE** **ROOT** , then  $\leftarrow$  **EQ** **SOLVR**. To find  $F(.01)$ , press **EEX** 2 **+/-** then **X** **EXPR=** to see 1.97058823529.

To find  $F(.0001)$ , press EEX 4 +/- then X EXPR= to see  
1.99970005999.

To find  $F(.000001)$ , press EEX 6 +/- then X EXPR= to see  
1.99999700001.

To find  $F(.00000001)$ , press EEX 8 +/- then X EXPR= to see  
1.99999997.

Clearly, we can see that  $f(x)$  is approaching 2 as we let  $x$  approach 0 from the right. What happens to  $f(x)$  as we let  $x$  approach 0 from the left? Experiment to find out.

When using the SOLVR, if you store an equation, say 'expression 1 = expression 2', in EQ instead of a single expression, pressing EXPR= for a given value of  $X$  will return two values, one for the left side of the equation and one for the right side. This provides a convenient way to compare the outputs of two functions at various input values of  $X$ .

You should be aware that whenever you use the calculator's SOLVE application, the last value for  $X$  is stored under the variable name 'X' in user memory. You need not make explicit use of the SOLVR for this to occur: pressing ROOT on the FCN submenu automatically activates the SOLVR (as do the commands ISECT, EXTR and F' which appear as menu keys on the FCN submenu). You can see this variable by pressing VAR to go to the VAR menu, where you will see the menu key X. Press X to recall the value stored for  $X$ . Our recommendation is that before going on to the next application you *immediately* purge this variable to avoid trouble later on. Purge by pressing ' X PURGE.

## User-Defined Functions

Another way to represent functions on the HP-48G/GX is by creating *user-defined functions*. In HP-48 parlance, a user-defined function is simply 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 «  $\rightarrow X$  '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 ' $F(X) = 2 * \text{SIN}(X) + \text{SIN}(4 * X)$ ' onto level 1 of the stack and press  $\leftarrow$  **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  $\rightarrow$   $F$ ; 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 ' $T \wedge 2$ ' on level 1 and press  $F$  to see ' $2 * \text{SIN}(T \wedge 2) + \text{SIN}(4 * T \wedge 2)$ '. Likewise, press 4  $F$  to see  $2 \sin 4 + \sin(4 * 4)$  evaluated as -1.80150830728. Note that you can enter the equation ' $F(X) = \text{expression in } X$ ' directly or by first entering ' $F(X)$ ', then the 'expression in  $X$ ' and pressing  $\leftarrow$  **=**. In either case the **DEF** key automatically creates the user-defined function from the equation.

User-defined functions of two or more variables are constructed in the same way. For instance, to represent  $G(s, t) = s/t^2$  enter ' $G(S, T) = S/T \wedge 2$ ' and press  $\leftarrow$  **DEF**.

To evaluate  $G$  we input a value for  $S$ , then for  $T$ . Try it for yourself:  
 $G(2, 3) = .222222222222$ .

Piecewise-defined functions often occur in applications and are introduced early in calculus to illustrate the ideas of one-sided limits and points of discontinuity. The best way to represent them on the HP-48G/GX is to use the IFTE command, found on the second page of the PRG BRCH menu. The IFTE command is an abbreviation for the "if ... then ... else ... end" construction and executes one of two procedures that you specify, according as a "test clause" is true or false. The IFTE command takes three arguments: a test argument and two procedural arguments, as in IFTE (test, procedure 1, procedure 2). You should interpret this as "If *test clause* is true, then *execute procedure 1* else *execute procedure 2*".

To represent the function  $p(x) = \begin{cases} x^2 - 2x & x < 0 \\ 1 - x^2 & 0 \leq x \end{cases}$ , the desired expression is

'IFTE(X < 0, X ^ 2 - 2 \* X, 1 - X ^ 2)'. Begin with a tick, then go to the second page of the PRG BRCH menu and press IFTE, followed by the three required arguments separated by commas (α ← 2 will produce the < symbol or you can go to the PRG TEST menu), then ENTER. This expression can now be treated like any other function. For instance, with 'IFTE(X < 0, X ^ 2 - 2 \* X, 1 - X ^ 2)' displayed on stack level 1, enter 'P(X)', then press SWAP, = and finally DEF to create a user-defined function. Try evaluating  $P(X)$  using values to the left and right of 0 to discover the behavior of the function  $p$  as  $x$  approaches 0.

The general construction for a piecewise-defined function with two pieces like

$$f(x) = \begin{cases} f_1(x) & x \leq a_1 \\ f_2(x) & a_1 < x \end{cases}$$

is IFTE(  $x \leq a_1, f_1(x), f_2(x)$  ).

For three or more pieces, you can nest the IFTE commands:

$$\text{for } f(x) = \begin{cases} f_1(x) & x < a_1 \\ f_2(x) & a_1 \leq x < a_2 \\ f_3(x) & a_2 \leq x \end{cases},$$

use  $\text{IFTE}(X < a_1, f_1(X), \text{IFTE}(X < a_2, f_2(X), f_3(X)))$ .

## Activity Set 1.1

1. What happens to values of  $f(x) = \frac{\sin x}{x}$  as  $x$  approaches 0? Use the SOLVR to find out. Let  $x$  approach 0 through values  $x = 10^{-2}, 10^{-3}, \dots, 10^{-6}$ , then their negatives. Press EEX 2 +/- to input  $10^{-2}$ , etc.
2. What happens to values of  $f(x) = \frac{\cos x - 1}{x}$  as  $x$  approaches 0? Use the SOLVR to find out. Let  $x$  approach 0 through values  $x = 10^{-2}, 10^{-3}, \dots, 10^{-6}$ , then their negatives. Press EEX 2 +/- to input  $10^{-2}$ , etc.
3. What happens to values of  $f(x) = xe^x$  as  $x$  approaches  $-\infty$ ? Use the SOLVR to find out. Let  $x$  approach  $-\infty$  through values  $x = -1, -10, -1,000$  and  $-10,000$ .
4. Repeat Activities 1-3, but this time with a user-defined function for  $f(x)$ .
5. (a) Evaluate  $f(x) = (1 + 1/x)^x$  for  $x = 10^2, 10^4, \dots, 10^{11}$ . Then make a conjecture about what happens to  $f(x)$  as  $x \rightarrow \infty$ .  
(b) Now evaluate  $f(x)$  for  $x = 10^{12}$ . Can you explain the result?
6. Investigate the behavior of the following function as  $x \rightarrow 0$ :

$$f(x) = \frac{x + |1 - \sqrt{x+1}|}{|1 - \sqrt{x+1}|}.$$

7. (a) Use the IFTE command to build an expression for

$$f(x) = \begin{cases} x^2 & \text{if } x < 0 \\ \cos x & \text{if } x \geq 0 \end{cases}.$$

- (b) Evaluate  $f(x)$  for a sequence of values that approaches 0 from the left; what does  $f(x)$  approach?
- (c) Now evaluate  $f(x)$  for a sequence of values that approaches 0 from the right; what does  $f(x)$  approach?
- (d) In view of your results in (b) and (c), does  $\lim_{x \rightarrow 0} f(x)$  exist?

8. The *greatest integer* function, often denoted by  $\lfloor x \rfloor$ , is defined by

$$\lfloor x \rfloor = \text{the greatest integer } \leq x.$$

It is executed on the HP-48G/GX by the **FLOOR** command (a menu key appears on the third page of the **MTH REAL** menu). Use the **FLOOR** command to calculate  $\lfloor x \rfloor$  for each of the following values of  $x$ :

$$(a) \pi^e \quad (b) -\pi^e \quad (c) e^\pi \quad (d) (\sqrt{\pi})^{10}$$

9. The *least integer* function, often denoted by  $\lceil x \rceil$ , is defined by

$$\lceil x \rceil = \text{the least integer } \geq x.$$



It is executed on the HP-48G/GX by the **CEIL** command (a menu key appears on the third page of the **MTH REAL** menu). Use the **CEIL** command to calculate  $\lceil x \rceil$  for each of the following values of  $x$ :

$$(a) \pi^e \quad (b) -\pi^e \quad (c) e^\pi \quad (d) (\sqrt{\pi})^{10}$$



## 1.2 FUNCTION GRAPHING

The single most important application of the HP-48G/GX to a study of calculus is to create visual images of the wide variety of functions under study. More than anything else, the ability to graph quickly and easily adds a powerful new dimension to the traditional analytical approach to calculus. Many of the important aspects of functional behavior — maximum and minimum values, rates of change, etc. — can be effectively displayed by the graph of the function. With the HP-48, graphical representations can be used from the beginning of the course.

To get informative representations of graphs on the HP-48 you must set the viewing window to display the part of the graph that you want to see. The default settings of the plotting ranges for points  $(x, y)$  are  $-6.5 \leq x \leq 6.5$  and  $-3.1 \leq y \leq 3.2$ , with a common unit scaling of each axis. Since there are 131 columns and 64 rows of pixels, the default settings produce square pixels of size .1 and your visual intuition of slope and area is preserved on the screen. The default settings also work well for trigonometric functions of amplitude 3 or less. You can, of course, change the settings in a variety of ways, some of which will be illustrated in the examples. To accommodate trigonometric graphs, make sure your calculator is set to radians mode. The   key will toggle between degrees and radians; when radian mode is set, the message RAD will appear in the top left corner of the screen.

### Basic Plotting

Functions are represented graphically as *plots* in the PICTURE environment. The general procedure to produce a plot of a function of a single independent variable is as follows:

- Access the PLOT application;
- Make sure the plot type is set to FUNCTION;

- Enter the expression that defines the function;
- Set the plotting parameters: the independent variable, horizontal and vertical plotting ranges, etc.;
- ERASE ( if desired) any previous plots;
- Execute the DRAW command.

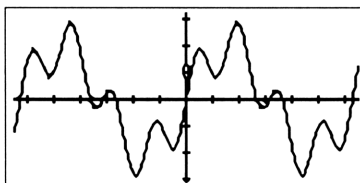
The HP-48G/GX calculators allow you to access the PLOT application in two different ways in order to enter a function's expression and to set the plotting parameters: with  $\boxed{\rightarrow}$   $\boxed{\text{PLOT}}$  to interact directly with the main PLOT screen, or with  $\boxed{\leftarrow}$   $\boxed{\text{PLOT}}$  to use the various commands on the PLOT menu. We will illustrate both approaches in our first two examples.

**EXAMPLE 1.** Graph  $y = 2 \sin x + \sin 4x$  with the default plotting parameters.

### Using the PLOT screen

From the stack display screen, go to the PLOT application with  $\boxed{\rightarrow}$   $\boxed{\text{PLOT}}$ . The main PLOT screen will show the current plot type, current angle mode, current expression in EQ (if any), the independent variable (X, by default), and the current horizontal and vertical display ranges. If the current plot type does not show Function, press  $\boxed{\Delta}$   $\boxed{\text{CHOOS}}$ , highlight FUNCTION and press  $\boxed{\text{OK}}$ . If necessary, use  $\boxed{\triangleright}$  and a similar procedure to set the angle mode to RAD. Now highlight the field EQ: and type '2 \* SIN( X ) + SIN(4 \* X)' and press  $\boxed{\text{ENTER}}$ . Notice that when you are using the PLOT screen, you do not have to begin the algebraic expression that defines the function with a tick mark. If the default plotting parameters are current, the independent variable will appear as INDEP: X, the horizontal display range as H-VIEW: -6.5 6.5, and the vertical display range as V-VIEW: -3.1 3.2. If any of these settings appears otherwise, go to the next page of

the PLOT menu with **NXT**, press **RESET** and highlight Reset Plot and press **OK**. Once the default plotting parameters are set, return to the previous page with **PREV** and press **ERASE** to erase any previous plot. Now press **DRAW**. You should see a plot like this:



When the plot is complete and the menu labels appear, press **TRACE** and then use the right and left cursor keys to *trace* along the plot. Press **(X, Y)** to obtain *coordinate readouts* for the cursor. The *x*: value is the pixel location of the cursor but the *y*: value is the value of the function in EQ computed at the *x*: value. Press **+** to restore the menu keys. When you have finished viewing the plot, press **ON** twice to return to the stack environment. You can bring back the plot by using the **◀** key to access the PICTURE environment.

## Using the PLOT menu

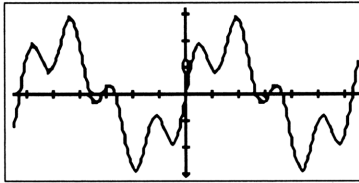
Enter '2 \* SIN( X ) + SIN(4 \* X)' on level 1 (you will have to start with a tick mark), and press **↵** **PLOT** to access the PLOT menu. If you do not now read "Ptype: FUNCTION" at the top of your screen press **PTYPE** and then **FUNC**. Then use **↵** **EQ**, to store the expression on level 1 into EQ. Now press **PPAR** to see the plotting parameters. If you do not now read

```

Indep:  'X'
Depnd:  'Y'
Xrng:   -6.5    6.5
Yrng:   -3.1    3.2
Res:    0

```

on your screen, press **RESET** to return your screen to the default settings. Now press **↶** **PREV** to turn back a page and open the PLOT menu with **PLOT**. Press **ERASE** to erase any plot previously drawn, then **DRAX** to draw axes and **DRAW**. You should see the graph we had before:



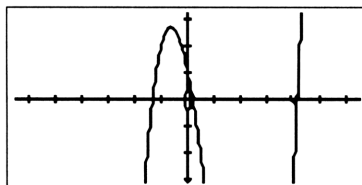
Now trace along the plot with coordinate readouts. When you have finished viewing the plot, return to the stack display screen by pressing **ON**. You can always bring back the plot by using the **◀** key.

Often, in order to see more of a plot you can compress or expand the viewing screen vertically or horizontally by using commands from the ZOOM menu, as in the next example.

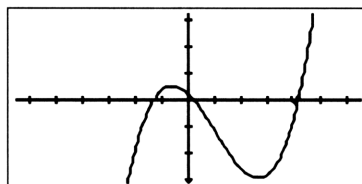
**EXAMPLE 2.** Graph  $y = x^3 - 3x^2 - 5x + 1$ .

### Using the PLOT screen

Open the PLOT application with **➞** **PLOT**. Since the default settings are current from our previous example, we need only enter the new function. With the EQ field highlighted, type 'X ^ 3 - 3 \* X ^ 2 - 5 \* X + 1' and press **ENTER**. If you made a mistake in entering the expression, simply highlight the EQ field, press **EDIT**, and use the cursor keys and the **DEL** key to correct the entry. Use **OK** to insert the corrected expression into the EQ field. Press **ERASE** and **DRAW** to produce this plot:



The lower right part of the plot is not visible, so to see more we will zoom out on the vertical axis but leave the x-axis unchanged. Open the **ZOOM** menu, then the **ZFACT** submenu. Set the **H-FACTOR** to 1, the **V-FACTOR** to 5, then press **OK**. Move to the next page and zoom out on the vertical axis with the **VZOUT** command. You will get the following plot:

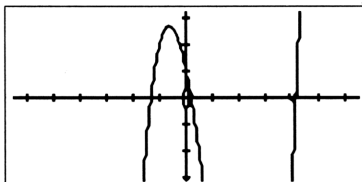


Use the subtraction key **-** to remove the labels from the bottom of the screen that hide the plot; use **+** to put the labels back. To verify that we have expanded the height of the graphing screen by a factor of 5 activate the coordinate read-out with the menu key **(X, Y)**, and use the **Δ** key to move the cursor up to the first tick mark on the *y*-axis. Notice that this tick mark records the zoom factor. The zoom factor 5 was determined by trial and error; a smaller factor failed to show the low point of the graph. Press **ON** to return to the stack display screen when you've finished.

## Using the PLOT menu

Begin with ' $X^3 - 3 * X^2 - 5 * X + 1$ ' on level 1 of the stack and press **↵** **PLOT** to access the **PLOT** menu. Then use **↵** **EQ** to load the expression on level 1 into **EQ** and use **PPAR** to see the current plotting parameters. Since we

wish to plot first with the default settings use **RESET** to set the plotting parameters to their default settings. Return to the previous page and open the **PLOT** menu, then use **ERASE**, **DRAX** and **DRAW** to produce this plot:



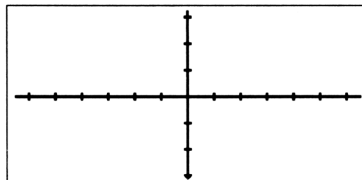
Now zoom out as before to see more of the local behavior.

The above two examples convey the major differences in using the **PLOT** screen ( **→** **PLOT** ) and the **PLOT** menu ( **↵** **PLOT** ) to access the **PLOT** application. Using **→** **PLOT** lets you interact with the main **PLOT** screen, and using **↵** **PLOT** provides direct access to the commands on the **PLOT** menu. Many beginners prefer to interact with the **PLOT** screen, but more experienced users tend to prefer the menu commands because of their speed and versatility. From here on, we shall leave the choice to you, the reader.

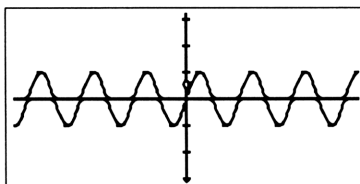
## Zoom Operations

As EXAMPLE 2 shows, after producing a plot we may have to modify one or more of the plotting parameters in order to better see some portion of the plot. Here are two examples of plots that require adjustment on the range of  $x$  values:

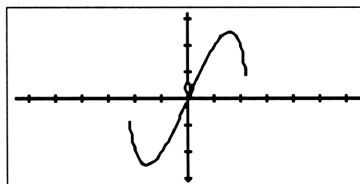
**EXAMPLE 3.** Graph  $y = \sin(10\pi x)$  on the default viewing screen. You will see:



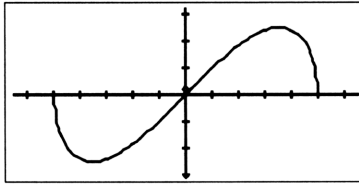
No plot appears. With the default plotting parameters the HP-48 calculates values of  $y$  for each of the 131 values of  $x$  from  $x = -6.5$  to  $x = 6.5$ , .1 unit apart. Since  $10\pi$  times each of these numbers is an integer multiple of  $\pi$ , the sine function is 0 at each of these values of  $x$ . Thus the plot lies along the  $x$ -axis. You can see this by turning off the axes and redrawing the plot. To get a better picture we can compress the viewing window in the  $x$  direction. Zoom in on the horizontal axis by a factor of 10 (set the H-FACTOR, then use HZIN) to see:



**EXAMPLE 4.** If you plot  $y = x\sqrt{5-x^2}$  with the default plot parameters, you will see:



Why does the plot fail to touch the  $x$ -axis? From the function,  $y$  is 0 when  $x = \pm\sqrt{5}$ , but these points do not show on the plot. To four decimal places,  $\sqrt{5} = 2.2361$ . With the default plotting parameters, the HP-48 will plot a point for  $x = 2.2$ ; but for  $2.3 \leq x$ ,  $y$  is a complex number so no points will be plotted. We can "tie down" the plot to the  $x$ -axis by modifying the scale along the  $x$ -axis. For example, if we zoom in on  $x$  by a factor of 2.2361 we will be rescaling the  $x$ -axis so that 5 units on the  $x$ -axis is approximately  $\sqrt{5}$ , and will see the following plot:



## The Zoom Menu

Several of the commands on the **ZOOM** menu are fairly self-evident:

**ZIN** and **ZOUT**: Zoom in or zoom out on both axes according to the **ZOOM FACTORS**.

**HZIN** and **HZOUT**: Zoom in or zoom out on the horizontal axis according to the **H-FACTOR**.

**VZIN** and **VZOUT**: Zoom in or zoom out on the vertical axis according to the **V-FACTOR**.

**ZDFLT**: Zoom to the default plotting screen.

**ZLAST**: Zoom to the last plotting screen.

But some of the other commands are not so obvious:

**ZSQR**: Leaves *Xrng* unchanged but changes *Yrng* so that each pixel is square.

**ZDECI**: Leaves *Yrng* unchanged but resets *Xrng* to its default state: -6.5 6.5 . Pixels are .1 unit along the horizontal axis.

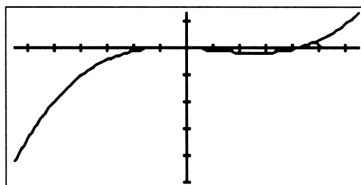
**ZINTG**: Leaves *Yrng* unchanged but sets *Xrng* to -65 65 so that each pixel along the horizontal axis is 1 unit.

**ZTRIG**: Resets *Xrng* so that every 10 pixels equals  $\pi/2$  units and resets *Yrng* so that every 10 pixels equals 1 unit.



**ZAUTO:** Leaves  $Xrng$  unchanged but rescales the vertical axis by sampling the expression in EQ at 40 equally spaced values across the  $x$ -axis plotting range, resets the  $Yrng$  to include the maximum and minimum sampled values, and then redraws the plot.

**Caution:** It is tempting for beginning users of the HP-48G/GX to use the **ZAUTO** command instead of adjusting the vertical display range in other ways. But *we urge restraint and caution in the use of ZAUTO* because it tends to excessively "flatten" a plot due to the narrow vertical dimension of the display screen. For instance, if we return to the function of EXAMPLE 2,  $y = x^3 - 3x^2 - 5x + 1$ , and apply the **ZAUTO** command to the plot obtained with the default plotting parameters, we obtain the following "flattened" plot:

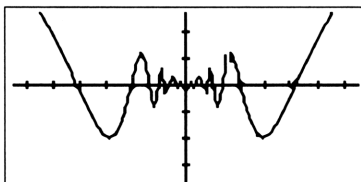


Compare this with the plot we obtained by rescaling the vertical axis with a zoom-out factor of 5. Which would you prefer to see?

The **BOXZ** application on the **ZOOM** menu of the HP-48G/GX is especially helpful for zooming in on a particular region of a plot. The basic idea is to capture the region of interest within a small "box", then zoom in on the box. Here's an example.

**EXAMPLE 5.** Begin by plotting  $y = x \sin \frac{1}{x}$  on the default screen. To better see what's happening near the origin, begin by opening the **ZOOM** menu. Now move the cursor 5 pixels to the left of the origin, then down 3 pixels and open **BOXZ** file. Now

move the cursor 5 pixels to the right of the origin, then 3 pixels above the origin. Notice that the cursor drags a small box that has the origin as its center. Now press **ZOOM** to zoom in on the box, and obtain the following plot:



Repeat this zooming-in process with **BOXZ** by moving to a corner of a box 5 pixels to the left and 3 pixels below the origin, then moving to the diagonally opposite corner 5 pixels to the right and 3 pixels above the origin and pressing **ZOOM**. You should by now be ready to explain the behavior of  $y = x \sin \frac{1}{x}$  as  $x$  approaches 0.

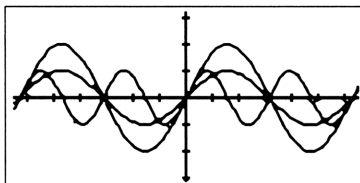
## Superimposing Plots

To superimpose the plots of the graphs of two or more functions you can plot them individually without erasing. A better procedure is to put a list {F G H . . . etc.} of the functions  $F, G, H, \dots$  etc. to be graphed into EQ and set the calculator to *sequential plotting mode* (the default mode). When the **DRAW** command is executed, the functions in the list are plotted sequentially, left-to-right. The following example will illustrate this, both from the **PLOT** screen and the **PLOT** menu.

**EXAMPLE 6.** Superimpose plots of the graphs of  $\sin x$ ,  $2 \sin x$  and  $\sin 2x$  on the same set of coordinate axes using the default parameters.

- (a) **Using the PLOT menu.** Put ' $\sin( X )$ ', ' $2 * \sin( X )$ ' and ' $\sin(2 * X)$ ' on the stack press the **Δ** key to engage the interactive stack. Then move the pointer to level 3 and press **→LIST** **ENTER** to build the list

{ 'SIN( X ) '2 \* SIN( X )' 'SIN(2 \* X)' }. Now go to the PLOT menu and store this list into EQ. Open the FLAG menu on the second page of the PLOT menu and make sure that the middle menu key reads **SIMU**. If **SIMU** appears, toggle off the key. Then return to the PLOT menu, reset the default plotting parameters, and press **ERASE**, **DRAX** and **DRAW** to see the plots. Observe how the plots are drawn sequentially from the list.



To draw the plots in the list simultaneously instead of sequentially, go to the second page of the PLOT menu, open the FLAG submenu and toggle on the middle key to show **SIMU**.









- (b) **Using the PLOT screen.** Open the PLOT screen and insert the list { 'SIN( X ) '2 \* SIN( X )' 'SIN(2 \* X)' } into the EQ field (note that tick marks are required in the list). Open the OPTS (Options) submenu and make certain that there is no check mark (✓) in the SIMLT field. Return to the previous screen, set the default plotting parameters, then **ERASE** and **DRAW**. You should observe the list being plotted sequentially.

## Disconnected Plots

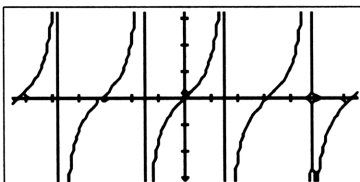
All of our function plots so far have been done in *connected* mode, which means that any spaces between the pixels activated by the plot EQ were filled in with short line segments. But there are times when it is desirable to plot in disconnected mode, so that no filling in will be done. In connected mode, the HP-48 connects

adjacent pixels with short line segments and sometimes extraneous lines can appear on the plot.

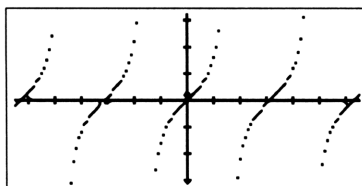
The choice between connected versus disconnected plotting modes is specified by system flag -31. When flag -31 is clear (the default state), plotting is done in connected mode. But if flag -31 is set, then plotting is done in disconnected mode.

If you are using the PLOT screen (   ), use the  menu key to access the various PLOT OPTIONS. A check (✓) in the CONNECT field indicates that connected plotting mode is active. If you are using the PLOT menu (   ), go to the second page of the PLOT menu use the  key to access the three flags that are pertinent to basic plotting (AXES, CNCT, and SIMU). If the second menu key shows  then connected plotting mode is active. If the key reads  , then disconnected mode is active; simply press the key to change the status of the flag. Here is an example.

**EXAMPLE 7.** If you plot  $y = \tan x$  in connected mode using the default screen, you will see



Notice that this plot contains vertical lines that are not part of the graph of  $y = \tan x$ . The vertical lines appear because the HP-48 connects adjacent plotted pixels. A graph of the tangent function without these extraneous lines is obtained by plotting in disconnected mode:



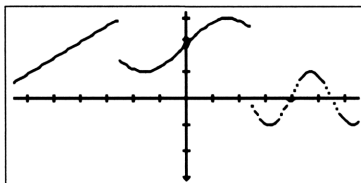
Although the disconnected plot is a little "dotty", it is nevertheless a better representation of the graph of  $y = \tan x$  than the plot above.

## Piecewise Plots

Piecewise-defined functions are plotted by putting the defining IFTE expression into EQ and proceeding as usual. To get the plot shown in the next example, use the default viewing screen and the disconnected plotting mode; in the connected mode the calculator will connect the pixels on opposite sides of the two discontinuities and give you an inaccurate representation. To set the HP-48G/GX to plot in disconnected mode, go to the second page of the **PLOT** menu and open the **FLAG** submenu. Press the second white menu key so that **CNCT** appears in the second menu box.

**EXAMPLE 8.** To plot the graph of  $f(x) = \begin{cases} .6x + 4.5 & x < -2.5 \\ 2 + \sin x & -2.5 \leq x < 2.5 \\ -\cos 2x & 2.5 \leq x \end{cases}$ , use the expression

'IFTE ( X < -2.5, .6 \* X + 4.5, IFTE ( X < 2.5, 2 + SIN( X ), -COS(2 \* X) ) )'. This gives the plot:



When you have finished, reset your calculator to plot in connected mode.

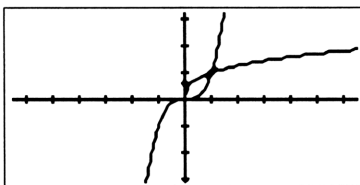
## Plotting Inverse Functions

When a function  $f$  is one-to-one (different input values produce different output values), it has an inverse function  $f^{-1}$  that satisfies

$$f^{-1}(y) = x \quad \text{iff} \quad f(x) = y.$$

Whenever  $(x, y)$  is a point on the graph of  $f$  then  $(y, x)$  will be a point on the graph of  $f^{-1}$ . Thus, the graphs of  $f$  and  $f^{-1}$  will be reflections of one another across the line  $y = x$ .

To compare the graph of  $f(x) = x^3$  with that of its inverse  $g(x) = \sqrt[3]{x}$ , you can begin by plotting the list { 'X ^ 3' 'X ^ (1/3)' }. Using the default settings you will see:

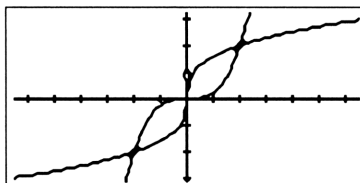


This fails to show the left branch of  $g(x) = \sqrt[3]{x}$ . The reason is that for each negative value of  $X$ ,  $X ^ (1/3)$  is calculated as the *principal cube root* of  $x$ , a complex number, and so no pixel is activated. Although you may at first find this a bit disquieting, the ability of the HP-48G/GX to return complex values for odd roots and for natural logarithms of negative numbers is but one of the many features that makes the unit so appropriate for post-calculus mathematics.

To obtain *real* odd roots of negative numbers, use the XROOT command, given by the  $\sqrt[x]{y}$  key (the  $\rightarrow \sqrt{x}$  key). To see both branches of the graph of  $g(x) = \sqrt[3]{x}$ , we must plot the expression 'XROOT(3, X)'. There are two ways to enter this:

- (i) Put 'X', then 3 on the stack and press  $\sqrt[x]{y}$ .
- (ii) Alternatively, go to the Equation Writer with  $\leftarrow$  **ENTER**, and enter the expression  $\sqrt[3]{x}$  with the keystrokes  $\rightarrow$   $\sqrt{x}$  3  $\rightarrow$   $\alpha$   $1/x$ . Press **ENTER** to convert this to the expression 'XROOT(3, X)' on stack level 1.

When the list {'X ^ 3' 'XROOT(3, X)'} is plotted with the default screen, then enlarged by a factor of 2 with the **ZOOM** menu, we see the following:



Notice that the original plot and its inverse meet on the line  $y = x$  and that the plots are reflections across this line.

To help plot the graph of an inverse function, you can use the following program INV.F.<sup>1</sup> To use INV.F, begin by storing an expression for the original function  $f$  in EQ and drawing a "good" plot of EQ, i.e., a plot on which you wish to superimpose a plot of  $f^{-1}$ . Then execute INV.F. Since the program uses the expression stored in EQ and the plotting parameters from the reserved variable PPAR, make certain that you produce your original plot from the same user directory in which INV.F resides, preferably, your CALC directory. The program will redraw the original plot of  $f$ , overlay the line  $y = x$ , and then overdraw a plot of  $f^{-1}$ . In case  $f$  is not a one-to-one function, INV.F will overdraw the *inverse relation* for  $f$ .

<sup>1</sup> Thanks to William C. Wickes of Hewlett Packard for suggesting this version that uses parametric plotting.

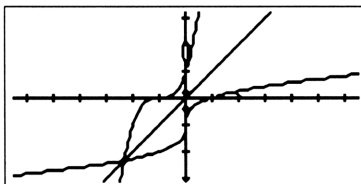
**INV.F**

*Inputs:* An expression for a function  $f$ , stored in EQ; and the desired plotting parameters, stored in PPAR.

*Outputs:* Draws, over the plot of  $y = f(x)$ , plots of the line  $y = x$  and of the inverse relation  $f^{-1}$  to  $f$ .

```
« RCEQ PPAR → eq1 ppar1 « PARAMETRIC eq1 i * 'X' + 'X' i * 'X' +
eq1 'i*X' + 3 →LIST STEQ ERASE DRAX DRAW eq1 STEQ ppar1 'PPAR'
STO FUNCTION PICTURE » »
```

**EXAMPLE 9.** Plot  $f(x) = (x + 1)^3$  with the default viewing window. To see its inverse, clear the graphing screen with **ON** and go to the VAR menu with **VAR**. Press **INV.F** to see plots of  $f$ , the line  $y = x$ , and  $f^{-1}$  drawn sequentially:



## Parametric Curves

Not every curve in the  $xy$ -plane is the graph of a function. For example, a circle is not the graph of a function. More generally, imagine a point  $P$  moving in the  $xy$ -plane in such a way that its coordinates are given as functions of time  $t$ :

$$x = f(t) \text{ and } y = g(t).$$

We call  $t$  a *parameter* and call the curve that is traced by the moving point a *parametric curve*.



**EXAMPLE 10.** The coordinates of a moving point are given by

$$x = 2 \cos 2t, \quad y = t - 3 \sin 2t \quad \text{for } 0 \leq t \leq 4.5.$$

Plot the curve and determine the location of the point at  $t = 2$ .

### Using the Parametric Plot Form

Access the PLOT screen with . If the plot type does not already read Parametric, open the CHOOS box and select Parametric. Set the angle display mode to Rad (for this example and any others that use trigonometric functions). Parametric plots require that the expression(s) for EQ appear as complex-valued functions: functions like

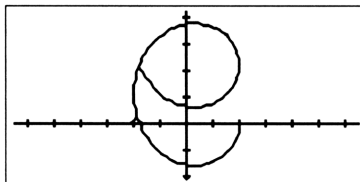
$$f(t) + i * g(t)$$

where  $x = f(t)$  and  $y = g(t)$  give the  $x$ - and  $y$ - coordinates of the moving point.

Therefore, enter the following expression into the EQ field:

$$'2 * \cos(2 * T) + i * (T - 3 * \sin(2 * T))'$$

Now set the independent variable (INDEP:) to 'T', set H-VIEW: -6.5 6.5, and set V-VIEW: -3 6. Open the OPTS submenu and set the independent variable to range from LO: 0 to HI: 4.5. Check AXES and CONNECT. Return to the previous screen with  and ERASE and DRAW to see the following parametric plot. Note that it is traced in a clockwise direction:



To get the *approximate* location of the moving point when  $t = 2$ , trace clockwise along the plot with coordinate readouts active. Notice that the screen shows pixel

coordinates as  $t$  varies. We can only get close to  $t = 2$  with  $t = 2.01$ , and for this value of  $t$  the pixel coordinates of the point are approximately  $(-1.28, 4.31)$ . We can determine the *exact* location of the point when  $t = 2$  as follows.

Return to the stack display screen and access the **SOLVE** menu with  $\boxed{\leftarrow}$  **SOLVE**. Open the **ROOT** menu, then the **SOLVR** submenu. Now input the value 2 for  $\boxed{T}$  and press  $\boxed{\text{EXPR=}}$  to see

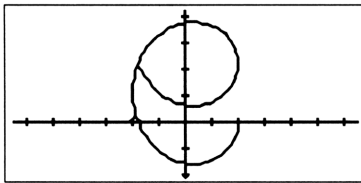
Expr:  $-1.30728724173 + i * 4.27040748592$ .

Thus the exact coordinates of the point when  $t = 2$  are

$$x = -1.30728724173 \text{ and } y = 4.27040748592.$$

## Using the PLOT menu

Access the **PLOT** menu with  $\boxed{\leftarrow}$  **PLOT**. If necessary, open the **PTYPE** menu and press  $\boxed{\text{PARA}}$  to select **PARAMETRIC** plot mode. Enter ' $2 * \cos(2 * T) + i * (T - 3 * \sin(2 * T))$ ' onto stack level 1 and load it into **EQ** with  $\boxed{\leftarrow}$  **EQ**. Now open the **PPAR** submenu. Key in the expression  $\{ T \ 0 \ 4.5 \}$  and touch  $\boxed{\text{INDEP}}$  to specify the independent variable as  $T$  with a range from 0 to 4.5. Type  $-6.5 \ 6.5$  and touch  $\boxed{\text{XRNG}}$  to set the  $Xrng$ , then type  $-3 \ 6$  and touch  $\boxed{\text{YRNG}}$  to set the  $Yrng$ . Return to the previous page, open the **PLOT** submenu, and **ERASE**, **DRAX**, and **DRAW** the plot:



You can obtain the coordinates when  $t = 2$  as above.

When plotting in **PARAMETRIC** mode, you are free to specify any variable as the independent variable, not just ' $T$ '. Unless you specify a range of values for that

variable, the HP-48G/GX will by default use the values specified by the *Xrng*; this is likely not to be the best choice.

You should recall that the ellipse given by  $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$  has the parameterization

$$x = a \cos t, \quad y = b \sin t \quad \text{for } 0 \leq t \leq 2\pi.$$

If we take  $a = b$  then the circle  $x^2 + y^2 = a^2$  has the parameterization

$$x = a \cos t, \quad y = a \sin t \quad \text{for } 0 \leq t \leq 2\pi.$$

Of course, any function  $y = f(x)$ ,  $a \leq x \leq b$  can be parameterized by

$$x = t, \quad y = f(t) \quad \text{for } a \leq t \leq b.$$

We shall consider more exotic parametric curves in the activities.

## Activity Set 1.2

1. (a) Plot the graph of  $y = \frac{\sin x}{x}$  using the default plotting parameters.  
 (b) ERASE and then plot the graph of  $y = \frac{\sin(-2x)}{x}$ .  
 (c) ERASE and then plot the graph of  $y = \frac{\cos x - 1}{x}$ .
2. (a) Plot the list { 'SIN(4 \* X)' '-2\*SIN(X)' } using the default plotting parameters.  
 (b) ERASE and plot the sum of the two functions in the list.  
 (c) Overdraw your plot in (b) with the plot of  $y = -2 \sin x$ .  
 (d) ERASE and plot the product of the two functions in the list.  
 (e) Overdraw your plot in (d) with the plot of  $y = -2 \sin x$ .

3. Plot  $y = \cos(10\pi x)$  on the default plotting screen. Why does the plot look this way? Adjust the screen to make the plot look more like a cosine curve.
4. Set your calculator to degree mode and plot  $y = \sin(x^\circ)$  using the default screen. Without changing back to radian mode, zoom on  $X$  to make the plot look like  $\sin x$ ,  $x$  in radians. When you're done, reset to radian mode.
5. Graph  $y = x^3 - 1.3x^2 + .32x - .02$  using the default plotting screen. Examine the behavior of this function near the origin by using **BOXZ** several times.
6. To appreciate how "steep" are the graphs of simple polynomial functions, begin by plotting  $y = 34x^3 - 91x^2 - 117x + 54$  on the default screen. Now zoom out along the  $y$ -axis as necessary until you can see all the high points and low points (*local extreme points*).
7. Plot  $y = \cos(\cos^{-1}x)$  using the default screen. Is the plot what you expected? Now **ERASE** and plot  $\cos^{-1}(\cos x)$ . Can you explain what you see?
8. Investigate, graphically, the following limit:

$$\lim_{x \rightarrow 0} \left( \frac{x + |1 - \sqrt{x+1}|}{|1 - \sqrt{x+1}|} \right)$$

(see Activity 6 in Activity Set 2.1):

9. Graphically investigate the behavior of  $f(x) = \sin\left(\frac{1}{x}\right)$  near  $x = 0$ . Begin by plotting on the default screen, then use **BOXZ**. What is your conclusion?
10. (a) Plot  $y = \begin{cases} -x & x < 0 \\ \sin x & 0 \leq x < \pi \\ x - \pi & \pi \leq x \end{cases}$ , using the default plotting screen.
- (b) Recall **EQ** to the stack, change it's sign with  $\boxed{+/-}$  and then overdraw the original plot with this expression.

11. Plot  $y = x\sqrt{3 - x^2}$ . Adjust the viewing screen to make the plot touch the  $x$ -axis at the end points of the domain.
12. Graph  $y = x^3 - 9x^2 + 2x + 48$  with the default plotting screen, then zoom out on the vertical axis by a factor of 16 to see the local maximum. Now move the cursor to the point  $(4, 0)$ , open the **ZOOM** menu and press the menu key CNTR on the second page to relocate the center of the viewing window. You may want to remove the menu key labels to see the local minimum. When you have finished, use **ZLAST** to zoom to the last screen. When the plot is done, use **ZLAST** again.
13. (a) Plot  $y = x^2 + \frac{4}{x}$  on the default plotting screen, then zoom out on the vertical axis by a factor of 4. Use **TRACE** to approximate the local minimum value to the right of the origin.  
 (b) Plot  $y = \frac{x^3 - 1}{x - 1}$  on the default plotting screen, then relocate the center of the viewing rectangle at  $(0, 2)$ . Where is the "hole" in the graph?
14. (a) Use the default plotting screen to plot  $f(x) = 2x - 3$ , then use the **INV.F** program to plot  $f^{-1}$ .  
 (b) Write an equation for  $f^{-1}$ .  
 (c) **ERASE**, then plot  $g(x) = -.6x + 1$  and its inverse. When you've finished, write an equation for  $g^{-1}$ .  
 (d) What is your observation about the slopes of non-parallel lines that are symmetric to the line  $y = x$ ? Prove it.  
 (e) Is the converse to your observation true?

15. Let  $u(x) = x^2 + x + 1$  and  $v(x) = \sin x$ .

- (a) Plot the composite function  $f(x) = u[v(x)]$  on the default plotting screen and compare it with the graph of  $v(x)$ .
- (b) Now **ERASE**, plot the composite function  $g(x) = v[u(x)]$  on the default plotting screen and compare it with the plot of  $u(x)$ .

16. Use the default plotting parameters to graph

(a)  $y = x^{2/3}$       (b)  $y = 3(x - 2)^{2/3} + 1$

17. (a) Use the **XROOT** command to plot  $y = 2(x + 2)^{2/3} + \frac{x - 4}{x^2 + 1}$ . Use the default plotting screen.

(b) Zoom in on both axes by a factor of .6. Trace to obtain an approximation to the local maximum to the left of the origin.

(c) Now trace to find the approximate location of the local minimum that is nearest to the origin; with the cursor resting at that point, press ENTER to record the coordinates on the stack.

18. The HP-48G/GX command **IP** will return the integer part of any real number on the stack. Thus, to determine whether a real number  $X$  is an integer, we need only test  $X$  against **IP(X)**:  $X$  is an integer iff  $X$  is the same as **IP(X)**. The syntax to test  $X$  against **IP(X)** is ' $X == \text{IP}(X)$ ', and you can find the **==** command on the **PRG TEST** menu. Use these ideas to graph each of the following functions on the default plotting screen.

(a) 
$$f(x) = \begin{cases} 1 & \text{if } x \text{ is an integer} \\ x^2 + 2x - 1 & \text{if } x \text{ is not an integer} \end{cases}$$

(b) 
$$g(x) = \begin{cases} 1 - x & \text{if } x \text{ is an integer} \\ 1 + x & \text{if } x \text{ is not an integer} \end{cases}$$

19. Plot the parametric curve traced by a point  $P$  moving in such a way that the coordinates are given by the equations

$$x = t - 2 \sin 3t, \quad y = 2 \cos 2t \quad \text{for } 0 \leq t \leq 6.3.$$

Give the exact location of  $P$  when  $t = 3$ . Use  $Xrng$ : -3.5 9.5 and  $Yrng$ : -2 2.

In activities 20 - 25, draw a plot of the indicated parametric curves on the default plotting screen. Go to the MTH CONS menu to get a 12-digit approximation for  $\pi$ .

20.  $x = 4 \cos t, \quad y = 2 \sin t \quad \text{for } 0 \leq t \leq 2\pi$

21.  $x = 3 \cos t + 2 \cos 3t, \quad y = 3 \sin t - 2 \sin 3t \quad \text{for } 0 \leq t \leq 2\pi$

22.  $x = 2 \cos 3t, \quad y = \sin 7t \quad \text{for } 0 \leq t \leq 2\pi$

(When done, zoom in using ZOOM factors of 2.)

23.  $x = 3 \cos^3 t, \quad y = 3 \sin^3 t \quad \text{for } 0 \leq t \leq 2\pi$

24.  $x = \sec t, \quad y = \tan t \quad \text{for } 0 \leq t \leq 2\pi$

25.  $x = 2 \cos t - 1.5 \cos 3t, \quad y = 2 \sin t - 1.5 \sin 3t \quad \text{for } 0 \leq t \leq 2\pi$

# 2

## DERIVATIVES

Elementary calculus is concerned with the mathematics of continuous change. For example, given a function  $f$  whose graph is smooth, the rate of change of  $f$  at a point  $P = (x, f(x))$  on the graph is given by the intuitive notion of the *slope of the graph* at point  $P$ . Calculus provides us with a precise mathematical meaning for this intuitive notion by defining the derivative  $f'(x)$  of  $f$  at  $x$ , then declaring the slope of the graph at  $P$  to be the derivative.

### 2.1 APPROXIMATING SLOPES

#### Difference Quotients

Given a function  $f$ , the *derivative* of  $f$  is the function  $f'$  given by

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}.$$

The geometry is clear enough. The difference quotient

$$\frac{f(x+h) - f(x)}{h}$$

appearing in the definition of  $f'$  is the slope of the secant line joining the point  $(x, f(x))$  on the graph of  $f$  with some nearby point  $(x+h, f(x+h))$  on the graph. Thus the derivative can be viewed geometrically as the limiting position of the slopes of nearby secant lines. For a given  $x$ , we can approximate  $f'(x)$  numerically by evaluating the difference quotient  $\frac{f(x+h) - f(x)}{h}$  for suitably small values of  $h$ .

A simple way to do this on the HP-48G/GX is to evaluate a user-defined function for the difference quotient:



$$DQ(X,H) = \frac{F(X+H) - F(X)}{H}$$

This procedure requires that we also build a user-defined function  $F$  for the given function  $f$ .

To illustrate, consider the function  $f(x) = (x^2 + 5)^3$ . We create a user-defined function  $F$  for  $f$ : «  $\rightarrow X \text{ ' } (X^2 + 5)^3 \text{ '}$  »; and another,  $DQ$ , for the difference quotient: «  $\rightarrow X \text{ H ' } (F(X+H) - F(X))/H \text{ '}$  ». To approximate  $f'(2)$ , we simply evaluate  $DQ$  using input values  $(2, H)$  for varying values of  $H$ .

H	DQ(2, H)
.001	972.67528
.0001	972.0675
.00001	972.0067
.000001	972
-.001	971.32528
-.0001	971.9325
-.00001	971.9933
-.000001	972

This numerical investigation should convince you that  $f'(2) = 972$ .

However, you must exercise caution with the numerical computation of difference quotients because they are susceptible to serious *cancellation error* with the finite precision arithmetic used in any machine computation. For example, consider the function

$$f(x) = \frac{\sqrt[3]{1 + \cos^2 x}}{x^3}.$$

If you build a user-defined function for  $f$  and then evaluate DQ for the following input values  $(1, H)$  you will obtain these results:

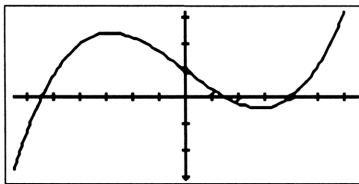
H	DQ(1, H)
$10^{-4}$	-3.5221718
$10^{-5}$	-3.522835
$10^{-6}$	-3.52291
$10^{-7}$	-3.523

The correct value is  $f'(1) = -3.5229074056$ , so you can see that we are losing digits with each successive evaluation of the difference quotient.

## Slopes by Zooming

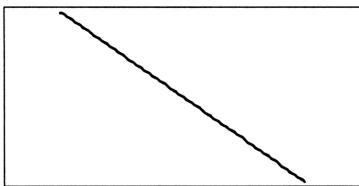
Because there is a strong element of geometry underlying the definition of the derivative, it is not surprising that graphical investigations can often help in building an understanding of the concepts that surround derivatives. By zooming in on a graph, we can often "see" the slope at a point.

**EXAMPLE 1.** We wish to "see" the slope of the graph of  $f(x) = 2x^3 - 3x + 1$  at the point  $(-2, 1.584)$ . Begin by drawing a plot of the graph of  $f(x) = 2x^3 - 3x + 1$  on the default screen, then zoom in on the horizontal axis by a factor of 4 to obtain a better view:



Activate TRACE and the coordinate readout  $(X, Y)$ , trace to the point on the curve where  $x = -2$ , reset both zoom factors to 100 with recentering at the crosshairs, and

then zoom in. Again, with the cursor resting on the curve at  $x = -.2$ , zoom in to see the following approximation to the tangent line at  $x = -.2$ :



To calculate the slope of this line, we choose two points on the line. Trace left to the point  $P$  where  $x = .20001$  and press **ENTER** to record the precise coordinates on the stack. Then trace right to the point  $Q$  where  $x = -.19999$  and use **ENTER** to record the precise location. Return to the stack with **ON** and press  $\boxed{-}$  to calculate the ordered pair  $(\Delta x, \Delta y)$ , where  $\Delta x$  and  $\Delta y$  are the differences in the  $x$  and  $y$  coordinates of  $P$  and  $Q$ , respectively. Use the **C→R** (complex into real) command on the **MTH CMPL** menu to put  $\Delta x$  and  $\Delta y$  on the stack, then **SWAP** and divide to obtain the approximation  $\frac{\Delta y}{\Delta x} = -2.76$  to the slope of the curve at  $x = -.2$ . In this case, we are very accurate: the slope of the curve at  $x = -.2$  is  $-2.76$ .

In the activities that follow, we will use this zooming technique to investigate the slopes of several important functions.

## Activity Set 2.1

1. Can you see the slope of  $y = \sin x$  at  $(0, 0)$ ?
  - (a) Draw a plot of  $y = \sin x$  using the default screen, then zoom in by factors of 100 three times.
  - (b) Trace left to the point  $P$  on the curve where  $x = -.000001$  and record the coordinates on the stack with  $\boxed{\text{ENTER}}$ , then trace right to the point  $Q$  on the curve where  $x = .000001$  and record the coordinate.

- (c) Use  $P$  and  $Q$  to calculate your approximation  $\frac{\Delta y}{\Delta x}$  to the slope as in EXAMPLE 1.
- (d) Find the slope of  $y = \sin x$  at  $(0, 0)$  by evaluating difference quotients.
- (e) What is the slope of  $y = \sin x$  at  $(0, 0)$ ? Express your answer in mathematical terms as the formal limit of a difference quotient involving the *Sine* function.
2. Can you see the slope of  $y = \cos x$  at  $(0, 0)$ ?
- (a) Draw a plot of  $y = \cos x$  using the default screen, then zoom in on the horizontal axis by a factor of 100 (no vertical axis zoom). Zoom in again on the horizontal axis by a factor of 100 (no vertical zoom). With the coordinate readout active, trace along the curve to determine its slope at  $(0, 0)$ .
- (b) Find the slope of  $y = \cos x$  at  $(0, 0)$  by evaluating difference quotients.
- (c) Express the slope of  $y = \cos x$  at  $(0, 0)$  in mathematical terms as the formal limit of a difference quotient involving the *Cosine* function.
3. Use zoom in (with the same horizontal and vertical factors) to estimate the slope of each of the following functions at the point where  $x = 1$ . Be sure to check (✓) RECENTER AT CROSSHAIRS to keep the zooming region centered on the screen, and always make sure that the cursor is resting on the curve at the desired point.
- (a)  $y = x^{-2/3}$       (b)  $y = \frac{1}{x^2} - 2$       (c)  $y = \sin(x^2 - 1)$       (d)  $y = \sin e^x$

4. Use difference quotients to estimate the following slopes:

(a) the slope of  $y = \sqrt{3x - 2}$  at  $x = 2$

(b) the slope of  $y = \frac{2x}{x^3 - 1}$  at  $x = -1$

## 2.2 DERIVATIVES WITH THE HP-48

It is important that students learn the basic mechanics of finding derivatives without their calculators. However, there are times when it is perfectly natural to use the calculator to take derivatives; for example, when we want to plot a function and its first two derivatives, and then find the roots. Since the plotting and root-finding will be done on the HP-48, we may as well do the differentiation process there also.

### The Derivative Function $\partial$

The HP-48 uses the derivative function  $\partial$  ( $\partial$  is the right-shifted  $\boxed{\text{SIN}}$  key) to perform symbolic differentiation. The differentiation can be executed all at once or in step-by-step fashion following the chain rule. In either case, you must specify the expression that is to be differentiated and also the variable of differentiation. In order to obtain symbolic results, the HP-48 must be set to display symbolic results (the default state) and no numerical value should be stored for the variable of differentiation.

### Using the Stack

To perform symbolic differentiation on the stack all at once, the two inputs are specified on the stack:

level 1: 'the expression to be differentiated'

level 2: 'the variable of differentiation'

Then execute the  $\partial$  command with  $\boxed{\partial}$ .

**EXAMPLE 2.** To differentiate  $f(x) = \sin x^2$ , arrange the stack as follows:

level 2: 'SIN(X ^ 2)'

level 1: 'X'

then press the  $\partial$  key to see the symbolic result

level 1: 'COS(X ^ 2) \* (2 \* X)'.

Recall that the chain rule says (in mixed notation) that

$$\frac{d}{dx} f[g(x)] = f'[g(x)] g'(x).$$

To perform symbolic differentiation of  $f(x) = \sin x^2$  on the stack in step-by-step fashion following the chain rule, begin with the expression ' $\partial X(\text{SIN}(X ^ 2))$ ' on level 1:

level 1: ' $\partial X(\text{SIN}(X ^ 2))$ '.

Press  $\boxed{\text{EVAL}}$  to perform one step of the differentiation and obtain:

level 1: 'COS(X ^ 2) \*  $\partial X(X ^ 2)$ '.

Press  $\boxed{\text{EVAL}}$  again to perform the second step:


level 1: 'COS(X ^ 2) \* ( $\partial X(X) * 2 * X^{(2 - 1)}$ )'.

Finally, press  $\boxed{\text{EVAL}}$  again to execute the final step:

level 1: 'COS(X ^ 2) \* (2 \* X)'.

As an alternative to keying in ' $\partial X(\text{SIN}(X ^ 2))$ ' directly to stack level 1, you can use the *Equation Writer*. The Equation Writer is an environment that enables you to enter mathematical expressions and text in much the same way they are written by hand. Activate the Equation Writer with  $\boxed{\leftarrow}$   $\boxed{\text{ENTER}}$ , then use the  $\partial$  key to begin the expression. When you see the form



$$\frac{\partial}{\partial \square}$$

respond by entering  $X$  and then press the  key to move away from the denominator and obtain




$$\frac{\partial}{\partial x} (\square).$$

Now enter  $\text{SIN}$  and then  $X$  to see





$$\frac{\partial}{\partial x} (\text{SIN}(X \square).$$

Close the two parentheses with   to obtain


$$\frac{\partial}{\partial x} (\text{SIN}(X)) \square.$$

Our use of the  key in this illustration is typical: in the Equation Writer, the  key is used to complete any *subexpression* and move on to the next part. Press  to convert the expression into ' $\partial X(\text{SIN}(X))$ ' on level 1.

## Using the Symbolic Differentiate Screen

Go to the Symbolic application with  , highlight *Differentiate* and press . When the Differentiate screen appears, enter ' $\text{SIN}(X \wedge 2)$ ' into the **EXPR** field and ' $X$ ' into the **VAR** field. With the result type specified as Symbolic press  to see the result returned all at once to stack level 1:

$$1: \text{'COS}(X \wedge 2) * (2 * X) \text{'}$$

To obtain the symbolic derivative in step-by-step fashion, return to the Symbolic application and select *Differentiate* as before. After entering ' $X$ ' into the **VAR** field, press . The result of the first step of the differentiation process will appear on level 1:

$$1: \text{'COS}(X \wedge 2) * \partial X(X \wedge 2) \text{'}$$

As before, each succeeding press of the **Eval** key will perform another step of the differentiation.

**EXAMPLE 3.** If you put your calculator in degree mode and take the derivative of a trigonometric function, say  $f(x) = \sin x$ , you will see 'COS(X) \* ( $\pi/180$ )'. Why the factor  $\pi/180$ ? There are several ways to explain this.

When  $x$  is measured in radians, we know that  $\frac{d}{dx}(\sin x) = \cos x$ . Thus by the chain rule, we have  $\frac{d}{dx}[\sin(x^\circ)] = \frac{d}{dx}[\sin(\frac{\pi}{180}x)] = \cos(\frac{\pi}{180}x) \frac{d}{dx}(\frac{\pi}{180}x) = \cos(x^\circ)(\frac{\pi}{180})$ .

For an explanation at the more fundamental level, recall the derivation of the derivative of the sine function:

$$\begin{aligned}\frac{d}{dx}(\sin x) &= \lim_{h \rightarrow 0} \frac{\sin(x+h) - \sin x}{h} \\ &= \lim_{h \rightarrow 0} \frac{(\sin x \cos h + \cos x \sin h) - \sin x}{h} \\ &= \lim_{h \rightarrow 0} \left( \sin x \cdot \frac{\cos h - 1}{h} \right) + \lim_{h \rightarrow 0} \left( \cos x \cdot \frac{\sin h}{h} \right) \\ &= \sin x \cdot \left( \lim_{h \rightarrow 0} \frac{\cos h - 1}{h} \right) + \cos x \cdot \left( \lim_{h \rightarrow 0} \frac{\sin h}{h} \right).\end{aligned}$$

Thus, the result depends upon the two limits

$$\lim_{h \rightarrow 0} \frac{\cos h - 1}{h} \quad \text{and} \quad \lim_{h \rightarrow 0} \frac{\sin h}{h}.$$

Whether  $h$  is measured in radians or degrees, we have

$$\lim_{h \rightarrow 0} \frac{\cos h - 1}{h} = 0.$$

When  $h$  is measured in radians, we have seen (see Activity 1 in Section 3.1) that

$$\lim_{h \rightarrow 0} \frac{\sin h}{h} = 1.$$



But when  $h$  is measured in degrees,

$$\lim_{h \rightarrow 0} \frac{\sin h}{h} = \frac{\pi}{180}$$

(see Activity 1, Section 3.2).

Thus, using degree measure we have

$$\begin{aligned} \frac{d}{dx}(\sin x) &= \sin x \cdot \left( \lim_{h \rightarrow 0} \frac{\cos h - 1}{h} \right) + \cos x \cdot \left( \lim_{h \rightarrow 0} \frac{\sin h}{h} \right) \\ &= \sin x \cdot 0 + \cos x \cdot \frac{\pi}{180} \\ &= \cos x \cdot \frac{\pi}{180}. \end{aligned}$$

## Differentiating the XROOT Function

Although the XROOT function is built into the HP-48G/GX, its derivative is *not*. You can, however, differentiate XROOT if you have the following program stored in your HOME directory. It is important that the name *derXROOT* use lowercase letters for *d*, *e*, and *r* followed by XROOT in uppercase letters because this is the syntax recognized by the HP-48's differentiation routine. (*Note*: to obtain lowercase alphabetical characters, use  $\alpha$   $\leftarrow$  D,  $\alpha$   $\leftarrow$  E, etc.)

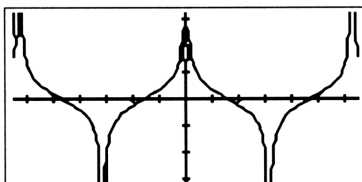
### derXROOT

*Input:* 'XROOT(N, F(X)) on level 1, where  $N > 0$  is an integer

*Effect:* returns  $\frac{d}{dx}(\sqrt[N]{F(X)})$  on level 1

«  $\rightarrow$  n w y z 'INV(n) \* XROOT(n, w) ^ (1 - n) \* z' »

**EXAMPLE 4.** Plot the derivative of  $f(x) = \sqrt[3]{5 \sin x}$  on the default screen. With the program `derXROOT` in the `HOME` directory of your 48G/GX, put '`XROOT(3, 5 * SIN(X))`' on the stack, enter '`X`' and press  $\boxed{\partial}$  to see the derivative '`.333333333333 * XROOT(3, 5 * SIN(X)) ^ - 2 * (5 * COS(X))`'. Now plot on the default screen to see



Notice that the derivative is not defined at the values  $x = n\pi$ ,  $n = 0, \pm 1, \dots$ .

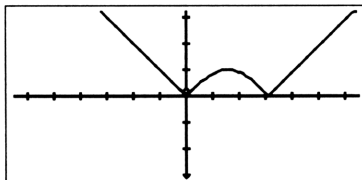
## Piecewise Differentiation

Although the HP-48G/GX will not completely symbolically differentiate a function defined with the `IFTE` command, it will correctly plot the derivative. Here is an example.

**EXAMPLE 5.** Find the derivative of  $f(x) = \begin{cases} -x & x < 0 \\ \sin x & 0 \leq x < \pi \\ x - \pi & \pi \leq x \end{cases}$  and then plot both  $f$

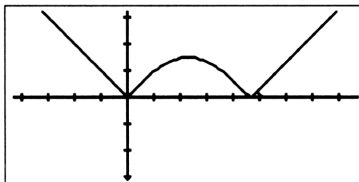
and its derivative.

Put two copies of '`IFTE(X < 0, -X, IFTE(X < pi, SIN(X), X - pi))`' on the stack and graph in *disconnected* mode with the default parameters to see



For greater clarity, especially after we overdraw  $f'$ , trace along the curve to the point where  $x = 1.5$ , open the `ZOOM` menu and press  $\boxed{\text{CNTR}}$ . The plot will be

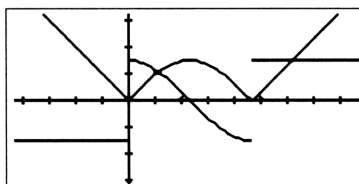
redrawn with the point you choose as center. Now **ZOOM** in on both axes by .67 to see



Press **ON** to return to the stack, put 'X' on level 1 and press **∂** to differentiate.

This gives 'IFTE( $X < 0$ ,  $\partial X(-X)$ ,  $\partial X(\text{IFTE}(X < \pi, \text{SIN}(X), X - \pi))$ )'. Notice that the differentiation is not complete. Use of **EVAL** does not change the expression.

However, if we plot  $f'$  without erasing we see the plot of the derivative superimposed on the plot of  $f$ :



Notice that  $f$  has local minima at values of  $x$  where  $f'(x)$  does not exist.

## Implicit Differentiation

Implicit differentiation is a technique that is used to obtain the derivative  $y' = \frac{dy}{dx}$  when  $y$  is *implicitly* defined as a function of  $x$ .

For example, the equation  $y^3 - xy^2 - y = 5$  implicitly defines  $y$  as a function of  $x$ . To use implicit differentiation, we think of  $y$  as an implicit function of  $x$  and apply the chain rule to differentiate both sides of the equation. The resulting equation can then be solved, if necessary for  $y'$ .

To implicitly differentiate  $y^3 - xy^2 - y = 5$  we proceed as follows:

$$(i) \quad 3y^2y' - (y^2 + x \cdot 2yy') - y' = 0 \quad (\text{apply chain rule})$$

$$(ii) \quad (3y^2 - 2xy - 1)y' - y^2 = 0 \quad (\text{algebra})$$

$$(iii) \quad y' = \frac{y^2}{3y^2 - 2xy - 1} \quad (\text{solve for } y')$$

The HP-48G/GX cannot remember that  $y$  is an implicit function of  $x$ . Instead, we must specify that  $Y$  depends upon  $X$  by using  $Y(X)$  instead of simply  $Y$ . When the calculator takes the derivative, the symbolic derivative of  $Y(X)$  will appear as the expression  $\text{derY}(X, 1)$ . Try it. Put ' $Y(X) \wedge 2$ ' on level 2, and ' $X$ ' on level 1 and press  $\boxed{\partial}$ . You will see ' $\text{derY}(X, 1) * 2 * Y(X)$ ' returned to level 1, the calculator's version of

$$\frac{d}{dx}(y^2) = 2yy'.$$

To avoid having to type  $Y(X)$  in place of  $Y$ , and to make the result appear more like what we are accustomed to writing, we can use a short calculator program. The program given below does the following:

- replaces  $Y$  with  $Y(X)$ ;
- takes the derivative with respect to  $X$ ; then
- replaces  $Y(X)$  with  $Y$  and  $\text{derY}(X, 1)$  with  $y'$  in the resulting expression

#### **IM.y'**

*Input:* level 1: an expression involving  $X$  and  $Y$

*Effect:* differentiates the expression on level 1 with respect to  $X$ ;  
returns an expression for the derivative that uses  $X$ ,  $Y$  and  $y'$ .

```
« { Y 'Y(X)' } ↑MATCH DROP 'X' ∂ { 'Y(X)' Y } ↑MATCH DROP
  { 'derY(X, 1)' y' } ↑MATCH DROP »
```

Thus, with ' $X^2 + Y^2$ ' on level 1, program IM.y' returns ' $2 * X + y' * 2 * Y$ '.

**EXAMPLE 6.** To implicitly differentiate the equation  $y^3 - xy^2 - y = 5$  on the HP-48G/GX, put the equation ' $Y^3 - X * Y^2 - Y = 5$ ' on level 1 and run program IM.y' to see:

$$'y' * 3 * Y^2 - (Y^2 + X * (y' * 2 * Y)) - y' = 0'$$

This is the calculator's version of equation (i) above. You can now isolate  $y'$  and then solve for  $y'$  as in equations (ii) - (iii).

An alternative to the above way of performing implicit differentiation on the HP-48G/GX, we can use a more advanced result that relates implicit differentiation to partial derivatives. Given a function of two independent variables, say  $F(x, y)$ , the *partial derivative*  $F_x$  with respect to  $x$  is obtained by regarding  $y$  as a constant and taking the derivative with respect to  $x$ . For the function  $F(x, y) = y^3 - xy^2 - y - 5$  the partial derivative with respect to  $x$  is  $F_x = -y^2$ . Similarly, we obtain the *partial derivative*  $F_y$  with respect to  $y$  by regarding  $x$  as a constant and differentiating with respect to  $y$ :  $F_y = 3y^2 - 2xy - 1$ . The following result relates implicit differentiation to partial derivatives:

*If the equation  $F(x, y) = 0$  defines  $y$  as a differentiable function of  $x$ , then at any point where  $F_y \neq 0$  we have*

$$\frac{dy}{dx} = \frac{-F_x}{F_y}.$$

Using this result we see that for the example  $F(x, y) = y^3 - xy^2 - y - 5$  we have

$$y' = \frac{dy}{dx} = \frac{-F_x}{F_y} = \frac{y^2}{3y^2 - 2xy - 1}$$

which agrees with our earlier calculation in (iii). Given an equation  $F(x, y) = 0$  that implicitly defines  $y$  as a function of  $x$ , the following program takes as input the algebraic expression  $F(x, y)$  and returns the result  $y' = \frac{-F_x}{F_y}$ .

**y'**

*Input:* level 1: an algebraic expression in terms of  $X$  and  $Y$   
representing  $F(x, y)$

*Effect:* returns to level 1 an algebraic expression of the form  
 $y' = \frac{-F_x}{F_y}$  for the derivative

« { X Y } PURGE DUP 'X' ∂ SWAP 'Y' ∂ NEG / 'y'' →TAG »

**EXAMPLE 7.** To obtain the derivative  $y'$  for the function  $y$  of  $x$  that is implicitly defined by the equation  $y^3 - xy^2 - y - 5 = 0$ , put ' $Y^3 - X * Y^2 - Y - 5$ ' on level 1 and press Y' to see the following result returned to level 1:

$$1: y': ' - (Y^2) / - (3 * Y^2 - X * (2 * Y) - 1) '$$

Compare this to our expression in (iii) above.

As the above example illustrates, implicit differentiation usually results in an expression for the derivative  $y'$  in terms of  $x$  and  $y$ , say  $y' = G(x, y)$ . To evaluate the two-variable function  $G(x, y)$  at a particular point  $(a, b)$ , we can use the following program F.XY.

**F.XY**

*Inputs:* level 3: an algebraic expression  $F(x, y)$  in variables  $X$  and  $Y$

level 2: a real number " $a$ "

level 3: a real number " $b$ "

*Effect:* returns the number  $F(a, b)$  to level 1 and the original expression  $F(x, y)$  to level 2

« 'Y' STO 'X' STO DUP EVAL { X Y } PURGE »

**EXAMPLE 8.** To find the derivatives  $y'(-1, 2)$  and  $y'(3, -4)$  of the function  $y$  implicitly defined by the equation  $y^3 - xy^2 - y = 5$ , at the points  $(-1, 2)$  and  $(3, -4)$ , put 'Y ^ 3 - X \* Y ^ 2 - Y - 5' on level 1 and press  $\boxed{Y'}$  to obtain the symbolic derivative  $y'$ ; '-(Y ^ 2 / - (3 \* Y ^ 2 - X \* (2 \* Y) - 1))' on level 1. now press 1  $\boxed{+/-}$   $\boxed{SPC}$  2  $\boxed{F.XY}$  to see the derivative  $y'(-1, 2) = .26666666666667$  on level 1 and the original symbolic derivative on level 2. Now SWAP levels 1 and 2 and use 3  $\boxed{SPC}$  4  $\boxed{+/-}$   $\boxed{F.XY}$  to see  $y'(3, -4) = .225352112676$  on level 1.

**Activity Set 2.2**

1. (a) Calculate a full precision decimal approximation to  $\pi/180$ .
- (b) Set your calculator to DEGREE mode. Use the SOLVR to numerically investigate  $\lim_{x \rightarrow 0} \frac{\sin x}{x}$  in degree mode. Complete the table:

X	SIN(X)/X
.01	
.001	
.0001	
-.01	
-.001	
-.0001	

(c) Keep your calculator in degree mode and do a graphical investigation of  $\lim_{x \rightarrow 0} \frac{\sin x}{x}$ , as follows.

- Plot 'SIN(X)/X' using the default plotting parameters. What do you see? We need to zoom in.
  - Put the full precision decimal approximation to  $\pi/180$  on level 1. Go to the PLOT menu with  $\boxed{\leftarrow}$   $\boxed{\text{PLOT}}$ , open  $\boxed{\text{PPAR}}$ , go to the next page and press  $\boxed{* \text{H}}$  to rescale the vertical axis so that each tick mark represents  $\pi/180$ . Use  $\boxed{\text{NXT}}$   $\boxed{\text{PLOT}}$  and  $\boxed{\text{DRAW}}$  to redraw your plot of 'SIN(X)/X'. What is  $\lim_{x \rightarrow 0} \frac{\sin x}{x}$  in degree mode?
2. For each of the following functions, find the derivative by hand calculation. As a check on your result, use the HP-48G/GX to calculate the symbolic derivative in a single step as follows:
- using the stack
  - using the Symbolic Differentiate Screen.



$$(a) \quad y = \frac{x}{x^2 + 1} \qquad (b) \quad y = \cos \sqrt{x^2 + 1}$$

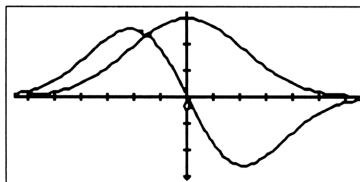
$$(c) \quad y = \sqrt[3]{\sin^2 x} \qquad (d) \quad y = e^{-x/3} \sin(2x)$$

3. (a) Plot  $y = \sqrt[3]{\sin^2 x}$  on the default screen.
- (b) Recall **EQ** to the stack, take its derivative with the HP-48 and then overdraw the plot from (a) with a plot of the derivative.
- (c) For what values of  $x$  is the derivative undefined? What can you say about the function at these values?
4. (a) Plot  $y = \sin |x|$  on the default screen. (Use **ABS(X)** for  $|x|$ .) By examining the plot, can you tell where the derivative will not be defined?
- (b) Use the HP-48G/GX to take the derivative. The term **SIGN(X)** is interpreted as follows:

$$\text{SIGN}(X) = \begin{cases} +1 & \text{for } x > 0 \\ 0 & \text{for } x = 0 \\ -1 & \text{for } x < 0 \end{cases}$$

Overdraw your plot in (a) with a plot of the derivative. Where is the derivative not defined?

5. The following plot shows two graphs, a function  $f$  and its derivative  $f'$ . Which plot is  $f$  and which is  $f'$ ?



6. (a) Draw, on the default screen, a plot of  $y = \frac{3}{2} \tan^{-1} 2x$ . Try hard to visualize a plot of the derivative.
- (b) Confirm (or refute) your visualization efforts by overdrawing a plot of the derivative.
7. Each of the following equations implicitly defines  $y$  as a function of  $x$ . Use program IM.y' to implicitly differentiate the equation, then solve for the derivative  $y'$  by hand.
- (a)  $3x^2 - 9y^3 = 17$  (d)  $y^3 - \sqrt{x} + \cos xy^2 = 8$
- (b)  $x(y^2 + 5x) = 9$  (e)  $x^{2/3} + y^{3/2} = 7$
- (c)  $xe^y + \sin x^2 y - y^2 = 1$  (f)  $\cos x = \sin^2 y$
8. Use program y' to obtain the derivative  $\frac{dy}{dx}$  of each of the implicitly defined functions in Activity 7, then use program F.XY to evaluate the derivative at the indicated point.
- (a) (5, -3) (c)  $(3, \pi/2)$  (e) (8, 9)
- (b) (-2, -3) (d)  $(\pi, 2)$  (f)  $(\pi/4, \pi/4)$

## 2.3 USING THE DERIVATIVE

### Maxima, Minima and Inflection Points

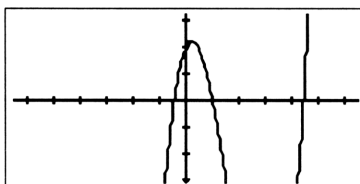
The derivative of a function is the source of considerable information about the behavior of the graph of the function. It can tell us where the graph of the function is increasing and decreasing, help pinpoint the location of local maximum and minimum values on the graph, and show where the graph is concave up and concave down. It is thus advantageous to consider functions and their derivatives from the very beginning of a study of calculus.

A plot of the graph of a function produced on a calculator's screen can often provide valuable information about the behavior of the function. When graphical techniques are carefully combined with an understanding of the derivative as a rate of change, we have a powerful tool for analyzing a function's behavior in considerable detail.

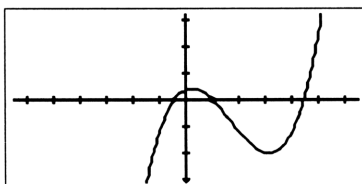
When the **DRAW** command is executed and the HP-48G/GX draws a plot of the graph of a function, the calculator enters the **PICTURE** environment and displays the **PICTURE** menu. In addition to the zoom operations accessible through the **ZOOM** submenu, the **FCN** (= function) submenu contains a number of commands that are helpful in analyzing a function's behavior with calculus without leaving the **PICTURE** environment. Commands such as **ROOT** (to find roots of equations), **ISECT** (for finding intersections of curves), **SLOPE** (for the slope of a graph), **AREA** (for calculating areas of regions beneath curves), **EXTR** (for finding extreme points (i.e., local maxima or minima) on curves), **F'** (to calculate and plot the derivative of a function), and **TANL** (to plot the line tangent to a curve at a point).

**EXAMPLE 9.** Find the  $x$ -intercepts, local maxima and minima, and any inflection points for the function  $f(x) = x^3 - 5x^2 + 2x + 2$ .

Use the default parameters in connected mode to obtain the following plot:



Since the plot goes off screen, we zoom out on the vertical axis. A zoom factor of 5 gives the plot:



This plot shows all of the graph between  $x = -2$  and  $x = 5$ . Before proceeding, we pause to consider what calculus tells us about the graph of this function.

The function is a cubic polynomial, so has at most three real roots. Since we see the plot crossing the  $x$ -axis three times, all the  $x$ -intercepts are displayed. The derivative is a second-degree polynomial and thus has at most two real roots. So the graph of  $f$  can have at most two local extreme points, and because we see a high point and a low point, we certainly have all the local extrema displayed.

The second derivative is a nonconstant linear function having one real root, so the graph of  $f$  has only one inflection point. Since the graph is concave down at the origin and concave up near  $x = 3$ , the inflection point lies between these two points.

With the plot still displayed open the FCN menu.

*To find the  $x$  intercepts:* Move the cursor to the point to the left of 0 where the plot appears to cross the  $x$ -axis and press ROOT. You will see a twelve digit approximation to this root displayed at the bottom of the screen:

ROOT: -.449489742783.

This has also been entered onto the stack. Go to the stack and you will see:

1: Root: -.449489742783.

Now return to the graph with PICTURE (the ◀ key) and find the other two roots in the same way. When you're done, return to the stack to see all three roots. When you find a root of a function in this way, the HP-48 uses the

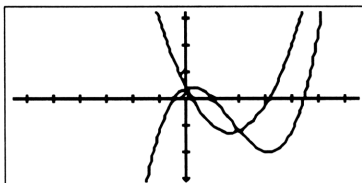
$x$ -coordinate of the cursor as a first approximation for its own ROOT program to numerically approximate the root.

*To find the coordinates of the local extrema:* Move the cursor to the apparent high point of the graph located above the  $x$ -axis just to the right of the origin and press EXTR. The message EXTRM: (.213700352153, 2.2088207353) appears below the graph. This point was also entered on the stack. Now move the cursor to the apparent low point of the graph and again press EXTR. The new message EXTRM: (3.11963298118, -10.0606725872) appears below the graph and this point was entered on the stack.

When you execute the EXTR command, the HP-48 finds the extreme point by a well-known procedure. It finds the derivative  $f'$  of  $f$  and then uses the  $x$ -coordinate of the cursor as a first approximation for the ROOT program to find a root of  $f'$ . Finally, it calculates the value of  $f$  at this root and displays the two coordinates.

*To find the inflection point:* There is no key on the FCN menu to do this so we must use our knowledge of the relation between the function and its derivatives. We know that inflection points occur where the graph of the function changes concavity. And for functions that are everywhere differentiable (such as this one), concavity will change at points where the derivative  $f'$  changes its direction, i.e., at points where the graph of  $f'$  has a local extremum. We can therefore locate the inflection point on the graph of  $f$  by locating the extreme point on the graph of the derivative  $f'$ . This can be done in the PICTURE environment.

With the graph of  $f$  displayed, go to the second page of the FCN menu and press the F' key. This will plot the derivative  $f'$  and then replot  $f$ .



When you press the  $\boxed{F'}$  key, EQ becomes a list  $\{f' f\}$  containing  $f'$  and  $f$ , in this order. The HP-48 commands ROOT, EXTR, etc., apply only to the *first* function in the list, which is now  $f'$ . So move the cursor to the apparent low point of  $f'$  and press  $\boxed{\text{EXTR}}$ . You will see **EXTRM:** (1.66666666667,-6.333333333) displayed at the bottom of the screen. This is the low point on the graph of  $f'$  and we want to use its  $x$ -coordinate as the  $x$ -coordinate of the inflection point of  $f$ .

To get the  $y$ -coordinate of the inflection point  $P$  we must evaluate the function  $f$  at the  $x$ -coordinate of  $P$ . Perhaps the easiest way to do this is to use a short program. The following program assumes that EQ is a list  $\{f' f\}$  composed of  $f'$  and  $f$  in order, and that the coordinates of an extreme point of  $f'$  are displayed on stack level 1. With this input, the program returns the corresponding inflection point of  $f$  with the tag "Infl". The 1 in the name "INFL1" indicates that the first derivative is used in the process.

#### **INFL1** ( Inflection point of $f$ )

*Input:* level 1: the coordinates  $(x_0, y_0)$  of an extreme point of  $f'$

As a stored variable EQ: the list  $\{f' f\}$  consisting of  $f'$  and  $f$

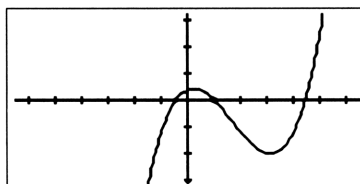
*Effect:* returns to level 1 the point  $(x_0, f(x_0))$  tagged as 'Infl'

« RE EQ 2 GET OVER 'X' STO EVAL R→C 'Infl' →TAG 'X' PURGE »

With this program in your calculator and the extreme point of  $f'$  on stack level 1, press **INFL1** to see

level 1: Infl: ( 1.666666666667, -3.925925926 ).

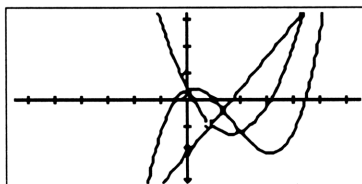
The stack now displays, on levels 6 through 1, the three  $x$ -intercepts (roots), the two extreme points and the inflection point of  $f$ , all with identifying tags. In the display below, we have set the display mode to show only two decimal places to avoid running off the right of the screen, and have again shown the plot of  $f$  to coordinate it with the information about the points of interest:



6:	Root: -0.45
5:	Root 1.00
4:	Root 4.45
3:	Extrm: (0.21, 2.21)
2:	Extrm: (3.12, -10.06)
1:	Infl: (1.67, -3.93)

You will, of course, have to scroll with the **Δ** key to see all six stack levels.

Since the coordinates of extrema can be found from the FCN menu with a single keystroke, it was convenient to find the inflection points of  $f$  from the extrema of  $f'$ . But another way to find inflection points of  $f$  is to find the  $x$ -intercepts of the second derivative  $f''$ , because  $f$  has an inflection point at the values of  $x$  where the graph of  $f''$  crosses the  $x$  axis. We will now use this method to again locate the inflection point of  $f$ . With the graphs of  $f'$  and  $f$  displayed on the screen, press **F'** again and the calculator will plot  $f''$ , then  $f'$  and finally  $f$ .



EQ is now the list  $\{f'' f' f\}$  consisting of  $f''$ ,  $f'$  and  $f$ , in this order. Move the cursor to the root of  $f''$  and press ROOT to display the message ROOT: 1.66666666667 below the graph. To find the value of  $f$  at this  $x$  and then display both coordinates as an inflection point, we will use the following program, INFL2. The 2 in the name indicates that the second derivative was used.

**INFL2** (Inflection point of  $f$ )

*Input:* level 1: the coordinate  $x_0$  of a root of  $f''$

As a stored variable EQ: the list  $\{f'' f' f\}$  consisting of  $f''$ ,  $f'$  and  $f$ .

*Effect:* returns to level 1 the point  $(x_0, f(x_0))$  tagged as 'Infl'

« EQ 3 GET OVER 'X' STO EVAL R→C 'Infl' →TAG 'X' PURGE »

This program assumes that EQ contains a list of  $f''$ ,  $f'$  and  $f$ , in this order, and that a root of  $f''$  is displayed on stack level 1. With this input it returns the corresponding inflection point of  $f$ , tagged "Infl". With the root of  $f''$  displayed on stack level 1, executing INFL2 will give:

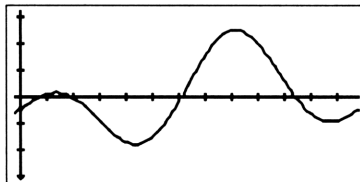
1: Infl : (1.67, -3.93) (the display was again fixed at two decimal places).

The next example uses a trigonometric function and would not be appropriate without the use of technology because of the difficulty of finding roots. With the HP-48G/GX, the procedure is the same as in EXAMPLE 9.



**EXAMPLE 10.** Plot the graph of  $f(x) = \sin(2x) + \cos(x + 2)$ . Find the  $x$ -intercepts and the coordinates of the local extreme points and inflection points.

Since this is a periodic function with period  $2\pi$ , so it is sufficient to find the desired points on the interval  $[0, 2\pi)$ . Plot  $f$  with the  $x$ -range set to  $- .1 \leq x \leq 6.29$  and the  $y$ -range set to  $- 2.5 \leq y \leq 2.5$  to see:



*First, the intercepts:* Move the cursor to each of the four points between 0 and  $2\pi$  where the plot appears to cross the  $x$ -axis and press ROOT on the FCN submenu at each point. Return to the stack display screen to see:

```

4: Root:  .429203673203
3: Root:  .904129660127
2: Root:  2.99852476252
1: Root:  5.09291986492

```

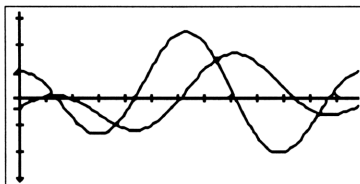
*To find the extreme points:* We proceed as in EXAMPLE 9. Move the cursor to each of the four apparent extreme points and press EXTR at each one. Return to the stack display screen and set the display to show two decimal places. You will see these results:

```

4: Extrm: (0.67, 0.08)
3: Extrm: (2.14, -1.45)
2: Extrm: (4.00, 1.95)
1: Extrm: (5.76, -0.77)

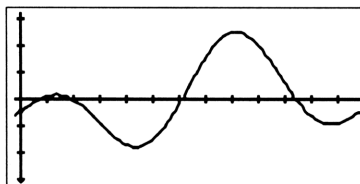
```

*To find the inflection points:* It should be clear from the plot that there is an inflection point between each consecutive pair of extreme points. We will proceed as before by finding the extreme points of  $f'$  and then using program INFL1 to build the inflection points of  $f$ . Retrieve the graph of  $f$  with PICTURE. When we use F' to plot both  $f'$  and  $f$ , the high point of  $f'$  is off screen, so we zoom out on the vertical axis with a factor of 1.5 to see:



Move the cursor to each of the four extreme points of the graph of  $f'$  and press EXTR at each point. Return to the stack display and convert each of the extreme points of  $f'$  to inflection points of  $f$  with program INFL1. You must do some stack manipulation in order to move the extrema of  $f'$  to level 1 in left-to-right order to use with the program. Here's an easy way: with the four extreme points of  $f'$  on the stack, press INFL1 to convert the point on level 1 to an inflection point of  $f$ . Then press the Δ key to activate the *interactive stack* and then use the Δ key to position the pointer ► on level 4 and press ROLLD ENTER. This *rolls* the first four levels of the stack *downward*, moving the first inflection point up to level 4. Now repeat this entire process three more times until all four extreme points are converted and appear in their natural order, left-to-right.

We display again the graph of  $f$  and the points that we have found:

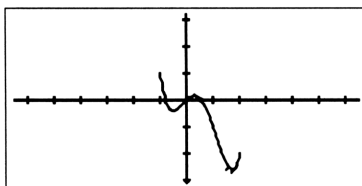


12:	Root:	0.43
11:	Root:	0.90
10:	Root:	3.00
9:	Root:	5.09
8:	Extrm:	(0.67, 0.08)
7:	Extrm:	(2.14, 1.45)
6:	Extrm:	(4.00, 1.95)
5:	Extrm:	(5.76, -0.77)
4:	Infl:	(0.06, -0.35)
3:	Infl:	(1.45, -0.71)
2:	Infl:	(3.09, 0.28)
1:	Infl:	(4.82, 0.64)

In the activities we will examine a function whose inflection points occur where the derivative is not defined. Sometimes we need to find the absolute maximum and absolute minimum values of a function  $f$  on a closed interval  $[a, b]$ .

**EXAMPLE 11.** Find the absolute maximum and minimum values of the function  $f(x) = x^4 - 2x^3 - x^2 + x$  on the interval  $[-1, 2]$ .

We *could* graph  $f$  with  $-1 \leq x \leq 2$  as the  $x$ -range but, instead, we will keep the default  $x$ -range and *restrict the independent variable* to plot only those points satisfying  $-1 \leq x \leq 2$ . The correct plot is

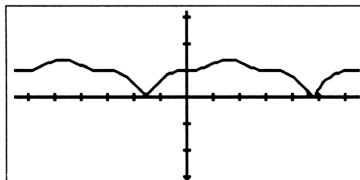


Clearly the absolute maximum value of  $f$  on  $[-1, 2]$  occurs at the left endpoint  $x = -1$ , and the absolute minimum value occurs at the minimum point where  $x$  is approximately 1.5. To find  $f(-1)$ , move the cursor to any point whose  $x$ -coordinate is  $-1$  and press  $\boxed{\text{F(X)}}$  on the second page of the PICTURE FCN menu to see the message  $\text{F(X): 1}$  displayed at the bottom of the screen. So the absolute maximum value of  $f$  on  $[-1, 2]$  is 1, occurring when  $x = -1$ . To find the absolute minimum value of  $f$  from the graph, position the cursor near the apparent minimum point whose  $x$ -coordinate is close to 1.5 and press  $\boxed{\text{EXTR}}$  on the PICTURE FCN menu. The message  $\text{EXTRM: (1.70710678119, -2.66421356232)}$  will appear. Thus the absolute minimum value of  $f$  on  $[-1, 2]$  is  $-2.66421356232$ , occurring when  $x = 1.70710678119$ . Before moving on to the next example, clear your stack and be sure to reset the independent variable to its default state with  $\boxed{\text{RESET}}$ .

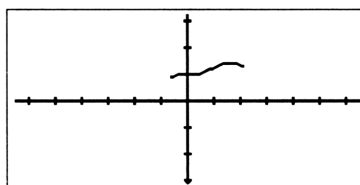
In a situation like that of EXAMPLE 11, if the plot of a function shows an absolute extreme value occurring at the left endpoint of the interval  $[a, b]$  but  $a$  is not a pixel coordinate, say,  $a = \sqrt{2}$  or  $\pi$  for example, then you will have to exit the PICTURE environment to evaluate the function at  $a$ . Of course, the HP-48 will never evaluate a function at  $\pi$ , only at its 12-digit rational number approximation.

**EXAMPLE 12.** Find the absolute maximum and absolute minimum values of the function  $f(x) = \sqrt{1 + \sin^3 x}$  on the interval  $[-\pi/5, 2\pi/3]$ .

We first PLOT the graph with the default parameters to see:



Now restrict the independent variable to plot only the interval  $[-\pi/5, 2\pi/3]$  and redraw:



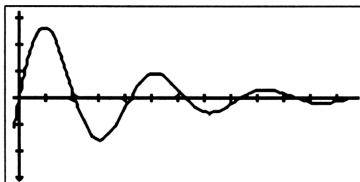
It is clear from the plot that we can use the EXTR command to obtain the absolute maximum:

Extrm: (1.57079632679, 1.41421356237)

We recognize this as the decimal approximation to  $(\pi/2, \sqrt{2})$ . The absolute minimum occurs at the left endpoint of the interval. To evaluate  $f$  there, we return to the stack display screen and go to the SOLVE menu with  $\leftarrow$  SOLVE. Open the ROOT submenu, then the SOLVR submenu. Build the decimal approximation to  $-\pi/5$  with  $\leftarrow \pi 5 + +/ - \rightarrow \text{NUM}$ . Use ENTER to make a duplicate copy, then touch X and EXPR= to see  $y$ -coordinate .892706665066. Thus the absolute minimum point of the graph on the interval  $[-\pi/5, 2\pi/3]$  is approximately  $(-.628318530718, .892706665066)$ .

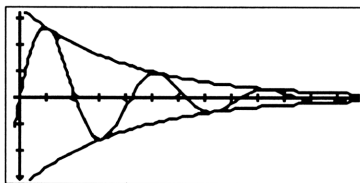
**EXAMPLE 13.** Plot the graph of  $f(x) = 1.7 e^{-x/2} \sin(3x)$  for  $0 \leq x$ .

Since we are interested in the graph only for non-negative values of  $x$ , we set the  $x$ -range as  $-1 \leq x \leq 6.4$  and the  $y$ -range as  $-1.55 \leq y \leq 1.6$ . This halving of both ranges retains equal unit distances (number of pixels per coordinate unit) on both axes and produces the graph:



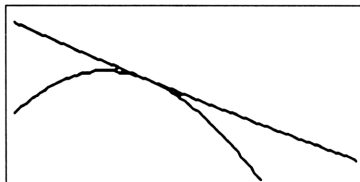
This function (which represents damped harmonic motion) is not periodic and has infinitely many roots, extrema and inflection points for values of  $x \geq 0$ . We could find any of these that we desired by using the techniques described earlier. But in this example, we will use the HP-48G/GX to analyze another aspect of the function's behavior.

Since  $-1 \leq \sin(3x) \leq 1$ , the graph of  $f$  lies between the graphs of  $u(x) = 1.7 e^{-x/2}$  and  $v(x) = -1.7 e^{-x/2}$ , coinciding with the graph of  $u$  when  $\sin(3x) = 1$  and with the graph of  $v$  when  $\sin(3x) = -1$ . We can illustrate this by plotting the list  $\{f u v\}$  using the same plotting parameters that we used for  $f$ . Exit the PICTURE environment, recall  $f$  to the stack with  $\boxed{\rightarrow}$   $\boxed{\text{EQ}}$ , and use  $\boxed{\text{ENTER}}$  to put a second copy on the stack. Edit the copy on level 1 to read ' $1.7 * \text{EXP}(-X/2)$ ', make a second copy of the newly edited expression and then press  $\boxed{+/-}$  to change sign. Use the  $\boxed{\Delta}$  key and the  $\rightarrow\text{LIST}$  command to build the list  $\{f u v\}$ . Now store the list in EQ and graph it to see:



The roots of  $f$  occur where  $\sin(3x) = 0$ , that is, at the roots of  $\sin(3x)$ . **Question:** *do the extrema of  $f$  occur at the extrema of  $\sin(3x)$ , that is, at the points of coincidence of  $f$  with  $u$  or  $v$ ?*

We investigate this question both analytically and graphically. Move the cursor to the first maximum point to the right of the  $y$ -axis and press EXTR to see **EXTRM:** (.468549216461, 1.32664626947) at the bottom of the plot screen. If this were the point where  $\sin(3x) = 1$ , then its first coordinate should be  $\pi/6$ . But  $\pi/6 \approx .523598775598$ , so the extreme points of  $f$  do not coincide with those of  $\sin(3x)$ . We can illustrate this graphically by using **BOXZ** to zoom in on the region of the graph around the first maximum point to the right of the  $y$ -axis:



The maximum point of  $f$  is clearly seen to be to the left of the point where the graph of  $f$  intersects the graph of  $u$ . With some analysis of the derivative, you can show that successive extrema of  $f$  occur every  $\pi/3$  units along the  $x$ -axis, as do successive points of coincidence of  $f$  with  $u$  or  $v$ . So the spacing shown between an extreme point and the corresponding point of intersection with one of the bounding graphs is constant.

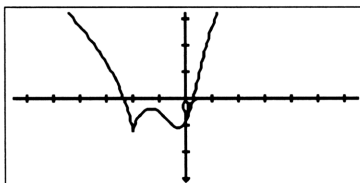
## Caution

When you execute the **EXTR** command on the **PICTURE FCN** menu, the HP-48G/GX takes the derivative of the expression stored in **EQ** and then finds the value of  $x$  closest to the cursor that causes the derivative to evaluate to 0. Thus, if the  $x$ -coordinate of the extreme point that you are finding is a root of the derivative, you are using the **EXTR** command in the way in which it was designed to be used. But, if

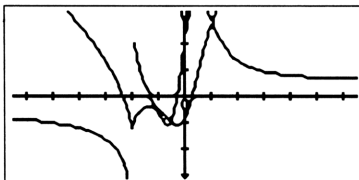
the extreme value of  $f$  does *not* occur at a root of the derivative, you should not use this command.

**EXAMPLE 14.** Find the roots, extrema and inflection points of the function  $f(x) = 2(x + 2)^{2/3} + \frac{x - 4}{x^2 + 1}$ .

Put '2 \* XROOT(3, (X + 2)^2) + (X - 4)/(X ^ 2 + 1)' on level 1 and plot with the default parameters to see:



We can find the two roots in the usual way, by moving the cursor to each of them and pressing **ROOT** on the PICTURE FCN menu. We can find the local maximum point near  $x = -1$  and the local minimum near  $x = -3$  by moving the cursor near these points and pressing **EXTR**. However, if we move the cursor to the minimum point where  $x = -2$  and press **EXTR**, we get **EXTRM: (-7.52928344591E213, 7.68302819356E142)** which is nonsense. From the graph,  $f$  clearly has a minimum at  $x = -2$  and  $f(-2) = 0 + \frac{-6}{5} = -\frac{6}{5}$ . The problem is that  $f$  has no derivative at  $x = -2$  so the **EXTR** approach is not appropriate. If we press **F'** on the PICTURE FCN menu to plot both  $f$  and  $f'$  we see:



The plot makes it clear that  $f'$  does not exist at  $x = -2$ . Since the inflection points of  $f$  occur at values of  $x$  where  $f'$  has extrema, we move the cursor near the local



minimum of  $f'$  to the left of the origin and press EXTR to obtain (-.661278286618, -1.07868129833). Now return to the stack, open the VAR menu and use INFL1 to build the inflection point as

Infl: (-.661278286618, -.81375384108).

Similarly, we find the inflection point that lies to the right of the origin to be

Infl: (.464327883331, .74032208835).

### Activity Set 2.3.1

For each of the functions given in Activities 1-18 below, plot the graph and find all local extreme values and inflection points. When a closed interval is given, also find the absolute extreme values on that interval.

1.  $f(x) = x^3 - x + 2$

2.  $f(x) = x^3 - (1.3)x^2 + (.32)x - .02$

3.  $f(x) = x^4 - 2x^3 + 3x - 2$

4.  $f(x) = x^5 + 3x^4 - x^3 - 3x^2 - x + 3$

5.  $f(x) = \frac{4}{2 + x^2}$

6.  $f(x) = \frac{4}{x^2 - 5}$

7.  $f(x) = \sqrt[3]{1 - x^2}$

8.  $f(x) = x + 3 \sin x$ , on the interval  $[0, 2\pi]$ .

9.  $f(x) = \sin x + 2 \cos(3x)$  on  $[0, \pi]$

10.  $f(x) = \cos 2x - \sin x$  on  $[0, \pi]$

11.  $f(x) = \sin(3x) - \cos(2x)$ ,  $0 \leq x \leq 2\pi$

12.  $f(x) = \begin{cases} 1 + x^2 & x < 0 \\ \cos x & 0 \leq x < \pi \\ \pi - x & \pi \leq x \end{cases}$

13.  $f(x) = 3e^{-x^2/4}$

14.  $f(x) = x^x$

15.  $f(x) = \cos(4 \cos^{-1} x)$ .

16.  $f(x) = 1.5 \tan^{-1} 2x$

17.  $f(x) = x^{\sin x}$ ,  $0 \leq x \leq 2\pi$

18.  $f(x) = \frac{5}{1 + 5e^{-x}}$

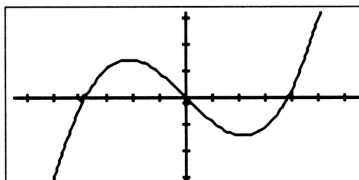
*Activities 19-21 are printed with thanks to Jim Nicholson.*

19. A telephone company plans to run a new telephone line to a customer whose house is located one mile off the straight road along which the telephone lines are run. The new line must go from a junction box on the road to the customer's house. The junction box that is nearest to the house is three miles down the road from the point on the road that is closest to the house. It costs \$100 per mile to run telephone cable along the road and \$150 per mile to run cable off the road. What cable route minimizes total costs?
20. The owner of Big Sky Farm wants to build a rectangular paddock using one side of her horse barn as part, or all, of one side of the paddock. Her barn side is 50 feet in length. There is enough material on hand to build 200 feet of paddock fencing. What dimensions will give a paddock with maximum turn-out area?
21. As an afterthought, the owner of Big Sky Farm needs to use enough of her material to repair 70 feet of existing paddock fencing elsewhere, leaving only enough material to build 130 feet of fencing for the new barn paddock. With only 130 feet available, what dimensions will maximize turn-out area?

## Newton's Method

The technique known as Newton's method has become a classic topic for inclusion in calculus. It is important because it not only invokes the notion of the derivative to produce a simple geometric procedure for finding roots of many functions, but also because it introduces several important ideas: algorithms, recursion, iteration. And it is especially easy to implement on the HP-48G/GX. The iteration formula for Newton's method is  $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ , so we need only iterate the new function  $F(x) = x - \frac{f(x)}{f'(x)}$ .

**EXAMPLE 15.** To use Newton's method to find the roots of  $f(x) = 3x - 4 \sin x$ , we first graph  $f$  to see how many roots there are and to supply first guesses. The plot below is the result of plotting with the default parameters and then zooming in by a factor of .333 on both axes:



We will now create a user-defined function for  $NM(x) = x - \frac{f(x)}{f'(x)}$ . An easy way to do this is to put 'NM(X)', 'X', and two copies of '3 \* X - 4 \* SIN(X)' on the stack, then take the derivative, divide, subtract and equate. The result is 'NM(X) = X - (3 \* X - 4 \* SIN(X))/(3 - 4 \* COS(X))'. Now press **DEF** to create the function NM on the VAR menu.

From the graph, 1.4 appears to be a reasonable first guess, so put 1.4 on the stack and press **NM** to see 1.28871273546 as the next approximation. Press **ENTER** to make a duplicate copy to keep. Now press **NM** again to obtain a second approximation and then **ENTER** to keep a copy. If you repeat this for three more iterations of Newton's method, you will have:

```

5: 1.28871273546
4: 1.27587035767
3: 1.27569814018
2: 1.27569810928
1: 1.27569810928

```

Five iterations have given us successive approximations that agree to 11 decimal places.

The above procedure for building a user-defined function to implement Newton's method for a given function  $f$  can be automated with a short program. Program **NEWTON**, given below, takes an expression for  $f(x)$  from level 1 of the stack and constructs a user-defined function  $NM$  to perform the iteration.

### **NEWTON**

*Input:* level 1: an expression for  $f(x)$

*Effect:* constructs the user-defined function  $NM$  to implement Newton's method.

« 'X' PURGE 'NM(X)' 'X' ROT DUP 'X'  $\partial$  / - = DEFINE 'X' PURGE »




If, for instance, you put ' $3 * X - 4 * \text{SIN}(X)$ ' on level 1 and press NEWT, you can then execute Newton's method from the menu key NM as above.

Newton's method has its limitations. It will obviously not converge if we ever obtain  $f'(x_n) = 0$ . But there can be other causes for its failure. We shall examine some of these in the next set of Activities.

## **Roots**

You should appreciate Newton's method for what it is: a simple iterative procedure, based upon the geometric interpretation of the derivative as the slope of the tangent line, for finding roots of an equation  $y = f(x)$ . But it pales in comparison to the more powerful, robust and sophisticated root-finder that is built in to the HP-48G/GX. We called upon this root-finder when we used the ROOT key on the FCN submenu of the PICTURE environment to find the roots of a function whose plot was displayed. The location of the cursor on the graphics screen provided the initial guess for the procedure which, like Newton's method, is iterative.

The HP-48's root-finder program is the heart of the HP Solve System, which allows you to use menu keys to obtain a numerical solution to any problem that can be expressed in terms of an equation that includes only one unknown variable. The root-finder can be activated in either of two ways:

- with  **SOLVE** , to gain access to the **SOLVE EQUATION** screen;
- with  **SOLVE** **ROOT**, to gain access to the **SOLVE** command menu.

No matter which way you activate the HP Solve system, the general procedure for using it is the same:

- enter the equation you want to solve;
- enter values for all known variables;
- optional: enter an initial guess for the unknown variable;
- solve for the unknown variable.

The local procedures that support this general scheme depend upon how you activate the HP Solve system. Here is a worked example in which the HP Solve system is activated and used each way.

**EXAMPLE 16.** The following equation is often used in financial calculations that involve loan payments:

$$A = P \left[ \frac{(r/12) (1 + r/12)^n}{(1 + r/12)^n - 1} \right]$$

where:  $A$  = monthly payment made at the end of each month

$P$  = total amount of the loan




$n$  = total number of monthly payments




$r$  = annual interest rate (e.g., for 7%,  $r = .07$ ).

Suppose that you are considering different options associated with buying a new car.

- (a) How much would your monthly payments be if you were to borrow \$10,000 for 4 years at 7.5% annual interest?
- (b) How much could you borrow for 4 years at 7.5% interest if you could only afford to pay \$175 per month?
- (c) What annual interest rate would you have to obtain in order to borrow \$9,000 for 4 years with monthly payments of \$215?

### Using the SOLVE EQUATION screen

Use    to gain access to the SOLVE EQUATION screen. The EQ dialogue box will reflect the contents of the current EQ. Enter ' $A = P * (R/12) * (1 + R/12)^N / ((1 + R/12)^N - 1)$ ' into the EQ field.

- (a) Enter .075 into the  $R$  field, 48 into the  $N$  field, and 10,000 into the  $P$  field. *Optional:* as an initial guess for the amount  $A$  of monthly payment, enter 100. If you make no initial guess, the root finder will use the default guess of 0. Now highlight the  $A$  field and press  to see the correct monthly payment of \$241.79 returned to the  $A$  field.
- (b) With the screen set from part (a), enter 175 into the  $A$  field. When the  $P$  field is highlighted, press  to see the correct principal amount \$7,237.71 returned to the  $P$  field.
- (c) With the screen set from part (b), enter 9,000 into the  $P$  field and 215 into the  $A$  field. Highlight the  $R$  field and press  to see the correct interest rate of a little over 6.876% returned to the  $R$  field.

