NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

MATHEMATICAL MODELING USING MAPLE

by

Robert Edward Beauchamp

September 1996

Thesis Advisor: Maurice Weir

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1996	Master's Thesis

4. TITLE AND SUBTITLE
MATHEMATICAL MODELING USING MAPLE

5. FUNDING NUMBERS

6. AUTHOR(S)
Beauchamp, Robert Edward

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release, distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The area of higher mathematics begins with successive courses in calculus; however, rarely does the calculus student recognize the applications or impetus for the mathematical skills that are taught. Giordano and Weir produced *A First Course in Mathematical Modeling*, the first text which addressed this shortcoming in the curriculum of every science and engineering field. With the advent of powerful classroom computers, Fox, Maddox, Giordano and Weir produced *Mathematical Modeling With Minitab*, which assists the student in translating the theory into a computer language. At the Naval Postgraduate School, Maple is the software used most commonly in the Mathematics Department, requiring a similar instructing tool. Mathematical Modeling Using Maple follows the lead of *Mathematical Modeling With Minitab*, and assists the student in grasping the concepts of the modeling class without getting slowed down by the syntax of Maple.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Model Fitting, Proportionality, Least-Squares, Cubic Spline	112
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

Approved for public release; distribution is unlimited.

MATHEMATICAL MODELING USING MAPLE

Robert Edward Beauchamp
Lieutenant, United States Navy
B.S., United States Naval Academy, 1990

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
September 1996

Author: _____
Robert Edward Beauchamp

Approved by: _____
Maurice Weir, Thesis Advisor

_____
Carlos Borges, Second Reader

_____
Richard Franke, Chairman
Department of Mathematics

ABSTRACT

The area of higher mathematics begins with successive courses in calculus; however, rarely does the calculus student recognize the applications or impetus for the mathematical skills that are taught. Giordano and Weir produced *A First Course in Mathematical Modeling*, the first text which addressed this shortcoming in the curriculum of every science and engineering field. With the advent of powerful classroom computers, Fox, Maddox, Giordano and Weir produced *Mathematical Modeling With Minitab*, which assists the student in translating the theory into a computer language. At the Naval Postgraduate School, Maple is the software used most commonly in the Mathematics Department, requiring a similar instructing tool. Mathematical Modeling Using Maple follows the lead of *Mathematical Modeling With Minitab*, and assists the student in grasping the concepts of the modeling class without getting slowed down by the syntax of Maple.

TABLE OF CONTENTS

I. AN INTRODUCTION TO MAPLE

A. BACKGROUND

Experimental mathematical modeling, a natural stepping stone to statistical analysis, has an obvious coupling with computers, which can quickly solve equations, and plot and display data to assist in model test and evaluation. The software computer algebra system, Maple, is a powerful tool to assist in this process. When dealing with real world problems, data requirements can be immense. When evaluating immigration trends, for example, and the political, social and economic effects of these trends, thousands of data points are used; in some cases, millions of data points. Such problems cannot be analyzed by hand, effectively or efficiently. The manipulation required to plot, curve fit, and statistically analyze with goodness of fit techniques, cannot feasibly be done without the assistance of a computer software system.

Maple is designed to service two main calls for software service; namely, DOS and Microsoft Windows users. As a result of the different formats, there are many subtle and many obvious differences between the two systems, and occasionally there are differences in the Maple commands and output from one system to the other. This manuscript is intended to assist with the Microsoft Windows version of Maple, which is identical to the system found on NPS Sun systems. This manuscript is also suitable as a reference source for the DOS user; although there may be small differences in the coding syntax. The Windows version of Maple allows the user to create documents by combining input, output, graphics and text.

The Maple system is easy to learn and can be applied in many mathematical applications. While these demonstrate its versatility, Maple is also an extremely powerful software package. Maple provides over 2500 built-in mathematical functions, covering every mathematical interest: calculus, differential equations, linear algebra, statistics, and group theory to mention only a few. The statistical package reduces many standard time-consuming statistical questions into one step solutions, including mean, median, percentile, kurtosis, moments, variance, standard deviation, and so forth. There are many references for Maple, and a short list would include:

- Maple Quick Reference
- Maple Flight Manual
- Maple Language Reference Manual
- Maple Library Reference Manual
- Maple Release 3 Release Notes
- Maple Release 3 Getting Started

B. THE STRUCTURE OF MAPLE

Maple is an example of a Computer Algebra System (CAS). It is composed of thousands of commands, to execute operations in algebra, calculus, differential equations, discrete mathematics, geometry, linear algebra, numerical analysis, linear programming, statistics, and graphing. It has been logically designed to minimize storage allocation, while remaining user friendly. Maple allows a user to solve and evaluate complicated equations and calculations, analytically or numerically, such as optimization problems, least square solutions to equations, and solving equations that involve Gamma functions, and other special functions.

C. NOTATION AND CONVENTIONS

Throughout this manuscript, different type fonts and styles are used to distinguish between Maple commands, Maple output, and other information. *Italics* are used to indicate Maple commands and the output will immediately follow the Maple commands. In the first example in Table 1, the variable *a* has been assigned the value $2x^3 + \frac{5}{6}x$. Notice that the symbol := is used to indicate this assignment. In the second example, the value of *a*, or $2x^3 + \frac{5}{6}x$, is differentiated with respect to *x*.

Command	Output
*a:=2*x^3 + 5*x/6;* *diff(a,x);*	$a := 2x^3 + \frac{5}{6}x$ $6x^2 + \frac{5}{6}$

Table 1. Maple Command Examples.

This chapter will assist in the description of the user interface features of Maple. A brief review of standard Window's terms will be conducted prior to a description of the Maple software. "Clicking" refers to a single click of the button on the computer mouse when the mouse is located on top of the item of interest on the computer screen. A double-click is simply clicking twice very quickly on top of the item of interest.

D. A GENERAL INTRODUCTION TO MAPLE

Maple has five types of windows: the worksheet window, help window, 2-D window, 3-D window, and the animation window. In this manuscript, the worksheet, help and 2-D windows are used most often.

The worksheet is where all interaction between Maple and the user occurs. Within a worksheet, commands (input) and text (remarks for clarification) are entered by the user, and results numerical or symbolic (output and graphics) are produced. The user may manipulate these interactions to create a flowing document, which can be saved by click File, then click SaveAs followed by naming the document. The name of the document can be any word or group of letters and/or numbers which has a length of less than nine characters. Once a document has initially been saved, it can be retrieved by clicking File, clicking Open, and entering the document name. Then it can be re-saved after modification by clicking File, followed by clicking Save.

The input and text regions of the worksheet can be modified to change a document, but the graphic and output regions cannot be modified once Maple inserts them into a document. The input region is identified by the > prompt which proceeds all command entries into Maple. (Note: Maple only recognizes the Maple generated > prompt. If the symbol > itself is typed by the user, Maple does not respond to it as an input prompt, but as the "greater than" relation.) The input commands, output characters and text regions are all of different font size and color to assist the user in distinguishing between them. Text regions assist in documentation and explanation of the input/output regions of the document and they may be placed anywhere in a document. Graphic regions, once they are generated by input commands, can be copied and pasted into a worksheet. Once the graphic is pasted into the worksheet, it can no longer be edited or

manipulated. The output regions are generated by the user's input commands and cannot be manipulated once they appear in a document, although the user is allowed to delete these results.

The Maple menu bar is located at the top of the screen, immediately below the Maple title. The menu bar provides easy access and collocation of many commonly used options. The menu bar is composed of File, Edit, View, Insert, Format, Options, Window and Help, much like any Windows software menu. Immediately below the Maple menu bar is the Maple tool bar. The tool bar provides accelerated access to the most commonly used options.

Maple syntax requires either a colon or a semicolon following every statement. Maple will not process any command without either a colon or a semicolon. Multiple statements may be on the same line, but each must have its own colon or semicolon. The colon suppresses output of the command, while the semicolon signals that the results are to be printed to the screen immediately after the enter key for the command is pressed. There is a large set of commands either readily available in Maple memory or stored separately in Maple packages, which assist in more efficient memory storage. Standard commands such as addition and multiplication, are not contained in packages. When using commands stored in packages (such as graph plotting commands), the command *with(package name)*: is issued prior to using any of the commands in that package. This command is required only once. A few of the Maple packages will be used in this manuscript. Specifically, *plot,* will be used for all plotting commands, *linalg*, for all linear algebra commands, and *stats*, for all linear regression commands.

1. The Help Command

The Maple help database can provide all information found in the Maple Library Reference Manual. However, help can be obtained immediately to assist the user in solving problems without leaving the document. Help can be acquired by a number of methods: click on Help in the menu bar, type help at the > prompt, or type ? at the > prompt. By using the ? or help at the > prompt, the user must type the keyword for the help search. If a specific syntax command is in question, (for example, the user desires to learn Maple's syntax for differentiation) the user need only type *?differentiate* at the > prompt. This procedure is perhaps the most convenient one for help on syntax. The Help on the menu bar gives the user more options.

2. Data Entry

Commonly, a set of data will describe a process. Entering the data is the first step to analyzing the data. Maple provides a convenient method for manually entering data via a list. A list is a group of data to which Maple's many operations are applied. Suppose a group of five college students' ages are known; 18, 20, 25, 19, 19. Table 2 demonstrates the command required to enter the data into Maple.

Command	Output
ages:=[18,20,25,19,19];	ages:=[18,20,25,19,19]

Table 2. Entering A Data List.

The use of brackets in the command indicates a list. The brackets will maintain an initial ordering of the data and allows for duplicate values, and may be manipulated by the methods described in Section D.3.a of this chapter. Commas are required between elements, neither blanks, nor periods, nor any other separators may be substituted. Any alphanumeric may be included in a list, to include integers and rational numbers. If any data should be missing, a place holder can be entered in its place, such as the letter x. If there was a sixth student in the group but the age was unknown, the symbol x could be used to represent the sixth student's age. Table 3 presents the required command.

Command	Output
ages:=[18,20,25,19,19,x];	ages:=[18,20,25,19,19,x]

Table 3. Entering Data With A Missing Data Point.

a. *Recovering Data From A File*

Typically, data collection is done by an outside agency and the analyst analyzes the data that has been collected. When a survey is conducted on the state or national level, the number of data points is typically in the thousands; which would lead to tremendous time loss if manual input was required. To alleviate this problem, Maple has two separate commands to recover data from a file, *importdata* and *readdata*.

Importdata is part of Maple's statistical package and requires the *with(stats):* command. To use *importdata,* the file name and the number of columns or data sets are required, the default is one data set. Missing data in the file will be

converted to the keyword "missing." Given a data file, named "data1," which contains two sets of data {2,4,6} and {3,5} with the third element in the second set missing. Table 4 presents the Maple command and output.

Command	Output
newdata:=importdata(`data1`,2);	newdata:=[2.0, 4.0, 6.0], [3.0, 5.0, 'missing']

Table 4. Reading Data From A File Using *Importdata*.

Readdata is a more flexible command. The file name, number of columns and type of data are required to use the command. The type of data indicates either integer or float, with float being the default. *Readdata* requires a separate command prior to use, (similar to the *with( ):* command); the command is *readlib(readdata):* which is required only once. An example of this command follows, given a data file named "data" containing 12 data points in three columns. Table 5 presents four separate commands which describe the possible manipulation of the data file using the *readdata* command. The file "data2" is composed of three columns of data, each including four data points, described in the three sets: {1,11,15,17}, {3,22,5,78}, and {5,55,55,70}.

Command	Action
readdata(data2,3);	read 3 columns of floats from the file 'data'
readdata(data2,integer,3);	read 3 columns of integers
readdata(data2,integer);	read first column of integers
readdata(data2);	read first column of floats

Table 5. Reading Data From A File Using *Readdata*.

b. *Data Entry And Verification*

Figure 1 illustrates entering, verifying and naming data pertaining to the length and weight of bass caught during a fishing derby. In Chapter III, several models for predicting the weight of a bass as a function of some readily measurable dimension are suggested. In Figure 1 the data is entered in columns, the columns are named, and the data is printed to verify correct entry.

```
len:=[14.5,12.5,17.25,14.5,12.625,17.75,14.125,12.625]:
wt:=[27,17,41,26,17,49,23,16]:
bass:=array(1..2,1..8,[len,wt]);
```

$$bass := \begin{bmatrix} 14.5 & 12.5 & 17.25 & 14.5 & 12.625 & 17.75 & 14.125 & 12.625 \\ 27 & 17 & 41 & 26 & 17 & 49 & 23 & 16 \end{bmatrix}$$

If titles for columns are desired:

```
len:=[length,14.5,12.5,17.25,14.5,12.625,17.75,14.125,12.625]:
wt:=[weight,27,17,41,26,17,49,23,16]:
bass:=array(1..2,1..9,[len,wt]);
```

$$bass := \begin{bmatrix} length & 14.5 & 12.5 & 17.25 & 14.5 & 12.625 & 17.75 & 14.125 & 12.625 \\ weight & 27 & 17 & 41 & 26 & 17 & 49 & 23 & 16 \end{bmatrix}$$

Figure 1. Data Entry And Verification.

c. *Correcting Erroneous Entries*

In the above illustrations, Maple displayed the bass data immediately after its input. This is done to help the user verify that all elements have been entered correctly. Reentry of each command is typically the best method for correcting an erroneous entry with many errors. However, there are other commands to expedite smaller scale

error correction, including the use of the following commands: *op* and *subsop*. This section demonstrates that by using these two commands on a defined data set, data can be inserted, deleted and replaced with ease. Once the data set has been entered, a data point may have been left out or forgotten. Appending an element to the *wt* data set from Figure 1 is done by the command, *[op(wt), X]*, where X is the new element.

