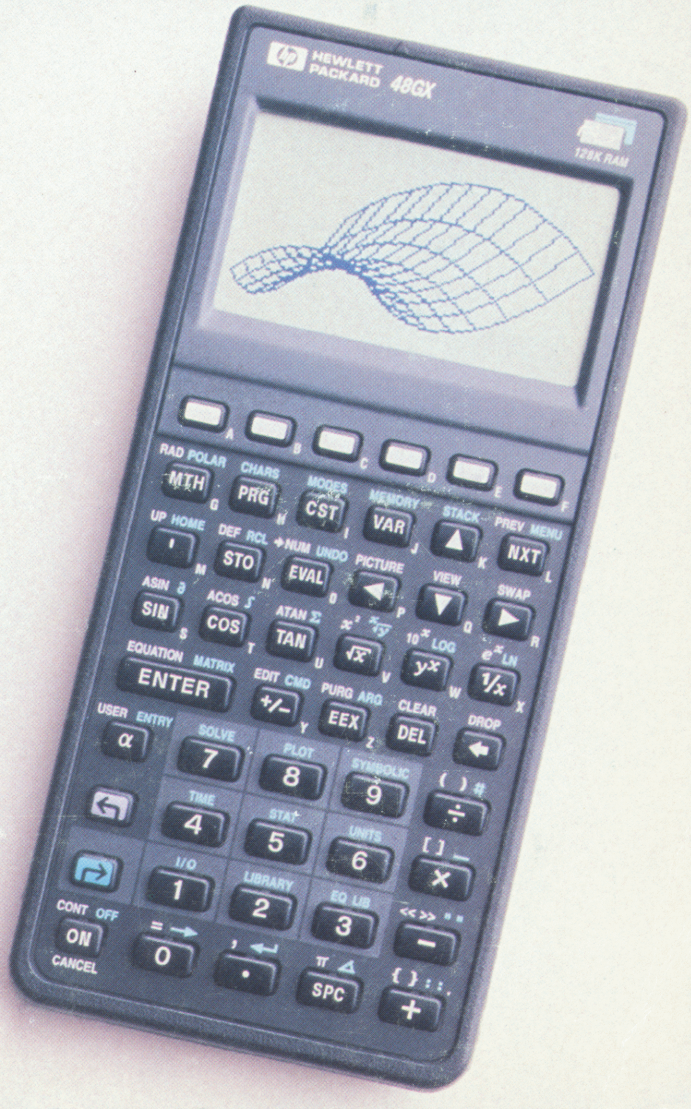
DIFFERENTIAL EQUATIONS USING THE HP-48G/GX

T. GILMER PROCTOR



DIFFERENTIAL EQUATIONS

using the

HP-48G/GX

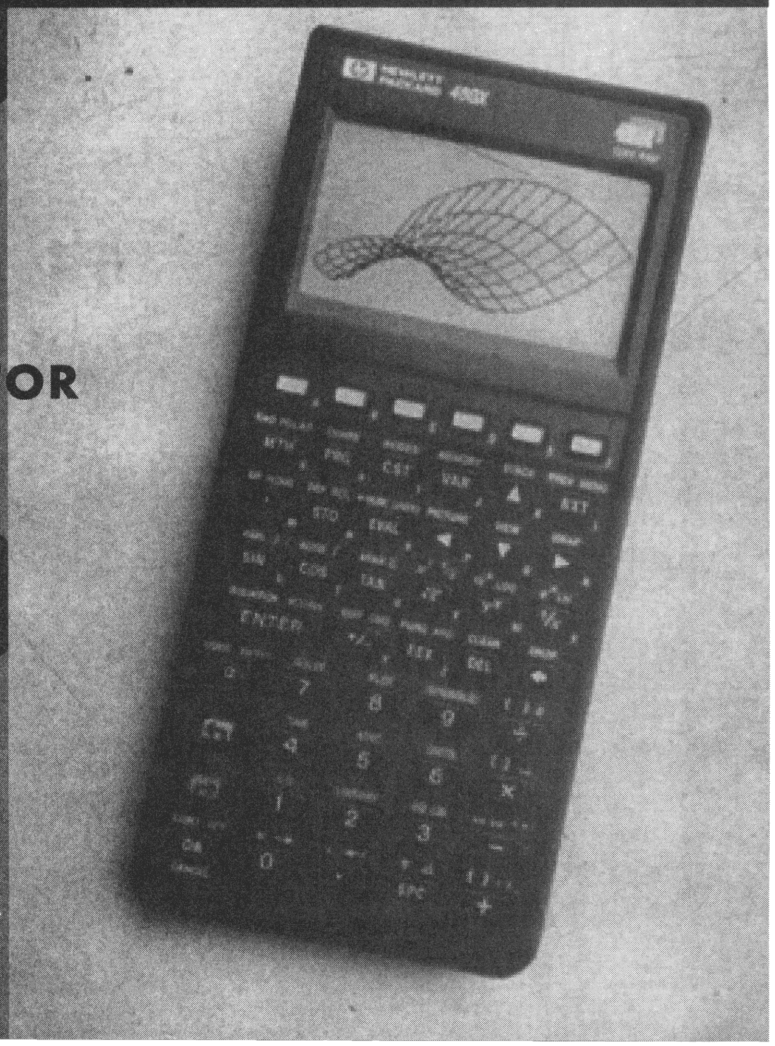
DIFFERENTIAL EQUATIONS

using the

HP-48G/GX

T. GILMER PROCTOR

Clemson University



CHARLES RIVER MEDIA, INC.
Rockland, Massachusetts

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

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

This book is printed on acid-free paper.

Differential Equations using the HP-48G/GX T. Gilmer Proctor

ISBN: 1-886801-19-3

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

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

CONTENTS

INTRODUCTION 1

1 PLOTTING SOLUTIONS FOR DIFFERENTIAL EQUATIONS ON THE HP-48 5

Elementary User Programs 15

2 FIRST ORDER DIFFERENTIAL EQUATIONS 25

Population Problems 32

Motion of a Particle in One Dimension 36

Input Output Problems 41

3 SECOND ORDER DIFFERENTIAL EQUATIONS 50

Second Order Input Output Problems 55

Trajectories in the y_1-y_2 Plane 60

Linear Variational Systems in the y_1-y_2 Plane 66

4 LINEAR SYSTEMS OF DIFFERENTIAL EQUATIONS WITH CONSTANT COEFFICIENTS 76

Homogeneous Systems 76

Non-homogeneous Systems 88

5 MISCELLANEOUS SYSTEMS 95

A Nonlinear Problem: the Lorentz Equations 100

A Nonlinear Problem: Earth, Moon, Satellite Motion 103

Discrete Dynamical Systems 106

Parameter Identification Problems Revisited 112

References 120

APPENDICES

1 User Menu Housekeeping and Organization 121

2 Direction Fields 128

3 Distinguished Oscillations for a Forced Harmonic Oscillator 135

4 Runge-Kutta Adaptive Step Size Algorithm 140

5 Programs for Three Dimensional Trajectories 144

6 Answers to Selected Exercises 148

INDEX 158

PROGRAMS Inside Back Cover

INTRODUCTION

This is a manual of EXERCISES for a graphics calculator to supplement the elementary differential equations course. You will learn early in such a course that important mathematical models for scientific problems often contain differential equations and that particular solutions of these equations describe the behavior of the model. The problem solver often has some intuition concerning how the system should behave and the graphical properties of a single solution or a family of solutions are an important clue to the correctness of the model and provide qualitative properties of the solution. Even if analytical expressions can be obtained for the solutions, their graphs may reveal behavior a scientist may not discover from these expressions.

The HP-48G/GX calculator is a great graphics and computational tool in this course. It can be used in class to illustrate concepts. It can be used for homework in your favorite study area, for example, a library or a dormitory room. The graphs and computations that are created on the calculator can be stored or recreated on a microcomputer. This manual contains only some of the possible uses of this tool and illustrates the material which my students have helped develop in the past five years. Some of the material is taken from [4], but many of the EXERCISES and presentations are new. The presentation does not require that the reader be a good HP 48 programmer since nearly all of the programs are explained within this manual.

One of the distinctive features of the HP-48G/GX is a built-in program for calculating and displaying in the same graph approximate solutions to one or more initial value problems containing differential equations. The HP-48S/SX does not have such a program but we will present user programs which will accomplish the same purposes. To emphasize the statement given above, the capability to easily display solutions of several problems allows us to study how the solutions depend on various parameters and to focus on geometrical characteristics of a system.

The first part of Chapter 1 describes the programs that have been provided (in the HP-48G/GX) for obtaining and plotting approximate solutions of initial value problems. Then we show how to construct programs using elementary algorithms (the Euler and improved Euler algorithms) for obtaining and plotting approximate solutions. These programs can be used on both the HP-48S/SX and HP-48G/GX. They can be adapted for various special purposes such as creating other graphical displays or for use in higher order numerical methods for differential equations such as the Runge-Kutta algorithms.

The second chapter 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 third chapter concerns the solution of two first order differential equations or a single second order equation with initial conditions. Because the HP-48 processes vector and scalar quantities with many of the same commands, it is possible to use some programs for scalar systems as vector programs, including the algorithms for solving initial value problems (Euler, improved Euler, Runge-Kutta, etc.). To solve and plot the solutions of a second order differential equation with initial conditions, we borrow from problems involving particle motion. For displacement x(t) and velocity v(t), of the solution of the pair of equations

$$dx/dt = v,\ dv/dt = g(t, x, v),\quad x(t_0) \text{ and } v(t_0) \text{ given,}$$

the function x(t) satisfies the second order initial value problem

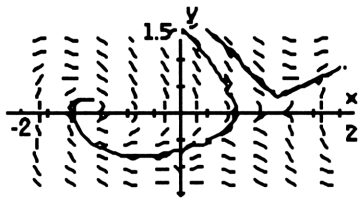
$$\frac{d^2x}{dt^2} = g(t, x, \frac{dx}{dt}),\qquad x(t_0) \text{ and } \frac{dx}{dt}(t_0) \text{ given,}$$

and conversely, if we have a solution x(t) of the second order initial value problem, the pair x(t), dx/dt(t) = v(t) satisfies the pair of first order differential equations. Thus, to solve the second order initial value problem we make a vector $[Y_1, Y_2\ ] = [x, v]$ and solve the vector system

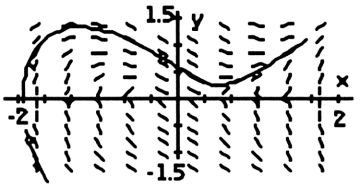
$$\frac{dY}{dt} = \begin{bmatrix} \frac{dx}{dt} \\ \\ \frac{dv}{dt} \end{bmatrix} = F(t, Y) = \begin{bmatrix} v \\ \\ g(t, x, v) \end{bmatrix} = \begin{bmatrix} Y_2 \\ \\ g(t, Y_1, Y_2) \end{bmatrix}$$

with $Y(t_0)$ = column $[x(t_0), v(t_0)]$ as given using the same algorithms as for a first order scalar problem. In chapter three, EXERCISES are provided to study in particular the solutions of the second order differential equations encountered in linear and nonlinear models of mechanical springs and electrical circuits.

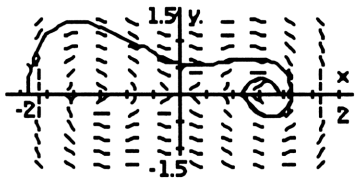
The fourth chapter contains programs that construct solutions of linear vector systems of differential equations of the form dy/dt = A y + f(t). The fifth chapter contains several problems that extend the earlier material and result in three or more differential equations. Finally, Appendix 2 contains a set of programs that can be used to sketch the direction field for a pair of differential equations, and Appendix 4 contains a program that HP-48S owners my use when a more accurate algorithm is needed for computing the solution of a differential equation. This algorithm is already encoded into the HP-48G calculators.



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



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



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

Assorted Direction Fields with Solution Overlays

Many topics traditionally associated with an introductory course in differential equations are not included; among these delay differential equations, and control problems. Problems in these particular areas are presented in [1].

Many students use a graphics programmable calculator in more than one course. It is important that you know where the appropriate programs are stored in memory. It is easy to collect a set of programs that you use frequently in a particular course into a directory. See your calculator Owners Manual or Appendix 1. We strongly suggest that the reader create a subdirectory for the programs that use the built-in algorithms for solving initial value problems contained in Chapter 1, and a subdirectory that includes the elementary user programs also presented in Chapter 1. We will make specific suggestions in appropriate locations within this manual.

This manual would not have been possible without the strong encouragement and help from my editor D.R. LaTorre, without the support from the HP-48 development team (Diana Byrne, Paul McClellan, Charlie Pattan, and William C. Wickes) and without the patience and word-processing skill of Mrs. April Haynes. Thanks.

1 PLOTTING SOLUTIONS FOR DIFFERENTIAL EQUATIONS ON THE HP-48

Suppose we wish to plot an approximate solution of an initial value problem

$$\frac{dy}{dt} = F(t, y), \quad y(t_0) = y_0$$

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

- A program that gives the value of F when t and y are specified.
- The initial quantities t_0, y_0 and criteria for completion (e. g. the final value of t_f).
- The plot window must be specified and the plot screen may have to be erased.
- It may also be necessary to specify an appropriate algorithm for computing the approximate solution and any necessary inputs to the algorithm such as a global error tolerance and a starting value of the step size.
- The command to draw.

There are three methods on the HP-48G/GX to provide these inputs and obtain a plot of an approximate solution. The built-in method prompts the user for the necessary information with input forms and choose boxes. Alternatively, we can

construct a set of programs that take or generate some of the required inputs from the stack and then call the basic algorithm to calculate solutions in a plotting program. This alternative method is particularly useful when only one or two of the inputs must be modified or when the stopping criterion is nonstandard. The third method is to construct programs that employ easy algorithms for computing and plotting approximate solutions to initial value problems. This last method can be used on the HP-48S/SX calculators. We will illustrate each of these methods with EXERCISES. The user must make a decision on the appropriate method for the other EXERCISES. In this section we begin with the built-in method, then pass on to the second alternative. The third method will be featured in a separate section of this chapter.

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**.

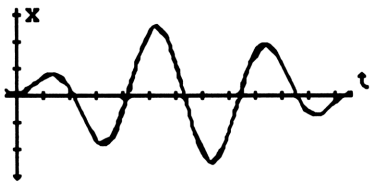
EXERCISE 1.1: Construct a plot of the solution of y ' + 3y = cos t, y(0) = .3 for 0 ≤ t ≤ 6.283 using the calculator's differential equation plot feature. Enter **COS(T) - 3*Y** in the F field of the input form. (Note that the calculator automatically places this function within ' marks.) Make sure T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

After completing this EXERCISE, you should check to see that the program which calculates the values of F(T,Y) is stored in the variable **EQ**. To get some confidence in the calculator solution, we can plot the exact solution y = .3 cos(t) + .1 sin(t). Press **ON** to return to the **PLOT** application, change the **TYPE** to **Function**, and enter **.3*COS(T) + .1*SIN(T)** as **EQ**. (Again the calculator places ' marks around the function.) In this case we want to overlay the new plot over the old one so do not ERASE. Press **DRAW**. Note the good agreement between the approximate solution and the exact solution.

EXERCISE 1.2: Construct a solution of y ' = sin (ty), y(0) = 2 for 0 ≤ t ≤ 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 ≤ t ≤ 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 **T, SOLN** name **W** and the **INIT** vector [0 1] for **SOLN**. If we want a graph of W(1) versus T we choose **INDEP** for the **H-VAR** and **SOLN(1)** for the **V-VAR** on the **OPTS** page. **V-VIEW** should be set at -.8 .8. (If we want a W(1) versus W(2) plot we choose **SOLN(1)** for **H-VAR** and **SOLN(2)** for **V-VAR** on the **OPTS** page.)



$x'' + 49x = 12\cos(5t),\ x(0) = x'(0) = 0$

EXERCISE 1.4: The figure shown above is a plot 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 individual or multiple steps in either of two basic algorithms for approximating the solution to a differential equation. These programs can be embedded in **user** programs to produce variations of the basic program described above. The advantages gained by this process include some speedup when most parameters are already set and any modifications of the basic problem not treated easily by the first method. For example, the final time t_f may be "when some condition is satisfied" rather than a simple number which is known beforehand.

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

Program Name:	**IN.FN**
Purpose:	The user supplies program **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 responses might be **<< '-T*Y' EVAL >>** (or the reverse Polish notation program **<< T Y * NEG >>**) for the function F(T,Y) = - T*Y or

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

for the function F(T,Y) = 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

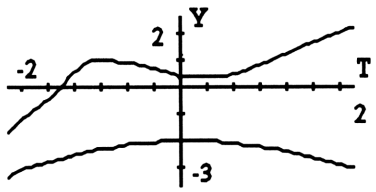
No input stack is required. The appropriate response for the first query is a pair of numbers, H-min and H-max. The response for the second query is a pair of numbers V-min and V-max.

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 scalar differential equation. The TY designation indicates that the plot is a (T, Y) plot.

Program Name:	**G1.TY**		
Purpose:	Generate a T Y plot of the solution to T_f		
Stored Quantities:	**XRNG YRNG FN TOL HS**		
Input	level 3	level 2	level 1
	T_0	Y_0	T_f

The output stack is empty, the variables T Y contain updated values.

```
<< { # 0d # 0d } PVIEW DRAX 3 ROLLD 'Y' STO 'T' STO
  → TF << { T Y FN } TOL HS T Y R→C 4 ROLLD DO
 RKFSTEP T Y R→C DUP 6 ROLLD 5 ROLL LINE DUP T +
 TF UNTIL > END DROP TF T - RKFSTEP T Y R→C DUP 6
 ROLLD 5 ROLL LINE DROP TF T - RKFSTEP T Y R→C 5
        ROLL LINE 3 DROPN >> PICTURE >>
```

Notes: Typical numbers for **HS** and **TOL** are .005 and .00005 and are to be stored before execution of this program. If the user wants other names for the variables other than **T Y FN,** such changes can be made by substituting for **{ T Y FN}, T,** and **Y,** the desired alternate notation. The command **RKFSTEP** invokes the built-in Runge-Kutta-Feldberg program for one step.

EXERCISE 1.1a: Construct a plot 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) - 3*Y' EVAL >>** and press **ENTER**. Execute **IN.PP**, respond by typing 0 6.283 and **ENTER**, respond by typing -.5 .5 and press **ENTER**. Put 0 .3 6.283 on the stack and execute **G1.TY**. As in EXERCISE 1.1, plot the exact solution y = .3 cos(t) + .1 sin(t). Press ↱ **PLOT**, change the **TYPE** to **Function**, and enter **.3*COS(T) + .1*SIN(T)** as **EQ**. In this case we want to overlay the new plot on the old one so do not ERASE. Now press **DRAW**. Note the good agreement between the approximate solution and the exact solution.

EXERCISE 1.2a: Construct and plot 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(I)) for a vector system Y' = F(T,Y). We will call the program **G.0I** where the 0 represents the T variable and the I represents the Y(I) variable. The modification consists of changes made to **G1.TY** in four locations in which the **Y** number in **G1.TY** is changed to **'Y(I)' EVAL** and adding the first **<<** → **I** and the last **>>**. The user can avoid retyping the whole program by pressing **'G1.TY' RCL, EDIT**, typing the corrections and additions pressing **ENTER**, then **'G.0I' STO.**

Program Name:	**G.0I**			
Purpose:	Generate a T Y(1) plot of the solution to T_f			
Stored Quantities:	**XRNG YRNG FN TOL HS**			
Input	level 4	level 3	level 2	level 1
	T_0	vector Y_0	T_f	I

The output stack is empty, the variables T and Y contain updated values.

```
<< → I << { # 0d # 0d } PVIEW DRAX 3 ROLLD 'Y'
STO 'T' STO → TF << { T Y FN } TOL HS T 'Y(I)' EVAL
R→C 4 ROLLD DO RKFSTEP T 'Y(I)' EVAL R→C DUP 6
ROLLD 5 ROLL LINE DUP T + TF UNTIL > END DROP TF T
- RKFSTEP T 'Y(I)' EVAL R→C DUP 6 ROLLD 5 ROLL LINE
DROP TF T - RKFSTEP T 'Y(I)' EVAL R→C 5 ROLL
LINE 3 DROPN >> PICTURE >> >>
```

EXERCISE 1.3a: Construct a composite (t, x) plot 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], 6.283 and 1 on the stack and execute **G.0I**. We change **R** to each of the numbers 2 and 2.5 and place input quantities 0, [1 0], 6.283 and 1 on the stack and execute **G.0I** twice more to overlay plots of the other two solutions.

A similar change to **G.0I** gives the plot program **G.12** in which the component Y(1) of the solution is plotted against the component Y(2). The change is made in four places and commands **T 'Y(1)' EVAL** are changed to **'Y(1)' EVAL 'Y(2)' EVAL** and by removing the first **<< →** I and the last **>>**.

Program Name:	**G.12**		
Purpose:	Generate a (Y(1), Y(2)) plot of the solution from T_0 to T_f		
Stored Quantities:	**XRNG YRNG FN TOL HS**		
Input:	level 3	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 DROP TF T - RKFSTEP 'Y(1)' EVAL 'Y(2)' EVAL R→C
DUP 6 ROLLD 5 ROLL LINE DROP TF T - RKFSTEP 'Y(1)'
EVAL 'Y(2)' EVAL R→C 5 ROLL LINE 3 DROPN >>
PICTURE >>
```

For consistency, from this point we will use notation as follows: for first order differential equations, Y will be the dependent variable and T will be the independent variable. For higher order differential equations, x will be the dependent variable, t will be the independent variable and we will reserve Y as a vector with components which may be constructed from the x, x', etc. variables.

EXERCISE 1.5: Construct an x vs x' plot 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 **T, SOLN** name **Y** and the **INIT** vector [0 1] for **SOLN**. We choose **SOLN(1)** for **H-VAR** and **SOLN(2)** for **V-VAR** on the **OPTS** page. **HVIEW** should be set at -1 1 and **V-VIEW** should be set at -1 1. **ERASE** and **DRAW**. This approximate solution of the differential equation can be compared to the exact solution by overlaying the parametric curve

$$e^{-.25t}\ 1.0328\ (\sin(.9862t)+i*(.9862\ \cos(.9862t) - .25\sin(.9862t))).$$

on the same picture. (Use **Parametric** type in the **PLOT** environment.) The user should notice that the plot of the approximate solution consists of a set of points

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

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.0I, G1.TY, ER.SE, IN.FN, IN.PP, T, Y, FN, TOL, HS, EQ,** and

PPAR in this order. The program **ER.SE** given by **<< ERASE >>** is placed in this directory for convenience. The subdirectory can be created by placing the name '**DE.1**' on the stack, then pressing ◀ **MEMORY**, pressing **DIR**, then **CRDIR**. To obtain the desired order, press { and enter the program names in order, press **ENTER**, then **ORDER** (located in the same **MEMORY DIR** menu).

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 1.6: 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

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

(Again y_k is the approximation of $y(t_k)$ and t_0, y_0 and h are given so the algorithm may be initiated.) This program is named **IULER** and takes t y from the stack and gives (t+h) (y+h*[f(t,y)+f(t+h,y+h*f(t,y)]/2). Note **EULER** is part of this program.

Program Name:	**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 HP-48G calculator. Press [↱] **SOLVE**, use the [∇] arrow key to select **Solve Diff eq...**, press **OK**, enter the F function **Y**, set the range of the independent variable to 0 1 and set the initial value of the solution to 1. Move the cursor to **FINAL** and press **SOLVE**. Press the **ON** key and you should see the value 2.718... on the stack. Put 1 on the stack, press the e^x key and subtract to see the apparent error -.000019... . This error was achieved with the default tolerance .0001. The performance of the differential equation algorithm depends on the problem and is not always this good.

A comment on the built-in algorithm on the HP-48G calculator for solving differential equations is in order at this point. There is a default program based on the well known Runge-Kutta Feldberg algorithm which automatically selects step

size to keep the perceived error below the specified tolerance. There is also a second built-in calculator program for solving stiff differential equations that we will discuss briefly later.

EXERCISE 1.7: Try **EULER** on the problem $y' = (y^2 + y)/t$, $y(1) = 1$ with $h = .2$. Execute 5 times, then reduce h to .1 and execute 10 times. Next execute from the initial value 20 times with **H** = .05. What is being indicated? Hint: this problem can be solved exactly and has an asymptote at t = 2. Here **F.N** could be given by

```
<< → T Y ' (Y^2+Y)/T ' >>
```

EXERCISE 1.7a: Try the HP-48G calculator's **Solve diff eq...** algorithm for the F function and initial condition given in EXERCISE 1.7 for the value of the solution at t = 2. Change the tolerance **TOL** to .1 and try to **SOLVE** for **FINAL.** The calculator will take over 10 seconds and returns a value of 1743.5... . If you change the value of **TOL** to .05 and resolve for **FINAL**, the calculator will take over 20 seconds and returns a value of 2187.8... . The long execution time tells us that the calculator is struggling to achieve good results and in this case can not achieve accuracy for good reason.

Programs to obtain graphical output are easy on the HP-48. The following program, which we will call **GRAF,** requires t_0, y_0 from the stack and uses **IULER** (or **EULER**) to advance **N** steps of size **H**. (**N** is also stored.) The user should pre-enter the numbers t_{min} t_{max} as **XRNG** and numbers y_{min} y_{max} as **YRNG** for the graph.

Program Name:	**GRAF**	for scalar equation	
Purpose:	Plot N values of (x,y) obtained using Euler algorithm		
Stored quantities:	**N, H, F.N, IULER XRNG YRNG**		
Input		Output	
level 2	level 1	level 2	level 1
t_0	y_0	t_N	y_N
		and graph with cursor	

<< { # 0d # 0d } PVIEW DRAX DUP2 R→C 3 ROLLD 1 N START IULER DUP2 R→C DUP 5 ROLLD 4 ROLL LINE NEXT PICTURE >>

GRAF contains a loop in which N new points (t,y) are calculated and plotted. **You may want to ERASE the graphics screen before executing the program.** The program **EULER** may be inserted in place of **IULER** so that **GRAF** uses whichever algorithm is desired. Notice also that the last values of t and y remain on the stack after **GRAF** is executed. To restore the stack screen, press **ON**.

As a footnote to this section, the following program can be used to remind the user for the ingredients required for **GRAF**. As written, the user must enter an expression for f(T, Y) (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→ 'F.N(T,Y)' SWAP =
  DEFINE "KEY IN # OF STEPS" " " INPUT OBJ→ 'N' STO
  "KEY IN STEP SIZE" " " INPUT OBJ→ 'H' STO "KEY IN
  XRNG" " " INPUT OBJ→ XRNG "KEY IN YRNG" " "
INPUT OBJ→ YRNG "KEY IN INITIAL T" " " INPUT OBJ→
  "KEY IN INITIAL Y" " " INPUT OBJ→ ERASE >>
```

As we have already indicated, it is often desirable to plot solutions of several initial value problems on the same plot. Of course, plots can be combined simply by not erasing the previous result.

EXERCISE 1.8: Consider the following differential equation together with several initial conditions and plot the solutions on the same graph.

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

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

F.N: `<< → T Y ' Y*(1-Y) >>`.

Put 0 and .2 on the stack, then execute **GRAF**. (Remember **H** = .05 and **N** = 100 are stored before execution.) Place another initial condition on the stack and add the second solution graph. Notice the solution y = 1 is an attracting solution, i. e., nearby solutions collapse to y = 1 as time increases.

The **EULER** and **IULER** programs also work for the vector case when the **F.N** program has the proper form and when the initial y input is a vector. Here again **F.N** requires input T Y. We can modify the **GRAF** program to the following form:

Program Name:	**G.TYI**
Purpose:	Plot N values of (T,Y(I)) resulting from the improved Euler algorithm which creates a sequence of N values of t, and Y.
Stored Quantities:	**N H F.N EULER IULER XRNG YRNG**

Input			Output	
level 2	level 1	level 3	level 2	level 1
t_0	initial vector Y	I	t_n	final vector Y & graph

```
<< → I << { # 0d # 0d } PVIEW DRAX DUP2 I GET R→C 3
ROLLD 1 N START IULER DUP2 I GET R→C DUP 5 ROLLD
4 ROLL LINE NEXT GRAPH >> >>
```

Program **INIT.I** is to set the plotting parameters for **G.TYI**. Notice that the construction of the **F.N** program is to be done later since it is felt that **F.N** could be a complicated program.

Program Name: **INIT.I**

Purpose: Set parameters for **G.TYI**.

```
<< "KEY IN # OF STEPS" "" INPUT OBJ→ 'N' STO
"KEY IN STEP SIZE" "" INPUT OBJ→ 'H' STO
"KEY IN XRNG" "" INPUT OBJ→ XRNG
"KEY IN YRNG" "" INPUT OBJ→ YRNG
"KEY IN INITIAL T" "" INPUT OBJ→
"KEY IN INITIAL Y" "" INPUT OBJ→
"KEY I FOR Y(I) GRAPH" "" INPUT OBJ→
"PRESS ENTER, CONSTRUCT PROGRAM FN
EXECUTE G.TYI" "" INPUT OBJ→ >>
```

The program **G.TYI** as given above works for vector initial value problems with two or more components. For example consider the following

EXERCISE 1.9: Set F.N to be

```
<< → T Y << ' -.00001*Y(1)*Y(2)' EVAL '.00001*Y(1)*Y(2) - Y(2)/14' EVAL
'Y(2)/14' EVAL 3 →ARRY >> >>
```

set initial **T, Y** to 0 [45400 2100 2400] with **XRNG** 0 25 **YRNG** 0 45400 and set the number of steps to be N = 25 and step size to H = 1. Obtain a T-Y(1) plot, overlay a T-Y(2) plot, etc.