## Using the SOLVE command menu

Return to the stack display screen and purge the variables  $N$ ,  $R$ ,  $P$ , and  $A$  from our previous work on this problem; only the  $EQ$  should remain. Use  $\leftarrow$  **SOLVE** **ROOT** to activate the SOLVE menu. Press **EQ** to verify that the desired expression is present in  $EQ$ . Clear the stack (if necessary) and open the **SOLVR**. You will see input boxes for each of the variables  $A$ ,  $P$ ,  $R$ ,  $N$  at the bottom of the screen, along with an **EXPR=** box. The top of the screen will display the contents of  $EQ$ .

- (a) Store the initial guess 100 into variable  $A$  with 100 **A**, then store the values 10,000 into  $P$ , .075 into  $R$ , and 48 into  $N$  by a similar procedure. Since the value for  $A$  is only an initial (and optional) guess, we need to solve for the correct value of variable  $A$ . To do this, press  $\leftarrow$  **A**. The message at the top of the screen will say Solving for  $A$ . When done, the top of the screen will read **Zero** (to indicate that an exact root of the equation was found) and show the root on level 1: 241.789019379. Thus the monthly payment would be \$241.79.
- (b) Now store the value 175 into  $A$  and solve for  $P$  to see the value 7237.71494872 returned to level 1. Thus, you could only borrow \$7,237.71 with \$175 monthly payments.
- (c) Finally, store the value 9,000 into  $P$ , the value 215 into  $A$ , and solve for variable  $R$ . The correct value is 6.876284944E-2, so you would have to obtain an annual interest rate of 6.876%. Now press **VAR** and purge the variables used in this problem.

To avoid confusion, you should know that there is another **ROOT** command on the HP-48G/GX. It appears on the **ROOT** submenu of the **SOLVE** menu and is useful

for solving in programs. It solves an expression (on level 3) for an unknown (on level 2) using a first guess (on level 1). Try it out now for the function of EXAMPLE 15,  $f(x) = 3x - 4 \sin x$ , with an initial guess in the vicinity of  $x = 1.5$ .

### Activity Set 2.3.2

1. Use Newton's method to find all roots of the following functions:

(a)  $f(x) = x^3 - 3x^2 - 5x + 15$

(b)  $f(x) = \sin x - 2 \cos 3x$  in the interval  $[0, 2\pi]$

(c)  $f(x) = e^x - 2 \cos x$  in the interval  $[-2\pi, 1]$

(d) the *Legendre Polynomial* of degree 3:

$$P_3(x) = \frac{5}{2}x^3 - \frac{3}{2}x$$

(e) the *Chebyshev Polynomial* of degree 4:

$$T_4(x) = 8x^4 - 8x^2 + 1$$

2. Because Newton's method relies on tangent lines to generate a sequence of successive approximations  $x_0, x_1, x_2, \dots$  to a desired root  $r$ , you might expect that the method is somewhat sensitive to the slopes of these tangent lines. Indeed, tangent lines with small slopes often lead us *away* from the root we seek. To see this, try to locate the root  $r = 0$  of  $f(x) = \sin x$  by Newton's method, using the following initial guesses:

(a)  $x_0 = \pi/2$  [What happens here? Why?]

(b)  $x_0 = 1.6$

(c)  $x_0 = 1.5$

(d)  $x_0 = 1.4$

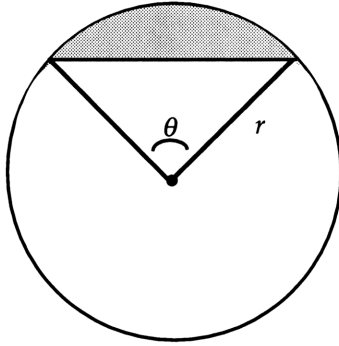


- (e)  $x_0 = 1.3$
  - (f)  $x_0 = 1.2$
  - (g)  $x_0 = 1.1$
3. Apply Newton's method to  $f(x) = \sqrt[3]{x}$ . The graph of  $f$  should help you to understand what is happening here.
4. Apply Newton's method to the function

$$f(x) = \begin{cases} -\sqrt{2-x} & \text{for } x < 2 \\ \sqrt{x-2} & \text{for } x \geq 2 \end{cases}.$$

- (a) Use any convenient initial guess, say  $x_0 = 3$ . When program NM returns a symbolic result, simply press EVAL to evaluate that result and obtain a numerical result.
  - (b) Experiment with several other initial guesses. What is taking place here?
  - (c) To "see" what is taking place, plot the graph of  $f$  on the default plotting screen. Trace along the plot to the point  $P$  where  $x = 3$  (our first initial guess), open the FCN submenu and use TANL to plot the tangent line to  $f$  at  $P$ . Return to PICT, trace along the plot to the point  $Q$  where  $x = 1$  (our second guess when  $x_0 = 3$ ), and use TANL to draw the tangent line to  $f$  at  $Q$ . What is apparent about these tangent lines? Return to the stack and examine their slopes.
5. (a) Plot the list { 'EXP (-X ^ 2)' '.75/(1 + X ^ 2)' } on the default screen, then zoom in using zoom factors of 3 to enlarge the plot.
- (b) Use the command ISECT on the FCN submenu to find the points of intersection of the two plots. ISECT uses the HP Solve system to produce its results.

6. Find the value for  $\theta$  (in degrees) that will give the shaded region an area of  $1.5 \text{ in}^2$  if  $r = 4 \text{ in.}$  (The command  $R \rightarrow D$  on the MTH REAL menu will convert radians to degrees.)



7. It is well-known that the centroid of the St. Louis arch is in the shape of an inverted catenary (hyperbolic cosine). The outside surface is much thicker at the base than at the top and thus is not a true catenary. Nevertheless, we shall model the outside surface as a catenary having both its height and base equal to 630 ft. Since a catenary hanging above the origin with lowest point at  $(0, a)$  has an equation  $y = a \cosh \frac{x}{a}$ , it is easy to see that an equation for the St. Louis arch is

$$y = 630 + a \left( 1 - \cosh \frac{x}{a} \right)$$

for some positive parameter  $a$ . To help determine this parameter, we use the fact that the point  $(315, 0)$  lies on the arch. Use the HP Solve system to determine the parameter  $a$  and then write an equation for the St. Louis arch that is free of unknown parameters. Remember, the HP Solve system only needs an initial guess. The  $\cosh$  command resides on the MTH HYP menu.

## Polynomial Approximations

A great deal of calculus is concerned with approximations. Indeed, approximations lie at the heart of the two main ideas of calculus, the derivative and the integral. The derivative is defined as a limit of approximating slopes and the integral is defined as a limit of approximating sums.

Aside from familiar approximations like  $.3, .33, .333, .3333, \dots \rightarrow \frac{1}{3}$  the simplest approximations in calculus occur when we approximate differentiable functions  $f$  by their tangent lines at points  $x = a$ . Recall that the slope of the tangent line of a function  $f$  at  $x = a$  is given by

$$f'(a) = \lim_{x \rightarrow a} \frac{f(x) - f(a)}{x - a}.$$

Thus, for values of  $x$  close to  $a$ , we have

$$f(x) \approx \frac{f(x) - f(a)}{x - a} (x - a) + f(a),$$

so that

$$(1) \quad f(x) \approx f(a) + f'(a)(x - a).$$

The expression on the right hand side of (1) is a linear polynomial in  $(x - a)$ :

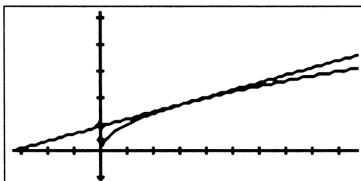
$$P_1(x) = f(a) + f'(a)(x - a)$$

and its graph is the tangent line  $y = f(a) + f'(a)(x - a)$  to  $f$  at  $x = a$ . Of all possible linear polynomials in  $(x - a)$ ,  $P_1(x)$  is the only one that satisfies the two conditions:

- (i)  $P_1(a) = f(a)$   $[P_1 \text{ and } f \text{ agree at } x = a]$
- (ii)  $P_1'(a) = f'(a)$   $[P_1 \text{ and } f \text{ have the same derivative at } x = a]$

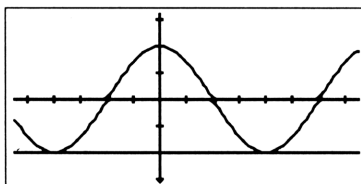
Because conditions (i) and (ii) are enough to completely determine the form of the polynomial  $P_1(x)$ , we call  $P_1(x)$  *the best linear approximation to  $f$  at  $x = a$* . By zooming in enough near the point where  $x = a$ , the graphs of  $P_1$  and  $f$  appear almost identical.

For some functions  $f$ , the best linear approximation at  $x = a$  can be a good one



The best linear approximation to  $f(x) = \sqrt{x}$  at  $x = \pi$  is a good fit to the graph for values of  $x$  near  $x = \pi$ .

But for functions  $f$  having more *curvature* at  $x = a$ , the best linear approximation can be poor:



The best linear approximation to  $f(x) = \cos x$  at  $x = \pi$  is a poor fit to the graph for values of  $x$  near  $x = \pi$  because of the high degree of curvature.

To account for a higher degree of curvature at a point  $x = a$ , we need an approximating polynomial whose higher order derivatives are not all zero at that point.

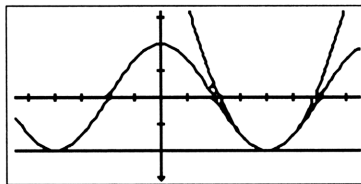
The *best quadratic approximation to the function  $f$  at  $x = a$*  is the quadratic polynomial  $P_2(x)$  in  $(x - a)$  satisfying the three conditions:

- (i)  $P_2(a) = f(a)$  [ $P_2$  and  $f$  agree at  $a$ ]
- (ii)  $P_2'(a) = f'(a)$  [ $P_2$  and  $f$  have the same first and second
- (iii)  $P_2''(a) = f''(a)$  derivatives at  $x = a$ ]

The defining expression is

$$P_2(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2!} (x - a)^2.$$

The plot below shows the best linear and best quadratic approximations to  $f(x) = \cos x$  at  $x = \pi$ .



The best quadratic approximation to  $f(x) = \cos x$  at  $x = \pi$  is a better fit than the best linear approximation.




The best linear and quadratic approximations to a function  $f$  at  $x = a$  are also called the *Taylor Polynomials* of orders 1 and 2 for  $f$  at  $x = a$ . More generally, given a function  $f$  whose first  $n$  derivatives exist in a neighborhood of  $x = a$ , the *Taylor Polynomial of order  $n$  at  $x = a$*  is the polynomial

$$P_n(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2!} (x - a)^2 + \dots + \frac{f^{(n)}(a)}{n!} (x - a)^n.$$

As our last plot suggests, higher order Taylor polynomials at  $x = a$  extend the range of values near  $x = a$  for which we can expect to get reasonably good approximations to  $f$ .


The HP-48G/GX will find Taylor polynomials at  $x = 0$  for any function that it can differentiate, and it is easy to write short programs that extend this capability to the more general case of Taylor polynomials at an arbitrary value  $x = a$ .

## Using the Taylor Polynomial Screen

Access the Taylor Polynomial screen with  **SYMBOLIC**, highlight Taylor Polynomial and press . Enter an expression for the function into the **EXPR** field, say **EXPR:** 'SIN(X)', the variable of differentiation into the **VAR** field, **VAR:** 'X', and the desired order of the Taylor polynomial, say **ORDER:** 3. With the result set to **RESULT:** Symbolic, press  to see the Taylor Polynomial at  $x = 0$  on stack level 1:






level 1: 'X - 1/3! \* X ^ 3'

## Using the TAYLR command

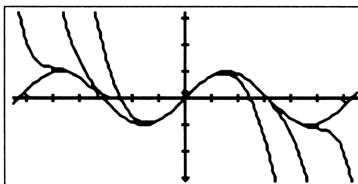
The command **TAYLR**, located on the first page of the  **SYMBOLIC** menu, requires a threefold input: on level 3 the function  $f$  whose Taylor polynomial at  $x = 0$  is desired, on level 2 the independent variable, and on level 1 the degree of the desired polynomial. This command produces Taylor polynomials about  $x = 0$ :

```

3: 'SIN(X)'
2: 'X'
1: 3
1: 'X - 1/3! * X ^ 3'
```

To efficiently graph plot  $f(x) = \sin x$  and its Taylor polynomials  $P_3$ ,  $P_7$  and  $P_{11}$  of orders 3, 7 and 11 at  $x = 0$ , begin with 'SIN(X)' on level 1 and press  three times to make three additional copies. Build the list {'SIN(X)'} by pressing the  followed by  **LIST** . **SWAP** levels 1 and 2, enter 'X' and 3, then press  to build  $P_3(x)$ : 'X - 1/3! \* X ^ 3'. Insert this as the second

element of the list with  $\boxed{+}$ . Now **SWAP** levels 1 and 2 and proceed as before to build  $P_7(x)$  and  $P_{11}(x)$ , adding them to the list as they become available. You can then store the final list into **EQ** and plot with the default viewing screen to see:



Displaying plots is a dramatic way of showing how a function can be approximated by its Taylor polynomials, but you should remember that with the default plotting parameters two plots will coincide for a value of  $x$  if their  $y$ -coordinates are the same when rounded to one decimal digit; ordinarily, this is not good enough for serious numerical approximations.

To find Taylor polynomials centered about an arbitrary point  $x = a$ , you can use program **TAY.A**. Make sure that the independent variable is set to  $X$  and that no value is stored for  $X$  before using the program.

### **TAY.A**

*Input:* Level 3: an algebraic expression for a function  $f$ ,  
in terms of 'X'.

Level 2: the order  $n$  of the desired Taylor polynomial.

Level 1: the new center point,  $a$ .

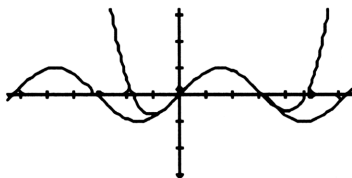
*Effect:* Returns the Taylor polynomial of order  $n$  for function  $f$  at  $x = a$ .

```
« → n a « 'Y' a + 'X' STO EVAL 'Y' n TAYLR 'X' PURGE 'X'
a - 'Y' STO EVAL 'Y' PURGE » »
```

For example, to find the fourth order Taylor polynomial for  $f(x) = \sin x$ , at the point  $x = 2$ , put 'SIN(X)' on the stack, then enter 4 and 2 and press TAY.A to see the calculator's version of

$$0.909297 - 0.416147(x - 2) - .454649(x - 2)^2 + 0.069358(x - 2)^3 + 0.037887(x - 2)^4$$

on level 1. (We set the display to show 6 decimal places.) Plot the list containing  $\sin x$  and this polynomial with the default parameters to see:



Notice that the graphs appear to coincide from near  $x = -0.5$  to near  $x = 3.5$ , that is, on an interval centered about  $x = 2$ .

Although TAY.A does the obvious by making a change of variables  $X = Y + a$  to translate the Taylor polynomial from  $x = 0$  to  $x = a$ , you should be aware of the fact that the symbolic computations required to calculate higher order Taylor polynomials at points  $x = a$  away from  $x = 0$  can be substantial. Thus, as a symbolic processor, you may sometimes find the HP-48G/GX not quite up to the task of finding the Taylor polynomials that you desire if you use TAY.A. For example, the HP-48G runs out of memory (32K RAM) before it can produce the Taylor polynomial of order 7 for  $f(x) = x^{-1}$  at  $x = 2$  and the HP-48GX (with 128K RAM) requires almost 25 minutes to produce this polynomial. The solution is to be a bit more clever in how we approach the symbolics. Program TAYLAT ("Taylor at") is due to Charlie Patton of Hewlett Packard and uses the  $\downarrow$ MATCH and  $\downarrow$  commands to rearrange the symbolic computations. With TAYLAT, you can produce the Taylor polynomial of order 7 for  $f(x) = x^{-1}$  at  $x = 2$  in less than 30 seconds.



**TAYLAT**

*Input:* Level 4: an expression for a function  $f$ .

Level 3: the independent variable.

Level 2: the order  $n$  of the desired Taylor polynomial

Level 1: the new center point  $a$ .

*Effect:* Returns the Taylor polynomial of order  $n$  for function  $f$ , centered about  $x = a$ .

« → XP VA ORD PT « XP VA VA PT + 2 →LIST ↓MATCH DROP  
VA ORD TAYLR VA VA PT - 2 →LIST | » »

**Activity Set 2.3.3**

1. Plot  $f(x) = \tan^{-1}(x)$  on the default screen. Then overdraw the Taylor Polynomials of orders 1 and 3 for  $f$  at  $x = 0$ .
2. Plot  $f(x) = \sec x$  on the interval  $[-\pi, \pi]$ . Then overdraw the Taylor Polynomials of orders 2 and 4 for  $f$  at  $x = 0$ .
3. (a) Plot  $f(x) = \sin 2x - 2 \sin x$  on the default plotting screen and then overdraw the Taylor polynomials of orders 3, 5, 7 and 9 for  $f$  at  $x = 0$ .  
(b) ERASE the plots from part (a) and plot  $f(x) = \sin 2x - 2 \sin x$  using  $Xrng: -2$  6 and  $Yrng: -3.1$  3.2. Now use TAY.A to overdraw with the plot of the Taylor polynomials of orders 2 and 5 for  $f$  centered at  $x = 2$ .

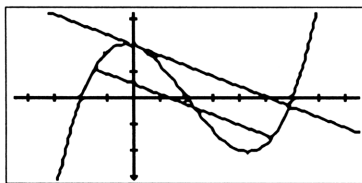
4. (a) Plot  $f(x) = e^{-x^2/2}$  on the default screen, then zoom in with factors of 3.
- (b) Overdraw your plot in (a) with the Taylor polynomials of orders 2 and 4 for  $f$  at  $x = 0$ .
- (c) Now overdraw your plot in (b) with the Taylor polynomial of order 6 for  $f$  at  $x = 0$ . The HP-48G/GX takes quite some time to produce this sixth degree polynomial (approximately 2.5 min), an illustration of how complex the computation of such polynomials can be. To convince yourself, try finding this polynomial by hand.

## Discovering the Mean Value Theorem

The Mean Value Theorem is one of the "gems" of elementary calculus. Its statement is simple, its geometric character makes it believable, and it is an extremely useful result. Indeed, the Mean Value Theorem provides the theoretical basis for a host of other theorems that comprise an important part of differential calculus.

What does the Mean Value Theorem say?

*Given a function  $f$  that is continuous on the closed interval  $[a, b]$  and differentiable between  $a$  and  $b$ , then for some point  $c$  between  $a$  and  $b$ , the tangent line to  $f$  at  $x = c$  has slope equal to  $\frac{f(b) - f(a)}{b - a}$ .*



**Geometrically:** At some point  $c$  between  $a$  and  $b$  the tangent line to  $f$  at  $x = c$  is parallel to the secant line joining  $(a, f(a))$  and  $(b, f(b))$ .

**EXAMPLE 17.** In this example we will apply the Mean Value Theorem to the function  $f(x) = x^3 - 3x^2 - x + 3$  on the interval  $[-.75, 2.6]$  and produce the above plot of the result. Before beginning, purge  $X$  from the current directory and its ancestors.

Begin by plotting  $f(x) = x^3 - 3x^2 - x + 3$  with  $Xrng$ :  $-2.25 \ 4.25$  and  $Yrng$ :  $-4.65 \ 4.8$ . Trace left along the plot to the point  $P$  where  $x = -.75$ , press **ENTER** to record the coordinates  $(a, f(a))$  of  $P$  on the stack, then press **×** to mark the location of the cursor on the plot screen. Now trace right along the plot to the point  $Q$  where  $x = 2.6$  and again press **ENTER** to record the coordinates  $(b, f(b))$  of  $Q$  on the stack. Press **NXT**, open the EDIT menu, and press **LINE** to draw the secant line joining points  $P$  and  $Q$ .

To calculate the slope  $\frac{f(b) - f(a)}{b - a}$  of the secant line, first exit to the stack display screen to see the coordinates of  $P$  on level 2 and  $Q$  on level 1:

2:  $(-.75, 1.640625)$

1:  $(2.6, -2.304)$

Press **−** to calculate  $(a - b, f(a) - f(b))$ :

1:  $(-3.35, 3.944625)$

then divide 3.944625 by -3.35 to obtain the slope  $m = -1.1775$  (Here is an easy way to do the division without retyping the numbers: open the MTH CMPL (= complex) menu, press **C→R** to separate the ordered pair into its two components, then SWAP and divide.) Leave -1.1775 on level 1.

To find a point  $c$  between  $x = a$  and  $x = b$  where the tangent line to  $f$  at  $x = c$  has slope equal to -1.1775, recall EQ to the stack and press **ENTER** to make a duplicate copy. Now take the symbolic derivative of EQ, move -1.1775 from level 3

to level 1 with the command 3 ROLL; and then use  $\boxed{\leftarrow} \boxed{=}$  to equate -1.1775 to the derivative:

$$1: '3 * X ^ 2 - 3 * (2 * X) - 1 = -1.1775'$$

Go to the SOLVE menu with  $\boxed{\leftarrow} \boxed{\text{SOLVE}}$ , open  $\boxed{\text{ROOT}}$  and store the equation on level 1 into EQ. Open  $\boxed{\text{SOLVR}}$ . Since  $a = -.75$  and  $b = 2.6$ , we can use  $x = 0$  as an initial guess for the root-finder; so put 0 into  $\boxed{X}$  and then solve for  $x = c$  with  $\boxed{\leftarrow} \boxed{X}$ . You will see  $X: 3.00343648694\text{E-}2$  returned to level 1. Now SWAP the value for  $c$  on level 1 with the original  $f(x)$  on level 2, restore  $f(x)$  as the EQ with the command STEO, open the VAR menu and purge the value stored in  $\boxed{X}$ . (You should still have the value for  $c$  on level 1.)

To build an equation for the tangent line to  $f$  at  $x = c$ , use the following program TAN.L. Recall that the HP-48G/GX has a menu key on the PICTURE FCN menu for drawing the tangent line to a function whose plot appears on the screen. But this built-in feature uses as input the  $x$ -coordinate of the cursor, and our value  $c$  is not such a point; hence the need for a more general purpose program. At any rate, run program TAN.L now. The program will use the value of  $c$  from level 1 and the expression  $f(x)$  in EQ to calculate and overdraw a plot of the tangent line to  $f$  at  $x = c$ ; for convenience, a copy of the equation of the tangent line is left on level 1 of the stack.

$$1: '-2.9672865388 - 1.1775 * (X - 3.00343648694\text{E-}2)'$$

**TAN.L**

*Input:* Level 1: a real number  $c$  or a complex number  $(c, d)$

As the stored variable EQ: an algebraic expression for a function of  $f$ .

*Effect:* Calculates an expression for the tangent line to  $f$  at  $x = c$ , plots the expression on the existing plotting screen, and returns the expression to level 1 of the stack.

```
« DTAG DUP IF TYPE 0 == THEN 'X' STO ELSE C→R DROP 'X'
STO END EQ DUP EVAL EQ 'X' ∂ EVAL 'X' X - * + 'X' PURGE DUP
STEQ DRAW SWAP STEQ PICTURE »
```

**Activity Set 2.3.4**

1. (a) Plot the function  $y = \frac{1}{x^2}$  using *Xrng*: -5 3 and *Yrng*: 0 4.2.  
 (b) Apply the Mean Value Theorem (as in the last Example) to  $f$  on the interval  $[a, b]$ , where  $a, b$  are the  $x$ -coordinates of points  $P, Q$  obtained as follows: trace left along the curve to the point  $P$  where  $x = .496$ , then trace right along the curve to the point  $Q$  where  $x = 1.44$ . As in the last example, overlay plots of the secant line and tangent line on the plot of  $f$ .
2. (a) Plot the function  $f(x) = \sqrt{x} \sin x$  using *Xrng*: 0 3.14 and *Yrng*: 0 1.5.  
 (b) Apply the Mean Value Theorem to  $f$  on the interval  $[a, b]$  where  $a, b$  are the  $x$ -coordinates of points  $P, Q$  determined as follows: trace left along the curve to the point  $P$  where  $x = .725$ , then trace right along the curve to the point  $Q$  where  $x = 2.15$ . Plot the secant line joining  $P$  and  $Q$ . When you get ready to use the root-finder, use  $x = 0$  as your initial guess.

Overdraw plots of the secant and tangent lines. Does the location of the tangent line at  $x = c$  surprise you? Now seed the root finder with  $x = 1.5$  to find another value for  $c$ . Is the tangent line to  $f$  at this new  $x = c$  the one you expected originally?

3. As Activity 2 shows, there may be more than one value of  $c$  between  $x = a$  and  $x = b$  that meets the conditions of the Mean Value Theorem. Here is a spectacular example.

- (a) Plot  $f(x) = \sin x - 2 \cos 3x$  using *Xrng*: 0 6.28 and *Yrng*: -3.1 3.2.
- (b) Apply the Mean Value Theorem to  $f$  over the interval  $[a, b]$  where  $a, b$  are the  $x$ -coordinates of points  $P, Q$  determined as follows: trace left along the curve to the point  $P$  where  $x = 2.75$ , then trace right along the curve to the point  $Q$  where  $x = 4.64$ . Plot the secant line joining  $P$  and  $Q$ .
- (c) Now find six values of  $c$  between  $a$  and  $b$  that meet the conditions of the Mean Value Theorem. Do this by using the initial guesses for the root-finder of  $x = 1, 2, 3, 4, 5$  and  $6$ . Plot all six tangent lines.

## Parametric Differentiation

How can we find the slope of a smooth parametric curve

$$x = f(t), \quad y = g(t), \quad a \leq t \leq b$$

at a point  $(x_0, y_0)$  on the curve?

If the coordinate functions  $f$  and  $g$  are reasonably "well-behaved", then  $y$  can be expressed as a function of  $x$ , say  $y = F(x)$ . Then, since  $y$  is a function of  $x$  and  $x$  is a function of  $t$  the Chain rule tells us that

$$\frac{dy}{dt} = \frac{dy}{dx} \frac{dx}{dt}.$$

At a point where  $\frac{dx}{dt} \neq 0$  we can then obtain

$$\boxed{\frac{dy}{dx} = \frac{dy}{dt} / \frac{dx}{dt}}.$$

In this last equation,  $\frac{dy}{dt}$  and  $\frac{dx}{dt}$  are the rates of change of the coordinates with respect to the parameter  $t$ , while  $\frac{dy}{dx}$  is the rate of change of  $y$  with respect to  $x$  — the slope of the curve. In case  $\frac{dy}{dt} = 0$  and  $\frac{dx}{dt} \neq 0$ , the slope of the curve is 0, meaning a horizontal tangent line. On the other hand, if  $\frac{dx}{dt} = 0$  and  $\frac{dy}{dt} \neq 0$  then the curve has a vertical tangent line. The case that both derivatives  $\frac{dy}{dt}$  and  $\frac{dx}{dt}$  are 0 is ruled out when we have a smooth curve.

**EXAMPLE 18.** Consider the parametric curve given by  $x = 2 \cos 2t$ ,  $y = t - 3 \sin 2t$  for  $0 \leq t \leq 4.5$ . We first met this curve in Chapter 1.

The slope of the curve at any value  $t$  is given by

$$\frac{dy}{dx} = \frac{dy}{dt} / \frac{dx}{dt} = \frac{1 - 6 \cos 2t}{-4 \sin 2t}.$$

Thus, the slope of the curve at the point where  $t = 2$  is

$$m = \frac{1 - 6 \cos 2(2)}{-4 \sin 2(2)} = 1.62587390888.$$

The point on the curve corresponding to  $t = 2$  is  $(x_0, y_0) = (-1.30728724173, 4.27040748592)$ , so an equation for the tangent line  $y = y_0 + m(x - x_0)$  to the curve at this point is

$$y = 4.27040748592 + 1.62587390888(x + 1.30728724173)$$

or

$$y = 1.62587390888x + 6.39589170366.$$

In addition to plotting parametric curves, the HP-48G/GX can calculate the slope of a smooth parametric curve at a point  $(x_0, y_0)$  and then overdraw a plot of the tangent line to the curve at this point. Instead of performing all of the calculations on the stack, we can use a program to do most of the work. Program PAR' [= PARametric derivative], given below, will calculate the slope  $m = \frac{dy}{dx}$  of a smooth parametric curve at a point  $(x_0, y_0)$  and then determine an equation for the tangent line  $y = y_0 + m(x - x_0)$ .

### PAR'

*Input:* Level 2: a parametric curve '(f(T), g(T))' in terms of the parameter 'T'

Level 1: a value  $t_0$  of the parameter 'T'

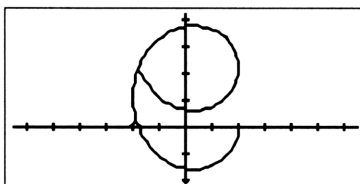
*Effect:* Calculates the slope  $m = \frac{dy}{dx}(x_0, y_0)$  of the curve at  $(x_0, y_0)$ , where  $x_0 = f(t_0)$  and  $y_0 = g(t_0)$ , and returns to level 1 an expression for the tangent line  $y = y_0 + m(x - x_0)$  at  $(x_0, y_0)$ . Displays the message "VERTICAL TANGENT" in the case of a vertical tangent.

```
« 'T' PURGE SWAP OBJ→ DROP2 OBJ→ DROP2 2 →LIST πLIST i
NEG * COLCT DUP2 2 →LIST 'T' ∂ OBJ→ DROP 5 ROLL →NUM 'T' STO
EVAL 9 RND SWAP EVAL DUP ABS IF 1E-10 ≤ THEN 4 DROPN
"VERTICAL TANGENT" ELSE / 3 ROLLD EVAL SWAP EVAL 3 PICK * -
SWAP 'X' * SWAP + END 'T' PURGE »
```



**EXAMPLE 19.** we shall apply program PAR' to the parametric curve of EXAMPLE 18.

Begin with a parametric plot of the curve using  $Xrng$ : -6.5 6.5,  $Yrng$ : -3 6, and independent variable  $T$  restricted by  $\{ T \ 0 \ 4.5 \}$ . The plot should appear as follows:

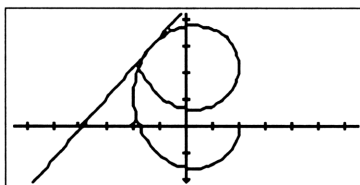


Now recall the EQ to level 1, enter 2 and run program PAR' to see the expression

$$'1.62587390881 * X + 6.39589170357$$

for the tangent line when  $T = 2$ .

To overlay the tangent line on the plot of the curve, change the plot type to FUNCTION, put the above expression into EQ and set the independent variable to  $X$ . Without erasing the original plot, execute the DRAW command to obtain the following:



**Activity Set 2.3.5**

1. (a) Plot the parametric curve given by

$$x = t - 2 \sin 3t, \quad y = 2 \cos 2t, \quad 0 \leq t \leq 2\pi$$

using *Xrng*: -3.5 9.5 and *Yrng*: -2 2.

- (b) Calculate and overdraw plots of the tangent lines to the curve at points corresponding to  $t = 0$ ,  $t = \pi/2$ ,  $t = 5\pi/6$ , and  $t = 7\pi/6$ .

2. (a) Plot the parametric curve given by

$$x = 4 \cos t, \quad y = 2 \sin t, \quad 0 \leq t \leq 2\pi$$

using the default plotting screen.

- (b) Calculate and overdraw plots of the tangent lines to the curve at points corresponding to the following values of  $t$ :

(i)  $t = 0$  and  $t = \pi$

(ii)  $t = \pi/2$  and  $t = 3\pi/2$

(iii)  $t = \pi/4$  and  $t = 5\pi/4$

3. (a) Plot the parametric curve given by

$$x = 3 \cos^3 t, \quad y = 3 \sin^3 t, \quad 0 \leq t \leq 2\pi$$

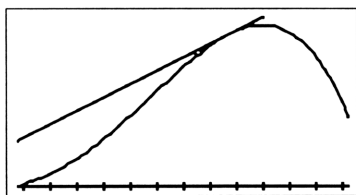
using the default plotting screen.

- (b) Calculate and overdraw plots of the tangent lines to the curve at points corresponding to  $t = 0$ ,  $\pi/2$ ,  $\pi$  and  $3\pi/2$ ; then at points corresponding to  $t = 3\pi/4$  and  $7\pi/4$ .

# 3

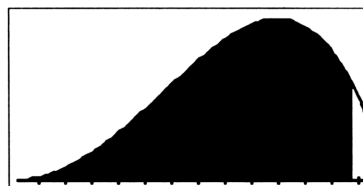
## INTEGRALS

Calculus is rich in its connections to geometry. The two main ideas of calculus — the derivative and the integral — arose from simple geometric questions: what is the slope of a curve? what is the area beneath a curve?



The derivative  $\frac{dy}{dx}$  gives the slope of  $y = f(x)$

Figure 1(a)



The integral  $\int_a^b f(x)dx$  gives the area between  $y = f(x)$  and the  $x$ -axis from  $x = a$  to  $x = b$

Figure 1(b)

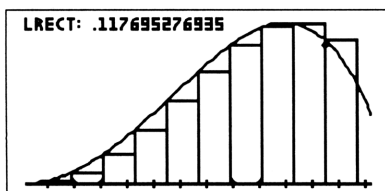
### 3.1 APPROXIMATING AREA

#### Rectangle Approximations

To approximate the area of the region lying between the curve  $y = f(x)$  and the  $x$ -axis from  $x = a$  to  $x = b$  (the shaded region in Figure 1(b)), we can sum areas of rectangles.

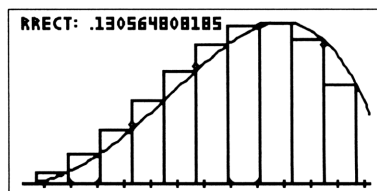
Divide the interval  $[a, b]$  into  $n$  equal subintervals of length  $h = \frac{b-a}{n}$  with points  $a = x_0 < x_1 < \dots < x_{n-1} = b$ . On each subinterval, build a rectangle whose width is  $h$  and whose height is given by a value  $f(x^*)$  of the function for some  $x^*$  chosen within the subinterval. If  $x^*$  is always chosen as the left endpoint of the

subinterval we build *left rectangles*; if  $x^*$  is always chosen as the right endpoint of the subinterval we build *right rectangles*. The sum of the areas of the rectangles is an approximation to the area of the region.



Approximation with 10 left rectangles

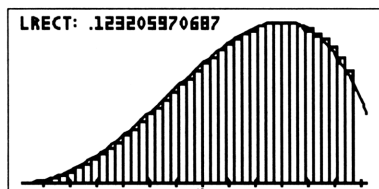
Figure 2(a)



Approximation with 10 right rectangles

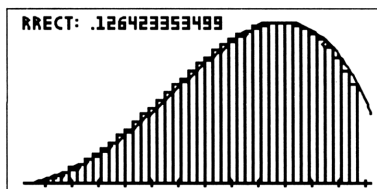
Figure 2(b)

By increasing the number of rectangles we can improve the approximations.



Approximation with 40 left rectangles

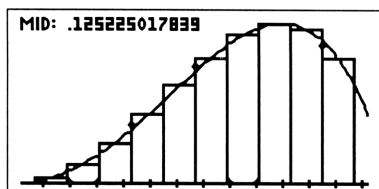
Figure 3(a)



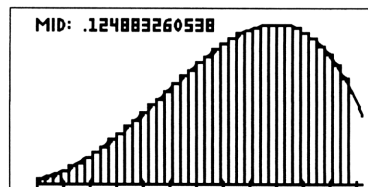
Approximation with 40 right rectangles

Figure 3(b)

When a graph is increasing, left rectangles will clearly *underestimate* the area, while right rectangles will *overestimate* the area. Exactly the opposite occurs when a graph is decreasing: left rectangles will overestimate and right rectangles underestimate. Thus, a convenient way to balance the errors is to use *midpoint rectangles*, rectangles whose heights are calculated by  $f(x^*)$  where  $x^*$  is always chosen as the midpoint of each subinterval.



Approximation with 10 midpoint rectangles  
Figure 4(a)



Approximation with 40 midpoint rectangles  
Figure 4(b)

Since it is impractical to calculate a large number of rectangle areas by hand, we can use the HP-48G/GX. Below we present a sequence of HP-48 programs to do this. The first one, **GRECT**<sup>1</sup> [= Graphing RECTangles] provides a graphical/numerical interface for rectangle sum approximations. Depending upon your choice, it calls upon programs **LRECT** [= Left RECTangles], **RRECT** [Right RECTangles], or **MID** [= MIDpoint rectangles] to do the numerical calculations. These three programs call upon program **SUM** to do the actual summing and **SUM** calls upon program **F.val** to evaluate the input function  $f$  at the appropriate values. Two other utility programs are given: **FABSTO**, used to store the expression for  $f$  and values for  $a$  and  $b$ ; and **NSTO**, used to store the number  $n$  of subintervals. All of these programs (and two others) can be found in the special **INTG** subdirectory of the main calculus directory **CALC**, on the teaching code diskette (available from the publisher). Following a listing of these programs, we will work an example.

---

<sup>1</sup>The **GRECT** program was written by Robert E. Simms of Clemson University. We are indebted to him for permission to use it here.

**FABSTO**

*Input:* Level 3: an expression for  $f(x)$ , in terms of 'X'

Level 2: the lower limit of integration,  $a$

Level 1: the upper limit of integration,  $b$

*Effect:* stores  $f$ ,  $a$  and  $b$  as EQ,  $A$  and  $B$ .

« 'B' STO 'A' STO STEQ »

**NSTO**

*Input:* level 1: a positive integer  $n$

*Effect:* stores  $n$  as the number of subintervals  $N$   
and stores  $h = (b - a)/n$  as  $H$

« 'N' STO B A - N / 'H' STO »

**GRECT**

*Input:* As stored variables: an expression for  $f(x)$  in EQ and values for  $a$  and  $b$  in  $A$  and  $B$ , respectively, from the program FABSTO; a value for  $n$  in  $N$  from program NSTO.

*Effect:* Prompts the user for a rectangle type; based upon the choice, produces an autoscaled plot of the function in EQ, overlays the approximating rectangles on the plot, and calculates the sum of their signed areas; puts the sum on stack level 1 as a tagged object. (By *signed* areas we mean that areas of rectangles lying below the  $x$ -axis carry a negative ( - ) sign while areas of rectangles lying above the  $x$ -axis carry a positive ( + ) sign.)

*Comment:* In order for the program to properly draw the approximating rectangles, the number  $n$  of subintervals (rectangles) must be a divisor of 120:  $N = 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 25, 30, 40, 60$  or 120. When  $n = 120$ , the entire region beneath the graph from  $x = a$  to  $x = b$  is shaded because each rectangle is exactly one pixel in width.

```
« N 120 MIN 1 MAX DUP B A - SWAP / → n h « CLLCD "Select rectangle type 1
for Left 2 for Mid 3 for Right" 1 DISP 7 FREEZE IFERR 0 WAIT THEN DROP ELSE →c «
CASE c 82.1 == THEN 'LRECT' A END c 83.1 == THEN 'MID' A h 2 / + END c 84.1 ==
THEN 'RRECT' A h + END KILL END » 0 0 10 FOR z A B A - 10 / z * + 'X' STO EQ
EVAL NEXT 12 DUPN 1 11 START MAX NEXT 13 ROLLD 1 11 START MIN NEXT DUP2
DUP2 - 2 60 / * 5 ROLLD - 1 60 / * - 3 ROLLD + YRNG B A - 5 120 / * DUP NEG A +
SWAP B + XRNG # 131d # 64d PDIM {# 0d # 0d} PVIEW DRAX DRAW 'X' STO IFERR A
1 n START DUP 0 R→C C→PX SWAP h + DUP 3 ROLLD EQ EVAL R→C C→PX BOX 'X'
h STO+ NEXT DROP PICT {# 0d # 0d} 3 ROLL EVAL DUP 4 ROLLD 1 →GROB REPL 7
FREEZE THEN END 'X' PURGE END » »
```

**LRECT**

*Input:* none from the stack

*Effect:* uses SUM to compute the Riemann sum for the  $f$ ,  $a$ ,  $b$  and  $n$  stored, with  $f$  evaluated at the left end point of each subinterval

« A SUM 'lrect' →TAG »

**RRECT**

*Input:* none from the stack

*Effect:* uses SUM to compute the Riemann sum for the  $f$ ,  $a$ ,  $b$  and  $n$  stored, with  $f$  evaluated at the right end point of each subinterval

« A H + SUM "rrect" →TAG »

**MID**

*Input:* none from the stack

*Effect:* uses SUM to compute the Riemann sum for the  $f$ ,  $a$ ,  $b$  and  $n$  stored, with  $f$  evaluated at the midpoint of each subinterval

« A H 2 / + SUM "mid" →TAG »



**F.val**

*Input:* none from the stack

*Effect:* a utility program used by other programs to evaluate  $f$  at a specified number

« 'X' STO EQ EVAL »

**SUM**

*Input:* none from the stack

*Effect:* a utility program used for computation by each of the Riemann sum programs and by TRAP and SIMP. It takes the initial value of  $x$  from the other program,  $a$  for LRECT,  $a + h$  for RRECT and  $a + h/2$  for MID.

« → X « 0 1 N START X F.val + X H + 'X' STO NEXT H \* » 'X' PURGE »

**EXAMPLE 1.** To approximate the area of the region beneath the graph of  $f(x) = x^2 - x^4$  over the interval  $[.05 \ .9]$ , first with  $n = 10$  rectangles and then with  $n = 40$  rectangles, arrange the stack like

3: 'X ^ 2 - X ^ 4'

2: .05

1: .9

and press **FABST** to store 'X ^ 2 - X ^ 4' as EQ, .05 as  $A$  and .9 as  $B$ . Now enter 10 and press **NSTO** to store 10 as the number  $N$  of rectangles. Press

`GRECT`; at the prompt, press 1 to choose left rectangles. You should obtain the plot shown in Figure 2(a). Exit to the stack to see the approximating sum on stack level 1. Press `GRECT` again, and this time select right rectangles. You should obtain the plot shown in Figure 2(b). Run `GRECT` again and select midpoint rectangles. You should obtain the plot of Figure 4(a). Return to the stack environment and store 40 into  $N$  with `40 NSTO`. Select 1 to obtain the plot of Figure 3(a). Running `GRECT` and selecting 3 and then 2 will produce the plots of Figure 3(b) and Figure 4(b). Clear the stack when you have finished plotting.

Although program `GRECT` will only plot approximating rectangles for values of  $n$  that divide 120, we can use the programs `LRECT`, `RRECT` and `MID` by themselves to obtain rectangle approximations to signed areas for *arbitrary* values of  $n$ . As with `GRECT`, we must first use `FABSTO` and `NSTO` to store  $EQ$ ,  $A$ ,  $B$  and  $N$ .

**EXAMPLE 2.** In EXAMPLE 1 we applied `GRECT` to  $f(x) = x^2 - x^4$  over  $[.05 \ .9]$  to graphically view the approximating rectangles and calculate the sum of their areas for  $n = 10$  and  $n = 40$ . Here we use programs `LRECT`, `RRECT` and `MID` by themselves to simply calculate the approximating sums. Using  $n = 100, 200$  and  $500$  you should verify the following results:

$n$	LRECT	RRECT	MID
100	.124209601096	.125496554221	.124864054865
200	.124536827979	.125180304542	.124861310615
500	.124731407787	.124988798412	.124860542199

Which column of approximations appears to be the most accurate? The exact area is .124860395833 to twelve decimal places.

### Activity Set 3.1.1

1. Consider the function  $f(x) = 1/x^2$  over the interval  $[1, 3]$ .

- (a) Use GRECT with  $n = 15, 30, 60$  and  $120$  rectangles to complete the first four lines of the following table.

n	LRECT	RRECT	MID
15			
30			
60			
120			
200			
500			
1000			

- (b) Now use programs LRECT, RRECT and MID by themselves to complete the last three lines of the table.
- (c) Which of the three columns in the table appears to be producing the best approximations? The exact area is  $\frac{2}{3} = .666666666666$ .
2. Consider the function  $f(x) = \sqrt{x} \sin x$  over the interval  $[0, \pi]$ .
- (a) Use GRECT with  $n = 15, 30, 60$  and  $120$  rectangles to complete the first four lines of the following table.

n	LRECT	RRECT	MID
15			
30			
60			
120			
200			
500			
1000			

- (b) Now use programs LRECT, RRECT and MID by themselves to complete the last three lines of the table.
- (c) Which of the three columns in the table appears to be producing the best approximations? The exact area is 2.43532116417 to twelve decimal places.
3. Consider the function  $f(x) = 3x - 4 \sin x$  over the interval  $[-2, 2]$ .
- (a) Use program GRECT to obtain midpoint rectangle approximations to the signed area over  $[-2, 2]$  for  $n = 5, 20$  and  $40$ . Explain your answers.
- (b) Now use GRECT to compare the left rectangle and right rectangle approximations to the signed area over  $[-2, 2]$  for  $n = 5, 20$  and  $40$ .

n	LRECT	RRECT
5		
20		
40		

- (i) Explain these results.
- (ii) Are these results what you expected? Why don't the left and right rectangle approximations more closely match the results from the midpoint rectangle approximations?
- (c) What is the exact signed area of  $f(x) = 3x - 4 \sin x$  over  $[-2, 2]$ ? Why?
4. Consider the function  $f(x) = 2 \cos 2x - \sin(x + 2)$  on the interval  $[0, 4]$ .
- (a) Use GRECT to obtain left, right and midpoint rectangle approximations to the signed area over  $[0, 4]$  for  $n = 10, 25$ , and  $40$ . Then use LRECT, RRECT and MID by themselves to obtain approximations for  $n = 100, 200$  and  $500$ .

n	LRECT	RRECT	MID
10			
25			
40			
100			
200			
500			

- (b) Which of the three columns in the table appears to be producing the best approximations? The exact signed area is 2.36567536982.

## Riemann Sums

Whenever rectangles are used to approximate a region lying between a curve  $y = f(x)$  and the  $x$ -axis over an interval  $[a, b]$ , the sum of the areas of the approximating rectangles is given by an expression of the form

$$(1) \quad \sum_{k=1}^N f(x_k^*) \Delta x_k .$$

This sum is based upon a division of the interval  $[a, b]$  into  $N$  subintervals  $[x_0, x_1]$ ,  $[x_1, x_2]$ ,  $\dots$ ,  $[x_{N-1}, x_N]$  using points  $a = x_0 < x_1 < \dots < x_N = b$ . The meaning of the terms in the sum (1) are as follows:

- $\Delta x_k$ : the width of the  $k^{th}$  rectangle
- $x_k^*$ : a point somewhere in the  $k^{th}$  subinterval
- $f(x_k^*)$ : the height of the  $k^{th}$  rectangle
- $f(x_k^*) \Delta x_k$ : the area of the  $k^{th}$  rectangle

To acquire a better understanding of such sums we can use the HP-48G/GX to create user-defined functions for them.

**EXAMPLE 3.** Given  $f(x) = x^2 - x^4$  on the interval  $[0, 1]$ , create user-defined functions  $S(N)$  and  $T(N)$  for sums like (1) that use  $N$  equally spaced right rectangles and  $N$  equally spaced midpoint rectangles.

The key ingredients for the right rectangle sum in HP-48G/GX notation are:


- width of each rectangle ( $\Delta x_k$ ):  $\frac{1}{N}$
- right endpoint of the  $k^{th}$  rectangle ( $x_k^*$ ):  $\frac{K}{N}$
- height of the  $k^{th}$  rectangle ( $f(x_k^*)$ ):  $\left(\frac{K}{N}\right)^2 - \left(\frac{K}{N}\right)^4$
- area of the  $k^{th}$  rectangle:  $\left[\left(\frac{K}{N}\right)^2 - \left(\frac{K}{N}\right)^4\right]\left(\frac{1}{N}\right)$

The user-defined function is




$$S(N) = \sum_{K=1}^N \left( \left( \frac{K}{N} \right)^2 - \left( \frac{K}{N} \right)^4 \right) \left( \frac{1}{N} \right).$$

If you use the Equation Writer to create this expression, the right-hand side will appear like


$$\sum_{K=1}^N \left( \left( \frac{K}{N} \right)^2 - \left( \frac{K}{N} \right)^4 \right) \cdot \left( \frac{1}{N} \right)$$

Remember to use the  key to end each subexpression. If you build the expression on the stack, it will appear as:

$$1: 'S(N) = \sum (K = 1, N, ((K/N) ^ 2 - (K/N) ^ 4) * (1/N))'$$

If you build the expression with the Equation Writer, it will be put on the stack when you press . Use   to complete the construction. Now evaluate  $S$  for values of  $N = 10, 20, 50$ , and  $100$  to obtain the following table:

$N$	$S(N)$
10	.13167
20	.132916875
50	.13326672
100	.13316667

*Suggestion:* The *easiest* way to build a complicated user-defined function like this one is to use RPN on the stack. Put 'S(N)' on level 1, then enter 'K', 1, and 'N'. Then put 'K/N' on level 1 and press  to make a duplicate copy. Now build

' $((K/N) \wedge 2 - (K/N) \wedge 4) * (1/N)$ ' using RPN. Press  $\boxed{\rightarrow}$   $\boxed{\Sigma}$  to complete the sum. Do  $\boxed{\leftarrow}$   $\boxed{=}$  to equate 'S(N)' with the sum, then use  $\boxed{\leftarrow}$   $\boxed{\text{DEF}}$  to complete the task.

For the midpoint rectangles, the only difference is that we have

- midpoint of the  $k^{\text{th}}$  rectangle ( $x_k^*$ ):  $\frac{2K-1}{2N}$ .

Thus, in this case the user-defined function will be

$$T(N) = \Sigma(K = 1, N, (((2 * K - 1)/(2 * N)) \wedge 2 - ((2 * K - 1)/(2 * N)) \wedge 4) * (1/N))'$$

Evaluating  $T$  for  $N = 10, 20, 50$  and  $100$  we have

$N$	$T(N)$
10	.13416375
20	.133541484376
50	.133366662
100	.133341666383

For an arbitrary choice of points  $x_1 < x_2 < \dots < x_{N-1}$  between  $a$  and  $b$  and an arbitrary choice of a point  $x_k^*$  in the  $k^{\text{th}}$  subinterval, the sum (1) is called a *Riemann sum*. Sums constructed from subintervals of equal width, or with a uniform choice of points  $x_k^*$  in each subinterval, or both, are special kinds of Riemann sums. But arbitrary Riemann sums work just as well, with no restrictions whatsoever on the widths of the subintervals or the location of the  $x_k^*$ 's. The general result is that for a "well-behaved" function on the interval  $[a, b]$ , e.g., a function  $f$  that is continuous on  $[a, b]$ , Riemann sums have a limit  $I$ :

$$\lim_{N \rightarrow \infty} \sum_{k=1}^N f(x_k^*) \Delta x_k = I.$$

The number  $I$  is called the *definite integral of  $f$  from  $a$  to  $b$* , and is denoted by



$$I = \int_a^b f(x)dx.$$

The integral  $\int_a^b f(x)dx$  is the *signed area* of the region between the curve  $y = f(x)$  and the  $x$ -axis over the interval  $[a, b]$ .

For a given value of  $n$ , the midpoint rectangle approximation  $M_n$  to an integral  $I = \int_a^b f(x)dx$  will be more accurate than the left or right rectangle approximations,  $L_n$  or  $R_n$ . Upper bounds for the errors are well-known and are related to the first and second derivatives of  $f$ . If  $|f'(x)| \leq B_1$  and  $|f''(x)| \leq B_2$  for all  $x$  in  $[a, b]$  then

$$|L_n - I| \leq \frac{B_1(b-a)^2}{2n} \text{ and } |R_n - I| \leq \frac{B_1(b-a)^2}{2n} \text{ but } |M_n - I| \leq \frac{B_2(b-a)^3}{24n^2}.$$

### Activity Set 3.1.2

1. Consider the region between the curve  $y = x^3$  and the  $x$ -axis on the interval  $[0, 4]$ . We want to create a user-defined function for a Riemann sum like (1) that uses  $N$  equally spaced right rectangles.

(a) Express the following in terms of  $N$ :

- the width of each rectangle ( $\Delta x_k$ ):
- the right endpoint of the  $k^{th}$  rectangle ( $x_k^*$ ):
- the height of the  $k^{th}$  rectangle ( $f(x_k^*)$ ):
- the area of the  $k^{th}$  rectangle  $f(x_k^*) \Delta x_k$ :

- (b) Create a user defined function  $S(N) = \sum_{K=1}^N$  (area of the  $k^{th}$  rectangle) on your HP-48G/GX and use it to complete the table below:

$N$	$S(N)$
10	
50	
100	
200	

- Repeat Activity 1 using  $y = \frac{1}{x}$  on  $[1, 3]$ .
- Repeat Activity 1 using  $y = e^x$  on  $[-1, 3]$ .
- Repeat Activity 1 for  $y = \frac{3}{x^2} + x$  on  $[-3, -1]$  using  $N$  equally spaced *left rectangles*.

## Trapezoid and Simpson's Approximations

Recall that for an increasing (decreasing) function, the left rectangle approximation underestimates (overestimates) the area and the right rectangle approximation does exactly the opposite. Thus, midpoint rectangles were introduced as a way of "balancing" the two errors. But instead of using midpoint rectangles, we can simply average the left and right rectangle results.

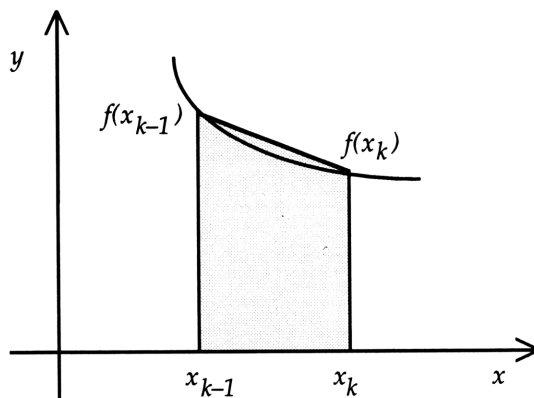
For an evenly spaced division of the interval  $[a, b]$  into  $N$  subintervals of length  $\Delta x$  we have

$$\text{LRECT} = \sum_{k=1}^N f(x_{k-1})\Delta x \quad \text{and} \quad \text{RRECT} = \sum_{k=1}^N f(x_k)\Delta x .$$

Their average is

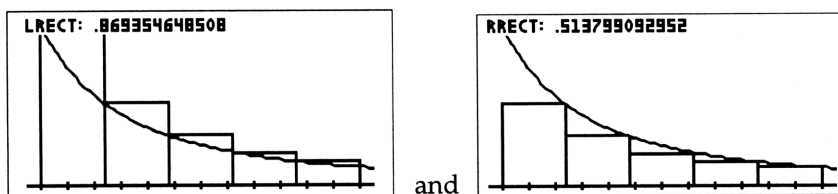
$$\text{Average} = \sum_{k=1}^N \frac{[f(x_{k-1}) + f(x_k)]}{2} \Delta x.$$

This summand is the area of a trapezoid sitting on the  $k^{\text{th}}$  subinterval  $[x_{k-1}, x_k]$ :

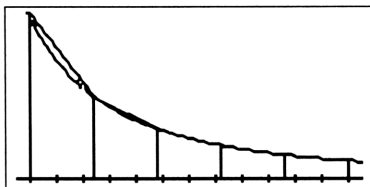


For this reason, the average is called the *Trapezoid approximation*. It is easy to see that it can be a much better approximation than the left and right rectangle approximations. Consider, for example, the case  $y = \frac{1}{x^2}$  over the interval  $[1, 3]$ .

With  $n = 5$  we have:



But the Trapezoidal approximation looks like this:



TRAP: .69157687073

The following program TRAP appears in the INTG subdirectory of the main CALC directory. Like LRECT, RRECT, and MID, it requires that you first use FABSTO and NSTO.

### TRAP

*Input:* none from the stack

*Effect:* uses SUM to compute the trapezoidal approximation  
for the stored quantities  $f$ ,  $a$ ,  $b$ , and  $n$

« A SUM B F.val A F.val - 2 / H \* + 'X' PURGE "trap" →TAG »

**EXAMPLE 4.** We return to EXAMPLE 2 where we used LRECT, RRECT and MID to calculate approximating sums for the integral  $\int_{.05}^9 (x^2 - x^4)dx$  using  $N = 100, 200$  and 500.

Applying TRAP we obtain

$N$	TRAP
100	.124853077658
200	.12485856626
500	.1248601031

Because the Trapezoid approximation simply sums the areas of trapezoids, it is possible to give a formula for the approximation. Given that the interval  $[a, b]$  is divided into  $n$  subintervals of equal length  $\Delta x = \frac{b-a}{n}$  by points  $a = x_0 < x_1 < \dots < x_{n-1} < x_n = b$ , let  $y_j = f(x_j)$  for  $j = 0, 1, \dots, n$ . Then the Trapezoid approximation  $T_n$  to  $\int_a^b f(x)dx$  is given by

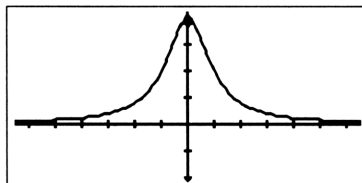
$$T_n = \frac{\Delta x}{2} [y_0 + 2y_2 + \dots + 2y_{n-1} + y_n].$$

This formula is frequently used in hand calculations for small values of  $n$ .

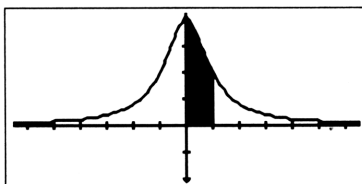
It is well-known that

$$\int_0^1 \frac{4}{1+x^2} dx = \pi.$$

You can verify this with your HP-48G/GX as follows: Draw a plot of  $y = \frac{4}{1+x^2}$  using  $Xrng$ : -6.5 6.5 and  $Yrng$ : -2.1 4.2.



Move the cursor to  $(0, 0)$  and press  $\boxed{\times}$  to mark its location, then reposition the cursor at  $(1, 0)$  and press  $\boxed{\text{AREA}}$  on the FCN menu. The message **AREA:** 3.14159265359 will appear on the lower left of your screen (and on level 1 of the stack). With the origin marked and the cursor still at  $(1, 0)$ , press  $\boxed{\text{NXT}}$  to return to the FCN menu and press  $\boxed{\text{SHADE}}$  to shade the region whose area is  $\pi$ .



This use of the **AREA** key on the HP-48G/GX requires that the lower and upper limits of integration be given by pixel coordinates.

We shall use this example to compare the errors made by the trapezoid and midpoint approximations. Open the INTG subdirectory and use FABSTO to store ' $4/(1 + X^2)$ ' into EQ and 0, 1 into A, B respectively. Now enter the following program and store it under the name 'ERROR':

«  $\pi$  →NUM – "error" →TAG ».

For a calculated approximation  $A$  on level 1, program ERROR will calculate  $(A - \pi)$  and display it on level 1 with the tag "error".

To compare the Trapezoid and midpoint approximations for  $N = 50, 100, 150$  and 200, proceed as follows:

- (i) Use NSTO to store 50 for  $N$ . Press **TRAP** **ENTER**, then **ERROR** to see

2: trap: 3.1415259869  
1: error: -.00006666669

Now press **MID** **ENTER**, then **ERROR** to see

2: mid: 3.14162598694  
1: error: .00003333335

Notice that the magnitude of the error from MID is one-half of that from TRAP.

(ii) - (iv). Repeat step (i) using  $N = 100, 150$  and  $200$  in succession.

The following table summarizes the results:

$N$	TRAP	ERROR	MID	ERROR
50	3.1415259869	-.00006666669	3.14162598694	.00003333335
100	3.14157598691	-.00001666668	3.1416009869	.00000833331
150	3.1415852461	-.00000740749	3.14159635725	.00000370366
200	3.1415884869	-.00000416669	3.14159473692	.00000208333

The TRAP and MID columns tell us that the trapezoid estimates are too low, while the midpoint estimates are too high. And the two ERROR columns show that the midpoint error is consistently one-half the trapezoid error in magnitude.

This is not surprising if we examine the upper bounds on the errors. We noted earlier that for a given value of  $n$ , a bound on the error by the midpoint rectangle approximation  $M_n$  to an integral  $I = \int_a^b f(x)dx$  is given by

$$|M_n - I| \leq \frac{B_2(b-a)^3}{24n^2}, \text{ where } |f''(x)| \leq B_2 \text{ for all } x \text{ in } [a, b].$$

For the Trapezoid approximation  $T_n$  with  $n$  subintervals, a bound on the error is

$$|T_n - I| \leq \frac{B_2(b-a)^3}{12n^2}.$$

To get an improved estimate of the integral that balances the errors, we can use a "weighted" average of the trapezoid and midpoint estimates:

$$\text{weighted average} = \frac{1}{3} (\text{trapezoid}) + \frac{2}{3} (\text{midpoint}).$$

Averaging will tend to balance the low versus high estimates, and we weight the midpoint estimate twice as much because its error is only half that of the trapezoid estimate.

This particular weighted average is known as *Simpson's approximation*. It produces approximations to the integral  $\int_a^b f(x)dx$  that are far more accurate than those by the other methods that we have considered. A bound on the error involves the fourth derivative of  $f$  on  $[a, b]$ . If  $|f^{(iv)}(x)| \leq B_4$  for all  $x$  in  $[a, b]$ , then Simpson's approximation  $S_n$  using  $n$  subintervals satisfies

$$|S_n - I| \leq \frac{B_4(b-a)^5}{180n^4}.$$

Because of this, Simpson's approximation produces *exact* results for any integral  $\int_a^b f(x)dx$  where  $f^{(iv)}(x) = 0$ . In particular, it gives exact results for all linear, quadratic and cubic polynomial functions.

The following HP-48G/GX program **SIMP** resides in the INTG subdirectory of the main CALC directory.

#### **SIMP**

*Input:* none from the stack

*Effect:* uses MID and TRAP to compute Simpson's approximation for the stored quantities  $f$ ,  $a$ ,  $b$  and  $n$

« MID 2 \* TRAP + 3 / "simp" →TAG »



To appreciate the accuracy of Simpson's approximation, we apply program SIMP to the integral  $\int_0^1 \frac{4}{1+x^2} dx = \pi$ :

$N$	SIMP	ERROR
5	3.14159261393	-.00000003966
10	3.14159265297	-.00000000062
15	3.14159265354	-.00000000005
20	3.14159265359	0

Like the Trapezoid approximation, there is an easy formula for Simpson's approximation. The formula is based upon dividing the interval  $[a, b]$  into an *even number*  $n$  of subintervals of equal width  $\Delta x = \frac{b-a}{n}$ :

$$S_n = \frac{\Delta x}{3} (y_0 + 4y_1 + 2y_2 + \dots + 2y_{n-2} + 4y_{n-1} + y_n).$$

Observe, carefully, the pattern 1, 4, 2, 4, 2, ... in the coefficients; and how it ends: 2, 4, 1. This pattern *requires* that  $n$  be an even number. This formula for Simpson's approximation to the integral  $\int_a^b f(x)dx$  is useful when  $f$  is given only in graphical or tabular forms.

**Activity Set 3.1.3**

In each of the following Activities, use the AREA command on the FCN submenu to obtain an accurate twelve (12) digit approximation to the integral. Then calculate Trapezoid and Simpson's approximations to the integral using the indicated number of subintervals. Keep your numeric display mode set to STD to show full precision.

$$1. \int_1^3 \frac{1}{x^2} dx, n = 50, 100$$

$$2. \int_0^{\pi} \sqrt{x} \sin x \, dx, n = 100$$

$$3. \int_0^4 (4 \sin x - x) dx, n = 100$$

$$4. \int_0^4 (2 \cos 2x - \sin(x + 2)) dx, n = 100$$

$$5. \int_0^3 \frac{1}{1+x^3} dx, n = 100$$

$$6. \int_{-3}^{-1} \left( x + \frac{3}{x^2} \right) dx, n = 50, 100$$

$$7. \int_0^3 e^{-x^2} dx, n = 100$$

$$8. \int_0^1 \sqrt{1 + \sec^2 x} \, dx, n = 100$$

$$9. \int_0^2 \sqrt{1 + \sin^2 x} \, dx, n = 100$$

$$10. \int_1^2 \sqrt[3]{\left( 1 + \frac{1}{x^4} \right)} dx, n = 100, 200$$

11. Consider

$$f(x) = \begin{cases} \cos(\pi x^2/2) & x < 1 \\ x^2 - 3x + 2 & x \geq 1 \end{cases}$$

(a) Find the integral  $\int_0^3 f(x)dx$  using the AREA key on the FCN submenu.

(b) Now approximate the integral  $\int_0^3 f(x)dx$  using Simpson's approximation with  $n = 100$  and  $n = 200$  subintervals.

### 3.2 INTEGRATION ON THE HP-48G/GX

#### Numerical Integration

In the application of calculus to fields such as engineering, physics, probability and statistics there is often a need to obtain fairly accurate estimates of definite integrals. The integrands in question may be simple in appearance, but usually lack elementary, closed-form antiderivatives so that the fundamental theorem of calculus cannot be applied. Simple examples are:

- the standard normal integral  $\int_0^z \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$  from probability theory
- the period  $T = \int_0^\alpha \frac{2\sqrt{2} dy}{\sqrt{\cos y - \cos \alpha}}$  of a simple pendulum
- the electrostatic potential  $V$  at a point  $P(x,y)$  due to a variable charge density  $\lambda(s)$  applied along a straight wire over an interval  $[-a, a]$ :

$$V = \int_{-a}^a \frac{\lambda(s)ds}{\sqrt{(x-s)^2 + y^2}}$$







The HP-48G/GX has a built-in numerical integration routine that uses a Romberg numerical integration technique. The routine is iterative, producing increasingly accurate estimates derived from values of the integrand at points sampled within the interval of integration until three successive estimates agree to within an error tolerance specified by the user. The error tolerance  $e$  is specified by setting the numeric display mode as follows:  $n$  FIX specifies an error tolerance  $e = 10^{-n}$  and STD specifies an error tolerance  $e = 10^{-11}$ .

For example, setting the numeric display to 5 FIX will specify an error tolerance  $e = .00001$ . In general, the smaller the error tolerance, the longer the calculation time and the more accurate the result. When the calculation is finished, an estimate of the error in the result is given in the variable IERR.

There are two ways to perform a numerical integration on the 48G/GX: with the INTEGRATE Form on the SYMBOLIC Application or on the Stack. We illustrate each way with the integral  $\int_0^{\pi} 3x \sin 2x \, dx$ . The exact answer and its decimal approximation are

$$\int_0^{\pi} 3x \sin 2x \, dx = -3\pi/2 \approx -4.71238898038.$$

## Using the INTEGRATE Form

Open the Symbolic Application with  **SYMBOLIC** and press  to select Integrate. Type in '3 \* X \* SIN(2 \* X)' and use  to enter it into the EXPR: field. Then enter 'X' into the VAR: field and 0,  $\pi$  into the LO: and HI: fields. When the RESULT: field is highlighted press  if it reads Numeric, otherwise press  to change to Numeric. Set the NUMBER FORMAT: field to Std, using the CHOOS box if necessary. Press  to perform the numerical integration.

The result will be shown on stack level 1.

1: -4.71238898038

Press **VAR** then **IERR** to see the error estimate 9.4E-11.

## Using the Stack

Arrange the stack as follows:

4:	0
3:	$\pi$
2:	'3 * X * SIN(2 * X)'
1:	'X'

Press **→** **∫** to see the symbolic expression

'∫ (0,  $\pi$ , 3 \* X \* SIN(2 \* X), X)'

returned to level 1. Use **↵** **→NUM** to see the numerical result -4.71238898038

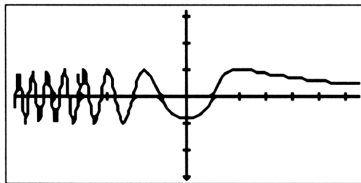
returned to level 1. Press **VAR** then **IERR** to see the error estimate 9.4E-11.

If you wish, you can use the Equation Writer to key in the integral:

$$\int_0^{\pi} 3 \cdot X \cdot \sin(2 \cdot X) \, dX$$

Use the **▶** to end each subexpression. When the expression is complete, press **ENTER** to view it on the stack and then **↵** **→NUM** to evaluate. Alternatively, with the expression showing in the Equation Writer press **↵** **→NUM** to bypass the stack and obtain the numerical result.

**EXAMPLE 5.** As another example, we graph the function  $f(x) = \begin{cases} \sin(x^2 - 1) & x < 1 \\ \sin(\pi/x) & 1 \leq x \end{cases}$  with the default parameters and calculate the integral  $\int_{-2}^2 f(x) dx$ . Enter the function as 'IFTE(X < 1, SIN(X ^ 2 - 1), SIN( $\pi$ /X))'. Plotting with the default parameters, we see:



To calculate the integral, we first set the numeric display mode to 5 FIX, then press  $\leftarrow$  to again view the plot. Activate coordinate read-out with  $(X, Y)$ , move the cursor to (-2, 0) and press  $\times$  to mark the location. Now move the cursor to (2, 0), press **NXT** to return the menu labels, open the FCN submenu and press **AREA**. In approximately 1 minute, 12 seconds you will see the result .20163 displayed at the bottom left of the screen and on stack level 1. IERR shows the error to be approximately .00003.

To use the numerical integration routine in this way (while viewing a plot of the integrand), the limits of integration must be pixel coordinates. It is also interesting to note that if you perform the same integration on the stack, its execution is a little faster, approximately 1 minute, 8 seconds.

Why did we set the numeric display to 5 FIX instead of asking for full twelve digit precision in STD mode? We actually tried for twelve digit precision, but gave up and interrupted the integration process at the end of one hour. When we seek twelve digit precision, many more points on the integrand are sampled than with five digit precision, and the "break point"  $x = 1$  in the definition of the integrand

causes problems. How can we obtain twelve digit precision? The trick is to split the integral into two parts at the break point:

$$\int_{-2}^2 f(x)dx = \int_{-2}^1 f(x)dx + \int_1^2 f(x)dx.$$

The first integral on the right hand side is found in 36 seconds:

$$\int_{-2}^1 f(x)dx = -.546976060733.$$

And the second integral on the right hand side takes only 14 seconds:

$$\int_1^2 f(x)dx = .748600792238.$$

Thus, we can add to obtain

$$\int_{-2}^2 f(x)dx = .201624731505$$

in a little over 50 seconds.

This trick of splitting the integral into several other integrals is standard practice with almost all numerical integration routines on calculators or computers. Obvious separation points are any break points in the definition of the integral (as in the above example), as well as any points where the function is not defined or is non-differentiable.

**Activity Set 3.2.1**

1. A roller coaster has part of its track in the shape of the curve  $y = x + \sin 2x^2$  when plotted using  $Xrng: 0\ 2$  and  $Yrng: 0\ 3$ .
  - (a) Plot the curve in this viewing window.
  - (b) Calculate the area of the region between the track and the  $x$ -axis (the ground) over the interval  $[0, 2]$ .
  - (c) Calculate the area of the region between the track and the ground over the interval between the two local maxima.
2. Find the volume of the solid of revolution generated by revolving the curve  $y = e^{\sin x}$  around the  $x$ -axis over the interval  $[0, 3]$ .
3. Find the volume of the solid generated by revolving about the  $y$ -axis the region between the graph of  $y = e^{-x^2}$  and the  $x$ -axis over the interval  $[1/3, 1]$ .
4. Calculate the "arch length" of the St. Louis Arch using the formula from Activity 7 in ACTIVITY SET 3.2.2.
5. Imagine a point  $P$  moving along a parametric curve  $C: x = f(t), y = g(t)$  in such a way that it traces the curve only once from  $t = a$  to  $t = b$ . Then the length of the curve from  $t = a$  to  $t = b$  is given by the formula

$$\int_a^b \sqrt{[f'(t)]^2 + [g'(t)]^2} dt .$$

Find the lengths of the following curves.

- (a)  $x = 3 \cos^3 t, y = 3 \sin^3 t$  from  $t = 0$  to  $t = 2\pi$ .
- (b)  $x = 3 \cos t + 2 \cos 3t, y = 3 \sin t - 2 \sin 3t$  from  $t = 0$  to  $t = 2\pi$ .



6. The following formula gives the period  $T$  of a simple pendulum of length  $L$  that is released from rest at an angle  $\alpha$  with the vertical axis ( $g$  is the constant acceleration due to gravity):

$$T = \frac{2\sqrt{2}}{\sqrt{g/L}} \int_0^{\alpha} \frac{1}{\sqrt{\cos y - \cos \alpha}} dy.$$

Find the approximate period for a pendulum of length  $L = 1.5m$  that is released at an angle of  $\pi/4$  rad from the vertical axis. (Use 3 FIX.)

## Symbolic Integration

Symbolic integration refers to calculating an integral  $\int_a^b f(x)dx$  by finding an antiderivative  $F(x)$  of the integrand  $f(x)$  and then returning a symbolic expression for  $F(b) - F(a)$ . Because of its restricted memory, the HP-48G/GX can perform symbolic integration for only the following restricted set of integrands:

- All built-in functions that have an antiderivative expressible in terms of built-in functions (except LNP1);
- sums, differences, negatives, linear combinations and other selected patterns of the above functions;
- all derivatives of built-in functions;
- polynomials whose base term is linear.

The HP-48G/GX will not, for example, perform symbolic integration on such simple integrals as

$$\int_a^b x \sin x \, dx, \quad \int_a^b x e^x dx, \quad \text{or} \quad \int_a^b \sin x \cos x \, dx.$$

These integrands are not included in the above list. On the other hand, the HP-48 will perform symbolic integration on the integral

$$\int_a^b \frac{1}{\sin x \cos x} dx$$

because the integrand is one of the selected patterns that is built-in. Because of all this, you should not view the HP-48G/GX as a serious symbolic integrator. Nevertheless, we will briefly outline some of its symbolic integration features so that you will be familiar with them.

Whether or not the  $\int$  function performs numerical or symbolic integration depends upon whether numerical or symbolic execution mode is active. The default state of the HP-48G/GX is for symbolic execution (flag -3 clear). In this state, the  $\int$  function uses a built-in system of pattern matching and returns a symbolic result (which may be nothing more than the original symbolic input). If you specify numerical results mode by setting flag -3, then the  $\int$  function will return a numerical result. No matter what the setting of flag -3, you can temporarily achieve numerical results by applying the  $\rightarrow\text{NUM}$  command to evaluate an integral.

#### EXAMPLE 6.

- (i) Make certain that your HP-48G/GX is in its default state for symbolic results. With ' $\int (0, \pi, \text{SIN}(X), X)$ ' on level 1, EVAL returns the following symbolic result.

$$\begin{aligned} 1: & \text{'-COS(X) / } \partial \text{ X(X) | (X = } \pi \\ & \text{) - (-COS(X) / } \partial \text{ X(X) | (X} \\ & \text{= 0 ))}' \end{aligned}$$

The vertical stroke  $|$  is the "where" command, used to substitute values in an expression. You can recognize this result as the HP-48 version of the familiar symbolic expression

$$-\cos x \left| \begin{array}{l} x = \pi \\ x = 0 \end{array} \right. .$$

Press **EVAL** again to effectively substitute the values  $\pi$  and 0 into  $-\cos(X)$  and obtain the numerical result 2.

- (ii) Again, with '(0,  $\pi$ , SIN(X), X)' on level 1 and symbolic results active, press **↶** **→NUM** to temporarily set numerical results mode and obtain the numerical result 2.
- (iii) You can achieve the same symbolic results as in (i) by using the INTEGRATE form in the SYMBOLIC Application. Do **→** **SYMBOLIC** **OK** and then enter 'SIN(X)' into the EQ field, 'X' into the VAR field, and 0 and  $\pi$  into the LO and HI fields. With Symbolic highlighted, press **OK** to see the same symbolic results as in (i), then use **EVAL** to effect the substitution and obtain the numerical result 2.

Occasionally, you may want to use your HP-48 to obtain an antiderivative for a function  $f(x)$ . Recall that Part 1 of the Fundamental Theorem of Calculus tells us that every continuous function  $f(x)$  on an interval  $[a, b]$  has an antiderivative  $F(x)$ , namely

$$F(x) = \int_a^x f(t)dt.$$

Therefore, if the HP-48G/GX can symbolically integrate  $f(t)$ , we can obtain a symbolic expression for the antiderivative  $F(x)$ .

**EXAMPLE 7.** To obtain an antiderivative for  $f(x) = \ln x$ , perform the symbolic integration

$$\int_a^x \ln t \, dt.$$

Use 'T' for the variable of integration and make certain that the upper limit 'X' is a *formal variable*, i.e., no value for 'X' is stored in the current directory or any of its ancestral directories. Use lowercase 'a' for the lower limit of integration. When the first symbolic result appears, press PRG TYPE OBJ→, then DROP DROP DROP. The remaining symbolic result will be

$$'(T * \text{LN}(T) - T) / \partial \, T(T) \mid (T = X)'$$

A final EVAL will return the desired antiderivative ' $X * \text{LN}(X) - X$ '.

### Activity Set 3.2.2

In activities 1-10, use your HP-48G/GX to find the indicated antiderivatives.

1.  $\int \frac{7}{\sqrt{5x-1}} \, dx$

6.  $\int 3(5x - \pi)^5 \, dx$

2.  $\int (2x + 3)^{-3/2} \, dx$

7.  $\int \frac{dx}{(\tan .5x)(\sin .5x)}$

3.  $\int \tan^2 x \, dx$

8.  $\int (x - \tanh x) \, dx$

4.  $\int \frac{\tan x}{\cos x} \, dx$

9.  $\int 2^{x+1} \, dx$

5.  $\int \tan^{-1}(2x + 3) \, dx$

10.  $\int x^{s-1} \, dx$

## 3.3 THE FUNDAMENTAL THEOREM OF CALCULUS

Chief among the significant contributions of Newton and Leibniz to the "invention" of calculus in the 17th century was their clarification of the inverse relationship between differentiation and integration. This relationship, which is the intended focus of the Fundamental Theorem of Calculus, is often obscured by a

failure to focus on Part 1 of that Theorem, which asserts that continuous functions have antiderivatives:

$$(*) \quad \frac{d}{dx} \left[ \int_a^x f(t) dt \right] = f(x).$$

Traditionally, our calculus textbooks have focused instead on Part 2 of the Theorem, which says that integration "undoes" differentiation, up to a constant:

$$\int_a^x F'(t) dt = F(x) - F(a).$$

Indeed, it is because of their focus on Part 2 that students often come to view integration as simply a search for antiderivatives rather than as the limit of Riemann sums. In retrospect, this has been a somewhat natural occurrence because, in the teaching process, teachers tend to seek out activities that students can *do* to help reinforce their understanding of the theory. And without computing power, the activities that reinforce Part 1 are restricted, for the most part, to purely analytical investigations.

But certainly, the HP-48G/GX provides enough computing power for students to engage in graphical and numerical activities that support Part 1 of the Fundamental Theorem. The midpoint approximation can be used to construct a symbolic expression  $F(x)$  that approximates the antiderivative  $\int_a^x f(t) dt$ , i.e.,  $F(x) \approx \int_a^x f(t) dt$ . This approximation and its derivative  $F'$  can then be plotted and we can observe to what extent  $F'$  approximates  $f$ . Not only does such an activity bring to the fore the mathematical content of Part 1 of the Theorem, but it also reinforces our desired goal of understanding the integral as a limit of approximating sums.

The algebraic formulation of the midpoint approximation using  $n$  subintervals of equal length  $\Delta x = \frac{x-a}{n}$  is

$$\int_a^x f(t)dt \approx \sum_{i=1}^n f\left(a + (2i-1) \frac{\Delta x}{2}\right) \Delta x.$$

When  $f$  is stored in memory as a user-defined function  $F$ , program FTC, given below, takes  $a$  and  $n$  as inputs and returns an algebraic expression for the midpoint approximation on level 2 and its derivative on level 1.

The program is due to William C. Wickes of Hewlett Packard and is about fifteen times as fast as the original program that we devised for the task. It is a marvel of extremely clever programming, and is written at a level that will not be obvious to a casual HP-48 programmer. We are indebted to Dr. Wickes for his permission to use it here.

**FTC** (Fundamental Theorem of Calculus)

*Input:* level 2: the lower limit of integration,  $a$

level 1: the number of rectangles,  $n$

As a user-defined function  $F$ : an algebraic expression for  $f(x)$ .

*Effect:* Returns to level 2 an expression that is algebraically equivalent to the midpoint approximation

$$\sum_{i=1}^n f\left(a + (2i-1) \frac{\Delta x}{2}\right) \Delta x.$$

and to level 1 the derivative of that expression.

```
« 'X' PURGE 'X' 3 PICK - → a n z « a '←i' n / z * + F a '↑↓' n /
z * + F 'X' ∂ { ↑↓ ←i } ↓MATCH DROP → f g « 0 0 .5 n .5 - FOR ←i f
EVAL ROT + SWAP g EVAL + NEXT OVER z * n / 3 ROLLD z * + n / »
» »
```

Consider the elementary function  $f(x) = \sin x$ . Since  $f$  is continuous everywhere, Part 1 of the Fundamental Theorem tells us that the function

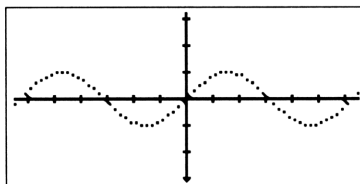
$$G(x) = \int_a^x \sin t \, dt$$

is an antiderivative of  $\sin x$  on any interval  $(a, b)$ . We take  $a = 0$  for convenience. Then

$$G(x) = \int_0^x \sin t \, dt = -\cos t \Big|_0^x = 1 - \cos x$$

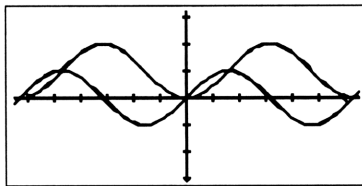
and we can readily verify that  $G'(x) = \sin x$ .

To apply program FTC, begin by constructing a user-defined function  $F$  for  $\sin x$  and then plotting  $y = \sin x$  using the default settings for  $Xrng$  and  $Yrng$ . To better see what is happening, plot in disconnected mode with the resolution set to .2 (from the PLOT menu, use .2 RES; from the PLOT screen, select OPTS and enter .2 into the STEP field).



Now run program FTC with inputs 0 (for  $a = 0$ ) and 6 (for  $n = 6$  rectangles). The program will return to level 2 an expression in 'X' that represents the midpoint approximation to  $G(x)$  using  $n = 6$  rectangles, and its derivative on level 1. Change the plotting resolution back to its default state of 0 and then overdraw the derivative on the original plot. Note the close agreement. Now change the resolution back to .2 and overdraw with a plot of the approximate antiderivative. Finally, to see how closely the approximate antiderivative agrees with the known

exact antiderivative  $1 - \cos x$ , change the resolution back to 0 and overdraw a plot of  $1 - \cos x$ :



## Differential Equations and the Fundamental Theorem

Equations that contain derivatives of one or more unknown functions are called *differential equations*. The simplest differential equations have the form  $y' = f(x)$  and their general solution is given by  $y = \int f(x)dx + C$ . Since  $C$  is a constant, there are infinitely many solutions. But we can always obtain a particular solution by specifying an *initial condition* that  $y$  is required to meet:  $y(x_0) = y_0$ . Together, the differential equation with an initial condition

$$y' = f(x), \text{ where } y(x_0) = y_0$$

is called an *initial value problem*.


Part 1 of the Fundamental Theorem of Calculus is really an initial value problem. For if we adopt the notation  $y(x) = \int_a^x f(t)dt$ , then equation ( \* ) of Part 1 of the Theorem reads

$$\frac{dy}{dx} = f(x), \text{ or simply } y' = f(x).$$

Since  $y(x)$  represents the (signed) area between the graph of  $f$  and the horizontal axis over the interval  $[a, x]$ , we have the initial condition  $y(a) = 0$ . Thus, equation ( \* ) of the Fundamental Theorem is really the initial value problem

$$y' = f(x), \quad y(a) = 0.$$



The HP-48G/GX will not only find numerical solutions to initial value problems but will also plot their solutions. To plot a solution to the initial value problem  $y'(t) = f(t, y)$ ,  $y(t_0) = y_0$  we go to the  PLOT screen and choose Diff Eq for the plot TYPE. Although the screen will show

$$\text{PLOT } Y'(T) = F(T, Y)$$

at the top, the default independent variable is 'X', which is fine for our application to the Fundamental Theorem. The application we are referring to, of course, is to simply plot the antiderivative given by Part 1, using its reformulation as an initial value problem.

The special Diff Eq plot screen is designed to let you plot a solution to the general initial value problem  $y'(t) = f(t, y)$ , subject to the initial condition  $y(t_0) = y_0$ , over the  $t$ -interval  $[t_0, t_f]$ . For our purposes, we will use the default variable 'X' instead of 'T', and take the initial value of  $Y$  to be 0.

**EXAMPLE.** To illustrate the use of the Diff Eq plotting routine, we will plot an antiderivative  $\int_a^x f(t)dt$  for  $f(x) = \ln x$ . For convenience, we take  $a = 1$ . Thus we want

to plot the solution to the initial-value problem

$$y'(x) = \ln x, \quad y(1) = 0.$$

Go to the Diff Eq plot screen and set the screen like this:

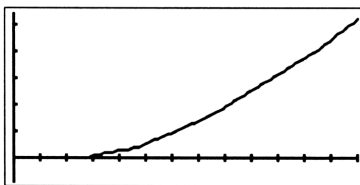
TYPE:	Diff Eq	✕ : Rad	
F:	'LN(X)'		
INDEP:	X	INIT: 1	FINAL: 6
SOL:	Y	INIT: 0	_ STIFF

Open **OPTS** and set the PLOT OPTIONS like this:

TOL:	.000001	STEP:	Dflt	<input checked="" type="checkbox"/> AXES
H-VAR:	0	H-VIEW:	0	6
V-VAR:	1	V-VIEW:	-1	6
H-TICK:	10	V-TICK:	10	<input checked="" type="checkbox"/> PIXELS

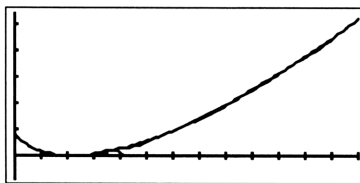
**Note:** *H-VIEW* and *V-VIEW* correspond to *Xrng* and *Yrng*, respectively.

Press **OK** to return to the previous screen, then **ERASE** and **DRAW** to see a plot of the antiderivative.



The differential equations plotter leaves values stored in *X* and *Y*, so you should now purge *X* and *Y* from your **VAR** menu.

In this case, we know a closed-form expression for the antiderivative:  $\int \ln x dx = x \ln x - x + C$ . To meet the initial condition  $y(1) = 0$ , we must choose  $C = 1$ . If you now overdraw your plot of the initial value solution with a plot of  $y = x \ln x - x + 1$  (first, be sure to reset the function type to **FUNCTION**, and the independent variable to 'X' and the dependent variable to 'Y'), you will see that the two plots are in close agreement for  $x \geq 1$ .



This use of the built-in plotter for numerical solutions to initial value problems is especially helpful for viewing plots of antiderivatives that have no elementary closed-form expressions.

### Activity Set 3.3

1. (a) Build a user-defined function  $F$  for  $y = x \cos x$ .
  - (b) Plot  $y = x \cos x$  in disconnected mode using Resolution .1 (if you are plotting from the PLOT menu) or STEP size .1 (if you are plotting from the PLOT screen), with  $Xrng$ : 0 6.28 and  $Yrng$ : -6.3 1.
  - (c) Run program FTC with inputs 0 and 6 to construct an approximation to the antiderivative  $\int_0^x t \cos t \, dt$  using the midpoint rule with  $n = 6$  rectangles.
  - (d) Change the resolution (or STEP size) back to 0 and overdraw with a plot of the derivative of your approximate antiderivative. How closely does it appear to approximate  $y = x \cos x$ ?
  - (e) Now reset the resolution to .1 and overdraw with a plot of your approximate antiderivative found by FTC in (c).
  - (f) Use integration by parts to find an elementary antiderivative of  $y = x \cos x$ . Choose an initial condition so your antiderivative will pass through (0, 0). Reset the resolution to 0 and overdraw your plot in (e) with this exact antiderivative. How closely do the two plots appear to agree?
2. Repeat Activity 1 using the function  $y = x \sin x$ . Use  $Yrng$ : -5.3 2. Zoom out on the vertical axis by a factor of 1.5 after plotting in part (b).

In Activities 3 - 5, the function  $f$  is known to have no elementary, closed-form antiderivative. Proceed as in Activity 1, parts (a) - (e). Note any special conditions.

3. Let  $f(x) = e^{-x^2}$ . Draw the initial plot in disconnected mode with resolution .1 using  $Xrng$ : -2 2 and  $Yrng$ : -2 2. Use  $a = 0$  and  $n = 6$  for program FTC. Reset the resolution to 0 before plotting the results of FTC.
4. Let  $f(x) = \sin x^2$ . Plot everything in connected mode with default resolution 0 over  $Xrng$ : 0 6.28 and  $Yrng$ : -2.5 2.5. Use  $a = 0$  and  $n = 20$  for program FTC. The higher value for  $n$  is needed because of the more frequent oscillations in the graph of  $f$ . Notice that the derivative of the approximate antiderivative begins to deviate from  $f$  as the oscillations increase in frequency.
5. Let  $f(x) = \frac{e^x}{x}$ . Draw the initial plot in disconnected mode with resolution .1 using  $Xrng$ : -6.5 6.5 and  $Yrng$ : 0 6.3. Use  $a = .1$  and  $n = 6$  for program FTC. Before plotting the results from FTC, reset to connected mode with resolution 0 and the independent variable restricted to plot only from 0 to 6.5.
6. Use the Differential Equations Plot Screen to plot the following antiderivatives; use the indicated settings for H-VIEW and V-VIEW.

(a)  $\int_0^x e^{-x^2} dx$ ; H-VIEW: -2 2 and V-VIEW: -2 2

(b)  $\int_0^x \sin x^2 dx$ ; H-VIEW: 0 6.28 and V-VIEW: -2.5 2.5

(c)  $\int_{.1}^x x^{-1} e^x dx$ ; H-VIEW: -6.5 6.5 and V-VIEW: 0 6.3

### 3.4 IMPROPER INTEGRALS

In applications of calculus we often meet improper integrals like  $\int_a^\infty f(x)dx$ . The meaning is clear:

$$\int_a^\infty f(x)dx = \lim_{t \rightarrow \infty} \int_a^t f(x)dx.$$

If the limit is the real number  $L$  then we say that the improper integral  $\int_a^\infty f(x)dx$  converges to  $L$  and write

$$\int_a^\infty f(x)dx = L$$

If the limit does not exist (is not a real number), we say that the improper integral  $\int_a^\infty f(x)dx$  diverges.

Assume that we have a convergent improper integral, say  $\int_a^\infty f(x)dx = L$ . Then

for any value of  $N > a$  we have

$$L = \int_a^N f(x)dx + \int_N^\infty f(x)dx.$$

The second integral in this sum is called the "tail" and if we can choose  $N$  so that the tail is "sufficiently small", then we can approximate  $L$  with the ordinary integral  $\int_a^N f(x)dx$ . We measure "sufficiently small" by specifying an acceptable error tolerance  $\epsilon > 0$ , and then attempt to find a value of  $N$  for which

$$\left| \int_N^\infty f(x) dx \right| = \left| L - \int_a^N f(x) dx \right| < \epsilon.$$

Thus, to within the tolerance specified by  $\epsilon$ , we have  $\int_a^N f(x) dx \approx L$ . The following result is of some help.

### Absolute Comparison Theorem

Suppose that  $f$  and  $g$  are continuous functions for  $x \geq a$  and there is a constant  $K$  such that  $|f(x)| \leq K g(x)$  whenever  $x$  is sufficiently large. Then if  $\int_a^\infty g(x) dx$  converges, so does  $\int_a^\infty f(x) dx$  and

$$\left| \int_a^\infty f(x) dx \right| \leq \int_a^\infty |f(x)| dx \leq K \int_a^\infty g(x) dx.$$

Two convergent improper integrals that are useful for such comparisons are:

$$\int_a^\infty \frac{1}{x^p} dx = \frac{1}{(p-1)a^{p-1}} \quad \text{for } p > 1.$$

$$\int_a^\infty e^{-cx} dx = \frac{1}{ce^{ca}} \quad \text{for } c > 0.$$

(1) Suppose that for sufficiently large  $x$ ,  $|f(x)| \leq \frac{K}{x^p}$  for some  $K > 0$  and  $p > 1$ . Then by the Absolute Comparison Theorem we have

$$\left| \int_N^\infty f(x) dx \right| \leq \int_N^\infty |f(x)| dx \leq K \int_N^\infty \frac{1}{x^p} dx = \frac{K}{(p-1)N^{p-1}}.$$

With  $N > 0$ , to have  $\frac{K}{(p-1)N^{p-1}} < \epsilon$  we choose  $N > \left\{ \frac{K}{\epsilon} \cdot \frac{1}{p-1} \right\}^{1/(p-1)}$

(2) Suppose that for sufficiently large  $x$ ,  $|f(x)| \leq Ke^{-cx}$  for some  $K > 0$  and  $c > 0$ .

Then we have

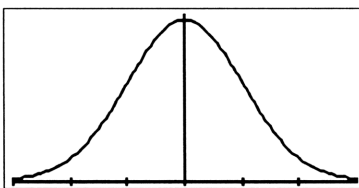
$$\left| \int_N^\infty f(x) dx \right| \leq \int_N^\infty |f(x)| dx \leq K \int_N^\infty e^{-cx} dx = \frac{K}{ce^{cN}}.$$

To have  $\frac{K}{ce^{cN}} < \epsilon$ , choose  $N > \frac{1}{c} \ln \left( \frac{K}{c\epsilon} \right)$ .

**EXAMPLE 8.** The improper integral

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$$

is important in probability theory. A plot of the integrand  $f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$  over the interval  $[-3, 3]$  appears below.



Now  $\int_{-\infty}^{\infty} f(x) dx = \int_{-\infty}^0 f(x) dx + \int_0^{\infty} f(x) dx = \lim_{t \rightarrow -\infty} \int_t^0 f(x) dx + \lim_{t \rightarrow \infty} \int_0^t f(x) dx$ . Since  $\int_0^t f(x) dx$

is the area of the region between the graph of  $y = f(x)$  and the  $x$ -axis over the finite interval  $[0, t]$ , we can interpret the improper integral  $\int_0^{\infty} f(x) dx$  as the area of the

"infinite" region between the graph of  $y = f(x)$  and the  $x$ -axis to the right of  $x = 0$ .

Thus the improper integral  $\int_{-\infty}^{\infty} f(x)dx$  is the area of the entire infinite region between the graph of  $y = f(x)$  and the  $x$ -axis. By symmetry, to show that  $\int_{-\infty}^{\infty} f(x)dx$  converges, it suffices to show that  $\int_0^{\infty} f(x)dx$  converges.

Now  $|f(x)| = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} < \frac{1}{\sqrt{2\pi}} e^{-x/2} < e^{-x/2}$  for  $x > 1$ . Thus  $\int_0^{\infty} f(x)dx$  converges because  $\int_0^{\infty} e^{-x/2} dx$  converges. Moreover, we can take  $K = 1$  and  $c = \frac{1}{2}$  in (2) above, so that

$$N > \frac{1}{c} \ln\left(\frac{K}{c\epsilon}\right) = 2 \ln\left(\frac{2}{\epsilon}\right).$$

With  $\epsilon = 10^{-11}$ ,  $N > 2 \ln\left(\frac{2}{10^{-11}}\right) \approx 52.04$  and we can approximate

$$\int_0^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx \text{ with } \int_0^{52.04} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx.$$

Evaluating this last integral with the HP-48G/GX we obtain .500000000001, so

$$\int_0^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx \approx 1.00000000002 \text{ to within } \epsilon = 10^{-11}.$$

The exact value is 1.



### Activity Set 3.4

In each of the following, establish the convergence and then evaluate the improper integral to within the specified tolerance  $\epsilon$ .

1.  $\int_{\pi/4}^{\infty} \frac{\sin x}{x^4} dx$ ; use  $\epsilon = .01$

2.  $\int_0^{\infty} \frac{x}{\sqrt{x^6 + 4}} dx$ ; use  $\epsilon = .001$

3.  $\int_0^{\infty} \frac{xe^{-2x}}{\sqrt[3]{x^3 + 1}} dx$ ; use  $\epsilon = .001$

4.  $\int_2^{\infty} \frac{1}{\sqrt{x^5 - 1}} dx$ ; use  $\epsilon = .001$

(Hint:  $\frac{1}{\sqrt{x^5 - 1}} < \frac{2}{\sqrt{x^5}}$  for what values of  $x$ ?)

5.  $\int_0^{\infty} \sqrt{x+1} e^{-2x} dx$ ; use  $\epsilon = .005$

(Hint:  $\sqrt{x+1} e^{-2x} = (\sqrt{x+1} e^{-(x+1)}) e^{-x}$ ; what is the maximum value of  $\sqrt{x+1} e^{-(x+1)}$ ?)

# 4

## INFINITE SERIES

Approximations by infinite processes are central to calculus. The concept of the limit of a function

$$\lim_{x \rightarrow c} f(x) = L ,$$

which is fundamental to the development of so much in calculus, has its roots in the intuitive notion that as  $x$  approaches the number  $c$  through an infinite succession of increasingly better approximations

$$x_1, x_2, x_3, \dots \rightarrow c$$

then the corresponding function values are an infinite succession of increasingly better approximations to the limit  $L$  :

$$f(x_1), f(x_2), f(x_3), \dots \rightarrow L .$$

Infinite series, which are expressions of the form

$$\sum_{k=1}^{\infty} a_k = a_1 + a_2 + a_3 + \dots ,$$

also represent approximations by an infinite process. Since the core of any such series is the sequence of terms

$$a_1, a_2, a_3, \dots ,$$

we usually begin a study of series by first considering sequences.

## 4.1 SEQUENCES

A *sequence* of numbers

$$(1) \qquad a_1, a_2, a_3, \dots$$

is simply an *infinite ordered list*. We often use the compact notation  $\{a_k\}_{k=1}^{\infty}$  to represent the sequence (1), and  $a_k$  denotes the  $k^{\text{th}}$  term of the sequence.

More precisely, we can view the sequence (1) as the output values of a function  $f$  that is defined only for the positive integers  $k = 1, 2, 3, \dots$ :

$$\begin{array}{cccc} a_1, & a_2, & a_3, & \dots \\ f(1), & f(2), & f(3), & \dots \end{array}$$

We can use the HP-48G/GX to calculate and view the terms of a sequence. Program **SHO**, given below, will calculate and show the consecutive terms of a sequence  $\{a_k\}_{k=1}^{\infty}$  from a specified starting value of  $k$  to a specified ending value.

This program, and all the others in this chapter, can be found in the **SERIES** subdirectory of the main **CALC** directory.

**SHO** (Show sequence terms)

*Input:* Level 3-5: expressions for the  $k^{\text{th}}$  terms of 1-3 sequences  $\{f_k\}$ ,  $\{g_k\}$  and  $\{h_k\}$  in terms of the variable 'K'

Level 2: a starting value for 'K'

Level 1: an ending value for 'K'

*Effect:* dynamically displays (every two seconds) the successive terms of 1-3 sequences  $\{f_k\}$ ,  $\{g_k\}$  and  $\{h_k\}$  from the starting value of  $k$  to the ending value, beneath the index  $k$ ; leaves everything on the stack.

```
« DEPTH → b n d « CLLCD IF d 3 == THEN → f « b n FOR j j
'K' STO K DUP 1 DISP f EVAL DUP 3 DISP 2 WAIT NEXT » 'K' PURGE
ELSE IF d 4 == THEN → f g « b n FOR j j 'K' STO K DUP 1 DISP f
EVAL DUP 3 DISP g EVAL DUP 5 DISP 2 WAIT NEXT » 'K' PURGE
ELSE → f g h « b n FOR j j 'K' STO K DUP 1 DISP f EVAL DUP 3
DISP g EVAL DUP 5 DISP h EVAL DUP 7 DISP 2 WAIT NEXT » 'K'
PURGE END END » »
```


**EXAMPLE 1.**

- (a) Consider the sequence  $\{f_k\}_{k=1}^{\infty}$  where  $f_k = \frac{1}{k}$ . To see the first 25 terms of this sequence, arrange the stack as follows and press **SHO**

3: '1/K'

2: 1

1: 25

The display will show, in timed two second intervals, the first 25 terms of the index  $k$  and the sequence  $\{1/k\}$ . When done, the stack will contain everything that was displayed, so that you can scroll upward with the  and view any particular term.

- (b) Consider the two sequences  $\{f_k\}_{k=1}^{\infty}$  and  $\{g_k\}_{k=1}^{\infty}$ , where  $f_k = \frac{1}{k}$  and  $g_k = \frac{1}{k^2}$ .

To see terms 10 through 20 of these two sequences, arrange the stack as follows and run program SHO.

```

4: '1/K'
3: '1/K ^ 2'
2: 10
1: 20

```

The display will show, in timed two second intervals, terms 10 through 20 of each sequence, with sequence  $\{f_k\}$  being above sequence  $\{g_k\}$  on the display screen, just below the index  $k$ . When done, everything is left on the stack for your perusal.

It is helpful to regard the terms of the sequences  $\{f_k\}$  and  $\{g_k\}$  in EXAMPLE 1 as being sample values from the two ordinary functions  $f(x) = \frac{1}{x}$  and  $g(x) = \frac{1}{x^2}$ , defined for all  $x \geq 1$ . The graphs of the functions  $f$  and  $g$  consist of all points in the plane  $\left(x, \frac{1}{x}\right)$  and  $\left(x, \frac{1}{x^2}\right)$  for  $x \geq 1$ , respectively. Therefore, the graphs of the sequences  $\{f_k\}$  and  $\{g_k\}$  will consist of the *discrete* points  $\left(k, \frac{1}{k}\right)$  and  $\left(k, \frac{1}{k^2}\right)$  for  $k = 1, 2, 3, \dots$ .

To plot the graph of a sequence  $\{g_k\}_{k=1}^{\infty}$ , we can use program GSEQ<sup>1</sup>.

---

<sup>1</sup>Program GSEQ and a later one GPS were written by Mr. Robert E. Simms of Clemson University. We are indebted to Mr. Simms for permission to use his programs here.

**GSEQ** (Sequence Graph)

*Input:* Level 1: an expression for the  $k^{th}$  term  $a_k$  of a sequence  $\{a_k\}$  in terms of the variable 'K'

Level 2: the number of discrete points on the graph of the sequence  $\{a_k\}$  that you wish to see.

As a stored variable: program SDRW (below), which scales and plots the points that are created by GSEQ and stored in the variable  $\Sigma DAT$

*Effect:* draws coordinate axes and plots, in sequential order, the specified number of discrete points  $(k, a_k)$  on the graph of the sequence  $\{a_k\}$

```
« # 131d # 64d PDIM 0 → eq n k « eq {K k} ↑MATCH DROP 'eq'
STO [0 0] 1 n FOR k k NEXT n →LIST 1 « DUP 'k' STO eq →NUM 2
ROW→ » DOLIST » OBJ→ 1 + ROW→ 'ΣDAT' STO 1 XCOL 2 YCOL
SDRW 7 FREEZE »
```

**SDRW** (a utility subroutine)

*Effect:* Used by GSEQ and GPS to scale and plot the  $xy$ -data in the matrix  $\Sigma DAT$

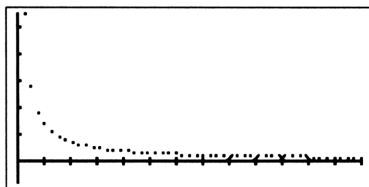
```
« SCLΣ DRAX { # 0d # 0d } PVIEW ΣDAT SIZE 1 GET 1 SWAP FOR
i ΣDAT i ROW- OBJ→ DROP R→C PIXON DROP NEXT »
```

**EXAMPLE 2.** To plot the first 50 terms of the sequence  $\{a_k\}$  where  $a_k = \frac{1}{k}$ , arrange the stack as follows:

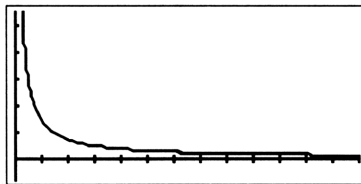
2: '1/K'

1: 50

Now press GSEQ to see the following plot develop:



To graphically verify that the graph of  $a_k = \frac{1}{k}$  is simply a discrete sampling of points from the graph of the function  $f(x) = \frac{1}{x}$ , overdraw the above plot with a plot of the graph of  $f(x) = \frac{1}{x}$ .