Used in a slightly different fashion, *op* can also select a portion of the data set. This is done by, *op(j..k, wt)*, where *wt* is the data set and j and k are row values in the data set, and j < k. As a result, the command selects a range from the *wt* data set. Table 6 presents two methods of selecting this range.

Command	Output
newweight:= op(2..5,wt);	newweight:= 17, 41, 26, 17
newweight:= wt[2..5];	newweight:= [17, 41, 26, 17]

Table 6. Selecting A Range From A Data Set.

Subsop has two main purposes; to replace a data point with a new value and to delete an element from a data set. Table 7 demonstrates an example of each, using the *wt* data set.

Command	Output
newwt:=subsop(2=15,wt);	newwt:=[27, 15, 41, 26, 17, 49, 23, 16]
newwt:=subsop(7=NULL,wt);	newwt:=[27, 17, 41, 26, 17, 49, 16]

Table 7. Replacing Data In A Data Set.

3. Transformations And Functions

As described in Section C of this chapter, the symbol := is used to assign the value on the right hand side of the statement to the name on the left hand side of the statement. An example, *x:=3;* assigns the value 3 to *x*, to unassign *x*, use the command *x:='x'*. Functions can be defined using the mapping arrow symbol →. The function can be evaluated numerically or symbolically, Table 8 provides a short example.

Command	Output
f:= x -> x^2+2;	f:= x-> $x^2 + 2$
f(5);	27
f(x-1);	$(x-1)^2 + 2$

Table 8. The Mapping Command.

A mathematical model suggests an existing relationship among the selected variables. In the two dimensional case, the model suggests a functional relationship between a dependent and an independent variable. Given values of an independent variable, a function can transform the given data to yield predicted values for the dependent variable. Transforming data requires the understanding of algebraic operations and functions that are used in Maple. Table 9 presents the regular algebraic operations that Maple recognizes.

Symbol	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponents

Table 9. Maple Algebraic Functions.

Many other functions are also recognized by Maple. Table 10 is an abbreviated listing.

Command	Operation
abs	absolute value of real or complex argument
argument	argument of a complex number
ceil(x)	smallest integer ≥ x
conjugate	conjugate of a complex number
exp	the exponential function: exp(x) = sum(x^i/i!,i=0..∞)
factorial	the factorial function, factorial(n) = n!
floor	floor(x) = greatest integer ≤ x
ln	natural logarithm (with base E = 2.718...)
log	logarithm to arbitrary base
log10	log to the base 10
max, min	maximum/minimum of a list of real values
RootOf	function for expressing roots of algebraic expressions

Table 10. Other Maple Functions.

Some other functions that are available will be discussed in the next chapter when considering statistical operations that may be performed on columns of data: sums, mean, standard deviation, and so forth. Table 11 presents a few examples of data transformation, with the correlating Maple commands.

Expression	Maple Command
x^2	*x^2*
$2x^2 + 2.5x + 9$	*2*x^2 + 2.5*x + 9*
$(x^2 + 2)^{0.5}$	*(x^2 + 2) ^ (.5)*

Table 11. Examples Of Maple Commands.

a. *Column Operations*

In this section, the operations that can be performed directly on worksheet columns are presented. Among the available operations are: *matadd, evalm,* and *scalarmul.* These operations all require the *with(linalg):* command prior to use. Table 12 demonstrates these commands with six examples using two columns of data; c1:=[1,2,3] and c2:=[4,5,6].

Operation	Command	Output
Summing Columns: c1 + c2	*c3:= matadd(c1,c2,1,1);*	c3:=[5 7 9]
Summing a Constant Into a Column	*c4:= evalm(3+c2);*	c4:=[7 8 9]
Multiplying a Constant Into a Column	*c5:= scalarmul(c1, 3);*	c5:=[3 6 9]
Adding Multiples of Columns	*c6:=matadd(c1,c2,4,2);*	c6:=[12 18 24]
Using a Function on a Column	*c7:=map(x->ln(x),c1): evalf(");*	c7:=[0 .693 1.099]
Taking Each Value in a Column to a Power	*c8:=map(x->x^2,c1);*	c8:=[1 4 9]

Table 12. Column Operations.

4. Arrays and Matrices

Arrays and matrices are structured devices used to store and manipulate data. An array is a specialization of a table; a matrix is a two dimensional array. Both *array* and *matrix* are part of the linear algebra package, and require the *with(linalg):* command prior to use. Table 13 presents a few examples of the use of the *array* and *matrix* commands.

Command	Output
with(linalg): *a:=array([1,2,2,3]);* *b:=array([[1,2],[3,5]]);* *c:=array(1..5,[9,3,1,8,3]);* *d:=matrix([[7,3],[2,3]]);*	a := [1 2 2 3] $b := \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix}$ c := [9 3 1 8 3] $d := \begin{bmatrix} 7 & 3 \\ 2 & 3 \end{bmatrix}$

Table 13. Array And Matrix Commands.

To observe multiple columns of data in a chart an *array* or *matrix* command may be used. Table 14 presents an example with output.

Command	Output
a:=['i','i^3']$'i'=1..4: *with(linalg):* *achart:=matrix(4,2,[a]);*	no output no output achart = $\begin{bmatrix} 1 & 1 \\ 2 & 8 \\ 3 & 27 \\ 4 & 64 \end{bmatrix}$

Table 14. Charts Using Matrices.

5. Saving And Printing A Worksheet

To start Maple from Windows, enter Maple by double-clicking on the Maple icon. Once Maple has been started, it will automatically open an empty worksheet, with a flashing cursor to the right of a character prompt >. To save the worksheet, click on File and then click on SaveAs and then specify a name for the worksheet. Once the worksheet

has been named and saved, click on File and then click on Save to re-save the worksheet. To open a previously saved worksheet, click on File and then click on Open, and then specify the name of the worksheet.

After re-opening a document, the document will contain only the commands and not have the initial results. This is because of Maple's kernel, or internal state. After a command is executed, the result is stored in memory (the kernel). If the document is closed and then reopened, Maple recovers the commands but does not recall the results, unless the kernel itself is saved. Saving the kernel is done by selecting "Save Kernel State" under the Options menu followed by saving the document. This saves some time if you are reusing the same document time and time again; however, it does take up a lot of memory in the computer.

After the completion of a document, involving commands and results, the document can be printed by clicking File, clicking Print (which calls up a print window), and then clicking again on print.

To exit Maple click on File and then click on Exit, or type *quit*, *done*, or *stop* at the Maple prompt. Note: there is no opportunity to save your worksheet using one of these latter three commands, but saving your worksheet file is automatic by simply clicking on Exit.

II. GRAPHING MODELS

A. GRAPHICAL OVERVIEW

> In the mathematical modeling process, it is often desirable to graph a model in order to gain a qualitative feel for interpreting the model, or perhaps to use the graph for making predictions. In another situation you may wish to test a proposed model against some observed data. It is helpful, in such cases, to plot the observed data and overlay the data scatterplot on the graph of the model. (Fox, et al., Mathematical Modeling With Minitab, page 5)

By evaluating a graph, many interesting properties can be determined, including detecting local maxima and minima. In this chapter, an in depth review of the Maple commands required to accomplish these tasks is presented.

Maple has an extremely detailed and developed plot command, which provides plots in both two and three dimensions. For these modeling purposes, 2-D plots will typically be used. The plot function is called by *plot(f, hr, vr, options)*, where *f* is the function to be plotted, *hr* is the horizontal range and *vr* is the vertical range. Additionally, many other calls can be added after the vertical range to control a variety of options. Table 15 presents a short list of these options (defaults are in bold font).

Option	Description
scaling =	constrained or **unconstrained**
style =	point, **line,** patch or patchnogrid
title = 'title'	**no title**
thickness =	**0**, 1, 2, or 3
axes =	framed, boxed, **normal** or none
view =	[xmin..xmax, ymin..ymax], **entire curve**

Table 15. Plot Command Options.

1. Continuous Plots

The plot command, *plot(f);* will generate a 2-D plot of *f* with a default range of -10..10 on the x-axis. No other command information is required; however, the axes ranges and options may be specified to generate a specific plot. The default range can be specified as a finite range or an infinite range. By constraining the scale, equal units occur in both the x and the y direction. However, a plot is generally easier to see when the scale is unconstrained, although it would be distorted (e.g., a circle would appear as an ellipse.) Maple automatically scales the axes to spread the data over as large a space as possible, but this procedure does not imply that the area of interest will be plotted most effectively. As a result, the view option must be employed to ensure that the correct portion of the plot is best displayed. To demonstrate the plot command with a variety of options, a few examples are provided in Figures 2, 3 and 4. Note the distortion in both Figures 2 and 3.

plot(sin,style=point);

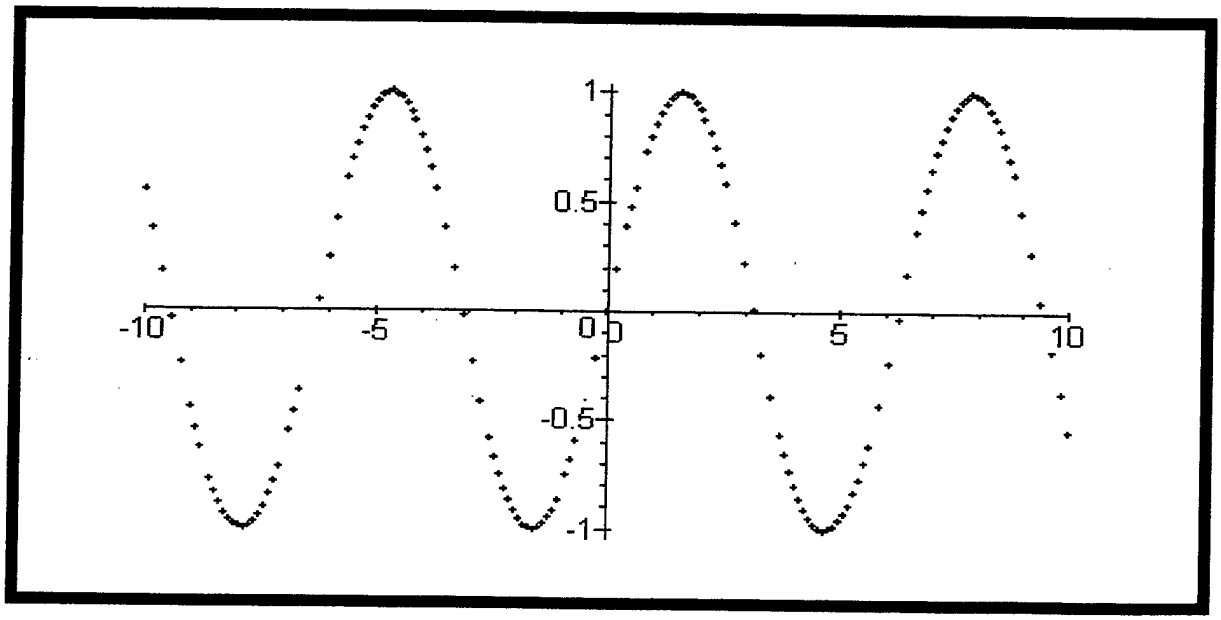

Figure 2. A Sine Plot With The Default Range.

plot(sin(x),x=0..infinity,scaling = unconstrained, axes= boxed);

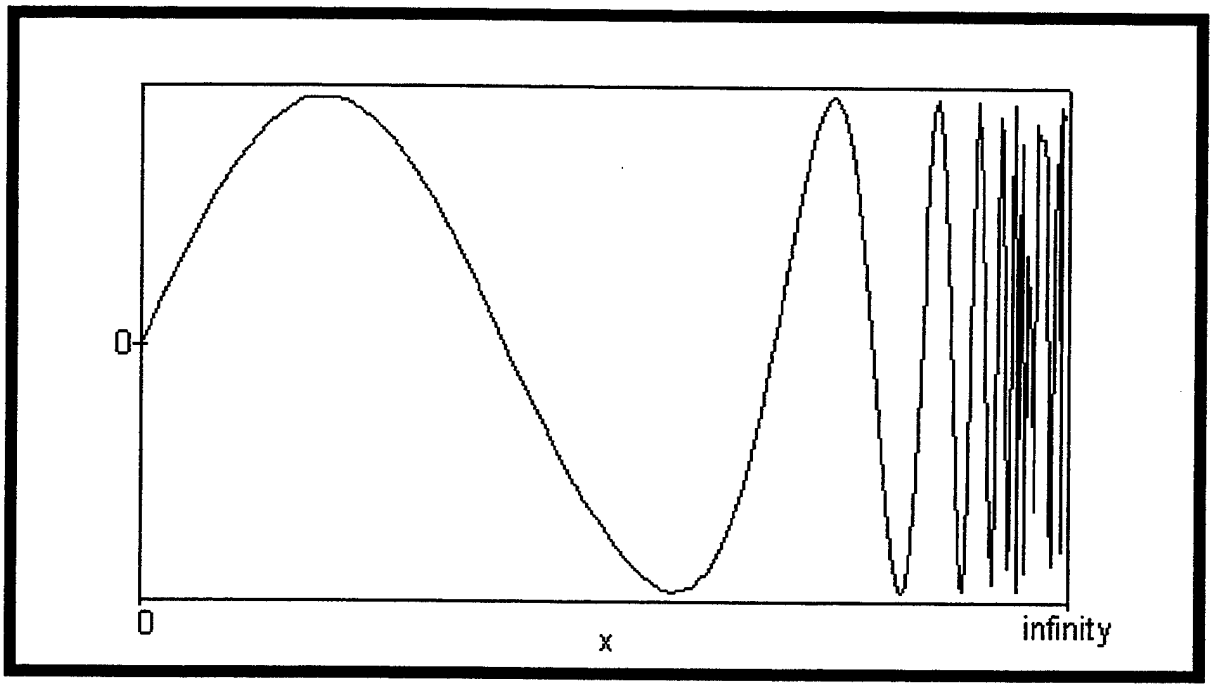

Figure 3. A Sine Plot With An Infinite Range.

