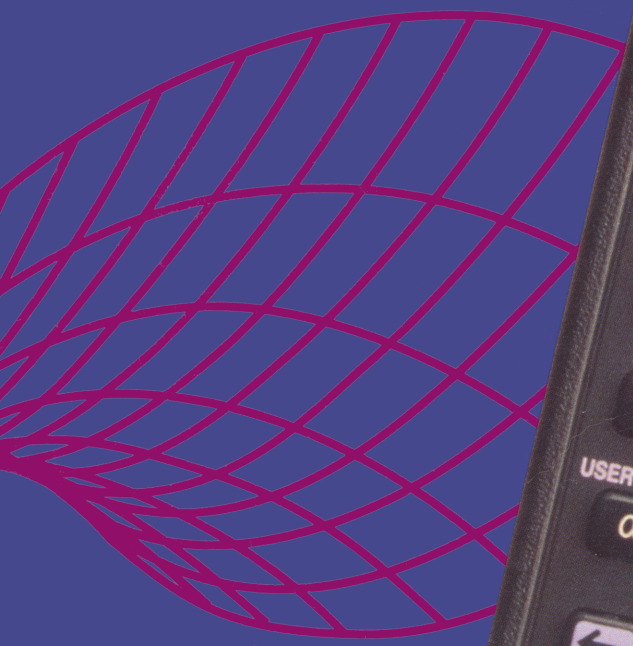
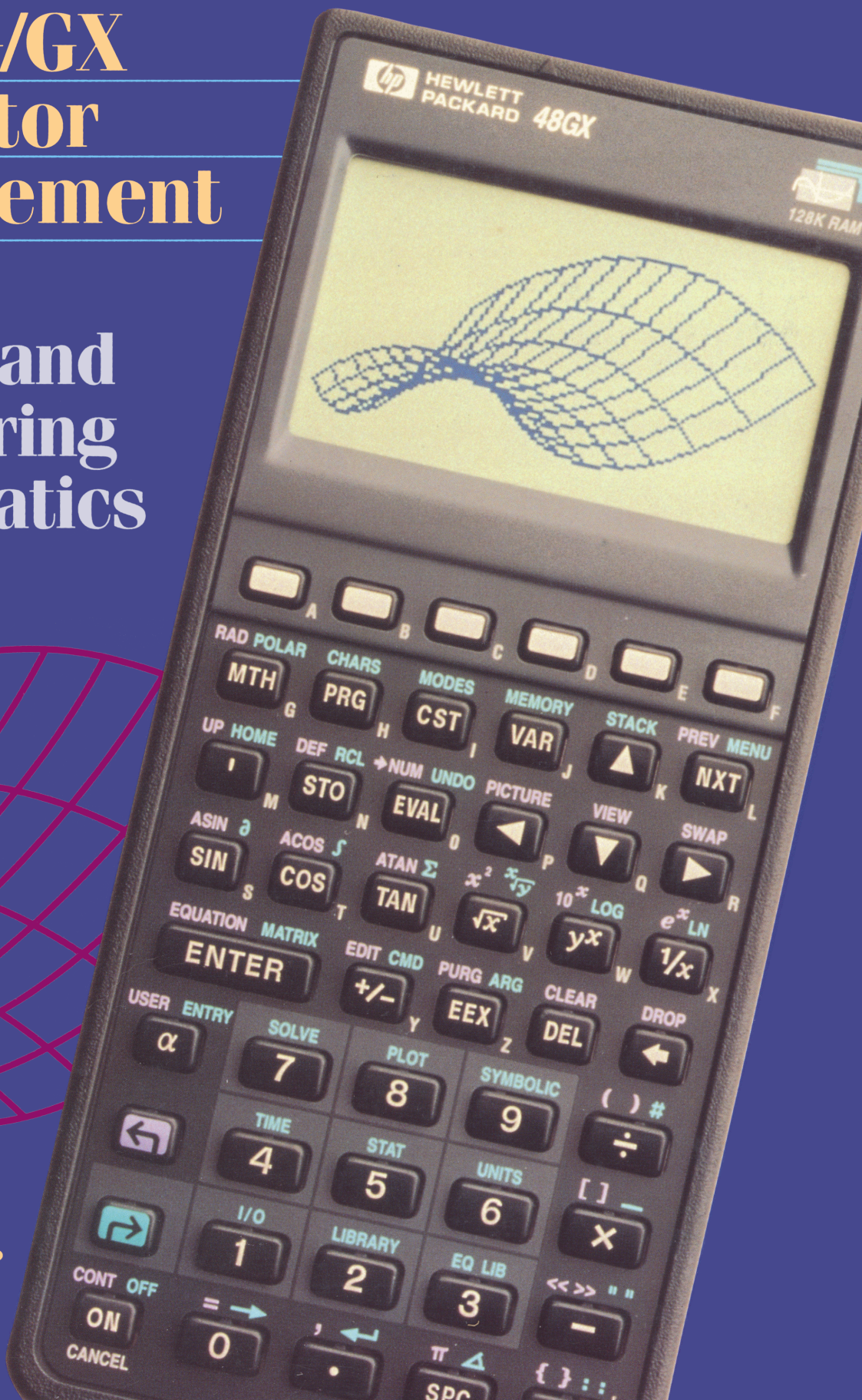
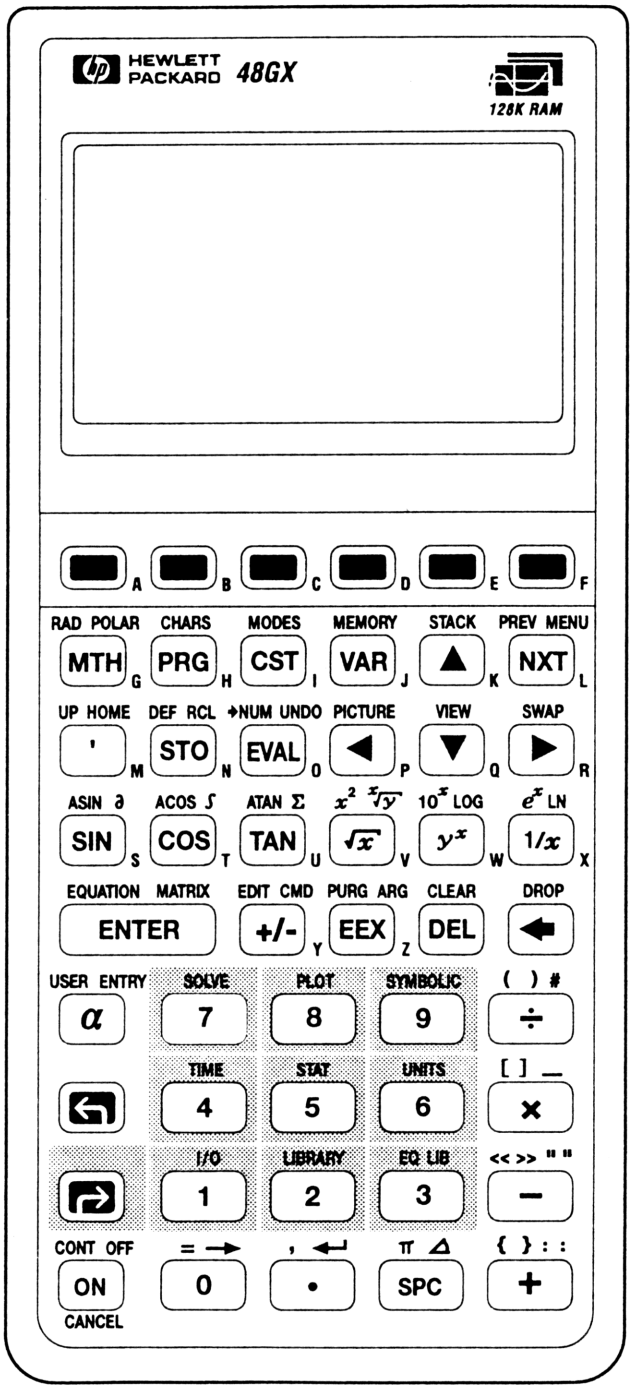
HP-48G/GX Calculator Enhancement for Science and Engineering Mathematics





Edited by **Donald R. LaTorre**


HEWLETT PACKARD 48GX
128K RAM
A
B
C
D
E
F
RAD POLAR
MTH
G
CHARS
PRG
H
MODES
CST
I
MEMORY
VAR
J
STACK
K
PREV MENU
NXT
L
UP HOME
'
M
DEF RCL
STO
N
→NUM UNDO
EVAL
O
PICTURE
P
VIEW
Q
SWAP
R
ASIN ∂
SIN
S
ACOS ∫
COS
T
ATAN Σ
TAN
U
x^2 $\sqrt[x]{y}$
$\sqrt{x}$
V
10^x LOG
y^x
W
e^x LN
$1/x$
X
EQUATION MATRIX
ENTER
EDIT CMD
PURG ARG
+/-
Y
EEX
Z
CLEAR
DEL
DROP
USER ENTRY
α
SOLVE
7
PLOT
8
SYMBOLIC
9
( ) #
÷
TIME
4
STAT
5
UNITS
6
[ ] _
×
I/O
1
LIBRARY
2
EQ LIB
3
<< >> " "
−
CONT OFF
ON
CANCEL
= →
0
, ←
.
π Δ
SPC
{ } : :
+

HP-48G/GX Calculator Enhancement for Science and Engineering Mathematics

Edited by

Donald R. LaTorre
Clemson University

SAUNDERS COLLEGE PUBLISHING
Harcourt Brace College Publishers

Fort Worth Philadelphia San Diego New York Orlando
San Antonio Toronto Montreal London Sydney Tokyo

Cover photo courtesy of Hewlett-Packard

Printed in the United States of America

HP-48G/GX Calculator Enhancement for Science and Engineering Mathematics

0-03-097000-8

Library of Congress Catalog Card Number: 93-085550

CONTENTS

PREFACE

It has been over three years since the Hewlett-Packard HP-48S series calculators were introduced (March, 1990) to the North American scientific community. Assessment of their impact leaves little doubt that the affect upon collegiate undergraduate science, engineering and mathematics has been both significant and substantial. With features such as 32K of expandable memory, two-way infrared communication, serial link to personal computers, a sophisticated operating system and a structured programming language that supports extensive symbolic manipulation capabilities, these original HP-48 units were accurately termed *supercalculators*. The HP-48G series units offer even more: a new, easy-to-use input forms environment for beginners, built-in 128K RAM with two memory expansion ports (in the HP-48GX), enhanced graphics that include the first calculator-based 3D capabilities (and hence the G designation), professional code for differential equations and for matrix operations, and a host of other innovative features. The G series units will most certainly achieve a widespread acceptance in undergraduate education because of the potential for their creative use, on a personal level, by faculty and students alike to enhance teaching and learning.

This volume is one of a growing number of publications that are appearing in support of the use of the high level calculators in undergraduate mathematics. But, unlike most others, this volume is not dedicated to a comprehensive, in-depth discussion of how the HP-48G/GX units can be effectively used in any *one* particular course of instruction. Rather, it is a collection of six independent chapters, each devoted to a particular course and authored by faculty who are experienced in the use of the HP-48 in the material of that course. The chapter titles reflect the courses: Single-variable Calculus, Multivariable Calculus, Differential Equations,

Linear Algebra, Advanced Engineering Mathematics, and Probability and Statistics. Five of the six authors are faculty at Clemson University, which requires all students in the calculus sequence for science and engineering to have their own HP-48's. We are extremely pleased to have join us as author of the chapter on Advanced Engineering Mathematics, Dr. Donald L. Kreider, Professor of Mathematics and Computer Science at Dartmouth College. Don is well-known for his textbooks in that area.

Each of the six chapters is self-contained, and written in the spirit of *showing the potential for using the HP-48G series calculators* in a mainstream mathematics course. We have tried to avoid "teaching the mathematics", and have instead written "about teaching the mathematics". Most of the chapters survey the main topics of a course, and each chapter includes many activities, exercises, explorations and projects that can be engaged by students in a calculator-enhanced treatment of the material. For the first four chapters, Single-variable Calculus through Differential Equations and Linear Algebra, supporting student-oriented material will be available in 1994 from the publisher, Saunders College Publishing, by the same authors. Chapter 6, Probability and Statistics, is new material; only a small portion of Iris Fetta's work in this area appears outside the present volume. Chapter 5, Advanced Engineering Mathematics, is also new material, having no existing counterpart published elsewhere. It addresses the use of the HP-48G/GX in an important and significant area of undergraduate mathematics, especially for students in analysis, sciences or engineering.

It has been a joy for me to serve as Consulting Editor for this volume. In a deliberate attempt to foster the creativity of each of the authors, I have refrained from imposing editorial restrictions in terms of format, structure and style. You will thus notice a wide variation in these features from chapter to chapter, and I hope you will find this to be somewhat refreshing.

I especially wish to express my appreciation to Bill Wickes of Hewlett-Packard, who served as head of the original design and development team for the HP-48. His wisdom and genius have been a source of inspiration for us all.

The timely appearance of this volume could not have taken place without the splendid support and coorporation that the authors received from Hewlett-Packard and Saunders. Hewlett-Packard provided prototypes of the HP-48G series units and members of the software development team helped us sort through many of the new routines. I wish to recognize Diana Byrne, Charlie Patton, and Paul McLellan of that team for all their help. Our editor, Jay Ricci of Saunders, was especially supportive of our efforts from the very beginning to bring forth this new material. And finally, I express special thanks to Mrs. April K. Haynes who, with great skill and patience, word-processed several chapters in this volume, as well as all the many revisions, corrections and peripheral material to be found here.

Clemson University

Don LaTorre
June, 1993

CALCULATOR PRELIMINARIES

Although the chapters in this volume are self-contained, the authors have assumed that readers will have a basic familiarity with the HP-48G/GX and its operation, at least to the extent of being able to do elementary numerical calculations and to enter algebraic expressions. For those inexperienced with HP-calculators, this basic familiarity can best be acquired by a hands-on study of Chapters 1, 2 and the first five pages of chapter 3 of The HP-48G Series Users' Guide. For convenience, we briefly review the basics here, and include as an appendix some material on program housekeeping.

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 three 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 blank 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 will turn you to the second page of

the MTH menu and another [NXT] will cycle you back to the beginning. The small horizontal tabs above the labels in the MTH menu indicate that each of the boxes contains a submenu (a file, or *subdirectory* in HP parlance). Open the HYP submenu by pressing the white menu key beneath it to access the various 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.

Display Settings

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

To display numbers in standard form, set your unit to STD display mode (the default setting) by pressing [↰] [MODES], opening the FMT (= Format) menu and checking to see that the left-most menu box reads [STD☐]. The small box next to STD indicates that STD mode is active. If the menu simply reads [STD] press the associated white menu key to activate STD mode. 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 [↱] and the left-shift key [↰] are color coded to many of the keyboard labels, and the [α] key is used to obtain alphabetical characters.

Adjacent to [ENTER] is [+/−] for changing signs, then [EEX] for entering exponents, [DEL] for deleting characters (and clearing the stack), and [⌫] for backspace-and-delete (and dropping objects from level 1). The [SIN], [COS], [TAN], and [√x] keys are just above, as are [Y^X] (for obtaining powers) and [1/x]. 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 left-and right-shifted labels printed above them, but many have only one of the two.

The keys that have only green labels above them represent *applications*, e.g., I/O, PLOT, SOLVE, TIME, UNITS. The right-shifted version of an application key invokes a specially designed user-interface that lets you interact directly with the named application, often through the use of *input forms*, which are the HP

equivalent of the familiar computer "dialogue boxes". Alternatively, the left-shifted version of an application key gives you access to the various commands on the *command menu* that is associated with the particular application. The commands may be included in programs or executed directly from the keyboard while viewing the stack display screen.

Numerical Calculations

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

To edit this result, use [↰] [EDIT] (the purple [+/-] key), use the right cursor key to move the cursor over the 9 , delete the 9 with [DEL] and press 3 [ENTER]. Now use [↱] [LN] (the right-shifted [1/x] key) to obtain the natural logarithm. To multiply by π, press [↰] [π] (π is obtained with the left-shift [SPC] key) then [×]. Notice the symbolic result '6.35396147609 * π' on level 1, enclosed in tick marks. To convert this to a numerical result, use [↰] [→NUM] (the left-shift [EVAL] key). Now drop the 19.9615586945 from level 1 with [⇦]. The [⇦] key drops objects from level 1; the adjacent key (labeled CLEAR in purple)

clears the entire stack. Normally, you need not left-shift these keys; shifting is required only when the command line is active.

Algebraic Expressions

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

To enter the algebraic expression 'SIN(X)', press ['] [SIN] [α] [1/x] [ENTER]. Notice the location of the cursor after each keystroke; after [1/x] the cursor is still inside the right parenthesis. To move it outside, use the right cursor key [▷]. But, pressing [ENTER] does it all for you. As a more complicated example, try 'COS(X^2)/(2*X^3)'. The keystroke sequence is:

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

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

HP-48G/GX Calculator Enhancement
for
Single-Variable Calculus

Donald R. LaTorre, John W. Kenelly, James H. Nicholson

Calculus of a single variable has proven to be the mathematical common denominator for students in practically every scientific field. Exceedingly rich in terms of concepts and ideas, calculus was for almost three centuries linked to every major development in mathematics, science and technology. Its applications are diverse and widespread and today a study of elementary calculus forms the basic mathematical foundation for careers in mathematics, science and engineering.

But the teaching of calculus has not kept pace with the times. All too often our courses have catered mainly to traditional analytic presentations and largely ignored the strong graphical and numerical aspects that have always been present but which are now readily accessible with microcomputer or calculator technology.

This chapter is an introduction to how the HP-48G/GX supercalculator can be used to effectively enhance the teaching and learning of the graphical and numerical aspects of single-variable calculus. Our presentation is, of necessity, somewhat brief and makes no claim to being comprehensive. It is intended only to point the way for teachers to use the high-level Hewlett Packard units by showing some of their power and versatility, as well as some examples of where and how they can make a difference. Reference [1] is an expansion suitable for classroom use by students.

The chapter is divided into four sections. Section 1 is concerned with representing functions numerically and graphically. We begin by representing mathematical functions on the HP-48 as user-defined functions, and then show how the SOLVE application can also be used to evaluate functions. The HP-48's PLOT application lets you represent functions as plots in the PICTURE environment and we discuss plotting via the main plot screen (which uses input forms and choose boxes) as well as with the commands on the plot menu.

Section 2 discusses derivatives: from the simple evaluation of difference quotients and their susceptibility to cancellation errors to the calculator's ability to perform symbolic differentiation. Root finding and the analysis of the graphical behavior of functions in terms of local extrema and inflection points are also examined.

Section 3 considers integration. We provide a calculator directory of short routines that produce various kinds of Reimann sum approximations to definite integrals - left and right rectangle, trapezoidal, midpoint, and Simpson's approximations - and then discuss the HP-48's built-in numerical integration routine. This section concludes with a calculator-based activity designed to reinforce students' understanding of Part 1 of the Fundamental Theorem of Calculus.

Section 4 focuses on Taylor polynomials and infinite series, illustrating the ability of the HP-48G/GX to contribute graphically and numerically to the teaching of these important topics. Programs are included that show the calculation of partial sums dynamically.

Along the way, we have included a number of sets of EXERCISES. Although not inclusive, they are fairly representative of the types of activities that students can be called upon to conduct with the HP-48 as they become active participants in the construction of their own personal understandings of the material.

1. FUNCTIONS

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

Representation and Evaluation

A convenient way to represent many mathematical functions on the HP-48 is through the creation of *user-defined functions*. In HP-48 parlance, a user-defined function is a short program that captures the essence of the formal way that we define a function by an equation like F(x) = 2 sin x + sin 4x. Here, F is the name of the function, x is the input variable, and the expression to the right of the = sign is an algebraic description of the desired output for a given input x.

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

To evaluate this function, enter the desired input and press the menu key [F]. For example, put 'T^2' on level 1 and press [F] to see '2*SIN(T^2) + SIN(4*T^2)'.

Likewise, press 4 [F] to see 2 sin 4 + sin(4*4) evaluated as -1.80150830728. You may enter the equation 'F(X) = expression in X' directly or by first entering 'F(X)', then the 'expression in X' and pressing [↰] [=]. 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 $G(x,h) = \frac{\sin(x+h) - \sin x}{h}$ enter 'G(X,H) = (SIN(X+H) – SIN(X))/H'and press [↰] [DEF].

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-48 is to use the IFTE command, found on the third 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 $p(x) = \begin{cases} x^2 - 2x & x < 0 \\ 1 - x^2 & 0 \le x \end{cases}$, the desired command is 'IFTE(X<0, X^2 – 2*X, 1 – X^2) ' on stack level 1. Begin with ', then go to the second page of the PRG BRCH menu and press [IFTE] , followed by the three required arguments separated by commas, then [ENTER]. The inequality relations are on the first page of the PRG TEST menu. This expression can now be treated like any other function and evaluated, graphed, differentiated or integrated. 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 whether the function has a limit as x approaches 0.

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

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

is IFTE($x \le a_1$, $f_1(x)$, $f_2(x)$).
For three or more pieces, you can nest the IFTE commands:

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

use IFTE(X < a_1, f_1(X), IFTE(X < a_2, f_2(X), f_3(X))).

Although you can evaluate functions by representing them as user-defined functions, you may also use the SOLVR . The SOLVR is designed to solve equations, but the format of its menu makes it convenient for evaluating functions. With the function on level 1, press [↰] [SOLVE] [ROOT] then load the function on level 1 into EQ by pressing [↰] [EQ]. Now press [SOLVR]. 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 investigate the behavior of the function $f(x) = \frac{x+2}{2x+1}$ as $x \to 0$:

Put '(X + 2) / (2*X + 1)' on level 1 and press [↰] [SOLVE] [ROOT] , then [↰] [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.

What is the limit of f(x) as $x \to 0^+$? Now evaluate f for values of $x \to 0^-$.

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 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 values of two functions at various 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 this 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].

EXERCISES 1.1

1. Create a user-defined function for f(x) = xlnx and evaluate it for $x = 10^{-2}, 10^{-4}, \ldots, 10^{-14}$ (Suggestion: press [EEX] 2 [+/-] to input 10^{-2}, etc.) What does the limit of xlnx, as $x \to 0^+$ appear to be? l'Hospital's rule will determine the limit analytically.

2. Investigate, numerically, the following limit:

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

3. Evaluate $f(x) = (1 + 1/x)^x$ for $x = 10^2, 10^4, \ldots, 10^{10}$ and 10^{11}. Then make a conjecture for

$$\lim_{x\to\infty} (1 + 1/x)^x.$$

Now evaluate f(x) for $x = 10^{12}$ and watch what happens. Can you explain this?

4. Use the IFTE command to represent the function $f(x) = \begin{cases} x^2 & x<0 \\ \cos x & 0\le x \end{cases}$. Evaluate for a sequence of values approaching 0 from the left; then use a sequence of values approaching 0 from the right. Do any of the limits, $\lim_{x\to 0^-} f(x)$, $\lim_{x\to 0^+} f(x)$, and $\lim_{x\to 0} f(x)$, exist?

5. Conduct a numerical investigation of $\lim_{x\to 0} \frac{\sin x}{x}$.

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 analytic 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 calculator, graphical representations can be used extensively 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 $-6.5 \le x \le 6.5$ and $-3.1 \le y \le 3.2$, with a common unit scaling of each axis. Since there are 131 columns and 64 rows of pixels, these 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 nicely 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 accomodate trigonometric graphs, make sure your calculator is set to radians mode. The [↰] [MTH] 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.

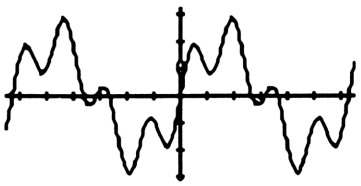
Functions are represented graphically as plots in the PICTURE environment. The general procedure to produce a plot of a function of a single independent variable is as follows:

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

The HP-48G/GX allows you to access the PLOT application in two different ways to enter a function's expression and to set the plotting parameters: with [↱] [PLOT] to interact directly with the main PLOT screen, or with [↰] [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 4 x with the default plotting parameters.

Using the PLOT screen: From the stack environment, go to the PLOT application with [↱] [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 [Δ] [CHOOS], highlight Function and press [OK]. If necessary, use [▷] 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 [ENTER]. 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 appear otherwise, go to the next page of the PLOT menu with [NXT], press [RESET] and activate Reset Plot with [OK]. Once the default plotting parameters are set, return to the previous page with [PREV] and press [ERASE] to erase any previous plot and [DRAW]. You should see a graph like this:



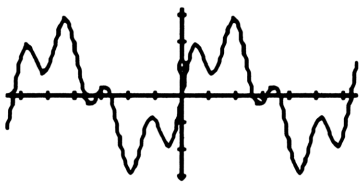
Press [ON] twice to return to the stack environment.

Using the PLOT menu: Enter '2*SIN(X) +SIN(4*X)' on level 1, and press [↰] [PLOT] then [↰] [EQ], to store this expression into EQ. If you do not now read "Ptype: FUNCTION " at the top of your screen press [PTYPE] and then

[FUNC]. 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 graph previously drawn, then [DRAX] and [DRAW]. You should see a graph like this:



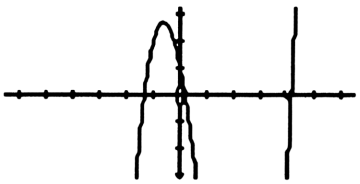
When you have finished viewing the graph, return to the stack by pressing [ON] twice; you can always bring back the graph by using the [◀] key.

Often, you can see more of a graph if you compress or expand the viewing screen vertically or horizontally by using the [ZOOM] key, as in the next example.

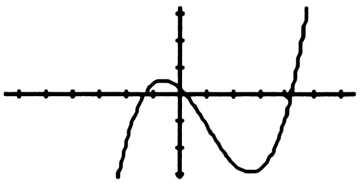
EXAMPLE 2. Graph $y = x^3 - 3x^2 - 5x + 1$.

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]. Press [ERASE] and [DRAW] to produce this graph.



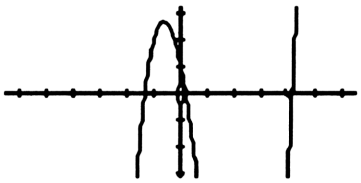
To see more of the lower right part of the graph we will zoom out on the vertical-axis but leave the x-axis unchanged. Open the [ZOOM] menu, then the [ZFACT] menu. Toggle down to the V-FACTOR with [▽], change it to 5 with 5 [ENTER], then press [OK]. Move to the next page with [NXT] and zoom out on the vertical axis with [VZOUT]. You will get the graph:



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] twice 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] [↰] [EQ] [PPAR]. 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 [PLOT], then use [ERASE], [DRAX] and [DRAW] to produce this graph:

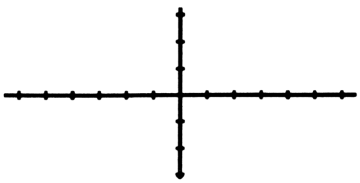


Now we can zoom out as before to see more of the local behavior.

The above two examples convey the major differences in using [↱] [PLOT] and [↰] [PLOT] to access the PLOT application. The [↱] [PLOT] lets you interact with the main PLOT screen, and the [↰] [PLOT] provides direct access to the various commands on the PLOT menu. While beginners may prefer to interact with the PLOT screen, more experienced users tend to prefer the menu commands. From here on, we leave the choice to the reader.

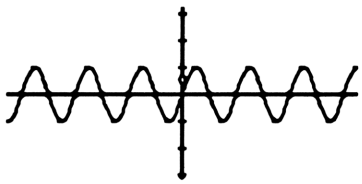
Here are two examples of graphs requiring adjustment on the range of x values:

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

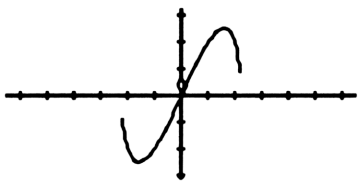


Where are the points? In the default mode, the HP-48 calculates values of y for each of the 131 values of x, .1 unit apart from x = -6.5 to x = 6.5. Since 10π times each of these numbers is an integer multiple of π, the sine function is 0 at each of these values of x. To get a better picture of the graph we can compress the viewing

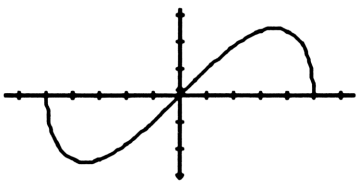
window in the x direction. Zoom in on the horizontal axis by a factor of 10 (set the H-FACTOR, then use [HZIN] to see:



EXAMPLE 4. Graph $y = x\sqrt{5 - x^2}$ with the default settings to see:



From the function, y is 0 when $x = \pm\sqrt{5}$, but these points do not show on the graph. Correct to four places, $\sqrt{5} = 2.2361$. With the default viewing screen, the HP-48 will plot a point for x = 2.2, but for $2.3 \le x$, y is a complex number so no points will be plotted. If we zoom in on x by a factor of 2.2361 we see

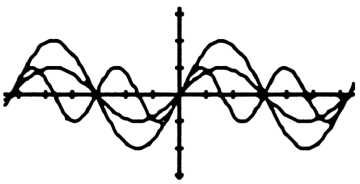


The viewing window was reset so that 5 units on the x axis is approximately $\sqrt{5}$.

To superimpose the graphs of two or more functions you can graph them individually without erasing. A better procedure is to build a list { F G H . . .

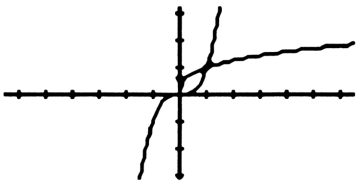
etc.} of the functions to be graphed, and store it into EQ. When DRAW is activated, the functions in the list are drawn sequentially, left-to-right.

EXAMPLE 5. Graph sin x, 2 sin x and sin 2x on the same set of coordinate axes with the default parameters. Put 'SIN(X)', '2*SIN(X)' and 'SIN(2*X)' on the stack and execute 3 →LIST (on the PRG LIST menu). You will see the list { 'SIN(X) '2*SIN(X)' 'SIN(2*X)'}. Now store this list into EQ and then press ERASE , DRAX and DRAW to see the graphs. Observe how the graphs are drawn sequentially from the list.



To draw the grpahs in the list simulatneously instead of sequentially, go to the second page of the ↰ PLOT menu, open the FLAG submenu and toggle on SIMU .

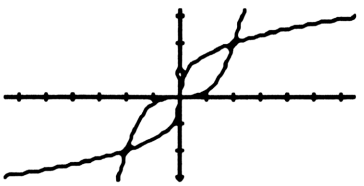
To compare the graph of $y = x^3$ with that of its inverse $y = \sqrt[3]{x}$, you may begin by graphing the list { 'X^3' 'X^(1/3)'}. Using the default settings you will see



which fails to show the left branch of $y = \sqrt[3]{x}$. 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. Thus, no pixel is activated. Although you may at first find this a bit disquieting, the ability of the HP-48 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 [↱] [$\sqrt{x}$] key). For instance, with -8 on level 2 and .333333333333 on level 1, the [y^x] key will return the principal cube root of -8: (1, 1.73205080757). But with -8 on level 2 and 3 on level 1, the [$\sqrt[x]{y}$] key will return the real cube root of -8 : -2. To see both branches of the graph of $y = \sqrt[3]{x}$, graph the expression 'XROOT(3,X)'. To enter this, put 'X', then 3 on the stack and press [$\sqrt[x]{y}$]. When the list { 'X^3' 'XROOT(3,X)'}is graphed with the default screen, then enlarged by a factor of 2 with the [ZOOM] menu, we see



Notice that this function and its inverse meet on the line y = x and that the graphs are reflections across this line. Here is a program which, when given a function f whose graph is displayed on the graphing screen and an interval $a \le x \le b$ contained in the current x range, will graph the line y = x and then the graph of the inverse *relation* for f with x restricted to $a \le x \le b$. If f is a one-to-one function on the given interval, then the inverse relation will be the inverse function of f restricted to $a \le x \le b$.

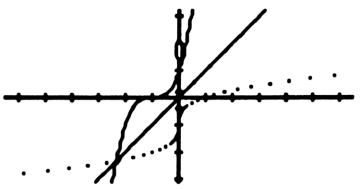
INV.F

Input: level 2: a, a real number
level 1: b, a real number > a
As a stored variable EQ: a function f.

Effect: Draws, over the graph of f, the graph of y = x and the graph of the inverse relation of f, with x restricted to the interval [a,b].

```
« → A B « RCEQ → EQ1 « CLLCD 'X' STEQ DRAW EQ1 STEQ A
B FOR I I 'X' STO X RCEQ EVAL SWAP R→C PIXON .1 STEP 'X'
PURGE PICTURE » » »
```

EXAMPLE 6. Graph $y = (x + 1)^3$ with the default viewing window. To see its inverse, clear the graphing screen with [ON] and go to the VAR menu with [VAR]. Enter the "screen domain" of the plotted graph: -3, 1 and press [INV.F] to see

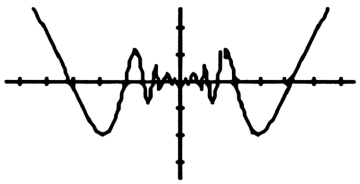


Since the graph of the inverse function is plotted point-by-point, it will appear dotty as in this example.

The ZOOM menu of the HP-48G/GX calculators contains assorted commands for zooming on a graph and the BOXZ applicaiton is especially helpful for zooming in

on a particular region of a graph. The basic idea is to capture the region of interst within a samll "box", then zoom in on the box. Here's an example.

EXAMPLE 7. Begin by graphing $x \sin \frac{1}{x}$ 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 (0,0), then down 3 pixels and open [BOXZ]. Now move the cursor 5 pixels to the right of (0,0), then 3 pixels above (0,0). Notice that the cursor "drags" a small box that has the origin as its center. Now press [ZOOM] to zoom in on the box.



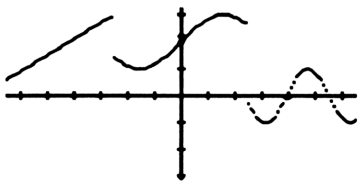
Now repeat this zooming-in process with BOXZ by moving to a corner of a box 5 pixels to the left and 3 pixels below the origin, then moving to the diagonally opposite corner 5 pixels to the right and 3 pixels above the origin and pressing [ZOOM]. Are you ready to give an answer to

$$\lim_{x \to 0} x \sin \frac{1}{x} = ?$$

Piecewise-defined functions are graphed by storing the defining IFTE expression into EQ and proceeding as usual. To get the graph shown in the next example, use the default viewing screen and the disconnected graphing 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 graph 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 graph $f(x) = \begin{cases} .6x + 4.5 & x < -2.5 \\ 2 + \sin x & -2.5 \le x < 2.5 \\ -\cos 2x & 2.5 \le x \end{cases}$, use the expression 'IFTE (X < -2.5, .6*X + 4.5, IFTE (X < 2.5, 2 + SIN(X), -COS(2*X))'. This gives the graph:



When you have finished, reset to graph in connected mode.

EXERCISES 1.2

1. (a) Graph the list { 'SIN(4*X)' '-2*SIN(X)'} using the default graphing screen.

   (b) ERASE and graph the sum of the two functions in the list.

   (c) Overdraw your graph in (b) with the graph of y = -2 sin x.

   (d) ERASE and graph the product of the two functions in the list.

   (e) Overdraw your graph in (d) with the graph of y = -2 sin x.

2. Graph y = cos(10πx) on the default graphing screen. Why does the representation of the graph look this way? Adjust the screen to make the representation look more like a cosine curve.

3. Set your calculator to degree mode and then graph y = sin(x°) using the default screen. Without changing back to radian mode, how should you zoom on X to

make this graph look like sin x, x in radians? (When you're done, set to radian mode.)

4. Graph $y = x^3 - 1.3x^2 + .32x - .02$ using the default screen. Clarify the behavior of this function near the origin by using BOXZ several times.

5. To appreciate how "steep" are the graphs of simple polynomial functions, begin by graphing $y = 34x^3 - 91x^2 - 117x + 54$ in the default screen. Now zoom out along the y-axis as necessary until you can see all local extreme points.

6. Graph $y = \cos(\cos^{-1}x)$ using the default screen. The result is what you expected, isn't it? Now, ERASE and graph $\cos^{-1}(\cos x)$. Can you explain what you see?

7. Investigate, graphically, the following limit:

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

(See Exercise 2 in EXERCISE 1.1)

8. Graphically investigate the behavior of $f(x) = \sin\left(\frac{1}{x}\right)$ near $x = 0$. Begin by graphing in the default screen, then use BOXZ . What is your conclusion?

9. Graph $y = \begin{cases} -x & x < 0 \\ \sin x & 0 \le x < \pi \\ x - \pi & \pi \le x \end{cases}$, using the default screen.

10. Graph $y = x\sqrt{3 - x^2}$. Adjust the viewing screen to make the graph touch the x axis at the end points of the domain.

11. Graph $y = x^3 - 9x^2 + 2x + 48$ with the default screen, then zoom out on y by a factor of 16 to see the local maximum. Now move the cursor to (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 wish to press [–] to remove the menu key labels.

12. (a) Graph $y = x^2 + \frac{4}{x}$ on the default screen, then zoom out on y by a factor of 4.

    (b) Graph $y = \frac{x^3 - 1}{x - 1}$ on the default screen, then relocate the center of the viewing rectangle at (0,2) to see the "hole".

13. (a) Use the default screen to graph $f(x) = 2x - 3$, then use the INV.F program to graph f^{-1}.

    (b) Write an equation for f^{-1}.

    (c) ERASE, then graph $g(x) = -.6x + 1$ 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?

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

    (a) Graph the composite function $f(x) = u[v(x)]$ and compare with the graph of $v(x)$.

    (b) Graph the composite function $g(x) = v[u(x)]$ and compare with the graph of $u(x)$.

15. Use the XROOT command to graph $y = 2(x + 2)^{2/3} + \frac{x - 4}{x^2 + 1}$. Use the default screen, then zoom out on xy by $\frac{6}{10}$.

2. DERIVATIVES

The derivative f' of a function f is defined by $f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}$. The difference quotient on the right-hand side is the average rate of change of f, with

respect to x, over the interval [x, x+h] and is also the slope of the secant line joining the points (x,f(x)) and (x + h, f(x + h)) on the graph of f. For a given x, we may approximate f'(x) numerically by evaluating $\frac{f(x+h)-f(x)}{h}$ for suitably small values of h.

A simple way to do this on the HP-48 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. (See pages 3-4 to refresh on user-defined functions)

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

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

However, you must exercise caution because the numerical computation of difference quotients is susceptible to serious cancellation error with finite precision arithmetic. 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 input values (1,H) you will obtain

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

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 take derivatives with the calculator. For example, when we want to graph a function, its first two derivatives, and then find their roots. Since the graphing and root-finding will be done on the calculator, we may as well carry out the differentiation process there also.

The HP-48 uses the [∂] key (the [↱] [SIN] for differentiation and requires two inputs: the function to be differentiated (in symbolic mode between ' ') on level 2, and the variable of differentiation on level 1:

2: 'an expression in X'
1: 'X'

When the [∂] key is pressed, the derivative will appear on level 1.

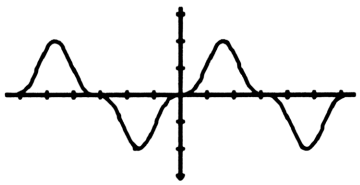
EXAMPLE 9. To graph $f(x) = 2\sin^3 x$ and its derivative f', we may proceed as follows:

Put two copies of '2*SIN(X) ^ 3' on the stack (press [ENTER] to duplicate level 1), then enter 'X' and press [∂] to see

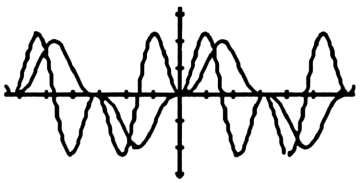
2. '2*SIN(X)^3

1. '2*(COS(X)*3*SIN(X)^2)'

To graph f first, execute SWAP and then graph with the default viewing screen:



Now draw the graph of f' over this without erasing:

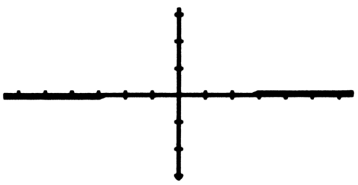


Notice that the graph of f has a maximum or minimum point at those values of x where the graph of f' crosses the x axis. There are also three points where the graph of f has a horizontal tangent but no extreme value. At the x coordinates of these points, f' has value 0 but its graph does not cross the x axis. Finally, at the values of x where f' has an extreme value, f has an inflection point.

EXAMPLE 10. If you put your calculator in degree mode and take the derivative of f(x) = sin x, you will see 'COS(X)*(π/180)'. Why? There are several explanations, each addressing the question from a different aspect.

Analytically, we know that, for x in radians, $\frac{d}{dx}(\sin x) = \cos x$. So, $\frac{d}{dx}[\sin(x°)] = \frac{d}{dx}[\sin(\frac{\pi}{180}x)] = \cos(\frac{\pi}{180}x)\frac{d}{dx}(\frac{\pi}{180}x) = \cos(x°)(\frac{\pi}{180})$.

Graphically, with the calculator still in degree mode, if you draw the graph of sin x with the default parameters you will see



For $-2.8° \le x \le 2.8°$, sin x = 0 to the nearest tenth of a unit, which is the difference in y coordinates between adjacent rows of pixels for the default viewing screen. Note that $f'(5) = \cos(5°)(\frac{\pi}{180})$, which is approximately .017. Certainly, this value seems reasonable for the slope of the tangent line to the above graph at the point where x = 5.

For a more basic explanation, you can return to the original derivation of $\frac{d}{dx}$ (sin x). The derivation often uses the result that, for h in radians, $\lim_{h\to 0} \frac{\sin h}{h} = 1$. This limit is customarily proved using an inequality involving the areas of two triangles and the area of a certain sector of the unit circle. When h is in radians this sector has area h/2, but if h is in degrees, this sector has area $\pi h/360$.

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. (Note: to obtain lowercase alphabetical characters, use [α] [↰] [D], [α] [↰] [E], etc.)

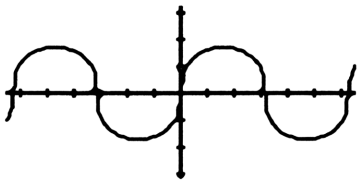
derXROOT

Input: 'XROOT(N,F(X)) on level 1

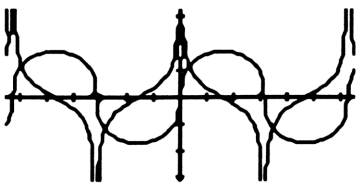
Effect: puts $\frac{d}{dx}\left(\sqrt[N]{F(X)}\right)$ on level 1

« → N W Y Z 'INV(N)*XROOT(N,W)^(1 – N)*Z'»

EXAMPLE 11. Find the derivative of $f(x) = \sqrt[3]{5 \sin x}$ and graph both f and f'. With the program derXROOT in the HOME directory of your 48, put two copies of 'XROOT(3, 5*SIN(X))' on the stack, enter 'X' on 1 and press [∂] to see the derivative '.333333333333*XROOT(3, 5*SIN(X))^–2*(5*COS(X))'. Now press [SWAP] to put f on level 1 and graph it on the default screen to see



Now graph f' without erasing to see

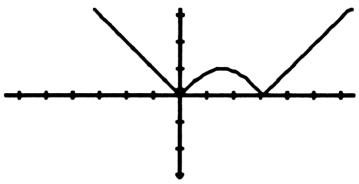


Here, f' has no extreme values, but f has inflection points at those values of x for which f' is not defined.

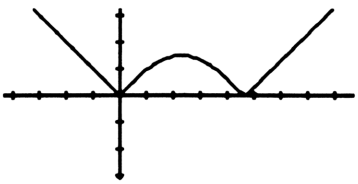
Although the HP-48 will not completely symbolically differentiate a function defined with the IFTE command, it will correctly graph the derivative.

EXAMPLE 12. Find the derivative of $f(x) = \begin{cases} -x & x < 0 \\ \sin x & 0 \le x < \pi \\ x - \pi & \pi \le x \end{cases}$ and then graph both f and the derivative.

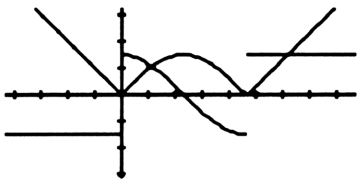
Put two copies of 'IFTE(X < 0, -X, IFTE(X < π, sin(X), X – π)) on the stack and graph in disconnected mode with the default parameters to see



For greater clarity, particularly after we overdraw f', move the cursor to a point on the x axis approximately under the high point of the graph and press CNTR. The graph will be redrawn with the point you chose 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, ∂X(-X), ∂X(IFTE(X < π, SIN(X), X – π)))'Use of EVAL does not change the expression. However, if we graph f' without erasing we see the graph of the derivative superimposed on the graph of f:

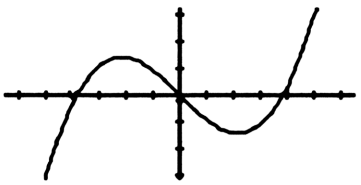


Notice that the minimum values of f occur at values of x where f'(x) does not exist, and that f has no inflection points.

Newton's Method

The technique known as Newton's method has almost 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 effectively introduces students to several important ideas: algorithms, recursion, iteration. And it is especially easy to implement on the HP-48.

EXAMPLE 13. 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 graph below is the result of graphing with the default parameters and then zooming in by a factor of .333 on *both* x and y:



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 good 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 for 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 agreeing 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 the desired user-defined function as NM.

NEWTON

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

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

```
« 'NM(X)' 'X' ROT DUP 'X' ∂ / – = DEFINE 'X' PURGE »
```

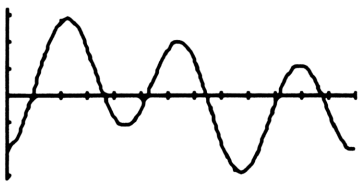
If, for instance, you put '3*X – 4*SIN(X)' on level 1 and press [NEWT], you can execute Newton's method from the menu key [NM] as above.

Roots

Students of calculus often need to find the real roots of a function. But, without ready access to numerical root-finding procedures, they have traditionally been constrained to work with polynomials that factor easily and with simple exponential, logarithmic and trigonometric functions. The HP-48 Solve application provides an advanced level of root-finding capability that enables students to expand their investigations to almost any function that they may encounter.

The Solve application is accessible through the SOLVE menu and requires an initial estimate of the root in question. Often, the best way to get such an estimate is from a graph of the function.

To illustrate, consider the problem of finding all roots of the equation sin x – 2 cos 3 x that lie in the interval [0, 2π]. Begin by graphing the function f(x) = sin x – 2 cos 3 x using the XRNG : 0 6.28 and the default YRNG : -3.1 3.2 . After storing the function in EQ, open [PPAR], press 0 [SPC] 6.28 [XRNG] to set this x range.



To get an initial estimate of the left-most root, move the cursor to the apparent crossing of the graph of y = f(x) with the x-axis and press [ENTER]. [ON] will now exit you from the picture environment and you will see the pixel coordinates of the estimate on level 1 of the stack. Now enter the Solve application with [↰] [SOLVE], then press the [ROOT] and [SOLVR] menu keys. Since 'SIN(X) – 2* COS(3*X)' was stored into the reserved variable EQ in order to produce the graph, the contents of EQ will appear at the top of the screen. Two menu labels appear at the bottom left: [X] and [EXPR=]. Press [X] to enter the initial estimate; a

message reflecting the result of that action will appear at the top of the screen. Now press [↰] [X] to activate the SOLVR's root-finding routine; notice the temporary message "Solving for X" near the top. When the root-finder is finished, the message "Sign Reversal" will appear near the top and the decimal approximation to the root will appear on level 1 as X: .450451159135. To evaluate the expression stored in EQ at this value of X, press [EXPR=] and see Expr: -.000000000002. The message "Sign Reversal": indicates that the HP Solve application was unable to find a point where the value of the expression in EQ is 0 to within the calculator's 12-digit precision; it found two points where the value of the expression has opposite signs, but could not find a point between them where the value is 0.

For convenience, most of the above procedure has been programmed into the command ROOT on the PICTURE FCN menu. Thus to quickly find the other roots, press [◁] to view the graph, move the cursor to the apparent root that is second from the left, and press [FCN] and [ROOT]. You will see ROOT: 1.74250596672 displayed at the bottom of the screen. If you exit to the stack with two presses of the ENTER key, you will see this last root displayed on the stack as a "tagged" object. You can now find the other four roots in this way. When you're finished, purge X from your user memory.

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 in programs. It solves an expression (on level 3) for an unknown (on level 2) using a first guess (on level 1). You may want to try it out now for the example at hand.

EXERCISES 2.1

1. For $f(x) = (x^2 + 5)^3$, estimate $f'(.6)$ by setting up user-defined functions to evaluate the difference quotient.

2. For the following functions f, use the HP-48G/GX to obtain the derivative f'. Graph f and f' in the same viewing window and examine the graphs. By noting roots of f' or points where f' fails to exist, estimate local extreme values of f. By noting where f' has horizontal tangents, estimate inflection points of f.

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

   (b) $f(x) = \dfrac{4}{2+x^2}$

   (c) $f(x) = \dfrac{4}{x^2-5}$

   (d) $f(x) = \cos 2x - \sin x$

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

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

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

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

3. Use Newton's method to find all roots of

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

   (b) the equation $2^x = x^{10}$ (how many roots are there?)

   (c) $f(x) = \sqrt[3]{x-2}$. Suggestion: Start with $x_0 = 2$ and explain what happens. Then begin with $x_0 = 2.1$ and explain (geometrically) what happens.

4. Use the Solver and a graphical estimate to obtain the roots of

   (a) $e^{x+1} = \cos^{-1}(x+2)$

   (b) $3e^{-x^2/4} = \frac{4}{2+x^2}$

   (c) $x = \ln \frac{1}{x}$

5. It is well-known that the centroid of the St. Louis arch is in the shape of an inverted catenary (hyperbolic cosine). The oustide 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 may use the fact that the point (315,0) lies on the arch. Use the Solver to determine the parameter a and then write an equation for the St. Louis arch that is free of unknown parameters. Remember, the Solver only needs an initial guess. The cosh command resides on the MTH HYP menu.

Analyzing Functions

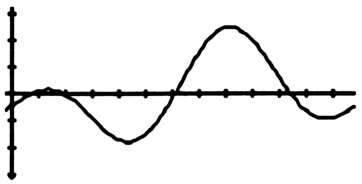
The graphical representation of a function produced on a calculator's screen often provides valuable information about the function's behavior. When graphical techniques are effectively combined with an understanding of the derivative as a

rate of change, we have a powerful tool for analyzing a function's behavior in considerable detail.

After the DRAW command is executed and the HP-48G/GX draws a graph, the calculator enters the PICTURE environment and displays the PICTURE menu. In addition to the zooming operations accessible through the ZOOM submenu and the BOXZ key, the FCN submenu contains commands appropriate for analyzing a function's behavior with calculus without leaving the PICTURE environment.

We begin with an example that would not be appropriate without technology.