### Activity Set 4.1

1. (a) Use program SHO to calculate and view the first 25 terms of the sequence  $\{a_k\}$ , where  $a_k = \frac{k^2}{2^k}$ . Do these terms seem to be approaching a limit?
- (b) Use program GSEQ to plot the first 25 points on the graph of the sequence  $\{a_k\}$ . Does the graph suggest that  $\lim_{k \rightarrow \infty} a_k$  exists?
- (c) Consider the graph in (b) as a discrete sampling of points on the graph of the continuous function  $f(x) = \frac{x^2}{2^x}$  for  $x \geq 0$ . Overdraw your plot in (b) with the graph of  $f$ . What does the graph of  $f$  say about  $\lim_{x \rightarrow \infty} \frac{x^2}{2^x}$ ?
- (d) Use l'Hospital's Rule to analytically find  $\lim_{x \rightarrow \infty} \frac{x^2}{2^x}$ . What is  $\lim_{k \rightarrow \infty} \frac{k^2}{2^k}$ ?

2. (a) Use program SHO to calculate and view the first 25 terms of the sequence  $\{a_k\}$  where  $a_k = \frac{\ln k}{k}$ .
- (b) Now use program GSEQ to plot the first 50 points on the graph of  $\{a_k\}$ . Does the graph suggest a limit for  $\{a_k\}$ ?
- (c) Overdraw your plot in (b) with the graph of the function  $f(x) = \frac{\ln x}{x}$ . Does the graph suggest that  $\lim_{x \rightarrow \infty} \frac{\ln x}{x}$  exists?
- (d) Use l'Hospital's Rule to investigate the limit  $\lim_{x \rightarrow \infty} \frac{\ln x}{x}$ . Based on your results, what can you conclude about  $\lim_{k \rightarrow \infty} \frac{\ln k}{k}$ ?
3. (a) Use program SHO to calculate and view the first 25 terms of the sequence  $\{a_k\}$  where  $a_k = \frac{\sin k}{e^{.05k}}$ . Do these numbers convey to you a sense of what is happening to the terms  $a_k$  as  $k \rightarrow \infty$ ?
- (b) Use program GSEQ to plot the first 50 points on the graph of  $\{a_k\}$ . What does the plot suggest?
- (c) Overdraw your plot in (b) with the graph of  $f(x) = \frac{\sin x}{e^{.05x}}$ , for  $x \geq 0$ . What does the new plot suggest about the limit  $\lim_{x \rightarrow \infty} \frac{\sin x}{e^{.05x}}$  and the limit  $\lim_{k \rightarrow \infty} \frac{\sin k}{e^{.05k}}$ ? Is l'Hospital's Rule of any help here?
4. (a) Use SHO to calculate and view the first 25 terms of the sequence  $\{a_k\}$  where  $a_k = (-1)^{k+1} \left( \frac{1}{k} \right)$ . Do these numbers convey to you a sense of the limit  $\lim_{k \rightarrow \infty} (-1)^{k+1} \left( \frac{1}{k} \right)$ ?



- (b) Use GSEQ to plot the first 50 points on the graph of the sequence  $\{a_k\}$ . Is  $\lim_{k \rightarrow \infty} (-1)^{k+1} \left( \frac{1}{k} \right)$  any more apparent?
- (c) Overdraw your plot in (b) with the graphs of a function  $f$  and its negative  $-f$ . Now what can you conclude about the limit of the sequence  $\{a_k\}$ ?
5. Use program GSEQ to investigate the limits of the following sequences  $\{a_k\}$ :
- (a)  $a_k = \frac{k}{\sqrt{k} + 1}$                       (b)  $a_k = \frac{\cos k}{\sqrt{k}}$
6. (Just for fun!) Plot the first 300 terms of the graph of the sequence  $\{a_k\}$ , where  $a_k = \sin k$ .

## 4.2 SERIES

What do we mean by an *infinite series*?

$$(1) \quad \sum_{k=1}^{\infty} a_k = a_1 + a_2 + a_3 + \dots$$

How can we possibly sum infinitely many numbers?

This is exactly the same kind of question we face when confronted with an improper integral of the form

$$(2) \quad \int_1^{\infty} f(x) dx.$$

How can we possibly integrate from 1 to  $\infty$ ?

Infinite series and improper integrals have much more in common than mere superficial appearances. Indeed, they behave in very similar ways.

The improper integral (2) is a limit of ordinary integrals:

$$\int_1^{\infty} f(x)dx = \lim_{t \rightarrow \infty} \int_1^t f(x)dx .$$

In a similar way, the infinite series (1) is a limit of ordinary sums:

$$(3) \quad \sum_{k=1}^{\infty} a_k = \lim_{N \rightarrow \infty} \sum_{k=1}^N a_k .$$

The ordinary sum  $\sum_{k=1}^N a_k$  is called a *partial sum* of the series. Indeed, the partial sums of the series form a sequence  $\{s_N\}$ , where the terms are:

$$\begin{aligned} s_1 &= a_1 \\ s_2 &= a_1 + a_2 \\ s_3 &= a_1 + a_2 + a_3 \\ &\vdots \\ s_N &= a_1 + a_2 + \dots + a_N = \sum_{k=1}^N a_k \end{aligned}$$

If the sequence of partial sums  $\{s_N\}$  has a limit  $S$

$$\lim_{N \rightarrow \infty} \{s_N\} = S ,$$

then we call  $S$  the *sum of the infinite series*  $\sum_{k=1}^{\infty} a_k$  and write

$$S = \sum_{k=1}^{\infty} a_k .$$

In this case we also say that the series *converges* to the sum  $S$ . If the sequence of partial sums fails to converge to a limit (a real number), then we say that the series *diverges*. It is no wonder that students find infinite series difficult to study. We are combining the terms of one sequence  $a_1, a_2, a_3, \dots$  to form the terms of a new sequence of partial sums  $s_1, s_2, s_3, \dots$ , and then have to consider the limit of the sequence of partial sums:

$$S = \lim_{N \rightarrow \infty} s_N = \lim_{N \rightarrow \infty} \sum_{k=1}^N a_k.$$

How can the HP-48G/GX be of use in a study of a subject that is so analytically complex as infinite series?

To begin, we can use the built-in  $\Sigma$  function to quickly calculate partial sums. For example, consider the series

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots = \sum_{k=1}^{\infty} \frac{1}{2^{k-1}}.$$

Go to the Equation Writer and build the following expression for the  $N^{\text{th}}$  partial sum

$$s_N = \sum_{k=1}^N \frac{1}{2^{k-1}} :$$

$$\sum_{K=1}^N \frac{1}{2^{K-1}}$$

Press **ENTER** to throw it onto the stack:

$$1: \text{'}\Sigma (K = 1, N, 1/2 \wedge (K - 1))\text{'}$$

Open the **SOLVE** application with **↵** **SOLVE**, press **ROOT** and load the expression on level 1 into **EQ** with **↵** **EQ**. Now open the **SOLVR** where you will see boxes labeled **K** **N** **EXPR=**. Ignore the box **K**. Put 10 into **N** and press **EXPR=** to see the 10<sup>th</sup> partial sum

$$1: \text{Expr: } 1.998046875 .$$

Repeat by putting 20, 30, 35 into **N** and using **EXPR=** to obtain the 20<sup>th</sup>, 30<sup>th</sup> and 35<sup>th</sup> partial sums:

$$4: \text{Expr: } 1.998046875$$

$$3: \text{Expr: } 1.99999809266$$

$$2: \text{Expr: } 1.99999999814$$

1: Expr: 1.99999999995

Is there any doubt that the sequence  $\{s_N\}$  of partial sums is converging to  $S = 2$ ?

$$\sum_{k=1}^{\infty} \frac{1}{2^{k-1}} = 2$$

**EXAMPLE 3.** The series  $\sum_{k=1}^{\infty} \frac{1}{k^4}$  is an example of a *p-series*  $\sum_{k=1}^{\infty} \frac{1}{k^p}$ , known to converge if and only if  $p > 1$ .

What is the sum? Using the SOLVR as above to calculate partial sums, we have

$$s_{100} = 1.08232290538$$

$$s_{200} = 1.08232319242$$

$$s_{300} = 1.08232322151$$

$$s_{400} = 1.08232322861$$

$$s_{500} = 1.08232323119$$

$$s_{600} = 1.08232323227$$

$$s_{700} = 1.08232323295$$

$$s_{800} = 1.08232323295$$

Since these last two partial sums agree to 11 decimal places, we have determined that the sum is  $S = 1.08232323295$  to within the precision of the HP-48G/GX.

As an alternative to using the SOLVR, you can use the following program INFSUM. This program, a modification of one due to William C. Wickes of Hewlett Packard in [1], shows the convergence of the series dynamically, by showing the partial sums as a single number with the last digits changing as more terms are added. The program sums series that begin with initial index  $k = 1$ , so you will

have to make adjustments for series that do not start there. It should only be used with series that are *known to converge*.

### INFSUM

*Input:* the term  $a_k$ , for the infinite series  $\sum_{k=1}^{\infty} a_k$ , in terms of the variable 'K'

*Effect:* calculates partial sums  $\sum_{k=1}^N a_k$  until two successive sums agree, displays the last partial sum and the value of  $n$  at which agreement was reached

```
« →f « 1 'K' STO f EVAL 2 'K' STO DO DUP f EVAL + DUP 3
DISP 1 'K' STO+ SWAP UNTIL OVER == END K 1 - 'K' PURGE »
```

Continuing with EXAMPLE 3, put '1/K ^ 4' on level 1 and run program INFSUM. You will see the partial sums accumulate dynamically at the top left of the display screen, until two consecutive sums agree to the precision of the HP-48G/GX. This agreement is reached when  $N = 669$ . Thus to the precision of the machine, the sum is  $S \approx 1.08232323295 = \sum_{k=1}^{669} \frac{1}{k^4}$ .

**EXAMPLE 4.** The series  $\sum_{k=1}^{\infty} (-1)^{k+1}(1/k^6)$  has terms that alternate in sign. It converges by the alternating series test. As an alternating series, we know that the error made in using any partial sum  $s_n$  as the sum of the series is less than the absolute value of the term to be added to get the next partial sum  $s_{n+1}$ . For twelve place accuracy, we must take  $n$  large enough so that  $1/(n+1)^6 < 5 \times 10^{-13}$ . Using the HP-48 for the calculation, we find that  $1/113^6$  is approximately  $4.8 \times 10^{-13}$ . Thus, calculating the 112<sup>th</sup> partial sum with the SOLVR we obtain .985551091299 as our estimate of the

sum, to 12 decimal places. (You can calculate the 113<sup>th</sup> partial sum to see if you get agreement or run program INFSUM.)

Since the partial sums of a series  $\sum_{k=1}^{\infty} a_k$  form a sequence  $\{s_k\}$ , it is also helpful to plot the graph of this sequence. Program GPS does that. Like its predecessor GSEQ for sequences, the code is due to Robert E. Simms.

**GPS** (Graphical partial sums)

*Input:* Level 2: an expression for the  $k^{\text{th}}$  term  $a_k$  of the series

$$\sum_{k=1}^{\infty} a_k, \text{ in terms of the variable 'K'}$$

Level 1: the number of partial sums of the series

$$\sum_{k=1}^{\infty} a_k \text{ that you wish to plot}$$

*Effect:* draws coordinate axes and plots, in sequential order, the specified number of points  $(k, s_k)$  on the graph of the

sequence of partial sums  $\{s_k\}$  for the series  $\sum_{k=1}^{\infty} a_k$

```
« # 131d # 64d PDIM 0 0 → eq n s k « eq { K k } ↑MATCH DROP
'eq' STO [ 0 0 ] 1 n FOR k k NEXT n →LIST 1 « DUP 'k' STO eq →NUM
s + DUP 's' STO 2 ROW→ » DOLIST » OBJ→ 1 + ROW→ 'ΣDAT' STO 1
XCOL 2 YCOL SDRW 7 FREEZE »
```

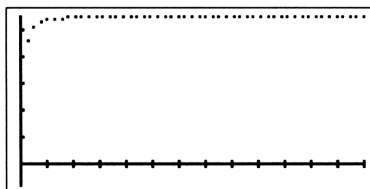
**EXAMPLE 5.** To show the use of program GPS, we graph the first 50 partial sums of the series  $\sum_{k=1}^{\infty} \frac{1}{k^3}$  and  $\sum_{k=1}^{\infty} (-1)^k \frac{1}{k}$ .

For the first one, arrange the stack as follows:

2: '1/K ^ 3'

1: 50

Now press GPS to see



Despite the fact that these partial sums appear to level off rather quickly, convergence is *extremely* slow. Program INFSUM will approximate the sum  $S$  to the full precision of the HP-48 by the 5,849<sup>th</sup> partial sum

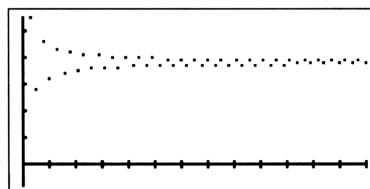
$$\sum_{k=1}^{5849} \frac{1}{k^3} = 1.20205689144 .$$

For the alternating harmonic series,  $\sum_{k=1}^{\infty} (-1)^{k+1} \frac{1}{k}$ , arrange the stack like

2: '(-1) ^ (K + 1) \* (1/K)'

1: 50

and press GPS to see



**Activity Set 4.2.1**

1. Consider the series  $\sum_{k=0}^{\infty} 10^k / k!$ 
  - (a) Apply a standard test to show that the series converges.
  - (b) Plot a graph of the first 25 partial sums of the series.
  - (c) Use the SOLVR to obtain a 12-digit approximation to the sum of the series.
  - (d) Use program INFSUM to obtain a 12-digit approximation to the sum. Which partial sum gives a full precision approximation?
  - (e) Overdraw your plot in (b) with the sum of the series.
2. Repeat Activity 1 with the following series:
  - (a)  $\sum_{k=1}^{\infty} 1/k!$
  - (b)  $\sum_{k=1}^{\infty} 5^k / k!$
  - (c)  $\sum_{k=1}^{\infty} (-1)^{k+1} (k/2^k)$
3. Consider the series  $\sum_{k=1}^{\infty} (-1)^{k+1} \left( \frac{1}{k!} \right)$ .
  - (a) Prove that the series converges.
  - (b) Find a value for  $n$  so that the  $n^{th}$  partial sum will approximate the sum of the series to 12 decimal digits.
  - (c) Use the SOLVR to calculate the partial sum  $s_n$  for the value of  $n$  in (b).
4. Prove that the series  $\sum_{k=1}^{\infty} (k+1)/k^{10}$  converges. What is the sum to 12 decimal digits?



5. (a) Prove that  $\sum_{k=1}^{\infty} k!/(k+2)!$  converges.
- (b) Plot a graph of the first 25 partial sums of this series.
- (c) Apply program INFSUM to find the sum of the series. Watch closely what takes place. How can you explain this?

## Series and Improper Integrals

We mentioned earlier that the connection between series and improper integrals was more than cosmetic:

$$\sum_{k=1}^{\infty} a_k \quad \text{versus} \quad \int_1^{\infty} f(x)dx .$$

Indeed, for series of positive terms we have the *integral test*.

### Integral Test

Let  $f(x)$  be a continuous, positive, decreasing function for  $x \geq 1$  and let  $a_k = f(k)$  for  $k = 1, 2, \dots$ . Then if either  $\int_1^{\infty} f(x)dx$  or  $\sum_{k=1}^{\infty} a_k$  converges, both converge. If either  $\int_1^{\infty} f(x)dx$  or  $\sum_{k=1}^{\infty} a_k$  diverges, both diverge.

In other words, the series  $\sum_{k=1}^{\infty} a_k$  and the integral  $\int_1^{\infty} f(x)dx$  converge or diverge together.

Suppose that we have a convergent pair,  $S = \sum_{k=1}^{\infty} a_k$  and  $I = \int_1^{\infty} f(x)dx$ , as above.

In Chapter 3 (Section 3.4.) we saw that

$$I = \int_1^{\infty} f(x)dx = \int_1^N f(x)dx + \int_N^{\infty} f(x)dx.$$

(the “tail”)

If we can make the tail of the integral small enough, then we can approximate the improper integral  $I$  with the ordinary integral  $\int_1^N f(x)dx$  :  $I \approx \int_1^N f(x)dx$ .

Similarly,

$$S = \sum_{k=1}^{\infty} a_k = \sum_{k=1}^N a_k + \sum_{k=N+1}^{\infty} a_k$$

(the “tail”,  $R_N$ )

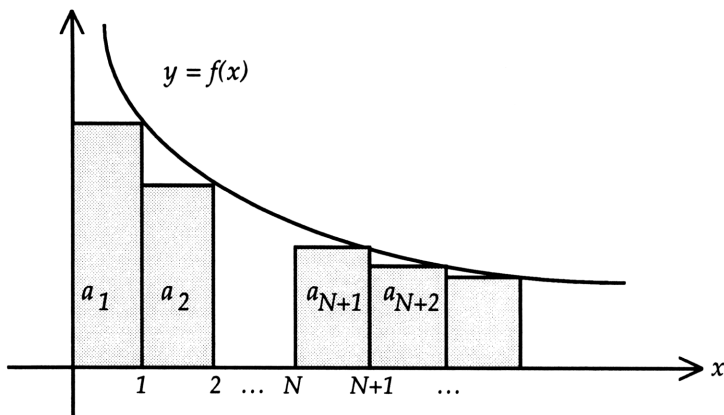
And if we can make the tail of the series small enough, then we can approximate the series with the ordinary sum  $\sum_{k=1}^N a_k$  :  $S \approx \sum_{k=1}^N a_k$ .

As with improper integrals, we measure “small enough” by specifying an acceptable error tolerance  $\epsilon > 0$ , and then attempt to find a value for  $N$  so that

$$\left| \sum_{k=N+1}^{\infty} a_k \right| = \left| S - \sum_{k=1}^N a_k \right| < \epsilon .$$

Then, to within the tolerance specified by  $\epsilon$ ,  $S \approx \sum_{k=1}^N a_k$ .

The trick, then, is to relate the tail  $R_N$  of the series to the tail of the integral. The following picture tells all:



$$R_N = \sum_{k=N+1}^{\infty} a_k = a_{N+1} + a_{N+2} + \dots \leq \int_N^{\infty} f(x) dx$$

(tail of the integral)

Since  $f(x) \geq 0$  for all  $x$ , the **Absolute Comparison Theorem** for improper integrals applies (see Section 4.4).

- (1) Suppose that for sufficiently large  $x$ ,  $f(x) \leq \frac{K}{x^p}$  for some  $K > 0$  and  $p > 1$ . Then by (1) in Section 4.4, we have

$$R_N \leq \int_N^{\infty} f(x) dx \leq \frac{K}{(p-1)N^{p-1}}.$$

Thus, to have  $R_N \leq \frac{K}{(p-1)N^{p-1}} < \epsilon$ , we choose

$$N > \left\{ \frac{K}{\epsilon} \cdot \frac{1}{p-1} \right\}^{1/(p-1)}$$

- (2) Suppose that for sufficiently large  $x$ ,  $f(x) \leq Ke^{-cx}$  for some  $K > 0$  and  $c > 0$ . Then by (2) in Section 4.4, we have

$$R_N \leq \int_N^{\infty} f(x) dx \leq \frac{K}{ce^{cN}}.$$

To have  $R_N \leq \frac{K}{ce^{cN}} < \epsilon$ , we choose  $N > \frac{1}{c} \ln \left( \frac{K}{c\epsilon} \right)$ .

**EXAMPLE 6.** Show that  $\sum_{k=1}^{\infty} \frac{\cos(k)}{k^5}$  converges and find its sum  $S$  to within  $\epsilon = 10^{-6}$ .

The natural integral to use is  $\int_1^{\infty} \frac{\cos(x)}{x^5} dx$ , which converges on comparison with

$\int_1^{\infty} \frac{1}{x^5} dx$ . Thus, the series  $\sum_{k=1}^{\infty} \frac{\cos(k)}{k^5}$  also converges. Since  $f(x) = \frac{\cos(x)}{x^5} \leq \frac{1}{x^5}$  for

$x \geq 1$ , to approximate the sum  $S$  by  $\sum_{k=1}^N \frac{\cos(k)}{k^5}$  to within  $\epsilon = 10^{-6}$ , we need to choose

$$N > \left\{ \frac{1}{\epsilon} \cdot \frac{1}{5-1} \right\}^{1/(5-1)} = \left\{ \frac{10^6}{4} \right\}^{1/4} \approx 22.36.$$

Thus  $S \approx \sum_{k=1}^{23} \frac{\cos(k)}{k^5} \approx .522820670966$  (Equation Writer,  $\rightarrow$ NUM)

### Activity Set 4.2.2

In each of the following, establish the convergence of the series and then find the sum to within the specified tolerance  $\epsilon$ .

1.  $\sum_{k=1}^{\infty} \frac{\sin k}{k^4}$ ,  $\epsilon = .01$
2.  $\sum_{k=1}^{\infty} \frac{k}{\sqrt{k^6 + 4}}$ ,  $\epsilon = .001$
3.  $\sum_{k=1}^{\infty} a_k$ , where  $a_k = \frac{k e^{-2k}}{\sqrt[3]{k^3 + 1}}$

$$4. \sum_{k=2}^{\infty} \frac{k}{\sqrt{k^5 - 1}}, \epsilon = .001$$

$$5. \sum_{k=1}^{\infty} \ln\left(\frac{1}{k^2}\right), \epsilon = .0001$$

## The Ratio Test

The well-known *ratio test* states that for a series  $\sum_{k=1}^{\infty} a_k$  of *positive* terms, if the ratios  $\frac{a_{k+1}}{a_k}$  approach a limit  $r$ ,

$$\lim_{k \rightarrow \infty} \frac{a_{k+1}}{a_k} = r$$

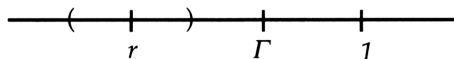
then the series converges for  $r < 1$  and diverges for  $r > 1$ .

Suppose we have a series of positive terms  $\sum_{k=1}^{\infty} a_k$  that is known to converge by the ratio test. As in the preceding section, we wish to approximate the sum  $S$  of the series by a finite sum  $\sum_{k=1}^N a_k$  to within a specified tolerance  $\epsilon$ . We must therefore choose a value for  $N$  that will make the absolute value of the tail less than  $\epsilon$ :

$$(*) \quad \left| \sum_{k=N+1}^{\infty} a_k \right| = \sum_{k=N+1}^{\infty} a_k < \epsilon.$$

Let  $\Gamma$  be any number so that  $r < \Gamma < 1$ . Since  $\lim_{k \rightarrow \infty} \frac{a_{k+1}}{a_k} = r$ , we can choose  $N$  so that

$$(1) \quad \frac{a_{k+1}}{a_k} < \Gamma \text{ for all } k \geq N.$$



For all  $k \geq N$ , the ratios  $\frac{a_{k+1}}{a_k}$  will lie in the open interval centered at  $r$ .

If we can *also* choose  $N$  so that

$$(2) \quad a_N \left( \frac{\Gamma}{1-\Gamma} \right) < \epsilon$$

then we will have our desired result (\*). The justification is as follows.

Since we have chosen  $N$  so that  $\frac{a_{k+1}}{a_k} < \Gamma$  for  $k \geq N$  then we have

$$a_{N+1} \leq \Gamma a_N$$

$$a_{N+2} \leq \Gamma a_{N+1} \leq \Gamma^2 a_N$$

$$a_{N+3} \leq \Gamma a_{N+2} \leq \Gamma^3 a_N$$

$\vdots$

etc.

Thus

$$(i) \quad a_{N+1} + a_{N+2} + \dots + a_{N+M} < (\Gamma + \Gamma^2 + \dots + \Gamma^M) a_N.$$

Since  $|\Gamma| < 1$ , the geometric series  $\Gamma + \Gamma^2 + \Gamma^3 + \dots$  converges to  $\frac{\Gamma}{1-\Gamma}$ . In fact, since the sequence of partial sums of this geometric series is bounded and increasing, we have

$$(ii) \quad \Gamma + \Gamma^2 + \dots + \Gamma^M < \frac{\Gamma}{1-\Gamma} \text{ for all } M.$$

Combining (i) and (ii) we have

$$(iii) \quad a_{N+1} + \dots + a_{N+M} < a_N \left( \frac{\Gamma}{1-\Gamma} \right).$$

Therefore, we can obtain (\*) by choosing  $N$  so that, in addition to (1), also  $a_N \left( \frac{\Gamma}{1-\Gamma} \right) < \epsilon$ .

**EXAMPLE 7.** Consider the series  $\sum_{k=1}^{\infty} \frac{k^{10}}{10^k}$ . By the ratio test,  $\lim_{k \rightarrow \infty} \frac{a_{k+1}}{a_k} = \frac{1}{10}$ , so the series converges. We wish to approximate the sum of the series to within  $\epsilon = 10^{-4}$ . Choose  $\Gamma = \frac{1}{2}$ . To satisfy condition (1), choose  $N$  so that  $\frac{1}{10} \left( \frac{k+1}{k} \right)^{10} < \frac{1}{2}$  for all  $k \geq N$ . This reduces to  $\left( 1 + \frac{1}{k} \right)^{10} < 5$  for  $k \geq N$ , and the smallest such  $N = 6$ . To satisfy condition (2) we must *also* choose  $N$  so that  $\frac{N^{10}}{10^N} < \epsilon = 10^{-4}$ . Build a user-defined function for  $G(N) = \frac{N^{10}}{10^N}$  and evaluate  $G$  for different values of  $N$ , starting with the value  $N = 6$ . We find that  $N = 17$  is the first value that gives  $G(N) < 10^{-4}$ . Now evaluate the sum  $\sum_{k=1}^{17} \frac{k^{10}}{10^k}$  on the HP-48G/GX to obtain  $S \approx 376.17943$ .

### Activity Set 4.2.3

Establish the convergence of each of the following series by the ratio test and then find the sum of the series to within the specified tolerance  $\epsilon$ .

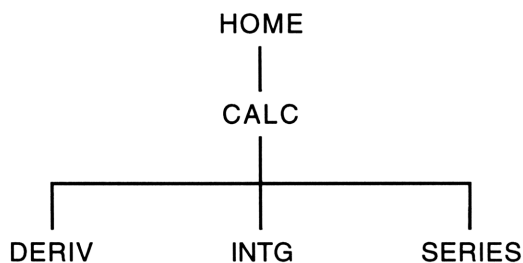
1.  $\sum_{k=1}^{\infty} \frac{k}{2^k}$ ,  $\epsilon = 10^{-6}$
2.  $\sum_{k=1}^{\infty} \frac{k^3}{e^k}$ ,  $\epsilon = 10^{-6}$
3.  $\sum_{k=1}^{\infty} \frac{(k+1)(k+2)}{k!}$ ,  $\epsilon = 10^{-6}$
4.  $\sum_{k=1}^{\infty} \frac{10^k k!}{(2k+1)!}$ ,  $\epsilon = 10^{-6}$
5.  $\sum_{k=1}^{\infty} \frac{.7^k (k+1)}{k}$ ,  $\epsilon = 10^{-6}$

## APPENDIX

# TEACHING CODE FOR PART I

The special-purpose HP-48G/GX programs for teaching single variable calculus 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 calculus is stored in a directory called CALC. The CALC directory contains three subdirectories, each one containing teaching code related to a major topic.



- Subdirectory DERIV. Contains the teaching code related to differentiation (see Chapters 1-2): INV.F, derXROOT, IM.y', Y', F.XY, INFL1, INFL2, NEWTON, TAY.A, TAYLAT, TAN.L, and PAR'.




- Subdirectory **INTG**. Contains the teaching code related to integration (see Chapter 3): **FABSTO**, **NSTO**, **GRECT**, **LRECT**, **RRECT**, **MID**, **TRAP**, **SIMP**, **SUM**, **F.val** and **FTC**.
- Subdirectory **SERIES**. Contains the teaching code related to series (see Chapter 4): **SHO**, **GSEQ**, **INFSUM**, **GPS**, and **SDRW**.

To execute any of these programs, open the **CALC** directory with the **CALC** key, then open the appropriate subdirectory with its menu key. Put the necessary inputs to a particular program on the stack, execute the name of the program by typing the correct name and use the **ENTER** key, or using the appropriate menu key.

You have access to built-in commands from any **CALC** subdirectory without exiting from the subdirectory. Type the command, 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, return directly to the subdirectory you are in with the **VAR** key.

Because the HP-48 needs program **derXROOT** in order to differentiate the **XROOT** function, we recommend that you move it from your **DERIV** subdirectory into your **HOME** directory, where it will be accessible from anywhere. Here is how to do that. Open your **DERIV** subdirectory and recall **derROOT** to the stack with **↗** **DERXR**, then put its name on the stack using **'** **DERXR** **ENTER**. Use **↗** **HOME** to go **HOME**. Press **STO** to store the program in **HOME** directory.

To move up from a particular **CALC** subdirectory to the main **CALC** directory, use the **UP** key (the left-shifted **↵** key). To rearrange any of the variables in a subdirectory of **CALC** (including the **CALC** 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!

### HP-48G/GX Teaching code

derXROOT	Derivative of XROOT
F.val	Utility for SUM
F.XY	Evaluate $F(x,y)$
FABSTO	Store $f, a, b$
FTC	Fundamental Theorem of Calculus
GPS	Graphical partial sums
GRECT	Graphical rectangles
GSEQ	Sequence graph
IM.y'	Implicit differentiation
INFL1	Inflection point via $f'$
INFL2	Inflection point via $f''$
INFSUM	Sum a series
INV.F	Inverse function
LRECT	Left rectangle sum
MID	Midpoint rectangle sum
NEWTON	Newton's Method
NSTO	Store $n$
PAR'	Parametric derivative
RRECT	Right rectangle sum
SDRW	Utility for GSEQ, GPS
SHO	Show sequence
SIMP	Simpson's rule
SUM	Utility for LRECT, RRECT, etc.
TAN.L	Tangent line
TAY.A	Taylor polynomial at $x = a$
TAYLAT	Taylor polynomial at $x = a$ (alternate version)
TRAP	Trapezoid sum
Y'	Implicit derivative

## **PART**

## **II**

# **DIFFERENTIAL EQUATIONS**

Part II contains examples and exercises which require a graphics calculator and supplement the elementary differential equations course. Students learn early in such a course that important mathematical models for scientific problems often contain differential equations and that particular solutions of these equations describe the behavior of the model. The problem solver often has some intuition concerning how the system should behave and the graphical properties of a single solution or a family of solutions are an important clue to the correctness of the model and provide qualitative properties of the solution. Even if analytical expressions can be obtained for the solutions, their graphs may reveal behavior a scientist may not discover from these expressions.

The HP-48G/GX calculator is a great graphics and computational tool in this course. It can be used in class to illustrate concepts. It can be used for homework in your favorite study area, for example, a library or a dormitory room. The graphs and computations that are created on the calculator can be stored or recreated on a microcomputer. The presentation does not require that the reader be a good HP 48 programmer since nearly all of the programs are explained within this manual.

One of the distinctive features of the HP-48G/GX is a built-in program for calculating and displaying in the same graph approximate solutions to one or more initial value problems containing differential equations. The HP-48S/SX does not have such a program but we will present user programs which will accomplish the same purposes. To emphasize the statement given above, the capability to easily display solutions of several problems allows us to study how the solutions depend on various parameters and to focus on geometrical characteristics of a system.

The first part of Chapter 5 describes the programs that have been provided (in the HP-48G/GX) for obtaining and plotting approximate solutions of initial value problems. Then we show how to construct programs using elementary algorithms (the Euler and improved Euler algorithms) for obtaining and plotting approximate solutions. These programs can be used on both the HP-48S/SX and HP-48G/GX. They can be adapted for various special purposes such as creating other graphical displays or for use in higher order numerical methods for differential equations such as the Runge-Kutta algorithms.

Chapter 6 contains examples and exercises to illustrate graphical study of the characteristics of solutions obtained in the portion of the course dealing with first order differential equations. Chapter 7 concerns the solution of two first order differential equations or a single second order equation with initial conditions. Because the HP-48 processes vector and scalar quantities with many of the same commands, it is possible to use some programs for scalar systems as vector programs, including the algorithms for solving initial value problems (Euler, improved Euler, Runge-Kutta, etc.). To solve and plot the solutions of a second order differential equation with initial conditions, we borrow from problems involving particle motion. For displacement  $x(t)$  and velocity  $v(t)$ , of the solution of the pair of equations

$$dx/dt = v, \quad dv/dt = g(t, x, v), \quad x(t_0) \text{ and } v(t_0) \text{ given,}$$

the function  $x(t)$  satisfies the second order initial value problem

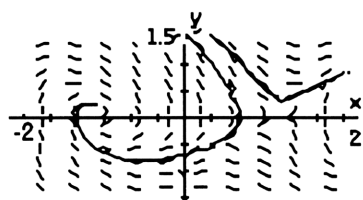
$$\frac{d^2 x}{dt^2} = g(t, x, \frac{dx}{dt}), \quad x(t_0) \text{ and } \frac{dx}{dt}(t_0) \text{ given,}$$

and conversely, if we have a solution  $x(t)$  of the second order initial value problem, the pair  $x(t)$ ,  $dx/dt(t) = v(t)$  satisfies the pair of first order differential equations. Thus, to solve the second order initial value problem we make a vector  $[Y_1, Y_2] = [x, v]$  and solve the vector system

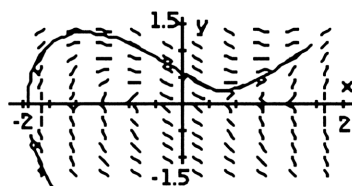
$$\frac{dY}{dt} = \begin{bmatrix} \frac{dx}{dt} \\ \frac{dv}{dt} \end{bmatrix} = F(t, Y) = \begin{bmatrix} v \\ g(t, x, v) \end{bmatrix} = \begin{bmatrix} Y_2 \\ g(t, Y_1, Y_2) \end{bmatrix}$$

with  $Y(t_0) = \text{column } [x(t_0), v(t_0)]$  as given using the same algorithms as for a first order scalar problem. In chapter 7, exercises are provided to study in particular the solutions of the second order differential equations encountered in linear and nonlinear models of mechanical springs and electrical circuits.

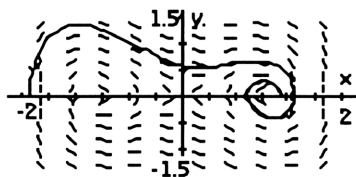
Chapter 8 contains programs that construct solutions of linear vector systems of differential equations of the form  $dy/dt = A y + f(t)$ . Chapter 9 contains several problems that extend the earlier material and result in three or more differential equations and contains a set of programs that can be used to sketch the direction field for a pair of differential equations.



$$x'' + x' - x^2 + 1 = 0$$



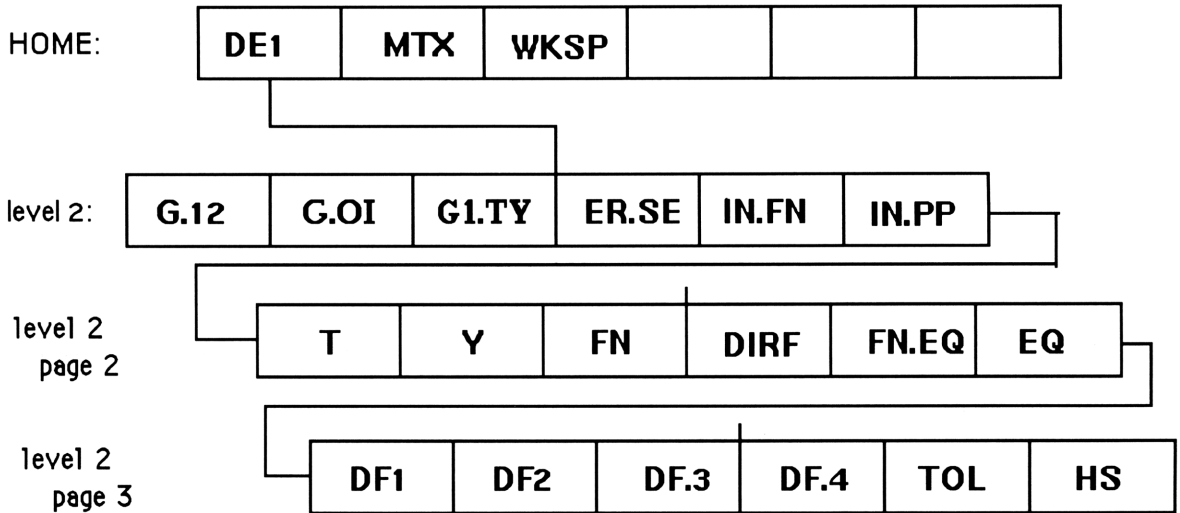
$$x'' + x' - x^2 = 0$$



$$x'' + .4x' + x^3 - x = 0$$

Assorted Direction Fields  
with Solution Overlays

A convenient directory structure for the material in Part II is as follows:



HOME contains various entries, perhaps one being the DE1 subdirectory – in which you may group together all the stored quantities for differential equations. HOME is the *parent* directory of subdirectory DE1. HP-48S owners should create directory DE.2 instead of DE.1 (see below).

To create subdirectory DE1, press , DE1 CRDIR<sup>M</sup> (note that CRDIR<sup>M</sup> appears on the MEMORY menu): a new label DE1<sup>M</sup> appears in the original VAR menu. Pressing DE1<sup>M</sup> will send you to this subdirectory, which is now empty.

You should enter the programs in DE1: IN.FN, IN.PP, G1.TY, G.OI, G.12 ER.SE, FN, T, Y, TOL, HS, PPAR, and EQ. The programs INIT1, GRAF, N, H, PPAR, EULER, IULER, INIT.I, G.TYI, etc. (if used) should be entered into a separate directory, DE2. Subdirectory MTX should contain the PIV, ROKL, and CHAR programs. You may want an WKSP directory for odds and ends.

# 5

## PLOTTING SOLUTIONS FOR DIFFERENTIAL EQUATIONS ON THE HP-48

Suppose we wish to plot an approximate solution of an initial value problem

$$\frac{dy}{dt} = F(t, y), \quad y(t_0) = y_0$$


for some interval  $t_0 \leq t \leq t_f$ , where  $t_f$  may not be predetermined. What inputs to a calculator are required?

- A program that gives the value of  $F$  when  $t$  and  $y$  are specified.
- The initial quantities  $t_0, y_0$  and criteria for completion (e. g. the final value of  $t_f$ ).
- The plot window must be specified and the plot screen may have to be erased.
- It may also be necessary to specify an appropriate algorithm for computing the approximate solution and any necessary inputs to the algorithm such as a global error tolerance and a starting value of the step size.
- The command to draw.

There are three methods on the HP-48G/GX to provide these inputs and obtain a plot of an approximate solution. The built-in method prompts the user for the necessary information with input forms and choose boxes. Alternatively, we can

construct a set of programs that take or generate some of the required inputs from the stack and then call the basic algorithm to calculate solutions in a plotting program. This alternative method is particularly useful when only one or two of the inputs must be modified or when the stopping criterion is nonstandard. The third method is to construct programs that employ easy algorithms for computing and plotting approximate solutions to initial value problems. This last method can be used on the HP-48S/SX calculators. We will illustrate each of these methods with EXERCISES. The user must make a decision on the appropriate method for the other EXERCISES. In this section we begin with the built-in method, then pass on to the second alternative. The third method will be featured in a separate section of this chapter.

## 5.1 USING BUILT IN PROGRAMS

Open the PLOT application with  PLOT. The cursor keys can then be used to move around the screen and highlight the desired fields. Highlight the TYPE field, press CHOOS, highlight Diff Eq and press OK. If the STIFF field is checked, highlight it and press CHK to remove the check. This will cause the Runge-Kutta Feldberg algorithm to be used for the initial value problem.

- Highlight the F field, type in the desired function  $F(T,Y)$  and press OK.
- Set the INDEP variable to T and specify its initial and final values. Set the SOLN field to Y and specify its initial value.
- Press OPTS, set the H-VAR (by pressing CHOOS, highlight the desired field and OK) and the V-VAR variables in a similar manner, set the limits of H-VIEW and V-VIEW.
- Press ERASE and DRAW.



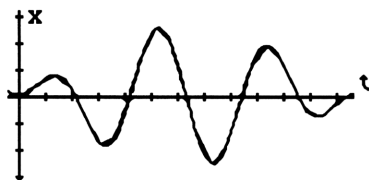
**EXERCISE 5.1:** Construct a plot of the solution of  $y' + 3y = \cos t$ ,  $y(0) = .3$  for  $0 \leq t \leq 6.283$  using the calculator's differential equation plot feature. Enter  $\text{COS}(T) - 3*Y$  in the F field of the input form. (Note that the calculator automatically places this function within ' marks.) Make sure T is the INDEP variable, the H-VIEW is set to 0 6.283 and the V-VIEW is set to -.5 .5, then ERASE and DRAW.

After completing this EXERCISE, you should check to see that the program which calculates the values of  $F(T,Y)$  is stored in the variable EQ. To get some confidence in the calculator solution, we can plot the exact solution  $y = .3 \cos(t) + .1 \sin(t)$ . Press ON to return to the PLOT application, change the TYPE to Function, and enter  $.3*\text{COS}(T) + .1*\text{SIN}(T)$  as EQ. (Again the calculator places ' marks around the function.) In this case we want to overlay the new plot over the old one so do not ERASE. Press DRAW. Note the good agreement between the approximate solution and the exact solution.

**EXERCISE 5.2:** Construct a solution of  $y' = \sin(ty)$ ,  $y(0) = 2$  for  $0 \leq t \leq 6$ . Choose the program 'SIN(T\*Y)' (or  $\ll \text{'SIN(T*Y)'} \text{ EVAL } \gg$ ) and V-VIEW as 0 8. Now overlay the solution of the same differential equation which satisfies the initial condition  $y(0) = 4$ , then overlay a third solution of the same differential equation which satisfies the condition  $y(0) = 6$ . (Note: We do not know a formula for the exact solutions of this differential equation and this overlay process will be used frequently in this chapter to indicate the sensitivity of a problem to its inputs.)

**EXERCISE 5.3:** Construct a graph of the solution of  $y'' + .5 y' + y = 0$ ,  $y(0) = 0$ ,  $y'(0) = 1$ , for  $0 \leq t \leq 6.283$ . We convert this problem to a first order format using the variables  $y$  and  $y'$  as components of a vector  $w = [y, y']$ . Then  $w' = [w(2), -(0.5 w(2) + w(1))]$  and  $w(0) = [0 \ 1]$ . Our procedure calls for an appropriate F function which in this case will be a 2-vector. Then we provide the program  $\ll \text{'W(2)'} \text{ EVAL } '.5*W(2)+W(1)'\text{ EVAL NEG } 2 \rightarrow \text{ARRY } \gg$  for F together with the

INDEP variable name T, SOLN name W and the INIT vector [0 1] for SOLN. If we want a graph of W(1) versus T we choose INDEP (i. e. 0) for the H-VAR and SOLN(1) for the V-VAR on the OPTS page. V-VIEW should be set at -.8 .8. (If we want a W(1) versus W(2) plot we choose SOLN(1) for H-VAR and SOLN(2) for V-VAR on the OPTS page.)



$$x'' + 49x = 12 \cos(5t), x(0) = x'(0) = 0$$

**EXERCISE 5.4:** The figure shown above is a plot of the solution of the indicated problem for  $0 \leq t \leq \pi$ . What function F in the variables T and Y (vector with 2 components) is appropriate for the calculator input form ?

Hewlett Packard has also provided several "smaller" programs that perform either individual or multiple steps in either of two basic algorithms for approximating the solution to a differential equation. These programs can be embedded in user programs to produce variations of the basic program described above. The advantages gained by this process include some speedup when most parameters are already set and any modifications of the basic problem not treated easily by the first method. For example, the final time  $t_f$  may be "when some condition is satisfied" rather than a simple number which is known beforehand.

You can construct programs that ask for part of the total information required for a solution plot. For example, the first program IN.FN asks you to write a program for the function F(T,Y) (in variables T Y) which is then stored in FN.

Program Name:	<b>IN.FN</b>
Purpose:	The user supplies program FN
Stored Quantities:	none
No input is required. The appropriate response is a program.	
<pre>&lt;&lt; " ENTER PRG FOR FN IN T Y" " "</pre>	
<pre>INPUT OBJ→ 'FN' STO &gt;&gt;</pre>	

Example responses might be `<< '-T*Y' EVAL >>` ( or the reverse Polish notation program `<< T Y * NEG >>`) for the function  $F(T,Y) = -T*Y$  or

`<< 'Y(2)' EVAL 'Y(1)' EVAL NEG 2 →ARRY >>`

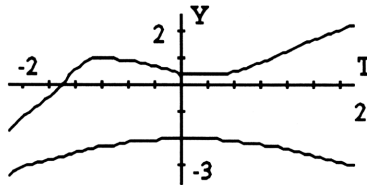
for the function  $F(T,Y) = \text{column } [Y(2), -Y(1)]$ .

The next program asks the user to set the viewing window for the plot.

Program Name:	<b>IN.PP</b> (plot parameters)
Purpose:	The user supplies XRNG and YRNG
Stored Quantities:	none
No input stack is required. The appropriate response for the first query is a pair of numbers, H-min and H-max. The response for the second query is a pair of numbers V-min and V-max.	
Output: New values for XRNG, YRNG. PICT has been erased.	
<pre>&lt;&lt; " KEY IN XRNG" " " INPUT OBJ→ XRNG</pre>	
<pre>" KEY IN YRNG" " " INPUT OBJ→ YRNG ERASE &gt;&gt;</pre>	

*Note:* The reader has probably correctly inferred that the commands XRNG and YRNG set the H-VIEW and V-VIEW variable ranges.

We wish to present a program to give a composite graph in the (T, Y) plane for a differential equation with one or more initial conditions such as indicated in the figure shown below.



**Solutions of  $Y' = \text{SIN}(T^2 - Y^2)$   
with  $Y(-2) = -3$  and  $Y(-2) = -1.5$**

The following program contains the basic ingredients of a user plotting program. The number 1 in the name indicates that the program is for a scalar differential equation. The TY designation indicates that the plot is a (T, Y) plot.

Program Name: **G1.TY**

Purpose: Generate a T Y plot of the solution to  $T_f$


Stored Quantities: XRNG YRNG FN TOL HS

Input	level 3	level 2	level 1
	$T_0$	$Y_0$	$T_f$

The output stack is empty, the variables T Y contain updated values.

```
<< { # 0d # 0d } PVIEW DRAX 3 ROLLD 'Y' STO 'T' STO
  → TF << { T Y FN } TOL HS T Y R→C 4 ROLLD DO
RKFSTEP T Y R→C DUP 6 ROLLD 5 ROLL LINE DUP T +
TF UNTIL > END DROP TF T - RKFSTEP T Y R→C DUP 6
  ROLLD 5 ROLL LINE DROP TF T - RKFSTEP T Y R→C 5
  ROLL LINE 3 DROPN >> PICTURE >>
```

*Notes:* Typical numbers for HS and TOL are .005 and .00005 and are to be stored before execution of this program. If the user wants other names for the variables other than T Y FN, such changes can be made by substituting for { T Y FN}, T, and Y, the desired alternate notation. The command RKFSTEP invokes the built-in Runge-Kutta-Feldberg program for one step.

**EXERCISE 5.1a:** Construct a plot of the solution of  $y' + 3y = \cos t$ ,  $y(0) = .3$  for  $0 \leq t \leq 6.283$  using the programs described above. Execute IN.FN, respond by typing `<< 'COS(T) - 3*Y' EVAL >>` and press ENTER. Execute IN.PP, respond by typing 0 6.283 and ENTER, respond by typing -.5 .5 and press ENTER. Put 0 .3 6.283 on the stack and execute G1.TY. As in EXERCISE 1.1, plot the exact solution  $y = .3 \cos(t) + .1 \sin(t)$ . Press  PLOT, change the TYPE to Function, and enter `.3*COS(T) + .1*SIN(T)` as EQ. In this case we want to overlay the new plot on the old one so do not ERASE. Now press DRAW. Note the good agreement between the approximate solution and the exact solution.

**EXERCISE 5.2a:** Construct and plot solutions of  $y' = \sin(t^{1.5}y^R)$ ,  $y(0) = 2$  for  $0 \leq t \leq 8$  when R has the values .75, .5 and .33, in the same picture as follows: Execute IN.FN, respond by typing `<< 'SIN(T^1.5*Y^R)' EVAL >>` and press ENTER. Execute IN.PP, respond by typing 0 8 and ENTER, respond by typing 0 4 and press ENTER. Put .75 on the stack and press 'R' STO, then put 0 2 8 on the stack and execute G1.TY. Now put .5 on the stack, press 'R' STO then put 0 2 8 on the stack and execute G1.TY. Finally put .33 on the stack, press 'R' STO, put 0 2 8 on the stack and execute G1.TY. Notes: Here we are observing the solution for three values of a parameter. The process of storing a value for R and placing appropriate input on the stack for the graph program can be abbreviated in various ways. For example, store the program `<< 'R' STO 0 2 8 >>` under a name, say P.1. Then put one of the values of R on the stack, execute P.1, then execute G1.TY, etc.

Suppose that the user wants to plot  $(T, Y(I))$  for a vector system  $Y' = F(T, Y)$ . We will call the program **G.0I** where the 0 represents the  $T$  variable and the I represents the  $Y(I)$  variable. The modification consists of changes made to **G1.TY** in four locations in which the  $Y$  number in **G1.TY** is changed to ' $Y(I)$ ' EVAL and adding the first  $\ll \rightarrow I$  and the last  $\gg$ . The user can avoid retyping the whole program by pressing '**G1.TY**' RCL, EDIT, typing the corrections pressing ENTER, '**G.0I**' STO.

Program Name:	<b>G.0I</b>			
Purpose:	Generate a $T$ $Y(I)$ plot of the solution to $T_f$			
Stored Quantities:	XRNG	YRNG	FN	TOL HS
Input	level 4	level 3	level 2	level 1
	$T_0$	vector $Y_0$	$T_f$	I

The output stack is empty, the variables  $T$  and  $Y$  contain updated values.

```

<<  $\rightarrow$  I << { # 0d # 0d } PVIEW DRAX 3 ROLLD 'Y' STO '
T' STO  $\rightarrow$  TF << { T Y FN } TOL HS T 'Y(I)' EVAL R $\rightarrow$ C 4
ROLLD DO RKFSTEP T 'Y(I)' EVAL R $\rightarrow$ C DUP 6 ROLLD 5
ROLL LINE DUP T + TF UNTIL > END DROP TF T -
RKFSTEP T 'Y(I)' EVAL R $\rightarrow$ C DUP 6 ROLLD 5 ROLL LINE
DROP TF T - RKFSTEP T 'Y(I)' EVAL R $\rightarrow$ C 5 ROLL
LINE 3 DROPN >> PICTURE >> >>

```

**EXERCISE 5.3a:** Construct a composite  $(t, x)$  plot of the solutions of  $x'' + R x' + x = 0$ ,  $x(0) = 0$ ,  $x'(0) = 1$ , for  $0 \leq t \leq 6.283$  when  $R = .5$ , when  $R = 2$  and when  $R = 2.5$ . We convert this problem to a first order format using the variables  $y$  and  $y'$  as components of a vector  $y = [x, x']$ . Then  $y' = [y(2), -(R y(2) + y(1))]$  and  $y(0) = [0 \ 1]$ . Our procedure calls for an appropriate  $F$  function which in this case will be a 2-vector. Then we provide the program  $\ll$  ' $Y(2)$ ' EVAL ' $R*Y(2)+Y(1)$ ' EVAL

NEG 2 →ARRY >> as a response to the query in the IN.FN program and the responses to set 0 6.283 for XRNG and -.8 .8 for YRNG in IN.PP. We store the value .5 in the variable R and place the objects 0, [0 1], 6.283 and 1 on the stack and execute G.0I. We change R to each of the numbers 2 and 2.5 and place input quantities 0, [1 0], 6.283 and 1 on the stack and execute G.0I twice more to overlay plots of the other two solutions.

A similar change to G.0I gives the plot program G.12 in which the component Y(1) of the solution is plotted against the component Y(2). The change is made in four places and commands T 'Y(1)' EVAL are changed to 'Y(1)' EVAL 'Y(2)' EVAL and by removing the first << → I and the last >>.

Program Name:	<b>G.12</b>		
Purpose:	Generate a (Y(1), Y(2)) plot of the solution from $T_0$ to $T_f$		
Stored Quantities:	XRNG	YRNG	FN TOL HS
Input:	level 3 $T_0$	level 2 vector $Y_0$	level 1 $T_f$

The output stack is empty, the variables T and Y contain updated values.

```

<< { # 0d # 0d } PVIEW DRAX 3 ROLLD 'Y' STO 'T' STO
→ TF << { T Y FN } TOL HS 'Y(1)' EVAL 'Y(2)' EVAL R→C 4
  ROLLD DO RKFSTEP 'Y(1)' EVAL 'Y(2)' EVAL R→C DUP 6
  ROLLD 5 ROLL LINE DUP T + TF UNTIL > END DROP TF T
  - RKFSTEP 'Y(1)' EVAL 'Y(2)' EVAL R→C DUP 6 ROLLD 5
  ROLL LINE DROP TF T - RKFSTEP 'Y(1)' EVAL 'Y(2)' EVAL
  R→C 5 ROLL LINE 3 DROPN >> PICTURE >>

```

For consistency, from this point we will use notation as follows: for first order differential equations, Y will be the dependent variable and T will be the independent variable. For higher order differential equations, x will be the

dependent variable,  $t$  will be the independent variable and we will reserve  $Y$  as a vector with components which may be constructed from the  $x, x'$ , etc. variables.

**EXERCISE 5.5:** Construct an  $x$  vs  $x'$  plot of the solution of  $x'' + .5 x' + x = 0$ ,  $x(0) = 0$ ,  $x'(0) = 1$ , for  $0 \leq t \leq 6.283$ . As before, for vector  $y = [x, x']$  we have

$$y' = [y(2), -(.5 y(2) + y(1))], \quad y(0) = [0 \ 1].$$


An appropriate F function is given by the program `<< 'Y(2)' EVAL '.5*Y(2)+Y(1)' EVAL NEG 2 →ARRY >>` with the INDEP variable name  $T$ , SOLN name  $Y$  and the INIT vector  $[0 \ 1]$  for SOLN. We choose SOLN(1) for H-VAR and SOLN(2) for V-VAR on the OPTS page. HVIEW should be set at -1 1 and V-VIEW should be set at -1 1. ERASE and DRAW. This approximate solution of the differential equation can be compared to the exact solution by overlaying the parametric curve

$$e^{-.25t} 1.0328 (\sin(.9862t) + i*(.9862 \cos(.9862t) - .25\sin(.9862t))).$$

on the same picture. (Use Parametric type in the PLOT environment.) The user should notice that the plot of the approximate solution consists of a set of points

$$\{(y_1(t_i), y_2(t_i)) : i = 1, 2, \dots\}$$

connected with straight lines. The parametric plot also has this form; however, the points are spaced much closer.

We recommend that the user create a subdirectory for the programs in this section. A possible subdirectory name is DE.1. This subdirectory should contain the programs G.12, G.OI, G1.TY, ER.SE, IN.FN, IN.PP, T, Y, FN, TOL, HS, EQ, and PPAR in this order. The program ER.SE given by `<< ERASE >>` is placed in this directory for convenience. The subdirectory can be created by placing the name 'DE.1' on the stack, then pressing  MEMORY, pressing DIR, then CRDIR. To



obtain the desired order, press { and enter the program names in order, press ENTER, then ORDER (located in the same MEMORY DIR menu).

## 5.2 ELEMENTARY USER PROGRAMS

We present several user programs that are useful in a differential equations course. The students should have some experience with algorithms used to calculate approximate solutions to initial value problems containing differential equations and with programs to implement these algorithms. The simplicity of the programs presented here should help the reader whenever more complicated programs are required for other purposes.

The Euler algorithm for the solution of an initial value problem results from assuming the slope of the solution of a differential equation  $dy/dt = F(t,y)$  is well approximated by the constant  $F(t_k, y_k)$  in the interval  $t_k \leq t \leq t_k + h$  and the algorithm is given by  $t_{k+1} = t_k + h$ ,  $y_{k+1} = y_k + hF(t_k, y_k)$ . (Here  $y_k$  is the approximation of  $y(t_k)$  and it is assumed that initial values  $t_0$  and  $y_0$  and the step size  $h$  are given so the algorithm may be initiated.) Our program is called EULER and takes  $t, y$  from the stack and returns the results of a single step using Euler's algorithm. It will use the step size  $H$ , which is stored, and a stored program F.N that takes  $t,y$  from the stack and returns  $F(t,y)$ .

Program Name:	<b>EULER</b>			
Purpose:	Generate new values of x and y resulting from one step in the Euler algorithm.			
Stored Quantities:	H	F.N		
	Input		Output	
	level 2	level 1	level 2	level 1
	$t_n$	$y_n$	$t_{n+1}$	$y_{n+1}$
	<< DUP2 F.N H * + SWAP H + SWAP >>			

Notice that the structure of F.N is different from the FN program given in the first section of this chapter. F.N requires input from the stack, whereas the programs for FN in section 1 require stored values for  $t$  and  $y$ .

The reader should test this program using the F.N program  $\ll \rightarrow T \ Y \ 'Y' \gg$  for the step size .1 stored in  $H$  and initial conditions  $y(0) = 1$ . (Put 0 1 on the stack and execute EULER EULER EULER, etc.) Note: Here we are solving  $y' = y$ ,  $y(0) = 1$ , using steps  $H = .1$  and obtain the following results:

$t$	$y$	$t$	$y$	$t$	$y$
.1	1.1	.4	1.46	.7	1.95
.2	1.21	.5	1.61	.8	2.14
.3	1.33	.6	1.77	.9	2.36

and  $y$  at  $t = 1.0$  is 2.59, a crude approximation of 2.718....

**EXERCISE 5.6:** To obtain approximate values of the solution of  $y' = \sin(ty)$ ,  $y(0) = 3$ , enter the program F.N given by  $\ll \rightarrow T \ Y \ 'SIN(T*Y)' \gg$ , put initial values 0 3 on the stack and execute EULER, EULER, etc. You should get .1 3, then .2 3.03, then .3 3.09, etc. (Make sure the calculator is in RAD mode.)

Suppose we want to execute EULER, say,  $N$  times and observe the output only at  $t = t_0 + NH$ . The following program, called RPT (for repeat), requires that  $N$  be stored, requires initial values of  $t$  and  $y$  as input, and outputs the final values of  $t$  and  $y$ :  $\ll 1 \ N \ \text{START EULER NEXT} \gg$ .

The Improved Euler algorithm is another method for approximating the solution of an initial value problem. The method results from assuming the slope of the solution is well approximated by the average of  $f(t_k, y_k)$  and a guess at  $f(t_{k+1}, y_{k+1})$  in the interval  $t_k \leq t \leq t_k + h$ . The algorithm is given by

$$t_{k+1} = t_k + h, \quad y_{k+1} = y_k + h[f(t_k, y_k) + f(t_{k+1}, y_k + hf(t_k, y_k))]/2.$$



(Again  $y_k$  is the approximation of  $y(t_k)$  and  $t_0$ ,  $y_0$  and  $h$  are given so the algorithm may be initiated.) This program is named IULER and takes  $t$   $y$  from the stack and gives  $(t+h)$   $(y+h*[f(t,y)+f(t+h,y+h*f(t,y))/2])$ . Note EULER is part of this program.

Program Name:	<b>IULER</b>			
Purpose:	Generate new values of $x$ and $y$ resulting from one step in Improved Euler algorithm.			
Stored Quantities:	H	F.N	EULER	
	Input		Output	
	level 2	level 1	level 2	level 1
	$t_n$	$y_n$	$t_{n+1}$	$y_{n+1}$
<u>Instruction</u>	<u>Resulting stack</u>			
<< DUP2 DUP2 EULER	$t$ $y$ $t+h$ $y+hf(t,y)$			
F.N 3 ROLL	$t$ $y$ $f(t+h, y+hf(t,y))$ $t$ $y$			
F.N + 2 /	$t$ $y$ $(f(t+h, y+hf(t,y))+f(t,y)) / 2$			
H * + SWAP H + SWAP	$[y+h*\{ f(t+h,y+h*f(t,y))+f(t,y)\}/2]$ $t+h$			
>>				

Just as in the EULER program we require that the program F.N and the step size  $H$  be stored before execution. A multiple step program can be obtained by substituting IULER for EULER in the program RPT given just after EXERCISE 2.1.

Try IULER using the F.N program  $\ll \rightarrow T \ Y \ 'Y' \ \gg$ ,  $H = .1$  and initial data 0 1. Execute 9 times. You should get 1 2.7140808---. (Euler gives about 2.593742---, not nearly so good an approximation of  $e = 2.71828$ ---.) In general, the improved Euler method can be shown to be a better approximation when  $h$  is small.

How is an appropriate value of  $h$  chosen? If it is decided to use a constant step size throughout the interval of interest  $[t_0, x_f]$ , one common way to select  $h$  is to try a nominal size of  $h$ , say  $(t_f - t_0)/50$ , and calculate the solution approximate  $y_f$  at  $t_f$ . Then reduce  $h$  by half and recalculate the approximate at  $t_f$ . If the values agree to your satisfaction (for example, to three decimal places), use the last set of values obtained; if not, reduce  $h$  by half and try again.

This is a good time to check on the accuracy of the built-in differential equation algorithm used by the HP-48G calculator. Press  SOLVE, use the  arrow key to select Solve Diff eq..., press OK, enter the F function Y, set the range of the independent variable to 0 1 and set the initial value of the solution to 1. Move the cursor to FINAL and press SOLVE. Press the ON key and you should see the value 2.718... on the stack. Put 1 on the stack, press the  $e^x$  key and subtract to see the apparent error -.000019... . This error was achieved with the default tolerance .0001. The performance of the differential equation algorithm depends on the problem and is not always this good.

A comment on the built-in algorithm on the HP-48G calculator for solving differential equations is in order at this point. There is a default program based on the well known Runge-Kutta Feldberg algorithm which automatically selects step size to keep the perceived error below the specified tolerance. There is also a second built-in calculator program for solving stiff differential equations that we will discuss briefly later.

**EXERCISE 5.7:** Try EULER on the problem  $y' = (y^2 + y)/t$ ,  $y(1) = 1$  with  $h = .2$ . Execute 5 times, then reduce  $h$  to .1 and execute 10 times. Next execute from the initial value 20 times with  $H = .05$ . What is being indicated? Hint: this problem can be solved exactly and has an asymptote at  $t = 2$ . Here F.N could be given by

$$<< \rightarrow T \ Y \ ' (Y^2+Y)/T \ ' >>$$

**EXERCISE 5.7a:** Try the HP-48G calculator's Solve diff eq... algorithm for the F function and initial condition given in EXERCISE 1.7 for the value of the solution at  $t = 2$ . Change the tolerance TOL to .1 and try to SOLVE for FINAL. The calculator will take over 10 seconds and returns a value of 1743.5... . If you change the value of TOL to .05 and resolve for FINAL, the calculator will take over 20 seconds and returns a value of 2187.8... . The long execution time tells us that the calculator is struggling to achieve good results and in this case can not achieve accuracy for good reason.

Programs to obtain graphical output are easy on the HP-48. The following program, which we will call GRAF, requires  $t_0, y_0$  from the stack and uses IULER (or EULER) to advance N steps of size H. (N is also stored.) The user should pre-enter the numbers  $t_{\min} \ t_{\max}$  as XRNG and numbers  $y_{\min} \ y_{\max}$  as YRNG for the graph.

Program Name:	<b>GRAF</b> for scalar equation			
Purpose:	Plot N values of (x,y) obtained using Euler algorithm			
Stored quantities:	N, H, F.N, IULER XRNG YRNG			
	Input		Output	
	level 2	level 1	level 2	level 1
	$t_0$	$y_0$	$t_N$	$y_N$
	and graph with cursor			
	<< { # 0d # 0d } PVIEW DRAX DUP2 R→C 3 ROLLD 1 N			
	START IULER DUP2 R→C DUP 5 ROLLD 4 ROLL LINE NEXT			
	PICTURE >>			

GRAF contains a loop in which  $N$  new points  $(t,y)$  are calculated and plotted. You may want to ERASE the graphics screen before executing the program. The program EULER may be inserted in place of IULER so that GRAF uses whichever algorithm is desired. Notice also that the last values of  $t$  and  $y$  remain on the stack after GRAF is executed. To restore the stack screen, press ON.

As a footnote to this section, the following program can be used to remind the user for the ingredients required for GRAF. As written, the user must enter an expression for  $F(T, Y)$  (for example 'SIN(T\*Y)') which will be stored by the program as

`<< → T Y 'SIN(T*Y)' >>` in the variable F.N. The program will also prompt for initial conditions, step size, number of steps, etc.

Program Name: **INIT1**

Initialization Program to set required ingredients for GRAF

```
<< "ENTER F.N IN T,Y" " " INPUT OBJ→ 'F.N(T,Y)' SWAP =
DEFINE "KEY IN # OF STEPS" " " INPUT OBJ→ 'N' STO "KEY
IN STEP SIZE" " " INPUT OBJ→ 'H' STO "KEY IN XRNG" " "
INPUT OBJ→ XRNG "KEY IN YRNG" " " INPUT OBJ→
YRNG "KEY IN INITIAL T" " " INPUT OBJ→ "KEY IN INITIAL
Y" " " INPUT OBJ→ ERASE >>
```

As we have already indicated, it is often desirable to plot solutions of several initial value problems on the same plot. Of course, plots can be combined simply by not erasing the previous result.

**EXERCISE 5.8:** Consider the following differential equation together with several initial conditions and plot the solutions on the same graph.

$$dy/dt = y(1-y), \quad y(0) = .2, .4, .6, 1.5$$

where the solutions are plotted for  $0 \leq t \leq 5$  and step size  $h = .05$  is used. Try

F.N: `<< → T Y ' Y*(1-Y) >>.`

Put 0 and .2 on the stack, then execute **GRAF**. (Remember  $H = .05$  and  $N = 100$  are stored before execution.) Place another initial condition on the stack and add the second solution graph. Notice the solution  $y = 1$  is an attracting solution, i. e., nearby solutions collapse to  $y = 1$  as time increases.

The **EULER** and **IULER** programs also work for the vector case when the F.N program has the proper form and when the initial  $y$  input is a vector. Here again F.N requires input  $T \ Y$ . We can modify the **GRAF** program to the following form:

Program Name:	<b>G.TYI</b>				
Purpose:	Plot N values of (T,Y(I)) resulting from the improved Euler algorithm which creates a sequence of N values of t, and Y.				
Stored Quantities:	N H F.N EULER IULER XRNG YRNG				
	Input			Output	
	level 2	level 1	level 3	level 2	level 1
	$t_0$	initial vector Y	I	$t_n$	final vector Y & graph
	<code>&lt;&lt; → I &lt;&lt; { # 0d # 0d } PVIEW DRAX DUP2 I GET R→C 3</code>				
	<code>ROLLD 1 N START IULER DUP2 I GET R→C DUP 5 ROLL</code>				
	<code>4 ROLL LINE NEXT GRAPH &gt;&gt; &gt;&gt;</code>				

Program **INIT.I** is to set the plotting parameters for **G.TYI**. Notice that the construction of the F.N program is to be done later since it is felt that F.N could be a complicated program.

Program Name: **INIT.I**

Purpose: Set parameters for G.TYI.

```
<< "KEY IN # OF STEPS" "" INPUT OBJ→ 'N' STO
    "KEY IN STEP SIZE" "" INPUT OBJ→ 'H' STO
    "KEY IN XRNG" "" INPUT OBJ→ XRNG
    "KEY IN YRNG" "" INPUT OBJ→ YRNG
    "KEY IN INITIAL T" "" INPUT OBJ→
    "KEY IN INITIAL Y" "" INPUT OBJ→
    "KEY I FOR Y(I) GRAPH" "" INPUT OBJ→
    "PRESS ENTER, CONSTRUCT PROGRAM F.N
    EXECUTE G.TYI" "" INPUT OBJ→ >>
```

The program G.TYI as given above works for vector initial value problems with two or more components. For example consider the following

**EXERCISE 5.9:** Set F.N to be

```
<< → T Y << ' -.00001*Y(1)*Y(2)' EVAL '.00001*Y(1)*Y(2) - Y(2)/14' EVAL
    'Y(2)/14' EVAL 3 →ARRY >> >>
```

set initial T, Y to 0 [45400 2100 2400] with XRNG 0 25 YRNG 0 45400 and set the number of steps to be  $N = 25$  and step size to  $H = 1$ . Obtain a T-Y(1) plot, overlay a T-Y(2) plot, etc.

Students who use the Euler and improved Euler algorithm will also need a program to compute the solution and plot the components  $y_1(t)$  versus  $y_2(t)$  as  $t$  increases. Such a program, call G.Y12 is presented below. The initialization program INIT.I also works for this program, except the input I is not needed and should be deleted before execution of G.Y12.




Program Name:	<b>G.Y12</b>			
Purpose:	Plot N values of (Y(1),Y(2)) resulting from the improved Euler algorithm which creates a sequence of N values of Y(1) and Y(2).			
Stored Quantities:	N H F.N EULER IULER XRNG YRNG			
	Input		Output	
	level 2	level 1	level 2	level 1
	t <sub>0</sub>	initial vector Y	t <sub>n</sub>	last vector Y & graph

```

<< { # 0d # 0d } PVIEW DRAX DUP DUP 1 GET SWAP 2
GET R→C 3 ROLLD 1 N START IULER DUP DUP 1 GET
SWAP 2 GET R→C DUP 5 ROLLD 4 ROLL LINE NEXT
GRAPH >>

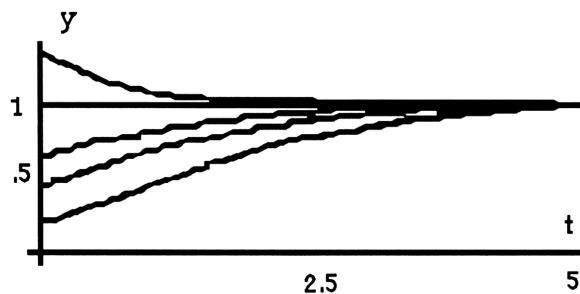
```

We recommend that the user create a subdirectory for the programs in this section. A possible subdirectory name is DE.2. This subdirectory should contain the programs INIT1, GRAF, INIT.I, G.TYI, ER.SE, G.Y12, T, Y, F.N, EQ, N, H, IULER, EULER and PPAR in this order. The program ER.SE given by << ERASE >> is placed in this directory for convenience. The subdirectory can be created by placing the name 'DE.2' on the stack, then pressing  MEMORY, pressing DIR, then CRDIR. To obtain the desired order, press { and enter the program names in order, press ENTER, then ORDER (located in the same MEMORY DIR menu).

# 6

## FIRST ORDER DIFFERENTIAL EQUATIONS

Now that we can construct approximate solutions of a differential equation we can suggest exercises and activities that use these graphical and numerical computations to enhance the study of differential equations.



Five solutions of  $dy/dt = y(1-y)$

We will often be interested in constructing graphs of several solutions of a differential equation. The figure shown above, constructed for the differential equation  $y' = y(1-y)$  is an example. There may be solutions  $y(t)$  that remain constant as time increases. Such solutions are called *equilibrium solutions*. In the case just mentioned, the constant solutions are  $y(t) = 0$  and  $y(t) = 1$ . Clearly the solutions  $y(t) = y_e$  of  $dy/dt = F(t, y)$ , which are constant, satisfy

$$F(t, y_e) = 0.$$

If we wish to understand how solutions of a differential equation change as the initial condition  $y(0)$  is varied, one of the first tasks is to find the equilibrium solutions. If the function  $F(t, y)$  is continuous and has continuous derivatives then

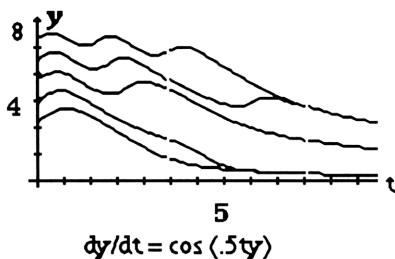
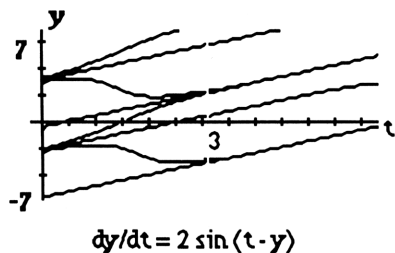
any solutions  $y_1(t)$  and  $y_2(t)$  which satisfy different initial conditions do not intersect. Consequently, constant solutions restrict the region where nearby solutions can proceed.

**EXERCISE 6.1:** Plot the solutions of the three initial value problems  $dy/dt = y^2(1 - y^2)$ , that satisfy either  $y(0) = -1$ ,  $y(0) = 0$ , and  $y(0) = 1$  for  $0 \leq t \leq 5$  all in the same picture. Then overlay plots of the solutions of the same differential equation that satisfy  $y(0) = -.25$  and  $y(0) = .25$ .

**EXERCISE 6.2:** Consider the differential equation  $y' = y - y^3$ . The equilibrium solutions are 0, -1 and 1. For an initial condition  $y(0)$  not in the set  $\{-1, 0, 1\}$ , use the separation of variables technique to obtain the result

$$y^2|1-y| = |1+y|Ke^{2t}, \quad K = \frac{y_0^2|1-y_0|}{|1+y_0|}.$$

This equation defines the solution  $y$  implicitly as a function of  $t$ . For example, if  $0 < y_0 < 1$ , then since  $y(t)$  will not leave the interval  $0 < y < 1$ , we have a cubic equation for  $y$  as a function of  $t$ . We can use one of our calculator programs to construct a plot of an approximate solution of the problem with such an initial condition or if we only want a crude idea of the solution graph, we can sketch in an increasing function proceeding from  $y(0)$  up toward the asymptotic value  $y(\infty) = 1$ . Why?



The solutions of the two differential equations pictured above show interesting structure. The straight lines  $y = t + a$  are solutions of  $dy/dt = 2 \sin(t - y)$  for  $a = 3.665$ ,  $a = -.524$ ,  $a = -2.618$ , or  $a = -6.81$ . (Hint: make the transformation  $t - y = w$  to get the differential equation  $w' = 1 - 2\sin w$ . What are the equilibrium solutions of the  $w$  differential equation?) Solutions starting near  $t = 0$ ,  $y = -.524$  collapse to the straight line solution  $y = t - .524$ , while solutions starting near  $t = 0$ ,  $y = -2.618$  are repelled away from the straight line solution  $y = t - 2.618$ , etc.

As for the second plot shown just above, even though the functions  $y = (2n+1)\pi/t$  ( $n = 0, \pm 1, \pm 2, \dots$ ) are not solutions of  $dy/dx = \cos(.5ty)$ , when  $t$  is large such a function  $y$  has small derivative and we can see these approximate solutions emerge for large  $t$ . Moreover when the initial value  $y(0)$  is large, there are more values of  $t$  on the graph when  $.5ty(t) = (4n+1)\pi/2$  and the slope  $y'$  is zero.

**EXERCISE 6.3:** Plot the solutions starting from  $y(0) = -7.85$ ,  $y(0) = -1.57$ ,  $y(0) = 4.71$ ,  $y(0) = -1.9$ ,  $y(0) = -2.5$ ,  $y(0) = 2.5$  and  $y(0) = 4.3$  that satisfy the differential equation  $dy/dt = \sin(t - y)$  for  $0 \leq t \leq 8$ . Use vertical dimension to show  $-8 \leq y \leq 8$ . *Hint:* the transformation  $w = t - y$  gives a differential equation with equilibrium solutions  $w_e = \pi/2, 5\pi/2, -3\pi/2$ , etc.

**EXERCISE 6.4:** Plot the graph of the differential equation  $dy/dt = \sin(ty)$  with initial condition  $y(0) = 3$  with plot parameters to show  $0 \leq t \leq 6$ ,  $0 \leq y \leq 5$ . Select a new starting point  $y(0) = 3.5$  and add the new trajectory. Now choose  $y(0) = 1.5$ , get the new combination graph, then choose  $y(0) = 1$  and get another combination graph: we see the bottom two trajectories approach each other.

The graphical study of solutions of  $dy/dt = \sin(ty)$  led to an journal article that gives mathematical proofs for some of the interesting behavior observed in the graphs. See Mills, B. Weisfeiler and A. Krall, "Discovering Theorems with a Computer", *The American Mathematical Monthly*, volume 86 (1979), pages 733-739.

**EXERCISE 6.5:** Consider the differential equation  $y' = y - .3t - (y - .3t)^3$ . The transformation  $w = y - .3t$  gives the new differential equation  $w' = w - w^3 - .3$ . What are the equilibrium solutions of the new differential equation? Sketch several solutions of the  $w$  differential equation including the equilibrium solutions, a solution with  $w(0)$  above the largest equilibrium solution, one with  $w(0)$  below all of the equilibrium solutions and one with  $w(0)$  near but not on the middle equilibrium solution. Now make a sketch of the corresponding solutions of the  $y$  differential equation.

**EXERCISE 6.6:** Repeat as much as possible of the previous EXERCISE for the differential equation  $y' = y - .5t - (y - .5t)^3$ .

We will wish to compare the graphs of solutions of different differential equations. For example if we graph the solutions of the two initial value problems  $dy/dt = y(1 - y)$ ,  $y(0) = .25$  and  $dy/dt = y^2(1 - y^2)$ ,  $y(0) = .25$  (graphic screen parameters  $0 \leq t \leq 5$  and  $0 \leq y \leq 1.2$ ) on the same plot, we notice that the solutions are structurally similar. In which case is a change of concavity apparent?

**EXERCISE 6.7:** Plot the solutions of the two initial value problems  $dy/dt = y(1 - y)$ ,  $y(0) = .25$  and  $dy/dt = -y \ln y$ ,  $y(0) = .25$  (plot screen parameters  $0 \leq t \leq 5$  and  $0 \leq y \leq 1.2$ ) on the same plot. In what ways are the solution graphs similar? In what way are they different?

There are many problems in a differential equations course in which a number (or numbers) satisfying a somewhat complicated equation is needed. In one type of example we can use the graphing capability of the calculator to display the inverse of a particular function and thus graph a desired solution. We will give an example of this below. In a second type of problem we may simply use the equation solver routine contained in the calculator. An example of this type of problem is also given below.

Implicitly defined solutions may arise in the study of first order differential equations, particularly in those problems in which variables are "separated and integrated" or in exact equations.

**EXAMPLE:** Plot the solution of

$$dx/dt = 1 - x^{3/2}, \quad x(0) = 1/2$$

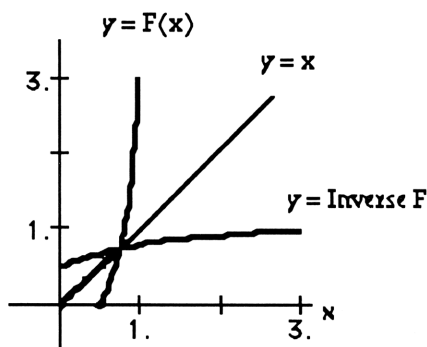
for  $0 \leq t \leq 2.5$ . Clearly the solution  $x(t)$  will approach 1 as  $t$  increases. Using separation of variables, we obtain

$$\int \frac{dx}{1 - x^{3/2}} = t + C.$$

We make the substitution  $x = y^2$  and use a partial fraction decomposition for the fraction to obtain the implicit equation  $F(x) = t$  where  $F(x) = f(x) - f(.5)$  and

$$1.5 f(x) = \ln \left\{ \frac{\sqrt{(1 + x + \sqrt{x})}}{1 - \sqrt{x}} \right\} - \sqrt{3} \operatorname{Arctan} \left\{ \frac{1 + 2\sqrt{x}}{\sqrt{3}} \right\}.$$

We enter the formula for  $F(x)$  in the calculator and notice the range of  $F$  for  $.5 \leq x \leq .99$  is  $[0, 2.79]$ . We specify plotting parameters so that  $XRNG$  and  $YRNG$  are  $0 \ 3$  and restrict the values of  $x$  to be graphed by entering  $\{X \ .5 \ .99\}$  as  $INDEP$  and draw a plot of  $F(x)$ . A plot of the inverse function can be overdrawn by altering  $EQ$  to ' $F(X) + i * X$ ' and changing the plot type to  $PARAM$ . Finally the graph of  $y = x$  is also shown as part of the construction of  $F^{-1}$  from  $F$ .



**EXERCISE 6.8:** Determine the solution of  $x' = 1 - x^r$ ,  $x(0) = .5$  using separation of variables technique for  $r = 5/4$  and for  $r = 5/2$ . Then use the inverse function to plot  $x(t)$ . Hint:  $(x^2 + .5(1 - \sqrt{5})x + 1)(x^2 + .5(1 + \sqrt{5})x + 1) = (x^4 + x^3 + x^2 + x + 1)$ .

Suppose we wish a number  $x$  so that an equation  $f(x) = g(x)$  is satisfied. Construct a list which contains expressions for  $f(x)$  and for  $g(x)$ . Then store this list in the variable  $EQ$ . Set plot parameters so that when both sides of the equation are drawn, a crossing is shown. Use the cursor to locate the approximate crossing coordinates and the  $ISECT$  command to obtain the result.

**MIXING PROBLEM:** Initially a large tank holds 2000 gallons of pure water. An stream of 5 gallons per minute with salt content of 2 #/gallon is input into the tank and 4 gallons per minute of the well mixed solution is drained from the tank. When is there  $Q_0$  pounds present in tank? The usual model  $dQ/dt = \text{input rate} - \text{output rate}$  gives

$$Q = 2 \left[ 2000 + t - \frac{(2000)^5}{(2000 + t)^4} \right].$$

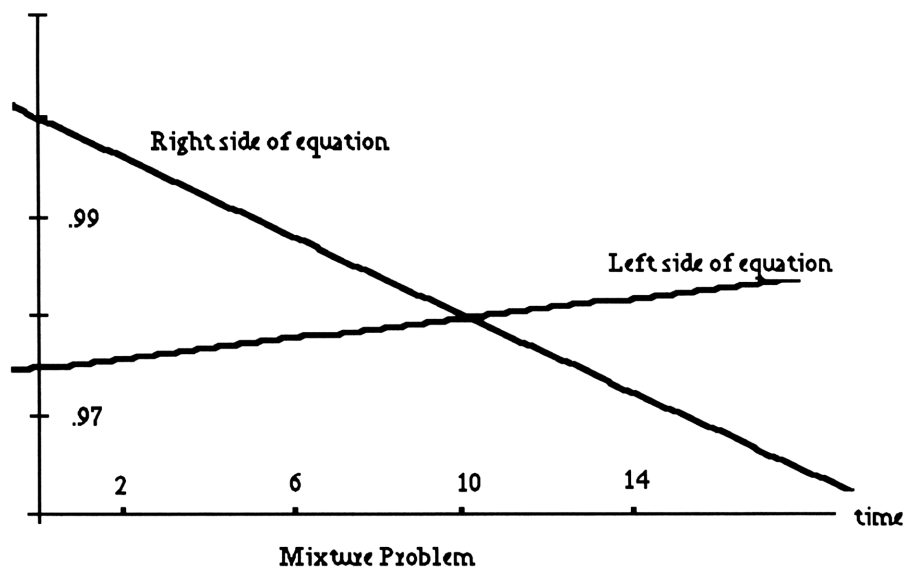
Putting  $Q(t) = Q_0$  gives

$$\frac{2000 - \frac{Q_0}{2} + t}{2000} = \left[ \frac{2000}{2000 + t} \right]^4$$

to solve for  $t$ . For  $Q_0 = 100$ , we get

$$\frac{1950 + t}{2000} = \left[ \frac{2000}{2000 + t} \right]^4;$$

and if we use plotting parameters to show  $0 \leq x \leq 20$ ,  $.9 \leq y \leq 1$ , we get an intersection at about 10 as shown:





**EXERCISE 6.9:** A tank initially contains 300 gallons of pure water. Brine containing 1.5# of salt per gallon enters the tank at 2 gallons/minute and the well mixed solution leaves at 3 gallons per minute. When will the tank contain 21 # of salt? (There may be more than one solution.)

## 6.1 POPULATION PROBLEMS

EXERCISE 2.7 gives two initial value problems that model population growth in a food limited environment. Which model is appropriate? Some input from biologists or some observation data could be used to answer this question. Suppose that from experimental data, we can determine the limiting value of the population and that we can also estimate at what fraction of the limiting value of  $y$  an inflection point occurs. In EXERCISE 2.7, the inflection points occur at 36.8% (for the logarithm model) and 50% (for the quadratic model) of the limiting value of  $y$ , which in both cases is  $y = 1$ . We will further explore this question below.

Suppose we are given the assignment of explaining how the population of a species evolves in time and we note that the environment will only support a finite number of the population. Two much studied models of this type are:

- The logistic model:

$$\frac{dp}{dt} = ap - bp^2, \quad p(0) = p_0 : \quad p = \frac{ap_0}{bp_0 + (a - bp_0)e^{-at}}$$

- The Gompertz model:

$$\frac{dp}{dt} = p(A - B \ln p), \quad p(0) = p_0 : \quad p = e^{A/B} \left[ \frac{p_0}{e^{A/B}} \right]^{\exp(-Bt)}$$

The parameters have different meanings: equating the carrying capacity of the model (i. e., the value of the population that is reached in infinite time) gives  $e^{A/B}$

in the Gompertz model and  $a/b$  in the logistic model. Which of these models is better? Or should we look for another model?

These are not easy questions in general. Probably the first step is to pick a model, use data to determine what the model parameters should be (e.g. the constants  $a$ ,  $b$  or  $A$ ,  $B$ ) and graph the solution. Then change the model, use data to determine that model's parameters and graph the solution again, etc.

**POPULATION PROBLEM:** Suppose we use a logistic population model  $dp/dt = ap - bp^2$  with parameters  $a$ ,  $b$  and data taken from the following table:

<i>Year</i>	<i>Population</i>	<i>Year</i>	<i>Population</i>
1790	3.93	1900	75.99
1800	5.31	1910	91.97
1810	7.24	1920	105.71
1820	9.64	1930	122.78
1830	12.87	1940	131.67
1840	17.07	1950	151.33
1850	23.19	1960	179.32
1860	31.44	1970	203.21
1870	39.83	1980	226.50
1880	50.16	1990	248.71
1890	62.95		

To determine the parameters  $a$  and  $b$ : if we use  $p_0 = 3.93$  and  $p(50) = 17.07$  we get

$$bp_0 = \frac{a \left( \frac{3.93}{17.07} - e^{-50a} \right)}{1 - e^{-50a}} ;$$

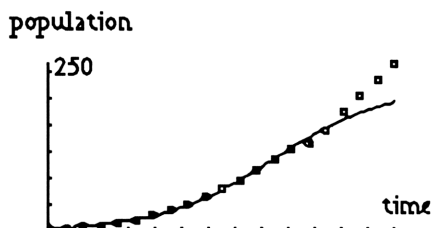
and if we also take  $p = 75.99$  at  $t = 110$ , we have

$$\left[ \frac{3.93}{17.07} - e^{-50} \right] (1 - e^{-110a}) = \left[ \frac{3.93}{75.99} - e^{-110a} \right] (1 - e^{-50}) .$$

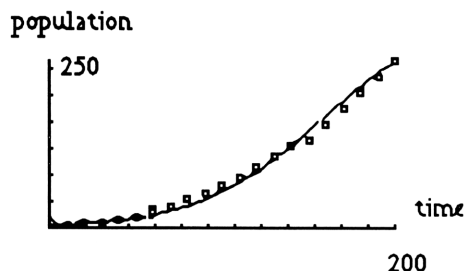
By setting  $x = 50a$  (which gives  $110a = 2.2x$ ) we obtain the equation

$$[e^{-x} - .23023] (1 - e^{-2.2x}) = [e^{-2.2x} - .05172] (1 - e^{-x}) .$$

For plotting parameters  $1 \leq x \leq 2$ ,  $-0.05 \leq y \leq .02$ , we observe a solution at  $x = 1.53$ , which means that  $a = .031$  and  $b = .00014$ . We show below the data and the solution curve for these values of the parameters, and also the solution curve for the values of  $a$  and  $b$  obtained by fitting to data at time  $t = 140$  and  $t = 200$  ( $a = .0279$  and  $b = .0000855$ ).



Data and Logistic Curve for fit  
to  $p(50)=17.07, p(110)=75.99$



Data and Logistic Curve for fit  
to  $p(140)=122.78, p(200)=248.71$

*Note:* A program D.GRF to produce a graph of data with "fat pixels" will be suggested. A list of the data coordinate pairs is stored in a variable L1. After XRNG and YRNG are set and the screen is erased the following program will plot the data points:

```
PGM D.GRF    << PPAR DUP 2 GET SWAP 1 GET - C→R
              64 / SWAP 132 / MIN 1.2 * → RADIUS
<< DRAX L1 OBJ→ 1 SWAP START RADIUS 0 6.28 ARC NEXT >> >>.
```

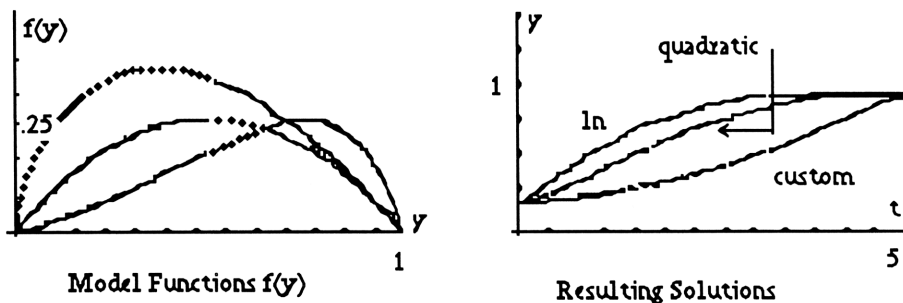
**EXERCISE 6.10.** Suppose again that  $p_0$ ,  $(t_i, p_i)$ , and  $(t_k, p_k)$  are known. Determine the constants  $A$  and  $B$  in the Gompertz model. (Hint: put  $s = A/B$  and solve for  $e^{-Bt}$  in the expression for the solution, then for  $B$ . Then evaluate the expression at each time and set them equal.) Find the value of  $A$  and  $B$  for  $p_0 = 1$ ,  $(t_i, p_i) = (1, 1.46)$  and  $(t_k, p_k) = (2, 1.5)$ .

How might other models be constructed? Here is a suggestion if data  $\{(t_1, p_1), (t_2, p_2), \dots, (t_n, p_n)\}$  is given and a graph of the data indicates the location of an inflection point and the carrying capacity  $K$ . Population models may have the form  $dp/dt = f(p)$  with  $f(0) = 0$ ,  $f(K) = 0$  for  $K > 0$ , and  $f(p) > 0$  for  $0 < p < K$ . Notice that inflection points come at those points  $p$  with  $f'(p)f(p) = 0$ . Since  $f(p) > 0$ , we get inflection points when  $f'(p) = 0$ . In the logistic model this occurs when  $p = .5 a/b$  and in the Gompertz model when  $\ln p = A/B - 1$ .

To get a model with inflection at  $p = .6K$ , we could try  $p' = f(p) = (.6K)^2 - (p - .6K)^2$  for  $0 < p \leq .6K$ , and also  $p' = f(p) = 2.25 ( (.4K)^2 - (p - .6K)^2 )$  for  $.6K < p \leq K$ .

**EXERCISE 6.11.** Use your calculator to obtain a plot of the solution of this model for  $K = 1$ ,  $p(0) = .2$ ,  $XRNG = -.2 \quad 5$ ,  $YRNG = -.1 \quad 1.1$ . We will suggest here a method to enter an appropriate  $F$  function using the HP-48G input form format. Press **PLOT**, **CHOOS**, **Diff Eq**, press **OK**, then position the highlighted field to  $F$ . Press **NXT**, press **CALC**, and place the following on the stack: `'IFTE(Y<.6, .36 - (Y-.6)^2, 2.25*(.16- (Y-.6)^2))'` Note: The command **IFTE** can be located by pressing **PRG BRCH NXT**. The **<** command is located by pressing **PRG TEST**. When this step is complete, press the **ON (CONT)** key, then you should see the desired stack entry and the **ON** key. Press **ON**. The student should complete the EXERCISE from this point. If you wish to invoke the user program, an appropriate **FN** program might be `<< 'IFTE(Y<.6, .36 - (Y-.6)^2, 2.25*(.16- (Y-.6)^2))' EVAL >>`.

**EXERCISE 6.12.** Suppose we have the following (time, population) data point measurements  $\{(0, .2), (.5, .37), (1, .61), (1.5, .88), (2, .98), (2.5, 1), (3, 1)\}$ . Use your calculator to plot the data and estimate the location of the inflection point. Then construct a model that will give an inflection point at this value and overlay the solution of the model with the data for comparison.



Logistic, Gompertz and Custom Models (exercise 2.8)

**EXERCISE 6.13:** Set plot parameters to show  $-0.5 \leq t \leq 12.56$ ,  $-0.5 \leq y \leq 4$ . Graph the solutions of  $y' = .5y(\exp(\sin t) - y)$  with  $y(0) = 1$ , and  $y(0) = 3$ . What initial condition gives periodicity? (This differential equation is a potential model for an environment where the birth rate is periodic in time.)

## 6.2 MOTION OF A PARTICLE IN ONE DIMENSION

Mathematical models for the velocity of a particle falling from rest under gravity with air resistance have the form

$$\frac{dv}{dt} = g - f(v), \quad v(0) = 0,$$

where the force exerted on the particle by the resistive medium,  $f(v)$ , is determined by experimental means. We will assume that  $f$  is an increasing function with

$f(0) = 0$ . The velocity will increase toward a terminal value which is given by  $f^{-1}(g)$ .

**EXERCISE 6.14:** For simplicity we take physical units so that  $g = 2$ . We want to compare the trajectories from different models in which the  $f(v)$  functions are given by:

(a)  $f(v) = v$

(b)  $f(v) = .5 v^2$

(c)  $f(v) = \text{IFTE}(v \leq 1, (1.5)^{.5} v, (2.5 v - 1)^{.5})$

(d)  $f(v) = \text{IFTE}(v \leq 1, .75 v^{1.5}, 1.25 v - .5)$

Notice that these models have been chosen so that all have terminal velocity 2. Use the calculator's function **DRAW** program to plot each  $f(v)$  function for  $0 \leq v \leq 2$ . Use **XRNG** = -.1 2 and **YRNG** = -.1 2.1. Accumulate these graphs on the same picture and label the graphs. Then use a differential equation plotting program to graph the solution of the initial value problem given above for each  $f(v)$  function for  $0 \leq t \leq 5$ . Accumulate them in the same picture for comparison. Again label the solutions. Use the **YRNG** as above and **XRNG** -.2 5.

A particle falls or is projected from a great height and observations are made on  $v$  for,  $n$  values of time. Two well-known models for such a problem are:

- linear air resistance model:  $dv/dt = g - kv$ ,  $v(0) = v_0$ . The solution is

$$v(t) = v_0 e^{-kt} + v_\infty (1 - e^{-kt}), \quad v_\infty = g/k.$$

- quadratic air resistance model:  $dv/dt = g - kv^2$ ,  $v(0) = v_0$ . The solution is

$$v(t) = v_{\infty} \frac{Me^{\sigma t} - 1}{Me^{\sigma t} + 1}, \quad M = \frac{v_{\infty} + v_0}{v_{\infty} - v_0}, \quad v_{\infty} = \sqrt{\frac{g}{k}}, \quad \sigma = 2\sqrt{gk}.$$

Suppose that  $v_{\infty}$  can be accurately determined from data, say  $\{(t_1, v_1), (t_2, v_2), \dots, (t_n, v_n)\}$ . In the case of the linear model  $k = g/v_{\infty}$  and we note that the graph of

$$z(t) = \ln(v_{\infty} - v(t)) = \ln(v_{\infty} - v_0) - kt$$

is a straight line with slope  $-g/v_{\infty}$ . Furthermore, in the case of the quadratic model,  $k = g/(v_{\infty})^2$  and the graph of

$$z(t) = \ln(v_{\infty} - v(t)) \approx \ln(2v_{\infty}/M) - (2g/v_{\infty})t$$

is a straight line with slope  $-2g/v_{\infty}$ . This is twice the slope of the linear model.

Suppose that (time, velocity) data is available. What model is appropriate? Maybe if we plot  $t_i$  vs  $\ln(v_{\infty} - v_i)$  a straight line will appear for large  $t$ , and we can choose a model with the appropriate slope.

**EXERCISE 6.15.** The data to be used for model selection is:

$$\{(0, 0), (1, 1.44), (2, 1.87), (3, 1.97), (4, 1.99), (5, 2)\}.$$

Consider models of the form  $dv/dt = 2 - av^r$ ,  $v(0) = 0$ , where  $r$  is a positive number and  $a$  is chosen so that  $v_{\infty} = 2$ . (In our units  $g = 2$ .) Plot the points  $(t_i, \ln(2 - v_i))$ . Determine the slope of a line which "fits" the plot for the latter data points and choose an appropriate value of  $r$ . Then plot the data points (no logarithms) and use a differential equation calculator graphing program to draw the trajectory of the model you have chosen as an overlay of the data point graph. Conclusions? (It is instructive to experiment with models of the form  $dv/dt = g - av^r$  and G1.TY can be modified to plot the  $\log(v_{\infty} - v(t))$  by inserting the commands `V∞ - ABS LN` in 4 locations each following the command `Y.`)

Consider the problem

$$M'(t)v + M(t)v' = T(t) - M(t)g - \sigma v, \quad v(0) = x(0) = 0$$

where  $M(t)$  is the mass of a rocket and has equation

$$M(t) = m - \alpha t \text{ for } 0 \leq t \leq t_0 \text{ and } M(t) = m_0 + \{m - \alpha t_0 - m_0\}e^{-\gamma(t-t_0)} \text{ for } t_0 < t$$

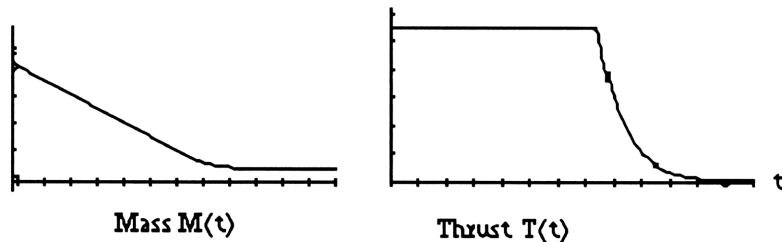
with  $\gamma = \alpha / (m - \alpha t_0 - m_0)$  (so that  $M'(t)$  is continuous) and  $T(t) = -\beta \, dM(t)/dt$ .

As an example we take  $m = 1$ ,  $\alpha = .19$ ,  $m_0 = .2$ ,  $t_0 = 4$ ,  $g = 1$ ,  $\sigma = .05$  and  $\beta = 22$  so

$$M(t) = 1 - .19t \text{ for } 0 \leq t \leq 4, \quad M(t) = .2 + .04e^{-4.75(t-4)} \text{ for } 4 \leq t, \text{ and}$$

$$T(t) = -22 \, dM(t)/dt.$$

The graphs are shown below.

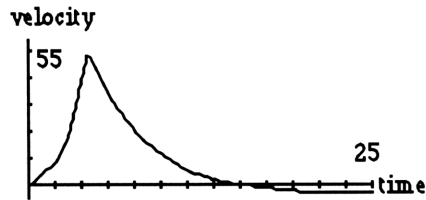


We notice that for these values of the parameters, after the thrust dissipates from mass burnoff, we have  $M_\infty = .2$  and the terminal velocity will be  $v_\infty = -4$ . Store the formulas for  $M(t)$  and  $M'(t)$  in user defined functions **MAS** and **MDOT**, and set  $F(T,Y)$  to be

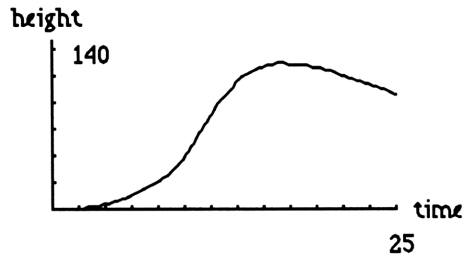
```
<< ' MAS(T) + (22+Y)*MDOT(T)+.05*Y' EVAL 'MAS(T)' EVAL / NEG >>.
```



Setting H-VIEW to 0 25 and V-VIEW to -5 55 respectively, gives the velocity graph:



The velocity rises to 53 begins to decrease at  $t = 4.25$  but remains positive until about  $t = 15$ . The velocity at  $t = 25$  is  $v = -3.65$ . The graph of height versus time is shown below. Recall that we have used specialized units (e.g.,  $g = 1$ ), so the actual height is not in a common physical unit.



**Rocket height versus time**

**EXERCISE 6.16 :** We may wish to study the sensitivity of the results we have to the parameter values used. Construct the velocity versus time graph shown above for the parameters as given, then change the parameter values of  $\alpha$  to  $\alpha = .16$ ,  $t_0$  to  $t_0 = 3.8$  and  $\beta$  to  $\beta = 18$ . Overlay the new velocity time graph on the first graph.

### 6.3 INPUT OUTPUT PROBLEMS

Suppose a tank, which contains  $V$  volume units of a mixture of water and a chemical substance, receives  $f(t)$  units (weight) of the chemical in solution per minute. The chemical is vigorously mixed in the tank and the mixture drains from the tank in such a way that constant volume in the tank is maintained. If  $y(t)$  is the weight of chemical in the tank at time  $t$ , a balance equation gives  $dy/dt$  as the rate that the chemical enters the tank minus the rate that the chemical exits from the tank. This is one example of an important problem, namely, to determine a particular solution of the equation

$$\frac{dy}{dt} + ry = f(t).$$

Here we assume  $r$  is a positive constant. Commonly, the function  $f(t)$  is called *input* to the problem and the solution  $y(t)$  is called the *output*. Other examples of this problem occur in electrical flow problems. The initial value problem solution is

$$y(t) = y(0) e^{-rt} + \int_0^t e^{-r(t-s)} f(s) ds.$$

If the function  $f(t)$  is periodic with period  $P$ , then we can choose  $y(0)$  so that the output is periodic. This is done by choosing  $y(0)$  so that  $y(0) = y(P)$ , which gives

$$y(0) = \frac{1}{1 - e^{-rP}} \int_0^P e^{-r(P-s)} f(s) ds.$$

The input function  $f$  is transformed to the output function  $y = Tf$ . Notice  $T(af + bg) = aTf + bTg$  when  $a$  and  $b$  are constants and  $f, g$  are input functions. This superposition property of the "operation"  $T$  shows the transformation  $T$  to be a *linear operator*.

Here we are interested in comparing the graphs of the input functions  $f$  to the graphs of output functions  $y = Tf$ . An important example,  $f(t) = \sin \alpha t$ , gives  $y(t) = \sin(\alpha t - \theta)/R^2$  with  $R^2 = (\alpha^2 + r^2)$  and  $\cos \theta = r/R$ ,  $\sin \theta = \alpha/R$ . In this example there is an obvious similarity between the graphs of the input and output functions.

**EXERCISE 6.17:** If a signal  $f(t) = \sin t$  is input into a device and produces output as described above, what value of  $r$  will produce the "delayed" output version of the signal,  $\sin(t - \pi/4)$ ? What distortion of the signal  $\sin 3t$  will be produced by this same device?

If the input signals are not sine or cosine in form, it may be difficult or impossible to find an analytical form of the output; however, a graph of the output may be found by using our differential equation graphing programs after using the calculator to evaluate the integral in  $y(0)$ .

**EXERCISE 6.18:** Let  $r = 1$ , and set the plot parameters so that  $0 \leq t \leq 3.14$  and  $0 \leq y \leq 1.2$ . Use the calculator to plot the following input functions with the Function DRAW program and the resulting output functions with a differential equation plotting program.