Students who use the Euler and improved Euler algorithm will also need a program to compute the solution and plot the components $y_1(t)$ versus $y_2(t)$ as t increases. Such a program, call **G.Y12** is presented below. The initialization program **INIT.I** also works for this program, except the input I is not needed and should be deleted before execution of **G.Y12**.

Program Name:	**G.Y12**
Purpose:	Plot N values of (Y(1),Y(2)) resulting from the improved Euler algorithm which creates a sequence of N values of Y(1) and Y(2).
Stored Quantities:	**N H F.N EULER IULER XRNG YRNG**

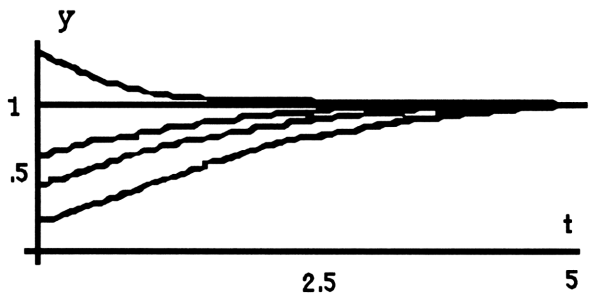
Input		Output	
level 2	level 1	level 2	level 1
t_0	initial vector Y	t_n	last vector Y & graph

<< { # 0d # 0d } PVIEW DRAX DUP DUP 1 GET SWAP 2 GET R→C 3 ROLLD 1 N START IULER DUP DUP 1 GET SWAP 2 GET R→C DUP 5 ROLLD 4 ROLL LINE NEXT GRAPH >>

We recommend that the user create a subdirectory for the programs in this section. A possible subdirectory name is **DE.2**. This subdirectory should contain the programs **INIT1**, **GRAF**, **INIT.I**, **G.TYI**, **ER.SE**, **G.Y12**, **T**, **Y**, **F.N**, **EQ**, and **PPAR** in this order. The program **ER.SE** given by **<< ERASE >>** is placed in this directory for convenience. The subdirectory can be created by placing the name '**DE.2**' on the stack, then pressing ↰ **MEMORY**, pressing **DIR**, then **CRDIR**. To obtain the desired order, press { and enter the program names in order, press **ENTER**, then **ORDER** (located in the same **MEMORY DIR** menu).

2 FIRST ORDER DIFFERENTIAL EQUATIONS

Now that we can construct approximate solutions of a differential equation we can suggest EXERCISES and activities that use these graphical and numerical computations to enhance the study of differential equations.



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

We will often be interested in constructing graphs of several solutions of a differential equation. The figure shown above, constructed for the differential equation y' = y(1–y) is an example. There may be solutions y(t) that remain constant as time increases. Such solutions are called *equilibrium solutions*. In the case just mentioned, the constant solutions are y(t) = 0 and y(t) = 1. Clearly the solutions $y(t) = y_e$ of dy/dt = F(t, y), which are constant, satisfy

$$F(t, y_e) = 0.$$

If we wish to understand how solutions of a differential equation change as the initial condition y(0) is varied, one of the first tasks is to find the equilibrium solutions. If the function F(t, y) is continuous and has continuous derivatives then

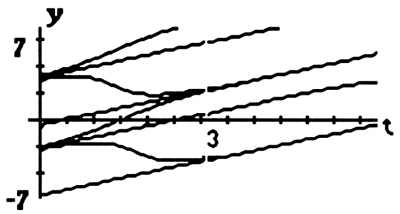
any solutions $y_1(t)$ and $y_2(t)$ which satisfy different initial conditions do not intersect. Consequently, constant solutions restrict the region where nearby solutions can proceed.

EXERCISE 2.1: Plot the solutions of the three initial value problems $dy/dt = y^2\,(1 - y^2)$, that satisfy either $y(0) = -1$, $y(0) = 0$, and $y(0) = 1$ for $0 \le t \le 5$ all in the same picture. Then overlay plots of the solutions of the same differential equation that satisfy $y(0) = -.25$ and $y(0) = .25$.

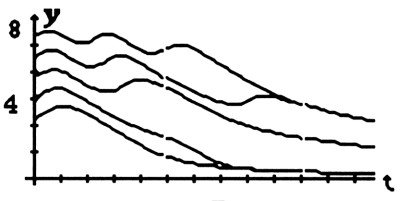
EXERCISE 2.2: Consider the differential equation $y' = y - y^3$. The equilibrium solutions are 0, -1 and 1. For an initial condition $y(0)$ not in the set $\{-1, 0, 1\}$, use the separation of variables technique to obtain the result

$$y^2\,|1 - y| = |1 + y|\,K e^{2t}, \quad K = \frac{y_0^2\,|1 - y_0|}{|1 + y_0|}.$$

This equation defines the solution y implicitly as a function of t. For example, if $0 < y_0 < 1$, then since $y(t)$ will not leave the interval $0 < y < 1$, we have a cubic equation for y as a function of t. We can use one of our calculator programs to construct a plot of an approximate solution of the problem with such an initial condition or if we only want a crude idea of the solution graph, we can sketch in an increasing function proceeding from $y(0)$ up toward the asymptotic value $y(\infty) = 1$. Why?



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 $a = -6.81$. (Hint: make the transformation $t - y = w$ to get the differential equation $w' = 1 - 2\sin w$. What are the equilibrium solutions of the w differential equation ?) Solutions starting near $t = 0$, $y = -.524$ collapse to the straight line solution $y = t - .524$, while solutions starting near $t = 0$, $y = -2.618$ are repelled away for the straight line solution $y = t - 2.618$, etc.

As for the second plot shown just above, even though the functions $y = (2n+1)\pi/t$ ($n = 0, \pm 1, \pm 2, \ldots$) 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. Moreover when the initial value y(0) is large, there are more values of t on the graph when $.5ty(t) = (4n+1)\pi/2$ and the slope y' is zero.

EXERCISE 2.3: 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 2.4: 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 $y(0) = 3.5$ and add the new trajectory. Now choose $y(0) = 1.5$, get the new combination graph, then choose $y(0) = 1$ and get another combination graph: we see the bottom two trajectories approach each other.

The graphical study of solutions of $dy/dt = \sin(ty)$ led to an journal article that gives mathematical proofs for some of the interesting behavior observed in the graphs. See Mills, B. Weisfeiler and A. Krall, "Discovering Theorems with a Computer", *The American Mathematical Monthly*, volume 86 (1979), pages 733-739.

EXERCISE 2.5: Consider the differential equation $y' = y - .3t - (y - .3t)^3$. The transformation $w = y - .3t$ gives the new differential equation $w' = w - w^3 - .3$. What are the equilibrium solutions of the new differential equation ? Sketch several solutions of the w differential equation including the equilibrium solutions, a solution with w(0) above the largest equilibrium solution, one with w(0) below all of the equilibrium solutions and one with w(0) near but not on the middle equilibrium solution. Now make a sketch of the corresponding solutions of the y differential equation.

EXERCISE 2.6: Repeat as much as possible of the previous EXERCISE for the differential equation $y' = y - .5t - (y - .5t)^3$.

We will wish to compare the graphs of solutions of different differential equations. For example if we graph the solutions of the two initial value problems $dy/dt = y(1 - y)$, $y(0) = .25$ and $dy/dt = y^2(1 - y^2)$, $y(0) = .25$ (graphic screen parameters $0 \le$ "t" ≤ 5 and $0 \le y \le 1.2$) on the same plot, we notice that the solutions are structurally similar. In which case is a change of concavity apparent ?

EXERCISE 2.7: Plot the solutions of the two initial value problems $dy/dt = y(1 - y)$, $y(0) = .25$ and $dy/dt = -y \ln y$, $y(0) = .25$ (plot screen parameters $0 \le$ "t" ≤ 5 and $0 \le y \le 1.2$) on the same plot. In what ways are the solution graphs similar ? In what way are they different ?

There are many problems in a differential equations course in which a number (or numbers) satisfying a somewhat complicated equation is needed. In one type of example we can use the graphing capability of the calculator to display the inverse of a particular function and thus graph a desired solution. We will give an example of this below. In a second type of problem we may simply use the equation solver routine contained in the calculator. An example of this type of problem is also given below.

Implicitly defined solutions may arise in the study of first order differential equations, particularly in those problems in which variables are "separated and integrated" or in exact equations.

EXAMPLE: Plot the solution of

$$dx/dt = 1 - x^{3/2}, \; x(0) = 1/2$$

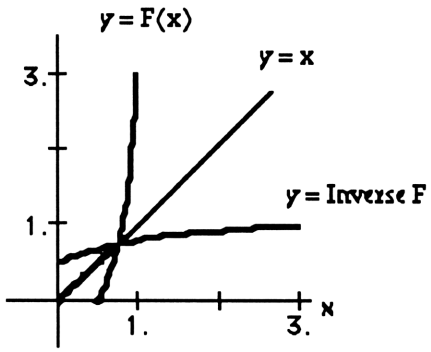
for $0 \le t \le 2.5$. Clearly the solution $x(t)$ will approach 1 as t increases. Using separation of variables, we obtain

$$\int \frac{dx}{1 - x^{3/2}} = t + C.$$

We make the substitution $x = y^2$ and use a partial fraction decomposition for the fraction to obtain the implicit equation $F(x) = t$ where $F(x) = f(x) - f(.5)$ and

$$1.5\, f(x) = \ln \left\{ \frac{\sqrt{(1 + x + \sqrt{x})}}{1 - \sqrt{x}} \right\} - \sqrt{3}\, \text{Arctan} \left\{ \frac{1 + 2\sqrt{x}}{\sqrt{3}} \right\}.$$

We enter the formula for F(x) in the calculator and notice the range of F for $.5 \le x \le .99$ is [0, 2.79]. We specify plotting parameters so that XRNG and YRNG are 0 3 and restrict the values of x to be graphed by entering {X .5 .99} as INDEP and draw a plot of F(x). A plot of the inverse function can be overdrawn by altering EQ to 'F(X) +i*X' and changing the plot type to PARAM. Finally the graph of y = x is also shown as part of the construction of F^{-1} from F.



EXERCISE 2.8: Determine the solution of $x' = 1 - x^r$, $x(0) = .5$ using separation of variables technique for $r = 5/4$ and for $r = 5/2$. Then use the inverse function to plot x(t). Hint: $(x^2+.5(1-\sqrt{5})x+1)(x^2+.5(1+\sqrt{5})x+1) = (x^4+x^3+x^2+x+1)$.

Suppose we wish a number x so that an equation f(x) = g(x) is satisfied. Construct a list which contains expressions for f(x) and for g(x). Then store this list in the variable **EQ**. Set plot parameters so that when both sides of the equation are drawn, a crossing is shown. Use the cursor to locate the approximate crossing coordinates and the **ISECT** command to obtain the result.

MIXING PROBLEM: Initially a large tank holds 2000 gallons of pure water. An stream of 5 gallons per minute with salt content of 2 #/gallon is input into the tank and 4 gallons per minute of the well mixed solution is drained from the tank. When is there Q_0 pounds present in tank ? The usual model dQ/dt = input rate - output rate gives

$$Q = 2\left[\, 2000 + t - \frac{(2000)^5}{(2000 + t)^4}\right].$$

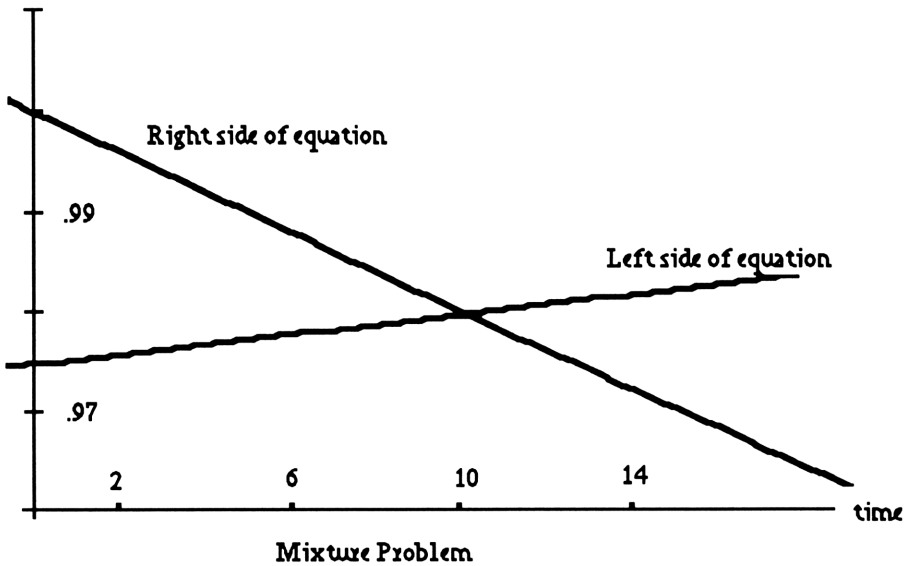
Putting $Q(t) = Q_0$ gives

$$\frac{2000 - \frac{Q_0}{2} + t}{2000} = \left[\frac{2000}{2000 + t}\right]^4$$

to solve for t. For $Q_0 = 100$, we get

$$\frac{1950 + t}{2000} = \left[\frac{2000}{2000 + t}\right]^4;$$

and if we use plotting parameters to show $0 \le x \le 20$, $.9 \le y \le 1$, we get an intersection at about 10 as shown:



Mixture Problem

EXERCISE 2.9: A tank initially contains 300 gallons of pure water. Brine containing 1.5# of salt per gallon enters the tank at 2 gallons/minute and the well mixed solution leaves at 3 gallons per minute. When will the tank contain 21 # of salt? (There may be more than one solution.)

Population Problems

EXERCISE 2.7 gives two initial value problems that model population growth in a food limited environment. Which model is appropriate? Some input from biologists or some observation data could be used to answer this question. Suppose that from experimental data, we can determine the limiting value of the population and that we can also estimate at what fraction of the limiting value of y an inflection point occurs. In EXERCISE 2.7, the inflection points occur at 36.8% (for the logarithm model) and 50% (for the quadratic model) of the limiting value of y, which in both cases is y = 1. We will further explore this question below.

Suppose we are given the assignment of explaining how the population of a species evolves in time and we note that the environment will only support a finite number of the population. Two much studied models of this type are:

- The logistic model:

$$\frac{dp}{dt} = ap - bp^2, \quad p(0) = p_0 : \quad p = \frac{ap_0}{bp_0 + (a - bp_0)\, e^{-at}}$$

- The Gompertz model:

$$\frac{dp}{dt} = p(A - B \ln p), \quad p(0) = p_0 : \quad p = e^{A/B}\left[\frac{p_0}{e^{A/B}}\right]^{\exp(-Bt)}.$$

The parameters have different meanings: equating the carrying capacity of the model (i. e., the value of the population that is reached in infinite time) gives $e^{A/B}$

in the Gompertz model and a/b in the logistic model. Which of these models is better? Or should we look for another model?

These are not easy questions in general. Probably the first step is to pick a model, use data to determine what the model parameters should be (e.g. the constants a, b or A, B) and graph the solution. Then change the model, use data to determine that model's parameters and graph the solution again, etc.

POPULATION PROBLEM: Suppose we use a logistic population model $dp/dt = ap - bp^2$ with parameters a, b and data taken from the following table:

Year	*Population*	*Year*	*Population*
1790	3.93	1900	75.99
1800	5.31	1910	91.97
1810	7.24	1920	105.71
1820	9.64	1930	122.78
1830	12.87	1940	131.67
1840	17.07	1950	151.33
1850	23.19	1960	179.32
1860	31.44	1970	203.21
1870	39.83	1980	226.50
1880	50.16	1990	248.71
1890	62.95		

To determine the parameters a and b: if we use $p_0 = 3.93$ and $p(50) = 17.07$ we get

$$bp_0 = \frac{a\left(\frac{3.93}{17.07} - e^{-50a}\right)}{1 - e^{-50a}} \; ;$$

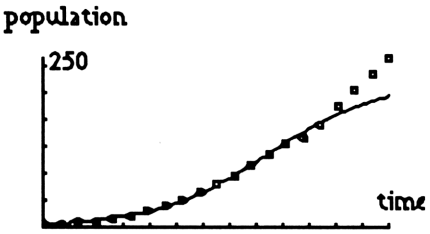
and if we also take $p = 75.99$ at $t = 110$, we have

$$\left[\frac{3.93}{17.07} - e^{-50a}\right]\left(1 - e^{-110a}\right) = \left[\frac{3.93}{75.99} - e^{-110a}\right]\left(1 - e^{-50}\right).$$

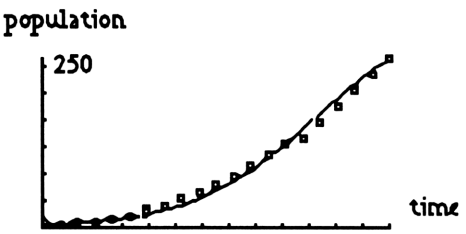
By setting x = 50a (which gives 110a = 2.2x) we obtain the equation

$$[\, e^{-x} - .23023\,]\,(1 - e^{-2.2x}) = [\, e^{-2.2x} - .05172\,]\,(1 - e^{-x}).$$

For plotting parameters $1 \le x \le 2$, $-0.05 \le y \le .02$, we observe a solution at x = 1.53, which means that a = .031 and b = .00014. We show below the data and the solution curve for these values of the parameters, and also the solution curve for the values of a and b obtained by fitting to data at time t = 140 and t = 200 (a = .0279 and b = .0000855).



Data and Logistic Curve for fit to p(50)=17.07, p(110) = 75.99



Data and Logistic Curve for fit to p(140)=122.78, p(200)=248.71

Note: A program **D.GRF** to produce a graph of data with "fat pixels" will be suggested. A list of the data coordinate pairs is stored in a variable **L1**. After **XRNG** and **YRNG** are set and the screen is erased the following program will plot the data points:

```
PGM D.GRF    << PPAR DUP 2 GET SWAP 1 GET - C→R
64 / SWAP 132 / MIN 1.2 * → RADIUS
<< DRAX L1 OBJ→ 1 SWAP START RADIUS 0 6.28 ARC NEXT >> >>
```

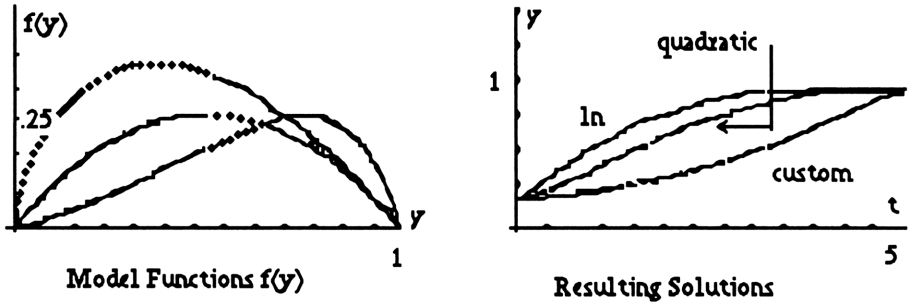
EXERCISE 2.10. Suppose again that p_0 , (t_i, p_i), and (t_k, p_k) are known. Determine the constants A and B in the Gompertz model. (Hint: put s = A/B and solve for e^{-Bt} in the expression for the solution, then for B. Then evaluate the expression at each time and set them equal.) Find the value of A and B for $p_0 = 1$, $(t_i, p_i) = (1, 1.46)$ and $(t_k, p_k) = (2, 1.5)$.

How might other models be constructed ? Here is a suggestion if data $\{(t_1, p_1), (t_2, p_2), \ldots (t_n, p_n)\}$ 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 with inflection at p = .6K, we could try p' = f(p) = $(.6K)^2 - (p - .6K)^2$ for 0 < p ≤ .6K, and also p' = f(p) = 2.25 ($(.4K)^2 - (p - .6K)^2$) for .6K < p ≤ K.

EXERCISE 2.11. Use your calculator to obtain a plot of the solution of this model for K = 1, p(0) = .2, **XRNG** = -.2 5, **YRNG** = -.1 1.1. We will suggest here a method to enter an appropriate F function using the HP-48G input form format. Press **PLOT**, **CHOOS**, **Diff Eq**, press **OK**, then position the highlighted field to **F**. Press **NXT**, press **CALC**, and place the following on the stack: **'IFTE(Y<.6, .36 – (Y–.6)^2, 2.25*(.16– (Y-.6)^2)))'** Note: The command **IFTE** can be located by pressing **PRG BRCH NXT**. The < command is located by pressing **PRG TEST**. When this step is complete, press the **ON** (**CONT**) key, then you should see the desired stack entry and the **ON** key. Press **ON**. The student should complete the EXERCISE from this point. If you wish to invoke the user program, an appropriate FN program might be **<< 'IFTE(Y<.6, .36 – (Y-.6)^2, 2.25*(.16– (Y–.6)^2))' EVAL >>**.

EXERCISE 2.12. Suppose we have the following (time, population) data point measurements {(0, .2), (.5, .37), (1, .61), (1.5, .88), (2, .98), (2.5, 1), (3, 1)}. Use your calculator to plot the data and estimate the location of the inflection point. Then construct a model that will give an inflection point at this value and overlay the solution of the model with the data for comparison.



Logistic, Gompertz and Custom Models (exercise 2.8)

EXERCISE 2.13: Set plot parameters to show $-.5 \le t \le 12.56$, $-.5 \le y \le 4$. Graph the solutions of $y' = .5y(\exp(\sin t) - y)$ with $y(0) = 1$, and $y(0) = 3$. What initial condition gives periodicity ? (This differential equation is a potential model for an environment where the birth rate is periodic in time.)

Motion of a Particle in One Dimension

Mathematical models for the velocity of a particle falling from rest under gravity with air resistance have the form

$$\frac{dv}{dt} = g - f(v), \quad v(0) = 0,$$

where the force exerted on the particle by the resistive medium, **f(v), is determined by experimental means.** We will assume that f is an increasing function with

$f(0) = 0$. The velocity will increase toward a terminal value which is given by $f^{-1}(g)$.

EXERCISE 2.14: For simplicity we take physical units so that $g = 2$. We want to compare the trajectories from different models in which the f(v) functions are given by:

(a) $f(v) = v$

(b) $f(v) = .5\ v^2$

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

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

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 **XRNG** = -.1 2 and **YRNG** = -.1 2.1. Accumulate these graphs on the same picture and label the graphs. Then use a differential equation plotting program to graph the solution of the initial value problem given above for each f(v) function for $0 \le t \le 5$. Accumulate them in the same picture for comparison. Again label the solutions. Use the **YRNG** as above and **XRNG** -.2 5.

A particle falls or is projected from a great height and observations are made on v for, 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 2.15. The data to be used for model selection is:

$$\{ (0, 0), (1, 1.44), (2,1.87), (3, 1.97), (4, 1.99), (5, 2) \}.$$

Consider models of the form $dv/dt = 2 - a\ v^r$, $v(0) = 0$, 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 **G1.TY** can be modified to plot the log $(v_\infty - v(t))$ by inserting the commands **V_∞ - ABS LN** in 4 locations each following the command **Y**.)

Consider the problem

$$M'(t)v + M(t)v' = T(t) - M(t)g - \sigma v, \quad v(0) = x(0) = 0$$

where M(t) is the mass of a rocket and has equation

$$M(t) = m - \alpha t \text{ for } 0 \le t \le t_0 \text{ and } M(t) = m_0 + \{m - \alpha t_0 - m_0\}e^{-\gamma(t-t_0)} \text{ for } t_0 < t$$

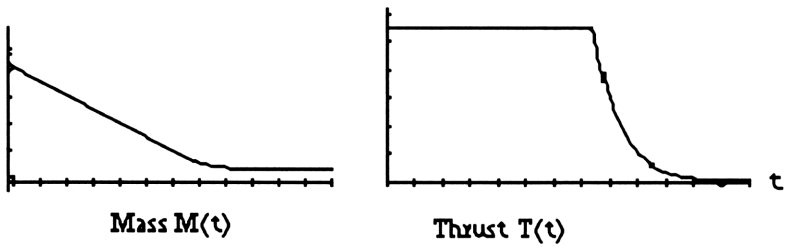
with $\gamma = \alpha/(m - \alpha t_0 - m_0)$ (so that M'(t) is continuous) and $T(t) = -\beta\, dM(t)/dt$.

As an example we take $m = 1$, $\alpha = .19$, $m_0 = .2 = .8\ m$, $t_0 = 4$, $g = 1$, $\sigma = .05$ and $\beta = 22$ so

$$M(t) = 1 - .19t \text{ for } 0 \le t \le 4, \quad M(t) = .2 + .04e^{-4.75(t-4)}) \text{ for } 4 \le t, \text{ and}$$

$$T(t) = -22\ dM(t)/dt.$$

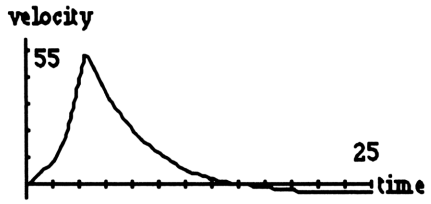
The graphs are shown below.



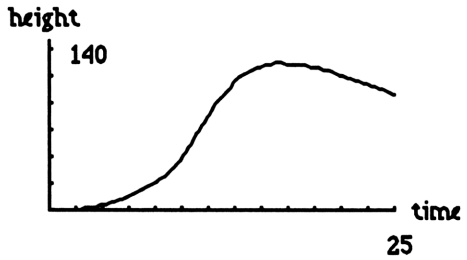
We notice that for these values of the parameters, after the thrust dissipates from mass burnoff, we have $M_\infty = .2$ and the terminal velocity will be $v_\infty = -4$. Store the formulas for M(t) and M'(t) in user defined functions **MAS** and **MDOT**, and set F(T,Y) to be

<< ' MAS(T) + (22+Y)*MDOT(T)+.05*Y' EVAL 'MAS(T)' EVAL / NEG >>.

Setting **XRNG** to 0 25 and **YRNG** to -5 55 respectively, gives the velocity graph:



The velocity rises to 53 begins to decrease at t = 4.25 but remains positive until about t = 15. The velocity at t = 25 is v = -3.65. The graph of height versus time is shown below. Recall that we have used specialized units (e.g., g = 1), so the actual height is not in a common physical unit.



Rocket height versus time

EXERCISE 2.16 : We may wish to study the sensitivity of the results we have to the parameter values used. Construct the velocity versus time graph shown above for the parameters as given, then change the parameter values of α to α = .16, t_0 to t_0 = 3.8 and β to β = 18. Overlay the new velocity time graph on the first graph.

Input Output Problems

Suppose a tank, which contains V volume units of a mixture of water and a chemical substance, receives f(t) units (weight) of the chemical in solution per minute. The chemical is vigorously mixed in the tank and the mixture drains from the tank in such a way that constant volume in the tank is maintained. If y(t) is the weight of chemical in the tank at time t, a balance equation gives dy/dt as the rate that the chemical enters the tank minus the rate that the chemical exits from the tank. This is one example of an **important** problem, namely, to determine a particular solution of the equation

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

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

$$y(0) = \frac{1}{1 - e^{-r\,P}} \int_0^P e^{-r\,(P\,-\,s)}\, f(s)\, ds.$$

The input function f is transformed to the output function y = Tf. Notice T(af + bg) = aTf + bTg when a and b are constants and f, g are input functions. This superposition property of the "operation" T shows the transformation T to be a *linear operator*.

Here we are interested in comparing the graphs of the input functions f to the graphs of output functions y = Tf. An important example, $f(t) = \sin \alpha t$, gives $y(t) = \sin (\alpha t - \theta)/R^2$ with $R^2 = (\alpha^2 + r^2)$ and $\cos \theta = r/R$, $\sin \theta = \alpha/R$. In this example there is an obvious similarity between the graphs of the input and output functions.

EXERCISE 2.17: If a signal $f(t) = \sin t$ is input into a device and produces output as described above, what value of r will produce the "delayed" output version of the signal, $\sin (t - \pi/4)$? What distortion of the signal sin 3t will be produced by this same device?

If the input signals are not sine or cosine in form, it may be difficult or impossible to find an analytical form of the output; however, a graph of the output may be found by using our differential equation graphing programs after using the calculator to evaluate the integral in y(0).

EXERCISE 2.18: Let r = 1, and set the plot parameters so that $0 \le t \le 3.14$ and $0 \le y \le 1.2$. Use the calculator to plot the following input functions with the **Function DRAW** program and the resulting output functions with a differential equation plotting program.

(a) $f(t) = 1 - \sin^4 (3t)$, (c) $f(t) = \text{Max} (\sin 6t, 0)$.