EXAMPLE 1. Graph $f(x) = \sin(2x) + \cos(x + 2)$, find the x intercepts and the coordinates of its local extreme points and inflection points. Since this is a periodic function with period 2π, it is sufficient to find the desired points on the interval $[0,2\pi)$. Graphing f with the x range set to $-.1 \le x \le 6.29$ and the y range set to $-2.5 \le y \le 2.5$, we see:



Find the roots using the ROOT command on the FCN submenu. You should get:

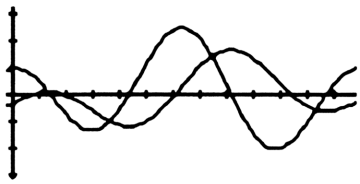
4: Root: 429203673203
3: Root: .904129660127
2: Root: 2.99852476252
1: Root: 5.09291986492

To find the coordinates of the local extrema: To simplify the display, FIX the display mode at 2 places; full 12 digit accuracy is retained in memory and can be

recalled at any time. Return the picture with [◁], activate the trace cursor with [TRACE], then use [◁] to move the cursor to the first apparent high point of the graph. Open the FCN submenu and press [EXTR]. The message EXTRM: (0.67,0.08) appears below the graph. Now move the cursor to the next apparent extremum, a low point, and press [EXTR] to see EXTRM: (2.14,-1.45) displayed. Find the last two extrema in the same way to see EXTRM: (4.00,1.95) and EXTRM: (5.76,-0.77). Now return to the stack display with [ON] [ON], and you will see that these four points have been entered on the stack, each labeled "Extrm".

To find the inflection points. 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. Since an inflection point of f has the same x coordinate as an extreme point of f', we will find the extreme points of f' and calculate the value of f at each of their x coordinates to get the coordinates of the corresponding inflection points of f. The first part can be accomplished within the PICTURE environment.

With the graph of f displayed, go to the second page of the FCN menu and execute [F']. This will plot the derivative f' and then replot f. The high point of f' is offscreen, so we zoom out on y with a factor of 1.5 to see



When you execute [F'], EQ becomes a list {f' f} containing f' and f in order. The function analysis operations ROOT, EXTR, etc., apply only to the first function in the list, which is now f'. Move the cursor to the apparent high point of f' near the origin and press [EXTR]. You will see EXTRM: (0.06,1.10)displayed at the

bottom of the screen and on stack level 1. The x coordinate is the x coordinate of the corresponding inflection point of f.

The following program automates the procedure for obtaining the coordinates of the inflection point. The program assumes that EQ is the list {f' f} and that the coordinates of an extreme point of f' are displayed on stack level 1. With this input, it returns the corresponding inflection point of f with the tag "Infl". The 1 in the name "INFL1" simply 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

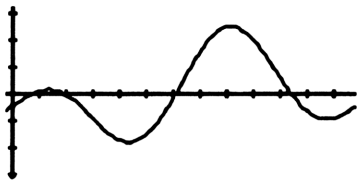
Effect: returns to level 1 the point $(x_0, f(x_0))$ tagged as 'Infl'

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

With this program in your calculator and the above extreme point of f' displayed on stack level 1, press [INFL1] to see 1: Infl: (0.06, -0.35) displayed.

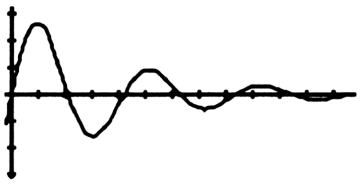
Return the graph to the screen by pressing [PICTURE]. Now move the cursor to each of the remaining three extreme points of the graph of f' and press [EXTR] on the FCN menu at each point. Return to the stack display with [ON] and convert each of the three extreme points of f' to inflection points of f with the use of INFL1. You must do some stack manipulation to move the extrema of f' to level 1 for use with the program. If you want to keep the inflection points in their order on the graph, here's an easy way: with the three extreme points of f' on the stack, press

the [Δ] key to activate the *interactive stack*, use the same [Δ] key to position the pointer on level 3 and then press [ROLL]. This *rolls* the first three levels of the stack *upward* pushing the extreme point on level 3 down to level 1. Press [ENTER] to leave the interactive stack, then press [INFL1] to convert the extreme point now on level 1. Now repeat this entire process twice more until all the extreme points are converted. We display again the graph of f and the points that we have found:



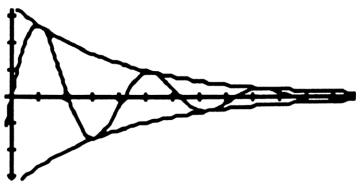
12:	Root:	0.43
11:	Root:	0.90
10:	Root:	3.00
9:	Root:	5.09
8:	Extrm:	(0.67,0.08)
7:	Extrm:	(2.14,-1.45)
6:	Extrm:	(4.00, 1.95)
5:	Extrm:	(5.76,-0.77)
4:	Infl:	(0.06,-0.35)
3:	Infl:	(1.45, -0.71)
2:	Infl:	(3.09,0.28)
1:	Infl:	(4.82,0.64)

EXAMPLE 2. Plot the graph of $f(x) = 1.7\ e^{-x/2} \sin(3x)$ for $0 \le x$. Since we are interested in the graph only for non-negative x, we set the x range as $-.1 \le x \le 6.4$ and the y range as $-1.55 \le y \le 1.6$. 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 $0 \le x$. We could find any of these 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 $-1 \le \sin(3x) \le 1$, 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 coinciding with the graph of v when $\sin(3x) = -\ 1$. We can illustrate this by graphing the list { f u v } with the same plotting parameters we used for f. Exit the PICTURE environment with [ON], recall f to the stack with [↱] [EQ], and use [ENTER] to put a second copy on the stack. Edit the copy on level 1 to read '1.7*EXP(-X/2)', make a second copy of the newly edited expression with [ENTER] and then press [+/-] to change sign. Go to the PRG LIST menu and press 3 [→LIST] to build the list { f u v }. Now store the list into 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 ?*

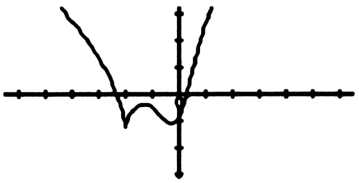
We investigate this question both analytically and graphically. Move the cursor to the first maximum point to the right of the y axis and press EXTR on the FCN menu to see EXTRM: (.468549216461, 1.32664626947) at the bottom of the graphing screen. If this were the point where sin(3x) = 1, then its first 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 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 f', 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 for $0 \le x \le \infty$.

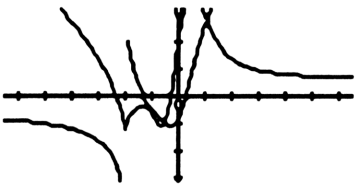
Caution

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 x value closest to the cursor that causes the derivative to evaluate to 0. Thus, if the x coordinate of the extreme point 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 f', you should not use this command.

EXAMPLE 3. Find the roots, extrema and inflection points of $f(x) = 2(x + 2)^{2/3} + \frac{x-4}{x^2+1}$. Put '2*XROOT(3, (X + 2)^2) + (X – 4)/(X^2 + 1)' on level 1 and graph f 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. If we press F ' on the PICTURE FCN menu to graph both f and f' we see:



Notice that f' does not exist when 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 to the right of the origin to be

Infl: (.464327883331, .74032208835).

EXERCISES 2.2

For each of the functions given below, plot the graph and find all local extreme values and inflection points.

1. $f(x) = x + 3 \sin x$, on the interval $[0,2\pi]$.
2. $f(x) = x^3 - x + 2$.
3. $f(x) = \sin x + 2\cos(3x)$ on $[0,\pi]$
4. $f(x) = x^3 - (1.3)x^2 + (.32)x - .02$
5. $f(x) = x^5 + 3x^4 - x^3 - 3x^2 - x + 3$
6. $f(x) = \sin(3x) - \cos(2x)$, $0 \le x \le 2\pi$
7. $f(x) = x^x$
8. $f(x) = x^{\sin x}$, $0 \le x \le 2\pi$
9. $f(x) = \cos(4 \cos^{-1} x)$.

3. INTEGRATION

The HP-48G/GX calculator can be effectively used to enhance the study of the definite integral. To illustrate the basic limiting process that defines the definite integral, short programs can be used to facilitate the rapid calculation of various kinds of Riemann sums to produce numerical approximations: the left rectangle, right rectangle, trapezoidal, mid-point, and Simpson's approximations.

Create a directory named INTG by keying in 'INTG' and pressing [CRDIR] (this key is on the DIR submenu of the [↰] [MEMORY] menu). Now press [INTG] on the VAR menu and enter the following programs (originally provided by Tom Tucker and John Kenelly):

FABSTO

Input: Level 3: the integrand, 'f(x)'.

Level 2: the lower limit of integration, a.

Level 1: the upper limit of integration, b.

Effect: stores f, a and b.

```
« 'B' STO 'A' STO STEQ »
```

NSTO

Input: level 1: a positive integer, n

Effect: sets n, the number of subintervals and stores (b – a)/n as h

```
« 'N' STO B A - N / 'H' STO »
```

LRECT

Input: none

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

```
« A SUM »
```

RRECT

Input: none

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

```
« A H + SUM »
```

TRAP

Input: none

Effect: uses SUM to compute the trapezoidal rule approximation for the f, a, b, and n already stored.

```
« A SUM B F A F - 2 / H * + 'X' PURGE »
```

MID

Input: none

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

```
« A H 2 / + SUM »
```

SIMP

Input: none

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

```
« MID 2 * TRAP + 3 / »
```

F

Input: none

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

```
« 'X' STO EQ EVAL »
```

SUM

Input: none

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 + X H + 'X' STO NEXT H * » 'X'
PURGE »
```

We shall use the programs in INTG to obtain some approximations to the integral $\int_0^4 \frac{1}{1+x^3}\,dx$. Begin with '1/(1+X^3)' on level 1 and press 0 [SPC] 4 [FABST]. This stores the function and the lower and upper limits of integration. Now enter a value for n with NSTO. We start with n = 10 and use LRECT. Press 10 [NSTO] then [LRECT] to see the result 1.37470502665 on level 1. This is the left rectangle Riemann sum approximation for the function $f(x) = \frac{1}{1+x^3}$ on the interval [0,4], when the interval is partitioned into ten subintervals of equal length. Since $\frac{1}{1+x^3}$ is a decreasing function for all positive x, this sum will be larger than the actual integral. Press [RRECT] to see the result .9808588728. This is like the LRECT approximation except that the evaluation point in each subinterval is chosen to be the right end point. Since the function is decreasing, this sum will be less than the actual integral. Keying in 40 [NSTO] [LRECT] gives 1.22736347304 and [RRECT] gives 1.12890193457 when the interval [1,4] is partitioned into 40 subintervals. Repeating with 100 [NSTO] [LRECT] gives 1.19783377435 and [RRECT] gives 1.15844915896 for 100 subintervals. Thus $1.15844915896 \le \int_0^4 \frac{1}{1+x^3}\,dx \le 1.19783377435$. Using larger values of n will, of course, narrow the gap still further.

For an increasing or decreasing function, evaluating the function at the left and right end points of each subinterval has the advantage of bracketing the answer. This suggests that a better approximation may be obtained by using a simple average of these two approximations – the trapezoidal rule – or by evaluating the function at the midpoint of each subinterval – the midpoint rule.

To approximate $\int_0^4 \frac{1}{1+x^3}\,dx$ with the trapezoidal rule TRAP with with n = 10, 40 and 100, we have

10 NSTO TRAP gives 1.17778194973,

40 NSTO TRAP gives 1.17813270381, and

100 NSTO TRAP gives 1.7814146666.

Using the midpoint rule MID with n = 10, 40 and 100, we have

10 NSTO TRAP gives 1.17840171046,

40 NSTO TRAP gives 1.178148463398, and

100 NSTO TRAP gives 1.17814377875.

Notice that the midpoint approximations all agree to three decimal places, and the last two to five places.

Although the trapezoidal approximation is geometrically appealing, it can be shown with techniques that we shall not discuss here that the error in the approximation provided by the midpoint rule is roughly half the size of the error produced by the trapezoidal approximation. This suggests that a weighted average which assigns twice as much weight to the midpoint approximation as to the trapezoidal approximation would take advantage of the errors "cancelling" each other. This procedure is incorporated into the widely used formula known as Simpson's rule, and program SIMP does this.

To approximate $\int_0^4 \frac{1}{1+x^3}\,dx$ using Simpson's rule with n = 10, 40 and 100:

10 NSTO SIMP gives 1.18040998105,

40 NSTO SIMP gives 1.17814632836, and

100 NSTO SIMP gives 1.17814308787.

You may want to compare these with the values found earlier.

For this example, the approximations provided by Simpson's rule appear to be in close agreement and, indeed, Simpson's rule is the "best" of these techniques. As a rule of thumb, the error with the left or right-end point Riemann sum approximation is proportional to $h = \frac{b-a}{n}$ and the error with the midpoint Riemann sum or the trapezoidal approximation is proportional to h^2. But the error with the approximation provided by Simpson's rule is proportional to h^4.

The Numerical Integration Routine on the HP-48

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 Part II of the fundamental theorem of calculus cannot be applied. Simple examples are

(1) the standard normal integral $\int_0^z \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\, dx$ from probability theory;

(2) the period $T = \int_0^\alpha \frac{2\sqrt{2}\, dy}{\sqrt{\cos y - \cos \alpha}}$ of a simple pendulum released from an initial angle α; and

(3) the electrostatic potential V at a point P(x,y) due to a variable charge density $\lambda(s)$ applied along a straight wire over [-a, a]:

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

The HP-48 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 sampled points 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 of $E = 10^{-n}$
- STD specifies an error tolerance of $E = 10^{-11}$

For example, setting the numeric display to 5 FIX will specify an error tolerance of .00001. In general, the smaller the error tolerance, the longer the calculation time. When the calculation is finished, the uncertainty of the result is expressed in a variable IERR, where

$$IERR \leq E \int |f(x)| dx .$$

After specifying the tolerance, you enter the symbolic expression

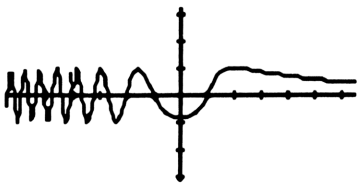
'∫ (*lower, upper, integrand, variable*)'

where *lower* and *upper* are the limits of integration, and *variable* specifies the variable of integration, e.g.,

'∫ (0, 4, 1/(1+x^3), x)'

If desired, you can use the Equation Writer to key in the integral; when you press ENTER, it will be placed on level 1 in the above algebraic syntax. The calculation is activated by pressing [→NUM]. With a tolerance of 10^{-5}, the integral $\int_0^4 \frac{1}{1+x^3}\,dx$ is calculated to be 1.17814. Enter the VAR menu and press [∫ERR] to see 10^{-5} as the uncertainty in this result.

As another example, we graph the function $f(x) = \begin{cases} \sin(x^2-1) & x < 1 \\ \sin(\pi/x) & 1 \le x \end{cases}$ with the default parameters and evaluate $\int_0^2 f(x)\,dx$ with error tolerance 10^{-5}. Enter the function as 'IFTE(X < 1, SIN(X^2 - 1), SIN(π/X))'. Graphing with the default parameters, we see:



To evaluate the integral, enter 0 and 2 onto the stack, recall the function to level 1 from EQ, enter 'X' and press the ∫ key to obtain the symbolic integral. Now press [→NUM] to evaluate the integral as .15511 with uncertainty 10^{-5}.

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 when students fail to focus on Part 1 of that theorem, which asserts that continuous functions have antiderivatives:

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

and focus instead on Part 2, 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 a concentration on Part 2 that many students come to view integration as simply a search for antiderivatives rather than as a limiting process. In retrospect, this has been a somewhat natural occurrence because, in the pedagogical process, teachers tend to seek out activities that students can *do* to help reinforce their understanding of the theory. And without computing power, what activity can they possibly do to reinforce Part 1?

But certainly, the HP-48 provides enough personal computing power for students to engage in activities that support Part 1 of the Fundamental Theorem (FTC). Equipped with the mid-point rule for approximating integrals, students can use it to construct a symbolic expression F(x) that approximates the antiderivative $\int_a^x f(x)dt$,

i.e., $F(x) \approx \int_a^x f(t)dt$. They can then graphically represent this approximation and its derivative F' and 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 FTC, but it also reinforces the desired goal of understanding the integral as a limit of approximating sums.

The algebraic formulation of the mid-point approximation using n subintervals of equal length is

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

When f is stored in memory as a user-defined function F, program FTC.1, given below, takes a and n as inputs and returns the algebraic expression for the mid-point approximation. FTC.1 calls upon subroutine SUMF , also given below, to construct the actual expression following the summation symbol.

FTC.1

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 the algebraic expression $\sum_{i=1}^{n} f\left[a + (2i-1)\frac{(x-a)}{2n}\right]\frac{(x-a)}{n}$.

```
« 'N' STO 'A' STO 0 1 N FOR I I SUMF + NEXT COLCT { A N }
PURGE »
```

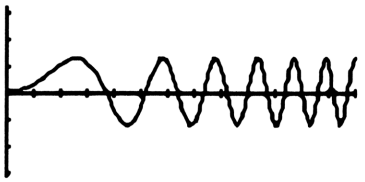
SUMF

Inputs: none

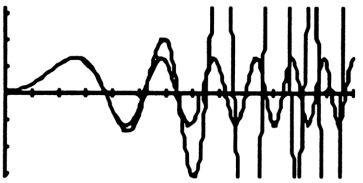
Effect: a utility program used by FTC.1 to construct the desired algebraic summand.

```
« → I ' F (A + (X–A) * (2*I–1) / 2 / N) * (X–A) / N ' »
```

As an example, we shall use $\int_0^{6.28} \sin x^2\,dx$, which has no elementary antiderivative. Begin by building a user-defined function F for $\sin x^2$ and then graphing $\sin x^2$ using XRNG : 0 6.28 and YRNG : -2.5 2.5, to see

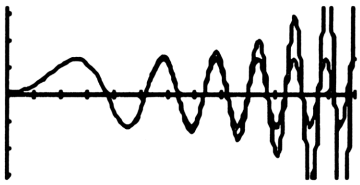


Now execute program FCT.1 with inputs 0 and 5, for a and n respectively. With only 5 approximating rectangles we do not expect a good approximation. When an algebraic expression appears, take its derivative with 'X' ENTER ∂, then overdraw the graph of $\sin x^2$ with this derivative to see:



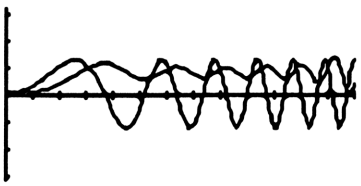
Not surprisingly, the 5 rectangle approximation becomes increasingly worse as the oscillations in $\sin x^2$ increase.

Now ERASE, redraw sin x^2 and run FTC.1 with inputs 0 and 13, for n = 13 rectangles. Overdraw the graph of sin x^2 with the graph of the derivative of the expression to see a much improved approximation:



(Be patient: the derivative and its graph unfold slowly.) Note that although we are using only 13 rectangles, the approximation is dramatically improved.

Although we did not draw the graph of the approximation to the antiderivative (simply to keep from having a too-cluttered screen), students should overdraw the graph of sin x^2 with the approximating antiderivative for n = 13. As with the previous graph, they will be viewing a scene that has been denied to students of calculus for centuries:



EXERCISES 3.1

1. Evaluate $\int_0^1 \frac{2}{1+x^2}\,dx$ by hand. Now approximate this integral as follows:

   (a) using LRECT and RRECT with n = 50, 100 and 200.

   (b) using TRAP with n = 50, 100 and 200

(c) using MID with n = 50, 100 and 200,

(d) using SIMP with n = 50, 100 and 200

(e) using the built-in numerical integration routine in STD mode.

2. Repeat parts (a) through (e) of Exercise 1 for $\int_0^3 e^{-x^2}\,dx$. For part (e) use an error tolerance of .000001.

3. Approximate the arc length of y = cos x from x = 0 to x = 2.

(a) using Simpson's rule with n = 100

(b) using the built-in numerical integration program with error tolerance .000001.

4. Graph FLOOR(X) with the default x range, the y range set as $-.1 \le y \le 6.2$ and the disconnected mode (FLOOR is on the MTH REAL menu). Approximate $\int_1^4 \text{FLOOR}(X)\,dx$ with the LRECT, RRECT and MID programs for n = 10, 50 and 100. Evaluate this integral with the built-in numerical integration program. Now evaluate this integral using only geometry. At the right hand end point, x = 4, the function jumps in value; why does this not affect the integral?

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

evaluate $\int_0^3 f(x)\,dx$ with the built-in numerical integration program.

6. (a) Graph the function y = xcos x in disconnected mode using XRNG : 0 6.28 and YRNG : -6.3 1.

(b) Overdraw the approximation to the antiderivative $\int_0^x t\cos t\, dt$ obtained by using the mid-point rule with n= 5 rectangles.

(c) Change to connected mode and overdraw the derivative of the function graphed in (b). How closely does it approximate xcos x ?

(d) Use integration by parts to obtain an elementary antiderivative for y = xcos x; choose an initial condition so that your antiderivative will pass through the origin. Now overdraw the above graphs with this antiderivative. How closely does the approximation in (b) match this antiderivative?

4. INFINITE SERIES

The approximation of functions by polynomials is an important topic in elementary calculus. From the simple notion of linear approximations by tangent lines to the subtleties of higher-order approximations by polynomials whose derivatives mimic the function's behavior, students are ultimately led to consider power series representations. With the HP-48 they can effectively exploit the graphical representation of the partial sums of these series as Taylor polynomials, and witness the dynamics of numerical convergence at the end points of the interval of convergence.

Taylor Polynomials

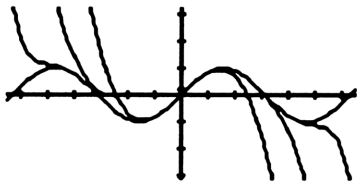
The approximation of functions by series representations parallels the approximation of numbers by decimal representations. For example, in a non-symbolic setting, we must approximate π to an appropriate number of decimal places, i.e., 3.14, 3.14159, 3.14159265, etc. A selection is made on the level of accuracy, but this selection is balanced with the computational complications that we are willing to

tolerate. In ancient times, mathematicians failed to realize that an infinite sequence of finite approximations could converge to a finite number and thus the inadequacy of their mathematics yielded the famous Zeno paradox. They could not understand that $1/2 + 1/4 + 1/8 + 1/16 + ... + 1/2^n + ...$ represented 1 and discussed at length how an arrow could never arrive "because it had to first get half-way there, then half-way again, then half-way again....., etc. Their mathematics generated a contradiction with what was clearly reality, which showed a need for further improvement in their mathematics. This eventually led to an understanding of infinite decimal representations of numbers and finite approximations to their values. That is exactly what we now do with functions, except that the "decimal" entries are polynomials of higher and higher degrees. Just as before, we take the approximations to the heights that we need, balanced with the computational complications. And again, just as we did with π, we look at a function in its symbolic form, e.g., sin x, and when need be, look at its polynomial approximations: $1 - x$, $1 - x + x^3/3!$, etc.

The HP-48G/GX will find Taylor polynomials about x = 0 for any function that it can differentiate, and it is easy to write a short program that extends this capability to the more general case of polynomials about x = a. 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 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.

For instance, to efficiently graph y = sin x and its Taylor polynomials P_3, P_7 and P_{11} of degrees 3, 7 and 11, begin with 'SIN(X)' on level 1 and press ENTER three times to make three additional copies. Build the list {'SIN(X)'} by pressing the Δ followed by →LIST ENTER. SWAP levels 1 and 2, enter 'X' and 3, then

press [TAYLR] to build $P_3(x)$: 'X - 1/3!* X^3'. Insert this as the second element of the list with [+]. 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 graph it with the default viewing screen to see



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

To find Taylor polynomials centered about a point x = c, you can use the next program. Make sure the independent variable is set to X before using the program.

TAY.C

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

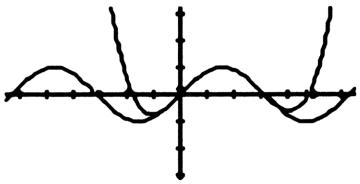
Level 2: the order n of the desired Taylor polynomial.

Level 1: the new center point, c.

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

```
« → N C « 'Y' C + 'X' STO EVAL 'Y' N TAYLR 'X' PURGE 'X'
C - 'Y' STO EVAL 'Y' PURGE » »
```

For example, to find the fourth order Taylor polynomial for sin x, centered about x = 2, put 'SIN(X)' on the stack, then enter 4 and 2 and press TAY.C 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.) Graph the list containing sin x and this polynomial with the default parameters to see:



Notice that the graphs coincide from about x = -.5 to near x = 3.5, that is, on an interval centered about x = 2.

Although TAY.C does the obvious by making a change of variables X = Y + C to translate the center of the Taylor series expansion from the origin to x = c, you should note that the symbolic computations involved in calculating higher order Taylor polynomials centered away form the origin are substantial. Thus, as a symbolic processor, you may sometimes find the HP-48 not quite up to the task of finding the Taylor polynomials that you desire if you use TAY.C. For example, the HP-48G runs out of memory (32K RAM) before it can produce the Taylor polynomial of order 7 for $f(x) = x^{-1}$ centered about x = 2 and the HP-48GX (with 128K RAM) requires about 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 ↓MATCH and | commands to rearrange the symbolic computations. With it, you can produce the Taylor polynomial of order 7 for $f(x) = x^{-1}$ centered about x = 2 in 18 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 c.

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

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

Sequences and Series

The analysis of numerical series requires a clear understanding of the size of very large and very small numbers, and the HP-48G/GX is an ideal instrument to help expand this understanding. The next two programs, FSHO and HGFSHO, can be used for exploratory work with numerical sequences.

FSHO

Input: Level 3: an expression for a_K, in terms of the variable K

Level 2: a starting value for K

Level 1: an ending value for K

Effect: dynamically displays the successive terms of the sequence $\{a_K\}$ from the starting value to the ending value beneath the index K.

```
« → F B N « B N FOR J J 'K' STO F EVAL DUP CLLCD K 1
DISP 3 DISP 1 WAIT NEXT » 'K' PURGE »
```

For example, to see the first 25 terms of the sequence $\left\{ \frac{K^5}{K!} \right\}$, enter 'K^5/K!', then 1, 25 and press [FSHO]. The index K appears on line 1 of the display screen and the terms of $\frac{K^5}{K!}$ on line 3.

HGFSHO

Input: Level 5: an expression for h_K, in terms of the variable K

Level 4: an expression for g_K, in terms of the variable K

Level 3: an expression for f_K, in terms of the variable K

Level 2: a starting value for K

Level 1: an ending value for K

Effect: dynamically displays the successive terms of the three sequences, $\{h_k\}$, $\{g_k\}$ and $\{f_k\}$ from the starting value to the ending value, beneath the index K.

```
« → H G F B N « B N FOR J J 'K' STO CLLCD K DUP 1 DISP H
EVAL DUP 3 DISP G EVAL DUP 5 DISP F EVAL DUP 7 DISP 1
WAIT NEXT » 'K' PURGE »
```

To experience, for example, the differences in the growth rates of the sequences $\{K^5\}$, $\{K!\}$ and $\{K^k\}$ we may examine the corresponding terms between K = 1 and K = 20. Thus, enter 'K^5', 'K!', 'K^K', 1, 20 and press HGFSHO. Watch carefully! Notice the subtantial growth that K^K has over K!.

Another very effective exploration involving very large numbers uses the units feature of the HP-48G/GX and can be turned into an interesting class competition to see who can generate the largest number with a *verbal* description. For example, adding one cubic light year in stack level 2 to one teaspoon in stack level 1 will give the result in teaspoons, i.e., the number of teaspoons in a cubic light year -- a very large number! This is easily accomplished by opening the UNITS Catalogue menu, then the LENGTH submenu, entering one light year , 1_lyr, three times on the stack

and multiplying to get a cubic light year, 1_lyr^3. Now return to the UNITS Catalog menu, open the VOL submenu, enter the other volume measure, one teaspoon 1_tsp, and add to get 1.71788573061E53 tsp, the number of teaspoons in a cubic light year. This exercise can be extended to estimating the answer in terms of factorials, and a quick search will locate the above number between 43! and 44!. In many successive calculus classes, students have been unable to generate *with words* any number that is larger than 100!. This gives them real insight into the study of series that have factorials in the denominators of the terms. They really begin to sense the awesome nature of n! and the effects that it will have on the terms of the series.

For a numerical series which is known to converge, you can often use program P.SUM to calculate a twelve-digit approximation to the sum by evaluating partial sums of the series for increasing values of n until agreement is reached for two successive values of n. Program INFSUM, given later, carries out these calculations automatically and returns the sum of the series, accurate to the twelve-digit display, along with the value of n at which agreement of successive partial sums was reached. *Before beginning either of these procedures, you should already have determined that the series in question converges* by applying one of the standard convergence tests.

P.SUM

Input: level 3: an expression for a_K, in terms of the variable K
level 2: a starting value for K, 0 or 1
level 1: an ending value for K

Effect: returns the partial sum $\sum_{K=start}^{n} a_K$ on level 2 and the expression for a_K on level 1. Also, dynamically shows the summation.

```
« → F B N « 0 B N FOR J J 'K' STO F EVAL + DUP 3 DISP
NEXT F » 'K' PURGE »
```

For example, the infinite series $\sum_{k=1}^{\infty}(1/k^4)$ is known to converge by the integral test. Put '1/K^4' on level 1 and press 1 [SPC] 100 [P.SUM]. After dynamically showing the summation process, the program will show 1.08232290538 on level 2 and '1/K^4' on level 1. To calculate the 250th partial sum, simply press 1 [SPC] 250 [P.SUM]. You will see 1.08232321257 returned to level 2. Finally, to obtain the 500th partial sum, press 1 [SPC] 500 [P.SUM] and see 1.08232323119. Since these last two sums agree to 7 decimal places, the sum of this series is 1.0823232 to 7 decimal places.

The series $\sum_{k=1}^{\infty}(-1)^{k+1}(1/k^6)$ converges by the alternating series test. Since it is an alternating series, we know that the error made in using any partial sum S_k 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_{k+1}. For twelve place accuracy, we must take n large enough so that $1/(n+1)^6 < 5\times10^{-13}$. Using the HP-48 for the calculation, we find that $1/113^6$ is approximately 4.8×10^{-13}. Thus, calculating the 112th partial sum with P.SUM we obtain .985551091299 as our estimate of the sum, to 12 decimal places. (You may calculate the 113th partial sum to see if you get agreement.)

Program INFSUM , given below, also 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 with initial index k = 1, so you will have to make adjustments for series that do not start there. As before, the program should only be used with series that are *known to converge.*

INFSUM

Input: the summand, 'f(k)', for the infinite series $\sum_{k=1}^{\infty} f(k)$

Effect: calculates partial sums $\sum_{k=1}^{n} f(k)$ until two successive sums agree, displays the last partial sum and the value of n at which agreement was reached.

```
« 'F' STO 1 'K' STO F EVAL 2 'K' STO DO DUP F EVAL + DUP 3
DISP 1 'K' STO+ SWAP UNTIL OVER == END K 1 - { F K }
PURGE »
```

As an example, we know that the series $\sum_{k=0}^{\infty} 10^k/k!$ converges by the ratio test. Put '10^K/K!' on level 1 and press **INFSU** to see 22025.4657948 on level 2 and 39 on level 1. Although the partial sums are large in value, agreement to 12 figures is reached quickly, at n = 39. The number 22025.4657948 is the sum of the series starting at k = 1, so we must add $\frac{10^0}{0!} = 1$ to get the sum starting at k = 0. Thus $\sum_{k=0}^{\infty} 10^k/k! =$ 22026.4657948, correct to 12 figures.

EXERCISES 4.1

1. Find the Taylor polynomials $P_6(x)$ and $P_{10}(x)$ for $f(x) = e^x$ about a = 0, and graph f and the two polynomials on the same axes. Make a table of values for all three functions using x = .1, .2, .6, 1 and 1.5.

2. Use program TAY.A to find the Taylor polynomials $P_5(x)$ and $P_9(x)$ for $f(x) = \sin 2x - 2\sin x$ about $a = 2$ and graph f and the two polynomials on the same set of axes. Use XRNG : -2 6 and YRNG : -3.1 3.2.

3. Find the Taylor polynomials $P_6(x)$ and $P_{10}(x)$ for $f(x) = e^{-x^2}$ at $a = 0$ and graph these two polynomials and f on the same axes.

4. (a) Verify, analytically, that each of the following sequences has a limit and find the limit:

$$\left\{\left(\frac{K}{K+1}\right)^K\right\} \quad \left\{\left(\frac{1}{1+1/K}\right)^K\right\}$$

(b) Investigate, numerically, the limits of these sequences with FSHO. Let K go from 10^6 to $10^6 + 10$, 10^8 to $10^8 + 10$, ..., 10^{11} to $10^{11} + 10$.

(c) What happens if you let K go from 10^{12} to $10^{12} + 10$? Why?

5. Prove that the infinite series $\sum_{k=1}^{\infty} (k + 1)/k^{10}$ converges. Then use program P.SUM to find the partial sums for n = 10, 50 and 100. What is the sum of this series to 12 places?

6. Prove that the infinite series $\sum_{k=1}^{\infty} (-1)^{k+1}(k!)$ converges. Then, find a value of n for which S_n will approximate the sum, correct to 12 places. Use P.SUM to calculate this S_n.

7. Show that each of the following series converges and find the sums using program INFSUM .

a. $\sum_{k=1}^{\infty} 1/k^3$. (slow convergence)

b. $\sum_{k=0}^{\infty} 1/k!$. (rapid convergence)

c. $\sum_{k=1}^{\infty} (-1)^{k+1} k^2 / 2^k$.

d. $\sum_{k=1}^{\infty} 100^k / k!$

8. (a) What will happen if you use INFSUM on $\sum_{k=1}^{\infty} 1/k$? Try it and see for yourself. (You may interrupt the program at any time by pressing [ON]; when the program is interrupted, it leaves [F] and [K] on your VAR menu.)

   (b) Prove that $\sum_{k=1}^{\infty} \frac{k!}{(k+2)!}$ converges. Now apply program INFSUM and watch what happens. Explain what is taking place here.

REFERENCES

1. Kenelly, John W. and Donald R. LaTorre, *Calculus with the HP-48G/GX*, Saunders College Publishing, 1994, Philadelphia, PA.

HP-48G/GX Calculator Enhancement

for

Multivariable Calculus

James A. Reneke

The chapter is divided into three sections. The first treats applications of the HP-48G/GX to curves and surfaces. The material exploits the plotting utilities for conic sections and parametric curves. The second section considers maximum/minimum problems, with an emphasis on the role of level curves, and multiple integrals. The geometry of constrained optimization problems is explored with examples. The final section discusses line integrals and makes extensive use of the symbolic manipulation capability of the calculator. Thus the graphical, numerical and symbolic capabilities of the HP-48G/GX are all exercised in the context of multivariable calculus.

SECTION 1. CURVES AND SURFACES

1.1 Curves, Surfaces and Functions

In one dimensional calculus we apply the methods of calculus to problems associated with functions. For instance, we study maximum/minimum problems for functions and the problem of computing the average value of a function. We usually do not distinguish between a function and its graph nor elaborate upon the application of calculus to implicitly defined curves, for instance to the conic sections. That is, implicit differentiation is introduced as a natural extension of

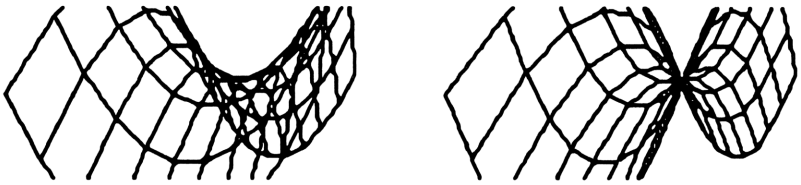
differentiation of functions even though the relationship of an equation to an implicitly defined curve is fundamentally different from the relationship of a function and its graph. A consequence of this blurring of distinctions is a heightened emphasis on functions as opposed to geometric curves.

In multidimensional calculus the more fundamental objects of our study are geometric curves and surfaces. Many of the concepts to be introduced are inherently geometric and functions can be used in several different ways for the study of these concepts, although some particular way will likely be most useful in any given discussion. Thus the task now is to shift our attention from functions to curves and surfaces, solving some of the same problems considered in one dimensional calculus.

However, we also take up problems in multidimensional calculus that do not have a one dimensional counterpart. Functions of several variables have more than one kind of derivative, for instance, directional derivatives. And line and surface integrals are not just multidimensional versions of the Riemann integral. Obviously, in thinking about curves and surfaces we are not going to abandon what we know of differentiating and integrating functions. So, how can we use functions to describe curves and surfaces in higher dimensional spaces? How is calculus applied to functions to study the curves and surfaces they describe? Using the HP-48G/GX effectively for multivariable calculus requires clear answers to these questions.

Geometry in spaces of dimension higher than three uses the language of three dimensions. Since we have no way to visualize geometric objects in higher dimensions, spending time to understand three dimensions is important for building intuition for higher dimensions. The HP-48G/GX graphics calculator provides useful tools for exploring curves and surfaces in 3-space. We will begin with the WIREFRAME plot type which produces pictures of the graphs of functions of two variables. This will give us some concrete examples to use in a more abstract discussion of the relation of functions and geometric curves and surfaces.

The surface $S = \{(x, y, z) \mid z = x^2 - y^2\}$ is the graph of $z = f(x, y) = x^2 - y^2$. Two views of S are shown below.



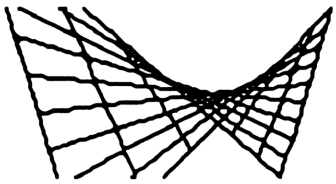
Exercises. Use the WIREFRAME Plot Screen to do the following exercises.

1) Reproduce the pictures given above. Open the Main Plot Screen with [↱] PLOT, choose WIREFRAME plot type and enter 'X^2 - Y^2' as the current equation. Choose 'X' and 'Y' as the independent and dependent variables, respectively. Choose -2 and 2 for X-left and X-right, -2 and 2 for Y-near and Y-far, -2 and 2 for Z-low and Z-high so that the View Volume surrounds the origin. Choose 3, -6, 1 for Xe, Ye, and Ze to give a distant oblique view of the graph. Choose 6, -6, 1 to give an alternate view.

2) Crucial to producing a recognizable picture for the surface in the previous exercise is the choice of View Volume and View Point. Enter 'X^2 - Y^2 + 4*X + 4*Y' as the current equation. Without changing the View Volume or View Point from the previous exercise draw the figure. Of course, the critical point (-2, 2) is not centered in the View Volume so we are not looking where the action is. Choose -4 and 0 for X-left and X-right, 0 and 4 for Y-near and Y-far, -2 and 2 for Z-low and Z-high so that the View Volume surrounds the critical point. The View Point has to be chosen relative to the new View Volume. Try 4, -6, 1 for Xe, Ye, and Ze.

3) Enter 'X*Y - 2*X + 2*Y - 4' as the current equation. The critical point is

   (-2, 2). Try the previous View Volume and View Point. What happened? Try producing the following picture.



The lesson here is about choosing the View Point; in trying various possiblities one gets a real sense of the surface.

Basic definitions with examples. A *function* is best understood as a collection of ordered pairs no two of which have the same first term. In the one dimensional calculus, functions are ordered pairs of real numbers that can be displayed graphically as subsets of the Cartesian plane. This graphical representation of the function is called the *graph of the function.* For real valued functions of more than one variable, the first members of the ordered pairs (elements of the domain of the function) will themselves be n-tuples. In the case of real valued functions of two variables, the first members will be 2-tuples or ordered pairs, and the graphical representation of such functions will be in 3-space. A surface S is said to be given *explicitly* by a function f, usually from R^n to R, provided S is the graph of f.

a) From the one-dimensional calculus consider the function $f(x) = x^2 + x + 1$. The curve $C = \{(x, y) \mid y = x^2 + x + 1\}$, the graph of f, is said to be given explicitly by f.

b) The surface $S = \{(x, y, z) \mid z = x^2 - y^2\}$ is given explicitly by the function $f(x, y) = x^2 - y^2$.

Exercises.

1) Produce the surface $z = \sin(x + y)$ using the View Volume [0, 6.28] by [0, 3.14] by [-2, 2]. Start with 6, -6, 4 for Xe, Ye, and Ze. Is there another choice for Xe, Ye, and Ze that improves the picture?

2) Try some of the other explicit quadratic surfaces from your calculus text.

Among the conic sections, the ellipses and hyperbolas are usually defined implicitly by equations. For example, the unit circle is given by the equation $x^2 + y^2 = 1$. Note that the graph of a circle cannot be the graph of a function, because there are vertical lines that intersect a circle in more than one point. A surface S is said to be given *implicitly* by a function f, usually from R^n into R, provided S is a level set of f, i. e., there is a number k such that $S = \{ x \mid f(x) = k\}$. Note that S is a subset of the domain of f.

a) The unit circle C given by $x^2 + y^2 = 1$ is a level set of the function $f(x, y) = x^2 + y^2$, i.e., $C = \{(x, y) \mid f(x, y) = x^2 + y^2 = 1\}$.

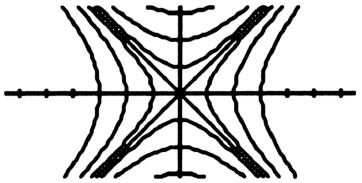
b) A portion of the implicitly defined surface $S = \{(x, y, z) \mid x^2 + y^2 + z^2 = 1\}$ is given below.

This picture was produced with the WIREFRAME Plot Screen by solving $x^2 + y^2 + z^2 = 1$ explicitly for z. Of course, each point (x, y) produces two values of z and we used only the nonnegative values of z. Thus we can use software for producing explicit surfaces to represent portions of implicitly defined surfaces.

Exercise. Use the WIREFRAME Plot Screen to produce a portion of some of the implicitly defined quadratic surfaces in your calculus text, for instance, $x^2 + y^2 + 4z^2 = 1$.

Implicitly defined curves can also be used to study explicitly defined surfaces. Given an explicit surface $z = f(x, y)$ we produce the implicitly defined level curves $k = f(x, y)$. In this case, the level curves of a surface are a two dimensional

representation of a three dimensional surface. Consider the level curve representation of the surface $z = x^2 - y^2$ given below.



Later, we will use the conic graphing ability of the HP-48G/GX to produce level curves for quadratic surfaces like the one shown above.

A curve C (sometimes a surface) is said to be given *parametrically* by a function f, usually from some subset U of R to R^n, provided C is the image of U under f, i.e., C = f(U). Note that C is a subset of the range of f.

a) The straight line through $P(x_0, y_0, z_0)$ and $Q(x_1, y_1, z_1)$ is given parametrically by the set of functions $\begin{cases} x = x_0 + t(x_1 - x_0) \\ y = y_0 + t(y_1 - y_0) \\ z = z_0 + t(z_1 - z_0) \end{cases}$, or in vector form by $x = (x_0, y_0, z_0) + t(x_0 - x_1, y_0 - y_1, z_0 - z_1)$.

b) The plane consisting of all linear combinations of two linearly independent vectors u and v is given parametrically by $f(a, b) = au + bv$, (a, b) in $R \times R$.

Some standard representations

a) Lines (explicit: $y = mx + b$; implicit: $Ax + By + C = 0$; parametric: $r(t) = u + tv$, t a real number)

b) Conic sections

Implicit:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \quad \text{and} \quad \frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$$

Parametric:

$$\begin{cases} x = a\cos\theta \\ y = b\sin\theta \end{cases}, 0 \le \theta \le 2\pi \quad \text{and} \quad \begin{cases} x = a\cosh t = \frac{a}{2}\left(e^t + e^{-t}\right) \\ y = b\sinh t = \frac{b}{2}\left(e^t - e^{-t}\right) \end{cases}, -\infty < t < \infty$$

Shifting the representation

Many times, problems come with some "natural" representation of the curve or surface. In most applications, including the HP-48G/GX, a certain representation is required; for instance, the WIREFRAME Plot Type requires an explicit representation of the surface. The first step in these problems becomes that of shifting the given representation of the curve or surface to the representation appropriate to the application.

a) A curve C given explicitly by $y = f(x)$, x in D, can be described implicitly by $C = \{ (x, y) \mid y - f(x) = 0\}$ or parametrically by

$$\begin{cases} x = x \\ y = f(x), \ x \text{ in } D \end{cases}$$

b) A curve C given implicitly by $f(x, y) = 0$ might not have have an explicit representation; for instance, the circle $x^2 + y^2 = 1$. Even so, we can frequently give a branch of the curve an explicit representation; for example, as the upper semicircle $y = \sqrt{1 - x^2}$, $-1 \le x \le 1$.

c) Sometimes a curve C originally given parametrically can also be represented implicitly by "eliminating the parameter". See the conic sections above.

Exercises.

1) Find both parametric and explicit representations of the line segment between P = (1, 1) and Q = (2, -3).

2) Find both implicit and parametric representations of the ellipse with vertices (-1, 0), (1, 0), (0, -2) and (0, 2).

3) Find both explicit and implicit representations of the plane that contain (1, 0, 0), (0, 1, 0) and (0, 0, 1).

1.2 Three Dimensional Parametric Curves

We want to use the 2D parametric plot capability of the HP-48G/GX to produce some 3D plots. We will be working with the Stack Interface, i.e., from the ↰ PLOT menu. Our approach to plotting 3D parametric curves will be to create a general function in terms of X, Y and Z, called PARA3D. We can obtain the plot of any 3D parametric curve by plotting PARA3D after specifying X, Y and Z. To accomplish this enter

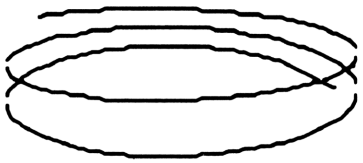
'(X - Xe) / (Y - Ye) + i * (Z - Ze) / (Y - Ye)'

'PARA3D' STO

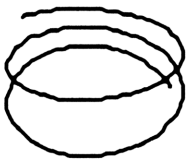
Note that we are not writing a program, merely plotting a "generic" parametric curve that will be completely specified only when 'X', 'Y' and 'Z' are specified. To make PARA3D the current equation enter 'PARA3D' 'EQ' STO.

Examples. Store 0, -2, and 25 as 'Xe', 'Ye', and 'Ze', respectively.

1) A spiral wrapped around a cylinder: r = (cos t, sin t, t), $0 \le t \le 5\pi$. Enter 'COS(T)' 'X' STO 'SIN(T)' 'Y' STO 'T' 'Z' STO. Change PYTPE to PARAMETRIC, open PPAR and change INDEP to {T 0 15.71}. Then AUTO ERASE DRAW produces the following.



We can change the x scale with ZOOM. Open ZOOM and set the H-factor as 2 and the V-factor as 1 in ZFACT. Then press ZOUT to produce



The unwanted axis was eliminated with ERASE DRAW.

2) A spiral wrapped around a cone: r = ((1 - t/6π) cos t, (1 - t/6π) sin t, t), $0 \le t \le 6\pi$.

Exercises. Plot the following 3D parametric curves:

1) r = (1 - t, 2t + 1, t + 1), $-3 \le t \le 3$. Hint: If you wish to speed up the plotting, change the resolution with 1 RES.

2) $r = (t, 0, t^2)$, $0 \le t \le 2$. Try .1 RES.

3) $r = (\cos t, \sin t, \cos^2 t - \sin^2 t)$, $0 \le t \le 2\pi$. Hint: Use AUTO and then rescale with ZOUT assuming H-factor is 2 and V-factor is 1. Try to understand the 3D figure by plotting it with different choices of the viewing point.

1.3 Level Curves of Quadratic Surfaces

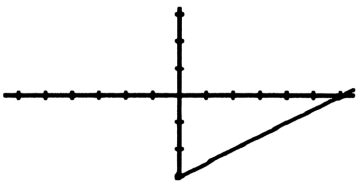
Continuing to use the Stack Interface, begin by going to the HP-48G/GX's PLOT menu, open PPAR and press RESET. Do ↰ PLOT again, open the PTYPE submenu and press the menu key CONIC.

Example. Sketch $4y^2 - 8x - x^2 + 40y + 85 = 0$.

Enter

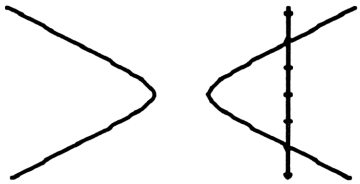
'4 * Y ^ 2 - 8 * X - X ^ 2 + 40 * Y + 85' 'EQ' STO

ERASE DRAX DRAW

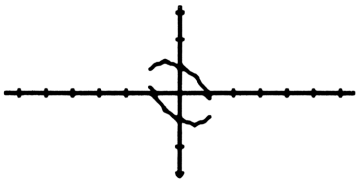


The resulting sketch isn't very satisfying. How can we improve it? Completing squares, our equation becomes $(x + 4)^2 - 4(y + 5)^2 = 1$, so the figure must be a hyperbola with center (-4, -5). Set the center in PPAR and redraw as follows:

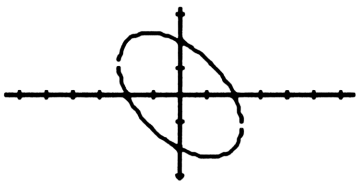
(-4, -5) CENT ERASE DRAX DRAW



Example. Graph $x^2 + xy + y^2 - 1 = 0$ using the default graphing parameters. 'X ^ 2 + X * Y + Y ^ 2 - 1' 'EQ' STO ERASE DRAX DRAW results in a graph like the following.

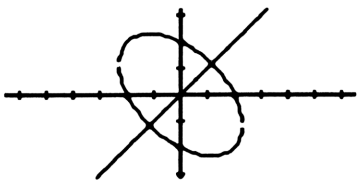


The picture can be improved with ZOOM. Set the H-factor as 2 and V-factor as 2 in ZFACT, then press ZIN.



Note. ON takes us out of the PICTURE environment and the PICTURE key returns us to it.

Example. 'Y - X' 'EQ' STO DRAW overdraws the last graph with a line:

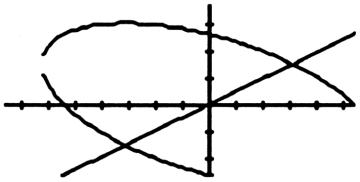


A better way to produce this graph is with {'X ^ 2 + X * Y + Y ^ 2 - 1' 'X - Y'} 'EQ' STO ERASE DRAX DRAW.

Example. The upper region bounded by the curves can be regraphed with

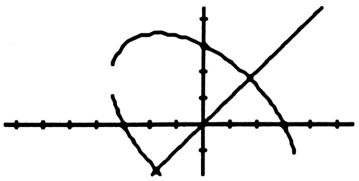
(-1.4, -1) PMIN (1, 1.4) PMAX ERASE DRAX DRAW

The commands PMIN and PMAX must be typed in; they capture the coordinates of the lower left and upper right corners of the plotting screen.



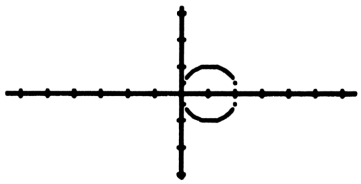
This method of choosing the graphing window captures the region of interest but can distort the picture. A better way is illustrated by the following example.

Example. To return to the original parameters press RESET DRAX DRAW. Then to enlarge the upper bounded region, move the cursor to the approximate center of the region of interest press ENTER (to place the coordinates of the cursor on the stack) and then CENT. Now use ZIN with H-factor 3 and V-factor 3 to produce something like the following:

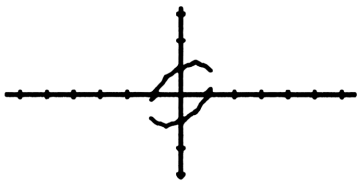


Examples. Two of the standard figures are graphed below. Try to improve the pictures by moving the center and using ZIN or ZOUT.

1) $x^2 + y^2 - 2x = 0$



2) $x^2 - xy + y^2 - 1 = 0$



Exercises

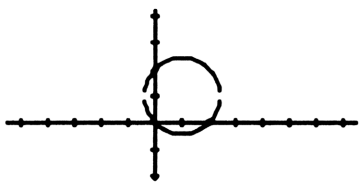
1) Redo some of the other standard figures: $x + y + 1 = 0$, $y^2 + x - 1 = 0$, $x^2 + y^2 - 1 = 0$ and $2x^2 - y^2 - 2 = 0$.

2) Produce a graph containing both $x + y - 1 = 0$ and $y = 2 - x^2$ on the same axis.

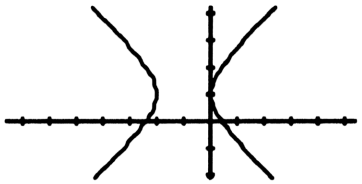
All second degree polynomials $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$ (allowing for degenerate figures) define conic sections.

Examples. Use the default settings to produce the following graphs. Notice the center is not always at (0, 0).

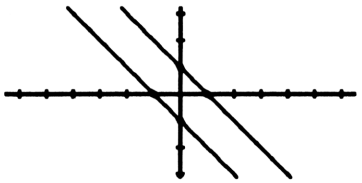
1) $x^2 + y^2 - 2x - 2y = 0$



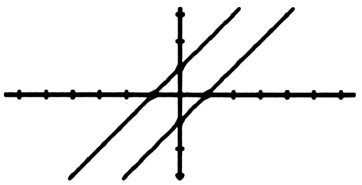
2) $x^2 - y^2 + 2x + 2y - 1 = 0$



3) $x^2 + 2xy + y^2 - 1 = 0$



4) $x^2 - 2xy + y^2 - 1 = 0$



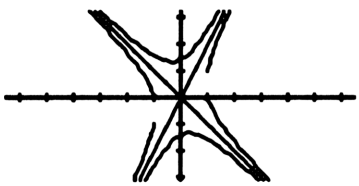
The level curves of explicitly defined quadratic surfaces of the form $z = Ax^2 + Bxy + Cy^2 + Dx + Ey + F$ are conic sections.

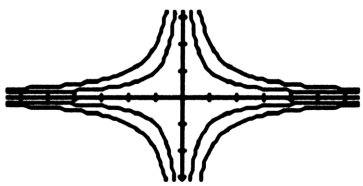
Exercises. Match the following surfaces with the sets of level curves given below:

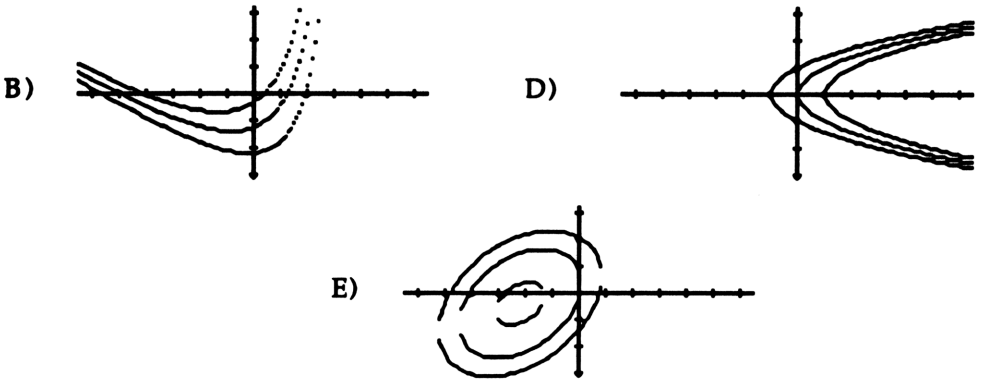
1) $z = 2x^2 + xy - y^2$ 2) $z = y^2 - x$ 3) $z = xy$

4) $z = 16x^2 + 24xy + 60x - 80y - 100$

5) $z = 29x^2 - 24xy + 36y^2 + 118x - 24y - 55$

A) 

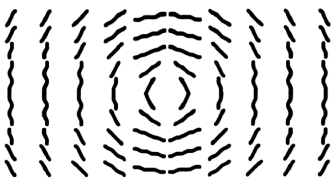
C) 



1.4 Pseudolevel Curves of Explicitly Defined 3D Surfaces

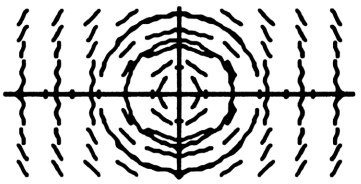
Strictly speaking, finding the level curves of a general, explicitly defined, 3D surface z = f(x, y) with the HP-48G/GX requires material from differential equations.

Example. Go to the Screen Interface with [↱] PLOT and choose Ps-Contour. Store ' X^2 + Y^2' as the current equation in EQ. Open OPTS and set -4, 4 for X-left and X-right and -2, 2 for Y-near and Y-far. Use ten steps for both the independent variable X and 'dependent' variable Y. then ERASE DRAW produces the following:



We can explore the relationship of this graph to the level curves of the quadratic surface $z = x^2 + y^2$ by proceeding as follows. Move the cursor away from the origin and enter the coordinates of the cursor onto the stack with ENTER. Now press ON twice to return to the stack. Produce the z value with OBJ→ x^2 SWAP x^2 + 'Z' STO. Now overlay the level curve produced with CONIC as follows. Using the

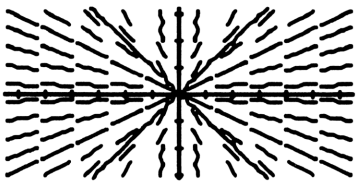
Stack Interface choose CONIC. Open PPAR and set Xrng and Yrng to agree with X-left, etc. Change EQ to 'X ^ 2 + Y ^ 2 - Z' and press DRAW. Drawing the conic sections for a couple of different values for Z produces



Notice that the level curves do not cross any of the line segments.

Exercise 1. Work the last example with the explicitly defined surface $z = x^2 - y^2$.

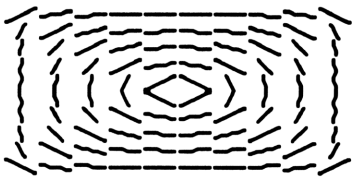
Exercise 2. A useful example to explore is $z = \frac{2xy}{x^2 + y^2}$. Start by storing '2 * X * Y / (X^2 + Y^2)' in EQ. Repeat the steps using Ps-Contour outlined in the example. In order to use CONIC store ' Z * (X^2 + Y^2) - 2 * X * Y ' in EQ. See if you can produce something like the following. Save the z values on the stack.



What's happening at the origin?

Of course, Ps-Contour works for surfaces more general than the quadratic surfaces. The problem is that the user has to construct mentally the actual level curves using the picture produced by Ps-Contour.

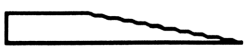
Example. Store 'SIN(X) * SIN(Y)' as the current equation. Choose 0, 3.14 for X-left and X-right and 0, 3.14 for Y-near and Y-far. The Ps-Contour Plot type produces



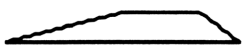
Can you visualize the level curves of this surface?

1.5 Explicitly Defined 3D Surfaces

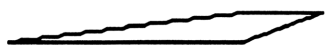
Before returning to WIREFRAME we will make use of the Pr-Surface plot type to explore further the interaction of View Point with the pictures produced. Go to the Screen Interface with [↱] PLOT and choose Pr-Surface as the plot type. Enter the current equation as { U V 0 }, U for INDEP, V for DEPND, and use 2 for STEPS in both U and V. Open OPTS and set the View Volume to -1, 2; -1, 2; and -2, 2. To set the ranges of U and V you must use the Stack Interface. Set INDEP as {U 0 1} and DEPN as {V 0 1}. Return to the Screen Interface and try various choices for the View Point. We are looking at a square in the xy-plane with vertices (0, 0, 0) and (1, 1, 0) from various view points.



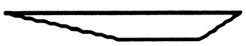
Xe: -1, Ye: -2, Ze: 1



Xe: 0.5, Ye: -2, Ze: 1



Xe: 2, Ye: -2, Ze: 1



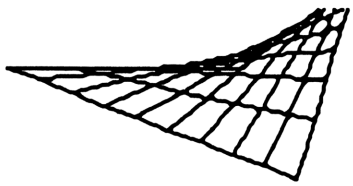
Xe: 0.5, Ye: -2, Ze: -1

Xe: 0.5, Ye: -2, Ze: 0

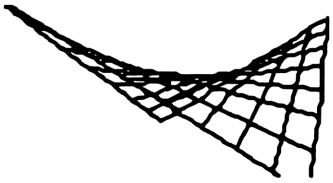
Xe: 2, Ye: -2, Ze: 2

Exercise. Try the same thing with { U 0 V } as the current equation. In this exercise you are viewing the square in the xz-plane with vertices (0, 0, 0) and (1, 0, 1).

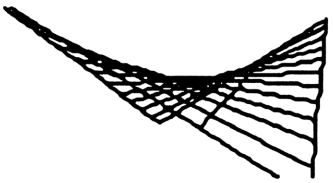
To use WIREFRAME effectively requires some knowledge of the surface. The View Volume should contain one or more critical points, that is, where the action is. We will work with the surface z = xy and see how choices of the View Volume, View Point and number of Steps interact to improve the visualization of the surface in a region containing (0, 0, 0). Using the Screen Interface, choose WIREFRAME as a PTYPE and enter ' X * Y ' as the current equation. Set the independent variable to X and use 10 steps for X. Set the dependent variable to be Y and use 8 steps for Y. Since (0, 0, 0) is a critical point, choose the View Volume as -2, 2; -2, 2; and -2, 2. Thinking about the symmetry of the figure we can try 3, -3, 3 for the View Point. These choices result in the following:



We are expecting a hyperbolic paraboloid, so what happened? The picture does not capture the essential behavior of the surface at (0, 0, 0), namely, that (0, 0, 0) is a saddlepoint. The problem is that the figure is "too symmetric" in X and Y. Earlier we saw that an oblique view helps our eye orient the figure in three space. Try -5, 5; -1, 1; and -4, 4, for the View Volume. To have an oblique view we must change the View Point. Since in the previous picture we were viewing from too high a vantage point, try 6, -3, 2:



The picture captures the saddlepoint at the origin, but if you look away and then look back there is some confusion about which part of the surface is in front. One way to improve this is to choose Steps unequally. Try 4 steps for X and 12 steps for Y:



1.6 Intersections of Surfaces

Slices of explicitly defined 3D surfaces. The intersection of a cylinder with a vertical generator and an explicitly defined surface is a curve in three dimensions. If the cylinder is a plane then the intersection is said to be a *slice* (or *section*). We want to use our representations of 3D parametric curves to plot various standard slices.

Producing really good 3–dimensional pictures requires patience and judgment. In the examples we will concentrate on two surfaces that should be familiar, the paraboloid $z = x^2 + y^2$ and the hyperbolic paraboloid $z = x^2 - y^2$. Depending on the reader's intuition, this should provide an opportunity to show that reasonable choices can lead to unexpected results which require interpretation.

We begin with the intersection of a vertical plane $Ax + By = C$ with a surface $z = f(x, y)$. There are two ways to parameterize the curve:

1) $x = t$; $y = ax + b$; $z = f(x, y)$ or $r = (t, at + b, f(t, at + b))$ and

2) $x = ay + b$; $y = t$; $z = f(x, y)$ or $r = (at + b, t, f(at + b,t))$.

We need both ways. If the plane is perpendicular to the y-axis then we use 1) with $a = 0$. Similarly, if the plane is perpendicular to the x-axis then we use 2) with $a = 0$.

Example. Slice the surface $z = x^2 + y^2$ with various vertical planes perpendicular to the y-axis, say $y = 0$. Store 'T' in 'X', 'X^2 + Y^2' in 'Z' and 0 in 'Y'. Store 0 into 'Xe', -1.5 into 'Ye', and 0 into 'Ze'. Reset PPAR to the default settings and then change INDEP to {T -1 1}. Set EQ as 'PARA3D'. Use AUTO to autoscale, then ERASE and DRAW. Use ZOUT with H-FACTOR 2 and V-FACTOR 1. since ZOUT inserts axes, remove them with ERASE DRAW. This produces something like the following:

Be careful in examining the picture to realize that the curve is in a plane perpendicular to the y-axis.

Of course, we may choose another plane, say $y = x/2$. Simply store '0.5 * X' in 'Y' and enter ERASE DRAW. This produces something like the following:

Comparing this graph with the one above, we see there are only subtle differences in the pictures even though the two curves come from different slices.

Moving the View Point a bit, say to –3 –2 0, produces the following graph for the first curve:

The second curve looks like this:

Certainly, no one would confuse the two curves. This suggests that frequently we will want to view 3D objects from more than one perspective.

Exercise. Repeat the last example using the surface $z = x^2 - y^2$.

Example. Critical points of a surface $z = f(x, y)$, i.e., points (x_0, y_0) where $\nabla z(x_0, y_0) = 0$, are places where important things happen. Slices through critical points can be revealing. For example, slicing $z = x^2 + y^2$ through (0, 0) along $y = x$ produces the following:

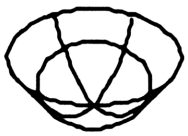
Slicing along $y = -x$ produces

In fact, any slice through (0, 0) will yield a parabola with vertex at the origin. Thus, one can easily believe that $z = x^2 + y^2$ has a local minimum at (0, 0).

Exercise. Repeat the above example for $z = x^2 - y^2$. Slice the surface with the planes $x \pm y = 0$.

Intersections of cylinders with explicitly defined 3D surfaces. We can use PARA3D for the intersection of a cylinder generated by any parametric curve in the plane and a 3D surface. For instance, the curve C: $x = u(t)$, $y = v(t)$, $a \le t \le b$, or $r = (u(t), v(t), f(u(t), v(t)))$, $a \le t \le b$. Of special interest are the circles C: $x = a \cos t$, $y = a \sin t$, $0 \le t \le 2\pi$.

Example. With some care the following piece of the paraboloid $z = x^2 + y^2$ can be produced with PARA3D.



For the circular sections we chose $a = \sqrt{2}/2$ and $a = 1$.

Exercise. Try to reproduce the picture above.

Suppose $\{P_i\}_0^n$ is a sequence of points in the plane and C_i, for $1 \le i \le n$, is the curve from P_{i-1} to P_i defined by C_i: $r = (i - t)P_{i-1} + (t - i + 1)P_i$, $i - 1 \le t \le i$. The curve $C = \sum_{i=1}^{n} C_i$ is called

a *polygonal curve* from P_0 to P_n. The polygonal curves in the plane form another important family of curves generating 3D cylinders. We will construct polygonal curves using the following two programs VERTS and POLIG.

level n + 1	. . .	level 2	level 1		level 1
(x_1, y_1)		(x_n, y_n)	n	⇒	*list*

VERTS:

```
<< →LIST 'Vrts' STO >>          |Stores the vertices as a list named
                                 |'Vrts'.
```

POLIG:

```
<<      Vrts T CEIL DUP 0 ==
   <<1 + >> IFT GETI             |Gets the first vertex.
      DUP2 DROP 1 - T - *        |Multiplies that vertex by
                                 |(1 - T) and stores the result in 'p'.
   'p' STO SWAP DUP2 DROP        |Gets the next vertex.
   GET SWAP  T SWAP 2 - - *      |Multiplies by (T - 1).
   p + 'p' PURGE                 |Adds the result to the vertex in 'p'.
                                 |Clears 'p'.
>>
```

Example. A reasonable case can be made for studying surfaces using the analogy of walking along paths in the mountains. Consider the surface z = sin(x)sin(y) which

looks like a rumpled (but more regular) blanket. For the analogy, think of z as a deviation from some reference altitude. Maybe the units are 1000 ft. We are interested in following paths, say from (0, 0, 0) to (2π, 2π, 0). Enter

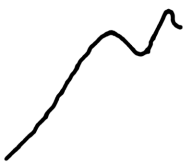
```
(0, 0)  2 'π' * →NUM  DUP  R→C  2  VERTS

<< POLIG OBJ→ DROP >> 'X' STO

<< POLIG OBJ→ SWAP DROP >> 'Y' STO

'SIN(X )*SIN(Y)'  'Z'  STO
```

Now graph PARA3D in the parametric mode with the independent variable set as {T 0 1}, i.e., T ranges from 0 to one less than the number of vertices. This produces

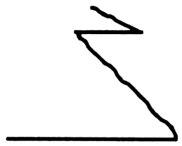


Suppose you object to the up/down nature of the path. Can you find a path from (0, 0, 0) to (2π, 2π, 0) with less climbing? Perhaps a path that is level for its entire length?

Suggestion: You may find it easier to store << POLIG OBJ→ DROP >> in plgX and << POLIG OBJ→ SWAP DROP >> in plgY. Then before graphing PARA3D to produce the curve on the surface, store 'plgX' in 'X' and 'plgY' in 'Y'.

We want to follow the intersection of the surface with the vertical cylinder whose intersection with the xy-plane is parameterized by the polygonal curve C from (0, 0) to (π, 0) to (π, π) to (2π, π) to (2π, 2π). On the surface the previous path and the new path look like the following:

(0, 0, 0) to $(2\pi, 2\pi, 0)$

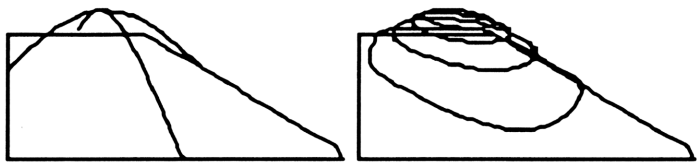


(0, 0) to $(\pi, 0)$ to (π, π) to $(2\pi, \pi)$ to $(2\pi, 2\pi)$

Plotted Together

Since $z = 0$ on the new path, the new path never leaves the xy-plane in moving from (0, 0, 0) to $(2\pi, 2\pi, 0)$.

Suppose at some point, say $(\pi/2, 0, 0)$, you want to climb to the top of the "hill" at $(\pi/2, \pi/2, 1)$. Two ways of visualizing the hill, slicing the surface, are as follows.



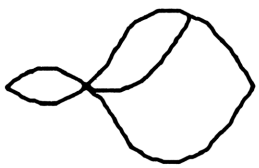
Two paths lead to the top. Both are intersections of the surface with vertical cylinders whose intersections with the xy-plane are parameterized by polygonal curves. The first, C_1, goes from $(\pi/2, 0)$ to $(\pi/2, \pi/2)$. The second, C_2, goes from $(\pi/2, 0)$ to $(3\pi/4, \pi/2)$ to $(\pi/2, 2\pi/3)$ to $(3\pi/8, \pi/2)$ to $(\pi/2, \pi/2)$. On the surface the curves look like the following:

Which would be the easier walk?

Exercise 1. Graph paths from (0, 0, 0) to (2, 2, 0) on the surface $z = x^2 - y^2$ that are intersections of the surface with vertical cylinders whose intersections with the xy-plane are parameterized by polygonal curves C_1 and C_2. The first, C_1, goes from (0, 0) to (2, 2). The second, C_2, goes from (0, 0) to (2, 0) to (2, 2). Which would be the easier walk?

Exercise 2. Produce the pseudolevel curves of $z = x^2 - y^2$ using Ps-Contour. Overlay with the curves C_1 and C_2. How can you distinguish between C_1 and C_2 in terms of the level curves?

3D surfaces. We can graph surfaces by graphing one or more judiciously chosen slices. Three general rules apply. Fewer, more widely separated slices usually look better. You may need to try several viewing perspectives before something satisfactory emerges. A piece of the surface might be sufficient to tell the whole story. This picture of a hyperbolic paraboloid looks reasonable if you already know what the surface looks like.



The following program was designed to convey a sense of a surface by animating several different views of a slice to give an illusion of motion. PICS can be slow to run, so animation is of limited utility. The ideas work best for

intersections of cylinders generated with simple figures, straight lines, circles and rectangles parameterized with the reserved variable K.

PICS:

```
<< 1 5 FOR i                          |Generates five
   i 'K' STO ERASE                    |different views of a slice of
   DRAW PICT RCL                      |the surface.
   1 STEP 5 →LIST DUP                 |Arranges the views on the
   REVLIST + OBJ→                     |stack for ANIMATE.
>>
```

Example. We want to animate the intersection of the surface z = sin(x)sin(y) with the vertical plane y = Kπ/6. Set PTYPE to PARAMETRIC, Xe to 0, Ye to -1 and Ze to 2. Enter

```
'SIN(X) * SIN(Y)' 'Z' STO 'K * 3.14 / 6' 'Y' STO RESET
{ X 0 3.14 } INDEP .1 RES 1 +/- 2 XRNG 1.5 +/- 0 YRNG
PICS ANIMATE
```

Exercise. Redo the example using Y-SLICE. Using the Screen Interface set TYPE to Y-SLICE and EQ to 'SIN(X) * SIN(Y)'. Set X-LEFT to 0, X-RIGHT to 3.14, Y-NEAR to 0, Y-FAR to 3.14, Z-LOW to 0 and Z-HIGH to 2. How do the two animations differ?

For some figures you might want to generate different views by moving the View Point. Try something like Xe = a + (b - a)K/5.

Exercise. Consider the intersection of the surface $z = y^2 + x^2$ with $y = x/2$. Let Xe = -4 + 4K/5, Ye = -2 and Ze = 0.

Intersections of planes with quadratic surfaces. The intersection of two surfaces in 3-space will usually be a curve. To treat general surfaces would be too difficult at this time. Instead, we will look at the problem of describing regions bounded by a quadratic surface and a plane.

Example. Describe the region D of R^3 between the quadratic surface $z^2 = 2x^2 + 3y^2 + 1$ and the plane $x + y + z = 2$.

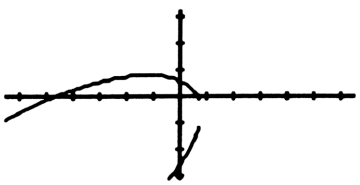
These two surfaces intersect in a 3-dimensional curve. We begin by looking at the cylinder set parallel to the z-axis that contains this intersection. The equation of the cylinder set is found by eliminating z from the equations; i.e., solve the equation of the plane for z and then substitute the result into the equation for the elliptic paraboloid of two sheets. We want to look at the xy-section of the cylinder set to see that we have a bounded region. Begin with the default plotting parameters

'X' PURGE 'Y' PURGE

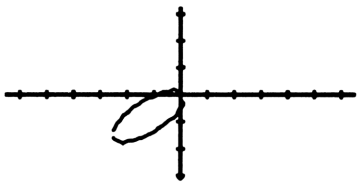
'X + Y + Z - 2' 'Z' ISOL DEF

'2 * X ^ 2 + 3 * Y ^ 2 - Z ^ 2 + 1' EVAL ↰ PLOT

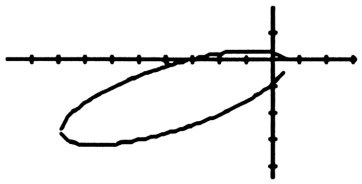
CONIC ERASE DRAX DRAW



The resulting picture is difficult to analyze. Step back by reducing the resolution, i.e., in ZFACT set H-factor and V-factor to 5 and then ZOUT.



The region appears to be an ellipse, and we can confirm this by shifting the axis. Move the crusor to the center of the figure and press ENTER to record the coordinates of the cursor on the stack. Press ON, open PPAR and press CENT (on the second page) to relocate the center of the screen to the coordinates location. Then execute ERASE DRAX DRAW. The resulting picture is convincing, i.e., the cylinder set intersects the xy-plane in a bounded elliptic region D_2.



The surface $z^2 = 2x^2 + 3y^2 + 1$ is an elliptic paraboloid. For $x = 0$, $y = 0$ we have $z^2 = 1$. The corresponding point on the plane $x + y + z = 2$ is (0, 0, 2), i.e., the plane intersects the upper sheet of the elliptic paraboloid. Hence for (x, y) in the bounded region D_1, whose boundary is the ellipse obtained using the calculator, we must have

$$(2x^2 + 3y^2 + 1)^{1/2} \leq 2 - x - y.$$

In order to complete the description of D_1, we must obtain a description of the region D_1 in set builder notation. Recall what is stored in EQ and execute

'Y' 2 TAYLR 'Y' QUAD DEF

Y 1 +/- 's1' STO EVAL

Now execute EXPA and COLCT several times to get a simplified version, maybe something like '-1 - 0.25 * √ (40 - 4 * X ^ 2 - 48 * X) + 0.50 * X'. Proceed in the same manner with 's1' having the value 1. Thus for (x, y) in D_1 we have

$$-1 + (x - \sqrt{10 - 12x - x^2})/2 \le y \le -1 + (x + \sqrt{10 - 12x - x^2})/2.$$

Call this interval of y values I_X. To find the interval of x values proceed as follows. The x-coordinates of the vertices of the ellipse will occur when the radical is zero. Why? Use the editor to obtain just the radical.

```
EDIT '√ (40 - 4 * X ^ 2 - 48 * x)'
x²  'X'  QUAD
```

We find the x-interval with

```
ENTER -1 's1' STO EVAL
SWAP 1 's1' STO EVAL
```

The stack is 0.7823 ... , -12.7823 ... . We obtain the right order with SWAP.

Thus $D_1 = \{(x, y) \mid -12.7823 \le x \le 0.7823$ and y in $I_x\}$. Finally, $D = \{(x, y) \mid -12.7828 \le x \le 0.7828$, y in I_x and $(2x^2 + 3y^2 + 1)^{1/2} \le z \le 2 - x - y\}$.

SECTION 2. OPTIMIZATION AND INTEGRATION

2.1 Classification of Critical Points for Functions of Two Variables

Taylor series for functions of two variables. The Taylor series expansion of f(x) about x_0 is given by

$$f(x) = \sum_{p=0}^{\infty} \frac{f^{(p)}(x_0)}{p!}) (x - x_0) .$$

The sum of the first $n + 1$ terms, i.e., for $p = 0, 1, \ldots, n$, is called the nth degree Taylor polynomial $P_n(x)$. Of course, $P_1(x)$ is the line tangent to the graph of $f(x)$ at $(x_0, f(x_0))$. Similarly, each of the polynomials approximates the function.

We look for critical points of $y = f(x)$ by finding values for which the graph of $P_1(x)$ is a horizontal line. A critical point x_0 of $y = f(x)$ is classified as a local minimum if the second degree Taylor polynomial $P_2(x)$ for $f(x)$ is a parabola that opens up. Notice that $P_2(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2$, which opens up provided $f''(x_0) > 0$. Similarly, x_0 is classified as a local maximum if $P_2(x)$ opens down.

Example 2. TAYLR always computes the polynomial expansions about zero, i.e., the MacLaurin expansions. In order to find an expansion about some other point we must introduce a change of variables. Suppose we want the expansion of $1/x$ about 1. Notice there is no expansion about 0.

Enter the following commands:

```
'X' 1/x 'Y + 1' 'X' STO EVAL
'Y' 3 TAYLR
'X' PURGE 'X - 1' 'Y' STO EVAL
```

Exercise. Try expanding sec x about π.

For $z = f(x, y)$ we proceed as follows: let $w(t, x, y) = f(x_0 + t(x - x_0), y_0 + t(y - y_0))$, a section of $z = f(x, y)$ in the direction of (x, y). The second degree Taylor polynomial at 0 for $w(t, x, y)$ (holding (x, y) fixed) is

$$P_2(t, x, y) = f(x_0, y_0) + [f_x(x_0, y_0)(x - x_0) + f_y(x_0, y_0)(y - y_0)]t + (1/2)[f_{xx}(x_0, y_0)(x - x_0)^2 + 2f_{xy}(x_0, y_0)(x - x_0)(y - y_0) + f_{yy}(x_0, y_0)(y - y_0)^2]t^2$$

which reduces to

$$P_2(t, x, y) = f(x_0, y_0) + (1/2)[f_{xx}(x_0, y_0)(x - x_0)^2 + 2f_{xy}(x_0, y_0)(x - x_0)(y - y_0) + f_{yy}(x_0, y_0)(y - y_0)^2]t^2$$

when (x_0, y_0) is a critical point.

Example. Find the second degree Taylor polynomial for w(t, x, y) given z = (sin x)(sin y) and $(\pi/2, \pi/2)$ is a critical point. Enter the following:

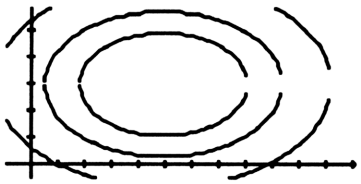
'SIN(U) * SIN(V)' 'Z' STO
'π/2' →NUM DUP +/- 'X' + 'T' × + 'U' STO
U EDIT (change X to Y) ENTER 'V' STO
Z EVAL 'T' 2 TAYLR

Classifying critical points. In order to classify $(\pi/2, \pi/2)$ as either a local maximum, local minimum or a saddlepoint, we must determine if $P_2(t, x, y)$ is a parabola opening down (or up) for all (x, y) or opening down for some (x, y) and up for others. We can decide by sketching the level curves of

$P_2(1, x, y)$. Proceed as follows:

1 'T' STO EVAL
'K' PURGE 'K' + NEW T2 ENTER CONIC
0.5 +/- 'K' STO DRAW
0 'K' STO DRAW
1 'K' STO DRAW

Clearly, we have produced the level curves of an elliptic paraboloid, i.e., the coefficient of t^2 in $P_2(t)$ must always be either positive or negative. Therefore $(\pi/2, \pi/2)$ must be a local extremum. Furthermore, since the paraboloid looks down $(\pi/2, \pi/2)$ must be a local maximum.



The example illustrates this general result: if $f_{xx}(x_0, y_0)(x - x_0)^2 + 2f_{xy}(x_0, y_0)(x - x_0)(y - y_0) + f_{yy}(x_0, y_0)(y - y_0)^2$ is an ellipse then (x_0, y_0) is an extremum. In this case, (x_0, y_0) is a maximum if $f_{xx}(x_0, y_0)$ is negative and a minimum if $f_{xx}(x_0, y_0)$ is positive. If $f_{xx}(x_0, y_0)(x - x_0)^2 + 2f_{xy}(x_0, y_0)(x - x_0)(y - y_0) + f_{yy}(x_0, y_0)(y - y_0)^2$ is a hyperbola then (x_0, y_0) is a saddlepoint. This can be developed as an efficiently applied criterion as follows: Let $\Delta = (f_{xy}(x_0, y_0))^2 - f_{xx}(x_0, y_0)f_{yy}(x_0, y_0)$.

1) If $\Delta > 0$, then (x_0, y_0) is an extremum, a maximum if $f_{xx}(x_0, y_0) < 0$ and a minimum if $f_{xx}(x_0, y_0) > 0$.

2) If $\Delta > 0$, then (x_0, y_0) is a saddlepoint.

3) If $\Delta = 0$ then the test fails.

Exercise. Use the method outlined above to classify the critical points (-1, 11/6) and (1, 1/2) of $z = x^3 + y^2 + 2xy - 4x - 3y + 5$ as either local extrema or saddlepoints.

2.2 Polya's Problems[1]

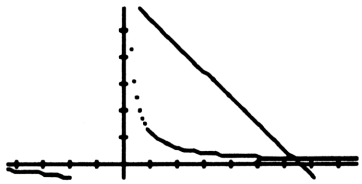
Example. If the sum of two numbers is 6, what is the maximum of their product?

Let the numbers be x and y. We are given that $x + y = 6$ and we wish to maximize the function $f(x, y) = xy$ subject to that constraint. We produce a graph containing both $x + y = 6$ and $xy = 1$, i.e., the constraint and an arbitrary level curve of f(x, y).

---

[1] George Polya, Mathematics and Plausible Reasoning, Vol. 1, Princeton University Press, Princeton, NJ, 1954.

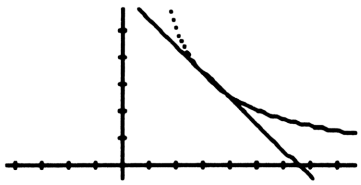
```
'PPAR' PURGE CONIC
'X + Y - 6' 'CONSTR' STO
'K' PURGE 'X*Y - K' 'FUNCL' STO
{ 'CONSTR' 'FUNCL' } 'EQ' STO
1 'K' STO DRAW
```



Of course, xy = 1 is not tangent to the constraint x + y = 6 at any point. We can choose a more appropriate level curve of f(x, y) = xy by moving the cursor to a point on the constraint x + y = 6 where we think some level curve is tangent. Capture that point with the menu key (x,y) and then continue. (Our guess resulted in (3.1, 2.9).)

```
ON OBJ→ × 'K' STO ERASE DRAW
```

This will produce something like the following:



The picture strongly suggests that (3.1, 2.9) is either a point of tangency or near such a point. To check this, we can zoom in by setting the center at (3.1, 2.9) and repeating the process from the top.

Exercise. Find the minimum of $x^2 + y^2$ on the curve $x = y^2 + 1$.

Example. Find the distance from the point (1, 2) to the curve $y = x^3 - 3x^2 + 2x$. It is sufficient to find the point that minimizes the square of the distance. This latter problem can be stated as:

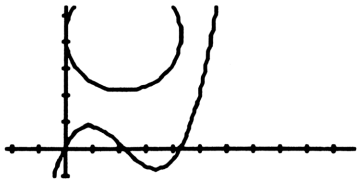
Minimize: $f(x, y) = (x - 1)^2 + (y - 2)^2$

Subject to: $g(x, y) = y - x^3 + 3x^2 - 2x = 0$

The method of Lagrange multipliers leads to a system of polynomial equations with no rational solution. Let us try the graphical method outlined above.

```
'PPAR' PURGE -6.8 6.8 XRNG -1 2.1 YRNG
'X * (X - 1) * (X - 2) - Y' 'CONSTR' STO
'K' PURGE '(X - 1)' x² '(Y - 2)' x² + 'K' -
'FUNCL' STO
{ 'CONSTR' 'FUNCL' } 'EQ' STO
1 'K' STO DRAW
```

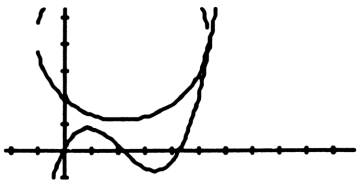
This produces something like the following:



Use the cursor to estimate the point on the curve nearest (1, 2), for us (2.4, 1.8). We use CENTER ZOOM to move in on the region of the graph of most interest. We further update the graph as follows:

```
OBJ→ 2 →ARRY 1 2 2 →ARRY - DUP DOT 'K' STO
ERASE DRAW
```

This produces the following:



The current estimate of the minimum seems pretty good, but we can improve our estimate by repeating the process with a better guess. Our current estimate of the distance is the square root of K, i.e., 1.50.

Exercise. Find the distance from the point (1, 2) to the curve $y = \ln x$.

Example. We want to find a graphical solution to a more difficult version of the milkmaid problem. Suppose a house is located at P(0, 1), a barn at Q(-2, 1) and a river bank is given by $y = \sin x$. If each morning the milkmaid walks in a straight line from the house P to a point R on the riverbank to fill her pail and then in a straight line to the barn Q, the total distance she must travel is d(P, R) + d(R, Q). The problem is to choose the point R on the riverbank that minimizes the total distance she must travel.

The problem can be stated as follows:

Minimize: $f(x, y) = ((x + 2)^2 + (y - 1)^2)^{1/2} + (x^2 + (y - 1)^2)^{1/2}$

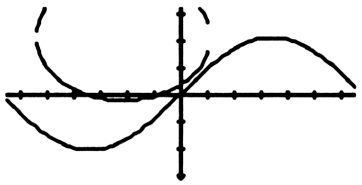
Subject to: $g(x, y) = y - \sin x = 0$

The method of Lagrange Multipliers leads to some difficult equations. However, the graphical method outlined above works reasonably well.

The level curves of the distance the milkmaid walks are ellipses. Recall from the derivation of the general form $\frac{(x - h)^2}{a^2} + \frac{(y - k)^2}{b^2} = 1$ of an ellipse with major axis horizontal that for this problem (h, k) = (-1, 1), 2a is the distance the milkmaid walks, 2c = 2 is the distance between the house and barn (the foci), and $b^2 = a^2 - 1$. The calculator analysis proceeds as follows:

'PPAR' PURGE FUNC 'SIN(X)' 'CONSTR' STO
'K' PURGE 'X' 1 + x^2 'K' x^2 + 'Y' 1 - x^2 'K' x^2
1 - + + 1 - 'Y' 2 TAYLR 'Y' QUAD (Edit the
result to eliminate Y =.) 'FUNCL' STO
{ 'CONSTR' 'FUNCL' } 'EQ' STO
1 's1' STO 1.5 'K' STO ERASE DRAW

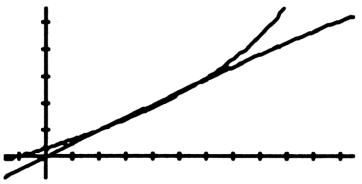
The graph looks something like.



Use the cursor to capture a guess for the point that minimizes the distance and then BOXZ to bracket the region of interest. Our guess is (.2, .3). To compute the updated K proceed as follows:

OBJ→ 2 →ARRY DUP 0 1 2 →ARRY - DUP DOT $\sqrt{x}$
SWAP 2 +/– 1 2 →ARRY - DUP DOT $\sqrt{x}$ + 2 + 'K' STO
ERASE DRAW

The graph looks something like this.



Of course, the estimate can be improved by repeating the above process.

Exercise. Find a graphical solution to the milkmaid problem, given that the house is at (0,1), the barn at (0, 2) and the river bank is given by $y = \ln x$.

2.3 Integration

There are two approaches to numerical integration on the HP-48: using the stack or the Equation Writer. Both require setting the numerical mode (Enter -3 SF) and specifying an accuracy factor. The latter is specified by setting the display mode to n FIX , where n is number of decimal digits to be displayed. In the examples and exercises, we suggest n = 3 to avoid lengthy calculation times.

Integration on the stack requires the following arguments.

Integral

level 4	level 3	level 2	level 1		level 1
a	b	f(x)	'x'	⇒	integral

Examples

1) Use ∫ to evaluate $\sqrt{2}\int_0^{2\pi}\sqrt{1-\cos}\ dt$.

3 FIX 2 √x

0 2 'π' →NUM × '1 - COS(T)' √x 'T'

-3 SF ∫ -3 CF

Notice that we wanted the symbolic mode (-3 CF) in order to construct the integrand using the symbol capabilities of the machine. We will assume from this point that the accuracy factor has been set.

The following program is convenient.

Integral

level 4	level 3	level 2	level 1		level 1
a	b	f(x)	'x'	⇒	integral

IGL:

```
<<   DUP PURGE SWAP          | Evaluates the nested
     EVAL EVAL SWAP          | functions down to the
                             | variable of integration.
     -3 SF ∫ -3 CF           | Evaluates the integral
>>                           | numerically. If an error
                             | occurs reset the symbolic
                             | mode.
```

2) Compute the circumference of the ellipse given parametrically by $x = 2\cos t$, $y = \sin t$, $0 \le t \le 2\pi$.

0 2 'π' →NUM × '4 * SQ(SIN(T)) + SQ(COS(T))' √x 'T' IGL

Exercises

1) Compute the circumference of the unit circle using the standard parameterization.

2) Compute the length of the parabola $y = x^2$ from ((1, 1) to (3, 9).

3) Compute the surface area of the figure generated by revolving about the x-axis the curve $y = \ln x$, $1 \le x \le 2$.

4) Compute the surface area of the figure generated by revolving about the x-axis the curve parameterized by $x = t^2$, $y = t^3$, $0 \le t \le 1$.

Example 3. Compute the length of the curve $y = \frac{1}{1 + x^2}$ from (0, 1) to (3, 0.1). Enter the following:

0 3 '1 + SQ(X)' 1/x 'X' PURGE 'X' ∂ x² 1 + √x 'X' IGL

In the second approach to numerical integration use the Equation Writer to produce

$$\int_0^3 \sqrt{1 + SQ(\frac{\partial}{\partial X}(\frac{1}{1 + SQ(X)}))}\ dX.$$

Press ENTER to put '∫ (0, 3, √ (1 + SQ (∂X (1 / (1 + SQ (X))))), X)' on stack level 1. To evaluate the integral enter −3 SF →NUM −3 CF. Again the following program is convenient.

IGLEW:

```
<< −3 SF →NUM −3 CF        | If an error occurs reset
   >>                      | the symbolic mode.
```

Exercises

5) Compute the surface area of the figure generated by revolving about the x-axis the curve discussed in the example above using the Equation Writer.

6) Redo the previous set of exercises using the Equation Writer.

7) Compute the circumference of the circle given in polar coordinates by r = sin θ. Which approach is easier to use?

Example 4. Evaluate $\int_0^1 \int_x^{\sqrt{x}} \sqrt{x+y}\ dydx$. The Equation Writer produces

'∫ (0, 1, ∫ (X, √ X, √ (X + Y), Y), X)'

Notice that, after executing IGLEW, the calculator takes a while to return the answer of 0.152, about 12 seconds. Iterated integrals are harder to obtain numerically than definite integrals for functions of a single variable. Also, since the stack is complicated you should DUP the argument on the stack before executing IGLEW. The argument can also be edited.

Exercises

8) Find the volume of the figure bounded by the planes $x + y + z = 1$, $x = 0$, $y = 0$ and $z = 0$.

9) Find the volume of the figure bounded by $z = 0$, $x^2 + y^2 = 1$ and $z = x^2 + y^2$.

10) Compute the mass of a flat plate, the quarter disk $x^2 + y^2 \leq 1$, $x \geq 0$ and $y \geq 0$, with density $\rho(x, y) = xy$.

SECTION 3. VECTOR FIELDS AND LINE INTEGRALS

3.1 Vector Fields

A vector valued function defined on a subset of R^n, $n > 1$, is called a *vector field.* Similarly, a scalar valued function is called a *scalar field.* We will use a standard notation, namely, $f(x, y) = P(x, y)\mathbf{i} + Q(x, y)\mathbf{j}$. Of course, $\mathbf{f}$ is a vector field with component functions P and Q, which are scalar fields.

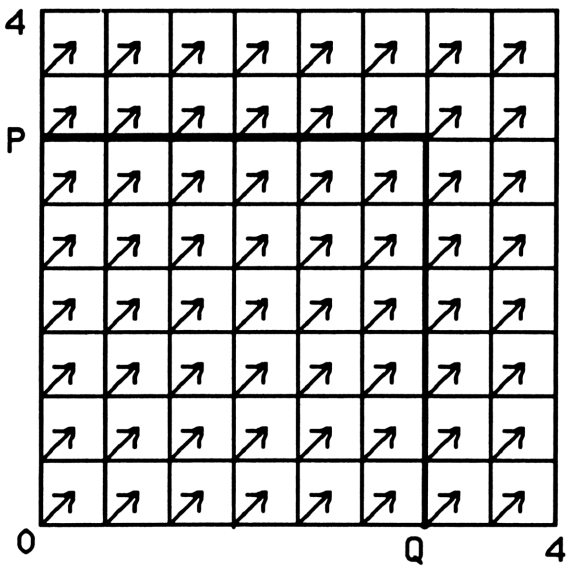
If a constant force c (constant in both direction and magnitude) is applied is applied in moving a particle along a straight line (the x-axis) from a to b ($a < b$) then the work W done is $c(b - a)$. Notice that if c is positive then W is positive and the physical interpretation is that we have done work on the system. If c is negative then the system does work on us.

Any problem where motion is in a straight line and the force acts in a direction parallel to the direction of motion can be recoordinatized to fit our standard formulation.

Suppose the particle is to be moved from $\mathbf{P} = (a, b)$ to $\mathbf{Q} = (c, d)$ along the straight line connecting $\mathbf{P}$ and $\mathbf{Q}$, but the force $\mathbf{f}$ no longer is assumed to act in a direction parallel to $\mathbf{u} = [c - a, b - d]$. The component of the force in the direction $\mathbf{u}$ is

given by $(f \cdot u)/\| u \|$. The distance traveled in moving from P to Q is $\|u\|$. Hence the work done is $((f \cdot u)/\|u\|)\|u\| = f \cdot u$.

Example 1. Compute the work required to move a particle in the force field illustrated below from the point P to the point Q.

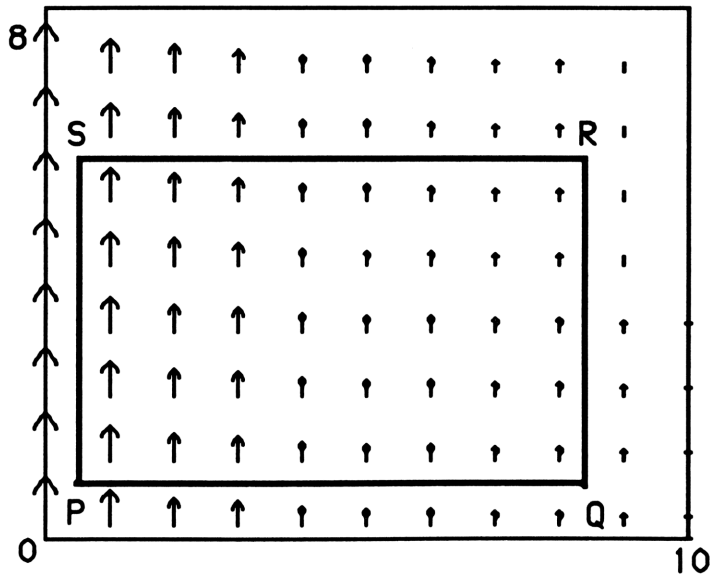


The force field is constant, say $f = (1/4)i + (1/4)j$. The path the particle must travel can be split up into two parts, from P = (0, 3) to (3, 3) and from (3, 3) to Q = (3, 0). The total work W is the sum of the work on each part. Let $u = 3i$ and $v = -3j$. The force to be exerted in moving the particle must balance the force exerted by the field, i.e., the force exerted in moving the particle must be -f. Thus $W = -f \cdot u - f \cdot v = -(3/4) + (3/4) = 0$.

Example 2. The picture below represents the force field $f = (1/(1 + x))j$. Compute the work done in moving a particle around the path PQRSP.

Of course, in moving from P to Q and from R to S, no work is done. Again, a force which balances the field must be exerted on the particle, but work is the

component of the force exerted in the direction of motion. On those two segments of the path the force is orthogonal to the direction of motion so no work is done. Assume that P = (1/2, 1), Q = (17/2, 1), R = (17/2, 6) and S = (1/2, 6). Let $\mathbf{u} = 5\mathbf{j}$ and $\mathbf{v} = -\mathbf{u} = -5\mathbf{j}$. Since the field on the path from Q to R is $f(17/2, y) = (2/19)\mathbf{j}$ and from S to P is $f(1/2, y) = (2/3)\mathbf{j}$, we have $W = -f(17/2, y)\cdot\mathbf{u} - f(1/2, y)\cdot\mathbf{v} = -(10/19) + (10/3) = 160/57$. Note that in moving from Q to R the system does work on us. From S to P we do work on the system. Since the total work is positive, we do work in traversing the path PQRSP. If we move in the opposite direction PSRQP then the system does net work on us.

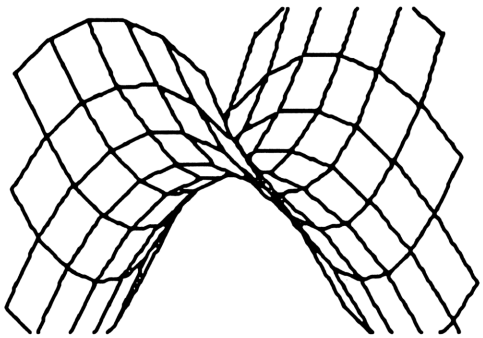


Exercise 1. Show that W = 0 if the particle is moved in a straight line from P to Q.

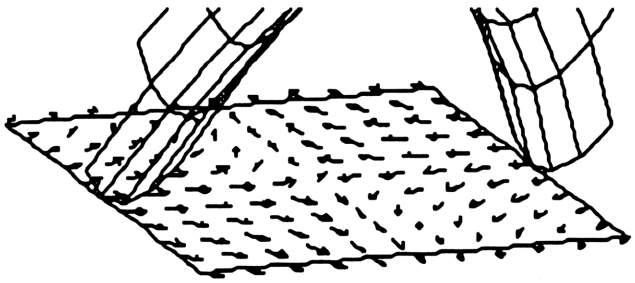
Exercise 2. Which path, PQR or PSR, requires the most work to move a particle from P to R?

Of course, in order to construct a vector field $f(x, y) = P(x, y)\mathbf{i} + Q(x, y)\mathbf{j}$ one only has to specify P and Q. An interesting way to do this is to start with a function

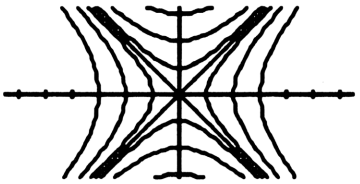
$z = g(x, y)$ and let $f = \nabla g = [g_x, g_y] = g_x i + g_y j$. A vector field defined this way is called a *gradient field*. Consider the surface below, given explicitly by $z = x^2 - y^2$.



The surface with attached gradient field appears below. All vectors have been normalized to simplify the picture.



Exercise 3. Use the fact that a gradient vector at a point is normal to the level curve through the point to add normalized vectors to the level curves of $z = x^2 - y^2$ given below and recapture the previous picture. Remember which direction is up hill.



Exercise 4. Produce the gradient fields for $z = y^2$ and $z = x^2 + y^2$.

3.2 Line Integrals

We use the numerical integration program IGL to compute the various line integrals in the next three examples. Notice that the calculator performs the task of constructing the integrand symbolically from the pieces, i.e., P(x, y), Q(x, y), x = x(t) and y = y(t). In fact, the keystrokes for computing much more complex line integrals differs very little from these simple examples.

Example 1. $\int_C x^2y\,ds$; C: $x = \cos t$, $y = \sin t$, $0 \le t \le \pi/2$.

'X ^ 2 * Y' 'P' STO
'COS(T)' 'X' STO
'SIN(T)' 'Y' STO
'T' PURGE
0 'π' 2 + P
'X' 'T' ∂ x²
'Y' 'T' ∂ x² + √x ×
'T' IGL

Example 2. $\int_C (x^2y\,dx + xy\,dy)$; C: $x^2 + y^2 = 1$ from (1 ,0) to (0, 1).

C: $y = \sqrt{1 - x^2}$, $0 \le x \le 1$

```
'X ^ 2 * Y'  'P'  STO
'X * Y'  'Q'  STO
'( 1  -  X ^ 2 ) ^ .5'  'Y'  STO

1  0
'P'  'Q'  'X'  PURGE
'Y'  'X'  ∂  ×  +
'X'  IGL
```

C: $x = \cos t$, $y = \sin t$, $0 \le t \le \pi/2$

```
'X ^ 2 * Y'  'P'  STO
'X * Y'  'Q'  STO
'COS( T )'  'X'  STO
'SIN( T )'  'Y'  STO

0  'π'  2  +
'T'  PURGE  'P'
'X'  'T'  ∂  ×
'Q'  'Y'  'T'  ∂  ×  +
'T'  IGL
```

Example 3. $\int_C F \cdot dr$; $F(x, y) = (x + 2y)i + (2x + y)j$, C: $r(t) = ti + t^2 j$, $0 \le t \le 1$.