(a)  $f(t) = 1 - \sin^4(3t)$ ,      (c)  $f(t) = \text{Max}(\sin 6t, 0)$ .

(b)  $f(t) = 1 - \sin^{10}(3t)$ ,

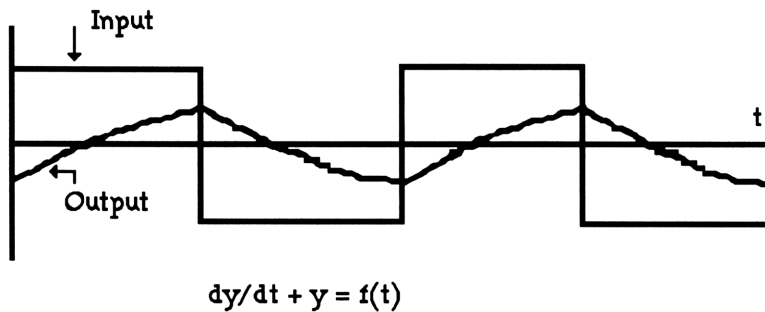
If  $f(t)$  is stored in EQ and  $P = 1.047 \approx \pi/3$ , the following program can be used to calculate  $y(0)$ :

```
<< 3 FIX 'T' PURGE 0 1.047 EQ 'EXP(T)' * 'T' 3 NEG
      SF J 3 NEG CF 1 1.047 EXP SWAP - / STD >>
```

The input signals in (a) and (b) are periodic, spike-like disturbances of a constant input and the input in (c) is a half-wave rectified sine function.

An observant student may notice that if we start with incorrect initial conditions then the solution approaches the periodic output after some time. This suggests that the starting condition  $y(0) = 0$  is ignored and that the resulting motion will become periodic. This is a result of the theorem that any solution of the non-homogeneous problem is the sum of a particular solution and a solution of the homogeneous problem.

The function  $f(t) = 2 * \text{CEIL}(\text{SIN}(t * \pi)) - 1$  has values given by: for  $0 < t < 1$ ,  $f(t) = 2 - 1 = 1$ , for  $1 < t < 2$ ,  $f(t) = -1$ , for  $2 < t < 3$ ,  $f(t) = 2 - 1 = 1$ , etc. This is called a *square wave*. The calculator numerical integration "key" and graphing program can handle such a function even though it is not defined at  $t = 1$ ,  $t = 2$ , etc. The periodic input function and its periodic output are shown for  $0 \leq t \leq 4$ .

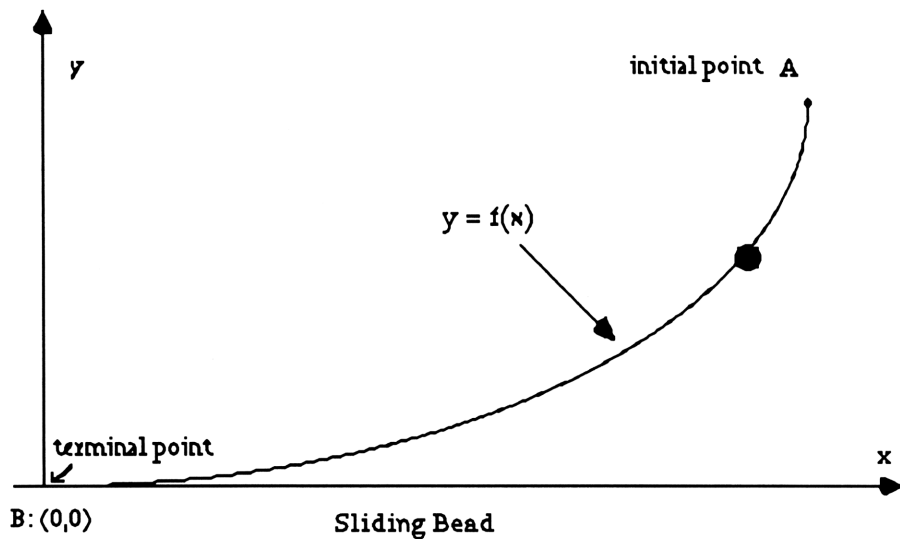


The student can also construct the input function shown above as  $\text{IFTE}(T \text{ MOD } 2 \leq 1, 1, -1)$ . In the same way, the switch function  $u_a(t) = 0$  when  $t \leq a$  and 1 otherwise can be given by an IFTE function or by  $u_a(t) = .5[1 + (t-a)/|t-a|] = 0$  when  $t < a$  and  $u_a(t) = 1$  when  $t > a$ . This could be called a switch-on function. Other interesting functions can be obtained using the MOD function on the calculator. For example,  $f(x) = '2 * X \text{ MOD } 1'$  will produce repeated ramps of height 2. Finally,

functions defined by different formulae in different intervals can be produced by the IFTE command: for example,  $f(x) = 2x$  for  $0 \leq x \leq .5$ ,  $f(x) = 1 - \sin(x - .5)$  for  $.5 \leq x \leq .5 + 1.571$ ,  $x^2$  for  $x > .5 + 1.571$  is produced by 'IFTE( $X \leq .5$ ,  $2 * X$ , IFTE( $X \leq .5 + 1.571$ ,  $\sin(X - .5)$ ,  $X^2$ ))'.

**PROJECT EXERCISE: Travel time for a sliding bead as a function of trajectory shape:**

We want to specify the shape of a curved wire by a function  $y=f(x)$ , which connects the point A with coordinates  $(x_1, y_1)$  to the origin  $(0,0)$  (denoted as point B) so that a bead of mass  $m$  will slide along the wire from A to B in a minimum amount of time. The bead begins with initial velocity zero and slides with no friction under the force of gravity  $g$ .



The well-known formula for arclength ,

$$s(x) = \int_x^{x_1} \sqrt{1 + \left( \frac{df(r)}{dr} \right)^2} dr,$$

the principal of conservation of energy (this is a conservative system),

$$\frac{1}{2} m v^2 + mgy = mgy_1 ,$$

and the expression for the travel time,

$$T = \int_0^T dt = \int_0^{s(0)} \frac{dt}{ds} ds = \int_0^{s(0)} \frac{1}{v} ds,$$

leads to the equation

$$T(f) = \int_0^{x_1} \sqrt{\frac{1 + \left( \frac{df(x)}{dx} \right)^2}{2 g(y_1 - f(x))}} dx ,$$

where we have explicitly noted that  $T$  depends on the curve  $y = f(x)$ . (We have used the technique of changing the variable of integration and the Fundamental Theorem of Calculus.)

We further specialize the example by taking  $x_1 = 200$ ,  $y_1 = 100$  and  $g = 1$ . Later, for this case we will find there is a curve such that the travel time  $T$  is approximately 25.231. For these parameters the following curves connect the points A and B:

(a)  $f(x) = x^2/400$

$$(b) \quad f(x) = \exp((x \ln 101)/200) - 1$$

$$(c) \quad f(x) = 100 [1 - \cos (\pi x/400)].$$

We will evaluate the travel time integrals for each of these curves using numerical integration.

For  $f(x) = x^2/400$  the descent time integral becomes

$$T = \int_0^{200} \sqrt{\frac{1 + \frac{x^2}{4 \left[ 10^4 \right]}}{2 \left[ 100 - \frac{x^2}{400} \right]}} dx .$$

Put  $x/200 = z$ , evaluate the integral to get  $T = 26.779$  (attempted accuracy: 0.01). Find  $T$  for the functions given in (b) and (c). (Note that so far, no differential equation has arisen.)

The differential equation

$$\frac{dy}{dx} = \sqrt{\frac{k^2 - (y_1 - y)}{y_1 - y}}$$

can be shown to give the minimum time of descent. We have

$$dx = \frac{\sqrt{y_1 - y}}{\sqrt{k^2 - (y_1 - y)}} dy.$$

Use the change of variables

$$y_1 - y = k^2 \sin^2 \frac{\phi}{2}$$

to get

$$dx = -k^2 \sin^2 \frac{\varphi}{2} d\varphi, \quad x = C - \frac{k^2}{2} (\varphi - \sin \varphi).$$

At  $\varphi = 0$ ,  $x = x_1$  and  $y = y_1$  and at  $\varphi = \varphi_1$  (to be determined),  $x = y = 0$ .

Thus the solution to the differential equation, along with the transformation  $y$  to  $\varphi$ , yields a parametric representation  $(x(\varphi), y(\varphi))$  of the curve of minimum descent time. There are two constants to be determined,  $k^2$  and  $\varphi_1$ ; then

$$x = x_1 - \frac{k^2}{2} [\varphi - \sin \varphi], \quad y = y_1 - \frac{k^2}{2} [1 - \cos \varphi], \quad 0 \leq \varphi \leq \varphi_1$$

The constraint  $x(\varphi_1) = y(\varphi_1) = 0$  leads to the equation

$$k^2 = \frac{y_1}{\sin^2 \frac{\varphi}{2}} = \frac{2y_1}{1 - \cos \varphi_1} = \frac{2x_1}{\varphi_1 - \sin \varphi_1}$$

We solve

$$G(\varphi) = \varphi - \sin \varphi - \frac{x_1}{y_1} (1 - \cos \varphi) = 0$$

for  $\varphi = \varphi_1$  with the calculator. Notice there is a positive solution. Now determine  $k^2$ . Using the differential equation in the expression for the descent time

$$T(y) = \int_0^{x_1} \sqrt{\frac{1 + \left( \frac{dy(x)}{dx} \right)^2}{2g(y_1 - y(x))}} dx,$$

we obtain the expression



$$T = \int_0^{x_1} \sqrt{\frac{1 + \frac{k^2 - (y_1 - y)}{y_1 - y}}{2(y_1 - y)}} dx = \frac{k}{\sqrt{2}} \int_0^{x_1} \frac{dx}{y_1 - y(x)} .$$

But  $y_1 - y = k^2 [1 - \cos \phi]/2$ , and  $dx = -k^2 [1 - \cos \phi]d\phi$  so that

$$T(y_{\text{opt}}) = \sqrt{\frac{k^2}{2g}} \phi_1 .$$

Use the values you get for  $\phi_1$  and  $k^2$  to evaluate  $T$ . You should find

$$T(y_{\text{opt}}) = 25.231.$$

The reader may note that the slope of the optimal curve is infinite at the initial point. This results in a quick start for the sliding bead. The optimal curve is called a cycloid. Optimality is shown in the study of the calculus of variations. Notice however, the exponential curve gives a travel time similar to the optimal curve.

### PROJECT EXERCISE: Travel time up a hill versus initial energy

This project is also concerned with the shape of a unknown function  $f(x)$ . Suppose we give to a particle with initial position at the origin, energy  $E$  in the form of initial velocity. The particle sliding on the shape function  $f(x)$  (we assume that  $f$  is an increasing function) leaves the origin, travels up the "hill"  $f(x)$ , reaches the limit of its travel at which all its energy has been converted into potential energy and then returns to the origin. We observe the time required to do this (as a function of the initial energy). (This problem is also illustrated by the figure given for the previous project.) The time between the particle's leaving and arriving back at the origin is given by

$$T(E) = \sqrt{2m} \int_0^{x(E)} \frac{\sqrt{1 + [f'(x)]^2}}{E - mgf(x)} dx,$$

and  $x(E) = f^{-1}(E/mg)$ , that is  $E - mgf(x) = 0$ .

Suppose the shape of the hill (i. e. the curve  $f(x)$ ) is one of the functions given in the previous project (i.e., a parabola, an exponential function, or a trigonometric function). Graph  $T$  vs  $E$  for  $E = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100$  (numerical integration required).

Now read the first part of the article by Keller on Inverse Problems in the *American Mathematician Monthly*, volume 83, 1976, pages 107-118, and describe what is meant by the inverse problem.

# 7

## SECOND ORDER DIFFERENTIAL EQUATIONS

We saw in chapter 5 that it is easy to program the HP-48 to treat a vector differential equation. Consider the case where the vectors have two components,  $y = [y_1, y_2]$ ; that is, initial value problems consisting of two first order differential equations and the initial values of the two dependent variables:

$$\frac{dy_1}{dt} = F_1(t, y_1, y_2), \quad \frac{dy_2}{dt} = F_2(t, y_1, y_2)$$

where  $y_1(t_0)$  and  $y_2(t_0)$  are given. You should note that a second order initial value problem

$$\frac{d^2x}{dt^2} = g\left(t, x, \frac{dx}{dt}\right), \quad \text{with } x(t_0) \text{ and } \frac{dx}{dt}(t_0) \text{ given,}$$

can be reduced to a first order system of differential equations

$$\frac{dy_1}{dt} = y_2, \quad \frac{dy_2}{dt} = g(t, y_1, y_2)$$

(with initial values of  $y_1$  and  $y_2$ ) by using the identification  $y_1 = x$  and  $y_2 = x'$ . A significant part of this chapter will be concerned with the study of solutions of second order differential equations. We will obtain approximations of these solutions using the calculator by studying the associated vector systems. This is a common practice on all kinds of calculators and computer programs. We will plot trajectories and study the solution characteristics of such systems. Of course, in this case we can plot  $y_1$  versus  $t$ ,  $y_2$  versus  $t$ , or plot  $y_1$  versus  $y_2$  as the parameter  $t$  varies.

As in the case of a single differential equation, the user has three choices: use the built-in plotting form, the user programs described in the first section of chapter 5, or the Euler or modified Euler algorithms in plotting programs as described in the second section of chapter 5. If you want to use the EULER and IULER programs as written for the vector case, make sure the F.N program will give a vector  $dY/dT$  when the input is a number  $T$  and a vector  $Y$ . Consider

$$\frac{dy_1}{dt} = y_2, \quad \frac{dy_2}{dt} = \cos t - y_1, \quad y_1(0) = y_2(0) = 0,$$

over the interval  $0 \leq t \leq 2\pi$ . An appropriate F.N program is

```
<< → T Y << 'Y(2)' EVAL 'Y(1)' EVAL NEG 2 →ARRAY >> >> .
```

(The first two stack items are used as local variables, and the inner program creates the first and second components of the output and forms a vector from these components.) After a value for  $H$  is stored, the stack input 0 [0 0] to either of the programs EULER or IULER will produce the values at  $T = H$ .

Suppose we wish to plot the vector solution of  $dy_1/dt = y_2$ ,  $dy_2/dt = -y_1$ ,  $y_1(0) = 1$ ,  $y_2(0) = 0$  using the built-in HP-48G algorithm on the interval  $0 \leq t \leq 2\pi$ . We can choose the input form Diff Eq under PLOT as indicated in chapter 5 or we can use the user programs constructed in chapter 5 which eliminate some of the inconveniences of the input form. In the latter case we execute the program IN.FN and respond with

```
<< 'Y(2)' EVAL 'Y(1)' EVAL NEG 2 →ARRAY >>
```

which will be stored in FN. We execute IN.PP, respond to set H-VIEW with -1.2 1.2, and respond with set V-View to -1.2 1.2. Then we enter 0 [1 0] 6.283 on the stack and execute G.12. The solution is  $y_1(t) = \cos t$ ,  $y_2(t) = -\sin t$  and the  $y_1$  versus  $y_2$  plot

should be a "circle". The actual figure is a set of points connected by straight lines. Exit to the stack and press T and Y in the VAR menu to get (approximately) 6.283 [1 0]. This will be more accurate than the result as drawn by G.Y12, say with H = .0628 and N = 100 which uses the improved Euler algorithm.

Probably the first type of second order differential equation you will study is a linear homogeneous equation with constant coefficients. Such an equation can be solved by finding appropriate values of a constant  $r$  so that  $x = e^{rt}$  is a solution of the problem. The calculator can be used to determine unknown coefficients in constructing general solutions. You should also use the calculator to become familiar with the plots of solutions that occur in common problems. This type of exercise will use the function grapher in the calculator and we will not study these kinds of problems here. But here is an associated problem. How are the coefficients in these differential equations obtained?

Several mathematical models lead to second order ordinary differential equations with constant coefficients. The coefficients are usually obtained from measurements either directly on the physical system or on solutions of the system. How could we deduce approximate values of these coefficients using measurements on the solutions? Consider the differential equation

$$\frac{d^2 y}{dt^2} + B \frac{dy}{dt} + Cy = 0.$$

A common solution function has the form

$$y(t) = ae^{-P t} + be^{-Q t},$$

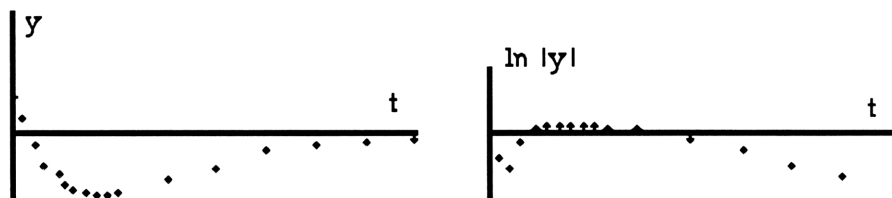
where  $a, b, p, q$  are parameters. Suppose a set of values  $\{(t, y)\}$  is obtained by making measurements. There is usually some experimental error in measurements so the entire set  $\{(t, y)\}$  will be used to find the parameters. In this example the measurements are:

$$\{(0, 1), (.1, .30), (.2, -.20), (.3, -.60), (.4, -.88), (.5, -1), (.6, -1.2), (.7, -1.2), (.8, -1.3),$$

$$(.9, -1.3), (1, -1.2), (1.5, -1), (2, -.70), (2.5, -.50), (3, -.3), (3.5, -.17), (4, -.11)\}$$

We want to deduce first approximate values of the parameters  $a, b, p, q$ , then use these to determine the parameters  $B$  and  $C$  in the equation above.

We may be able to learn something from a plot of the data. A program for producing such a plot is given below. We also recall that if  $y = Ae^{-kt}$  then a graph of  $\ln|y|$  versus  $t$  is a straight line with slope  $-k$ . So a graph of data  $\{(t, \ln|y|)\}$  may reveal information. Suppose we construct a list of the data and store it as **L1**.



**Graphs of Solution Observations**

The graphs shown are generated with the HP-48 program

```
<< ERASE DRAX L1 LIST→ 1 SWAP START PIXON NEXT GRAPH >>
  << ERASE DRAX L1 LIST→ 1 SWAP START C→R ABS LN
    R→C PIXON NEXT GRAPH >>
```

after setting H-VIEW to 0.4 and V-VIEW in the first problem to -1.4 1.1 and in the second program to -2.1 .28 (approximate values for  $\ln |-.11|$  and  $\ln 1.3$ ).

To get approximations for  $a$ ,  $b$ ,  $p$ , and  $q$  we proceed as follows: suppose  $p > q$ , then  $y(t) = e^{-q t}(ae^{-(p-q)t} + b)$ . Since the first term becomes negligible as  $t$  increases,  $b < 0$  (we replace  $b$  with  $-|b|$ ) and a plot of  $(t, \ln |y|)$  is a straight line for large  $t$  with slope  $-q$ . From the second graph we get  $q = -.5 \ln (.11/.7) = .925$  from the data points  $(4, -.11)$  and  $(2, -.7)$ . The first data point gives  $a = |b| + 1$  (which is, of course, an approximate equation), and  $dy/dt(.85) = 0$  gives  $p(1+|b|)e^{-.85p} = .42|b|$ . Finally we use the approximate equation  $\ln -y(t) = \ln |b| - .925 t = 0$  at  $t = 1.5$  which gives  $|b| = 4$ ,  $a = 5$ ,  $p e^{-.85p} = .336$ . This equation has two solutions  $p = .525$  and  $p = 2.2$ . Since we want  $p > q$ , we take  $p = 2.2$ . This yields the equation

$$y(t) = 5 e^{-2.2t} - 4e^{-.925t}.$$

You should now plot this equation together with the plot of the data for comparison. These approximate values for  $a$ ,  $b$ ,  $p$  and  $q$  can be taken as starting values to an iterative process to determine the parameter values by a least squares fit to data. See Chapter 9. It is easy to use the values of  $p$  and  $q$  to determine the corresponding values of  $B$  and  $C$  in the differential equation.

**EXERCISE 7.1:** Suppose the following data is collected on the solution of the second order differential equation given above.

{ (0, 11.04), (.4, 12), (.8, 11.06), (1.2, 8.47), (1.6, 4.75), (2, .54), (2.4, -3.48), (2.8, -6.71), (3.2, -8.7), (3.6, -9.22), (4, -8.29), (4.4, -6.15), (4.8, -3.2), (5.2, .05), (5.6, 3.09) }

Find approximate values of  $B$  and  $C$  in the differential equation.

## 7.1 SECOND ORDER INPUT OUTPUT PROBLEMS

We now consider constructing the solution of a non-homogeneous second order differential equation with constant coefficients. The problem is treated in many textbooks for special types of forcing, usually sine or cosine forcing functions. A model for an elastic spring with damping and with external forcing  $f(t)$  or a model for a simple electrical circuit loop with external voltage is:

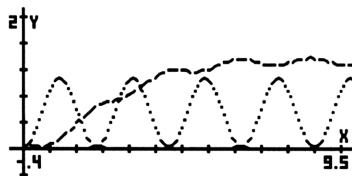
$$\frac{d^2x}{dt^2} + 2r \frac{dx}{dt} + \omega^2 x = f(t), \quad x(0) = \frac{dx}{dt}(0) = 0, \quad \omega^2 > r^2.$$

The solution is given by

$$x_q(t) = \frac{1}{\mu} \int_0^t e^{-r(t-s)} \sin \mu(t-s) f(s) ds, \quad \mu = \sqrt{\omega^2 - r^2}.$$

As indicated before, this problem is equivalent to the pair of differential equations  $dy_1/dt = y_2$ ,  $dy_2/dt = f(t) - 2ry_2 - \omega^2 y_1$ ,  $y_1(0) = y_2(0) = 0$ .

**EXAMPLE:** Take  $\omega^2 = .41$ ,  $r = .5$  (so  $\mu^2 = .16$ ) and  $f(t) = \sin^2(1.5t)$ . Set FN as `<< 'Y(2)' EVAL 'SIN(1.5*T)^2 - Y(2) - .41*Y(1)' EVAL 2 →ARRY >>` and the plotting parameters to show  $0 \leq t \leq 9.42$ ,  $0 \leq y \leq 2$ . Put `0 [0 0] 9.42 1` on the stack and execute G.OI. Next overlay a plot of the input function. The forcing function (input) and solution (output) resulting from this program are shown below.



**EXERCISE 7.2:** Find the output graph for  $f(t) = 1 - \sin^4(3.14t)$  for  $\mu = 1$ , and  $r = .5$ . Choose plot parameters to show  $0 \leq t \leq 6$  and  $0 \leq y_1 \leq 1$ . Add the input function



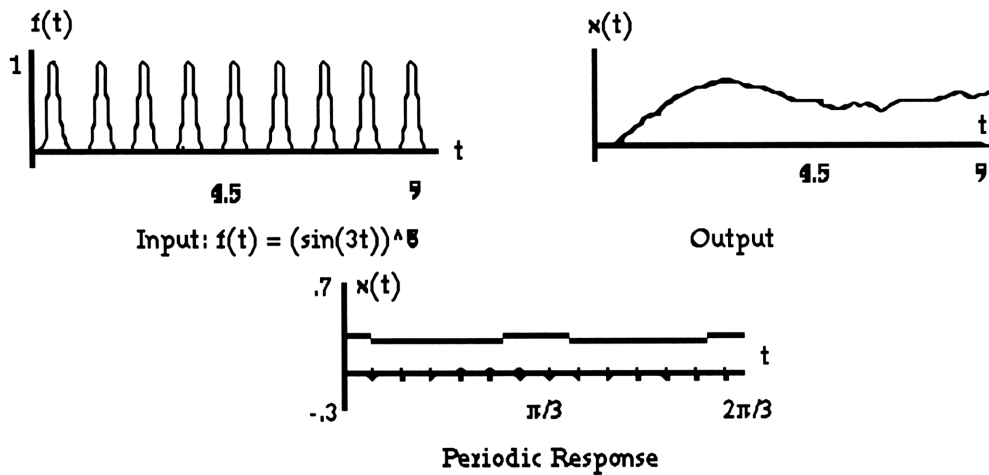
graph as an overlay. Comment: The output function for this input can be obtained from a table of integrals after several substitutions using the method of undetermined coefficients. But, an output function for an input  $f(t) = 1/(2 - \sin^4(3.14t))$  can not be found this way.

Suppose the forcing function  $f(t)$  is periodic with period length  $P$ . If we can change the initial conditions so that  $x(P) = x(0)$  and  $x'(P) = x'(0)$ , then the resulting solution is periodic. And if the damping coefficient  $r > 0$ , then all solutions will eventually be close approximations to the periodic solution when viewed over one period. We may want to view such a solution without waiting for asymptotic behavior to emerge. Suppose we determine solutions  $x_1(t)$  and  $x_2(t)$  of the associated homogeneous system so that  $x_1(0) = x_2'(0) = 1$  and  $x_1'(0) = x_2(0) = 0$ ; then a general solution is  $x(t) = a x_1(t) + b x_2(t) + x_q(t)$  where  $x_q(t)$  is the solution constructed above for 0 initial conditions for  $x$  and  $x'$ . Expressions for  $x_1(t)$  and  $x_2(t)$  are  $x_1(t) = e^{-rt} [\cos \mu t + (r/\mu) \sin \mu t]$  and  $x_2(t) = (1/\mu) e^{-rt} \sin \mu t$ . We can use the calculator to compute the integrals in  $x_q(P)$  and  $x'_q(P)$ , then we can use the calculator to solve the periodicity condition for  $a$  and  $b$ :

$$\begin{bmatrix} 1 - x_1(P) & -x_2(P) \\ -x'_1(P) & 1 - x'_2(P) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x_q(P) \\ x'_q(P) \end{bmatrix}.$$

The periodic response can be obtained by using G.0I with input 0 [a b] P 1 on the stack.

Output for  $f(t) = (\sin 3t)^8$ ,  $r = .25$  and  $\mu = 1$  with the initial conditions  $x(0) = dx/dt(0) = 0$  is shown below. This input has period  $\pi/3$ . The periodic response is also shown over two periods. The average value for this forcing  $f(t)$  is  $\pi/6$  and  $f(t) = [f(t) - \pi/6] + \pi/6$ , so a portion of the periodic response is the constant  $\pi/(6\omega^2)$ .



**EXERCISE 7.3 :** Find and plot the periodic output response for  $f(t) = 1 - \sin^4(3.14t)$  for  $\mu = 1$ , and  $r = .5$ . Then overlay a plot of the input forcing function.

The friction/resistance term in the spring/circuit model that we have been considering is given by  $2r dx/dt$  and the restoring force term is  $\omega^2 x$ . These terms are usually approximations for nonlinear phenomena. What happens to the periodic response in the mathematical model driven by periodic input when the terms are replaced by nonlinear functions? The method of calculating the correct initial conditions no longer applies; however, in some cases the solution to the differential equation with a variety of initial conditions will settle toward a periodic steady state solution as time increases.

**EXERCISE 7.4 :** Find a periodic solution to nonlinear problems of the form

$$\frac{d^2 x}{dt^2} + R\left(\frac{dx}{dt}\right) + K(x) = 2 \cos t.$$

Set H-VIEW (i. e., XRNG) = -2 6.283, H-VIEW (i. e., YRNG) = -2.1 2.1. Let

$$R_1(dx/dt) = 2 * \text{IFTE}(dx/dt \leq -1, dx/dt + .5, \text{IFTE}(dx/dt \leq 1, .5 * dx/dt, dx/dt - 1)).$$

- (a) Take  $R = R_1(dx/dt)$  and  $K(x) = x$ . Use initial condition  $t = 0, x = 0, x' = 1.56$ .
- (b) Take  $R(dx/dt) = dx/dt$  and  $K(x) = x$ . Use initial condition  $t = 0, x = 0, dx/dt = 2$ .
- (c) Take  $R(dx/dt) = dx/dt$  and  $K(x) = \sin x$ .
- (d) Take  $R(dx/dt) = R_1(dx/dt)$  and  $K(x) = \sin x$ .

Note: In each case, if the solution you get is not periodic then use the values of  $x$  and  $dx/dt$  at  $t = 6.283$  as initial conditions and generate another solution. Which nonlinearities caused a phase shift from the linear case (b)?

Suppose that in the system  $dY/dt = F(t, Y)$ ,  $F$  is periodic in  $t$ . Since the system has a periodic rhythm, perhaps the rhythm will also exist in the solutions. It is easy to use the calculator to illustrate the idea of locating a solution for  $t =$  any multiple of a given time period. For example, the differential equation

$$\frac{d^2 x}{dt^2} + \omega^2 x = .5 \cos t, \quad \omega \neq 1,$$

together with the initial condition  $x(0) = \xi, dx/dt(0) = 0$  has solution

$$x(t) = \left[ \xi + \frac{1}{2(1 - \omega^2)} \right] \cos \omega t - \frac{1}{2(1 - \omega^2)} \cos t$$

$$\frac{dx}{dt}(t) = -\omega \left[ \xi + \frac{1}{2(1 - \omega^2)} \right] \sin \omega t + \frac{1}{2(1 - \omega^2)} \sin t$$

What are the properties of such a solution? For  $t = 2\pi n$  we have

$$\frac{x(2\pi n) + \frac{1}{2(1 - \omega^2)}}{\xi + \frac{1}{2(1 - \omega^2)}} = \cos 2\pi n \omega, \quad \frac{\frac{dx}{dt}(2\pi n)}{\omega \left[ \xi + \frac{1}{2(1 - \omega^2)} \right]} = -\sin 2\pi n \omega.$$

By squaring both sides, we see that the points  $x(2\pi n)$ ,  $x'(2\pi n)$  lie on an ellipse.

**EXERCISE 7.5:** Plot the points  $x(2\pi n)$ ,  $x'(2\pi n)$  for  $\xi = 0$  and  $n = 1, 2, \dots$  (several values of  $n$ ) for  $\omega = 1/\text{Sqrt}(5)$  and for  $\omega = 1/3$ . Note that the points cycle around the ellipse. If  $\omega n$  is an integer  $m$  for some integer  $n$ , then you can see the solution is periodic, but what happens when  $\omega$  is irrational?

The graph of  $\{ (x(nT), dx/dt(nT)) : n = 0, 1, 2, \dots \}$  of the solution of a differential equation  $dx/dt = f(x, y, t)$ ,  $dy/dt = g(x, y, t)$  when the function  $f$  and  $g$  have period  $T$  in  $t$  is called a *Poincare section*. The following program collects 10 points for such a graph on the stack:

```
<< { T Y FN } TOL 1 10 FOR N  $\Pi$  N * 2 *  $\rightarrow$ NUM  $\rightarrow$  TF
<< TF RKF 'Y(1)' EVAL 'Y(2)' EVAL RC 3 ROLLD >> NEXT >>
```

Executing this program for the FN function

```
<< 'Y(2)' EVAL '12*COS(T) - X -  $\epsilon$ *('Y(1)^2-1)*Y(2)' EVAL 2  $\rightarrow$ ARRAY >>
```

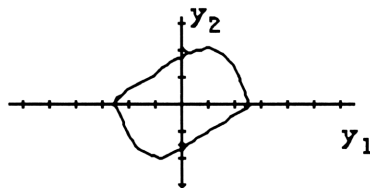
(periodically forced Van der Pol equations) for different values of  $\epsilon$  gives the data:

	$\epsilon = .05$	$\epsilon = .1$	$\epsilon = .15$
initial value	$Y = [1, 1]$	$Y = [1, 1]$	$Y = [1, 1]$
first section point	$Y = [-2.1, 10.4]$	$Y = [-2.24, 7.97]$	$Y = [-2.21, 6.08]$
second section point	$Y = [-2.19, 10.28]$	$Y = [-1.99, 8.28]$	$Y = [-1.82, 6.65]$
third section point	$Y = [-2.19, 10.28]$	$Y = [-1.98, 8.29]$	$Y = [1.79, 6.86]$

There were no further changes in the section points coordinates (to 6 places). In each case the solution with initial value  $Y = [1, 1]$  collapsed to a periodic solution.

*Warning:* the program's execution is about 30 minutes.

It is interesting to plot a solution starting at one of the section points over a period of the system. The following figure is such a plot for one of the examples above.



**Forced Van der Pol with  $\epsilon = .1$**

We have already noted that if periodic forcing with period  $T$  is imposed on the undamped oscillator, solutions may result which have a periodic behavior of a period inherited from the natural frequency of the oscillator and  $T$ . To be specific, the solution for

$$\frac{d^2 x}{dt^2} + \omega^2 x = F_0 \cos \gamma t, \quad x(0) = \frac{dx}{dt}(0) = 0$$

for  $\gamma \neq \omega$  is

$$x(t) = \frac{2F_0}{\omega^2 - \gamma^2} \sin\left(\frac{\omega + \gamma}{2} t\right) \sin\left(\frac{\omega - \gamma}{2} t\right).$$

This special solution for a forced harmonic oscillator is quite unusual. The homogeneous differential equation associated with this problem has periodic solutions of period  $2\pi/\omega$ , the forcing has period  $2\pi/\gamma$ . The presence of this, possibly periodic solution with a different period, although surprising, comes from the interaction of the forcing function with the solutions of the associated homogeneous

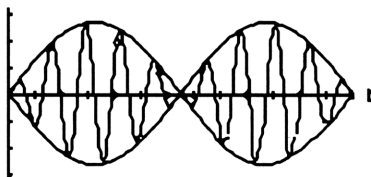
problem. Moreover the structure of this special solution is intriguing because it has the form of a relatively slowing amplitude,  $\sin((\omega-\gamma)t/2)$  multiplying a factor which varies at a faster rate.

**EXERCISE 7.6:** (a) Store the functions

$$\frac{2}{\omega^2 - \gamma^2} \sin\left(\frac{\omega + \gamma}{2} t\right) \sin\left(\frac{\omega - \gamma}{2} t\right), \quad \frac{2}{\omega^2 - \gamma^2} \sin\left(\frac{\omega - \gamma}{2} t\right) \text{ and} \\ \frac{-2}{\omega^2 - \gamma^2} \sin\left(\frac{\omega - \gamma}{2} t\right)$$

in the function grapher of your calculator. These function are, of course,  $x(t)$  as given above, and two additional curves which are bounding curves for  $x(t)$ . Plot each of the functions for  $0 \leq t \leq 4\pi/|\gamma - \omega|$  for the cases: (i)  $\omega = 2, \gamma = 22/9$ , (ii)  $\omega = 2, \gamma = 20/7$ . Keep a copy of these graphs for comparison with later work.

The case  $\omega = 2, \gamma = 22/9$  is graphed below.



**Beats Vibration**

(b) Suppose  $.5|\gamma - \omega| \tau = 2\pi$ . Then  $.5(\gamma + \omega) \tau = 2\pi (\gamma + \omega) / |\gamma - \omega|$ . What is the difference of the behavior of  $x(t)$  for  $t$  near  $\tau$  between the case when  $(\gamma + \omega) / |\gamma - \omega|$  is an even integer and the case when  $(\gamma + \omega) / |\gamma - \omega|$  is an odd integer? Try some examples to discover the answer. For example graph the solution given above for cases (i) and (ii) in the classroom problem given above and in the cases (iii)  $\gamma = 18/7$  and  $\omega = 2$ , (iv)  $\gamma = 8/3$  and  $\omega = 2$ . Give a partial answer based on your result to the

question: does an input forcing of period  $T = 2\pi/\gamma$  gives periodic output of least period  $4\pi/|\gamma - \omega|$ .

Many questions concerning the presence of a "special" solution to a forced oscillator may occur to you. We mention several below.

The external forcing in the oscillator mentioned above is periodic (with least period  $2\pi/\gamma$ ) and has average value zero. Is it possible to replace  $F_0 \cos \gamma t$  with another function with these properties and discover a solution which resembles the special solution studied above? As an example we consider the function

$$f(t) = F_0 \sum_{n=0}^{\infty} (-1)^n \delta(t - n \frac{\pi}{\gamma})$$

consisting of a sum of Dirac delta functions. You should show the transform of the solution of  $y'' + \omega^2 y = f(t)$ ,  $y(0) = y'(0) = 0$  is

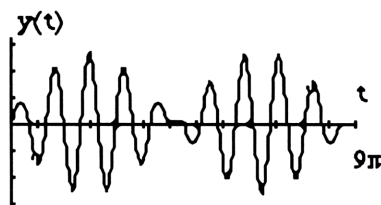
$$Y(s) = \frac{F_0}{\omega} \sum_{n=0}^{\infty} (-1)^n \frac{\omega e^{-(n\pi/\gamma)s}}{s^2 + \omega^2}$$

and the corresponding solution is

$$y(t) = \frac{F_0}{\omega} \sum_{n=0}^{\infty} (-1)^n \sin\{\omega(t - n\pi/\gamma)\} u(t - n\pi/\gamma).$$

Here  $u(t) = 0$  for  $t \leq 0$ ,  $= 1$  otherwise. For  $\omega = 2$ ,  $\gamma = 22/9$ , we obtain the graph shown below for  $0 \leq t \leq 9\pi = 28.274$ . Note the input function changes sign every 1.285 units so we must sum at least 25 terms to graph this interval. We note that the graph has generally the same shape as that arising from the forcing  $f(t) = \cos \gamma t$ ; however there is some variation for positive  $t$  near  $t = 0$  where the slope of the graph is 2 rather than 0 as in the cosine case, near  $t = 9\pi/2$  with a flat spot and near  $t = 9\pi$  with a flat spot. The width of the flat spot at  $4.5\pi$  is approximately 1.1 units

and the graphs of  $\pm 1.4 \sin(2t + .55)$  are good enveloping curves for this "distorted beats vibration.



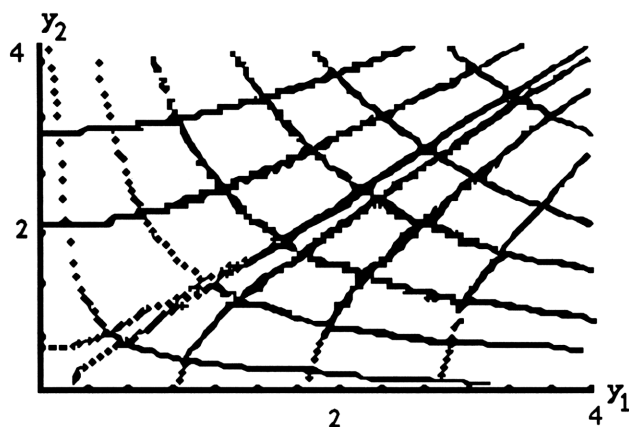
**Response of Harmonic Oscillator to Periodic Impulses**

If the graph is extended to  $0 \leq t \leq 18\pi$ , it is seen that the response apparently has period  $9\pi$ . Next question: for a case in which  $(\gamma + \omega)/|\gamma - \omega|$  is an odd integer, does the resulting solution have least period  $2\pi/|\omega - \gamma|$ ? To get an indication of the answer, try the case  $\omega = 2$  and  $\gamma = 20/7$ .

## 7.2 TRAJECTORIES IN THE $y_1$ - $y_2$ PLANE

A topic occurring early in many differential equation textbooks is that of determining trajectories that are orthogonal to the members of a one-parameter family of curves, say  $W(y_1, y_2, p) = 0$ . The usual technique is to first find the differential equation satisfied by the members of the given curve family, say  $dy_2/dy_1 = m(y_1, y_2)$ ; then curves that are orthogonal satisfy the differential equation  $dy_2/dy_1 = -1/m(y_1, y_2)$ . If the original family is given by equations of the form  $dy_1/dt = f(y_1, y_2)$ ,  $dy_2/dt = g(y_1, y_2)$ , trajectories for orthogonal curves satisfy  $dy_1/dt = -g(y_1, y_2)$ ,  $dy_2/dt = f(y_1, y_2)$ . This latter form is preferred if the curves in either family must be specified in terms of a parameter  $t$ . Clearly, the program G.12 (or G.Y12) can be used to plot members of both the given family of curves and the orthogonal trajectories. This is our first example of what is called an *autonomous system*. A specific example is shown.





Orthogonal Trajectories  $dy_1/dt = -y_2/y_1$ ,  $dy_2/dt = y_1/y_2$

**EXERCISE 7.7:** Set the plot parameters to show both H-VIEW and V-VIEW as  $-.5 \quad 3.5$  and enter the following FN:

<< 'Y(1)\*(Y(1)^2-Y(2)^2)' EVAL 'Y(2)\*(3\*Y(1)^2-Y(2)^2)' EVAL 2 →ARRY >>.

Create a composite plot in the  $y_1$ - $y_2$  plane resulting from the inputs to the G.12:

$t_0$	0	0	0	0	0
$y_0$	[.5 .1]	[.75 .1]	[1 .1]	[1 .4]	[1.5 .5]
$t_f$	4	4	2	2	2

These are five solution trajectories (ovals) for the system

$$dy_1/dt = y_1(y_1^2 - y_2^2), \quad dy_2/dt = y_2(3y_1^2 - y_2^2).$$

Now overlay the solution trajectories of the orthogonal system corresponding to the following inputs to the G.12 program:

$t_0$	0	0	0	0	0
$y_0$	[0 3.4]	[0 2.5]	[0 1.5]	[2 0]	[3.4 0]
$t_f$	1.2	.8	.8	.8	.8

Plots in the  $y_1$ - $y_2$  plane of solutions  $(y_1(t), y_2(t))$  of differential equations  $y_1' = F_1(y_1, y_2)$ ,  $y_2' = F_2(y_1, y_2)$  are called *phase plane* plots. If  $F_1(y_1, y_2)$ , and  $F_2(y_1, y_2)$ , have continuous partial derivatives, solutions to initial value problems are unique and it is elementary to show that under such circumstances solution trajectories arising from different initial points either coincide or do not intersect. If fact, it is easy to see that if  $(y_1(t), y_2(t))$  is a solution of an equation of this form and  $a$  is any constant, then  $(y_1(t+a), y_2(t+a))$  is also a solution. *Closed trajectories* in the phase plane indicate periodic solutions. Constant solutions, that is, points  $(y_1, y_2)$  such that  $F_1(y_1, y_2) = F_2(y_1, y_2) = 0$  are called *critical point solutions* (also equilibrium solutions). Other trajectories of particular interest are those nearby to a critical point.

- If trajectories arising at all points within some circle around a critical point  $(y_{1c}, y_{2c})$  leave the vicinity of  $(y_{1c}, y_{2c})$  as  $t \rightarrow \infty$ , then  $(y_{1c}, y_{2c})$  is called a *repelling solution*, i.e., unstable.
- If trajectories arising at all points within some circle around a critical point  $(y_{1c}, y_{2c})$  approach  $(y_{1c}, y_{2c})$  as  $t \rightarrow \infty$ , then  $(y_{1c}, y_{2c})$  is called an *attracting solution*, i.e., asymptotically stable.

Some well-studied examples of autonomous are presented below. Note the asymptotic behavior of the solution trajectories as indicated by the graphs.

**EXERCISE 7.8 :** Systems called Lotka-Volterra systems may be scaled to the form

$$dy_1/dt = y_1(3 - y_2), \quad dy_2/dt = y_2(y_1 - 3).$$

Such systems arise in the study of populations of two species, one of which feeds on the other. Trajectories that begin in the first quadrant are periodic. Plot the solution that starts at  $(0, 2)$ , for  $0 \leq t \leq 2.25$ , after setting the plot parameters to show H-VIEW 0 6, and V-VIEW 0 6, by using the plot program G.12.

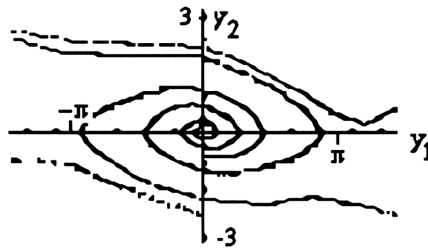
**EXAMPLE.** The differential equations

$$x'' + cx' + \sin x = 0 \quad \text{or} \quad y_1' = y_2, \quad y_2' = -\sin y_1 - cy_2$$

arise in the study of the displacements of damped (or undamped) pendulums. The critical points are  $(0,0)$  and  $(n\pi, 0)$ . For  $c > 0$ ,  $(0, 0)$  is an attracting solution. We use G.12,  $c = .3$ , and FN given by

```
<< 'Y(2)' EVAL 'sin(Y(1))+.3*Y(2)' EVAL NEG 2 →ARRY >>
```

to obtain the following graph. (For  $c = 0$ , there is a family of periodic solutions.)



**Damped Pendulum Motion ( $c = .3$ )**

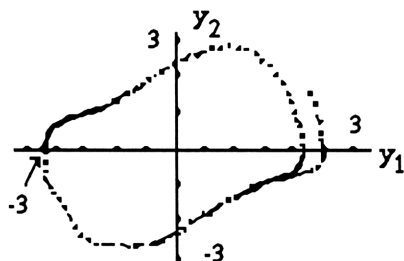
**EXAMPLE.** The system

$$dy_1/dt = -2y_2 + y_1(1-r^2)/r, \quad dy_2/dt = 2xy_1 + y_2(1-r^2)/r:$$

where  $(r^2 = y_1^2 + y_2^2)$  has an isolated periodic solution  $r = 1$ . Here ,nearby solutions spiral towards the circle  $r = 1$ . To obtain graphs use **G.12** and the function FN given by

```
<< '-2*Y(2)+Y(1)*(1-Y(1)^2-Y(2)^2)/(Y(1)^2+Y(2)^2)^.5' EVAL
'2*Y(1)+Y(2)*(1-Y(1)^2-Y(2)^2)/(Y(1)^2+Y(2)^2)^.5' EVAL 2 →ARRY >>.
```

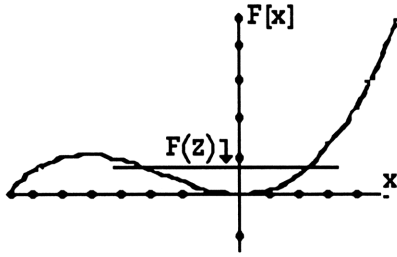
Another problem that has an isolated attracting periodic solution is the Van der Pol differential equation. This equation was studied in connection with its application to an electronic component. This example is usually studied as a function of a parameter  $\mu$  contained in the "damping" term. Our figure shows a



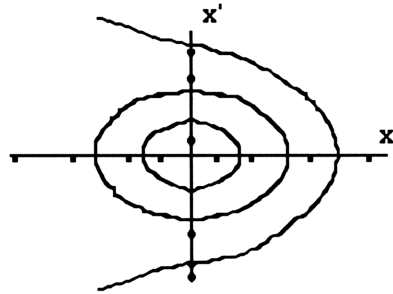
$$\frac{dy_1}{dt} = y_2, \quad \frac{dy_2}{dt} = -[y_1 + .3(y_1^2 - 1)y_2]$$

typical graph: here  $\mu = .3$ . Notice that the motion is counterclockwise and that the solution was started at  $(x, y) = (2, 2)$ . The solution quickly moves close to its asymptotic shape and is periodic. Solutions starting inside the closed curve (except from  $(0, 0)$ ) also move out to the periodic solution. Variation of the parameter  $\mu$  causes dramatic changes in the shape and period of the solution.

**EXERCISE 7.9:** In this exercise we will examine the cycle times of periodic solutions of several special differential equations. The equations under consideration have solutions that resemble the trajectories graphed in the figure below.



Construction for Trajectories  
(See region between  $F[z]$  &  $F[x]$ )



Trajectories

Here we assuming that  $y(t)$  satisfies the initial value problem

$$\frac{d^2 x}{dt^2} + f(x) = 0, \quad x(0) = z, \quad \frac{dx}{dt} = 0,$$

where the essential feature of  $f(x)$  is that it changes sign from negative to positive as  $y$  increases through zero. We multiply by  $dx/dt$  and integrate from 0 to  $t$  to obtain

$$\frac{dx}{dt} = \pm \sqrt{F(z) - F(x)}, \quad \text{where } F(x) = 2 \int_0^x f(s) ds.$$

If we denote by  $P/2$  the time for the trajectory to proceed from the starting point to the state  $x(P/2) = z_1$ ,  $dx/dt(P/2) = 0$ , then

$$P = 2 \int_0^{P/2} dt = 2 \int_{z_1}^z \frac{dx}{\sqrt{F(z) - F(x)}}.$$

We list the value  $z_1$  for several examples:

- (a)  $f(x) = x$ ,  $F(x) = x^2$ ,  $z_1 = -z$   
 (b)  $f(x) = \sin x$ ,  $F(x) = 2[1 - \cos x]$ ,  $z_1 = -z$   
 (c)  $f(x) = x + x^2$ ,  $F(x) = x^2 + 2x^3/3$ ,  $z_1$  = largest negative root of  $\frac{2}{3}x^2 + [1 + \frac{2}{3}]x + [z + \frac{2}{3}z^2] = 0$ .  
 (d)  $f(x) = x + x \cos 4x + .25 \sin 4x$   $F(x) = x^2 + .5x \sin 4x$ ,  $z_1 = -z$

Notice that in (a), (b) and (d), the function  $F$  is even in  $x$ , but in (c) it is not. Calculate and plot the values of  $P$  for one of the examples (a), (b), or (c) listed above for several values of  $z$ . Use the numerical integration key (program) on your calculator with a tolerance of 0.005. The following values of  $P$  are for part (d) above:

$z$ values	.25	.5	.75	1	1.25	1.5	1.75	2	2.25
$P$ values	3.94	5.29	12.74	21.54	8.29	5.74	5.04	5.45	8.37

Note that  $dx/dt = 0$  and  $x = \pi/4$  and  $dx/dt = 0$ ,  $x = 3\pi/4$  are equilibrium points.

## 7.3 LINEAR VARIATIONAL SYSTEMS IN THE $y_1$ - $y_2$ PLANE

*Linear autonomous systems* can be solved analytically. These systems have the form:

$$dy_1/dt = a_{11}y_1 + a_{12}y_2, \quad dy_2/dt = a_{21}y_1 + a_{22}y_2$$

We will consider the case  $\det(A) \neq 0$ , which means that the origin (0,0) is the only critical point. Special solutions have the form  $w = \text{column} [y_1, y_2] = e^{\lambda t} v$  where  $\lambda$  is a solution of the equation  $\det(A - \lambda I) = 0$  and  $v$  will be given below. Such a number  $\lambda$  is called an *eigenvalue* of the system. The equation  $\det(A - \lambda I) = 0$  is called the *characteristic equation* or the *eigenvalue equation* for the system. If  $\lambda$  is an

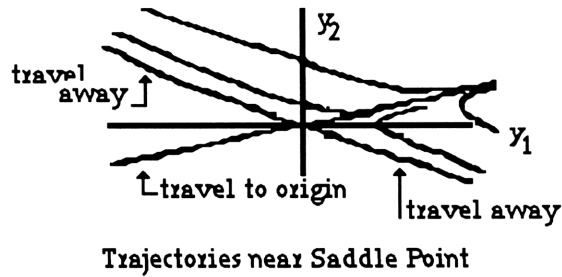
eigenvalue for the system then the column vector  $v = [c, d]$  is a non-zero solution of  $(A - \lambda I)v = 0$ . Other solutions of our system are linear combinations of these special solutions (in most cases).

The solution graphs of such systems near the origin (0,0) are particularly interesting. Examples fall into the following cases: closed trajectories (indicating a family of periodic solutions), spiraling trajectories (inward or outward spirals) and curved spoke-like trajectories (again traveling toward or away from the origin). The cases correspond to the type of eigenvalues for the system, viz. purely imaginary values, complex numbers with non-zero real parts and real eigenvalues.

**EXAMPLE:** Consider the system

$$dy_1/dt = y_1 - 4y_2, \quad dy_2/dt = -y_1 + 2y_2.$$

The associated matrix  $A$  has eigenvalues  $\lambda = .5(3 \pm 17.5i)$  and corresponding eigenvectors  $c = \text{column } [4, 1.56]$  and  $c = \text{column } [-4, 2.56]$ . When a solution starts on a multiple of the first eigenvector, it proceeds toward the origin exponentially. When a solution starts on a multiple of the second eigenvector it travels away from the origin exponentially. Other solutions are a linear combination of these two solutions and eventually proceed away from the origin. Typical trajectories are shown in the figure below. The procedure was to start on the eigenvector solution and trace that trajectory. Other solutions starting very near these special solutions were followed for short periods.



**EXERCISE 7.10:** For the case  $\lambda$  is complex and has negative real part the origin,  $(0,0)$  is called a spiral point critical point (so we have an attracting critical point). Use G.12 to study

$$dy_1/dt = -.5y_1 + 4y_2, \quad dy_2/dt = -4y_1 - .5y_2$$

Start at  $(t, y) = (0, [0, 1])$  after setting the plot parameters to show H-VIEW -2 2 and V-VIEW -1 1 and plot for  $0 \leq t \leq 3$ .

**EXERCISE 7.11 :** Use G.12 to graph the trajectories initiating at  $(t, [y_1, y_2]) = (0, [0, 1])$  and at  $(0, [-1, -1])$  for the system

$$\frac{dy_1}{dt} = -(2y_1 + y_2), \quad \frac{dy_2}{dt} = -y_1 + 2y_2.$$

What are the eigenvectors for this system associated with the  $[0, 0]$  critical point? Can you see them on the graphs? The graph should show that the origin  $(0,0)$  is neither an attracting or repelling critical point solution for the system.

**EXERCISE 7.12 (a)** Use the calculator to draw a graph of the solution of

$$\frac{dx}{dt} = x - \frac{5}{2}y, \quad \frac{dy}{dt} = \frac{3}{2}x - 3y$$



with initial conditions  $x(0) = -.5$ ,  $y(0) = .5$  for  $0 \leq t \leq 4$ . Use XRNG scale  $-1.4 \leq x \leq 1.4$ , and YRNG  $-1 \leq y \leq 1$ . When the plotting program is completed, record the final values of the solution  $x(4)/y(4)$ . Consider the matrix of coefficients  $A = \{\text{column}[1, 1.5], \text{column}[-2.5, -3]\}$ . What is the characteristic equation  $\det(A - rI) = 0$ ? What are the solutions  $r_1, r_2$ ? Give nontrivial solutions of  $(A - rI)\mathbf{v} = 0$  for  $r = r_1$  and for  $r_2$ . Calculate  $v_1/v_2$  for each solution. Compare with the answer you obtained for  $x(4)/y(4)$ .c) Give the general solution of  $dw/dt = Aw$ . Which term tends to vanish first as  $t$  increases?

(b) Use the calculator to draw a plot of the solution of

$$\frac{dx}{dt} = 5.7x - 10y, \quad \frac{dy}{dt} = 4x - 6.3y$$

with initial conditions  $x(0) = .3$ ,  $y(0) = -.2$  for  $0 \leq t \leq 6$ . Use XRNG scale  $-1.47 \leq x \leq 1.7$ , and YRNG  $-1 \leq y \leq 1$ . Consider the matrix of coefficients  $A = \{\text{column}[5.7, 4], \text{column}[-10, -6.3]\}$ . What is the characteristic equation  $\det(A - rI) = 0$ ? What are the solutions  $r_1, r_2$ ? Give nontrivial solutions of  $(A - rI)\mathbf{v} = 0$  for  $r = r_1$  and for  $r_2$ . Give the general solution of  $dw/dt = Aw$ .

Solution graphs of *nonlinear autonomous systems* near a critical point solution can be studied using a linear approximation. Let the vector  $y = \text{column}[y_1, y_2]$  and suppose we have the system  $dy/dt = F(y)$  for  $F(y) = \text{column}[F_1(y_1, y_2), F_2(y_1, y_2)]$ , and  $F_1(y_{1c}, y_{2c}) = F_2(y_{1c}, y_{2c}) = 0$ . Solution behavior near the critical point  $y_c = (y_{1c}, y_{2c})$  can be determined by studying the *linear variational matrix*  $J(y_c) = F_y(y_c)$  defined below. If all eigenvalues of this matrix have negative real parts, the solution  $y = y_c$  is an attracting solution. If one of the eigenvalues has a positive real part, some solutions leave immediate neighborhoods of the critical point. The matrix  $J(y_c)$  has  $(i, j)$  element

$$\frac{\partial F_i}{\partial y_i}(y_c)$$

**EXAMPLE:** Consider the system  $dy_1/dt = 2y_1^2 + y_2^2 - 9$ ,  $dy_2/dt = y_1^2 + y_2^2 - 5$ , which has critical point solutions  $(2, 1)$ ,  $(-2, 1)$ ,  $(2, -1)$ ,  $(-2, -1)$ . The variational matrix for the last critical point has eigenvalue equation  $\lambda^2 + 16\lambda + 8 = 0$ . The roots of this equation clearly are negative so that  $(-2, -1)$  is an attracting critical point.

The calculator can be used to find the matrix  $J$  associated with any equilibrium point  $y_c$  by using the sequence of programs given below. Because such information is also useful for a vector system  $dy/dt = F(y)$  where  $y$  and  $F(y)$  are vectors with  $m$  components, we present the programs for the vector case. We further will present the programs in a form where the labeling of the independent variables can be specified by the user. For example, instead of  $y_1, y_2$ , etc. the user might prefer  $u, v, \dots$ . The user's preference will be entered into a stored list as shown. After the matrix  $J$  is determined then the calculator can be used to find the eigenvalues as explained in the next section of this chapter.

Here is an outline of the procedure, assuming that we know the point  $y_c$ . We store the value of  $m$  in  $M$  and store the names of the  $m$  components in a list called  $PL$ . For example,  $PL = \{ U \ V \}$ . Make sure each of the variables in  $PL$  have been purged. Store the components of the  $F$  function in a list  $FL$ . For instance, in the example given above  $FL = \{ '2*U^2+V^2-9' \ 'U^2 + V^2 - 5' \}$  where  $U$  replaces  $y_1$  and  $V$  replaces  $y_2$ . Then execute the program **DER** below:

Subprogram Name:   **DER**

Purpose:               Creates list **JL** puts the **FL** functions on the  
stack and executes **DERA**  $M$  times.

<< { } 'JL' STO FL OBJ→ 1 SWAP START DERA NEXT>>

Program DER calls the subprograms DERA and DERB.

Subprogram Name: **DERA**

Purpose: Creates M -1 more copies of the first element on the stack for use in the next subprogram.

```
<< 1 M 1 - START DUP NEXT DERB >>
```

Subprogram Name: **DERB**

Purpose: Takes M copies of a function in FL, creates the derivatives with respect to each parameter in PL and stores them in JL.

```
<<1 M FOR I PL I GET ∂ M 1 + I - ROLLD NEXT  
M →LIST JL + 'JL ' STO>>.
```

At this point, for  $m = 2$ ,  $JL = \{F_{1u}(u,v) \ F_{1v}(u,v) \ F_{2u}(u,v) \ F_{2v}(u,v)\}$ . Now store the values of the variables in PL at  $Y_c$  (e. g.  $U = -2, V = -1$ ) and create matrix JMAT with a program called JEV given by

```
<< JL OBJ→ 1 SWAP START →NUM M SQ ROLLD NEXT  
{M M} →ARRAY 'JMAT' STO >>
```

At this point we have constructed the matrix JMAT. There is a straight forward procedure for finding the eigenvalues of JMAT. See the next chapter. For  $m = 2$ , the eigenvalues are the roots of the quadratic polynomial

$$\lambda^2 - (JMAT[1,1] + JMAT[2,2])\lambda + (JMAT[1,1]*JMAT[2,2] - JMAT[1,2]*JMAT[2,1]).$$

Finding critical points is not always easy. Newton's method for solving simultaneous nonlinear equations may be used to find critical points of a system if an approximate location  $y_0 = \text{column } [u_0 \ v_0]$  of the critical point is known. Then better approximations of the critical point may result from one or more applications of the following algorithm:

$$y_n = y_0 - J^{-1}(y_0) F(y_0), \quad y_n \rightarrow y_0.$$

The same programs listed above can be used to create the JL list for the components of the J matrix. We need additional programs to calculate the  $F(y_0)$  vector. The program FEV that will be used to create the vector FVEC is given by

```
<< FL OBJ→ 1 SWAP START →NUM M ROLLD NEXT
      {M} →ARRAY 'FVEC' STO >>.
```

Put an approximation of the critical point  $[U \ V]$  on the stack and execute the program NWTN given by

```
<< DUP OBJ→ DROP 'V' STO 'U' STO JEV FEV FVEC JMAT / >>.
```

At this point you have an incremental vector  $[U - U_n, V - V_n]$  on the first level of the stack and the old vector  $[U, V]$  on the second level. If the incremental vector is sufficiently small, create the new vector  $[U_n, V_n]$ , by the command  $-$  (a minus command). If not, execute  $-$ , then NWTN again, etc.

EXERCISE 7.13: Find a critical point of the system

$$du/dt = \sin u + \cos v - u, \quad dv/dt = \cos u - \sin v - v$$

near  $u = 1.9$  and  $v = .2$ , and determine the eigenvalues of the variational matrix.

(Answer  $u = 1.9235$ ,  $v = -.17315$ ,  $\lambda = -1.66 \pm i .244$ )

**EXERCISE 7.14:** Find a critical point of the system

$$du/dt = u - \sin u * \cosh v, \quad dv/dt = v - \cos u * \sinh v$$

near  $u = 7$  and  $v = 2.5$ , and determine the eigenvalues of the variational matrix.

(Answer  $u = 7.49768$ ,  $v = 2.76868$ ,  $\lambda = -1.79 \pm i 7.4$ )

How does one find starting values for such a procedure? If the equilibrium is attracting, then for a variety of initial conditions the output of G.12 will indicate an approximate location. If the equilibrium is repelling, then running the system backwards in time will yield the approximate location for many initial conditions. If the equilibrium is neither attracting or repelling, then the same procedure will work if care is used in choosing the initial conditions.

Recall that the Runge-Kutta Feldberg algorithm attempts to set a step size for which the perceived error is below a tolerance level. There are cases for which this algorithm is not efficient: the step selected is too small and too much time is required to proceed from the initial time to a desirable termination. We have previously discussed systems of the form

$$dy_1/dt = a_{11}y_1 + a_{12}y_2, \quad dy_2/dt = a_{21}y_1 + a_{22}y_2.$$

The failure of the default algorithm occurs for such systems when the eigenvalues  $\lambda_1$ ,  $\lambda_2$  of the matrix  $A$  made from the coefficients are both negative and  $\lambda_1/\lambda_2$  is a large number. This indicates that there are two solutions of the differential equation that approach zero as time increases at widely differing rates. Such a system of differential equations is called *stiff* and Hewlett Packard has provided a second algorithm to handle such cases. Nonlinear systems can also be stiff. For example, a system  $dy/dt = F(y)$  which has an equilibrium  $y_c$  for which the matrix  $J(y_c)$  discussed above has eigenvalues with  $\lambda_1/\lambda_2$  large is stiff in the neighborhood of  $y_c$ .

An algorithm for a stiff system is somewhat less efficient than the default algorithm when operating on a nonstiff case. Consequently Hewlett Packard's alternate differential equation program attempts to use the default algorithm whenever possible and switches to a stiff algorithm when stiffness is 'detected'.

To execute the alternate differential equation program, the user must provide a program F for the function  $F(y)$ , a program for  $J(y)$  and a program for  $\partial F/\partial t$ . We will illustrate for the problem

$$y_1' = y_2, \quad y_2' = -1000 y_1 - 1001 y_2.$$

The matrix J here does not depend on y: a program for J is `<< 0 1 -1000 -1001 {2 2} →ARRY >>`. A program for  $\partial F/\partial t$  is `<< 0 0 2 →ARRY >>`. We store these programs under the names FNY and FNT respectively. The following adaptation of G.01 is constructed for this problem. The reader should recall G.01 and edit. Note that the stack command `{T Y FN FNY FNT}` replaces `{T Y FN}` in the default algorithm and that the stack input to RRKSTEP consist of four elements. The last element is an indicator variable (in this case 2) which determines the method to be used.

EXERCISE 7.15: Use IN.FN to store an appropriate function FN. Store FNY and FNT as given above. Use IN.PP to set XRNG to 0 1 and YRNG to 0 1. Put the entries `0 [1 -1 ] 1` on the stack and execute GS.01 (see next page). An alternative is to use the form provided by HP for plotting the solution of a differential equation. To do this enter the FN function as given above for F and check the STIFF box. Then enter the FNY and FNT functions given above in the  $\partial F/\partial Y$  and  $\partial F/\partial T$  boxes respectively. The exact solution is  $y_1 = e^{-t}$ ,  $y_2 = -e^{-t}$ . Use the Function mode to overlay the solution as an accuracy check.

Program Name: **GS.01**

Purpose: Generate a T Y(1) graph of the solution to T<sub>f</sub>.

Stored Quantities: XRNG YRNG FN FNY FNT TOL HS

Input level 3 level 2 level 1

T<sub>0</sub> vector Y<sub>0</sub> T<sub>f</sub>

The output stack is empty, the variables T and Y contain updated values

```
<< { # 0d # 0d } PVIEW DRAX 3 ROLLD 'Y' STO 'T' STO
→ TF << { T Y FN FNY FNT } TOL HS 0 T 'Y(1)' EVAL R→C
5 ROLLD DO RRKSTEP T 'Y(1)' EVAL R→C DUP 7 ROLLD
6 ROLL LINE SWAP DUP 3 ROLLD T + TF UNTIL > END
SWAP DROP TF T - SWAP RRKSTEP T 'Y(1)' EVAL R→C
DUP 7 ROLLD 6 ROLL LINE SWAP DROP TF T - SWAP
RRKSTEP T 'Y(1)' EVAL R→C 6 ROLL LINE 4 DROPN >>
PICTURE >>
```

**EXERCISE 7.16:** Use the calculator to graph several solutions in the x y plane of the 'non-stiff' system

$$\frac{dx}{dt} = x(1 - x - y), \quad \frac{dy}{dt} = y(.5 - .75x - .25y)$$

showing XRNG  $0 \leq x \leq 1.5$ , and YRNG  $0 \leq y \leq 2.5$ . For  $x(0) = .1$ ,  $y(0) = .2$ , plot for  $0 \leq t \leq 25$ , for  $x(0) = .1$ ,  $y(0) = .3$  plot for  $0 \leq t \leq 15$ , for  $x(0) = y(0) = 1.5$  plot for  $0 \leq t \leq 15$ , for  $x(0) = 1.5$ ,  $y(0) = 1.0$ , plot for  $0 \leq t \leq 20$  and for  $x(0) = 1.5$ ,  $y(0) = .8$  plot for  $0 \leq t \leq 20$ . Here notice that  $(x, y) = (.5, .5)$ ,  $(x, y) = (0, 2)$  and  $(x, y) = (1, 0)$  are equilibrium solutions.

# 8

## LINEAR SYSTEMS OF DIFFERENTIAL EQUATIONS WITH CONSTANT COEFFICIENTS

In this chapter we consider linear systems of differential equations of the form  $y' = Ay + f(t)$  where  $y$  and  $f(t)$  are vectors with, say,  $n$  components and  $A$  is an  $n$  by  $n$  matrix. Solutions can be constructed from the eigenvalues and eigenvectors of  $A$ . There are built-in programs in the HP-48G calculator for these eigenvalues and eigenvectors. However, a differential equations student may wish to know just how these quantities could be calculated. Consequently, we will present several special programs to illustrate steps involved in obtaining eigenvalues and eigenvectors. We recommend that beginning students use these special programs at first to become comfortable with the mathematical concepts then use the built-in programs to avoid the computational pitfalls that are sometimes encountered.

### 8.1 HOMOGENEOUS SYSTEMS

Consider first the vector problem  $dy/dt = Ay$ . Here we want to find all solutions of the differential equation. It is readily shown that if  $n$  independent vector functions satisfying the differential equation can be determined and a matrix  $Y(t)$  is constructed with these columns, then all solutions have the form  $Y(t)c$  where  $c$  is a vector with  $n$  components. The "educated guess"  $y(t) = e^{\lambda t}v$  (here  $y(t)$  and  $v$  are vectors) leads to the  $n$ th order polynomial equation  $\det(A - \lambda I) = 0$  which is called the eigenvalue or characteristic equation, and to the problem of determining nontrivial solution vectors  $v$  to the problem  $(A - \lambda I)v = 0$  (where  $\lambda$  is a solution to the eigenvalue equation). Thus the problem breaks into several parts: (1) find the eigenvalue equation, (2) find the solutions of the eigenvalue equation, (3) for each



solution  $\lambda$ , find a corresponding eigenvector  $v$ , and (4) assemble the matrix  $Y(t)$ . We will illustrate the solution process first for  $n = 2$  and then for  $n = 3$  component systems. Then we will outline a procedure that uses the calculator's built-in routine for eigenvalues and eigenvectors for all  $n \geq 2$ . EXAMPLES/EXERCISES are given.

A calculator program to display the eigenvalue equation for a 2 by 2 matrix is:

Program Name: **EIG2**  
 Purpose: Display the eigenvalue equation.  
 Stored Quantities: 2 by 2 matrix A  

$$\ll 'X' \text{ PURGE } A \text{ DET } 8 \text{ RND} \rightarrow D1 \ll 'X^2 - (A(1,1) + A(2,2))*X + D1' \text{ EVAL } \gg \gg$$

**EXERCISE 8.1:** Find the eigenvalue (or characteristic) equation for the matrices

$$\begin{bmatrix} -3 & 4 \\ -2 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ -1000 & -1001 \end{bmatrix}$$

A calculator program to display the eigenvalue equation in the 3 by 3 case is:

Program Name: **EIG3**  
 Purpose: Display the eigenvalue equation  
 Stored Quantities: 3 by 3 matrix A  

$$\ll 'X' \text{ PURGE } A \text{ DET } 8 \text{ RND} \rightarrow D1 \ll 'X^3 - (A(1,1) + A(2,2) + A(3,3))*X^2 + (A(1,1)*A(2,2) - A(1,2)*A(2,1) + A(1,1)*A(3,3) - A(1,3)*A(3,1) + A(2,2)*A(3,3) - A(3,2)*A(2,3))*X - D1' \text{ EVAL } \gg \gg$$

The program will display the eigenvalue equation as a cubic in X.

**EXERCISE 8.2:** Find the characteristic equation for the matrices

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 1 & 0 & 1 \\ 4 & -4 & 5 \end{bmatrix}, \quad A = \begin{bmatrix} -1 & -6 & 3 \\ 3 & 8 & -3 \\ 6 & 12 & -4 \end{bmatrix}, \quad A = \begin{bmatrix} -5 & -8 & -12 \\ -6 & -10 & -10 \\ 6 & 10 & 13 \end{bmatrix}.$$

(The first matrix has the eigenvalue equation  $\lambda^3 - 6\lambda^2 + 11\lambda - 6$ .)

We can find the roots of the eigenvalue equation simply by executing the PROOT program on the HP-48G calculator (left-shift SOLVE, then POLY) — see below — or by storing the equation and using the DRAW and/or SOLVR programs. You may have to try several settings of the plot parameters XRNG, YRNG.

**EXERCISE 8.3:** Find the eigenvalues of the matrices given in EXERCISE 8.2.

(Eigenvalues for the first matrix are 1, 2, 3.)

Consider the matrix

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

The eigenvalue equation in the variable x is  $x^3 - 3x - 1$ . A simple way to obtain the roots on the HP-48G is to press  $\boxed{\rightarrow}$  SOLVE, move to Solve poly... and press OK. Enter the vector of coefficients [1 0 -1 -1] and press OK and SOLVE to get all roots. (You may want to go to EDIT MODES 3 FIX to see all the roots.) Another way to obtain the roots is to use the ROOT command (under FCN on the graphics screen) after plotting the polynomial from -2 to 2. A root is  $x = 1.3247\text{---}$ . If we divide the polynomial  $x^3 - x - 1$  by  $(x - 1.3247\text{---})$  we obtain the quotient  $x^2 + 1.3247\text{---}x + (1.3247\text{---})^2 - 1$ . Zeros of this quadratic are complex eigenvalues. At this point the x has a value stored in it. To avoid confused notation we take an extra step: bring the

value in  $x$  to the stack and store it in  $R$ . Now place ' $x^2 + r*x + (r^2-1)$ ' on the stack, and key in the command ' $X$ ' PURGE. You now have the desired quadratic on the stack, enter ' $X$ ' and execute QUAD (on the SYMBOLIC menu). Follow the usual procedure for the QUAD program to obtain the roots  $-.662 \pm i .563$ .

**EXERCISE 8.4:** Determine the eigenvalues for each of the matrices

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 4 & 3 & 0 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 3 & 0 \end{bmatrix}, \quad A = \begin{bmatrix} -11 & -8 & -12 \\ 2 & 1 & 4 \\ 6 & 4 & 5 \end{bmatrix}.$$

When an eigenvalue  $\lambda$  is determined, the matrix  $(A-\lambda I)$  is singular and the linear system solver is not appropriate to solve the equation  $(A-\lambda I)v = 0$ . Place  $(A-\lambda I)$  on the stack and use the programs named PIV and ROKL given below to obtain the Gauss-Jordan echelon form to determine the row space of  $(A-\lambda I)$  and nontrivial solution vectors  $v$ . Alternately you can use the program RREF on the HP-48G (see below).

Program Name:	<b>PIV</b>	(Adapted from D. R. LaTorre)
Purpose:	Gauss pivot on element $K L$	
Input:	Matrix <b>A</b> , integers <b>K L</b>	Output: Altered matrix <b>A</b>
<pre> &lt;&lt; → A K L &lt;&lt; IF 'A(K,L)' EVAL 0 == THEN "PIVOT ENTRY IS 0" ELSE A SIZE 1 GET → M &lt;&lt; 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 * &gt;&gt; 8 RND END &gt;&gt; &gt;&gt;                     </pre>		

Program Name:       **ROKL** (Adapted from D. R. LaTorre)  
 Purpose:             Interchange rows K and L  
 Input: Matrix A, integers K L Output: Altered matrix A

```

    << → A K L << A SIZE 2 GET → N << A 1 N FOR I
    'A(K,I)' EVAL { L I } SWAP PUT NEXT 1 N FOR J 'A(L,J)'
    EVAL { K J } SWAP PUT NEXT >> >> >>
  
```

Notice that the programs PIV and ROKL given above are valid for any size square matrix.

**EXAMPLE:** The first matrix in the EXERCISE 8.2 has eigenvalues 1, 2, and 3.

For  $\lambda = 1$  an equation for  $v$  is

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

If this matrix is placed on the stack and the command 1, 2 ROKL (to interchange rows) is given we get

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

Next give the command 1,1 PIV (creating 0's in the first column) to get

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

Now the command 2,2 PIV gives

$$\begin{bmatrix} 1 & 0 & .5 \\ 0 & 1 & -.5 \\ 0 & 0 & 0 \end{bmatrix}.$$

The solution relations  $v_1 = -.5 v_3$ ,  $v_2 = .5 v_3$  result: i. e.,  $v = [-1, 1, 2]$  or any nonzero multiple of this vector. Alternately the command RREF (on the HP-48G) will accomplish the same result as the ROKL and PIV commands. Similarly for  $\lambda = 2$ , we find that any multiple of  $v = [-2, 1, 4]$  is a corresponding eigenvector; for  $\lambda = 3$ , we find that any multiple of  $v = [-1, 1, 4]$  is a corresponding eigenvector.

**EXAMPLE:** The matrix

$$A = \begin{bmatrix} 6 & 7 & 8 \\ -2 & 0 & -2 \\ -4 & -6 & -6 \end{bmatrix}$$

has eigenvalues  $\lambda = -2$  and  $1 \pm i$ . The procedure shown above gives the eigenvector  $v = \text{column } [1, 0, -1]$  corresponding to  $\lambda = -2$ . For  $\lambda = 1 + i$ , the matrix  $A - \lambda I$  is

$$A = \begin{bmatrix} (5,-1) & 7 & 8 \\ -2 & (-1,-1) & -2 \\ -4 & -6 & (-7,-1) \end{bmatrix}.$$

When we use RREF (or 1 1 PIV, then 2 2 PIV) we obtain

$$\begin{bmatrix} (1, 0) & (0, 0) & (1, -.5) \\ (0, 0) & (1, 0) & (.5, .5) \\ 0 & 0 & 0 \end{bmatrix}.$$

This leads to an eigenvector  $v = \text{column } [(-1, .5), ((-.5, -.5), 1]$ . Recall that for the conjugate eigenvalue, there is a eigenvector conjugate to this vector  $v$ .

The next step is to assemble a fundamental matrix of solutions  $Y(t)$  that has as its columns the vector solutions determined above. For the first matrix in EXERCISE 8.2 we determined eigenvalues and corresponding eigenvectors in the example just after the ROKL program. Thus

$$Y(t) = \begin{bmatrix} -e^t & -2e^{2t} & -e^{3t} \\ e^t & e^{2t} & e^{3t} \\ 2e^t & 4e^{2t} & 4e^{3t} \end{bmatrix}.$$

The solution of  $y' = Ay$ ,  $y(0) = \text{column}[1 \ 3 \ -5]$  is  $y(t) = Y(t)Y^{-1}(0) \text{column}[1 \ 3 \ 5]$ .

For the matrix example given just above the preceding paragraph (one real and a pair of complex eigenvalue) we proceed as follows. If a matrix  $A$  has eigenvalues  $\lambda = \alpha \pm \beta i$  and corresponding eigenvectors  $c = a \pm ib$ , then by adding the exponential solutions obtained it is known that the quantities  $e^{\alpha t} (\cos \beta t \ a - \sin \beta t \ b)$  and  $e^{\alpha t} (\sin \beta t \ a + \cos \beta t \ b)$  are real valued solutions of the differential equation  $y' = Ay$ . Consequently, for this example we get the fundamental matrix of solutions

$$Y(t) = \begin{bmatrix} -e^t (\cos t + .5 \sin t) & e^t (-\sin t + .5 \cos t) & e^{-2t} \\ .5e^t (-\cos t + \sin t) & -.5e^t (\sin t + \cos t) & 0 \\ e^t \cos t & e^t \sin t & -e^{-2t} \end{bmatrix}.$$

**EXERCISE 8.5:** Find a fundamental matrix of solutions of  $dy/dt = Ay$  for

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

When there is a eigenvalue  $\lambda$  of multiplicity two, either there are two independent eigenvectors  $c$  such that  $(A - \lambda I)c = 0$  or there is a solution of the form  $y(t) = e^{\lambda t} (vt + d)$ . In the latter case, we derive the following requirements by

substitution:  $(A - \lambda I) v = 0$  and  $(A - \lambda I) d = v$ . To determine the vector  $d$  we can augment the matrix  $(A - \lambda I)$  with the additional column  $v$  and use Gauss elimination to determine  $d$ . For

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

$\lambda = -2$  is a eigenvalue of multiplicity 2 and  $\lambda = -1$  is a simple eigenvalue. The eigenvectors corresponding to  $\lambda = -2$  are multiples of  $v = \text{column } [1, -1, 2]$  and the eigenvectors corresponding to  $\lambda = -1$  are multiples of  $v = \text{column } [1, -1, 3]$ . The equation  $(A + 2I) d = \text{column } [1, -1, 2]$  has a solution  $d = \text{column } [0, 1, 0]$ . (Such a solution vector is easily obtained on the calculator, first by calculating  $(A + 2I)$ , augmenting the matrix with the column  $[1, -1, 2]$  then using **RREF** on the HP-48G or **PIV** as listed above to obtain  $d$ .) For this matrix  $A$  we have a fundamental matrix of solutions

$$Y(t) = \begin{bmatrix} e^{-t} & e^{-2t} & te^{-2t} \\ -e^{-t} & -e^{-2t} & (1-t)e^{-2t} \\ 3e^{-t} & 2e^{-2t} & 2te^{-2t} \end{bmatrix}.$$

The matrix eigenvalues for

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

are  $\lambda = -3$  (multiplicity 2) and  $\lambda = -2$ . The eigenvectors corresponding to  $\lambda = -3$  are linear combinations of  $c = \text{column } [1, -1, 0]$  and  $c = \text{column } [-3, 0, 2]$ . The eigenvectors corresponding to  $\lambda = -2$  are multiples of  $c = \text{column } [1, 0, -1]$ . A fundamental matrix of solutions is

$$Y(t) = \begin{bmatrix} e^{-2t} & e^{-3t} & -3e^{-3t} \\ 0 & -e^{-3t} & 0 \\ -e^{-2t} & 0 & 2e^{-3t} \end{bmatrix}.$$

**EXERCISE 8.6:** Find a fundamental matrix of solutions for the system  $y' = Ay$  for each of the following matrices:

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} -3 & 1 & 0 \\ 0 & -3 & 1 \\ 4 & -8 & 2 \end{bmatrix}, \begin{bmatrix} -1.25 & -.5 & .75 \\ .5 & -1 & .5 \\ .25 & .5 & -1.75 \end{bmatrix}.$$

We wish to present a program which accepts an  $n$  by  $n$  matrix as input and generates its eigenvalue equation. There is an algorithm for the coefficients of this equation which combines many subdeterminants to form the coefficients. Such an algorithm seems cumbersome for the calculator; however another less well known algorithm involves products and sums of  $n$  by  $n$  matrices and the computation of the traces of some of these matrices, something this calculator does with little trouble. The following algorithm is taken from Cullen, *Linear Algebra with Applications*, Scott Foresman and Company, 1988: Let  $A$  be an  $n$  by  $n$  matrix, set  $B_0 = I$  and then for  $k = 1, 2, \dots, n$  let

$$A_k = AB_{k-1}, c_k = -(1/k) \operatorname{tr}(A_k), B_k = A_k + c_k I.$$

Then the characteristic polynomial is given by

$$\lambda^n + c_1 \lambda^{n-1} + c_2 \lambda^{n-2} + \dots + c_{n-1} \lambda + c_n.$$

The following program will generate the coefficients in the eigenvalue equation:



Program Name	<b>CHAR</b>
Purpose	Find the eigenvalue equation for a matrix in level 1 on the stack
Input stack: square matrix	Output stack: list of the coefficients in characteristic equation
<pre> &lt;&lt; DUP SIZE 1 GET { 1 } → mtx n poly &lt;&lt; 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 &gt;&gt; &gt;&gt;                     </pre>	

**EXAMPLE:** Place the matrix

$$A = \begin{bmatrix} 3 & 4 & 6 & 4 \\ 1 & -1 & 1 & 1 \\ 6 & 8 & 7 & 8 \\ -11 & -12 & -15 & -14 \end{bmatrix}$$

on the stack and execute **CHAR**. You should receive output { 1 5 13 19 10 }, meaning that the eigenvalue equation is  $\lambda^4 + 5\lambda^3 + 13\lambda^2 + 19\lambda + 10 = 0$ . This equation has two real zeros,  $\lambda = -1$  and  $\lambda = -2$ . Dividing  $\lambda^2 + 3\lambda + 2$  into the eigenvalue equation gives a factor  $\lambda^2 + 2\lambda + 5$  so  $\lambda = -1 \pm i$  are two remaining eigenvalues. The procedure given above for 3 by 3 matrices extends to n by n matrices and therefore a fundamental matrix of solutions is

$$Y(t) = \begin{bmatrix} 4e^{-t} & 0 & e^{-t} \cos 2t & e^{-t} \sin 2t \\ e^{-t} & e^{-2t} & 0 & 0 \\ -2e^{-t} & 0 & -e^{-t} \sin 2t & e^{-t} \cos 2t \\ -2e^{-t} & -e^{-2t} & e^{-t} (\sin 2t - \cos 2t) & -e^{-t} (\cos 2t + \sin 2t) \end{bmatrix}.$$

Recall that an HP-48G calculator has a program to obtain solutions of a polynomial equation. After obtaining the characteristic equation using **CHAR**, the roots of the equation can be determined using the program **PROOT** located in the **POLY** directory on the **SOLVE** menu. The output of **CHAR** is a list of the coefficients. This list should be converted to an array by using the keystrokes **PRG TYPE OBJ**→ **→ARRAY** before using **PROOT**.

**EXERCISES 8.7:** Find a fundamental matrix of solutions for  $y' = Ay$  when

$$A = \begin{bmatrix} -2.5 & 2.5 & -3.5 & -.5 \\ 1 & 2 & 2 & 1 \\ 5 & 4 & 6 & 6 \\ -4.5 & 8.5 & -5.5 & -7.5 \end{bmatrix}.$$

We noted earlier that Hewlett Packard has provided professional programs to calculate the eigenvalues and eigenvectors of  $n$  by  $n$  matrices ( $n \geq 2$ ). These programs are located by pressing the **MTH MATR NXT** keys. **EGV** determines the eigenvalues and eigenvectors of the matrix on level 1, **EGVL** determines only the eigenvalues.

**EXAMPLE:** Place the matrix

$$A = \begin{bmatrix} 3 & 4 & 6 & 4 \\ 1 & -1 & 1 & 1 \\ 6 & 8 & 7 & 8 \\ -11 & -12 & -15 & -14 \end{bmatrix}$$

on the stack and execute **EGV**. You should receive output  $[(-1, 2) (-1, -2) (-1, 0) (-2, 0)]$  for the eigenvalues and output for corresponding eigenvectors

$$\begin{bmatrix} (-.5, .5) & (-.5, -.5) & (1, 0) & (0, 0) \\ (0, 0) & (0, 0) & (.25, 0) & (1, 0) \\ (-.5, -.5) & (-.5, .5) & (-.5, 0) & (0, 0) \\ (1, 0) & (1, 0) & (-.5, 0) & (-1, 0) \end{bmatrix}.$$

The first two eigenvalues are complex  $-1 \pm 2i$  and have conjugate eigenvectors which are the first two columns of the matrix. The procedure given above for 3 by 3 matrices extends to  $n$  by  $n$  matrices and therefore a fundamental matrix of solutions is

$$Y(t) = \begin{bmatrix} 4e^{-t} & 0 & e^{-t} \cos 2t & e^{-t} \sin 2t \\ e^{-t} & e^{-2t} & 0 & 0 \\ -2e^{-t} & 0 & -e^{-t} \sin 2t & e^{-t} \cos 2t \\ -2e^{-t} & -e^{-2t} & e^{-t} (\sin 2t - \cos 2t) & -e^{-t} (\cos 2t + \sin 2t) \end{bmatrix}.$$

**EXERCISE 8.8:** Find a fundamental matrix of solutions for  $y' = Ay$  when

$$A = \begin{bmatrix} -2.5 & 2.5 & -3.5 & -.5 \\ 1 & 2 & 2 & 1 \\ 5 & 4 & 6 & 6 \\ -4.5 & 8.5 & -5.5 & -7.5 \end{bmatrix}.$$

Remark on EXERCISE 8.8. Suppose you use EGV on the HP-48G to obtain eigenvectors on stack level two and eigenvalues on level one. We note that the first eigenvalue is approximately 5.964. If we want to find an eigenvector corresponding to  $\lambda = 5.964$  with fourth component equal to 1 we can proceed as follows: Bring the matrix of eigenvectors to stack level one and create a copy by pressing ENTER. Press MTH MATR and execute the command 1 COL-. You will now have the first column of the eigenvector matrix on level one (and the eigenvector matrix without column 1 on level two). Create a copy of this column by pressing ENTER. Then execute 4 GET and / to bring the fourth element of the column to stack level one and to then divide the eigenvector by its fourth component to get [6.049, -8.871, -21.106, 1] as an eigenvector corresponding to the first eigenvalue.

## 8.2 NON-HOMOGENEOUS SYSTEMS

If the functions in a vector  $f(t)$  are elementary, we can use the method of undetermined coefficients to construct a particular solution to  $y' = Ay + f(t)$ . The computation of the coefficients will require the solution of linear algebraic equations. For more complicated functions  $f(t)$  to obtain solutions of the nonhomogeneous equation suppose that a fundamental matrix  $Y(t)$  of solutions for the associated homogeneous equation is known (so  $Y'(t) = AY(t)$ ). It is easy to see that

$$y(t) = Y(t)c + \int_0^t Y(t-s) Y^{-1}(0) f(s) ds$$

is a solution of the nonhomogeneous system for any vector  $c$ . In the general case a program that uses the numerical integration capability of the calculator can produce values at various times  $t$  for the components of the integral listed above.

**EXAMPLE:** Suppose  $A$  is the matrix given by

$$A = \begin{bmatrix} -.85 & .85 & -2.05 \\ .1 & 0 & -.9 \\ .55 & 1.5 & -.25 \end{bmatrix}$$

(a) We calculate the eigenvalues (one real and a pair of complex conjugate eigenvalues, viz.  $\{-.786... , -.1568... \pm i 1.527...\}$ ) and corresponding eigenvectors ( $v_1 = \text{column}\{1, -.3096... , -.159..., \}$ , and  $a \pm ib = \text{column}\{1, .467... \pm i .1985... , -.144... \pm i -.827...\}$ ) for the matrix and determine 3 independent vector solutions  $u(t)$ ,  $v(t)$ ,  $w(t)$  of the homogeneous system  $dy/dt = Ay$ . We take  $Y(t) = \{e^{\lambda_1 t} v_1, e^{\alpha t}(\cos \beta t a - \sin \beta t b), e^{\alpha t}(\sin \beta t a + \cos \beta t b)\}$ , so that  $Y(0) = [v_1, a, b]$ . (To obtain this matrix put  $A$  on the stack, execute **EGV**, **SWAP** and **OBJ→**, then use the **OBJ→** to put the real and imaginary parts of the first two columns of the matrix of eigenvectors on the stack.

Finally use the (up-arrow)STK and the ECHO commands in the matrix editor to construct  $Y(0)$ .)

(b) For simple forcing functions we can use the method of undetermined "coefficients" to find a particular solution: for example we can choose vectors  $\alpha, \beta$  so that  $y(t) = \alpha \cos 2\pi t + \beta \sin 2\pi t$  is a particular solution of

$$\frac{dy}{dt} = Ay + \begin{bmatrix} 2 \sin 2\pi t \\ 0 \\ -3 \cos 2\pi t \end{bmatrix}$$

as follows: we split the nonhomogeneous term into a column vector  $\{2, 0, 0\} \sin 2\pi t +$  a column vector  $\cos 2\pi t$  then by substituting the prescribed form for  $y$  into the differential equation we get the equations

$$2\pi \beta = A \alpha + \begin{bmatrix} 0 \\ 0 \\ -3 \end{bmatrix}, \quad -2\pi \alpha = A \beta + \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}.$$

If we use the first equation in the second equation we get

$$\alpha = -\text{column} \{ .485\dots, .08\dots, .026 \}, \quad \beta = \text{column} \{ .0317\dots, -.002\dots, -.269 \}.$$

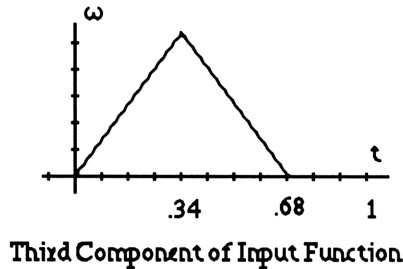
(c) For more complicated forcing functions we use the variation of parameters method to find a particular solution. Suppose

$$f(t) = \text{column} [0, 0, \omega(t)]$$

where  $\omega(t) = \text{IFTE}(t \leq .34, 5t, \text{IFTE}(t \leq .68, 1.7 - 5(t-.34), 0))$  for  $0 \leq t \leq 1$  and  $\omega(t)$  is periodic with period 1. Then if we choose  $c$  so that  $y(0) = y(1)$  in the equation for  $y(t)$  given above we will have a particular solution of  $dy/dt = A y + f(t)$  which is periodic with period 1. This gives the following equation for  $c$  (which forms part of the appropriate initial condition)

$$[Y(0) - Y(1)]c = \int_0^1 Y(1-s)Y^{-1}(0)f(s)ds$$

where for example, we let  $Y(t) = \text{column } [u(t), v(t), w(t)]$  and  $u, v, w$  were obtained in (a). A graph of the input function is shown below.



After obtaining the value of  $c$ , the initial condition  $Y(0)c$  will produce a periodic response over  $0 \leq t \leq 1$ .

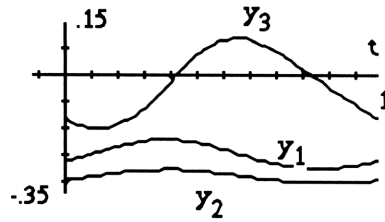
To accomplish this computation we calculate numerically the integral values

$$\begin{aligned} EW &= \int_0^1 e^{\lambda_1(1-s)} \omega(s) ds = .346, \\ SEW &= \int_0^1 e^{\alpha(1-s)} \sin \beta(1-s) \omega(s) ds = .430, \\ CEW &= \int_0^1 e^{\alpha(1-s)} \cos \beta(1-s) \omega(s) ds = .2738. \end{aligned}$$

Then

$$\int_0^1 \omega(s) Y(1-s) \begin{bmatrix} .310.. \\ -.310.. \\ -1.214.. \end{bmatrix} ds = .310..EW v_1 - .310..(CEW a - SEW b) - 1.214 (SEW a + CEW b)$$

which we label as RHS. The equation  $\{Y(0) - Y(1)\}c = \text{RHS}$  gives  $c = [.19735, -.45535, .19735]$  and the appropriate initial condition for the periodic solution is  $Y(0)c = [-.258, -.313, -.129]$ . A graph of the resulting components is shown.



Components of Periodic Solution

As noted above, the change in the third component is pronounced, whereas the other components are influenced to a lesser extent to the change in  $\omega(t)$  at  $t = .34$  and  $.68$ .

**EXERCISE 8.9:** Suppose that  $f(t) = \omega(t)$  column  $\{ 0, 1, 0 \}$  in the problem given above. Obtain a graph of the resulting periodic solution.

**EXERCISE 8.10.** Use the method of undetermined coefficients to determine a particular solution of

$$y' = \begin{bmatrix} -3 & 1 & 0 \\ 0 & -3 & 1 \\ 4 & -8 & 2 \end{bmatrix} y + e^{-2t} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

*Hint:* Try a solution of the form  $y = e^{-2t} \{ t a + b \}$  where vectors  $a$  and  $b$  are to be determined. By substitution the requirements for vectors  $a$  and  $b$  are  $Aa = -2a$  and  $A b = -2 b + a - \text{column } \{1, 1, 0\}$ . We choose  $a = \text{column } \{1, 2, 4\}$ , then  $b = \text{column}\{\sigma, \sigma, \sigma\} - \text{column } \{1, 1, 0\}$  for any  $\sigma$ . The reader might ponder the case where the nonhomogeneous term has the form  $e^{-2t}$  column  $\{g_1, g_2, g_3\}$  in the case  $4(g_1 - g_2) \neq g_3$ .

**EXERCISE 8.11:** A linear model for the angular displacements in a double pendulum (see *The Differential Equation Problem Solver* published by Research and Education

Association, 1978) for a system of two pendulums with lengths  $l_1, l_2$ , a ratio of pendulum masses given by  $\delta = 1 + m_2/m_1$  is given by the differential equations

$$m_1 l_1 \theta_1'' = g\{m_2 \theta_2 - (m_1 + m_2) \theta_1\}$$

$$m_2 (l_1 \theta_1'' + l_2 \theta_2'') = -m_2 g \theta_2.$$

These equations may be recast as

$$l_1 \theta_1'' = g\{m_2/m_1 \theta_2 - \delta \theta_1\}$$

$$(l_1 \theta_1'' + l_2 \theta_2'') = -g \theta_2$$

and by standard elimination techniques, we obtain

$$l_1 l_2 \theta_1^{(iv)} + g(l_1 + l_2) \delta \theta_1'' + g^2 \delta \theta_1 = 0$$

$$\theta_2 = \frac{1}{\delta - 1} \left\{ \frac{l_1}{g} \theta_1'' + \delta \theta_1 \right\}.$$

We note  $\delta > 1$ . Show that for  $\theta_1 = e^{rt}$  the characteristic equation is quadratic in  $r^2$ ; viz.

$$r^2 = \frac{g\delta}{2l_1\tau} \left\{ -(1+\tau) \pm \sqrt{(1+\tau)^2 - \frac{4\tau}{\delta}} \right\}$$

where  $\tau$  is defined by the equation  $l_2 = \tau l_1$ . Test these values of  $r^2$  for values of  $\tau$  and  $\delta$  by putting  $\delta = 1.5$  (i. e.  $m_2 = .5 m_1$ ) and graph the term in brackets for  $\tau$  in the interval  $0 \leq \tau \leq 1$ . These graphs show the roots for  $r^2$  are negative and thus there are two pairs of conjugate purely imaginary roots for  $r$ . If we denote the roots of the characteristic equation as  $r = \pm i\mu, r = \pm i\nu$ , show the solution  $\theta_1$  and  $\theta_2$  will contain terms of the form

$$a_j \cos \mu t + b_j \sin \mu t + c_j \cos \nu t + d_j \sin \nu t, j = 1, 2.$$

with



$$\begin{aligned} a_2 &= \left[ \delta - \frac{l_1 \mu^2}{g} \right] a_1 & b_2 &= \left[ \delta - \frac{l_1 \mu^2}{g} \right] b_1 \\ c_2 &= \left[ \delta - \frac{l_1 v^2}{g} \right] c_1 & d_2 &= \left[ \delta - \frac{l_1 v^2}{g} \right] d_1 \end{aligned}$$

With appropriate initial conditions, we can find terms of the form  $\cos \mu t - \cos vt$  which leads to the form  $\sin \{.5(\mu+v)t\} \sin \{.5(\mu-v)t\}$ .

Alternately we put  $y_1 = \theta_1$ ,  $y_2 = \theta_1'$ ,  $y_3 = \theta_2$ ,  $y_4 = \theta_2'$  to derive the equation  $y' = Ay$  where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{\delta g}{l_1} & 0 & \frac{g(\delta-1)}{l_1} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{g\delta}{\tau l_1} & 0 & -\frac{g\delta}{\tau l_1} & 0 \end{bmatrix}$$

For  $g = 1$ ,  $l_1 = .75$ ,  $\tau = \delta - 1 = .10059..$ , the matrix  $A$  has eigenvalues  $\lambda = \pm i \mu$ ,  $\lambda = \pm i v$  with  $\mu = 4.53817...$ ,  $v = 1.13454..$  and corresponding eigenvectors

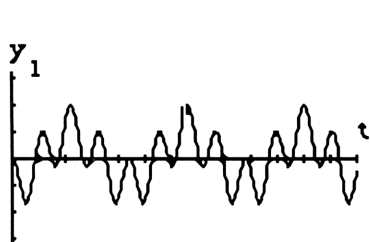
$$\begin{bmatrix} 0 \\ -.035852.. \\ 0 \\ 1 \end{bmatrix} + i \begin{bmatrix} .0079001.. \\ 0 \\ -.22035.. \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ .93526.. \\ 0 \\ 1 \end{bmatrix} + i \begin{bmatrix} -.82435.. \\ 0 \\ -.8814.. \\ 0 \end{bmatrix}$$

We denote the vectors listed above as  $a + i b$ ,  $c + i d$ . There is a solution of  $y' = Ay$

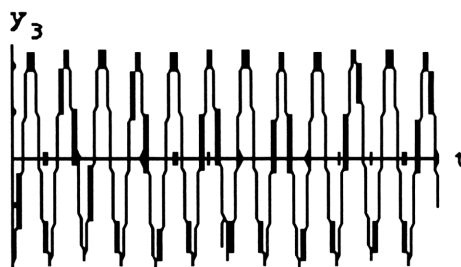
$$y(t) = \sigma \{ \sin \mu t a + \cos \mu t b - b(1) (\sin vt c + \cos vt d) / d(1) \},$$

and we note the first component has the form  $y_1(t) = \sigma b(1) \{ \cos \mu t - \cos vt \}$ . Thus the angular displacement of the first pendulum oscillates with a beats motion. It is

interesting to plot the motion of the angular displacement of the second pendulum which looks deceptively like a regular periodic motion of a sine wave.



Motion of pendulum 1



Motion of pendulum 2

# 9

## MISCELLANEOUS SYSTEMS

This chapter contains a set of applied problems that are appropriate in the study of differential equations. The arrangement of the problems is somewhat random and many, if not most, of the problems can be studied as soon as the concept of solving a vector initial value problem is discussed. The first two problems might occur more naturally just after Chapter 7. We delayed the presentation of these problems so that students could keep the time schedule used in many classes. Some classes use a nonstandard order of topics. For example, a class may choose to study discrete systems early in the course. This chapter includes a short discussion of such systems, but covers only an introduction to the concept of chaos.

Consider the motion of a particle in two dimensions ( $x$  and  $y$ ) in a gravitational field. We assume that the height attained by the particle is relatively low so that gravity is constant, and motion occurs in a plane. Put

$$v(t) = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}, \quad \theta(t) = \tan^{-1} \frac{dy}{dx}(t).$$

The force on the projectile in the direction of the tangent to the trajectory is  $F_1 = T(t) - av^r$  (thrust and air resistance). There are also horizontal and vertical forces. The equations resulting from Newton's law of motion are:

$$m \frac{d^2 x}{dt^2} + \frac{dm}{dt} \frac{dx}{dt} = F_1 \cos \theta - \omega, \quad m \frac{d^2 y}{dt^2} + \frac{dm}{dt} \frac{dy}{dt} = F_1 \sin \theta - m.$$

Here  $\omega$  is a force in the horizontal direction such as wind or an artificially imposed control on the trajectory as described below for rocket flight. Since

$$\frac{dx}{dt} = v \cos \theta, \quad \frac{dy}{dt} = v \sin \theta, \quad (9.1-9.2)$$

$$\begin{aligned} \frac{d^2x}{dt^2} &= \frac{1}{m} (T(t) - av^r) \cos \theta - \frac{\frac{dm}{dt}}{m} \frac{dx}{dt} - \frac{\omega}{m} \\ &= dv/dt \cos \theta - v \sin \theta d\theta/dt \end{aligned}$$

$$\begin{aligned} \frac{d^2y}{dt^2} &= \frac{1}{m} (T(t) - av^r) \sin \theta - \frac{\frac{dm}{dt}}{m} \frac{dy}{dt} - g \\ &= dv/dt \sin \theta + v \cos \theta d\theta/dt \end{aligned}$$

Multiplying the right side of the  $x$  acceleration expression by  $\cos \theta$  and the right side of the  $y$  acceleration expression by  $\sin \theta$  and adding gives

$$\frac{dv}{dt} = \frac{1}{m} (T(t) - av^r) - \frac{\frac{dm}{dt}}{m} v - g \sin \theta - \frac{\omega}{m} \cos \theta \quad (9.3)$$

If we differentiate the expression for  $\theta$  with respect to  $t$  we obtain

$$\frac{d\theta}{dt} = \frac{1}{v} \left( \frac{\omega}{m} \sin \theta - g \cos \theta \right). \quad (9.4)$$

Differential equations 9.1-9.4, together with initial conditions  $x(0) = y(0) = 0$ , and  $v(0)$ ,  $\theta(0)$  prescribed, will give the trajectory  $(x(t), y(t))$ . The reader may note that when the initial velocity is low the initial values of  $\theta$  will be quite sensitive to the  $v(0)$  value.

**EXERCISE 9.1:** (*Artillery shell EXERCISE for the HP-48G*) Take  $r = 2$ ,  $m = 500$  kg,  $a = .04$ ,  $\omega = 0$  and initial condition  $Y = [\theta, v, x, y] = [\theta, 200, 0, 0]$  for several values of

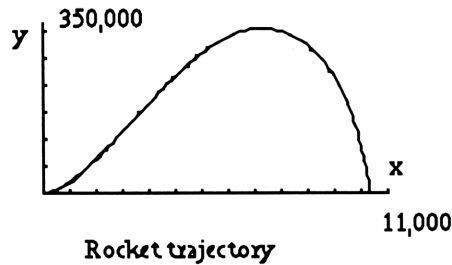
$\theta$ , say between .5 and 1.2. At each elevation, determine the range and maximum height of the shell.

**EXERCISE 9.2:** (*Baseball trajectory EXERCISE for the HP-48G*) Take  $r = 1.2$ ,  $a = .035$ ,  $m = .25$  kg, and  $v(0) = 50$  meters /second. Determine the trajectories that start at  $x = 0$ ,  $y = 0$  and terminate when the ball hits the ground for selected values of  $\theta$  and  $\omega = 0$ , then for  $\omega = \pm 4$  meters/second.