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

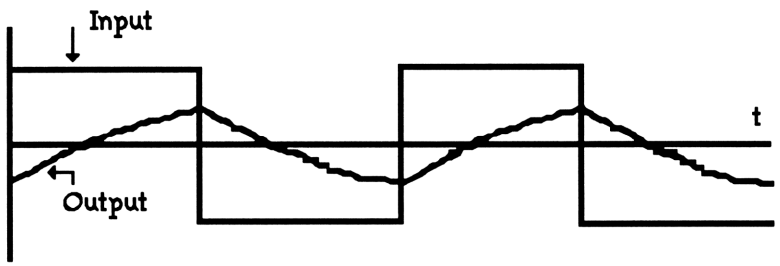
If f(t) is stored in **EQ** and $P = 1.047 \approx \pi/3$, the following program can be used to calculate y(0):

```
<< 3 FIX 'T' PURGE 0 1.047 EQ 'EXP(T)' * 'T' 3 NEG
SF ∫ 3 NEG CF 1 1.047 EXP SWAP - / STD >>
```

The input signals in (a) and (b) are periodic, spike-like disturbances of a constant input and the input in (c) is a half-wave rectified sine function.

An observant student may notice that if we start with incorrect initial conditions then the solution approaches the periodic output after some time. This suggests that the starting condition y(0) = 0 is ignored and that the resulting motion will become periodic. This is a result of the theorem that any solution of the non-homogeneous problem is the sum of a particular solution and a solution of the homogeneous problem.

The function f(t) = **2*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 ≤ t ≤ 4.



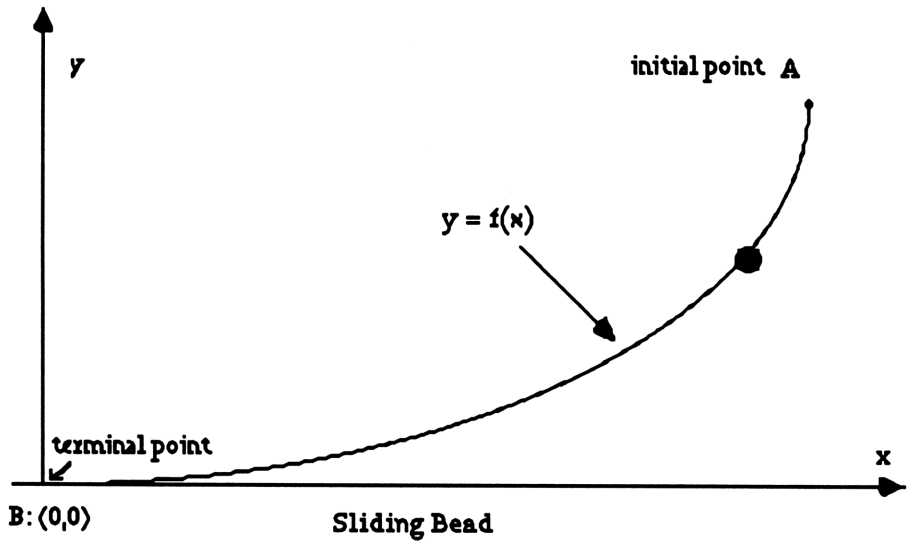
dy/dt + y = f(t)

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 ≤ 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))'**.

PROJECT EXERCISE: Travel time for a sliding bead as a function of trajectory shape:

We want to specify the shape of a curved wire by a function y=f(x), which connects the point A with coordinates (x_1,y_1) to the origin (0,0) (denoted as point B) so that a bead of mass m will slide along the wire from A to B in a minimum amount of time. The bead begins with initial velocity zero and slides with no friction under the force of gravity g.



The well-known formula for arclength ,

$$s(x) = \int_{x}^{x_1} \sqrt{1 + \left(\frac{df(r)}{dr}\right)^2}\, dr,$$

the principal of conservation of energy (this is a conservative system),

$$\frac{1}{2} m v^2 + mgy = mgy_1 ,$$

and the expression for the travel time,

$$T = \int_0^T dt = \int_0^{s(0)} \frac{dt}{ds}\, ds = \int_0^{s(0)} \frac{1}{v}\, ds,$$

leads to the equation

$$T(f) = \int_0^{x_1} \sqrt{\frac{1 + \left(\frac{df(x)}{dx}\right)^2}{2\, g(y_1 - f(x))}}\, dx ,$$

where we have explicitly noted that T depends on the curve y = f(x). (We have used the technique of changing the variable of integration and the Fundamental Theorem of Calculus.)

We further specialize the example by taking $x_1 = 200$, $y_1 = 100$ and $g = 1$. Later, for this case we will find there is a curve such that the travel time T is approximately 25.231. For these parameters the following curves connect the points A and B:

(a) $f(x) = x^2/400$

(b) $f(x) = \exp((x \ln 101)/200) - 1$

(c) $f(x) = 100\,[\,1 - \cos(\pi x/400)]$.

We will evaluate the travel time integrals for each of these curves using numerical integration.

For $f(x) = x^2/400$ the descent time integral becomes

$$T = \int_0^{200} \sqrt{\frac{1 + \dfrac{x^2}{4\left[10^4\right]}}{2\left[100 - \dfrac{x^2}{400}\right]}}\; dx\,.$$

Put $x/200 = z$, evaluate the integral to get $T = 26.779$ (attempted accuracy: 0.01). Find T for the functions given in (b) and (c). (Note that so far, no differential equation has arisen.)

The differential equation

$$\frac{dy}{dx} = \sqrt{\frac{k^2 - (y_1 - y)}{y_1 - y}}$$

can be shown to give the minimum time of descent. We have

$$dx = \frac{\sqrt{y_1 - y}}{\sqrt{k^2 - (y_1 - y)}}\; dy.$$

Use the change of variables

$$y_1 - y = k^2 \sin^2 \frac{\varphi}{2}$$

to get

$$dx = -k^2 \sin^2 \frac{\varphi}{2}\, d\varphi\,, \quad x = C - \frac{k^2}{2}\,(\varphi - \sin\varphi).$$

At $\varphi = 0$, $x = x_1$ and $y = y_1$ and at $\varphi = \varphi_1$ (to be determined), $x = y = 0$.

Thus the solution to the differential equation, along with the transformation y to φ, yields a parametric representation $(x(\varphi),y(\varphi))$ of the curve of minimum descent time. There are two constants to be determined, k^2 and φ_1; then

$$x = x_1 - \frac{k^2}{2}[\varphi - \sin\varphi], \quad y = y_1 - \frac{k^2}{2}[1 - \cos\varphi], \quad 0 \le \varphi \le \varphi_1$$

The constraint $x(\varphi_1) = y(\varphi_1) = 0$ leads to the equation

$$k^2 = \frac{y_1}{\sin^2 \frac{\varphi}{2}} = \frac{2y_1}{1 - \cos\varphi_1} = \frac{2x_1}{\varphi_1 - \sin\varphi_1}$$

We solve

$$G(\varphi) = \varphi - \sin\varphi - \frac{x_1}{y_1}(1 - \cos\varphi) = 0$$

for $\varphi = \varphi_1$ with the calculator. Notice there is a positive solution. Now determine k^2. Using the differential equation in the expression for the descent time

$$T(y) = \int_0^{x_1} \sqrt{\frac{1 + \left(\frac{dy(x)}{dx}\right)^2}{2g\left(y_1 - y(x)\right)}}\; dx,$$

we obtain the expression

$$T = \int_0^{x_1} \sqrt{\frac{1 + \dfrac{k^2 - (y_1 - y)}{y_1 - y}}{2(y_1 - y)}}\, dx = \frac{k}{\sqrt{2}} \int_0^{x_1} \frac{dx}{y_1 - y(x)} \quad .$$

But $y_1 - y = k^2 [1 - \cos \varphi]/2$, and $dx = - k^2 [1 - \cos \varphi] d\varphi$ so that

$$T(y_{opt}) = \sqrt{\frac{k^2}{2g}}\ \varphi_1 \, .$$

Use the values you get for φ_1 and k^2 to evaluate T. You should find

$$T(y_{opt}) = 25.231.$$

The reader may note that the slope of the optimal curve is infinite at the initial point. This results in a quick start for the sliding bead. The optimal curve is called a cycloid. Optimality is shown in the study of the calculus of variations. Notice however, the exponential curve gives a travel time similar to the optimal curve.

PROJECT EXERCISE: Travel time up a hill versus initial energy

This project is also concerned with the shape of a unknown function f(x). Suppose we give to a particle with initial position at the origin, energy E in the form of initial velocity. The particle sliding on the shape function f(x) (we assume that f is an increasing function) leaves the origin, travels up the "hill" f(x), reaches the limit of its travel at which all its energy has been converted into potential energy and then returns to the origin. We observe the time required to do this (as a function of the initial energy). (This problem is also illustrated by the figure given for the previous project.) The time between the particle's leaving and arriving back at the origin is given by

$$T(E) = \sqrt{2m} \int_0^{x(E)} \frac{\sqrt{1 + [f'(x)]^2}}{E - mgf(x)}\, dx,$$

and $x(E) = f^{-1}(E/mg)$, that is $E - mgf(x) = 0$.

Suppose the shape of the hill (i. e. the curve f (x)) is one of the functions given in the previous project (i.e., a parabola, an exponential function, or a trigonometric function). Graph T vs E for E = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 (numerical integration required).

Now read the first part of the article by Keller on Inverse Problems in the *American Mathematician Monthly*, volume 83, 1976, pages 107-118, and describe what is meant by the inverse problem.

3 SECOND ORDER DIFFERENTIAL EQUATIONS

We saw in chapter 1 that it is easy to program the HP-48 to treat a vector differential equation. Consider the case where the vectors have two components, $y = [\, y_1, y_2]$; that is, initial value problems consisting of two first order differential equations and the initial values of the two dependent variables:

$$\frac{dy_1}{dt} = F_1(\, t,\ y_1,\ y_2\,), \qquad \frac{dy_2}{dt} = F_2(\, t,\ y_1,\ y_2\,)$$

where $y_1(t_0)$ and $y_1(t_0)$ are given. You should note that a second order initial value problem

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

can be reduced to a first order system of differential equations

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

(with initial values of y_1 and y_2) by using the identification $y_1 = x$ and $y_2 = x'$. A significant part of this chapter will be concerned with the study of solutions of second order differential equations. We will obtain approximations of these solutions using the calculator by studying the associated vector systems. This is a common practice on all kinds of calculators and computer programs. We will plot trajectories and study the solution characteristics of such systems. Of course, in this case we can plot y_1 versus t, y_2 versus t, or plot y_1 versus y_2 as the parameter t varies.

As in the case of a single differential equation, the user has three choices: use the built-in plotting form, the user programs described in the first section of chapter 1, or the Euler or modified Euler algorithms in plotting programs as described in the second section of chapter 1. If you want to use the **EULER** and **IULER** programs as written for the vector case, make sure the **F.N** program will give a vector dY/dT when the input is a number **T** and a vector **Y**. Consider

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

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

<< → T Y << 'Y(2)' EVAL 'Y(1)' EVAL NEG 2 →ARRY >> >> .

(The first two stack items are used as local variables, and the inner program creates the first and second components of the output and forms a vector from these components.) After a value for **H** is stored, the stack input 0 [0 0] to either of the programs **EULER** or **IULER** will produce the values at **T** = **H**.

Suppose we wish to plot the vector solution of $dy_1/dt = y_2$, $dy_2/dt = -y_1$, $y_1(0) = 1$, $y_2(0) = 0$ using the built-in HP-48G algorithm on the interval $0 \le t \le 2\pi$. We can choose the input form **Diff Eq** under **PLOT** as indicated in chapter 1 or we can use the user programs constructed in chapter 1 which eliminate some of the inconveniences of the input form. In the latter case we execute the program **IN.FN** and respond with

<< 'Y(2)' EVAL 'Y(1)' EVAL NEG 2 →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 plot should be a "circle". The actual figure is a set of points connected by straight lines. Exit to the stack and press **T** and **Y** in the **VAR** menu to get (approximately) 6.283 [1 0]. This will be more accurate that the result as drawn by **GR.12**, say with **H** = .0628 and **N** = 100. (The program **GR.12** is obtained from **GR.01** by substituting **DROP** for **DROP2** and inserting **DROP** after **PIXON**.)

Probably the first type of second order differential equation you will study is a linear homogeneous equation with constant coefficients. Such an equation can be solved by finding appropriate values of a constant r so that $x = e^{rt}$ is a solution of the problem. The calculator can be used to determine unknown coefficients in constructing general solutions. You should also use the calculator to become familiar with the plots of solutions that occur in common problems. This type of EXERCISE will use the function grapher in the calculator and we will not study these kinds of problems here. But here is an associated problem. How are the coefficients in these differential equations obtained?

Several mathematical models lead to second order ordinary differential equations with constant coefficients. The coefficients are usually obtained from measurements either directly on the physical system or on solutions of the system. How could we deduce approximate values of these coefficients using measurements on the solutions ? Consider the differential equation

$$\frac{d^2y}{dt^2} + B\frac{dy}{dt} + Cy = 0.$$

A common solution function has the form

$$y(t) = ae^{-pt} + be^{-qt},$$

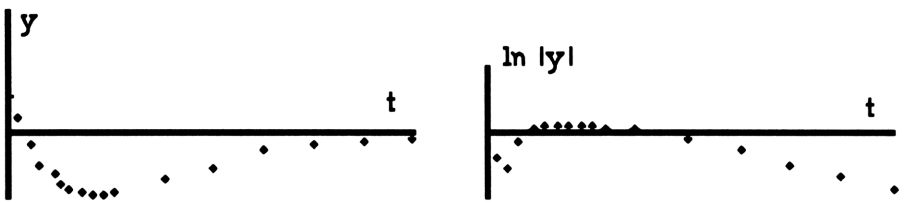
where a, b, p, q are parameters. Suppose a set of values { (t, y) } is obtained by making measurements. There is usually some experimental error in measurements so the entire set { (t, y) } will be used to find the parameters. In this example the measurements are:

{ (0, 1), (.1, .30), (.2, -.20), (.3, -.60), (.4, -.88), (.5, -1), (.6, -1.2), (.7, -1.2), (.8, -1.3),

(.9, -1.3), (1, -1.2), (1.5, -1), (2, -.70), (2.5, -.50), (3, -.3), (3.5, -.17), (4, -.11)}

We want to deduce first approximate values of the parameters a, b, p, q, then use these to determine the parameters B and C in the equation above.

We may be able to learn something from a plot of the data. A program for producing such a plot is given below. We also recall that if $y = Ae^{-kt}$ then a graph of $\ln|y|$ versus t is a straight line with slope -k. So a graph of data $\{(t, \ln|y|)\}$ may reveal information. Suppose we construct a list of the data and store as **L1**.



Graphs of Solution Observations

The graphs shown are generated with the HP-48 program

<< ERASE DRAX L1 LIST→ 1 SWAP START PIXON NEXT GRAPH >>

<< ERASE DRAX L1 LIST→ 1 SWAP START C→R ABS LN R→C PIXON NEXT GRAPH >>.

after setting **XRNG** to 0 4 and **YRNG** in the first problem to -1.4 1.1 and in the second program to -2.1 .28 (approximate values for ln |-.11| and ln 1.3).

To get approximations for a, b, p, and q we proceed as follows: suppose $p > q$, then $y(t) = e^{-qt}(ae^{-(p-q)t} + b)$. Since the first term becomes negligible as t increases, $b < 0$ (we replace b with $-|b|$) and a plot of $(t, \ln|y|)$ is a straight line for large t with slope -q. From the second graph we get $q = -.5 \ln(.11/.7) = .925$ from the data points (4, -.11) and (2, -.7). The first data point gives $a = |b| + 1$ (which is, of course, an approximate equation), and $dy/dt(.85) = 0$ gives $p(1+|b|)e^{-.85p} = .42|b|$. Finally we use the approximate equation $\ln -y(t) = \ln|b| - .925t = 0$ at $t = 1.5$ which gives $|b| = 4$, $a = 5$, $pe^{-.85p} = .336$. This equation has two solutions $p = .525$ and $p = 2.2$. Since we want $p > q$, we take $p = 2.2$. This yields the equation

$$y(t) = 5e^{-2.2t} - 4e^{-.925t}.$$

You should now plot this equation together with the plot of the data for comparison. These approximate values for a, b, p and q can be taken as starting values to an iterative process to determine the parameter values by a least squares fit to data. See Chapter 5. It is easy to use the values of p and q to determine the corresponding values of B and C in the differential equation.

EXERCISE 3.1: Suppose the following data is collected on the solution of the second order differential equation given above.

{ (0, 11.04), (.4, 12), (.8, 11.06), (1.2, 8.47), (1.6, 4.75), (2, .54), (2.4, -3.48), (2.8, -6.71), (3.2, -8.7), (3.6, -9.22), (4, -8.29), (4.4, -6.15), (4.8, -3.2), (5.2, .05), (5.6, 3.09) }

Find approximate values of B and C in the differential equation.

Second Order Input Output Problems

We now consider constructing the solution of a non-homogeneous second order differential equation with constant coefficients. The problem is treated in many textbooks for special types of forcing, usually sine or cosine forcing functions. A model for an elastic spring with damping and with external forcing f(t) or a model for a simple electrical circuit loop with external voltage is:

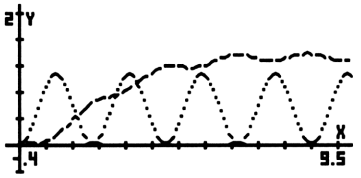
$$\frac{d^2x}{dt^2} + 2r\frac{dx}{dt} + \omega^2 x = f(t),\ \ x(0) = \frac{dx}{dt}(0) = 0,\ \omega^2 > r^2 .$$

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 plot of the input function. The forcing function (input) and solution (output) resulting from this program are shown below.



EXERCISE 3.2: Find the output graph for $f(t) = 1 - \sin^4(3.14t)$ for $\mu = 1$, and $r = .5$. Choose plot parameters to show $0 \le t \le 6$ and $0 \le y_1 \le 1$. Add the input function

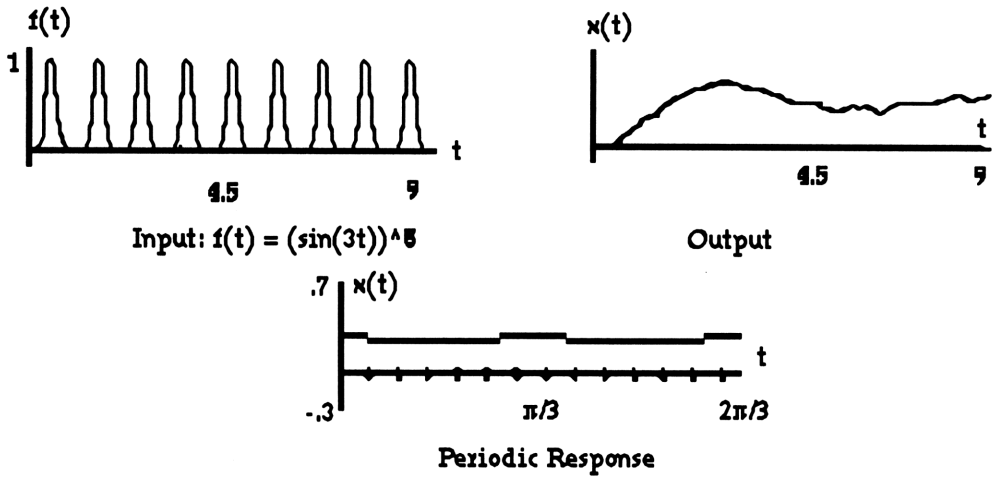
graph as an overlay. Comment: The output function for this input can be obtained from a table of integrals after several substitutions using the method of undetermined coefficients. But, an output function for an input $f(t) = 1/(2 - \sin^4(3.14t))$ can not be found this way.

Suppose the forcing function f(t) is periodic with period length P. If we can change the initial conditions so that $x(P) = x(0)$ and $x'(P) = x'(0)$, then the resulting solution is periodic. And if the damping coefficient $r > 0$, then all solutions will eventually be close approximations to the periodic solution when viewed over one period. We may want to view such a solution without waiting for asymptotic behavior to emerge. Suppose we determine solutions $x_1(t)$ and $x_2(t)$ of the associated homogeneous system so that $x_1(0) = x'_2(0) = 1$ and $x'_1(0) = x_2(0) = 0$; then a general solution is $x(t) = a\,x_1(t) + b\,x_2(t) + x_q(t)$ where $x_q(t)$ is the solution constructed above for 0 initial conditions for x and x'. Expressions for $x_1(t)$ and $x_2(t)$ are $\mathbf{x_1(t) = e^{-rt}[\cos \mu t + (r/\mu) \sin \mu t]}$ and $\mathbf{x_2(t) = (1/\mu)\, e^{-r\,t} \sin \mu t}$. We can use the calculator to compute the integrals in $x_q(P)$ and $x'_q(P)$, 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 has period $\pi/3$. The periodic response is also shown over two periods. The average value for this forcing f(t) is $\pi/6$ and $f(t) = [f(t) - \pi/6] + \pi/6$, so a portion of the periodic response is the constant $\pi/(6\omega^2)$.



EXERCISE 3.3 : Find and plot the periodic output response for $f(t) = 1 - \sin^4(3.14t)$ for $\mu = 1$, and $r = .5$. Then overlay a plot of the input forcing function.

The friction/resistance term in the spring/circuit model that we have been considering is given by $2rdx/dt$ 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 3.4 : Find a periodic solution to nonlinear problems of the form

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

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

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

(a) Take $R = R_1(dx/dt)$ and $K(x) = x$. Use initial condition $t = 0$, $x = 0$, $x' = 1.56$.

(b) Take $R(dx/dt) = dx/dt$ and $K(x) = x$. Use initial condition $t = 0$, $x = 0$, $dx/dt = 2$.

(c) Take $R(dx/dt) = dx/dt$ and $K(x) = \sin x$.

(d) Take $R(dx/dt) = R_1(dx/dt)$ and $K(x) = \sin x$.

Note: In each case, if the solution you get is not periodic then use the values of x and dx/dt at t = 6.283 as initial conditions and generate another solution. Which nonlinearities caused a phase shift from the linear case (b)?

Suppose that in the system $dY/dt = F(t,Y)$, F is periodic in t. Since the system has a periodic rhythm, perhaps the rhythm will also exist in the solutions. It is easy to use the calculator to illustrate the idea of locating a solution for t = any multiple of a given time period. For example, the differential equation

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

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

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

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

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

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

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

EXERCISE 3.5: Plot the points x(2πn), x'(2πn) for ξ = 0 and n = 1, 2, ... (several values of n) for ω = 1/Sqrt(5) and for ω = 1/3. Note that the points cycle around the ellipse. If ωn is an integer m for some integer n, then you can see the solution is periodic, but what happens when ω is irrational?

The graph of { (x(nT), dx/dt(nT)) : n = 0, 1, 2, ... } of the solution of a differential equation dx/dt = f(x,y,t), dy/dt = g(x, y, t) when the function f and g have period T in t is called a *Poincare section*. The following program collects 10 points for such a graph on the stack:

```
<< { T Y FN } TOL 1 10 FOR N Π N * 2 * →NUM → TF
<< TF RKF 'Y(1)' EVAL 'Y(2)' EVAL RC 3 ROLLD >> NEXT >>
```

Executing this program for the **FN** function

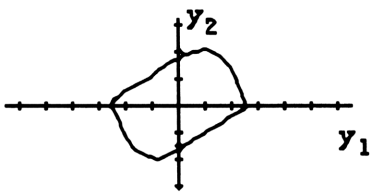
```
<< 'Y(2)' EVAL '12*COS(T) – X – ε*('Y(1)^2–1)*Y(2)' EVAL 2 →ARRY >>
```

(periodically forced Van der Pol equations) for different values of ε gives the data:

	ε = .05	ε = .1	ε = .15
initial value	Y = [1, 1]	Y = [1, 1]	Y = [1, 1]
first section point	Y = [-2.1, 10.4]	Y = [-2.24, 7.97]	Y = [-2.21, 6.08]
second section point	Y=[-2.19, 10.28]	Y= [-1.99, 8.28]	Y = [-1.82, 6.65]
third section point	Y = [-2.19, 10.28]	Y = [-1.98, 8.29]	Y = [1.79, 6.86]

There were no further changes in the section points coordinates (to 6 places). In each case the solution with initial value Y = [1,1] collapsed to a periodic solution. *Warning*: the program's execution is about 30 minutes.

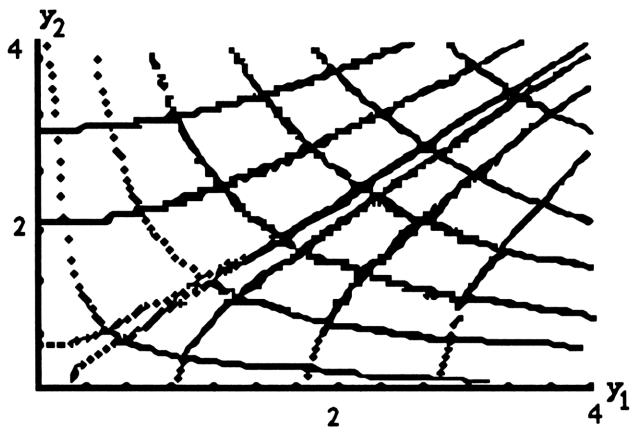
It is interesting to plot a solution starting at one of the section points over a period of the system. The following figure is such a plot for one of the examples above.



Forced Van der Pol with c = .1

Trajectories in the y_1-y_2 Plane

A topic occurring early in many differential equation textbooks is that of determining trajectories that are orthogonal to the members of a one-parameter family of curves, say $W(y_1, y_2, p) = 0$. The usual technique is to first find the differential equation satisfied by the members of the given curve family, say $dy_2/dy_1 = m(y_1, y_2)$; then curves that are orthogonal satisfy the differential equation $dy_2/dy_1 = -1/m(y_1, y_2)$. If the original family is given 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 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 plot members of both the given family of curves and the orthogonal trajectories. This is our first example of what is called an *autonomous system*. A specific example is shown.



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

EXERCISE 3.6: 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 plot in the y_1-y_2 plane resulting from the inputs to the **G.12**:

t_0	0	0	0	0	0
y_0	[.5 .1]	[.75 .1]	[1 .1]	[1 .4]	[1.5 .5]
t_f	4	4	2	2	2

These are five solution trajectories (ovals) for the system

$$dy_1/dt = y_1(y_1^2 - y_2^2), \quad dy_2/dt = y_2(3y_1^2 - y_2^2).$$

Now overlay the solution trajectories of the orthogonal system corresponding to the following inputs to the **G.12** program:

t_0	0	0	0	0	0
y_0	[0 3.4]	[0 2.5]	[0 1.5]	[2 0]	[3.4 0]
t_f	1.2	.8	.8	.8	.8

Plots in the y_1-y_2 plane of solutions $(y_1(t), y_2(t))$ of differential equations $y_1' = F_1(y_1, y_2)$, $y_2' = F_2(y_1, y_2)$ are called *phase plane* plots. If $F_1(y_1, y_2)$, and $F_2(y_1, y_2)$, have continuous partial derivatives, solutions to initial value problems are unique and it is elementary to show that under such circumstances solution trajectories arising from different initial points either coincide or do not intersect. If fact, it is easy to see that if $(y_1(t), y_2(t))$ is a solution of an equation of this form and a is any constant, then $(y_1(t+a), y_2(t+a))$ is also a solution. *Closed trajectories* in the phase plane indicate periodic solutions. Constant solutions, that is, points (y_1, y_2) such that $F_1(y_1, y_2) = F_2(y_1, y_2) = 0$ are called *critical point solutions* (also equilibrium solutions). Other trajectories of particular interest are those nearby to a critical point.

- If trajectories arising at all points within some circle around a critical point (y_{1c}, y_{2c}) leave the vicinity of (y_{1c}, y_{2c}) as $t \to \infty$, then (y_{1c}, y_{2c}) is called a **repelling solution**, i.e., unstable.
- If trajectories arising at all points within some circle around a critical point (y_{1c}, y_{2c}) approach (y_{1c}, y_{2c}) as $t \to \infty$, then (y_{1c}, y_{2c}) is called an **attracting solution**, i.e., asymptotically stable.

Some well-studied examples of autonomous are presented below. Note the asymptotic behavior of the solution trajectories as indicated by the graphs.

EXERCISE 3.7 : 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 begin 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**.