```
'X + 2 * Y' 'P' STO
'2 * X + Y' 'Q' STO
'T' 'X' STO
'T ^ 2' 'Y' STO
0 1
'T' PURGE
'P' 'X' 'T' ∂ ×
'Q' 'Y' 'T' ∂ × +
'T' IGL
```

Exercises

1) Compute $\int_C F \cdot dr$ given $F(x, y) = (x + 2y)i + (2x + y)j$, C: $x = \sqrt{2}\, t$, $y = \sqrt{2}$ sin t, $0 \le t \le \pi/4$. Compare with Example 3.

2) Compute $\int_C F \cdot dr$ given $F(x, y) = x^2 yi + xyJ$, $C = C_1 + C_2$, C_1: $x = 1 + (\sqrt{2} - 2)\, t/2$, $y = \sqrt{2}t\ /2$, and C_2: $x = \sqrt{2}\,(1 - t)\ /2$, $y = \sqrt{2}\,(1 - t)\ /2 + t$.

Compare with Example 2. What happens as C is approximated with shorter straight line segments?

Example 4. Compute $\int_C F \cdot dr$, where $F(x, y) = xyi + (x - y)j$, and C is given in polar coordinates by $r = \cos \theta$.

Here, our first task is to obtain a parametric representation of C in rectangular coordinates. This is easily done since $x(\theta) = r(\theta)\cos \theta = \cos^2(\theta)$ and $y = r(\theta)\sin \theta = \cos \theta \sin \theta$.

```
'X * Y'  'P'  STO
'X - Y'  'Q'  STO
'SQ( COS( T ) )'  'X'  STO
'COS( T ) * SIN( T )'  'Y'  STO
0  'π'
'T'  PURGE
'P'  'X'  'T'  ∂  ×
'Q'  'Y'  'T'  ∂  ×  +
'T'  IGL
```

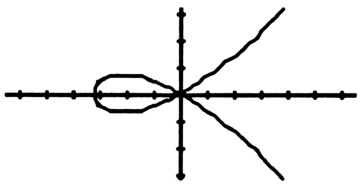
3.3 Green's Theorem

Green's Theorem may be stated as follows: Suppose F(x, y) is a vector field, i.e., F(x, y) = P(x, y)i + Q(x, y)j where P(x, y) and Q(x, y) are scalar functions (fields). Assume that $P_y(x, y)$ and $Q_x(x, y)$ are continuous in a bounded region R with a piecewise smooth boundary C that is oriented positively. (C is given parametrically by r(t) = x(t)i + yt)j, $a \le t \le b$, and x(t) and y(t) are piecewise smooth. Furthermore, as t varies from a to b, r(t) traces out C keeping R on the left.) Then $\int_C F \bullet dr = \int_a^b P(x, y)\,dx + Q(x, y)\,dy = \int\int_R [Q_x(x, y) - P_y(x, y)]\,dxdy$.

Example 1. Use Green's Theorem to compute the area of the unit disc R. Since the area is given by $\int\int_R dxdy$, we can apply Green's Theorem provided P(x, y) and Q(x, y) can be found so that $Q_x(x, y) - P_y(x, y) = 1$. Of course, this can be done in many ways. Why not P(x, y) = 0 and Q(x,y) = x? The boundary of the unit disc R is the unit circle C, which can be parameterized with r(t) = (cos t)i + (sin t)j, $0 \le t \le 2\pi$. Hence the area is

$$\int_C x\,dy = \int_0^{2\pi} (\cos t)^2\,dt = \left(\frac{t}{2} + \frac{\sin 2t}{4}\right)\Bigg|_0^{2\pi} = \pi$$

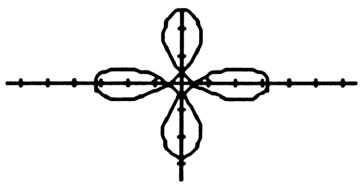
Example 2. Find the area inside the loop of Tschirnhausen's cubic C parameterized by $\mathbf{r}(t) = (t^2 - 3)\mathbf{i} + (t^3/3 - t)\mathbf{j}$, $-3 \le t \le 3$. The curve C looks something like



We need to restrict the range of t to the values that give the boundary of just the loop, call it C_1. This can be done by solving $x(t) = y(t) = 0$. Clearly, $t = \pm\sqrt{3}$, i.e., we restrict the parameter to the interval $-\sqrt{3} \le t \le \sqrt{3}$. Check to see that the loop (the boundary of the region inside) has a positive orientation. The area is then given by

$$\int_C x\,dy = \int_{-\sqrt{3}}^{\sqrt{3}} (t^2 - 3)(t^2 - 1)\,dt = \left(\frac{t^5}{5} + \frac{\sin 4t^3}{3} + 4t\right)\Bigg|_{-\sqrt{3}}^{\sqrt{3}} = \pi$$

Example 3. Find the area of the four loops in the hypotrochoid C given parametrically by $\mathbf{r}(t) = (6\cos t + 5\cos 3t)\mathbf{i} + (6\sin t - 5\sin 3t)\mathbf{j}$, $0 \le t \le 2\pi$. The figure looks something like



The loops are all equal. The boundary of each loop is negatively oriented. Only the boundary of the little region in the center of the figure is positively oriented. Let's work with the top loop. The first step is to restrict the range of the parameter and call the resulting boundary C_1. We need to solve the equation $x(t) = 6\cos t + 5\cos 3t = 0$. Using the calculator, proceed as follows:

'6 * COS(T) + 5 * COS(3 * T)' 'EQ' STO
'PPAR' PURGE DRAW

The resulting picture isn't much help, but you can improve it by changing the plotting parameters. Use the cursor and the (x,y) menu key to capture the bottom of the y-axis. It should be approximately (0, −3.1). Store the result in PPAR by hitting PMIN. Redraw the graph. Again you probably want to change PPAR. This time move the cursor to the top of a vertical line just to the right of the first two zeros, say (2, 3.2). Capture this point with the (x,y) key and store in PPAR with PMAX. Now we can estimate the first two zeros of x(t), i.e., the beginning and end of the top vertical loop of the hypotrochoid.

We use ROOT to estimate the two zeros more accurately. My answer comes back as 0.83548. Repeating this process for the zero on the right (remember to capture your best guess of the zero *first)* I got 1.5708. Of course, the true answer is $\pi/2$. The area is

$$\int_{C_1} x\,dy = \int_{.83548}^{1.5708} (6\cos t + 5\cos 3t)(6\cos t - 15\cos 3t)\,dt$$

which can be evaluated as follows: (assuming that the two zeros we found are still on the stack)

'6 * COS(T) + 5 * COS(3 * T)' '6 * SIN(T) - 5 * SIN(3 * T)'
'T' PURGE 'T' ∂ × 'T' IGL

The answer returned is -15.831. We're trying to find an area and we've ended up with a negative number. What's wrong? As noted earlier, C_1 has a negative orientation, so the area is 15.831. The area of the region bounded by the *four* loops is 63.324.

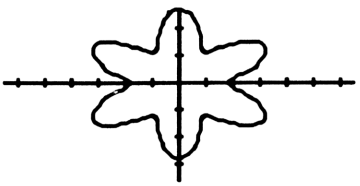
Exercises.

1) Compute the area of the region R bounded by the ellipse $\frac{x^2}{9} + \frac{y^2}{4} = 1$. Our standard parameterization of the boundary C of R is given by $\mathbf{r}(t) = (3\cos t)\mathbf{i} + (2\sin t)\mathbf{j}$, $0 \le t \le 2\pi$.

2) Find the area bounded by one arch of the cycloid generated with a circle of radius one and the x-axis. The portion of the cycloid of interest, call it C_1, is parameterized by $\mathbf{r}(t) = (t - \sin t)\mathbf{i} + (1 - \cos t)\mathbf{j}$, $0 \le t \le 2\pi$. For the relevant portion of the x-axis C_2 use $\mathbf{R}(t) = t\mathbf{i}$, $0 \le t \le 2\pi$. The boundary of the region C with positive orientation then becomes $C = C_2 - C_1$.

Sometimes one side of the equation in Green's Theorem is easier to evaluate than the other. This usually comes about because the integral on one side or the other is easier to set up.

Example 4. Evaluate $\int\int_R y - x \, dxdy$, where R is the region bounded by the curve C given in polar coordinates by $r(\theta) = 2 - \cos^2(3\theta)$. The region looks something like



Clearly, the 'snowflake' region R would be difficult to describe in rectangular coordinates. We proceed as follows:

```
'X * Y' ENTER 'Q' STO 'P' STO
'2 - COS( 3 * T ) ^ 2'
ENTER 'COS( T )' × 'X' STO
'SIN( T )' × 'Y' STO
'T' PURGE 0 6.3
'P' 'X' 'T' ∂ ×
'Q' 'Y' 'T' ∂ × +
'T' IGL
```

After a wait of some time the answer 0.000 returns.

Exercises.

3) Integrate y - x over the region bounded by the loop of Tschirnhausen's cubic parameterized by $\mathbf{r}(t) = (t^2 - 3)\mathbf{i} + (t^3/3 - t)\mathbf{j}$, $-3 \le t \le 3$. (Use the boundary C_1 developed in Example 2 above.)

4) Calculate $\int_C x\,dy$, where C is the polygonal path from (0, 0) to (1, 0) to (1, 1) to (0, 1) to (0, 0).

REFERENCES

1. J.A. Reneke, *Calculator Enhancement for a Course in Multivariable Calculus*, Saunders College Publishing, 1992, Philadelphia, PA.

HP-48G/GX Calculator Enhancement
for
Differential Equations

T. Gilmer Proctor

How can a graphics calculator be used effectively in an elementary differential equations course ? We will only give a partial answer here: hopefully you can add to our comments after some experimention with the exercises that are provided in this chapter. 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 can be used in class to illustrate concepts. It can be used for homework in the study areas that students use: libraries and dormitory rooms. The graphs and computations that are created on the calculator by the students can be stored or recreated on a microcomputer. This chapter contains only some of the possible uses of this tool and illustrates the material which my students have been given in every differential equations class for the past five years. Some of the material is taken from [1], but most of the exercises and presentations are new. Our

presentation does not require that the reader be a good HP 48 programmer since nearly all of the programs are explained within this chapter.

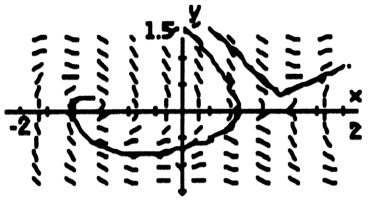
Distinctive features of the HP-48G/GX include built-in programs for calculating and displaying in the same graph approximate solutions to one or more initial value problems containing differential equations. To emphasize the statement given above, the capability to easily display solutions of several problems allows the student to study how the solutions depend on various parameters and to focus on geometrical characteristics of a system.

The first section of this chapter describes programs that have been provided for obtaining approximate solutions of initial value problems. The next section describes elementary algorithms (the Euler and improved Euler algorithms) for obtaining approximate solutions and gives elementary calculator programs to compute and plot these solutions. This material is included so that the user will become accustomed to programming. We do not give programs using higher order numerical methods for differential equations such as the Runge-Kutta algorithms. The third section 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.

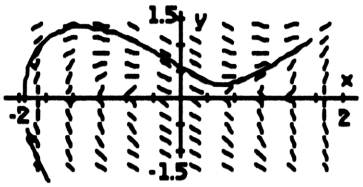
The fourth section of the chapter concerns the solution of two first order differential equations with initial conditions. Exercises are provided to aid in the study of the solutions of the second order differential equations encountered in linear and nonlinear models of mechanical springs and electrical circuits.

The fifth section contains programs that construct solutions of linear vector systems of differential equations of the form $dy/dt = A\ y + f(t)$. Finally the

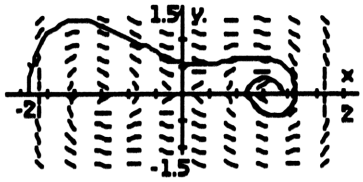
appendix contains a set of programs that can be used to sketch the direction field for a pair of differential equations.



$x'' + x' - x^2 + 1 = 0$



$x'' + x' - x^2 = 0$



$x'' + .4x' + x^3 - x = 0$

Assorted Direction Fields with Solution Overlays

Many topics associated with an introductory course in differential equations are not included. Among these are: discrete dynamical systems, delay differential equations, parameter estimation using observations of the solution, and control problems. Problems in these particular areas are presented in [1].

We suggest that the reader create a subdirectory for the programs contained in the next section and a miscellaneous subdirectory which includes the programs in the second section.

1. Introduction to the Plot Feature for Differential Equations

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 $t_0 \le t \le t_f$, where t_f may not be predetermined. What inputs to a calculator are required ?

- A program which 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 **two methods** on the HP-48G/GX to provide these inputs and obtain the graph 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 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. We will illustrate each of these methods with exercises. The user must make a decision on the appropriate method for the other exercises. We begin with the built-in method.

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**.

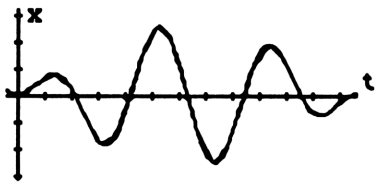
Exercise 1.1: Construct a graph of the solution of $y' + 3y = \cos t$, $y(0) = .3$ for $0 \le t \le 6.283$ using the calculator's differential equation plot feature. Enter **COS(T) - 3*Y** in the F field of the input form. (Note that the calculator automatically places this function within ' marks.) 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**.

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*COS(T) + .1*SIN(T)** as **EQ**. (Again the calculator

places ' marks around the function.) In this case we want to overlay the new graph on the old one so do not ERASE. Press **DRAW**. Note the good agreement between the approximate solution and the exact solution.

Exercise 1.2: Construct a solution of y ' = sin (ty), y(0) = 2 for $0 \le t \le 6$. Choose the program **'SIN(T*Y)'** (or **<< 'SIN(T*Y)' EVAL >>**) 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 1.3: Construct a graph of the solution of y'' + .5 y' + y = 0, y(0) = 0, y'(0) = 1, for $0 \le t \le 6.283$. 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), -(.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 **<< 'W(2)' EVAL '.5*W(2)+W(1)' EVAL NEG 2 →ARRY >>** for F together with the **INDEP** variable name W and the **INIT** vector [0 1] for **SOLN**. If we want a graph of W(1) versus X we choose **INDEP** 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) graph 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 1.4: The figure shown above is a graph of the solution of the indicated problem for $0 \le t \le \pi$. 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 indivdual 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.

The user can construct programs that ask the user for part of the total information required for a solution plot. For example, the first program **IN.FN** asks the user to write a program for the function F(T,Y) (in variables T Y) which is then stored in **FN.**

Program Name:	**IN.FN**
Purpose:	The user supplies a program which is stored in **FN**
Stored Quantities:	none

No input is required. The appropriate response is a program.

```
<< " ENTER PRG FOR FN IN T Y" " "
    INPUT OBJ→ 'FN' STO >>
```

Example repsonses 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) = column [Y(2), -Y(1)].

The next program asks the user to set the viewing window for the plot.

Program Name:	**IN.PP** (plot parameters)
Purpose:	The user supplies XRNG and YRNG
Stored Quantities:	none

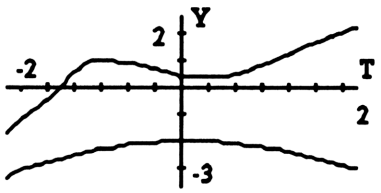
No input stack is required. The appropriate response for the first query is a pair of numbers, H-min and H-max. The response for the second query is a pair of numbers V-min and V-max.

Output: New values for **XRNG, YRNG. PICT** has been erased.

```
<< " KEY IN XRNG" " " INPUT OBJ→ XRNG
   " KEY IN YRNG" " " INPUT OBJ→ YRNG ERASE >>
```

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' = 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 scaler differential equation. The TY desination indicates that the plot is a (T, Y) plot.

Program Name:	**G1.TY**		
Purpose:	Generate a T Y graph 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 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 1.1a: Construct a graph of the solution of $y' + 3y = \cos t$, $y(0) = .3$ for $0 \le t \le 6.283$ using the programs described above. Execute **IN.FN**, respond by typing **<< 'COS(T*Y) - 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, 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 graph on the old one so do not ERASE. Now press **DRAW**. Note the good agreement between the approximate solution and the exact solution.

Exercise 1.2a: Construct and graph solutions of $y' = \sin(t^{1.5}y^R)$, $y(0) = 2$ for $0 \le t \le 8$ 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(1)) for a vector system, say with vectors Y and FN. We will call the program **G.01** where the 0 represents the T variable and the 1 represents the Y(1) 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(1)' EVAL** . The user can avoid retyping the whole program by pressing **'G1.TY' RCL, EDIT,** typing the corrections, pressing **ENTER,** then **'G.01' STO.**

Program Name:	**G.01**		
Purpose:	Generate a T Y(1) graph of the solution to T_f		
Stored Quantities:	**XRNG YRNG FN TOL HS**		
Input	level 3	level 2	level 1
	T_0	vector Y_0	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 T 'Y(1)' EVAL R→C 4
ROLLD DO RKFSTEP T 'Y(1)' EVAL R→C DUP 6 ROLLD 5
   ROLL LINE DUP T + TF UNTIL > END DROP TF T -
RKFSTEP T 'Y(1)' EVAL R→C DUP 6 ROLLD 5 ROLL LINE
   DROP TF T - RKFSTEP T 'Y(1)' EVAL R→C 5 ROLL
            LINE 3 DROPN >> PICTURE >>
```

Exercise 1.3a: Construct a composite graph of the solutions of x" + R x' + x = 0, x(0) = 0, x'(0) = 1, for $0 \le t \le 6.283$ 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 **<< '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], and 6.283 on the stack and execute **G.01**. We change **R** to each of the numbers 2 and 2.5 and place input quantities 0, [1 0], 6.283 on the stack and execute **G.01** twice more to overlay graphs of the other two solutions.

A similar change to **G.01** 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**.

Program Name:	**G.12**		
Purpose:	Generate a (Y(1), Y(2)) graph of the solution from T_0 to T_f		
Stored Quantities:	**XRNG YRNG FN TOL HS**		
Input:	level 3	level 2	level 1
	T_0	vector Y_0	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 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 1.5: Construct an x vs x' graph of the solution of x'' + .5 x' + x = 0, x(0) = 0, x'(0) = 1, for $0 \le t \le 6.283$. 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 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 -.8 .8. **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 graph of the approximate solution consists of a set of points

$$\{(y_1(t_i)\ ,\ y_2 t_i)\ ):\ 1 = 1, 2, ...\}$$

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.01**, **G.TY**, **IN.FN**, **IN.PP**, **T**, **Y**, **FN**, **TOL**, **HS**, **EQ**, and **PPAR** in this order. 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).

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 $t_k \le t \le t_k + h$ 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:	**EULER**			
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 **<< → T Y 'Y' >>** 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 2.1: To obtain approximate values of the solution of y' = sin(ty), y(0) = 3, enter the program **F.N** given by **<< → T Y 'SIN(T*Y)' >>**, 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: **<< 1 N START EULER NEXT >>.**

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 $t_k \le t \le t_k + h$. The algorithm is given by

$$t_{k+1} = t_k + h,\ y_{k+1} = y_k + h[f(t_k, y_k) + f(t_{k+1}, y_k + hf(t_k, y_k))]/2.$$

(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:	**IULER**			
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}

Instruction	Resulting stack
<< DUP2 DUP2 EULER	t y t y t+h y+hf(t,y)
F.N 3 ROLLD	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 **<< → T Y 'Y' >>**, **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 calculator. Press [↱] **SOLVE**, use the [∇] arrow key to select **Solve Diff eq...**, press **OK**, enter the F function **Y**, set the range of the independant 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 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 2.2: 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

```
<< → T Y ' (Y^2+Y)/T ' >>
```

Exercise 2.3: Try the calculator's **Solve diff eq...** algorithm for the F function and initial condition given in exercise 2.2 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:	**GRAF**			
Purpose:	Graph 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 1 N START IULER DUP2 R→C PIXON 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) (e. g. '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→ 'FN(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 indicated in section 1, it is often desirable to plot solutions of several initial value problems on the same graph. Of course, graphs can be combined simply by not erasing the previous result.

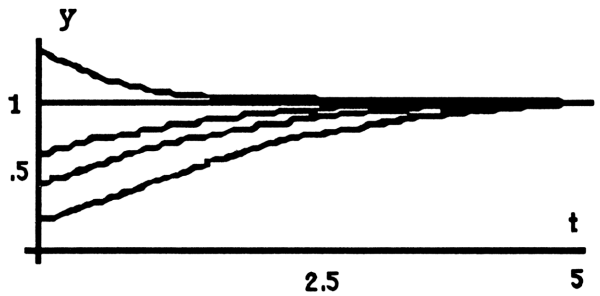
Exercise 2.4: Consider the following differential equation together with several initial conditions and plot the solutions on the same graph.

$$dy/dt = y(1-y), \quad y(0) = .2, .4, .6, 1.5$$

where the solutions are plotted for $0 \le t \le 5$ and step size h = .05 is used. Try

F.N: `<< → X 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.



Five solutions of dy/dt = y (1 - y)

3. First Order Differential Equations

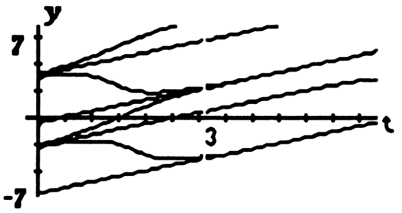
Now that we have introduced the reader to the differential equation features on the HP-48G series calculators and to the construction of some simple programs, it is time to suggest exercises and activities that use graphical and numerical computations to enhance the study of differential equations.

We will often be interested in constructing graphs of several solutions of an initial value problem. The figure constructed above for the differential equation $y' = y(1-y)$ is an example. It is often the case that there are 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 $F(t, y_e) = 0$. In fact, any solution of this equation is an equilibrium solution. 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. Moreover, if the function $F(t, y)$ is continuous and has continuous derivatives then solutions $y_1(t)$ and

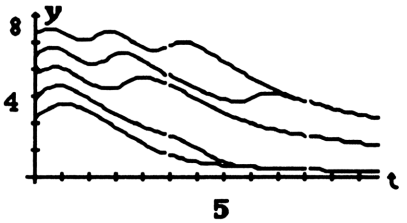
$y_2(t)$ which satisfy different initial conditions do not intersect; consequently, constant solutions restrict where nearby solutions can proceed.

Exercise 3.1: Graph 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 $0 \le t \le 5$ all in the same picture. Then add the graphs of the solutions of the same differential equation that satisfy $y(0) = -.25$ and $y(0) = .25$.

Exercise 3.2: 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 $0 \le$ "t" ≤ 5 and $0 \le y \le 1.2$) on the same plot. In this case, we notice that the solutions are similar. In which case is a change of concavity apparent ?



dy/dt = 2 sin (t - y)



dy/dt = cos (.5ty)

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 $a = -6.81$. (Make the transformation $t - y = w$ to see why.) 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 for the straight line solution $y = t - 2.618$, etc. And even though the functions $y = (2n+1)\pi/t$ ($n = 0, \pm 1, \pm 2, \ldots$) 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.

Exercise 3.3: 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 $0 \le t \le 8$. Use vertical dimension to show $-8 \le y \le 8$. 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 3.4: Plot the graph of the differential equation $dy/dt = \sin(ty)$ with initial condition y(0) = 3 with plot parameters to show $0 \le t \le 6$, $0 \le y \le 5$. Select a new starting point y(0) and add the new trajectory. (If we 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 3.5: Graph 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 (graphic screen parameters $0 \le$ "t" ≤ 5 and $0 \le y \le 1.2$) on the same plot. Notice that the solutions are similar.

Exercises 2.4, 3.1 and 3.5 give 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 3.5, 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. See [1] for HP-48S/SX calculator programs that can be used to determine parameter values to data points containing more than three points by the least squares method.

Exercise 3.6. Suppose p_0 is known and two other population data points, say (t_i, p_i) and $(t_k\ p_k)$. How can the constants a and b in the logistic model be determined from

this information? From the formula for p, we solve for bp_0, then evaluate the expression at the point t_i and t_k and set them equal to get

$$(1 - e^{-at_k})(\frac{p_0}{p_i} - e^{-at_i}) = (1 - e^{-at_i})(\frac{p_0}{p_k} - e^{-at_k})$$

This expression in the unknown a can be solved by graphing both sides and using the **ISECT** key. Find the value of a when $p_0 = 1$, $(t_i, p_i) = (1, 1.46)$ and $(t_k, p_k) = (2, 1.5)$, then find b from the expression for bp_0.

Exercise 3.7. 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 $\{(t_1, p_1), (t_2, p_2), \ldots (t_n, p_n)\}$ 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 $\ln p = A/B - 1$.

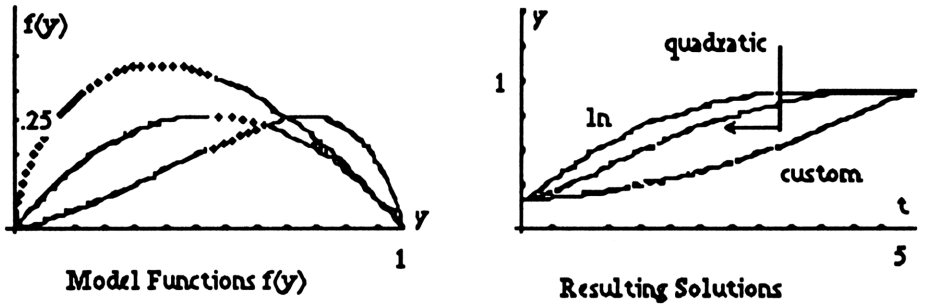
To get a model, say with inflection at p = .6 K, we could try $p' = f(p) = (.6K)^2 - (p - .6K)^2$ for $0 < p \leq .6K$, and $p' = f(p) = 2.25\ (\ (.4K)^2 - (p - .6K)^2)$ for $.6K < p \leq K$.

Exercise 3.8. Use your calculator to obtain a graph of the solution of this model for K = 1, p(0) = .2, **H-VIEW** = .2 5, **V-VIEW** = -.1 1.1. We will suggest a method to enter an appropriate F function using the HP input form format. Press **PLOT**,

CHOOS, Diff Eq, press **OK**, then position the highligted 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 3.9. 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. Hint: to plot data first, store the data list in a variable in **D.LST**, set the **XRNG** and **YRNG** parameters, **ERASE**, and then **EVAL** the program

<< D.LST OBJ→ 1 SWAP START PIXON NEXT DRAX PICTURE >>.



Logistic, Gompertz and Custom Models (exercise 3.8)

Exercise 3.10: Set plot parameters to show $-.5 \le t \le 12.56$, $-.5 \le y \le 3$. 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.)

Exercise 3.11: 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 $f^{-1}(g)$. **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) $f(v) =$ **IFTE**$(v \le 1, (1.5)^{.5}\ v, (2.5\ v - 1)^{.5})$

(d) $f(v) =$ **IFTE**$(\ v \le 1, .75\ v^{1.5}, 1.25\ v - .5)$ (e) $f(v) = 2^{1-r}\ v^r$.

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 $0 \le v \le 2$. Use **H-VIEW** = -.1 2 and **V-VIEW** = -.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 $0 \le t \le 5$. Accumulate them in the same picture for comparison. Again label the solutions. Use the **V-VIEW** as above and **H-VIEW** -.2 5.

A particle falls or is projected from a great height and observations are made on v for, say, 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

$$v(t) = v_0\, e^{-kt} + v_\infty (1 - e^{-kt}),\ v_\infty = g/k\ .$$

- quadratic air resistance model: $dv/dt = g - kv^2$, $v(0) = v_0$. The solution is

$$v(t) = v_\infty \frac{Me^{\sigma t} - 1}{Me^{\sigma t} + 1}\ ,\ M = \frac{v_\infty + v_0}{v_\infty - v_0}\ ,\ v_\infty = \sqrt{\frac{g}{k}}\ ,\ \sigma = 2\sqrt{gk}\ .$$

Suppose that v_∞ can be accurately determined from data, say $\{(t_1, v_1), (t_2, v_2), \ldots (t_2, v_2)\}$. 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 3.12. 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 $dv/dt = 2 - a\, v^r$, $v(0) = 0$, 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 $dv/dt = g - a\,v^r$ and **G.TY** can be modified to plot the log $(v_\infty - v(t))$ by inserting the commands **V_∞ - ABS LN** after the **Y** instructions.)

Suppose a tank 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 – rate that the chemical exits from the tank. This application gives one example of an **important** problem, namely, to determine a particular solution of

$$\frac{dy}{dy} + ry = f(t).$$

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 αt, gives y(t) = sin $(\alpha t-\theta)/R^2$ with $R^2 = (\alpha^2+r^2)$ and cos θ = r/R, sin $\theta = \alpha/R$.

There is an obvious similarity between the graphs of the input and output functions. If a signal f(t) = sin t is input into a device and produces output as described above and it is desired to produce a "delayed" version of the signal, say sin $(t-\pi/4)$, after the second term dies out, what value of r will give this delayed signal? 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 3.13: Let r = 1, and set the plot parameters so $0 \le t \le 3.14$, $0 \le y \le 1.2$. Use the calculator to draw the following input functions with the **Function DRAW** program and the resulting output functions with a differential equation plotting program.

(a) $f(t) = 1 - \sin^4(3t)$, (b) $f(t) = 1 - \sin^{10}(3t)$,

(c) f(t) = Max (sin 6t, 0).

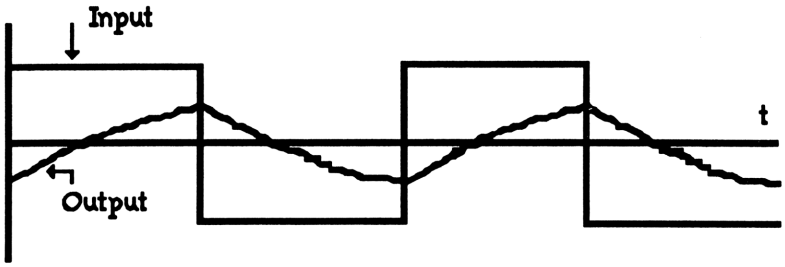
If f(t) is stored in **EQ** and P = 1.047 ≈ $\pi/3$.) The following program can be used to calculate y(0):

```
<< 3 FIX 'T' PURGE 0 1.047 EQ 'EXP(T)' * 'T' 3 NEG
SF ∫  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 leads one to suspect that the starting condition y(0) = 0 is being forgotten and 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*CEIL(SIN(t*π)) - 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 $0 \le t \le 4$.



dy/dt + y = f(t)

The student can also construct the input function shown above as **IFTE**(T MOD 2 ≤ 1, 1, -1). In the same way, the switch function $u_a(t) = 0$ when $t \le a$ 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 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 $0 \le x \le .5$, f(x) = 1 - sin (x-.5) for $.5 \le x \le .5 + 1.571$, x^2 for $x > .5 + 1.571$ is produced by **'IFTE(X ≤ .5, 2*X, IFTE(X ≤.5+1.571, SIN(X-.5), X^2))'**.

4. Initial Value Problems: Two Differential Equations

As the reader may have guessed from section 1 of this chapter, it is easy to program the calculator to treat a vector differential equation. In this section we 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.

(The student should note that a second order initial value problem

$$\frac{d^2x}{dt^2} = g(t, x, \frac{dx}{dt}), \quad \text{with } x(t_0), \ \frac{dx}{dt}(t_0) \text{ given}$$

can be reduced to a first order system of differential equations

$$\frac{dy_1}{dt} = y_2, \qquad \frac{dy_2}{dt} = g(t, y_1, y_2)$$

and initial values of y_1 and y_2 by using the identification $y_1 = x, y_2 = x'$.) We will graph trajectories and study the solution characteristics of such systems. Of course, in this case we can graph y_1 versus t, y_2 versus t or graph y_1 versus y_2 as the parameter t varies.

As in the case of a single differential equation, the reader may decide to use the built-in plotting form, the user programs as created in section 1 of this chapter, or to incorporate the Euler or modified Euler algorithms in graphing programs. 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. Consider

$$\frac{dy_1}{dt} = y_2, \qquad \frac{dy_2}{dt} = \cos t - y_1, \ y_1(0) = y_2(0) = 0,$$

over the interval $0 \le t \le 2\pi$. An appropriate **F.N** program is

```
<< DUP → Y 'Y(2)' → T Y 'COS(T) - Y(1)' 2 →ARRY >> .
```

(The first stack item, namely Y, is duplicated and is used as local variable to create the first component of the output and the input pair T Y is used as local variables to create the second component of the output.) 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**. We can modify the **GRAF** program to the following form:

Program Name:	**GR.01**
Purpose:	Graph N values of (T,Y(1)) resulting from the improved Euler algorithm which creates a sequence of N values of t, y1 and y2.
Stored Quantities:	**N H F.N EULER IULER XRNG YRNG**

Input		Output	
level 2	level 1	level 2	level 1
t_0	[$y1_0$ $y2_0$]	t_n	[$y1_n$ $y2_n$] & graph

<< { # 0d # 0d } PVIEW DRAX 1 N START IULER DUP2 OBJ→ DROP2 R→C PIXON NEXT PICTURE >>

Consider the problem of graphing the vector solution of $dy_1/dt = y_2$, $dy_2/dt = -y_1$, $y_1(0) = 1$, $y_2(0) = 0$ on the interval $0 \le t \le 2\pi$. We execute the program **IN.FN** and respond with

<< 'Y(2)' EVAL 'Y(1)' EVAL NEG 2 →ARRY >>

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 graph 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 that the result as drawn by **GR.12**, say with **H** = .0628 and **N** = 100.

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:

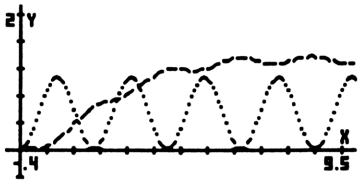
$$\frac{d^2x}{dt^2} + 2r\frac{dx}{dt} + \omega^2 x = f(t),\ x(0) = \frac{dx}{dt}(0) = 0,\ \omega^2 > r^2 .$$

The solution is given by

$$x_q(t) = \frac{1}{\mu}\int_0^t e^{-r(t-s)} \sin \mu(t-s)\, f(s)\, ds\ ,\ \mu = \sqrt{\omega^2 - r^2}\ .$$

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 $0 \le t \le 9.42$, $0 \le y \le 2$. Put 0 [0 0] 9.42 on the stack and execute **G.01**. Next overlay a graph for the input function. The forcing function (input) and solution (output) resulting from this program are shown below.



Exercise 4.1: Find the output graph for $f(t) = 1 - \sin^4(3.14*t)$ for $\mu = 1$, and $r = .5$. Choose plot parameters to show $-.4 \le t \le 6$, $-.4 \le y_1 \le 1.2$. Add the input function graph as an overlay. Comment: The output function for this input function can be

obtained from a table of integrals after several substitutions using the method of undetermined coefficients and a lot of work. But, an output function for an input function such as $f(x) = 1/(2 - \sin^4(3.14*t))$ could not be found this way.

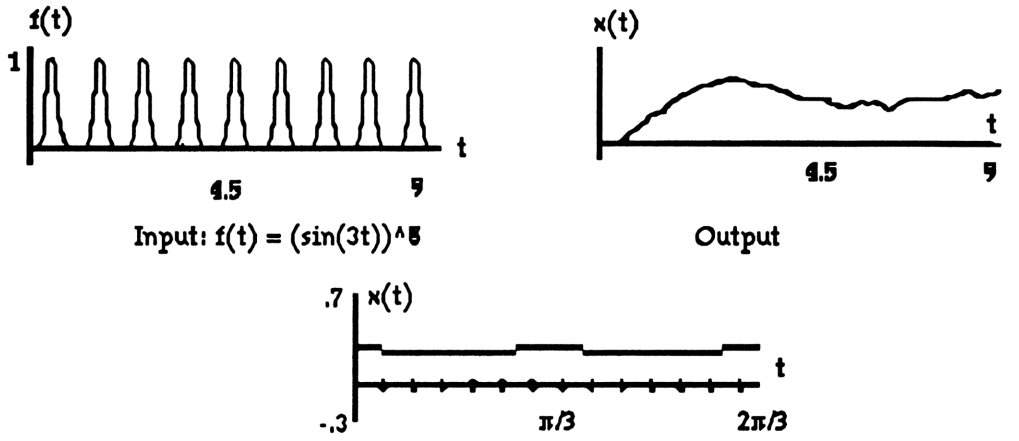
Suppose the forcing function f(t) is periodic with period length P for the differential equation. If we can change the initial conditions so that x(P) = x(0) and x'(P) = x'(0), then the resulting solution is periodic. Moreover, if the damping coefficient r > 0, then all solutions will eventually be close 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 $x_2(t) = (1/\mu) e^{-r\,t} \sin \mu t$. We can use the calculator to compute the integrals in $x_q(P)$ and $x'_q(P)$ numerically, 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.01** with input 0 [a b] P 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 is periodic with 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

function with value $\pi/(6\omega^2)$ = .493. This seems to be the constant part of the periodic solution as shown.



Periodic Response

Exercise 4.2 : Find and graph the periodic output response for $f(x) = 1 - \sin^4(3.14t)$ for $\mu = 1$, and $r = .5$. Then add the graph of the input forcing function.

The friction/resistance term in the spring/circuit model that we have been considering is given by 2rdx/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 4.3 : Find a periodic solution to nonlinear problems of the form

$$\frac{d^2x}{dt^2} + R(\frac{dx}{dt}) + K(x) = 2 \cos t.$$

Set **H-VIEW** (i. e., **XRNG**) =-.2 6.283, **H-VIEW** (i. e., **YRNG**) = -2.1 2.1 . Let

$R_1(dx/dt) = 2*IFTE(dx/dt \leq -1, dx/dt + .5, IFTE(dx/dt \leq 1, .5*dx/dt, dx/dt - 1))$.

(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)?

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

$$\frac{d^2x}{dt^2} + \omega^2 x = .5 \cos t, \quad \omega \neq 1,$$

together with the initial condition $x(0) = \xi$, $dx/dt(0) = 0$ has solution

$$x(t) = [\xi + \frac{1}{2(1 - \omega^2)}] \cos \omega t - \frac{1}{2(1 - \omega^2)} \cos t$$

$$\frac{dx}{dt}(t) = -\omega[\xi + \frac{1}{2(1 - \omega^2)}] \sin \omega t + \frac{1}{2(1 - \omega^2)} \sin t$$

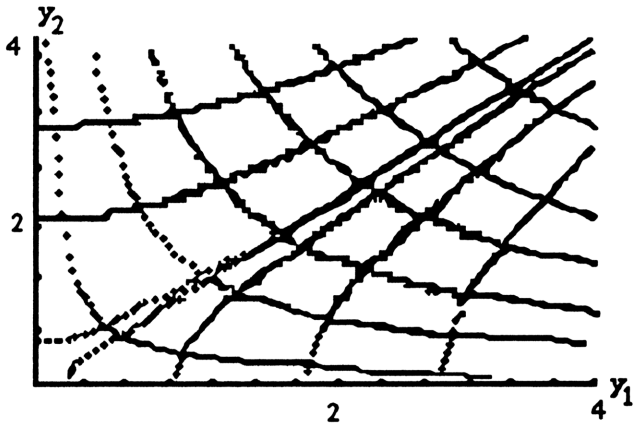
What are the properties of such a solution ? For $t = 2\pi n$ we have

$$\frac{x(2\pi n) + \dfrac{1}{2(1-\omega^2)}}{\xi + \dfrac{1}{2(1-\omega^2)}} = \cos 2\pi n\omega, \qquad \frac{\dfrac{dx}{dt}(2\pi n)}{\omega\left[\xi + \dfrac{1}{2(1-\omega^2)}\right]} = -\sin 2\pi n\omega.$$

By squaring both sides, we see that the points $x(2\pi n)$, $x'(2\pi n)$ lie on an ellipse.

Exercise 4.4: Plot the points $x(2\pi n)$, $x'(2\pi n)$ for $\xi = 0$ and $n = 1, 2, \ldots$ (several values of n) for $\omega = 1/\mathrm{Sqrt}(5)$ and for $\omega = 1/3$. Note that the points cycle around the ellispe. If ωn = an integer m for some integer n, then you can see the solution is periodic, but what happens when ω is irrational ?

A topic occuring 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 which are orthogonal satisfy the differential equation $dy_2/dxy_1 = -1/m(y_1, y_2)$. If the original family is given in 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



Orthogonal Trajectories $dy_1/dt = -y_2/y_1$, $dy_2/dt = y_1/y_2$

preferred if the curves in either family must be specified in terms of a parameter t. Clearly, the program **G.12** can be used to sketch 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 above.

Exercise 4.5: Set the plot parameters to show both **H-VIEW** and **V-VIEW** as -.5 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 graph in the y_1-y_2 plane consisting of the following inputs to the **G.12** program.

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

Graphs 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 graphs. 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 $t \to \infty$, (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 $t \to \infty$, (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 4.6: Systems called **Lotka-Voltera** 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 initiate in the first quadrant are periodic. Plot the solution that starts at 0 [2 2], for $0 \le t \le 2.25$, after setting the plot parameters to show **H-VIEW** 0 6, and **V-VIEW** 0 6, by using the plot program **G.12.**

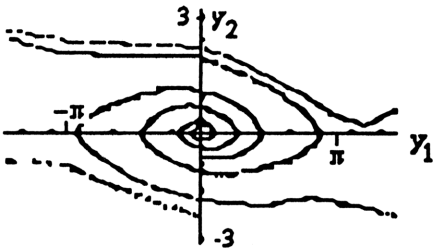
Example. The differential equations

$$x'' + cx' + \sin x = 0 \text{ or } y_1' = y_2,\ y_2' = -\sin y_1 - cy_2$$

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)

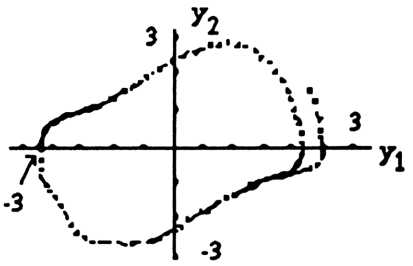
Example. The system

$$dy_1/dt = -2y_2 + y_1(1-r^2)/r,\ dy_2/dt = 2xy_1 + y_2(1-r^2)/r:$$

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 >>.
```

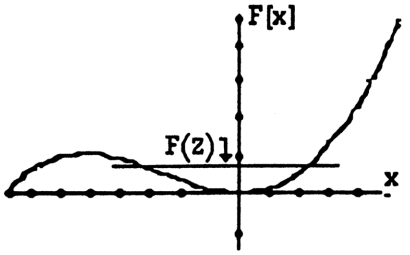
Another problem which has an isolated attracting periodic solution is the Van der Pol differential equation. This equation was studied in connection with its application to an electronic component. This example in usually studied as a function of a parameter μ contained in the "damping" term. Our figure shows



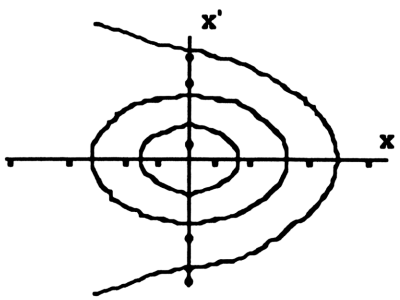
$$dy_1/dt = y_2,\ dy_2/dt = -[y_1 + .3(y_1^2 - 1)y_2]$$

typical graph: here $\mu = .3$. Note the motion is counterclockwise and 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 μ causes dramatic changes in the shape and period of the solution.

Exercise 4.7: In this exercise we will examine the cycle time 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 suppose that y(t) satisfies the initial value problem

$$\frac{d^2x}{dt^2} + f(x) = 0, \quad x(0) = z, \quad \frac{dx}{dt} = 0,$$

where the essential feature of f(x) is that it changes sign from negative to positive as y increases thru zero. We multiply by dx/dt and integrate from 0 to t to obtain

$$\frac{dx}{dt} = \pm\sqrt{F(z) - F(x)}, \text{ where } F(x) = 2\int_0^x f(s)\,ds.$$

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 y_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) 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.

Linear autonomous systems can be solved analytically. These systems have the form:

$$dy_1/dt = a_{11}y_1 + a_{12}y_2, \quad dy_2/dt = a_{21}y_1 + a_{22}y_2$$

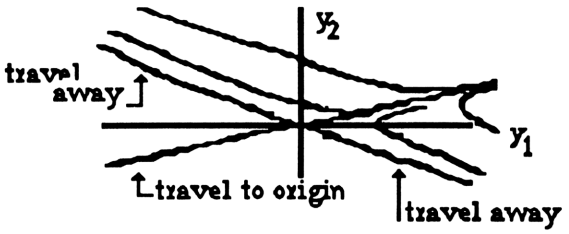
We will consider the case $\det(A) \neq 0$, which means that the origin (0,0) is the only critical point. Special solutions have the form $w = \text{column}\,[y_1, y_2] = e^{\lambda t} v$ where λ is a solution of the equation $\det(A - \lambda I) = 0$ and v will be given below. Such a number λ is called 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 λ 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

$$dy_1/dt = y_1 - 4y_2\,, \qquad dy_2/dt = -y_1 + 2y_2.$$

The associated matrix A has eigenvalues $\lambda = .5(3 \pm 17^{.5})$ and corresponding eigenvectors c = column [4, 1.56] and c = 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.



Trajectories near Saddle Point

Exercise 4.8. For the case λ 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

$$dy_1/dt = -.5y_1 + 4y_2\,, \qquad dy_2/dt = -4y_1 - .5y_2$$

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 $0 \le t \le 3$.

Exercise 4.9. Use G.12 to graph the trajectories initiating at (t, $[y_1, y_2]$) = (0, [0, 1]) and at (0, [-1, 0]) 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.

Solution graphs of ***nonlinear autonomous systems*** near a critical point solution can be studied using a linear approximation. Let the vector y = column $[y_1, y_2]$ and suppose we have the system dy/dt = F(y) for F(y) = 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

$$\frac{\partial F_i}{\partial y_i}(yc)$$

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 labelling of the independant variables can be specified by the user. For example, instead of y1, y2, etc. the user might prefer u, v, .. The user's preference will be entered into a stored list as shown. After the matrx 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} →ARRY 'JMAT' STO >>
```

For m = 2, the eigenvalues are the roots of the quadratic polynomial λ^2 - (**JMAT**[1,1]+**JMAT**[2,2])λ + (**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_o = column $[u_o\ v_o]$ of the critical point is known. Then better approximations of the critical point may result from one or more applications of the following algorithm:

$$y_n = y_0 - J^{-1}(y_0)\,F(y_0),\quad y_n \to y_0.$$

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} →ARRY '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 4.10: 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 4.12: Find a critical point of the system

$$du/dt = u - \sin u * \cosh v,\ dv/dt = v - \cos u * \sinh v$$

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

$$dy_1/dt = a_{11}y_1 + a_{12}y_2,\ dy_2/dt = a_{21}y_1 + a_{22}y_2\ .$$

The failure of the default algorithm occurs for such systems when the eigenvalues λ_1, λ_2 of the matrix A made from the coefficients are both negative and λ_1/λ_2 is a large number. This inducates 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 λ_1/λ_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

$$y_1' = y_2,\ y_2' = -1000\, y_1 - 1001\, y_2.$$

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 adaption 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 **RSBSTEP** consist of four elements. The last element is an indicator variable (in this case 2) which determines the method to be used.

Program Name:	**GS.01**
Purpose:	Generate a T Y(1) graph of the solution to T_f.
Stored Quantities:	**XRNG YRNG FN FNY FNT TOL HS**
Input	level 3 level 2 level 1
	T_0 vector Y_0 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 FNY FNT } TOL HS 2 T 'Y(1)'
EVAL R→C 5 ROLLD DO RSBSTEP DROP 2 T 'Y(1)' EVAL
R→C DUP 7 ROLLD 6 ROLL LINE SWAP DUP T + TF
UNTIL > END DROP TF T - RKFSTEP T 'Y(1)' EVAL R→C
DUP 6 ROLLD 5 ROLL LINE DROP TF T - RKFSTEP T 'Y(1)'
EVAL R→C 5 ROLL LINE 3 DROPN >> PICTURE >>
```

Exercise 4.13: 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. The exact solution is $y_1 = e^{-t}$, $y_2 = -e^{-t}$. Use the **Function** mode to overlay the solution as an accuracy check.

5. Linear Autonomous Systems of Differential Equations

In this section 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 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 that are constructed to avoid the computational pitfalls which are sometimes encountered.

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 satisfing the differential equation can be determined and placed in the columns of a matrix Y(t), 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 nth 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 λ 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 λ, 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 will be 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

```
<< 'λ' PURGE A DET 8 RND → δ1 << 'λ^2 + (A(1,1) +
A(2,2))*λ - δ1' EVAL >> >>
```

Note: The Greek letters can be entered in your program by pressing [↱] **CHAR**, using the cursor to highlight the appropriate character, pressing **ECHO** and then the **ON** key.

Exercise 5.1: Find the eigenvalue (or characteristic) equation for the matrices

$$\begin{bmatrix} -3 & 4 \\ -2 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ -1000 & -1001 \end{bmatrix}$$

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

```
<< 'λ' PURGE A DET 8 RND → δ1 << 'λ^3 - (A(1,1) + A(2,2) +
  A(3,3))*λ^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))*λ - δ1' EVAL >>
                              >>
```

The program will display the eigenvalue equation as a cubic in λ.

Exercise 5.2: 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 λ^3 - 6λ^2 + 11λ - 6.)

We can find the roots of the eigenvalue equation graphically by isolating one or more roots, by storing the equation (**STEQ**), and using the **DRAW** and **SOLVR** programs. You may have to try several settings of the plot parameters **XRNG**, **YRNG**.

Exercise 5.3: Find the eigenvalues of the matrices given in exercise 5.2. (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 - x - 1. A zero of this equation obtained by **ROOT** after drawing the curve from -2 to 2 or by the **SOLVE** routine is x = 1.3247---. If we divide x - 1.3247--- into x^3 - x - 1 we obtain the quotient x^2 + 1.3247---x +(1.3247--^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 ± i .563. An alternative method is to press [↱] **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.)

Exercise 5.4: 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 λ 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** to obtain the Gauss-Jordon echelon form to determine the row space of $(A-\lambda I)$ and nontrivial solution vectors v.