*Rocket flight example for the HP-48G:* We take  $r = 2$ ,  $m_0 = 90,000$  kg,  $M(t) = m_0 - \text{slope } t$  for  $0 \leq t \leq 120$ ,  $= .1m_0 + .05*m_0e^{-\gamma(t-120)}$  for  $t > 120$  where  $\text{slope} = .85 m_0/120 = 637.5$  kg/sec, and  $.05m_0\gamma = 637.5$  or  $\gamma = .85/.05 * 1/120 = .14167$  and  $a = .05$ . A reasonable model for thrust is  $T(t)$  proportional to  $dM(t)/dt$ , but some easy experiments will convince the reader that unless some control is exerted on the horizontal direction, say in the form of providing thrust in this direction, the rocket will soon tilt into the ground with unburned fuel. To prevent this from happening we take  $\omega = -\beta \epsilon dM(t)/dt \cos \theta$  and  $T(t) = -\beta dM(t)/dt (1 - \epsilon \cos \theta)$ . Further we assume that special conditions are placed on the system so that the rocket lifts off and achieves a velocity of 10 meters per second as it clears the liftoff tower. From this point we assume the equations of motion given above apply. For  $\epsilon = .825$  and initial vector  $Y$  with components  $\theta, v, x, y$  having values  $[1.5, 10, 0, 0]$  and  $XRNG = 0$  11,000,  $YRNG = 0$  350,000, we observe the following data:

$t = 120,$	$\theta = 1.555,$	$v = 3,178,$	$x = 1,666,$	$y = 67,800$
$t = 240,$	$\theta = 1.542,$	$v = 711,$	$x = 5,758,$	$y = 316,279$
$t = 300,$	$\theta = 1.302,$	$v = 67,$	$x = 6,878,$	$y = 338,756$
$t = 306,$	$\theta = .3,$	$v = 13,$	$x = 7,000,$	$y = 338,960$
$t = 654.7,$	$\theta = -1.569,$	$v = 1313,$	$x = 10416,$	$y = 150$

The graph of the rocket's trajectory is shown below.



**EXERCISE 9.3 :** Take  $\epsilon = .8$ , the initial  $Y$  to be  $[1.5, 10, 0, 0]$ , and plot the trajectory. Record the values of  $Y$  at times  $t = 120, 240, 300, 660$ . Then take  $\epsilon = .8125$ , initial  $Y = [1.5, 10, 0, 0]$  and observe the values of  $Y$  for successive time steps to see whether the control will give permit to a full trajectory, i.e., to a trajectory with flight termination after fuel burnout (approximately 150 seconds). To do this place the calculator in 2 FIX MODE, place  $\{T \ Y \ FN\}$  on stack level five, a tolerance .005 in level four, starting step size, say .01 in level three, 0 on level two and  $[1.5 \ 10 \ 0 \ 0]$  on level one, and execute the program

`<< 'T' STO 'Y' STO RKFSTEP T Y >> .`

Repeat execution of this program for several times to see that the first component of  $Y$  shown on level one remains near 1.5 and the second component increases at each execution.

**EXERCISE 9.4 (Pursuit Problem):** A rabbit starts at  $(0,1)$  and runs along  $y = 1$  with speed 1. At the same time a dog starts at  $(0, 0)$  and pursues the rabbit with speed 1.3. The dog attempts to point at the rabbit at all times but is constrained by his momentum. That is, for the angle  $z$  between the dog's direction and the  $x$  axis,  $dz/dt$  is restricted. What is the path of the dog? The equations of motion are:

$$\begin{aligned} dx/dt &= 1.3 \cos z, & dy/dt &= 1.3 \sin z, & dz/dt &= -(z - \theta(t,x,y)) \\ x(0) &= 0, & y(0) &= 0, & z(0) &= 2.2. \end{aligned}$$

Here  $\theta(t,x,y)$  is the angle between the dog-rabbit vector and the x axis. We take  $H = .15$ ,  $N = 60$ , plot parameters to show  $-1 \leq x \leq 8$ ,  $-.5 \leq y \leq 2$ , repeatedly apply IULER to the differential equation  $dw/dt = F(t,w)$  and watch the trajectory. Suitable functions for  $F_1(t, x, y, z)$ ,  $F_2(t, x, y, z)$ , and  $F_3(t, x, y)$  (here  $F(t, w) = \text{column } [F_1, F_2, F_3]$ ) are given by:

**F.N1:** <<  $\rightarrow$  T W '1.3\* COS(W(3))' >>

**F.N2:** <<  $\rightarrow$  T W '1.3\* SIN(W(3))' >>

**F.N3:** <<  $\rightarrow$  T W << 'W(3)' EVAL T 'W(1)' EVAL - DUP SQ 'W(2)' EVAL 1  
- DUP SQ 3 ROLL +  $\sqrt{3}$  ROLLD SWAP DUP 3 ROLLD THTA - NEG >>

**THTA:** << 0 IF  $\geq$  THEN THT1 ELSE THT2 END >>

**THT1:** << 0 IF  $\leq$  THEN SWAP / ACOS ELSE SWAP / ACOS NEG END >>

**THT2:** << 0 IF  $<$  THEN NEG SWAP / ACOS  $\pi \rightarrow$ NUM SWAP - ELSE  
NEG SWAP / ACOS  $\pi \rightarrow$ NUM + END >>

**F.N:** << DUP2 DUP2 F.N1 5 ROLLD F.N2 3 ROLLD F.N3 3  $\rightarrow$  ARRAY >>

An easy modification of the GRAF program can be used to follow the action. The program requires a starting stack of 0, [0, 0, 1.8] and the trajectories of both dog and rabbit are shown dynamically as follows:

<< { # 0d # 0d } PVIEW 0 1 R $\rightarrow$ C 3 ROLLD DRAX 1 N START IULER  
DUP OBJ $\rightarrow$  DROP2 R $\rightarrow$ C PIXON 3 ROLL H 0 R $\rightarrow$ C + DUP PIXON 3  
ROLLD NEXT PICTURE >>

## 9.1 THE LORENTZ EQUATIONS

Consider the problem:

$$dx/dt = \sigma(y-x), \quad dy/dt = (r-z)x - y, \quad dz/dt = xy - bz.$$

where  $\sigma$ ,  $r$  and  $b$  are parameters. This set of equations was proposed by E. Lorentz (1963) in connection with convective heat transfer between the earth's surface and the atmosphere. A point  $(x, y, z)$  represents convection velocities and temperature profile, vertical and horizontal. An equilibrium, or periodic, solution to the system represents predictable behavior. The graphs of particular solutions gave particularly surprising results because most trajectories never seem to approach such predictability. The paper has been one of the seminal studies in the area of chaos.

First the three critical points (that is, points  $(x, y, z)$  where the right sides of the differential equations are zero) are  $(0, 0, 0)$  and  $(\pm(b(r-1))^{.5}, \pm(b(r-1))^{.5}, r-1)$ . The variational matrix at  $(0, 0, 0)$  is  $\begin{bmatrix} -\sigma & \sigma & 0 \\ r & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$ , with system eigenvalues of  $-b$  and  $.5(-(\sigma+1) \pm \sqrt{(\sigma-1)^2 + 4\sigma r})$ .

The variational matrices near the remaining critical points are

$$\begin{bmatrix} -\sigma & \sigma & 0 \\ 1 & -1 & -\delta \\ \delta & \delta & -b \end{bmatrix} : \delta = \pm \sqrt{b(r-1)}.$$

Here, the eigenvalues satisfy the equation

$$\lambda^3 + (\sigma+b+1)\lambda^2 + b(r+\sigma)\lambda + 2\sigma b(r-1) = 0.$$

For  $r > 1$ ,  $b > 0$ ,  $\sigma > 0$  one of the variational eigenvalues near  $(0, 0, 0)$  is positive, so most trajectories starting near  $(0, 0, 0)$  leave that neighborhood. For  $r$  sufficiently



near 1, the eigenvalues of the remaining variational matrices are negative and the corresponding critical points are attracting solutions. When  $r$  is slightly larger, one of the variation eigenvalues has a positive real part and the critical point is repelling. It can be shown that solutions starting near the critical points stay bounded and thus the trajectories have interesting behavior as  $t$  increases.

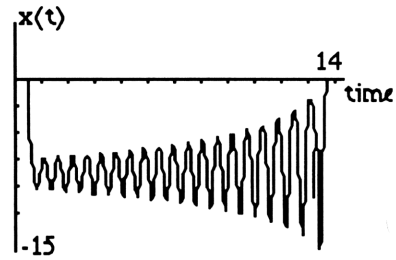
**EXERCISE 9.5:** Compute a solution for initial values  $x(0) = 1$ ,  $y(0) = 1$ ,  $z(0) = 1$  for  $\sigma = 10$ ,  $r = 28$ , and  $b = 8/3$  using the built-in program for  $0 \leq t \leq 15$ . Use viewing box  $-15 \leq x \leq 0$ ,  $-15 \leq y \leq 0$ ,  $20 \leq z \leq 40$  and position of the eye along the vector  $[1, 1, 5]$ . (See Appendix 5 for programs that give three dimensional plots.) Part of a typical trajectory is shown below. A continuation of the trajectory reveals no asymptotic pattern other than a continual looping in two of the  $x, y, z$  octants. The following table shows partial results rounded to two decimals. At time  $t$  approximately .7, the trajectory enters the viewing box, starts in a tight spiral that winds outward, finally leaving the quadrant at about  $t = 14.4$ .

time	x	y	z
.608	-5.07	-8.06	26.44
3.90	-8.67	-10.42	24.76
5.36	-9.87	-8.19	30.65
7.68	-8.80	-11.43	23.55
9.45	-5.24	-6.22	21.38
11.48	-4.79	-6.82	18.16
13.12	-13.16	-18.40	26.71
13.81	-6.50	-.12	8.40
14.37	-15.96	-25.44	26.07
14.55	-5.18	6.93	35.46
14.68	3.76	6.90	25.36



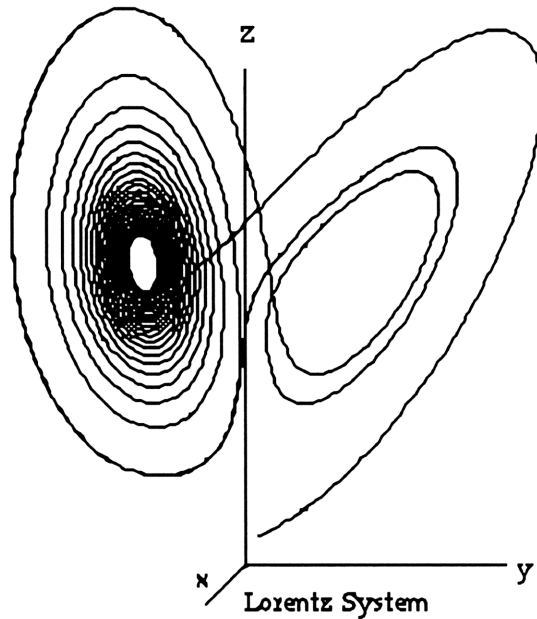
xyz perspective

Lorenz trajectory enters viewing window  
near center of picture winds outward to  $t = 14$



x-t perspective

An extension to later times is shown below. Subsequently the trajectory winds in the right hand portion of the picture making excursions into the left portion. The trajectory never intersects itself so there is layering going on. *Warning:* the development of this trajectory takes considerable time on the HP-48.



## 9.2 EARTH, MOON, SATELLITE MOTION

Consider the model problem where the moon is circling the earth and a satellite is in motion in the plane of the earth-moon orbit. The "restricted three body" problem results when the satellite mass can be ignored when compared to the masses of the moon and earth. (See *Celestial Mechanics*, Part II by S. Sternberg, W. A. Benjamin Company, 1969.) If a rotating coordinate system is used so the coordinates of the earth are  $(-\mu, 0)$ , the coordinates of the moon are  $(1-\mu, 0)$ , and the coordinates of the satellite are denoted by  $(x(t), y(t))$ , then the equations of motion are

$$\frac{dx}{dt} = u, \quad \frac{du}{dt} = 2v + x - \frac{(1-\mu)}{r^3}(x+\mu) - \mu \frac{(x+\mu-1)}{\rho^3}$$

$$\frac{dy}{dt} = v, \quad \frac{dv}{dt} = -2u + y - \frac{(1-\mu)}{r^3}y - \mu \frac{y}{\rho^3}$$

where

$$r^2 = (x + \mu)^2 + y^2, \quad \rho^2 = (x + \mu - 1)^2 + y^2$$

The constant  $\mu$  is a ratio of the masses of the earth and moon ( $= m_p / (m_p + m_m) = 1/82.45$ ). For the "earth-moon" system it has been discovered that a solution with period approximately 6.1922 results from the initial conditions

$$x(0) = 1.2 \quad u(0) = 0, \quad y(0) = 0, \quad v(0) = -1.04936\dots$$

We take the vector  $w = [x, u, y, v]$  and create the subprogram

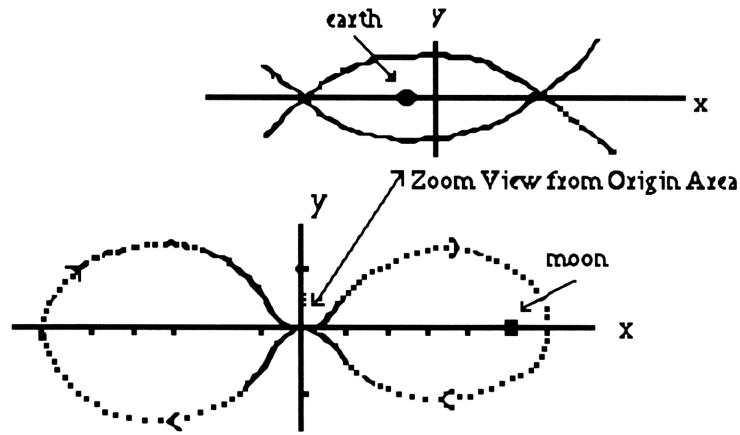
Subprogram Name	<b>RN</b>
Stored Quantities	none
input w:	output: $R1 = r^3$ $R2 = \rho^3$
<< OBJ→ DROP2 SWAP DROP SQ SWAP 82.45 INV + DUP2	
SQ + 1.5 ^ 3 ROLLD 1 - SQ + 1.5 ^ >>	

The program F.N that takes  $w$  to  $F(w)$  can be as follows:

```
<< DUP RN 82.45 INV → R1 R2 MU << OBJ→ DROP 3 ROLLD MU DUP
R2 / SWAP 1 - R1 / - 1 - NEG * SWAP DUP 5 ROLLD 2 * - 3 ROLLD
SWAP DUP MU + 1 - R2 / MU * SWAP DUP 3 ROLLD MU + 1 MU - R1
/ * + - SWAP DUP 2 * 3 ROLL + 3 ROLLD SWAP 4 →ARRY SWAP DROP
>> >>
```

Since this program is complex, we provide a stack status at various program steps:

<u>Instruction</u>	<u>Stack contents</u>
DUP RN 82.45 INV	t w R1 R2 MU
→ R1 R2 MU	t w
OBJ→ DROP	t x u y v
3 ROLLD MU DUP R2 /	t x v u y $\mu$ $\mu/R2$
SWAP 1 - R1 /	t x v u y $\mu/R2$ $(\mu-1)/R1$
- 1 - NEG *	t x v u $y*(-1)[\mu/R2-(\mu-1)/R1 -1]$
SWAP DUP 5 ROLLD 2 * -	t u x v $\{y*(-1)[\mu/R2-(\mu-1)/R1 -1]-2u\}$
3 ROLLD SWAP	t u last equation v x
DUP MU + 1 - R2 / MU *	t u last equation v x $\mu[x+\mu-1]/R2$
SWAP DUP 3 ROLLD	t u last equation v x $\mu[x+\mu-1]/R2$ x
MU + 1 MU - R1 /	t u last equation v x $\mu[x+\mu-1]/R2$ $x+\mu$ $(1-\mu)/R1$
* + - SWAP DUP	t u last equation $x-\mu[x+\mu-1]/R2+(x+\mu)(1-\mu)/R1$ v v
2 * 3 ROLL + 3 ROLLD SWAP	
4 →ARRY SWAP DROP >>	



Satellite near Earth-Moon System

The graphs shown above require considerable execution time. The tolerance parameter in the variable stepsize method was initially set to 0.01 and adjusted when the satellite approaches the earth: later it was reduced back to 0.01, etc.

Another interesting periodic solution occurs for the approximate initial conditions

$$x(0) = 0, \quad u(0) = 1.58, \quad y(0) = 1.2, \quad v(0) = 0.$$

We note that equilibrium points for the restricted three body satisfy

$$u = v = 0, \quad y [1 - (1-\mu)/r^3 - \mu/\rho^3] = 0, \quad x = (1-\mu)(x+\mu)/r^3 + \mu(x-1+\mu)/\rho^3.$$

**PROJECT:** Find the equilibrium solutions of this system and determine their stability properties. Notice that for  $y = 0$ , we need only solve an equation of the form  $g(x) = 0$ , but for other solutions, we need to find the solution of a pair of nonlinear algebraic equations. We can use Newton's method as presented in Chapter 7 for finding such values of  $x$  and  $y$  if we can find a suitable starting point  $x_0, y_0$ . (Two of the equilibrium points are located at  $x = .48787, y = \pm .86603$ .) To determine

the stability, we need to find the eigenvalues of the 4 by 4 variational matrices associated with each critical point solution.

## 9.3 DISCRETE DYNAMICAL SYSTEMS

In some situations, the problem of interest is to determine information concerning the asymptotic behavior (i.e., for large values of  $t$ ) of solutions. This occurs often in differential equations; however we will illustrate this type of problem by introducing a new type of problem.

The sequence  $\{y_n\}_{0}^{\infty}$  where  $y_n$  is given by a recursive function of the form  $y_{n+1} = F(y_n)$ , is called a **discrete dynamical system**. Euler's method, the improved Euler method, and Newton's method for finding roots may give such systems. Concepts such as constant "solutions", attractive or repelling solutions taken from differential equations, are also present in the study of such systems. Discrete dynamical systems (in one dimension) have solutions with more complicated structure than do differential equations. For example,  $y_n = \alpha$  (for all  $n$ ) is called a period one solution,  $y_{\text{odd } n} = \alpha$  and  $y_{\text{even } n} = \beta$  is called a period two solution,  $y_{3m} = \alpha$ ,  $y_{3m+1} = \beta$ ,  $y_{3m+2} = \chi$  is a period three solution, etc. Consider

$$y_{n+1} = (1 + a) y_n - a (y_n)^2 \text{ and } y_0 = .1.$$

We want to regard  $a$  as a parameter and study the effect on the "solution sequence  $\{y_n\}$ " as  $a$  is varied. In particular we want to study  $a = 1.8, 2.3, 2.5$  and  $3$  as well as nearby values of  $a$ .

After several numerical experiments, we find that for  $a = 1.8$  the terms of the sequence approach 1, but for  $a = 2.3$ , the terms of the sequence approach 1.18 for even  $n$  and .69 when  $n$  is odd, etc. This leads us to the following graphical program:

Set the value of  $a$ . Calculate the first 50 terms, then plot the values of the next 100, then change the value of  $a$  and repeat. To get these cases on a single plot, we plot the values of  $y_n$  on the horizontal axis and the values of  $a$  on the vertical axis. The following programs can be used: Store  $A = 1.8$  and  $N = 100$ , set the plot parameters so that  $-0.5 \leq x \leq 1.5$  and  $1.6 \leq y \leq 3.2$ , then enter the programs.

Program Name	<b>DDS</b>
Purpose	Plot the asymptotic value of a discrete system for several values of the parameter $a$
Stored Quantities	FN1 DDS1 DDS2 A N XRNG YRNG
No input:	Output is a graph
<pre>&lt;&lt; { # 0d # 0d } PVIEW .1 1 12 START DDS1 A .1 + 'A' STO       .1 NEXT PICTURE &gt;&gt;</pre>	

Subprograms are:

Subprogram Name	<b>FN1</b>
Purpose	Create the new value of $y$ , given the previous value
<pre>&lt;&lt; DUP SQ A * SWAP A 1 + * SWAP - &gt;&gt;</pre>	

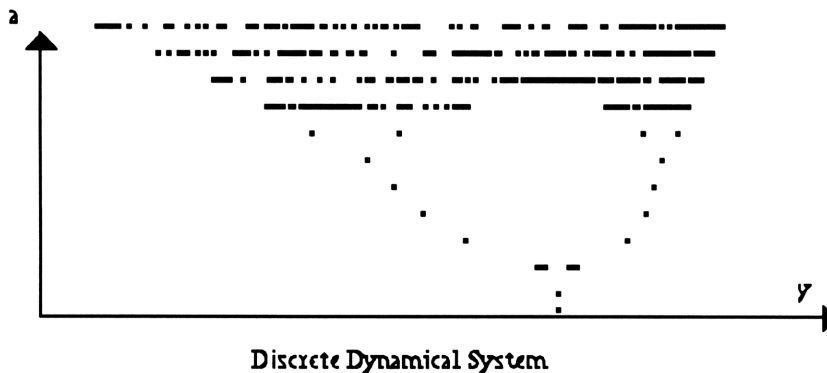
Subprogram Name	<b>DDS1</b>
Purpose	Execute FN1 50 times, call DDS2
<pre>&lt;&lt; 1 50 START FN1 NEXT DDS2 &gt;&gt;</pre>	

Subprogram Name     **DDS2**

Purpose                 Repeat FN1 N times, plot points

<< 1 N START FN1 DUP A R→C PIXON NEXT >>

(The 50 executions of FN1 without graphing allows the sequence to come to "steady state" before the graphing begins.) Execute DDS.



**EXERCISE 9.6.** Change FN1 to repeat the process for the system

$$y_{n+1} = (1 + a) y_n + 2a (\cos y_n - 1) \text{ and } y_0 = .1.$$

An appropriate range of the parameter  $a$  begins at 2.0.

The objects in a discrete dynamical system may be vectors. One example was discussed (without labeling it as a discrete dynamical system) in EXERCISE 9.5. Particular attention has been given to the case when the  $y_n$  are two-dimensional vectors. In this case we can easily plot the vectors  $(y_{n1}, y_{n2})$ ,  $n = 1, 2, \dots$  on a graph. In addition to the cases of an attracting or repelling equilibrium solution, other types



of interesting behavior may occur. The reader should consult [1] or [2] for more details. We will present the example, called the Henon map, in which the point  $(x, y)$  is "mapped" to the point

$$\begin{aligned}\bar{x} &= x \cos \alpha - (y - x^2) \sin \alpha \\ \bar{y} &= x \sin \alpha + (y - x^2) \cos \alpha\end{aligned}$$

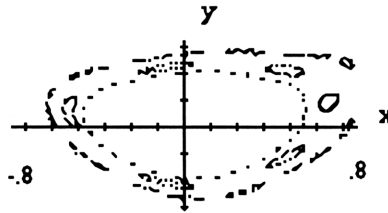
Here  $\alpha$  and the starting values  $(x_0, y_0)$  are parameters. For fixed  $\alpha$ , the trajectory  $(x_n, y_n)$ ,  $n = 0, 1, 2, \dots$  may have significantly different structure for different values of the initial point. Usually trajectories arising from several initial points are shown on the same plot. And different values of  $\alpha$  may result in quite different plots. In our example we take  $\alpha = \cos^{-1}(.4)$ . We modify the GRAF program from Chapter 5 to a program called **GRF.H** as follows:

```
<< { # 0d # 0d } PVIEW DRAX DUP2 R→C PIXON 1 N START HMAP DUP2
      R→C PIXON NEXT PICTURE >>
```

where **HMAP** is given by

```
<< → X Y << ' X*COS(ALFA) - (Y - X^2)*SIN(ALFA)' EVAL
      ' X*SIN(ALFA) + (Y - X^2)* COS(ALFA)' EVAL >>
```

If we set **XRNG** and **YRNG** each to  $-.8 \ .8$  and use  $N = 400$ ,  $x_0 = .63$ ,  $y_0 = .2$  and execute **GRF.H** we obtain a curious picture with six islands. Repeating the program with  $x_0 = .55$ ,  $y_0 = 0$ , then with  $x_0 = .75$ ,  $y_0 = 0$  give two "closed" curves. (The outer curve seems to contain another island.) These three trajectories are shown on the following plot:



**Henon Map for  $\cos \alpha = .4$**

**EXERCISE 9.7:** Set  $\alpha = \text{Cos}^{-1}(-.05)$ , take initial points  $(.32, .9)$ ,  $(.30, .9)$ ,  $(.28, .9)$ ,  $(.30, 0)$  and  $(.2, 0)$  with  $\text{XRNG} = \text{YRNG} = -1 \ 1$ : execute GRF.H.

Next we consider here the case  $z_{n+1} = f(z_n, c)$ ,  $n = 0, 1, 2, \dots$  for given  $z_0$ , where all the  $z$  elements are complex numbers (a two component vector with special algebraic rules) and the complex parameter number  $c$  is given. The purpose is again to focus attention on the asymptotic behavior of the dynamical system solution sequence  $\{z_n\}$ . In particular, we will study systems of the form  $z_{n+1} = z_n^2 + c$ ,  $z_0$  given, where  $c$  will be fixed in each system. For each number  $c$  depending on the starting position  $z_0$ , one of three things can happen: (1)  $\lim |z_n| = \infty$ , (2)  $\lim |z_n| =$  some number, or (3) neither of the above. We show the dependence of the elements of the system on the starting value  $z_0 = \alpha$  by using the notation  $z_n(\alpha)$  for the elements. The "Julia" set for this sequence is the boundary of the set  $A = \{ \alpha : |z_n(\alpha)| \rightarrow \infty \}$ . (Elements on the boundary of this set do not belong to the set.) Many of the elements of the Julia set have a wandering property, that is, they do not have a limit and so their behavior is called chaotic. (Iterates  $f(\alpha)$ ,  $f(f(\alpha))$ ,  $f(f(f(\alpha)))$ ,  $\dots$  wander around the Julia set.)

The following program is to graph members of the Julia set. The procedure of finding lots of members of this set may seem tricky at first because the requirement to be in the set is quite delicate and any roundoff error may cause a sequence  $\alpha$ ,  $f(\alpha)$ ,  $f(f(\alpha))$ ,  $f(f(f(\alpha)))$ ,  $\dots$  beginning with  $\alpha$  in the Julia set to drift out of the set. The

algorithm to be given is based on the property that for any fixed  $c$ , the inverse images of a repelling fixed point belong to the Julia set, that is, for  $w$  in the set, the images  $f^{-1}(w)$ ,  $f^{-1}(f^{-1}(w))$ ,  $f^{-1}(f^{-1}(f^{-1}(w)))$ , ... also belong to the set. (Here  $f^{-1}(w) = \pm\sqrt{w-c}$ .) It can be shown that the latter sequence is stable for this function  $f$ , whereas the sequence of direct images is not stable to roundoff error.

Suppose the mapping  $z \rightarrow f(z) = z^2 + c$  has a fixed point, that is  $z^2 + c = z$ . There are two such values of  $z$ , namely

$$z = \frac{1 \pm \sqrt{1-4c}}{2}$$

The student should obtain this number  $z$  or several values of the complex number  $c$  to see how the HP-48 handles complex square roots and to verify that the absolute value  $|f(\text{one of these two values of } z)| > 1$ . Such a value of  $z$  is called a repelling fixed point of the mapping  $f$ . (Complex numbers  $w$  near such  $z$  have the property that  $f(w)$ ,  $f(f(w))$ ,  $f(f(f(w)))$ , ... get further and further from  $w$ .)

We call the repelling fixed point located by  $z$ , compute and graph members of the sequence  $f^{-1}(z)$ ,  $f^{-1}(f^{-1}(z))$ ,  $f^{-1}(f^{-1}(f^{-1}(z)))$ , ... . It can be shown that all such complex numbers satisfy  $|w| \leq (1 + \sqrt{1+4|c|})/2$ . For a complex number  $w$  since there are two inverse images, viz.  $\pm\sqrt{w-c}$ , the particular sequence of inverse iterates chosen involves a random choice of  $\pm$ .

Choose and store a complex number **C1**. To set the drawing screen and compute the repelling fixed point  $z$  we execute the following subprogram, named **PREP**.

```
<< C1 DUP ABS 4 * 1 + √ 1 + 2 / DUP NEG SWAP DUP2
      XRNG YRNG 4 * 1 SWAP - √ 1 + 2 / >>
```

The output is a fixed point of  $f$ . If the output has absolute value larger than .5 call the number  $Z$ , if not put  $Z = 1 - \text{the output}$ . Store  $Z$  and execute the program **BACK** given by

```
<< { # 0d # 0d } PVIEW Z BCK1 >> .
```

Here **BCK1** is

```
<< 1 500 START C1 - √ ONE * DUP PIXON NEXT DROP PICTURE >>
```

and the subprogram **ONE** is given by

```
<< (1,0) IF RAND .5 < THEN NEG END >>.
```

For  $c = (-.12256, .7449)$ , the following "graph" results



**Douady's Rabbit:**  $c = (-.123, .745)$

**EXERCISE 9.8:** Execute the program sequence given above for  $c = (-1,0)$ ,  $c = (-.5, -.1)$ .

## 9.4 PARAMETER IDENTIFICATION PROBLEMS REVISITED

Suppose  $\{ (t, y) \}$  data is given for the solution of an initial value problem and we wish to determine appropriate values of a parameter vector  $w = [p, q, a, b, \dots]$  in a function  $y = g(t, w)$  to fit the data, say in a least squares sense. That is, we want to choose  $w$  to minimize the sum

$$\sum_{i=1}^N [y_i - g(t_i, w)]^2.$$

By taking partial derivatives with respect to the components of  $w$  and setting them to 0 we obtain equations

$$\sum_{i=1}^N [y_i - g(t_i, w)] \frac{\partial g}{\partial w_k}(t_i, w) = 0, \quad k = 1, 2, \dots, M.$$

We take the left sides of these equations as components of a vector  $F$ , and attempt to solve the vector  $F(w) = 0$ . We will assume we have a starting values for the parameter vector  $w$  and give an iterative process.

The reader should review the algorithm we developed in Chapter 7 for finding a solution of a vector equation  $F(w) = 0$ . In this case also we assume there are  $m$  parameters and  $w$  will be the  $m$  vector of parameters and  $F$  will be the  $m$  vector given above involving the partial derivatives. Just as before we assume the components of the function  $F$  are smooth, and we have an approximate solution  $w_0$ , so that Taylors theorem gives the approximate formula

$$F(w) = F(w_0) + J(w_0) (w - w_0)$$

where the matrix  $J$  has  $i, j$  element  $\partial F_i / \partial w_j$ . If  $w$  is to be a good approximation of the solution, the left side of this equation is zero and we get a "formula" for an improved vector solution  $w$  in terms of the old approximate  $w_0$ . Just as in Chapter 7 we will provide an algorithm to construct and evaluate the function  $F(w)$  and the associated Hessian matrix.

Here is an outline the problem: First we create calculator programs for

$$[y - g(t, w)] \frac{\partial g}{\partial w_k}(t, w), \quad k = 1, 2, \dots, M$$

then we will use the program called DER created in Chapter 7 for finding the derivatives of these functions with respect to  $p = w_1, q = w_2, a = w_3, b = w_4$ , etc.

After execution, the derivatives can be used to create terms in the Hessian matrix  $J$  used in Newton's method.

Next we form the list  $\{ (t_1, y_1), (t_2, y_2), \dots, (t_n, y_n) \}$  by entering the number pairs on the stack, then entering  $n$  and the command  $\rightarrow \text{LIST}$  and store this as  $\text{DTA1}$ .

Finally we create programs called  $\text{JACM}$ ,  $\text{JEVP}$ ,  $\text{FACM}$  and  $\text{FEVP}$  to accumulate the data sums in the Hessian matrix and the function  $F$  after assigning values to  $p$  and  $q$ . (These programs will replace  $\text{JEV}$  and  $\text{FEV}$  in the chapter three procedure.) Now we have the ingredients of the Newton formula:

$$w_{\text{new}} = w - J(w)^{-1} F(w)$$

and can find a new vector  $w$ .

Many engineering and science problems require the solution of several nonlinear equations. Newton's method is one such algorithm. Most methods to accomplish this can fail under a variety of conditions. Good starting guesses are essential.

HP-48 programs are listed below for Newton's method for this problem. Here we assume there are  $m$  parameters and  $n$  data points

1. Store the value of  $m$  in  $M$  and the names of the  $m$  parameters in a list named  $\text{PL}$ , say  $\{ P \ Q \}$  for 2 parameters or in the case of four parameters, say  $\text{PL} = \{ P \ Q \ A \ B \}$ . Make sure each of the parameter "variables" in  $\text{PL}$  has been purged.
2. Purge the variables  $T$  and  $Y$  and store the components for the  $m$  functions  $F(T, Y, P, Q)$  in a list named  $\text{FL}$ . For example,

$$\{ '(Y - 3 \cdot \text{EXP}(-P \cdot T) + \text{EXP}(-Q \cdot T)) \cdot 3 \cdot T \cdot \text{EXP}(-P \cdot T)'$$

$$'(Y - 3 \cdot \text{EXP}(-P \cdot T) + \text{EXP}(-Q \cdot T)) \cdot T \cdot \text{EXP}(-Q \cdot T)'$$

would result from trying to fit  $y = 3 e^{-P t} - e^{-Q t}$  to data.

3. The program DER given in Chapter 7 will use the calculator's ability to take appropriate derivatives of the functions in FL.
4. Store the the N elements of data { (T, Y) } in a list DTA1.
5. Now create the programs (which assume values are assigned to P, Q)

Program Name	<b>JACM</b>
Purpose	Create matrix JMAT, gets a data point t,y and calls the subprogram JEV and does this for each data point
<pre>&lt;&lt; {M M} 0 CON 'JMAT' STO 1 N FOR I DTA1 I GET C→R       'Y' STO 'T' STO JEV NEXT &gt;&gt;</pre>	

Subprogram Name	<b>JEV</b>
Purpose	Evaluates the elements in JMAT at the data point and adds it to the value to the previous sum in the JMAT element
<pre>&lt;&lt; JL OBJ→ 1 SWAP START →NUM M SQ ROLL NEXT {M M}       →ARRY JMAT + 'JMAT' STO &gt;&gt;</pre>	

JMAT will be the Hessian matrix of derivatives.

6. Now for FVEC (F vector). In the following P, Q values have been assigned.

Subprogram Name	<b>FACM</b>
Purpose	Create FVEC, get a data point t, y and call FEV: do this for each data point
<pre>&lt;&lt; {M} 0 CON 'FVEC' STO 1 N FOR I DTA1 I GET C→R 'Y'       STO 'T' STO FEVP NEXT&gt;&gt;</pre>	

Subprogram Name	<b>FEVP</b>
Purpose	Evaluate the functions in FL at t,y, P, Q, ...
and	add the value to the previous value stored in FVEC
<pre>&lt;&lt; FL OBJ→ 1 SWAP START →NUM M ROLLD NEXT {M}       →ARRY FVEC + 'FVEC' STO &gt;&gt;</pre>	

Procedure: Store PL, FL, N, M and execute DER to get JL (J list). Put a vector [p,q] with initial values of P and Q on the stack and execute a program **NST1** given by

```
<< DUP OBJ→ DROP 'Q' STO 'P' STO JACM FACM FVEC JMAT / >>
```

The result is a copy of the old value of [P,Q] and the increment [ $\Delta P$ ,  $\Delta Q$ ]. Execute the command – and repeat.

**EXERCISE 9.9:** Use starting values  $p = .25$ ,  $q = 2$  and the data to determine appropriate values of p and q in

$$y = 3 e^{-pt} - 2 e^{-qt} :$$



to fit data  $\{ (0, 1), (.4, 1.89), (.8, 2.01), (1.2, 1.9), (1.6, 1.72), (2, 1.53), (2.4, 1.34), (2.8, 1.18), (3.2, 1.03), (3.6, .903), (4, .79), (5, .57) \}$

**EXERCISE 9.10:** Use starting data  $p = 2.2$ ,  $q = .925$ ,  $a = 5$  and  $b = -4$  to determine appropriate parameter values in

$$y = a e^{-pt} + b e^{-qt}$$

to fit data  $\{(0, 1), (.1, .3), (.2, -.2), (.3, -.6), (.4, -.88), (.5, -1), (.6, -1.2), (.7, -1.2), (.8, -1.3), (.9, -1.3), (1, -1.2), (1.5, -1), (2, -.7), (2.5, -.5), (3, -.3), (3.5, -1.7), (4, -1.1)\}$

**PROJECT EXERCISE: Data Fit in a Population Problem.** Population data  $\{ p_i \}$  at times  $\{ t_i \}$  (in ten year intervals between 1790 and 1990) was given in Chapter 6 just before problem 2.10. Suppose we use a model  $dp/dt = ap - bp^2$  : the form of the solution was also given in Chapter 6. We wish to minimize the payoff function

$$P(p_0^*, a, b) = \sum_{i=0}^n \left\{ p_i - \frac{ap_0^*}{bp_0^* + (a - bp_0^*) e^{-at_i}} \right\}^2$$

Starting values for the equations obtained by setting the partial derivatives with respect to  $p_0$ ,  $a$ , and  $b$  to zero can be obtained by using the data in the year  $t = 0$  (1790) and the years when  $t = 50$  and  $t = 100$ . We use Newton's iterative method.

We do not know the form of the solution for some of the population models given in Chapter 6. See EXERCISES 6.7, 6.9, 6.10 and 6.11. We have just discussed how to choose problem parameters to achieve a fit to data observations when the functional form of the solution  $g(t, p, q)$  is known. Consider the new problem of choosing  $p$  and  $q$  so that the solution of

$$dy/dt = q y (1 - y^p), \quad y(0) = .2$$

best fits the  $(t, y)$  data  $(1, .4)$ ,  $(2, .5)$ ,  $(3, .75)$ , and  $(4, .9)$ .

Here we have an initial value problem, say for population growth, of the form  $dy/dt = f(y, p, q)$ ,  $y(0) = y_0$  and wish to choose the parameters  $p$  and  $q$  so that a close fit to data is achieved. The solution must be obtained by using a numerical method (improved Euler, Runge Kutta, etc.) and the functional form of the solution is unknown. Even though we may attack a vector initial value problem of this type, for simplicity we will assume  $y$ ,  $p$ , and  $q$  are real numbers and for values of  $y$ ,  $p$ ,  $q$  in the domain of  $f$ ,  $f(y, p, q)$  is a real number. If we also assume  $f$  is a smooth function, we can differentiate the differential equation with respect to  $p$  to obtain

$$\frac{du}{dt} = f_y(y, p, q) u + f_p(y, p, q)$$

where  $u = \partial y / \partial p$  and  $f_y$  and  $f_p$  denote the partial derivatives of  $f$  with respect to  $y$  and  $p$  respectively. A similar equation holds for  $v = \partial y / \partial q$ . Consider the vector initial value problem  $dw/dt = F(w)$  with  $w = w(0)$  at  $t = 0$  for

$$w = \begin{bmatrix} y \\ u \\ v \end{bmatrix}, \quad F(w) = \begin{bmatrix} f(y, p, q) \\ f_y(y, p, q) u + f_p(y, p, q) \\ f_y(y, p, q) v + f_q(y, p, q) \end{bmatrix}, \quad w(0) = \begin{bmatrix} y_0 \\ 0 \\ 0 \end{bmatrix}.$$

Our criterion for best fit is to choose  $p$ ,  $q$  to minimize the payoff function

$$J = \sum_{i=1}^N [y_i - y(t_i, p, q)]^2.$$

Here the data observations are  $\{(t_1, y_1), (t_2, y_2), \dots, (t_N, y_N)\}$ , and  $y(t, p, q)$  is the solution of the original problem (and the first component of the solution of the vector equation). If we set the derivatives of  $J$  with respect to  $p$  and  $q$  to zero, we get

$$\sum_{i=1}^N [y_i - y(t_i, p, q)] u(t_i, p, q) = 0, \quad \sum_{i=1}^N [y_i - y(t_i, p, q)] v(t_i, p, q) = 0.$$

to determine  $p$  and  $q$ .

Our first thought here may be to apply Newton's method to determine solutions  $p, q$  of these two nonlinear equations. However, recall that Newton's method would require partial derivatives of  $u$  and  $v$  with respect to  $p$  and  $q$ . This could be done by differentiating the original differential equation more times, but then we would need to solve an initial value problem containing 6 differential equations !

Alternately, suppose that we have a trial set of parameters  $p$  and  $q$  and wish to choose better values  $p + \Delta p, q + \Delta q$ . If the incremental values are small then

$$y(t_i, p+\Delta p, q+\Delta q) \approx y(t_i, p, q) + \frac{\partial y}{\partial p}(t_i, p, q) \Delta p + \frac{\partial y}{\partial q}(t_i, p, q) \Delta q$$

Define a vector  $z$  with components  $y_i - y(t_i, p, q)$ ,  $i = 1, 2, \dots, N$  and an  $N$  by 2 matrix  $A$  with components  $A(i, 1) = \partial y / \partial p(t_i, p, q)$  and  $A(i, 2) = \partial y / \partial q(t_i, p, q)$ ,  $i = 1, 2, \dots, N$ . The payoff at  $p + \Delta p, q + \Delta q$  is

$$J = \sum_{i=1}^N [y_i - y(t_i, p+\Delta p, q+\Delta q)]^2$$

and we wish to choose  $\Delta p$  and  $\Delta q$  so that the new value of  $J$  is minimized. If we substitute the approximate value of  $y(t_i, p+\Delta p, q+\Delta q)$  into  $J$ , we need to choose the vector  $\delta = \text{column} [\Delta p, \Delta q]$  so that the quantity  $\|A\delta - z\|^2$  is minimized. Here  $\|A\delta - z\|^2$  is the sum of squares of the components of the vector  $A\delta - z$ . This is called a linear least squares problem. The solution is determined by solving  $A^t A \delta = A^t z$  where  $A^t$  is  $A$  transpose. (There is a better numerical method to determine  $\delta$  presented in standard linear algebra textbooks.) We change to the new values of  $p$  and  $q$  and repeat the process several times until either this process converges to good values of  $p$  and  $q$  or until it is clear the process is not converging. In the latter case, we need new starting values of  $p$  and  $q$ .

Thus our procedure is to take an initial guess for  $p$  and  $q$  and solve the initial value problem for  $w$  and recording the values of  $y$ ,  $u$ ,  $v$  at the various  $t_i$  measurement points, forming the matrix  $A$  and vector  $z$  and solving for the incremental vector  $\delta$ , correcting  $p$  and  $q$  and cycling thru this sequence of steps until a conclusion arises.

**EXERCISE 9.11:** Choose  $p$  and  $q$  so that the solution of

$$dy/dt = qy(1 - yP), \quad y(0) = .2$$

best fits the  $(t, y)$  data  $(1, .4)$ ,  $(2, .5)$ ,  $(3, .75)$ , and  $(4, .9)$ . Our criterion is to minimize

$$P = [y_1 - y(1, p, q)]^2 + [y_2 - y(2, p, q)]^2 + [y_3 - y(3, p, q)]^2 + [y_4 - y(4, p, q)]^2$$

where  $y_1 = .4$ ,  $y_2 = .5$ ,  $y_3 = .75$ ,  $y_4 = .9$  and  $y(t, p, q)$  is the solution of the original problem. Choose starting values of  $p$  and  $q$  near 1 and 1.

## 9.5 DIRECTION FIELDS

The following is a set of programs that will construct a direction field for

$$\frac{dy_1}{dt} = F_1(y_1, y_2), \quad \frac{dy_2}{dt} = F_2(y_1, y_2).$$

We construct the programs so that a student may conveniently overlay a plot of one or more solution trajectories on the direction field. This means that we should present one set of programs for those students who will use the *built-in algorithm* for constructing solutions and a second set of programs for those users who will use the Euler or improved Euler method to produce the solutions.

To use the HP-48G built-in algorithm suppose that  $FN$  gives the values of the vector  $[F_1(Y), F_2(Y)]$  (as illustrated in the first part of Chapter 5). That is  $FN$  takes values of the variable  $Y$  from memory. Set appropriate values in  $XRNG$  and

YRNG (program IN.PP presented in Chapter 5 can be used for this task), enter program FN (a method is to use IN.FN also from Chapter 5) and execute DIRF.

Program Name: **DIRF**

Purpose: Generate a direction field in the region prescribed by XRNG, YRNG.

Stored Quantities: FN

Input: none Output: direction field

```
<< [ 0 0 ] 'Y' STO ERASE { # 0d # 0d } DRAX PVIEW PPAR 2
GET PPAR 1 GET DUP 3 ROLLD - OBJ→ 9 / 3 ROLLD DUP
40 / 4 ROLLD 11 / SWAP OBJ→ DF1 PICTURE >>
```

Subprogram Name: **DF1**

```
<< → R DY2 DY1 Y1L Y2L << Y1L 1 10 START DY1 + DUP
1 12 FOR J DUP Y2L DY2 J * + R 3 ROLLD DF2 NEXT
DROP NEXT DROP >> >>
```

Subprogram Name: **DF2**

```
<< DUP2 'Y(2)' STO 'Y(1)' STO FN OBJ→ DROP SWAP DUP2
IF 0 == THEN DF.3 ELSE DF.4 END >>
```

Subprogram Name: **DF.3**

```
<< 0 IF == THEN DROP2 R→C PIXON DROP
ELSE DROP2 3 ROLL 3 DUPN DUP2 - R→C 4 ROLLD
+ RC LINE END >>
```

Subprogram Name: **DF.4**

```
<< DROP / ATAN DUP COS 3 ROLLD SIN 5 ROLL DUP 4
  ROLLD * DUP2 - 6 ROLLD + 4 ROLLD * DUP2 - 5 ROLLD
      + SWAP R→C
      3 ROLLD R→C LINE >>
```

Note that you can overlay a solution trajectory of the system using **G.12** as described in Chapter 5 after the inputs  $t_0$ ,  $y_0$ ,  $t_f$  are placed on the stack or one can use the input forms and the choose boxes. In the latter case we need to load the appropriate program into EQ by executing **FN.EQ** given by

```
<< 'FN' RCL 'EQ' STO >>.
```

The user may want to place these programs in the directory DE.1 containing the programs from chapter 5, placing DIRF early in the menu order and the subprograms late in the order.

For those users who wish to use the Euler or improved Euler algorithm given in the second part of Chapter 5, the programs D.RF, DF.1 and DF.2 are alternatives to DIRF, DF1 and DF2. To execute these programs we require a program F.N that takes the number T and the vector Y from the stack and produces  $[F_1(Y), F_2(Y)]$ .

Program Name: **D.RF**

Purpose: Generate a direction field in the region prescribed by XRNG, YRNG.

Stored Quantities: F.N

Input: none Output: direction field

```
<< ERASE { # 0d # 0d } DRAX PVIEW PPAR 2 GET PPAR 1 GET
  DUP 3 ROLLD - OBJ→ 9 / 3 ROLLD DUP 40 / 4 ROLLD 11
      / SWAP OBJ→ DF.1 GRAPH >>
```

Subprogram Name: **DF.1**

```
<< → R DY2 DY1 Y1L Y2L << Y1L 1 10 START DY1 + DUP
  1 12 FOR J DUP Y2L DY2 J * + R 3 ROLLDF.2 NEXT
      DROP NEXT DROP >> >>
```

Subprogram Name: **DF.2**

```
<< DUP2 2 →ARRY 0 SWAP FN OBJ→ DROP SWAP DUP2 IF
      0 == THEN DF.3 ELSE DF.4 END >>
```

Note that you can overlay a solution trajectory of the system using G.Y12 as described in Chapter 5 after the inputs  $t_0$ ,  $y_0$ ,  $t_f$  are placed on the stack. The user may want to place these programs in the directory DE.2 containing the programs from the second part of Chapter 5, placing D.RF early in the menu order and the subprograms late in the order.

**EXAMPLE:** Consider the system

$$\frac{dy_1}{dt} = y_2, \quad \frac{dy_2}{dt} = y_1^2 - \varepsilon - y_2.$$

The equilibrium points are  $y_2 = 0$ ,  $y_1 = \varepsilon$ . Let  $\varepsilon = 1$  and draw the direction field for  $-2 \leq y_1 \leq 2$ ,  $-1.5 \leq y_2 \leq 1.5$ . Overlay the trajectory which begins,  $t = 0$  at  $y_1 = 0$ ,  $y_2 = 1.5$  and ends when  $t = 2$ . Notice from the direction field that solution which initiates at  $y_1 = .5$ ,  $y_2 = 1.5$  has significantly different behavior for  $t > 0$ . Overlay such a solution. Now see the first figure in the introduction.

**EXERCISE:** Consider the system

$$\frac{dy_1}{dt} = y_2^2 - 1, \quad \frac{dy_2}{dt} = y_1^2 - 1.$$

The equilibrium points are  $y_1 = y_2 = \pm 1$ . Draw the direction field for  $-3 \leq y_1 \leq 3$ ,  $-3 \leq y_2 \leq 3$ .

**EXERCISE:** Consider the system

$$\frac{dy_1}{dt} = y_1^2 - 1, \quad \frac{dy_2}{dt} = y_1 y_2.$$

The equilibrium points are  $y_1 = \pm 1, y_2 = 0$ . Draw the direction field for  $-2 \leq y_1 \leq 2$ ,  $-3 \leq y_2 \leq 3$ .

**EXERCISE:** Consider the system

$$\frac{dy_1}{dt} = y_2, \quad \frac{dy_2}{dt} = y_1^2 - 3y_1^2.$$

The equilibrium points are  $y_2 = 0, y_1 = 0$  and  $y_2 = 0, y_1 = 1/3$ . Draw the direction field for  $-.4 \leq y_1 \leq .6$ ,  $-.5 \leq y_2 \leq .5$ . Overlay the four trajectories which begin ( $t = 0$ ) at  $y_1 = .15, y_2 = 0$  and ends when  $t = 7$ , begin at  $y_1 = -.3, y_2 = .3$  and ends when  $t = 3$ , begin at  $y_1 = -.3, y_2 = .4$  and ends when  $t = 10$ , and begin at  $y_1 = .5, y_2 = 0$  and ends when  $t = 3.6$ . (The last one satisfies  $y_1 = .5 \operatorname{sech}^2(t/2)$ .)

## Project Exercise in acoustical dynamics

The speed of sound traveling underwater depends on depth. We will use a ray model for underwater acoustic propagation and let  $z(x)$  denote the depth of a sound ray at position  $x$ , measured along the ocean surface. Snell's law can be written in the



form  $\cos \theta / C(z)$  is a constant where  $\tan \theta$  is the slope  $dz/dx$  and  $C(z)$  denotes the speed of sound transmission at depth  $z$ . Change the variable by  $y = C(z) dz/dx$  to obtain

$$\frac{dz}{dx} = \frac{y}{C(z)}, \quad \frac{dy}{dx} = -C'(z).$$

Determine  $C(z)$  by a least squares fit of the form  $C(z) = a e^{-bz} + c + mz$  ( $a, b, c$ , and  $m$  are constants to be determined) using the data

$z$	0	500	1000	1500	2000	2500	3000	3500	4000	5000
$C(z)$	5042	4995	4948	4887	4868	4863	4865	4869	4875	4875
$z$	6000	7000	8000	9000	10000	11000	12000			
$C(z)$	4887	4905	4918	4933	4949	4973	4991			

Next, find a value of  $\theta_0$  for the initial conditions:

$$z(0) = z_0, \quad y(0) = C(z_0) \tan \theta_0,$$

so that for  $z(x_f) = z_f$ , a prescribed number. For this problem take  $z_0 = 2000$ ,  $x_f = 24(5280)$ ,  $z_f = 3000$ . Plot the ray trajectory.

The function  $C(z)$  can be approximated by a least squares fit to data to another type of function (see Forsythe, George, Michael Malcolm and Cleve Moler, *Computer Methods for Mathematical Computations*, Prentice Hall, 1977.) Such a fit is given by

$$C(z) = 4779 + 0.01668 x + 160,295/(x+600).$$

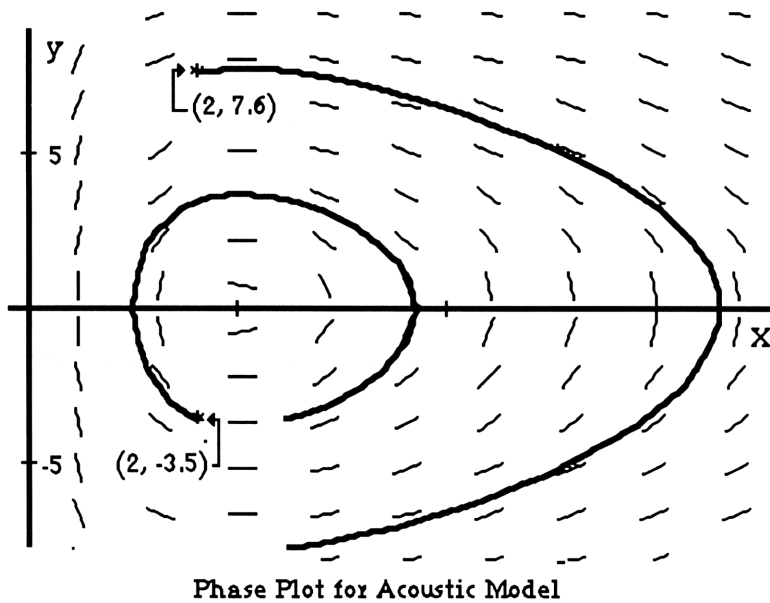
Re-scale the variables  $x$ , and  $z$  by  $t = x/10^4$ ,  $X = z/1000$ : the equation become

$$dy/dt = -10 f(X), \quad dX/dt = 100y/f(X)$$

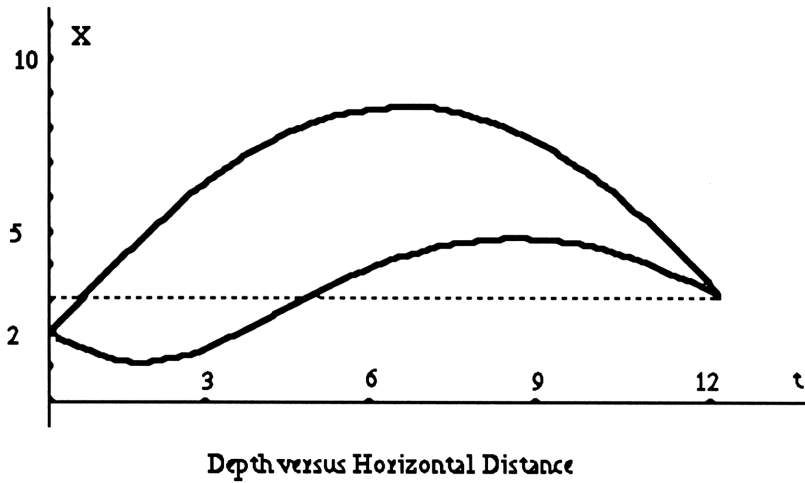
where  $f(X) = C(1000X)$ . Now we want to find  $y(0) = f(2) \tan \theta$  and  $X(0) = 2$  so that  $X(12.672) = 3$ . Dividing the differential equations gives  $dy/dX = -f'(X) f(X)/(10y)$

integration gives  $10 y^2 + f^2(x) = \text{a constant}$ . Phase plane graphs are shown. (An adaptive step method was used to obtain the graphs.)

It is interesting to compare the graph of the solution  $z(x)$  obtained by using the  $C(z)$  given above with that obtained using the  $C(z)$  function given in the project EXERCISE above. Any interpolation formula used for  $C(z)$  instead of a least squares fit also gives an interesting comparison.



The  $X$  vs.  $t$  graph is:



## 9.6 PROGRAMS FOR THREE DIMENSIONAL TRAJECTORIES

The programs given below may be used to produce graphs of solution trajectories in three dimensions. The user executes IN.P to input the high and low values of  $X$ ,  $Y$ ,  $Z$  for the view box to be shown and the position of the eye  $X_E$ ,  $Y_E$ ,  $Z_E$ . The utility program UTL1 defines an orthogonal set of vectors  $U.1$ ,  $V.1$  and  $W.1$  and sets the associated two dimensional plot parameters. Subprograms SCAL, UVW and PJ.1 are called by the other programs. The program G.Y3 is similar to other programs given earlier to graph the solution trajectories.

Program Name **G.Y3**

Stored Quantities FN HS TOL PJ.1

Input  $T_0$  (initial time)  
 $Y_0$  (initial value as a three vector)  
 $T_f$  (final time)

Output : Solution graph of  $Y' = F(T, Y)$ ,  $Y(T_0) = Y_0$ ,  
 $T_0 \leq T \leq T_f$ .

```
<< { # 0d # 0d } PVIEW 3 ROLLD 'Y' STO 'T' STO → TF <<
{ T Y FN } TOL HS Y PJ.1 R→C 4 ROLLD DO RKFSTEP Y
PJ.1 R→C DUP 6 ROLLD 5 ROLL LINE DUP T + TF UNTIL
> END DROP TF T - RKFSTEP Y PJ.1 R→C DUP 6 ROLLD
5 ROLL LINE DROP TF T - RKFSTEP Y PJ.1 R→C 5 ROLL
LINE 3 DROPN >> PICTURE >>
```

Program Name **IN.P**

Stored Quantities none Input none

Output coordinates of the corners of the viewing box  
and the position of the eye have been stored

```
<< "ENTER XL XU" " " INPUT OBJ→ 'XU' STO 'XL' STO
"ENTER YL YU" " " INPUT OBJ→ 'YU' STO 'YL' STO
"ENTER ZL ZU" " " INPUT OBJ→ 'ZU' STO 'ZL' STO
"ENTER XE" " " INPUT OBJ→ "ENTER YE" " " INPUT OBJ→
"ENTER ZE" " " INPUT OBJ→ 3 →ARRY 'W.1 STO >>
```

Program Name	<b>UTL1</b>
Stored Quantities	variables from IN.P and programs SCAL, UVW
Input	none
Output	the parameters XRNG YRNG have been set.  << UVW SCAL SCATRPLOT 1.125 DUP *W *H ERASE FUNCTION CLΣ >>

Subprogram Name	<b>SCAL</b>
Stored Quantities:	Output from IN.P and UVW      Input: none
Output	a matrix ΣDAT has been defined containing the projections of the corners of the viewing box.  << CLΣ XL YL DUP2 ZL →V3 PJ.1 →V2 Σ+ ZU →V3 PJ.1 →V2 Σ+ XL YU DUP2 ZL →V3 PJ.1 →V2 Σ+ ZU →V3 PJ.1 →V2 Σ+ XU YL DUP2 ZL →V3 PJ.1 →V2 Σ+ ZU →V3 PJ.1 →V2 Σ+ XU YU DUP2 ZL →V3 PJ.1 →V2 Σ+ ZU →V3 PJ.1 →V2 Σ+ >>

Subprogram Name	<b>UVW</b>
Stored Quantities	Output from IN.P and UVW
Input	none
Output	an orthogonal set of vectors U.1, V.1 and W.1 have been constructed.  << W.1 DUP ABS / DUP DUP 'W.1' STO [ 0 0 1 ] DUP 3 ROLL DOT 3 ROLL * – DUP ABS / DUP 'V.1' STO W.1 CROSS 'U.1' STO >>

Subprogram Name	<b>PJ.1</b>
Stored Quantities	Output from UVW
Input	any vector $Z$ in $\mathbb{R}^3$
Output	the coordinates of the projection of $Z$ in $\mathbb{R}^2$ .
<< DUP U.1 DOT SWAP V.1 DOT >>	

**EXAMPLE:** For view box  $-2 \leq x \leq 2$ ,  $-2 \leq y \leq 2$ ,  $-5 \leq z \leq 2$ , eye position  $[1, 1, .5]$ ,  $Y' = AY$ , with  $A$  given below,  $Y(0) = [2, 0, 2]$  and  $0 \leq t \leq 12.56$  we obtain the following picture:



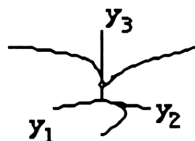
**xyx solution for**  
 $Y' = AY, Y(0) = [2, 0, 2]$

Here the first row of  $A$  is  $[-.1, 2, 0]$ , the second row is  $[-2, -.1, 0]$ , and the third row is  $[0, 0, -.2]$ . Here the solution is given by  $x(t) = 2 e^{-.1 t} \cos t$ ,  $y(t) = 2 e^{-.1 t} \sin t$ , and  $z(t) = 2 e^{-.2 t}$ .

**EXERCISE:** Use the same view box and eye position as given above. For  $0 \leq t \leq 6$  and

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -.5 & 0 \\ 0 & 0 & -.25 \end{bmatrix}$$

plot the solutions of  $Y' = AY$  with initial conditions  $[2, 0, 2]$ ,  $[0, 2, 2]$  and  $[2, 2, 0]$ .



**Normal Mode Solutions**  
to  $Y' = AY$

## **PART III**

# **ENGINEERING MATHEMATICS**

Applications of the HP-48G Scientific Calculator to problems in elementary mathematics, calculus, probability, linear algebra, and differential equations were the subject of earlier chapters. Examples were chosen to demonstrate the power of computational methods for deducing useful information from mathematical models and for reinforcing mathematical concepts.

These chapters pursue further examples often associated with a course in advanced calculus or “advanced engineering mathematics”. Typical examples include numerical methods for solving differential equations, use of infinite series and integrals to study non-elementary functions, boundary-value problems, and problems from vector field theory. When students first encounter such problems they often underestimate the power of the mathematical methods involved. The apparent resistance of the problems to pencil and paper techniques can lead them to conclude that the tools at hand lack real practical utility. It is in this setting that a few well-selected computational algorithms, and knowledge of how to implement them on the HP-48G calculator, can contribute enormously to the student’s confidence. The marriage of mathematical theory and computational methods gives students power to solve the kind of scientific and engineering problems that arise in real world situations.

Examples in the chapters illustrate the power of the HP-48G calculator—both its built-in functions and its programming potential. No attempt was made to include all possible topics from a typical engineering mathematics course. But the author hopes that the examples hold intrinsic interest and are of sufficient variety to stimulate further mathematical explorations.

# 10

## MORE ON SOLUTIONS FOR DIFFERENTIAL EQUATIONS

Differential equations are commonly used to model the dynamic behavior of physical systems. Beginning courses in the subject classify such equations according to their *order* and whether they are *linear* or *nonlinear*. Students learn to solve a variety of simple cases that apply to problems in physics, engineering, chemistry and biology. But they also soon learn that, more likely than not, differential equations encountered in real applications do not yield easily to simple techniques, and indeed they frequently do not possess solutions expressible in a finite closed form in terms of elementary functions familiar to the student. This is disquieting when first learning the subject. One can easily have concern that around every corner there lurk inaccessible problems. Fortunately, numerical methods can be applied in many such cases. And, combined with a few mathematical theorems concerning the existence and global behavior of solutions, numerical techniques restore the student's ability to extract information from mathematical models.

Even simple numerical methods are useful. For example the Euler and Improved Euler methods, introduced in Chapter 5, enable one to generate numerical solutions for quite general initial-value problems. And the built-in functions of the HP-48G for solving and plotting differential equations provide sophisticated and powerful tools. Our examples will compare both elementary methods, such as the second-order Runge-Kutta algorithm that has useful teaching value, and the more powerful Runge-Kutta-Fehlberg algorithm that is built-in.



## 10.1 A RUNGE KUTTA METHOD FOR TWO EQUATIONS

Consider the following initial-value problem

$$y'' + x^2 y = 0, \quad y(0) = 1, \quad y'(0) = 0. \quad (10.1)$$

It occurs in problems of bending beams and in optics. It does not have a closed form solution expressible in elementary functions.<sup>1</sup> Nevertheless the graph of  $y(x)$  can be generated using a Runge-Kutta method. And important features, such as the location of its zeros, can be determined (approximately). The HP-48G program below is a simple implementation of a second-order Runge-Kutta algorithm. It solves an initial-value problem for a system of two first-order differential equations.

### PROGRAM 10.1. A Runge-Kutta method.

The program generates a numerical solution of the initial-value problem

$$\begin{aligned} \frac{dy}{dx} &= f_1(x, y, z) \\ \frac{dz}{dx} &= f_2(x, y, z) \\ y'(x_0) &= y_0 \\ z'(x_0) &= z_0 \end{aligned} \quad (10.2)$$

on the interval  $x_0 \leq x \leq xmax$  with stepsize  $h$ . It takes as inputs the functions  $f_1$  and  $f_2$ , programmed as user-defined functions, and the initial conditions and graph parameters, as shown in the following table.

---

<sup>1</sup> Later in the section we will see that its solutions can be expressed in terms of Bessel functions.

Inputs		Outputs
4:	<< → x y z 'f1(x,y,z)'>>	Draws graph of solution of (10.2)
3:	<<→ x y z 'f2(x,y,z)'>>	
2:	{x0 y0 z0 h}	
1:	{xmin xmax ymin ymax}	

```

<< 4 DUPN OBJ→ DROP
YRNG DUP 3 ROLLD
XRNG SWAP OBJ→ DROP 0 0
→ f1 f2 b x y z h p q
<< x y R→ C
  CLLCD {(0,0) {1 1}} AXES ERASE DRAX LABEL

  WHILE x b <
  REPEAT
    'y+h*f1(x,y,z)' →NUM 'p' STO
    'z+h*f2(x,y,z)' →NUM 'q' STO
    x h + 'x' STO
    '(y+p)/2+.5*h*f1(x,p,q)' →NUM 'y' STO
    '(z+q)/2+.5*h*f2(x,p,q)' →NUM 'z' STO

    x y R→C DUP 3 ROLLD LINE
    "(" x "," y ")" + + + + 2 DISP
  END
  >> DROP PICTURE
  >>

```

- Unpack input lists, set x and y range, and load inputs into variables.
- Initialize graph
- Loop to generate the points on the curve.
- Generate the next point on the curve, using the Runge-Kutta method.
- Draw line to next point, and display it.

The graph that follows was generated by this program on the interval  $0 \leq x \leq 6$ , using the value  $h = 0.1$ . The given second-order initial-value problem was first converted to an equivalent system of two first-order differential equations:

$$\begin{aligned}
 \frac{dy}{dx} &= z \\
 \frac{dz}{dx} &= -x^2 y \\
 y(0) &= 1, \quad z(0) = 0,
 \end{aligned}
 \tag{10.3}$$

and the program was provided with initial data by placing the four objects

```
<< → x y z 'z' >> << → x y z '-x^2*y' >> {0 1 0 .1} {0 6 -1 1}
```

on the stack.

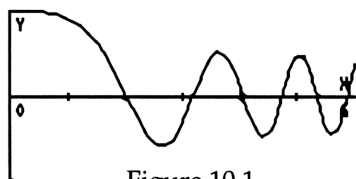


Figure 10.1

Graph of the solution of the initial-value problem (1.1), or (1.3), drawn by the HP-48G using program 10.1.

The oscillatory nature of the solution is easily observed in the graph, as is the damping of the oscillations and the increase in their frequency as  $x$  increases. Such behavior is predicted by the *Sturm Comparison Theorem* which implies that the zeros of the solution separate, on any interval on which  $x > k$ , the zeros of any solution of  $y'' + k^2 y = 0$ . In particular they separate the zeros of  $\sin kx$ . The damping in Figure 10.1 is implied by the *Sonin-Polya theorem*<sup>2</sup>.

The zeros of the solution shown in Figure 10.1 can be found approximately using the COORD key in the HP-48G's PICTURE environment. They are found at 2.0, 3.2, 4.1, ..., approximately. We shall determine them more accurately later.

As appealing as the example and graph, above, appear, we will discover that things are not so simple as we might hope. When the solution is graphed on a larger interval something clearly goes wrong.

---

<sup>2</sup> For the Sturm Comparison Theorem, see [Kreider, et al.]. A special case of the Sonin-Polya theorem covering our example is also included there. A more general proof can be found in Birkhoff and Rota, *Ordinary Differential Equations*, Ginn, Boston, 1962.

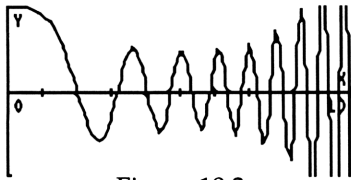


Figure 10.2

Graph of the solution of (10.1), drawn on the interval  $0 \leq x \leq 10$  by the same program.

**EXERCISE 10.1.**

Enter Program 10.1 into your HP-48G and reproduce the graphs shown above. Then generate the graph of the solution of (10.1) on the interval  $0 \leq x \leq 15$ . How much confidence do you now have in the zeros estimated using the HP-48G's COORD key? What explanation would you give as to why matters have deteriorated so badly? What happens if a smaller value of  $h$  is used? Try using  $h = 0.01$ . What disadvantage does this entail?

**EXERCISE 10.2.**

To regain your confidence slightly, apply Program 10.1 to graphing on the interval  $0 \leq x \leq 15$  the solution of the initial-value problem

$$y'' + y = 0, \quad y(0) = 1, \quad y'(0) = 0,$$

again using the value  $h = 0.1$ . How accurate is the fifth zero  $9\pi/2$ ? (In this case we know that the solution is  $\cos x$ .)

In Section 3 we will see that the difficulties we met in applying the simple Runge-Kutta algorithm disappear when the requirement of constant step size  $h$  is abandoned. Indeed, the built-in differential equation functions of the HP-48G implement a variable step size algorithm, and we will see in Section 3 how effectively this solves the problem. Nevertheless the simple Runge-Kutta algorithm is useful for many situations when accuracy is not the principal concern, and it remains a valuable teaching tool.

## 10.2 SERIES SOLUTIONS

The foregoing examples are typical of differential equations that arise in practice. When the equations are nonlinear, or when they are linear but do not have constant coefficients, it is a rare accident if their solutions can be expressed in closed form in terms of elementary functions. In such cases, however, we abandon our demand for neat little formulas for solutions. After all, a differential equation itself completely determines the solution to the given initial-value problem. A number of mathematical theorems stand ready to predict the solution's global behavior. And numerical methods are available, finally, to obtain useful local values and behavior.

Nevertheless we often seek explicit representations of solutions in more general forms—infinite series or integral representations being commonly employed. For example we might try to find solutions  $y(x)$  that can be expressed in the form of Taylor series

$$y(x) = \sum_{k=0}^{\infty} \frac{y^{(k)}(x_0)}{k!} (x - x_0)^k = \sum_{k=0}^{\infty} a_k (x - x_0)^k. \quad (10.4)$$

Under suitable conditions a solution is represented by such a series within its interval of convergence. And the coefficients can be determined in a straightforward way—by determining the values of  $y^{(k)}(x_0)$  directly from the differential equation, or by using a recurrence relation for  $a_k$ . Several examples will illustrate the method.

### EXAMPLE 10.2.

Find a Taylor series solution

$$y = \sum_{k=0}^{\infty} \frac{y^{(k)}(0)}{k!} (x - x_0)^k \quad (10.5)$$

that satisfies the initial-value problem  $y'' + x^2 y = 0$ ,  $y(0) = 1$ ,  $y'(0) = 0$ , of Example 10.1. We compute the derivatives of  $y(x)$  successively. The differential equation yields

immediately  $y''(0) = 0$  when we set  $x = 0$ . In the same way, successive derivatives of the equation give

$$\begin{aligned} y'' + x^2 y' + 2xy &= 0: & y''(0) &= 0, \\ y^{(4)} + x^2 y'' + 4xy' + 2y &= 0: & y^{(4)}(0) &= -2, \\ &\dots & & \end{aligned} \quad (10.6)$$

and in general

$$\begin{aligned} y^{(n+2)} + x^2 y^{(n)} + 2nxy^{(n-1)} + n(n-1)y^{(n-2)} &= 0: \\ y^{(n)}(0) &= -(n-2)(n-3)y^{(n-4)}(0). \end{aligned} \quad (10.7)$$

From equations (10.6) and (10.7) we then obtain

$$\begin{aligned} 0 &= y''(0) = y^{(6)}(0) = y^{(10)}(0) = \dots, \\ 0 &= y'''(0) = y^{(7)}(0) = y^{(11)}(0) = \dots, \end{aligned}$$

and the remaining derivatives depend on either  $y(0)$  or  $y'(0)$ . Thus we can find two linearly independent solutions  $y_0(x)$  and  $y_1(x)$ , corresponding to the two sets of initial conditions:  $y(0) = 1, y'(0) = 0$  or  $y(0) = 0, y'(0) = 1$ . In the first case we obtain

$$y^{(4k)}(0) = -(4k-2)(4k-3)y^{(4k-4)}(0), \quad k = 1, 2, 3, \dots,$$

and we then quickly calculate as many of the non-zero derivatives as we wish. The desired solution is then

$$y_0(x) = 1 + \sum_{k=1}^{\infty} \frac{y^{(4k)}(0)}{(4k)!} x^{4k}, \quad (10.8)$$

and the function  $y_0(x)$  can be computed by a simple program on the HP-48G.

**PROGRAM 10.2. Series solution of an initial-value problem.**

The program computes values of the solution  $y_0(x)$  (Equation (10.8)).

Inputs	Outputs
1: $x$	1: $y(x)$

Store the program in the variable Y0. Since the program was written in the form of a user-defined function, i.e. with the syntax `<< → x '<<... >> >>`, it can be executed either by placing its argument  $x$  on the stack and pressing the user menu key Y0, or by evaluating the algebraic expression 'Y0(X)'. Thus the PLOT and SOLVE environments of the HP-48G are available to plot the graph of  $y_0(x)$  or to find its zeros. (Since its definition is in the form of a *program* rather than an *expression*, operations involving differentiation, such as finding relative maximum and minimum points, are excluded.)

`<< → x`

`<< 0 1 1 1 → k Y T S`

`<< DO`

`'k+4' EVAL 'k' STO`

`Y k 2 - k 3 - * NEG * 'Y' STO`

`Y k ! / x k ^ * 'T' STO`

`T S + 'S' STO`

`UNTIL T ABS 1E-12 <`

`END`

`S`

`>>`

`>>`

`>>`

- Make it a user-defined function
- Initial values

- Add terms until they are small

- Put the final sum on the stack

### EXERCISE 10.3.

Enter Program 10.2 into your HP-48G and experiment with computing various values of  $y_0(x)$ . Plot the function on the interval  $0 \leq x \leq 5$ . Find the zeros near 2.0, 3.2, and 4.1 more exactly using the SOLVE application or the function ROOT (SOLVE:ROOT:ROOT).

**EXERCISE 10.4.**

If the initial conditions  $y(0) = 0$ ,  $y'(0) = 1$  are used with the same differential equation, a second linearly independent solution

$$y_1(x) = x + \sum_{k=1}^{\infty} \frac{y^{(4k+1)}(0)}{(4k+1)!} x^{4k+1}$$

is obtained. Exactly the same recurrence relation (10.7) determines all the derivatives, hence only the initial conditions need to be changed in Program 10.2. (In fact only the initial values of  $k$  and  $S$  need to be changed. The initial value of  $T$  is not used. Initializing  $T$  serves only to create  $T$  as a local variable.) Change the initial values of  $k$  and  $S$  to 1 and  $x$ , respectively, and have the HP-48G draw the graph of  $y_1(x)$  on the interval  $0 \leq x \leq 5$ . Find its smallest zeros.

One source of inaccuracy in Program 10.2 arises from the way terms of the series are computed. Each of the quantities  $y^{(k)}(0)$ , and  $k!$  becomes very large as  $k$  increases, although their *quotient* becomes vanishingly small. Avoiding large intermediate results in computations is generally helpful in achieving accuracy. In this case we observe that the coefficients  $a_k = y^{(k)}(0) / k!$  in (10.4) satisfy the recurrence relation

$$a_k = -\frac{a_{k-4}}{k(k-1)}. \quad (10.9)$$

This leads to an alternative program (below) for computing the solutions  $y_0(x)$  and  $y_1(x)$ .

**PROGRAM 10.3. Series solution of an initial-value problem.**

The program defines a user-defined function to compute two linearly independent solutions  $y_n(x)$ ,  $n = 1, 2$ , for the differential equation  $y'' + x^2 y = 0$ .

Inputs		Outputs	
2:	n	1:	$y_n(x)$
1:	x		



```
<< → n x
```

```
<< n IF n THEN x ELSE 1 END DUP  
→ k T S
```

- Make it a user-defined function
- Initialize k, T and S

```
<< DO  
  'k+4' EVAL 'k' STO  
  T k k 1 - * / NEG x 4 ^ * 'T' STO  
  T S + 'S' STO  
  UNTIL T ABS 1E-12 <  
  END  
  S  
>>  
>>  
>>
```

- Add terms until they are small
- Put the final sum on the stack

### EXAMPLE 10.3.

Store Program 10.3 in the variable Y. Then use it to graph the functions  $y_0(x)$  and  $y_1(x)$  on the intervals  $0 \leq x \leq 5$  (Fig 10.3) and  $0 \leq x \leq 8$  (Fig 1.4). They can be graphed simultaneously by entering the list

{ 'Y(0,X)' 'Y(1,X)' }

into the EQ field of the HP-48G's PLOT application.

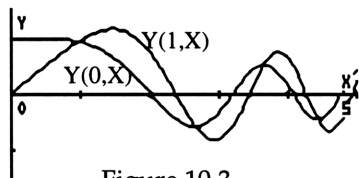


Figure 10.3

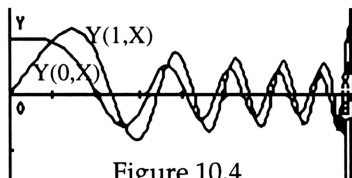


Figure 10.4

Again, on a sufficiently small interval the program apparently behaves well, but for larger values of  $x$  something is still going wrong. We have avoided the problem caused by the large values of  $y^{(k)}(0)$  and  $k!$ , but when  $x$  is large, the terms  $a_k x^k$  of the series initially grow very large before eventually settling down and approaching zero. We thus have a situation in which an alternating series of very large terms is adding up to a very small final value. The HP-48G represents real numbers to about 15 significant digits of accuracy, thus when  $x$  is large enough that the magnitude of intermediate terms in the computation of the sum of the series are themselves 15-digit numbers, all accuracy in the final sum is lost!

### EXERCISE 10.5.

Enter Program 10.3 into your HP-48G and verify the results in Figures 10.3 and 10.4. Modify the program so that each partial sum  $S$  is displayed as the program runs. (Insert `S 1 DISP -1 WAIT` immediately before the `UNTIL` statement.) For what value of  $x$  do the partial sums grow to 12 digits ( $\approx 10^{12}$ ). How is this related to Figure 10.4? How accurate would you expect the computed values of  $y_0(x)$  and  $y_1(x)$  to be when  $x = 5$ ?

The method of Taylor series for solving differential equations is quite general. Most texts on the subject show, for example, that the *linear* differential equation

$$a_n(x)y^{(n)} + a_{n-1}(x)y^{(n-1)} + \dots + a_1(x)y' + a_0(x)y = h(x) \quad (10.10)$$

has two linearly independent Taylor series solutions about any point  $x_0$  at which the coefficients and  $h(x)$  themselves possess such expansions and the leading coefficient  $a_n(x_0) \neq 0$ . The radius of convergence of the series solutions can be shown to be the distance from  $x_0$  (in the complex plane) to the nearest point at which such regularity conditions fail. Such a point  $x_0$  is called a *regular point* of equation (10.10), and we can develop programs for the Taylor series solutions in the same manner as was done in the examples above. Note, by the way, that the *origin* is a regular point for the differential equation in Example 10.2 and that there are no points in the complex plane where the

regularity conditions fail. Thus the radius of convergence of the series expansions of  $y_0(x)$  and  $y_1(x)$  is *infinity*. This would seem to settle the question completely of finding solutions on the interval  $-\infty \leq x \leq \infty$ . As the examples show, however, numerical issues often become the limiting factor rather than the issue of mathematical convergence.

We conclude with one final example to show that the series method is still useful when the recurrence relation is substantially more complicated—when each coefficient of the series can depend on several preceding coefficients, not just one as in Equation (10.9).

#### EXAMPLE 10.4

We propose to solve the initial-value problem

$$y'' + e^x y = 0, \quad y(0) = y'(0) = 1. \quad (10.11)$$

The origin is a regular point of the differential equation, thus two linearly independent Taylor series solutions exist. They converge for all values of  $x$ .

Since we seek solutions of the form  $y(x) = \sum_{k=0}^{\infty} a_k x^k$  we substitute into (10.11):

$$\sum_{k=0}^{\infty} k(k-1)a_k x^{k-2} + \sum_{k=0}^{\infty} \frac{x^k}{k!} \sum_{k=0}^{\infty} a_k x^k = 0.$$

Multiplying the two series and collecting terms we obtain

$$\sum_{k=0}^{\infty} \left[ (k+2)(k+1)a_{k+2} + \sum_{j=0}^k \frac{a_j}{(k-j)!} \right] x^k = 0.$$

This leads, finally, to a recurrence relation from which each coefficient  $a_k$  can be computed from the values of coefficients with smaller index:

$$a_{k+2} = \frac{-1}{(k+2)(k+1)} \sum_{j=0}^k \frac{a_j}{(k-j)!}, \quad k \geq 0. \quad (10.12)$$

The initial conditions determine  $a_0 = a_1 = 1$ , and the recurrence relation (1.12) then permits us to calculate as many of the remaining coefficients as we need:

$$\begin{aligned}a_2 &= -\frac{a_0}{2} = -\frac{1}{2}, \\a_3 &= -\frac{a_0 + a_1}{6} = -\frac{1}{3}, \\a_4 &= -\frac{1}{12} \left( \frac{a_0}{2} + a_1 + a_2 \right) = -\frac{1}{12}, \\&\dots\end{aligned}$$

The initial terms of the desired solution are, therefore,

$$y(x) = 1 + x - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{12} + \dots$$

The following program provides a user-defined function for  $y(x)$ :

**PROGRAM 10.4. Handling a many-term recurrence relation.**

The program defines a user-defined function to compute the solution  $y(x)$  of the initial-value problem (10.11).

Inputs		Outputs	
1:	x	1:	y(x)

The program stores the coefficients  $a_0, a_1, \dots, a_n$  used in the computation in the variable COEFFS.

```
<< → X
```

```
<< 1 X X 1 + 0
→ k T S nextA
```

```
<< {1 1} 'COEFFS' STO
DO
  'k+1' EVAL 'k' STO

  0 'nextA' STO
  0 k 2 - FOR j
    COEFFS j 1 + GET
    '(k-2-j)!' EVAL /
    nextA + 'nextA' STO
  NEXT
```

```
nextA k k 1 - * / NEG 'nextA' STO
COEFFS nextA + 'COEFFS' STO
nextA x k ^ * 'T' STO
T S + 'S' STO
UNTIL T ABS 1E-6 <
END
S
```

```
>>
>>
>>
```

- Make it a user-defined function
- Initialize k, T and S and nextA
- Initialize COEFFS
- Start adding series
- Update k
- Next coefficient is nextA. It is a sum. The FOR loop evaluates the sum.
- Finish computation of nextA
- Update T and S
- Put the final sum on the stack

We store the program in the variable YMT and, in the HP-48G's PLOT application, enter 'YMT(X)' as the current equation. On the interval  $0 \leq x \leq 3$  the following graph is drawn. (We note, by examining the variable COEFFS that the program uses 52 terms of the series to achieve the desired accuracy when  $x = 3$ . Since each coefficient in the series is itself a sum, the computation is of considerable size! (It is useful to increase the stepsize in the PLOT application to reduce the time needed for generating the graph.) Using the HP-48G's SOLVE application, the first zero of the solution is found at  $x = 1.58656$ .

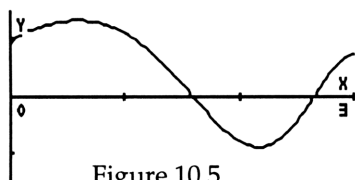


Figure 10.5

Graph of the solution of the initial-value problem (10.1), drawn by the HP-48G using program 10.4.

### EXERCISE 10.6.

Enter Program 10.4 into your HP-48G, and verify the example above. Find the *second* zero of the solution lying in the interval  $0 \leq x \leq 3$ . Add to the program at an appropriate place the commands `S 3 DISP`, and then monitor the size of the partial sums during the course of the computation. What can you say about the accuracy achieved in the computation of  $y(3)$ ?

### EXERCISE 10.7.

Find a recurrence relation for Taylor series solutions about the origin of the differential equation  $y'' - 3xy' + y = 0$ . Write a program for the solution satisfying the initial conditions  $y(0) = 0$ ,  $y'(0) = 1$ . Use your HP-48G to plot this solution on a suitable interval, and find any zeros that lie in the interval.

### EXERCISE 10.8.

Find a recurrence relation for the Taylor series solution about the origin of the initial-value problem

$$3y''' - xy' + x^2y = e^x,$$

$$y(0) = y'(0) = 0, \quad y''(0) = \frac{1}{4}.$$

Write a program for the HP-48G that defines this function as a user-defined function (as in the examples above). Explore the solution graphically.

# 11

## BESSEL FUNCTIONS

### 11.1 BESSEL'S EQUATION: SOLUTIONS OF THE FIRST KIND

Bessel's equation of order  $p$  is

$$x^2 \frac{d^2 y}{dx^2} + x \frac{dy}{dx} + (x^2 - p^2)y = 0. \quad (11.1)$$

Its solutions are encountered in problems involving temperature distributions over regions with cylindrical symmetry, in determining fundamental buckling modes of columns with non-uniform cross section, and in many other problems involving damped oscillatory motion with non-uniform frequency.

Bessel's equation differs mathematically from those explored in Chapter 10 in that the origin is a *singular* point of the equation. Thus we cannot in general expect it to possess power series solutions about the origin. On the other hand the singularity of Bessel's equation is a modest one—it is a *regular singular point* in the sense of the following definition.

**Definition.** A point  $x_0$  is said to be a *regular singular point* of a second-order linear differential equation if the equation can be written in the form

$$(x - x_0)^2 y'' + (x - x_0)a_1(x)y' + a_2(x)y = h(x),$$

where  $a_1(x)$ ,  $a_2(x)$  and  $h(x)$  have power series expansions about  $x_0$ .

Although there may be no power series solution about a regular singular point, there is always at least one solution of the form

$$y = (x - x_0)^c \sum_{k=0}^{\infty} a_k (x - x_0)^k, \quad (11.2)$$

where  $c$  is a constant. Both  $c$  and the coefficients  $a_k$  can be determined by *the method of undetermined coefficients*, i.e. by substituting (11.2) into the differential equation and determining the coefficients from the requirement that the equation be satisfied. We illustrate this in the case of Bessel's equation.

**EXAMPLE 11.1.**

Substituting (11.2) into Bessel's equation we obtain

$$x^2 \sum_{k=0}^{\infty} (k+c)(k+c-1)a_k x^{k+c-2} + x \sum_{k=0}^{\infty} (k+c)a_k x^{k+c-1} + (x^2 - p^2) \sum_{k=0}^{\infty} a_k x^{k+c} = 0,$$

and, after simplifying and collecting terms,

$$(c^2 - p^2)a_0 + (2p+1+c^2-p^2)a_1 + \sum_{k=0}^{\infty} [(k+c)^2 - p^2]a_k x^{k+c} + \sum_{k=2}^{\infty} a_{k-2} x^{k+c} = 0.$$

For  $a_0$  to remain arbitrary we must have  $c^2 - p^2 = 0$ , or  $c = \pm p$ . The quadratic equation is called the *indicial* equation for Bessel's equation and the roots  $c$  the *indices*. The positive root determines a solution, with

$$\begin{aligned} a_0 & \text{ arbitrary,} \\ a_1 & = 0, \quad \text{and} \\ a_k & = -\frac{a_{k-2}}{k(2p+k)}, \quad k \geq 2. \end{aligned} \tag{11.3}$$

We could stop at this point, using the recurrence relations (11.3), in the manner of the foregoing examples, to develop an infinite series for the solution. But further simplification is possible. From (11.3) we obtain



$$\begin{aligned}
 a_1 = a_3 = a_5 = \dots = 0, \quad \text{and} \\
 a_{2k} &= (-1)^k \frac{a_0}{2 \cdot 4 \cdot 6 \cdots (2k) \cdot (2p+2)(2p+4) \cdots (2p+2k)} = \\
 &= \frac{(-1)^k}{2^{2k} k!} \cdot \frac{a_0}{(p+1)(p+2) \cdots (p+k)}.
 \end{aligned}$$

A final simplification will result from expressing the product in the denominator as  $\Gamma(p+k+1) / \Gamma(p+1)$ , where  $\Gamma$  is the celebrated *gamma function* defined by

$$\Gamma(x) = \int_0^{\infty} e^{-t} t^{x-1} dt, \quad x > 0.$$

It is easily verified that  $\Gamma(1) = 0$  and (integrating by parts) that  $\Gamma(x+1) = x\Gamma(x)$ . Thus if  $n$  is a non-negative integer it follows that  $\Gamma(n+1) = n(n-1)(n-2) \cdots 2 \cdot 1 = n!$ , earning the gamma function the distinction of generalizing the factorial function to a continuous function of  $x$  for  $x > 0$ .<sup>1</sup> Finally, choosing  $a_0 = 2^{-p} / \Gamma(p+1)$ , we are led to the solution

$$J_p(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{k! (k+p)!} \left(\frac{x}{2}\right)^{2k+p}, \quad (11.4)$$

usually called the *Bessel function of order  $p$  of the first kind*. In equation (11.4) we have followed the usual convention of abbreviating  $\Gamma(k+p+1)$  as  $(k+p)!$ . Indeed, the HP-48G includes the gamma function as a built-in function  $x!$  under the menu [MTH][PROB]. Try it out, verifying that for integer arguments it returns the usual factorial values, but for non-integer arguments  $x$  it also returns a value ( $\Gamma(x+1)$ ).

---

<sup>1</sup> Under suitable conditions of regularity, namely a continuous, positive second derivative (i.e. very smooth and concave-up), the gamma function can be shown to be the unique such generalization of the factorial function to the positive real numbers.

Equation (11.4) gives *one* solution of Bessel's equation. It can be shown that the method of undetermined coefficients will always thus determine one solution  $y_1(x)$  of the form (11.2) about a *regular singular point* of a second-order differential equation, corresponding to the larger root  $c_1$  of the *indicial equation*. It can also be shown that a second linearly independent solution  $y_2(x)$  of the same form is determined by the remaining root  $c_2$  of the indicial equation whenever  $c_1 - c_2$  is not an integer. Finally, if  $c_1 - c_2$  is an integer there is always a second solution of the form

$$y_2(x) = |x|^{c_2} \sum_{k=0}^{\infty} b_k x^k + C y_1(x) \ln|x|, \quad (11.5)$$

where  $C$  is a constant. The method of undetermined coefficients can be applied in any of these cases to determine the constants. This effectively gives us a method for developing two linearly independent series solutions about any regular singular point. Details of this method are found in most of the textbooks listed at the end of the chapter.

#### PROGRAM 11.1. Bessel functions of the first kind.

The program computes  $J_p(x)$ , when  $p$  is not a negative integer. (In general when  $x$  is negative and  $p$  is not an integer we must replace  $x^{-p}$  by  $|x|^{-p}$ .)

The program monitors the size of the intermediate partial sums and estimates the accuracy of the final result. It sets the display mode to the approximate number of digits of accuracy achieved, and issues a warning if all accuracy is lost.

Store the program in the variable JS. Since the form of the program is that of a user-defined function, it can be executed by placing its two arguments on the stack and pressing the user menu key JS or by evaluating the expression 'JS(P,X)'. The HP-48G's PLOT and SOLVE applications are also available. (Its definition as a *program* instead of an *expression*, however, excludes operations that involve the derivative (e.g. SLOPE).)

Inputs		Outputs	
2:	p	2:	
1:	x	1:	$J_p(x)$

<< → p x

<< 1 0 x 2 / p ^ p ! / DUP

→ maxT k T S

<< WHILE T ABS 1E-12 >

REPEAT

T x 2 / DUP \* \*

'k' INCR / k p + / NEG 'T' STO

T S + 'S' STO

IF T ABS maxT >

THEN T ABS 'maxT' STO

END

END

10 maxT LOG 1 + IP - DUP

IF 0 >

THEN FIX S

ELSE CLLCD

STD " ALL ACCURACY LOST"

1600 .5 BEEP

3 DISP 7 FREEZE

END

>>

>>

>>

- Make it a user-defined function
- Initialize maxT, k, T, and S
- Start adding series
- Update k, T, and S
- Update maxT to monitor size of intermediate sums
- Estimate accuracy of the value returned

### EXAMPLE 11.2.

Plot the functions  $J_0(x)$ ,  $J_1(x)$  and  $J_2(x)$  on the interval  $0 \leq x \leq 10$ . We do this in the HP-48G's PLOT application, entering the list

$$\{'JS(0,X)' 'JS(1,X)' 'JS(2,X)'\}$$

as the current equation. The graphs are shown in Figure 11.1, below.

Note, in the graph, that  $J_0(x)$  has zeros at 2.4, 5.5, and 8.7, approximately. Indeed, using the HP-48G's SOLVE application, we can find these zeros more accurately as 2.404825558, 5.52007811, and 8.6537279. Even more difficult computations can be carried out easily. For example, the point of intersection of the graphs of  $J_0(x)$  and  $J_1(x)$  can be found at the point (1.434695651, 0.547946450), using the [PICTURE][FCN][ISECT] function.

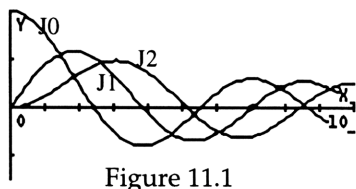


Figure 11.1

Graph of  $J_0(x)$ ,  $J_1(x)$ , and  $J_2(x)$ , drawn by the HP-48G using Program 11.1

### EXERCISE 11.1

Enter Program 11.1 into your HP-48G and verify the results of Example 11.2. Find the zeros of  $J_1(x)$  and  $J_2(x)$  visible in the above graph. Find the second point of intersection of  $J_0(x)$  and  $J_1(x)$ . Find the first point of intersection of  $J_1(x)$  and  $J_2(x)$ . (Hint: The HP-48G finds the nearest intersection of the first two functions in the current equation list. Thus it is necessary to change the order of the list.)

### EXAMPLE 11.3.

Plot the functions  $J_{\frac{1}{4}}(x)$  and  $J_{-\frac{1}{4}}(x)$  on the interval  $0.001 \leq x \leq 10$ . (Why this interval?) In the HP-48G's PLOT application, enter the list

$$\{'JS(-.25,X)' 'JS(.25,X)'\}$$

as the current equation. The graphs are shown in Figure 11.2.

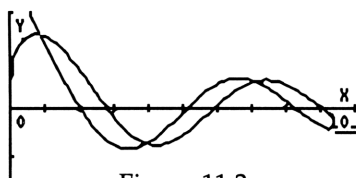


Figure 11.2

Graph of  $J_{\frac{1}{4}}(x)$  and  $J_{-\frac{1}{4}}(x)$ ,  
drawn by the HP-48G using  
Program 11.1

## 11.2 BESSEL'S EQUATION: GENERAL SOLUTIONS

The function  $J_p(x)$  is one solution of Bessel's equation, and when  $p$  is not an integer the function  $J_{-p}(x)$  is a second linearly independent solution. In other words, the general solution is  $y = c_1 J_p(x) + c_2 J_{-p}(x)$ . When  $p$  is an integer, a second solution can be found in the form of equation (11.5), containing a logarithmic term. Equation (11.6), below, gives a common form of this solution, called *Weber's form of the Bessel function of order  $n$  of the second kind*. It is described in most of the textbooks listed at the end of the chapter, and it is fully described and tabulated in [Abramowitz and Stegun].

$$Y_n(x) = \frac{2}{\pi} \left( \ln \frac{x}{2} + \gamma \right) J_n(x) - \frac{1}{\pi} \sum_{k=0}^{n-1} \frac{(n-k-1)!}{k!} \left( \frac{x}{2} \right)^{2k-n} - \frac{1}{\pi} \sum_{k=0}^{\infty} (-1)^k \frac{H_k + H_{n+k}}{k!(n+k)!} \left( \frac{x}{2} \right)^{2k+n} \quad (11.6)$$

The general solution of Bessel's equation is then  $y = c_1 J_n(x) + c_2 Y_n(x)$  when  $p = n$  is an integer. An HP-48G program that defines  $Y_n(x)$  follows. The function  $H_k$  that appears in (11.6) is the partial sum of the harmonic series  $H_k = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{k}$ ,  $H_0 = 0$ . The constant  $\gamma$  is Euler's constant  $\gamma = \lim_{m \rightarrow \infty} (H_m - \ln m) \approx 0.577215664902$ .

### PROGRAM 11.2.

The program computes  $Y_n(x)$  when  $n$  is a nonnegative integer and  $x > 0$ . It calls a subprogram to compute the function  $H_k$ . Store this subprogram in the variable H:

```

<< → n << IF n 0 ==
      THEN 0
      ELSE 0 1 n
      FOR k 1 k / +
      NEXT
      END
    >>
  >>

```

The program monitors the size of the individual terms that are generated and estimates the accuracy of the final result. It sets the display mode to the approximate number of digits of accuracy achieved, and issues a warning if all accuracy is lost.

The program itself should be stored in the variable YS. Since the form of the program is that of a user-defined function, it can be executed by placing its two arguments on the stack and pressing the user menu key YS or by evaluating the expression 'YS(N,X)'. The HP-48G's PLOT and SOLVE applications are also available. (Its definition as a *program* instead of an *expression*, however, excludes operations that involve the derivative (e.g. SLOPE).)

Inputs		Outputs	
2:	n	2:	
1:	x	1:	$Y_n(x)$

```
<< → n x
```

```
<< 1 0 → maxT S
```

```
<< 'JS(n,x)' EVAL  
x 2 / LN .577215664902 + * 'S' STO
```

```
1 → T  
<< IF n 0 >  
  THEN 0 n 1 -  
  FOR k  
    '(n-k-1)!*(x/2)^(2*k-n)/k!/2'  
    EVAL NEG  
    'T' STO  
    T S + 'S' STO  
  NEXT  
END  
>>
```

```
0 1 → k T  
<< WHILE T ABS 1E-12 >  
  k 2 < OR  
  REPEAT  
    '(-1)^(k+1)*(H(k)+H(n+k))  
    *(x/2)^(2*k+n)/k!/(n+k)!/2'  
    EVAL 'T' STO  
    T S + 'S' STO  
    'k' INCR DROP  
    IF T ABS maxT >  
    THEN T ABS 'maxT' STO  
  END  
END  
>>
```

(continued)

- Make it a user-defined function
- Initialize maxT and S
- First term of (11.6)
- Second term of (11.6) is a finite sum. Add it to S next. Skip it if  $n=0$ . (Don't monitor size of these terms—they are all of the same sign.)
- Third term of (11.6) is an infinite series. The WHILE loop adds it to S.

(continued)

```

10 maxT LOG 1 + IP - DUP
IF 0 >
THEN
  FIX 'S*2/π' →NUM
ELSE
  STD CLLCD
  "  ALL ACCURACY LOST"
  1600 .5 BEEP 3 DISP 7 FREEZE
END
>>
>>
>>

```

- Finally, estimate the accuracy achieved and set the display mode.

**EXAMPLE 11.4.**

Plot the functions  $J_0(x)$  and  $Y_0(x)$  together on the interval  $0 \leq x \leq 10$ . Also plot  $J_1(x)$  and  $Y_1(x)$  together on  $0 \leq x \leq 10$ . For the first, make the current equation the list  $\{JS(0,X) \ YS(0,X)\}$ . For the second, make it the list  $\{JS(1,X) \ YS(1,X)\}$ .

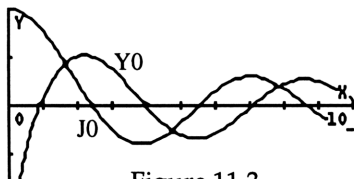


Figure 11.3

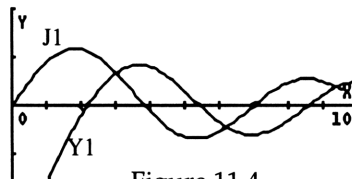


Figure 11.4

The programs that define  $J_p(x)$  and  $Y_n(x)$ , although having the form of user-defined functions, are not of the kind for which the HP-48G can calculate derivatives. Thus built-in functions that use derivatives, such as SLOPE, cannot be used. We could, of course, obtain the derivative functions by differentiating the defining series, but that would only defer the obvious need to study Bessel functions more closely. Are there, for example, simple differentiation formulas for Bessel functions? What relationships exist between Bessel functions for different values of  $p$ ? Are there other ways of representing



Bessel functions (and perhaps of computing them) besides infinite series? The student is invited to think about other important classes of functions, for example the trigonometric functions, and what it is important to know besides numerical methods for computing their values. In the case of trigonometric functions it is their *differentiation formulas* and the many *identities* relating them that render the class of functions so useful in myriad applications. Bessel functions, also, are widely studied, and many of their properties can be found in the textbooks listed at the end of the chapter as well as in reference works like [Abramowitz and Stegun]. We refer the student to these references but include here a few properties of Bessel functions that help us in our numerical calculations.

### 11.3 SELECTED PROPERTIES OF BESSEL'S FUNCTIONS

1. Behavior for small values of  $x$ :

$$\begin{aligned} J_p(x) &\sim \frac{1}{2^p p!} x^p, & J_{-p}(x) &\sim \frac{2^p}{(-p)!} x^{-p}, \\ Y_n(x) &\sim -\frac{2^n (n-1)!}{\pi} x^{-n} \quad (n \neq 0), & Y_0(x) &\sim \frac{2}{\pi} \ln x. \end{aligned} \quad (11.7)$$

2. Behavior for large values of  $x$ :

$$J_p(x) \sim \sqrt{\frac{2}{\pi x}} \cos\left(x - \frac{\pi}{4} - \frac{p\pi}{4}\right), \quad Y_n(x) \sim \sqrt{\frac{2}{\pi x}} \sin\left(x - \frac{\pi}{4} - \frac{n\pi}{4}\right). \quad (11.8)$$

3. Differentiation formulas:

$$\begin{aligned} \frac{d}{dx} J_p(x) &= -J_{p+1}(x) + \frac{p}{x} J_p(x), \\ \frac{d}{dx} Y_n(x) &= -Y_{n+1}(x) + \frac{n}{x} Y_n(x). \end{aligned} \quad (11.9)$$

## 4. Recurrence relations:

$$\begin{aligned}
 J_{p-1}(x) + J_{p+1}(x) &= \frac{2p}{x} J_p(x), \\
 Y_{n-1}(x) + Y_{n+1}(x) &= \frac{2n}{x} Y_n(x).
 \end{aligned}
 \tag{11.10}$$

## 5. Integral representations:

$$\begin{aligned}
 J_p(x) &= \frac{(\frac{1}{2}x)^p}{\sqrt{\pi}\Gamma(p + \frac{1}{2})} \int_0^\pi \cos(x \cos t) \sin^{2p} t \, dt \quad (p > -\frac{1}{2}), \\
 Y_n(x) &= \frac{1}{\pi} \int_0^\pi \sin(x \sin t - nt) \, dt - \frac{1}{\pi} \int_0^\infty [e^{nt} + e^{-nt} \cos n\pi] e^{-x \sinh t} \, dt.
 \end{aligned}
 \tag{11.11}$$

The asymptotic behaviors for small and large values of  $x$ , given in (11.7) and (11.8), are useful in limit calculations and in interpreting the graphs of Bessel functions. The differentiation formulas in (11.9) are representative of a larger number of such formulas found in the references. The recurrence relations (11.10) are useful for rewriting Bessel functions in terms of others of higher or lower orders, often to achieve more satisfactory convergence properties of the series or integral representations. The integral forms of representation (11.11) provide an alternative way to compute values of Bessel functions, and as we will show in examples are often essential for this purpose. All of these properties can be proved by direct consideration of the differential equation or of the series definitions of the functions. Details are in the references cited at the end of the chapter.

**EXAMPLE 11.5.**

Plot  $J_{-\frac{1}{3}}(x)$  with its derivative, and find the coordinates of its first relative minimum point. The differentiation formula (11.9) gives

$$\frac{d}{dx} J_{-\frac{1}{3}}(x) = -J_{\frac{2}{3}}(x) - \frac{1}{3x} J_{-\frac{1}{3}}(x).$$

Thus we can enter the list

$$\{ '-JS(2/3,X)-1/(3*X)*JS(-1/3,X)' 'JS(-1/3,X)' \}$$

as the current equation in the HP-48G's PLOT application, draw its graph on a suitable interval, and use the function [PICTURE][FCN][ROOT] (or the SOLVE application) to find the first zero of the derivative. The first relative minimum is thus found at (3.27468213, -0.43476438), and the graph, drawn by the HP-48G is shown in Figure 2.5.