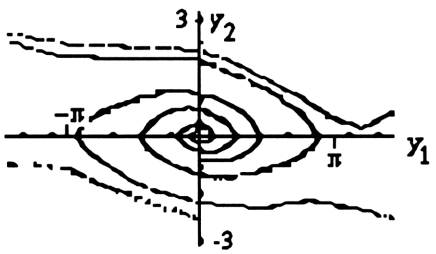
EXAMPLE. The differential equations

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

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)

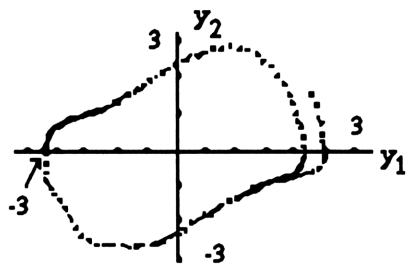
EXAMPLE. The system

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

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 >>.
```

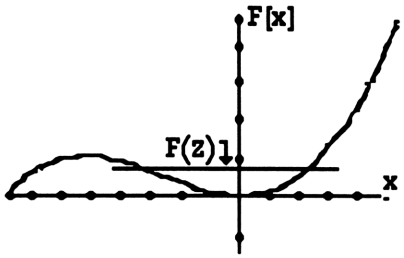
Another problem that has an isolated attracting periodic solution is the Van der Pol differential equation. This equation was studied in connection with its application to an electronic component. This example is usually studied as a function of a parameter μ contained in the "damping" term. Our figure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



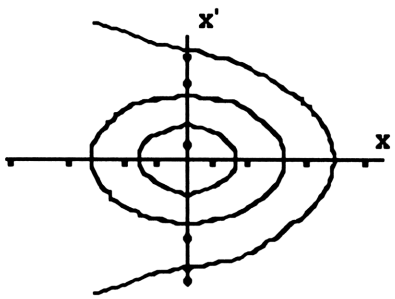
$$dy_1/dt = y_2,\ dy_2/dt = -[y_1 + .3(y_1^2 - 1)y_2]$$

typical graph: here μ = .3. Notice that the motion is counterclockwise and that the solution was started at (x, y) = (2, 2). The solution quickly moves close to its asymptotic shape and is periodic. Solutions starting inside the closed curve (except from (0, 0)) also move out to the periodic solution. Variation of the parameter μ causes dramatic changes in the shape and period of the solution.

EXERCISE 3.8: In this EXERCISE we will examine the cycle times of periodic solutions of several special differential equations. The equations under consideration have solutions that resemble the trajectories graphed in the figure below.



Construction for Trajectories
(See region between F[z] & F[x])



Trajectories

Here we assuming that y(t) satisfies the initial value problem

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

where the essential feature of f(x) is that it changes sign from negative to positive as y increases through 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) it is not. Calculate and plot the values of P for one of the examples (a), (b), or (c) listed above for several values of z. Use the numerical integration key (program) on your calculator with a tolerance of 0.005. The following values of P are for part (d) above:

z values	.25	.5	.75	1	1.25	1.5	1.75	2	2.25
P values	3.94	5.29	12.74	21.54	8.29	5.74	5.04	5.45	8.37

Note that $dx/dt = 0$ and $x = \pi/4$ and $dx/dt = 0$, $x = 3\pi/4$ are equilibrium points.

Linear Variational Systems in the y_1-y_2 Plane

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

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

We will consider the case $\det(A) \neq 0$, which means that the origin (0,0) is the only critical point. Special solutions have the form $w = \text{column}[y_1, y_2] = e^{\lambda t} v$ where 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

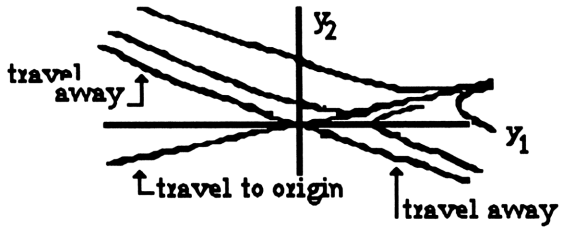
eigenvalue for the system then the column vector v = [c, d] is a non-zero solution of $(A - \lambda I)v = 0$. Other solutions of our system are linear combinations of these special solutions (in most cases).

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 3.9 : 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

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

Start at (t, y) = (0, [0, 1]) after setting the plot parameters to show **H-VIEW** -2 2 and **V-VIEW** -1 1 and plot for $0 \le t \le 3$.

EXERCISE 3.10 : Use **G.12** to graph the trajectories initiating at (t, $[y_1, y_2]$) = (0, [0, 1]) and at (0, [-1, -1]) for the system

$$\frac{dy_1}{dt} = -(2y_1 + y_2), \quad \frac{dy_2}{dt} = -y_1 + 2y_2.$$

What are the eigenvectors for this system associated with the [0, 0] critical point? Can you see them on the graphs ? The graph should show that the origin (0,0) is neither an attracting or repelling critical point solution for the system.

EXERCISE 3.11 (a) Use the calculator to draw a graph of the solution of

$$\frac{dx}{dt} = x - \frac{5}{2}\,y, \quad \frac{dy}{dt} = \frac{3}{2}x - 3\,y$$

with initial conditions x(0) = -.5, y(0) = .5 for $0 \le t \le 4$. Use XRNG scale $-1.4 \le x \le 1.4$, and YRNG $-1 \le y \le 1$. When the plotting program is completed, record the final values of the solution x(4)/y(4). Consider the matrix of coefficients A= {column[1, 1.5], column[-2.5, -3]} What is the characteristic equation det (A–rI) = 0 ? What are the solutions r_1, r_2? Give nontrivial solutions of (A–rI)v = 0 for r = r_1 and for r_2. Calculate v_1/v_2 for each solution. Compare with the answer you obtained for x(4)/y(4).c) Give the general solution of dw/dt = Aw. Which term tends to vanish first as t increases ?

(b) Use the calculator to draw a plot of the solution of

$$\frac{dx}{dt} = 5.7\,x - 10\,y, \quad \frac{dy}{dt} = 4\,x - 6.3\,y$$

with initial conditions x(0) = .3, y(0) = -.2 for $0 \le t \le 6$. Use XRNG scale $-1.47 \le x \le 1.7$, and YRNG $-1 \le y \le 1$. Consider the matrix of coefficients A= {column[5.7, 4], column[-10, -6.3]} What is the characteristic equation det (A–rI) = 0 ? What are the solutions r_1, r_2 ? Give nontrivial solutions of (A–rI)v = 0 for r = r_1 and for r_2. Give the general solution of dw/dt = Aw.

Solution graphs of ***nonlinear autonomous systems*** near a critical point solution can be studied using a linear approximation. Let the vector y = 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

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

EXAMPLE: Consider the system $dy_1/dt = 2y_1^2 + y_2^2 - 9$, $dy_2/dt = y_1^2 + y_2^2 - 5$, which has critical point solutions (2, 1), (-2, 1), (2, -1), (-2, -1). The variational matrix for the last critical point has eigenvalue equation $\lambda^2 + 16\lambda + 8 = 0$. The roots of this equation clearly are negative so that (-2,-1) is an attracting critical point.

The calculator can be used to find the matrix J associated with any equilibrium point y_c by using the sequence of programs given below. Because such information is also useful for a vector system dy/dt = F(y) where y and F(y) are vectors with m components, we present the programs for the vector case. We further will present the programs in a form where the labeling of the independent variables can be specified by the user. For example, instead of y_1, y_2, etc. the user might prefer u, v, ... The user's preference will be entered into a stored list as shown. After the matrix J is determined then the calculator can be used to find the eigenvalues as explained in the next section of this chapter.

Here is an outline of the procedure, assuming that we know the point y_c. We store the value of m in **M** and store the names of the m components in a list called **PL**. For example, **PL** = { **U V** }. Make sure each of the variables in **PL** have been purged. Store the components of the F function in a list **FL.** For instance, in the example given above **FL** = { **'2*U^2+V^2–9' 'U^2 + V^2 – 5'** } where U replaces y_1 and V replaces y_2. Then execute the program **DER** below:

Subprogram Name:	**DER**
Purpose:	Creates list **JL** puts the **FL** functions on the stack and executes **DERA** M times.

<< { } 'JL' STO FL OBJ→ 1 SWAP START DERA NEXT>>

Program **DER** calls the subprograms **DERA** and **DERB**.

Subprogram Name:	**DERA**
Purpose:	Creates M -1 more copies of the first element on the stack for use in the next subprogram.

<< 1 M 1 - START DUP NEXT DERB >>

Subprogram Name:	**DERB**
Purpose:	Takes M copies of a function in **FL**, creates the derivatives with respect to each parameter in **PL** and stores them in **JL**.

<<1 M FOR I PL I GET ∂ M 1 + I - ROLLD NEXT
M →LIST JL + 'JL ' STO>>.

At this point, for m = 2, **JL**= $\{F_{1u}(u,v)\ F_{1v}(u,v)\ F_{2u}(u,v)\ F_{2v}(u,v)\}$. Now store the values of the variables in **PL** at Y_c (e. g. **U** = -2, **V** = -1) and create matrix **JMAT** with a program called **JEV** given by

<< JL OBJ→ 1 SWAP START →NUM M SQ ROLLD NEXT
{M M} →ARRY 'JMAT' STO >>

At this point we have constructed the matrix **JMAT**. There is a straight forward procedure for finding the eigenvalues of **JMAT**. See the next chapter. For m = 2, the eigenvalues are the roots of the quadratic polynomial

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_0 = column $[u_0\ v_0]$ of the critical point is known. Then better approximations of the critical point may result from one or more applications of the following algorithm:

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

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 $[\mathbf{U - U_n, V - V_n}]$ 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 $[\mathbf{U_n, V_n}]$, by the command – (a minus command). If not, execute –, then **NWTN** again, etc.

EXERCISE 3.12: 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 3.13: 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 indicates that there are two solutions of the differential equation that approach zero as time increases at widely differing rates. Such a system of differential equations is called *stiff* and Hewlett Packard has provided a second algorithm to handle such cases. Nonlinear systems can also be stiff. For example, a system dy/dt = F(y) which has an equilibrium y_c for which the matrix $J(y_c)$ discussed above has eigenvalues with 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 adaptation of **G.01** is constructed for this problem. The reader should recall **G.01** and edit. Note that the stack command **{T Y FN FNY FNT}** replaces **{T Y FN}** in the default algorithm and that the stack input to **RRKSTEP** consist of four elements. The last element is an indicator variable (in this case 2) which determines the method to be used.

EXERCISE 3.14: Use **IN.FN** to store an appropriate function **FN**. Store **FNY** and **FNT** as given above. Use **IN.PP** to set **XRNG** to 0 1 and **YRNG** to 0 1. Put the entries 0 [1 -1] 1 on the stack and execute **GS.01** (see next page). An alternative is to use the form provided by HP for plotting the solution of a differential equation. To do this enter the **FN** function as given above for F and check the STIFF box. Then enter the **FNY** and **FNT** functions given above in the $\partial F\partial Y$ and $\partial F\partial T$ boxes respectively. The exact solution is $y_1 = e^{-t}$, $y_2 = -e^{-t}$. Use the **Function** mode to overlay the solution as an accuracy check.

Program Name:	**GS.01**
Purpose:	Generate a T Y(1) graph of the solution to T_f.
Stored Quantities:	**XRNG YRNG FN FNY FNT TOL HS**

Input	level 3	level 2	level 1
	T_0	vector Y_0	T_f

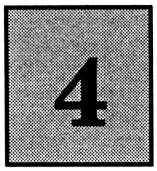
The output stack is empty, the variables T and Y contain updated values

```
<< { # 0d # 0d } PVIEW DRAX 3 ROLLD 'Y' STO 'T'
STO → TF << { T Y FN FNY FNT } TOL HS 0 T 'Y(1)'
EVAL R→C 5 ROLLD DO RRKSTEP T 'Y(1)' EVAL R→C
DUP 7 ROLLD 6 ROLL LINE SWAP DUP 3 ROLLD T + TF
UNTIL > END SWAP DROP TF T - SWAP RRKSTEP T
'Y(1)' EVAL R→C DUP 7 ROLLD 6 ROLL LINE SWAP DROP
TF T - SWAP RRKSTEP T 'Y(1)' EVAL R→C 6 ROLL LINE
4 DROPN >> PICTURE >>
```

EXERCISE 3.15: Use the calculator to graph several solutions in the x y plane of the 'non-stiff' system

$$\frac{dx}{dt} = x(1 - x - y), \quad \frac{dy}{dt} = y(.5 - .75\,x - .25\,y)$$

showing XRNG $0 \le x \le 1.5$, and YRNG $0 \le y \le 2.5$. For x(0) = .1, y(0) = .2, plot for $0 \le t \le 25$, for x(0) = .1, y(0) = .3 plot for $0 \le t \le 15$, for x(0) = y(0) = 1.5 plot for $0 \le t \le 15$, for x(0) = 1.5, y(0) = 1.0, plot for $0 \le t \le 20$ and for x(0) = 1.5, y(0) = .8 plot for $0 \le t \le 20$. Here notice that (x, y) = (.5, .5), (x, y) = (0, 2) and (x, y) = (1, 0) are equilibrium solutions.



LINEAR SYSTEMS OF DIFFERENTIAL EQUATIONS WITH CONSTANT COEFFICIENTS

In this chapter we consider linear systems of differential equations of the form $y' = Ay + f(t)$ where y and f(t) are vectors with, say, n components and A is an n by n matrix. Solutions can be constructed from the eigenvalues and eigenvectors of A. There are built-in programs in the HP-48G calculator for these eigenvalues and eigenvectors. However, a differential equations student may wish to know just how these quantities could be calculated. Consequently, we will present several special programs to illustrate steps involved in obtaining eigenvalues and eigenvectors. We recommend that beginning students use these special programs at first to become comfortable with the mathematical concepts then use the built-in programs to avoid the computational pitfalls that are sometimes encountered.

Homogeneous Systems

Consider first the vector problem $dy/dt = Ay$. Here we want to find all solutions of the differential equation. It is readily shown that if n independent vector functions satisfying the differential equation can be determined and a matrix Y(t) is constructed with these columns, then all solutions have the form Y(t)c where c is a vector with n components. The "educated guess" $y(t) = e^{\lambda t}v$ (here y(t) and v are vectors) leads to the 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 are given.

A calculator program **to display the eigenvalue equation** for a 2 by 2 matrix is:

Program Name:	**EIG2**
Purpose:	Display the eigenvalue equation.
Stored Quantities:	2 by 2 matrix A

```
<< 'X' PURGE A DET 8 RND → D1 << 'X^2 - (A(1,1) +
A(2,2))*X + D1' EVAL >> >>
```

EXERCISE 4.1: 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

```
<< 'X' PURGE A DET 8 RND → D1 << 'X^3 – (A(1,1) + A(2,2)
+ A(3,3))*X^2 + (A(1,1)*A(2,2) – A(1,2)*A(2,1) + A(1,1)*A(3,3) –
A(1,3)*A(3,1) + A(2,2)*A(3,3) – A(3,2)*A(2,3))*X – D1' EVAL >>
>>
```

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

EXERCISE 4.2: Find the characteristic equation for the matrices

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 1 & 0 & 1 \\ 4 & -4 & 5 \end{bmatrix}, \quad A = \begin{bmatrix} -1 & -6 & 3 \\ 3 & 8 & -3 \\ 6 & 12 & -4 \end{bmatrix}, \quad A = \begin{bmatrix} -5 & -8 & -12 \\ -6 & -10 & -10 \\ 6 & 10 & 13 \end{bmatrix}.$$

(The first matrix has the eigenvalue equation $\lambda^3 - 6\lambda^2 + 11\lambda - 6$.)

We can find the roots of the eigenvalue equation simply by executing the **PROOT** program on the HP-48G calculator (left-shift **SOLVE**, then **POLY**) — see below — or by storing the equation and using the **DRAW** and/or **SOLVR** programs. You may have to try several settings of the plot parameters **XRNG, YRNG**.

EXERCISE 4.3: Find the eigenvalues of the matrices given in EXERCISE 4.2. (Eigenvalues for the first matrix are 1, 2, 3.)

Consider the matrix

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

The eigenvalue equation in the variable x is $x^3 - 3x - 1$. A simple way to obtain the roots on the HP-48G is to press [↱] **SOLVE**, move to **Solve poly...** and press **OK**. Enter the vector of coefficients [1 0 -1 -1] and press **OK** and **SOLVE** to get all roots. (You may want to go to **EDIT MODES 3 FIX** to see all the roots.) Another way to obtain the roots is to use the **ROOT** command (under **FCN** on the graphics screen) after plotting the ploynomial from -2 to 2. A root is x = 1.3247---. If we divide the polynomial $x^3 - x - 1$ by (x – 1.3247...) 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.

EXERCISE 4.4: 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–λI) is singular and the linear system solver is not appropriate to solve the equation (A–λI)v = 0. Place (A–λI) on the stack and use the programs named **PIV** and **ROKL** given below to obtain the Gauss-Jordon echelon form to determine the row space of (A–λI) and nontrivial solution vectors v. Alternately you can use the program **RREF** on the HP-48G (see below).

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 4.2 has eigenvalues 1, 2, and 3.

For $\lambda = 1$ an equation for v is

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

If this matrix is placed on the stack and the command 1, 2 **ROKL** (to interchange rows) is given we get

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

Next give the command 1,1 **PIV** (creating 0's in the first column) to get

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

Now the commands 2,2 **PIV** gives

$$\begin{bmatrix} 1 & 0 & .5 \\ 0 & 1 & -.5 \\ 0 & 0 & 0 \end{bmatrix}.$$

The solution relations $v_1 = -.5\ v_3$, $v_2 = .5\ v_3$ result: i. e., v = [-1, 1, 2] or any nonzero multiple of this vector. Alternately the command **RREF** (on the HP-48G) will accomplish the same result as the **ROKL** and **PIV** commands. Similarly for $\lambda = 2$, we find that any multiple of v = [-2, 1, 4] is a corresponding eigenvector; for $\lambda = 3$, we find that any multiple of v = [-1, 1, 4] is a corresponding eigenvector.

EXAMPLE: The matrix

$$A = \begin{bmatrix} 6 & 7 & 8 \\ -2 & 0 & -2 \\ -4 & -6 & -6 \end{bmatrix}$$

has eigenvalues $\lambda = -2$ and $1 \pm i$. The procedure shown above gives the eigenvector v = column [1, 0, -1] corresponding to $\lambda = -2$. For $\lambda = 1 + i$, the matrix $A - \lambda I$ is

$$A = \begin{bmatrix} (5,-1) & 7 & 8 \\ -2 & (-1,-1) & -2 \\ -4 & -6 & (-7,-1) \end{bmatrix}.$$

When we use **RREF** (or 1 1 **PIV**, then 2 2 **PIV**) we obtain

$$\begin{bmatrix} (1, 0) & (0, 0) & (1, -.5) \\ (0, 0) & (1, 0) & (.5, .5) \\ 0 & 0 & 0 \end{bmatrix}.$$

This leads to an eigenvector v = 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 4.2 we determined eigenvalues and corresponding eigenvectors in the example just after the **ROKL** program. Thus

$$Y(t) = \begin{bmatrix} -e^{t} & -2e^{2t} & -e^{3t} \\ e^{t} & e^{2t} & e^{3t} \\ 2e^{t} & 4e^{2t} & 4e^{3t} \end{bmatrix}.$$

The solution of y' = Ay, y(0) = column[1 3 -5] is $y(t) = Y(t)Y^{-1}(0)$ column[1 3 5].

For the matrix example given just above the preceding paragraph (one real and a pair of complex eigenvalue) we proceed as follows. If a matrix A has eigenvalues $\lambda = \alpha \pm \beta i$ and corresponding eigenvectors $\mathbf{c} = \mathbf{a} \pm i\mathbf{b}$, then by adding the exponential solutions obtained it is known that the quantities $e^{\alpha t}$ (cos βt **a** – sin βt **b**) and $e^{\alpha t}$ (sin βt **a** + cos βt **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 4.5: Find a fundamental matrix of solutions of dy/dt = Ay for

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

When there is a eigenvalue 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 $y(t) = e^{\lambda t}(vt + d)$. In the latter case, we derive the following requirements by

substitution: $(A - \lambda I)\, v = 0$ and $(A - \lambda I)\, d = v$. To determine the vector d we can augment the matrix $(A - \lambda I)$ with the additional column v and use Gauss elimination to determine d. For

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

$\lambda = -2$ is a eigenvalue of multiplicity 2 and $\lambda = -1$ is a simple eigenvalue. The eigenvectors corresponding to $\lambda = -2$ are multiples of v = column [1, -1, 2] and the eigenvectors corresponding to $\lambda = -1$ are multiples of v = column [1, -1, 3]. The equation $(A + 2I)\, d$ = column [1, -1, 2] has a solution d = column [0, 1, 0]. (Such a solution vector is easily obtained on the calculator, first by calculating $(A + 2I)$, augmenting the matrix with the column [1, -1, 2] then using **RREF** on the HP-48G or **PIV** as listed above to obtain d.) For this matrix A we have a fundamental matrix of solutions

$$Y(t) = \begin{bmatrix} e^{-t} & e^{-2t} & te^{-2t} \\ -e^{-t} & -e^{-2t} & (1-t)e^{-2t} \\ 3e^{-t} & 2e^{-2t} & 2te^{-2t} \end{bmatrix}.$$

The matrix eigenvalues for

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

are $\lambda = -3$ (multiplicity 2) and $\lambda = -2$. The eigenvectors corresponding to $\lambda = -3$ are linear combinations of c = 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 4.6: Find a fundamental matrix of solutions for the system y' = Ay for each of the following matrices:

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} -3 & 1 & 0 \\ 0 & -3 & 1 \\ 4 & -8 & 2 \end{bmatrix}, \begin{bmatrix} -1.25 & -.5 & .75 \\ .5 & -1 & .5 \\ .25 & .5 & -1.75 \end{bmatrix}.$$

We wish to present a program which accepts an n by n matrix as input and generates its eigenvalue equation. There is a algorithm for the coefficients of this equation which combines many subdeterminants to form the coefficients. Such an algorithm seems cumbersome for the calculator; however another less well known algorithm involves products and sums of n by n matrices and the computation of the traces of some of these matrices, something this calculator does with little trouble. The following algorithm is taken from Cullen, *Linear Algebra with Applications*, Scott Foresman and Company, 1988: Let A be an n by n matrix, set $B_0 = I$ and then for $k = 1, 2, \ldots, n$ let

$$A_k = AB_{k-1},\ c_k = -(1/k)\ \mathrm{tr}(A_k),\ B_k = A_k + c_kI.$$

Then the characteristic polynomial is given by

$$\lambda^n + c_1\lambda^{n-1} + c_2\lambda^{n-2} + \ldots + c_{n-1}\lambda + c_n.$$

The following program will generate the coefficients in the eigenvalue equation:

Program Name	**CHAR**
Purpose	Find the eigenvalue equation for a matrix in level 1 on the stack
Input stack: square matrix	Output stack: list of the coefficients in characteristic equation