*plot(sin(x),x=0..2*Pi,scaling=constrained);*

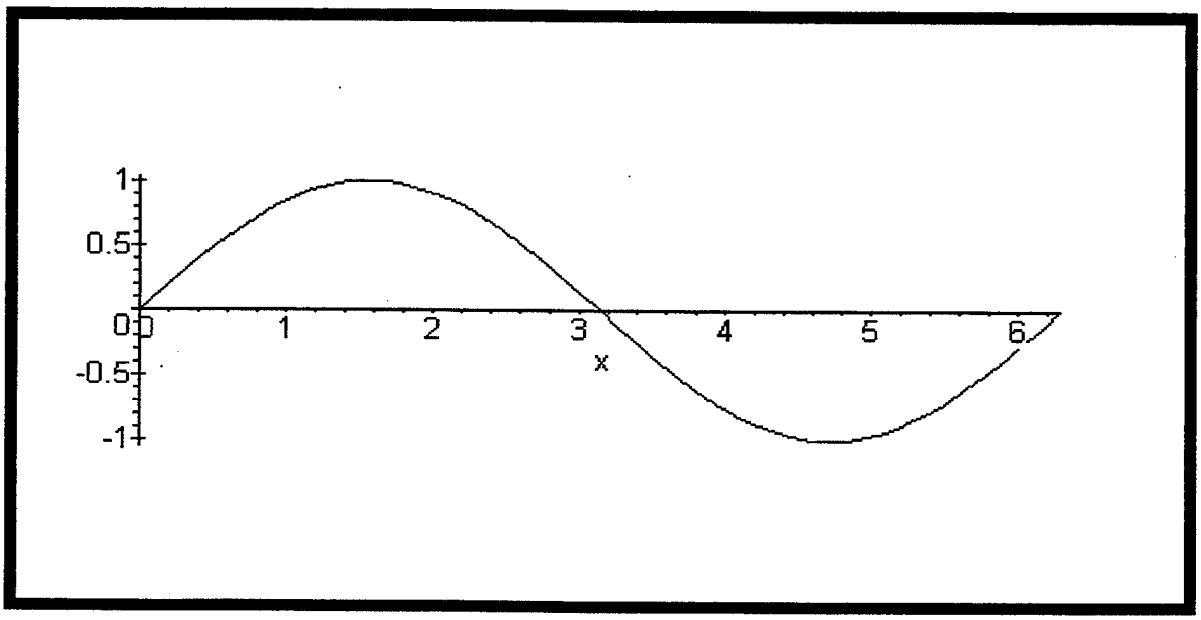

Figure 4. A Sine Plot With A Constrained Scale.

2. Scatterplots

The previous plots demonstrate functions with continuous x values; either defaulted to -10..10 or selected by the user. However, Maple can also plot discontinuous sets of data. Using the data in Table 16, Figure 5 presents an example of plotting the ages of five people versus their respective weights.

Age (years)	1	5	13	17	24
Weight (lbs.)	15	40	90	160	180

Table 16. Age-Weight Data.

```
age:=[1,5,13,17,24]:
wt:=[15,40,90,160,180]:
agewt:={seq([age[i],wt[i]],i=1..5)}:
plot(agewt,style=point, symbol =diamond);
```

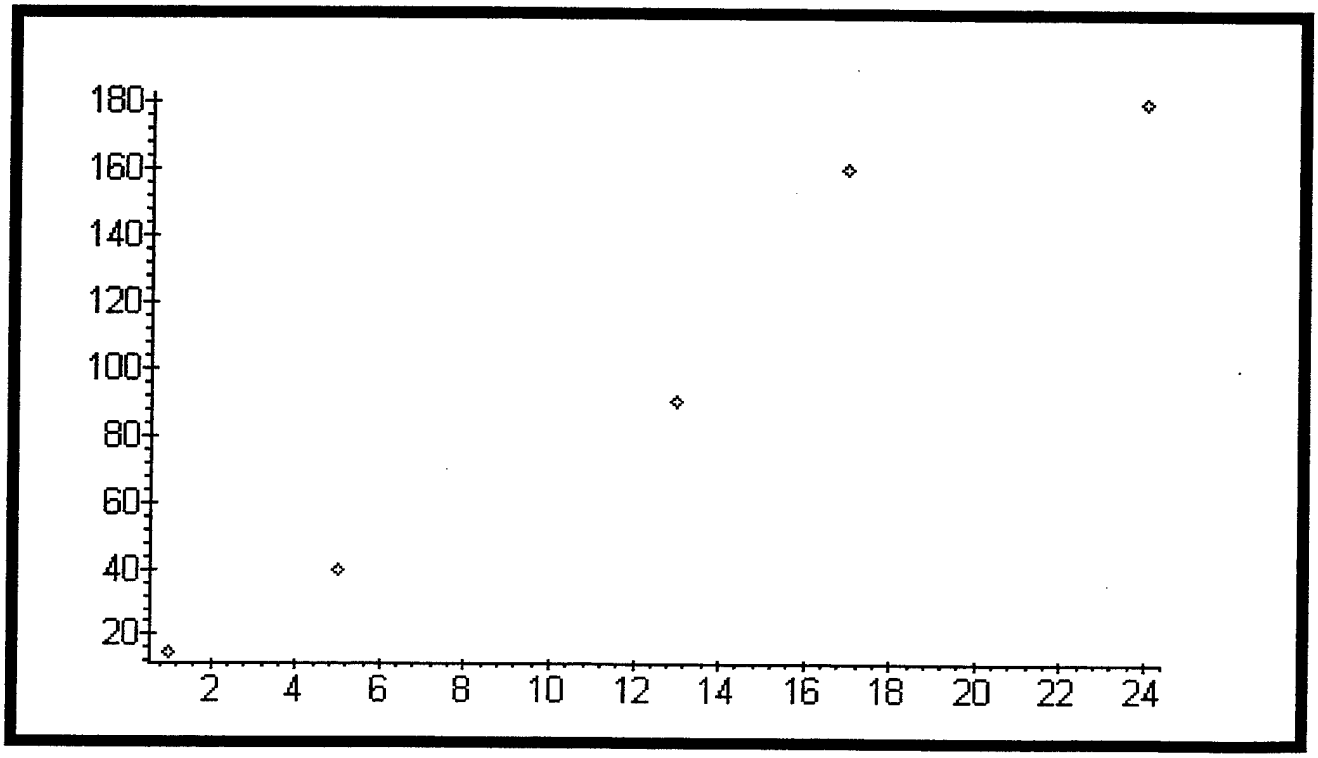

Figure 5. Age vs. Weight Data Plot.

3. Multiple Plots

It is also an option to plot multiple functions on one set of axes. One method requires that both functions have the same domain, presented in Figure 6. The second method uses the display command which requires the *with(plots):* command be called once. This method does not restrict the domains, as presented in Figure 7.

plot({sin(x),x^(.5)}, x=0..5, scaling = constrained, axes = frame);

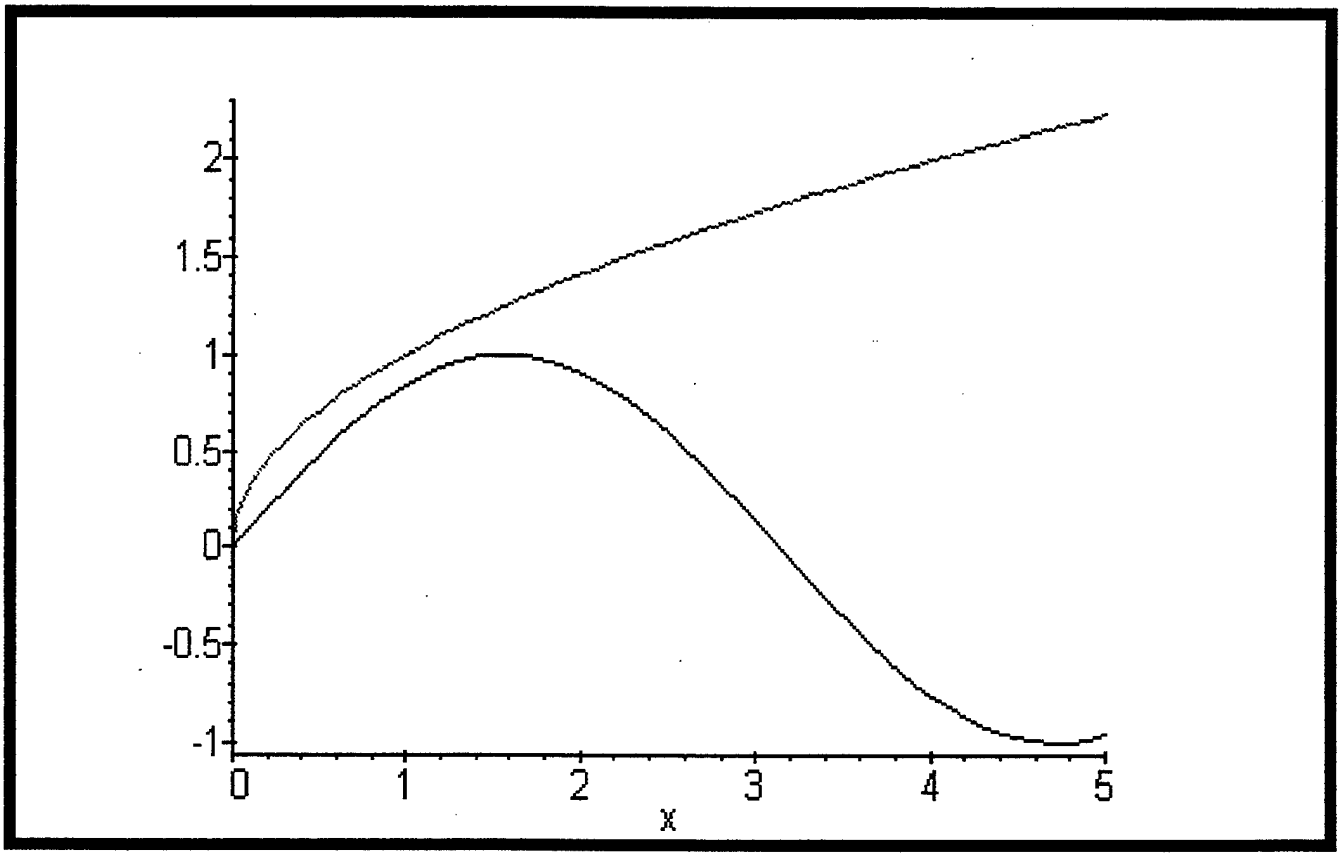

Figure 6. Multiple Functions On One Plot.

```
with(plots):
plot1:=plot(agewt,style=point, symbol =diamond):
plot2:=plot(30*sin(x), x=0..30):
display({plot1,plot2});
```