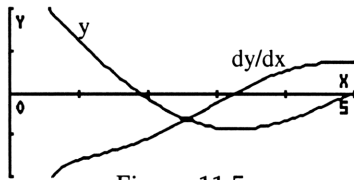


Figure 11.5

Graph of  $J_{-\frac{1}{3}}(x)$  plotted with its derivative.

#### EXAMPLE 11.6.

Examine the behavior of  $J_0(x)$  on the larger interval  $0 \leq x \leq 35$ , and find its largest zero in this interval. The plot of the function in this interval, Figure 11.6 below, exhibits the expected failure of the series for large values of  $x$ . (For this example the monitoring of the size of terms in the series was turned off.) The *smearing* effect, caused by large terms early in the series, obscures the final value which is close to zero. Less apparent is the loss of accuracy for intermediate values of  $x$ . The zero near  $x = 20$  is reported by the HP-48G, using Program 11.1, as 21.21, accurate to only 2 decimal places! The zero near  $x = 24$  is given as 24.0 (24.35 is better). Near  $x = 35$  the values are worthless.

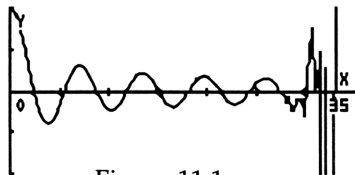


Figure 11.1

$J_0(x)$  plotted on  $0 \leq x \leq 35$ , showing the disastrous effect of *smearing* in the computation of the series defining  $J_0(x)$ .

In this situation consider using the integral formula (11.11) for computing  $J_0(x)$ . Programs 11.3 and 11.4, below, implement the integral representations for  $J_p(x)$  and  $Y_n(x)$ . They are slower than their series counterparts, however they retain *full accuracy* in their computations! The same zeros of  $J_0(x)$  reported above using the series definition are given by the integral representation as 21.2116366299 and 24.352471531. The zeros in the vicinity of  $x = 30$  are found successfully to be 27.4934791320, 30.6346064684 and 33.7758202136. The author enjoyed lunch and a short nap while the HP-48G worked to find these zeros. But all digits in the results obtained are accurate!<sup>2</sup>

#### PROGRAMS 11.3 and 11.4.

The two programs below calculate values of  $J_p(x)$  and  $Y_n(x)$  using the integral representations (11.11).

Store the programs in the variables JI and YI, respectively. The syntax of the programs permits them to be executed either from the stack or in algebraic mode. Note that the infinite integral in the equation for  $Y_n(x)$  is replaced by the finite limits 0 and 10. The integrand of this improper integral can be shown to be negligible for  $t > 10$  (cf Exercise 11.3, below).

Inputs for JI		Outputs	
2:	p	2:	
1:	x	1:	$J_p(x)$
Inputs for YI		Outputs	
2:	n	2:	
1:	x	1:	$Y_n(x)$

---

<sup>2</sup> As tabulated in [Abramowitz and Stegun].

```
<< → p x
<< RAD
'(x/2)^p/√π/(p-.5)!*
 ∫ (0,π,COS(x*COS(t))*SIN(t)^(2*p),t)'
→NUM
>> >>
```

- User-defined function for the first equation (11.11)

```
<< → n x
<< RAD 10 → U
' ∫ (0,π,SIN(x*SIN(t)-n*t),t)' →NUM
' ∫ (0,U,(EXP(n*t)+EXP(-n*t))*COS(r*π))
  *EXP(-x*SINH(t)),t)' →NUM
- π / →NUM
>> >> >>
```

- User-defined function for the second equation (11.11)

When using the integral forms, above, for drawing graphs, extreme accuracy is not needed, so it is best to limit the accuracy of the integral calculations by setting the display format to only a few decimal places. When an approximate value of a zero of the solution function is determined from the graph, the SOLVE application can then be used to find the zero to full accuracy.

### EXERCISE 11.2.

Enter the programs given above into your HP-48G and experiment with their use in computing values, drawing graphs, finding zeros, and finding maximum and minimum points. In particular compare the series and integral programs with regard to speed, accuracy, and range. Can you find the 20th zero of  $J_0(x)$ ? (Answer: 62.0484691902. Hint: equation (11.8) can suggest approximate values of the zeros.)

### EXERCISE 11.3.

Define the integrand of the infinite integral in equation (11.11) as a function in your HP-48G. Demonstrate that it is very small when  $t > 10$ , for all values of  $n$  and  $x$  of interest.

**EXAMPLE 11.7.**

Many differential equations can be solved in terms of Bessel functions. The textbooks and references each list a variety of such equations. For example [Abramowitz and Stegun] note that the differential equation

$$\frac{d^2 y}{dx^2} + \lambda^2 x^{p-2} y = 0$$

has the solution  $y = \sqrt{x} Z_{1/p}(2\lambda x^{p/2}/p)$ , where  $Z$  denotes  $J_p$  or  $Y_n$  as appropriate to the value of  $1/p$ . This enables us to give an explicit solution of the differential equation  $y'' + x^2 y = 0$  studied in Examples 1.1 and 1.2. Setting  $\lambda = 1$  and  $p = 4$  we see that the general solution of  $y'' + x^2 y = 0$  is

$$y = c_1 \sqrt{x} J_{\frac{1}{4}}(\tfrac{1}{2} x^2) + c_2 \sqrt{x} J_{-\frac{1}{4}}(\tfrac{1}{2} x^2).$$

To satisfy the initial conditions  $y(0) = 1$ ,  $y'(0) = 0$  of Example 1.1, therefore, we deduce from the formulas (11.7) that  $c_1 = 0$  and  $c_2 = \Gamma(\frac{3}{4})/\sqrt{2}$ . Hence the solution  $y_0(x)$  obtained earlier, both by the Runge-Kutta method and by Taylor series methods, is

$$y_0(x) = \frac{\Gamma(\frac{3}{4})}{\sqrt{2}} \sqrt{x} J_{-\frac{1}{4}}(\tfrac{1}{2} x^2). \quad (11.12)$$

We tried earlier, but failed, to draw the graph of this function on the interval  $0 \leq x \leq 10$ . The simple Runge-Kutta method was unable to “track” the rapid oscillations of the function. And the series solution was unable to avoid catastrophic *smearing* effects. Can we do it now? We have several ways of computing values of  $J_{-\frac{1}{4}}(x)$ . Since for  $x$  in the interval  $0 \leq x \leq 10$  the argument of  $J_{-\frac{1}{4}}$  in (11.12) will range from 0 to 50, we cannot expect to use the series program JS. So we turn to the integral representation, implemented through the program JI. With high expectations we enter, in the HP-48G’s PLOT application, the current equation

$$'(-1/4)!/\sqrt{(X/2)}*JI(-1/4,X^2/2)'$$

The calculation proceeds apace, exceeding the duration of any lunch and nap time that a reasonable person might entertain. This prompts a closer look at the integral formula, and we soon notice the trouble—the integral is *improper* when  $p$  is negative. It converges mathematically, but that is small comfort. At this point we recall the recurrence relations. From Equation (11.10) we can write

$$J_{-\frac{3}{4}}(x) = \frac{3}{2x} J_{\frac{1}{4}}(x) - J_{\frac{7}{4}}(x).$$

Thus we enter as the current equation the somewhat more complicated expression

$$(-1/4)! \sqrt{X/2} * ((3/X^2) * JI(3/4, X^2/2) - JI(7/4, X^2/2))'$$

This time we succeed! Figure 11.7 shows the graph of the function  $y_0(x)$  on the interval  $6 \leq x \leq 10$  that was problematical in Example 10.3. And we are able to compute the zeros in this interval with full precision! For example, the last zero in the interval was determined by the HP-48G's SOLVE application to be 9.90851094691.

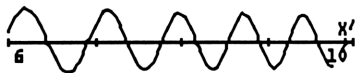


Figure 11.7

Graph of the solution of the initial-value problem (10.1) on  $6 \leq x \leq 10$ .

## 11.4 POSTSCRIPT ON NUMERICAL SOLUTION OF DIFFERENTIAL EQUATIONS

In Chapter 10 we employed a rather naive numerical method in the cause of solving an initial-value problem. Our first effort, which we called a *second-order Runge-Kutta algorithm*, was rewarded nicely, and Figure 10.1 gave reason for optimism. The algorithm used had the merit of great simplicity and intuitive appeal. It behaved well in relatively simple cases, such as in Exercise 10.2, where the solution function changed uniformly over the interval of interest. But it failed, finally, when pushed too hard—when

we asked it to plot the solution of the initial-value problem (10.1) whose solution oscillates with ever increasing frequency as  $x \rightarrow \infty$ .

In general we wish to solve initial-value problems of the form

$$\frac{dY}{dt} = F(t, Y), \quad Y(0) = Y_0 \quad (11.13)$$

in which  $Y(t)$  is a vector function. We may think of such a system of equations as determining a curve beginning at the point  $Y_0$  and having a tangent vector given by (11.13) at each point  $Y(t)$  on the curve. Most numerical differential equation solvers, then, generate an approximate solution by taking small steps along the curve, each step using the tangent vectors given by (11.13) to guide its direction. The only thing in question is the size of the step and the particular way the tangent vectors are used.

In Chapter 5 a number of examples were given using *Euler's method* and *Improved Euler's method*. Euler's method is the simplest of all, using the tangent vector given by (11.13) directly and taking steps of constant size  $h$  (in the independent variable  $t$ ). The second-order Runge-Kutta method of Section 10.1 uses a weighted average of two tangent vectors to improve accuracy. It also maintains constant step size. Further improvement is possible by increasing the mathematical sophistication of the algorithm for taking one step, and we include below as an example the *fourth order Runge-Kutta method*. Such improvements help. But our benchmark example, the initial-value problem (10.1), shows that any algorithm that insists on taking steps of constant size will eventually fail to track the oscillations. A rather full treatment of Runge-Kutta methods is given in [Numerical Recipes], where the *second-order* and *fourth-order* versions are described in detail. But the authors of that reference assert that any good integrator for ordinary differential equations should exert some *adaptive control* over its own progress, frequently changing its stepsize to match the current behavior of the solution. The built-in differential equation solver of the HP-48G does this, implementing an *adaptive fourth-order* algorithm known as the *Runge-Kutta-Fehlberg method* (RKF). Our examples, below,



compare the simple non-adaptive methods with the built-in RKF algorithm. The power, flexibility and speed of the HP-48G's built-in functions will be apparent.

For the sake of completeness (and for use with the HP-48S that does not have built-in differential equation functions) we have included programs in the Appendix that implement an adaptive fourth-order Runge-Kutta algorithm as well as the simpler non-adaptive versions. Each such differential equation solver provides an algorithm for taking a single step along the solution curve, from a point  $(t, Y(t))$  to the next point  $(t + h, Y(t + h))$ . And it also provides an integrator—a program that manages taking multiple steps to generate the solution curve to a specified stopping point. The built-in solver of the HP-48G, for example, provides the function RKFSTEP to take a single step and the function RKF to take multiple steps to a specified value of the independent variable. The HP-48G also provides user interfaces in the form of SOLVE and PLOT applications, simplifying the task of using the built-in functions. Many will find the applications sufficient for all of their purposes and will never use the functions RKFSTEP and RKF directly. We will see, however, that it is often very useful to do so.

The programs in the appendix, mainly of interest to users of the HP-48S, provide algorithms RK2, RK4 and RK4A for taking a single step along a solution curve. They also provide an integrator SOLV that generates and graphs a solution curve using any one of the single-step algorithms. And a SETUP procedure makes it convenient to apply the programs to an arbitrary system of  $n$  first-order differential equations. The Appendix contains instructions for entering and using the programs given there.

The graphs below, drawn by the HP-48G, compare the second-order and fourth-order Runge-Kutta algorithms with the built-in RKF algorithm, as applied to our benchmark initial-value problem (10.1). The zeros of the solution lying in the interval  $0 \leq x \leq 10$  are also obtained for the purpose of comparing accuracy of the algorithms. The improved stability of the built-in RKF algorithm is evident.

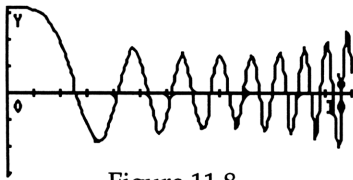


Figure 11.8

Graph of the solution of the initial-value problem (10.3) on  $0 \leq x \leq 10$  drawn by the HP-48G using the second-order Runge-Kutta method.

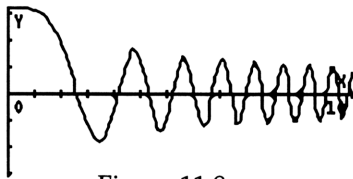


Figure 11.9

Graph of the solution of the initial-value problem (10.3)  $0 \leq x \leq 10$  drawn by the HP-48G using the fourth-order Runge-Kutta method.

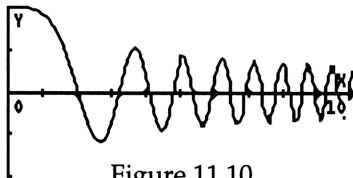


Figure 11.10

Graph of the solution of the initial-value problem (10.3)  $0 \leq x \leq 10$  using the built-in RKF algorithm.

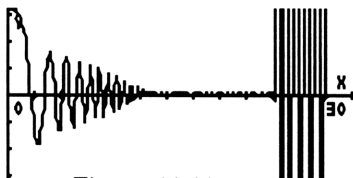


Figure 11.11

Graph of the solution of the initial-value problem (10.3) on the interval  $0 \leq x \leq 30$  using the fourth-order Runge-Kutta method.

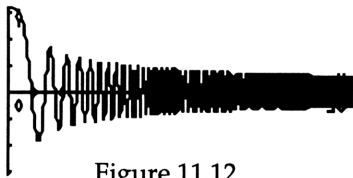


Figure 11.12

Graph of the solution of the initial-value problem (10.3) on the interval  $0 \leq x \leq 30$  using the built-in RKF algorithm.

Figures 11.8, 11.9 and 11.11 were drawn by the programs in the Appendix, following the instructions given there. The special program SETUP prompts for the functions on the right hand side of the system (10.3) as well as for the initial conditions. Figures 11.10 and 11.12 were drawn by the HP-48G's PLOT application. In this case it was necessary to provide a program to evaluate the function  $F(t, Y)$  in (11.13). For a single first-order equation this usually takes the form of an expression 'F(t,y)', entered in the PLOT application's EQ field. When a system of more than one equation is given, however, Y is a vector, and the entry into the EQ field more typically takes the form of a program. For the system of first-order equations in (10.3), for example, with T as independent variable and the *vector* Y as dependent variable, the program

```
<< Y(2)' EVAL 'T^2*Y(1)' EVAL 2 →ARRY >>
```

is the appropriate entry. And the initial values of T and Y are provided as 0 and [1, 0], respectively.

Note how badly Figure 11.11 represents the solution! The constant stepsize algorithms do not handle this example at all well except for very limited domains. Another measure of the problem, presented in the following table, is the achievable accuracy in computing zeros of the solution in the interval  $0 \leq x \leq 10$ . For the 2nd and 4th order Runge-Kutta methods the zeros were generated by the programs in the Appendix. For the built-in RKF algorithm we used the HP-48G's SOLVER application in a very appealing way that we now describe.

First, it is useful to know how the HP-48G's RKF function works. It takes three arguments from the stack as shown in the table:

Inputs	Outputs
3: {T Y F}	{T Y F}
2: accuracy desired	
1: final value of T	

The list in level three contains the names of the independent variable, the dependent variable, and the variable containing the program (or expression) that was entered, above, in the EQ field of the PLOT application. The number in level two is the desired accuracy to be maintained as the solution is generated. The number in level one is the desired final value of the independent variable. RKF starts with initial conditions stored in T and Y, generates the solution as far as the given final value of T, and terminates with T and Y holding the final point on the curve. For convenience in generating further extensions of the solution, it leaves the first two of its arguments on the stack.

We can now define a user-defined function SOL(T1) that represents the solution of the initial-value problem. Merely save the following program in the variable SOL.

```
<< → T1 << {T Y F} .00000005 T1 RKF CLEAR 'Y(1)' EVAL >>
```

If we initialize T and Y to any point on the solution curve, for example to the given initial values, then SOL(T1) will return the value of the solution at any other point T1. Behind the scenes RKF is wielding its magic, invisibly generating the solution curve between the two points, stepping along the curve with steps adapted to the local nature of the solution so as to maintain the specified accuracy. Finally, the user-defined function clears from the stack the arguments left there by RKF and leaves the value of Y(1), instead. (Note that any of the other components of the solution vector Y could have been returned instead. If we were to return Y(2) instead of Y(1) in this example, the function SOL(T1) would define the *derivative* of the solution function.)

Since this solution SOL(T1) has the form of a user-defined function, it has all the privileges of such functions. In particular the algebraic expression 'SOL(T1)' can be entered directly in the EQ field of the SOLVE application to find zeros or other characteristics of the solution. This is how the last column of the table of zeros, below, was generated.

Such direct use of the HP-48G's function RKF illustrates the great flexibility and power of its differential equation solver. Indeed, although the PLOT application is very useful for graphing the solution of a single differential equation (or system of first-order

differential equations), the direct use of the RKF functions in programs is, perhaps, their more important use. In the application at hand, the finding of zeros of the solution function, the accuracy specified for the RKF algorithm was 5E-8. The 16th zero that it found is, indeed, accurate to 1 part in 100,000,000!

### Zeros of the initial-value problem (10.1)

2nd Order	4th Order	built-in RKF
2.00203644891	2.00313594472	2.00314729270
3.19492413783	3.2009154135	3.20095692562
4.05128652588	4.06405777033	4.06397614690
4.75358445731	4.77440169509	4.77419471552
5.36231454325	5.39209207961	5.39190129467
5.90668117367	5.94602222796	5.94588151508
6.40268633618	6.45310927629	6.45252653428
6.86131541392	6.92251066537	6.92221834643
7.28957420209	7.36299391049	7.36202330847
7.69178778272	7.77815861927	7.77700811227
8.07357868843	8.17170813129	8.17095208315
8.43599505705	8.54848304089	8.54676317595
8.78272725853	8.90884476027	8.90673565134
9.11562873391	9.25522004356	9.25271736799
9.43417449644	9.58850392663	9.58622265412
9.74356252066	9.91106744133	9.90851094399

### EXERCISE 11.4.

Enter the programs of the Appendix into your HP-48G (or HP-48S). Use them to plot the solution of these initial-value problems:

$$y'' + y = 0, \quad y(0) = 0, \quad y'(0) = 1,$$

$$y'' + \frac{1}{x}y' + y = 0, \quad y(.00001) = 1, \quad y'(.00001) = 0,$$

$$y'' + e^x y = 0, \quad y(0) = 0, \quad y'(0) = 1,$$

$$3y''' - xy' + x^2y = e^x, \quad y(0) = y'(0) = 0, \quad y''(0) = \frac{1}{4}.$$

In each case, also use the built-in differential equation solver and compare results. Examine the list of the zeros of the solution accumulated by the programs (or obtained as shown above using the HP-48G's SOLVE application.) How well do the zeros found by the different algorithms agree with your expectations? The second of the above equations is Bessel's equation of order zero. The solution in that case is started slightly to the right of the point (0, 1). Why? How well does the solution agree with  $J_0(x)$ ? How well do the zeros agree with those of  $J_0(x)$ ?

## 11.5 BOUNDARY VALUE PROBLEMS

Initial-value problems arise naturally in studying physical systems whose behavior is determined by a differential equation and a complete description of the initial state of the system. In contrast to these, many problems in engineering and science lead instead to *boundary-value problems* where *initial* conditions give way to conditions imposed at several different points. A rotating shaft, for example, is modeled by a fourth-order differential equation that embodies the elasticity properties of the shaft, together with conditions imposed at each end of the shaft specifying how the shaft is constrained.

Examples frequently studied involve a linear differential equation accompanied by boundary conditions of the form

$$\begin{aligned}\alpha_1 y(a) + \alpha_2 y(b) + \alpha_3 y'(a) + \alpha_4 y'(b) &= \gamma_1, \\ \beta_1 y(a) + \beta_2 y(b) + \beta_3 y'(a) + \beta_4 y'(b) &= \gamma_2,\end{aligned}$$

where  $\alpha_i$ ,  $\beta_i$ , and  $\gamma_i$  are constants. A simple example would be the following boundary-value problem on the interval  $0 \leq x \leq L$ :

$$\begin{aligned}y'' + \lambda y &= 0, \\ y(0) &= 0, \\ hy(L) + y'(L) &= 0,\end{aligned}\tag{11.14}$$

where  $h$  is a constant. As is typical of such problems, we seek *non-trivial* solutions of the differential equation that satisfy the two end point conditions—solutions that pass

through the point  $(0, 0)$  and that satisfy the relation specified between the ordinate and slope at the other end of the interval. In general such solutions exist only for discrete values of  $\lambda$ , called *eigenvalues* or *critical values* of the boundary-value problem. The corresponding solutions are called *eigenfunctions* or *characteristic modes* of the problem.

Sometimes, as in the case of Equation (11.14), the boundary-value problem can be solved explicitly. For in this very simple case, we can find the general solution of the differential equation and apply the two boundary conditions directly. We obtain:

$$\begin{aligned} y &= C_1 \cos \sqrt{\lambda} x + C_2 \sin \sqrt{\lambda} x, \\ C_1 \cdot 1 + C_2 \cdot 0 &= 0, \\ h(C_1 \cos \sqrt{\lambda} L + C_2 \sin \sqrt{\lambda} L) + \sqrt{\lambda} (-C_1 \sin \sqrt{\lambda} L + C_2 \cos \sqrt{\lambda} L) &= 0 \end{aligned}$$

(Note that when  $\lambda$  is zero or negative the general solution takes a different form. These cases must be considered also if one wants to find *all* eigenvalues of the problem. We leave it as an exercise to show that there are *no* non-trivial solutions in these cases, i.e. no eigenvalues  $\lambda \leq 0$ .) The equations above imply immediately that  $C_1 = 0$  and  $C_2(h \sin \sqrt{\lambda} L + \sqrt{\lambda} \cos \sqrt{\lambda} L) = 0$ . Hence, there can be non-trivial solutions only if  $\lambda$  is chosen so that

$$h \sin \sqrt{\lambda} L + \sqrt{\lambda} \cos \sqrt{\lambda} L = 0.$$

With  $\mu = \sqrt{\lambda} L$  we thus seek solutions of the equation  $\tan \mu = (-1/hL)\mu$ . As expected the eigenvalues form a discrete set—corresponding to the points of intersection  $\mu_1, \mu_2, \mu_3, \dots$  of the curves as shown in the graph below (drawn by the HP-48G). The eigenvalues are then  $\lambda_i = \mu_i^2 / L^2$ ,  $i = 1, 2, 3, \dots$ . The first few values are easily computed, using the HP-48G's SOLVE function.

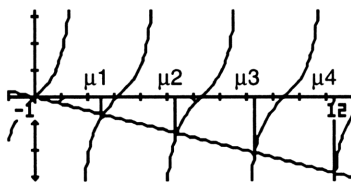


Figure 11.13

In many physical problems it may be only the first few eigenvalues that have physical significance, and in simple cases these might be computed analytically, as above. When the differential equation itself yields only to numerical methods of solution, however, we turn to numerical algorithms for finding the smaller eigenvalues. Of the two commonly used methods—*shooting methods* and *relaxation methods*—we illustrate only the former.<sup>3</sup> Consider, for example, the boundary-value problem

$$y'' + \lambda xy = 0, \quad y(0) = y(2) = 0. \quad (11.15)$$

The shooting method seeks to find the first few positive eigenvalues by choosing an additional initial condition at  $x = 0$  and determining  $\lambda$  by trial-and-error so that the (unique) solution of the *initial-value* problem satisfies the other boundary condition at  $x = 2$  as well. (The analogy with shooting a rifle bullet at a 45 degree angle, experimenting with the muzzle velocity that will cause the bullet to land at a prescribed point, comes to mind.) Which second initial condition is chosen is immaterial (so long as it does not lead to trivial solutions) since the eigenfunction corresponding to a given eigenvalue is determined only up to an arbitrary constant. Thus we will determine solutions of the initial-value problem  $y'' + \lambda xy = 0$ ,  $y(0) = 0$ ,  $y'(0) = 1$  for various values of  $\lambda$ , experimenting until the condition  $y(2) = 0$  is satisfied. The plots below, drawn by the HP-48G using the built-in differential equation PLOT application, illustrate the method. Figures 11.14 to 11.17 show the solutions for  $\lambda = 1, 2, 3, 4$ , while Figures 11.18 to 11.21 show solutions for  $\lambda = 8, 9, 10, 11$ . Notice that when  $\lambda$  is 1 or 2 the solution

---

<sup>3</sup>Both methods are discussed in [Numerical Recipes].



*overshoots* the target point  $(2, 0)$ , whereas when  $\lambda$  is 3 or 4 it *undershoots* the target. The first eigenvalue is thus seen to be between 2 and 3. And it would not be difficult (only a bit time consuming) to narrow in on the eigenvalue with further trials. We will show, below, that the HP-48G's SOLVE application can be used to find the eigenvalue more efficiently. The second group of four figures show, by similar reasoning, that the second eigenvalue lies between 9 and 11. This is the essence of the shooting method.

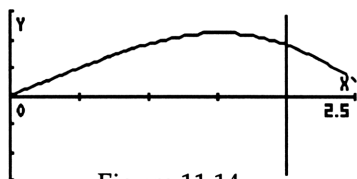


Figure 11.14

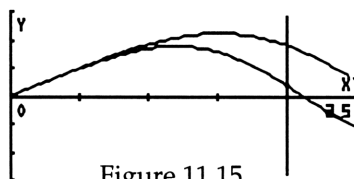


Figure 11.15

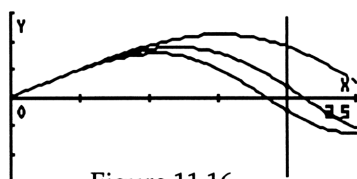


Figure 11.16

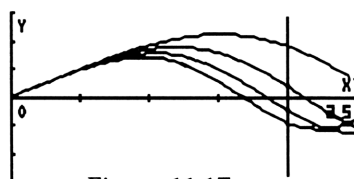


Figure 11.17



Figure 11.18

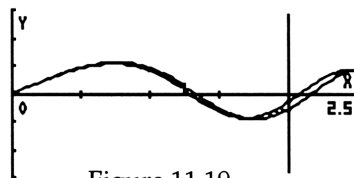


Figure 11.19



Figure 11.20

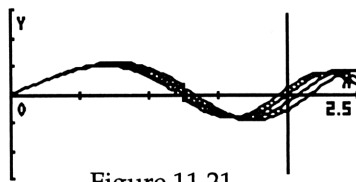


Figure 11.21

To find the eigenvalues more exactly we will define a user-defined function for the HP-48G that takes  $\lambda$  as input and returns  $y(2)$ , the value of the solution at  $x = 2$ :

```
<< → λ1
<< λ1 'λ' STO 0 'X' STO [0 1] 'Y' STO
      {X Y F} .00000005 2 RKF
      CLEAR 'Y(1)' EVAL
>> >>
```

Save this program in a variable EV. The program assumes that variables X, Y, and F exist and that F contains the function that defines the differential equation. The roots of the function  $EV(\lambda)$  are the desired eigenvalues of the boundary-value problem, thus we can use the SOLVE application with EQ set to 'EV( $\lambda$ )' to find the roots. With the display mode set to 6 digits, the initial guess 2.5 yields the value 2.369533 for the smallest eigenvalue. The guess 9.5 yields 10.235823 as the second eigenvalue. (In exercise 11.5 we will see that these values are, indeed, correct to all digits shown.)

#### EXERCISE 11.5.

Use the program, above, to verify on your HP-48G calculator that the first two eigenvalues of the boundary-value problem (3.3) are the values stated. Can you find the third eigenvalue?

#### EXERCISE 11.6.

Use Example 11.7 to show that the general solution of the differential equation is

$$y = \sqrt{x} \left( C_1 J_{\frac{2}{3}} \left( \frac{2}{3} \sqrt{\lambda} x^{\frac{3}{2}} \right) + C_2 J_{-\frac{2}{3}} \left( \frac{2}{3} \sqrt{\lambda} x^{\frac{3}{2}} \right) \right).$$

From equations (11.7) deduce that the boundary condition  $y(0) = 0$  implies  $C_2 = 0$ . Thus the second boundary condition implies that  $C_1$  remains arbitrary only if

$$J_{\frac{2}{3}} \left( \frac{2}{3} \sqrt{\lambda} 2^{\frac{3}{2}} \right) = 0.$$

Find the first several eigenvalues accurately by using your HP-48G and the program JS for computing  $J_{\frac{2}{3}}(x)$ . How do these values compare with those obtained by the shooting method?

The eigenvalues  $\lambda_0, \lambda_1, \lambda_2, \dots$  of the boundary-value problem (11.15) have been “found” in the sense that, at least in principle, they can be calculated numerically. The corresponding eigenfunctions  $\varphi_0, \varphi_1, \varphi_2, \dots$  have also been “found”—they are exactly the solutions arrived at, above, by the *shooting* method. Exercise 11.5 also showed, in fact, that the eigenvalues are given by  $\lambda_n = 9\mu_n^2/32, n = 1, 2, 3, \dots$ , where  $\mu_1, \mu_2, \mu_3, \dots$  are the positive roots of  $J_{\frac{2}{3}}(x) = 0$ ; and the eigenfunctions are

$$\varphi_n(x) = \sqrt{x} J_{\frac{2}{3}} \left( \frac{2}{3} \sqrt{\lambda_n} x^{\frac{3}{2}} \right), \quad n = 1, 2, 3, \dots \quad (11.16)$$

It is clear that there is great merit in expressing the solution of the boundary-value problem in terms of known functions. A little mathematics goes a long way! But the point of the example is to show that we are not helpless in the face of a problem that just happens to elude the class of functions in our current repertoire.

# 12

## ORTHOGONAL FUNCTIONS

Treatments of boundary-value problems lead naturally to the study of *orthogonal sequences* of functions. Two functions  $f(x)$  and  $g(x)$  are defined to be *orthogonal* (on the interval  $a \leq x \leq b$ , with respect to the *weighting function*  $p(x)$ ) if

$$\int_a^b p(x)f(x)g(x)dx = 0. \quad (12.1)$$

Most of the boundary-value problems that arise in applications belong to the class of *Sturm-Liouville Problems*, whose determining characteristic is that the eigenfunctions form an orthogonal sequence. Sturm-Liouville problems involve a second-order differential equation of the form  $\frac{d}{dx}\left(p(x)\frac{dy}{dx}\right) + [q(x) + \lambda r(x)]y = 0$ , together with boundary conditions that ensure that Equation (12.1) holds.<sup>1</sup>

Our sample problem (11.15) is a Sturm-Liouville problem with  $p(x) = 1$ , thus the sequence (11.16) of its eigenfunctions is orthogonal. It is this property that enables us to expand quite general functions  $f(x)$  in a series

$$f(x) = \sum_{n=1}^{\infty} a_n \varphi_n(x) = \sum_{n=1}^{\infty} a_n \sqrt{x} J_{\frac{1}{3}}\left(\frac{2}{3} \sqrt{\lambda_n} x^{\frac{3}{2}}\right), \quad (12.2)$$

where the coefficients are given by

---

<sup>1</sup>In the various textbook references listed it is shown that admissible boundary conditions include the common forms  $y(a) = 0$ ,  $y'(a) = 0$ , and  $\alpha_1 y(a) + \alpha_2 y'(a) = 0$ .

$$a_n = \frac{\int_0^2 f(x) \varphi_n(x) dx}{\int_0^2 \varphi_n^2(x) dx} = \frac{\int_0^2 f(x) \sqrt{x} J_{\frac{1}{3}}\left(\frac{2}{3} \sqrt{\lambda_n} x^{\frac{3}{2}}\right) dx}{\int_0^2 x J_{\frac{1}{3}}^2\left(\frac{2}{3} \sqrt{\lambda_n} x^{\frac{3}{2}}\right) dx}.$$

Note, again, that we can, in principle, calculate the coefficients  $a_n$  numerically, using the built-in integration function of the HP-48G. As might be expected, however, the integrations can often be handled by use of general formulas available in comprehensive works such as [Abramowitz and Stegun].

### EXERCISE 12.1.

Use the values of  $\lambda_1, \lambda_2, \lambda_3, \dots$  that you determined in Exercise 11.5 to determine the first few terms of the series (12.2). Use the built-in integration function of your HP-48G and the user-defined function JS (or JI) given by Program 11.1.

## 12.1 FOURIER SERIES

The common (trigonometric) Fourier series

$$\begin{aligned} f(x) &= \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx), \\ a_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos kx dx, \\ b_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin kx dx, \end{aligned} \tag{12.3}$$

arises from the Sturm-Liouville problem  $y'' + \lambda y = 0$ ,  $y(-\pi) = y(\pi)$ ,  $y'(-\pi) = y'(\pi)$ . The series effects the analysis of the function  $f(x)$  into its *fundamental vibrational modes*, and all of the popular textbooks in the subject include many applications. We include one example here to demonstrate the capacity of the HP-48G to handle Fourier series.

Consider the step function

$$f(x) = \begin{cases} -1 & -\pi < x < 0 \\ 1 & 0 < x < \pi. \end{cases}$$

Equations (12.3) in this case lead to

$$\begin{aligned} b_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin kx \, dx = \frac{2}{\pi} \int_0^{\pi} \sin kx \, dx \\ &= \frac{2}{k\pi} (1 - \cos k\pi) \\ &= \begin{cases} \frac{4}{k\pi}, & k = 1, 3, 5, \dots, \\ 0, & k = 2, 4, 6, \dots \end{cases} \end{aligned}$$

Hence the Fourier series expansion of  $f(x)$  is

$$\begin{aligned} f(x) &= \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{\sin(2k-1)x}{2k-1} \\ &= \frac{4}{\pi} \left[ \sin x + \frac{\sin 3x}{3} + \frac{\sin 5x}{5} + \dots \right]. \end{aligned}$$

We define a user-defined function FS for the HP-48G by entering the following program and storing it in the variable FS:

```
<< → N X << '4/π*Σ(K=1,N,SIN((2*K-1)*X)/(2*K-1))' → NUM >> >>
```

Then the  $n$ th partial sum of the series for a given value of  $x$  can be calculated either by entering  $n$  and  $x$  into the stack and pressing the user menu key FS, or by evaluating the algebraic expression 'FS(N,X)'. More conveniently, the algebraic expression can be made the *current equation* in the HP-48G's SOLVE or PLOT applications.

Graphs of the first several, and the 50th partial sums, drawn by the HP-48G, are shown below. Note that the pointwise convergence of the series to  $f(x)$  is nicely demonstrated. The non-uniformity of the convergence is also visible, the Gibb's phenomenon showing up clearly. The student should repeat the steps leading to these

graphs and generate additional partial sums. Further examples of Fourier series can also be found in the textbooks, and they can be explored graphically in the same manner.

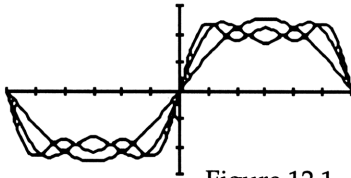


Figure 12.1

Figure 12.1 shows the first three partial sums, Figure 12.2 the 6th partial sum, and Figure 12.3 the 50th partial sum.

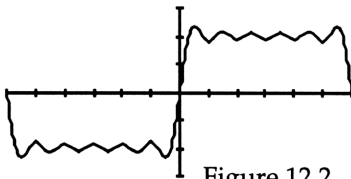


Figure 12.2

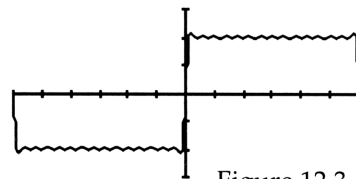


Figure 12.3

### EXERCISE 12.2.

Save the graphs of the 1st, 2nd, 3rd, ...,  $n$ th partial sums on the stack as individual pictures, (graphic objects) and then "animate the convergence" of the series. (As each graph is drawn you can save it to the stack by pressing [PICTURE] [EDIT] [NEXT] [NEXT] [PICT→]. Then enter the number of pictures on the stack and execute the ANIMATE command by pressing [PROG] [GROB] [NEXT] [ANIM].) Note: The number of graphic objects that you will be able to save on the stack depends on the amount of memory that your calculator has. With 32K of memory you may be limited to a dozen or so pictures. With 128K of memory you can save many more.

## 12.2 LEGENDRE POLYNOMIALS

Important among sequences of orthogonal functions are a number of sequences of *orthogonal* polynomials. Legendre polynomials appear, for example, as the eigenfunctions  $P_0(x)$ ,  $P_1(x)$ ,  $P_2(x)$ , ... of the Sturm-Liouville problem

$$\begin{aligned} (1-x^2)y'' - 2xy' + n(n+1)y &= 0, \\ y(-1) \text{ and } y(1) &\text{ finite.} \end{aligned} \tag{12.4}$$

The textbooks derive from the differential equation many of the important properties of these polynomials, important ones for our immediate purposes being

$$\begin{aligned} P_n(x) &\text{ is a polynomial of degree } n, \\ P_n(x) &\text{ has } n \text{ distinct zeros in the interval } -1 < x < 1, \\ P_n(x) &\text{ is an odd or even function according as } n \text{ is odd or even,} \\ P_n(1) &= 1, \text{ and } P_n(-1) = (-1)^n, \\ P_0(x) &= 1, P_1(x) = x, \text{ and } P_n(x) = \frac{2n-1}{n} P_{n-1}(x) - \frac{n-1}{n} P_{n-2}(x), \\ \int_{-1}^1 P_n(x) P_m(x) dx &= 0 \text{ if } n \neq m, \quad = \frac{2}{2n+1} \text{ if } n = m. \end{aligned} \tag{12.5}$$

The last of these equations states the orthogonality of the Legendre polynomials. We can thus expect to represent functions  $f(x)$  in *Fourier-Legendre* series

$$\begin{aligned} f(x) &= \sum_{k=1}^{\infty} a_k P_k(x), \\ a_k &= \frac{2k+1}{2} \int_{-1}^1 f(x) P_k(x) dx. \end{aligned}$$

To develop programs for computing Legendre polynomials efficiently, we turn to the recurrence relation (12.5). Starting with  $P_0(x) = 1$  and  $P_1(x) = x$ , we can compute successively as many of the polynomials as desired. Program 12.1 carries out this plan.



**PROGRAM 12.1. Generation of Legendre Polynomials.**

Program to generate the first  $n$  Legendre polynomials, store them in a subdirectory GRF, and plot  $P_n(x)$ .

Inputs		Outputs
1:	n	The subdirectory GRF contains the first n Legendre polynomials.

The program assumes that the subdirectory GRF is created before running. It also calls three subprograms SETPP, EXCO. and MULTI. The program SETPP sets the graphing parameters and is included below. The programs EXCO and MULTI completely expand an expression algebraically. They are included with Program 13.2 in chapter 13. If subdirectory GRF has not been created, the *current directory* will be cleared.

```

<< GRF CLVAR 1 'P0' STO 'X' 'P1' STO
1 'X' ROT 2 SWAP

FOR n
  DUP ROT SWAP 'X' * '(2*n-1)/n' EVAL
  * SWAP '(n-1)/n' EVAL * -
  EXCO
  CLLCD DUP "P" n + DUP " " SWAP
  " stored" + + 3 DISP OBJ→ STO
NEXT

DUP STEQ SET PP
ERASE DRAX LABEL DRAW 7 FREEZE
CLEAR UPDIR
>>

<< -1 1 XRNG -1 1 YRNG {X -1 1} INDEP >>

```

- Initialize the stack.
  
  
  
- Use the recurrence relation to generate and store n Legendre polynomials.
  
  
  
  
  
  
- Store  $P_n(x)$  in the current plotting equation and draw it.
  
  
  
  
  
- SETPP  
Program to set the plot parameters

**EXERCISE 12.3.**

Enter the first of the Programs 12.1 and save it in a variable LGN. Also enter the programs SETPP, EXCO and MULTI and create a subdirectory named GRF. Experiment! In particular run LGN with input 16 to generate the first sixteen Legendre polynomials. The subdirectory now contains the polynomial expressions.

**EXAMPLE 12.1.**

Using the expression for the sixteenth Legendre polynomial from the subdirectory GRF, we enter the following program into the HP-48G that defines a user-defined function PP.

```
<< → X '.196380615234-
      26.7077636719*X^2+
      592.022094728*X^4-
      4972.9855957*X^6+
      20424.7622681*X^8-
      45388.3605956*X^10+
      55703.8970947*X^12-
      35503.5827636*X^14+
      9171.7588806*X^16' >>
```

Store it in the variable PP. Now we can evaluate  $P_{16}(x)$  by either putting its argument on the stack and pressing the user menu key PP or by evaluating the algebraic expression 'PP(X)'. Using the HP-48G SOLVE application, we can find the roots of  $P_{16}(x)$ , only the positive roots being listed since the polynomial is an even function. The values returned are:

```
0.095012509838
0.281603550774
0.458016777679
0.617876243864
0.755404411444
0.865631200614
0.944575030377
0.989400931244
```

Comparing these roots with values listed in [Abramowitz and Stegun] we find that the last one is accurate to only 8 significant digits. The purpose of this example is to demonstrate the loss of accuracy that ensues from evaluating a polynomial naively, i.e. from its standard form. We have seen, earlier, the *smearing effect* that reduces accuracy when computing a sum—the result of adding large intermediate terms of opposite sign that contribute to a final result that is small. In the calculation carried out by the above program, we would expect to lose about five or six digits of precision since the largest terms can be about that large and the final answer is less than 1 in magnitude. Except for polynomials of quite small degree, one avoids computing them in the straightforward way. Example 12.2 shows a correct way.

### EXAMPLE 12.2.

Again we use the recurrence relation (12.5), this time avoiding the symbolic expressions for the Legendre polynomials. The program uses the recurrence relation for each value of  $x$  individually.

### PROGRAM 12.2. Evaluation of Legendre Polynomials.

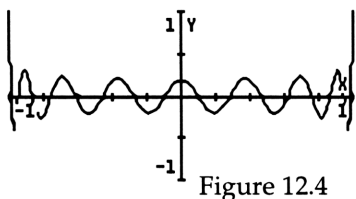
User-defined function to compute the value of  $P_n(x)$ .

Inputs		Outputs	
2:	n	1:	$P_n(x)$
1:	x		

```
<< → n x
<< 1 x 2 n
FOR k
  DUP ROT SWAP x * 2 k * 1 - k / *
  SWAP k 1 - k / * -
NEXT
SWAP DROP
>>
>>
```

- Uses the recurrence relations (5.1) for each value of  $x$  rather than symbolically.

The expression 'P(16,X)' can now be entered as the current equation in the SOLVE or PLOT application. Again we find the 8 positive zeros of the function. But this time they are accurate to the full precision of the calculator. We will list the values obtained for these zeros, below, when we use them in an important application—Gaussian 16 point quadrature.



Graph of  $P_{16}(x)$  drawn on the interval  $-1 \leq x \leq 1$ .

Applications of Legendre polynomials are nearly always related to their orthogonality properties. They arise, for example, in solving partial differential equations using the method of separation of variables. It is their origin as eigenfunctions of the Sturm-Liouville problem (12.4) that explains their appearance. Another example is their use in *Gaussian quadrature*, an important technique for numerical integration. It nicely complements the built-in integration function of the HP-48G, as we will show in the final examples in the chapter.

A continuous function  $f(x)$  can be approximated by a unique interpolating polynomial  $p(x)$  of degree  $n$  that agrees with  $f(x)$  at  $n+1$  points  $x_0, x_1, \dots, x_n$ . The existence of such a polynomial is proved most easily by exhibiting it explicitly:

$$f(x) = \sum_{k=0}^n f(x_k) L_k(x), \text{ where} \quad (12.6)$$

$$L_k(x) = \frac{(x-x_0)(x-x_1) \dots (x-x_{k-1})(x-x_{k+1}) \dots (x-x_n)}{(x_k-x_0)(x_k-x_1) \dots (x_k-x_{k-1})(x_k-x_{k+1}) \dots (x_k-x_n)}$$

$L_k(x)$  is a product of exactly  $n$  factors, thus it is a polynomial of degree  $n$ . Moreover, it is clear that it has the following properties:

$$\begin{aligned} L_k(x_k) &= 1, \quad \text{and} \\ L_k(x_i) &= 0 \quad i \neq k. \end{aligned}$$

The approximation for  $f(x)$  in (12.6) follows from these facts. It is the famous Lagrange interpolation formula for  $f(x)$ .

Many numerical integration formulas are derived by integrating the interpolating polynomial approximation (12.6) for  $f(x)$ . This yields

$$\begin{aligned} \int_a^b f(x) dx &= \sum_{k=0}^n f(x_k) \int_a^b L_k(x) dx \\ &= \sum_{k=0}^n w_k f(x_k). \end{aligned} \tag{12.7}$$

In other words the approximation to the integral of  $f(x)$  is obtained as a weighted sum of  $n+1$  values of  $f(x)$ . The weights  $w_k$  do not involve the function  $f(x)$  at all. Thus, once the weights are determined, equation (12.7) provides a scheme for numerically integrating any continuous function  $f(x)$ . The student will recognize in (12.7) many of the common integration routines such as the trapezoid rule, Simpson's rule, and others. The weights do depend, of course, on the interpolating points  $x_0, x_1, \dots, x_n$  chosen. The point of departure for Gaussian quadrature is to ask the question "For a given number  $n$ , what is the *best way* to choose the interpolating points?"

Let us take as interpolating points the sixteen zeros of  $P_{16}(x)$  in  $-1 \leq x \leq 1$ . This might seem strange at first, since so many common integration methods begin with *equally spaced* interpolation points. But we will see that it pays off handsomely. We arrive at an integration method from (12.7), known as *16 point Gaussian Quadrature*. It is only necessary that we know the weights  $w_k$ . As one might expect, they can be found in [Abramowitz and Stegun]. But we can also compute them ourselves on the HP-48G, and

it is important to realize how simply this can be done. For in some future application we may find it necessary to use a sequence of orthogonal polynomials different from the Legendre polynomials in order to meet the requirements of the application. The set of Programs 12.3, below, accomplish the computation of the required weights. And Program 12.4 implements the Gaussian quadrature integration method.

**PROGRAM 12.3. Calculation of weights for 16 point Gaussian quadrature.**

The program takes no arguments from the stack and returns none. It generates the eight weights corresponding to the eight positive zeros of  $P_{16}(x)$  (from symmetry, we know that the zeros  $\pm\alpha$  of  $P_{16}(x)$  have the same weight) and stores the weights as a list in the variable W. It assumes that the 16 zeros of  $P_{16}(x)$  are stored in the variable P16R before the program runs. Computing the weights also requires a user-defined function to compute  $L_k(x)$ . Enter this program first, and store it in the variable L.

```
<< → k x
<< 1 → v
  << 1 16 FOR i
    IF i k ≠
      THEN
        P16R i GET DUP x - SWAP
        P16R k GET - / * 'v' STO
      END
    NEXT v
  >>
>>
>>
```

- User-defined function
- Compute the product that defines  $L_k(x)$

Finally, store the following program in the variable BLDW:

```
<< { } 'W' STO
  9 16 FOR i
    'j (-1,1,L(i,X),X)' NUM
    DUP W SWAP + 'W' STO
    i 1 DISP
  NEXT
>>
```

- Initialize list W
- Calculate weights for the 8 positive roots

**PROGRAM 12.4. Gaussian quadrature (two versions).**

The two programs given here evaluate  $\int_a^b f(x) dx$ . Their use differs only in the form of the inputs required. The first expects the function  $f(x)$  in the form of an algebraic expression, for example 'SIN(X)'. The second expects a *program* for the function  $f(x)$  that takes one input from the stack and returns one value to the stack. The second is more useful in further programming applications, for example in computing double integrals (below). Store the programs in the variables GIN and PGIN.

Inputs for GIN (version 1)		Outputs
3:	'f(x)'	1: $\int_a^b f(x) dx$
2:	a	
1:	b	
Inputs for PGIN (version 2)		Outputs
3:	<< program for f(x) >>	1: $\int_a^b f(x) dx$
2:	a	
1:	b	

**GIN:** << → f a b  
 << 0 << a b DUP2 SWAP - 2 / 4 ROLL  
 \* 3 ROLLD + 2 / + >>  
 → sum u  
 <<1 8 FOR k  
 Z k GET DUP NEG  
 u →NUM 'X' STO f →NUM SWAP  
 u →NUM 'X' STO f →NUM +  
 W k GET \* sum + 'sum' STO  
 NEXT  
 sum b a - 2 / \* → NUM >>  
 >> 'X' PURGE >>

- User-defined function
- u changes coordinates from [-1, 1] to [a, b]
- Compute sum (12.7)
- Get next zero
- Multiply by weight
- Finish coordinate transformation

**PGIN:**

```

<< → f a b
<< 0 << a b DUP2 SWAP - 2 / 4 ROLL
  * 3 ROLL + 2 / + >>
→ sum u

<< 1 8 FOR k
  Z k GET u → NUM f EVAL
  W k GET * sum + 'sum' STO
  Z k GET NEG u → NUM f EVAL
  W k GET * sum + 'sum' STO
NEXT
sum b a - 2 / * → NUM
>>
>>
>>

```

- User-defined function
- u changes coordinates from [-1, 1] to [a, b]
- Compute sum (12.7)
- Next zero  $\alpha$  and its weight
- Next zero  $-\alpha$  and its weight
- Finish coordinate transformation

**EXERCISE 12.4.**

Enter the programs GIN and PGIN, above, into your HP-48G, and use them to evaluate several integrals, including the following. In each case compare the result with the value given by the HP-48G's built-in integration function. How do the computing times compare?

$$(a) \int_0^{\pi} \sin x \, dx,$$

$$(b) 4\sqrt{\frac{L}{g}} \int_0^{\frac{\pi}{2}} \frac{d\phi}{\sqrt{1 - k^2 \sin^2 \phi}}, \text{ where } \begin{cases} L = 1 \text{ meter} \\ g = 9.80665 \text{ m/s}^2 \\ k = \sin \frac{\pi}{4} \end{cases}$$

The second integral is an example of an *elliptic integral of the first kind*. With the values of  $L$ ,  $g$ , and  $k$  given, it represents the period of a simple pendulum of length 1 meter swinging with a maximum amplitude of  $\pi/2$ . It is a non-elementary integral. Note that our 16 point Gaussian integration routine gives the correct result to 12 digits of precision!



In fact, it is shown in numerical analysis texts that 16 point Gaussian quadrature gives exactly correct results for polynomials of degree  $\leq 33$ ! Simpson's rule, by way of contrast, gives exact results only for polynomials of degree  $\leq 3$ .

### EXERCISE 12.5.

The function  $F(k, x)$  defined by the integral, below, is called the *elliptic integral of the first kind*.

$$F(k, x) = \int_0^x \frac{d\phi}{\sqrt{1 - k^2 \sin^2 \phi}}$$

Write a program for your HP-48G that makes it a user-defined function that can be evaluated either from the stack or in algebraic form. Then study it by generating graphs, for various values of  $k$  with independent variable  $x$  (have the HP-48G draw a family of curves), and for various values of  $x$  with independent variable  $k$ . Find the reference work by Abramowitz and Stegun (cf. bibliography) and compare your results with the ones it catalogs.

# 13

## APPLICATIONS TO VECTOR CALCULUS

The HP-48G handles matrices and vectors well, and has built-in functions for handling the common operations on two and three dimensional vectors. Some of these were illustrated in earlier chapters, and further applications should suggest themselves to the student.

### EXAMPLE 13.1.

As an example, the problem of finding the distance between two skew lines  $L_1$  and  $L_2$  in 3-space involves evaluating the vector expression

$$dist = \frac{(\mathbf{P}_1 - \mathbf{P}_2) \cdot (\mathbf{V}_1 \times \mathbf{V}_2)}{\|\mathbf{V}_1 \times \mathbf{V}_2\|},$$

where  $\mathbf{P}_1$  and  $\mathbf{P}_2$  are points on  $L_1$  and  $L_2$ , and  $\mathbf{V}_1$  and  $\mathbf{V}_2$  are vectors parallel to  $L_1$  and  $L_2$ , respectively. On the HP-48G we can store in the variable DIST the program

```
<< → P1 P2 V1 V2 << V1 V2 CROSS DUP P1 P2 - DOT SWAP ABS / >>
```

creating a user-defined function for evaluating the distance.

### EXERCISE 13.1.

Enter the program above into your HP-48G, and find the distance between the lines determined by the vectors  $\mathbf{P}_1 = [1 \ 2 \ 3]$ ,  $\mathbf{P}_2 = [-2 \ 3 \ -5]$ ,  $\mathbf{V}_1 = [1 \ 0 \ 2]$ , and  $\mathbf{V}_2 = [-3 \ 5 \ 8]$ . (You should obtain the value 1.33955 for the distance.)

### 13.1 ANALYSIS OF SPACE CURVES

Let  $\gamma(t) = [a \cos t, a \sin t, bt]$  be a curve in 3-space. The curve is a helix spiraling around the  $z$ -axis. We can study the geometry of this curve by computing the following quantities that characterize its shape and orientation:

$$\begin{aligned} \mathbf{V} &= \text{velocity vector} = \frac{d\gamma}{dt}, \\ v &= \text{speed} = \|\mathbf{V}\|, \\ \mathbf{U} &= \text{unit tangent vector} = \frac{\mathbf{V}}{\|\mathbf{V}\|}, \\ \frac{d\mathbf{U}}{dt} &= v\kappa\mathbf{N}, \quad \mathbf{N} = \text{unit normal vector}, \quad \kappa = \text{curvature}, \\ \mathbf{B} &= \mathbf{U} \times \mathbf{N} = \text{unit binormal vector}, \\ \frac{d\mathbf{B}}{dt} &= -v\tau\mathbf{N}, \quad \tau = \text{torsion}. \end{aligned} \tag{13.1}$$

The various textbooks discuss these quantities and their relation to the curve  $\gamma(t)$ . The vectors  $\mathbf{U}$  and  $\mathbf{N}$  determine the osculating plane of the curve  $\gamma(t)$  at a point  $\mathbf{P}$ . The constant  $1/\kappa$  is the radius of curvature of the curve at  $\mathbf{P}$ . The vector  $\mathbf{B}$  is perpendicular to the osculating plane, and its rate of change measures the rate at which the curve tends to twist out of its osculating plane (hence the term *torsion*).

To compute these quantities for a given space curve  $\gamma(t) = [x(t), y(t), z(t)]$ , using the HP-48G, we assume that the three coordinate functions  $x(t)$ ,  $y(t)$  and  $z(t)$  are defined as user-defined functions. Thus, for example, the three variables X, Y and Z might contain programs

$$\begin{aligned} &<< \rightarrow t 'A1*\text{COS}(t)' >> \\ &<< \rightarrow t 'A1*\text{SIN}(t)' >> \\ &<< \rightarrow t 'B1*t' >> \end{aligned} \tag{13.2}$$

and the variables A1 and B1 contain constants. The essential feature of these programs is only that they take one argument from the stack and return their value to the stack. We could write them in many different ways, for example the first, which is written in (14.2) as a user-defined function, could be written `<< COS A1 * >>` instead. The following programs then compute the various quantities that describe the space curve.

**PROGRAM 13.1. Analysis of a space curve.**

The programs below compute the various elements of a space curve defined in Equation (6.1). A numerical differentiation routine is used in place of the HP-48G's built-in symbolic differentiation, which is not readily used with the vector functions. All of the programs take a single input  $t$  from the stack and return their result to the stack, with the exception of the numerical differential function DER. It takes two arguments from the stack—the quoted name of a vector function and a value of  $t$ . It returns to the stack the vector derivative of the named vector function, evaluated at  $t$ .

Each program is labeled with the variable name in which it should be stored. To use the programs, store definitions of the three coordinate functions  $x(t)$ ,  $y(t)$  and  $z(t)$  in the variables X, Y and Z.

**P:**

```
<< → t
  << 'X(t)' →NUM
    'Y(t)' →NUM
    'Z(t)' →NUM
    3 →ARRY
  >> >>
```

- Return the vector  $P(t)$  to the stack

**DER:**

```
<< .0001 → v t h
  << t h + v EVAL t h - v EVAL -
    2 h * /
>> >>
```

**V:**

```
<< → t
  << 'P' t DER
>> >>
```

**U:**

```
<< → t
  << 'V(t)' →NUM DUP ABS /
>> >>
```

**N:**

```
<< → t
  << 'U' t DER DUP ABS /
>> >>
```

**B:**

```
<< → t
  << 'U(t)' →NUM 'N(t)' →NUM CROSS
>> >>
```

**SPD:**

```
<< → t
  << 'V(t)' →NUM ABS
>> >>
```

**CURV:**

```
<< → t
  << 'U' t DER ABS 'SPD(t)' →NUM /
>> >>
```

**TORS:**

```
<< → t
  << 'B' t DER ABS NEG 'SPD(t)' →NUM /
>> >>
```

- Numerical differentiation

- $\mathbf{V} = \frac{d\mathbf{P}}{dt}$

- $\mathbf{U} = \frac{\mathbf{V}}{\|\mathbf{V}\|}$

- $\mathbf{N} = \frac{\frac{d\mathbf{U}}{dt}}{\left\| \frac{d\mathbf{U}}{dt} \right\|}$

- $\mathbf{B} = \mathbf{U} \times \mathbf{N}$

- $v = \|\mathbf{V}\|$

- $\kappa = \frac{1}{v} \frac{d\mathbf{U}}{dt}$

- $\tau = \frac{1}{v} \left\| \frac{d\mathbf{B}}{dt} \right\|$

**EXERCISE 13.2.**

Enter the programs 13.1 into your HP-48G, and use them to compute the velocity vector, speed, unit normal vector, curvature, etc. for various values of  $t$ . The helix is a very simple curve for which the curvature and torsion are constants. Verify that the vectors  $\mathbf{U}$ ,  $\mathbf{N}$ , and  $\mathbf{B}$  are indeed perpendicular. What accuracy is obtained by the programs?

**EXERCISE 13.3.**

Study the space curve  $\gamma(t) = [t, 1+t^2, t^3-3t+1]$ ,  $0 \leq t \leq 1$ . What are its curvature and torsion when  $t = \frac{1}{2}$ ? What is the length of the curve? (Hint:  $L = \int_0^1 v(t) dt$ . Since SPD is a user-defined function the HP-48G is able to evaluate the expression  $\int (0,1,SPD(T),T)$ .)

**13.2 LINE INTEGRALS**

Line integrals of the form

$$\int_{\gamma} P dx + Q dy + R dz,$$

where  $\gamma$  is a curve in 3-space and  $P$ ,  $Q$  and  $R$  are functions of  $x$ ,  $y$  and  $z$ , occur often in applications of vector calculus. In particular, if

$$\mathbf{F}(x, y, z) = [P(x, y, z), Q(x, y, z), R(x, y, z)]$$

is a vector field that represents a force exerted on a particle at the point  $(x, y, z)$ , then the line integral

$$\int_{\gamma} P dx + Q dy + R dz = \int_{\gamma} \mathbf{F} \cdot d\gamma$$

expresses the work done by the force field on the particle as it moves along  $\gamma$ .

As a specific example, let us compute the line integral  $\int_{\gamma} \mathbf{F} \cdot d\gamma$  where  $\gamma$  is the curve  $\gamma(t) = [t, 1+t^2, t^3-3t+1]$ ,  $0 \leq t \leq 1$  of Exercise 13.3, and the force field is given by  $\mathbf{F}(x, y, z) = [-x, y, xy + z^2]$ . Then the work done by the force field as the particle moves along the curve is

$$W = \int_{\gamma} \mathbf{F} \cdot d\gamma = \int_a^b [P(x, y, z)x'(t) + Q(x, y, z)y'(t) + R(x, y, z)z'(t)] dt$$

It is tempting to try performing this calculation symbolically on the HP-48G. We do this by defining the functions  $x(t)$ ,  $y(t)$ ,  $z(t)$  and  $P(x, y, z)$ ,  $Q(x, y, z)$ ,  $R(x, y, z)$  as user-defined functions:

```
'X(T)=T' [DEF]
'Y(T)=1+T^2' [DEF]
'Z(T)=T^3-3*T+1' [DEF]
'P(X,Y,Z)=-X' [DEF]
'Q(X,Y,Z)=Y' [DEF]
'R(X,Y,Z)=X*T+Z^2' [DEF]
```

We also store the limits of integration in variables:

```
0 'A' STO
'2*π' 'B' STO
```

The variables A B X Y Z P Q R now appear on the user menu, and they are user-defined functions. (They can therefore be evaluated from algebraic notation and can be differentiated by the HP-48G. ) We now enter the integrand of the line integral as an expression, and store it in a variable L3IN (so we never have to type it in again):

```
'P(X(T),Y(T),Z(T))*∂T(X(T))+Q(X(T),Y(T),Z(T))*∂T(Y(T))
+R(X(T),Y(T),Z(T))*∂T(Z(T))'
```

Finally, we enter the following programs:

**PROGRAM 13.2 Computation of a Line Integral in 3-space.**

The program LLL evaluates the integrand L3IN of the line integral repeatedly, carrying out the indicated differentiations and substitutions. It then computes the integral of the resulting function of  $t$ .

Each program is labeled with the variable name in which it should be stored. They assume that you have defined the variables A B X Y Z P Q R as described above.

**LLL:**

```
<< L3IN EVCO A B 3 ROLL 'T' ] →NUM >>
```

- Simplify and integrate.

**EVCO:**

```
<< << EVAL >> MULTI >>
```

- EVCO evaluates the expression repeatedly.

**EXCO:**

```
<< << EXPAN >> MULTI
<< COLCT >> MULTI >>
```

- See the HP-48G Owner's Manual, Volume II (p 569) for EXCO and MULTI. They accomplish the complete algebraic expansion and collection of terms.

**MULTI:**

```
<< → p
<< DO
    DUP p EVAL DUP ROT
    UNTIL SAME
    END
>>
>>
```

**EXERCISE 13.4.**

Enter the programs 13.2 and evaluate the line integral of  $F(x, y, z) = [-x, y, xy + x^2]$  over the curve given in Example 13.3 over various intervals. Also evaluate the following line integrals.



$$\int_{\gamma} \mathbf{F} \cdot d\gamma \quad \text{where} \quad \begin{cases} \mathbf{F}(x, y, z) = [x, -yz, e^z] \text{ and} \\ \gamma(t) = [t^3, -t, t], 0 \leq t \leq 1 \end{cases}$$

$$\int_{\gamma} xyz^2 dz \quad \text{where } \gamma(t) = [t^2, t^3, 2t^2].$$

It is instructive to evaluate the integrals, above, “manually” rather than using the program LLL. Press L3IN to put the integrand on the stack. Then press EVCO to completely evaluate the expression (doing all differentiations and substitutions). Finally press EXCO to expand algebraically and collect terms.

Note: Try integrating the integrand L3IN without first completely evaluating it. The integration routine is confused by the extra variables and gives erroneous answers. The use of the evaluation routine EVCO is *essential* in the program LLL.

### EXERCISE 13.5.

Construct a version of Program 13.2 for computing a line integral in two dimensions. The modifications are minor. Store the modified version in the variable LL. We will use it in the next section. (Note that the 3-dimensional version LLL can be used to compute line integrals in the plane. Simply set  $Z(t) = 0$  and  $R(x, y, z) = 0$ . It is nevertheless convenient to have a version dedicated to two-dimensional problems.)

## 13.3 DOUBLE INTEGRALS

Green’s Theorem in the plane states that, under suitable conditions of continuity and smoothness, an important relationship holds between a *line integral*

$$\oint_{\gamma} \mathbf{F} \cdot d\gamma = \oint_{\gamma} P dx + Q dy$$

around a simple closed curve  $\gamma$  in the plane, and a *double integral* over the region  $D$  of the plane enclosed by the curve:

$$\oint_{\gamma} F \cdot d\gamma = \iint_D \left( \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) dx dy.$$

This relationship is one of several forms that the fundamental theorem of calculus takes in higher dimensions. It, along with its higher dimensional cousins (Stokes theorem), has important applications and interpretations for fluid flow and conservative force fields. Note indeed that it gives immediately a sufficient condition that a force field be conservative, namely that  $\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} = 0$  hold throughout the region  $D$ .

We will verify Green's theorem by calculating the integrals in a number of examples. We already know how to evaluate the line integral, of course. So we now consider the calculation of the double integral. This is often done by expressing the multiple integral as an *iterated integral*

$$\begin{aligned} \iint_D f(x, y) dx dy &= \int_a^b \int_{c(x)}^{d(x)} f(x, y) dy dx \\ &= \int_a^b dx \int_{c(x)}^{d(x)} f(x, y) dy \end{aligned}$$

or a sum of several such integrals, depending on the shape of the region  $D$ . The latter can then be evaluated by entering the expression

$$' \int (A, B, \int (C(X), D(X), F(X, Y), Y), X) '$$

into the HP-48G and executing  $\rightarrow \text{NUM}$ . The key, of course, is to define the variables  $A$ ,  $B$  and the *user-defined functions*  $C$ ,  $D$ , and  $F$  before evaluating the integral expression. The following program is arranged to make this convenient.

**PROGRAM 13.3. Evaluation of a Double Integral.**

The program takes six arguments from the stack, as shown below, and returns the value of the iterated integral. It creates the user-defined functions needed.

Inputs		Outputs
6:	'F(X,Y)'	$\int_a^b dx \int_{c(x)}^{d(x)} f(x,y) dy$
5:	'C(X)'	
4:	'D(X)'	
3:	A	
2:	B	
1:	<number of decimal places>	1:

```
DBL: << 6 DUPN FIX
      'B' SWAP = DEFINE
      'A' SWAP = DEFINE
      'D(X)' SWAP = DEFINE
      'C(X)' SWAP = DEFINE
      'F(X,Y)' SWAP = DEFINE
      ' ∫ (A,B, ∫ (C(X),D(X),F(X,Y),Y),X)' →NUM
      {F C D A B} PURGE
      >>
```

- Copy arguments, set output mode
- Define the functions from expressions on the stack
- Evaluate the integral
- Clean up the mess

**EXAMPLE 13.2.**

Find the volume under the paraboloid  $z = 1 + 2x^2 + 3y^2$  and lying over the region of the  $xy$ -plane bounded by the  $x$ -axis and the curve  $y = \sin(x)$ . We enter the following inputs into the HP-48G:

```
'1+2*X^2+3*Y^2'
0
'SIN(X)'
0
'π'
6
```

The program returns the value 15.072542. The computation time is a couple of minutes.

**EXAMPLE 13.3.**

The area in the  $xy$ -plane lying below the parabola  $y = 4 - x^2$  and above the hyperbolic cosine curve  $y = \cosh x$  can be found by evaluating a double integral with  $f(x, y) = 1$ . The points of intersection of the two curves can first be found with the HP-48G's SOLVE application. The value 1.37617667019 is returned, and we store this in a variable RT. We finally enter the inputs

```
1
'COSH(X)'
'4-x^2'
'-RT'
'RT'
5
```

and after a few moments the result 5.56470 is returned. An enormous amount of computation is done easily!

**EXAMPLE 13.4.**

Use Green's Theorem to evaluate the line integral  $\int_{\gamma} \mathbf{F} \cdot d\gamma$  around the curve  $\gamma$  consisting of pieces of the  $x$ -axis and of the parabola  $y = 4 - x^2$  (the student should draw a sketch of these curves), the curve being traversed in a counter-clockwise direction. Let the vector function be

$$F(x, y) = [2y^3 e^x, 3x^2 - 4y^2].$$

We must evaluate the double integral

$$\iint_D \left( \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) dx dy = \iint_D (6x - 6y^2 e^x) dx dy.$$

With the inputs to the program DBL:

```
'6*X-6*Y^2*EXP(X)'
0
'4-X^2'
-2
2
5
```

we obtain -291.04903. Using Program 13.2 we also evaluate the line integral around the boundary, breaking the computation into 2 parts corresponding to the piece of the  $x$ -axis and the parabolic part. The line integral along the  $x$ -axis is zero (verify this). For the parabolic piece we use the parameterization  $\gamma(t) = [-t, 4 - t^2]$ ,  $-2 \leq t \leq 2$ . Program 13.2 gives exactly the same answer as did the double integral, above, in confirmation of Green's Theorem.

We conclude this survey of examples of the use of the HP-48G by returning to the Gaussian integration routine, Program 12.4, of Chapter 12. We are motivated by the very long computing times for double integrals experienced when using the built-in integration function of the HP-48G. A second difficulty in computing such integrals arises in cases where the functions  $c(x)$  and  $d(x)$  have some irregularity such as an infinite derivative at points of the interval  $a \leq x \leq b$ . In such cases the computing time of the program DBL can be prohibitive. In Program 13.4 we evaluate iterated integrals, using Gaussian quadrature for both the inside and outside integral. In doing this we use the second version of Gaussian integration that requires *programs* as inputs rather than *expressions*.

**PROGRAM 13.4. Evaluation of a Double Integral using Gaussian quadrature.**

The program takes five arguments from the stack, as shown below, and returns the value of the iterated integral. It creates the variables used in the Gaussian quadrature routine by defining them as user-defined function.

Inputs		Outputs
5:	'F(X,Y)'	$\int_a^b dx \int_{c(x)}^{d(x)} f(x,y) dy$
4:	'C(X)'	
3:	'D(X)'	
2:	A	
1:	B	

**DINT:**

<< 5 DUPN	• Copy the arguments
'B' SWAP = DEFINE 'A' SWAP = DEFINE 'D(X)' SWAP = DEFINE 'C(X)' SWAP = DEFINE 'F(X,Y)' SWAP = DEFINE	• Define the functions from expressions on the stack
<< G >> A →NUM B →NUM PGIN	• Do the outside integration.
{F C D A B} PURGE >>	• Clean up the mess
G: << → x << 1 16 START x NEXT << F >> x C →NUM x D →NUM PGIN >> >>	• Inside integration. • F(x, y) is called 16 times by PGIN

**EXAMPLE 13.5.**

Let us compare the programs DBL and DINT by computing the double integral

$$\int_0^1 \int_1^{e^x} \sin(x+y) dy dx.$$

We place the inputs 'SIN(X+Y)' 1 'EXP(X)' 0 1 11 on the stack and execute DBL. After 30 minutes (!) of computing we obtain the value 0.49242316000. We then place the first five of these inputs on the stack and execute DINT. In about 40 seconds we obtain the result 0.492423159996!

**EXAMPLE 13.6.**

Find problems involving Green's Theorem in your textbook and use Programs 13.2 to 13.4 to evaluate the line integrals and double integrals. Compare Programs 13.3 and 13.4.

**EXAMPLE 13.7.**

Find the volume of a sphere of radius 1 by writing the volume as the iterated integral

$$volume = \int_{-1}^1 \int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} 2 \sqrt{1-x^2-y^2} dy dx.$$

The infinite derivatives of  $c(x)$  and  $d(x)$  at the end points of the interval  $-1 \leq x \leq 1$  are troublesome for most numerical integration methods. How well does the HP-48G do? How well does Gaussian quadrature do?

## PROGRAMS FOR THE ADAPTIVE 4th ORDER RUNGE-KUTTA METHOD

The programs here are based on algorithms presented in *Numerical Recipes*, Press, Flannery et al. Three algorithms are included—the 2nd order Runge-Kutta method, the 4th order Runge-Kutta method, and a 4th order Runge-Kutta method with adaptive stepsize. A single driver program SOLV is provided in which the call to take a single step by way of one of these algorithms can be any of STP2, STP4 or STP4A.

The driver program plots the solution as it progresses, and it accumulates a list of all zero-crossings encountered. The zeros are determined by linear interpolation between pairs of points for which a sign change occurred. It is relatively easy to modify the driver so that it also stores lists of zeros of the derivatives.

In practice the 2nd order Runge-Kutta algorithm is useful for drawing simple graphs quickly when a high degree of accuracy is not required. It works quite well for solution functions that are relatively smooth and that have bounded rate of change. The 4th order Runge-Kutta algorithm often performs more satisfactorily, its larger admissible stepsize more than making up for the greater computing cost for each step.

### PROGRAM Adaptive 4th Order Runge-Kutta Algorithm.

A package of programs is given for solving an initial-value problem involving a system of first-order differential equations:



$$\begin{aligned}
\frac{dy_1}{dt} &= f_1(t, y_1, y_2, \dots, y_n) \\
\frac{dy_2}{dt} &= f_2(t, y_1, y_2, \dots, y_n) \\
&\dots \\
\frac{dy_n}{dt} &= f_n(t, y_1, y_2, \dots, y_n) \\
y_1(t_0) &= y_1', \quad y_2(t_0) = y_2', \quad \dots, \quad y_n(t_0) = y_n'
\end{aligned}$$

The user will normally use three programs — SETUP, INIT, and SOLV, in that order — by pressing the user menu keys bearing those names. SETUP prompts for the inputs defining the system of equations and puts them on the stack in appropriate form. INIT initializes all global variables. And SOLV does the rest. The comments below are merely explanatory with regard to the other programs in the package.

The programs RK2, RK4 and RK4A, perform one step by the *second-order*, *fourth-order* or *adaptive fourth-order* Runge-Kutta algorithms, respectively. RK2 and RK4 each take four arguments from the stack — the current value of  $t$ , a vector  $[Y1, Y2, \dots, Yn]$  giving the current values of the dependent variables, a vector  $[Y1', Y2', \dots, Yn']$  giving the current values of the derivatives of the dependent variables, and the current value of the stepsize  $H$ . RK4A takes only the first three of these arguments. They all return the new values of  $t$  and  $[Y1, Y2, \dots, Yn]$  to the stack. In addition, RK4A modifies the global variable  $H$ .

The driver program SOLV orchestrates the setting up of the graph, taking repetitive steps, and doing the necessary bookkeeping. Choose the desired step algorithm RK2, RK4 or RK4A by editing the first line of SOLV. It assumes that global variables  $H$ ,  $N$ , and  $F$  are available —  $N$  containing the number of equations in the system,  $H$  containing the current stepsize, and  $F$  containing a *list* of programs for the right-hand sides of the differential equations.

The programs SETUP and INIT take care of all the details of setting up the variables called for above. SETUP prompts the user for the functions  $f_1, f_2, \dots, f_n$  in the form of algebraic expressions (SETUP converts them to program form), the initial conditions in the form of a list  $\{t_0, y_0, y_1, \dots, y_n\}$  and the graph parameters in the form of a list  $\{x_{\min}, x_{\max}, y_{\min}, y_{\max}, h_{\max}\}$ . The value of  $h_{\max}$  is used by RK2 and RK4 as the stepsize  $h$ . It is used by RK4A as an upper bound on the stepsize. (RK4A decides what size steps to take, but it is often useful to limit the maximum stepsize according to the requirements of plotting; for example the stepsize might be limited to 1 pixel.) The program SETUP merely puts the appropriate inputs on the stack. INIT actually sets up the global variables used by all the other routines.

The programs REVW (review) and SVST (save stack) make it convenient to save the list of inputs and to recall them to the stack. SVST is executed automatically by the other routines. Thus REVW can be used at any time to restore the list of inputs originally constructed by SETUP. In situations where the user is solving many related initial-value problems, for example changing only the initial conditions, it is easier to use REVW to restore the inputs, edit them directly, and then use INIT and SOLV again.

**RE VW:**

```
<< STK OBJ→ DROP >>
```

- Recall previous inputs

**INIT:**

```
<< SVST
  OBJ→ DROP DUP 'HMAX' STO 'H' STO
  YRNG XRNG ERASE DRAX LABEL
  OBJ→ 1 - 'N' STO N →ARRY
  N 2 + ROLLD N 2 + ROLLD
  N →LIST 'F' STO >>
```

- Put starting point on the stack and initialize the graph

```

SOLV: << << STP2 >> → stepAlg
      << ERASE DRAX LABEL {#0 #0} PVIEW
      { } 'RTLS' STO
      DO
        DUP2 stepAlg EVAL
        4 PICK 4 PICK 1 GET R→C
        3 PICK 3 PICK 1 GET R→C LINE
        IF 3 PICK 1 GET 2 PICK 1 GET * 0 ≤
        THEN INTRP RTLS SWAP + 'RTLS' STO
        END
        4 ROLL 4 ROLL DROP2
      UNTIL
      OVER PPAR 2 GET RE >
    END

```

```
>> >>
```

```

STP2:
  << DUP2 DYDX H RK2 >>

```

```

STP4:
  << DUP2 DYDX H RK4 >>

```

```

STP4A:
  << DUP2 DYDX RK4A >>

```

```

RK2:
  << → x Y DY h
    << DY h * 2 / Y +
      x h 2 / + SWAP DYDX
      h * Y + x h + SWAP

```

```
>>>>
```

```

RK4:
  << 0 0 → x Y DY h h2 x2
    << x h 2 / + 'x2' STO
    DY h * DUP 2 / Y +
    x 2 SWAP DYDX h * DUP 2 / Y +
    x 2 SWAP DYDX h * DUP Y +
    x h + SWAP DYDX h * SWAP
    2 * + SWAP 2 * + + 6 / Y +
    x h + SWAP
  >> >>

```

- **CHANGE ME**  
to STP4 or STP4A
- **Main program loop.**  
Generate points  
until the graph  
is finished.
- Take one step of  
RK2
- Take one step of  
RK4
- Take one step of  
RK4A
- RK2 step
- RK4 step

**RK4A:**

```

<< 0 0 0 → x Y DY scale err done
<< 1 N FOR k
  '1+ABS(Y(k))+H*ABS(DY(k))' EVAL
NEXT N →ARRAY 'scale' STO

DO

  x Y DY H 2 / RK4
  DUP2 DYDX H 2 / RK4 DUP
  IF 3 PICK x ==
  THEN
    DROP 880 .5 BEEP HALT

  ELSE
    x Y DY H RK4
    SWAP DROP - OBJ→ DROP 0 'err' STO
    N 1 FOR k
      ABS scale k GET / DUP
      IF err >
      THEN 'err' STO
      ELSE DROP
      END -1
    STEP

    err EPS / 'err' STO
    IF err 1 ≤
    THEN
      err -.2 ^ .9 * H * 'H' STO
      1 'done' STO
    ELSE
      err -.25 ^ .9 * H * HMAX MIN
      'H' STO DROP2
    END
  END
UNTIL done 1 ==
END
>>
>>

```

- RK4A step
- Set up scaling
- Loop until the right stepsize is found
- Take 2 halfsize RK4 steps
- Abort if no progress is being made
- Take a full RK4 step and estimate the error by comparing it with the 2 halfsize steps
- If error is small, reduce stepsize and proceed
- Otherwise decrease stepsize and try again

**SETUP:**

```

<< STD "How many equations?"
" :N: " INPUT OBJ→ 'N' STO
1 N FOR i
CASE
  N 1 ==
  THEN
    "<< → T X" "f" i + "(T,X)" +
  END
  N 2 ==
  THEN
    "<< → T X Y" "f" i + "(T,X,Y)" +
  END
  N 3 ==
  THEN
    "<< → T X Y Z" "f" i + "(T,X,Y,Z)" +
  END
    "<< → T" "f" i + "(T" +
    1 N FOR j
      ",Y" j + + SWAP " Y" j + + SWAP
    NEXT
    ") ?" +
  END
  { "" 2 ALG V } INPUT
">>" + + OBJ→
NEXT

  CLLCD " Initial conditions?"
  3 DISP .5 WAIT
CASE
  N 1 == THEN "{T X} ?" END
  N 2 == THEN "{T X Y} ?" END
  N 3 == THEN "{T X Y Z} ?" END

  "{T"
  1 N FOR j
    " Y" + j +
  NEXT "}" ?" +
END
{ "{}" 2 V } INPUT OBJ→
      (continued)

```

- SETUP routine
- Prompt user for the input equations, and put the appropriate form of definition on the stack (user-defined function)
- Prompt user for initial conditions and package them into a list

(continued)

```

      CLLCD " Graph parameters ?"
      3 DISP .5 WAIT
      "{X1 X2 Y1 Y2 H} ?"
      { "{}" 2 V } INPUT OBJ→
  >>

```

- Prompt user for the graph parameters and for the value of h

**DYDX:**

```

  << 1 N FOR k
    DUP2 OBJ→ DROP F k GET EVAL 3 ROLLD
  NEXT
  DROP2 N →ARRAY
  >>

```

- Calculate the next direction vector

**INTRP:**

```

  << 4 PICK 4 PICK 1 GET
    4 PICK 4 PICK 1 GET
    DUP 4 PICK - 5 ROLLD
    4 ROLL * 3 ROLLD * - SWAP /
  >>

```

- Interpolate between the last two points generated

**SVST:**

```

  << { } 'STK' STO STK
    1 N 2 + START
    SWAP 1 →LIST SWAP +
  NEXT
  'STK' STO REVW
  >>

```

- Save the input arguments for review or editing

**RE VW:**

```

  << STK OBJ→ DROP

```

- Recall inputs to the stack for editing

## SOME TEXTBOOK REFERENCES FOR PART III

1. Handbook of Mathematical Functions; Edited by Milton Abramowitz and Irene A. Stegun; Dover Publications, Inc., New York
2. Numerical Recipes in Pascal, The Art of Scientific Computing; Press, Flannery, Teukolsky, Vetterling; Cambridge University Press, Cambridge, New York, Melbourne, Sidney.
3. Introduction to Applied Mathematics; Gilbert Strang; Wellesley-Cambridge Press, Wellesley, Massachusetts.
4. An Introduction to Linear Analysis; Kreider, Kuller, Ostberg, Perkins; Addison-Wesley, Reading, Massachusetts.
5. Advanced Calculus for Engineers; Francis B. Hildebrand; Prentice-Hall, New York.
6. Advanced Calculus for Applications; Francis B. Hildebrand; Prentice-Hall, New York.
7. Advanced Engineering Mathematics; Peter V. O'Neil; Wadsworth, Belmont, California.
8. Mathematics of Physics and Modern Engineering; Sokolnikoff and Redheffer; McGraw-Hill, New York, San Francisco, London.
9. Mathematical Methods for Physicists; George Arfken; Academic Press, New York, San Francisco, London.
10. Advanced Engineering Mathematics; Erwin Kreyszig; John Wiley and Sons, New York, London.
11. Foundations of Applied Mathematics; Michael Greenberg; Prentice-Hall, Englewood Cliffs, New Jersey.
12. Advanced Engineering Mathematics; C. Ray Wylie; McGraw-Hill, New York, San Francisco, London.

## **PART IV**

# **LINEAR ALGEBRA**

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. 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.

Part IV 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.

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 in the appendix for part IV. The code is readily available from the editor for downloading to an HP-48G/GX from a microcomputer.



The material is an outgrowth of 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. The 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. The teaching style concentrates 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 many are included in the course, but the instructors exercise some care to prevent “theorems” and “proofs” from intimidating these beginning students.

Although the course in linear algebra is at the sophomore level, many of the students are further along in their studies. They can be nudged beyond normal expectations in linear algebra into treating certain topics in more depth: e.g., orthogonality, least squares, and real symmetric matrices. The material on iterative methods and singular value decompositions for honors students is provided for outside class projects.

The material in Part IV supplements whatever textbook is being used. 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, students should be required to use their HP-48G/GX units on a regular, almost daily basis.

# 14

## 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.

### 14.1 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  $2 \times 3$  real matrix

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

Similarly, the  $3 \times 2$  complex matrix

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

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  $\boxed{\leftarrow}$   $\boxed{[ ]}$  1, 2, 3  $\boxed{\text{ENTER}}$ . Although we show commas to separate the three numbers, you should instead insert spaces with the  $\boxed{\text{SPC}}$  key. To enter a matrix, start with [ [ by pressing the  $\boxed{[ ]}$  key twice (the left-shifted  $\boxed{\times}$  key), enter the first row and press  $\boxed{\rightarrow}$  to move the cursor beyond the innermost bracket, then continue entering the remaining entries in row order and press  $\boxed{\text{ENTER}}$ .

**EXAMPLE:** Keystrokes  $\boxed{[ ]}$   $\boxed{[ ]}$  1, 2, 3  $\boxed{\rightarrow}$  4, 5, 6, 7, 8, 9  $\boxed{\text{ENTER}}$  will produce the following matrix:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

The  $\boxed{\rightarrow}$  key simply defines the number of columns. Now press  $\boxed{\text{DROP}}$  to drop this matrix from the stack. (When no command line is present you need not press the  $\boxed{\leftarrow}$  to DROP.)

To put the complex number (1, 2) on the stack, use the left-shifted  $\boxed{\div}$  key to get the double parentheses ( ), and press 1  $\boxed{\text{SPC}}$  2  $\boxed{\text{ENTER}}$ . When building a matrix with complex numbers, use the right cursor key  $\boxed{\rightarrow}$  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:

$$\begin{bmatrix} (1,2) & (3,4) \\ (5,0) & (0,6) \end{bmatrix}$$

## Using the MatrixWriter

Enter the MatrixWriter application by pressing  $\boxed{\rightarrow}$   $\boxed{\text{MATRIX}}$ . This activates a spreadsheet-type display, with a dark cursor resting in the 1-1 position. Check to see that the  $\boxed{\text{G}\odot\leftrightarrow\Box}$  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  $\boxed{\text{G}\odot\leftrightarrow}$  label to activate it.) Key in the numbers of the first row of the matrix in row order separated by spaces and then press  $\boxed{\text{ENTER}}$ . When you are ready to go to the second row press  $\boxed{\nabla}$ . 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  $\boxed{\text{ENTER}}$ . A final  $\boxed{\text{ENTER}}$  will put the matrix onto the stack.

**EXAMPLE.** The keystrokes  $\boxed{\rightarrow}$   $\boxed{\text{MATRIX}}$  1, 2, 3  $\boxed{\text{ENTER}}$   $\boxed{\nabla}$  4, 5, 6, 7, 8, 9  $\boxed{\text{ENTER}}$   $\boxed{\text{ENTER}}$  will produce this matrix on the stack:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

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  $\boxed{\rightarrow}$   $\boxed{\text{MATRIX}}$ . Note that the menu key  $\boxed{\text{VEC}\square}$  appears. If you press  $1\ 2\ 3\ 4$  (separate the digits with spaces)  $\boxed{\text{ENTER}}\ \boxed{\text{ENTER}}$ , the *vector*  $[1\ 2\ 3\ 4]$  will show on the stack. The presence of the white box in  $\boxed{\text{VEC}\square}$  indicates that vector entry is active. If you toggle off this key to see  $\boxed{\text{VEC}}$  without the box, the keystrokes  $1, 2, 3, 4\ \boxed{\text{ENTER}}\ \boxed{\text{ENTER}}$  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  $\boxed{\rightarrow}$   $\boxed{\text{MATRIX}}$ , the vector entry mode  $\boxed{\text{VEC}\square}$  is active by default. But if there is an array on level 1 and you enter the MatrixWriter by pressing  $\boxed{\nabla}$  to review that array, the status of  $\boxed{\text{VEC}}$  reflects the nature of the array:  $\boxed{\text{VEC}\square}$  for a vector and  $\boxed{\text{VEC}}$  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  $\boxed{\nabla}$  to enter the MatrixWriter, then changing the status of  $\boxed{\text{VEC}}$  and pressing  $\boxed{\text{ENTER}}$ .

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

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

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  $Z_{10} = \{ 0, \pm 1, \pm 2, \dots, \pm 9 \}$ . 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  $\pm$  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

$$\begin{bmatrix} 4 & -2 & 5 & -8 & 5 \\ -4 & 7 & 8 & 0 & -6 \\ -4 & 0 & 8 & -5 & -2 \\ 5 & 6 & 0 & 2 & 3 \end{bmatrix}$$

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

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

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 }      CON returns [[ 5 5 5 ]
1: 5                  [ 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 section Getting Started with the HP-48G/GX).

**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  $\pm$  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

$$\begin{bmatrix} -5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & -7 \end{bmatrix}$$

**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  $\pm$  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

```
[ [-8  8  9 -4]
  [ 0 -9  0  1]
  [ 0  0  4 -4]
  [ 0  0  0  2]
```

**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  $\pm$  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  $\pm$  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  $\pm$  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 } → ARRAY 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 }
→ ARRAY » »
```

**EXAMPLE.** Press 4 HILB to see the approximation to the  $4 \times 4$  Hilbert matrix

$$\begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix}$$

### Activity Set 14.1

- 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)  $\begin{bmatrix} 3 & -1 & 2 \\ -4 & 0 & 1 \\ 2 & -3 & 5 \end{bmatrix}$

(d)  $\begin{bmatrix} -9 & 8 & -7 \\ 6 & -5 & 4 \\ -3 & 2 & -1 \\ 0 & -1 & 2 \\ -3 & 4 & -5 \\ 6 & -7 & 8 \end{bmatrix}$

(c)  $\begin{bmatrix} 4 & 2 & -1 & 7 \\ -5 & 0 & 3 & 8 \\ -6 & 9 & -2 & 5 \\ 0 & -4 & 1 & -3 \end{bmatrix}$

(e)  $\begin{bmatrix} -3 & -2 \\ -1 & 0 \\ 1 & 2 \\ 3 & 3 \\ 5 & 6 \\ 7 & 8 \\ 9 & 0 \end{bmatrix}$

$$\begin{array}{l} \begin{bmatrix} -9 & 6 & -3 & 0 & -3 & -6 \\ 8 & -5 & 2 & -1 & 4 & 7 \\ -7 & 4 & -1 & 2 & 5 & 8 \end{bmatrix} \\ \text{(f)} \end{array}$$

$$\begin{array}{l} \begin{bmatrix} (6, -5) & (-2, -1) & (4, -8) \\ (-3, 0) & (7, 9) & (0, -11) \end{bmatrix} \\ \text{(g)} \end{array}$$

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.

$$\begin{array}{l} \begin{bmatrix} -2 & 4 & 0 & 1 \\ 3 & -5 & 2 & 7 \\ 1 & 3 & 10 & -6 \\ -4 & 5 & 1 & -1 \end{bmatrix} \\ \text{(a)} \end{array}$$

$$\begin{array}{l} \begin{bmatrix} -3 & -8 \\ 1 & -1 \\ -4 & 6 \\ -2 & -7 \\ 5 & -4 \\ -8 & 5 \\ 6 & 0 \end{bmatrix} \\ \text{(b)} \end{array}$$

$$\begin{array}{l} \begin{bmatrix} (-1, 0) & (1, -2) & (-3, 4) \\ (5, -6) & (-7, 8) & (0, -9) \end{bmatrix} \\ \text{(c)} \end{array}$$

$$\begin{array}{l} \begin{bmatrix} -4 & 3 & -2 & 1 \end{bmatrix} \\ \text{(d)} \end{array}$$

$$\begin{array}{l} \begin{bmatrix} -4 \\ 3 \\ -2 \\ 1 \end{bmatrix} \\ \text{(e)} \end{array}$$

$$\begin{array}{l} \begin{bmatrix} e^{4.1} & \sqrt{11} \\ \pi/3 & \sin 4 \end{bmatrix} \\ \text{(f)} \end{array}$$

$$\begin{array}{l} \begin{bmatrix} \ln(-2) & \sqrt{-2} & \cos^{-1}(-2) \end{bmatrix} \\ \text{(g)} \end{array}$$

## 14.2 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  $\text{OBJ} \rightarrow$ , 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  $\text{OBJ} \rightarrow$  is the inverse command to  $\rightarrow \text{ARRAY}$ .

**EXAMPLE.** With  $[ 0 \ 2 \ 4 \ 6 ]$  on level 1 the command  $\text{OBJ} \rightarrow$  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

$$\begin{bmatrix} 1 & -7 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

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  $\rightarrow$ ROW, located on the MTH MATR ROW menu, and  $\rightarrow$ COL, located on the MTH MATR COL menu. For example, with

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

on stack level 1, press  **$\rightarrow$ ROW** to separate the matrix into its four row vectors. Notice that stack level 1 contains the number of rows. Press **△** 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  $\rightarrow$ ROW and  $\rightarrow$ COL are ROW $\rightarrow$  and COL $\rightarrow$ , located next to the  $\rightarrow$ ROW and  $\rightarrow$ COL commands on the appropriate menus. With four vectors on levels 1 through 4, simply press 4 **COL $\rightarrow$**  to build the matrix having the four vectors as columns:

4: [ 0 2 4 -6 ]			
3: [ 5 -1 8 3 ]			[[ 0 5 -7 6 ]
2: [-7 9 -4 2 ]	4	<div style="border: 1px solid black; padding: 2px 5px; display: inline-block;">COL→</div>	returns [ 2 -1 9 -3 ]
1: [ 6 -3 5 8 ]			[ 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  $\rightarrow$ DIAG command (on the MTH MATR menu) will extract the main diagonal (as a vector) from any matrix on level 1. For example,

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

- The  $\rightarrow$ 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:

$$\begin{array}{ll} 2: [2 \ 2 \ 2] & \text{DIAG} \rightarrow \text{returns} \\ 1: 3 & \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \end{array}$$

2: [ 2 2 2 ]      DIAG→ returns      [[ 2 0 0 0 ]  
 1:      4                              [ 0 2 0 0 ] , and  
    [ 0 0 2 0 ]  
    [ 0 0 0 0 ]]

2: [ 2 2 2 ]      DIAG→ returns      [[ 2 0 ]  
 1: { 3 2 }                              [ 0 2 ] .  
    [ 0 0 ]]

## Activity Set 14.2

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  $4 \times 5$  matrix  $B$  that preserves row order.
  - (b) Disassemble  $B$  into its entries, drop the last five entries, and reassemble the remaining entries into a  $5 \times 3$  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  $5 \times 4$  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  $4 \times 3$  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, then a row of 4's.
- (c) Are the results in (a) and (b) the same?

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

- (a) Use the ROW+ and COL+ commands to build the block matrices

$$[A \ B] \text{ and } \begin{bmatrix} A \\ B \end{bmatrix}.$$

- (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  $3 \times 4$  matrix  $A$ . Now use RANM to generate a  $3 \times 2$

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

6. Seed your calculator's random number generator with 6 and use program SYMM to generate a  $3 \times 3$  symmetric matrix  $A$ . Then use program TRIDIA to generate a  $3 \times 3$  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  $3 \times 4$  matrix  $E$  that preserves row order.

## 14.3 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  $\boxed{+}$ . Press  $\boxed{-}$  instead of  $\boxed{+}$  to calculate  $A - B$ . Note that the commands  $\boxed{+}$  and  $\boxed{-}$  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  $\boxed{A} \boxed{B} \boxed{+}$  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  $\boxed{\times}$ . Multiplying by -1 can be done in a single keystroke with the  $\boxed{+/-}$  key.

## Dot Products and Length

The dot product of two real or complex vectors  $[x_1 \ x_2 \ \dots \ x_n]$  and  $[y_1 \ y_2 \ \dots \ y_n]$  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  $[x_1 \ x_2 \ \dots \ x_n]$  and  $[y_1 \ y_2 \ \dots \ y_n]$ , (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  $[x_1 \ x_2 \ \dots \ x_n]$  the command ABS (see the MTH VECTR menu) will calculate the Euclidean length (norm)

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

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  $\boxed{\times}$  instead of  $\boxed{+}$ . 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:

$$\begin{bmatrix} 7 & -9 & -8 \\ -5 & 8 & -1 \end{bmatrix}$$

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

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

Press  $\boxed{\times}$  to see

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

## Matrix by Vector Multiplication

As a matter of convenience, the HP-48G series calculators will let you premultiply any  $n$ -vector  $x = [x_1 \ x_2 \ \dots \ x_n]$  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  $k = 1, 2, 3, \dots$ , you can use the following program.

**$A^{\uparrow k}$**       ( $k^{\text{th}}$  power of a matrix)

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

*Effect:*    returns the  $k^{\text{th}}$  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

$$A = \begin{bmatrix} 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 \end{bmatrix},$$

then  $A^8$  can be found correctly on the calculator to be the constant  $4 \times 4$  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  $p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ , we sometimes want to find  $p(A) = A^n + a_{n-1} A^{n-1} + \dots + a_1 A + a_0 I$ . The following program, P.of.A, does just that.

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

*Inputs:* level 2: a vector  $[a_n \ a_{n-1} \ \dots \ a_1 \ a_0]$  of coefficients

level 1: a square matrix  $A$

*Effect:* returns  $p(A) = a_n A^n + a_{n-1} A^{n-1} + \dots + a_1 A + a_0 I$

```
« → 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

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

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

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

## 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:

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

To obtain the conjugate transpose, execute TRN:

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

$$\begin{bmatrix} (7, -4) & (2, 0) \end{bmatrix}$$

$$\begin{bmatrix} (3, 0) & (5, 1) \end{bmatrix}$$

### Activity Set 14.3

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

2. (a) Create a  $5 \times 4$  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  $3 \times 4$  matrix  $A$  over  $Z_{10}$  and calculate  $A^T A$ ; carefully observe your result.  
 (b) Repeat part (a) using random  $4 \times 5$  and  $5 \times 6$  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  $4 \times 4$  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  $A, A^2, A^3, \dots$  to find  $\lim_{n \rightarrow \infty} \{A^n\}$ .

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

(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  $\frac{u - v}{\|u - v\|}$
- (e) Verify the triangle inequality:  $\|u + v\| \leq \|u\| + \|v\|$ .
- (f) Verify the Cauchy-Schwarz inequality:

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

## 14.4 DETERMINANTS AND INVERSES

With a square matrix  $A$  on stack level 1, the command DET will return the determinant of  $A$ , and pressing  $\boxed{1/x}$  to execute the INV command will return  $A^{-1}$  in the event that  $\det A \neq 0$ . 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  $\det A = 256$ .

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

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  $\det C = 6.1E-17$ . 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,  $\det C = 0$ , 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-off 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,  $1 \leq n \leq 11$ . For example, to round the matrix

$$\begin{bmatrix} 1 & 0 & -.000000000001 \\ 0 & 1 & .000000000001 \\ 0 & 0 & .999999999999 \end{bmatrix}$$

to 11 decimal places, simply enter 11 RND to obtain

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

### Activity Set 14.4

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  $n \times n$  matrix  $A$ ,  $\det[kA] = k^n \det A$  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 14.4.

## 14.5 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

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

we obtain the matrix

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

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  $\alpha$  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

$$\ll 1 + \text{SQ} \sqrt{\phantom{x}} \gg.$$



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 the Getting Started section 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   to recall the program to the stack, press    to put the name 'APLY' on level 1, go to { HOME } and press . Touch  to see the copy in your { HOME } directory. (At this point you may wish to purge the **EXAMPLES** directory.)

## Activity Set 14.5

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

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

by writing a short RPN program and using program **APLY**. Note that the command  $\rightarrow \text{NUM}$  (the   key) will convert the symbolic  $\pi$  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

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

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

$$f(x) = \lceil \sqrt{\frac{x+1}{x}} \rceil$$
 to the matrix

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

# 15

## 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,  $A = LU$ .

### 15.1 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:

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



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

$$\begin{bmatrix} 2 & 3 & 2 & -1 & 4 \\ 0 & 0 & 5 & 0 & 7 \\ 4 & 8 & 7 & 2 & -3 \\ 2 & 4 & 1 & -4 & 2 \end{bmatrix}$$

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

$$\begin{bmatrix} 2 & 3 & 2 & -1 & 4 \\ 0 & 0 & 5 & 0 & 7 \\ 0 & 2 & 3 & 4 & -11 \\ 0 & 1 & -1 & -3 & -2 \end{bmatrix}$$

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

$$\begin{bmatrix} 2 & 3 & 2 & -1 & 4 \\ 0 & 2 & 3 & 4 & -11 \\ 0 & 0 & 5 & 0 & 7 \\ 0 & 0 & -2.5 & -5 & 3.5 \end{bmatrix}$$

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




$$\begin{bmatrix} 2 & 3 & 2 & -1 & 4 \\ 0 & 2 & 3 & 4 & -11 \\ 0 & 0 & 5 & 0 & 7 \\ 0 & 0 & 0 & -5 & 7 \end{bmatrix}$$

Back substitution by hand shows the solution vector to be  $[7.1 \quad -4.8 \quad 1.4 \quad -1.4]$ . 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  , then backsolves for  $x_{n-1}$  and halts, etc. After  $x_n, x_{n-1}, \dots, x_1$  are on the stack, a final  returns  $x = [x_1, x_2, \dots, x_n]$ .

```
« → 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  to split off the rightmost column, then press  to see the last component of the solution vector, -1.4. Each press of   will return the next component. When all four components are on the stack, a final   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:

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

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

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

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

$$\begin{bmatrix} -4 & -6 & 1 & 2 & -1 \\ 0 & 0 & 2.5 & 0 & 3.5 \\ 0 & 2 & 8 & 4 & -4 \\ 0 & 1 & 1.5 & -3 & 1.5 \end{bmatrix}$$

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

$$\begin{bmatrix} -4 & -6 & 1 & 2 & -1 \\ 0 & 2 & 8 & 4 & -4 \\ 0 & 0 & 2.5 & 0 & 3.5 \\ 0 & 0 & -2.5 & -5 & 3.5 \end{bmatrix}$$

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

$$\begin{bmatrix} -4 & -6 & 1 & 2 & -1 \\ 0 & 2 & 8 & 4 & -4 \\ 0 & 0 & 2.5 & 0 & 3.5 \\ 0 & 0 & 0 & -5 & 7 \end{bmatrix}$$

Notice how this matrix differs from the one we obtained 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

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

begin with the augmented matrix  $[A|b]$

$$\begin{bmatrix} 5 & -9 & 16 & 6 & 48 \\ -5 & 9 & -16 & -8 & -45 \\ 10 & -9 & 24 & 8 & 72 \\ -5 & -9 & 8 & 8 & 3 \end{bmatrix}$$

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  $[U|b']$

$$\begin{bmatrix}
 10 & -9 & 24 & 8 & 72 \\
 0 & -13.5 & 20 & 12 & 39 \\
 0 & 0 & -2.6 & -2 & -1 \\
 0 & 0 & 0 & -2 & 3
 \end{bmatrix}$$

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

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  $[A|b]$  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  $[0 \quad 0 \quad \dots \quad 0 \quad *]$  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

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

begin with the augmented matrix  $[A|b]$

$$\begin{bmatrix} 4 & 1 & 3 & -2 & 1 & 5 \\ 8 & 2 & 1 & -5 & 0 & 13 \\ 4 & 1 & 8 & -1 & 2 & 7 \\ 8 & 2 & -9 & -7 & -2 & 9 \end{bmatrix}$$

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:

$$\begin{bmatrix} 8 & 2 & 1 & -5 & 0 & 13 \\ 0 & 0 & -10 & -2 & -2 & -4 \\ 0 & 0 & 0 & 0 & .5 & -2.5 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

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 15.1**

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.

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

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

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.

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

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



3. Repeat Activity 2 for the following linear systems.

$$(a) \quad x_1 + 2x_2 + 3x_3 + 4x_4 = 5$$

$$x_1 + 3x_2 + 4x_3 + 5x_4 = 5$$

$$3x_1 + 6x_2 + 9x_3 + 2x_4 = -5$$

$$2x_1 + 4x_2 + 6x_3 + x_4 = -4$$

$$(b) \quad -2x_1 + 3x_2 + 5x_3 - 2x_4 + 3x_5 - 9x_6 = 6$$

$$2x_1 - 3x_2 + x_3 + 4x_4 - 7x_5 + x_6 = 2$$

$$6x_1 - 9x_2 + 11x_4 - 19x_5 + 3x_6 = 0$$

$$4x_1 - 6x_2 + 5x_3 + 9x_4 - 16x_5 - 2x_6 = 8$$

## 15.2 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  $\rightarrow LP$  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  $\rightarrow LP$ (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  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  $\rightarrow \text{LP}$  to create appropriate starting matrices  $L$  and  $P$ . Interchange rows 1 and 2 in  $A$  with 1, 2  $\text{RSWP}$ , recall  $P$  to the stack and make the same row interchange, then store the updated result in  $P$  with  $\leftarrow \text{P}$ . Now press 1, 1  $\text{L.U}$  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  $\text{RSWP}$  to effect the interchange, then bring  $P$  to level 1, make the same row interchange with  $\text{RSWP}$  and store the result in  $P$ . Now bring  $L$  to level 1 with  $\text{L}$ , interchange multipliers with 2, 4  $\text{L.SWP}$  and store the result in  $L$  with  $\leftarrow \text{L}$ .

Step 3: Now execute 2, 2  $\text{L.U}$  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  $\boxed{L}$ , then do  $\boxed{U}$   $\boxed{\times}$  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  $P^{-1}LU = P^T LU = A$ . Recall  $P$  to level 1 and get  $P^T$ , SWAP levels with  $LU$  and then use  $\boxed{\times}$  to see  $P^T LU = A$ .

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

- (i) In the case that  $A = LU$ , all the information regarding Gaussian elimination on  $A$  is stored in the factors  $L$  and  $U$ . Matrix  $L$  maintains a record of the multipliers used in the elimination 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  $n \times n$  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  $U$ :  $n^3$  versus  $\frac{n^3}{3}$ . 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  , then solves for  $x_2$  and halts, etc.

After  $x_1, x_2, \dots, x_n$  are on the stack, a final  returns  $x = [x_1, x_2, \dots, x_n]$ .