Program Name: **PIV** (Adapted from D. R. LaTorre)
Purpose: Gauss pivot on element **K L**
Input: Matrix **A**, integers **K L** Output: Altered matrix **A**

```
<< → 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 >> >>
```

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 5.2 has eigenvalues 1, 2, and 3.

For $\lambda = 1$ the 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** is given we get

$$\begin{bmatrix} 1 & -1 & 1 \\ 0 & 2 & -1 \\ 4 & -4 & 4 \end{bmatrix};$$

now give the command 1,1 **PIV** to get

$$\begin{bmatrix} 1 & -1 & 1 \\ 1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix};$$

now 2,2 **PIV** gives

$$\begin{bmatrix} 1 & 0 & .5 \\ 0 & 1 & -.5 \\ 0 & 0 & 0 \end{bmatrix}.$$

The solution relations $v_1 = -.5\ c_3$, $v_2 = .5\ v_3$ result: i. e., v = [-1, 1, 2] or any nonzero multiple of this vector. 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 = 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 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 the eigenvector v = 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 5.2 we determined eigenvalues and corresponding eigenvectors in the example just after the **ROKL** program. Thus

$$Y(t) = \begin{bmatrix} -e^t & -2e^{2t} & -e^{3t} \\ e^t & e^{2t} & e^{3t} \\ 2e^t & 4e^{2t} & 4e^{3t} \end{bmatrix};$$

Exercise 5.5: Give the solution of $y' = Ay$, with conditions: $y(0) = [1 \;\; 3 \;\; -5]^t$.

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

$$e^{\alpha t} (\cos \beta t\, a - \sin \beta t\, b) \text{ and } e^{\alpha t} (\sin \beta t\, a + \cos \beta t\, b)$$

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 5.6: 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 λ 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 $y(t) = e^{\lambda t} (vt + d)$. Here, v will be an eigenvector and $(A - \lambda I)^2 d = 0$. 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 = column [1, -1, 2] and the

eigenvectors corresponding to $\lambda = -1$ are multiples of v = column [1, -1, 3]. The equation $(A+2I)^2\ d = 0$ has a solution d = column [0, 1, 0]. (Such a vector is easily obtained on the calculator, first by calculating $(A+2I)^2$, then using **PIV** 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 = column [1, -1, 0] and c = column [-3, 0, 2]. The eigenvectors corresponding to $\lambda = -2$ are multiples of c = 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 5.7: 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 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.

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 $-1 \pm i2$ and have congugate 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 5.8: Find a fundamental matrix of solutions for y' = Ay when

$$A = \begin{bmatrix} -2.5 & 2.5 & -3.5 & -.5 \\ 1 & 2 & 2 & 1 \\ 5 & 4 & 6 & 6 \\ -4.5 & 8.5 & -5.5 & -7.5 \end{bmatrix}$$

To obtain solutions of the nonhomogeneous equation y' = Ay + f(t), 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 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. If the functions in f(t) are elementary, we can use the method of undetermined coefficients to construct a particular solution. The computation of the coefficients will require the solution of linear algebraic equations.

References

1. Proctor, T. G. , *Calculator Enhancement for a Course in Differential Equations,* (Philadelphia: Saunders College Publishing, 1992).

Appendix to Chapter 3. Direction Fields

The following is a set of programs that will construct a direction field for

$$\frac{dy_1}{dt} = FN_1(y_1, y_2), \qquad \frac{dy_2}{dt} = FN_2(y_1, y_2).$$

We assume that **FN** is a stored program of the type discussed in section 1 of this chapter. These programs are provided to allow the user to use the built-in differential equation solver to overlay particular solutions with minimal additional instructions.

Program Name:	**DIRF**
Purpose:	Generate a direction field in the region prescribed by **XRNG, YRNG.**
Stored Quantities:	**FNA GNA**
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→ DF.1 PICTURE >>
```

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 ROLLD DF.2
NEXT DROP NEXT DROP >> >>
```

Subprogram Name: **DF.2**

```
<< 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 section 1 of this chapter 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 containing the programs from section 1, placing **DIRF** 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 $-2 \le y_1 \le 2$, $-1.5 \le y_2 \le 1.5$. Overlay the trajectory which begins, $t = 0$ at $y_1 = 0$, $y_2 = 1.5$ and 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 this chapter.

HP-48G/GX Calculator Enhancement
for
Linear Algebra

Donald R. LaTorre

Linear Algebra has long been an established staple in the undergraduate mathematics curriculum. But, as the pervasive growth of computers has stimulated the widespread applicability of matrices, students in a variety of disciplines now need an understanding of linear algebra. Since most students will take only one course in the subject, it is important that the course address not only fundamental concepts but also be responsive to the needs of the client disciplines it serves. Thus, there is a considerable and growing impetus for introductory courses to be matrix oriented [1].

This chapter is intended to show how the HP-48G and 48GX super calculators may be used in such a matrix-oriented course. We do not intend the material to be definitive; only that it show some of the possibilities for the effective use of the calculator. A more extensive treatment can be found in [2].

After a brief preliminary section on calculator issues, section 2 is concerned with procedures for entering, editing and manipulating matrices and vectors. In addition to routine matrix arithmetic, we also consider the computation of determinants and inverses.

Section 3 is devoted to systems of linear equations, and considers Gaussian elimination with partial pivoting and back substitution, its interpretation as an LU-factorization, and Gauss-Jordan reduction.

Section 4 addresses the important concept of orthogonality, including the Gram-Schmidt orthonormalization process, its interpretation as a QR-factorization, and applications to over determined systems arising from polynomial curve fitting.

Section 5 considers eigenvalues and eigenvectors, and ends with the question of diagonalization.

At the end of each section we have included some ACTIVITIES. These are short collections of HP-48G/GX based explorations, investigations and projects that students can engage to help reinforce and extend their understanding of the basic material.

1. PRELIMINARIES

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 [↰] [SOLVE]. 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

key	location
TRN	MTH MATR MAKE menu
RREF	MTH MATR FACTR menu

- Menus generally have more labels than can be shown above the six white menu keys, and the NEXT key will display the next row (page) of labels. Return to the previous page with PREV. We will not show these two keys.
- The four white arrow keys will be indicated by △, ▽, ◁ and ▷.
- Calculator operations and commands that appear in programs or in the text material will be in helvetica characters, e.g., DUP SWAP INV.

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 calculators recognize parentheses as object delimiters. *Unless we specify otherwise, the examples and exercises assume the calculator is set to STD display mode.*

Programming. Unless you obtain our programs by calculator-to-calculator infrared transmission, from a RAM card or by computer-to-calculator serial transmission, you will need to copy and enter them into your calculator. In doing so, you must be careful and copy the programs exactly as we show them. Special attention should be given to correct spacing because the calculators recognize commands that are separated by spaces. Instead of spelling out commands from the alphabet keyboard, we recommend that you use the keystroke commands that appear either as shifted keys (e.g., SWAP,

DROP, PURGE) or as labels on the various menus; keystroke commands will automatically insert spaces around each command.

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

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

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

Because this is a brief chapter directed towards the teaching of introductory courses in linear algebra, we shall not have occasion to use all of the built-in commands.

But we call them to your attention in the spirit of showing some of the possibilities for the effective use of the HP-48G/GX in both introductory and more advanced studies.

2. ARRAYS

On the HP-48 units, rectangular arrangements of real or complex numbers are called *arrays*. Arrays can be one-dimensional (vectors) or two-dimensional (matrices) and are considered to be single objects. Consequently, they can be manipulated with many of the same basic commands used in ordinary arithmetic. We shall begin by examining some of the ways of entering, editing, and manipulating arrays.

Entering Arrays

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

```
[ [ (1,1), (1,2) ]
  [ (2,1), (2,2) ]  .
  [ (3,1), (3,2) ] ]
```

Using the command line. The vector [1 2 3] is entered with keystrokes ↰ [ ] 1, 2, 3 ENTER. Be sure to insert spaces between the 1, 2, 3 with the SPC key. To enter a matrix, start with [[by pressing the [ ] key twice, enter the first row and press ▷, then continue entering the remaining entries in row order and press ENTER.

EXAMPLE: Keystrokes

[ ] [ ] 1, 2, 3 ▷ 4, 5, 6, 7, 8, 9 ENTER

will produce the matrix

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

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

The numbers may be any mixture of real or complex numbers (ordered pairs), but if any one entry is complex then the entire array will be complex.

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

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

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

Clearly, for entering simple matrices (say, with 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 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 extremely versatile environment for entering, reviewing and editing both *vectors and matrices*. To enter a vector using the MatrixWriter, say vector [1 2 3 4], start with an empty stack and enter the MatrixWriter environment with [↱] [MATRIX]. Note that the menu key [VEC□] appears. If you press 1, 2, 3, 4 [ENTER] [ENTER], *vector* [1 2 3 4] will show on the stack. The presence of the white box in [VEC□] indicates that vector entry is active. If you toggle off this key to see [VEC] without the box, the keystrokes 1, 2, 3, 4 [ENTER] [ENTER] will return the *matrix* [[1 2 3 4]].

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

A final note about entering arrays using the MatrixWriter application. Array entries may 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).

Note: As a matter of convenience, any n-vector $x = [x_1\ x_2\ \ldots\ x_n]$ may be premultiplied by any m×n matrix A to obtain Ax. Thus, in this context, x is treated as if it were an n×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].

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, where the white arrow keys then let you move to any desired entry and change it. You may 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 may 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.

Separating into Rows or Columns. It is often desirable to separate a matrix into its row or column vectors, and the appropriate commands are →**ROW**, located on the MTH MATR ROW menu, and →**COL**, located on the MTH MATR COL menu. For example, with

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

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

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

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

4 [COL→] returns

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

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

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

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

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

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

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

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

```
   [[ 1  2  3]                  [[ 11  12]
A= [ 4  5  6] on level 2 and B = [ 13  14] on level 1, press 2 [COL+] to insert
   [ 7  8  9]]                  [ 15  16]]
```

the columns of B into matrix A, starting at the column 2 position:

```
[[ 1  11  12  2  3]
 [ 4  13  14  5  6] .
 [ 7  15  16  8  9]]
```

ROW+ works similarly.

Matrix Arithmetic

Addition and subtraction of matrices proceeds just as for real numbers. To calculate A+B simply key in matrix A and press [ENTER], then key in B and press [+]. Pressing [−] instead of [+] calculates A–B. Note that the commands [+] and [−] add or subtract the

object on level 1 to or from the one on level 2. In case A and B are stored in user memory, press [A] [B] [+] to add.

To multiply a matrix by a scalar c, key in the matrix and press [ENTER], then key in scalar c and press [*]. Multiplying by -1 can be done with a single keystroke by pressing the [+/-] key.

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

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

A.KTH (K^{th} power of a matrix)

Inputs: level 2: a square matrix A
level 1: an integer K

Effect: returns A^K, the K^{th} power of A

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

EXAMPLE. To calculate B^5 for $B = \begin{bmatrix} 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \end{bmatrix}$, begin by entering the matrix onto level 1. Now press 5 [A.KTH] to see $\begin{bmatrix} 0 & 16 & 0 & -16 \\ 16 & 0 & -16 & 0 \\ 0 & -16 & 0 & 16 \\ -16 & 0 & 16 & 0 \end{bmatrix}$.

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

P.OFA (Polynomial evaluation at A)

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

level 1: a square matrix A

Effect: returns $p(A) = a_n A^n + a_{n-1}A^{n-1} + \ldots + a_1 A + a_0 I$

```
« → L A « A SIZE 1 GET → K « L 1 GET  2 L SIZE
OBJ→ DROP FOR N A *  L 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.OFA to see

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

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

CONJ command, on the MTH CMPL menu, returns the complex conjugate of its input argument.

Determinants and Inverses

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

EXAMPLE. Key in matrix A = $\begin{bmatrix} -4 & 4 & 8 & 8 \\ -16 & 12 & 16 & 16 \\ -8 & 4 & 12 & 8 \\ 8 & -4 & -8 & -4 \end{bmatrix}$ and press ENTER three times to put three copies of A on the stack. Now press DET to show detA = 256. As in this example, the HP-48G and 48GX will always return an integer for the determinant of a matrix having only integer entries. 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 $\begin{bmatrix} 1 & 1 & 1 \\ 3 & 6 & 4 \\ 3 & 6 & 4 \end{bmatrix}$ and multiply it by .101031 to obtain matrix B = $\begin{bmatrix} .101031 & .101031 & .101031 \\ .303093 & .606186 & .404124 \\ .303093 & .606186 & .404124 \end{bmatrix}$. Use ENTER to put two more copies of B on the stack then execute the DET command to obtain detB = 6.1E–17. Drop the determinant, then use 1/x to get

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

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

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

Matrix B, like A, has two identical rows. Thus, detB = 0, so B 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-0 value (the result of round-of error) for the determinant of a singular matrix and then a ridiculous candidate for an inverse.* The numerical calculation of matrix determinants and inverses is extremely sensitive to round-off error, scaling, and choice of numerical algorithm in a floating point environment. *Thus, our advice is to proceed with caution in a calculator environment and,* whenever possible, *avoid calculating determinants and inverses.*

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

$$\begin{bmatrix} 1 & 0 & -.000000000001 \\ 0 & 1 & .000000000001 \\ 0 & 0 & .999999999999 \end{bmatrix}$$

to 11 decimal places. Simply enter 11 RND to obtain

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

ACTIVITIES I

1. 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).

2. 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 partitioned matrices $[A \; B]$ and $\begin{bmatrix} A \\ B \end{bmatrix}$.

   (b) Build the partitioned 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).

3. For an input list {m n}, where m and n are positive integers, the RANM command (located on the MTH MATR MAKE menu) will generate a random m×n matrix with entries from the set $Z_{10} = \{0, \pm1, \pm2, \ldots, \pm9\}$.

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

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

   (c) Formulate a conjecture based upon your observations.

   (d) Prove your conjecture.

4. (a) Seed your calculator's random number generator with 1 by entering 1 RDZ, then generate two random 4×4 matrices A and B with the RANM command (see activity 3, above).

   (b) Combine A and B into the complex matrix A + iB by executing the command R → 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 → R (also on the MTH CMPL menu), SWAP levels 1 and 2 and then recombine the two matrices into a complex matrix with R → C. Now extract column 4.

3. SYSTEMS OF LINEAR EQUATIONS

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.

Gaussian elimination

In its traditional from, the Gaussian elimination algorithm for a square 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, then 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.

The HP-48G and 48GX units include built-in commands on the MTH MATR ROW menu to efficiently perform the row operations that transform [A| b] into [U| b']. With a matrix A on level 1, the RCIJ command is used to multiply row I of matrix A by scalar c and then add the result to row J, and the RSWP command is used to interchange rows I and J. The RCI command is used to rescale row I by multiplying it by scalar c.

To solve this linear system

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

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

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

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

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

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

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

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

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

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

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

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

BACK (Back substitution)

Inputs: level 2: an n×n upper triangular matrix U

level 1: an n-vector b

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

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

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

To speed up the elimination phase without losing control over the process, we encourage our beginning students to use the following program ELIM. Program ELIM pivots on a specified entry - the pivot - to produce zeros below that entry. It is written to 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 to solve the linear system

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

begin with the augmented matrix [A | b]

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

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

```
[[ 5 -9  16  6  48 ]
 [ 0  9  -8 -4 -24 ]
 [ 0  0   8  6   3 ]
 [ 0  0   0 -2   3 ]]
```

Press 5, [COL-] to split off the last column. Then, [BACK] followed by four applications of [CONT] shows [3 -2 1.5 -1.5] as the solution.

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.

We have already seen that 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. 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 any element in the pivot column 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, we require that our students adopt partial pivoting by using the RSWP command to reinforce their understanding of this technique.

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 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 →L P 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 L 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.

L.U (Used to get LU-factorizations)

Inputs: As stored variables: variables L and P, obtained from program →LP(below), each containing an identity matrix.
level 3: a square matrix A
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 diagonal. Press [L] to view L. Used iteratively to obtain an LU-factorization.

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

→ LP (Make L and P)

Input: level 1: a square matrix A

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

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

L.SWP (Interchange multipliers in L)

Input: level 1: a square matrix
level 2: an integer I
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; used to update L by interchanging multipliers.

```
« → A I J « 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 . Get an LU-factorization of $A = \begin{bmatrix} 2 & 3 & -1 & 2 \\ -4 & -6 & 2 & 1 \\ 2 & 4 & -4 & 1 \\ 4 & 8 & 2 & 7 \end{bmatrix}$; use partial pivoting.

Step 1: Enter A onto level 1, and press [→LP] to create appropriate starting matrices L and P. Interchange rows 1 and 2 in A with 1, 2 [RSWP], recall

P to the stack and make the same row interchange, then store the result in P with [↰] [P]. Now press 1, 1 [L.U] to see

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

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

Step 3: Now execute 2, 2 [L.U] to see
```
[[-4 -6  2   1 ]
 [ 0  2  4   8 ]
 [ 0  0 -5 -2.5 ]
 [ 0  0  0 2.5 ]]
```
. Store this as U.

Step 4: Get L =
```
[[  1  0  0  0 ]
 [ -1  1  0  0 ]
 [-.5 .5  1  0 ]
 [-.5  0  0  1 ]]
```
with [L], then do [U] [*] to see

```
LU =
[[-4 -6  2  1 ]
 [ 4  8  2  7 ]
 [ 2  4 -4  1 ]
 [ 2  3 -1  2 ]]
```
.

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

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 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. Why? Assume that A is n×n and that both A^{-1} and the factors L and U are available. Using A^{-1} to obtain $x = A^{-1}b$ requires n^2 multiplications. Solving Ly = b for y by forward substitution and then solving Ux = y for x by back substitution also requires n^2 multiplications. But the difference is seen in comparing the number of multiplications required to obtain A^{-1} to the number of multiplications required to obtain the factors L and U: n^3 verses $\frac{n^3}{3}$. For large n, the savings in using L and U is substantial.

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

FWD (Forward substitution)

Inputs: level 2: an n×n lower triangular matrix L
level 1: an n-vector b

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

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

EXAMPLE. To solve

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

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

```
A= [[ 2  3 -1  2]
    [-4 -6  2  1]
    [ 2  4 -4  1]
    [ 4  8  2  7]] .
```

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

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

L= [[ 1   0   0   0]
    [-1   1   0   0]
    [-.5 .5   1   0]
    [-.5  0   0   1]] ,

U= [[-4 -6  2   1 ]
    [ 0  2  4   8 ]
    [ 0  0 -5 -2.5]
    [ 0  0  0  2.5]] .
```

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 diagonal and the pivots appear on the diagonal of L. The method used to obtain this factorization is known as the *Crout* algorithm, 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 [L U] 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}.

Unlike our 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 non-singular linear system by applying an LU-factorization with a single keystroke. With an invertible matrix A or order n on stack level 1 and a matrix B having n rows on level 2, the command /, executed from the keyboard by pressing the [+] key, will effectively solve the linear system(s) Ax = B by the method cited earlier: 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.

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 was done to minimize 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.

Gauss-Jordan reduction, when applied to a non-zero matrix A, 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) the 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. Specifically, any variable (unknown) associated with a pivot is called a *pivot variable* while the other variables, if any, are called *free variables*. If the last non-zero row of [U| b'] looks like [0 0 . . . 0 1], the system has no solution. In any other case there is at least one solution: a unique solution if there are no free variables, and infinitely many when free variables are present. The pivot variables are usually expressed in terms of the free variables, and values for the free variables may be arbitrarily (*i.e.*, freely) chosen. Although impractical for large linear systems, Gauss-Jordan reduction is in popular use as a device to solve small systems. And it is easy to devise a calculator program for students to use 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. Students can use it in conjunction with the command RSWP to produce the RREF matrix. The program is written to accommodate both real and complex matrices.

PIVOT	(Gauss-Jordan Pivot)
Inputs:	level 3: a matrix
	level 2: an integer K
	level 1: an integer L
Effect:	converts the (K, L)-entry to 1 and then pivots on that entry to produce zeros above and below the pivot.

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

EXAMPLE . Solve the linear system

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

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

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

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

Although the reduced row-echelon form of a matrix is not in use at the professional level to solve linear systems, it can be an effective pedagogical tool in helping students understand the role of pivot point variables versus free variables and such vector space concepts as spanning, independence, bases and eigenspaces. The HP-48G and 48GX calculators provide access to this form by means of the RREF command, located on the MTH MATR FACTR menu. With a matrix A 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. After some initial experiences in producing this form with program PIVOT and the command RSWP, students should be encouraged to call upon the RREF command thereafter. The underlying code uses partial pivoting throughout.

ACTIVITIES II

1. Consider the linear system

$$\begin{aligned} -x_1 + x_2 + x_3 - 2x_4 &= -1 \\ 3x_1 - 3x_2 - 6x_3 + 12x_4 &= 3 \\ -2x_1 + 4x_2 + x_3 + 2x_4 &= 3 \\ -5x_1 + x_2 + 17x_3 - 24x_4 &= -20 \end{aligned}$$

   (a) Use Gaussian elimination with partial pivoting and back substitution to solve the system.

   (b) Solve the system by finding and applying an LU-factorization with partial pivoting; do this in three ways:

      (i) Use program L.U to construct an LU-factorization;

      (ii) Use the built-in command LU to obtain an LU-factorization;

(iii) Use the / command.

(c) Use Gauss-Jordan reduction to solve the system.

2. Find bases for the row space, column space and null space of the following matrix

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ -1 & -2 & -3 & -6 & -11 \\ 2 & 4 & 8 & 5 & 11 \\ -1 & -2 & -2 & -6 & -6 \end{bmatrix}.$$

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

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

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

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

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

(c) Find the change-of-basis matrix P from the B-basis to the B' basis for W. Then use P to express w in terms of the B'-basis.

4. ORTHOGONALITY CONCEPTS

Orthogonality concepts lie at the very heart of modern linear algebra. Orthogonal vectors, projections, bases, subspaces and matrices all combine to produce not only a rich and elegant theory, but also powerful numerical techniques that are widely used in the more serious applications of matrix-oriented linear algebra. We shall begin with projections and the Gram-Schmidt process.

The following program PROJ may be used to calculate the projection vector $P_y x$ of vector x onto vector y.

PROJ	(Projection vector)
Inputs:	level 2: a vector x
	level 1: a vector y
Effect:	Returns the projection vector $P_y x$ to level 1

« → X Y « X Y DOT Y * Y Y DOT / » »

EXAMPLE. For x = [5 15 5]^T and y = [3 4 5]^T find $P_y x$ and verify that $x-P_y x$ is orthogonal to y.

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

level 4: y
3: x
2: x
1: y

Press PROJ to see $P_y x$, then − to see $x-P_y x$, then DOT (on the MTH VECTR menu) to verify that $y \bullet (x-P_y x) = 0$.

The Gram-Schmidt Process

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

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

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

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

Let's look at several steps:

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

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

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

.
.
.

etc.

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

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

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

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

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

.
.
.

. . . and so on.

EXAMPLE. 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 may not recognize the entries, but the Q1, Q2 and Q3 you constructed 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 that order), pressing 3 COL→ to create a matrix $Q = \begin{bmatrix} | & | & | \\ Q_1 & Q_2 & Q_3 \\ | & | & | \end{bmatrix}$, then ENTER TRN SWAP * to see Q^TQ. Clean up round-off error with 11 RND and you should see I_3.

The Gram-Schmidt process, as presented above, is numerically unstable in floating point arithmetic. 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 obtain QR-factorizations.

QR-Factorizations

Just as Guassian elimination 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 invertible upper (or right) triangular.

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

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

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

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

.
.
.

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

Let A have $x_1, x_2, \ldots, x_n$ as its columns, left-to-right, and let Q have $q_1, q_2, \ldots, q_n$ as its columns, also left-to-right. Let R be the right triangular matrix

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

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

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

or

$$\mathbf{A = QR}\ .$$

Moreover, since $q_1, q_2, \ldots, q_n$ are orthonormal, we know that $Q^TQ = I$ and the i^{th} coefficient in

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

is $r_{ij} = q_i \bullet x_j$. Also, $r_{11} \neq 0$ because $r_{11} = \|x_1\|$, $r_{22} \neq 0$ because $r_{22} = \|x_2 - (x_2 \bullet q_1)q_1\|$, etc., and so R is invertible. Finally, from $Q^TQ = I$ and $A = QR$ we have $Q^TA = Q^TQ\,R = R$.

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

To obtain A = QR on the HP-48G or 48GX:

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

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

(3) Construct and store matrix Q:

Q1 Q2 ... QN N [COL→] [,] Q [STO]

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

You can verify that A = QR as follows:

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

Continuing with the vectors from our last example, construct A as

```
    [[ 2  0  2 ]
A=  [ 1  1  0 ]  .
    [ 0  1  2 ]]
```

After constructing Q you should see

```
    [[ .894427191     -.298142397      .333333333334 ]
Q=  [ .4472135955     .596284794      -.666666666665 ]  .
    [      0          .7453559925      .666666666665 ]]
```

After constructing R you should see

```
    [[ 2.2360679775     .4472135955     1.788854382 ]
R=  [       0           1.3416407865     .894427191 ]  .
    [ .000000000003         0                2      ]]
```

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, typically, the only type of QR-factorization encountered by students in introductory linear algebra. But QR-factorizations of more general matrices, obtained by more sophisticated numerical methods, are in widespread use by the professional matrix codes used in science and engineering. The HP-48G and 48GX

calculators incorporate such professional level code for a variety of applications, including producing least squares solutions to under-determined and over-determined 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×n matrix on level 1: AP = QR. Here, Q is an m×m orthogonal matrix, R is an m×n upper trapezoidal matrix, and P is an n×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×m permutation matrix, L is an m×n lower trapezoidal matrix, and Q is an n×n orthogonal matrix.

For example, with the matrix

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

on level 1, pressing the [QR] key will return the permutation matrix

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

to level 1, the upper trapezoidal matrix

```
     [[ 10.3440804328   -.676715542332   -.483368244523   -5.80041893427 ]
R =  [        0          6.96721293451   -3.06106659926    4.46014305107 ]
     [        0                0          -.27196004146    -.67990010365 ]]
```

to level 2, and the orthogonal matrix

```
      [ [ -.483368244523      -4.69488742217E-2     .874157276122 ]
Q =     [  9.66736489046E-2     -.995316133501          -8.E-16     ]
        [ -8.70062840141      -8.45079735991E-2     -.485642931179 ] ]
```

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

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 $\|Ax - b\|_2$ is as small as possible. Here, $\| \|_2$ denotes the 2-norm of the included vector and such an x is called a least squares solution. It can be shown that the least squares solutions to Ax = b are precisely the solutions to the associated system $A^TAx = A^Tb$, the so-called normal equations. Moreover, if A has full column rank than A^TA is invertible and there is a unique least squares solution. 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, located 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.

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

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

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

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

onto level 1 and press [LSQ]. The desired solution is

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

Fitting Curves to Data

Suppose we have n data points (x_1, y_1), (x_2, y_2), ... , (x_n, y_n) where all the x_j's are distinct. Consider the problem of finding a polynomial $P(t) = c_0 + c_1 t + \ldots + c_m t^m$ of degree $\leq m$ that passes through these data points, *i.e. fits* the data. We shall require $n \geq m+1$. Thus our requirements are $P(x_i) = y_i$ for i = 1, ..., n or

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

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

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

$$(*)\quad A=\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ & & \vdots & & \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix},\ c=\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_m \end{bmatrix} \text{ and } y=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

Since there are at least as many equations as unknowns, the system will, in general, be over determined and we naturally seek a least squares solution. However, A has independent columns, for if Ac = 0 had a non-0 solution, this would mean that there exists a non-0 polynomial P(t) of degree ≤ m having m+1 roots. Since A has independent columns, there is a *unique least squares solution*, given by the unique solution to the normal equations

$$(A^TA)c = A^Ty.$$

It is tempting to obtain the least squares solution by calculating $x = (A^TA)^{-1}A^Ty$ or by applying Gaussian elimination via the / command. But the coefficient matrix A^TA is likely to be ill-conditioned, so that solutions to $(A^TA)x = A^Ty$ are somewhat sensitive to perturbations caused by round-off errors. This is especially the case with large data sets where the x-values are equally spaced. Thus, good computational practice suggests that the above two approaches to solving the normal equations be abandoned in favor of the more sophisticated one provided by the LSQ command. We shall return to this conditioning question in the Activities.

The following program, P.FIT, creates the coefficient matrix A.

P.FIT (Polynomial Fit Matrix)

Input: level 2: an integer M
level 1: a list $\{x_1, x_2, ..., x_N\}$

Effect: Returns the matrix

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^M \\ 1 & x_2 & x_2^2 & \cdots & x_2^M \\ & & \vdots & & \\ 1 & x_N & x_N^2 & \cdots & x_N^M \end{bmatrix}$$

```
« DUP SIZE → M lst N « 1 N FOR J lst J GET → x « 1 1 M
FOR I x I ^ NEXT » NEXT N M 1 + 2 → LIST → ARRY » »
```

EXAMPLE. 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

```
     [[1  1   1    1 ]
      [1  2   4    8 ]
A =   [1  3   9   27 ]   .
      [1  4  16   64 ]
      [1  5  25  125]]
```

Now put y = [.6 1.2 2 2.8 4.1] on the stack and press [SWAP], then [LSQ] to see c = [-.16 .85 -.125 .025]. Thus the least squares cubic polynomial fit is $P_3(t) = -.16 + .85t - .125t^2 + .025t^3$.

ACTIVITIES III

1. (a) Generate a random 5×4 matrix over Z_{10} whose columns will be called x_1, x_2, x_3 and x_4.

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

   (c) Find the projection vector $P_W x_3$ of x_3 onto W: $P_w x_3 = \text{proj}_{q_1} x_3 + \text{proj}_{q_2} x_3 + \text{proj}_{q_3} x_3$.

   (d) Verify that $x_3 - P_W x_3$ is orthogonal to W by checking that it is orthogonal to x_1, x_2 and x_4.

2. (a) Fill-in the following table of values for $f(x) = (x+2)^2 e^{-x}$ (round to 3 decimal places).

x	-2.2	-1	.5	1.5	3
$y = (x+2)^2 e^{-x}$					

   (b) Plot the 5 data points.

   (c) Find the least squares cubic polynomial $P_3(x)$ for this data; overlay your data plot with the graph of $P_3(x)$.

   (d) Find the least squares polynomial $P_4(x)$ of degree 4 for this data; overlay your data plot with the graph of $P_4(x)$.

3. Augment the data in our last example with a sixth data point, 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^TA is ill-conditioned so that using Gaussian elimination or matrix inversion to solve the normal equations may produce inaccurate results.

(b) Find the least squares cubic polynomial by using the LSQ command, then with the / command; note that the two solutions agree to 12 decimal places.

(c) Now find the least squares cubic polynomial by applying the RREF command to the augmented matrix, then by inverting the coefficient matrix. Compare the accuracy of the two solutions with those obtained above in (b).

5. EIGENVALUES AND EIGENVECTORS

Eigenvalue-eigenvector considerations are of paramount importance in many real applications of linear algebra to science and engineering, especially in those involving systems of linear differential equations. The HP-48G/GX calculators can help students develop conceptual understanding 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.

The Characteristic Polynomial

Given a square, n×n matrix A, any real or complex number λ for which there is a non-zero vector x such that $Ax = \lambda x$ is called an *eigenvalue* of A, and the vector x is an associated *eigenvector*. To find such pairs (λ, x) we consider the equation $Ax = \lambda x$, which is clearly equivalent to $(A - \lambda I)x = 0$. 1 Thus λ is an eigenvalue, and x an eigenvector, iff x is a non-zero solution to the homogeneous linear system with coefficient matrix $A - \lambda I$. Such a solution exists iff $A - \lambda I$ is singular, which happens precisely when $\det(A - \lambda I) = 0$. The left-hand side, $\det(A - \lambda I)$, is a polynomial of degree n in λ, often called the *characteristic polynomial* of matrix A. Some authors prefer to use $\det(\lambda I - A)$ instead, but the difference is minor since these two polynomials differ only by a factor of $(-1)^n$. What really counts is that the eigenvalues of A are the roots of either of these polynomials, and for any such root λ the associated eigenvectors are the non-zero solutions to the linear system $(A - \lambda I)x = 0$.

Although the above is rather elegant from a purely algebraic viewpoint, 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 modest sized, or highly-specialized matrices. And secondly, it is no easy task to determine the roots of a polynomial.

Given an n×n matrix A, the following calculator program, CHAR, calculates the coefficients of $\det(\lambda I - A) = \lambda^n + c_{n-1}\lambda^{n-1} + \ldots + c_1\lambda + c_o$, 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×n matrix A

Effect: returns a vector [1 c_{n-1} ... c_1 c_0] of coefficients of $\det(\lambda I - A) = \lambda^n + c_{n-1}\lambda^{n-1} + \ldots + c_1\lambda + c_0$

```
« 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. Enter
```
[[ 4 -8  2  5]
 [ 0  1 -6 -2]
 [-9  0  7  1]
 [ 7  3 -8  9]]
```
. Press CHAR to see the coefficients vector [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.

Eigenvalue Calculations

Although low order matrices having integer entries are not typical of the matrices encountered in scientific and engineering applications, they serve us well in the learning process. But even with such matrices, finding the eigenvalues by hand as the roots of the characteristic polynomial is a difficult, if not impossible, task unless the matrices are highly contrived. To avoid such contrivance, we may use the polynomial root-finding routine PROOT on the HP-48G/GX calculators. PROOT will find all roots of an

arbitrary real or complex polynomial $a_n x^n + a_{n-1}x^{n-1} + \ldots a_1 x + a_0$. It requires as input the vector [a_n a_{n-1} . . . a_1 a_0] of coefficients.

EXAMPLE. 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 $p(\lambda) = \lambda^4 - 8\lambda^3 + 22\lambda^2 - 40\lambda + 25$. Go to the SOLVE menu with ↰ SOLVE and open the POLY subdirectory. Press PROOT to return 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 * −. Use ↰ EDIT to view the last column. After cleaning up round off error with 11 RND, we 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 is thus reasonably effective in returning these coefficients. 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, say, 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 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. We use
$$\begin{bmatrix} -14 & -16 & -26 & -9 \\ 16 & 19 & 28 & 12 \\ -7 & -8 & -11 & -7 \\ 13 & 14 & 24 & 14 \end{bmatrix}$$
as our matrix A. 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. With A on level 2, the command EGVL (on the MTH MATR menu) returns the vector [4.46410161514 -2.46410161514 3 3] of eigenvalues accurate to 12-digits.

Diagonalization

Given an n×n matrix A, how many independent eigenvectors can A have? Certainly no more than n because eigenvectors lie in R^n, which has dimension n. And the case where A has n independent eigenvectors, say $x_1, x_2, \ldots, x_n$, is especially nice. For then $P^{-1}AP = D = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$ where λ_i is the eigenvalue associated with x_i and P is the matrix having $x_1, x_2, \ldots, x_n$ as its columns: $P = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix}$.

The equation $P^{-1}AP = D$ is equivalent to saying that A has n independent eigenvectors. This equation is a rearrangement of $AP = PD$ which, when read column-by-column, simply says $Ax_i = \lambda_i x_i$. The x_i's are independent because they are the columns of the invertible matrix P. We are thus led to focus on the case where the n×n matrix A has n independent eigenvectors as the desirable one, and in this event call A diagonalizable. Any n×n matrix A having fewer than n independent eigenvectors is called *defective*.

Of fundamental help in determining if 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 and is diagonalizable. But it is also possible for A to be diagonalizable even when it has fewer than n distinct eigenvalues. There are two keys to understanding how this may happen:

(1) For any eigenvalue λ of A, dim NS(A – λI), *i.e.*, the dimension of the eigenspace associated with λ, does not exceed the multiplicity of λ as a root of the characteristic polynomial;

(2) If $\lambda_1, \lambda_2, ..., \lambda_k$ are the *distinct* eigenvalues of A and $B_1, B_2, ..., B_k$ are bases for the associated eigenspaces then the union of these bases is an independent set of eigenvectors of A.

Think about the characteristic polynomial of A in factored form:

$$\det(\lambda I - A) = (\lambda - \lambda_1)^{m_1}(\lambda - \lambda_2)^{m_2} \ldots (\lambda - \lambda_k)^{m_k}$$

where $\lambda_1, ..., \lambda_k$ are the *distinct* eigenvalues and $m_1, ..., m_k$ are their respective multiplicities. Since det(λI – A) is a polynomial of degree n, we have $n = m_1 + m_2 + ... + m_k$. According to (1), we have dim NS(A – λ_jI) $\leq m_j$, for each j = 1, ..., k. Thus, in the case where equality holds for every j, the bases in (2) will contain exactly m_j vectors and their union will produce n independent eigenvectors for A. But in the case where we have dim NS(A – λ_jI) < m_j for even *one* j, the union of the bases in (2) will fail to produce n independent eigenvectors and A will be defective.

A is defective iff for some eigenvalue λ there are not enough independent eigenvectors associated with λ.

EXAMPLE. Program CHAR returns [1 6 9 0 0] for matrix

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

Thus the characteristic polynomial 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 RREF to A – 0I and A + 3I shows that NS(A – 0I) and NS(A + 3I) each have dimension 1, so A is (*doubly*) defective.

EXAMPLE. Consider $A = \begin{bmatrix} 0 & 3 & -2 & 0 & -4 \\ -4 & 5 & -4 & 0 & -4 \\ 4 & -3 & 6 & 0 & 4 \\ 4 & -3 & 4 & 3 & 5 \\ -6 & 3 & -6 & 0 & -2 \end{bmatrix}$. The command EGVL returns 1, 2, 2, 3, 4 as eigenvalues. 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. Thus A is diagonalizable. You can use the RREF command to see that a basis for the eigenspace associated with $\lambda = 2$ consists of the vectors [-1 0 1 0 0] and [0 $1.\overline{3}$ 0 -1 1], and that the eigenspaces associated with $\lambda = 1$, 3 and 4 have [1 1 -1 -1 1], [0 0 0 1 0] and [-1 0 0 1 1] as bases, respectively. Put these five basis vectors on the stack and press 5 COL→ to build the diagonalizing matrix

$$P = \begin{bmatrix} -1 & 0 & 1 & 0 & -1 \\ 0 & 1.\overline{3} & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & -1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Verify that $P^{-1}AP = D = \begin{bmatrix} 2 & & & \\ & 2 & & \\ & & 1 & \\ & & & 3 \\ & & & & 4 \end{bmatrix}$ (clean up round off error with 10 RND).

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 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 [3 2 1 4 2] as the vector of eigenvalues and you will notice that columns 1, 3, and 4 of the matrix P that is returned to level 2 are precisely the vectors we constructed in the example. If you extract columns 2 and 5 and assemble them as the columns in a new matrix, the RREF command will show them to be independent.

ACTIVITIES IV

1. Adding a multiple of a row to another row will not change the determinant of a square matrix A. Will this change the eigenvalues? The characteristic polynomial? Use your calculator to investigate these questions by experimenting with random 3×3 and 4×4 matrices over Z_{10}.

2. (a) Generate two random 3×3 matrices A and B over Z_{10} and calculate the characteristic polynomials of AB and BA. What do you observe?

   (b) Repeat (a) for random 4×4 and 5×5 matrices over Z_{10}.

   (c) Repeat (a) – (b) for random 3×3, 4×4 and 5×5 *complex* matrices over Z_{10}.

(d) Formulate a conjecture based upon your observations. Discuss your conjecture and its implications with your instructor.

3. Determine whether the following matrices A are diagonalizable or defective. For each one that is diagonalizable, find an invertible P and a diagonal D for which $P^{-1}AP = D$.

(a)
```
[[ 0  1 -2  0 ]
 [-2  3 -4  0 ]
 [ 0  0  1  0 ]
 [-1 -1  0 -1 ]]
```

(c)
```
[[ 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 ]]
```

(d)
```
[[ 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 ]]
```

REFERENCES

1. Carlson, David, et al., *The Linear Algebra Curriculum Study Group Recommendations for the First Course in Linear Algebra*, The College Mathematics Journal, Vol. 24, No. 1, January 1993, 41-46.

2. D.R. LaTorre, *Linear Algebra with the HP-48G/GX*, Saunders College Publishing, 1994, Philadelphia, PA.

HP-48G/GX Calculator Enhancement for Advanced Engineering Mathematics

Donald L. Kreider, Dartmouth College

Introduction.

Applications of the HP-48G/GX 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.

This chapter pursues 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 series 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 chapter illustrate the power of the HP-48G/GX 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.

Section 1. Solution of 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 3, 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.

Example 1.1.

Consider the following initial-value problem

$$y'' + x^2 y = 0, \quad y(0) = 1, y'(0) = 0. \tag{1.1}$$

It occurs in problems of bending beams and in optics. It does not have a closed form solution expressible in elementary functions.[1] 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 1.1. A Runge-Kutta method.

The program generates a numerical solution of the initial-value problem

$$\begin{aligned} \frac{dy}{dx} &= f_1(x, y, z) \\ \frac{dz}{dx} &= f_2(x, y, z) \\ y'(x_0) &= y_0 \\ z'(x_0) &= z_0 \end{aligned} \tag{1.2}$$

---

[1] Later in the section we will see that its solutions can be expressed in terms of Bessel functions.

on the interval $x_0 \le x \le xmax$ 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.

Inputs		Outputs
4:	«→ x y z 'f1(x,y,z)'»	
3:	«→ x y z 'f2(x,y,z)'»	Draws graph of solution
2:	{x0 y0 z0 h}	of (1.2)
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 $0 \le x \le 6$, 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:

$$\begin{aligned} \frac{dy}{dx} &= z \\ \frac{dz}{dx} &= -x^2 y \\ y(0) &= 1, \quad z(0) = 0, \end{aligned} \tag{1.3}$$

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.

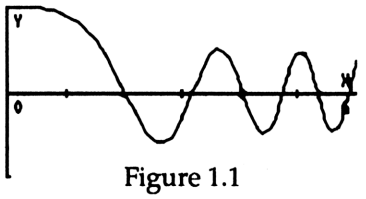


Figure 1.1

Graph of the solution of the initial-value problem (1.1), or (1.3), drawn by the HP-48G using program 1.1.

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 1.1 is implied by the *Sonin-Polya theorem* [2].

---

[2] 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.

The zeros of the solution shown in Figure 1.1 can be found approximately using the (X,Y) 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.

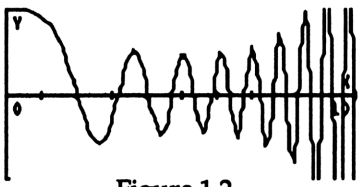


Figure 1.2

Graph of the solution of (1.1), drawn on the interval $0 \le x \le 10$ by the same program.

Exercise 1.1.

Enter Program 1.1 into your HP-48G and reproduce the graphs shown above. Then generate the graph of the solution of (1.1) on the interval $0 \le x \le 15$. How much confidence do you now have in the zeros estimated using the HP-48G's (X,Y) 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 1.2.

To regain your confidence slightly, apply Program 1.1 to graphing on the interval $0 \le x \le 15$ the solution of the initial-value problem

$$y'' + y = 0, \quad y(0) = 1, y'(0) = 0,$$

again using the value $h = 0.1$. How accurate is the fifth zero $2\pi / 9$? (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.

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

$$y(x) = \sum_{k=0}^{\infty} \frac{y^{(k)}(x_0)}{k!}(x - x_0)^k = \sum_{k=0}^{\infty} a_k (x - x_0)^k. \tag{1.4}$$

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 1.2.

Find a Taylor series solution

$$y = \sum_{k=0}^{\infty} \frac{y^{(k)}(0)}{k!}(x - x_0)^k \tag{1.5}$$

that satisfies the initial-value problem $y'' + x^2 y = 0, \quad y(0) = 1, y'(0) = 0,$ of Example 1.1. 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

$$\begin{aligned} &y''' + x^2 y' + 2xy = 0: && y'''(0) = 0, \\ &y^{(4)} + x^2 y'' + 4xy' + 2y = 0: && y^{(4)}(0) = -2, \\ &\quad \cdots \end{aligned} \tag{1.6}$$

and in general

$$\begin{aligned} y^{(n+2)} + x^2 y^{(n)} + 2nxy^{(n-1)} + n(n-1)y^{(n-2)} = 0: \\ y^{(n)}(0) = -(n-2)(n-3)y^{(n-4)}(0). \end{aligned} \tag{1.7}$$

From equations (1.6) and (1.7) we then obtain

$$\begin{aligned} 0 = y''(0) = y^{(6)}(0) = y^{(10)}(0) = \ \ldots, \\ 0 = y'''(0) = y^{(7)}(0) = y^{(11)}(0) = \ \ldots, \end{aligned}$$

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

$$y^{(4k)}(0) = -(4k-2)(4k-3)y^{(4k-4)}(0), \quad k = 1,\ 2,\ 3,\ \ldots,.$$

and we then quickly calculate as many of the non-zero derivatives as we wish. The desired solution is then

$$y_0(x) = 1 + \sum_{k=1}^{\infty} \frac{y^{(4k)}(0)}{(4k)!} x^{4k}, \tag{1.8}$$

and the function $y_0(x)$ can be computed by a simple program on the HP-48G.

Program 1.2. Series solution of an initial-value problem.

The program computes values of the solution $y_0(x)$ (Equation (1.8)).

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 1.3.

Enter Program 1.2 into your HP-48G and experiment with computing various values of $y_0(x)$. Plot the function on the interval $0 \le x \le 5$. 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 1..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 (1.7) determines all the derivatives, hence only the initial conditions need to be changed in Program 1.2. (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 $0 \le x \le 5$. Find its smallest zeros.

One source of inaccuracy in Program 1.2 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 (1.4) satisfy the recurrence relation

$$a_k = -\frac{a_{k-4}}{k(k-1)}. \tag{1.9}$$

This leads to an alternative program (below) for computing the solutions $y_0(x)$ and $y_1(x)$.

Program 1.3. 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 $y''+x^2y = 0$.

Inputs		Outputs	
2:	n		
1:	x	1:	$y_n(x)$

```
« → 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

(continued)

```
        (continued)
    « DO                                          • Add terms until
      'k+4' EVAL 'k' STO                            they are small
      T k k 1 - * / NEG x 4 ^ * 'T' STO
      T S + 'S' STO
     UNTIL T ABS 1E-12 <
     END
     S                                            • Put the final sum on
    »                                               the stack
  »
»
```

Example 1.3.

Store Program 1.3 in the variable Y. Then use it to graph the functions $y_0(x)$ and $y_1(x)$ on the intervals $0 \le x \le 5$ (Fig 1.3) and $0 \le x \le 8$ (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.

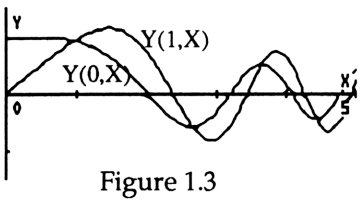


Figure 1.3

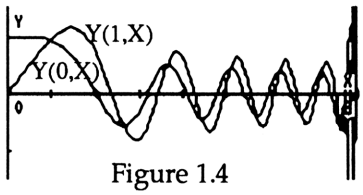


Figure 1.4

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 1.5.

Enter Program 1.3 into your HP-48G and verify the results in Figures 1.3 and 1.4. Modify the program so that each partial sum S is displayed as the program runs. (Insert S 1 DISP 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 1.4? 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

$$a_n(x)y^{(n)} + a_{n-1}(x)y^{(n-1)} + \ldots + a_1(x)y' + a_0(x)y = h(x) \qquad (1.10)$$

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 (1.10), 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 1.2 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 $-\infty \le x \le \infty$. 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 (1.9).

Example 1.4

We propose to solve the initial-value problem

$$y''+e^x y = 0, \quad y(0) = y'(0) = 1. \tag{1.11}$$

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 (1.11):

$$\sum_{k=0}^{\infty} k(k-1)a_k x^{k-2} + \sum_{k=0}^{\infty} \frac{x^k}{k!} \sum_{k=0}^{\infty} a_k x^k = 0.$$

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:

$$a_{k+2} = \frac{-1}{(k+2)(k+1)} \sum_{j=0}^{k} \frac{a_j}{(k-j)!}, \quad k \ge 0. \tag{1.12}$$

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:

$$a_2 = -\frac{a_0}{2} = -\frac{1}{2},$$

$$a_3 = -\frac{a_0 + a_1}{6} = -\frac{1}{3},$$

$$a_4 = -\frac{1}{12}\left(\frac{a_0}{2} + a_1 + a_2\right) = -\frac{1}{12},$$

$$\ldots$$

The initial terms of the desired solution are, therefore,

$$y(x) = 1 + x - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{12} + \ldots.$$

The following program provides a user-defined function for $y(x)$:

Program 1.4. Handling a many-term recurrence relation.

The program defines a user-defined function to compute the solution $y(x)$ of the initial-value problem (1.11).

Inputs		Outputs	
1:	x	1:	$y(x)$

The program stores the coefficients $a_0, a_1, \ldots, a_n$ 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 $0 \le x \le 3$ 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 $x = 1.58656$.

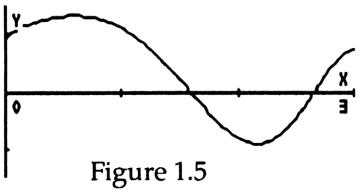


Figure 1.5

Graph of the solution of the initial-value problem (1.11), drawn by the HP-48G using program 1.4.

Exercise 1.6.

Enter Program 1.4 into your HP-48G, and verify the example above. Find the *second* zero of the solution lying in the interval $0 \le x \le 3$. 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 1.7.

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 1.8.

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.

Section 2. Bessel functions.

Bessel's equation of order p is

$$x^2\frac{d^2y}{dx^2}+x\frac{dy}{dx}+(x^2-p^2)y=0. \tag{2.1}$$

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 Section 1 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

$$(x-x_0)^2y''+(x-x_0)a_1(x)y'+a_2(x)y=h(x),$$

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

$$y=(x-x_0)^c\sum_{k=0}^{\infty}a_k(x-x_0)^k, \tag{2.2}$$

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 (2.2) 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 2.1.

Substituting (2.2) 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,

$$(c^2-p^2)a_0+\left(2p+1+c^2-p^2\right)a_1+\sum_{k=0}^{\infty}\left[(k+c)^2-p^2\right]a_k x^{k+c}+\sum_{k=2}^{\infty}a_{k-2}x^{k+c}=0.$$

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

$$\begin{aligned} &a_0 \quad \textit{arbitrary},\\ &a_1=0, \quad \textit{and} \\ &a_k=-\frac{a_{k-2}}{k(2p+k)}, \quad k\geq 2. \end{aligned} \tag{2.3}$$

We could stop at this point, using the recurrence relations (2.3), in the manner of the foregoing examples, to develop an infinite series for the solution. But further simplification is possible. From (2.3) we obtain

$$a_1=a_3=a_5=\ldots=0, \quad \text{and}$$

$$a_{2k}=(-1)^k\frac{a_0}{2\cdot 4\cdot 6\cdots(2k)\cdot(2p+2)(2p+4)\cdots(2p+2k)}=$$

$$=\frac{(-1)^k}{2^{2k}k!}\cdot\frac{a_0}{(p+1)(p+2)\cdots(p+k)}.$$

A final simplification will result from expressing the product in the denominator as $\Gamma(p+k+1)/\Gamma(p+1)$, where Γ 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 $x>0$.[3] Finally, choosing $a_0=2^{-p}/\Gamma(p+1)$, we are led to the solution

$$J_p(x)=\sum_{k=0}^{\infty}\frac{(-1)^k}{k!\,(k+p)!}\left(\frac{x}{2}\right)^{2k+p}, \tag{2.4}$$

usually called the *Bessel function of order p of the first kind* . In equation (2.4) 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 ($\Gamma(x+1)$).