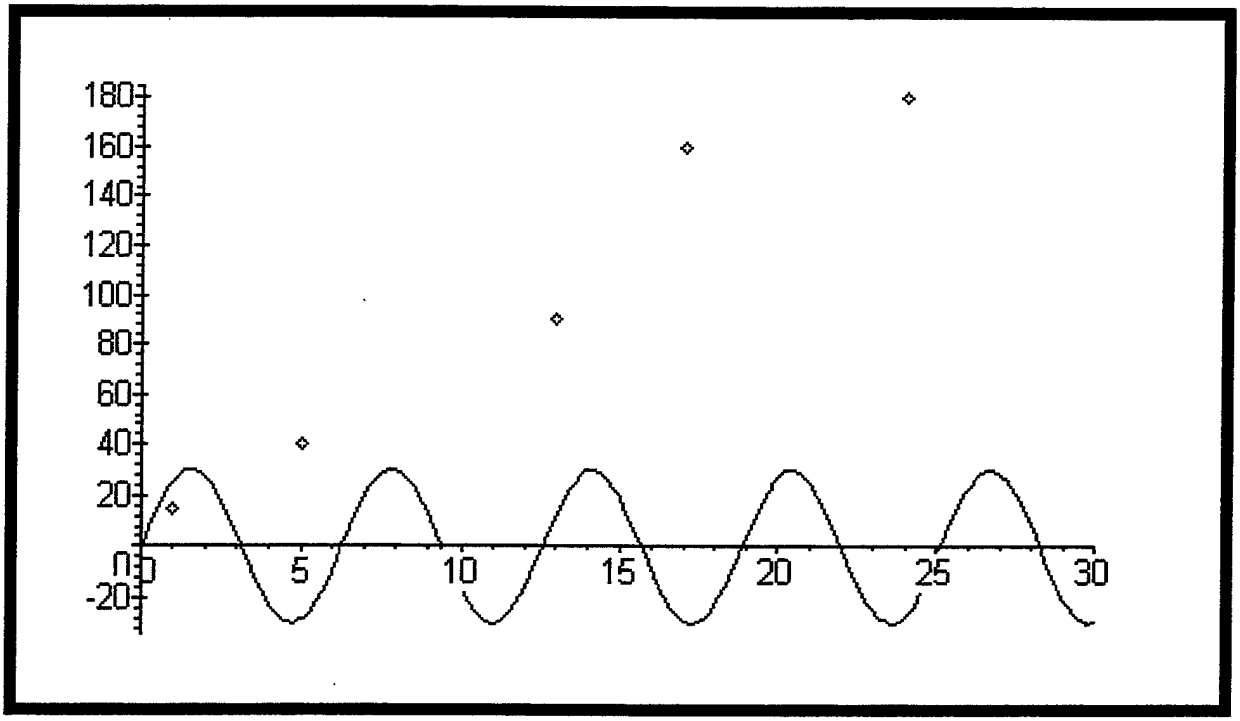

Figure 7. Multiple Functions On One Plot With Different Domains.

B. AN ILLUSTRATIVE EXAMPLE: MINIMIZING THE COST OF DELIVERY AND STORAGE

In this example, the graphing commands previously presented are used in a real world situation, minimizing the cost of gasoline. By graphically analyzing the amount of gasoline and interval of delivery of the gasoline, a service station can eliminate unnecessary costs, and sell the gasoline at a price which will capture a larger portion of the market. The two main costs for the service station are delivery costs and storage costs. The delivery cost is a charge per delivery which is independent of the amount of gasoline delivered. From Giordano and Weir, A First Course In Mathematical Modeling , page 136, the following model represents the average daily cost of gasoline: $c = \frac{d}{t} + \frac{srt}{2}$ Table 17 defines the variables used in this example.

Variable	Description
c	average daily cost
d	delivery cost in dollars per delivery
t	time in days between deliveries
s	storage cost per gallon per day
r	demand rate in gallons per day

Table 17. The Variables Of The Delivery And Storage Example.

As previously stated a service station wishes to minimize the average daily cost, c. From the above equation, this can be done by minimizing d, s, r, and t. The first summand of the model describes the effect of the delivery cost on the average daily cost, while the second summand describes the effect of the storage cost on the average daily cost. By plotting each summand separately as t changes, the model can be more completely understood. In this example, the storage cost, delivery cost and demand rate are known and are $s = 0.1$, $d = 800$ and $r = 1000$. Table 18 presents the Maple commands required to transform the submodels (Fox, et al., op. cit. , page 21).

Command	Output
s:=0.1;	s := 0.1
d:=800;	d := 800
r:=1000;	r := 1000
delivery:= d/t;	delivery := 800/t
*storage := s*r*t/2;*	storage := 50.0 t
daily := delivery + storage;	daily := 800/t + 50.0 t

Table 18. The Transformation Commands.

Now that the functions are ready to be plotted, a range must be specified. Having no feel for the models, the first plot will enter a range, which will be evaluated and changed accordingly to ensure the range for the next plot is more accurate for this model. After each plot, this process is repeated until the range is satisfactory for accurate analysis. Table 19 presents the commands to evaluate the range of each plot. Figures 8, 9 and 10 display the result of each command.

Commands
plot(delivery, t= 0..100);
plot(storage, t=0..20);
plot(daily, t=0..50);

Table 19. The Plot Commands.

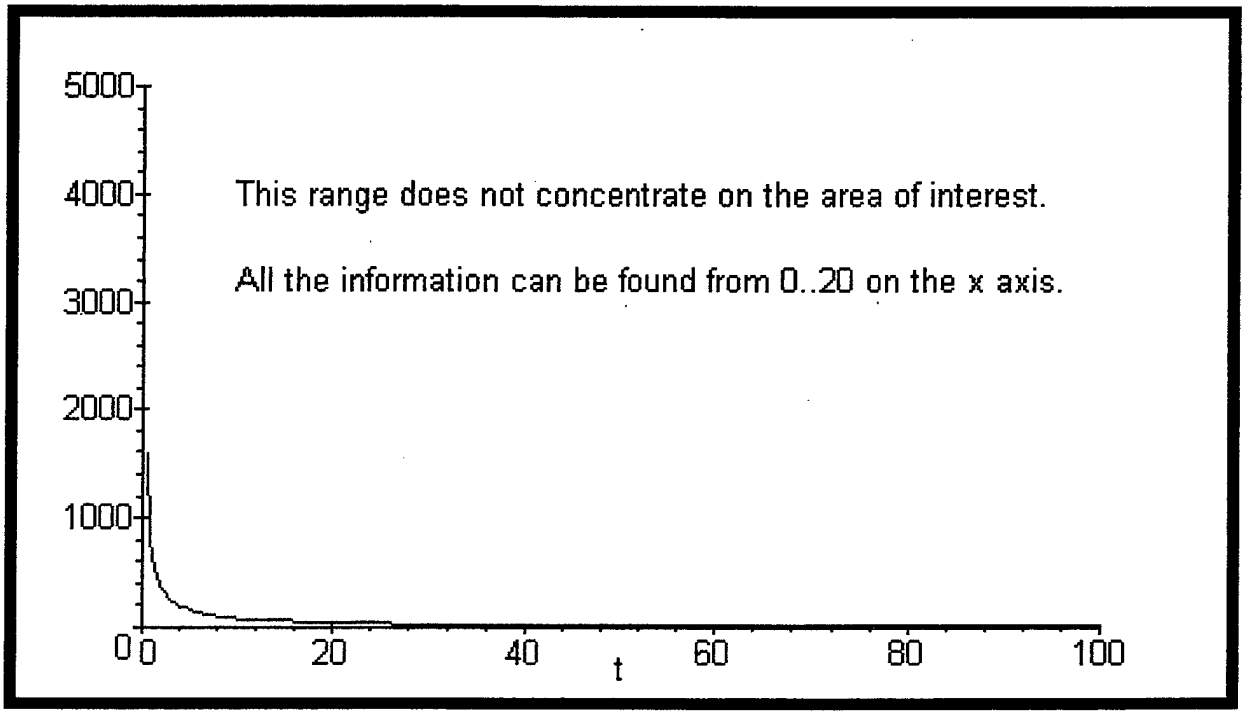

Figure 8. The Delivery Plot With An Initial Range Estimate.

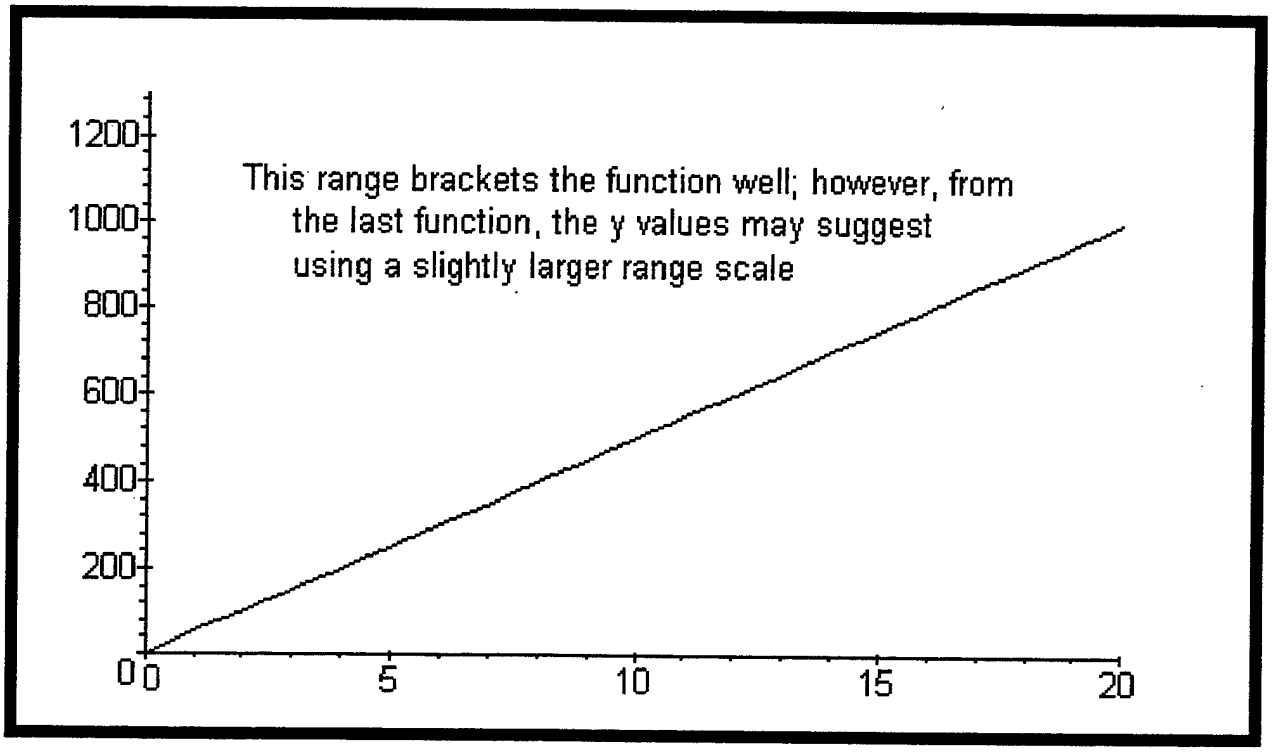

Figure 9. The Storage Plot With A Second Range Estimate.

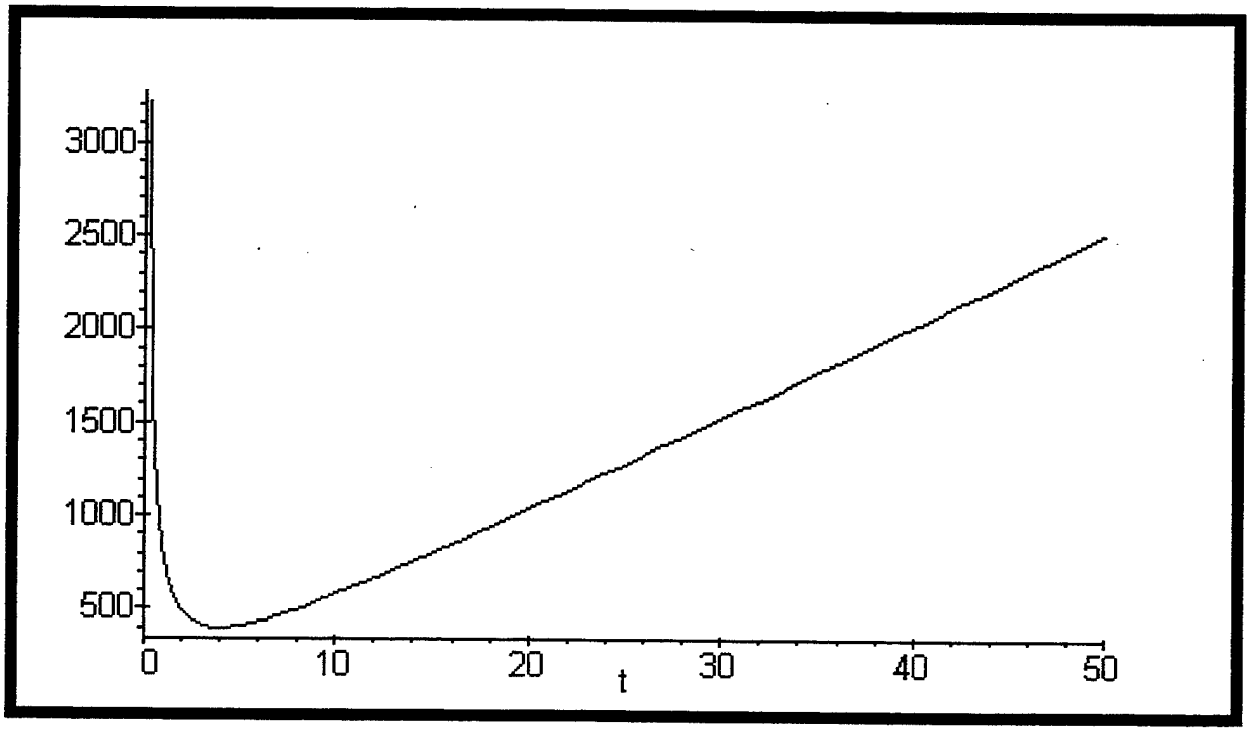

Figure 10. The Daily Plot With An Acceptable Range.