```
« → 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

$$2x_1 + 3x_2 - x_3 + 2x_4 = 1$$

$$-4x_1 - 6x_2 + 2x_3 + x_4 = 2$$

$$2x_1 + 4x_2 - 4x_3 + x_4 = 3$$

$$4x_1 + 8x_2 + 2x_3 + 7x_4 = 4$$

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  $u_{11}, u_{22}, \dots, u_{nn}$  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  $\boxed{+}$  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 15.2

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  $Ly = Pb$ .

$$(a) \quad A = \begin{bmatrix} -2 & 4 & 5 \\ -1 & 3.5 & 2.5 \\ 2 & -1 & 3 \end{bmatrix}$$

$$b = [10 \quad 3.5 \quad 11]$$

$$(c) \quad A = \begin{bmatrix} 0 & -2 & 7 & -3 \\ 3 & 0 & 3 & 5 \\ -3 & 6 & -3 & 4 \\ -9 & 6 & -3 & 6 \end{bmatrix}$$

$$b = [44 \quad -24 \quad -24 \quad -24]$$

$$(b) \quad A = \begin{bmatrix} -2 & 6 & 5 \\ -3 & 1 & -5 \\ -6 & -6 & -3 \end{bmatrix}$$

$$b = [26 \quad 15 \quad 18]$$

$$(d) \quad A = \begin{bmatrix} -2 & 0 & -3 & 0 & -5 \\ -6 & -3 & -12 & -9 & 3 \\ 6 & 6 & 6 & 12 & 6 \\ 0 & 3 & 3 & -3 & 6 \\ -2 & -1 & -4 & -1 & 1 \end{bmatrix}$$

$$b = [-7 \quad 36 \quad -6 \quad 27 \quad 10]$$

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.

$$(a) \quad A = \begin{bmatrix} -1 & -1 & 2 \\ -4 & 2 & -1 \\ -9 & 6 & 0 \end{bmatrix}, \quad b = [-7 \quad 11 \quad 21]$$

$$(b) \quad A = \begin{bmatrix} 1 & -5 & 2 & -1 \\ 2 & -7 & 4 & 1 \\ -7 & 9 & 1 & 3 \\ 5 & -8 & -9 & 8 \end{bmatrix}, \quad b = [-8 \quad -13 \quad 53 \quad -76]$$

$$(c) \quad A = \begin{bmatrix} -7 & -7 & -7 & 2 & -7 \\ -1 & 0 & -5 & -8 & 7 \\ 7 & -1 & -9 & 6 & 7 \\ 5 & 4 & 9 & 0 & -9 \\ 9 & -6 & -5 & -9 & 8 \end{bmatrix}, \quad b = [-16 \quad 45 \quad -80 \quad 7 \quad 17]$$

### 15.3 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

$$2x_1 - 3x_2 + x_3 - 3x_4 + 2x_5 = 6$$

$$-2x_1 + 3x_2 - x_3 + 4x_4 + x_5 = -5$$

$$6x_1 - 9x_2 + 7x_3 - 7x_4 + 5x_5 = 20$$

$$-2x_1 + 3x_2 + 3x_3 + 3x_4 - 9x_5 = -6$$

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:

$$\begin{bmatrix} 1 & -1.5 & 0 & 0 & 6.375 & 4.5 \\ 0 & 0 & 1 & 0 & -1.75 & 0 \\ 0 & 0 & 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

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 15.3

1. Perform Gauss-Jordan reduction with **PIVOT** to solve each of the linear systems in **ACTIVITY SET 15.1**. Verify your results by applying the **RREF** command.

# 16

## 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.

## 16.1 LINEAR COMBINATIONS AND SPANNING SETS

Recall that by a *linear combination* of vectors  $v_1, v_2, \dots, v_k$  in a vector space  $V$  (you may regard  $V$  as being  $R^n$  if it helps) we mean any vector of the form  $x_1v_1 + x_2v_2 + \dots + x_kv_k$  where the  $x_j$ 's are scalars. The set of all possible linear combinations of  $v_1, v_2, \dots, v_k$  is a subspace of  $V$ , often denoted by  $\text{Span}[v_1, v_2, \dots, v_k]$ , and the vectors  $v_i$  are said to *span* this subspace. To verify that  $\text{Span}[v_1, v_2, \dots, v_k]$  is a subspace of  $V$  we need only add two linear combinations of the vectors  $v_1, v_2, \dots, v_k$  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  $\text{Span}[v_1, v_2, \dots, v_k]$  we must determine whether  $u$  can be written as  $u = x_1v_1 + x_2v_2 + \dots + x_kv_k$  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  $A$ :

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

where  $A_j$  is column  $j$  of matrix  $A$  and  $x$  is the column vector  $= [x_1, x_2, \dots, x_n]$ . The column vectors  $A_1, A_2, \dots, A_n$  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

$$\begin{array}{ccc} \begin{bmatrix} 1 & -1 & 2 & 3 \\ -2 & 4 & 0 & 10 \\ 3 & 2 & -1 & -2 \\ 0 & 3 & 4 & 18 \end{bmatrix} & \rightarrow & \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

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

$$\begin{array}{ccc} \begin{bmatrix} 4 & 0 & 0 & -4 & 6 \\ -1 & 3 & 3 & 7 & -4.5 \\ 0 & -2 & -6 & -4 & 2 \\ 2 & 1 & 3 & 0 & 2 \end{bmatrix} & \rightarrow & \begin{bmatrix} 1 & 0 & 0 & -1 & 1.5 \\ 0 & 1 & 0 & 2 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

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



### Activity Set 16.1

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.

## 16.2 DEPENDENCE AND INDEPENDENCE

When a vector  $u$  is a linear combination of some other vectors  $v_1, v_2, \dots, v_k$  we say that  $u$  *depends* linearly upon the  $v_i$ 's and that the entire set of vectors is a "linearly dependent" set. More precisely, a set of vectors  $\{v_1, v_2, \dots, v_k\}$  ( $k > 1$ ) is called (*linearly*) *dependent* if one of these vectors is a linear combination of the others. To the contrary, the set of vectors  $\{v_1, v_2, \dots, v_k\}$  ( $k > 1$ ) 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  $\{v_1, v_2, \dots, v_k\}$  is dependent iff there are scalars  $x_1, x_2, \dots, x_k$ , *not all of which are 0*, such that  $x_1v_1 + x_2v_2 + \dots + x_kv_k = 0_v$ ; thus
- (b) the set  $\{v_1, v_2, \dots, v_k\}$  is independent iff whenever we have  $x_1v_1 + x_2v_2 + \dots + x_kv_k = 0_v$  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  $v_1, v_2, \dots, v_k$  as its columns then

- (a) the set  $\{v_1, v_2, \dots, v_k\}$  is dependent iff  $Ax = 0$  has a non-zero solution; and
- (b) the set  $\{v_1, v_2, \dots, v_k\}$  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  $\alpha$  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  $v_1 = x_2 v_2 + \dots + x_k v_k$ . Consequently, any linear combination of the vectors in  $\{v_1, v_2, \dots, v_k\}$  can be replaced by a linear combination of the vectors in the smaller

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

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

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 16.2

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  $\mathbb{R}^4$ . 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  $W$ .

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

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

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

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

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

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

### 16.3 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  $\mathbb{R}^n$  if it helps) we mean a collection of vectors  $w_1, w_2, \dots, w_k$  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  $w = x_1w_1 + x_2w_2 + \dots + x_kw_k$ . Suppose, however, that there is another way to do this, say  $w = y_1w_1 + y_2w_2 + \dots + y_kw_k$ . Then, easily

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

This last equation is a linear combination of the  $w_j$ 's equal to the zero vector. Therefore, because of the independence of  $\{w_1, w_2, \dots, w_k\}$ , all coefficients  $(x_j - y_j)$  in this combination must be zero:  $(x_j - y_j) = 0$ , so  $x_j = y_j$  for  $j = 1, \dots, k$  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  $B = \{v_1, v_2, \dots, v_n\}$ , 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  $A^Tx = 0$ .

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  $NS(A) = \{0\}$ , 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:

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



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

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

We therefore have the fundamental result that

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

Since this result 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 16.3

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

$$(a) \quad A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 6 & 3 & 0 & 1 \\ 10 & -1 & 4 & 1 \\ 7 & 0 & 2 & 1 \end{bmatrix}$$

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

$$\begin{array}{ll}
 \text{(c) } C = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 6 & 9 & 2 & -5 \\ 1 & 3 & 4 & 5 & 5 \\ 2 & 4 & 6 & 1 & -4 \end{bmatrix} & \text{(d) } D = \begin{bmatrix} 4 & 1 & 8 & -6 & -3 \\ 5 & -4 & 6 & -14 & -5 \\ 8 & 4 & 6 & 2 & 6 \\ 4 & -3 & 2 & -8 & -1 \\ -8 & 3 & -2 & 8 & -3 \end{bmatrix}
 \end{array}$$

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

$$\begin{array}{ll}
 \text{(a) } \begin{bmatrix} 1 & 3 & 1 & 2 \\ 2 & 6 & 3 & 4 \\ 3 & 9 & 4 & 6 \\ 4 & 2 & 5 & 1 \\ 5 & -5 & 5 & -4 \end{bmatrix} & \text{(b) } \begin{bmatrix} -1 & 1 & 2 & -1 \\ 2 & -1 & 1 & 3 \\ 1 & 0 & 3 & 2 \\ -3 & 2 & 2 & -3 \\ 1 & 1 & 6 & 1 \end{bmatrix}
 \end{array}$$

## 16.4 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  $B = \{ u_1, u_2, \dots, u_n \}$  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  $B$ :

$$w = x_1 u_1 + x_2 u_2 + \dots + x_n u_n.$$

The column vector  $[w]_B = [x_1 \ x_2 \ \dots \ x_n]$  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  $B' = \{v_1, v_2, \dots, v_n\}$ . The theorem describing how to do this is as follows:

Let  $B = \{u_1, u_2, \dots, u_n\}$  and  $B' = \{v_1, v_2, \dots, v_n\}$  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  $[u_1]_{B'}, [u_2]_{B'}, \dots, [u_n]_{B'}$ . If  $P$  is the  $n \times n$  matrix whose  $j^{\text{th}}$  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:

$$\begin{array}{cccc|cccc} [2 & 1 & -1 & 2 & : & 10 & -3 & -3 & 10] & & [1 & 0 & 0 & 0 & : & 2 & 1 & -1 & 2] \\ [1 & 3 & 2 & -3 & : & -3 & 23 & 10 & -21] & & [0 & 1 & 0 & 0 & : & 1 & 3 & 2 & -3] \\ [-1 & 2 & 1 & -1 & : & -3 & 10 & 7 & -13] & \rightarrow & [0 & 0 & 1 & 0 & : & -1 & 2 & 1 & -1] \\ [2 & -3 & -1 & 4 & : & 10 & -21 & -13 & 30] & & [0 & 0 & 0 & 1 & : & 2 & -3 & -1 & 4] \end{array}$$

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  $[w]_B$ :

$$\begin{bmatrix} 10 & -3 & -3 & 10 & 3 \\ -3 & 23 & 10 & -21 & -2 \\ -3 & 10 & 7 & -13 & -11 \\ 10 & -21 & -13 & 30 & 17 \end{bmatrix} \xrightarrow{\text{row operations}} \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & -3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

so that  $[w]_B = [-1 \ 2 \ -3 \ 1]$  and therefore  $P[w]_B = [5 \ -4 \ 1 \ -1]$ .

### Activity Set 16.4

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

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

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

- Show that both  $B$  and  $B'$  are independent sets of vectors and that  $\text{Span } B = \text{Span } B'$ .
- Let  $W = \text{Span } B = \text{Span } B'$ . Show that  $w = [1 \ 2 \ 2 \ -2]^T$  is in  $W$ .
- Find the change-of-basis matrix  $P$  from the  $B$ -basis to the  $B'$  basis for  $W$ .
- 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

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

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

and

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

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

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

# 17

## 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  $x = [x_1 \ x_2 \ \dots \ x_n]$  and  $y = [y_1 \ y_2 \ \dots \ y_n]$  is their dot product

$$x \bullet y = x_1 y_1 + x_2 y_2 + \dots + x_n y_n.$$

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  $x = [x_1 \ x_2 \ \dots \ x_n]$  the command ABS will return its Euclidean length (norm)  $\|x\|_2 = \sqrt{|x_1|^2 + |x_2|^2 + \dots + |x_n|^2}$  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.

## 17.1 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  $B = \{v_1, v_2, \dots, v_k\}$  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  $W = \text{Span}[v_1, v_2, \dots, v_k]$ . An attractive feature of such a basis is the ease with which we can obtain the coordinates of any vector  $w$  in  $W$ :

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

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

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

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

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

And clearly  $\{v_1, v_2, \dots, v_k\}$  is an orthonormal set iff  $A^T A$  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

$$A^T A = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 11 & 0 \\ 0 & 0 & 33 \end{bmatrix},$$

so the vectors are orthogonal.

More generally, since the  $(i, j)$ -entry in  $AB$  is the dot product (row of  $A$ )  $\bullet$  (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,

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

Thus, the nullspace  $NS(A)$  and the row space  $RS(A)$  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:

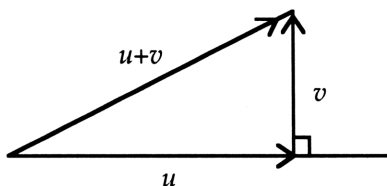
$$x \in NS(A^T) \iff x \text{ is orthogonal to } CS(A).$$

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

### Activity Set 17.1

- Let  $\langle x, y \rangle$  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*:  $|\langle x, y \rangle| \leq \|x\| \|y\|$ .
  - $x = [1 \ -2 \ 3 \ -5]$ ,  $y = [6 \ 9 \ -7 \ 3]$  in  $R^4$ ,
  - $x = [1 + i \ -2 + 3i \ i]$ ,  $y = [3 - 4i \ -i \ 5 + i]$  in  $C^3$ ,
  - 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  $CS(A)$ .

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

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

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

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

## 17.2 ORTHONORMAL BASES

We have already noted some of the advantages in having an orthonormal basis  $v_1, v_2, \dots, v_k$  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  $e_1, e_2, \dots, e_k$  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  $W$ :

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

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,  $u = x_1 v_1 + \dots + x_k v_k$  and  $w = y_1 v_1 + \dots + y_k v_k$ , then it is easy to see that the dot product of  $u$  and  $w$  is given by the simple equation

$$u \bullet w = x_1 y_1 + \dots + x_k y_k .$$

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:

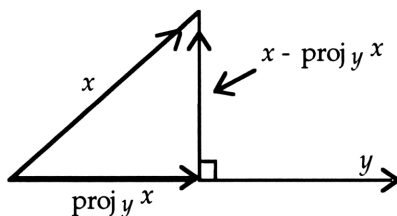


Figure 1.

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  $k$ :

$$0 = (x - ky) \bullet y \quad (\text{condition (ii)})$$

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

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

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 ROLL 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  $y \cdot (x - \text{proj}_y x)$ .

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

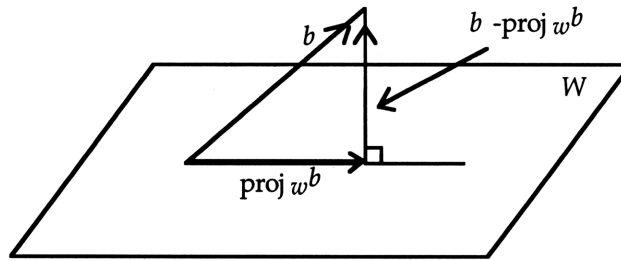


Figure 2.

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  $q_1, q_2, \dots, q_k$  from a given basis  $x_1, x_2, \dots, x_k$  for a subspace  $W$ . Here's how it works in  $R^n$ .

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

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

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



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,  $\text{Span} [x_1, \dots, x_j] = \text{Span} [q_1, \dots, q_j]$ , so when we're done,  $W = \text{Span} [x_1, \dots, x_k] = \text{Span} [q_1, \dots, q_k]$  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:

$$Q1: \begin{bmatrix} .894427191 & .4472135955 & 0 \end{bmatrix}$$

$$Q2: \begin{bmatrix} -.298142697 & .596284794 & .7453559925 \end{bmatrix}$$

$$Q3: \begin{bmatrix} .333333333334 & -.666666666665 & .666666666665 \end{bmatrix}.$$

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

$$Q = \begin{bmatrix} | & | & | \\ Q_1 & Q_2 & Q_3 \\ | & | & | \end{bmatrix}, \text{ then } \boxed{\text{ENTER}} \boxed{\text{TRN}} \boxed{\text{SWAP}} \boxed{\times} \text{ to see } Q^T Q. \text{ Clean up round-}$$

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  $\{v_1, v_2, \dots, v_k\}$  for  $W$ :

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

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 17.2

1. (a) Use the RANM command to generate a random  $4 \times 3$  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  $4 \times 3$  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  $W = \text{Span}[x_1 \ x_2]$ .
  - (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  $5 \times 4$  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  $W = \text{Span}[x_1 \ x_2 \ x_4]$ .
  - (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.

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

In general, we shall call a square matrix  $Q$  **orthogonal** if  $Q^T Q = I$ . 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:  $x = Q^{-1}b = Q^T b$ .

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 Gaussian 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  $q_1, q_2, \dots, q_j$ , say  $x_j = r_{1j}q_1 + r_{2j}q_2 + \dots + r_{jj}q_j$ .

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

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

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

$$\text{col 1 of } A = Q(\text{col 1 of } R)$$

$$\text{col 2 of } A = Q(\text{col 2 of } R)$$

$$\text{col 3 of } A = Q(\text{col 3 of } R)$$

$$\vdots$$

$$\vdots$$

$$\text{col } j \text{ of } A = Q(\text{col } j \text{ of } R)$$

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

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

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  $Q^T Q = I$  and  $A = QR$  we can obtain  $R$  as follows:  $Q^T A = Q^T Q R = R$ .

Thus, when the Gram-Schmidt process is applied to the columns of an  $m \times n$  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  $R = Q^T A$  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  $X_1, X_2, \dots, X_N$  stored in user memory.
- (2) Construct  $Q_1, Q_2, \dots, Q_N$  from  $X_1, X_2, \dots, X_N$  by the Gram-Schmidt process.
- (3) Construct and store matrix  $Q$ :

$Q_1 \ Q_2 \ \dots \ Q_N \ N$  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  $\backslash x$ (**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

$$Q = \begin{bmatrix} .894427191 & -.298142397 & .3333333334 \\ .4472135955 & .596284794 & -.66666666665 \\ 0 & .7453559925 & .66666666665 \end{bmatrix}.$$

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



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

to level 1, the upper trapezoidal matrix

$$R = \begin{bmatrix} [10.3440804328 & .676715542332 & -.483368244523 & -5.80041893427] \\ [0 & -6.96721293451 & -3.06106659926 & 4.46014305107] \\ [0 & 0 & -2.7196004146 & -.67990010365] \end{bmatrix}$$

to level 2, and the orthogonal matrix

$$Q = \begin{bmatrix} [-.483368244523 & -4.69488742217\text{E-}2 & .874157276122] \\ [9.66736489046\text{E-}2 & -.995316133501 & -8.\text{E-}16] \\ [-.870062840141 & -.845079735991\text{E-}2 & -.485642931179] \end{bmatrix}$$

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

### Activity Set 17.3

1. Consider the following matrix

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

- (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  $X_1, \dots, X_N$

level 1: the number  $n$  of vectors

*Effect:* Applies the modified Gram-Schmidt Algorithm to vectors  $X_1, \dots, X_N$  and returns the orthonormal vectors  $Q_1, \dots, Q_N$

```
« → 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.

## 17.4 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,  $\|\cdot\|_2$  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

$$A^T Ax = A^T b.$$

The linear system  $A^T Ax = A^T b$  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.*,  $\text{rank } A = n$ , we know that  $A^T A$  is invertible, so the system of normal equations  $A^T Ax = A^T b$  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

$$\begin{bmatrix} 2 & -3 & 1 & -3 & 2 \\ -2 & 3 & -1 & 4 & 1 \\ 6 & -9 & 7 & -7 & 5 \\ -2 & 3 & 3 & 3 & -9 \end{bmatrix}$$

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

$$x = [.47724708537 \quad -.715870628056 \quad .809514855209 \quad -.38779751786 \quad .462579917262].$$

## Fitting Curves to Data

Least squares solutions arise in curve fitting problems. Suppose we have  $n$  data points  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$  where all the  $x_j$ 's are distinct. Consider the problem of finding a polynomial  $P(t) = c_0 + c_1 t + \dots + c_m t^m$  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  $i = 1, \dots, n$  or

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

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

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

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

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

$$(A^T A)c = A^T y.$$

It is tempting to obtain the least squares solution by calculating  $x = (A^T A)^{-1} A^T y$  or by applying Gaussian elimination via the `/` command. But the coefficient matrix  $A^T A$  is likely to be *ill-conditioned*, in the sense that solutions to  $(A^T A)x = A^T y$  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  $\{x_1, x_2, \dots, x_n\}$

*Effect:* Returns the matrix

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix}$$

```
« 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 → ARRAY » »
```

**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 17.4

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  $f(x) = (x + 2)^2 e^{-x}$  (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  $A^T A$  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  $A^T A$  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).



# 18

## 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.

### 18.1 THE CHARACTERISTIC POLYNOMIAL

Given a square matrix  $A$  of order  $n$ , any real or complex number  $\lambda$  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  $(\lambda, x)$  we consider the equation  $Ax = \lambda x$ , which is clearly equivalent to  $(A - \lambda I)x = 0$ . For a given eigenvalue  $\lambda$ ,  $x$  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  $\lambda$  are just the non-zero vectors in the nullspace of  $A - \lambda I$ . We often call this nullspace the *eigenspace* of  $A$  determined by  $\lambda$ .

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  $\lambda$ , 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  $\lambda$  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  $\det(\lambda I - A) = \lambda^n + c_{n-1}\lambda^{n-1} + \dots + c_1\lambda + c_0$ , 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  $[1 \ c_{n-1} \ \dots \ c_1 \ c_0]$  of coefficients of  $\det(\lambda I - A) = \lambda^n + c_{n-1}\lambda^{n-1} + \dots + c_1\lambda + c_0$

```
« 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 18.1

1. (a) Generate and store random  $3 \times 3$  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  $4 \times 4$  matrices.
  - (d) Formulate a conjecture on the basis of your observations. Prove your conjecture.

2. (a) Generate a random  $3 \times 4$  matrix  $A$  and a random  $4 \times 3$  matrix  $B$ , both over  $Z_{10}$ .  
 (b) Compare  $\text{trace}(AB)$  and  $\text{trace}(BA)$ ; what do you observe?  
 (c) Repeat (a) - (b) for random  $3 \times 5$  and  $5 \times 3$  matrices over  $Z_{10}$ .  
 (d) Formulate a conjecture on the basis of your observations. Prove your conjecture.
3. (a) Generate a random  $3 \times 3$  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  $4 \times 4$  and  $5 \times 5$  matrices over  $Z_{10}$ .  
 (d) Formulate a conjecture based upon your observations.  
 (e) Prove your conjecture.
4. Generate and store random  $3 \times 3$ ,  $4 \times 4$ , and  $5 \times 5$  matrices over  $Z_{10}$ .  
 (a) For each of these matrices  $A$ , calculate  $\det A$ ,  $\text{trace} A$ , 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  $p(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + \dots + c_1\lambda + c_0$  its *companion matrix* is

$$C = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -c_0 & -c_1 & -c_2 & \dots & -c_{n-1} \end{bmatrix}.$$

- (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  $p(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + \dots + c_1\lambda + c_0$ ?
7. (a) Generate two random  $3 \times 3$  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  $4 \times 4$  and  $5 \times 5$  matrices over  $Z_{10}$ .
- (c) Repeat (a) – (b) for random  $3 \times 3$ ,  $4 \times 4$  and  $5 \times 5$  *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  $3 \times 3$  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  $4 \times 4$  and  $5 \times 5$  matrices over  $Z_{10}$ .

- (d) Repeat (a) – (c) using random  $3 \times 3$ ,  $4 \times 4$  and  $5 \times 5$  *complex* matrices over  $Z_{10}$ .
- (e) Formulate a conjecture based upon your observations. Discuss your conjecture with your instructor.

## 18.2 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  $a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ . It requires as input the vector  $[a_n \ a_{n-1} \ \dots \ a_1 \ a_0]$  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:  $\lambda = -2, 1, 3$  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  $\boxed{\times}$   $\boxed{-}$ . To obtain a basis for the nullspace of  $A - 3I$ , apply the RREF command. The result

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

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  $\boxed{\times}$   $\boxed{-}$ . Apply RREF. Use  $\boxed{\leftarrow}$  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  $A$

$$\begin{bmatrix} -14 & -16 & -26 & -9 \\ 16 & 19 & 28 & 12 \\ -7 & -8 & -11 & -7 \\ 13 & 14 & 24 & 14 \end{bmatrix}.$$

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 EGV (on the MTH MATR menu) returns the vector [ 4.46410161514 -2.46410161514 3 3 ] of eigenvalues accurate to 12-digits.

## Activity Set 18.2

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  $3 \times 3$  and  $4 \times 4$  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  $3 \times 3$  matrix  $A$  over  $Z_{10}$ . Calculate  $\det A$  and  $\text{trace} A$ .
- (b) Compare  $\text{trace} A$  with the sum of the eigenvalues of  $A$ ; what do you observe?
  - (c) Compare  $\det A$  with the product of the eigenvalues of  $A$ ; what do you observe?
  - (d) Repeat the above for random  $4 \times 4$  and  $5 \times 5$  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  $\lambda$  of maximum absolute value, the associated eigenspace.

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

$$B = \begin{bmatrix} -2 & -8 & -1 & 6 & 5 \\ 1 & 7 & 1 & -2 & -2 \\ -11 & -16 & 0 & 10 & 5 \\ -7 & -8 & -1 & 10 & 4 \\ 7 & 8 & 1 & -6 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 8 & 6 & 8 & 5 & -3 \\ -9 & -8 & -10 & -9 & 1 \\ -3 & -2 & -3 & -2 & 1 \\ 11 & 11 & 12 & 12 & 1 \\ -8 & -9 & -8 & -8 & -1 \end{bmatrix}$$

$$D = \begin{bmatrix} 54 & 22 & -4 & -2 & -12 & -6 \\ -91 & -34 & 9 & 2 & 22 & 8 \\ 170 & 70 & -12 & -7 & -37 & -20 \\ 92 & 47 & 0 & -8 & -15 & -18 \\ -27 & -7 & 5 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

## 18.3 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  $\{x_1, x_2, \dots, x_n\}$ , is especially nice. For then we have

$$P^{-1}AP = D = \begin{bmatrix} \lambda_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix},$$

where  $\lambda_i$  is the eigenvalue associated with  $x_i$  and  $P$  is the matrix having  $x_1, x_2, \dots, x_n$  as its columns:

$$P = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \dots & x_n \\ | & | & & | \end{bmatrix}.$$

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  $\lambda$  of  $A$ , the dimension of the eigenspace determined by  $\lambda$ ,  $\dim NS(A - \lambda I)$ , does not exceed the multiplicity of  $\lambda$  as a root of the characteristic polynomial;
- (2) If  $\lambda_1, \lambda_2, \dots, \lambda_k$  are the *distinct* eigenvalues of  $A$  and  $B_1, B_2, \dots, B_k$  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:

$$\det(\lambda I - A) = (\lambda - \lambda_1)^{m_1}(\lambda - \lambda_2)^{m_2} \dots (\lambda - \lambda_k)^{m_k}$$

where  $\lambda_1, \dots, \lambda_k$  are the *distinct* eigenvalues and  $m_1, \dots, m_k$  are their respective multiplicities. Since  $\det(\lambda I - A)$  is a polynomial of degree  $n$ , we have  $n = m_1 + m_2 + \dots + m_k$ . According to (1), we have  $\dim NS(A - \lambda_j I) \leq m_j$ , for each  $j = 1, \dots, k$ . 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  $\lambda$  there are not enough  
independent eigenvectors associated with  $\lambda$ .*

**EXAMPLE 5.** Consider the following matrix:

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

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 eigenvalues 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

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

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 18.3**

1. (a) Find matrix  $B = P^{-1}AP$  given the following matrices  $P$  and  $A$ :

$$A = \begin{bmatrix} 7 & 2 & 4 & 6 \\ -6 & -1 & -4 & -4 \\ 4 & 4 & 5 & -2 \\ -16 & -12 & -14 & -3 \end{bmatrix} \quad P = \begin{bmatrix} 0 & -1 & -1 & -1 \\ 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

- (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  $P^{-1}AP = D$ .

$$\begin{array}{lll} \text{(a)} \begin{bmatrix} 0 & 1 & -2 & 0 \\ -2 & 3 & -4 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & -1 & 0 & -1 \end{bmatrix} & \text{(b)} \begin{bmatrix} 10 & -5 & 12 & 0 \\ 9 & -4 & 12 & 0 \\ -5 & 3 & -5 & 0 \\ -7 & 2 & -9 & -2 \end{bmatrix} & \text{(c)} \begin{bmatrix} 6 & -2 & -4 & -2 \\ -2 & 3 & 2 & 1 \\ 2 & 1 & 0 & -3 \\ -2 & -1 & 2 & 5 \end{bmatrix} \end{array}$$

$$\begin{array}{ll} \text{(d)} \begin{bmatrix} 8 & -3 & 8 & 0 & 5 \\ 9 & -4 & 12 & 0 & 9 \\ -5 & 3 & -5 & 0 & -5 \\ -7 & 2 & -9 & 0 & -5 \\ 2 & -2 & 4 & 0 & 5 \end{bmatrix} & \text{(e)} \begin{bmatrix} 1 & 0 & 3 & 0 & -3 & 3 \\ -1 & 2 & 2 & -1 & -2 & 3 \\ 0 & 0 & 0 & -2 & 2 & 2 \\ -2 & 2 & 0 & 4 & 0 & 0 \\ -2 & 2 & -3 & -1 & 5 & 2 \\ 0 & 0 & -1 & -1 & 1 & 4 \end{bmatrix} \end{array}$$

## 18.4 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  $\lambda$  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  $\lambda$  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

$$Q^T A Q = \text{Diag} [ 2 \ 4 \ 1 \ 1 ].$$

As in this example,  $Q^{-1} = Q^T$  changes the similarity equation  $Q^{-1} A Q = D$  to  $Q^T A Q = D$ . More important is when we solve for  $A$ :

$$A = Q D Q^T.$$

If matrix  $Q$  has the orthonormal column vectors  $q_1, q_2, \dots, q_n$  as its columns

$$Q = [ q_1 \ q_2 \ \dots \ q_n ]$$

then we have

$$\begin{aligned} A = Q D Q^T &= [ q_1 \ \dots \ q_n ] \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \begin{bmatrix} q_1^T \\ \vdots \\ q_n^T \end{bmatrix} \\ &= \lambda_1 q_1 q_1^T + \lambda_2 q_2 q_2^T + \dots + \lambda_n q_n q_n^T. \end{aligned}$$

This is a *spectral decomposition* of matrix  $A$ , so-called because the set  $\{ \lambda_1, \lambda_2, \dots, \lambda_n \}$  of eigenvalues is sometimes called the *spectrum* of  $A$ . Each  $\lambda_j q_j q_j^T$  in the decomposition is an  $n \times n$  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  $q_1, \dots, q_n$ .

**EXAMPLE 8.** To compute a spectral decomposition of the matrix  $A$  in **EXAMPLE 7** using the particular  $Q$  matrix above, we have:

$$2q_1 q_1^T + 4q_2 q_2^T + 1q_3 q_3^T + 1q_4 q_4^T$$

$$\begin{aligned}
&= 2 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 4 \begin{bmatrix} \bar{.3} & -\bar{.3} & 0 & \bar{.3} \\ -\bar{.3} & \bar{.3} & 0 & -\bar{.3} \\ 0 & 0 & 0 & 0 \\ \bar{.3} & -\bar{.3} & 0 & -\bar{.3} \end{bmatrix} + 1 \begin{bmatrix} .5 & .5 & 0 & 0 \\ .5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
&+ 1 \begin{bmatrix} \bar{.16} & -\bar{.16} & 0 & -\bar{.3} \\ -\bar{.16} & \bar{.16} & 0 & \bar{.3} \\ 0 & 0 & 0 & 0 \\ -\bar{.3} & \bar{.3} & 0 & \bar{.6} \end{bmatrix} = \begin{bmatrix} 2 & -1 & 0 & 1 \\ -1 & 2 & 0 & -1 \\ 0 & 0 & 2 & 0 \\ 1 & -1 & 0 & 2 \end{bmatrix} \quad (\text{after 10 RND}).
\end{aligned}$$

### Activity Set 18.4

1. Find an orthogonal matrix  $Q$  and a diagonal matrix  $D$  such that  $Q^{-1}AQ = D$ .

$$\begin{aligned}
& \begin{bmatrix} 4 & -1 & -1 & -1 \\ -1 & 4 & -1 & -1 \\ -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & 4 \end{bmatrix} \\
& \text{(a)}
\end{aligned}$$

$$\begin{aligned}
& \begin{bmatrix} 3 & -1 & 1 & 0 \\ -1 & 3 & 1 & 0 \\ 1 & 1 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \\
& \text{(b)}
\end{aligned}$$

$$\begin{aligned}
& \begin{bmatrix} 1 & -2 & -3 & 2 \\ -2 & 1 & -2 & 3 \\ -3 & -2 & 1 & 2 \\ 2 & 3 & 2 & 1 \end{bmatrix} \\
& \text{(c)}
\end{aligned}$$

$$\begin{aligned}
& \begin{bmatrix} 2 & 2 & -2 & 2 & 2 \\ 2 & 1 & -3 & -1 & -1 \\ -2 & -3 & 1 & 1 & 1 \\ 2 & -1 & 1 & 5 & -1 \\ 2 & -1 & 1 & -1 & 5 \end{bmatrix} \\
& \text{(d)}
\end{aligned}$$

2. Find a spectral decomposition for each of the matrices in Activity 1(a) - 1(b).

## 18.5 POSITIVE DEFINITE MATRICES

A real symmetric matrix  $A$  of order  $n$  is called *positive definite* provided that

$$x^T A x > 0 \text{ for each non-zero } x \text{ in } R^n.$$

In the expression  $x^T A x$ , we are viewing vector  $x$  in  $R^n$  as a column matrix; thus  $x^T A x$  is a  $1 \times 1$  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  $x^T Ax = 0$ , contrary to  $x^T Ax > 0$ ). More importantly, any such matrix  $A$  has only positive eigenvalues (if  $Ax = \lambda x$  for some non-zero  $x$  then  $x^T Ax = x^T \lambda x = \lambda \|x\|^2$ , so that  $\lambda = x^T Ax / \|x\|^2 > 0$ ). 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 ( $x^T Ax > 0$  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  $A = M^T M$  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  $k = 1, 2, \dots, n$ . For  $k = 1$  this simply says that pivot  $a_{11} > 0$ . Then

$$A_2 = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \rightarrow \begin{bmatrix} a_{11} & * \\ 0 & a'_{22} \end{bmatrix}$$

and pivot  $a'_{22} > 0$  because  $\det A_2 = a_{11}a'_{22} > 0$  and  $a_{11} > 0$ .

Then we have

$$A_3 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \rightarrow \begin{bmatrix} a_{11} & * & * \\ 0 & a'_{22} & * \\ 0 & * & * \end{bmatrix} \rightarrow \begin{bmatrix} a_{11} & * & * \\ 0 & a'_{22} & * \\ 0 & 0 & a'_{33} \end{bmatrix}$$

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  $u_{11}, u_{22}, \dots, u_{nn}$ . If  $D$  is the diagonal matrix  $D = \text{diag} [u_{11} \ u_{22} \ \dots \ u_{nn}]$  then  $D^{-1} = \text{diag} [u_{11}^{-1} \ u_{22}^{-1} \ \dots \ u_{nn}^{-1}]$  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  $A = L_1 D U_1$ , we also have  $A = A^T = (L_1 D U_1)^T = U_1^T D^T L_1^T = U_1^T D L_1^T$  and the uniqueness tells us that  $L_1^T = U_1$ . Thus, the  $LDU$ -factorization in this case has the form  $A = L_1 D L_1^T$ . 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  $A = L_1 D L_1^T = (L_1 \sqrt{D}) (L_1 \sqrt{D})^T = LL^T$ , 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 18.5

1. Which of the following symmetric matrices are positive definite?

$$(a) \ A = \begin{bmatrix} 3 & 3 & 4 \\ 3 & -3 & 7 \\ 4 & 7 & 2 \end{bmatrix}$$

$$(b) \ A = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & 1 \\ -1 & 1 & 2 \end{bmatrix}$$

$$(c) \ A = \begin{bmatrix} -5 & -7 & -2 & 5 \\ -7 & 9 & -4 & -5 \\ -2 & -4 & 5 & 8 \\ 5 & -5 & 8 & -8 \end{bmatrix}$$

$$(d) \ A = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 1 \\ -1 & -1 & 2 & -1 \\ -1 & 1 & -1 & 2 \end{bmatrix}$$

$$\begin{array}{ll}
 \begin{array}{l}
 \begin{bmatrix} 5 & 8 & 2 & -9 & -9 \\ 8 & -8 & -7 & -1 & 4 \\ 2 & -7 & 0 & -8 & -2 \\ -9 & -1 & -8 & 9 & 3 \\ -3 & 4 & -2 & 3 & 7 \end{bmatrix} \\
 \text{(e) } A =
 \end{array}
 &
 \begin{array}{l}
 \begin{bmatrix} 4 & -10 & -4 & 3 & 3 \\ -10 & 32 & 12 & -2 & -4 \\ -4 & 12 & 6 & -3 & -3 \\ 3 & -2 & -3 & 9 & 7 \\ 3 & -4 & -3 & 7 & 6 \end{bmatrix} \\
 \text{(f) } A =
 \end{array}
 \end{array}$$

- 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.
- 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  $Ax = b$ .

$$\begin{array}{ll}
 \begin{array}{l}
 \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \\
 A =
 \end{array}
 &
 \begin{array}{l}
 b = [1 \ 1 \ 1 \ 1 \ 1]
 \end{array}
 \end{array}$$

## 18.6 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

$$(1) \quad A = U\Sigma V^T.$$

As we might expect from the case  $A = PDP^{-1}$ , the middle term  $\Sigma$  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,  $\Sigma$  will no longer contain the eigenvalues of  $A$ ; instead it will contain the *singular values* of  $A$ , non-negative

numbers  $\sigma_1 \geq \sigma_2 \geq \dots$ . 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  $U$ ,  $\Sigma$ , and  $V$  from the factorization itself? Since  $U$  and  $V$  are orthogonal, we expect that symmetric matrices are somehow involved. Indeed they are: both  $A^T A$  and  $A A^T$  are symmetric.

We begin with  $A^T A$ . From  $A = U\Sigma V^T$  we find that

$$\begin{aligned} A^T A &= (V\Sigma^T U^T)(U\Sigma V^T) \\ (2) \quad &= V(\Sigma^T \Sigma)V^T. \end{aligned}$$

Since  $\Sigma$  has entries  $\sigma_1, \sigma_2, \dots$  along the main diagonal,  $\Sigma^T \Sigma$  will have diagonal entries  $\sigma_1^2, \sigma_2^2, \dots$ . Thus, equation (2) is an orthogonal diagonalization of the real symmetric matrix  $A^T A$ , so the  $\sigma_1^2, \sigma_2^2, \dots$  are the eigenvalues of  $A^T A$  and the column vectors of  $V$  are orthonormal eigenvectors of  $A^T A$ .

Similarly, we find that

$$\begin{aligned} A A^T &= (U\Sigma V^T)(V\Sigma^T U^T) \\ (3) \quad &= U(\Sigma \Sigma^T)U^T. \end{aligned}$$

Thus, the  $\sigma_1^2, \sigma_2^2, \dots$  are also the eigenvalues of  $A A^T$  and the column vectors of  $U$  are orthonormal eigenvectors of  $A A^T$ .

If  $A$  is a  $5 \times 2$  matrix, then  $A^T A$  is  $2 \times 2$  while  $A A^T$  is  $5 \times 5$ . 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  $A^T A$  is the  $2 \times 2$  matrix

$$A^T A = \begin{bmatrix} 165 & 100 \\ 100 & 120 \end{bmatrix}$$

whose eigenvalues are  $\lambda = 245, 40$ . But  $AA^T$  is the  $5 \times 5$  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  $\lambda$  of  $A^T A$ , say  $(A^T A)x = \lambda x$  for some non-zero  $x$ . Then  $\lambda$  is a real number and we have

$$\begin{aligned} \|Ax\|^2 &= Ax \bullet Ax = (Ax)^T(Ax) \\ &= x^T(A^T Ax) \\ &= x^T \lambda x \\ &= \lambda \|x\|^2 \end{aligned}$$

so that  $\|x\| \neq 0$  implies  $\lambda \geq 0$ . Then, since the non-zero entries  $\sigma_1^2, \sigma_2^2, \dots$  of  $\Sigma^T \Sigma$  are just the non-zero eigenvalues of  $A^T A$ , each  $\sigma_j$  is the (positive) square root of a  $\lambda_j$ :  $\sigma_j = \sqrt{\lambda_j}$ ,  $j = 1, \dots, r$ . (Parenthetical note: notice that in the case where matrix  $A$  has full

column rank,  $A^T A$  is nonsingular, so that  $A^T A x \neq 0_v$ . Thus each eigenvalue  $\lambda$  of  $A^T A$  will satisfy  $\lambda > 0$ , so that  $A^T A$  will be positive definite.)

How many non-zero  $\lambda_j$ 's and  $\sigma_j$ 's are there? The notion of rank is the key:

$$\begin{aligned} \text{rank } A &= \text{rank } (U \Sigma V^T) \\ &= \text{rank } (\Sigma V^T), \text{ since } U \text{ is nonsingular} \\ &= \text{rank } \Sigma, \text{ since } V \text{ is nonsingular} \\ &= \text{the number } r \text{ of non-zero diagonal entries: } \sigma_1, \sigma_2, \dots, \sigma_r. \end{aligned}$$

Normally, we assume that the  $\sigma_j$ 's are arranged on the diagonal of  $\Sigma$  in decreasing order:

$$\Sigma^{m \times n} = \begin{bmatrix} \sigma_1 & & & & & \\ & \sigma_2 & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & \sigma_r & \\ \hline & & & & 0 & \\ & & & & 0 & \end{bmatrix}, \text{ where } \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r.$$

Using equation (2), we see in a similar way that

$$\begin{aligned} \text{rank } A^T A &= \text{rank } (\Sigma^T \Sigma) \\ &= \text{the number } r \text{ of non-zero diagonal entries: } \sigma_1^2, \sigma_2^2, \dots, \sigma_r^2. \end{aligned}$$

Thus  $\text{rank } A = \text{rank } A^T A$ . Replacing  $A$  in this last result with  $A^T$  we have  $\text{rank } A^T = \text{rank } A A^T$ . Thus, all four of these matrices have the same rank:

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

One final comment: although the singular values  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$  are uniquely determined by  $A$  (the  $\sigma_j^2$  are the non-zero eigenvalues of  $A^T A$  and  $A A^T$ ), 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  $3 \times 3$  matrix  $A^T A$  on level 1, the command EGV L 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  $4 \times 4$  matrix  $A A^T$  on level 1, the command EGV L 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

$$U = \begin{bmatrix} [-2.88503909637E-2 & -.157851068131 & -.434161002673 & .88642818039] \\ [.734808458395 & -.200855325398 & .586417975786 & .275368163386] \\ [.486192964128 & -.505806397342 & -.607810084379 & -.371945145644] \\ [.472059939039 & .823953922836 & -.313344990004 & 8.61767003328E-3] \end{bmatrix}$$

$$V = \begin{bmatrix} [-.230770214413 & -.417394627437 & -.878935056262] \\ [-.529224377248 & .811859013819 & -.246589740676] \\ [-.816496580928 & -.408248290464 & .408248290464] \end{bmatrix}$$

$$S = \begin{bmatrix} 23.9965775198 & 10.0580449062 & 5.06289164242E-14 \\ \sigma_1 & \sigma_2 & \sigma_3 \end{bmatrix}$$

- (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

$$AA^T * (\text{col 1 of } U) = \begin{bmatrix} -16.6130860183 & 423.128967009 & 279.967281716 \\ 271.828980859 \end{bmatrix}$$

with the vector

$$\sigma_1^2 * (\text{col 1 of } U) = \begin{bmatrix} -16.6130860182 & 423.128967008 & 279.967281715 \\ 271.828980858 \end{bmatrix}$$

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  $A^T A$  corresponding to the eigenvalue  $\sigma_2^2 = 101.164267335$ : compare the vector

$$A^T A * (\text{col 2 of } V) = \begin{bmatrix} -53.5385963797 & 82.1311223117 & -24.9460704477 \end{bmatrix}$$

with the vector



$$\sigma_2^2 * (\text{col 2 of } V) = \begin{bmatrix} -53.5384963801 & 82.1311223123 & -24.9460704478 \end{bmatrix}.$$

## 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  $\Sigma$  is the  $m \times n$  diagonal matrix

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix},$$

where,  $\Sigma_1$  is the  $r \times r$  ( $r = \text{rank} A$ ) diagonal matrix containing the singular values  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$ . 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

$$\begin{aligned} \|Ax - b\|_2^2 &= \|U^T Ax - U^T b\|_2^2 \\ &= \|\Sigma V^T x - U^T b\|_2^2 \\ &= \|\Sigma y - c\|_2^2, \text{ with } y = V^T x \text{ and } c = U^T b. \end{aligned}$$

Thus  $\|Ax - b\|$  will be minimized if  $\|\Sigma y - c\|_2$  is minimized. Exploiting the special structure of  $\Sigma$  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  $1 \leq j \leq r$ . Since  $x = Vy$ , we can minimize  $\|x\|_2$  by minimizing  $\|y\|_2$ . And since (\*) places no restriction upon the  $y_j$  for  $j = r + 1, \dots, m$ , we can minimize  $\|y\|_2$  by choosing  $y_{r+1} = \dots = y_m = 0$ .

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 turn,

- (i)  $c = U^T b = \begin{bmatrix} \bar{c} \\ d \end{bmatrix}$  ( $\bar{c}$  is an  $r$ -vector)
- (ii) use  $\bar{c}$  to build  $\bar{y} = \Sigma_1^{-1} \bar{c}$  ( $\bar{y}$  is also an  $r$ -vector)
- (iii) use  $\bar{y}$  to build  $x = V \begin{bmatrix} \bar{y} \\ 0 \end{bmatrix}$  (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

$$\bar{c} = [-1.97052724679 \quad -.580094340171].$$

Now delete the last entry in the vector of singular values and use the command `2 DIAG→` to construct the  $2 \times 2$  matrix

$$\Sigma_1 = \begin{bmatrix} 23.9965775198 & 0 \\ 0 & 10.0580449062 \end{bmatrix}.$$

Then calculate

$$\bar{y} = \Sigma_1^{-1} \bar{c} = [-8.21170121099\text{E-}2 \quad -5.76746619826\text{E-}2].$$

Finally, build the 3-vector  $y = [\bar{y} \ 0]$  and calculate the minimum norm least squares solution as

$$x = Vy = [ 4.94729975623\text{E-}2 \quad -1.25484945237\text{E-}2 \quad 8.63975006007\text{E-}2 ].$$

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  $m \times m$  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 18.6

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  $A^T A$  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  $\sigma_1^2$ , where  $\sigma_1$  is the singular value of greatest magnitude.
- (d) Verify that column 3 of  $V$  is an eigenvector of  $A^T A$  associated with the eigenvalue  $\sigma_3^2$ , where  $\sigma_1$  is the singular value of least magnitude.

2. (a) Solve the linear system  $Ax = b$  where

$$A = \begin{bmatrix} 4 & 8 & -6 \\ -6 & 1 & -4 \\ -3 & -7 & 7 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} -16 \\ -2 \\ 17 \end{bmatrix}.$$

- (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  $b = \begin{bmatrix} 1 & 0 & -2 & 1 & 2 \end{bmatrix}$ .
- (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 coefficient matrix  $A$  in a least squares problem  $Ax = b$  has full column rank, the matrix  $A^T A$  appearing in the system of normal equations  $A^T A x = A^T b$  is positive definite (see the parenthetical note on the third page of this section). Using a Cholesky factorization  $A^T A = LL^T$ , the normal equations are  $LL^T x = A^T b$  and can therefore be solved by forward and back substitutions: solve  $Ly = A^T b$  by forward substitution, then  $L^T x = y$  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**.

# 19

## 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.

## 19.1 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

$$(1) \quad Mx_k = Nx_{k-1} + b \quad (k \geq 1).$$

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:

$$\{x_k\} \rightarrow x \quad \text{if} \quad \lim_{k \rightarrow \infty} \|x_k - x\|_{\infty} = 0,$$

where for  $v = [v_1 \ v_2 \ \dots \ v_n]$  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  $\lambda$  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  $x_k = [x_1^{(k)} \ x_2^{(k)} \ \dots \ x_n^{(k)}]$ ). For the Jacobi iteration, we have

$$\begin{aligned} a_{11}x_1^{(k)} &= -a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - \dots - a_{1n}x_n^{(k-1)} + b_1 \\ a_{22}x_2^{(k)} &= -a_{21}x_1^{(k-1)} - a_{23}x_3^{(k-1)} - \dots - a_{2n}x_n^{(k-1)} + b_2 \\ &\vdots \\ a_{nn}x_n^{(k)} &= -a_{n1}x_1^{(k-1)} - a_{n2}x_2^{(k-1)} - \dots - a_{n,n-1}x_{n-1}^{(k-1)} + b_n \end{aligned}$$

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  $(k-1)^{\text{st}}$  iteration (on the right-hand side) to obtain the components of  $x_k$  (on the left-hand side) during the  $k^{\text{th}}$  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:

$$\begin{aligned} a_{11}x_1^{(k)} &= -a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - \dots - a_{1n}x_n^{(k-1)} + b_1 \\ a_{22}x_2^{(k)} &= -a_{21}x_1^{(k)} - a_{23}x_3^{(k-1)} - \dots - a_{2n}x_n^{(k-1)} + b_2 \\ &\vdots \\ a_{nn}x_n^{(k)} &= -a_{n1}x_1^{(k-1)} - a_{n2}x_2^{(k)} - \dots - a_{n,n-1}x_{n-1}^{(k)} + b_n \end{aligned}$$

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  $k^{\text{th}}$  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*:

$$(i) \quad \text{row diagonally dominant, } |a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| \text{ for } i \leq n$$

or

$$(ii) \quad \text{column diagonally dominant, } |a_{jj}| > \sum_{\substack{i=1 \\ i \neq j}}^n |a_{ij}| \text{ for } j \leq n.$$

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  $\epsilon$  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 → ARRAY 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 → ARRAY
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 / - → itmtx « IF itmtx RNRM 1 <
THEN "RNRM < 1" ELSE IF itmtx 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  $\epsilon$  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  $\epsilon$  or when 60 iterations have occurred; display is set to  $n$  FIX where  $n$  is the number of digits in  $\epsilon$ .

```
« → A b ∈ « 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 ∈ 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 + / ∈ < 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  $\epsilon$  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  $\epsilon$  or when 60 iterations have occurred; display is set to  $n$  FIX where  $n$  is the number of digits in  $\epsilon$ .

```
« → A b ∈ « 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 ∈ 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 + / ∈ < 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  $\epsilon = .00005$ , SEIDL shows the iterations converging to  $[-0.37499 \quad 1.71875 \quad -0.03125]$  after only 8 iterations.

**Activity Set 19.1**

1. Consider the following linear system:

$$-3x \quad \quad \quad + 9z + 3w + 2v = -29$$

$$-2x + y \quad \quad \quad + 2w - 6v = 29$$

$$10x + 2y - 4z + 2w - v = -26$$

$$x - 2y + 2z + 7w - v = 6$$

$$-2x + 8y + 4z - w \quad \quad \quad = 24$$

- (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  $5 \times 10^{-8}$ ; how many iterations were required?

(d) What is the exact solution?

4. Consider this tridiagonal system:

$$\begin{array}{rclcl} 2x_1 & + & x_2 & & = & 5 \\ x_1 & + & 2x_2 & + & x_3 & = & -2 \\ & & x_2 & + & 2x_3 & + & x_4 & = & -13 \\ & & & & x_3 & + & 2x_4 & + & x_5 & = & -10 \\ & & & & & & x_4 & + & 2x_5 & = & 8 \end{array}$$

(a) Apply D.DOM, JTEST and STTEST 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

$$\begin{array}{rcccccccl}
 x_1 & - & x_2 & & & & & = & -4 \\
 -x_1 & + & 3x_2 & - & x_3 & & & = & 11 \\
 & - & x_2 & - & 3x_3 & - & x_4 & = & -1 \\
 & & & - & x_3 & + & 3x_4 & - & x_5 & = & 3 \\
 & & & & & - & x_4 & + & 3x_5 & - & x_6 & = & -2 \\
 & & & & & & - & x_5 & + & x_6 & = & -3
 \end{array}$$

- Apply all our tests for convergence. What can you conclude?
- 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.
- What is the actual solution?

## 19.2 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  $\lambda_1$  satisfying

$$|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$$

where  $\lambda_1, \lambda_2, \dots, \lambda_n$  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  $A$ :

- $A$  is real and diagonalizable;



(b)  $A$  has a dominant eigenvalue  $\lambda_1$ .

We start with an arbitrary vector  $y_0$  written in terms of independent eigenvectors  $x_1, x_2, \dots, x_n$  as  $y_0 = a_1 x_1 + \dots + a_n x_n$  where  $a_1 \neq 0$ , and form the sequence of unit vectors  $\{y_k\}$  as  $y_k = \frac{Ay_{k-1}}{\|Ay_{k-1}\|}$ , for  $k = 1, 2, \dots$ . Here, we are using the usual Euclidean vector norm (or length). Thus  $y_1 = \frac{Ay_0}{\|Ay_0\|}$ ,  $y_2 = \frac{Ay_1}{\|Ay_1\|}$ ,  $y_3 = \frac{Ay_2}{\|Ay_2\|}$ , 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  $\lambda_1$ ;

and

(ii) the sequence  $\{Ay_k \bullet y_k\}$  converges to the dominant eigenvalue  $\lambda_1$ .

To see why, consider the sequence  $z_0 = y_0$ ,  $z_k = Az_{k-1}$  ( $k = 1, 2, \dots$ ) without normalization:

$$z_1 = Ay_0, z_2 = Az_1 = A^2y_0, \dots, z_k = Az_{k-1} = A^ky_0.$$

Using  $A^k x_j = \lambda_j^k x_j$  and our expression for  $y_0$  we have

$$\begin{aligned} z_k = A^k y_0 &= a_1 \lambda_1^k x_1 + \dots + a_n \lambda_n^k x_n, \text{ or} \\ (2) \quad z_k &= \lambda_1^k \left[ a_1 x_1 + a_2 \left( \frac{\lambda_2}{\lambda_1} \right)^k x_2 + \dots + a_n \left( \frac{\lambda_n}{\lambda_1} \right)^k x_n \right]. \end{aligned}$$

$$\text{Now } y_1 = \frac{Ay_0}{\|Ay_0\|} = \frac{z_1}{\|z_1\|}, \quad y_2 = \frac{Ay_1}{\|Ay_1\|} = \frac{A\left(\frac{z_1}{\|z_1\|}\right)}{\left\|A\left(\frac{z_1}{\|z_1\|}\right)\right\|} = \frac{Az_1}{\|Az_1\|} = \frac{z_2}{\|z_2\|},$$

and in general,  $y_k = \frac{z_k}{\|z_k\|}$ . Looking back at (2), we see that

$$(3) \quad y_k = \frac{\lambda_1^k \left[ a_1 x_1 + a_2 \left( \frac{\lambda_2}{\lambda_1} \right)^k x_2 + \dots + a_n \left( \frac{\lambda_n}{\lambda_1} \right)^k x_n \right]}{\left\| \lambda_1^k \left[ a_1 x_1 + a_2 \left( \frac{\lambda_2}{\lambda_1} \right)^k x_2 + \dots + a_n \left( \frac{\lambda_n}{\lambda_1} \right)^k x_n \right] \right\|}$$

Since  $|\lambda_1| > |\lambda_j|$  for  $j = 2, \dots, n$ , we have  $\left| \frac{\lambda_j}{\lambda_1} \right| < 1$  for  $j = 2, \dots, m$ . Thus from (3),

$$\text{as } k \rightarrow \infty, \quad y_k \rightarrow \pm \frac{a_1 x_1}{\|a_1 x_1\|} = v,$$

which is a unit eigenvector associated with  $\lambda_1$ .

Finally, since  $y_k \rightarrow v$ ,  $Ay_k \rightarrow Av = \lambda_1 v$ , so that  $Ay_k \bullet y_k \rightarrow \lambda_1 v \bullet v = \lambda_1 (v \bullet v) = \lambda_1 \|v\|^2 = \lambda_1$ . 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  $\lambda_1$ ;
- an associated sequence of numbers  $\{Ay_k \bullet y_k\}$  converging to the dominant eigenvalue  $\lambda_1$ .