```
<< 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 >> >>
```

EXAMPLE: Place the matrix

$$A = \begin{bmatrix} 3 & 4 & 6 & 4 \\ 1 & -1 & 1 & 1 \\ 6 & 8 & 7 & 8 \\ -11 & -12 & -15 & -14 \end{bmatrix}$$

on the stack and execute **CHAR**. You should receive output { 1 5 13 19 10 }, meaning that the eigenvalue equation is $\lambda^4 + 5\lambda^3 + 13\lambda^2 + 19\lambda + 10 = 0$. This equation has two real zeros, $\lambda = -1$ and $\lambda = -2$. Dividing $\lambda^2 + 3\lambda + 2$ into the eigenvalue equation gives a factor $\lambda^2 + 2\lambda + 5$ so $\lambda = -1 \pm i$ are two remaining eigenvalues. The procedure given above for 3 by 3 matrices extends to n by n matrices and therefore a fundamental matrix of solutions is

$$Y(t) = \begin{bmatrix} 4e^{-t} & 0 & e^{-t}\cos 2t & e^{-t}\sin 2t \\ e^{-t} & e^{-2t} & 0 & 0 \\ -2e^{-t} & 0 & -e^{-t}\sin 2t & e^{-t}\cos 2t \\ -2e^{-t} & -e^{-2t} & e^{-t}(\sin 2t - \cos 2t) & -e^{-t}(\cos 2t + \sin 2t) \end{bmatrix}.$$

Recall that an HP-48G calculator has a program to obtain solutions of a polynomial equation. After obtaining the characteristic equation using **CHAR**, the roots of the equation can be determined using the program **PROOT** located in the **POLY** directory on the **SOLVE** menu. The output of **CHAR** is a list of the coefficients. This list should be converted to an array by using the keystrokes **PRG TYPE OBJ→ →ARRY** before using **PROOT**.

EXERCISES 4.7: Find a fundamental matrix of solutions for y' = Ay when

$$A = \begin{bmatrix} -2.5 & 2.5 & -3.5 & -.5 \\ 1 & 2 & 2 & 1 \\ 5 & 4 & 6 & 6 \\ -4.5 & 8.5 & -5.5 & -7.5 \end{bmatrix}.$$

We noted earlier that Hewlett Packard has provided professional programs to calculate the eigenvalues and eigenvectors of n by n matrices ($n \geq 2$). These programs are located by pressing the **MTH MATR NXT** keys. **EGV** determines the eigenvalues and eigenvectors of the matrix on level 1, **EGVL** determines only the eigenvalues.

EXAMPLE: Place the matrix

$$A = \begin{bmatrix} 3 & 4 & 6 & 4 \\ 1 & -1 & 1 & 1 \\ 6 & 8 & 7 & 8 \\ -11 & -12 & -15 & -14 \end{bmatrix}$$

on the stack and execute **EGV**. You should receive output [(-1, 2) (-1, -2) (-1, 0) (-2, 0)] for the eigenvalues and output for corresponding eigenvectors

$$\begin{bmatrix} (-.5, .5) & (-.5, -.5) & (1, 0) & (0, 0) \\ (0, 0) & (0, 0) & (.25, 0) & (1, 0) \\ (-.5, -.5) & (-.5, .5) & (-.5, 0) & (0, 0) \\ (1, 0) & (1, 0) & (-.5, 0) & (-1, 0) \end{bmatrix}.$$

The first two eigenvalues are complex -1 ±2i and have conjugate eigenvectors which are the first two columns of the matrix. The procedure given above for 3 by 3 matrices extends to n by n matrices and therefore a fundamental matrix of solutions is

$$Y(t) = \begin{bmatrix} 4e^{-t} & 0 & e^{-t}\cos 2t & e^{-t}\sin 2t \\ e^{-t} & e^{-2t} & 0 & 0 \\ -2e^{-t} & 0 & -e^{-t}\sin 2t & e^{-t}\cos 2t \\ -2e^{-t} & -e^{-2t} & e^{-t}(\sin 2t - \cos 2t) & -e^{-t}(\cos 2t + \sin 2t) \end{bmatrix}.$$

EXERCISE 4.8: Find a fundamental matrix of solutions for y' = Ay when

$$A = \begin{bmatrix} -2.5 & 2.5 & -3.5 & -.5 \\ 1 & 2 & 2 & 1 \\ 5 & 4 & 6 & 6 \\ -4.5 & 8.5 & -5.5 & -7.5 \end{bmatrix}.$$

Remark on EXERCISE 4.8. Suppose you use **EGV** on the HP-48G to obtain eigenvectors on stack level two and eigenvalues on level one. We note that the first eigenvalue is approximately 5.964. If we want to find an eigenvector corresponding to $\lambda = 5.964$ with fourth component equal to 1 we can proceed as follows: Bring the matrix of eigenvectors to stack level one and create a copy by pressing **ENTER**. Press **MTH MATR** and execute the command 1 **COL-**. You will now have the first column of the eigenvector matrix on level one (and the eigenvector matrix without column 1 on level two). Create a copy of this column by pressing **ENTER**. Then execute **4 GET** and / to bring the fourth element of the column to stack level one and to then divide the eigenvector by its fourth component to get [6.049, -8.871, -21.106, 1] as an eigenvector corresponding to the first eigenvalue.

Non-homogeneous Systems

If the functions in a vector f(t) are elementary, we can use the method of undetermined coefficients to construct a particular solution to y' = Ay + f(t). The computation of the coefficients will require the solution of linear algebraic equations. For more complicated functions f(t) to obtain solutions of the nonhomogeneous equation suppose that a fundamental matrix Y(t) of solutions for the associated homogeneous equation is known (so Y'(t) = AY(t)). It is easy to see that

$$y(t) = Y(t)\, c + \int_0^t Y(t-s)\, Y^{-1}(0)\, f(s)\, ds$$

is a solution of the nonhomogenous system for any vector c. In the general case a program that uses the numerical integration capability of the calculator can produce values at various times t for the components of the integral listed above.

EXAMPLE: Suppose A is the matrix given by

$$A = \begin{bmatrix} -.85 & .85 & -2.05 \\ .1 & 0 & -.9 \\ .55 & 1.5 & -.25 \end{bmatrix}$$

(a) We calculate the eigenvalues (one real and a pair of complex conjugate eigenvalues, viz. {-.786... , -.1568... ± i 1.527...}) and corresponding eigenvectors (v_1 = column {1, -.3096... , -.159..., }, and a ± ib = column{1, .467... ± i .1985... , -.144...± i -.827...} for the matrix and determine 3 independent vector solutions u(t), v(t), w(t) of the homogeneous system dy/dt = Ay. We take Y(t) = {$e^{\lambda 1}tv_1$, $e^{\alpha t}$(cos βt a – sin βt b), $e^{\alpha t}$(sin βt a + cos βt b)}, so that Y(0) = [v_1, a, b] . (To obtain this matrix put A on the stack, execute **EGV**, **SWAP** and **OBJ→**, then use the **OBJ→** to put the real and imaginary parts of the first two columns of the matrix of eigenvectors on the stack.

Finally use the (up-arrow)**STK** and the **ECHO** commands in the matrix editor to construct Y(0).)

(b) For simple forcing functions we can use the method of undetermined "coefficients" to find a particular solution: for example we can choose vectors α, β so that $y(t) = \alpha \cos 2\pi t + \beta \sin 2\pi t$ is a particular solution of

$$\frac{dy}{dt} = Ay + \begin{bmatrix} 2 \sin 2\pi t \\ 0 \\ -3 \cos 2\pi t \end{bmatrix}$$

as follows: we split the nonhomogeneous term into a column vector {2, 0, 0} *sin 2πt + a column vector *cos 2πt then by substituting the prescribed form for y into the differential equation we get the equations

$$2\pi\,\beta = A\,\alpha + \begin{bmatrix} 0 \\ 0 \\ -3 \end{bmatrix}, \quad -2\pi\,\alpha = A\,\beta + \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}.$$

If we use the first equation in the second equation we get

$$\alpha = -\text{ column } \{ .485\ldots , .08\ldots , .026\}, \beta = \text{column } \{ .0317\ldots, -.002\ldots , -.269\}.$$

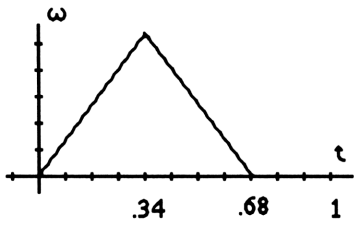
(c) For more complicated forcing functions we use the variation of parameters method to find a particular solution. Suppose

$$f(t) = \text{column } [\, 0, 0, \omega(t)]$$

where $\omega(t)$ = IFTE($t \le .34$, 5t , IFTE($t \le .68$, 1.7 - 5(t-.34), 0)) for $0 \le t \le 1$ and $\omega(t)$ is periodic with period 1. Then if we choose c so that $y(0) = y(1)$ in the equation for y(t) given above we will have a particular solution of **dy/dt = A y + f(t)** which is periodic with period 1. This gives the following equation for c (which forms part of the appropriate initial condition)

$$[\, Y(0) - Y(1)]\, c = \int_0^1 Y(1-s) Y^{-1}(0)\, f(s)\, ds$$

where for example, we let Y(t) = column [u(t), v(t), w(t)] and u, v, w were obtained in (a). A graph of the input function is shown below.



Third Component of Input Function

After obtaining the value of c, the initial condition Y(0) c will produce a periodic response over $0 \le t \le 1$.

To accomplish this computation we calculate numerically the integral values

$$EW = \int_0^1 e^{\lambda_1 (1-s)}\, \omega(s)\, ds = .346,$$

$$SEW = \int_0^1 e^{\alpha(1-s)} \sin \beta(1-s)\, \omega(s)\, ds = .430,$$

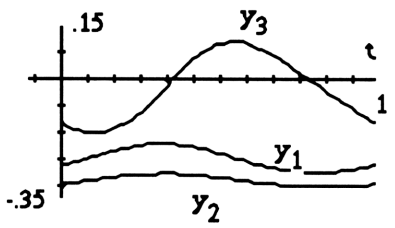
$$CEW = \int_0^1 e^{\alpha(1-s)} \cos \beta(1-s)\, \omega(s)\, ds = .2738.$$

Then

$$\int_0^1 \omega(s)\, Y(1-s) \begin{bmatrix} .310.. \\ -.310.. \\ -1.214.. \end{bmatrix} ds = .310..EW\ v_1 - .310..(CEW\ a - SEW\ b)$$

$$-1.214\ (SEW\ a + CEW\ b)$$

which we label as RHS. The equation {Y(0) – Y(1)} c = RHS gives c = [.19735, -.45535, .19735] and the appropriate initial condition for the periodic solution is Y(0) c = [-.258, .-313, -.129]. A graph of the resulting components is shown.



Components of Periodic Solution

As noted above, the change in the third component is pronounced, whereas the other components are influenced to a lesser extent to the change in $\omega(t)$ at t = .34 and .68.

EXERCISE 4.9: Suppose that f(t) = $\omega(t)$ column { 0, 1, 0 } in the problem given above. Obtain a graph of the resulting periodic solution.

EXERCISE 4.10. Use the method of undetermined coefficients to determine a particular solution of

$$y' = \begin{bmatrix} -3 & 1 & 0 \\ 0 & -3 & 1 \\ 4 & -8 & 2 \end{bmatrix} y + e^{-2t} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Hint: Try a solution of the form $y = e^{-2t}\{t\,a + b\}$ where vectors a and b are to be determined. By substitution the requirements for vectors a and b are Aa = -2a and A b = -2 b + a – column {1, 1, 0}. We choose a = column {1, 2, 4}, then b = column{σ, σ, σ} – column {1, 1, 0} for any σ. The reader might ponder the case where the nonhomogeneous term has the form e^{-2t} column $\{g_1, g_2, g_3\}$ in the case $4(g_1 - g_2) \neq g_3$.

EXERCISE 4.11: A linear model for the angular displacements in a double pendulum (see reference in Chapter 5: The Differential Equation Problem Solver) for a system

of two pendulums with lengths l_1, l_2, a ratio of pendulum masses given by $\delta = 1 + m_2/m_1$ is given by the differential equations

$$m_1 l_1 \theta_1'' = g\{m_2 \theta_2 - (m_1+m_2)\theta_1\}$$

$$m_2(l_1\theta_1''+l_2\theta_2'') = -m_2 g \theta_2.$$

These equations may be recast as

$$l_1\theta_1'' = g\{m_2/m_1\,\theta_2 - \delta q_1\}$$

$$(l_1\theta_1''+l_2\theta_2'') = -g\,\theta_2$$

and by standard elimination techniques, we obtain

$$l_1 l_2\,\theta_1^{(iv)} + g(l_1+l_2)\delta\,\theta_1'' + g^2\delta\,\theta_1 = 0$$

$$\theta_2 = \frac{1}{\delta-1}\left\{\frac{l_1}{g}\theta_1'' + \delta\,\theta_1\right\}.$$

We note $\delta > 1$. Show that for $\theta_1 = e^{rt}$ the characteristic equation is quadratic in r^2; viz.

$$r^2 = \frac{g\delta}{2l_1\tau}\left\{-(1+\tau) \pm \sqrt{(1+\tau)^2 - \frac{4\tau}{\delta}}\right\}$$

where τ is defined by the equation $l_2 = \tau\, l_1$. Test these values of r^2 for values of τ and δ by putting $\delta = 1.5$ (i. e. $m_2 = .5\, m_1$) and graph the term in brackets for τ in the interval $0 \le \tau \le 1$. These graphs show the roots for r^2 are negative and thus there are two pairs of conjugate purely imaginary roots for r. If we denote the roots of the characteristic equation as $r = \pm i\,\mu$, $r = \pm i\,\nu$, show the solution θ_1 and θ_2 will contain terms of the form

$$a_j \cos \mu t + b_j \sin \mu t + c_j \cos \nu t + d_j \sin \nu t,\ j = 1,2.$$

with

$$a_2 = \left[\delta - \frac{l_1 \mu^2}{g} \right] a_1 \qquad b_2 = \left[\delta - \frac{l_1 \mu^2}{g} \right] b_1$$

$$c_2 = \left[\delta - \frac{l_1 \nu^2}{g} \right] c_1 \qquad d_2 = \left[\delta - \frac{l_1 \nu^2}{g} \right] d_1$$

With appropriate initial conditions, we can find terms of the form cos μt – cos νt which leads to the form sin {.5(μ+ν)t }sin{.5(μ–ν)t}.

Alternately we put $y_1 = \theta_1$, $y_2 = \theta_1'$, $y_3 = \theta_2$, $y_4 = \theta_2'$ to derive the equation y' = Ay where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{\delta g}{l_1} & 0 & \frac{g(\delta-1)}{l_1} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{g\delta}{\tau l_1} & 0 & -\frac{g\delta}{\tau l_1} & 0 \end{bmatrix}$$

For g = 1, l_1 = .75, $\tau = \delta - 1$ = .10059.. , the matrix A has eigenvalues $\lambda = \pm i\,\mu$, $\lambda = \pm i\,\nu$ with μ = 4.53817..., ν = 1.13454.. and corresponding eigenvectors

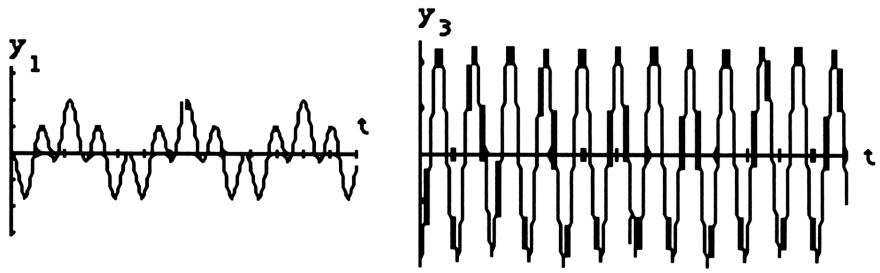
$$\begin{bmatrix} 0 \\ -.035852.. \\ 0 \\ 1 \end{bmatrix} + i \begin{bmatrix} .0079001.. \\ 0 \\ -.22035.. \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ .93526.. \\ 0 \\ 1 \end{bmatrix} + i \begin{bmatrix} -.82435.. \\ 0 \\ -.8814.. \\ 0 \end{bmatrix}$$

We denote the vectors listed above as a + i b, c + i d. There is a solution of y' = Ay

$$y(t) = \sigma \{ \sin \mu t\, a + \cos \mu t\, b - b(1) (\sin \nu t\, c + \cos \nu t\, d)/d(1) \},$$

and we note the first component has the form $y_1(t) = \sigma\, b(1)$ {cos μt – cos νt}. Thus the angular displacement of the first pendulum oscillates with a beats motion. It is

interesting to plot the motion of the angular displacement of the second pendulum which looks deceptively like a regular periodic motion of a sine wave.



Motion of pendulum 1 Motion of pendulum 2

5 MISCELLANEOUS SYSTEMS

This chapter contains a set of applied problems that are appropriate in the study of differential equations. The arrangement of the problems is somewhat random and many, if not most, of the problems can be studied as soon as the concept of solving a vector initial value problem is discussed. The first two problems might occur more naturally just after chapter three. We delayed the presentation of these problems so that students could keep the time schedule used in many classes. Some classes use a nonstandard order of topics. For example, a class may choose to study discrete systems early in the course. This chapter includes a short discussion of such systems, but covers only an introduction to the concept of chaos.

Consider the motion of a particle in two dimensions (x and y) in a gravitational field. We assume that the height attained by the particle is relatively low so that gravity is constant, and motion occurs in a plane. Put

$$v(t) = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}, \qquad \theta(t) = \tan^{-1}\frac{dy}{dx}(t).$$

The force on the projectile in the direction of the tangent to the trajectory is $F_1 = T(t) - av^r$ (thrust and air resistance). There are also horizontal and vertical forces. The equations resulting from Newton's law of motion are:

$$m\frac{d^2x}{dt^2} + \frac{dm}{dt}\frac{dx}{dt} = F_1\cos\theta - \omega, \quad m\frac{d^2y}{dt^2} + \frac{dm}{dt}\frac{dy}{dt} = F_1\sin\theta - mg$$

Here ω is a force in the horizontal direction such as wind or an artifically imposed control on the trajectory as described below for rocket flight. Since

$$\frac{dx}{dt} = v\cos\theta, \quad \frac{dy}{dt} = v\sin\theta, \tag{5.1, 5.2}$$

$$\frac{d^2x}{dt^2} = \frac{1}{m}(T(t) - av^r)\cos\theta - \frac{\frac{dm}{dt}}{m}\frac{dx}{dt} - \frac{\omega}{m}$$

$$= dv/dt\cos\theta - v\sin\theta\, d\theta/dt$$

$$\frac{d^2y}{dt^2} = \frac{1}{m}(T(t) - av^r)\sin\theta - \frac{\frac{dm}{dt}}{m}\frac{dy}{dt} - g$$

$$= dv/dt\sin\theta + v\cos\theta\, d\theta/dt$$

Multiplying the right side of the x acceleration expression by cos θ and the right side of the y acceleration expression by sin θ and adding gives

$$\frac{dv}{dt} = \frac{1}{m}(T(t) - av^r) - \frac{\frac{dm}{dt}}{m}v - g\sin\theta - \frac{\omega}{m}\cos\theta \tag{5.3}$$

If we differentiate the expression for θ with respect to t we obtain

$$\frac{d\theta}{dt} = \frac{1}{v}\left(\frac{\omega}{m}\sin\theta - g\cos\theta\right). \tag{5.4}$$

Differential equations (1-2), (3) and (4), together with initial conditions x(0)=y(0) = 0, and v(0) , θ(0) prescribed, will give the trajectory (x(t), y(t)). The reader may note that when the initial velocity is low the initial values of θ will be quite sensitive to the v(0) value.

EXERCISE 5.1 : (*Artillary shell EXERCISE for the HP-48G)* Take r = 2, m = 500 kg, a = .04, ω = 0 and initial condition Y = [θ, v, x, y] = [θ, 200, 0, 0] for several values of

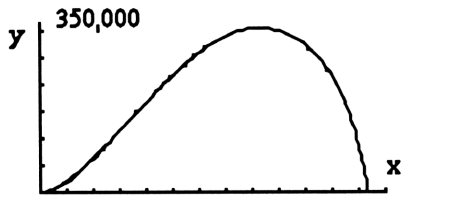
θ, say between .5 and 1.2. At each elevation, determine the range and maximum height of the shell.

EXERCISE 5.2: (*Baseball trajectory EXERCISE for the HP-48G*) Take r = 1.2, a = .035, m = .25 kg, and v(0) = 50 meters /second. Determine the trajectories that start at x = 0, y = 0 and terminate when the ball hits the ground for selected values of θ and $\omega = 0$, then for $\omega = \pm 4$ meters/second.

Rocket flight example for the HP-48G: We take r = 2, m_0 = 90,000 kg, $M(t) = m_0 - \text{slope } t$ for $0 \le t \le 120$, $= .1m_0 + .05 * m_0 e^{-\gamma(t-120)}$ for $t > 120$ where slope = $.85\, m_0/120 = 637.5$ kg/sec, and $.05 m_0 \gamma = 637.5$ or $\gamma = .85/.05 * 1/120 = .14167$ and a = .05. A reasonable model for thrust is T(t) proportional to dM(t)/dt, but some easy experiments will convince the reader that unless some control is exerted on the horizontal direction, say in the form of providing thrust in this direction, the rocket will soon tilt into the ground with unburned fuel. To prevent this from happening we take $\omega = -\beta \varepsilon\, dM(t)/dt \cos\theta$ and $T(t) = -\beta\, dM(t)/dt\, (1 - \varepsilon \cos\theta)$. Further we assume that special conditions are placed on the system so that the rocket lifts off and achieves a velocity of 10 meters per second as it clears the liftoff tower. From this point we assume the equations of motion given above apply. For $\varepsilon = .825$ and initial vector Y with components θ, v, x, y having values [1.5, 10, 0, 0] and XRNG 0 11,000, YRNG 0 350,000, we observe the following data:

t = 120,	θ = 1.555,	v = 3,178,	x = 1,666,	y = 67,800
t = 240,	θ = 1.542,	v = 711,	x = 5,758,	y = 316,279
t = 300,	θ = 1.302,	v = 67,	x = 6,878,	y = 338,756
t = 306,	θ = .3,	v = 13,	x = 7,000,	y = 338,960
t = 654.7,	θ = -1.569,	v = 1313,	x = 10416,	y = 150

The graph of the rocket's trajectory is shown below.



Rocket trajectory

EXERCISE 5.3 : Take ε = .8, the initial Y to be [1.5, 10, 0, 0], and plot the trajectory. Record the values of Y at times t = 120, 240, 300, 660. Then take ε = .8125, initial Y = [1.5, 10, 0, 0] and observe the values of Y for successive time steps to see whether the control will give permit to a full trajectory, i.e., to a trajectory with flight termination after fuel burnout (approximately 150 seconds). To do this place the calculator in **2 FIX MODE**, place **{T Y FN}** on stack level five, a tolerance .005 in level four, starting step size, say .01 in level three, 0 on level two and [1.5 10 0 0] on level one, and execute the program

<< 'T' STO 'Y' STO RKFSTEP T Y >> .

Repeat execution of this program for several times to see that the first component of Y shown on level one remains near 1.5 and the second component increases at each execution.

EXERCISE 5.4 (Pursuit Problem): A rabbit starts at (0,1) and runs along y = 1 with speed 1. At the same time a dog starts at (0, 0) and pursues the rabbit with speed 1.3. The dog attempts to point at the rabbit at all times but is constrained by his momentum. That is, for the angle z between the dog's direction and the x axis, dz/dt is restricted. What is the path of the dog? The equations of motion are:

$$dx/dt = 1.3 \cos z, \quad dy/dt = 1.3 \sin z, \quad dz/dt = -(z - \theta(t,x,y))$$
$$x(0) = 0, \quad y(0) = 0, \quad z(0) = 2.2.$$

Here $\theta(t,x,y)$ is the angle between the dog-rabbit vector and the x axis. We take **H** = .15, **N** = 60, plot parameters to show $-1 \le x \le 8$, $-.5 \le y \le 2$, repeatedly apply **IULER** to the differential equation $dw/dt = F(t,w)$ and watch the trajectory. Suitable functions for $F_1(t, x, y, z)$, $F_2(t, x, y, z)$, and $F_3(t, x, y)$ (here F(t, w) = column $[F_1, F_2, F_3]$) are given by:

```
F.N1:  << → T W '1.3* COS(W(3))' >>
F.N2:  << → T W '1.3* SIN(W(3))' >>

F.N3: << → T W << 'W(3)' EVAL T 'W(1)' EVAL - DUP SQ 'W(2)' EVAL
1 - DUP SQ 3 ROLL + √ 3 ROLLD SWAP DUP 3 ROLLD THTA - NEG >>
THTA: << 0 IF ≥ THEN THT1 ELSE THT2 END >>

THT1: << 0 IF ≤ THEN SWAP / ACOS ELSE SWAP / ACOS NEG END >>

THT2: << 0 IF < THEN NEG SWAP / ACOS π →NUM
SWAP - ELSE NEG SWAP / ACOS π →NUM + END >>

F.N: << DUP2 DUP2 F.N1 5 ROLLD F.N2 3 ROLLD F.N3 3 → ARRY >>
```

An easy modification of the **GRAF** program can be used to follow the action. The program requires a starting stack of 0, [0, 0, 1.8] and the trajectories of both dog and rabbit are shown dynamically as follows:

```
<< { # 0d # 0d } PVIEW 0 1 R→C 3 ROLLD DRAX 1 N START IULER
DUP OBJ→ DROP2 R→C PIXON 3 ROLL H 0 R→C + DUP PIXON 3
ROLLD NEXT PICTURE >>
```

A Nonlinear Problem: the Lorentz Equations

Consider the problem:

$$dx/dt = \sigma(y-x),\ dy/dt = (r-z)x - y,\ dz/dt = xy - bz.$$

where σ, r and b are parameters. This set of equations was proposed by E. Lorentz (1963) in connection with convective heat transfer between the earth's surface and the atmosphere. A point (x, y, z) represents convection velocities and temperature profile, vertical and horizontal. An equilibrium, or periodic, solution to the system represents predictable behavior. The graphs of particular solutions gave particularly surprising results because most trajectories never seem to approach such predictability. The paper has been one of the seminal studies in the area of chaos.

First the three critical points (that is, points (x, y, z) where the right sides of the differential equations are zero) are (0, 0, 0) and $(\pm(b(r-1))^{.5}, \pm(b(r-1))^{.5}, r-1)$. The variational matrix at (0, 0, 0) is $\begin{bmatrix} -\sigma & \sigma & 0 \\ r & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$, with system eigenvalues of -b and $.5(-(\sigma+1) \pm \sqrt{(\sigma-1)^2 + 4\sigma r}\,)$.

The variational matrices near the remaining critical points are

$$\begin{bmatrix} -\sigma & \sigma & 0 \\ 1 & -1 & -\delta \\ \delta & \delta & -b \end{bmatrix} : \delta = \pm\sqrt{b(r-1)}\,.$$

Here, the eigenvalues satisfy the equation

$$\lambda^3 + (\sigma+b+1)\lambda^2 + b(r+\sigma)\lambda + 2\sigma b(r-1) = 0.$$

For $r > 1$, $b > 0$, $\sigma > 0$ one of the variational eigenvalues near (0, 0, 0) is positive, so most trajectories starting near (0, 0, 0) leave that neighborhood. For r sufficiently

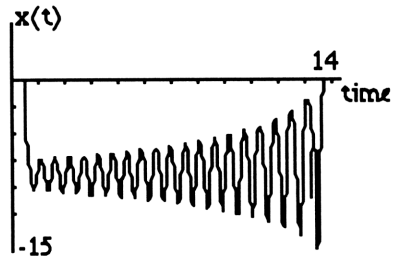
near 1, the eigenvalues of the remaining variational matrices are negative and the corresponding critical points are attracting solutions. When r is slightly larger, one of the variation eigenvalues has a positive real part and the critical point is repelling. It can be shown that solutions starting near the critical points stay bounded and thus the trajectories have interesting behavior as t increases.

EXERCISE 5.5: Compute a solution for initial values $x(0) = 1$, $y(0) = 1$, $z(0) = 1$ for $\sigma = 10$, $r = 28$, and $b = 8/3$ using the built-in program for $0 \le t \le 15$. Use viewing box $-15 \le x \le 0$, $-15 \le y \le 0$, $20 \le z \le 40$ and position of the eye along the vector [1, 1, 5]. (See Appendix 5 for programs that give three dimensional plots.) Part of a typical trajectory is shown below. A continuation of the trajectory reveals no asymptotic pattern other than a continual looping in two of the x, y, z octants. The following table shows partial results rounded to two decimals. At time t approximately .7, the trajectory enters the viewing box, starts in a tight spiral that winds outward, finally leaving the quadrant at about t = 14.4.

time	x	y	z
.608	-5.07	-8.06	26.44
3.90	-8.67	-10.42	24.76
5.36	-9.87	-8.19	30.65
7.68	-8.80	-11.43	23.55
9.45	-5.24	-6.22	21.38
11.48	-4.79	-6.82	18.16
13.12	-13.16	-18.40	26.71
13.81	-6.50	-.12	8.40
14.37	-15.96	-25.44	26.07
14.55	-5.18	6.93	35.46
14.68	3.76	6.90	25.36

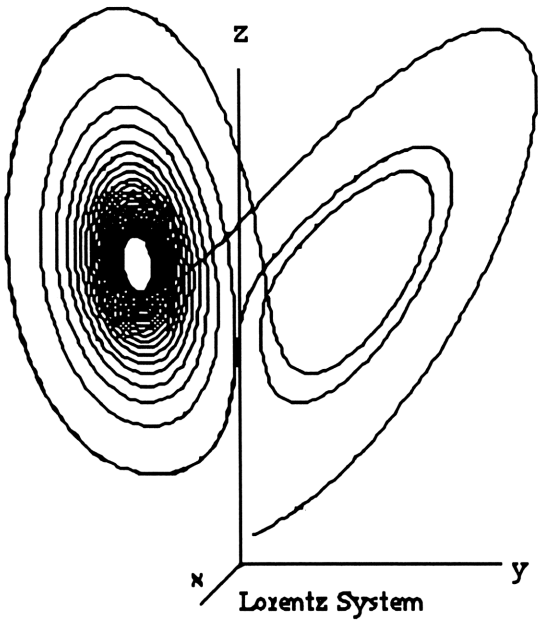
xyz perspective

Lorentz trajectory enters viewing window near center of picture winds outward to t = 14



x-t perspective

An extension to later times is shown below. Subsequently the trajectory winds in the right hand portion of the picture making excursions into the left portion. The trajectory never intersects inself so there is layering going on. *Warning*: the development of this trajectory takes considerable time on the HP-48.



A Nonlinear Problem: Earth, Moon, Satellite Motion

Consider the model problem where the moon is circling the earth and a satellite is in motion in the plane of the earth-moon orbit. The "restricted three body" problem results when the satellite mass can be ignored when compared to the masses of the moon and earth. (See *Celestial Mechanics*, Part II by S. Sternberg, W. A. Benjamin Company, 1969.) If a rotating coordinate system is used so the coordinates of the earth are $(-\mu,0)$, the coordinates of the moon are $(1-\mu,0)$, and the coordinates of the satellite are denoted by $(x(t), y(t))$, then the equations of motion are

$$\frac{dx}{dt} = u, \quad \frac{du}{dt} = 2v + x - \frac{(1-\mu)}{r^3}(x+\mu) - \mu\frac{(x+\mu-1)}{\rho^3}$$

$$\frac{dy}{dt} = v, \quad \frac{dv}{dt} = -2u + y - \frac{(1-\mu)}{r^3}\, y - \mu\frac{y}{\rho^3}$$

where

$$r^2 = (x+\mu)^2 + y^2, \quad \rho^2 = (x+\mu-1)^2 + y^2$$

The constant μ is a ratio of the masses of the earth and moon $(= m_p/(m_p+m_m) = 1/82.45)$. For the "earth-moon" system it has been discovered that a solution with period approximately 6.1922 results from the initial conditions

$$x(0) = 1.2 \qquad u(0) = 0, \qquad y(0) = 0, \qquad v(0) = -1.04936\ldots .$$

We take the vector w = [x, u, y, v] and create the subprogram

Subprogram Name	**RN**
Stored Quantities	none
input w:	output: **R1** = r^3 **R2** = ρ^3

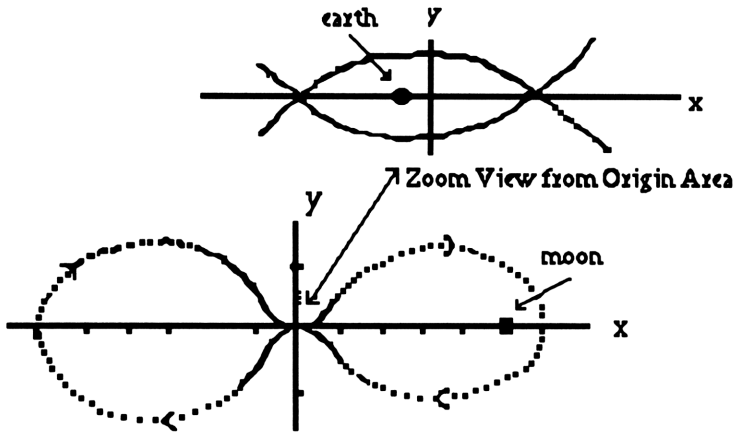
```
<< OBJ→ DROP2 SWAP DROP SQ SWAP 82.45 INV + DUP2
        SQ + 1.5 ^ 3 ROLLD 1 - SQ + 1.5 ^ >>
```

The program **F.N** that takes w to F(w) can be as follows:

```
<< DUP RN 82.45 INV → R1 R2 MU << OBJ→ DROP 3 ROLLD MU DUP
R2 / SWAP 1 - R1 / - 1 - NEG * SWAP DUP 5 ROLLD 2 * - 3 ROLLD
SWAP DUP MU + 1 - R2 / MU * SWAP DUP 3 ROLLD MU + 1 MU -
R1 / * + - SWAP DUP 2 * 3 ROLL + 3 ROLLD SWAP 4 →ARRY SWAP
DROP >> >>
```

Since this program is complex, we provide a stack status at various program steps:

Instruction	Stack contents
DUP RN 82.45 INV	t w R1 R2 MU
→ R1 R2 MU	t w
OBJ→ DROP	t x u y v
3 ROLLD MU DUP R2 /	t x v u y μ μ /R2
SWAP 1 - R1 /	t x v u y μ /R2 (μ −1)/R1
- 1 - NEG *	t x v u y*(-1)[μ /R2-(μ −1)/R1 -1]
SWAP DUP 5 ROLLD 2 * -	t u x v {y*(-1)[μ /R2−(μ −1)/R1 -1]−2u}
3 ROLLD SWAP	t u last equation v x
DUP MU + 1 - R2 / MU *	t u last equation v x μ[x+μ−1]/R2
SWAP DUP 3 ROLLD	t u last equation v x μ[x+μ−1]/R2 x
MU + 1 MU - R1 /	t u last equation v x μ[x+μ−1]/R2 x+μ (1−μ)/R1
*** + - SWAP DUP**	t u last equation x-μ[x+μ−1]/R2+(x+μ) (1−μ)/R1 v v
2 * 3 ROLL + 3 ROLLD SWAP 4 →ARRY SWAP DROP >>	



Satellite near Earth-Moon System

The graphs shown above require considerable execution time. The tolerance parameter in the variable stepsize method was initially set to 0.01 and adjusted when the satellite approaches the earth: later it was reduced back to 0.01, etc.

Another interesting periodic solution occurs for the approximate initial conditions

$$x(0) = 0, \qquad u(0) = 1.58, \qquad y(0) = 1.2, \qquad v(0) = 0.$$

We note that equilibrium points for the restricted three body satisfy

$$u = v = 0, \quad y\,[\,1 - (1-\mu)/r^3 - \mu/\rho^3] = 0, \quad x = (1-\mu)(x+\mu)/r^3 + \mu(x-1+\mu)/\rho^3.$$

PROJECT: Find the equilibrium solutions of this system and determine their stability properties. Notice that for $y = 0$, we need only solve an equation of the form $g(x) = 0$, but for other solutions, we need to find the solution of a pair of nonlinear algebraic equations. We can use Newton's method as presented in Chapter 3 for finding such values of x and y if we can find a suitable starting point x_0, y_0. (Two of the equilibrium points are located at $x = .48787$, $y = \pm .86603$.) To determine

the stability, we need to find the eigenvalues of the 4 by 4 variational matrices associated with each critical point solution.

Discrete Dynamical Systems

In some situations, the problem of interest is to determine information concerning the asymptotic behavior (i.e., for large values of t) of solutions. This occurs often in differential equations; however we will illustrate this type of problem by introducing a new type of problem.

The sequence $\{ y_n \}_0^\infty$ where y_n is given by a recursive function of the form $y_{n+1} = F(y_n)$, is called a **discrete dynamical system**. Euler's method, the improved Euler method, and Newton's method for finding roots may give such systems. Concepts such as constant "solutions", attractive or repelling solutions taken from differential equations, are also present in the study of such systems. Discrete dynamical systems (in one dimension) have solutions with more complicated structure than do differential equations. For example, y_n = a constant (for all n) is called a period one solution, $y_{odd\ n} = \alpha$ and $y_{even\ n} = \beta$ is a period two solution, $y_{3m} = \alpha$, $y_{3m+1} = \beta$, $y_{3m+2} = \chi$ is a period three solution, etc. Consider

$$y_{n+1} = (1 + a)\, y_n - a\, (y_n)^2 \text{ and } y_0 = .1.$$

We want to regard a as a parameter and study the effect on the "solution sequence $\{ y_n \}$" as a is varied. In particular we want to study a = 1.8, 2.3, 2.5 and 3 as well as nearby values of a.

After several numerical experiments, we find that for a = 1.8 the terms of the sequence approach 1, but for a = 2.3, the terms of the sequence approach 1.18 for even n and .69 when n is odd, etc. This leads us to the following graphical program:

Set the value of a. Calculate the first 50 terms, then plot the values of the next 100, then change the value of a and repeat. To get these cases on a single plot, we plot the values of y_n on the horizontal axis and the values of a on the vertical axis. The following programs can be used: Store **A** = 1.8 and **N** = 100, set the plot parameters so that $-.5 \le x \le 1.5$ and $1.6 \le y \le 3.2$, then enter the programs.

Program Name	**DDS**
Purpose	Plot the asymptotic value of a discrete system for several values of the parameter a
Stored Quantities	**FN1 DDS1 DDS2 A N XRNG YRNG**
No input:	Output is a graph

```
<< { # 0d # 0d } PVIEW .1 1 12 START DDS1 A .1 + 'A'
STO .1 NEXT PICTURE >>
```

Subprograms are:

Subprogram Name	**FN1**
Purpose	Create the new value of y, given the previous value

```
<< DUP SQ A * SWAP A 1 + * SWAP – >>
```

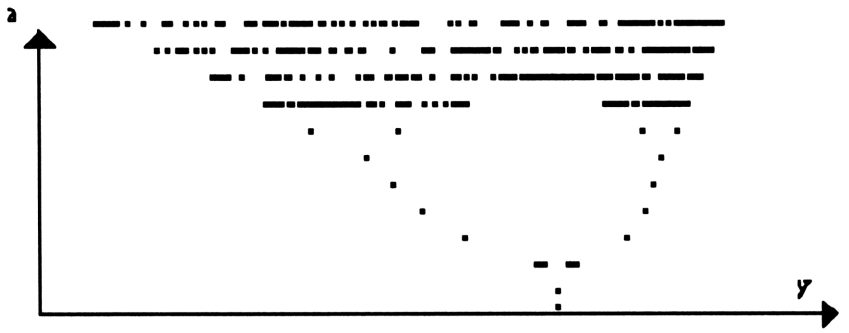
Subprogram Name	**DDS1**
Purpose	Execute **FN1** 50 times, call **DDS2**

```
<< 1 50 START FN1 NEXT DDS2 >>
```

Subprogram Name	**DDS2**
Purpose	Repeat **FN1** **N** times, plot points

<< 1 N START FN1 DUP A R→C PIXON NEXT >>

(The 50 executions of **FN1** without graphing allows the sequence to come to "steady state" before the graphing begins.) Execute **DDS**.



Discrete Dynamical System

EXERCISE 5.6. Change **FN1** to repeat the process for the system

$$y_{n+1} = (1 + a)\, y_n + 2a\, (\cos y_n - 1) \text{ and } y_0 = .1.$$

An appropriate range of the parameter a begins at 2.0.

The objects in a discrete dynamical system may be vectors. One example was discussed (without labeling it as a discrete dynamical system) in EXERCISE 3.5. Particular attention has been given to the case when the y_n are two-dimensional vectors. In this case we can easily plot the vectors (y_{n1}, y_{n2}), n = 1, 2, ... on a graph. In addition to the cases of an attracting or repelling equilibrium solution, other types

of interesting behavior may occur. The reader should consult [1] or [2] for more details. We will present the example, called the Henon map, in which the point (x, y) is "mapped" to the point

$$\overline{x} = x \cos \alpha - (y - x^2) \sin \alpha$$
$$\overline{y} = x \sin \alpha + (y - x^2) \cos \alpha$$

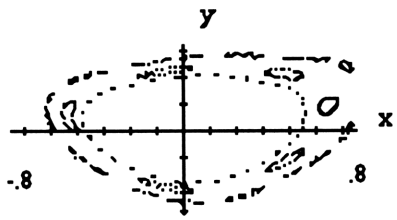
Here α and the starting values (x_0, y_0) are parameters. For fixed α, the trajectory (x_n, y_n), n =0, 1, 2, ... may have significantly different structure for different values of the initial point. Usually trajectories arising from several initial points are shown on the same plot. And different values of a may result in quite different plots. In our example we take $\alpha = \text{Cos}^{-1}(.4)$. We modify the **GRAF** program from Chapter 1 to a program called **GRF.H** as follows:

```
<< { # Θd  # Θd } PVIEW DRAX DUP2 R→C PIXON 1 N START HMAP
                DUP2 R→C PIXON NEXT PICTURE >>
```

where **HMAP** is given by

```
<< → X Y << ' X*COS(ALFA) – (Y – X^2)*SIN(ALFA)' EVAL
        ' X*SIN(ALFA) + (Y – X^2)* COS(ALFA)' EVAL >>
```

If we set **XRNG** and **YRNG** each to -.8 .8 and use **N** = 400, x_0 = .63, y_0 = .2 and execute **GRF.H** we obtain a curious picture with six islands. Repeating the program with x_0 = .55, y_0 = 0, then with x_0 = .75, y_0 = 0 give two "closed" curves. (The outer curve seams to contain another island.) These three trajectories are shown on the following plot:



Henon Map for cos α = .4

EXERCISE 5.7: Set $\alpha = \text{Cos}^{-1}(-.05)$, take initial points (.32, .9), (.30, .9), (.28, .9), (.30, 0) and (.2, 0) with **XRNG** = **YRNG** = -1 1: execute **GRF.H**.

Next we consider here the case $z_{n+1} = f(z_n, c)$, n = 0, 1, 2, . . . for given z_0, where all the z elements are complex numbers (a two component vector with special algebraic rules) and the complex parameter number c is given. **The purpose is again to focus attention on the asymptotic behavior of the dynamical system solution sequence $\{z_n\}$.** In particular, we will study systems of the form $z_{n+1} = z_n^2 + c$, z_0 given, where c will be fixed in each system. For each number c depending on the starting position z_0, one of three things can happen: (1) $\lim |z_n| = \infty$, (2) $\lim |z_n|$ = some number, or (3) neither of the above. We show the dependence of the elements of the system on the starting value $z_0 = \alpha$ by using the notation $z_n(\alpha)$ for the elements. **The "Julia" set for this sequence** is the boundary of the set $A = \{ \alpha: |z_n(\alpha)| \rightarrow \infty \}$. (Elements on the boundary of this set do not belong to the set.) Many of the elements of the Julia set have a wandering property, that is, they do not have a limit and so their behavior is called chaotic. (Iterates $f(\alpha)$, $f(f(\alpha))$, $f(f(f(\alpha)))$, . . . wander around the Julia set.)

The following program is to graph members of the Julia set. The procedure of finding lots of members of this set may seem tricky at first because the requirement to be in the set is quite delicate and any roundoff error may cause a sequence α, $f(\alpha)$,

$f(f(\alpha))$, $f(f(f(\alpha)))$, ... beginning with α in the Julia set to drift out of the set. The algorithm to be given is based on the property that for any fixed c, the inverse images of a repelling fixed point belong to the Julia set, that is, for w in the set, the images $f^{-1}(w)$, $f^{-1}(f^{-1}(w))$, $f^{-1}(f^{-1}(f^{-1}(w)))$, ... also belong to the set. (Here $f^{-1}(w) = \pm\sqrt{(w-c)}$.) It can be shown that the latter sequence is stable for this function f, whereas the sequence of direct images is not stable to roundoff error.

Suppose the mapping $z \rightarrow f(z) = z^2 + c$ has has a fixed point, that is $z^2 + c = z$. There are two such values of z, namely

$$z = \frac{1 \pm \sqrt{1-4c}}{2}$$

The student should obtain this number z or several values of the complex number c to see how the HP-48 handles complex square roots and to verify that the absolute value $|f(\text{one of these two values of } z)| > 1$. Such a value of z is called a repelling fixed point of the mapping f. (Complex numbers w near such z have the property that f(w), f(f(w)), f(f(f(w))), ... get further and further from w.)

We call the repelling fixed point located by z, compute and graph members of the sequence $f^{-1}(z)$, $f^{-1}(f^{-1}(z))$, $f^{-1}(f^{-1}(f^{-1}(z)))$, ... It can be shown that all such complex numbers satisfy $|w| \leq (1+\sqrt{(1+4|c|)})/2$. For a complex number w since there are two inverse images, viz. $\pm\sqrt{(w-c)}$, the particular sequence of inverse iterates chosen involves a random choice of $\pm$.

Choose and store a complex number **C1**. To set the drawing screen and compute the repelling fixed point z we execute the following subprogram, named **PREP**.

```
<< C1 DUP ABS 4 * 1 + √ 1 + 2 / DUP NEG SWAP DUP2
                XRNG YRNG 4 * 1 SWAP - √ 1 + 2 / >>
```

The output is a fixed point of f. If the output has absolute value larger than .5 call the number Z, if not put Z = 1 – the output. Store **Z** and execute the program **BACK** given by

<< { # 0d # 0d } PVIEW Z BCK1 >>.

Here **BCK1** is

<< 1 500 START C1 – √ ONE * DUP PIXON NEXT DROP PICTURE >>

and the subprogram **ONE** is given by

<< (1,0) IF RAND .5 < THEN NEG END >>.

For c = (-.12256, .7449), the following "graph" results

Douady's Rabbit: c = (-.123, .745)

EXERCISE 5.8: Execute the program sequence given above for c = (-1,0), c = (-.5, -.1).

Parameter Identification Problems Revisited

Suppose { (t, y) } data is given for the solution of an initial value problem and we wish to determine appropriate values of a parameter vector w = [p, q, a, b, ..] in a function y = g(t, w) to fit the data, say in a least squares sense. That is, we want to choose w to minimize the sum

$$\sum_{i=1}^{N} [\, y_i - g(t_i, w)]^2 .$$

By taking partial derivatives with respect to the components of w and setting them to 0 we obtain equations

$$\sum_{i=1}^{N} [\, y_i - g(t_i, w)] \frac{\partial g}{\partial w_k} (t_i, w) = 0, \quad k = 1, 2, \ldots M.$$

We take the left sides of these equations as components of a vector F, and attempt to solve the vector F(w) = 0. We will assume we have a starting values for the parameter vector w and give an iterative process.

The reader should review the algorithm we developed in chapter three for finding a solution of a vector equation F(w) = 0. In this case also we assume there are m parameters and w will be the m vector of parameters and F will be the m vector given above involving the partial derivatives. Just as before we assume the components of the function F are smooth, and we have an approximate solution w_0, so that Taylors theorem gives the approximate formula

$$F(w) = F(w_0) + J(w_0)\,(w - w_0)$$

where the matrix J has i, j element $\partial F_i / \partial w_j$. If w is to be a good approximation of the solution, the left side of this equation is zero and we get a "formula" for an improved vector solution w in terms of the old approximate w_0. Just as in chapter three we will provide an algorithm to construct and evaluate the function F(w) and the associated Hessian matrix.

Here is an outline the problem: First we create calculator programs for

$$[\, y - g(t,w)\,] \frac{\partial g}{\partial w_k} (t, w)\ , \ k = 1, 2, \ldots, M$$

then we will use the program called **DER** created in chapter three for finding the derivatives of these functions with respect to $p = w_1$, $q = w_2$, $a = w_3$, $b = w_4$, etc.

After execution, the derivatives can be used to create terms in the Hessian matrix J used in Newton's method.

Next we form the list { (t_1, y_1), (t_2, y_2), . . . , (t_n, y_n) } by entering the number pairs on the stack, then entering n and the command →**LIST** and store this as **DTA1**.

Finally we create programs called **JACM, JEVP, FACM** and **FEVP** to accumulate the data sums in the Hessian matrix and the function F after assigning values to p and q. (These programs will replace **JEV** and **FEV** in the chapter three procedure.) Now we have the ingredients of the Newton formula:

$$w_{new} = w - J(w)^{-1}\ F(w)$$

and can find a new vector w.

Many engineering and science problems require the solution of several nonlinear equations. Newton's method is one such algorithm. Most methods to accomplish this can fail under a variety of conditions. Good starting guesses are essential.

HP-48 programs are listed below for Newton's method for this problem. Here we assume there are m parameters and n data points

1. Store the value of m in **M** and and the names of the m parameters in a list named **PL**, say { P Q } for 2 parameters or in the case of four parameters, say **PL** = { P Q A B}. Make sure each of the parameter "variables" in **PL** has been purged.

2. Purge the variables T and Y and store the components for the m functions F(T, Y, P, Q) in a list named **FL**. For example,

{ '(Y – 3*EXP(-P*T) + EXP(-Q*T))*3*T*EXP(-P*T)'

'(Y – 3*EXP(-P*T)+ EXP(-Q*T))*T*EXP(-Q*T)'}

would result from trying to fit $y = 3\,e^{-pt} - e^{-qt}$ to data.

3. The program **DER** given in chapter three will use the calculator's ability to take appropriate derivatives of the functions in **FL**.

4. Store the the N elements of data { (T, Y) } in a list **DTA1.**

5. Now create the programs (which assume values are assigned to P, Q)

Program Name	**JACM**
Purpose and	Create matrix **JMAT**, gets a data point t,y calls the subprogram **JEVP** and does this for each data point

<< {M M} 0 CON 'JMAT' STO 1 N FOR I DTA1 I GET C→R
'Y' STO 'T' STO JEVP NEXT >>

Subprogram Name	**JEVP**
Purpose	Evaluates the elements in **JMAT** at the data point and adds it to the value to the previous sum in the **JMAT** element

<< JL OBJ→ 1 SWAP START →NUM M SQ ROLLD NEXT {M M}
→ARRY JMAT + 'JMAT' STO >>

JMAT will be the Hessian matrix of derivatives.

6. Now for **FVEC** (F vector). In the following P, Q values have been assigned.

Subprogram Name	**FACM**
Purpose	Create **FVEC**, get a data point t, y and call **FEV**: do this for each data point

```
<< {M} 0 CON 'FVEC' STO 1 N FOR I DTA1 I GET C→R 'Y'
STO 'T' STO FEVP NEXT>>
```

Subprogram Name	**FEVP**
Purpose and	Evaluate the functions in **FL** at t,y, P, Q, ... add the value to the previous value stored in **FVEC**

```
<< FL OBJ→ 1 SWAP START →NUM M ROLLD NEXT {M}
→ARRY FVEC + 'FVEC' STO >>
```

Procedure: Store **PL, FL, N, M** and execute **DER** to get **JL** (J list). Put a vector [p,q] with initial values of **P** and **Q** on the stack and execute a program **NST1** given by

```
<< DUP OBJ→ DROP 'Q' STO 'P' STO JACM FACM FVEC JMAT / >>
```

The result is a copy of the old value of [**P**,**Q**] and the increment [Δ**P**, Δ**Q**]. Execute the command – and repeat.

EXERCISE 5.9: Use starting values p = .25, q = 2 and the data to determine appropriate values 0f p and q in

$$y = 3\,e^{-pt} - 2\,e^{-qt}:$$

to fit data { (0, 1), (.4, 1.89), (.8, 2.01), (1.2, 1.9), (1.6,1.72), (2,1.53), (2.4, 1.34), (2.8, 1.18), (3.2, 1.03), (3.6, .903), (4, .79), (5, .57) }

EXERCISE 5.10: Use starting data p = 2.2, q = .925, a = 5 and b = -4 to determine appropriate parameter values in

$$y = a\, e^{-pt} + b\, e^{-qt}$$

to fit data {(0, 1), (.1, .3), (.2, -.2), (.3, -.6), (.4, -.88), (.5, -1), (.6, -1.2), (.7, -1.2), (.8, -1.3), (.9, -1.3), (1, -1.2), (1.5, -1), (2, -.7), (2.5, -.5), (3, -.3), (3.5, -1.7), (4, -.11)}

PROJECT EXERCISE: Data Fit in a Population Problem. Population data { p_i } at times { t_i } (in ten year intervals between 1790 and 1990) was given in chapter two just before problem 2.10. Suppose we use a model $dp/dt = ap - bp^2$: the form of the solution was also given in chapter two. We wish to minimize the payoff function

$$P(p_0^*, a, b) = \sum_{i=0}^{n} \left\{ p_i - \frac{ap_0^*}{bp_0^* + (a - bp_0^*)\, e^{-at_i}} \right\}^2$$

Starting values for the equations obtained by setting the partial derivatives with respect to p_0, a, and b to zero can be obtained by using the data in the year t = 0 (1790) and the years when t = 50 and t = 100. We use Newton's iterative method.

We do not know the form of the solution for some of the population models given in chapter 2. See EXERCISES 2.7, 2.9, 2.10 and 2.11. We have just discussed how to choose problem parameters to achieve a fit to data observations when the functional form of the solution g(t, p, q) is known. Consider the new problem of choosing p and q so that the solution of

$$dy/dt = q\, y\, (1 - y^p), \quad y(0) = .2$$

best fits the (t, y) data (1, .4), (2, .5), (3, .75), and (4, .9).

Here we have an initial value problem, say for population growth, of the form $dy/dt = f(y, p, q)$, $y(0) = y_0$ and wish to choose the parameters p and q so that a close fit to data is achieved. The solution must be obtained by using a numerical method (improved Euler, Runge Kutta, etc.) and the functional form of the solution is unknown. Even though we may attack a vector initial value problem of this type, for simplicity we will assume y, p, and q are real numbers and for values of y, p, q in the domain of f, f(y, p, q) is a real number. If we also assume f is a smooth function, we can differentiate the differential equation with respect to p to obtain

$$\frac{du}{dt} = f_y\,(y, p, q)\,u + f_p\,(y, p, q)$$

where $u = \partial y/\partial p$ and f_y and f_p denote the partial derivatives of f with respect to y and p respectively. A similar equation holds for $v = \partial y/\partial q$. Consider the vector initial value problem $dw/dt = F(w)$ with $w = w(0)$ at $t = 0$ for

$$w = \begin{bmatrix} y \\ u \\ v \end{bmatrix}, \quad F(w) = \begin{bmatrix} f(y, p, q) \\ f_y\,(y, p, q)\,u + f_p\,(y, p, q) \\ f_y\,(y, p, q)\,v + f_q\,(y, p, q) \end{bmatrix}, \quad w(0) = \begin{bmatrix} y_0 \\ 0 \\ 0 \end{bmatrix}.$$

Our criterion for best fit is to choose p, q to minimize the payoff function

$$J = \sum_{i=1}^{N} [y_i - y(t_i, p, q)]^2.$$

Here the data observations are $\{(t_1, y_1), (t_2, y_2), \ldots, (t_N, y_N)\}$, and y(t, p, q) is the solution of the original problem (and the first component of the solution of the vector equation). If we set the derivatives of J with respect to p and q to zero, we get

$$\sum_{i=1}^{N} [y_i - y(t_i, p, q)]\,u(t_i, p, q) = 0, \quad \sum_{i=1}^{N} [y_i - y(t_i, p, q)]\,v(t_i, p, q) = 0.$$

to determine p and q.

Our first thought here may be to apply Newton's method to determine solutions p, q of these two nonlinear equations. However, recall that Newton's method would require partial derivatives of u and v with respect to p and q. This could be done by differentiating the original differential equation more times, but then we would need to solve an initial value problem containing 6 differential equations !

Alternately, suppose that we have a trial set of parameters p and q and wish to choose better values p + Δp, q + Δq. If the incremental values are small then

$$y(t_i, p+\Delta p, q+\Delta q) \approx y(t_i, p, q) + \frac{\partial y}{\partial p}(t_i, p, q)\,\Delta p + \frac{\partial y}{\partial q}(t_i, p, q)\,\Delta q$$

Define a vector z with components $y_i - y(t_i, p, q)$, i = 1, 2, . . ., N and an N by 2 matrix A with components $A(i, 1) = \partial y/\partial p(t_i, p, q)$ and $A(i, 2) = \partial y/\partial q(t_i, p, q)$, i = 1, 2, . . ., N. The payoff at p + Δp, q + Δq is

$$J = \sum_{i=1}^{N} [y_i - y(t_i, p+\Delta p, q+\Delta q)]^2$$

and we wish to choose Δp and Δq so that the new value of J is minimized. If we substitute the approximate value of $y(t_i, p+\Delta p, q+\Delta q)$ into J, we need to choose the vector δ = column [Δp, Δq] so that the quantity $||A\delta - z||^2$ is minimized. Here $||A\delta - z||^2$ is the sum of squares of the components of the vector Aδ – z. This is called a linear least squares problem. The solution is determined by solving $A^tA\delta = A^t z$ where A^t is A transpose. (There is a better numerical method to determine δ presented in standard linear algebra textbooks.) We change to the new values of p and q and repeat the process several times until either this process converges to good values of p and q or until it is clear the process is not converging. In the latter case, we need new starting values of p and q.

Thus our procedure is to take an initial guess for p and q and solve the initial value problem for w and recording the values of y, u, v at the various t_i measurement points, forming the matrix A and vector z and solving for the incremental vector δ, correcting p and q and cycling thru this sequence of steps until a conclusion arises.

EXERCISE 5.11: Choose p and q so that the solution of

$$dy/dt = q\, y\, (1 - y^p), \quad y(0) = .2$$

best fits the (t, y) data (1, .4), (2, .5), (3, .75), and (4, .9). Our criterion is to minimize

$$P = [y_1 - y(1, p, q)]^2 + [y_2 - y(2, p, q)]^2 + [y_3 - y(3, p, q)]^2 + [y_4 - y(4, p, q)]^2$$

where $y_1 = .4$, $y_2 = .5$, $y_3 = .75$, $y_4 = .9$ and y(t, p, q) is the solution of the original problem. Choose starting values of p and q near 1 and 1.

REFERENCES

1. Henon, M. Numerical study of quadratic area-preserving maps, *Quarterly of Applied Mathematics*, 27, 291-312, 1969.

2. Hubbard, J. and B West, *MacMath: A Dynamical Systems Software Package for the Macintosh*, Springer-Verlag, New York, 1992.

3. Nagle, R. K. and E. B. Saff, *Fundamentals of Differential Equations*, Third Edition, Addison and Wesley, New York, 1993

4. Proctor, T. G. , *Calculator Enhancement for a Course in Differential Equations*, Saunders College Publishing, Philadelphia, 1992.

APPENDIX 1

USER MENU HOUSEKEEPING AND ORGANIZATION

The term user memory refers to that part of the calculator's memory which is accessible to a user through the **VAR** menu on the HP-48. User memory is where we 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) which you may regard as the name of the object. Here we are concerned with the basic "housekeeping" procedures associated with programs. By "housekeeping", we mean the simple procedures used to enter, name and store, run, edit and purge programs. The Owner's Manuals minimally address programming; but anyone desiring to become really proficient in developing and using programs across a broad spectrum of applications is strongly advised to study the books "HP-48 Insights" by William C. Wickes.

What is an HP-48 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**.

Naming and storing programs

To name and store a program which has been entered onto level 1 of the stack, press ['] to signify 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**] on the HP-48 will show a user 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; alternatively, key the *full* name into the command line and press [**ENTER**] [**EVAL**]. If the program happens to be on level 1, you may simply press [**EVAL**]. 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 which are named in the program body.

EXAMPLE. The program **« DUP 2 - NEG * »** takes a number "y" from level 1 as input data and returns the calculated value of y(2-y) to level 1. Key in the program by first pressing [« »] on the 48, followed by the other indicated keys. 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**]M.

Now, run the program using as input data the number 4: key in 4 and press [**PGM1**]M. The answer, -8, will be displayed on level 1. Notice that you did not have to enter the data onto the stack before pressing [**PGM1**]M. This is typical; pressing the menu key [**PGM1**]M automatically entered the data for you. Run the program with some more 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].

EXAMPLE. Key in : « → **ARRY** [ENTER]. Notice what happens. Now remove the message, delete the space after the → 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 [▽] key will copy it into 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**] M [**RCL**] will recall the program to level 1 and you can proceed as above.

EXAMPLE. Start this example with the program « **1 N START EULER NEXT** » stored in user memory as [**TRY1**] M.

(i) Recall it to stack level 1 with ['] [**TRY1**] M [**RCL**].

(ii) Copy it to the command line with [▽], and change **EULER** to **IULER**.

(iii) Copy back to level 1 with [ENTER] M, then replace the old version by pressing ['] [**TRY1**] M [**STO**].

(vi) Finally, check your last work by recalling to level 1, examining the result, then dropping it from the stack with **DROP**.

HP-48 Shortcuts

- You can recall to level 1 the contents of any stored variable, say **TRY1**, by pressing → **TRY1** M. Thus, *rightshift 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, *leftshift will load.* Try this by loading « + **SQ** **COS** » into **TRY1**; now recall the contents.

Purging

The object may be any one of the variety of objects recognized by the calculator: a real number, an array, a program, etc. To *purge* variable **PGM1** is to remove it and its contents entirely from user memory. Purging a single variable is usually done with the keystrokes ' **PGM1** M **PURGE**. The label disappears from the 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, then **ENTER**. **PURGE** to purge the variables in this 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** left shift **PURGE**; watch **X** M disappear. Now purge the two remaining variables at the same time by building a list { Y Z } and pressing **PURGE**. Watch these variables disappear.

EXERCISE. The following program takes numbers x, y from the stack and returns sin (xy).

« * SIN »

(a) Key in this program and store it under variable "**EX.1**".

(b) Run the program with inputs .5, π. You will get an expression instead of a number. To get a numerical value use the →**NUM** key.

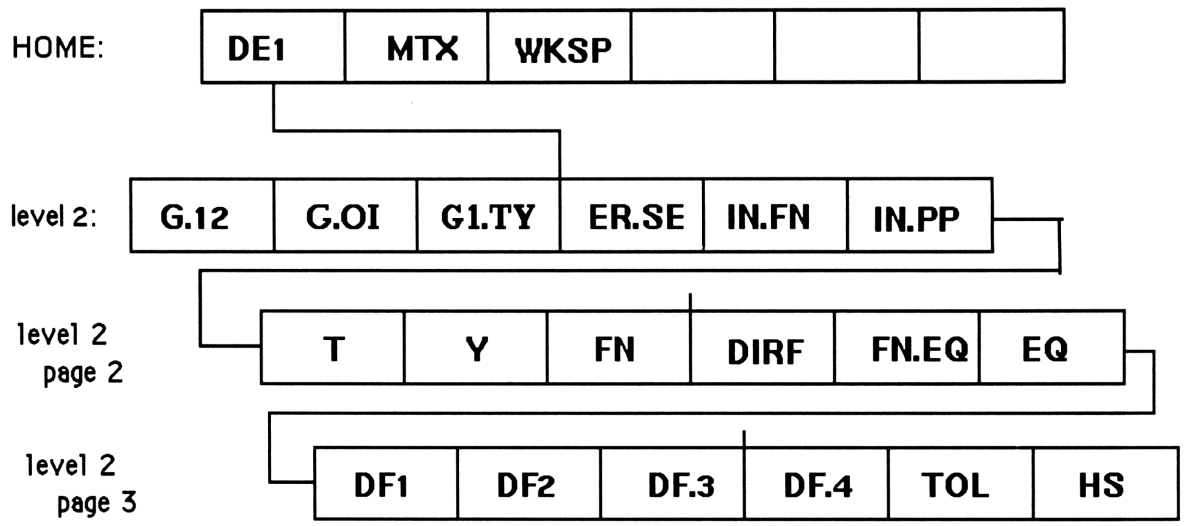
(c) Change the program body by adding **NEG** at the end.

(d) Run the new program with .5, π →**NUM**. What does the new program calculate?

(e) Purge programs **TRY1**, **PGM1**, and **EX.1**.

Just as a file cabinet organizes stored material in an office into convenient groupings, HP-48 *directories* enable you to organize the variables and programs that you store in memory. This memory is called **VAR**. The **VAR** memory is itself a directory – the **HOME** directory, and you can always go to **HOME**. Moreover, in much the same way that certain drawers in a file cabinet are further subdivided into sections, you can create *subdirectories* within the directories. The basic idea is to group together variables associated with a particular topic or subtopic.

A convenient directory structure for the material in this book is as follows:



HOME contains various entries, one being the **DE1** subdirectory – in which you may group together all the stored quantities for differential equations. **HOME** is the *parent* directory of subdirectory **DE1**.

Here's how to create our first subdirectory, subdirectory, **DE1**: press ' **DE1** **CRDIR** M (note that **CRDIR** M appears on the **MEMORY** menu). A new label **DE1** M appears in the original **VAR** menu. Pressing **DE1** M will send you to this new **DE1** subdirectory, which is now empty.

You should enter the programs in **DE1**: **IN.FN, IN.PP, G1.TY, G.0I, G.12 ER.SE, FN, T, Y, TOL, HS, PPAR,** and **EQ.** The programs **INIT1, GRAF, N, H, PPAR, EULER, IULER, EQ**, etc. (if used) should be entered into a separate directory, **DE2.** To transfer a variable from **HOME** to **DE1** first recall its contents to the stack with **RCL**, then recall its name to the stack, activate the **DE1** subdirectory and press **STO**, then purge it from **HOME**.

You may want to arrange the programs in a particular order in say **DE2**. To do this enter { } on the stack. If **GRAF** is to be first (i. e. on the left side of the menu), press ' **GRAF** **ENTER** + to create the list **{ GRAF }**. If the next program is to be **INIT1**, press ' **INIT1** **ENTER** + to create **{ GRAF INIT1 }**. Continue in this way until you have an ordered list of the elements of **DE2** that you care to order. Then press **ORDER** on the **MEMORY** menu to rearrange the menu.

Subdirectory **MTX** should contain the **PIV, ROKL,** and **CHAR** programs from Chapter 4. You may want an **WKSP** directory for odds and ends. The serious differential equations student should create a subdirectory for the programs in Appendix 4.

APPENDIX 2

DIRECTION FIELDS

The following is a set of programs that will construct a direction field for

$$\frac{dy_1}{dt} = F_1(y_1, y_2), \qquad \frac{dy_2}{dt} = F_2(y_1, y_2).$$

We construct the programs so that a student may conveniently overlay a plot of one or more solution trajectories on the direction field. This means that we should present one set of programs for those students who will use the *built-in algorithm* for constructing solutions and a second set of programs for those users who will use the Euler or improved Euler method to produce the solutions.

To use the HP-48G built-in algorithm suppose that **FN** gives the values of the vector $[F_1(Y), F_2(Y)]$ (as illustrated in the first part of Chapter 1). That is **F N** takes values of the variable Y from memory. Set appropriate values in **XRNG** and **YRNG** (program **IN.PP** presented in Chapter 1 can be used for this task), enter program **FN** (a method is to use **IN.FN** also from Chapter 1) and execute **DIRF**.

Program Name:	**DIRF**
Purpose:	Generate a direction field in the region prescribed by **XRNG, YRNG**.
Stored Quantities:	**FN**
Input: none	Output: direction field