Using the range 0..50, all three functions can then be presented on a single graph for quicker analysis of the effects of the individual variables on the daily cost. Figure 11 presents this combined plot, using the following commands:

plot({delivery, storage, daily}, t= 0..50, axes = boxed);

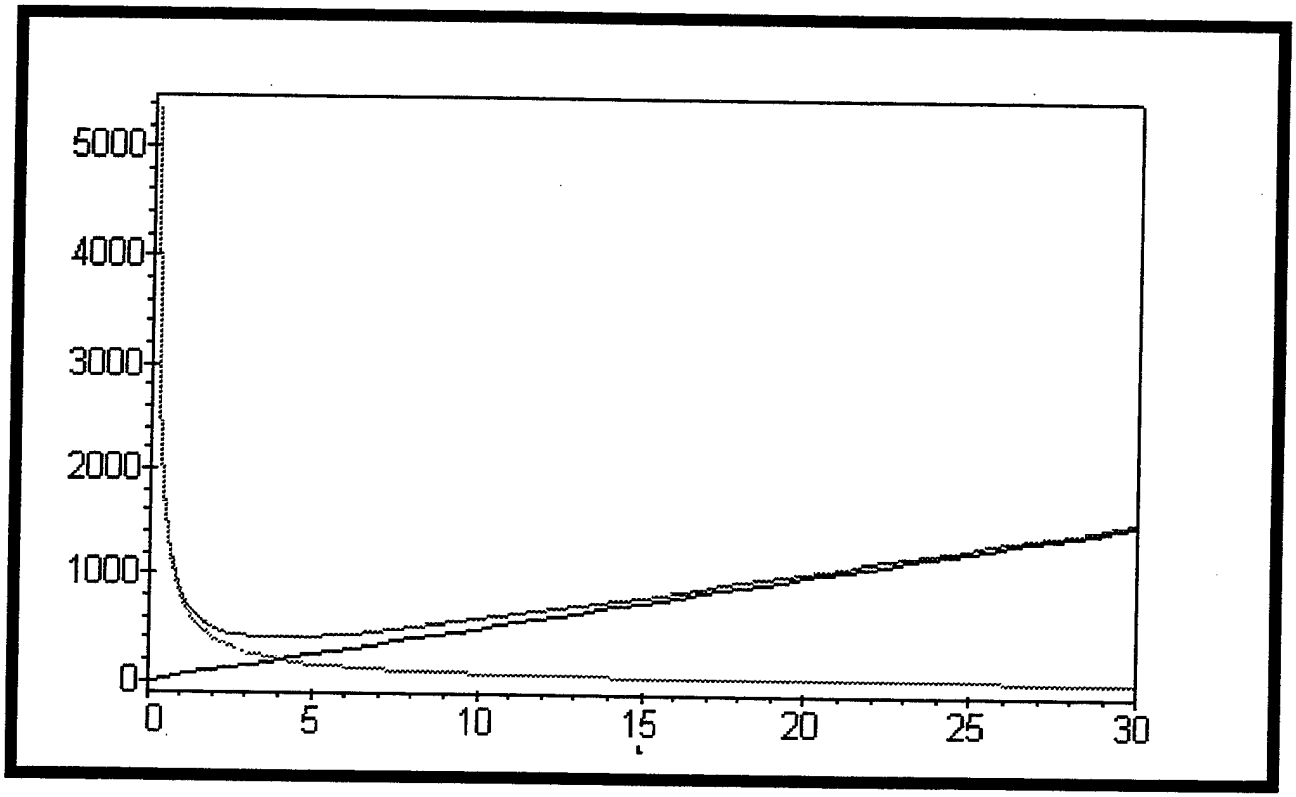

Figure 11. The Combined Plot Of The Daily, Delivery, And Storage Costs.

Immediately, it is noted that before day two, nearly all of the cost is derived from the delivery cost; while after day two, nearly all the cost is derived from the storage cost. This is the critical point, the time between orders that minimizes the average daily cost.

III. MODELING USING PROPORTIONALITY

The simplest method of constructing a model is by applying proportionality arguments to a set of data. By plotting a data set, an initial estimation of a proportionality constant can be acquired directly from the graph. The procedure for determining a proportionality constant for a model is presented in Table 20.

1.	Enter the data observed for the dependent and independent variables.
2.	Plot the raw data points to check for trends and to identify potential data outliers.
3.	Perform the transformations supporting a hypothesized proportionality.
4.	Plot the transformed data to test the hypothesized proportionality.
5.	Estimate the constant of proportionality.

Table 20. Procedure To Estimate Proportionality.

This chapter demonstrates the Maple commands necessary to perform this method of model construction. Section A presents the graphical plotting methods to check the proposed proportionality and the commands for estimating the constants of proportionality. In Section B, several modeling applications are investigated using proportionality. (Fox, et al., op.cit., page 24)

A. TESTING PROPORTIONALITY ARGUMENTS

> Two positive quantities x and y are said to be proportional (to one another) if one quantity is a constant positive multiple of the other; that is, if $y = kx$, for some positive constant k. We write $y \propto x$ in that situation, and say that y 'is proportional to' x. (Giordano and Weir, op.cit., page 50)

This description of proportionality graphically depicts a line through the origin. Thus a quick test of a proportionality model is the plot of the data, which will approximate a straight line projected through the origin. Other types of proportionality models involve transformations, such as $y \propto x^2$ or $y \propto \ln x$. Maple can quickly transform the original data, x, into many such functions, here x^2 and ln x, as displayed in Table 12 in Chapter I. A submodel involving a proportionality argument must approximate a line and it must pass through the origin.

Once a proportionality argument has been successfully identified, the next step is determining the slope of the line that "best" fits the data graphically. Initially a large plot is advantageous to determine if the data approximates a line. Quite often this plot will not include the origin, which necessitates a second plot to verify that the "best" line does pass through the origin. Consequently two plots may be required to accurately plot the transformed data. This procedure is illustrated by several examples.

1. Example 3.1.1: Testing A Proportionality Model

In Chapter 3 of Giordano and Weir, op.cit., a bass fishing derby was analyzed to determine which fishing club member caught the most fish by weight. In this example, an analyst wishes to test the model $vol \propto len^3$. Table 21 displays the data *len* and *vol*, representing length and volume.

len	0.6	1.0	2	4	7	20
vol	0.1	0.7	6	100	210	4000

Table 21. The Length And Volume Data Points.

Step 1 Table 22 presents the entry of the observed data and verification that the input is correct.

Command	Output
len:=[0.6,1,2,4,7,20]; *vol:=[0.1,0.7,6,100,210,4000];* *lenvol:=array(1..2,1..6,[len,vol]);*	len:=[.6, 1, 2, 4, 7, 20] vol:=[.1, .7, 6, 100, 210, 4000] $lenvol := \begin{bmatrix} .6 & 1 & 2 & 4 & 7 & 20 \\ .1 & .7 & 6 & 100 & 210 & 4000 \end{bmatrix}$

Table 22. Step 1.

Step 2 Table 23 presents the commands to plot the data checking for trends and identifying any potential outliers found in Figure 12.

Commands
datalv:={seq([len[i],vol[i]],i=1..6)}: *plot(datalv,style=point, symbol=box);*

Table 23. Step 2.

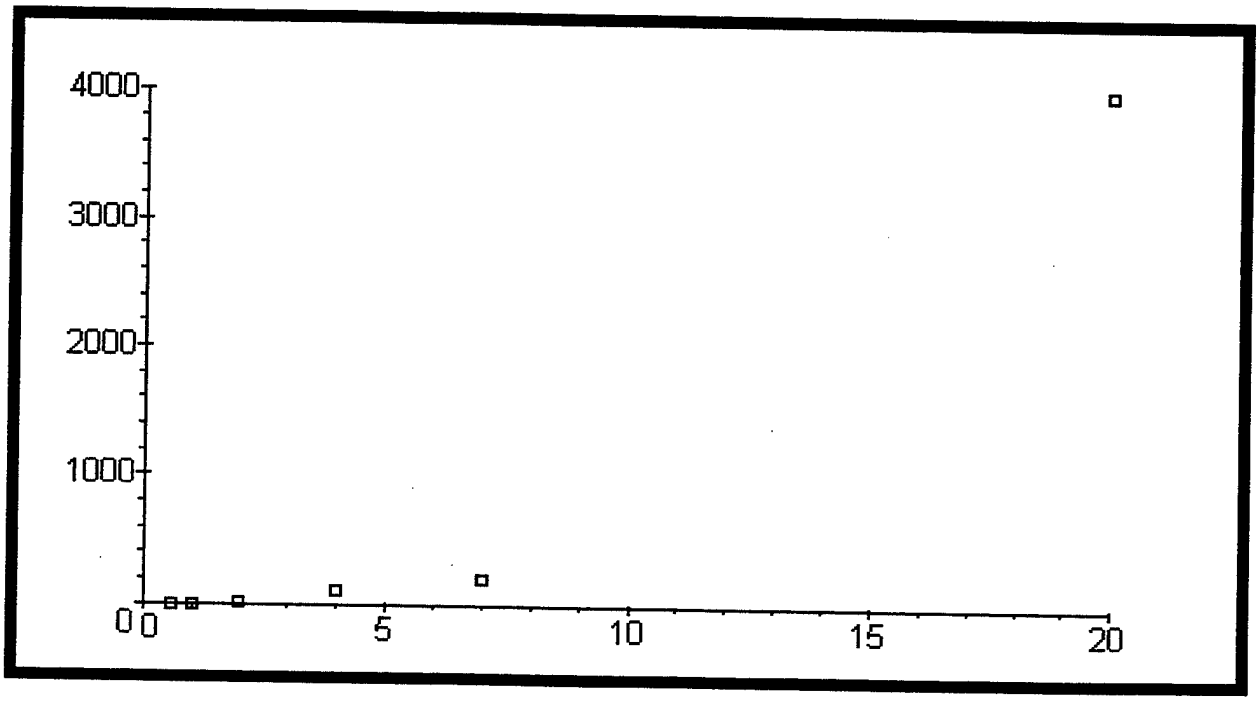

Figure 12. Length vs. Volume.

Step 3 From Figure 12, the data does not fall along a straight line, but it resembles a plot of x^2 or x^3. So try both transformations, to see if one of these relationships exists. Table 24 presents the commands and outputs for the transformations.

Command	Output
len2:=map(x->x^2,len): *len2vol:=array(1..2,1..6,[len2,vol]);*	$len2vol := \begin{bmatrix} .36 & 1 & 4 & 16 & 49 & 400 \\ .1 & .7 & 6 & 100 & 210 & 4000 \end{bmatrix}$
len3:=map(x->x^3,len): *len3vol:=array(1..2,1..6,[len3,vol]);*	$len3vol := \begin{bmatrix} .216 & 1 & 8 & 64 & 343 & 8000 \\ .1 & .7 & 6 & 100 & 210 & 4000 \end{bmatrix}$

Table 24. Step 3.

Step 4a. Table 25 presents the commands required to plot the transformed data to test for a straight line in the transformed data. The plots are presented in Figures 13 and 14.

Commands
data2:={seq([len2[i],vol[i]],i=1..6)}: *plot(data2,style=point, symbol=box);* *data3:={seq([len3[i],vol[i]],i=1..6)}:* *plot(data3,style=point, symbol=circle);*

Table 25. Steps 4a And 4b.

Step 4b. To evaluate if the data projects through the origin, a line is drawn through the data points on the graph. Notice a line does not represent well the len^2 data; however, a line can be drawn through the len^3 data points. Figures 15 and 16 are the same plots as Figures 13 and 14 with a line drawn in by hand to support our conclusions.