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  $\lambda$   
 level 2: an  $n$ -vector  $y_0$ , assumed to have a non-zero component in the direction of an eigenvector associated with  $\lambda$   
 level 1: an accuracy level  $\epsilon$  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  $\epsilon$  or when 180 iterations have occurred. Display is set to n FIX, where n is the number of digits in  $\epsilon$ .

```
« → 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

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

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 19.2

- Seed your random number generator with 2 and generate the following random  $6 \times 6$  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:

- Verify that  $A$  has a dominant eigenvalue  $\lambda$ .
  - Apply the power method via program POWER, starting with the vector of all 1's and using 8 decimal place accuracy.
  - Let  $\tilde{\lambda}$  and  $\tilde{x}$  denote the approximations to  $\lambda$  and an associated eigenvector obtained in (b). Verify that  $A\tilde{x} \approx \tilde{\lambda}\tilde{x}$ .
- 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:  $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$ ). The smaller this factor, the faster the convergence. That is, the convergence will be faster if  $\lambda_1$  *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.

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 6 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

(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  $y_0 = [1 \ 1 \ 1 \ 1 \ 1 \ 1]$ . Aim for 12 decimal place accuracy and note the number of iterations required.

(c) Now apply the power method to matrix  $B$  using  $y_0 = [1 \ 1 \ 1 \ 1 \ 1 \ 1]$ . 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  $5 \times 5$  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  $\|v\| = [x_1^2 + x_2^2 + x_3^2]^{1/2}$ . The well-known properties of vector lengths include:

- (1)  $\|v\| > 0$  if  $v \neq 0$ ,
- (2)  $\|\alpha v\| = |\alpha| \|v\|$  for any scalar  $\alpha$  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  $v = [x_1, x_2, \dots, x_n]$ ,  $\|v\| = [x_1^2 + x_2^2 + \dots + x_n^2]^{1/2}$ . 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  $\alpha$  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  $\|A^n\| \rightarrow 0$  as  $n \rightarrow \infty$ .

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<sup>1</sup> that:

- the vector sum norm induces the *column-sum norm* of  $A$ :

$$\|A\|_1 = \max_j \sum_i |a_{ij}|$$

---

<sup>1</sup>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

$\rho(A) = \max \{ |\lambda| : \lambda \text{ is an eigenvalue of } A \}$ , 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  $\infty$ -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

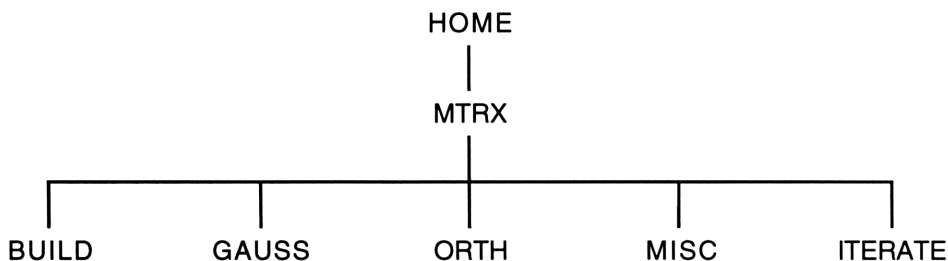
1. What is the Frobenius norm for an identity matrix? Experiment with identity matrices of orders 3, 4, 5, 9 and 16 to find out.
2. How do the row-sum and column-sum norms compare for symmetric matrices? Experiment with random  $3 \times 3$ ,  $4 \times 4$  and  $5 \times 5$  matrices over  $Z_{10}$  to find out.
3. Seed your random number generator with 3 and generate a random  $4 \times 4$  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  $3 \times 3$  matrices over  $Z_{10}$  to find a pair  $A, B$  for which the inequality  $\|AB\| \leq \|A\| \|B\|$  is invalid.

## APPENDIX B

# TEACHING CODE FOR PART IV

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 14): SYMM, L.TRI, U.TRI, DIAG, TRIDIA, HILB.

- Subdirectory **GAUSS**. Contains the teaching code related to Gaussian elimination (see Chapter 15): **ELIM**, **PIVOT**, **L.U**, **FWD**, **BACK**, **L.SWP**, **→LP**.
- Subdirectory **ORTH**. Contains the teaching code related to orthogonality concepts (see Chapter 17): **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 19): **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.

#### 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



# SOLUTIONS

## PART I EXERCISES

## Activity Set 1.1

1. 

$x$	$f(x)$
$\pm 10^{-2}$	.999983333417
$\pm 10^{-3}$	.99999933333
$\pm 10^{-4}$	.99999998333
$\pm 10^{-5}$	.99999999983
$\pm 10^{-6}$	1

2. 

$x$	$f(x)$
$\pm 10^{-2}$	$\mp .004999583$
$\pm 10^{-3}$	$\mp .0005$
$\pm 10^{-4}$	$\mp .00005$
$\pm 10^{-5}$	$\mp .000005$
$\pm 10^{-6}$	0

3. 

$x$	$f(x)$
-1	-.367879441171
-10	-4.5399297625E-4
-1,000	-5.07595889755E-432
-10,000	0

4. See numbers 1, 2, and 3.

5. (a) 

$x$	$f(x)$
$10^2$	2.70481382942
$10^4$	2.71814592683
$10^6$	2.71828046932
$10^8$	2.71828181487
$10^{10}$	2.71828182832
$10^{11}$	2.71828182845

as  $x \rightarrow \infty, f(x) \rightarrow e$

(b) When  $x = 10^{12}$ ,  $f(x)$  is evaluated as 1 by the HP48. Why? The precision of the HP-48 is 12 decimal digits. With  $x = 10^{12}$ ,  $1/x = .000000000001$  and  $1 + 1/x$  is evaluated as 1. Then  $1^x = 1$ .

6. (a)

$x$	$f(x)$	
$10^{-2}$	3.00498756295	
$10^{-3}$	3.00049988491	as $x \rightarrow 0^+, f(x) \rightarrow 3$ .
$10^{-4}$	3.00000400001	
$10^{-5}$	3	

$x$	$f(x)$	
$-10^{-2}$	-.994987437258	
$-10^{-3}$	-.99499873095	as $x \rightarrow 0^-, f(x) \rightarrow -1$ .
$-10^{-4}$	-.99995000125	
$-10^{-5}$	-.999994800014	
$-10^{-6}$	-1	

7. (a) 'IFTE (  $X < 0, X^2, \cos(X)$  )'

(b)

$x$	$f(x)$	
$-10^{-2}$	.0001	
$-10^{-4}$	.00000001	
$-10^{-6}$	.000000000001	as $x \rightarrow 0^-, f(x) \rightarrow 0$ .
$-10^{-8}$	1.E-16	
$-10^{-10}$	1.E-20	

(c)

$x$	$f(x)$	
$10^{-2}$	.999950000417	as $x \rightarrow 0^+, f(x) \rightarrow 1$ .
$10^{-4}$	.99999995	
$10^{-6}$	1	

(d)  $\lim_{x \rightarrow 0} f(x)$  does not exist.

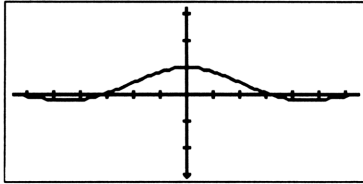
8. (a) 22 (b) -23 (c) 23 (d) 306

9. (a) 23 (b) -22 (c) 24 (d) 307

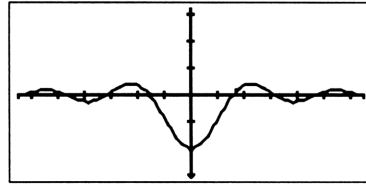


**Activity Set 1.2**

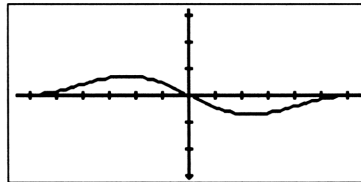
1. (a)



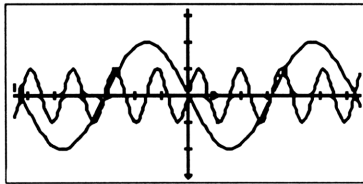
(b)



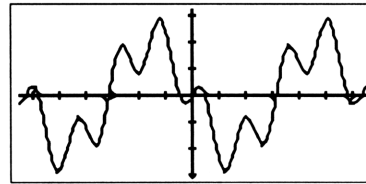
(c)



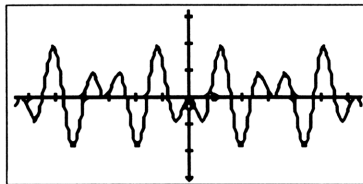
2. (a)



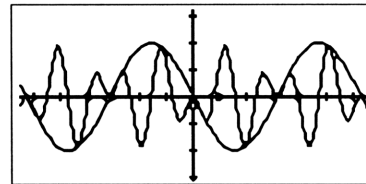
(b)



(d)



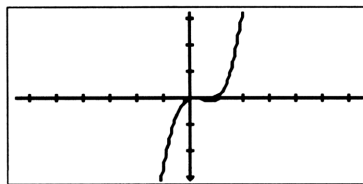
(e)



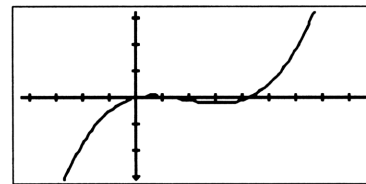
3. HZIN by a factor of 10.

4. Since  $1 \text{ rad} = 180^\circ/\pi \approx 57.2957795131$ , HZOUT by this factor.

5.

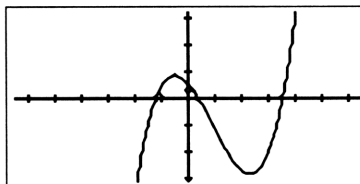


Original

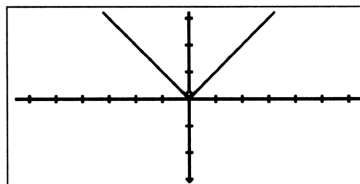
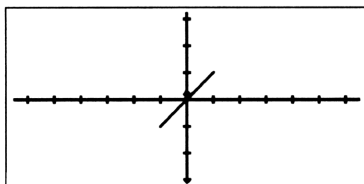


Xrng: -1 2  
Yrng: -1 1

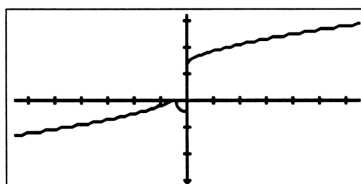
6. YZOUT by a factor of 100 to see the plot



- 7.

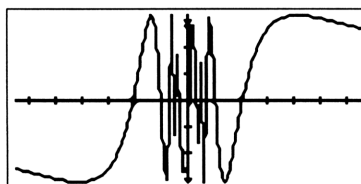


8. Graph with the default PPAR, then ZOUT by a factor of 2 to see



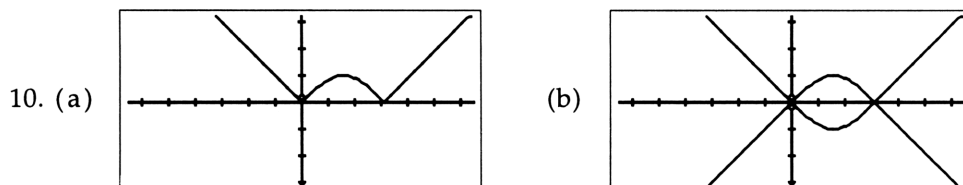
Clearly,  $\lim_{x \rightarrow 0^-} f(x) \neq \lim_{x \rightarrow 0^+} f(x)$ , so  $\lim_{x \rightarrow 0} f(x)$  does not exist.

9. Graph with the default PPAR, then use BOXZ:

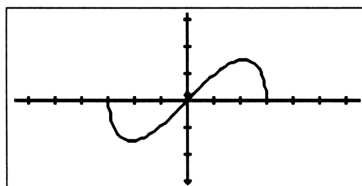


Xrng: -1 1  
Yrng: -1 1

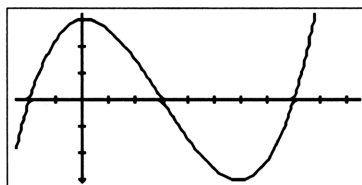
Conclusion:  $\lim_{x \rightarrow 0} f(x)$  does not exist.



11. Plot with the default PPAR, then HZIN by a factor of 1.733 to see



12. The final plot

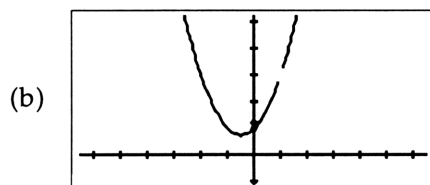


13. (a) Trace to  $x: 1.2$   $y: 4.77$  and press ENTER, then trace to  $x: 1.3$   $y: 4.77$  and press ENTER. Press ON to return to the stack and see

2: (1.2, 4.773333333333)

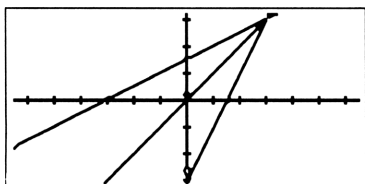
1: (1.3, 4.76692307692)

The local minimum is approximately the point on level 1.



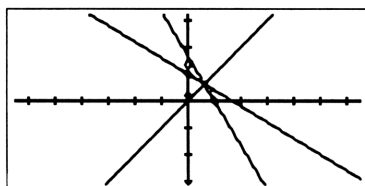
The "hole" is at  $x = 1$ .

14. (a)



$$(b) f_{-1}(x) = \frac{x+3}{2}$$

(c)

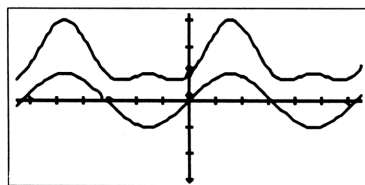


(d) Non-parallel lines that are symmetric to the line  $y = x$  have slopes that are reciprocals of one another.

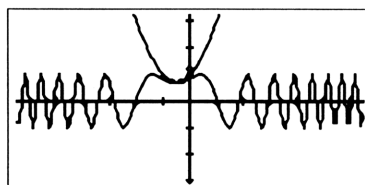
$$g^{-1}(x) = \frac{5}{3}(1-x)$$

(e) The converse is false.

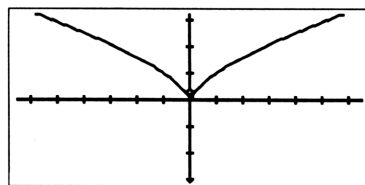
15. (a)


 $u[v(x)] \text{ and } v(x)$ 

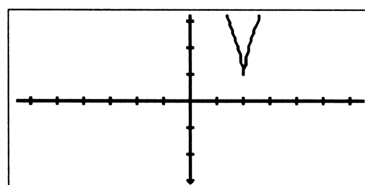
(b)


 $v[u(x)] \text{ and } u(x)$ 

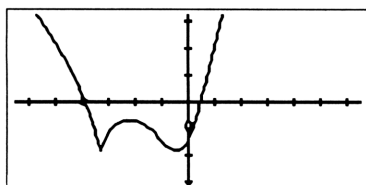
16. (a)



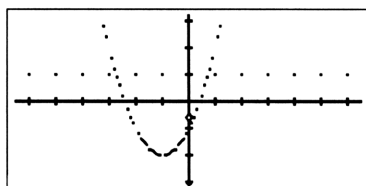
(b)



17. (b)

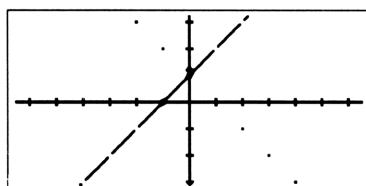
(c)  $(-.3, -1.09614986962)$ 

18. (a)

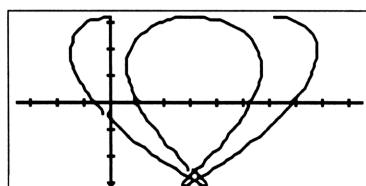


Use disconnected mode.

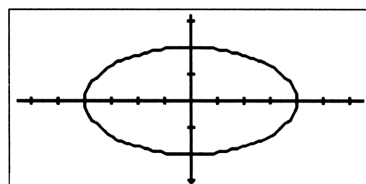
(b)



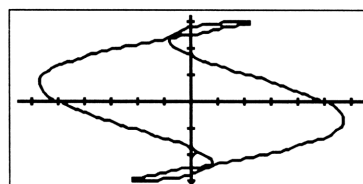
19.

When  $t = 3$ ,  $x = 2.17576302952$  and  $y = 1.9203405733$ .

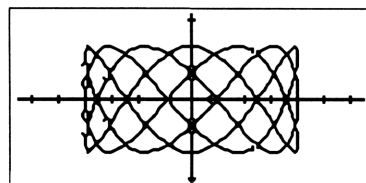
20.



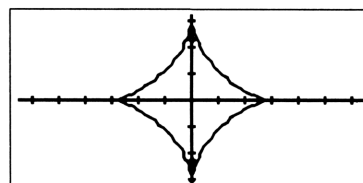
21.



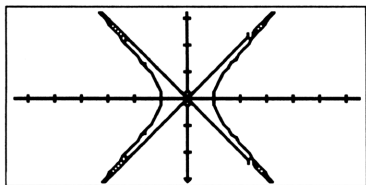
22.



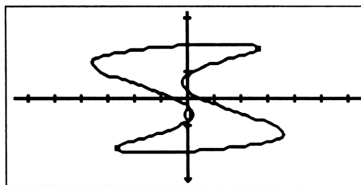
23.



24.



25.

**Activity Set 2.1**

1. (c)  $\frac{\Delta y}{\Delta x} = 1$

(d)	$H$	$DQ(0,H)$
	$\pm 10^{-2}$	.999983333417
	$\pm 10^{-4}$	.999999998333
	$\pm 10^{-5}$	.999999999983
	$\pm 10^{-6}$	1

(e)  $\lim_{h \rightarrow 0} \frac{\sin(0+h) - \sin 0}{h} = \lim_{h \rightarrow 0} \frac{\sin h}{h}$

2. (a) After zooming in on the horizontal axis twice by a factor of 100 each time, tracing shows that the  $y$ -coordinate remains constant at  $y = 1$ . Thus the tangent line at  $x = 0$  is horizontal with slope 0.

(b)	$H$	$DQ(0,H)$
	$\pm 10^{-2}$	$\mp .004999583$
	$\pm 10^{-3}$	$\mp .0005$
	$\pm 10^{-4}$	$\mp .00005$
	$\pm 10^{-5}$	$\mp .000005$
	$\pm 10^{-6}$	0

(c)  $\lim_{h \rightarrow 0} \frac{\cos(0+h) - \cos 0}{h} = \lim_{h \rightarrow 0} \frac{\cos h - 1}{h}$

3. (a) Graph on the default screen; ZIN twice by factors of 100 each time. Trace left to  $x = .99995$ , then right to  $1.00005$ . Calculate  $\frac{\Delta y}{\Delta x} = -.66666664$ . The slope when  $x = 1$  is  $-\frac{2}{3}$ .
- (b) Repeat the procedure in part (a), and calculate  $\frac{\Delta y}{\Delta x} = -2$ . The slope when  $x = 1$  is  $-2$ .
- (c) Repeat the procedure in part (a), and calculate  $\frac{\Delta y}{\Delta x} = 1.9999999666$ . The slope when  $x = 1$  is  $2$ .
- (d) Repeat the procedure in part (a), and calculate  $\frac{\Delta y}{\Delta x} = -2.4783497$ . The slope when  $x = 1$  is  $-2.47834973296$ .

4. (a)	<u><u>H</u></u>	<u><u>DQ(2,H)</u></u>	(b)	<u><u>H</u></u>	<u><u>DQ(-1,H)</u></u>
	$10^{-2}$	.748598999		$10^{-2}$	.492361819
	$10^{-4}$	.7499859		$10^{-4}$	.499925
	$10^{-6}$	.75		$10^{-6}$	.5
	$-10^{-2}$	.751411548		$-10^{-2}$	.5073631939
	$-10^{-4}$	.7500141		$-10^{-4}$	.50007498
	$-10^{-6}$	.75		$-10^{-6}$	.499999
				$-10^{-7}$	.5

The slope at  $x = 2$  is .75

The slope at  $x = -1$  is .5

**Activity Set 2.2**

1. (a)  $\pi/180 \approx .0174532925199$

X	SIN(X)/X
.01	.0174532924313
.001	.0174532925191
.0001	.017432925199
-.01	.017432924313
-.001	.017432925191
-.0001	.017432925199

(c) In degree mode,  $\lim_{x \rightarrow 0} \frac{\sin x}{x} = \frac{\pi}{180}$ .

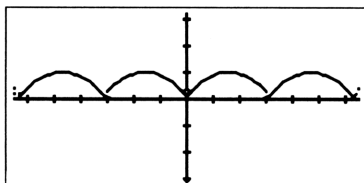
2. (a)  $y' = \frac{1-x^2}{(x^2+1)^2}$

(b)  $y' = \frac{-x \sin \sqrt{x^2+1}}{\sqrt{x^2+1}}$

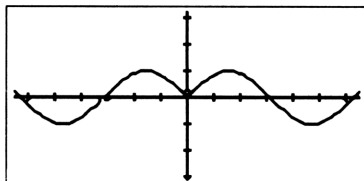
(c)  $y' = \frac{2 \sin x \cos x}{3 \sqrt[3]{\sin^4 x}}$

(d)  $y' = e^{-x/3} \left( 2 \cos 2x - \frac{1}{3} \sin 2x \right)$

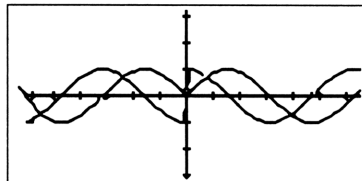
3. (a)

(b)  $y'$  is not defined for the values  $x = n\pi, n = \pm 1, \pm 2, \dots$   
 $y = 0$  for these values of  $x$ .

4. (a)

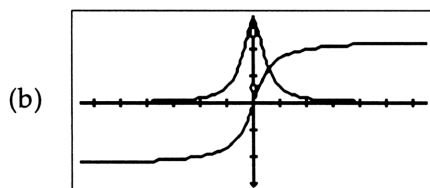
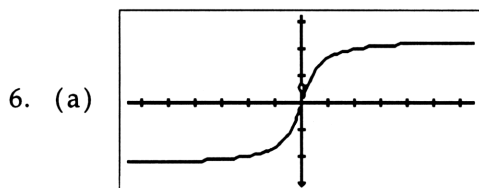


(b)

The derivative is not defined at  $x = 0$ .



5. A horizontal tangent line to the graph of  $f$  will cause the plot of  $f$  to cross the  $x$ -axis.



7. (a)  $y' = \frac{2x}{9y^2}$  (b)  $y' = \frac{-y^2 - 10x}{2xy}$  (c)  $y' = \frac{-e^y - 2xy \cos x^2 y}{xe^y + x^2 \cos x^2 y - 2y}$

(d)  $y' = \frac{\left(y^2 \sin xy^2 + \frac{1}{2\sqrt{x}}\right)}{(3y^2 - 2xy \sin xy^2)}$  (e)  $y' = \frac{-4}{9x^{1/3}y^{1/2}}$  (f)  $\frac{-\sin x}{2 \sin y \cos y}$

8. (a)  $\frac{dy}{dx}(5, -3) = .123456790123$  (b)  $\frac{dy}{dx}(-2, -3) = .91\bar{6}$   
 (c)  $\frac{dy}{dx}(3, \pi/2) = -.426089085256$  (d)  $\frac{dy}{dx}(\pi, 2) = .023507899329$   
 (e)  $\frac{dy}{dx}(8, 9) = -7.40740740748\text{E-}2$  (f)  $\frac{dy}{dx}(\pi/4, \pi/4) = .707106781188$

### Activity Set 2.3.1

- local max:  $(-.577, 2.385)$   
 local min:  $(.577, 1.615)$   
 infl. point:  $(0, 2)$
- local max:  $(.149, 2.127)$   
 local min:  $(.718, -.090)$   
 infl. point:  $(.433, -.044)$
- local max: none  
 local min:  $(-.598, -3.238)$   
 infl. point:  $(0, -2)$  and  $(1, 0)$
- local max:  $(-2.459, 21.968)$  and  $(-.202, 3.092)$   
 local min:  $(-.517, 3.031)$  and  $(.778, 1.319)$   
 infl. point:  $(-1.875, 14.819)$ ,  $(-.364, 3.061)$  and  $(.439, 2.026)$

5. local max:  $(0, 2)$   
local min: none  
infl. point:  $(-.816, 1.5)$  and  $(.816, 1.5)$
6. local max:  $(0, -.8)$   
local min: none  
infl. point: none
7. local max:  $(0, 1)$   
local min: none  
infl. point:  $(-1, 0)$  and  $(1, 0)$
8. absolute max:  $(0, 2\pi)$   
local max:  $(1.912, 4.739)$   
local min:  $(4.373, 1.544)$   
absolute min:  $(0, 0)$   
infl. point:  $(\pi, \pi)$
9. absolute max:  $(2.068, 2.873)$   
local max:  $(.056, 2.028)$   
local min:  $(1.018, -1.41)$   
absolute min:  $(\pi, -2)$   
infl. point:  $(.533, .452)$ ,  $(1.552, .889)$  and  $(2.627, .437)$
10. absolute max:  $(0, 1)$  and  $(\pi, 1)$   
absolute min:  $(\pi/2, -2)$   
infl. point:  $(.704, -.486)$  and  $(2.437, -.486)$
11. absolute max:  $(4.712, 2)$   
local max:  $(.767, .708)$ ,  $(2.375, .708)$  and  $(2\pi, -1)$   
local min:  $(0, -1)$   
absolute min:  $(3.510, -1.634)$  and  $(5.915, -1.634)$
12. local max: none  
local min: none  
infl. point:  $(0, 1)$  and  $(\pi/2, 0)$
13. local max:  $(0, 3)$   
local min: none  
infl. point:  $(-1.414, 1.819)$  and  $(1.414, 1.819)$
14. local max: none  
local min:  $(.368, .692)$   
absolute min:  $(-1, -1)$   
infl. point: none

15. absolute max:  $(-1, 1)$ ,  $(0, 1)$  and  $(1, 1)$   
 absolute min:  $(-.707, -1)$  and  $(.707, -1)$   
 infl. point:  $(-.408, -.111)$  and  $(.408, -.111)$
16. no local extrema  
 infl. point:  $(0, 0)$
17. absolute max:  $(2.128, 1.898)$   
 local max:  $(0, 1)$  and  $(2\pi, 1)$   
 local min: none  
 absolute min:  $(4.843, .209)$   
 infl. point:  $(1.395, 1.388)$  and  $(2.916, 1.270)$
18. none
19. Run the cable from the junction box along the road to a point .894 miles from the closest point on the road to the house, then run straight to the house.
20. The maximum area is achieved with a square pen of side length 62.5 feet situated on the back of the barn.
21. The maximum area is achieved with a rectangular pen  $40 \times 50$  feet in size.

### Activity Set 2.3.2

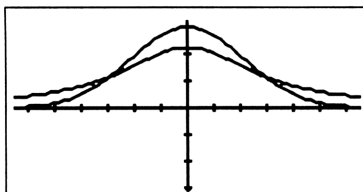
1.	<i>Starting value</i>	<i>Number of iterations</i>	<i>Convergence to</i>
(a)	-2.2	4	-2.2360679775
	2.3	5	between 2.23606797748 and 2.236067897751
	2.9	5	3
(b)	.483	4	.450451159135
	1.79	4	1.74250596672
	2.56	4	2.51943185453
	3.58	5	3.59204381272
	4.88	5	4.8840986203

(c)	-4.71	3	-4.7168606007
	-1.52	4	-1.45367366646
	.552	4	.539785160809
(d)	-.7	5	-.774596669242
	.8	6	.774596669242
(e)	-.9	6	-.923879532511
	-.3	5	-.382683432366

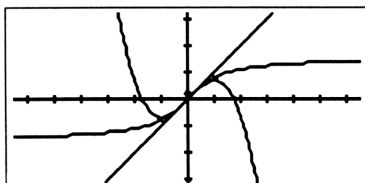
The other two roots are obtained by symmetry.

2. (a) At  $x_0 = \pi/2$ , the denominator in the iteration formula is equal to 0.  
 (b)  $\rightarrow 31.4159265359 (\approx 10\pi)$  after 8 iterations  
 (c)  $\rightarrow -12.5663706144 (\approx 4\pi)$  after 4 iterations  
 (d)  $\rightarrow 3.14159265359 (\approx \pi)$  after 7 iterations  
 (e)  $\rightarrow -3.14159265359 (\approx -\pi)$  after 6 iterations  
 (f)  $\rightarrow 3.14159265359 (\approx \pi)$  after 6 iterations  
 (g)  $\rightarrow 0$  after 6 iterations
3. Newton's method diverges away from 0 for any starting value  $x_0 \neq 0$ .
4. (a) Starting with  $x_0 = 3$ , the iterations oscillate between 3 and 1.  
 (b) Starting with  $x_0 = 2.5$ , the iterations oscillate between 2.5 and 1.5; and starting with  $x_0 = 1.8$ , the iterations oscillate between 1.8 and 2.2. In general, starting with  $x_0 = 2 + h$  or  $x_1 = 2 - h$ , the iterations oscillate between  $x_0$  and  $x_1$ .  
 (c) The tangent lines are parallel; they have the same slope.

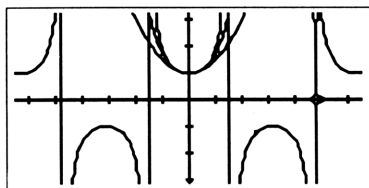
5. (a)

(b) The intersection points are  $(\pm 0.980448246014, 0.382403569603)$ .6.  $\theta \approx 60.72^\circ$ 7.  $a = 127.71148013$ . An equation for the arch is  $y = 630 + 127.7 \left( 1 - \cosh \frac{x}{127.7} \right)$ .**Activity Set 2.3.3**

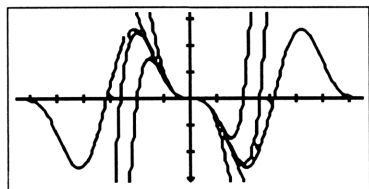
1.



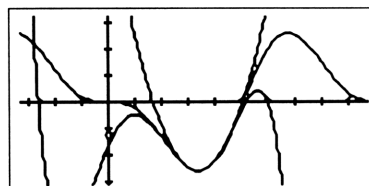
2.



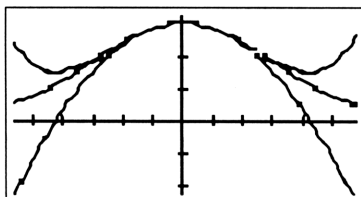
3. (a)



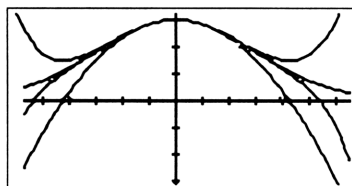
(b)



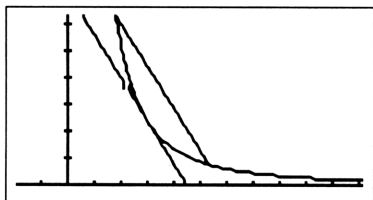
4. (a)



(b)

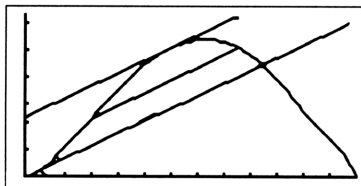
**Activity Set 2.3.4**

1.



Tangent line: ' 1.53352292311 - 3.7980913324 \* (X - .807522931339) '

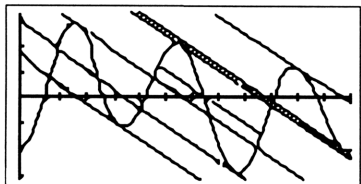
2.



Tangent line: ' 3.01253819463E-2 + .465272415481 \* (X - 9.69188952391E-2) '

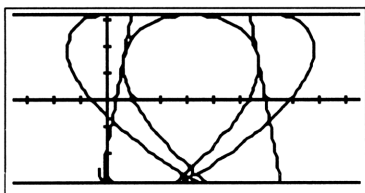
Tangent line: ' 1.23212271031 + .465272415482 \* (X - 1.52177971665) '

3.

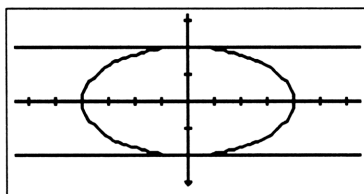


### Activity Set 2.3.5

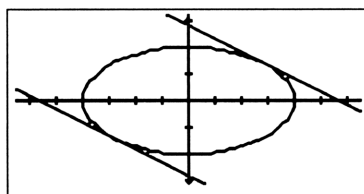
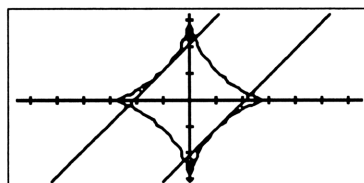
1.

2. (b) (i) Vertical tangents when  $t = 0$  and  $t = \pi$ 

(ii)



(iii)

3. Vertical tangents when  $t = 0, \frac{\pi}{2}, \pi$  and  $\frac{3}{2}\pi$ Tangents when  $t = \frac{3\pi}{4}$  and  $t = \frac{7\pi}{4}$

**Activity Set 3.1.1**

1. (a)

n	LRECT	RRECT	MID
15	.728768788418	.6102502699	.665249108684
30	.697008948527	.637749689263	.666310585448
60	.681659767039	.652030137409	.666577539299
120	.674118653051	.659303838236	.666644378016
200	.671127160162	.662238271273	.666658642265
500	.668447012312	.66489145676	.666665382712
1000	.667556197498	.66577841972	.666666345704

(c) The midpoint approximations are the best.

2. (a)

n	LRECT	RRECT	MID
15	2.42832476027	2.42832476027	2.43889647161
30	2.43361061595	2.43361061595	2.43618981423
60	2.43490021509	2.43490021509	2.43553398801
120	2.43521710155	2.43521710155	2.43537360984
200	2.43528393107	2.43528393107	2.43533989579
500	2.4531525233	2.43531525232	2.43532413036
1000	2.43531969458	2.43531969459	2.43532190445

(c) The midpoint approximations are the best.



3. (a) By symmetry, all midpoint approximations are 0.

(b)

n	LRECT	RRECT
5	-1.89024823416	1.89024823416
20	-.47256205854	.47256205854
40	-.23628102927	.23628102927

### Activity Set 3.1.2

1. (a)  $\text{width} = \frac{4}{N}$

$$\text{right endpoint} = \frac{4K}{N}$$

$$\text{height} = \left(\frac{4K}{N}\right)^3$$

$$\text{area} = \left(\frac{4K}{N}\right)^3 \left(\frac{4}{N}\right)$$

(b)  $S(N) = \sum_{K=1}^N (4 * K/N)^3 * (4/N)$

$N$	$S(N)$
10	77.44
50	66.5856
100	65.2864
200	64.6416

2. (a) width =  $\frac{2}{N}$

$$\text{right endpoint} = 1 + \frac{2K}{N} = \frac{2K + N}{N}$$

$$\text{height} = \frac{N}{2K + N}$$

$$\text{area} = \left( \frac{N}{2K + N} \right) \left( \frac{2}{N} \right)$$

(b)  $S(N) = \sum (K = 1, N, N/(2 * K + N) * (2/N))$

$N$	$S(N)$
10	1.0348956599
50	1.0853974528
100	1.0919752503
200	1.09528636265

3. (a) width =  $\frac{4}{N}$

$$\text{right endpoint} = -1 + \frac{4K}{N} = \frac{4K - N}{N}$$

$$\text{height} = e^{\left( \frac{4K - N}{N} \right)}$$

$$\text{area} = \left( e^{\left( \frac{4K - N}{N} \right)} \right) \left( \frac{4}{N} \right)$$

(b)  $S(N) = \sum (K = 1, N, \text{EXP}((4 * K - N)/N) * (4/N))$

$N$	$S(N)$
10	23.923392666
50	20.5168787439
100	20.1146395823
200	19.9154913077

4. (a) width =  $\frac{2}{N}$

$$\text{left endpoint} = -3 + \frac{2(K-1)}{N} = \frac{2K-3N-2}{N}$$

$$\text{height} = 3 + \left( \frac{2K-3N-2}{N} \right)^2 + \left( \frac{2K-3N-2}{N} \right)$$

$$\text{area} = \left( 3 + \left( \frac{2K-3N-2}{N} \right)^2 + \left( \frac{2K-3N-2}{N} \right) \right) \left( \frac{2}{N} \right)$$

(b)  $S(N) = \sum_{K=1}^N \left( 3 + \frac{2K-3N-2}{N} + \left( \frac{2K-3N-2}{N} \right)^2 \right) \left( \frac{2}{N} \right)$

$N$	$S(N)$
10	-2.44756241499
50	-2.09256321762
100	-2.04647408997
200	-2.0232851862

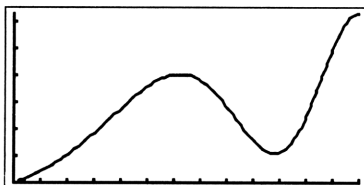
### Activity Set 3.1.3

1.	AREA = .666666666667	N	TRAP	SIMP
		50	.666923371902	.666666687877
		100	.666730858883	.66666667993
2.	AREA = 3.71221866457	N	TRAP	SIMP
		100	3.71108428728	3.71205883637
3.	AREA = -1.38542551654	N	TRAP	SIMP
		100	-1.38630748336	-1.38542551069

4.	AREA = 2.36567536982	N	TRAP	SIMP
		100	2.36496414195	2.36567538512
5.	AREA = 1.15444851259	N	TRAP	SIMP
		100	1.544445492303	1.5444851427
6.	AREA = -2	N	TRAP	SIMP
		50	-1.99922988429	-1.9999993636
		100	-1.99980742337	-1.999999996
7.	AREA = .886207348259	N	TRAP	SIMP
		100	.886207292754	.88620734825
8.	AREA = 1.58846779582	N	TRAP	SIMP
		100	1.58848892823	1.58846779602
9.	AREA = 2.50798211423	N	TRAP	SIMP
		100	2.50797278189	2.50798211428
10.	AREA = 1.08531739235	N	TRAP	SIMP
		100	1.08532405829	1.0853173924
		200	1.08531905887	1.08531739236
11.	AREA = 1.446560 (Using 6 FIX )	N	SIMP	
		100	1.44655963409	
		200	1.44656012035	

**Activity Set 3.2.1**

1. (a)



(b) AREA: 2.45375670998

(c) AREA: 1.52818183784

2. Volume: 41.1062399578 (units)<sup>2</sup>3. Volume: 1.65549327405 (units)<sup>2</sup>

4. The "arch length" is 1493.74 feet

5. (a) 18.001 (b) 40.095

6. Period  $\approx 2.554$  m**Activity Set 3.2.2**

1.  $\frac{14}{5} \sqrt{5x-1}$

2.  $-(2x+3)^{-1/2}$

3.  $\tan x - x$

4.  $\sec x$

5.  $\frac{1}{2} \left[ (2x+3) \tan^{-1}(2x+3) - \frac{1}{2} \ln(1+(2x+3)^2) \right]$

6.  $\frac{1}{10} (5x-\pi)^6$

7.  $-\frac{2}{\sin .5x}$

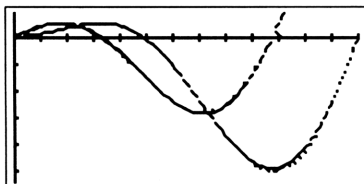
8.  $\frac{x^2}{2} - \ln \cosh x$

9.  $\frac{2^{x+1}}{\ln 2}$

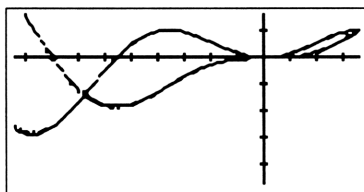
10. If  $s-1 = -1$ ,  $\ln x$ ; otherwise  $\frac{x^s}{s}$ .

### Activity Set 3.3

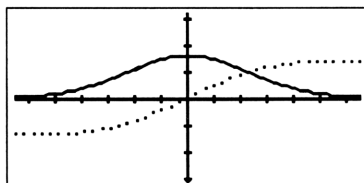
1. The final plot:



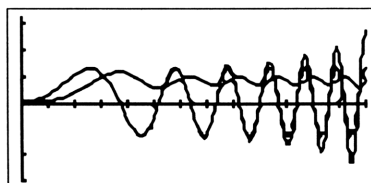
2. The final plot:



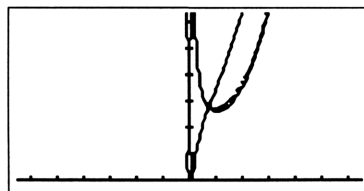
3. The final plot:

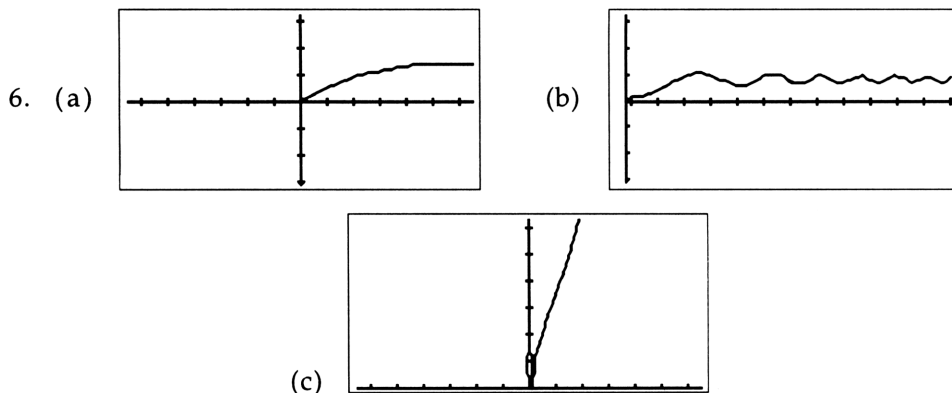


4. The final plot:



5. The final plot:





### Activity Set 3.4

1.  $\left| \frac{\sin x}{x^4} \right| \leq \frac{1}{x^4}$ . With  $K = 1$ ,  $p = 4$  and  $\epsilon = .01$  we have  $N > 3.22$ . Thus, the integral  $\approx .56$  to within  $\epsilon$ .
2.  $\left| \frac{x}{\sqrt{x^6 + 4}} \right| < \frac{1}{x^2}$ . With  $K = 1$ ,  $p = 2$  and  $\epsilon = .001$  we have  $N > 1000$ . Thus, the integral  $\approx 1.112$  to within  $\epsilon$ .
3.  $\left| \frac{xe^{-2x}}{\sqrt[3]{x^3 + 1}} \right| < e^{-2x}$ . With  $K = 1$ ,  $c = 2$  and  $\epsilon = .001$  we have  $N > 3.2$ . Thus, the integral  $\approx .198$  to within  $\epsilon$ .
4.  $\left| \frac{1}{\sqrt{x^5 - 1}} \right| < \frac{2}{x^{5/2}}$  for  $x \geq 2$ . With  $K = 2$ ,  $p = 5/2$  and  $\epsilon = .001$  we have  $N > 121.2$ . Thus, the integral  $\approx .236$ .
5. Using the hint, for  $x \geq 0$   $\left| \sqrt{x+1} e^{-2x} \right| < \left( \frac{1}{e} \right) e e^{-x}$ . Thus  $K = 1$  and  $c = 1$ . For  $\epsilon = .005$ ,  $N > 1n200 \approx 5.3$  and the integral  $\approx .605$ .

### Activity Set 4.1

- (a) - (b). The terms appear to approach 0.

(c) The graph approaches the  $x$ -axis.

(d) by l-Hospital's Rule,  $\lim_{x \rightarrow \infty} \frac{x^2}{2^x} = 0$ . Thus  $\lim_{k \rightarrow \infty} \frac{k^2}{2^k} = 0$ .
- (a) - (b). The terms appear to approach 0.

(c) The graph approaches the  $x$ -axis.

(d) by l-Hospital's Rule,  $\lim_{x \rightarrow \infty} \frac{\ln x}{x} = 0$ . Thus  $\lim_{k \rightarrow \infty} \frac{\ln k}{k} = 0$ .
- (a) - (b). The terms appear to approach 0.

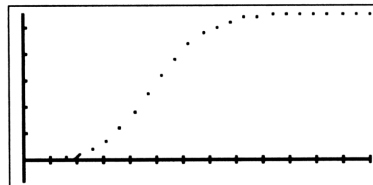
(c) The plot suggests that  $\lim_{k \rightarrow \infty} \frac{\sin k}{e^{.05k}} = 0$ .  
The Rule of l-Hospital is of no help.
- (a) - (b). The terms appear to approach 0.

(c) Use  $f(x) = \frac{1}{x}$ ;  $\lim_{k \rightarrow \infty} (-1)^k + 1 \left( \frac{1}{k} \right) = 0$ .
- (a)  $\left\{ \frac{k}{\sqrt{k}+1} \right\}$  diverges      (b)  $\left\{ \frac{\cos k}{\sqrt{k}} \right\}$  converges to 0.

### Activity Set 4.2.1

- (a) Use the ratio test

(b)

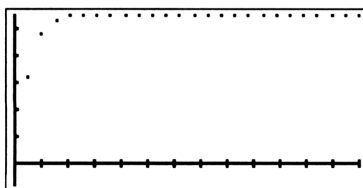


(c) sum  $\approx 2026.4657948$

(d) The 38<sup>th</sup> partial sum.

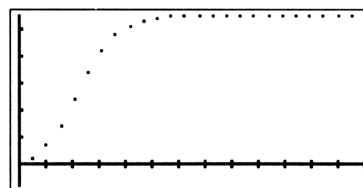


2. (a) Use the ratio test.



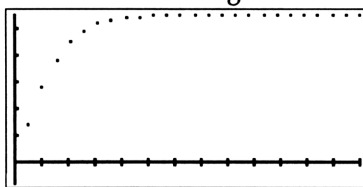
sum  $\approx 1.71828182846$   
The 14<sup>th</sup> partial sum

- (b) Use the ratio test



sum  $\approx 147.413159104$   
The 27<sup>th</sup> partial sum

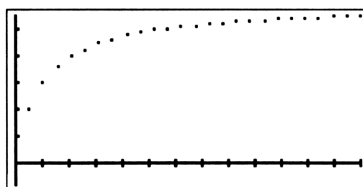
- (c) Use the alternating series test



sum  $\approx 2$   
The 42<sup>nd</sup> partial sum

3. (a) Use the alternating series test. (b)  $n = 15$  (c)  $S_{15} = 1.71828182846$

4. Sum  $\approx 2.00300296795$



5. (b)

- (c) For sufficiently large  $k$ ,  $\frac{k!}{(k+2)!}$  is sensed as 1 by the HP-48. From that point on, the partial sums accumulate by 1.

**Activity Set 4.2.2**

1.  $\int_1^{\infty} \frac{\sin x}{x^4} dx$  converges on comparison with  $\int_1^{\infty} \frac{1}{x^4} dx$  and  $\left| \frac{\sin x}{x^4} \right| < \frac{2}{x^4}$ ; use  $K = 2$

and  $p = 4$ . With  $\epsilon = .01$ ,  $N > 4.05$ . Thus the sum  $\approx \sum_{k=1}^5 \frac{\sin k}{k^4} = .90$  to

within  $\epsilon = .01$ .

2.  $\int_1^{\infty} \frac{x}{\sqrt{x^6 + 4}} dx$  converges on comparison with  $\int_1^{\infty} \frac{1}{x^2} dx$  since  $\left| \frac{x}{\sqrt{x^6 + 4}} \right| < \frac{1}{x^2}$  for

$x > 1$ ; use  $K = 1$  and  $p = 2$ . With  $\epsilon = .001$ ,  $N > 1000$ . Thus the sum  $\approx$

$$\sum_{k=1}^{1001} \frac{k}{\sqrt{k^6 + 4}} = 1.083 \text{ to within } \epsilon.$$

3. Let  $f(x) = \frac{xe^{-2x}}{\sqrt[3]{x^3 + 1}}$ . Then  $\int_1^{\infty} f(x) dx$  converges on comparison with  $\int_1^{\infty} e^{-2x} dx$  since  $|f(x)| < e^{-2x}$ ; use  $K = 1$  and  $c = 2$ . With  $\epsilon = .0012$ ,  $N > 3.1$ . Thus the sum  $\approx \sum_{k=1}^4 a_k = .128$  to within  $\epsilon$ .

4.  $\int_2^{\infty} \frac{x}{\sqrt{x^5 - 1}} dx$  converges on comparison with  $\int_2^{\infty} \frac{1}{x^{1.2}} dx$  since  $\frac{x}{\sqrt{x^5 - 1}} < \frac{1}{x^{1.2}}$  for  $x > 1.41$ ; use  $K = 1$  and  $p = 1.2$ . With  $\epsilon = .001$ ,  $N > 5.5$ . Thus the sum  $\approx \sum_{k=2}^6 \frac{k}{\sqrt{k^5 + 1}} = .835$  to within  $\epsilon$ .

5.  $\int_1^{\infty} \ln \left( \frac{1}{x^2} \right) dx$  converges on comparison with  $\int_1^{\infty} \frac{1}{x^2} dx$  since  $\ln \left( \frac{1}{x^2} \right) < \frac{1}{x^2}$  for  $x > 0$ ; use  $K = 1$  and  $p = 2$ . With  $\epsilon = .0001$ ,  $N > 10,000$ . Thus the sum  $\approx \sum_{k=1}^{10,001} \ln \left( \frac{1}{k^2} \right) = -164,236.277$  to within  $\epsilon$ .

**Activity Set 4.2.3**

1. Since  $r = \frac{1}{2}$  let  $\Gamma = \frac{3}{4}$ . The smallest value of  $N$  that satisfies (1) is  $N = 3$ .

The smallest value of  $N$  that satisfies (2) is  $N = 27$ . Thus the sum  $\approx$

$$\sum_{k=1}^{27} \left( \frac{k}{2^k} \right) = 2 \text{ to within } \epsilon = 10^{-6}.$$

2. Since  $r = \frac{1}{e}$  let  $\Gamma = \frac{1}{2}$ . The smallest value of  $N$  that satisfies (1) is  $N = 10$ .

The smallest value of  $N$  that satisfies (2) is  $N = 24$ . Thus the sum  $\approx$

$$\sum_{k=1}^{24} \left( \frac{k^3}{e^k} \right) = 6.006512 \text{ to within } \epsilon = 10^{-6}.$$

3. Since  $r = 0$  let  $\Gamma = \frac{1}{2}$ . The smallest value of  $N$  that satisfies (1) is  $N = 3$ .

The smallest value of  $N$  that satisfies (2) is  $N = 12$ . Thus the sum  $\approx$

$$\sum_{k=1}^{12} \frac{(k+1)(k+2)}{k!} = 17.027973 \text{ to within } \epsilon = 10^{-6}.$$

4. Since  $r = 0$  let  $\Gamma = \frac{1}{2}$ . The smallest value of  $N$  that satisfies (1) is  $N = 4$ .

The smallest value of  $N$  that satisfies (2) is  $N = 14$ . Thus the sum  $\approx$

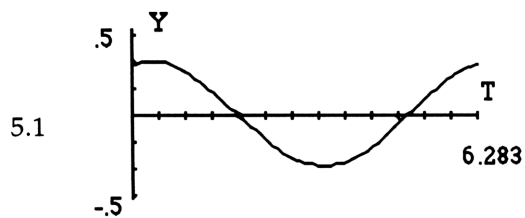
$$\sum_{k=1}^{14} \frac{10^k k!}{(2k+1)!} = 5.655198 \text{ to within } \epsilon = 10^{-6}.$$

5. Since  $r = .7$  let  $\Gamma = \frac{3}{4}$ . The smallest value of  $N$  that satisfies (1) is  $N = 1$ .

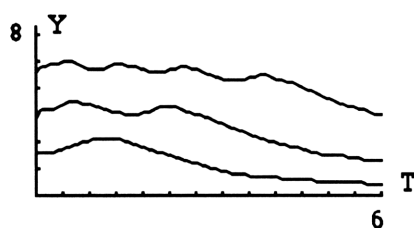
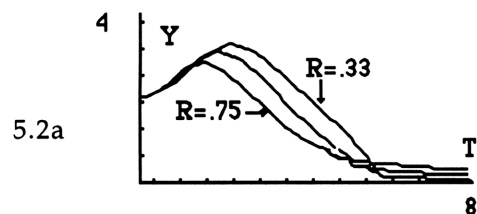
The smallest value of  $N$  that satisfies (2) is  $N = 42$ . Thus the sum  $\approx$

$$\sum_{k=1}^{42} \frac{.7^k (k+1)}{k} = 3.537305 \text{ to within } \epsilon = 10^{-6}.$$

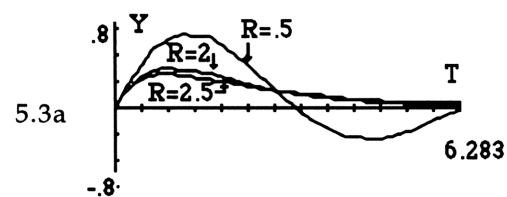
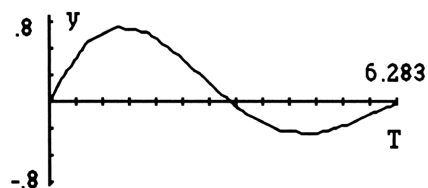
## PART II EXERCISES



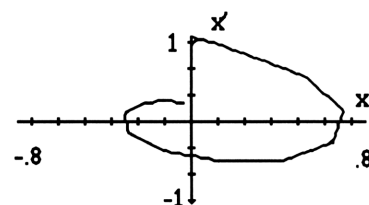
5.2

Note  $Y(6.283) = .3$ 

5.3



5.5

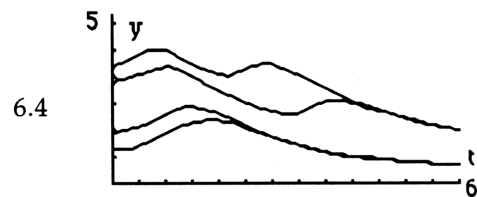
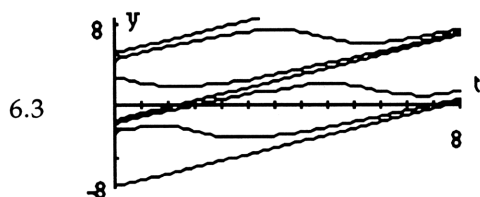
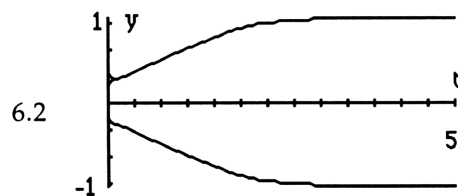
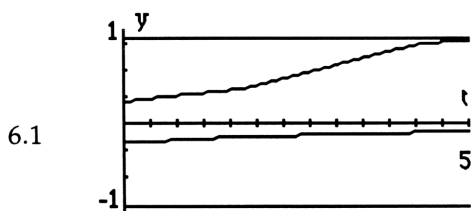
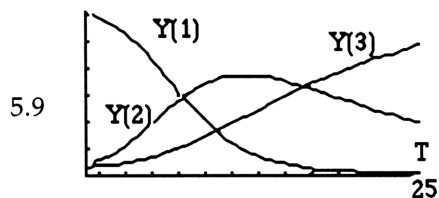


- 5.7 Starting at  $T = 1$ ,  $Y = 1$  for  $H = .2$  we get  $Y = 1.4$ ,  $Y = 1.96$ ,  $Y = 2.788$ ,  $Y = 4.110$ , and  $Y = 6.443$  when  $T = 1.4$ ,  $T = 1.6$ ,  $T = 1.8$ , and  $T = 2$  respectively.

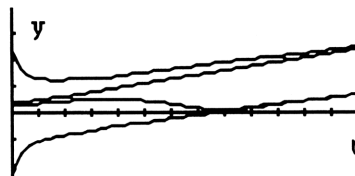
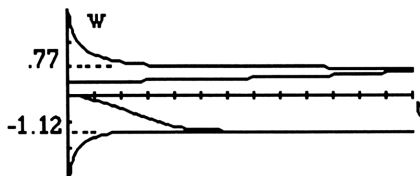
Starting at  $T = 1$ ,  $Y = 1$  for  $H = .1$  we get  $Y = 1.44$ ,  $Y = 2.097$ ,  $Y = 3.169$ ,  $Y = 5.168$ , and  $Y = 9.839$  when  $T = 1.4$ ,  $T = 1.6$ ,  $T = 1.8$ , and  $T = 2$  respectively.

Starting at  $T = 1$ ,  $Y = 1$  for  $H = .05$  we get  $Y = 1.467$ ,  $Y = 2.196$ ,  $Y = 3.484$ ,  $Y = 6.280$ , and  $Y = 15.693$  when  $T = 1.4$ ,  $T = 1.6$ ,  $T = 1.8$ , and  $T = 2$  respectively.

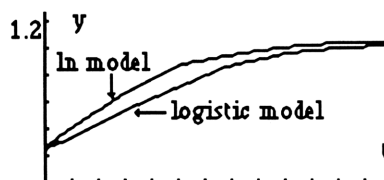
5.8 See page 200.



6.5 The equilibrium solutions are  $w = -1.125$ ,  $w = .339$  and  $w = .786$ .



6.6 There is only one 'equilibrium' solution 6.7



6.8 The substitution  $x = \omega^4$  leads to the equation

$$\int \frac{4\omega^3}{(1-\omega)(1+\omega+\omega^2+\omega^3+\omega^4)} = t + C$$

The integrand is 
$$\frac{A}{1-\omega} + \frac{B\omega+C}{\omega^2+.5(1+\sqrt{5})\omega+1} + \frac{D\omega+E}{\omega^2+.5(1-\sqrt{5})\omega+1}$$

for  $A = .8$ ,  $B = 2.3677$ ,  $C = 1.2944$ ,  $D = -1.5677$ ,  $E = -.4944$ . An antiderivative of the left side is  $.5B \ln \{\omega^2 + .5(1 + \sqrt{5})\omega + 1\} + (C - .25B(1 + \sqrt{5}))1.7 \tan^{-1} \sqrt{f(2\omega + 1.618, 1.1755)} + .5D \ln \{\omega^2 + .5(1 - \sqrt{5})\omega + 1\} + (E - .25D(1 - \sqrt{5}))1.0515 \tan^{-1} \frac{2\omega - .6}{1.1902}$

$-A \ln |1-\omega|$ . We replace  $\omega$  by  $x^{.25}$  and use the initial condition  $x(0) = .5$  to evaluate the constant of integration. This analytical procedure is already quite formidable and we must still invert the expression to graph  $x(t)$ . What if a parameter value in the equation changes? This adds further complication to the analytical procedure. The case  $r = 5/2$  is similar.

6.9 If  $Q(t)$  is the weight of salt in the tank at time  $t$  (in minutes), we have that  $\frac{dQ}{dt} = 2(1.5) - \frac{Q}{300-t}$ ,  $Q(0) = 0$ . This equation has solution  $Q = 450x(1-x^2)$  where  $x = \frac{300-t}{300}$ . If we put  $Q = 21$ , we get  $x$  solutions .04677 and .9758 which translate to  $t = 7.262$  and  $y = 286$ .

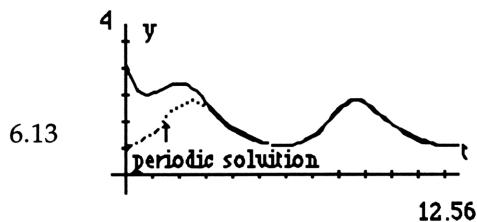
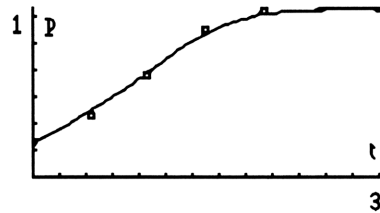
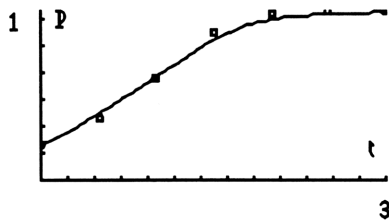
6.10 Following the hint gives  $t_k \ln \left\{ \frac{\ln(p(t_i)x)}{\ln(p(0)x)} \right\} = t_i \ln \left\{ \frac{\ln(p(t_k)x)}{\ln(p(0)x)} \right\}$  for  $x = e^{-s}$ .

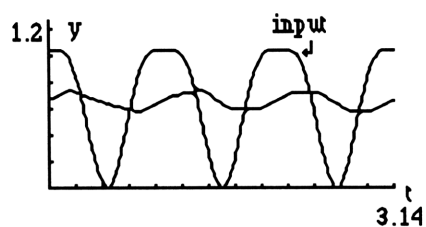
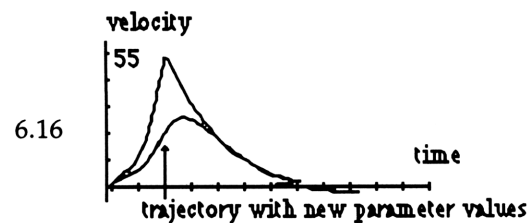
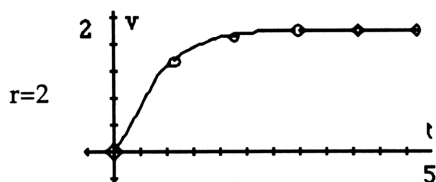
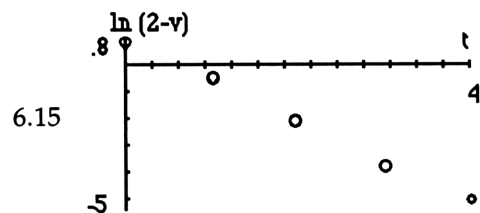
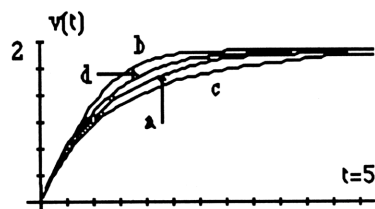
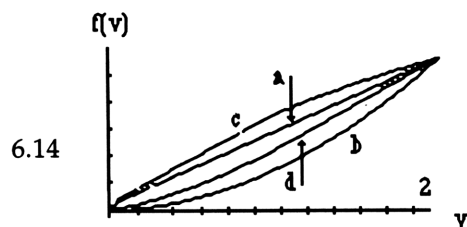
Using the solver we solve this equation for  $x$ . This gives a value for  $A/B = -\ln x$ .  $B$  is given by  $-Bt_i = \ln \left\{ \frac{\ln(p(t_i)x)}{\ln(p(0)x)} \right\}$ . For the case at hand, we get

$x = .665282$  which gives  $B = 2.6391$  and  $A = 1.07557$ . Plotting the resulting solution gives the expected results.

6.12 Clearly  $K = 1$ , so a quadratic model as constructed in Exercise 6.11 has the form

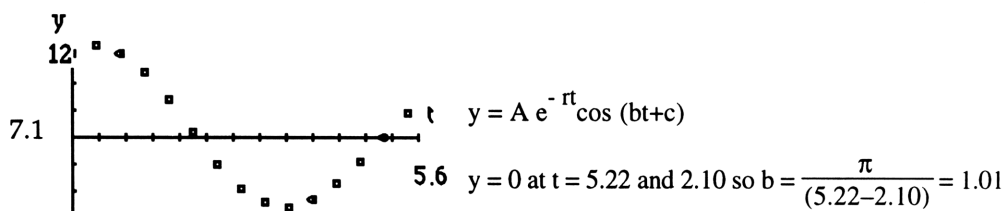
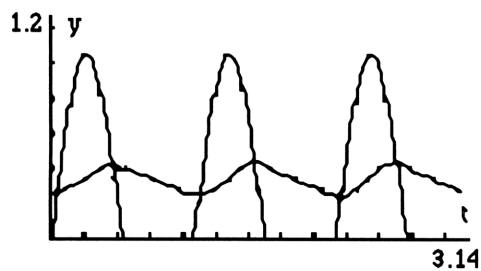
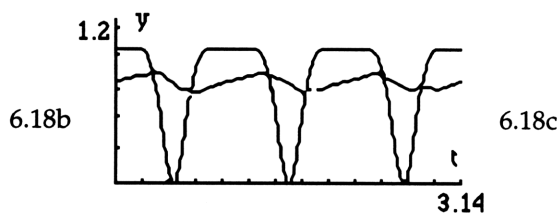
$Y' = \sigma \text{ IFTE}(Y < \theta, \theta^2 - (Y-\theta)^2, \frac{\theta^2}{(1-\theta)^2} [(1-\theta)^2 - Y-\theta)^2)$  for some  $s > 0$ . The following pictures were obtained by using  $\theta = .6$ ,  $\sigma = 1.4$  and  $\theta = .65$ ,  $\sigma = 1.25$ .



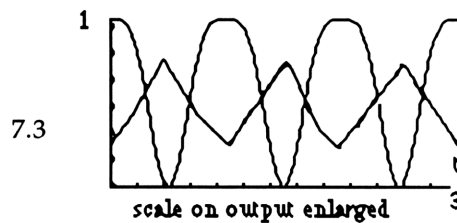
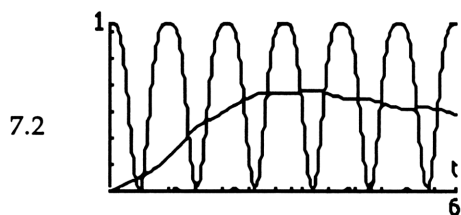


peak at  $t = 5.6$ ,  $v = 29$ , final velocity =  $-3.64$ .



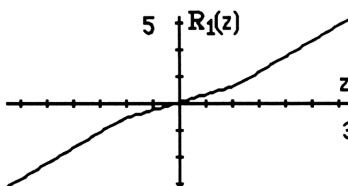


and  $12 e^{-.4r} = 9 e^{-r(.4+\pi/b)}$  so  $r = -b (\ln .75)/\pi = .092$ . Thus  $B = .184$  and  $C = (.092)^2 + (1.01)^2 = 1.029$ .

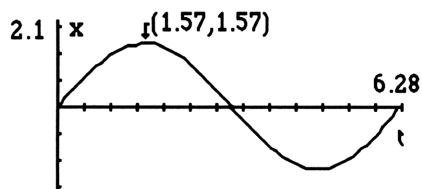


The calculator gives  $x_q(1) = .1973$ ,  $x_q'(1) = .3228$  so initial conditions are  $x(0) = .488$ ,  $x'(0) = .012$ .

7.4 A graph of  $R_1(z)$  vs  $z$  is



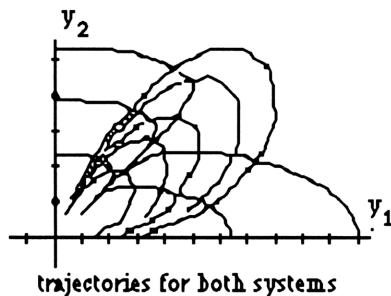
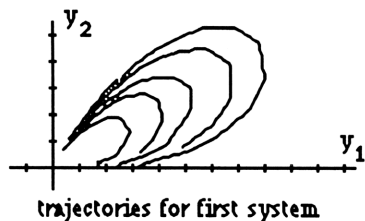
7.4a

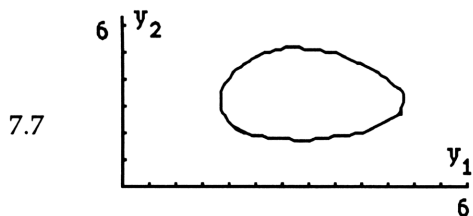


Contrast (a) with (c) solution  $x = 2 \sin t$ .

7.5 The program `<< → W << W DUP 2 ^ 1 - NEG 2 * INV → C << C 2 * NEG 0 XRNG C W * DUP NEG SWAP YRNG ERASE { # 0d # 0d } PVIEW 1 200 FOR N N 2 * P * W * → NUM DUP SIN SWAP COS 1 - C * SWAP W * C * NEG R→C PIXON NEXT >> >> >>` requires input  $W$  and outputs the desired graph and leaves a copy of  $W$  on the stack. Try for  $W = 19/20$ ,  $W = 6/7$ ,  $W = 1/\sqrt{7}$ .

7.6



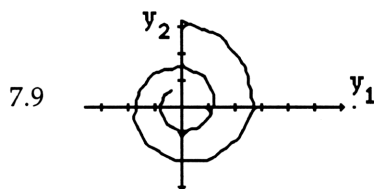


7.8b

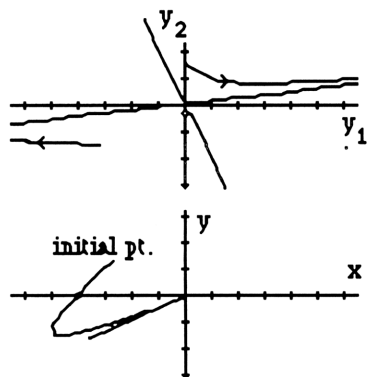
$$P(z) = \frac{4}{\sqrt{2}} \int_0^z \frac{dx}{\sqrt{\cos x - \cos z}}$$

$P(\pi/4) = 6.5340$  (takes HP about 9 minutes at 4 FIX to evaluate integral)

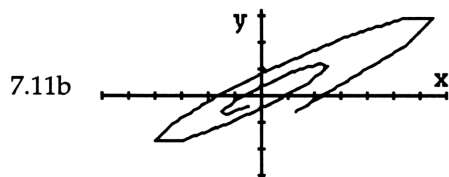
$P(\pi/2) = 7.4158$



7.10 The system has eigenvalues  $\pm \sqrt{5}$  and eigenvectors proportional to  $[4.236, 1]$  and  $[.2361, -1]$ . For initial conditions not proportional to the latter vector the trajectory will quickly approach the first vector as  $t$  increases.



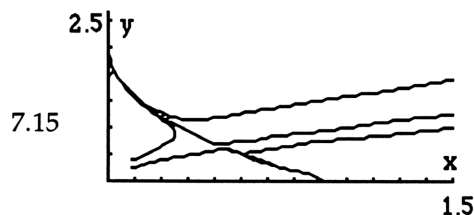
In 7.11a the final value of  $[x, y]$  was  $[-.333, -.198]$ . The ratio of  $y$  to  $x$  was 1.68. We show in the figure both the trajectory and the line  $y = 1.67x$ . The eigenvalues of the matrix are -1.5 and -.5 with associated eigenvectors  $[1, 1]$  and  $[1, .6]$ .



$$\lambda^2 + .6\lambda + 4.09$$

$$\lambda = -.3 \pm 2i$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = e^{-.3t} \left\{ \alpha \left( \begin{bmatrix} 1 \\ .6 \end{bmatrix} \cos 3t + \begin{bmatrix} 0 \\ .2 \end{bmatrix} \sin 3t \right) + \beta \left( \begin{bmatrix} 1 \\ .6 \end{bmatrix} \sin 3t - \begin{bmatrix} 0 \\ .2 \end{bmatrix} \cos 3t \right) \right\}$$



8.1a  $x^2 + 2x + 5 = 0$       8.1b  $x^2 + 1001x + 1000 = 0$       8.2a  $x^3 - 6x^2 + 11x - 6 = 0$

8.2b  $x^3 - 3x^2 + 4 = 0$       8.2c  $x^3 + 2x^2 - 21x - 6 = 0$

8.3a 2, 3, 1      8.3b -1, 2, 2      8.3c -5.574..., -.279..., 3.853...

8.4a -1.0979...  $\pm i$ .785..., 2.1958...      8.4b -.347..., 1.879..., -1.532...

8.4c  $-1 \pm 2i, -3$

8.5a 
$$Y(t) = \begin{bmatrix} -e^t & e^{-t} & -1 \\ 0 & .5 e^{-t} & 1 \\ e^t & -e^{-t} & 0 \end{bmatrix}$$

8.5b column 1 =  $e^{\alpha t} \begin{bmatrix} .1776 \cos \beta t - .5195 \sin \beta t \\ -.6027 \cos \beta t + .4309 \sin \beta t \\ \cos \beta t \end{bmatrix},$

column 2 =  $e^{\alpha t} \begin{bmatrix} .1776 \sin \beta t + .5195 \cos \beta t \\ -.6027 \sin \beta t - .4309 \cos \beta t \\ \sin \beta t \end{bmatrix}$

with  $\alpha = -1.079$ ,  $\beta = .785\dots$  and column 3 =  $e^{2.196t}$  column  $[.2074, .4554, 1]$ .

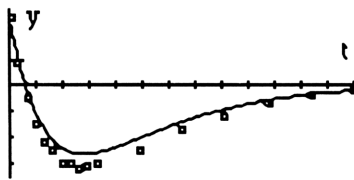
8.6a 
$$Y(t) = \begin{bmatrix} e^{-t} & e^{2t} & -.0174 e^{-t} \\ -.5e^{-t} & e^{2t} & -.983 e^{-t} \\ -.5 e^{-t} & e^{2t} & e^{-t} \end{bmatrix}$$

8.6b The eigenvalues are  $-1$  (repeated) and  $-2$ . Let  $\mathbf{a}$  = column  $[1, 2, 4]$ , an eigenvector corresponding to the eigenvalue  $-1$ . We try  $\mathbf{y} = e^{-t}(\mathbf{a}t + \mathbf{b})$  as a solution to  $\mathbf{y}' = \mathbf{A}\mathbf{y}$  and see this will be a solution if  $(\mathbf{A} + \mathbf{I})\mathbf{b} = \mathbf{a}$ . We create the 3 by 4 matrix  $[\mathbf{A}, \mathbf{a}]$  and reduce it by using the **RREF** command or a series of pivot operations to get the solution  $\mathbf{b}$  = column  $[0, 1, 4]$ . Finally an eigenvector corresponding to the eigenvalue  $-2$  is column  $[1, 1, 1]$ . Hence a fundamental matrix of solutions is

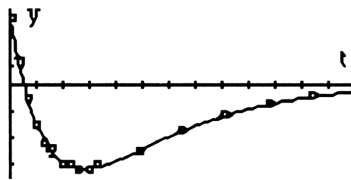
$$Y(t) = \begin{bmatrix} e^{-t} & e^{-t}t & e^{-2t} \\ 2e^{-t} & e^{-t}(2t+1) & e^{-2t} \\ 4e^{-t} & e^{-t}(4t+4) & e^{-2t} \end{bmatrix}.$$

- 8.6c Again the eigenvalues are -1 (repeated) and -2, and there is only one eigenvector corresponding to the eigenvalue -1, viz.,  $\mathbf{a} = \text{column } [-.5, 1, .5]$ . We try as a second solution  $y = e^{-t}(\mathbf{a}t + \mathbf{b})$  and proceed as in 4.3b.
- 8.7 The eigenvalues are 5.964, -1.529 and  $-3.218 \pm i 2.703$  with corresponding eigenvectors column  $[-.2866, .4203, 1, -.04738]$ , column  $[1, -.0063, -.1994, -.5790]$  and column  $[-.1846 \pm (-.5511i), .0255 \pm .0844i, -.5882 \pm i.0899, 1]$ .
- 9.10 The list **FL** should be { '(Y - A\*EXP(-P\*T)-B\*EXP(-Q\*T))\*A\*T\*EXP(-P\*T)'  
'(Y - A\*EXP(-P\*T)-B\*EXP(-Q\*T))\*B\*T\*EXP(-Q\*T)'  
'(Y - A\*EXP(-P\*T)-B\*EXP(-Q\*T))\*EXP(-P\*T)'  
'(Y - A\*EXP(-P\*T)-B\*EXP(-Q\*T))\*EXP(-P\*T)'}.

the parameter list **PL** should be {P Q A B},  $M = 4$ . The program **NST1** should be modified to << **DUP OBJ**→ **DROP 'B' STO 'A' STO 'Q' STO 'P' STO JACM FACM FVEC JMAT / >>. After several iterations, we obtain  $P = 2.07$ ,  $Q = .93$ ,  $A = 6.13$ ,  $B = -5.13$ .**

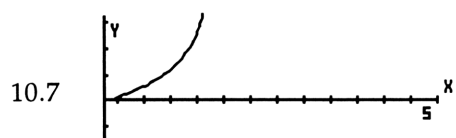
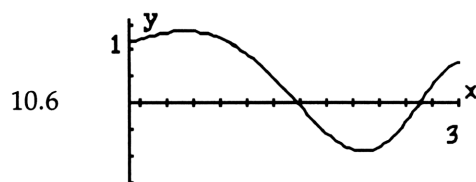
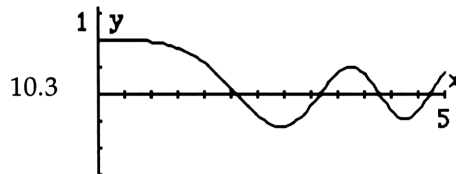
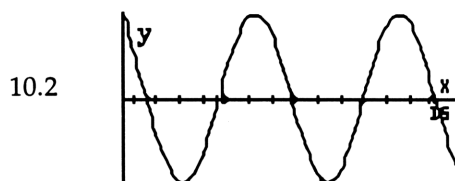
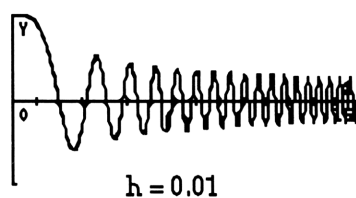
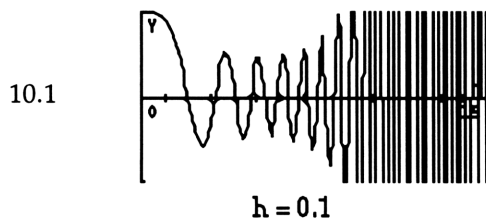


Fit using starting parameters

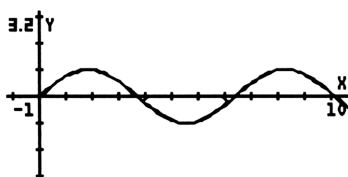
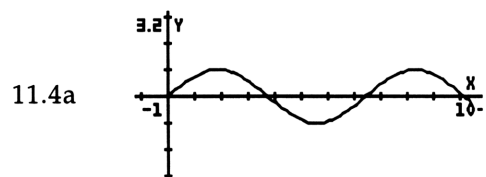


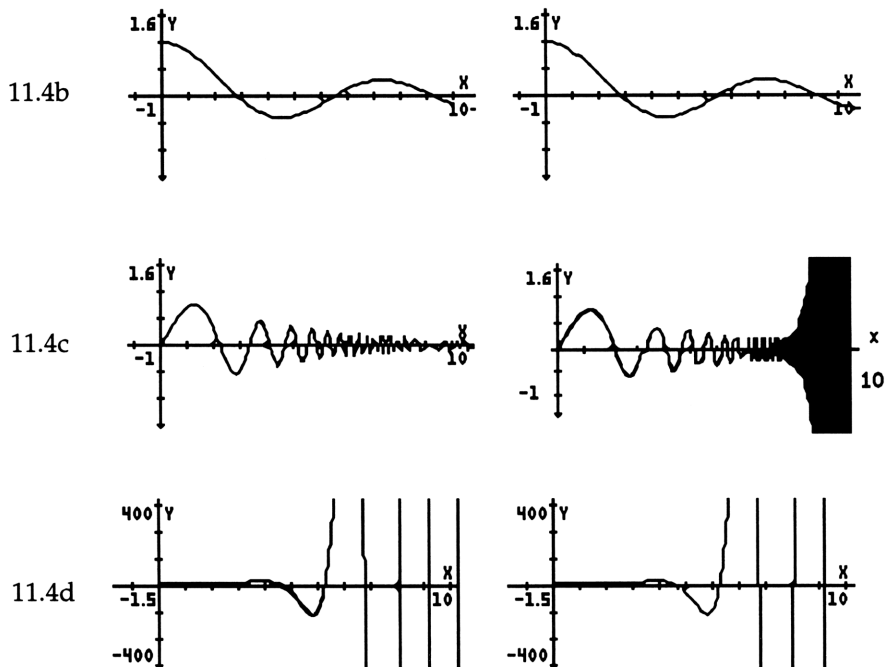
Fit using final parameters

## PART III EXERCISES



11.1 The first several zeros of  $J_1(x)$  are 3.831705970, 7.01558667, 10.17345681, the first several zeros of  $J_2(x)$  are 5.135622302, 8.4172441, the second intersection of  $J_1(x)$  and  $J_0(x)$  occurs at (4.680102554, -0.274841815) and the first intersection of  $J_1(x)$  and  $J_2(x)$  occurs at (2.629874112, 0.462386403).





11.6 The first three values of  $\lambda$  which give roots of  $J_{1/3}(z)$  with  $z = 2^{5/2} \sqrt{\lambda} / 3$  are 2.369533199, 10.23582292 and 23.6526167.

12.1  $\lambda_0 = 2.369533199$ ,  $\lambda_1 = 10.23582292$ ,  $\lambda_2 = 23.6526167$

12.4 The integral (a) is 2, the integral (b) is 2.36824634629

13.3 The curvature at  $t = 1/2$  is 0.443376024, the torsion is 0.21660881309. The length of the curve  $0 \leq t \leq 1$  is 2.64403204553.

13.4 The integral over  $F(x, y, z) = [-x, y, xy + z^2]$  on the curve of Example 13.3 is 4,072,716.93773; the integrals of the  $F$  functions of this exercise are 1.88498 and 1.454545 respectively.

13.5 Replace L3IN with  $P(X(T), Y(T)) * \partial T(X(T)) + Q(X(T), Y(T)) * \partial T(Y(T))$  in LLL program.

13.7 The HP-48G integration program gives 4.18879020478, Gaussian quadrature gives 4.18928084371.



## PART IV EXERCISES

## Activity Set 14.2

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

$$(b) \quad C = \begin{bmatrix} 3 & 2 & -1 \\ 5 & 0 & -6 \\ 7 & 3 & 0 \\ 1 & 2 & -4 \\ 8 & 7 & 9 \end{bmatrix}$$

$$(c) \quad D = \begin{bmatrix} 3 & 2 & -1 \\ -3 & 0 & -6 \\ 7 & 3 & 0 \\ 1 & 2 & -4 \\ 0 & 7 & 1.375 \end{bmatrix}$$

$$(d) \quad E = \begin{bmatrix} 3 & 2 & -1 \\ 7 & 3 & 0 \\ 0 & 7 & 1.375 \end{bmatrix}$$

$$2. (a) \quad A = \begin{bmatrix} .11 & .12 & .13 & .14 \\ .21 & .22 & .23 & .24 \\ .31 & .32 & .33 & .34 \\ .41 & .42 & .43 & .44 \\ .51 & .52 & .53 & .54 \end{bmatrix}$$

$$(b) \quad B = \begin{bmatrix} .21 & .22 & .23 & .24 \\ .31 & .32 & .33 & .34 \\ .51 & .52 & .53 & .54 \end{bmatrix}$$

$$(c) \quad C = \begin{bmatrix} .21 & .22 & .23 \\ .24 & .31 & .32 \\ .33 & .34 & .51 \\ .52 & .53 & .54 \end{bmatrix}$$

$$(d) \quad \begin{bmatrix} .62 & .63 & .64 \\ .21 & .22 & .23 \\ .33 & .34 & .51 \\ .52 & .53 & .54 \end{bmatrix}$$

$$3. (a) \quad \begin{bmatrix} 1 & 1 & 1 & 5 \\ 2 & 2 & 2 & 5 \\ 3 & 3 & 3 & 5 \\ 4 & 4 & 4 & 5 \end{bmatrix} \quad (b) \quad \begin{bmatrix} 1 & 1 & 1 & 5 \\ 2 & 2 & 2 & 5 \\ 3 & 3 & 3 & 5 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

(c) No; notice the (4,4)-entries

$$4. (a) \quad \begin{array}{ll} \text{level 2:} & A \\ \text{level 1:} & B \end{array} \quad 4 \text{ COL+ returns } [A \ B]$$

(b) Approaches vary

$$\begin{array}{ll} & \text{and} \\ \text{level 2:} & A \\ \text{level 1:} & B \end{array} \quad 4 \text{ ROW returns } \begin{bmatrix} A \\ B \end{bmatrix}$$

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

$$\begin{aligned}
 6. \quad A &= \begin{bmatrix} 0 & -2 & 4 \\ -2 & -3 & 3 \\ 4 & 3 & 6 \end{bmatrix} \quad B = \begin{bmatrix} -5 & 0 & 0 \\ -8 & 8 & 2 \\ 0 & -2 & -6 \end{bmatrix} \quad C = \begin{bmatrix} -0 & -2 & 4 \\ -5 & 0 & 0 \\ -8 & 8 & 2 \\ 0 & -2 & -6 \\ -2 & -3 & 3 \\ 4 & 3 & 6 \end{bmatrix} \\
 D &= \begin{bmatrix} -5 & 0 & 0 \\ 0 & -2 & -6 \\ -2 & -3 & 3 \\ 4 & 3 & 6 \end{bmatrix} \quad E = \begin{bmatrix} -5 & 0 & 0 & 0 \\ -2 & -6 & -2 & -3 \\ 3 & 4 & 3 & 6 \end{bmatrix}
 \end{aligned}$$

**Activity Set 14.3**

$$\begin{aligned}
 1. \quad A &= \begin{bmatrix} 7 & -9 \\ -8 & -5 \\ 8 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 5 & 3 \\ -3 & -7 & 7 \\ 1 & -2 & -5 \\ -2 & 9 & -7 \end{bmatrix} \quad \text{and} \quad BA = \begin{bmatrix} -16 & -28 \\ 91 & 55 \\ -17 & 6 \\ -142 & -20 \end{bmatrix} \\
 2. (a) \quad A &= \begin{bmatrix} 0 & -1 & -2 & -3 \\ 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 \\ 3 & 2 & 1 & 0 \\ 4 & 3 & 2 & 1 \end{bmatrix} \quad (b) \quad B = \begin{bmatrix} 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 \\ 4 & 3 & 2 & 1 \end{bmatrix} \quad (c) \quad C = \begin{bmatrix} 1 & 0 & -2 \\ 2 & 1 & -1 \\ 4 & 3 & 1 \end{bmatrix} \\
 (d) \quad C^2 &= \begin{bmatrix} -7 & -6 & -4 \\ 0 & -2 & -6 \\ 14 & 6 & -10 \end{bmatrix} \quad C^3 = \begin{bmatrix} -35 & -18 & 16 \\ -28 & -20 & -4 \\ -14 & -24 & -44 \end{bmatrix} \\
 3. \quad A &= \begin{bmatrix} 1 & 0 & -2 & 3 \\ 2 & -3 & 0 & -1 \\ 5 & -2 & 4 & 1 \end{bmatrix} \\
 (a) \quad B &= \begin{bmatrix} 0 & -3 & -2 \\ 2 & 0 & 4 \end{bmatrix}, \quad BA = \begin{bmatrix} -16 & 13 & -8 & 1 \\ 18 & -8 & 20 & -2 \end{bmatrix} \\
 (b) \quad C &= \begin{bmatrix} 1 & 3 \\ 2 & -1 \\ 5 & 1 \end{bmatrix} \quad \text{and} \quad CB = \begin{bmatrix} -6 & -3 & 10 \\ 2 & -6 & -8 \\ -2 & -15 & -6 \end{bmatrix} \\
 4. (a) \quad \text{For example, seed with 4 and generate the } 3 \times 4 \text{ matrix} \\
 A &= \begin{bmatrix} -8 & 8 & 9 & -4 \\ -9 & 0 & 1 & 4 \\ -4 & 2 & 8 & 6 \end{bmatrix}.
 \end{aligned}$$

$$\text{Then } A^T A = \begin{bmatrix} 161 & -72 & -113 & -28 \\ -72 & 68 & 88 & -20 \\ -113 & 88 & 146 & 16 \\ -28 & -20 & 16 & 68 \end{bmatrix}$$

(b) For the  $4 \times 5$  matrix

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

$$\text{we have } B^T B = \begin{bmatrix} 120 & 94 & -82 & -42 & 4 \\ 94 & 87 & -68 & -23 & -15 \\ -82 & -68 & 133 & 26 & 28 \\ -42 & -23 & 26 & 49 & -44 \\ 4 & -15 & 28 & -44 & 65 \end{bmatrix}$$

and for the  $5 \times 6$  matrix

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

$$\text{we have } C^T C = \begin{bmatrix} 115 & -84 & -46 & 22 & 3 & -44 \\ -84 & 123 & 18 & 59 & 34 & 44 \\ -46 & 18 & 118 & -69 & 1 & 76 \\ 22 & 59 & -69 & 202 & 15 & -90 \\ 3 & 34 & 1 & 15 & 62 & -10 \\ -44 & 44 & 76 & -90 & -10 & 169 \end{bmatrix}$$

(c) Conjecture: For any matrix  $A$ ,  $A^T A$  is symmetric.

(d) Proof:  $(A^T A)^T = A^T (A^T)^T = A^T A$ .

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

$$(b) \quad A + iB = \begin{bmatrix} (7, 9) & (-9, -7) & (-8, 5) & (-5, 0) \\ (8, 3) & (-1, 1) & (6, 3) & (5, 6) \\ (3, -3) & (-3, 5) & (-7, -6) & (7, -9) \\ (1, 4) & (-2, -1) & (-5, -3) & (-2, 5) \end{bmatrix}$$

$$(A + iB)^T = \begin{bmatrix} (7, 9) & (8, 3) & (3, -3) & (1, 4) \\ (-9, -7) & (-1, 1) & (-3, 5) & (-2, -1) \\ (-8, 5) & (6, 3) & (-7, -6) & (-5, -3) \\ (-5, 0) & (5, 6) & (7, -9) & (-2, 5) \end{bmatrix}$$

(c) After separating, SWAPing and recombining the new matrix is

$$\begin{bmatrix} (9, 7) & (-7, -9) & (5, -8) & (0, -5) \\ (3, 8) & (1, -1) & (3, 0) & (6, 5) \\ (-3, 3) & (5, -3) & (-6, -7) & (-9, 7) \\ (4, 1) & (-1, -2) & (-3, -5) & (5, -2) \end{bmatrix}$$

6. (a)  $C = \begin{bmatrix} 3 & 5 & 4 \\ 7 & 1 & 6 \end{bmatrix}$

(b)  $D = \begin{bmatrix} 1 & -2 & 3 & 5 & 4 & 3 & 7 \\ 7 & 9 & 0 & -1 & 3 & 5 & 1 \\ -3 & 8 & 6 & 2 & 1 & 4 & 6 \end{bmatrix}$   $E = \begin{bmatrix} 1 & 3 & 4 & 7 \\ 7 & 0 & 3 & 1 \\ -3 & 6 & 1 & 6 \end{bmatrix}$

7. (a)  $A^* = \begin{bmatrix} (5, -1) & (0, -4) \\ (2, 3) & (6, 1) \\ (1, 0) & (3, -4) \end{bmatrix}$   $B^T = \begin{bmatrix} (-3, 1) & (0, 0) & (4, -3) \\ (6, 0) & (2, -1) & (1, 1) \end{bmatrix}$

(b)  $A^* + B = \begin{bmatrix} (2, 0) & (6, -4) \\ (2, 3) & (8, 0) \\ (5, -3) & (4, -3) \end{bmatrix}$   $A + B^T = \begin{bmatrix} (2, 2) & (2, -3) & (5, -3) \\ (6, 4) & (8, -2) & (4, 5) \end{bmatrix}$

$AA^* = \begin{bmatrix} (40, 0) & (22, -40) \\ (22, 40) & (78, 0) \end{bmatrix}$   $B^TB = \begin{bmatrix} (15, -3) & (-11, 7) \\ (-11, 7) & (39, -2) \end{bmatrix}$

$(2-3i)A = \begin{bmatrix} (13, -13) & (-5, -12) & (2, -3) \\ (12, 8) & (9, -20) & (18, -1) \end{bmatrix}$

8.  $A^2 - 4A^* + 3A^T - I = \begin{bmatrix} (12, -10) & (19, 13) & (5, -2) \\ (0, 17) & (41, 24) & (17, 33) \\ (2, 17) & (9, 24) & (29, 13) \end{bmatrix}$

9. (a)  $A^4 - 8A^3 + 22A^2 - 40A + 25I = 0$

(b)  $A [A^3 - 8A^2 + 22A - 40I] = -25I$  so  $A^{-1} = \frac{-1}{25} [A^3 - 8A^2 + 22A - 40I]$

$$\begin{bmatrix} -4.68 & -3.28 & -4.48 & -2 \\ .56 & .76 & .16 & 0 \end{bmatrix}$$

$$(c) A^{-1} = \begin{bmatrix} 4.32 & 2.72 & 4.52 & 2 \\ 2.56 & 1.76 & 2.16 & 1 \end{bmatrix}$$

$$10. (a) \text{ To three decimal places, } \lim_{n \rightarrow \infty} \{A^n\} = \begin{bmatrix} .269 & .269 & .269 & .269 \\ .212 & .212 & .212 & .212 \\ .205 & .205 & .205 & .205 \\ .315 & .315 & .315 & .315 \end{bmatrix}$$

$$(b) \text{ To three decimal places, } \lim_{n \rightarrow \infty} \{A^n x\} = \begin{bmatrix} .269 & .212 & .205 & .315 \end{bmatrix}$$

$$(c) \text{ Each column in } \lim_{n \rightarrow \infty} \{A^n\} \text{ is } \lim_{n \rightarrow \infty} \{A^n x\}.$$

11. The conclusion is the same as in 10(c).

$$12. u = [4 \ -2 \ 5 \ -8 \ 5] \text{ and } v = [-4 \ 7 \ 8 \ 0 \ -6]$$

$$(a) u \bullet v = -20 \quad (b) u \bullet (u + v) = 114 \quad (c) v \bullet (v - u) = 185$$

$$(d) \frac{u - v}{\|u - v\|} = 1 \quad (e) 16.093476934 < 24.4210694815 \quad (f) 20 < 148.69431731$$

### Activity Set 14.4

$$1. (a) \det[1000A] = -1,536,000$$

(b) Using cofactors along column 1,

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

and since each of the three matrices on the right has two identical rows,  $\det A = 0$ . Thus  $\det[1000A] = 0$

$$\begin{aligned} (c) \text{ When } \det A \text{ is calculated as } -1.536 \times 10^{-12} \text{ due to round-off error,} \\ \det[1000A] &= \det[10^3 A] = (10^3)^6 \det A = 10^{18}(-1.536 \times 10^{-12}) \\ &= -1.536 \times 10^6 = -1,536,000. \end{aligned}$$

(d) With flag -54 clear,  $\det A$  is calculated to be 0.

### Activity Set 14.5

1. The RPN program is «  $\pi \rightarrow \text{NUM} * \text{COS} \gg$ . The result is

$$\begin{bmatrix} [-1 & 1 & -1] \\ [1 & -1 & 1] \\ [-1 & 1 & -1] \end{bmatrix}$$

2. The RPN program is «  $\sqrt{2} + \text{FLOOR} \gg$ . The result is

$$\begin{bmatrix} [3 & 3 & 3] \\ [4 & 4 & 4] \\ [4 & 4 & 5] \end{bmatrix}$$

3. The RPN program is «  $\text{DUP } 1 + \text{SWAP} / \sqrt{\text{CEIL}} \gg$ . The result is

$$\begin{bmatrix} [2 & 1 & 2] \\ [1 & 2 & 1] \\ [2 & 1 & 2] \end{bmatrix}$$

### Activity Set 15.1

1. (a)
- $$\begin{array}{l} \begin{bmatrix} [4 & 1 & 3 & 6] \\ [8 & -2 & 4 & -8] \\ [8 & -6 & -2 & -36] \end{bmatrix} \xrightarrow{\substack{1, 2 \text{ RSWP} \\ R_{21}(-.5) \\ R_{31}(-1)}} \begin{bmatrix} [8 & -2 & 4 & -8] \\ [0 & 2 & 1 & 10] \\ [0 & -4 & -6 & -28] \end{bmatrix} \\ \begin{array}{l} 2, 3 \text{ RSWP} \\ \rightarrow \end{array} \begin{bmatrix} [8 & -2 & 4 & -8] \\ [0 & -4 & -6 & -28] \end{bmatrix} \xrightarrow{R_{32}(.5)} \begin{bmatrix} [8 & -2 & 4 & -8] \\ [0 & -4 & -6 & -28] \end{bmatrix} = [U | b'] \\ \begin{array}{l} \\ \text{BACK} \rightarrow \end{array} \begin{bmatrix} [0 & 2 & 1 & 10] \\ [0 & 0 & -2 & -4] \end{bmatrix} \\ \begin{bmatrix} [0 & 2 & 1 & 10] \\ [0 & 0 & -2 & -4] \end{bmatrix} \end{array}$$
- (b)
- $$\begin{array}{l} \begin{bmatrix} [3 & 2 & -2 & 2 & -5] \\ [6 & 2 & -1 & -2 & -10] \\ [-3 & 1 & 2.5 & 0 & 8] \\ [6 & 0 & 2 & 4 & 2] \end{bmatrix} \xrightarrow{\substack{1, 2 \\ \text{RSWP}}} \begin{bmatrix} [6 & 2 & -1 & -2 & -10] \\ [3 & 2 & -2 & 2 & -5] \\ [-3 & 1 & 2.5 & 0 & 8] \\ [6 & 0 & 2 & 4 & 2] \end{bmatrix} \end{array}$$

$$\begin{array}{ll}
R_{21}(-.5) & \left[ \begin{array}{ccccc} 6 & 2 & -1 & -2 & -10 \end{array} \right] \quad 2, 3 \quad \left[ \begin{array}{ccccc} 6 & 2 & -1 & -2 & -10 \end{array} \right] \\
\rightarrow & \left[ \begin{array}{ccccc} 0 & 1 & -1.5 & 3 & 0 \end{array} \right] \text{ RSWP } \rightarrow \left[ \begin{array}{ccccc} 0 & 2 & -2 & 2 & -5 \end{array} \right] \\
R_{31}(.5) & \left[ \begin{array}{ccccc} 0 & 2 & 2 & -1 & 3 \end{array} \right] \quad R_{32}(-.5) & \left[ \begin{array}{ccccc} 0 & 0 & -2.5 & 3.5 & -1.5 \end{array} \right] \\
R_{41}(-1) & \left[ \begin{array}{ccccc} 0 & -2 & 3 & 6 & 12 \end{array} \right] \quad R_{42}(1) & \left[ \begin{array}{ccccc} 0 & 0 & 5 & 5 & 15 \end{array} \right] \\
3, 4 & \left[ \begin{array}{ccccc} 6 & 2 & -1 & -2 & -10 \end{array} \right] \quad \rightarrow \\
\text{RSWP} & \left[ \begin{array}{ccccc} 0 & 2 & 2 & -1 & 3 \end{array} \right] = [U | b'] \quad \text{BACK} & \left[ \begin{array}{cccc} -1 & 0 & 2 & 1 \end{array} \right] \\
\rightarrow & \left[ \begin{array}{ccccc} 0 & 0 & 5 & 5 & 15 \end{array} \right] \\
R_{43}(.5) & \left[ \begin{array}{ccccc} 0 & 0 & 0 & 6 & 6 \end{array} \right]
\end{array}$$

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:

$$\begin{array}{ll}
1, 3 \text{ RSWP} & \left[ \begin{array}{ccccc} 3 & 6 & 9 & 2 & -5 \end{array} \right] \\
1, 1 \text{ ELIM} & \rightarrow \left[ \begin{array}{ccccc} 0 & 1 & 1 & 4.\bar{3} & 6.\bar{6} \end{array} \right] \\
3, 4 \text{ ELIM} & \left[ \begin{array}{ccccc} 0 & 0 & 0 & 3.\bar{3} & 6.\bar{6} \end{array} \right] \\
& \left[ \begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \end{array} \right]
\end{array}$$

Thus  $x_3$  is a free variable  $x_4 = 2$ . Also

$$\begin{aligned}
x_2 &= 6.\bar{6} - x_3 - 4\bar{3}x_4 \\
&= 6.6 - x_3 - 8.\bar{6} \\
&= -2 - x_3.
\end{aligned}$$

$$\text{And } 3x_1 = -5 - 6x_2 - 9x_3 - 2x_4$$

$$= -5 - 6(-2 - x_3) - 9x_3 - 2(2)$$

$$= -5 + 12 + 6x_3 - 9x_3 - 4 = 3 - 3x_3, \text{ so } x_1 = 1 - x_3.$$

(b) 1, 3 RSWP

$$1, 1 \text{ ELIM} \quad \left[ \begin{array}{cccccc} 6 & -9 & 0 & 11 & -19 & 3 & 0 \end{array} \right]$$

$$2, 3 \text{ RSWP} \quad \left[ \begin{array}{cccccc} 0 & 0 & 5 & 1.\bar{6} & -3.\bar{3} & -8 & 6 \end{array} \right]$$

$$2, 3 \text{ ELIM} \rightarrow \left[ \begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 4 & 2 \end{array} \right]$$

$$\begin{array}{l} 3, 4 \text{ RSWP} \quad [ \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ ] \\ 3, 6 \text{ ELIM} \end{array}$$

Then  $x_2$ ,  $x_4$  and  $x_5$  are free variables and  $x_6 = .5$ .

$$\text{Also } 5x_3 = 6 - 1.\bar{6}x_4 + 3\bar{3}x_5 + 8x_6$$

$$= 6 - 1.\bar{6}x_4 + 3\bar{3}x_5 + 4$$

$$= 10 - 1.\bar{6}x_4 + 3\bar{3}x_5$$

$$\text{so } x_3 = 2 - .\bar{3}x_4 + \bar{6}x_5.$$

$$\text{Finally, } 6x_1 = 9x_2 - 11x_4 + 19x_5 - 3x_6$$

$$= 9x_2 - 11x_4 + 19x_5 - 3/2$$

$$\text{so that } x_1 = 1.5x_2 - 1.8\bar{3}x_4 + 3.1\bar{6}x_5 - .25.$$

## Activity Set 15.2

$$1. (a) \quad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ .5 & .5 & 1 \end{bmatrix} \quad U = \begin{bmatrix} -2 & 4 & 5 \\ 0 & 3 & 8 \\ 0 & 0 & -4 \end{bmatrix}$$

$$y = [ \ 10 \ 21 \ -12 \ ] \quad x = [ .5 \ -1 \ 3 \ ]$$

$$(b) \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ .\bar{3} & 1 & 0 \\ .5 & .5 & 1 \end{bmatrix} \quad U = \begin{bmatrix} -6 & -6 & -3 \\ 0 & 8 & 6 \\ 0 & 0 & -2 \end{bmatrix}$$

$$y = [ \ 18 \ 20 \ -4 \ ] \quad x = [ -5 \ 1 \ 2 \ ]$$

$$(c) \quad P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ .\bar{3} & 1 & 0 & 0 \\ 0 & -.5 & 1 & 0 \\ -. \bar{3} & .5 & .5 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} -9 & 6 & -3 & 6 \\ 0 & 4 & -2 & 2 \\ 0 & 0 & 6 & -2 \\ 0 & 0 & 0 & 7 \end{bmatrix} \quad y = [ -24 \ -16 \ 36 \ -42 \ ]$$

$$x = [ \ -2 \ 1 \ 4 \ -6 \ ]$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \end{bmatrix}$$



$$(d) \quad P = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad L = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ \bar{.3} & \bar{.3} & \bar{.3} & 1 & 0 \\ \bar{.3} & 0 & 0 & .5 & 1 \end{bmatrix}$$

$$\begin{aligned} & \begin{bmatrix} -6 & -3 & -12 & -9 & 3 \\ 0 & 3 & -6 & 3 & 9 \\ 0 & 0 & 9 & -6 & -3 \\ 0 & 0 & 0 & 4 & -8 \\ 0 & 0 & 0 & 0 & 4 \end{bmatrix} \\ U = & \begin{bmatrix} 0 & 0 & 9 & -6 & -3 \\ 0 & 0 & 0 & 4 & -8 \\ 0 & 0 & 0 & 0 & 4 \end{bmatrix} \quad y = \begin{bmatrix} 36 & 30 & -3 & -28 & 12 \end{bmatrix} \\ & x = \begin{bmatrix} -4 & 2 & 0 & -1 & 3 \end{bmatrix} \end{aligned}$$

$$2. (a) \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad L = \begin{bmatrix} -9 & 0 & 0 \\ -1 & -1.\bar{6} & 0 \\ -4 & -.6 & -1.8 \end{bmatrix} \quad U = \begin{bmatrix} 1 & -\bar{.6} & 0 \\ 0 & 1 & -1.2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$y = \begin{bmatrix} -2.\bar{3} & 5.6 & -3 \end{bmatrix} \quad x = \begin{bmatrix} -1 & 2 & -3 \end{bmatrix}$$

$$(b) \quad P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad L = \begin{bmatrix} -7 & 0 & 0 & 0 \\ 2 & -4.42857142857 & 0 & 0 \\ 5 & -1.57142857143 & -9.8064516129 & 0 \\ 1 & -3.71428571429 & -1.45161290323 & -3.53289473684 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & -1.28571428571 & -.142857142857 & -.428571428571 \\ 0 & 1 & -.967741935484 & -.419354893871 \\ 0 & 0 & 1 & -.867105263158 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$y = \begin{bmatrix} -7.57142857143 & -.483870964452 & 3.96710526316 & -1 \end{bmatrix}$$

$$x = \begin{bmatrix} -5 & 2 & 3 & -1 \end{bmatrix}$$

$$(c) \quad P = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$L = \begin{bmatrix} 9 & 0 & 0 & 0 & 0 \\ -7 & -11.\bar{6} & 0 & 0 & 0 \\ 7 & 3.\bar{6} & -8.5\bar{3} & 0 & 0 \\ -1 & -\bar{.6} & -4.9\bar{3} & -15.3214285714 & 0 \\ 5 & 7.\bar{3} & 4.9\bar{3} & 8.46428571429 & -9.41258741259 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & -\bar{.6} & -.5 & -1 & .\bar{8} \\ 0 & 1 & .9\bar{3} & .428571428571 & .0\bar{6} \\ 0 & 0 & 1 & -1.33928571429 & -.0625 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & -4.97668997669 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$y = [1.\bar{8} \quad .238095238096 \quad 11.0267857143 \quad -6.62121212126 \quad .270430906347]$$

$$x = [-2.99577087667 \quad .800891530589 \quad 2.35624071312 \quad -6.48662704316 \quad .270430906347]$$

### Activity Set 15.3

1. The RREF's for the augmented matrices are:

$$\begin{array}{ll} \text{a. } \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 2 \end{bmatrix} & \text{b. } \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

2. The RREF's for the augmented matrices are:

$$\begin{array}{ll} \text{a. } \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 & 4 \end{bmatrix} & \text{b. } \begin{bmatrix} 1 & 0 & 0 & 0 & 6 \\ 0 & 1 & 0 & 0 & -5 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 3 \end{bmatrix} \end{array}$$

3. The RREF's for the augmented matrices are:

$$\begin{array}{ll} \text{a. } \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \text{b. } \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1.\bar{3} & .6 \\ 0 & 1 & 0 & -1.\bar{2} & 2.\bar{1} & .\bar{1} & .\bar{2} \\ 0 & 0 & 1 & .\bar{3} & -.\bar{6} & -1.\bar{3} & -1.\bar{3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

### Activity Set 16.1

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

$$\left[ \begin{array}{cccccc|c} | & | & | & | & | & | & \\ v_1 & v_2 & v_3 & u_1 & u_2 & u_3 & \\ | & | & | & | & | & | & \end{array} \right] = \begin{bmatrix} -2 & 5 & 4 & -3 & -9 & 0 \\ 7 & -6 & -8 & 16 & 0 & -5 \\ 6 & -6 & 3 & 3 & 4 & 27 \\ -5 & 4 & 9 & -15 & -3 & 14 \end{bmatrix}.$$

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  $\text{Span}[v_1, v_2, v_3]$ , and  $u_3 = v_1 - 2v_2 + 3v_3$ .

$$\begin{bmatrix} 4 & 9 & 6 & -5 & -4 \\ -3 & 6 & 5 & 7 & 0 \\ -2 & -9 & 0 & -5 & -16 \\ 3 & -5 & 2 & -13 & -19 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & -2 & -1 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & -1 & -3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Thus  $p(x) = -2r(x) + s(x) - t(x)$  and  $q(x) = -r(x) + 2s(x) - 3t(x)$ .

## Activity Set 16.2

1. (a) Since the rows of  $A$  are the columns of  $A^T$ , we consider  $A^T$ :

$$\text{The reduced echelon form of } A^T \text{ is } \begin{bmatrix} 1 & 0 & 0 & 1/3 \\ 0 & 1 & 0 & -1/3 \\ 0 & 0 & 1 & -2/3 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

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

$$-\frac{1}{3}(\text{row } 1) + \frac{1}{3}(\text{row } 2) + \frac{2}{3}(\text{row } 3) + (\text{row } 4) = 0.$$

- b. The reduced echelon form of  $A$  is
- $$\begin{bmatrix} 1 & 0 & 0 & -.5 \\ 0 & 1 & 0 & .375 \\ 0 & 0 & 1 & .625 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

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

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

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

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

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

$$\begin{bmatrix} 1 & 0 & 0 & -.5 \\ 0 & 1 & 0 & .5 \\ 0 & 0 & 1 & -.5 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

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 16.3

1. (a) The reduced row echelon form of  $A$  is

$$E = \begin{bmatrix} 1 & 0 & 0 & \bar{3} \\ 0 & 1 & 0 & -\bar{3} \\ 0 & 0 & 1 & -\bar{6} \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

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  $A$ :  $[-\bar{3} \ \bar{3} \ \bar{6} \ 1]$ .

A basis for the left null space of  $A$ :  $[\ .5 \ -.375 \ -.625 \ 1 ]$ .

- (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  $D$ :

$$\begin{bmatrix} .586776859504 & -8.59504132231 & -.256198347107 & 1 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} -1.38016528926 & 1.28925619835 & .884297520661 & 0 & 1 \end{bmatrix}.$$

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 16.4

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  $\text{Span } B = RS(A^T) = RS(C^T) = \text{Span } B'$ . Indeed,  $\text{Span } B = CS(A) = CS(B) = \text{Span } B'$ .

- (b) Consider the augmented matrix  $[A \mid w]$ . 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  $W = \text{Span } B$  and  $[w]_B = [-2 \ 1 \ 1]^T$ .

- (c) The reduced row echelon form of the block matrix  $[C \mid A]$  is

$$\begin{bmatrix} 1 & 0 & 0 & .4 & 1.2 & .4 \\ 0 & 1 & 1 & .2 & -.4 & .2 \\ 0 & 0 & 0 & -.4 & -.2 & .6 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \text{ Thus } P = \begin{bmatrix} .4 & 1.2 & .4 \\ .2 & -.4 & .2 \\ -.4 & -.2 & .6 \end{bmatrix}.$$

- (d)  $[w]_{B'} = P[w]_B = [.8 \ -.6 \ 1.2]^T$ . The reduced row echelon form of  $[C \mid w]$  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  $\text{Span } B = RS(A^T) = RS(C^T) = \text{Span } B'$ .

- (b) The augmented matrix  $[A \mid w]$  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  $W = \text{Span } B$  and  $[w]_B = [72 \ 79 \ -3 \ -61]^T$ .

- (c) The reduced row echelon form of the block matrix  $[C \mid A]$  is

$$\begin{bmatrix} [ & 1 & 0 & 0 & 0 & -136 & 57.5 & 70 & -89.5 & ] \\ [ & 0 & 1 & 0 & 0 & 49 & -20.5 & -25 & 32.5 & ] \\ [ & 0 & 0 & 1 & 0 & -36 & 15 & 19 & -24 & ] \\ [ & 0 & 0 & 0 & 1 & 12 & -5 & -6 & -8 & ] \\ [ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ] \end{bmatrix}$$

$$\text{Thus } P = \begin{bmatrix} [ & -136 & 57.5 & 70 & -89.5 & ] \\ [ & 49 & -20.5 & -25 & 32.5 & ] \\ [ & -36 & 15 & 19 & -24 & ] \\ [ & 12 & -5 & -6 & -8 & ] \end{bmatrix}.$$

- (d)  $[w]_{B'} = P[w]_B = [0 \ 1 \ 0 \ -1]^T$ . The reduced row echelon form of  $[C \mid w]$  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 17.1

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  $A^T A = \text{diag}[7 \ 6 \ 8 \ 21]$ .  
 (b) Let  $A$  have  $v_1, v_2$ , and  $v_3$  as its columns, left-to-right. Then  $A^T A = I_3$ .
5. (i) The null space of  $A$  is the zero vector, so no basis exists.

- (ii) The reduced row echelon form of  $A$  is
- $$\begin{bmatrix} 1 & 2 & 0 & 4 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

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

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

A basis for  $CS(A)$  is  $\{[1 \ 1 \ -2], [-1 \ 0 \ 0]\}$

The reduced row echelon form of  $A^T$  is

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

A basis for  $NS(A^T)$  is  $\{[0 \ 2 \ 1]\}$ .  $CS(A)$  is orthogonal to  $NS(A^T)$  because

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

(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

$$\begin{bmatrix} 1 & -2 & 0 & 4 & 0 \\ 0 & 0 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 2 & -4 \\ 1 & 0 \\ 0 & -3 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

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

$$\begin{bmatrix} 1 & 2 & -1 & -3 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 2 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

(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

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

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

$$\begin{bmatrix} 1 & 0 & 1 & 2 & 1 \\ 2 & 1 & 2 & 3 & 1 \\ -1 & -1 & 0 & -2 & -2 \\ 2 & 0 & 3 & 2 & 2 \end{bmatrix} * \begin{bmatrix} -9 \\ 3 \\ 4 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

## Activity Set 17.2

1. (a) Seed the random number generator with 1 (1 RDZ) to produce

$$\begin{bmatrix} 7 & -9 & -8 \\ -5 & 8 & -1 \\ 0 & 5 & 3 \\ -3 & -7 & 7 \end{bmatrix}.$$

(b)  $\text{proj}_v u = [3.3698630137 \quad -2.99543378995 \quad -1.87214611872 \quad 2.62100456621]$

(c)  $\text{proj}_w u = [4.68292682927 \quad .585365853659 \quad -1.75609756098 \quad -4.09756097561]$

2. (a) Seed the random number generator with 2 to produce

$$\begin{bmatrix} 4 & -2 & 5 \\ -8 & 5 & -4 \\ 7 & 8 & 0 \\ -6 & -4 & 0 \end{bmatrix}.$$

(b)  $q_1 = [ .311399577664 \quad -.622799155328 \quad .544949260912 \quad -.467099366496 ]$ .

$q_2 = [ -.273777516016 \quad .646186669899 \quad .655153182431 \quad -.279755191039 ]$ .

(c)  $\text{proj}_w x_3 = [2.34302222747 \quad -5.07599787749 \quad -.38417546136 \quad -.78485938328]$ .

3. (a) Seed the random number generator with 3 to produce

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

- (b) Rounding to six decimal places:

$q_1 = [ .106600 \quad .213201 \quad -.746203 \quad .533002 \quad .31980 ]$

$q_2 = [ -.408126 \quad -.486755 \quad 1.248090\text{E-}3 \quad -.173485 \quad .752598 ]$

$q_3 = [ -.399263 \quad -.225475 \quad .365122 \quad .789454 \quad -.180902 ]$

$$(c) \text{proj}_w x_3 = [1.576232 \quad -.782662 \quad -2.477077 \quad -9.772820 \quad 5.504552]$$

$$4. (a) \begin{bmatrix} .25 \\ -.5 \\ .25 \\ -.75 \\ .25 \end{bmatrix} \quad (b) \begin{bmatrix} .875 & .25 & -.125 & .375 & -.125 \\ .25 & .5 & .25 & -.75 & .25 \\ -.125 & .25 & .875 & .375 & -.125 \\ .375 & -.75 & .375 & -.125 & .375 \\ -.125 & .25 & -.125 & .375 & .875 \end{bmatrix} \quad (c) H \text{ is symmetric} \\ (d) H^{-1} = H$$

### Activity Set 17.3

$$1. (a) \text{ The RREF of } A \text{ is } \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\ (b) Q = \begin{bmatrix} .5 & .5 & -.5 \\ .5 & -.5 & .5 \\ .5 & -.5 & -.5 \\ .5 & .5 & .5 \end{bmatrix} \quad R = \begin{bmatrix} 2 & 3 & 2 \\ 0 & 5 & -2 \\ 0 & 0 & 4 \end{bmatrix} \\ 2. Q = \begin{bmatrix} .2 & -.4 & -.8 \\ .4 & .2 & .4 \\ .4 & -.8 & .4 \\ .8 & .4 & .2 \end{bmatrix} \quad R = \begin{bmatrix} 5 & -2 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & 2 \end{bmatrix}$$

### Activity Set 17.4

1. (a) The RREF of the augmented matrix is  $I_4$ .

$$(b) A^T A = \begin{bmatrix} 26 & -2 & 12 \\ -2 & 20 & -12 \\ 12 & -12 & 12 \end{bmatrix} \text{ and } A^T b = [44 \quad -15 \quad 29] \text{ and } x = [-1.6 \quad 3.83 \quad 7.916]$$

$$2. (a) \begin{array}{c|c|c|c|c|c} x & -2.2 & -1 & .5 & 1.5 & 3 \\ \hline y & .361 & 2.718 & 3.791 & 2.733 & 1.245 \end{array}$$

$$(b) P_3(x) = 3.56125138045 + .168633453671x - .476477181213x^2 + .0530615957649x^3$$

$$(c) P_4(x) = .381750 + .347647x - .808918x^2 + .008864x^3 + .048199x^4 \\ (\text{rounded to six decimal places})$$

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  
 $[c_0 \ c_1 \ c_2 \ c_3] = [-.1\bar{3} \ .810582010582 \ -.109126984127 \ .0231481482]$ ,
- (c) Applying the RREF command we obtain  
 $[-.1\bar{3} \ .81058201058 \ -.109126984126 \ .0231481481481]$ .  
 Using  $x = (A^T A)^{-1}(A^T b)$  we obtain  
 $[-.133333334713 \ .8105820101475 \ -.109126983096 \ .0231481483408]$ .

### Activity Set 18.1

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  $(A + B) = 6$

$$(c) \quad A = \begin{bmatrix} 5 & 0 & 3 & 1 \\ 3 & 6 & -3 & 5 \\ -6 & -9 & 4 & -1 \\ -3 & 5 & -5 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 3 & 9 & 0 & -3 \\ 6 & -3 & 5 & 0 \\ 1 & 3 & 4 & -4 \\ 0 & 5 & 3 & -9 \end{bmatrix}$$

$$\text{Trace } A = 15, \text{ Trace } B = -5 \text{ and Trace } (A + B) = 10$$

- (d) Trace  $(A + B) = \text{Trace } A + \text{Trace } B$

$$\text{Proof: Trace } (A + B) = \sum_{i=1}^n (a_{ii} + b_{ii}) = \sum_{i=1}^n a_{ii} + \sum_{i=1}^n b_{ii} = \text{Trace } A + \text{Trace } B$$

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)  $\text{Trace}(A B) = -28 = \text{Trace}(B A)$

(c)  $A = \begin{bmatrix} 0 & -8 & 3 & -2 & 1 \\ 4 & 0 & 8 & -2 & 3 \\ 0 & 2 & -7 & -6 & 3 \end{bmatrix}$  and  $B = \begin{bmatrix} -6 & 2 & -3 \\ -8 & 6 & 1 \\ -7 & -7 & 8 \\ 0 & -2 & 8 \\ 6 & 0 & 8 \end{bmatrix}$

$\text{Trace}(A B) = -73 = \text{Trace}(B A)$

(d)  $\text{Trace}(A B) = \text{Trace}(B A)$

Proof: Let  $A$  be  $m \times n$  and  $B$  be  $n \times m$

$$\text{Trace}(A B) = \sum_{i=1}^m (A B)_{ii} = \sum_{i=1}^m \left( \sum_{j=1}^n A_{ij} B_{ji} \right) \text{ and}$$

$$\text{Trace}(B A) = \sum_{j=1}^n (B A)_{jj} = \sum_{j=1}^n \left( \sum_{i=1}^m B_{ji} A_{ij} \right). \text{ These are clearly the same.}$$

3. (a) Seed the random number generator with 3 and generate

$$A = \begin{bmatrix} 1 & -5 & 2 \\ -1 & 2 & -7 \\ 4 & 1 & -7 \end{bmatrix}$$

$$\det(\lambda I - A) = \lambda^3 + 4\lambda^2 - 25\lambda - 150 = \det(\lambda I - A^T)$$

(b)  $A = \begin{bmatrix} 9 & 1 & 3 & 5 \\ -8 & -9 & 8 & 3 \\ 3 & 9 & -6 & -7 \\ -7 & -7 & -2 & -7 \end{bmatrix}$

$$\det(\lambda I - A) = \lambda^4 + 13\lambda^3 - 70\lambda^2 - 1009\lambda + 3240 = \det(\lambda I - A^T)$$

(c)  $A = \begin{bmatrix} -1 & 0 & -5 & -8 & 7 \\ 7 & -1 & -9 & 6 & 7 \\ 5 & 4 & 9 & 0 & -9 \\ 9 & -6 & -5 & -9 & 8 \\ 0 & 4 & 1 & 7 & -5 \end{bmatrix}$

$$\det(\lambda I - A) = \lambda^5 + 7\lambda^4 + 24\lambda^3 - 1021\lambda^2 + 6257\lambda + 15082 = \det(\lambda I - A^T)$$

(d) - (e)  $A$  and  $A^T$  have the same characteristic polynomial.

Proof:  $\det(\lambda I - A) = \det(\lambda I^T - A^T) = \det(\lambda I - A)^T = \det(\lambda I - A)$

4. Seed the random number generator with 4 and generate

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

- (a)  $\det A = -479$ ,  $\text{Trace } A = -21$ ,  $\det(\lambda I - A) = \lambda^3 + 21\lambda^2 + 163\lambda + 479$ .  
 $\det B = -684$ ,  $\text{Trace } B = 3$ ,  $\det(\lambda I - B) = \lambda^4 - 3\lambda^3 - 134\lambda^2 + 696\lambda - 684$ .  
 $\det C = 87,902$ ,  $\text{Trace } C = -16$ ,  
 $\det(\lambda I - C) = \lambda^5 + 16\lambda^4 + 5\lambda^3 + 468\lambda^2 - 681\lambda - 87,902$ .
- (b) For an  $n \times n$  matrix  $A$ , the constant term in  $\det(\lambda I - A)$  is  $(-1)^n \det A$ ; the coefficient of  $\lambda^{n-1}$  is  $-(\text{Trace } A)$ .
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  $p(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + \dots + c_1\lambda + c_0$  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  $p(\lambda) = \lambda^3 + 48\lambda^2 - 133\lambda + 33,528$ .

$$(b) \quad AB = \begin{bmatrix} 96 & 63 & 57 & -124 \\ -36 & -45 & -63 & 68 \\ -52 & 96 & 16 & -27 \\ 54 & 49 & 9 & -71 \end{bmatrix} \quad \text{and} \quad BA = \begin{bmatrix} -21 & 23 & -65 & -2 \\ 77 & -5 & 63 & 74 \\ 12 & -76 & 60 & 8 \\ -24 & -84 & 74 & -38 \end{bmatrix}$$

have the same  $p(\lambda) = \lambda^4 + 4\lambda^3 + 6,626\lambda^2 + 116,180\lambda + 1,523,808$ .

$$(c) \quad AB = \begin{bmatrix} -28 & 0 & -34 & 40 & -77 \\ -72 & 46 & -104 & -70 & -38 \\ -36 & 5 & 32 & 39 & -3 \\ -30 & 2 & 69 & 68 & -59 \\ -16 & 72 & -88 & -56 & 4 \end{bmatrix} \quad \text{and} \quad BA = \begin{bmatrix} 137 & -9 & 10 & 57 & 17 \\ -5 & 26 & 78 & 23 & 17 \\ -139 & -45 & -84 & -11 & -27 \\ -98 & -74 & -118 & -2 & -10 \\ 131 & -46 & -34 & -17 & 45 \end{bmatrix}$$

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  $p(A) = 0$ .

$$(c) \quad \text{For } A = \begin{bmatrix} -4 & -6 & -8 & 0 \\ 3 & 4 & -3 & -8 \\ -8 & -4 & 1 & 2 \\ 2 & 2 & 7 & 2 \end{bmatrix}$$

$p(x) = x^4 - 3x^3 - 70x^2 + 174x + 720$  and  $p(A) = 0$ .

$$(d) \quad \text{For } A = \begin{bmatrix} 6 & -5 & 0 & 3 & -2 \\ -2 & 6 & -4 & -1 & -2 \\ 0 & -9 & -4 & -5 & 4 \\ -4 & 1 & 0 & 4 & -3 \\ 5 & -9 & -6 & -2 & 0 \end{bmatrix}$$

$p(x) = x^5 - 12x^4 + 69x^3 - 24x^2 + 7x + 5288$  and  $p(A) = 0$ .

(e) Every  $n \times n$  matrix satisfies its characteristic polynomial. This is known as the Cayley-Hamilton Theorem.



## Activity Set 18.2

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 eigenvalues 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  $\text{trace } A = -4$ . the sum of the eigenvalues is  $-3.\bar{9}$  and the product of the eigenvalues is 150.

- (e) The sum of the  $\lambda$ -values is the trace and the product of the  $\lambda$ -values is the determinant.
4. (a) The eigenvalues of  $A$  are 10, 2 and  $\pm 2.82842712475$ . 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 eigenvalues of  $C$  are 3.73205080756, 4,  $\pm i$  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  $\pm i$ . The eigenspace associated with  $\lambda = 3$  is spanned by  $[-2 \ 4 \ -6 \ -1 \ 1 \ 0]$ .

**Activity Set 18.3**

$$1. (a) B = P^{-1}AP = \begin{bmatrix} -1 & 4 & 2 & 8 \\ 0 & 1 & 0 & -6 \\ -4 & 4 & 3 & 2 \\ -0 & 0 & 0 & 5 \end{bmatrix}$$

(b) CHAR returns  $[1 \ -8 \ 22 \ -40 \ 25]$  for  $A$  and  $B$ .

(c) The  $\lambda$ -values are 1, 5 and  $1 \pm 2i$ ; trace = 8, det = 25 and rank = 4.

2. (a)  $A$  is diagonalizable. Its eigenvalues are  $\lambda = -1, 1, 1$  and 2.

For  $D = \text{diag}[-1 \ 1 \ 1 \ 2]$ , 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  $\lambda = -2, -1, 1$  and 1, but  $\dim NS[A - I] = 1$ .

(c)  $A$  is diagonalizable. Its eigenvalues are  $\lambda = 2, 2, 4$  and 6. For

$D = \text{diag}[2 \ 2 \ 4 \ 6]$ , a diagonalizing matrix  $P$  is

$$P = \begin{bmatrix} 1 & 1 & -\sqrt{6} & -2 \\ 0 & 1 & \sqrt{3} & 1 \\ 1 & 0 & -1 & -1 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

(d)  $A$  is defective. Its eigenvalues are  $\lambda = -1, 0, 1, 1$  and 3.  $\dim NS[A - I] = 1$ .

(e)  $A$  is diagonalizable. Its eigenvalues are  $\lambda = 4, 4, 3, 2, 2$  and 1. For

$D = \text{diag}[4 \ 4 \ 3 \ 2 \ 2 \ 1]$  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 18.4

1. (a) The eigenvalues of  $A$  are  $\lambda = 1, 5, 5, 5$ . A basis for the eigenspace of  $\lambda = 1$  is  $X_1 = [1 \ 1 \ 1 \ 1]$ , normalized to  $Q_1 = [.5 \ .5 \ .5 \ .5]$ . A basis for the eigenspace of  $\lambda = 5$  consists of the vectors  $X_2 = [-1 \ 1 \ 0 \ 0]$ ,  $X_3 = [-1 \ 0 \ 1 \ 0]$  and  $X_4 = [-1 \ 0 \ 0 \ 1]$ . Convert to an orthonormal basis using the modified Gram-Schmidt algorithm via program GS to obtain

$$Q_2 = [-.707106781188 \ .707106781188 \ 0 \ 0]$$

$$Q_3 = [-.408248290463 \ -.408248290466 \ .816496580929 \ 0]$$

$$Q_4 = [-.288675134593 \ -.288675134595 \ -.288675134595 \ .866025403784]$$

Then with  $Q = [Q_1 \ Q_2 \ Q_3 \ Q_4]$  (in column form),  $Q^T A Q = \text{diag} [1 \ 5 \ 5 \ 5]$

- (b) The eigenvalues of  $A$  are  $\lambda = 1, 4, 4, 4$ . A basis for the eigenspace of  $\lambda = 1$  is  $X_1 = [-1 \ -1 \ 1 \ 0]$ , normalized to

$$Q_1 = [-.577350269189 \ -.577350269189 \ .577350269189 \ 0].$$

A basis for the eigenspace of  $\lambda = 4$  consists of the vectors

$X_2 = [-1 \ 1 \ 0 \ 0]$ ,  $X_3 = [1 \ 0 \ 1 \ 0]$  and  $X_4 = [0 \ 0 \ 0 \ 1]$ . Apply program GS to obtain

$$Q_2 = [-.707106781188 \ .707106781188 \ 0 \ 0]$$

$$Q_3 = [.408248290463 \ .408248290466 \ .816496580929 \ 0]$$

$$Q_4 = [0 \ 0 \ 0 \ 1]$$

Then with  $Q = [Q_1 \ Q_2 \ Q_3 \ Q_4]$  (in column form),  $Q^T A Q = \text{diag} [1 \ 4 \ 4 \ 4]$ .

- (c) The eigenvalues are  $\lambda = -6, 2, 4, 4$ . A basis for the eigenspace of  $\lambda = -6$  is  $X_1 = [-1 \ -1 \ -1 \ 1]$ , normalized to  $Q_1 = [-.5 \ -.5 \ -.5 \ .5]$ . A basis for the eigenspace of  $\lambda = 2$  is  $X_2 = [1 \ -1 \ 1 \ 1]$  normalized to  $Q_2 = [.5 \ -.5 \ .5 \ .5]$ . A basis for the eigenspace of  $\lambda = 4$  consists of  $X_3 = [-1 \ 0 \ 1 \ 0]$  and  $X_4 = [0 \ 1 \ 0 \ 1]$ , already orthogonal. Normalizing we obtain

$$Q_3 = [-.707106781188 \ 0 \ .707106781188 \ 0]$$

$$Q_4 = [0 \ .707106781188 \ 0 \ .707106781188]$$

Then with  $Q = [Q_1 \ Q_2 \ Q_3 \ Q_4]$  (in column form) we obtain

$$Q^T A Q = \text{diag} [-6 \ 2 \ 4 \ 4].$$

- (d) The eigenvalues are  $\lambda = -2, -2, 6, 6, 6$ . A basis for the eigenspace of  $\lambda = 2$  consists of the vectors  $X_1 = [0 \ 1 \ 1 \ 0 \ 0]$  and  $X_2 = [-2 \ 2 \ 0 \ 1 \ 1]$ . Apply program GS to obtain (we round to six decimal places)

$$Q_3 = [0 \ .707107 \ .707107 \ 0 \ 0]$$

$$Q_4 = [-.707107 \ .353553 \ -.353553 \ .353553 \ .353553]$$

A basis for the eigenspace of  $\lambda = 6$  consists of the vectors  $X_3 = [-1 \ -1 \ 1 \ 0 \ 0]$ ,  $X_4 = [1 \ 0 \ 0 \ 2 \ 0]$  and  $X_5 = [1 \ 0 \ 0 \ 0 \ 2]$ . Apply program GS to obtain

$$Q_3 = [-.577350 \ -.577350 \ .577350 \ 0 \ 0]$$

$$Q_4 = [.3086067 \ -.154303 \ .154303 \ .925820 \ 0]$$

$$Q_5 = [.267261 \ -.133630 \ .133630 \ -.133630 \ .935414]$$

Then with  $Q = [Q_1 \ Q_2 \ Q_3 \ Q_4 \ Q_5]$  (in column form)

$$Q^T A Q = \text{diag} [-2 \ -2 \ 6 \ 6 \ 6].$$

2. (a) A spectral decomposition for matrix  $A$  is

$$\begin{aligned} & \left[ \begin{bmatrix} .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \end{bmatrix} + 5 \begin{bmatrix} .5 & -.5 & 0 & 0 \\ -.5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 5 \begin{bmatrix} .1\bar{6} & .1\bar{6} & .\bar{3} & 0 \\ .1\bar{6} & .1\bar{6} & -. \bar{3} & 0 \\ -. \bar{3} & -. \bar{3} & .\bar{6} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right] \\ & + 5 \begin{bmatrix} .08\bar{3} & .08\bar{3} & .08\bar{3} & -.25 \\ .08\bar{3} & .08\bar{3} & .08\bar{3} & -.25 \\ -.08\bar{3} & .08\bar{3} & .08\bar{3} & -.25 \\ -.25 & -.25 & -.25 & .75 \end{bmatrix} \end{aligned}$$

- (b) A spectral decomposition for matrix  $A$  is

$$\begin{aligned} & \left[ \begin{bmatrix} .\bar{3} & .\bar{3} & -. \bar{3} & 0 \\ .\bar{3} & .\bar{3} & -. \bar{3} & 0 \\ -. \bar{3} & -. \bar{3} & .\bar{3} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 4 \begin{bmatrix} .5 & -.5 & 0 & 0 \\ -.5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 4 \begin{bmatrix} .1\bar{6} & .1\bar{6} & .\bar{3} & 0 \\ .1\bar{6} & .1\bar{6} & .\bar{3} & 0 \\ .\bar{3} & .\bar{3} & .\bar{6} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right] \\ & + 4 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

(c) A spectral decomposition for matrix  $A$  is

$$\begin{aligned}
 & -6 \begin{bmatrix} .25 & .25 & .25 & -.25 \\ .25 & .25 & .25 & -.25 \\ .25 & .25 & .25 & -.25 \\ -.25 & -.25 & -.25 & .25 \end{bmatrix} + 2 \begin{bmatrix} .25 & -.25 & .25 & .25 \\ -.25 & .25 & -.25 & -.25 \\ .25 & -.25 & .25 & -.25 \\ .25 & -.25 & .25 & -.25 \end{bmatrix} + 4 \begin{bmatrix} .5 & 0 & -.5 & 0 \\ 0 & 0 & 0 & 0 \\ -.5 & 0 & .5 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
 & + 4 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & .5 & 0 & .5 \\ 0 & 0 & 0 & 0 \\ 0 & .5 & 0 & .5 \end{bmatrix}
 \end{aligned}$$

(d) A spectral decomposition for matrix  $A$  (we round to six decimal digits) is

$$\begin{aligned}
 & -2 \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & .5 & .5 & 0 & 0 \\ 0 & .5 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} - 2 \begin{bmatrix} .5 & -.25 & .25 & -.25 & -.25 \\ -.25 & .125 & -.125 & .125 & .125 \\ .25 & -.125 & .125 & -.125 & -.125 \\ -.25 & .125 & -.125 & .125 & .125 \\ -.25 & .125 & -.125 & .125 & .125 \end{bmatrix} \\
 & + 6 \begin{bmatrix} \sqrt{3} & \sqrt{3} & -\sqrt{3} & 0 & 0 \\ \sqrt{3} & \sqrt{3} & -\sqrt{3} & 0 & 0 \\ -\sqrt{3} & -\sqrt{3} & \sqrt{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
 & + 6 \begin{bmatrix} .095238 & -.047619 & .047619 & .285714 & 0 \\ -.047619 & .023809 & -.238095 & -.142857 & 0 \\ .047619 & -.023809 & .023809 & .142857 & 0 \\ .285714 & -.142857 & .142857 & .857143 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
 & + 6 \begin{bmatrix} .071429 & -.035714 & .035714 & -.035714 & .25 \\ -.035714 & .017857 & -.017857 & .017857 & -.125 \\ .035714 & -.017857 & .017857 & -.017857 & .125 \\ -.035714 & .017857 & -.017857 & .017857 & -.125 \\ .25 & -.125 & .125 & -.125 & .875 \end{bmatrix}
 \end{aligned}$$

## Activity Set 18.5

- Not positive definite
  - Positive definite
  - Not positive definite
  - Positive definite

- (e) Not positive definite
- (f) Positive definite
2. (b) The solution to  $Ly = b$  is  $y = [ .707106781188 \quad 2.04124145232 \quad 2.30940107676 ]$ .  
The solution to  $L^T x = y$  is  $x = [ 2 \quad 1 \quad 2 ]$ .
- (d) The solution to  $Ly = b$  is  $y = [ .577350269189 \quad 1.80739222822 \quad 6.71317113328 \quad 17.8978583435 ]$ . The solution to  $L^T x = y$  is  $x = [ 37 \quad 29 \quad 50 \quad 31 ]$ .
- (f) The solution to  $Ly = b$  is  $y = [ .5 \quad 1.70084012854 \quad 2.27093435774 \quad 3.22711724525 \quad 1.13389341893 ]$ . The solution to  $L^T x = y$  is  $x = [ -3 \quad -2 \quad 4 \quad 0 \quad 3 ]$ .
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  $y = [ 1 \quad 2 \quad 3 \quad 4 \quad 5 ]$ . The solution to  $L^T x = y$  is  $x = [ 15 \quad 14 \quad 12 \quad 9 \quad 5 ]$ .

### Activity Set 18.6

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  $x = [ -1 \quad 0 \quad 2 ]$ .
3. Both methods produce  
 $x = [ -.276893387225 \quad .0697403149571 \quad -.187791269405 \quad .128765792031 ]$ .

4. Using a Cholesky factorization, we find the vector of coefficients in  $p(x)$  to be  $[-10.6666667605 \quad 12.7609428657 \quad -4.14534963083 \quad .389570640143]$ .

Using the LSQ command we obtain

$[-10.6666666667 \quad 12.7609427609 \quad -4.1453495999 \quad .389570637505]$ .

The norms of these vectors agree to 6 decimal places.

## Activity Set 19.1

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  $\geq 1$ .
- (c) STTEST shows that both norms are  $\geq 1$ .
- (d) Swap rows one and three, then rows two and five.
- (e) After 25 iterations,  $x = [-8.762155 \quad 6.560882 \quad -8.564386 \quad 6.754029 \quad 2.265541]$ .
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,  
 $x = [3.10325718 \quad -2.39543980 \quad 9.21205220 \quad -5.18175900 \quad 5.30456021]$ .
- (c) After 13 iterations,  
 $x = [3.10325731 \quad -2.39543974 \quad 9.21205212 \quad -5.18175896 \quad 5.30456026]$ .
- (d) To full machine precision, the solution is  
 $x = [3.10325732899 \quad -2.39543973941 \quad 9.21205211726 \quad -5.18175895765 \quad 5.30456026059]$
4. (a) The coefficient matrix is not diagonally dominant and the JTEST fails. The STTEST succeeds with row-sum norm  $< 1$ .
- (b) After 55 iterations,  
 $x = [3.66666598 \quad -2.33333231 \quad -1.00000102 \quad -8.66666590 \quad 8.33333295]$ .

5. (a) All of the tests for convergence fail.
- (b) After 16 iterations, the Gauss-Seidel procedure converges to  
 $x = [ -0.846168 \quad 3.153840 \quad -0.692309 \quad -0.076918 \quad -2.538454 \quad -5.538454 ]$ .
- (c) To full machine precision, the solution is  
 $x = [ -0.846153846154 \quad 3.15384615385 \quad -0.692307692308 \quad -0.076923076923 \quad -2.53846153846 ]$ .

## Activity Set 19.2

1. (a) The dominant eigenvalue is  $\lambda = 22.644832095$ .
- (b) After 31 iterations, the power method converges to  $\tilde{\lambda} = 22.64483210$   
 $\tilde{x} = [ .03978762 \quad .56098114 \quad -.33459603 \quad .53336803 \quad .51329075 \quad -.15431704 ]$ .
2. (b) After 154 iterations,  $\tilde{\lambda} = 2.94188363485$  and  $\tilde{x} = [ .55065580727 \quad -.51865369330 \quad .45650931190 \quad -.36783426864 \quad .25778203471 \quad -.13274844593 ]$ .
- (c) After 30 iterations,  $\tilde{\lambda} = 6.19999998113$  and  $\tilde{x} = [ .97979591798 \quad -.19595916510 \quad .03919176386 \quad -.0078378681 \quad .005156516522 \quad -.000300099331 ]$ .
3. (a) - (b) After 180 iterations we have no convergence because the components of successive vectors agree to within a  $\pm$  sign.



# INDEX

- Absolute Comparison Theorem 148
- Acoustical dynamics 302
- Antiderivative 139
- Approximations
  - Polynomial 89
  - Rectangle 105
  - Trapezoid 120
  - Midpoint 106
  - Simpson's 120, 126
- Arc length
  - Parametric 134
  - Symbolic integration 135
- AREA 123
- Artillery shell 273
- Asymptotic 201, 281
- Attracting solutions 240, 246, 250, 284,
- Autonomous 238, 244, 247
- Back substitution 432
- Basis 460
- Beats vibration 236
- Bessell equation 325, 326, 328
- Bessell function 327, 328, 331, 335-6
- Bessell functions 340
- Bessell function, program 328-9, 331-4
- Boundary value problems 348
- BOXZ 31
- $C \rightarrow R$  49
- Cancellation errors 47
- Catenary 88
- Cauchy-Schwartz 473
- CEIL 22
- Change of basis 466
- Characteristic equation 244
- Characteristic polynomial 495
- Characteristic polynomial 262
- Chebyshev Polynomial 86
- Cholesky factorization 516
- Column space 453, 461
- Column sum norm 535, 551
- Companion matrix 499
- Conjugate transpose 418
- Critical point (see equilibrium point)
- Crout algorithm 446
- Curve fitting 490
- Data fit 209, 213, 228, 230, 288, 187
- Determinant 422
- Diagonal matrix 399
- Diagonalizable matrix 505
- Diagonally dominant 534
- Dimension 460, 461
- Damped harmonic motion 76
- DEFINE 19
- Definite Integral 118
- Derivative
  - Definition 46
  - $\partial$  function 51
  - Partial 59
- Difference quotient 46
- Differential equation 142

## Differentiation

- Using  $\partial$  51
- Using the stack 51
- Using the Symbolic Differentiate Screen 53
- of XROOT 55
- Piecewise 56
- Implicit 57
- Parametric 100

Direction fields 179, 298-302

Directory (subdirectory) 190, 199

Discrete dynamical system 284ff

Dominant eigenvalue 542

Dot product 414

Double integrals 375ff

DRPN 10

Echelon matrix 434, 449

Eigenfunctions 244, 254ff, 349

Eigenspace 495

Eigenvalues 244, 244ff, 349, 495

Eigenvector 495

Elliptic integral 366, 367

Equation Writer 52

Equilibrium solution 200-203, 237, 243, 301-2

## Errors

- Cancellation 46
- Left rectangle 119
- Right rectangle 119
- Midpoint rectangle 119
- Simpson's 126
- Trapezoid 125
- Integration 130

EXTR 77

Euler algorithm 191-3, 197, 227, 296

Flag -54 425, 451, 524

FLOOR 22

Fitting curves to data 490

Forward Substitution 443, 444

Free variable 436

Frobenius norm 470

Fundamental solution matrix 260-1, 263-5

## Function

- One-to-one 36
- Evaluation with SOLVR 17
- Inverse 36
- User-defined 19
- Two or more variables 19
- Piecewise-defined 20

## Fundamental Theorem

- of Calculus 138, 139

Gamma function 327

Gaussian quadrature 362ff, 379-81

Gauss-Jordan reduction 449

Gauss-Seidel iteration 532

Gaussian elimination 429, 435

Gram-Schmidt process 478

Greatest integer function 22

Greens theorem 376, 378

Harmonic oscillator 233, 235

Henon map 287

Hermitian product 470

Hilbert matrix 402

Householder matrix 482

HP Solve System 83

Hyperbolic cosine 88

HZIN and HZOUT 30

IFTE 20

Ill-conditioned 492

Implicitly defined solution 204

Improper integrals 147, 167

Improved Euler alg. 192-94, 197, 226, 296

- Inflection points 64
- Infinite
  - Series 159
  - Sequence 153
- Inflection points 210
- Initialization program 184-5, 198
- Initial value problem 142
- Inner product 470
- Input/output problems 216, 230
- Input signal delay 217
- Integer part 44
- Integral 118
  - Improper 147, 167
  - Test 167
- INTEGRATE Form 130
- Integration
  - Symbolic 135
  - Numerical 129
  - Error 130
  - Using the stack 131
- Interactive stack 9, 72
- Inverse function 36
- IP 44
- Jacobi iteration 532
- Julia set 288
- LAPACK code 524
- LDLT-factorization 517
- LDU-factorization 517
- Least integer function 22
- Least squares 492, 528
- Left null space 461
- Legendre polynomials 86, 358ff
- Linear autonomous system 244
- Linear combination 453
- Linear dependence 455
- Linearly independent sols 318, 321
- Linear independence 455
- Line integrals 372-75
- l'Hospital's Rule 157, 158
- Lorentz 278
- Lotka-Volterra model 240
- Lower triangular matrix 400
- LU-factorization 439
- LIST 10
- Matrix
  - change of basis 467
  - companion 535
  - defective 507
  - diagonalizable 505
  - echelon 434
  - Hilbert 402
  - Householder 482
  - ill-conditioned 492
  - inverse 422
  - iteration 532
  - lower triangular 400
  - orthogonal 514
  - permutation 439
  - positive definite 514
  - powers 416
  - symmetric 510
  - tridiagonal 402
  - upper triangular 400
- MatrixWriter 394
- Maxima 65
- Mean Value Theorem 96
- Midpoint Rule 140
- Minima 65
- Mixing problem 205-6
- Newton's method 80, 250, 292

Nonhomogeneous 266ff

Norms

- column-sum 535, 551
- Euclidean 414, 550, 551
- Frobenius 551
- matrix 550
- row-sum 550
- spectral 552
- sum 551
- vector 550
- vector-max 550
- vector-sum 550

Normal equations 489

Normal vector 471

$n$ th roots 7

Null space 461

Orthogonal 354

Orthogonal complement 473

Orthogonal matrix 482

Orthogonal projection 476

Orthogonal subspaces 472

Orthogonal trajectories 238

Orthogonal vectors 471

Orthonormal basis 475

Overlay 186, 189, 202

Parametric curves 38

Slopes of 101

Parameters 187-8, 208, 284

Partial derivative 59

Partial pivoting 433

Partial sum 160

Particle motion 211-5, 219-24, 290

Pendulum 241, 269

Periodic 23, 233, 241-3, 283

Permutation matrix 439

Phase plane 240

PICK 9

Piecewise Plots 35

Pivot 432

Pivot variable 436

PLOT menu 25

PLOT screen 24

Plots

Superimposing 32

Disconnected 32

Sequential 33

Simultaneous 33

Connected 34

Piecewise 35

Parametric 38

Poincare section 234

Polynomial

Approximations 89

Legendre 86

Chebyshev 86

Best Linear Approximation 89

Best Quadratic Approx. 91

Taylor 91

Population growth problems 207-12

Positive definite matrix 513

Power method 542

Power series solution 325

Principal cube root 36

Projectile motion 273-6

Projection 476

PURGE 13

Pursuit problem 276

QR-factorization 483, 486

Radius of convergence 320

Random Matrix Generator 396

- Rank 464
- Ratio test 171
- Recurrence 321, 322, 324, 326
- Reduced row echelon matrix 449
- Regular point 320, 321
- Regular singular point 325
- Repelling solutions 238, 240, 284, 286
- Restricted three body problem 281
- Riemann Sums 115
- RKF program 345
- Rocket flight 273
- ROLL 10
- ROLLD 10
- Root-finder 82
- Row echelon matrix 436
- Row space 461
- Row sum norm 535, 552
- RPN 10
- RREF 449
- Runge Kutta 311, 314, 341-3, 382-3
- Runge Kutta algorithm 187, 194, , 251
- Schur factorization 502
- Second order IVP 183, 225
- Series
  - Alternating 163
  - Convergent 160
  - Alternating harmonic 165
- Series solution of IVP 316, 318
- SHADE 123
- Shooting method 350
- SIGN(X) 62
- Similar matrices 505
- Simpson's approximation 120, 126
- Singular value decomposition 520
- Singular values 522
- Slope 46
- Solution structure 202-3
- SOLVE command menu 85
- SOLVE EQUATION screen 84
- SOLVR 17
- Sonin-Polya Theorem 313
- Souriau-Frame method 496
- Space curves 369-372
- Span 453
- Spectral decomposition 512
- Spectral radius 534, 552
- Spiral point 244
- Spring /circuit model 233
- Square wave, Switch function 218-9
- Step size selection 194
- Stiff differential equation 251
- St. Louis arch 88
- Sturm Comparison Theorem 313
- Sturm Liouville problems 354, 355
- Superimposing plots 32
- Symbolic Differentiate Screen 53
- Symbolic Execution Mode 5
- Symmetric matrix 402, 510
- Tail
  - Improper Integral 147
  - Infinite series 168
- Taylor Polynomial Screen 92
- Taylor series solution 315, 320, 321
- TAYLR 92
- Trace 418, 504
- Tridiagonal matrix 401
- Terminal velocity 211-2
- Trajectories in three dimensions 305ff
- Trapezoid approximation 120
- Undetermined coefficients 267

Unit lower triangular matrix 400  
Upper triangular matrix 400  
User-defined functions 19  
Vander Pol 235,242  
Variational matrix 247  
Vector space 452  
VZIN and VZOUT 30  
XROOT 36  
ZAUTO 31  
ZDECI 30  
ZDFLT 30  
ZIN and ZOUT 30  
ZINTG 30  
ZLAST 30  
ZOOM menu 30  
ZSQR 30  
ZTRIG 30



















# HP-48G/GX

## INVESTIGATIONS IN MATHEMATICS

Donald LaTorre, Don Kreider, and Gil Proctor

Use **HP-48 G/GX** graphing calculator technology to aid in the computation and visualization of Calculus, Differential Equations, Linear Algebra and Vector Calculus.

### FEATURES

- ▶ Includes calculator investigations that reinforce key topics in Engineering Mathematics—Calculus, Linear Algebra, Differential Equations, and Vector Calculus;
- ▶ Includes a DOS diskette with a collection of special purpose **HP-48G/GX** calculator programs contained in the book;
- ▶ Provides "activity sets" and their solutions for many computational and applied exercises;
- ▶ "Getting-started" section offers students a friendly introduction to the keyboard, data entry, RPN, and memory management;
- ▶ Surveys the main topics included in the standard one or two semester courses in these subject areas;
- ▶ May be used as a supplement with any textbook.

### SYSTEM REQUIREMENTS

IBM or compatible • 386 or higher • DOS • 4MB RAM

CHARLES RIVER MEDIA, INC  
PO Box 417  
Rockland, MA 02370



All trademarks and service marks are the property of their respective owners.

Printed in the USA by InterCity Press, Rockland, MA