Equation (2.4) 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 (2,2) 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

---

[3] 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.

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

$$y_2(x) = |x|^{c_2} \sum_{k=0}^{\infty} b_k x^k + C y_1(x) \ln|x|, \tag{2.5}$$

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 2.1. 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 2.2.

Plot the functions $J_0(x)$, $J_1(x)$ and $J_2(x)$ on the interval $0 \le x \le 10$. 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 2.1, 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.

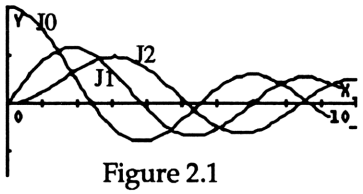


Figure 2.1

Graph of $J_0(x)$, $J_1(x)$, and $J_2(x)$, drawn by the HP-48G using program 2.1.

Exercise 2.1

Enter Program 2.1 into your HP-48G and verify the results of Example 2.2. 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 2.3.

Plot the functions $J_{\frac{1}{4}}(x)$ and $J_{-\frac{1}{4}}(x)$ on the interval $0.001 \le x \le 10$. (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 2.2.

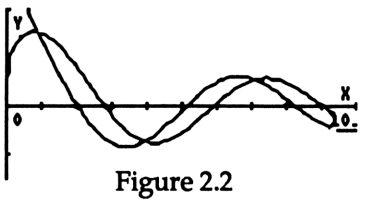


Figure 2.2

Graph of $J_{\frac{1}{4}}(x)$ and $J_{-\frac{1}{4}}(x)$, drawn by the HP-48G using program 2.1.

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 (2.5), containing a logarithmic term. Equation (2.6), 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].

$$Y_n(x) = \frac{2}{\pi}\left(\ln\frac{x}{2} + \gamma\right)J_n(x) - \frac{1}{\pi}\sum_{k=0}^{n-1}\frac{(n-k-1)!}{k!}\left(\frac{x}{2}\right)^{2k-n} - \frac{1}{\pi}\sum_{k=0}^{\infty}(-1)^k\frac{H_k + H_{n+k}}{k!(n+k)!}\left(\frac{x}{2}\right)^{2k+n} \tag{2.6}$$

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 (2.6) is the partial sum of the harmonic series $H_k = 1 + \frac{1}{2} + \frac{1}{3} + \ldots \frac{1}{k}$, $H_0 = 0$. The constant γ is Euler's constant $\gamma = \lim_{m\to\infty}(H_m - \ln m) \approx 0.577215664902$.

Program 2.2.

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

  « 'J(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 (2.6)
- Second term of (2.6) 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 (2.6) 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 2.4.

Plot the functions $J_0(x)$ and $Y_0(x)$ together on the interval $0 \le x \le 10$. Also plot $J_1(x)$ and $Y_1(x)$ together on $0 \le x \le 10$. 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)}.

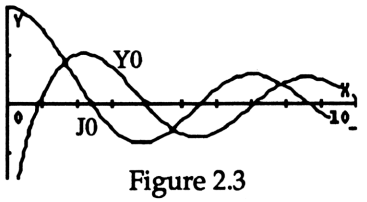


Figure 2.3

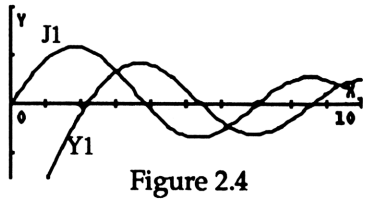


Figure 2.4

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.

Selected properties of Bessel functions.

1. Behavior for small values of x:

$$\begin{aligned} &J_p(x) \sim \frac{1}{2^p p!} x^p, && J_{-p}(x) \sim \frac{2^p}{(-p)!} x^{-p}, \\ &Y_n(x) \sim -\frac{2^n (n-1)!}{\pi} x^{-n} \quad (n \neq 0), && Y_0(x) \sim \frac{2}{\pi} \ln x. \end{aligned} \tag{2.7}$$

2. Behavior for large values of x:

$$J_p(x) \sim \sqrt{\frac{2}{\pi x}} \cos\left(x - \frac{\pi}{4} - \frac{p\pi}{4}\right), \quad Y_n(x) \sim \sqrt{\frac{2}{\pi x}} \sin\left(x - \frac{\pi}{4} - \frac{n\pi}{4}\right). \tag{2.8}$$

3. Differentiation formulas:

$$\begin{aligned} \frac{d}{dx} J_p(x) &= -J_{p+1}(x) + \frac{p}{x} J_p(x), \\ \frac{d}{dx} Y_n(x) &= -Y_{n+1}(x) + \frac{n}{x} Y_n(x). \end{aligned} \tag{2.9}$$

4. Recurrence relations:

$$J_{p-1}(x)+J_{p+1}(x)=\frac{2p}{x}J_p(x),$$
$$Y_{n-1}(x)+Y_{n+1}(x)=\frac{2n}{x}Y_n(x). \tag{2.10}$$

5. Integral representations:

$$J_p(x)=\frac{(\frac{1}{2}x)^p}{\sqrt{\pi}\Gamma(p+\frac{1}{2})}\int_0^\pi \cos(x\cos t)\sin^{2p}t\,dt \quad (p>-\tfrac{1}{2}),$$
$$Y_n(x)=\frac{1}{\pi}\int_0^\pi \sin(x\sin t-nt)\,dt-\frac{1}{\pi}\int_0^\infty [e^{nt}+e^{-nt}\cos n\pi]e^{-x\sinh t}\,dt. \tag{2.11}$$

The asymptotic behaviors for small and large values of x, given in (2.7) and (2.8), are useful in limit calculations and in interpreting the graphs of Bessel functions. The differentiation formulas in (2.9) are representative of a larger number of such formulas found in the references. The recurrence relations (2.10) 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 (2.11) 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 2.5.

Plot $J_{-\frac{1}{3}}(x)$ with its derivative, and find the coordinates of its first relative minimum point. The differentiation formula (2.9) 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.

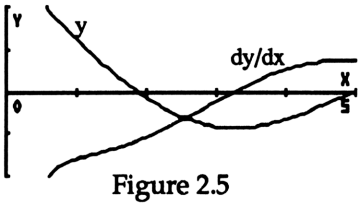


Figure 2.5

Graph of $J_{-\frac{1}{3}}(x)$ plotted with its derivative.

Example 2.6.

Examine the behavior of $J_0(x)$ on the larger interval $0 \le x \le 35$, and find its largest zero in this interval. The plot of the function in this interval, Figure 2.6 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 2.1, 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.

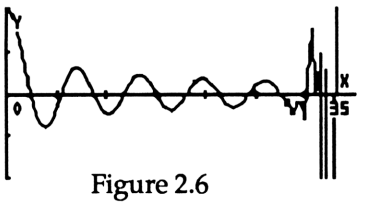


Figure 2.6

$J_0(x)$ plotted on $0 \le x \le 35$, showing the disastrous effect of *smearing* in the computation of the series defining $J_0(x)$.

In this situation consider using the integral formula (2.11) for computing $J_0(x)$. Programs 2.3 and 2.4, 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![4]

Programs 2.3 and 2.4.

The two programs below calculate values of $J_p(x)$ and $Y_n(x)$ using the integral representations (2.11).

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 2.3, below).

---

[4] As tabulated in [Abramowitz and Stegun].

Inputs for JI		Outputs	
2:	p	2:	
1:	x	1:	$J_p(x)$
Inputs for YI		**Outputs**	
2:	n	2:	
1:	x	1:	$Y_n(x)$

```
« → 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 (2.11)

```
« → n x
  « RAD 10 → U
    « '∫(0,π,SIN(x*SIN(t)-n*t),t)' → NUM
      '∫(0,U,(EXP(n*t)+EXP(-n*t)*COS(n*π))
       *EXP(-x*SINH(t)),t)' → NUM
      - π / →NUM
»»»»
```

- User-defined function for the second equation (2.11)

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 2.2.

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 (2.8) can suggest approximate values of the zeros.)

Exercise 2.3.

Define the integrand of the infinite integral in equation (2.11) 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 2.7.

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

$$\frac{d^2y}{dx^2} + \lambda^2 x^{p-2} y = 0$$

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 (2.7) 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

$$y_0(x) = \frac{\Gamma(\frac{3}{4})}{\sqrt{2}}\sqrt{x}J_{-\frac{1}{4}}(\tfrac{1}{2}x^2). \tag{2.12}$$

We tried earlier, but failed, to draw the graph of this function on the interval $0 \le x \le 10$. 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 $0 \le x \le 10$ the argument of $J_{-\frac{1}{4}}$ in (2.12) 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)!/√(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 (2.10) we can write

$$J_{-\frac{1}{4}}(x) = \frac{3}{2x}J_{\frac{3}{4}}(x) - J_{\frac{7}{4}}(x).$$

Thus we enter as the current equation the somewhat more complicated expression

'(-1/4)!*√(X/2)*((3/x^2)*JI(3/4,X^2/2)-JI(7/4,X^2/2))'

This time we succeed! Figure 2.7 shows the graph of the function $y_0(x)$ on the interval $6 \le x \le 10$ that was problematical in Example 1.3. 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.

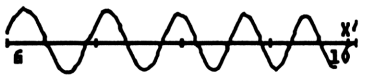


Figure 2.7

Graph of the solution of the initial-value problem (1.1) on $6 \le x \le 10$.

Section 3. Postscript on Numerical Solution of Differential Equations.

In Section 1 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 1.1 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 1.2, 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 (1.1) whose solution oscillates with ever increasing frequency as $x \to \infty$.

In general we wish to solve initial-value problems of the form

$$\frac{dY}{dt} = F(t, Y), \quad Y(0) = Y_0 \tag{3.1}$$

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 (3.1) 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 (3.1) 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 3 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 (3.1) directly and taking steps of constant size h (in the independent variable t). The second-order Runge-Kutta method of Section 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 (1.1), 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 (1.1). The zeros of the solution lying in the interval $0 \le x \le 10$ are also obtained for the purpose of comparing accuracy of the algorithms. The improved stability of the built-in RKF algorithm is evident.

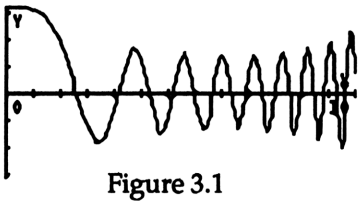


Figure 3.1

Graph of the solution of the initial-value problem (1.3) on $0 \le x \le 10$ drawn by the HP-48G using the second-order Runge-Kutta method.

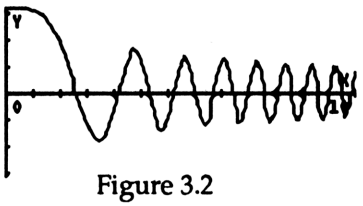


Figure 3.2

Graph of the solution of the initial-value problem (1.3) $0 \le x \le 10$ drawn by the HP-48G using the fourth-order Runge-Kutta method.

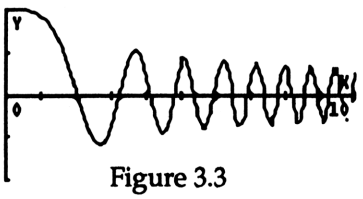


Figure 3.3

Graph of the solution of the initial-value problem (1.3) $0 \le x \le 10$ using the built-in RKF algorithm.

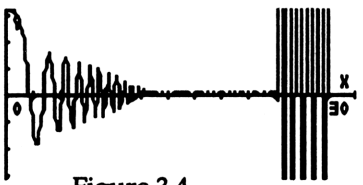


Figure 3.4

Graph of the solution of the initial-value problem (1.3) on the interval $0 \le x \le 30$ using the fourth-order Runge-Kutta method.

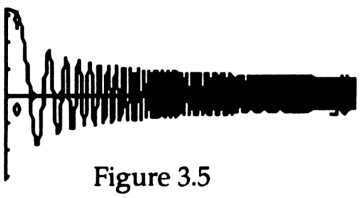


Figure 3.5

Graph of the solution of the initial-value problem (1.3) on the interval $0 \le x \le 30$ using the built-in RKF algorithm.

Figures 3.1, 3.2 and 3.4 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 (1.3) as well as for the initial conditions. Figures 3.3 and 3.5 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 (3.1). 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 (1.3), 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 value s of T and Y are provided as 0 and [1, 0], respectively.

Note how badly Figure 3.4 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 $0 \le x \le 10$. 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}	
2: accuracy desired	{T Y F}
1: final value of T	accuracy desired

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 (1.1)

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 3.1.

Enter the programs of the Appendix into your HP-48G (or HP-48S). Use them to plot the solution of these initial-value problems:

$$y''+y=0, \quad y(0)=0, y'(0)=1,$$

$$y''+\frac{1}{x}y'+y=0, \quad y(.00001)=1, y'(.00001)=0,$$

$$y''+e^x y=0, \quad y(0)=0, y'(0)=1,$$

$$3y'''-xy'+x^2y=e^x, \quad y(0)=y'(0)=0, y''(0)=\tfrac{1}{4}.$$

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 $J_0(x)$?

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

$$\alpha_1 y(a) + \alpha_2 y(b) + \alpha_3 y'(a) + \alpha_4 y'(b) = \gamma_1,$$
$$\beta_1 y(a) + \beta_2 y(b) + \beta_3 y'(a) + \beta_4 y'(b) = \gamma_2,$$

where α_i, β_i, and γ_i are constants. A simple example would be the following boundary-value problem on the interval $0 \le x \le L$:

$$\begin{aligned} y'' + \lambda y &= 0, \\ y(0) &= 0, \\ hy(L) + y'(L) &= 0, \end{aligned} \tag{3.2}$$

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 λ, 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 (3.2), 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:

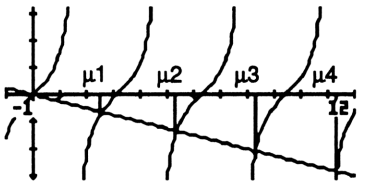
$$y = C_1 \cos\sqrt{\lambda}x + C_2 \sin\sqrt{\lambda}x,$$
$$C_1 \cdot 1 + C_2 \cdot 0 = 0,$$
$$h(C_1 \cos\sqrt{\lambda}L + C_2 \sin\sqrt{\lambda}L) + \sqrt{\lambda}(-C_1 \sin\sqrt{\lambda}L + C_2 \cos\sqrt{\lambda}L) = 0$$

(Note that when λ 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 λ 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 $\mu_1, \mu_2, \mu_3, \ldots$ of the curves as shown in the graph below (drawn by the HP-48G). The eigenvalues are then $\lambda_i = \mu_i^2 / L^2$, $i = 1, 2, 3, \ldots$. The first few values are easily computed, using the HP-48G's SOLVE function.



Figure

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.[5] Consider, for example, the boundary-value problem

$$y'' + \lambda xy = 0, \quad y(0) = y(2) = 0. \tag{3.3}$$

The shooting method seeks to find the first few positive eigenvalues by choosing an additional initial condition at $x = 0$ and determining λ by trial-and-error so that the

---

[5]Both methods are discussed in [Numerical Recipes].

(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 $y'' + \lambda x y = 0,\ y(0) = 0,\ y'(0) = 1$ for various values of λ, 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 3.7 to 3.10 show the solutions for $\lambda = 1, 2, 3, 4$, while Figures 3.11 to 3.14 show solutions for $\lambda = 8, 9, 10, 11$. Notice that when λ is 1 or 2 the solution *overshoots* the target point (2, 0), whereas when λ 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.

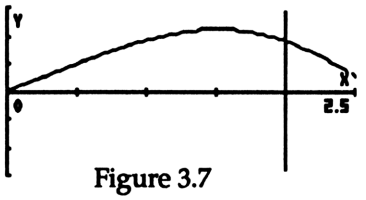


Figure 3.7

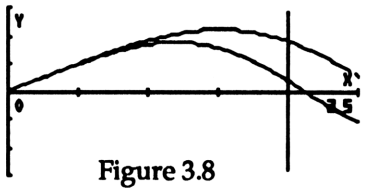


Figure 3.8

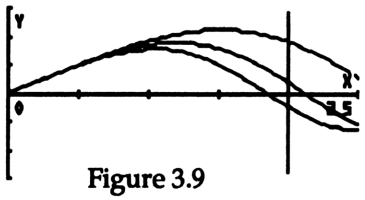


Figure 3.9

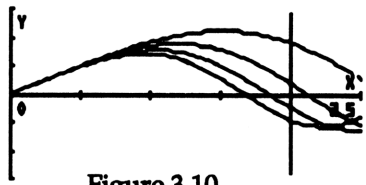


Figure 3.10

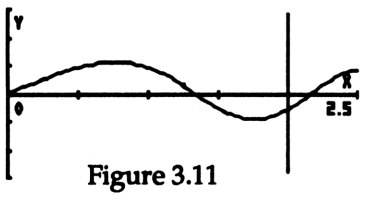


Figure 3.11

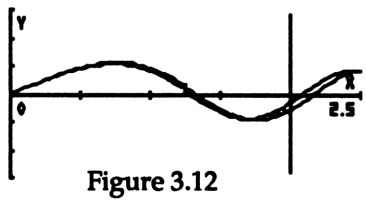


Figure 3.12

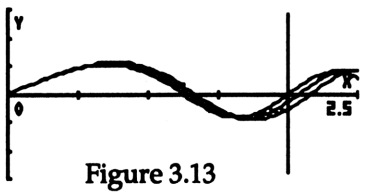


Figure 3.13

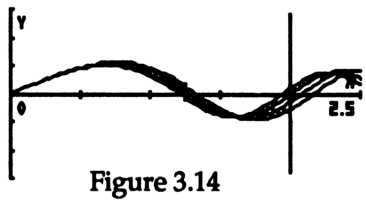


Figure 3.14

To find the eigenvalues more exactly we will define a user-defined function for the HP-48G that takes λ 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(λ)are the desired eigenvalues of the boundary-value problem, thus we can use the SOLVE application with EQ set to 'EV(λ1)' 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 3.2 we will see that these values are, indeed, correct to all digits shown.)

Exercise 3.2.

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 3.3.

Use Example 2.7 to show that the general solution of the differential equation is

$$y = \sqrt{x}\left(C_1 J_{\frac{1}{3}}(\tfrac{2}{3}\sqrt{\lambda}x^{\frac{3}{2}}) + C_2 J_{-\frac{1}{3}}(\tfrac{2}{3}\sqrt{\lambda}x^{\frac{3}{2}})\right).$$

From equations (2.7) 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

$$J_{\frac{1}{3}}(\tfrac{2}{3}\sqrt{\lambda}\,2^{\frac{3}{2}}) = 0.$$

Find the first several eigenvalues accurately by using your HP-48G and the program JS for computing $J_{\frac{1}{3}}(x)$. How do these values compare with those obtained by the shooting method?

The eigenvalues $\lambda_0, \lambda_1, \lambda_2, \ldots$ of the boundary-value problem (3.3) have been "found" in the sense that, at least in principle, they can be calculated numerically. The corresponding eigenfunctions $\varphi_0, \varphi_1, \varphi_2, \ldots$ have also been "found"—they are exactly the solutions arrived at, above, by the *shooting* method. Exercise 3.2 also showed, in fact,

that the eigenvalues are given by $\lambda_n = 9\mu_n^2/32, n = 1, 2, 3, \ldots$, where $\mu_1, \mu_2, \mu_3, \ldots$ are the positive roots of $J_{\frac{1}{3}}(x) = 0$; and the eigenfunctions are

$$\varphi_n(x) = \sqrt{x} J_{\frac{1}{3}}(\tfrac{2}{3}\sqrt{\lambda_n} x^{\frac{3}{2}}), \quad n = 1, 2, 3, \ldots \tag{3.4}$$

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.

Section 4. Fourier Series

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 $a \le x \le b$, with respect to the *weighting function* $p(x)$) if

$$\int_a^b p(x) f(x) g(x)\, dx = 0. \tag{4.1}$$

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 (4.1) holds.[6]

---

[6]In the various textbook references listed it is shown that admissible boundary conditions include the common forms $y(a) = 0$, $y'(a) = 0$, and $\alpha_1 y(a) + \alpha_2 y'(a) = 0$.

Our sample problem (3.3) is a Sturm-Liouville problem with $p(x) = 1$, thus the sequence (3.3) of its eigenfunctions is orthogonal. It is this property that enables us to expand quite general functions $f(x)$ in a series

$$f(x) = \sum_{n=1}^{\infty} a_n \varphi_n(x) = \sum_{n=1}^{\infty} a_n \sqrt{x} J_{\frac{1}{3}}\left(\tfrac{2}{3}\sqrt{\lambda_n} x^{\frac{3}{2}}\right), \tag{4.2}$$

where the coefficients are given by

$$a_n = \frac{\int_0^2 f(x)\varphi_n(x)\,dx}{\int_0^2 \varphi_n{}^2(x)\,dx} = \frac{\int_0^2 f(x)\sqrt{x} J_{\frac{1}{3}}\left(\tfrac{2}{3}\sqrt{\lambda_n} x^{\frac{3}{2}}\right)dx}{\int_0^2 x J_{\frac{1}{3}}{}^2\left(\tfrac{2}{3}\sqrt{\lambda_n} x^{\frac{3}{2}}\right)dx}.$$

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 4.1.

Use the values of $\lambda_1, \lambda_2, \lambda_3, \ldots$ that you determined in Exercise 3.2 to determine the first few terms of the series (4.2). Use the built-in integration function of your HP-48G and the user-defined function JS (or JI) given by Program 2.1.

Example 4.1.

The common (trigonometric) Fourier series

$$f(x)=\frac{a_0}{2}+\sum_{k=1}^{\infty}(a_k\cos kx+b_k\sin kx),$$

$$a_k=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\cos kx\,dx, \tag{4.3}$$

$$b_k=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\sin kx\,dx,$$

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 (4.3) in this case lead to

$$b_k=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\sin kx\,dx=\frac{2}{\pi}\int_{0}^{\pi}\sin kx\,dx$$

$$=\frac{2}{k\pi}(1-\cos k\pi)$$

$$=\begin{cases}\dfrac{4}{k\pi}, & k=1,3,5,\ldots,\\ 0, & k=2,4,6,\ldots.\end{cases}$$

Hence the Fourier series expansion of $f(x)$ is

$$f(x) = \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{\sin(2k-1)x}{2k-1}$$
$$= \frac{4}{\pi}\left[\sin x + \frac{\sin 3x}{3} + \frac{\sin 5x}{5} + \ldots\right].$$

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 nth 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.

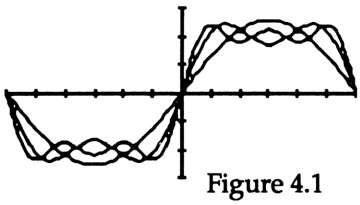


Figure 4.1

Figure 4.1 shows the first three partial sums, Figure 4.2 the 6th partial sum, and Figure 4.3 the 50th partial sum.

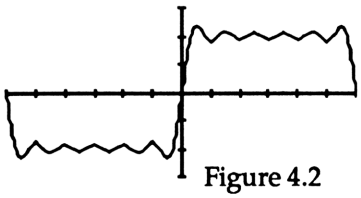


Figure 4.2

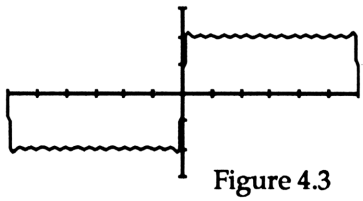


Figure 4.3

Exercise 4.2.

Save the graphs of the 1st, 2nd, 3rd, ..., nth 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.

Section 5. Legendre Polynomials, Gaussian Quadrature.

Important among sequences of orthogonal functions are a number of sequences of *orthogonal* polynomials. Legendre polynomials appear, for example, as the eigenfunctions $P_0(x), P_1(x), P_2(x), \ldots$ of the Sturm-Liouville problem

$$\begin{gathered}(1-x^2)y''-2xy'+n(n+1)y=0,\\ y(-1) \text{ and } y(1) \text{ finite.}\end{gathered} \tag{5.1}$$

The textbooks derive from the differential equation many of the important properties of these polynomials, important ones for our immediate purposes being

$$\begin{aligned}&P_n(x) \text{ is a polynomial of degree } n,\\ &P_n(x) \text{ has } n \text{ distinct zeros in the interval } -1<x<1,\\ &P_n(x) \text{ is an odd or even function according as } n \text{ is odd or even,}\\ &P_n(1)=1, \text{ and } P_n(-1)=(-1)^n,\\ &P_0(x)=1,\ P_1(x)=x, \text{ and } P_n(x)=\frac{2n-1}{n}P_{n-1}(x)-\frac{n-1}{n}P_{n-2}(x),\\ &\int_{-1}^{1}P_n(x)P_m(x)\,dx=0 \text{ if } n\neq m, \qquad =\frac{2}{2n+1} \text{ if } n=m.\end{aligned} \tag{5.2}$$

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

$$f(x)=\sum_{k=1}^{\infty}a_kP_k(x),$$

$$a_k=\frac{2k+1}{2}\int_{-1}^{1}f(x)P_k(x)\,dx.$$

To develop programs for computing Legendre polynomials efficiently, we turn to the recurrence relation (5.2). Starting with $P_0(x)=1$ and $P_1(x)=x$, we can compute successively as many of the polynomials as desired. Program 5.1 carries out this plan.

Program 5.1. Generation of Legendre Polynomials.

Program to generate the first n Legendre polynomials, store them in a subdirectory GRAF, and plot $P_n(x)$.

Inputs	Outputs
1: n	The subdirectory GRAF contains the first n Legendre polynomials.

The program assumes that the subdirectory GRAF 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 6.2 later in this chapter.

```
« GRAF 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
```

- Initialize the stack.
- Use the recurrence relation to generate and store n Legendre polynomials.

(continued)

(continued)

```
  DUP STEQ SETPP
  ERASE DRAX LABEL DRAW 7 FREEZE
  CLEAR UPDIR
»
```

- Store $P_n(x)$ in the current plotting equation and draw it.

```
« -1 1 XRNG -1 1 YRNG {X -1 1} INDEP »
```

- SETPP
  Program to set the plot parameters

Exercise 5.1.

Enter the first of the Programs 5.1 and save it in a variable LGN. Also enter the programs SETPP, EXCO and MULTI and create a subdirectory named GRAF. Experiment! In particular run LGN with input 16 to generate the first sixteen Legendre polynomials. The subdirectory now contains the polynomial expressions.

Example 5.1.

Using the expression for the sixteenth Legendre polynomial from the subdirectory GRAF, 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 5.2 shows a correct way.

Example 5.2.

Again we use the recurrence relation (5.1), this time avoiding the symbolic expressions for the Legendre polynomials. The program uses the recurrence relation for each value of x individually.

Program 5.2. Evaluation of Legendre Polynomials.

User-defined function to compute the value of $P_n(x)$.

Inputs		Outputs	
2:	n		
1:	x	1:	$P_n(x)$

```
« → 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.

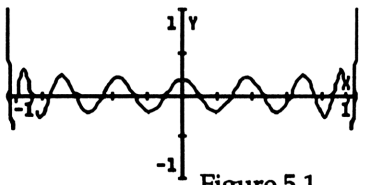


Figure 5.1

Graph of $P_{16}(x)$ drawn on the interval $-1 \le x \le 1$.

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 (5.1) 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 $x_0, x_1, \ldots, x_n$. The existence of such a polynomial is proved most easily by exhibiting it explicitly:

$$\begin{aligned} f(x) &= \sum_{k=0}^{n} f(x_k) L_k(x), \quad \text{where} \\ L_k(x) &= \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_1)(x_k-x_2)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} \end{aligned} \tag{5.3}$$

$L_k(x)$ 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:

$$\begin{aligned} L_k(x_k) &= 1, \quad \text{and} \\ \mathrm{L}_k(x_i) &= 0 \quad i \neq k. \end{aligned}$$

The approximation for $f(x)$ in (5.3) 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 (5.3) for $f(x)$. This yields

$$\begin{aligned} \int_a^b f(x)\,dx &= \sum_{k=0}^{n} f(x_k) \int_a^b L_k(x)\,dx \\ &= \sum_{k=0}^{n} w_k f(x_k). \end{aligned} \tag{5.4}$$

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 (5.4) provides a scheme for numerically integrating any continuous function $f(x)$. The student will recognize in (5.4) 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 $x_0, x_1, \ldots, x_n$ 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 (5.4), 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 5.3, below, accomplish the computation of the required weights. And Program 5.4 implements the Gaussian quadrature integration method.

Program 5.3. 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 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
  '∫(-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 5.4. 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)'		
2:	a	1:	$\int_a^b f(x)dx$
1:	b		
Inputs for PGIN (version 2)		**Outputs**	
3:	« program for f(x) »		
2:	a	1:	$\int_a^b f(x)dx$
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 (5.4)
- Get next zero
- Multiply by weight
- Finish coordinate transformation

```
PGIN:
 « → f a b
  « 0 « a b DUP2 SWAP - 2 / 4 ROLL
      * 3 ROLLD + 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 (5.4)
- Next zero α and its weight
- Next zero $-\alpha$ and its weight
- Finish coordinate transformation

Exercise 5.2.

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?

$$(a)\ \int_0^{\pi} \sin x\, dx,$$

$$(b)\ 4\sqrt{\frac{L}{g}}\int_0^{\frac{\pi}{2}} \frac{d\phi}{\sqrt{1-k^2\sin^2\phi}},\ \text{where}\ \begin{cases} L = 1\ \text{meter} \\ g = 9.80665\ {}^{m}/_{s} \\ k = \sin\dfrac{\pi}{4} \end{cases}$$

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 ≤ 33! Simpson's rule, by way of contrast, gives exact results only for polynomials of degree ≤ 3.

Exercise 5.3.

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.

Section 6. Some 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 6.1.

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 6.1.

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.)

Example 6.2.

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:

$$\begin{aligned}
&\mathbf{V} = \text{velocity vector} = \frac{d\gamma}{dt},\\
&v = \text{speed} = \|\mathbf{V}\|,\\
&\mathbf{U} = \text{unit tangent vector} = \frac{\mathbf{V}}{\|\mathbf{V}\|},\\
&\frac{d\mathbf{U}}{dt} = v\kappa\mathbf{N}, \quad \mathbf{N} = \text{unit normal vector}, \ \kappa = \text{curvature},\\
&\mathbf{B} = \mathbf{U}\times\mathbf{N} = \text{unit binormal vector},\\
&\frac{d\mathbf{B}}{dt} = -v\tau\mathbf{N}, \quad \tau = \text{torsion}.
\end{aligned} \tag{6.1}$$

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

```
« → t 'A1*COS(t)' »
« → t 'A1*SIN(t)' »
« → y 'B1*t' »
```
(6.2)

and the variables A1 and B1 contain constants. The essential feature of these programs in 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 (6.2) 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 6.1. 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 * /
  » »
```

- Numerical differentiation

```
V:
  « → t
    « 'P' t DER
  » »
```

- $\mathbf{V} = \frac{d\mathbf{P}}{dt}$

U:

```
« → t
  « 'V(t)' → NUM DUP ABS /
» »
```

- $\mathbf{U} = \dfrac{\mathbf{V}}{\|\mathbf{V}\|}$

N:

```
« → t
  « 'U' t DER DUP ABS /
» »
```

- $\mathbf{N} = \dfrac{\dfrac{d\mathbf{U}}{dt}}{\left\|\dfrac{d\mathbf{U}}{dt}\right\|}$

B:

```
« → t
  « 'U(t)' → NUM 'N(t)' →NUM CROSS
» »
```

- $\mathbf{B} = \mathbf{U} \times \mathbf{N}$

SPD:

```
« → t
  « 'V(t)' → NUM ABS
» »
```

- $v = \|\mathbf{V}\|$

CURV:

```
« → t
  « 'U' t DER ABS 'SPD(t)' → NUM /
» »
```

- $\kappa = \dfrac{1}{v}\dfrac{d\mathbf{U}}{dt}$

TORS:

```
« → t
  « 'B' t DER ABS NEG 'SPD(t)' → NUM /
» »
```

- $\tau = \dfrac{1}{v}\left\|\dfrac{d\mathbf{B}}{dt}\right\|$

Exercise 6.2.

Enter the programs 6.1 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 **U**, **N**, and **B** are indeed perpendicular. What accuracy is obtained by the programs?

Exercise 6.3.

Study the space curve $\gamma(t)=[t, 1+t^2, t^3-3t+1]$, $0 \le t \le 1$. 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 '∫(0,1,SPD(T),T)'.)

Example 6.3.

Line integrals of the form

$$\int_\gamma P\,dx+Q\,dy+R\,dz,$$

where γ 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

$$\int_\gamma P\,dx+Q\,dy+R\,dz=\int_\gamma F\cdot d\gamma$$

expresses the work done by the force field on the particle as it moves along γ.

As a specific example, let us compute the line integral $\int_\gamma F \cdot d\gamma$ where γ is the curve $\gamma(t) = [t, 1+t^2, t^3 - 3t + 1]$, $0 \le t \le 1$ of Exercise 6.3, 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))*àT(X(T))+Q(X(T),Y(T),Z(T))*àT(Y(T))
 +R(X(T),Y(T),Z(T))*àT(Z(T))'
```

Finally, we enter the following programs:

Program 6.2 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 6.4.

Enter the programs 6.2 and evaluate the line integral in Example 6.3 over various intervals. Also evaluate the following line integrals.

$$\int_\gamma \mathbf{F}\cdot d\gamma \quad \text{where} \begin{cases} \mathbf{F}(x,y,z)=[x,-yz,\ e^z] \text{ and} \\ \gamma(t)=[t^3,-t,\ t],\ 0\le t\le 1 \end{cases}$$

$$\int_\gamma xyz^2\,dz \quad \textit{where } \gamma(t)=[t^2,t^3,2t^2].$$

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 6.5.

Construct a version of Program 6.2 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.)

Example 6.4.

Green's Theorem in the plane states that, under suitable conditions of continuity and smoothness, an important relationship holds between a *line integral*

$$\oint_\gamma F \cdot d\gamma = \oint_\gamma P\,dx + Q\,dy$$

around a simple closed curve γ 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

'∫(A,B,∫(C(X),D(X),F(X,Y),Y),X)'

into the HP-48G and executing →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 6.3. 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)'		
5:	'C(X)'		
4:	'D(X)'		
3:	A		
2:	B	1:	$\int_a^b dx \int_{c(x)}^{d(x)} f(x,y)\,dy$
1:	<number of decimal places>		

```
DBL:« 6 DUPN FIX                                  • Copy arguments,
      'B' SWAP = DEFINE                             set output mode
      'A' SWAP = DEFINE                           • Define the functions
      'D(X)' SWAP = DEFINE                          from expressions
      'C(X)' SWAP = DEFINE                          on the stack
      'F(X,Y)' SWAP = DEFINE
    '∫(A,B,∫(C(X),D(X),F(X,Y),Y),X)' → NUM      • Evaluate the integral
      {F C D A B} PURGE                           • Clean up the mess
  »
```

Example 6.5.

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 lengthy.

Example 6.6.

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 6.7.

Use Green's Theorem to evaluate the line integral $\int_\gamma \mathbf{F} \cdot d\gamma$ around the curve γ 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

$$F(x,y) = \left[2y^3 e^x,\ 3x^2 - 4y^2\right].$$

We must evaluate the double integral

$$\iint_D \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right) dx\, dy = \iint_D \left(6x - 6y^2 e^x\right) dx\, dy.$$

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 6.2 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 $\gamma(t) = [-t, 4 - t^2],\ -2 \le t \le 2$. Program 6.2 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 5.4, of Section 5. We are motivated by the very long computing times for double integrals experienced when using the built-in

integration function of the HP-4G8. 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 6.4 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 6.4. 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)'		
4:	'C(X)'		
3:	'D(X)'		
2:	A	1:	$\int_a^b dx \int_{c(x)}^{d(x)} f(x,y)\,dy$
1:	B		

```
DINT:
  « 5 DUPN

    'B' SWAP = DEFINE
    'A' SWAP = DEFINE
    'D(X)' SWAP = DEFINE
    'C(X)' SWAP = DEFINE
    'F(X,Y)' SWAP = DEFINE

    « G »
    A →NUM B →NUM
    PGIN

    {F C D A B} PURGE
  »
```

- Copy the arguments
- Define the functions from expressions on the stack
- Do the outside integration.
- 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 6.8.

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 45 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 47 seconds we obtain the result 0.492423159996!

Exercise 6.6.

Find problems involving Green's Theorem in your textbook and use Programs 6.2 to 6.4 to evaluate the line integrals and double integrals. Compare Programs 6.3 and 6.4.

Exercise 6.7.

Find the volume of a sphere of radius 1 by writing the volume as the iterated integral

$$volume = \int_{-1}^{1} \int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} dy\,dx.$$

The infinite derivatives of $c(x)$ and $d(x)$ at the end points of the interval $-1 \le x \le 1$ are troublesome for most numerical integration methods. How well does the HP-48G do? How well does Gaussian quadrature do?

Some Textbook References

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.

Appendix. 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:

$$\frac{dy_1}{dt} = f_1(t, y_1, y_2, \ldots, y_n)$$
$$\frac{dy_2}{dt} = f_2(t, y_1, y_2, \ldots, y_n)$$
$$\ldots$$
$$\frac{dy_n}{dt} = f_n(t, y_1, y_2, \ldots, y_n)$$
$$y_1(t_0) = y_1', \; y_2(t_0) = y_2', \; \ldots, \; y_n(t_0) = y_n'$$

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,...,Yn] giving the current values of the dependent variables, a vector [Y1',Y2',...,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,...,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 $f_1, f_2, \ldots, f_n$ in the form of algebraic expressions (SETUP converts them to program form), the initial conditions in the form of a list $\{t_0\ y_0\ y_1 \ldots y_n\}$ 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.

```
REVW:
  « 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 â
         THEN INTRP RTLS SWAP + 'RTLS' STO
         END
         4 ROLL 4 ROLL DROP2
       UNTIL
         OVER PPAR 2 GET RE >
       END
    » »
```

- •• CHANGE ME to STP4 or STP4A
- Main program loop. Generate points until the graph is finished.

```
STP2:
  « DUP2 DYDX H RK2 »
```

- Take one step of RK2

```
STP4:
  « DUP2 DYDX H RK4 »
```

- Take one step of RK4

```
STP4A:
  « DUP2 DYDX RK4A »
```

- Take one step of RK4A

• RK2 step

```
RK2:
  « → x Y DY h
    « DY h * 2 / Y +
      x h 2 / + SWAP DYDX
      h * Y + x h + SWAP
    »
  »
```

• RK4 step

```
RK4:
  « 0 0 → x Y DY h h2 x2
    « x h 2 / + 'x2' STO
      DY h * DUP 2 / Y +
      x2 SWAP DYDX h * DUP 2 / Y +
      x2 SWAP DYDX h * DUP Y +
      x h + SWAP DYDX h * SWAP
      2 * + SWAP 2 * + + 6 / Y +
      x h + SWAP
    »
  »
```

```
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 →ARRY '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 â
      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 of 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 →ARRY
  »
```

- 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

```
REVW:
  « STK OBJ→ DROP
```

- Recall inputs to the stack for editing

HP48G/GX Calculator Enhancement
for
Probability and Statistics

Iris Brann Fetta

The subject of *statistics* deals with collecting, organizing, analyzing, displaying and interpreting information and the formulation of conclusions concerning the source of that information. Open today's newspaper, watch the television, listen to any sports report or look at any magazine. The chances are that you will see a graph or table presenting you with data or a conclusion that has been made as a result of the collection of data. Important decisions are made each day using the results of statistical surveys. Since any decision that is made should have an associated measure of the reliability of that decision, an understanding of the theory of *probability* is therefore necessary.

This chapter discusses the use of the HP48G's built-in statistics applications and also provides programs designed to enhance the capabilities of the calculator to explore many of the topics in an introductory course in probability and statistics. It is not the aim of this chapter to teach these topics. Likewise, the programs are not intended to give quick solutions, but are designed to help you think about the mathematics in a way that should provide insight into the underlying concepts. It is hoped that the material will stimulate you to explore and experiment beyond the specific applications presented here. We suggest that as you read through the chapter, you use your HP48G to follow the discussion and explorations.

The HP48G series (48G/48GX) offers a choice of two methods of accessing calculator applications: *screen interface* or *stack interface*. The screen interface, available for use when the green right-shift key [↱] is pressed before an application, provides

access through dialog boxes on the screen. The stack interface uses the standard HP48 softkeys and the stack to easily approach all commands related to a particular topic when the application key is preceded by the purple left-shift [↰] key. When applicable, instructions using both interfaces will be given through-out this chapter. Screen interface methods will be marked with the symbol [↱G] and stack interface methods will be denoted by the symbol [↰P]. We suggest you explore both operational methods and choose the one you prefer.

It will be helpful to create a statistics directory to hold the programs and data sets given here. Although this directory can be given any name, we suggest ISTAT. Before creating your directory, press [HOME] and enter ['] ISTAT on the stack.

- Note that whenever you are typing alphabetic characters, you will find it convenient to hold down the [α] key with one hand and continue holding it while you key in the letters with your other hand. When you finish typing, release the [α] key.

[↱G] Press [↱] [MEMORY] [NEW] and [▼]. Enter the directory name, press [OK] and [√CHK] [OK]. Press [ENTER] and you should see the name of the new directory in your variables ([VAR]) menu.

(If you are exploring both the screen and stack interface methods, you need to call the directory created by your second method a different name, say MSTAT.)

[↰P] Press [↰] [MEMORY] [DIR]. Enter the directory name and press [CRDIR]. Press [VAR] to see the name of the new directory in your variables menu.

(If you have created two directories ISTAT and MSTAT, delete MSTAT with the keystrokes ['] [MSTAT] [↰] [MEMORY] [DIR] [PGDIR].)

NUMERICAL DESCRIPTIVE MEASURES

When you describe a set of data with numbers that summarize its major features, you are using *numerical descriptive measures.* The statistics application of the HP48G/GX contains commands for calculating sample and population numerical descriptive measures. However, before any of these commands can be used, data must be entered into the calculator.

ENTERING AND EDITING ONE-VARIABLE DATA

One-variable data can be entered in the HP48G/GX as an array or as a list. Lists are more flexible than arrays, but the built-in statistics applications always use for calculations the data stored in a specific array (matrix) called ΣDAT. The statistical data matrix you create will reside in the active subdirectory of the VAR menu. Because ΣDAT is a variable, different ΣDAT matrices may exist in various directories in user memory. Any changes made to ΣDAT in the statistics application will update the ΣDAT matrix in the current directory but will not affect ΣDAT matrices stored in other directories. If you have changed directories, you should always recall ΣDAT to the stack by pressing [ΣDAT] to be sure you are working with the correct data. If you wish to use data other than the current statistical matrix, you can choose another matrix by entering new data, editing the current data or selecting another matrix. If you select a matrix with a name other than ΣDAT when using the stack interface, it must be designated the current statistical matrix by renaming it ΣDAT.

Let's explore the various methods of data entry. First, press [VAR] [ISTAT] to enter your statistics directory. Using either of the methods on the previous page, create a subdirectory called DATA to hold your data sets. Press [DATA] to enter this subdirectory.

The following table shows the lengths of term (in years) for the fifteen previous Chief Justices of the Supreme Court of the United States [4]:

Name	Appointed From	Term	Length of Term
John Jay	New York	1789-1795	5
John Rutledge	South Carolina	1795	0
Oliver Ellsworth	Connecticut	1796-1800	4
John Marshall	Virginia	1801-1835	34
Roger B. Taney	Maryland	1836-1864	28
Salmon P. Chase	Ohio	1864-1873	8
Morrison R. Waite	Ohio	1874-1888	14
Melville W. Fuller	Illinois	1888-1910	21
Edward D. White	Louisiana	1910-1921	10
William H. Taft	Connecticut	1921-1930	8
Charles E. Hughes	New York	1930-1941	11
Harlan F. Stone	New York	1941-1946	4
Fredrick M. Vinson	Kentucky	1946-1953	7
Earl Warren	California	1953-1969	15
Warren E. Burger	Virginia	1969-1986	17
William H. Rehnquist	Arizona	1986-	

Let's enter the 15 lengths of term for the former Chief Justices in the form of a list.

TO ENTER ONE-VARIABLE STATISTICAL DATA IN THE HP48G/GX AS A LIST:

- Press [↰] [{}] to begin the list.
- Key in the first data value and press [SPC].
- Key in the next data value, press [SPC], and continue in this manner until all values are entered.
- Press [ENTER] to place the list on the stack. Name and store this data list with the keystrokes ['] CJST [STO].

Lists of the same length can be added, subtracted, multiplied, divided, and operated on with functions. Additional information concerning lists can be found in your HP48G Series *User's Manual*. One list manipulation you will find beneficial is *sorting*. You may find this procedure helpful if you need to find the mode for a set of data or construct a stem-and-leaf plot. To sort the elements of a list in ascending order, place the list on level 1 of the stack by pressing the softkey corresponding to its name, and

press MTH LIST SORT. Sort the CJST list to obtain { 0 4 4 5 7 8 8 10 11 14 15 17 21 28 34 }. Sorting is a temporary action; the sorted list is not saved in your DATA directory unless you store it to the same name or to another name.

The table below gives data on serious football injuries (neck injuries) and deaths directly attributable to football (neck or head injury during game or practice).

Year	Serious Football Injuries	Deaths Directly Attributable to Football
1981	6	5
1982	7	7
1983	11	4
1984	5	4
1985	6	4
1986	3	10
1987	9	4
1988	10	7
1989	12	4
1990	11	0
1991	1	3

USA Weekend magazine, October 23-25, 1992

Enter the first column data (the 11 serious injury values) directly into ΣDAT using the screen interface by following the instructions below.

TO ENTER ONE-VARIABLE STATISTICAL DATA USING SCREEN INTERFACE:

- Press ↱ STAT and OK to choose Single-variable. If there is data already in ΣDAT (i.e., the highlighted space to the right of ΣDAT: is *not* empty), press DEL OK.
- Press EDIT to enter the MatrixWriter application. (You may first need to press NXT to find the menu page containing EDIT.)

- Key in the first data value, press [ENTER], and then press [▼] to move to the second row.
- Key in the remaining data values, pressing [ENTER] after each. (There is no need to press [▼] after the first row is entered.)
- Press [ENTER] to temporarily store the data into the ΣDAT matrix.
- Press [OK] to make the entered data the current ΣDAT matrix.

For future reference and use, let's save these data to the name INJ.

[G] **TO STORE ΣDAT TO A DIFFERENT NAME USING THE SCREEN INTERFACE:**

- Press [↱] [STAT] [OK]. You should see the current ΣDAT matrix partially displayed in the highlighted box.
- Press [NXT] [CALC] to place ΣDAT on the stack.
- Enter ['] followed by your choice of name for the data and press [STO].
- Press [OK] to return to the current statistics dialog box *or* press [CANCL] [CANCL] to return to the stack.

[P] **TO STORE ΣDAT TO A DIFFERENT NAME USING THE STACK INTERFACE:**

- Press [ΣDAT] [ENTER] to place the current ΣDAT on the stack.
- Enter ['] followed by your choice of name for the data and press [STO].

You should now see the softkey [INJ] in your [VAR] menu. Notice that [ΣPAR] has also appeared there. This variable is used in conjunction with ΣDAT for statistical plots. If you do not wish ΣPAR in your menu, press ['] [ΣPAR] [PURG].

Let's explore the stack interface data entry using the second column data (the 11 deaths directly attributable to football). After you input the data, name the matrix DEATHS using either of the two methods previously discussed.

TO ENTER ONE-VARIABLE STATISTICAL DATA USING STACK INTERFACE:

- Press ↰ STAT and DATA to bring up the statistics data entry menu.
- Press CLΣ *before* beginning data entry or entered values will be appended to the current ΣDAT matrix.
- Key in the first data value and press Σ+.
- Key in the remaining data values, pressing Σ+ after each to complete entry of data into ΣDAT.

If you make a mistake while keying in the data when using any of the data entry methods, you can correct the mistake before you enter the number by pressing ← and entering the correct value. To verify that the correct data values have been entered or to correct an incorrect value that has already been entered, use either of the two methods given below:

TO EDIT STATISTICAL DATA USING SCREEN INTERFACE ACCESS:

- Press ↱ STAT OK EDIT.
- Use the ▼ and ▲ cursor keys to scroll through the data and edit as necessary. Press ENTER to store the edited value.
- Press ENTER and OK to store the edited ΣDAT matrix.

[P] TO EDIT STATISTICAL DATA USING STACK INTERFACE ACCESS:

- Press [VAR] [ΣDAT] [▼].
- Use the [▼] and [▲] cursor keys to scroll through the data and edit as necessary. Press [ENTER] to store the edited value.
- Press [ENTER] to return to the stack and ['] [ΣDAT] [STO] to store the edited ΣDAT matrix.

Finding numerical descriptive measures is easy once data have been entered. Recall that the built-in statistics applications always use for calculations the data stored in ΣDAT. Therefore, if you wish to use previously entered data in calculations, you must designate that data as the current statistical matrix.

[G] TO DESIGNATE THE CURRENT ΣDAT MATRIX USING SCREEN INTERFACE:

- Press [↱] [STAT] [OK]. (Actually, any of the four choices may be used.)
- If there is already data in ΣDAT, press [DEL] [OK].
- Press [CHOOS] and highlight the matrix you wish to make the current statistics matrix.
- Press [OK] to temporarily store the matrix in ΣDAT for use in calculations from this interface. Pressing [OK] again replaces the ΣDAT in your variables menu with your chosen matrix; pressing [CANCL] cancels the ΣDAT change.

[P] TO DESIGNATE THE CURRENT ΣDAT MATRIX USING STACK INTERFACE:

- Press [VAR] (and if necessary [DATA]) to access your stored data.
- Enter the matrix you wish to make the current statistical matrix onto the stack by pressing the menu key corresponding to its name.

- Enter ' ΣDAT STO.

Now that you know about entering, editing, and manipulating one-variable data on the HP48G, let's use the data to find some numerical descriptive measures. Make the serious football injury data (INJ) the current ΣDAT and follow the instructions below to find the mean, standard deviation, variance, total, maximum, and/or minimum of the injury data using screen interface access.

TO FIND NUMERICAL DESCRIPTIVE MEASURES USING SCREEN INTERFACE:

- Press → STAT OK to choose Single-variable. Note that you can at this point choose the data set you wish to work with if you have not previously done so.
- Choose either Sample or Population in the second field with +/-.
- Use ▼, ▲, ►, and/or ◄ to move around the display pressing either √CHK or +/- to toggle on or off the measures you want calculated.
- Press OK and the tagged values appear on the stack. Pressing ▲ allows you to scroll through the stack values for viewing.

TO FIND NUMERICAL DESCRIPTIVE MEASURES USING STACK INTERFACE:

- Press ← STAT 1VAR to display the statistics numerical descriptive measures menu. You must have previously designated the current ΣDAT matrix.
- Press the softkey on the menu corresponding to the measure you desire, and the calculated value will appear on the stack. NXT scrolls you through the menu pages and STAT returns you to the main statistics menu.
- SDEV and VAR calculate sample standard deviation and variance respectively; PSDEV and PVAR calculate population standard deviation and variance.

Did you find $\bar{x} = 7.364$, $s = 3.557$, $s^2 = 12.655$, $\Sigma x = 81$, a maximum value of 12 and a minimum value of 1? If not, check the entry of your data. Note that Total = Σx provides a quick check on your entry of the data.

ENTERING AND EDITING MULTI-VARIABLE DATA

The ΣDAT matrix contains a row for each data *point* and a column for each *variable* measured at that point. Thus, each column of the matrix represents the values for a different variable in your data. The number of rows that are entered equals the number of data points. For instance, if you were entering the year, number of injuries and number of deaths in the football data as multi-variable data, the ΣDAT matrix would consist of 11 rows and 3 columns.

We consider another example. The following data relates gas mileage and weight of automobiles in a random sample of 15 automobiles:

Weight (in thousands of pounds)	Fuel Efficiency (in miles per gallon)
2.67	28.4
4.36	16.9
3.14	17.0
3.62	18.6
2.23	30.8
2.67	27.4
3.83	18.2
2.19	30.5
3.96	16.5
3.84	17.4
3.61	19.2
2.80	21.6
2.20	34.1
1.98	34.2
4.06	15.5

Using either of the two methods described below, enter these 15 data points, each consisting of the two variables weight and fuel efficiency, as multi-variable data.

Procedures for editing a statistical data matrix having more than one column are the same as those for one-variable data.

[G] TO ENTER MULTI-VARIABLE STATISTICAL DATA USING SCREEN INTERFACE:

- Press [→] [STAT] and [OK] to choose Single-variable. If there is data already in ΣDAT (i.e., the highlighted space to the right of **ΣDAT:** is *not* empty), press [DEL] [OK].
- Press [EDIT] to enter the MatrixWriter application.
- Key in the first data value, press [SPC], key in the next data value, press [SPC], etc. until you have entered all data in the first row of the matrix on the command line. Press [ENTER] and then press [▼] to move to the second row.
- Key in the remaining data values, separating those in each row with [SPC], and press [ENTER] after keying in the values for each individual row. (There is no need to press [▼] after the first row is entered.)
- Press [ENTER] to temporarily store the data into the ΣDAT matrix.
- Press [OK] to make the entered data the current ΣDAT matrix.

[P] TO ENTER MULTI-VARIABLE STATISTICAL DATA USING STACK INTERFACE:

- Press [←] [STAT] and [DATA] to bring up the statistics data entry menu.
- Press [CLΣ] *before* beginning data entry or entered values will be appended to the current ΣDAT matrix.
- Press [←] [[ ]] to let the HP48 know you are entering multi-variable data.
- Key in the first data value, press [SPC], key in the next data value, press [SPC], etc. until you have entered inside the brackets on the command line all data in the first row of the matrix. Press [Σ+].

- Key in the remaining data values, separating those in each row with [SPC], and press [Σ+] after keying in the values for each individual row. (There is no need to use the brackets after the first row is entered.)
- Press [ΣDAT] at any time during data entry to view the matrix.

For future reference and use, save these data to the name CARS. (When working with multi-variable data, storing ΣDAT to a different name and designating a matrix as the current ΣDAT matrix is the same as when working with one-variable data.) Press [VAR] and notice that both ΣDAT and CARS reside in your current user directory. Let's find some numerical descriptive information for the CARS data.

[→G] To find numerical descriptive measures for multi-variable data using the screen interface, follow the same instructions that are given for one-variable data. The only difference for multi-variable data is that after you enter the input screen, you must choose the column for which you wish the numerical measures calculated. Do this by using [▶] to highlight the field to the right of **COL:**, type in the number of the desired column, and press [OK].

Did you find a TOTAL of 47.16 for the weight (column 1) and a TOTAL of 346.3 for the fuel efficiency (column 2)? If not, check the entry of your data.

[→P] To find numerical descriptive measures for multi-variable data using the stack interface, follow the same instructions that are given for one-variable data. (Be certain that the data matrix you use is designated as the current ΣDAT matrix.) The only difference for multi-variable data is that numerical descriptive measures will be given for all columns when you press the proper keys in the STAT 1VAR menu. These will be given in the form of a vector with the individual statistics in the order of the columns.

For example, to find the mean fuel efficiency for the cars in the sample, press MEAN and observe the two-element vector of means. Since mpg is represented by the second column of ΣDAT, look in the second position of the means vector to find 23.0867. What is the variance in the weight of the sampled automobiles? Pressing NXT VAR gives the two-element vector of variances of this sample data as [.639 48.287]. The first column in the data is the weight, so the variance in the weight is the first of these numbers, .639.

Another numerical descriptive measure of a set of data is the *median*. Unlike the mean, which can be associated with the center of mass of a data distribution, the median measures the geometric center. Therefore, the median divides the data into two equal portions, the bottom 50% and the top 50%. The median is also known as the 50^{th} percentile. In describing the position of a particular data value in relation to the other measurements in a data set, you are using a numerical descriptive measure called a *percentile*. When your data set is arranged in order from smallest to largest, the p^{th} percentile is a number (which may or may not be one of the data values) that divides the bottom p% from the top $(1 - p)$% of the data.

There are two "hidden" programs, MEDIAN and %TILE, in your HP48G/GX that will calculate the median and other percentiles of a one-column set of data. Even though the median is the 50^{th} percentile and could be calculated using program %TILE, the input forms of these two programs are different. Sometimes it is more convenient to use one program than the other when calculating the median, so we will consider both. Program MEDIAN calls program %TILE, so both must be in the same directory.

To access these programs, first return to your home directory with HOME. Hold down α, and while holding it, type the word TEACH. Press ENTER and you should now notice EXAM in your home directory. The program to calculate the percentiles, %TILE, is in the EXAM subdirectory. Press EXAM PRGS and you should see the program to calculate the median, MEDIA. However, to use these programs, they

must be in the same directory as the current ΣDAT. Use the instructions below to copy programs MEDIAN and %TILE to the DATA directory.[1]

TO COPY OR MOVE VARIABLES USING THE VARIABLE BROWSER:

- Enter the directory that contains the variable (in this case, the program) you want to copy or move.
- Press [↱] [MEMORY] and a list of the variables in the current directory will appear.
- With the program name highlighted, press [COPY] (or [MOVE] if you wish to move rather than copy the variable.)
- Enter in the highlighted **COPY TO:** field the name of the directory in which you wish the variable to appear. (In this example, press [CHOOS] to obtain a list of the directories and their subdirectories. You should see DATA under ISTAT. Using [▲] and/or [▼], highlight DATA and press [OK]) .
- Press [OK] [NXT] [OK] to copy or move the variable and return to the stack.
- Press [HOME] and enter the directory or subdirectory to which you moved the variable to verify that it now appears where you placed it.

The CARS data should be your current ΣDAT. If not, designate it as such. Press [MEDIA] and the HP48G will return the vector [3.14 19.2] indicating that the median weight is 3.14 thousands of pounds and the median fuel efficiency is 19.2 mpg.

---

[1] You may need to correct program MEDIAN if you have an early version HP-48G/GX. Press [↱] [MEDIA] and see if STOΣ appears in the third line of the program between TRN and 1. If not, press [▼] and edit the program by inserting STOΣ immediately following TRN. Press [ENTER] [↰] [MEDIA] to store the corrected version of the program.

Now designate DEATHS as the current ΣDAT. Press MEDIA and you will find the median number of deaths due to football is 4. Notice that when using this program, the data does not need to be sorted; that is done within the program.

Program %TILE allows you to calculate percentiles other than the median. However, in order to use this program, data must be in the form of a list on the stack, *not* a matrix stored as ΣDAT. The list data is automatically sorted by program %TILE. Press CJST to place the Chief Justice list data on the stack. Enter the percentile you wish to calculate, say 25, on the stack and press %TILE to obtain 5 years as the 25th percentile for this data. Repeat the procedure for other percentiles, each time placing the list in level 2 of the stack and the desired percentile in level one before activating the program.

Suppose your data is not in the form of a list and you want a percentile other than the median or you wish to sort the data. You do not have to reenter the data as a list if you use one of the following two programs that will temporarily convert a specified column of a matrix to a list. Both programs should be placed in your DATA subdirectory. To store each program after it is entered on the stack, press ' *program name* STO.

Program SVTL ("single-variable to list" - converts a one-column matrix to a list)

<< OBJ→ OBJ→ DROP DROP →LIST >>

- Before using program SVTL, place a one-column data matrix on the stack.

Program MVTL ("multi-variable to list" - converts a specified column of a matrix of more than one column to a list)

<< COL⁻ OBJ→ OBJ→ DROP →LIST >>

- Before using program MVTL, place a matrix of more than one column in level 2 of the stack and the column number you want extracted in level one of the stack.

EXPLORATION EXERCISES

1. Compare the serious football injuries and the deaths directly attributable to football by calculating the following numerical descriptive measures for each. Give a written paragraph stating your conclusions.
   a) mean, median, mode
   b) range and standard deviation
   c) 25^{th}, 50^{th}, and 75^{th} percentiles

2. a) Would you choose the Sample or the Population setting in the screen interface method if calculating the mean and variance for the Chief Justice data? Support your answer.
   b) Recall that CJST is data in the form of a list. The statistics application uses data in ΣDAT, a matrix. Enter the following program which allows you to convert list data to a one-column matrix and store it as the current ΣDAT. (To type a lower case letter, press [α] [↰] before the letter key.)

Program LTSVM ("list to single-variable matrix" - converts a list to current ΣDAT)