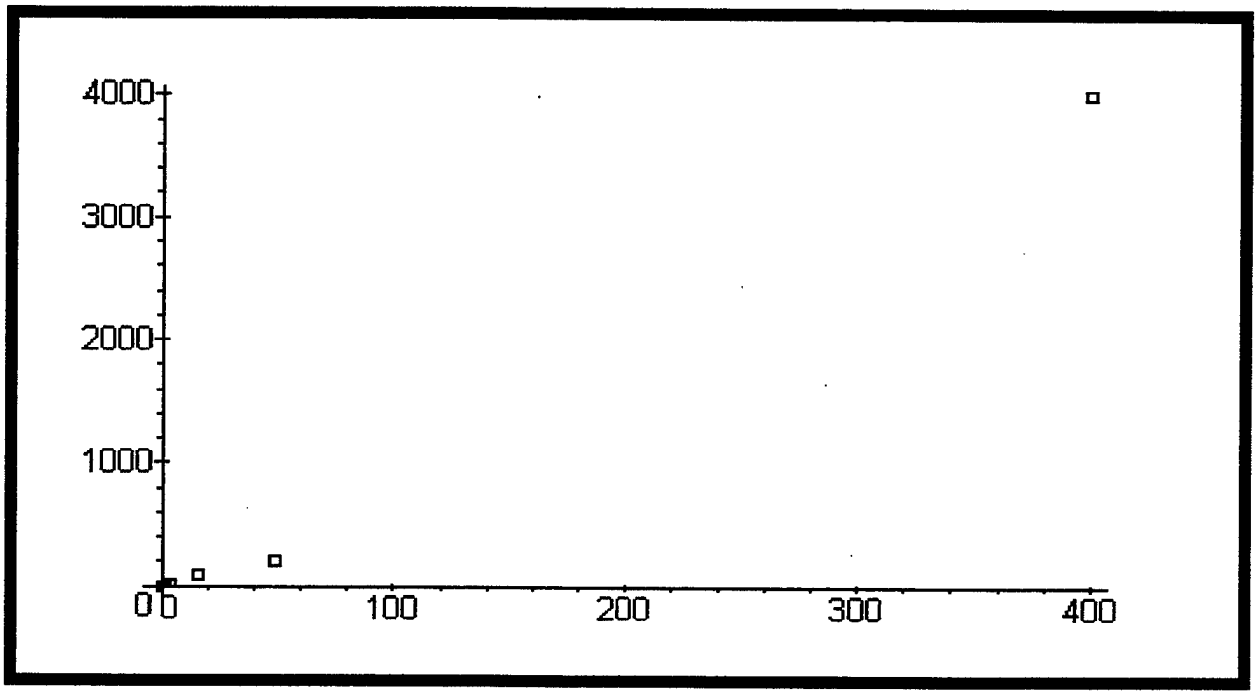

Figure 13. Length^2 vs. Volume.

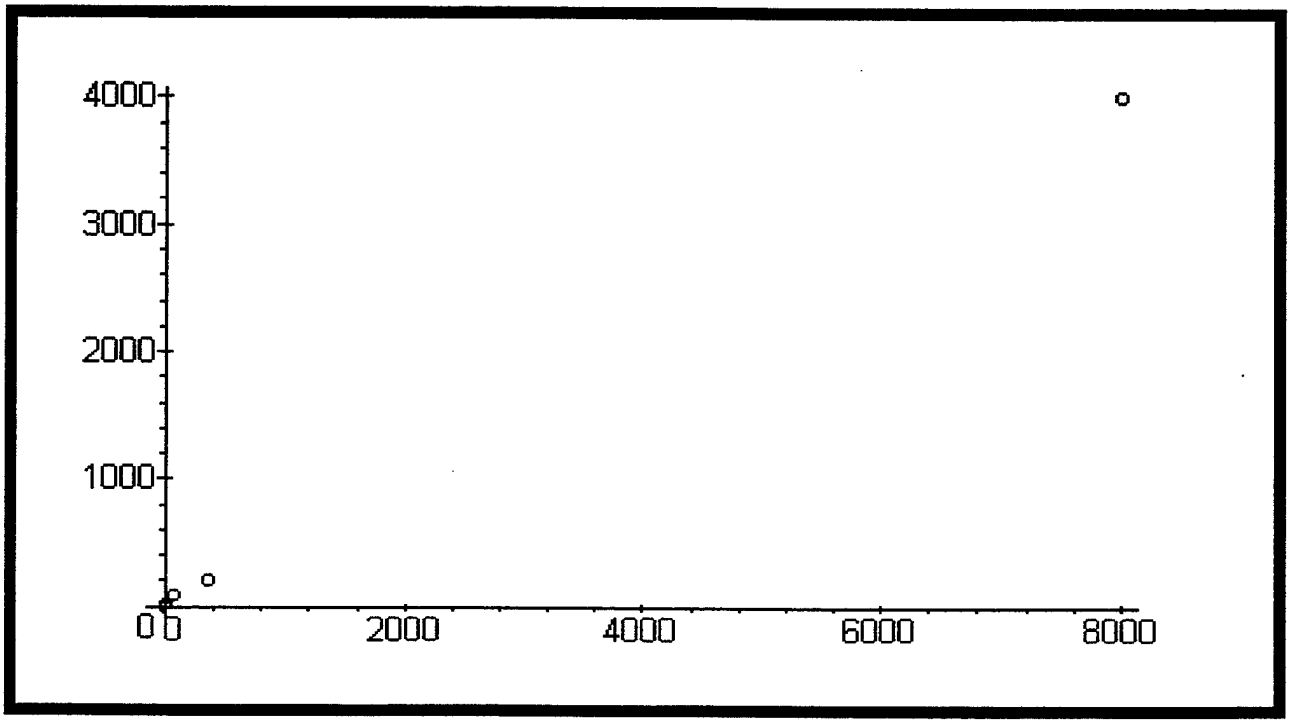

Figure 14. Length^3 vs. Volume.

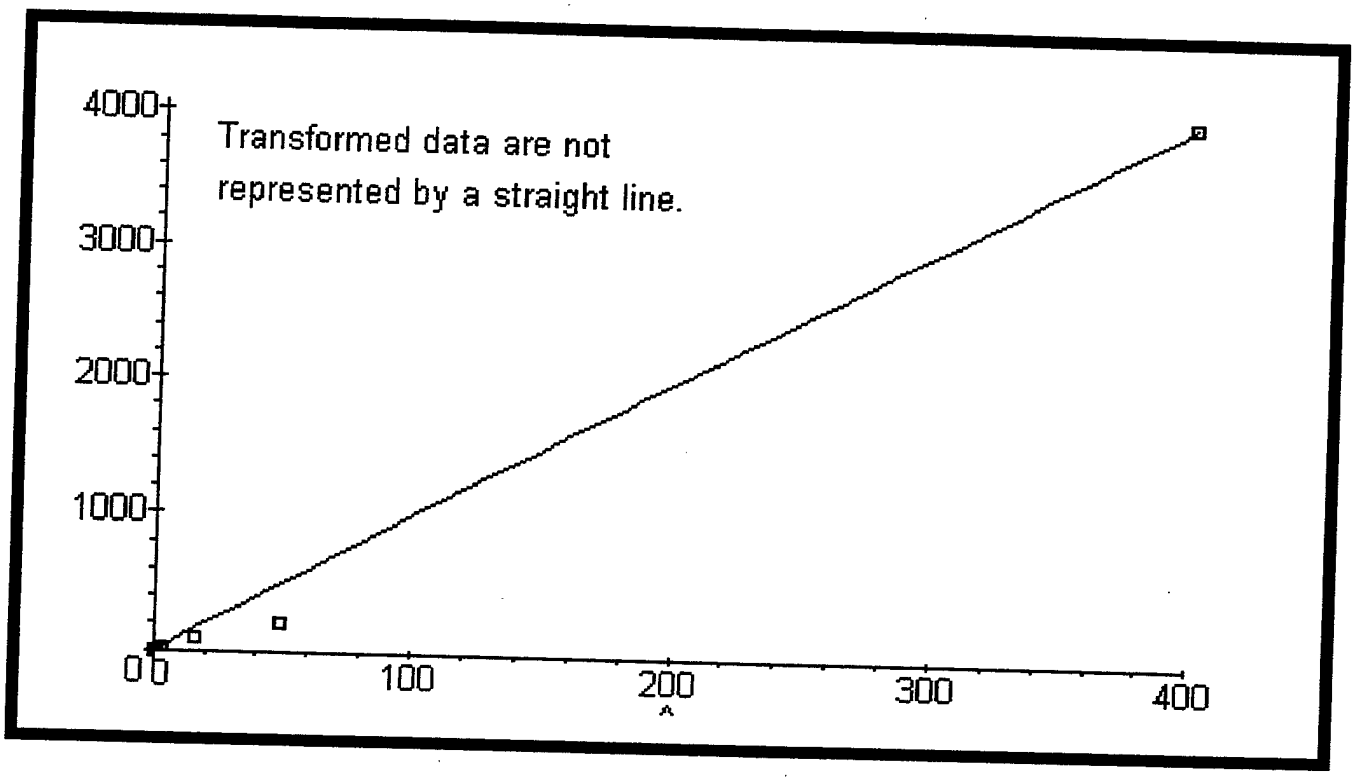

Figure 15. Length^2 vs. Volume.

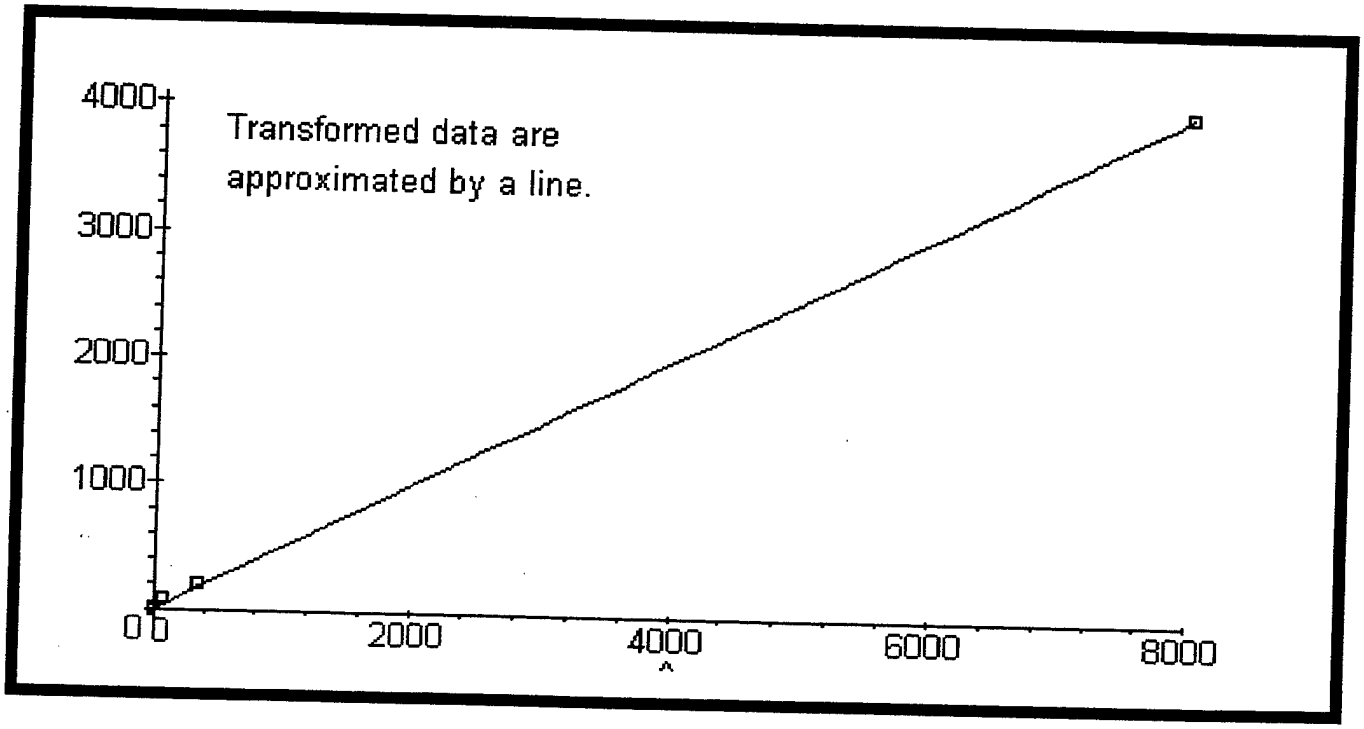

Figure 16. Length^3 vs. Volume.

Step 5 The proportionality constant. Use the line that was drawn in by hand onto the graphs of the data in Step 4; then select two points (x_1, y_1) and (x_2, y_2) on this line. The line segment between the two points approximates the hand drawn "best" line (see Figure 16). The proportionality constant is the slope of the line segment between the two chosen points, slope = $\frac{y_2 - y_1}{x_2 - x_1}$. Table 26 presents a solution for the slope using two points from the "best" line drawn onto Figure 16.

Command	Output
slope:=evalf((4000-100)/(8000-64));	slope:= .4914314516

Table 26. The Constant Of Proportionality.

Qualitatively, the proposed model is observed to be reasonable, as shown in Figure 16.

Using the procedures discussed in Chapter II displayed in Table 27, plotting the original data set overlaid by the continuous model provides a visual analysis of the model, located in Figure 17. As can be observed, the model represents quite accurately the trend of the original data.