```
<< [ 0 0 ] 'Y' STO ERASE { # 0d # 0d } DRAX PVIEW PPAR 2
 GET PPAR 1 GET DUP 3 ROLLD - OBJ→ 9 / 3 ROLLD DUP
    40 / 4 ROLLD 11 / SWAP OBJ→ DF1 PICTURE >>
```

Subprogram Name: **DF1**

```
<< → R DY2 DY1 Y1L Y2L << Y1L 1 10 START DY1 +
DUP 1 12 FOR J DUP Y2L DY2 J * + R 3 ROLLD DF2
NEXT DROP NEXT DROP >> >>
```

Subprogram Name: **DF2**

```
<< DUP2 'Y(2)' STO 'Y(1)' STO FN OBJ→ DROP SWAP
DUP2 IF 0 == THEN DF.3 ELSE DF.4 END >>
```

Subprogram Name: **DF.3**

```
<< 0 IF == THEN DROP2 R→C PIXON DROP
ELSE DROP2 3 ROLL 3 DUPN DUP2 - R→C 4 ROLLD + RC
LINE END >>
```

Subprogram Name: **DF.4**

```
<< DROP / ATAN DUP COS 3 ROLLD SIN 5 ROLL DUP 4
ROLLD * DUP2 - 6 ROLLD + 4 ROLLD * DUP2 - 5 ROLLD +
SWAP R→C 3 ROLLD R→C LINE >>
```

Note that you can overlay a solution trajectory of the system using **G.12** as described in Chapter 1 after the inputs t_0 , y_0 , t_f are placed on the stack or one can use the **input forms** and the **choose boxes**. In the latter case we need to load the appropriate program into **EQ** by excecuting **FN.EQ** given by

```
<< 'FN' RCL 'EQ' STO >>.
```

The user may want to place these programs in the directory **DE.1** containing the programs from chapter 1, placing **DIRF** early in the menu order and the subprograms late in the order.

For those users who wish to use the Euler or improved Euler algorithm given in the second part of Chapter 1, the programs **D.RF**, **DF.1** and **DF.2** are alternatives to **DIRF**, **DF1** and **DF2**. To execute these programs we require a program **F.N** that takes the number T and the vector Y from the stack and produces $[F_1(Y), F_2(Y)]$.

Program Name:	**D.RF**
Purpose:	Generate a direction field in the region prescribed by **XRNG, YRNG**.
Stored Quantities:	**F.N**
Input: none	Output: direction field

```
<< ERASE { # 0d # Od } DRAX PVIEW PPAR 2 GET PPAR 1
  GET DUP 3 ROLLD - OBJ→ 9 / 3 ROLLD DUP 40 / 4
      ROLLD 11 / SWAP OBJ→ DF.1 GRAPH >>
```

Subprogram Name: **DF.1**

```
<< → R DY2 DY1 Y1L Y2L << Y1L 1 10 START DY1 +
DUP 1 12 FOR J DUP Y2L DY2 J * + R 3 ROLLD DF.2
NEXT DROP NEXT DROP >> >>
```

Subprogram Name: **DF.2**

```
<< DUP2 2 →ARRY 0 SWAP FN OBJ→ DROP SWAP DUP2
      IF 0 == THEN DF.3 ELSE DF.4 END >>
```

Note that you can overlay a solution trajectory of the system using **G.Y12** as described in Chapter 1 after the inputs t_0 , y_0 , t_f are placed on the stack. The user may want to place these programs in the directory **DE.2** containing the programs from the second part of Chapter 1, placing **D.RF** early in the menu order and the subprograms late in the order.

EXAMPLE: Consider the system

$$\frac{dy_1}{dt} = y_2 , \quad \frac{dy_2}{dt} = y_1^2 - \varepsilon - y_2 .$$

The equilibrium points are $y_2 = 0, y_1 = \varepsilon$. Let $\varepsilon = 1$ and draw the direction field for $-2 \le y_1 \le 2$, $-1.5 \le y_2 \le 1.5$. 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 the introduction.

EXERCISE: Consider the system

$$\frac{dy_1}{dt} = y_2^2 - 1, \quad \frac{dy_2}{dt} = y_1^2 - 1.$$

The equilibrium points are $y_1 = y_2 = \pm 1$. Draw the direction field for $-3 \le y_1 \le 3$, $-3 \le y_2 \le 3$.

EXERCISE: Consider the system

$$\frac{dy_1}{dt} = y_1^2 - 1, \quad \frac{dy_2}{dt} = y_1 y_2 .$$

The equilibrium points are $y_1 = \pm 1, y_2 = 0$. Draw the direction field for $-2 \le y_1 \le 2$, $-3 \le y_2 \le 3$.

EXERCISE: Consider the system

$$\frac{dy_1}{dt} = y_2, \quad \frac{dy_2}{dt} = y_1^2 - 3y_1^2.$$

The equilibrium points are $y_2 = 0$, $y_1 = 0$ and $y_2 = 0$, $y_1 = 1/3$. Draw the direction field for $-.4 \le y_1 \le .6$, $-.5 \le y_2 \le .5$. Overlay the four trajectories which begin ($t = 0$) at $y_1 = .15$, $y_2 = 0$ and ends when $t = 7$, begin at $y_1 = -.3$, $y_2 = .3$ and ends when $t = 3$, begin at y1 = -.3 , y2 = .4 and ends when $t = 10$, and begin at $y_1 = .5$, $y_2 = 0$ and ends when $t = 3.6$. (The last one satisfies $y_1 = .5\ \mathrm{sech}^2(t/2)$.)

Project Exercise in acoustical dynamics

The speed of sound traveling underwater depends on depth. We will use a ray model for underwater acoustic propagation and let z(x) denote the depth of a sound ray at position x, measured along the ocean surface. Snell's law can be written in the form $\cos\theta/C(z)$ is a constant where $\tan\theta$ is the slope dz/dx and C(z) denotes the speed of sound transmission at depth z. Change the variable by $y = C(z)\ dz/dx$ to obtain

$$\frac{dz}{dx} = \frac{y}{C(z)}, \quad \frac{dy}{dx} = -C'(z).$$

Determine C(z) by a least squares fit of the form $C(z) = a\ e^{-bz} + c + mz$ (a, b, c, and m are constants to be determined) using the data

z	0	500	1000	1500	2000	2500	3000	3500	4000	5000
C(z)	5042	4995	4948	4887	4868	4863	4865	4869	4875	4875
z	6000	7000	8000	9000	10000	11000	12000			
C(z)	4887	4905	4918	4933	4949	4973	4991			

Next, find a value of θ_0 for the initial conditions:

$$z(0) = z_0,\ y(0) = C(z_0)\tan\theta_0,$$

so that for $z(x_f) = z_f$, a prescribed number. For this problem take $z_0 = 2000$, $x_f = 24(5280)$, $z_f = 3000$. Plot the ray trajectory.

The function C(z) can be approximated by a least squares fit to data to another type of function (see Forsythe, George, Michael Malcolm and Cleve Moler, *Computer Methods for Mathematical Computations*, Prentice Hall, 1977.) Such a fit is given by

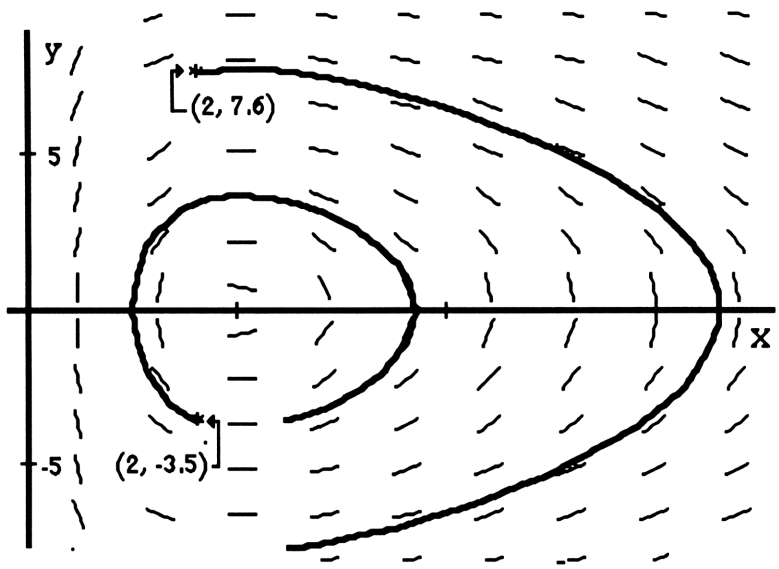
$$C(z) = 4779 + 0.01668\, x + 160{,}295/(x+600).$$

Re-scale the variables x, and z by $t = x/10^4$, $X = z/1000$: the equation become

$$dy/dt = -10\, f'(X), \quad dX/dt = 100y/f(X)$$

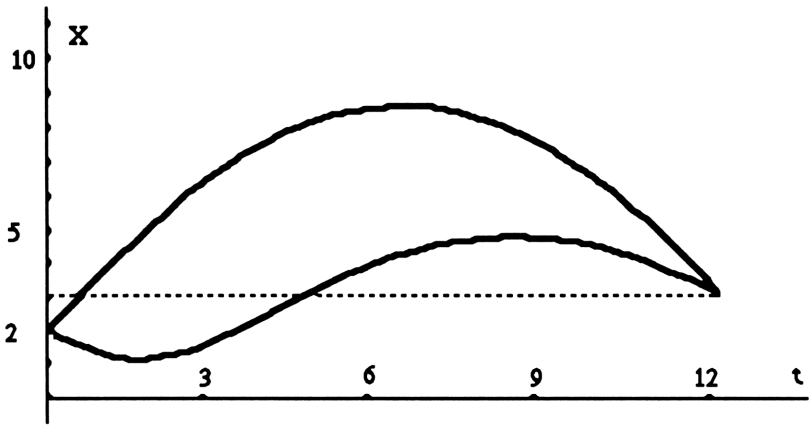
where $f(X) = C(1000X)$. Now we want to find $y(0) = f(2) \tan \theta$ and $X(0) = 2$ so that $X(12.672) = 3$. Dividing the differential equations gives $dy/dX = -f'(X)\, f(X)/(10y)$ integration gives $10\, y^2 + f^2(X) =$ a constant. Phase plane graphs are shown. (An adaptive step method was used to obtain the graphs.)

It is interesting to compare the graph of the solution z(x) obtained by using the C(z) given above with that obtained using the C(z) function given in the project EXERCISE above. Any interpolation formula used for C(z) instead of a least squares fit also gives an interesting comparison.



Phase Plot for Acoustic Model

The X vs. t graph is:



Depth versus Horizontal Distance

APPENDIX 3

DISTINGUISHED OSCILLATIONS OF A FORCED HARMONIC OSCILLATOR

In Chapter Three we noted that if periodic forcing with period T is imposed on the undamped oscillator, solutions may result which have a periodic behavior of a period inherited from the natural frequency of the oscillator and T. (See EXERCISE 3.4.) To be specific, the solution for

$$\frac{d^2x}{dt^2} + \omega^2 x = F_0 \cos \gamma t, \quad x(0) = \frac{dx}{dt}(0) = 0$$

for $\gamma \neq \omega$ is

$$x(t) = \frac{2F_0}{\omega^2 - \gamma^2} \sin\left(\frac{\omega+\gamma}{2} t\right) \sin\left(\frac{\omega - \gamma}{2} t\right).$$

This special solution for a forced harmonic oscillator is quite unusual. The homogeneous differential equation associated with this problem has periodic solutions of period $2\pi/\omega$, the forcing has period $2\pi/\gamma$. The presence of this, possibly periodic solution with a different period, although surprising, comes from the interaction of the forcing function with the solutions of the associated homogeneous problem. Moreover the structure of this special solution is intriguing because it has the form of a relatively slowing amplitude, $\sin((\omega-\gamma)t/2)$ multiplying a factor which varies at a faster rate.

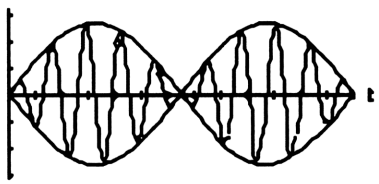
PROBLEM 1: (a) Store the functions

$$\frac{2}{\omega^2 - \gamma^2} \sin\left(\frac{\omega+\gamma}{2} t\right) \sin\left(\frac{\omega-\gamma}{2} t\right), \quad \frac{2}{\omega^2 - \gamma^2} \sin\left(\frac{\omega-\gamma}{2} t\right) \text{ and}$$

$$\frac{-2}{\omega^2 - \gamma^2} \sin\left(\frac{\omega-\gamma}{2} t\right)$$

in the function grapher of your calculator. These function are, of course, x(t) as given above, and two additional curves which are bounding curves for x(t). Plot each of the functions for $0 \le t \le 4\pi/|\gamma-\omega|$ for the cases: (i) $\omega = 2$, $\gamma = 22/9$, (ii) $\omega = 2$, $\gamma = 20/7$. Keep a copy of these graphs for comparison with later work.

The case $\omega = 2$, $\gamma = 22/9$ is graphed below.



Beats Vibration

(b) Suppose $.5|\gamma-\omega|\,\tau = 2\pi$. Then $.5(\gamma+\omega)\,\tau = 2\pi\,(\gamma+\omega)/|\gamma-\omega|$. What is the difference of the behavior of x(t) for t near τ between the case when $(\gamma+\omega)/|\gamma-\omega|$ is an even integer and the case when $(\gamma+\omega)/|\gamma-\omega|$ is an odd integer? Try some examples to discover the answer. For example graph the solution given above for cases (i) and (ii) in the classroom problem given above and in the cases (iii) $\gamma = 18/7$ and $\omega = 2$, (iv) $\gamma = 8/3$ and $\omega = 2$. Give a partial answer based on your result to the question: does an input forcing of period $T = 2\pi/\gamma$ gives periodic output of least period $4\pi/|\gamma-\omega|$.

Many questions concerning the presence of a "special" solution to a forced oscillator may occur to you. We mention several below.

The external forcing in the oscillator mentioned above is periodic (with least period $2\pi/\gamma$) and has average value zero. Is it possible to replace $F_0 \cos \gamma t$ with another function with these properties and discover a solution which resembles the special solution studied above? As an example we consider the function

$$f(t) = F_0 \sum_{n=0}^{\infty} (-1)^n \ \delta(t - n\frac{\pi}{\gamma})$$

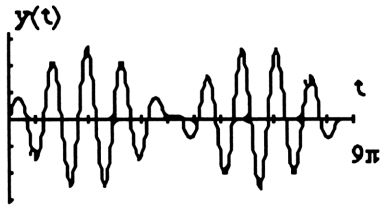
consisting of a sum of Dirac delta functions (see, for example, [3]). You should show the transform of the solution of $y'' + \omega^2 y = f(t)$, $y(0) = y'(0) = 0$ is

$$Y(s) = \frac{F_0}{\omega} \sum_{n=0}^{\infty} (-1)^n \frac{\omega \ e^{-(n\pi/\gamma)s}}{s^2 + \omega^2}$$

and the corresponding solution is

$$y(t) = \frac{F0}{\omega} \sum_{n=0}^{\infty} (-1)^n \ \sin\{\omega(t - n\pi/\gamma)\} \ u(t - n\pi/\gamma).$$

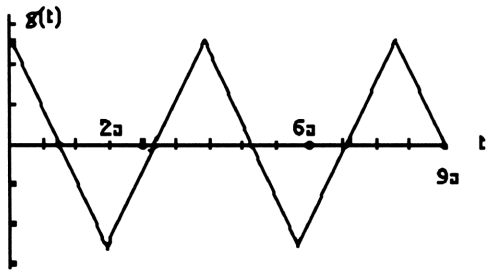
For $\omega = 2$, $\gamma = 22/9$, we obtain the graph shown below for $0 \le t \le 9\pi = 28.274$. Note the input function changes sign every 1.285 units so we must sum at least 25 terms to graph this interval. We note that the graph has generally the same shape as that arising from the forcing $f(t) = \cos \gamma t$; however there is some variation for positive t near $t = 0$ where the slope of the graph is 2 rather than 0 as in the cosine case, near $t = 9\pi/2$ with a flat spot and near $t = 9\pi$ with a flat spot. The width of the flat spot at 4.5π is approximately 1.1 units and the graphs of $\pm 1.4 \sin (2t + .55)$ are good enveloping curves for this "distorted beats vibration.



Response of Harmonic Oscillator to Periodic Impulses

If the graph is extended to $0 \le t \le 18\pi$, it is seen that the response apparently has period 9π. Next question: for a case in which $(\gamma+\omega)/|\gamma - \omega|$ is an odd integer, does the resulting solution have least period $2\pi/|\omega - \gamma|$? To get an indication of the answer, try the case $\omega = 2$ and $\gamma = 20/7$.

The reader is invited to find the response for a sawtooth wave input forcing function of the type indicated below. This case gives a response which is very close to the response from cosine forcing.



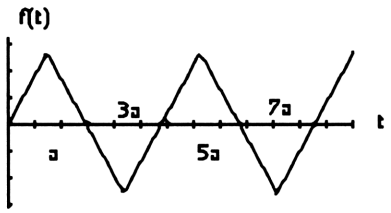
Sawtooth Input Forcing

We have just noticed that a harmonic oscillator forced with a function with geometric properties similar to a cosine function may have a special solution analogous to the special solution in the more familiar problem. What are other examples of such a phenomena?

We suggest the following type of problem. Suppose that a harmonic oscillator $y'' + \omega^2 y = f(t)$ is forced with a periodic function with geometric properties similar to sin ωt. Will there be a resonance solution? As an example find the solution of the problem forced by the function

$$f(t) = t/a \text{ for } 0 < t < a, \quad = 2 - t/a \text{ for } a < t < 3a, \quad -4 + t/a \text{ for } 3a < t < 5a$$
$$= 6 - t/a \text{ for } 5a < t < 7a, \quad -8 + t/a \text{ for } 7a < t < 9a, \text{ etc.}$$

with graph as shown.



We suggest you study the solution with $y(0) = y'(0) = 0$. Another type of input forcing could be patterned after the sum of impulse forces (Dirac delta functions).

APPENDIX 4

RUNGE-KUTTA ADAPTIVE STEP SIZE ALGORITHM

Elementary algorithms have been featured in this manual primarily for pedagogical reasons; however they also permit quick execution times. There is a loss of accuracy, but the simplicity of the programs will help students to learn valuable programming skills. In this appendix, we present an adaptive step method for the solution of initial value problems. The extension is a program for the popular Runge-Kutta Feldberg 4/5 algorithm which calculates the new value of y using a result which is accurate to fifth order in h whenever an estimated error test is passed. When the test is successful the step size is increased and the next step is attempted. If the test fails, then the step size is reduced and another attempt is made to find a step size in t which gives at most an extremely small error in the new value of y. For simplicity, in our programs no check is made to prevent extremely small steps. If the user wants to prevent ridiculously small steps, appropriate program steps should be incorporated. The algorithm is given by:

$$y_{new} = y + (16/135)k_1 + (6656/12825)k_3 + (28561/56430)k_4 - (9/50)k_5 + (2/55)k_6$$

and

$$esterr = (1/360)k_1 - (128/4275)k_3 + -(2197/75240)k_4 + (1/50)\,k_5 + 2/55\,k_6$$

where

$$k_1 = hf(t,y), \quad k_2 = hf(t+h/4,\ y+k_1/4) \quad k_3 = hf(t+3h/8,\ y+3k_1/32+9k_2/32)$$

$$k_4 = hf(t+12h/13,\ y+1932k_1/2197-7200k_2/2197+7296k_3/2197)$$

$$k_5 = hf(t+h,\ y+439k_1/216-8k_2+3680k_3/513-845k_4/4104)$$

$$k_6 = hf(t+h/2,\ y-8k_1/27+2k_2-3544k_3/2565+1859k_4/4104-11k_5/40)$$

Given t, y, and h, the process of calculating the term y_{new} using these formulae is coded into the following program. At the end of the program, the step size is divided by 2 and stored.

Subprogram Name	**FBG**
Stored Quantities	**FN H**
Input	**T** and **Y**
Output	**T Y esterr** (**H** has been reduced 50% & **DELY** has been stored)

```
<< → T Y
<< T Y FN H * DUP  DUP2 3 DUPN 4 / Y + T H 4 / +
SWAP FN H * DUP DUP2 9 * 5 ROLL 3 * + 32 / Y + T H
3 * 8 / + SWAP FN H *  DUP  DUP DUP2 7296 * 6 ROLL
7200 * - 8 ROLL 1932 * + 2197 / Y + T H 12 * 13 / +
SWAP FN H * DUP DUP2 845 * 4104 / 5 ROLL 3680 * 513
/ SWAP - 8 ROLL 8 * - 9 ROLL 439 * 216 / + Y + T H +
SWAP FN H * DUP DUP 11 * 40 / 4 ROLL 1859 * 4104 /
SWAP - 6 ROLL 3544 * 2565 / - 8 ROLL 2 * + 8 ROLL 8 *
27 / - Y + T H 2 / + SWAP FN H * DUP 2 * 55 / 3 ROLL
9 * 50 / - 4 ROLL 28561 * 56430 / + 5 ROLL 6656 * 12825
/ + 6 ROLL 16 * 135 / + 'DELY' STO 2 * 55 / SWAP 50 /
+ SWAP 2197 * 75240 / - SWAP 128 * 4275 / - SWAP 360
/ + ABS T Y ROT H 2 /  'H' STO END >> >>
```

The reader should notice that to calculate y_{new} and esterr, k_1 is needed 7 times (once each for k2, k3, k4, k5, k6 y_{new} and esterr), k_2 is needed 4 times (once each for k_3, k_4, k_5, k_6), k_3 is needed 5 times (once each for k_4, k_5, k_6, y_{new}, esterr), k_4 is

needed 4 times (once each for k_5, k_6, y_{new}, esterr), k_5 is needed 3 times (once each for k_6, y_{new}, esterr), and k_6 is needed 2 times (once for y_{new}, once for esterr). In the program listing, each time the commands **FN H *** are executed, one of the k terms has been calculated. The next commands create the appropriate number of copies of these numbers.

The **FBG** program is used as a subprogram to the program **RKFS** program listed below:

Subprogram Name	**RKFS**
Stored Quantities	**FN H FBG**
Input	**T** and **Y**
Output	$\mathbf{T_{new}}$ $\mathbf{Y_{new}}$ **H** (**H** has been altered)

```
<< DO FBG UNTIL .00001 < END DELY + SWAP H 2 * +
                SWAP H 4 * 'H' STO    >>
```

The tolerance used in the program (.00001) may require adjustment. This quantity can easily be changed to a variable and used as input to the program listed below:

Program Name	**RKF**
Stored Quantities	**FN H FBG RKFS**
Input: $\mathbf{T_{initial}}$ $\mathbf{Y_{initial}}$ $\mathbf{T_{final}}$	Output: **Stored List YV**

```
<< → TF << { } 'YV' STO DO RKFS DUP2 OBJ→ DROP
 2 →ARRY YV + 'YV' STO 2 PICK H + UNTIL TF > 2
PICK TF SWAP - 'H' STO FBG DROP DELY + SWAP 2 H *
  + OBJ→ DROP 2 →ARRY YV + 'YV' STO  >> >>
```

As written, the program will work for a vector system dy/dt = f(t,y) with y and f as vectors with M components except the user should replace in two places in **RKF** the sequence of commands **OBJ→ DROP 2 →ARRY** with **OBJ→ DROP** (M+1) **→ARRY** Here substitute the number M+1 into the command or use a variable named **M** and put **M 1 +** in for (M+1). Of course, the program **FN** takes **T** and the vector **Y** from the stack and returns the vector f (**T**,**Y**).

APPENDIX 5

PROGRAMS FOR THREE DIMENSIONAL TRAJECTORIES

The programs given below may be used to produce graphs of solution trajectories in three dimensions. The user executes **IN.P** to input the high and low values of X, Y, Z for the view box to be shown and the position of the eye XE, YE, ZE. The utility program **UTL1** defines an orthogonal set of vectors U.1, V.1 and W.1 and sets the associated two dimensional plot parameters. Subprograms **SCAL, UVW** and **PJ.1** are called by the other programs. The program **G.Y3** is similar to other programs given earlier to graph the solution trajectories.

Program Name	**G.Y3**
Stored Quantities	**FN HS TOL PJ.1**
Input	$\mathbf{T_0}$ (initial time)
	$\mathbf{Y_0}$ (initial value as a three vector)
	$\mathbf{T_f}$ (final time)
Output :	Solution graph of $Y' = F(T, Y)$, $Y(T_0) = Y_0$, $T_0 \le T \le T_f$.

```
<< { # 0d # 0d } PVIEW 3 ROLLD 'Y' STO 'T' STO → TF
<< { T Y FN } TOL HS Y PJ.1 R→C 4 ROLLD DO
RKFSTEP Y PJ.1 R→C DUP 6 ROLLD 5 ROLL LINE DUP T
+ TF UNTIL > END DROP TF T - RKFSTEP Y PJ.1 R→C
DUP 6 ROLLD 5 ROLL LINE DROP TF T - RKFSTEP Y PJ.1
R→C 5 ROLL LINE 3 DROPN >> PICTURE >>
```

Program Name	**IN.P**
Stored Quantities	**none** Input **none**
Output	coordinates of the corners of the viewing box and the position of the eye have been stored

```
<< "ENTER XL XU" " " INPUT OBJ→ 'XU' STO 'XL' STO
   "ENTER YL YU" " " INPUT OBJ→ 'YU' STO 'YL' STO
   "ENTER ZL ZU" " " INPUT OBJ→ 'ZU' STO 'ZL' STO
"ENTER XE" " " INPUT OBJ→ "ENTER YE" " " INPUT OBJ→
    "ENTER ZE" " " INPUT OBJ→ 3 →ARRY 'W.1 STO >>
```

Program Name	**UTL1**
Stored Quantities	variables from **IN.P** and programs **SCAL**, **UVW**
Input	none
Output	the parameters **XRNG YRNG** have been set.