```
<< CLΣ REVLIST OBJ→ →n
   << 1 n FOR k Σ+ NEXT >>
   ΣDAT >>
```

- Before using program LTSVM, place a list on the stack.

   c) Find the mean and variance for the Chief Justice data.

3. In 1990 every person in the U.S.A. produced an average of 3.5 pounds of solid waste per day. The problem of solid waste disposal worsens each year as the population and per-capita consumption continues to grow larger: [4]

Garbage in the United States

Year	1960	1970	1980	1990
Solid Waste (in millions of metric tons)	79.4	109.3	129.4	160

   a) Enter these data as a 4 by 2 matrix named TRSH.
   b) Find the mean amount of trash produced between 1960 and 1990. Give units with your answer.

c) Find the median amount of trash produced between 1960 and 1990. Give units with your answer.

d) i) Extract the solid waste column and convert it to a list. Leave this list on level 1 of the stack.

ii) Using the conversion 1 metric ton ≈ 1.102 tons (U.S.), change the solid waste data units to tons by entering 1.102 on the stack and pressing [×].

iii) Convert this list to the current ΣDAT matrix using program LTSVM. Name the converted data matrix TRTN.

e) Find the mean and median amount of trash produced between 1960 and 1990 in tons. Compare these values with your answers to b) and c) above.

4. Fifteen measurements taken at regular intervals of time of the level of asbestos fiber present in the air at an industrial plant gave the following data:

6 5 8 10 9 8 7 8 11 13 14 15 12 12 10

a) Enter this data in your calculator and save it as ASB.

b) What is the mean level of asbestos fiber in the air at the plant?

c) What is the median level of asbestos fiber in the air at the plant?

d) What is the variance of the level of asbestos fiber in the air at the plant?

e) What effect would changing the reading of 15 to 20 have on the mean? the median?

5. A mechanical engineer is studying the performance of an industrial process for manufacturing cement blocks. The following data is recorded:

Temperature (°C)	Concentration (%)	Response Yield (%)
70	21	78.4
60	25	79.5
90	21	78.3
110	22	78.9
120	21	78.1
70	25	79.3
90	23	78.9
110	25	80.3
90	25	79.8
60	22	78.0

a) Enter this data as a 10x3 matrix and name it CEMB.

b) Find the mean concentration and the mean response yield.

c) Find the median concentration.

d) Find the 25^{th}, 75^{th} and 90^{th} percentiles for the concentration data.

e) Find the 25^{th}, 75^{th} and 90^{th} percentiles for the response yield.

STATISTICAL GRAPHS

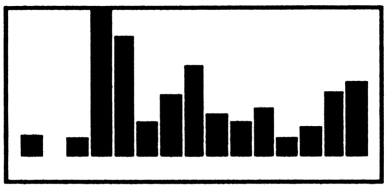
Bar charts, histograms and scatter plots are statistical graphs you can easily obtain from your HP48G/GX. The DRAW command takes the data for the statistical graph from the variable ΣDAT rather than from EQ that is used when plotting functions. You can also select and draw the graph of any of four regression models: linear, logarithmic, exponential or power. Other statistical graphs such as pie charts, box plots, and probability distribution graphs are readily obtained from user-entered programs.

BAR PLOTS

A bar plot drawn on the HP48 represents the data values with vertical bars of lengths proportional to the individual values. Let's explore this type of graph with the Chief Justice data. Before constructing the bar plot, the data you are using must be designated as the current ΣDAT. Use program LTSVM to convert the CJST list data to the current ΣDAT matrix.

- If the data is multi-variable, designate ΣDAT and the column of the matrix for which you wish the bar plot drawn by first pressing [↱] [STAT] [OK]. At this point, select ΣDAT and the appropriate column. Press [OK] to return to the stack and continue as if the data were single-variable.

Next, press ◀┐ STAT PLOT BARPL to draw the bar plot using the HP48G auto-scaling. Notice that a single bar is plotted for each data point in ΣDAT from left to right in the order in which the data appears in the matrix.



Press ON to return to the stack, and then press ◀. Notice the graph reappears. Until you erase this graph or draw another, you can recall the graph at any time by pressing ◀. Let's explore this bar plot. Press (X,Y) and you will see that the menu is replaced by the current location of the cursor. If you cannot see the cursor (+), it is because the cursor default setting is dark and it is on one of the dark bars of the graph. Press +/– and the cursor changes so that it appears dark against a light background and light against a dark background. Use the cursor keys (▲, ▼, ◀ and ▶) to move around the screen. To quickly move the cursor to the upper-left hand corner of the screen, press ↱ ◀ ↱ ▲, to quickly move it to the lower right-hand corner of the screen, press ↱ ▶ ↱ ▼, etc. Notice that auto-scaling the bar plot has set the x-range from 0 to *n*, where *n* is the number of points in ΣDAT. The y-range is set so that room is left for the menu at the bottom of the screen with the top of the tallest bar just reaching the top of the display screen. Pressing any of the white keys directly under the display returns the menu to the bottom of the graphics screen.

❒ HISTOGRAMS

A *histogram* is a statistical graph used to depict one-variable statistical data that has been grouped into classes. The classes appear on the horizontal axis and the frequencies or relative frequencies of the classes appear on the vertical axis. The HP48 can be used to construct a frequency histogram consisting of classes of equal width using

data in the current ΣDAT matrix. The classes into which the data are grouped are called *bins* with the number of data falling in each bin (that is, the *frequency*) depicted by a vertical bar.

There are several methods you can use to construct a histogram on your calculator. The first method we will discuss uses the PLOT menu, and the second method uses the STAT menu.

TO CONSTRUCT A HISTOGRAM USING THE PLOT MENU:

- Press [↱] [PLOT], use [▲] to move the the **TYPE:** field, and use [CHOOS] to select Histogram. Press [OK].
- Press [▼] and use [CHOOS] to select the desired data matrix. Press [OK].
- If the data matrix consists of more than one column, press [▶] and enter the column number containing the data for which you are constructing the histogram. (Column 1 is the default value.) Press [OK].
- In the **WID:** field, enter the width of each class. Dflt sets the width of each class of the histogram to one unit. Most statistics texts instruct to determine the equal class width by rounding *up* the value given by the formula

$$\text{width of class} = \frac{\max\Sigma - \min\Sigma}{\text{number of classes}}.$$

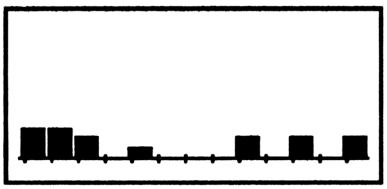
- In the **H-VIEW:** field, enter the horizontal display range (determined by the data values) and in the **V-VIEW:** field, enter the vertical display range (determined by the frequencies), *or*

  Check **AUTOSCALE** and press [ERASE] [DRAW] to draw the histogram.

The autoscale setting on the HP48G causes the display screen horizontal view to be between the smallest and largest entries in the chosen column of ΣDAT. The vertical view is then automatically chosen so that the setting for the top of the display screen

equals the number of entries in ΣDAT with the bottom of the screen being ⁻.15 times the top setting. This will usually make the bars appear fairly short.

Let's explore this histogram construction method with the CARS data. To draw a histogram of the fuel efficiency (mpg) values, choose CARS as the current ΣDAT and 2 as the column number. Choose Dflt in the width field and check AUTOSCALE to obtain



Notice that even though you only see 7 bars in this histogram, there are actually 13 intervals (bins) on the horizontal axis. Press ↱ PLOT and you will see that the horizontal view is 15.5 to 34.2 and the vertical view is ⁻2.25 to 15. Since there are 13 intervals and the width of the screen is 34.2 – 15.5 = 18.7, the width of each rectangle is approximately 18.7/13 (notice that a blank column of pixels is left between adjacent rectangles). Look now at the sorted mpg data:

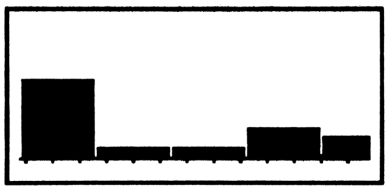
15.5 16.5 16.9 17 17.4 18.2 18.6 19.2 21.6 27.4 28.4 30.5 30.8 34.1 34.2

Since three of the data values are between 15.5 and 15.5 + 18.7/13 ≈ 16.94, namely 15.5, 16.5 and 16.9, the height or frequency of the first bar (rectangle) is 3. Return the histogram to the graphics screen with ◀.

Notice that when the histogram is on the screen you can estimate the heights of the rectangles with the graphic cursor. Realize, however, that the coordinates appearing at the bottom of the screen are pixel screen coordinates, not points on the graph you have drawn. Move the cursor to the top of the first bar, and press (X,Y) to display the cursor coordinates. The y-value will be a decimal number close to 3, and you should round this to the nearest whole number for the actual frequency. The second class, 16.94 to 16.94 + 18.7/13 = 18.39, contains the next three values in the sorted data

list. Therefore, the height of the second bar is 3. The data values have been grouped into classes (intervals) with the height of each rectangle being the number of data values in the particular interval. You should now verify that the frequencies shown by the 13 rectangles are 3, 3, 2, 0, 1, 0, 0, 0, 2, 0, 2, 0, and 2. The sum of the frequencies is 15, and there are 15 data values in ΣDAT.

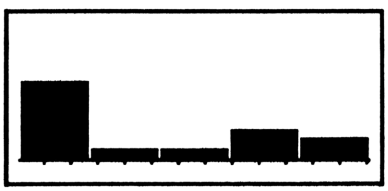
Suppose you wish the histogram to consist of 5 classes instead of 13. Determine the class width by the formula given in the histogram construction using the PLOT menu method to be (34.2 - 15.5)/5 = 3.74. There are several ways this value can be rounded up, but let's round up to the nearest integer, 4. Enter 4 in the width field and press ERASE DRAW to see



Why do you not see all of the last rectangle? The frequency distribution you are graphing is

mpg	frequency
15.5 - 19.5	8
19.5 - 23.5	1
23.5 - 27.5	1
27.5 - 31.5	3
31.5 - 35.5	2

Since the horizontal view is set as 15.5 to 34.2, you are not seeing all of the last bar. Change the highest horizontal value to 15.5 + 5(4) = 35.5 and redraw the graph with the keystrokes ERASE DRAW to obtain



Let's now explore using the STAT menu to construct a histogram. This method allows you to see the frequencies of the classes *before* the graph is drawn.

TO CONSTRUCT A HISTOGRAM USING THE STAT MENU:

- Press [↱] [STAT] to access the statistics screen interface and press [▼] to select Frequencies. Press [OK].
- Use [CHOOS] to select the desired data matrix. Press [OK] [OK] to store this matrix as ΣDAT.
- Repeat the first step to access Frequencies. If the data matrix consists of more than one column, press [▶] and enter the column number containing the data for which you are constructing the histogram. Press [OK].
- In the **X-MIN:** field, enter the smallest data value in the chosen ΣDAT column and press [OK]. Note that at any point during the construction of the histogram you can press [CALC] to temporarily return to the stack environment and use [↰] [STAT] to access information about the data.
- Enter the number of classes (bins) you wish in the histogram in the **BIN COUNT:** field and press [OK].
- Enter the width of each class in the **BIN WIDTH:** field and press [OK].
- Press [OK]. You will now be returned to the stack with the following output:

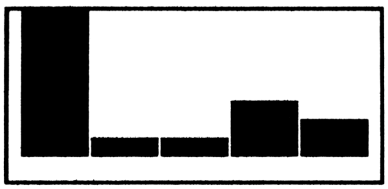
  Level 2: A "bins" matrix containing the frequencies of the classes.

  Level 1: An "excess" vector containing two numbers. The first number is the number of data values less than the minimum x-value you specified and the second number is the number of data values greater than the maximum x-value. This vector in level 1 should be [0 0] if you wish all data values included in your histogram.

Masking of values less than the minimum x-value or greater than the maximum x-value is accomplished by choosing an appropriate X-MIN value and class width and verified by observing the excess vector.

- To draw the histogram,
  1. Drop the excess vector from level 1 of the stack with [◀].
  2. Make the bins vector the new ΣDAT with [VAR] ['] [ΣDAT] [STO].
  3. Press [↰] [STAT] [PLOT] [BARPL].

Let's practice this method by reproducing the five-class histogram of the fuel efficiency data. After choosing Frequencies in the statistics screen interface, verify that the CARS data is the selected ΣDAT and that column 2 is the chosen column. Press [▼] to move to the **X-MIN**: field and then find the minimum value in the data to be 15.5 with [NXT] [CALC] [↰] [STAT] [1VAR] [MINΣ]. Press [ON] to return to the screen interface and enter the value 15.5. Since we are using 5 rectangles, enter 5 in the BIN COUNT field. Enter 4 as the BIN WIDTH (as we previously determined) and press [OK]. The excess vector is [0 0], indicating that all data values will be considered in the histogram. The vector [8 1 1 3 2] gives the frequencies of the distribution. Drop the excess vector and make the frequency vector the new ΣDAT. Draw the histogram with [↰] [STAT] [PLOT] [BARPL]. (If you get an Invalid ΣPAR error message, press [VAR] ['] [ΣPAR] [PURGE] and redraw the histogram with [↰] [STAT] [PLOT] [BARPL].)



Use the graphics cursor to note that the horizontal settings and cursor coordinates are now in terms of the number of bins, not the original data. Note that the graph is

essentially the same as was obtained using the PLOT menu method. The only visual difference in the graphs is that the BARPLOT command sets the height of the screen to be the height of the tallest rectangle.

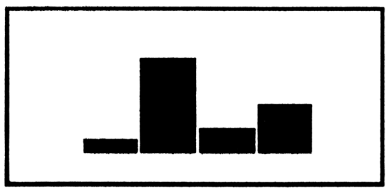
❒ APPLICATIONS OF THE STANDARD DEVIATION

It is often useful to compute the percentage of data values falling within one, two and three standard deviations of the mean of a set of data. You could use the graphic cursor to approximately locate the intervals $\bar{x} \pm s$, $\bar{x} \pm 2s$ and $\bar{x} \pm 3s$, and then roughly estimate of the number of data values falling in each of these intervals. You can, of course, look at the data and tally the number of values falling in each of these intervals to obtain the exact percentages. Why not have the HP48G do this?

Program DADSP sets a horizontal view from $\bar{x} - 3s$ to $\bar{x} + 3s$ with a class width equal to the standard deviation *s* and draws a histogram whose classes give the intervals $\bar{x} \pm s$, $\bar{x} \pm 2s$ and $\bar{x} \pm 3s$. The histogram constructed by this program will always consist of six intervals, $(\bar{x} - 3s, \bar{x} - 2s)$, $(\bar{x} - 2s, \bar{x} - s)$, $(\bar{x} - s, \bar{x})$, $(\bar{x}, \bar{x} + s)$, $(\bar{x} + s, \bar{x} + 2s)$, and $(\bar{x} + 2s, \bar{x} + 3s)$, since it counts three standard deviations on either side of the mean. The height of each bar can be found using the graphics cursor and represents the number (frequency) of data values falling in each of the indicated intervals.

```
<< STOΣ  SDEV  DUP  MEAN  SWAP  3 * DUP2 - 3
   ROLLD + XRNG  NΣ .8 * DUP  - .2 * SWAP
   YRNG RES HISTOGRAM ERASE DRAW 0 RES
   FUNCTION PICTURE >>
```

Program DADSP requires an nx1 ΣDAT matrix on level 1 of the stack as input. Suppose you wish to determine the number of data values falling within one, two and three standard deviations of the mean of the fuel efficiency values in the CARS data. Place the CARS matrix on level 1 of the stack, enter 2 to extract the mpg data, and press [MVTL] [LTSV]. Then execute program DADSP to obtain the graph



The mean of this data is 23.09 and the standard deviation is 6.95. Thus, the interval $\bar{x} \pm 3s$ is (2.24, 43.93). Notice from the graph on your calculator screen that there are no data values within two and three standard deviations on either side of the mean. Verify that the frequencies of the six intervals are, from left to right, 0, 1, 8, 2, 4, and 0. Thus, there are 8+2 = 10 data values in the interval $\bar{x} \pm s$, 1+8+2+4 = 15 values in the interval $\bar{x} \pm 2s$ and 0+1+8+2+4+0 = 15 values in the interval $\bar{x} \pm 3s$.

The *Empirical Rule* tells us that for mound (bell) shaped data, approximately 68% of the measurements fall within one standard deviation of the mean, approximately 95% of the data falls within two standard deviations of the mean, and approximately 99.7% of the data falls within three standard deviations of the mean. The percentage of data in these respective intervals for the fuel efficiency CARS data are 67%, 100%

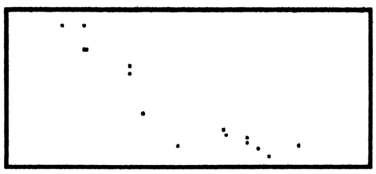
and 100%. We therefore see, as is indicated in the histogram, that this data is not really mound-shaped but slightly skewed.

SCATTER PLOTS

One of the best graphical displays of two-variable data is obtained with a *scatter plot* or *scatter diagram* that plots a point at each x-y coordinate pair. Information about the data set is obtained by looking at the scatter plot for a pattern and obvious deviations from that pattern. To construct a scatter plot on your HP48G/GX,

- Press [↱] [PLOT], use [▲] to move the the **TYPE:** field, and use [CHOOS] to select Scatter. Press [OK].
- Press [▼] and use [CHOOS] to select the desired data matrix. Press [OK].
- If the data matrix consists of more than two columns, press [▶] and enter the column numbers containing the data for which you are constructing the scatter plot. (Column 1 and 2 are the default values.) Press [OK].
- Check AUTOSCALE and draw the plot with [ERASE] [DRAW].

Let's see if we can observe a pattern in the weight of the automobiles and the fuel efficiency in the CARS data. Draw the scatter plot as instructed above. Notice that the autoscaling has placed points on the boundaries of the screen. If you wish a better view, uncheck AUTOSCALE and reset the horizontal and vertical views in the PLOT screen interface to values such as 1.5 to 5 and 12 to 36, respectively. Press [ERASE] and [DRAW] to redraw the scatter plot.



It certainly appears that as the weight of the car increases, the fuel efficiency decreases. We will return to further discuss this relationship later in this chapter.

EXPLORATION EXERCISES

1. The autoscaled barplot command BARPL on the HP48G is unique in that it plots bars of both positive and negative heights. Let's have the calculator construct a matrix consisting of both negative and positive values and construct the barplot to illustrate this type of graph. Press [MTH] [MATR] [MAKE]. Enter on the stack the dimensions of the matrix you wish to construct, say ten rows and one column, as the list { 10 1 }. Press [RANM] and use the methods of this section to construct the barplot.

2. Recall the cement block data CEMB and store it as the current ΣDAT matrix.

   a) Construct a scatter diagram of concentration versus response yield. What pattern do you observe? (Be certain that you set a range that allows you to clearly see all data points.)

   b) Construct a scatter diagram of temperature versus response yield. Does the temperature seem to have any effect on the response yield for this set of data?

   c) What are the mean and standard deviation for the response yield?

   d) Find the percentage of response yields falling within one, two and three standard deviations of the mean. How do these percentages compare with those given by the Empirical Rule for normally distributed data?

3. An agricultural scientist is studying the effect of diet on egg production. Twelve hens are fed with the current diet, Supergrow, and the egg production is measured. The same hens are then fed with a new diet, Growmore, for the same period of time. The results are

Hen	Egg Production	
	Supergrow Diet	Growmore Diet
1	140	138
2	155	160
3	132	135
4	138	145
5	150	152
6	125	125
7	141	135
8	173	189
9	160	165
10	150	156
11	148	154
12	163	178

a) Enter this data as a two-dimensional matrix and name it EGGS.

b) What is the total egg production for the hens when on the Supergrow diet? on the Growmore Diet?

c) Find the minimum and maximum value for the total egg production of hens on either diet.

d) Construct a histogram consisting of 5 classes of equal integer width for the Supergrow diet egg production data. Use a horizontal view of 125 to 190 and a vertical view of $^{-}2$ to 12.

e) Use the same horizontal and vertical views as in d) and construct a histogram consisting of five classes of equal integer width for the Growmore diet egg production data.

f) Compare the two histograms. What information is visually obtained about the egg production from hens on these two diets?

g) Compare the means, medians, and variances for the two diets. Comments?

h) What other questions could be asked about the egg production data? Choose what you consider to be the most interesting question and try to determine how to use your calculator to obtain a solution.

4. The following are the numbers of private aircraft which landed at a large metropolitan airport on fifteen consecutive days:

85 74 67 87 71 89 82 125 73 84 77 82 70 90 38

If you were an engineer hired to design a new runway, how might the information obtained from the following be helpful?

a) histogram

b) mean, median, mode, standard deviation

c) smallest x-value, 25^{th} percentile, 50^{th} percentile, 75^{th} percentile and maximum x-value

d) values more than 2 standard deviation away from the mean

■ PROBABILITY

Probability may be regarded as a numerical measure of the chance that a certain outcome (event) of an experiment will occur. The probability of an event E is denoted by $P(E)$ and is a real number *between 0 and 1* indicating the likelihood that E will happen. The closer the probability is to 1, the more likely the event is to happen, and the closer the probability is to 0, the less likely the event is to occur.

❐ SIMULATION TECHNIQUES

The *frequency* of an outcome is the number of times it occurs in repetitions of an experiment. When you divide the frequency by the number of repetitions, you obtain a fraction called the *relative frequency* of the outcome. Probability gives the *relative frequency* with which an event is *expected* to occur. The *Law of Large Numbers* states that if an experiment is repeated again and again under identical conditions, the relative frequency of an event will approach the theoretical probability of that event. However, many replications of the experiment under identical conditions may be difficult or impossible to perform.

Simulation is the process of representing an experiment with a model. The simulation technique has the advantage over the actual experiment in that many identical repetitions can be performed quite efficiently with the aid of a computer or in

this case, your calculator. Once simulations are performed, you can compare the outcomes of a large number of trials to the "theoretical" results. Simulation techniques usually involve *random numbers* - numbers chosen in such a way that each one is equally likely to be the one selected. Most statistics texts include a table of random numbers.

The HP48G has its own built-in program for generating pseudo-random numbers on the interval [0,1). A "true" random number generator on the interval [0,1) would select each *real* number in that interval with equal probability. Since it is impossible to simulate with the precision of real numbers, random number generators in calculators and computers generate pseudo-random numbers - that is, random numbers that are obtained to a fixed number of decimal places. These random outcomes look and behave for the most part like theoretical random numbers. There are $10^{12} - 1$ pseudo-random numbers on the interval [0,1) which may be obtained with the HP48G's random number generating function RAND. Access this function with MTH NXT PROB RAND. Press RAND several times and you will see some calculator-generated random numbers between 0 and 1.

The first random number that is generated is dependent on the value stored in the calculator memory. You may "seed" the random number generator (simulate randomly choosing a position in a table of random numbers) by entering any nonnegative number and pressing RDZ. You can repeat a particular series of random numbers by executing RDZ with the same argument.

To simulate the experiment of tossing one fair coin, let the outcome "tails" be represented by 0 and the outcome "heads" be denoted by the number 1. Since RAND yields a random number x such that $0 \le x < 1$, we have $0 \le 2x < 2$. Notice also that whenever $0 \le x < 0.5$, we have $0 \le 2x < 1$ and that whenever $0.5 \le x < 1$, it is true that $1 \le 2x < 2$. A command you will use with the random number generator is IP which is found on the second page of the MATH REAL menu. IP returns the integer portion of a

number, and entering RAND 2 * IP on the stack will return either a 0 or a 1 representing an outcome for the experiment of tossing of a fair coin.

Let's now look at a program that will allow the calculator to simulate tossing a coin. Return to your HOME directory and create a new directory called SIMU (simulation) to store the programs in this section. Store this program in SIMU with the name COINTOSS.

```
<< 1 RES ERASE 0 → n
   << { n } MENU HALT CLΣ n .8 * DUP -.2 *
      SWAP YRNG 0 2 XRNG 1 n FOR n RAND
      2 * IP Σ+ HISTOGRAM DRAW NEXT >>
   PICTURE 0 RES FUNCTION 2 MENU { ΣPAR
   PPAR CST } PURGE >>
```

- When this program is executed, you will see a menu containing the letter *N*. It is important to note that the number of tosses, *n*, must be entered in the program as a local variable (that is, using a lowercase letter obtained with [α] [↰] N). A local variable appears in the program menu as a capital letter.

- To run this program, enter the value of *n*, the number of times you wish to toss the coin, and press [↰] [N] to store the value as *n*. Notice the HALT message at the top of the display screen. Whenever HALT appears in this position, you should press [↰] [CONT] after the proper input to continue a program.

Use program COINTOSS to flip your calculator coin 10 times, 25 times, and then 50 times. For each experiment, record the number of heads (1's - the rightmost bar) and determine the relative frequency of the event {heads}. What value do the relative frequencies approach as the value of *n* increases?

Let's consider another simulation which this author calls the "histogram horse races". Enter the following two programs:

COINRACE

```
<< 1 RES ERASE 0 → n
   << { n } MENU HALT
      CLΣ n .8 * DUP
      -.2 * SWAP YRNG
      0 6 XRNG 1 n FOR k
      RAND 2 * IP RAND
      2 * IP RAND 2 * IP
      RAND 2 * IP RAND
      2 * IP + + + + Σ+
      HISTOGRAM DRAW
      NEXT >>
      PICTURE 0 RES
      FUNCTION 2 MENU >>
```

DIERACE

```
<< 1 RES ERASE 0 → n
   << { n } MENU HALT
      CLΣ n .6 * DUP
      -.2 * SWAP YRNG
      1 7 XRNG 1 n
      FOR k RAND 6 *
      1 + IP Σ+
      HISTOGRAM DRAW
      NEXT >>
      PICTURE 0 RES
      FUNCTION 2 MENU
      >>
   >>
```

When you execute each of these programs, you will see a histogram of 6 rectangles drawn one vertical block at a time. Think of each rectangle in the histogram as the moves for one horse on the race track. Thus, you are watching a race for 6 horses. Think of each block in the rectangles as a horse moving one unit. The total number of units moved by all the horses is *N*.

To execute either of the above programs, press the menu key in whose name you have stored the program. Load the value of *N* by entering the numerical value on the stack and pressing [↰] [N]. Press [↰] [CONT] to continue the program and run the 6 horse race. Good luck on picking the winner!

❒ PERMUTATIONS AND COMBINATIONS

The method of sampling, that is, how the elements of a sample are chosen, influences the number of outcomes for an experiment. If you sample *with replacement*, you return the element chosen to the population before you select the next element. If you sample *without replacement*, you do not return the element chosen to the population before choosing the next one. When the *order* in which the elements are chosen is important, the order in which the elements are selected must be considered.

A *permutation* is an *ordered* arrangement *without repetition of elements* of a set of distinct objects. The formula for the number of permutations of n different objects chosen r at a time is $nPr = n!/(n - r)!$ For the set {a, b, c}, the permutations are ab, ba, ac, ca, bc, and cb with 3P2 = 6. The permutation formula is in your HP48G in the MATH PROB menu. To use your calculator to find the number of permutations of n objects chosen r at a time, enter n, then r and then press PERM. Verify that you obtain 3P2 = 6.

A *combination* is a selection of the distinct objects of a set without regard to order. As with permutations, repetitions of elements are not allowed. The essential difference between permutations and combinations is that combinations ignore the order in which the objects are chosen. The formula for the number of combinations of n different objections chosen r at a time is $nCr = \frac{n!}{r!\,(n - r)!}$. For the set {a, b, c}, the combinations are ab, ac, and bc with 3C2 = 3. The combination formula is also in the MATH PROB menu. To use your calculator to find the number of combinations of n objects chosen r at a time, enter n, then r and then press COMB. Verify that you obtain 3C2 = 3.

EXPLORATION EXERCISES

1. Refer to programs COINRACE and DIERACE.

   a) How many coins are being tossed in the COINRACE program?

   b) If x is a random variable representing the number of heads obtained in the toss of these coins, what are the possible values for x?

   c) How many dice are being rolled in the DIERACE program?

   d) If y is the random variable representing the number of dots appearing on the upturned face of a die, what are the possible values for y?

   e) What is the relationship of x to y?

   f) Run each of these two programs using N = 10, 25, 40, and 55. Note for each the shape of the histogram as the value of N increases. Repeat if necessary. If you do not see a definite pattern emerging as the value of N gets larger, run the programs for N = 75 and N = 100. (This may take a while!)

   g) What theorem in statistics explains the different shapes of the histograms obtained from these two programs?

2. In order to determine whether a process producing bolts is stable, a quality control engineer selects a random sample of 15 bolts from a production lot consisting of 100 bolts.

   a) If the order in which the bolts are chosen is unimportant and each bolt is not replaced after examination, how many samples are possible?

   b) If the order in which the bolts are chosen matters and each bolt is not replaced after examination, how many samples are possible?

3. Suppose you want to simulate the birth of three children *N* times to estimate the probability of of obtaining two boys and 1 girl. Assume the birth of either sex is equally likely. Consider the variable of interest, x, to be the number of boys born. (For instance, if x = 2, that means two boys are born and the other child must be a girl.) Since the birth of either sex is equally likely, this simulation would be the same as tossing three coins and counting, say, the number of heads. Therefore, the simulation is accomplished with IP (2RAND) + IP (2RAND) + IP (2RAND).

a) Enter the following program and store it as BIRTH:

```
<< → n << CLΣ 1 n FOR k RAND 2 * IP RAND
   2 * IP RAND 2 * IP + + Σ+ NEXT >> >>
```

b) To simulate the birth of three children N times, enter the value of N on level 1 of the stack and execute program BIRTH. First let N = 10 and view the matrix ΣDAT containing the number of boys obtained in each of the three births. (For instance, you could consider the data in ΣDAT to be the results of a random survey of 10 families, each having three children. Each of the ten values in the matrix would then represent the number of boys in the family.) What are the possible values for x, the number of boys born?

c) Construct a histogram for the data using a class width of 1, a horizontal view from 0 to 4 and a vertical view from approximately 0 to .5N. Locate the rectangle corresponding to the event {the family has 2 boys}, and find the relative frequency of the event by dividing the frequency by N. Record this value.

d) Repeat parts b) and c) for N = 25, N = 50, and N = 75. What value do you find the relative frequency of the event {the family has 2 boys} approaching as N increases?

e) Would this experiment change if you let x be the number of defective items in a group of 3 items where each item is just as likely to be defective as good instead of being the number of boys in a family consisting of three children?

4. There is a game called MINEHUNT in your HP48G that is fun as well as educational. Press [↰] [EQ LIB] [UTILS] [MINE] to begin the game. The object of the game is to move from the upper left hand corner starting position to the lower right corner finish without being blown up. There are 20 mines randomly placed throughout the 8 by 16 grid. Your score increases by 1 each time you move to a "safe" light-colored block. Moves are made horizontally and vertically using the cursor keys. You can also move diagonally by pressing the 1, 3, 7, and 9 keys. The message at the upper left of the display screen tells you how many mines are possible in the squares to which it is possible to move. Play the game several times to become familiar with the possible moves.

a) If the 20 mines are randomly placed in the grid (there is never a mine in the starting or finishing position), use the combinations formula to determine the number of different ways the mines can be positioned in MINEHUNT.

b) If the 20 mines are randomly placed so that 5 must fall in the first row, 4 in the second row, 0 in the third row, 2 in the fourth row, 4 in the fifth row, 1 in the sixth row, 2 in the seventh row, and 2 in the eighth row, determine the number of different ways the mines can be positioned.

c) Suppose we number the squares according to the row and column in which they fall. For instance, in Figure 1 below, square A would be numbered 42, square B would be numbered 25 and square C would be numbered 44. When you start a new game, you are at the position of the darkened square in Figure 1. If you are told you are near 1 mine when you begin the game, what is the probability it is in square 21? (Assume the mines are randomly placed in the squares and that you do not know how many mines fall in any one row or column.)

d) Consider the game situation in Figure 2. You have already safely moved to each shaded square and given the information contained in that square. For instance, when you were in square 23, you were informed that you are near 3 mines. If the squares containing x's indicate the known location of mines,

i) which of the blank squares are safe to move into?

ii) which of the blank squares do you know contain mines?

e) Suppose, in Figure 2, you safely move to square 34 and are given the information that you are near 0 mines. What can you say about the blank squares in the fifth column?

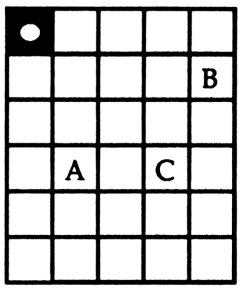


Figure 1

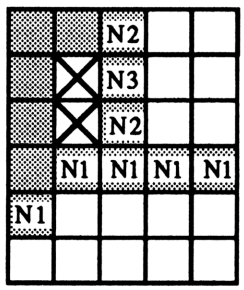


Figure 2

DISCRETE PROBABILITY DISTRIBUTIONS

Graphs of discrete probability distributions such as the binomial, Poisson and hypergeometric are readily obtained by programming the algebraic equation of the probability function. The probabilities for the distribution can be generated by a program and stored in the statistical matrix ΣDAT. This matrix may be renamed and stored for future reference if desired.

Because there are several programs used in this section, it will be helpful to create a directory to contain them. Return to the home directory, enter your statistics directory by pressing ISTAT, and create a subdirectory called DSCR. Press DSCR to enter the discrete graphs directory before entering the programs in this section.

BINOMIAL PROBABILITY DISTRIBUTION

The defining equation for the binomial probability distribution can be entered as an algebraic expression onto stack level 1 or can be entered into the HP48 Equation Writer. Access the EquationWriter with ↰ EQUATION and enter the binomial probability formula:

$$\text{COMB}(n,k)\,p^{k}\,(1-p)^{n-k}$$

Press ENTER and the equation is copied to the stack, ready for insertion into a program. Refer to your HP48G Series *User's Guide* if you are not familiar with the EquationWriter application.

Program BIDG, binomial distribution graph, will graph the binomial distribution probability histogram for the number of successes, x, between 0 and n. Notice that it also gives you a convenient way of calculating all the binomial probabilities p(x) for x = 0, 1, 2, ... , n. The program stores in the matrix ΣDAT the probabilities that are calculated. After the graph is drawn, press ΣDAT and you will see the matrix of probabilities which contains P(x = i) in row i+1 of the matrix.

Program BIDG should be entered in the DSCR directory and then stored with the keystrokes ' B I D G STO .

```
<< 0 DUP → n p
   << { n p } MENU HALT CLΣ 0 n FOR k
      'COMB ( n , k ) * p ^ k * (1 - p )
      ^ ( n - k )' EVAL Σ+ NEXT >>
   BARPLOT PICTURE 2 MENU { ΣPAR PPAR CST }
   PURGE >>
```

The program requires no user input before pressing BIDG .

To find the sum of several binomial probabilities, use program CUMPROB. Program CUMPROB will accumulate probabilities between a first value (F) and a last value (L) according to the value of x, not the position of $p(x)$, in the ΣDAT matrix.

Program CUMPROB (finds $\sum_{x=F}^{L} p(x)$ from the ΣDAT matrix of probabilities)

```
<< 1 + SWAP 1 + SWAP ΣDAT → b e s
   << s TRN STOΣ Σ- b e FOR i DUP i
      GET SWAP NEXT DROP e b - 1 SWAP
      FOR k + NEXT s STOΣ >>
>>
```

- Input for program CUMPROB is F, the first value in the sum, on level 2 and L, the last value in the sum, on level 1 of the stack.

Let's consider an example. Suppose you are taking a true-false test consisting of 20 questions. What is the probability distribution for the number of correct answers if you have not studied and only guess the answer to each question? Execute program BIDG for N = 20 and P = .5 and observe the graph. The mean number of correct answers is $\mu = 10$. (That's a score of 50% on the test.) Notice also that this distribution is symmetric about

its mean that occurs at the x value corresponding to the highest bar. What is the probability of scoring of at least 80%? To score 80% or better, you need 16, 17, 18, 19, or 20 correct answers. Run program CUMPROB with F = 16 and L = 20 to obtain 0.0059. Your chances aren't too good! Suppose that you have studied for the test and have a probability of 0.8 of obtaining the correct answer to any question. How does this affect the distribution of possible scores on the test? Execute program BIDG for N = 20 and P = 0.8. Notice that the graph has moved to the right, indicating that larger values of x are now more probable. The mean is $\mu = 16$. To find the probability that you obtain 16 or more correct answers, use program CUMPROB with F = 16 and L = 20 to obtain 0.6296. That's a better than even chance!

- The mean and standard deviation of the *probabilities* can be found using the screen interface or by pressing MEAN and SDEV with the stack interface. These, however, are *not* the mean and standard deviation of the probability distribution because the values of x have not been considered. You may modify the programs in this section to store the x-values as well as the binomial probabilities in a two-dimensional matrix ΣDAT or adapt techniques discussed in your text for the calculator.

❒ POISSON PROBABILITY DISTRIBUTION

The defining equation of the Poisson probability distribution may be entered in the EquationWriter application and copied into the following program or can be keyed in directly from the keyboard. The program below will graph the Poisson probability distribution histogram for the number of successes, x, between 0 and 10λ. Notice that, as in the case of the binomial distribution, the program gives you a convenient way of calculating the Poisson probabilities p(x) for x = 0, 1, 2, . . . , 10λ.

The following program, Poisson distribution graph, should be entered in the DSCR directory and then stored with the keystrokes ' P O D G STO .

```
<< 0 DUP → λ k
   << { λ } MENU HALT CLΣ 0 '10 * λ'
      FOR k 'λ ^ k * EXP(-λ) / k !'
      EVAL Σ+ NEXT >>
   BARPLOT PICTURE 2 MENU { ΣPAR CST }
   PURGE
>>
```

Special characters such as λ are obtained by pressing [↱] [CHARS]. To execute this program, press [PODG], enter the value of λ, the expected number of successes, and press [↰] [λ]. Press [↰] [CONT] to calculate the matrix ΣDAT containing the Poisson probabilities and draw the probability histogram. Program CUMPROB may be used to find cumulative Poisson probabilities.

❒ NORMAL DISTRIBUTION OVERLAY

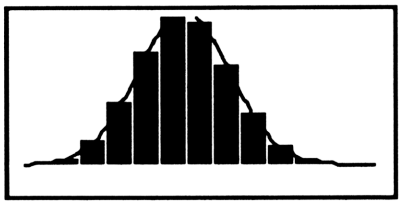
The normal distribution can in many cases be used to approximate probabilities for discrete random variables. Most statistics texts provide "rules-of-thumb" as to when such approximations are valid. Geometrically, whenever one considers an approximation by the normal distribution, the bell-shaped curve of the normal probability distribution should "fit" nicely over the graph of the distribution it is approximating. The following program, NDST, overlays a graph of the normal distribution on a probability histogram. The normal random variable should, of course, have the same mean and standard deviation as the discrete random variable for which you have drawn the histogram.

Enter the normal distribution graph program in the DSCR directory and store it with the keystrokes ['] N D S T [STO].