Commands
with(plots): *plot1:=plot(slope*x^3,x=0..20):* *plot2:=plot(datalv, style=point, symbol=circle):* *t1 := textplot([8,3000,`The model is plotted as the solid line. `],align=ABOVE):* *t2:= textplot([8,2500,`The length/volume data points are the circles.`],align = ABOVE):* *display({plot1,plot2,t1,t2});*

Table 27. The Model Over The Original Data.

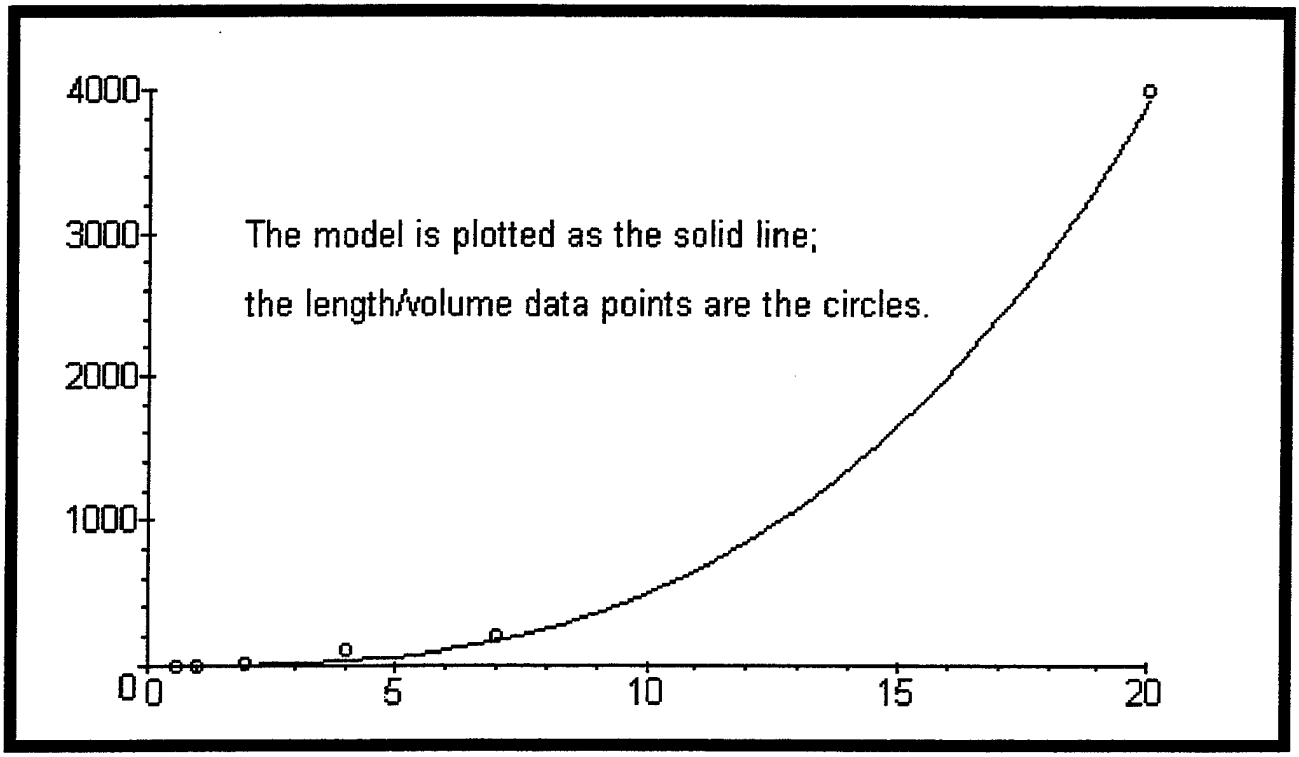

Figure 17. Bass Derby Proportionality Model.

2. Example 3.1.2: Rejecting A Poor Model

This sample demonstrates that the qualitative information obtained from a Maple plot can be sufficient to reject a poor model. In the following data set, p represents the population of fruit flies and t represents the time spent (in days) in incubation. The modeler suspects that the population of flies is proportional to the time spent in incubation: $p \propto t$. (Fox, et al., op.cit., page 42)

The data set is located in Table 28, while Table 29 presents the commands and outputs required for this analysis. Figure 18 is the resulting plot.

t	7	14	21	28	35	42
p	8	41	133	250	280	297

Table 28. The Time And Population Data Points.

Command	Output
t:=[7,14,21,28,35,42]: *p:=[8,41,133,250,280,297]:* *tp:=array(1..2,1..6,[t,p]);* *datatp:={seq([t[i],p[i]],i=1..6)}:* *plot(datatp,style=point,symbol=box);*	$tp := \begin{bmatrix} 7 & 14 & 21 & 28 & 35 & 42 \\ 8 & 41 & 133 & 250 & 280 & 297 \end{bmatrix}$

Table 29. The Plot Commands.

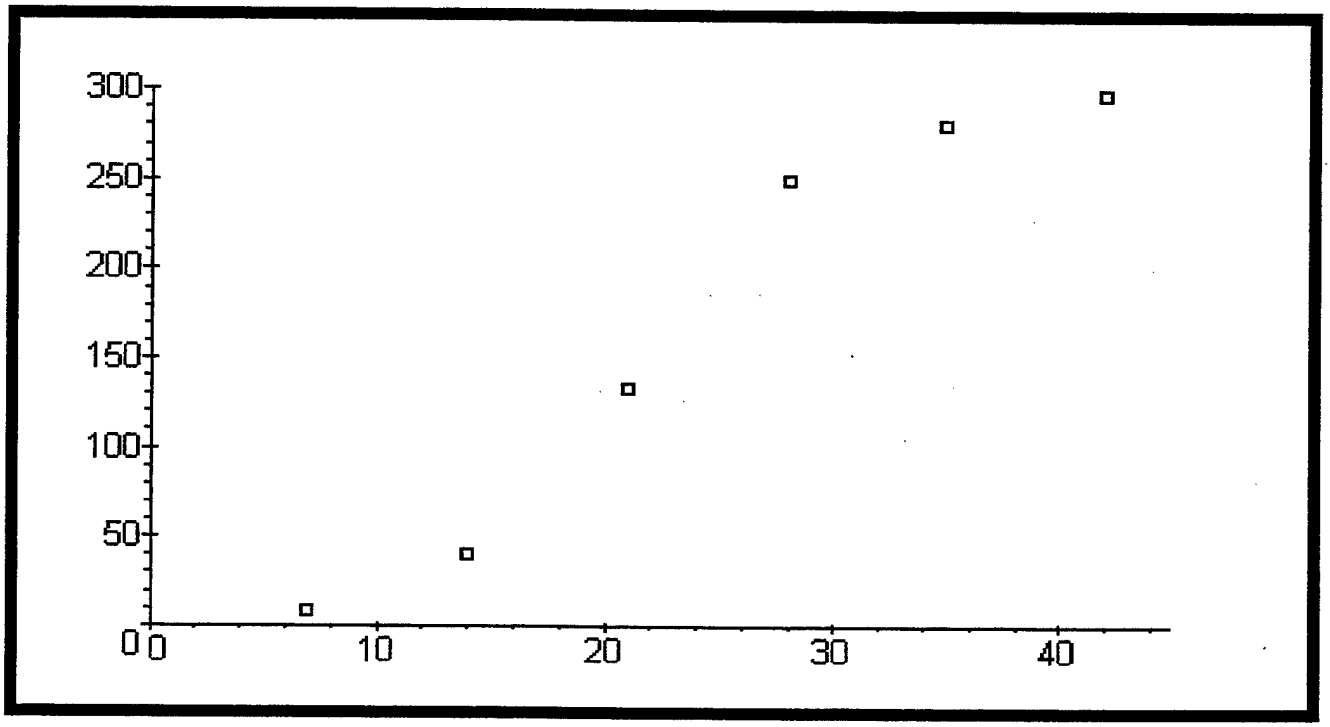

Figure 18. A Poor Proportionality Model.

It is clear from the graph that the data is not accurately approximated by a line through the origin. Therefore, the proportionality assumption is rejected. The modeling process must begin again to determine other possible relationships between the variables Chapter V will investigate and describe these other techniques.

3. Example 3.1.3: Checking If A Line Projects Through The Origin

After gathering the data found in Table 30, an analyst hypothesizes that a model $w \propto \ln z$ accurately describes the data. Using the commands found in Table 31 Maple is used to test the proportionality hypothesis. Figure 19 is the resulting plot (Fox, et al., op.cit., page 44).

z	8.1	22.1	60.1	165
w	1	2	3	4

Table 30. Example 3.1.3.

Command	Output
z:= [8.1,22.1,60.1,165]: *w:=[1,2,3,4]:* *zw:=array(1..2,1..4,[z,w]);*	$zw := \begin{bmatrix} 8.1 & 22.1 & 60.1 & 165 \\ 1 & 2 & 3 & 4 \end{bmatrix}$
lnz:=evalf(map(x->ln(x),z)); *data:={seq([lnz[i],w[i]],i=1..4)}:* *plot(data,style=point,symbol=circle);*	$lnz := [2.09, 3.10, 4.10, 5.11]$

Table 31. Example 3.1.3 Plot Commands.

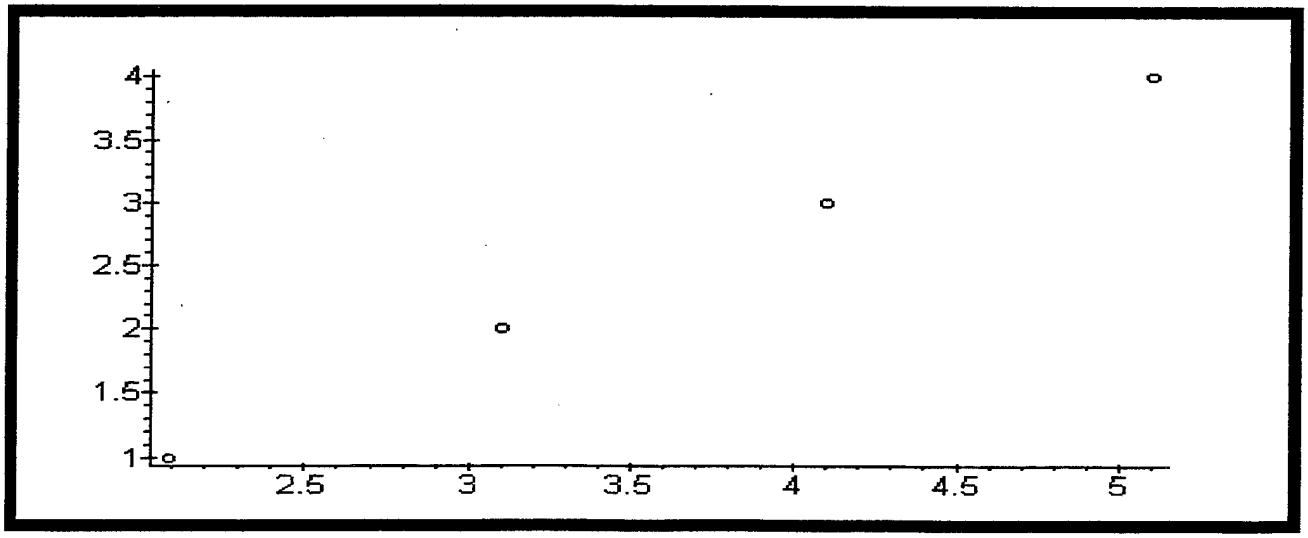

Figure 19. Example 3.1.3: The Raw Data Plotted.

From the graph it is observed that while the data does approximately lie along a straight line, it is difficult to determine if the proportionality model is accurate without the origin plotted on the graph. The command in Table 32 replots the data with a view to include the origin. In Figure 20, a line approximating the data has been drawn through the transformed data. It is obvious from Figure 20 that the transformed data does lie along a line, but it does not pass through the origin. Thus the proposed proportionality w $\propto$ ln z is rejected.

Command
plot(data,style=point,symbol=box,view=[-1..6,-2..5]);

Table 32. Investigating The Origin.