```
<< UVW SCAL SCATRPLOT 1.125 DUP *W *H
        ERASE FUNCTION CLΣ >>
```

Subprogram Name	**SCAL**
Stored Quantities:	Output from **IN.P** and **UVW** Input: none
Output	a matrix **ΣDAT** has been defined containing
the	projections of the corners of the viewing box.

<< CLΣ XL YL DUP2 ZL →V3 PJ.1 →V2 Σ+ ZU →V3 PJ.1 →V2 Σ+ XL YU DUP2 ZL →V3 PJ.1 →V2 Σ+ ZU →V3 PJ.1 →V2 Σ+ XU YL DUP2 ZL →V3 PJ.1 →V2 Σ+ ZU →V3 PJ.1 →V2 Σ+ XU YU DUP2 ZL →V3 PJ.1 →V2 Σ+ ZU →V3 PJ.1 →V2 Σ+ >>

Subprogram Name	**UVW**
Stored Quantities	Output from **IN.P** and **UVW**
Input	none
Output	an orthogonal set of vectors **U.1, V.1** and **W.1** have been constructed.

<< W.1 DUP ABS / DUP DUP 'W.1' STO [0 0 1] DUP 3 ROLL DOT 3 ROLL * - DUP ABS / DUP 'V.1' STO W.1 CROSS 'U.1 STO >>

Subprogram Name	**PJ.1**
Stored Quantities	Output from **UVW**
Input	any vector **Z** in R^3
Output	the coordinates of the projection of **Z** in R^2.

<< DUP U.1 DOT SWAP V.1 DOT >>

EXAMPLE: For view box $-2 \le x \le 2$, $-2 \le y \le 2$, $-.5 \le z \le 2$, eye position [1, 1, .5], differential equation $Y' = AY$, with A given below, $Y(0) = [2, 0, 2]$ and $0 \le t \le 12.56$ we obtain the following picture:

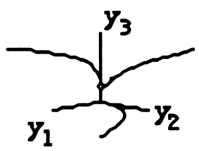
xyx solution for
$Y' = AY$, $Y(0) = [2, 0, 2]$

Here the first row of A is [-.1, 2, 0], the second row is [-2, -.1, 0], and the third row is [0, 0, -.2]. Here the solution is given by $x(t) = 2\, e^{-.1\, t} \cos t$, $y = 2\, e^{-.1\, t} \sin t$, and $z(t) = 2\, e^{-.2t}$.

EXERCISE: Use the same view box and eye position as given above. Change A to

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -.5 & 0 \\ 0 & 0 & -.25 \end{bmatrix}$$

and plot the solutions of $Y' = AY$ with initial conditions [2, 0, 2], [0, 2, 2] and [2, 2, 0] for $0 \le t \le 6$.

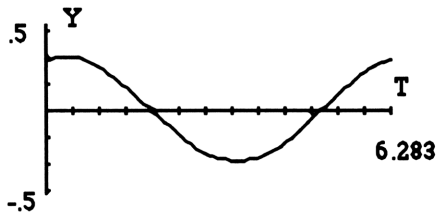


Normal Mode Solutions
to $Y' = AY$

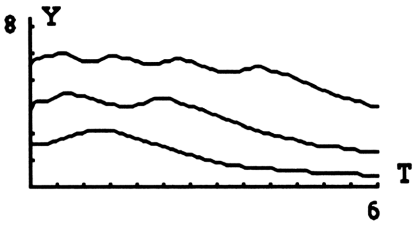
APPENDIX
6

ANSWERS TO SELECTED EXERCISES

1.1

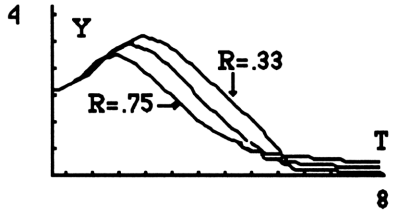


1.2

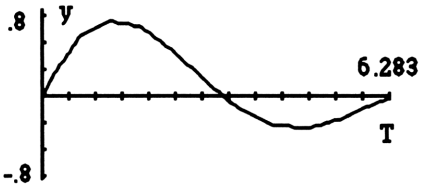


Note Y(6.283) = .3

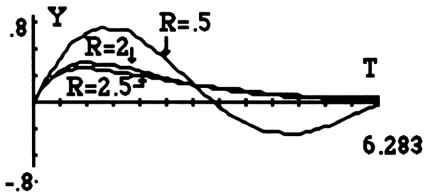
1.2a



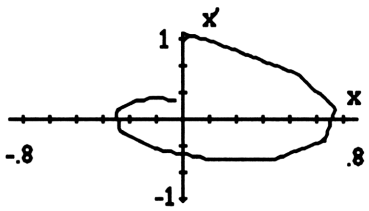
1.3



1.3a



1.5



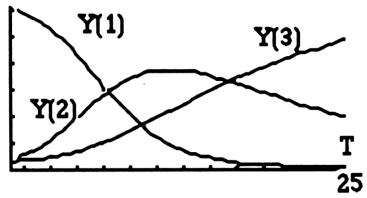
1.7 Starting at T = 1, Y = 1 for H = .2 we get Y = 1.4, Y = 1.96, Y = 2.788, Y = 4.110, and Y = 6.443 when T = 1.4, T = 1.6, T = 1.8, and T = 2 respectively.

Starting at T = 1, Y = 1 for H = .1 we get Y = 1.44, Y = 2.097, Y = 3.169, Y = 5.168, and Y = 9.839 when T = 1.4, T = 1.6, T = 1.8, and T = 2 respectively.

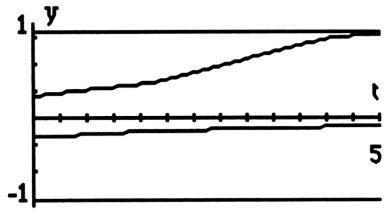
Starting at T = 1, Y = 1 for H = .05 we get Y = 1.467, Y = 2.196, Y = 3.484, Y = 6.280, and Y = 15.693 when T = 1.4, T = 1.6, T = 1.8, and T = 2 respectively.

1.8 See page 25.

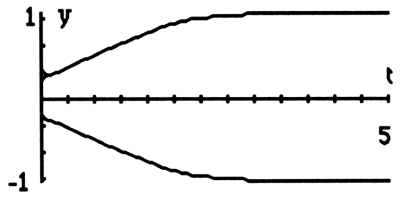
1.9



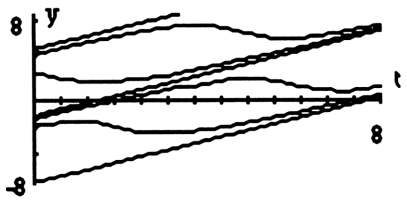
2.1



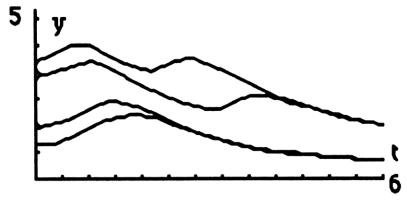
2.2



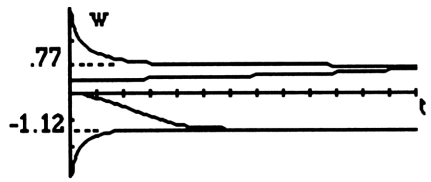
2.3

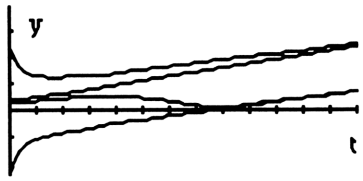


2.4

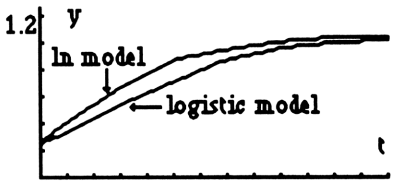


2.5 The equilibrium solutions are w = -1.125, w = .339 and w = .786.





2.6 There is only one 'equilibrium' solution 2.7



2.8 The substitution $x = \omega^4$ leads to the equation $\int \frac{4\omega^3}{(1-\omega)(1+\omega+\omega^2+\omega^3+\omega^4)} = t + C$

The integrand is $\frac{A}{1-\omega} + \frac{B\omega+C}{\omega^2+.5\,(1+\sqrt{5})\,\omega+1} + \frac{D\omega+E}{\omega^2+.5\,(1-\sqrt{5})\,\omega+1}$

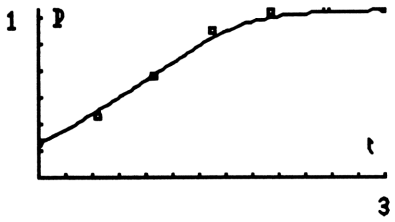
for A= .8, B = 2.3677, C = 1.2944, D = -1.5677, E = -.4944. An antiderivative of the left side is $.5B \ln\{\omega^2 + .5(1+\sqrt{5})\,\omega + 1\} + (C-.25B(1+\sqrt{5}))1.7 \tan^{-1}\frac{2\omega+1.618}{1.1755} + .5D \ln\{\omega^2 + .5(1-\sqrt{5})\,\omega + 1\} + (E-.25D(1-\sqrt{5}))1.0515 \tan^{-1}\frac{2\omega-.618}{1.1902}$ $-A \ln|1-\omega|$. We replace ω by $x^{.25}$ and use the initial condition x(0) = .5 to evaluate the constant of integration. This analytical procedure is already quite formidable and we must still invert the expression to graph x(t). What if a parameter value in the equation changes? This adds further complication to the analytical procedure. The case r = 5/2 is similar.

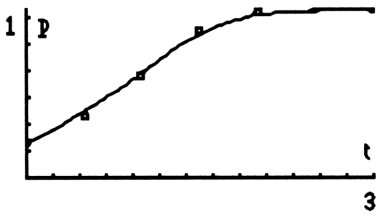
2.9 If Q(t) is the weight of salt in the tank at time t (in minutes), we have that $\frac{dQ}{dt} = 2(1.5) - \frac{Q}{300-t}3$, $Q(0) = 0$. This equation has solution $Q = 450x\,(1-x^2)$ where $x = \frac{300-t}{300}$. If we put Q = 21, we get x solutions .04677 and .9758 which translate to t = 7.262 and y = 286.

2.10 Following the hint gives $t_k \ln\left\{\frac{\ln(p(t_i)x)}{\ln(p(0)x)}\right\} = t_i \ln\left\{\frac{\ln(p(t_k)x)}{\ln(p(0)x)}\right\}$ for $x = e^{-s}$. Using the solver we solve this equation for x. This gives a value for

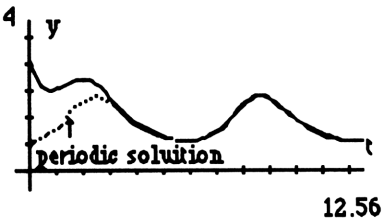
$A/B = -\ln x$. B is given by $-Bt_i = \ln\left\{\dfrac{\ln(p(t_i)x)}{\ln(p(0)x)}\right\}$ For the case at hand, we get $x = .665282$ which gives $B = 2.6391$ and $A = 1.07557$. Plotting the resulting solution gives the expected results.

2.12 Clearly $K = 1$, so a quadratic model as constructed in Exercise 2.11 has the form $Y' = \sigma\ \text{IFTE}(\ Y < \theta, \theta^2 - (Y-\theta)^2, \dfrac{\theta^2}{(1-\theta)^2}[(1-\theta)^2 - Y-\theta)^2)$ for some $s > 0$. The following pictures were obtained by using $\theta = .6$, $\sigma = 1.4$ and $\theta = .65$, $\sigma = 1.25$.

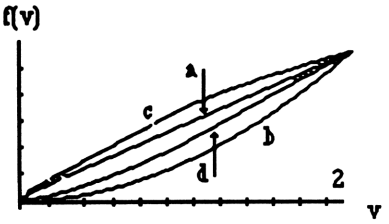


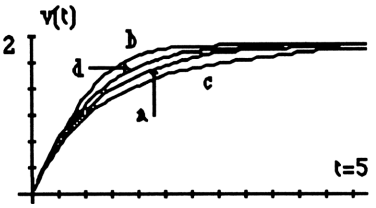


2.13

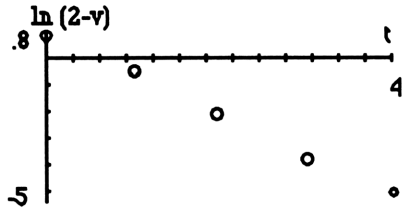


2.14

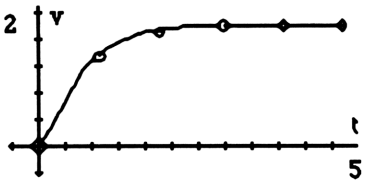




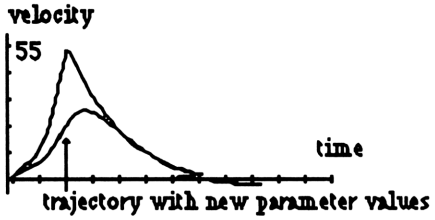
2.15



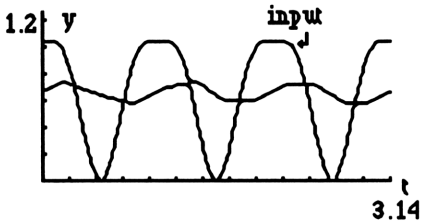
r=2



2.16

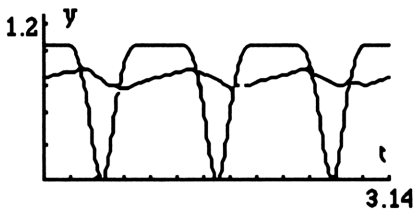


2.18a

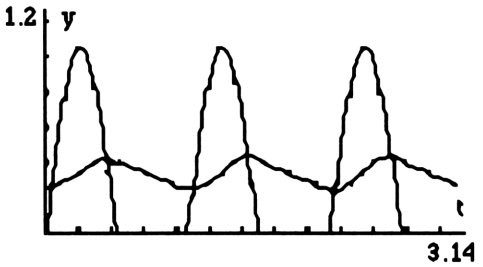


peak at t = 5.6, v = 29, final velocity = -3.64.

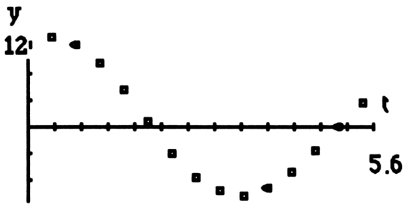
2.18b



2.18c



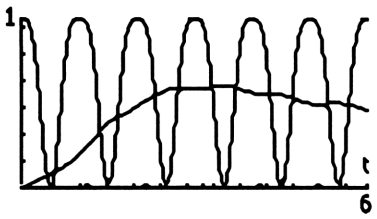
3.1

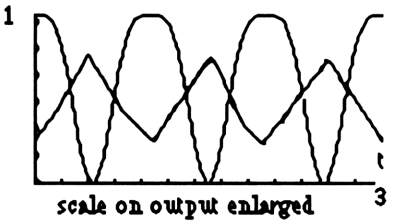


$y = A e^{-rt} \cos(bt+c)$

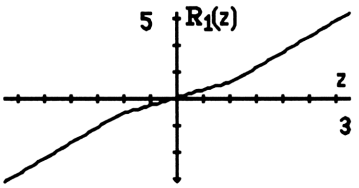
$y = 0$ at $t = 5.22$ and 2.10 so $b = \frac{\pi}{(5.22-2.10)} = 1.01$

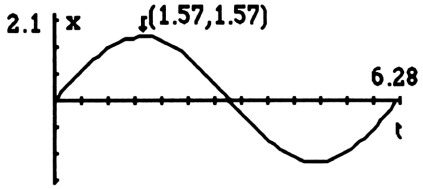
and $12 e^{-.4r} = 9 e^{-r(.4+\pi/b)}$ so $r = -b (\ln .75)/\pi = .092$. Thus $B = .184$ and $C = (.092)^2+(1.01)^2 = 1.029$.

3.2 

3.3 

The calculator gives $x_q(1) = .1973$, $x_q'(1) = .3228$ so initial conditions are $x(0) = .488$, $x'(0) = .012$.

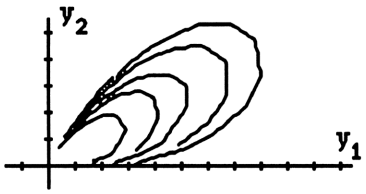
3.4 A graph of $R_1(z)$ vs z is 

3.4a 

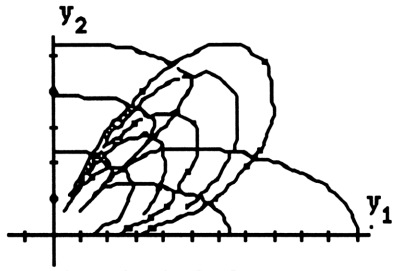
Contrast (a) with (c) solution x = 2 sin t.

3.5 The program **<< → W << W DUP 2 ^ 1 - NEG 2 * INV → C << C 2 * NEG 0 XRNG C W * DUP NEG SWAP YRNG ERASE { # 0d # 0d } PVIEW 1 200 FOR N N 2 * P * W * →NUM DUP SIN SWAP COS 1 - C * SWAP W * C * NEG R→C PIXON NEXT >> >> >>** requires input W and outputs the desired graph and leaves a copy of W on the stack. Try for W = 19/20, W = 6/7, W = $1/\sqrt{7}$.

3.6

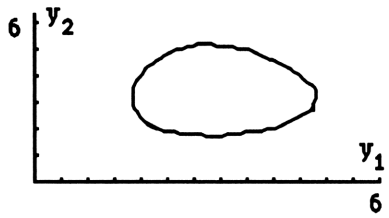


trajectories for first system



trajectories for both systems

3.7

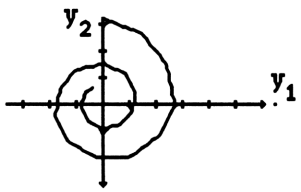


3.8b $$P(z) = \frac{4}{\sqrt{2}} \int_0^z \frac{dx}{\sqrt{\cos x - \cos z}}$$

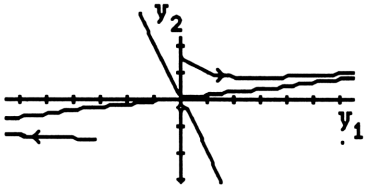
$P(\pi/4) = 6.5340$ (takes HP about 9 minutes at 4 FIX to evaluate integral)

$P(\pi/2) = 7.4158$

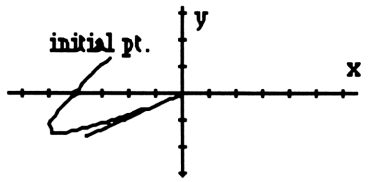
3.9



3.10 The system has eignevalues $\pm \sqrt{5}$ and eigenvectors proportional to [4.236, 1] and [.2361, -1]. For initial conditions not proportional to the latter vector the trajectory will quickly approach the first vector as t increases.

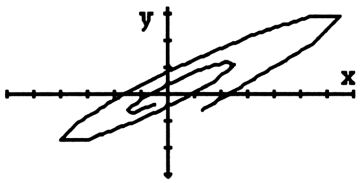


3.11a



In 3.11a the final value of [x,y] was [-.333, -.198]. The ratio of y to x was 1.68. We show in the figure both the trajectory and the line y = 1.67 x. The eigenvalues of the matrix are -1.5 and -.5 with associated eigenvectors [1, 1] and [1, .6].

3.11b

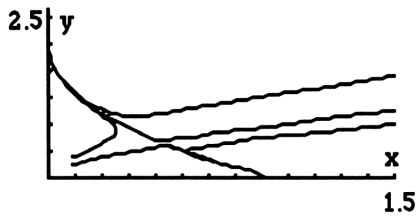


$\lambda^2 + .6\lambda + 4.09$

$\lambda = -.3 \pm 2i$

$$\begin{bmatrix} x \\ y \end{bmatrix} = e^{-.3t}\left\{\alpha\left(\begin{bmatrix} 1 \\ .6 \end{bmatrix}\cos 3t + \begin{bmatrix} 0 \\ .2 \end{bmatrix}\sin 3t\right) + \beta\left(\begin{bmatrix} 1 \\ .6 \end{bmatrix}\sin 3t - \begin{bmatrix} 0 \\ .2 \end{bmatrix}\cos 3t\right)\right\}$$

3.15



4.1a $x^2 + 2x + 5 = 0$ 4.1b $x^2 + 1001x + 1000 = 0$ 4.2a $x^3 - 6x^2 + 11x - 6 = 0$

4.2b $x^3 - 3x^2 + 4 = 0$ 4.2c $x^3 + 2x^2 - 21x - 6 = 0$

4.3a 2, 3, 1 4.3b -1, 2, 2 4.3c -5.574.., -.279.., 3.853...

4.4a -1.0979... ±i.785..., 2.1958... 4.4b -.347..., 1.879..., -1.532...

4.4c $-1 \pm 2i, -3$

4.5a $$Y(t) = \begin{bmatrix} -e^{t} & e^{-t} & -1 \\ 0 & .5\,e^{-t} & 1 \\ e^{t} & -e^{-t} & 0 \end{bmatrix}$$

4.5b $$\text{column } 1 = e^{\alpha t}\begin{bmatrix} .1776\cos\beta t - .5195\sin\beta t \\ -.6027\cos\beta t + .4309\sin\beta t \\ \cos\beta t \end{bmatrix},$$

$$\text{column } 2 = e^{\alpha t}\begin{bmatrix} .1776\sin\beta t + .5195\cos\beta t \\ -.6027\sin\beta t - .4309\cos\beta t \\ \sin\beta t \end{bmatrix}$$

with $\alpha = -1.079$, $\beta = .785...$ and column 3 = $e^{2.196t}$ column [.2074, .4554, 1].

4.6a $$Y(t) = \begin{bmatrix} e^{-t} & e^{2t} & -.0174\,e^{-t} \\ -.5e^{-t} & e^{2t} & -.983\,e^{-t} \\ -.5\,e^{-t} & e^{2t} & e^{-t} \end{bmatrix}$$

4.6b The eigenvalues are -1 (repeated) and -2. Let **a** = column [1, 2, 4], an eigenvector corresponding to the eigenvalue -1. We try $y = e^{-t}(\mathbf{a}t + \mathbf{b})$ as a solution to $y' = Ay$ and see this will be a solution if $(A + I)\,\mathbf{b} = \mathbf{a}$. We create the 3 by 4 matrix [A, a] and reduce it by using the **RREF** command or a series of pivot operations to get the solution **b** = column [0, 1, 4]. Finally an eigenvector corresponding to the eigenvalue -2 is column [1, 1, 1]. Hence a fundamental matrix of solutions is

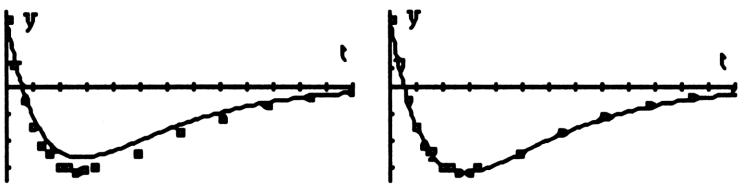
$$Y(t) = \begin{bmatrix} e^{-t} & e^{-t}t & e^{-2t} \\ 2e^{-t} & e^{-t}(2t+1) & e^{-2t} \\ 4e^{-t} & e^{-t}(4t+4) & e^{-2t} \end{bmatrix}.$$

4.6c Again the eigenvalues are -1 (repeated) and -2, and there is only one eigenvector corresponding to the eigenvalue -1, viz., **a** = column [-.5, 1, .5]. We try as a second solution $y = e^{-t}(\mathbf{a}t + \mathbf{b})$ and proceed as in 4.3b.

4.7 The eigenvalues are 5.964, -1.529 and -3.218 ± i 2.703 with corresponding eigenvectors column [-.2866, .4203, 1, -.04738], column [1, -.0063, -.1994, -.5790] and column [-.1846± (-.5511), .0255 ± .0844, -.5882 ± i.0899, 1].

5.10 The list **FL** should be { **'(Y - A*EXP(-P*T)-B*EXP(-Q*T))*A*T*EXP(-P*T)'**
'(Y - A*EXP(-P*T)-B*EXP(-Q*T))*B*T*EXP(-Q*T)'
'(Y - A*EXP(-P*T)-B*EXP(-Q*T))*EXP(-P*T)'
'(Y - A*EXP(-P*T)-B*EXP(-Q*T))*EXP(-P*T)'},

the parameter list **PL** should be **{P Q A B}**, **M** = 4. The program **NST1** should be modified to **<< DUP OBJ→ DROP 'B' STO 'A' STO 'Q' STO 'P' STO JACM FACM FVEC JMAT / >>.** After several iterations, we obtain P = 2.07, Q = .93, A = 6.13, B = -5.13.



Fit using starting parameters Fit using final parameters

INDEX

Acoustical dynamics 132
Artillary shell 96
Asymptotic 26, 106
Attracting solutions 62, 68, 108
Autonomous 50, 69
Beats vibration 137
Characteristic equation 66
Characteristic equation algorithm 84
Critical point (see equilibrium point)
Data fit 33, 38, 52, 54, 112, 117
Direction fields 3, 128-34
Directory (subdirectory) 4, 14, 24, 125-6
Discrete dynamical system 108ff
Eigenvalues, eigenvectors 66, 76ff
Equilibrium solution 25-28, 62, 68, 72, 131-2
Euler algorithm 2, 15-17, 22, 51, 128
Fundamental solution matrix 82, 84-85, 87
Graphics screen 20
Harmonic oscillator 135
Henon map 109
Implicitly defined solution 29
Improved Euler alg. 2, 17, 18, 22, 51, 128
Inflection points 35
Initialization program 9, 21, 23
Input/output problems 41, 55
Input signal delay 42
Julia set 110
Linear autonomous system 66
Lorentz 100
Lotka-Volterra model 63
Mixing problem 30-31
Newton's method 72, 95, 114
Nonhomogeneous 88ff
Numerical integration
Orthogonal trajectories 60
Overlay 11, 13-14, 26
Parameters 11-12, 33, 53, 108
Particle motion 36-40, 44-49, 95, 103
Pendulum 63, 92
Periodic 56, 58, 62, 90, 105, 136
Phase plane 62
Poincare section 59
Population growth problems 31-36
Projectile motion 95
Pursuit problem 98
Repelling solutions 62, 68, 108, 111
Restricted three body problem 103
Rocket flight 97
Runge Kutta algorithm 2, 6, 73
Sawtooth wave 138
Second order IVP 2, 3, 7, 50
Solution structure 27-28
Spiral point 68
Spring /circuit model 57
Square wave, Switch function 43
Step size selection 18
Stiff differential equation 73
Terminal velocity 37-38
Trajectories in three dimensions 144ff
Undetermined coefficients 91
Vander Pol 59, 64
Variational matrix 69

PROGRAMS

CHAR	Finds eigenvalue equation	85
DDS et al.	Generates plot for example discrete dynamical system	107
DER et al.	Creates list JL of derivatives of vector F(T,Y)	70
D.GRF	Generates graph of T, Y data	34
DIRF et al	Direction field programs using stored Y	128
D.RF et al	Direction field programs using Y as a local variable	130
EIG2	Eigenvalue equation for 2 by 2 matrices	77
EIG3	Eigenvalue equation for 3 by 3 matrices	77
EULER	Euler algorithm	16
IULER	Improved Euler algorithm	17
FACM, FEVP	Utility programs for NSTP1	114
FEV	Utility for NWTN	72
G.0I	Generates a T, Y(I) plot for vector Y	12
G.12	Generates a Y(1), Y(2) plot	13
G1.TY	Generates a T, Y plot	10
G.TYI	Generates a T, Y(I) plot using Iuler algorithm	22
G.Y3	Generates solution plot in three dimensions	144
GRAF	Graphics program for T, Y plot using IULER	20
GS.01	Generates T, Y(1) graph in stiff differential equation	75
G.Y12	Generates Y(1), Y(2) plot using IULER	24
IN.FN	Prompts user to write program for F(T,Y)	9
IN.PP	Prompts user to set plot parameters	9
INIT1	Prompt user to set parameters for GRAF	21
INIT.I	Prompts user to set parameters for G.TYI	23
JACM, JEVP	Utility program for NSTP1	115
JEV	Creates matrix of derivatives of F(T,Y)	71
NWTN	Newton step for vector solution of F(Y,Y) = 0	72
NSTP1	Newton step for parameter values	116
PIV	Gauss pivot on element KL	79
RKF et al.	Runge Kutta Feldberg algortihm	142
ROKL	Interchange rows K and L	80


ISBN 1-886801-19-3
51595
EAN
9 781886 801196