```
<< 0 DUP → μ σ
   << { μ σ } MENU HALT 'EXP(-(X-(μ + .5))
      ^ 2/(2 *σ ^ 2)) / (σ * √(2 * π))' STEQ
      2 MENU PPAR DUP 2 GET SWAP 1 GET
      C→R ROT C→R SWAP 3 ROLLD YRNG
      XRNG FUNCTION DRAW PICTURE >>
   { X EQ PPAR CST } PURGE >>
```

This program requires no user input before pressing [NDST]. When the program is executed with [NDST], you will see a menu containing the symbols μ and σ. Input the mean and standard deviation of the approximating normal distribution and continue the program to draw the graph. If you execute program NDST before erasing the contents of the graphics screen, the normal distribution graph will be drawn on top of the graph of the discrete probability distribution. You may press [◀] to retrieve the probability distribution graph with its normal overlay at any time before another graph is drawn.

As an example, we draw the graph of the binomial distribution for n = 12 and p = .45 using the program BIDG with the keystrokes [BIDG] 12 [↰] [N] .45 [↰] [P] [↰] [CONT]. After viewing the graph, press [ON] [NDST] 12 [ENTER] .45 [×] [↰] [μ] 12 [ENTER] .45 [×] 1 [ENTER] .45 [−] [×] [√] [↰] [σ] [↰] [CONT]. When the graph of the binomial distribution reappears on your screen, watch the overlay of the normal distribution whose mean and variance you have specified to be the same as that of the binomial distribution.



- If you look closely at the equation of the normal density function in the program NDST, you will notice that the graph has been shifted to the right 0.5 units. This is a standard procedure whenever you are graphing both a discrete probability distribution and continuous probability distribution on the same set of axes, and is called the *continuity correction.*

EXPLORATION EXERCISES

1. a) Use program BIDG to construct graphs for the binomial distributions with $n = 10$ and $p = 0.10$, $n = 10$ and $p = 0.35$, $n = 10$ and $p = 0.60$, and $n = 10$ and $p = 0.90$. How is the shape of the graph changing as p increases?

   b) Use the program BIDG to construct graphs for the binomial distributions with $n = 5$ and $p = 0.30$, with $n = 15$ and $p = 0.30$ and with $n = 25$ and $p = .30$. How is the shape of the graph changing as the value of n increases?

2. a) Use the program PODG to construct graphs for the Poisson distributions with $\lambda = 1.75$, $\lambda = 5$ and $\lambda = 10$. How is the shape of the graph changing as λ increases?

   b) Use the program PODG to construct graphs for the Poisson distributions with $\lambda = 2.63$ and $\lambda = 8$. In each case, enter the [↰] STAT menu and press [1VAR] [TOT] which will give you the sum of the probabilities. Do they sum to 1 as they should? Why or why not?

3. a) Use the program BIDG to construct graphs for the binomial distributions with $n = 5$ and $p = 0.20$ and with $n = 15$ and $p = 0.56$. In each case, overlay the graph of the normal distribution with the program NDST. Which bell-shaped curve best fits (in terms of the areas under the two being nearly the same) the underlying binomial distribution?

b) Use the program PODG to construct graphs for the Poisson distributions with $\lambda = 2.63$ and $\lambda = 8$. In each case, overlay the graph of the normal distribution with the program NDST. Which bell-shaped curve best fits (in terms of the areas under the two being nearly the same) the underlying Poisson distribution?

4. The probability that a certain type of aircraft engine fails during the first 10 years of operation is $p = .05$. Observe 20 of this type of engine and let x be the number that fail during the first 10 years of operation. Assume the status of one engine doesn't depend on the status of any other engine. Then, x has a binomial distribution with $N = 20$ and $p = .05$.

   a) Give the binomial probability distribution of x in tabular form. (Use BIDG.)

   b) Let $\lambda = np$ and give the Poisson probability distribution of x in tabular form.

   c) Would the Poisson distribution give a "good" approximation to the binomial distribution in this problem?

5. The number of daily plant shutdowns due to union problems follows a Poisson distribution with $\lambda = 2$. If the company loses $5000 for each shutdown, calculate the company's expected daily loss.

INFERENTIAL STATISTICS

Systematic methods which allow conclusions to be drawn from data while giving an associated measure of the reliability of those conclusions are the subject of *statistical inference*. The process of gathering data from an experiment with chance outcomes is called *sampling*. If all possible samples of a certain size were chosen from a given population and if, for each of those samples, a certain numerical value called the *test statistic* were computed, the value of this statistic would vary from sample to sample. The *sampling distribution* of the statistic is the probability distribution for all possible values of that statistic. The two major areas of inferential statistics are *estimation* and *hypothesis testing*. Conclusions drawn in both of these areas are based on the sampling variability of the test statistic.

❒ THE SAMPLING DISTRIBUTION OF THE SAMPLE MEAN

Many sets of data are approximately described by normal distributions, and many test statistics have sampling distributions that are close to a normal distribution. In particular, the sampling distribution of the sample mean $\bar{x}$ will approach a normal distribution as the sample size increases *regardless of the shape of the population distribution*. This result, perhaps the most important in all of statistical inference, is known as the *Central Limit Theorem*.

We can explore the Central Limit Theorem with a program that generates S random samples, each of size N, from a population having an exponential, standard normal, uniform, or Poisson distribution. The program draws a histogram of each sample, computes the mean of the sample, and then draws a histogram of the sample means computed from the S samples. The subroutines which generate the random values from the appropriate population are listed first. (You may wish to create a special directory to hold these programs.)

EXPRN (generates random values from an exponential distribution with mean λ and sets the range for the histograms of the samples)

```
<< → n  << CLΣ 1 n FOR k RAND LN NEG
   L * Σ+ NEXT >> 0 5 L * XRNG 0 NΣ 2
   / YRNG L 2 / RES >>
```

NMLRN (generates random values [5] from the standard normal distribution and sets the range for the histograms of the samples)

```
<< → n  << CLΣ 1 n FOR k RAND LN -2 *
   √ RAND 2 * π →NUM * RAD COS * Σ+
   NEXT >> -3.5 3.5 XRNG 0 NΣ 2 / YRNG
   1 RES { L } PURGE >>
```

UFMRN (generates random values from a uniform distribution and sets the range for the histograms of the samples)

```
<< → n << CLΣ 1 n FOR k RAND Σ+ NEXT
>> 0 1 XRNG 0 NΣ 3 / YRNG -1 RES >>
```

POSRN (generates random values [5] from a Poisson distribution with mean λ and sets the range for the histograms of the samples)

```
<< → n << CLΣ 1 n FOR k L NEG EXP -1
   1 DO SWAP 1 + SWAP RAND * UNTIL DUP
   4 PICK ≤ END DROP SWAP DROP Σ+ NEXT
   >> 0 L 4 * XRNG 0 NΣ .75 * YRNG L
   2 / RES >>
```

LBD (subroutine that prompts for λ value when exponential or Poisson populations are used)

```
<< 0 → λ
   << { λ } MENU HALT λ L STO 0 MENU >>
```

Program CLT (generates and graphs random samples from normal, exponential, uniform or Poisson populations and displays the graph of the sample means)

- Whenever you encounter the symbol ↵ in a program, you should begin a new line with the keystrokes [↱] [.].
- == is found in the PRG TEST menu , and the symbol $\bar{x}$ is accessed with the keystrokes [↱] [CHARS].

```
<< CLLCD "ENTER CHOICE AT PROMPT ↵ 1 FOR NORMAL , 2
FOR EXPONENTIAL , ↵ 3 FOR UNIFORM , ↵ 4 FOR POISSON " ↵
3 DISP 3 WAIT " CHOICE " "" INPUT OBJ→ DUP { NMLRN
EXPRN UFMRN POSRN } SWAP GET SWAP 2 / FP 0 IF
== THEN LBD END "# OF SAMPLES " "" INPUT OBJ→ 'S'
STO " SAMPLE SIZE " "" INPUT OBJ→ 'N' STO 1 S FOR
k DUP N SWAP EVAL HISTOGRAM ERASE DRAW MEAN
SWAP NEXT DROP CLΣ 1 S FOR i Σ+ NEXT RCLΣ DUP
MEAN TEXT " MEAN OF x̄ 's = " CLLCD 1 DISP 2 DISP 0
WAIT DROP ↵ " PRESS ENTER FOR THE SAMPLING DISTRIBUTION
OF THE SAMPLE MEAN " CLLCD 1 DISP 0 WAIT DROP
DADSP CLLCD { ΣPAR N S PPAR L CST } PURGE DROP
>>
```

- Program DADSP is a subroutine of this program and should be copied to the directory in which you place program CLT.

Since the sampling distribution of the sample mean is a probability distribution of *all* the possible means of the samples of size N chosen from a population, note that the graph of the sample means given by program CLT only *approximates* the actual sampling distribution of the sample mean and will vary with the values input for N and S. As the number of samples increases, the graph of the sample means given by the program will closer approximate the sampling distribution of the sample mean for samples of size N chosen from the specified population.

Consider what program CLT can tell you about the sampling distribution of the sample mean for samples chosen randomly from an population with an exponential distribution with mean λ. Execute program CLT, choosing 2 for the exponential population, input $\lambda = 2$ and let N = 5 and S = 10. Carefully observe the histograms of

the five values in each of the ten samples. Do you notice the rightward skewness of the data? Record the value of the mean of the distribution of the $\bar{x}$'s and press ENTER to observe the graph of the distribution of the means of the ten samples. Does the graph of the distribution of the means appear normally distributed? Probably not, because N = 5 is a small sample size. Repeat the experiment for N = 15, S = 10 and $\lambda = 2$. You will find the rightward skewness of the samples more pronounced, but again, the distribution of the sample means will not generally appear to be that of a mound-shaped normal distribution. Repeat the experiment once more, this time choosing N = 30, S = 10 and $\lambda = 2$. (You will get more consistent results using more samples, but the program takes longer to execute with larger values of *N* and *S*.) If you repeat this simulation several times, you will probably find that the distribution of the sample means for N = 30 will appear approximately mound-shaped most of the time.

Look at the values of the mean of the distribution of the $\bar{x}$'s each time you execute the program. Notice that most of them are close to λ, the mean of the exponential population. Repeat the entire experiment for the exponential population with $\lambda = 10$. You should obtain similar results with the exception that the mean of the distribution of the $\bar{x}$'s each time you execute CLT should now be close to 10. One possible histogram for a *sample* with N = 30 and $\lambda = 10$ is

A graph that could be obtained for the distribution of the sample *means* when N = 30, S = 15 and $\lambda = 10$ with mean of dist of $\bar{x}$'s = 9.622 is

Program CLT sets the same range as program DADSP when showing the graph of the distribution of the sample means at the end of the program. Recall that by the Empirical Rule, the percentage of sample means falling in the intervals $\bar{x} \pm s$, $\bar{x} \pm 2s$, and $\bar{x} \pm 3s$ should be close to 68%, 95% and 99.7%, respectively, provided N is large. Compare the percentages of sample means falling in each of these intervals for your last execution of the program to the Empirical Rule percentages. (Remember that you are performing a simulation with random values, and ten or fifteen samples is not infinitely many samples!)

Now perform the same experiment using the uniform distribution. The probability distribution resulting from the roll of a single die is an example of a discrete uniform distribution. The uniform distribution used in program UFMRN to generate random values for program CLT is an example of a *continuous* uniform distribution, where each number in the interval [0,1) has an equal chance of being selected for inclusion in the sample. The mean of this uniform population is 0.5. Observe the values for the mean of the distribution of the $\bar{x}$'s each time you execute program CLT for samples of size N = 5, 15 and 30 drawn from this uniform population. It is suggested that you again use 10 samples. Notice that the means of the distribution of the $\bar{x}$'s are close to 0.5 no matter what sample size you use. Repeated execution of this simulation for larger values of N and S should show the "level" or uniform nature of the graphs of the samples and, for large N, a mound- shape for the graph of the distribution of the sample means.

Consider now samples chosen randomly from a normal population. Subroutine NMLRN generates random values from the standard normal distribution with $\mu = 0$. What value would you expect for the mean of the distribution of the $\bar{x}$'s when you

execute program CLT for this population? Execute program CLT for samples of size N = 5, 15 and 30 drawn from the standard normal population. (We suggest that you again use 10 samples so that the program will not take too long to execute.) Carefully observe the shape of the samples and the shape of the sampling distribution of the $\bar{x}$'s for the small sample sizes N = 5 and 15. You should find an approximate mound shape for most of the histograms for the samples and for the distribution of the sample means, even when the sample size is small.

Finally, we explore the Poisson distribution with program CLT. Since the mean of the Poisson population is λ, the mean of the sampling distribution of the sample means should be close to λ. Execute program CLT, choosing the Poisson option with $\lambda = 2$, for N = 10 and N = 30. Again, choose S = 10. Repeat the experiment for $\lambda = 15$. Do you notice any difference in the shape of the graphs of the samples?

The above explorations with program CLT where intended to aid in understanding the following statistical theorems:

- The mean of the sampling distribution of the sample mean $\bar{x}$ *equals* the mean μ of the population from which the random samples are chosen.

- *Regardless of the size of the sample,* the sampling distribution of the sample mean is normal when the random samples are chosen from a normal population.

- *Regardless of the distribution of the population,* the sampling distribution of the sample mean approaches that of a normal distribution as the sample size increases. (The Central Limit Theorem)

❒ CONFIDENCE INTERVALS AND HYPOTHESIS TESTS

While statistical inference is the process of reaching conclusions about population values from evidence gathered in a sample, formal statistical reasoning is based upon a consideration of what would happen in many repetitions of an experiment that gathers evidence. Both interval estimation and tests of hypotheses use measures of reliability

that state what would happen if the decision were made many times under identical conditions. However, when the methods of inferential statistics are used to arrive at a conclusion, the results are based on a single sample, *not* the results of many repetitions of an experiment. We must therefore ask if the conclusions drawn from the single sample are convincing or are merely due to chance. The answer to this question involves the *confidence* that we have in the result, and the confidence is determined by the laws of probability and the sampling distribution of the appropriate test statistic. We need to understand the Central Limit Theorem in order to choose the proper test statistic and the appropriate formula. Because the HP48G/GX has built-in upper tail probabilities for the normal distribution, the t-distribution, the F-distribution and the chi-square distribution, it can be effectively used to construct confidence interval estimates and test statistics. The advanced equation solving capabilities of the SOLVE application make the HP48G especially useful in this regard. You may again find it helpful to create directories by application area to contain the programs that we shall use.

The following programs, NPRB and NVAL, call upon the calculator's built-in upper tail probabilities for the normal distribution to address hypothesis testing, confidence intervals and p-values (observed significance level).

Program NPRB (returns $P(x \geq a)$ where x has a normal distribution with mean μ and variance σ^2)

```
<< << μ σSQ X UTPN >> STEQ 30 MENU
   HALT 0 MENU { X μ σSQ EQ } PURGE >>
```

The program will place you in the SOLVE application and place the variables for the mean and variance of the normal distribution and the value of *X* to be used in the SOLVE menu. Execute the program and enter the values of the variables.

- Some input menus will contain dark letters on a white background (ordinary solver menu keys) and some will contain white letters on a dark background (local variables within lists that are active only within the program). To enter a value for a variable listed on a key with a *white background*, enter the value on the stack and press the menu key.

- To enter a value for a variable listed on a key with a *dark background*, enter the value on the stack, press [↰] and then the menu key of the variable. Whenever HALT appears at the top of the display screen you can continue program execution with [↰] [CONT].

After you have entered the values of the variables, press [EXPR=] and the HP48G will return the upper-tail normal probability for the entered value of *X*. For instance, to find P(z > 1) where *z* is the standard normal variable, enter μ = 0, σSQ = 1 and X = 1. Press [EXPR=] and you will see Expr: 0.158655253931. [↰] [CONT] returns you to the VAR menu and purges the values used in the calculation.

Program NVAL (returns the value of a such that P(x ≥ a) = P where *x* has a normal distribution with mean μ and variance σ^2)

```
<< << P μ σSQ X UTPN - >> STEQ 30 MENU
   HALT 0 MENU { X μ σSQ P EQ } PURGE >>
```

This program will place you in the SOLVE application and place the variables for the mean and variance of the normal distribution and the values of *X* and *P* to be used. This program is designed so that *P* equals the probability that the normal variable is *greater than or equal to X*; that is, the upper-tail probability. Execute the program and enter the values of the variables *P*, *μ*, and *σSQ*. Press [↰] [X] to solve for *X*. For instance, to find the value of z, say *a*, such that P(z > a) = 0.025, enter μ = 0,

σSQ = 1 and P = .025. Press [◀] [X] and you will see X: 1.95996398454. Again, [◀] [CONT] returns you to the VAR menu and purges the values used in the calculation.

A *confidence-interval estimate* of a *parameter* (population value) consists of an interval of numbers which is predicted to include the parameter and a probability that specifies how confident you are that the parameter lies in that interval. When the probability is expressed as a percentage, it is called the *confidence level.*

To find a confidence interval for the mean μ of a population where it is appropriate to use the z test statistic, program NVAL will give you the lower and upper endpoints of the confidence interval without you having to use formulas or tables from a textbook. Consider the problem: "A random sample of 80 observations from a population yielded a sample mean of 14.1 and a standard deviation of 2.6. Find a 95% confidence interval for the mean of this population." Recall that the mean, $\mu_{\bar{x}}$, of the sampling distribution of the sample mean equals the population mean μ and that the standard deviation, $\sigma_{\bar{x}}$, of the sampling distribution of the sample mean equals $\sigma/\sqrt{n}$. Use program NVAL and enter 14.1 [μ] 2.6 [ENTER] 80 [ENTER] [√] [÷] [x²] [σSQ] .975 [P]. Press [◀] [X] to solve for *X* and you will see the lower limit of the confidence interval as X: 13.5302603486. Key in .025 [P] and press [◀] [X] to solve for *X* and you will see the upper limit of the confidence interval as X: 14.6697396514.

Confidence intervals are easily calculated using program Zciμ from raw data that has been entered in an nx1 ΣDAT matrix. The program makes use of the built-in upper tail normal probabilities of the HP48G to give a large sample confidence interval. Raw data should be entered in the ΣDAT matrix in the statistics application. Before STAT is accessed to enter the data, be certain you are in the same directory in which program Zciμ resides. If not, you will have to store the ΣDAT matrix in that directory before you can execute this program. Input from the stack on level 1 is $(1-\alpha/2)$ to determine the

lower confidence limit and then the program is executed again with $\alpha/2$ on level 1 of the stack to determine the upper confidence limit for a $(1-\alpha)(100\%)$ confidence interval.

```
Program Zciμ    << 'P' STO MEAN 'M' STO SDEV SQ 'V' STO
                   << P M V X UTPN - >> 'X' M ROOT
                   { X P V M } PURGE 2 MENU >>
```

Confidence intervals using the t-distribution for one-sample tests are programmed and used in a similar manner except that the number of degrees of freedom must be used in the program, and UTPT is used instead of UTPN. The following program, TVAL, is similar to program NVAL except that the Student-t distribution is used instead of the normal distribution.

```
Program TVAL    << << P f X UTPT - >> STEQ 30 MENU HALT 2
                   MENU { X f P EQ } PURGE >>
```

This program will also place you in the SOLVE application and put the variables for the degrees of freedom (*f*) and the values of *X* and *P* on the menu. This program is designed so that *P* is the probability that the *t* variable is *greater than or equal to* X. Execute the program and enter the values of the variables *P* and *f*. Press [↰] [X] to solve for *X*. For instance, to find the value of *t*, say t_0 such that $P(t > t_0) = 0.025$ for 8 degrees of freedom, enter P = .025 and f = 8. Press [↰] [X] and you will see X: 2.3060041352. Press [↰] [CONT] to continue the program.

It is important to note that the command UTPT requires that the variable X is standardized. Thus, if you wish to use program TVAL to determine upper and lower *small sample* confidence limits, you must first standardize the variable. The following program, Tciμ, is used in such cases.

Program Tdiμ

```
<< 'P' STO { x̄ s n } MENU HALT n 1 - 'F' STO
   << P F X UTPT - >> 'X' 2 ROOT DUP IF 0 <
   THEN NEG END 'R' STO x̄ s n √ / R *
   DUP2 - 3 ROLLD + { X P F R x̄ n s CST }
   PURGE 2 MENU >>
>>
```

- Input from the stack is $\alpha/2$ to yield both the upper and lower confidence limits for a $(1-\alpha)(100\%)$ confidence interval.

When an interval estimate of the proportion p of successes in a binomial population is desired, program ZCIP can be used to obtain a $(1\text{-}100)\alpha\%$ confidence interval for p whenever the normal approximation to the binomial distribution is appropriate.

Program ZCIP

```
<< 'P' STO { x n } MENU HALT
   << P 0 1 X UTPN - >>
   'X' 2 ROOT DUP IF 0 < THEN NEG END
   'R' STO x n / DUP DUP 1 SWAP - * n
   / √ R * DUP2 - 3 ROLLD + { P x R n
   X CST } PURGE 2 MENU >>
```

- Input from the stack is $\alpha/2$ to yield both the upper and lower confidence limits for a $(1-\alpha)(100\%)$ confidence interval. The values of x, the number of successes in the sample and n, the sample size, are input from the menu called up by the program.

For confidence intervals involving two samples, you may find it easier to program the standard deviation of the test statistic as a separate quantity to avoid lengthy stack calculations. Of course, if you wish, the formulas may be programmed as algebraic expressions and solved using the SOLVE application. Several different

approaches to programming have been presented in this section. Refer to these when you are writing your own routines for evaluating the other confidence interval formulas found in elementary textbooks.

The second most common type of statistical inference is a test of significance, usually called a *hypothesis test,* which assesses the evidence provided by the data in regard to a claim or statement about a population parameter. Because this type of inference is based on sampling, there is always a chance that an error will be made in the decision. A *type I error* is made if the decision is to reject the null hypothesis when it is actually true. The probability of a type I error is denoted by α and is called the *level of significance* of the test. The value of α is usually predetermined by the experimenter before any sample results are obtained and represents the area under the sampling distribution of the test statistic corresponding to the set of values of the test statistic that lead to rejection of the null hypothesis. This region of values is called the *rejection region,* and the *critical value*(s) is the value(s) of the test statistic that separates the rejection region from the "do not reject" region.

A *type II error* is made if the decision is to accept the null hypothesis when it is actually false. The probability of type II error is denoted by β. Consult any of the standard textbooks for more information on hypothesis testing, the conditions under which each test statistic should be used, and the errors that are involved.

The probability of obtaining an outcome at least as far from what is expected if H_o were true is called the *p-value.* The smaller the p-value, the stronger is the evidence against H_o that is provided by the data. Therefore, the null hypothesis will be rejected for any choice α of that is greater than the p-value.

The following programs calculate the value of the appropriate test statistic for a hypothesis test of one population mean or proportion. Programs ZHTμ and tHTμ place you in the SOLVE application where you should input the quantities listed on the menu and press EXPR= to determine the value of the test statistic.

Program ZHTμ (computes the *z* test statistic for a large sample hypothesis test concerning the mean of a population - μ is the hypothesized value of the population mean)

```
<< '(x̄ - μ) * √ N / σ' STEQ 30 MENU HALT 0
   MENU { x̄ μ N σ EQ } PURGE >>
```

Program tHTμ (computes the *t* test statistic for a small sample hypothesis test concerning the mean of a population - μ is the hypothesized value of the population mean)

```
<< '(x̄ - μ) * √ n / s' STEQ 30 MENU HALT 0
   MENU { x̄ μ n s EQ } PURGE >>
```

Program ZHTp (computes the *z* test statistic for a hypothesis test concerning p, the proportion of successes in a binomial population - Po is the hypothesized value of the population proportion)

```
<< 0 0 0 → x n Po << { x n Po } MENU
   HALT x n / Po - n √ * Po DUP 1 SWAP
   - * √ / 0 MENU { CST } PURGE >> >>
```

Let's consider an example. For the hypothesis test H_0: $\mu = 12$ versus the alternative H_a: $\mu > 12$, suppose we are given that the mean of a random sample of size 45 chosen from the population of interest is 12.92 and that the sample standard deviation is 5.95. Execute program ZHTμ and input $\bar{x} = 12.92$, $\mu = 12$, $\sigma = 5.95$ (use s to approximate σ due to the large sample) and n = 45. You will find EXPR = z = 1.0372. This value can be compared to the critical value for *z* based on a given value of α = P(Type I error) and the appropriate conclusion reached.

To determine the observed significance level (p-value) for a hypothesis test, you can use the programs NPRB and NVAL (for *z* tests) or TVAL (for *t* tests). We shall find

the p-value for the hypothesis test conducted above: P(z > 1.0372) equals the p-value which can be obtained by either of two methods:

1) Using program NPRB, substitute, for the standard normal (z) distribution, $\mu = 0$, $\sigma SQ = 1$, $X = 1.0372$ and solve for the probability with [EXPR=]. You will obtain an answer of 0.1498 = P(z > 1.0372). Thus, the null hypothesis will be rejected for any value of $\alpha > .1498$.

2) Using program NVAL, substitute $\mu = 12$, $\sigma SQ = (5.95 / \sqrt{45})^2$, $X = 12.92$ and solve for the probability with [←] [P]. You will obtain $P(\bar{x} > 12.92) = 0.1498$. Thus, the null hypothesis will be rejected for any value of $\alpha > .1498$.

One of the conditions necessary to use the *t* test statistic is that the population from which the sample is drawn be approximately normally distributed. One way to assess the adequacy of the normal model is to construct a *normal quantile plot*, also called a *normal probability plot*. The general idea of this graph is to compare the sampled population distribution to the normal distribution by plotting their percentiles against one another. If the distributions are nearly the same, the data values will fall close to a straight line. If the distribution is negatively (leftward) skewed, the smallest observations fall distinctly to the left of a line drawn through the main body of the points. Positive (rightward) skewness or high outliers will cause the largest observations to fall distinctly to the right of a line drawn through the smaller data values. [2] A plot that bends down on the left and up on the right means that the data has longer tails than the normal distribution.

The construction of a normal quantile plot is usually done with computer soft-ware, but you can use program NQPLT and your HP48G to produce the plot. Programs SVTL and LTSVM in your DATA directory are subroutines of this program and must be copied to the directory in which you store program NQPLT and its subroutine ZVAL.

Program ZVAL (subroutine of program NQPLT)

```
<< << P 0 1 X UTPN - >> 'X' 2 ROOT >>
```

Program NQPLT (constructs normal quantile plot for assessing normality)

```
<< ΣDAT 'S' STO SVTL SORT LTSVM STOΣ MINΣ
   1 - MAXΣ 1 + XRNG -3 3 YRNG NΣ 'N'
   STO ΣDAT TRN { 2 N } RDM TRN DUP
   STOΣ 1 N FOR k k .375 - N .25 + /
   'P' STO P IF .5 ≤ THEN ZVAL NEG ELSE
   1 P - ZVAL NEG SWAP DROP END { k 2 }
   SWAP PUTI DROP NEXT STOΣ SCATTER
   ERASE DRAW PICTURE S STOΣ { X P N S
   ΣPAR PPAR } PURGE >>
```

- Stack level 1 input for program NQPLT is a single-column ΣDAT matrix. If you are working with a multi-variable ΣDAT, first use program MVTL to extract the desired column of data and then use program LTSVM to form the required nx1 ΣDAT input for program NQPLT.

Let's now consider a paired difference test. A common situation calling for a paired comparison experiment is a before-and-after study on the same subjects. A random sample is chosen from the population of pairs, and each pair is assigned a single value equal to the difference in the sampled pairs. The condition under which the result of this test is valid is that the population of paired differences is approximately normally distributed. Consider the following example.

Ten students are randomly selected from a large freshman engineering course at the beginning of the term and given a calculator test on evaluation of expressions. The same ten students are given a similar test one week after having received instruction in the use of the calculator. (A questionnaire administered with the test indicated that some

students were already familiar with the particular brand of calculator used on the test.) The results are as follows:

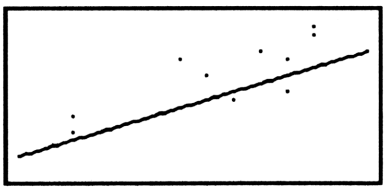
Student	Score Before Instruction	Score After Instruction
1	56	55
2	59	64
3	57	61
4	58	56
5	50	51
6	59	63
7	58	60
8	54	60
9	50	53
10	55	58

Using a level of significance of $\alpha = .01$ and the data above, can you conclude that the week of instruction in the use of the calculator significantly improves the test score? Let's call the mean of the population of test scores before receiving instruction μ_1 and the mean of the population of test scores after receiving the instruction μ_2. To see if the week of instruction significantly improves performance on the test, we test the hypothesis H_o: $\mu_1 - \mu_2 = 0$ versus the alternative hypothesis H_a: $\mu_1 - \mu_2 < 0$.

To perform the test, we need the differences in the data. Rather than risk making an arithmetic mistake in taking these differences, let's have the HP48 do the subtraction. There are several ways of accomplishing this task, but we will explore the method by which the data is first entered as a 10x2 matrix. After entering the data, name your matrix BFAFT.

Rather than just routinely performing computations, let's view the data points in relation to the line $y = x$. Why? If the points appear close to the line, you would probably believe that $x = y$ ($\mu_1 - \mu_2 = 0$). If most of the data points are above the line, you would suspect that $x < y$ ($\mu_1 - \mu_2 < 0$), and if the majority of the data points are below the line, you would suspect that $x > y$ ($\mu_1 - \mu_2 > 0$). Construct a scatter plot with the *before* scores (x) as column 1 and the *after* scores (y) as column 2 using the

AUTOSCALE feature. To obtain a slightly better view of all the points, reset the horizontal view to values like 48 to 61 and the vertical view to values such as 45 to 66. Redraw the scatter plot. To overlay the line y = x, press [↱] [PLOT] and choose **TYPE:** Function. In the **EQ:** field, enter X. Leave the other settings as determined for the scatter plot and press [DRAW]. Press [–] to eliminate the menu from the bottom of the screen for the best view.

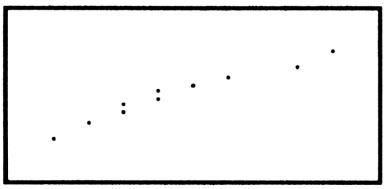


Notice that most of the data points are above the line indicating that "test score before instruction" – "test score after instruction" is negative most of the time.

To perform the hypothesis test, first find the data of differences by extracting each column with program MVTL, subtracting the resulting lists, and converting the list of differences to the 10x1 matrix DIFF with the keystrokes: [BFAFT] 1 [MVTL] [BFAFT] 2 [MVTL] [SWAP] [DROP] [–] [LTSVM] ['] D I F F [STO]. Since we are equivalently testing H_0: $\mu = 0$ versus H_a: $\mu < 0$ where μ is the mean of the population of differences, we will use program tHTμ. This program puts you in the SOLVE mode and requests input of the sample mean difference, sample standard deviation, and common sample size. Store the matrix of differences, DIFF, as the current ΣDAT and use either the screen or stack interface method to find $\bar{x} = -2.5$ and $s = 2.55$. Enter n = 10 and μ, the hypothesized value of the mean, as 0. Press [Expr=] to obtain $t = -3.1$. Now execute program TVAL by entering f = 9, X = -3.1, and solving for *P* with [↰] [P] to obtain P = .994. Recall that program TVAL is designed so that *P* is the probability that the *t* variable is *greater than or equal to* X. We can therefore state that $P(t \geq -3.1) = .994$ which is equivalent to saying that the p-value for this test is

$P(t < -3.1) = 1 - .994 = .006$. Since the p-value of .006 is less than $\alpha = .01$, we reject the null hypothesis and conclude that the calculator instruction significantly increases the score on the test.

Are the conditions of this test met; that is, can we consider the population of differences to be normally distributed? Press BFAFT to place the sample difference data on level 1 of the stack, and press NQPLT to construct the normal quantile plot. (The program takes a short time to execute.)



Do you feel you can consider the difference data approximately normally distributed based on this graph?

Many statistical applications concern comparing the means of two or more populations. The procedure for comparing these means involves analyzing the variation in the sample data. When comparing two population means, the sources of variability are the difference between the sample means and the variability within the two samples chosen from those populations. In analysis of variance procedures for comparing the means of three or more populations, variability is measured and allocated among its sources. The process of planning the experiment to collect the data, the *experimental design*, determines the proper analysis of variance, or ANOVA, procedure.

The hypotheses that are tested in the one-way ANOVA procedure for a completely randomized design are of the form H_o: $\mu_1 = \mu_2 = \mu_3 = \cdots = \mu_k$ versus H_a: Not all the means are equal (i.e. at least two of the means are unequal) where μ_i is the mean of the i^{th} population and k is the number of populations. In one-way ANOVA, the variation between the sample means is measured by a weighted average of the

squared deviations about the mean of the combined sample data, SSTR/(k-1). The measure of variation within the samples is the pooled estimate of the assumed common population variance and is denoted by SSE/(n-k) where n is the total number of data values from all the samples. An F test statistic equal to SSTR/(k-1) ÷ SSE/(n-k) and its p-value are used to determine if the alternative hypothesis is statistically significant. If the null hypothesis is true, the numerator and denominator of the F statistic should be approximately the same. Therefore, large values of F indicate the null hypothesis of equal population means should be rejected.

The ANOVA procedure is based on the assumptions that the randomly selected samples are independent of one another, each of the populations is normally distributed, and that the variances of the populations are equal. As long as the number of values in each sample is not too small, program NQPLT can be used for each sample to see if the data look reasonably normal.

The following program performs the lengthy computations of the one-way ANOVA procedure. Subroutine FVAL calculates areas associated with the F probability distribution using the built-in upper-tail F probabilities in the HP48G.

Program FVAL `<< << P N D X UTPF - >> 'X' 2 ROOT >>`

Program ANOVA (tests the hypothesis of equal population means for three or more populations)

```
<< 0 0 0 0 → T A B E
   << " Number of samples? "  " " INPUT  OBJ→ 'K'
   STO " ENTER INFO FOR EACH SAMPLE AT BEEP "
   CLLCD 1 DISP 1 WAIT 1 K FOR i 340 .3 BEEP
   { n x̄ s } MENU HALT n DUP DUP2 T + 'T'
   STO x̄ * A + 'A' STO x̄ SQ * B + 'B' STO
   1 - s SQ * E + 'E' STO NEXT K DUP 1 -
   SWAP T SWAP - B A SQ T / - * SWAP E
   * / DUP 'F' STO "F" →TAG K 1 - T K - F
   UTPF "p value" →TAG { F s x̄ n CST K } PURGE
   2 MENU >> >>
```

To execute this program, press [ANOVA] and input the number of samples (treatments) at the displayed prompt. Press [ENTER] and when the HP48G beeps, input the size, mean and standard deviation of the sample chosen from the population with mean μ_1. Press [↰] [CONT] and when the calculator again beeps, input the size, mean and standard deviation of the sample chosen from the population with mean μ_2. Continue this process until all sample information has been entered. The program will output the *F* statistic and p-value for the test. Whenever *F* is greater than the p-value, the null hypothesis of equal population means should be rejected.

EXPLORATION EXERCISES

1. The quality control engineer in a plant producing cans of cola wishes to estimate the mean amount of soft drink in 12 oz. cans filled by a machine in the plant.

   a) A random sample of 50 cans yields the results $\bar{x} = 11.95$ oz. and $s = 0.2$ oz. Find a 95% confidence interval for the true mean (μ) amount of cola in all cans filled by this machine.

b) What interval would you obtain if you used program Tciμ? Explain why the intervals are nearly identical.

c) Suppose the sample size in this problem had been 15 instead of 50. Use the same sample results and both programs to obtain a 95% confidence interval estimate of μ. Why is the interval obtained using the *t*-distribution wider (including more values) than the corresponding z interval? Which is the proper program to use to obtain the interval when n = 15?

2. The interpretation of the numerical value of the confidence in an interval estimate is often misunderstood. The population parameter μ is constant and does not have a sampling variability. The probability that μ falls in the generated confidence interval is either 0 or 1 depending on whether or not the actual value is between the two endpoints of the interval. Let's experiment to explore this very important idea. Suppose we could generate many interval estimates from a population with a known mean. We could then examine the intervals and determine the proportion (relative frequency) of those intervals that actually contain the population mean. Let's do it! Any simulation which generates random values from a distribution with a known mean could be used, so use subroutine UFMRN to generate N=30 random values from the uniform population with mean $\mu = 0.5$. Use program Zciμ to determine a 90% confidence interval for the mean of the population and record whether or not the value $\mu = .5$ falls within the interval. Repeat the procedure to obtain 10 confidence intervals. Divide the number of intervals which contain μ by 10 to obtain the simulated value of the confidence that μ will fall in the interval. You may not obtain 90%, but the more times you repeat this procedure, the closer you will find the simulated confidence approaching 90%.

3. We consider a *paired difference test*. Recall that whenever the elements of two populations are matched or paired by design, outcomes are compared within each matched pair. A common situation calling for a paired comparison experiment is a before-and-after study on the same subjects. To each pair is assigned a single value, the difference in the two individual values, a random sample is chosen from the population of pairs, and a *t*-test is used with the data consisting of the differences in the sampled pairs.

   Return to the EGGS data used to study egg production with diets Supergrow and Growmore. Use matrix methods to compute the matrix of difference data with entry x_i = x(Growmore) – x(Supergrow). To test the null hypothesis that there is no difference in the diets versus the alternative that the Growmore diet produces significantly more eggs than the Supergrow diet, the population of paired differences should be approximately normally distributed. What conclusion do you reach using program NQPLT? If appropriate, conduct the hypothesis test using a level of significance .05.

4. An environmental systems engineer is interested in making an estimate of the proportion of small streams with a pollution level higher than 10 ppm. He randomly samples 91 small streams and finds that 15 of them have pollution levels exceeding 10 ppm.

   a) Give a 90% confidence interval estimate for the proportion of all small streams with a pollution level higher than 10 ppm.

   b) Would a 98% confidence interval be wider or narrower than the 90% interval? Explain.

5. Tensile strengths of synthetic fiber used to make cloth for automobile upholstery is of interest to manufacturers. It is suspected that the strength is affected by the percentage of cotton in the fiber. Using the data given below (5 samples, one for each % cotton) collected from 20 experiments run in random order and program ANOVA, do you feel that there are differences in the strength due to the percentage of cotton used?

% Cotton	Tensile Strength (lbs/sq in)			
20	8	15	11	9
25	17	12	14	18
30	15	10	17	18
35	19	18	11	16
40	15	17	19	14

REGRESSION

Many statistical problems are concerned with the relation, if indeed a relation exists, between two or more variables. We usually wish to know if the variables of interest are related, and if so, what is the nature of the relationship? If an appropriate mathematical model can be found, how can information about one of the variables be used to predict another?

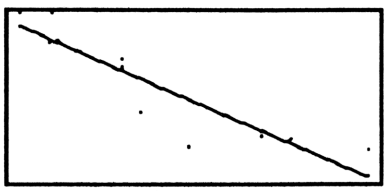
It is sometimes the case that a possible relationship between two variables can be visually identified by looking at a scatter diagram of the data points. When a scatter plot suggests that the dependence of the *response variable* y on the *explanatory variable* x

can be summarized by a straight line, the *linear regression* model is appropriate. The equation $y = a + bx$ with y-intercept a and slope b is the equation of the *regression line* determined using the *least squares criterion* which minimizes the sum of the squares of the vertical deviations (*residuals*) of the data points from the fitted line.

To calculate the linear regression coefficients *a* and *b* (or the coefficients of other regression models) using the HP48G and a ΣDAT matrix of at least two columns, access the statistics screen interface ([↱] [STAT]) and select Fit data. Choose the data matrix and enter the column number for the explanatory variable in the **X-COL:** field. Enter the column number for the response variable in the **Y-COL:** field. (The default values are column 1 for x and column 2 for y.) At this point, choose a model and press [OK]. The equation of the regression model is returned to level 3 of the stack, the correlation coefficient is given in level 2 and the covariance appears in level 1. These quantities can also be obtained from the stack interface ([↰] [STAT]) [FIT] folder after the regression model has been chosen. If you are dealing with population and not sample data, you should use [PCOV] for the covariance.

We shall explore these ideas with the CARS data. After designating CARS as the current ΣDAT matrix, select Linear Fit in the **MODEL:** field. Press [OK] and see that the calculated regression line for fuel efficiency on weight appears in level 3 of the stack as $y = 48.609 - 8.118x$. (Press [▲] [▲] [▲] [VIEW] or [◀] [◀] to see the entire expression.)

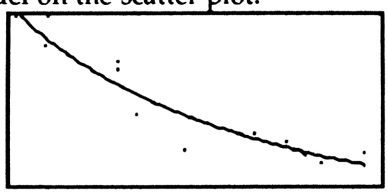
We next view the regression line on a scatter diagram of the data. Use the PLOT menu to draw the scatter diagram and then press [STATL] to observe



How good is this model? There are several conditions concerning the residuals which must be checked before you can give a complete answer to this question, and you should consult any standard statistics textbook on this issue. You can, however, obtain an indication of the fit of the model by simply viewing the line on the scatter plot. The correlation of -.933 and graph for the CARS data does not rule out the possibility of a linear fit.

You can obtain one of three other regression models on the HP48G. These are accessed by choosing the model in the **MODEL:** field of the Fit data screen interface. They are Logarithmic Fit ($y = a \ln x + b$), Exponential Fit ($y = be^{ax}$), and Power Fit ($y = bx^a$). The logarithmic, exponential and power regressions are calculated by applying logarithms to the data to transform the model to a linear one, performing a linear regression on the transformed data, and then inverse-transforming the computed parameters to the original model form [6].

The correlation coefficient, as well as the other regression menu keys, apply to the model you have chosen. You could consider each of the available models, compute the correlation coefficient for each, and then select the one for which the correlation has the largest absolute value as the "best-fitting" available model. The HP48G will do this for you if you select Best Fit. The chosen model appears in the third line of the stack and can also be accessed with [↰] [STAT] [FIT] [ΣLINE]. Verify that the power model with equation $y = 71.395x^{-1.05}$ and correlation coefficient -.952 is what Best Fit returns for the cars regression. Redraw the scatter diagram and use [STATL] to observe the graph of the power model on the scatter plot.



Once the appropriate conditions have been checked and a "best-fitting" model found, predicted values for the independent and response variables may be determined with the [↰] [STAT] [FIT] menu commands PREDX and PREDY. For instance, to predict the fuel efficiency for an automobile weighing 3000 pounds, enter 3 on the stack (remember the x-data is in thousands of pounds) and press [PREDY] to obtain y = 22.516 mpg. To find the predicted weight of an automobile whose fuel efficiency is y = 25, enter 25 on the stack and press [PREDX] to obtain x ≈ 2715 pounds. Predicted values can also be obtained from the statistics screen interface. After choosing a model, press [PRED]. If you wish to find the predicted weight of an automobile whose fuel efficiency is y = 30, enter 30 in the **Y:** field, move to the **X:** field and press [PRED]. To find a predicted y-value based on a given x-value, follow the same directions, but enter the known value in the proper location.

At this point, a word of warning is necessary. When you predict y-values for x-values that are within the range of data in your scatter plot, you are using a process called *interpolation*. Predicting y-values for x-values that are well beyond the range of the data is called *extrapolation*. Since you do not know how the data will behave outside the range you have plotted, extrapolation will very often lead to incorrect predictions.

EXPLORATION EXERCISES

1. Use the data on garbage in the United States (TRSH) to find a linear regression model. Is this the best fit for the data? If not, find the best fitting available model.

   b) Use the best-fitting model to predict the amount of garbage in the United States in the year 2000. Give units with your answer. Do you feel this prediction is realistic? Explain why or why not.

2. The National Center for Health Statistics lists years of life expected at birth as follows. [7]

Year	White Males	White Females	Black and Other Males	Black and Other Females
1920	54.4	55.6	45.5	45.2
1930	59.7	63.5	47.3	49.2
1940	62.1	66.6	51.5	54.9
1950	66.5	72.2	59.1	62.9
1960	67.4	74.1	61.1	66.3
1965	67.6	74.7	61.1	67.4
1970	68.0	75.6	61.3	69.4
1971	68.3	75.8	61.6	69.8
1972	68.3	75.9	61.5	70.1
1973	68.5	76.1	62.0	70.3
1974	69.0	76.7	62.9	71.3
1975	69.5	77.3	63.7	72.4
1976	69.9	77.5	64.2	72.7
1977	70.2	77.9	64.7	73.2
1979	70.8	78.4	65.4	74.1
1980	70.7	78.1	65.3	73.6
1981	71.1	78.4	66.1	74.4
1982	71.5	78.7	66.8	75.0
1983	71.7	78.7	67.2	74.3
1984	71.8	78.7	67.4	75.0
1985	71.9	78.7	67.2	75.0
1986	72.0	78.8	67.2	75.1
1987	72.2	78.9	67.3	75.2
1988	72.1	78.9	67.4	75.5
1989	72.6	79.1	67.5	75.7

If you are a male:

a) Construct a scatter plot for the year versus white male life expectancy at birth. The year in which the data was recorded should be the explanatory variable. *Store the graph.* What information does the graph for the white male life expectancy reveal? Discuss any pattern or deviations from a pattern you observe.

b) Look now at the scatter plot for the year in which the data was recorded versus black and other male life expectancy. *Store the graph.* What information does the graph for the black and other male life expectancy reveal? Discuss any pattern or deviations from a pattern you observe.

c) What historical events might account for any deviations from a general straight line pattern for these two data sets?

d) Recall the scatter diagram for white male life expectancy so that it is overdrawn on the scatter diagram for black and other male life expectancy. Do you see any pattern relating white male life expectancy to black and other male life expectancy? If so, are there any significant deviations from that pattern? Comments?

If you are a female:

Answer parts a), b) and d) of this exercise with the word *female* substituted for *male*.

c) Recently there has been very little increase in female life expectancy. Discuss possible reasons for this behavior of the data.

For both sexes:

e) Find the best-fitting regression model your calculator has to offer for the data for your sex and race. Predict the life expectancy in the year 1993. Predict the life expectancy in the year 2005. Do the estimates seem reasonable? Discuss why or why not.

3. Consider the following data [4] reported by the Bureau of the Census and the Bureau of Labor Statistics on women in the labor force.

Year	% of female population aged 16 and over in the labor force
1900	18.8
1910	21.5
1920	21.4
1930	22.0
1940	25.4
1950	33.9
1960	37.8
1970	43.4
1980	51.6
1988	56.6
1989	57.5
1990	57.5

Find the best-fitting available model and construct a graph of the model on a scatter plot of the data. Determine and interpret the correlation coefficient.

REFERENCES

1. Fetta, I., *Calculator Enhancement for Introductory Statistics: A Manual of Applications using the Sharp EL-5200, HP-28S and HP48S Graphing Calculators*, Saunders College Publishing, 1992, Philadelphia, PA.
2. Moore, D. and G. McCabe, *Introduction to the Practice of Statistics*, W. H. Freeman and Company, 1989, New York.
3. Raven, P., L. Berg and G. Johnson, *Environment*, Saunders College Publishing, 1992, Philadelphia.
4. *The 1992 Information Please Almanac, Atlas and Yearbook*, 45th edition, Houghton
5. Wickes, W. C., *HP-28 Insights, Principles and Programming of the HP-28C/S*, Larken Publications, 1988, Corvallis, Oregon.
6. Wickes, W. C., *HP48 Insights, Part II: Problem-Solving Resources*, Larken Publications, 1992, Corvallis, Oregon.
7. *World Almanac and Book of Facts*, Press Publishing Company, 1990, New York.

APPENDIX
PROGRAM HOUSEKEEPING

The term user memory refers to that part of the 48-G/GX's memory that we use to store the various types of objects recognized by the calculator, e.g., real or complex numbers, arrays, programs, lists, etc. These objects are stored as *global variables* (in calculator terminology) and you are provided access to them by pressing the [VAR] key to open the VAR (= variables) menu. Here we are concerned with the basic "housekeeping" procedures used to enter, name, store, run, edit and purge programs.

WHAT IS AN HP-48G/GX PROGRAM? A program is a sequence of data objects, procedures, commands and program structures - the *program body* - enclosed between program delimiters « »:

« *program body* » .

ENTERING PROGRAMS. Programs are keyed into the command line and entered onto the stack (level 1) with [ENTER]. You need not key in the necessary closing program delimiters because pressing [ENTER] will automatically insert them for you.

NAMING AND STORING PROGRAMS. To name and store a program that has been entered onto level 1 of the stack, press ['] to activate algebraic entry mode (suitable for entering names and expressions), then key in the desired name and press [STO]. The program will be stored in user memory under its name, and pressing [VAR] will show a menu key with an *abbreviated* name (up to 5 characters).

TO RUN A PROGRAM. To run a program, simply press the white menu key beneath the program's abbreviated name. Of course, if the program requires input data for its proper execution then you must first provide that data in an appropriate

way, either on the stack or as stored variables that are named in the program body.

EXAMPLE. The program « DUP SQ SWAP INV + √ INV » takes a number "a" from level 1 as input data and returns the calculated value of $\frac{1}{\sqrt{a^2 + 1/a}}$ to level 1. Key in the program by first pressing [↰] [« »], the other indicated commands. Press [ENTER] to add the closing program delimiters and copy the program to the stack.

Press ['] PGM1 [STO] to name this program PGM1 and store it in user memory under this label. Press [VAR] to see the menu key [PGM1].

Now, run the program using as input data the number 2: key in 2 and press [PGM1]. The answer, .471404520791, will be displayed on level 1. Notice that you did not have to enter the data onto the stack before pressing [PGM1]. This is typical; pressing the menu key [PGM1] automatically entered the data for you. Run the program with some other inputs.

SYNTAX ERRORS. When keying a program into the command line, if an object is accidentally entered in an invalid form, then pressing [ENTER] will cause the calculator to refuse to copy the program onto the stack and display a message indicating a syntax error. To remove the message from the screen so you can correct the syntax, simply press [ON], which cancels the message.

EXAMPLE. Key in : « → ARRY and press [ENTER] . Notice what happens. Now remove the message, delete the space after the → with [DEL], and press [ENTER].

EDITING PROGRAMS. To make any change in the body of an existing program you must *edit* the program.

- If the program is on stack level 1, the [EDIT] key will copy it to the command line where you can then make the required changes. Press [ENTER] to return the corrected version to level 1.

- If the program is not on stack level 1, but stored in user memory under, say, 'NAME' the keystrokes ['] [NAME] [RCL] will recall the program to level 1 and you can then proceed as above.

EXAMPLE. Start this example with the program « PROGRAM MODI » stored in user memory as [TRY1].

(i) Recall it to stack level 1 with ['] [TRY1] [RCL] .

(ii) Copy it to the command line with [EDIT], and change MODI to BODI.

(iii) Copy back to level 1 with [ENTER], then replace the old version with the new one by pressing ['] [TRY1] [STO] .

(iv) Now recall this new version to the stack, copy it to the command line, change BODI to BODY, and then replace the previous version with this newer one.

(v) Finally, check your last work by recalling to level 1, examine the result, then drop it from the stack with [DROP].

SHORTCUTS.

- You can recall to level 1 the contents of any stored variable, say TRY1, by pressing [↱] [TRY1]. Thus, *right shift will recall.*

- Likewise, you can store (or load) an object on level 1 into any stored variable, by pressing [↰] , then the variable's menu key. Thus, *left shift will load.* Try this by loading « + SQ COS » into [TRY1]; now recall the contents.

DELETING. Imagine that you have stored an object under variable PGM1 in your user memory. The object may be any one of the variety of objects recognized by the calculator: a real number, an array, a program, etc. You can delete this object from user memory by *purging* variable PGM1. Purging a single variable is usually done with the keystrokes ['] [PGM1] [PURGE]. The label disappears from the VAR menu and its contents are removed from user memory. To purge several variables at the same time press [{ }], then the menu key for each variable you wish to purge, and then [ENTER]. Now press [PURGE] to purge the variables in the list.

EXAMPLE. Start by storing the numbers 1, 2 and 3 in variables 'X' 'Y' and 'Z' in user memory. With your VAR menu active, purge 'X' by pressing ['] [X] [PURGE]; watch [X] disappear. Now purge the two remaining variables at the same time by building a list { Y Z } and pressing [PURGE].

ACTIVITY. The following program takes numbers x, y from the stack and returns $(x+y)^2\sqrt{x+y}$.

« + DUP SQ SWAP √ * »

(a) Key in this program and store it under variable "EX.1".

(b) Run the program with inputs 9, 16.

(c) Change the program body by replacing the * with / and adding NEG at the end (in algebraic entry mode press [+/–] to see NEG.

(d) Run the new program with 9, 16.

(e) In terms of x and y, what does the new program calculate?

(f) Purge this program.

(g) Purge programs TRY1 and PGM1 simultaneously.

PROGRAM INDEX

SINGLE-VARIABLE CALCULUS

(Alphabetical, by Topic Area)

MULTIVARIABLE CALCULUS

(Alphabetical Listing)

DIFFERENTIAL EQUATIONS

(Alphabetical Listing)

LINEAR ALGEBRA

(Alphabetical, by Topic Area)

Matrix Editing Routines

Linear Systems

Orthogonality

Eigenvalues and Eigenvectors

ADVANCED ENGINEERING MATHEMATICS

(Alphabetical, by Topic Area)

Differential Equations

PROBABILITY AND STATISTICS

(Alphabetical, by Topic Area)

SUBJECT INDEX

ISBN 0-03-097000-8

90021

9 780030 970009