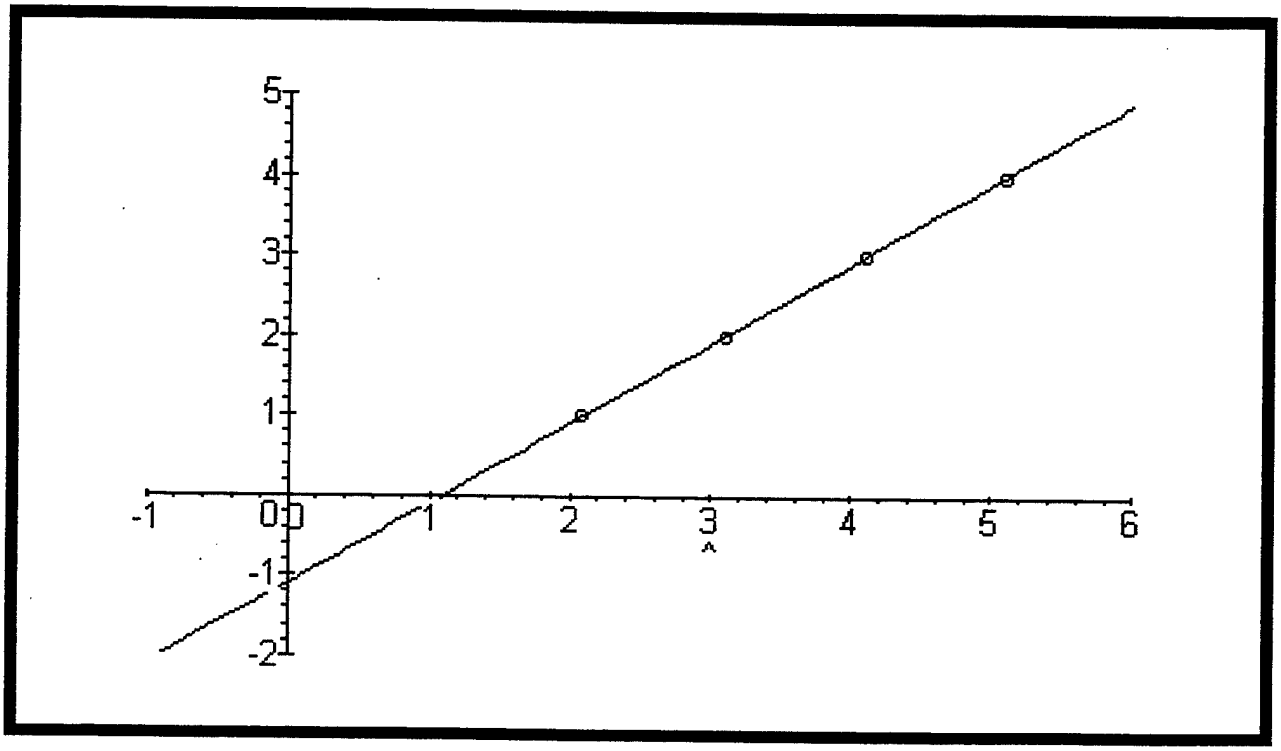

Figure 20. Example 3.1.3: Proportionality Rejected.

B. EXAMPLES OF PROPORTIONALITY SUBMODELS

1. Example 3.2.1: The Bass Fishing Derby

Consider a sport fishing club that for conservation purposes wishes to encourage its membership to release their fish immediately after catching them. On the other hand, the club wishes to make awards based on the total weight of fish that are caught. You might suggest that each individual carry a small portable scale. However, portable weight scales tend to be inconvenient and inaccurate, especially for smaller fish. Thus we define our problem as follows: Predict the weight of a fish in terms of some easily measurable dimensions. If we let W represent the weight of a fish in ounces, *len* represent the length of a fish in inches, and g its girth (circumference of the fish at its widest points) in inches, we can suggest the following models: (Fox, et al., op.cit., page 47)

- $W \propto len^3$
- $W \propto g^3$
- $W \propto g^2\, len$
- $W \propto g\, len^2$

Table 33 displays the data sets that have been collected and will be used to test each of the four models:

Length, *len*(in.)	14.5	12.5	17.25	14.5	12.625	17.75	14.125	12.625
Girth, *g*(in.)	9.75	8.375	11.0	9.75	8.5	12.5	9.0	8.5
Weight, *W*(oz.)	27	17	41	26	17	49	23	16

Table 33. The Bass Fishing Derby Data Points.

In this example, for Step 1, the bass data file used in Chapter I, Section D.2.b can be used, requiring only the entry of the girth data. The procedure discussed in Section A will be followed, illustrating a Maple solution analyzing the four suggested models. Once this procedure is completed, a new question arises; which model fits the data best? In Chapter

IV, a method of fitting models is demonstrated providing an analytical answer to the question, which model is best. Figures 21 through 33 and Table 34 present the commands and outputs for this analysis.

Step 1: Input the new data. *Digits:=4:* restricts the number of digits in any output to four, and eliminates unnecessary digits, (Maple's default is 10). Recall the data for length and weight has previously been inputted so only girth must be entered.

```
Digits:=4:
g:=[9.75,8.375,11,9.75,8.5,12.5,9,8.5]:
```

Step 2: Check for trends and identify potential outliers. Define the points to plot, from the bass array and execute the plot. In this case, a line through the origin is inserted to demonstrate that it does not graphically approximate the data.

```
lenwt:={seq([len[i],wt[i]],i=1..8)}:
plot(lenwt,style=point,symbol=box,view=[0..20,0..50]);
```

Figure 21. Example 3.2.1: The Bass Fishing Derby.

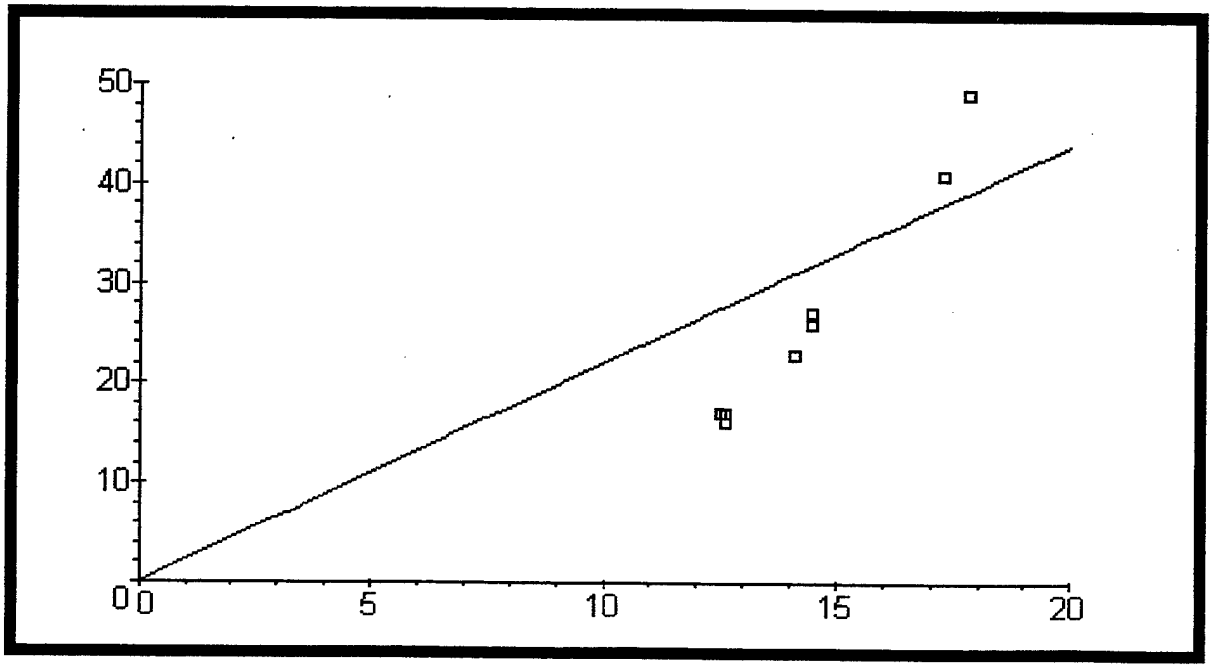

Figure 22. Length vs. Weight.

Step 3: Again, the data appears to be similar to some power x^2 or x^3, so a few hypothesized proportionalities could be tested; specifically, $V \propto len^3$, $V \propto g^3$, $V \propto g^2\,len$, and $V \propto g\,len^2$ as obtained by Giordano and Weir, op.cit., page 57. The following commands transform the data:

```
g2:=map(x->x^2,g):
g3:=map(x->x^3,g):
len2:=map(x->x^2,len):
len3:= map(x->x^3,len):
bass:=array(1..6,1..8,[len,len2,len3,g,g2,g3]);
```

bass:=

14.5	12.5	17.25	14.5	12.625	17.75	14.125	12.625	*length*
210.3	156.3	297.6	210.3	159.4	315.1	199.5	159.4	*length*^2
3049	1953	5133	3049	2012	5592	2818	2012	*length*^3
9.75	8.375	11	9.75	8.5	12.5	9	8.5	*girth*
95.06	70.14	121	95.06	72.25	156.3	81	72.25	*girth*^2
926.9	587.4	1331	9269	614.1	1953	729	614.1	*girth*^3

The commands for the two transformations $g^2\,len$ and $g\,len^2$ require the data from the above array.

```
lg2:= [seq(g2[i]*len[i],i=1..8)]:
gl2:= [seq(g[i]*len2[i],i=1..8)]:
bass:=array(1..5,1..8,[len,g,lg2,gl2,wt]);
```

bass:=

14.5	12.5	17.25	14.5	12.625	17.75	14.125	12.625	*length*
9.75	8.375	11	9.75	8.5	12.5	9	8.5	*girth*
1378	876.8	2087	1378	912.5	2774	1145	912.5	lg^2
2050	1309	3274	2050	1355	3939	1796	1355	*gl*^2
27	17	41	26	17	49	23	16	*weight*

Figure 23. Example 3.2.1: The Bass Fishing Derby.

Step 4: Plot the transformed data to test hypothesized transformations. The four plots will compare weight with len^3, g^3, g^2 len and g len^2, independently. The first step will define the points to plot, from the bass array, followed by the plot commands. On each plot an "eye-balled best" line has been drawn through the origin, to evaluate the proportionality of the transformations. This "best" line is a visual estimation of the line which passes through the data. In the next chapter you will learn how to obtain a more exact line.

```
len3wt:={seq([len3[i],wt[i]],i=1..8)}:
plot(len3wt,style=point,symbol=cross,view=[0..6000,0..50]);

g3wt:={seq([g3[i],wt[i]],i=1..8)}:
plot(g3wt,style=point,symbol=diamond,view=[0..2000,0..50]);

lg2wt:={seq([lg2[i],wt[i]],i=1..8)}:
plot(lg2wt,style=point,symbol=circle,view=[0..3000,0..50]);

gl2wt:={seq([gl2[i],wt[i]],i=1..8)}:
plot(gl2wt,style=point,symbol=diamond,view=[0..4000,0..50]);
```

Figure 24. Example 3.2.1: The Bass Fishing Derby.

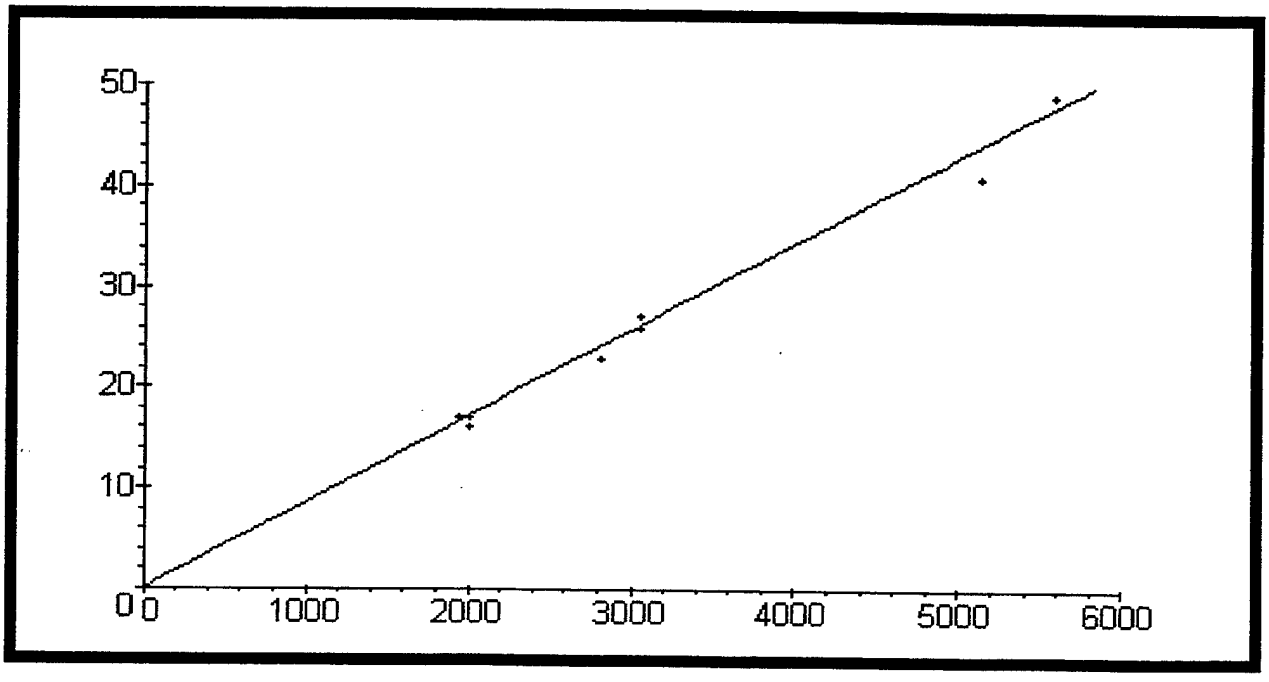

Figure 25. Weight vs. Length^3.

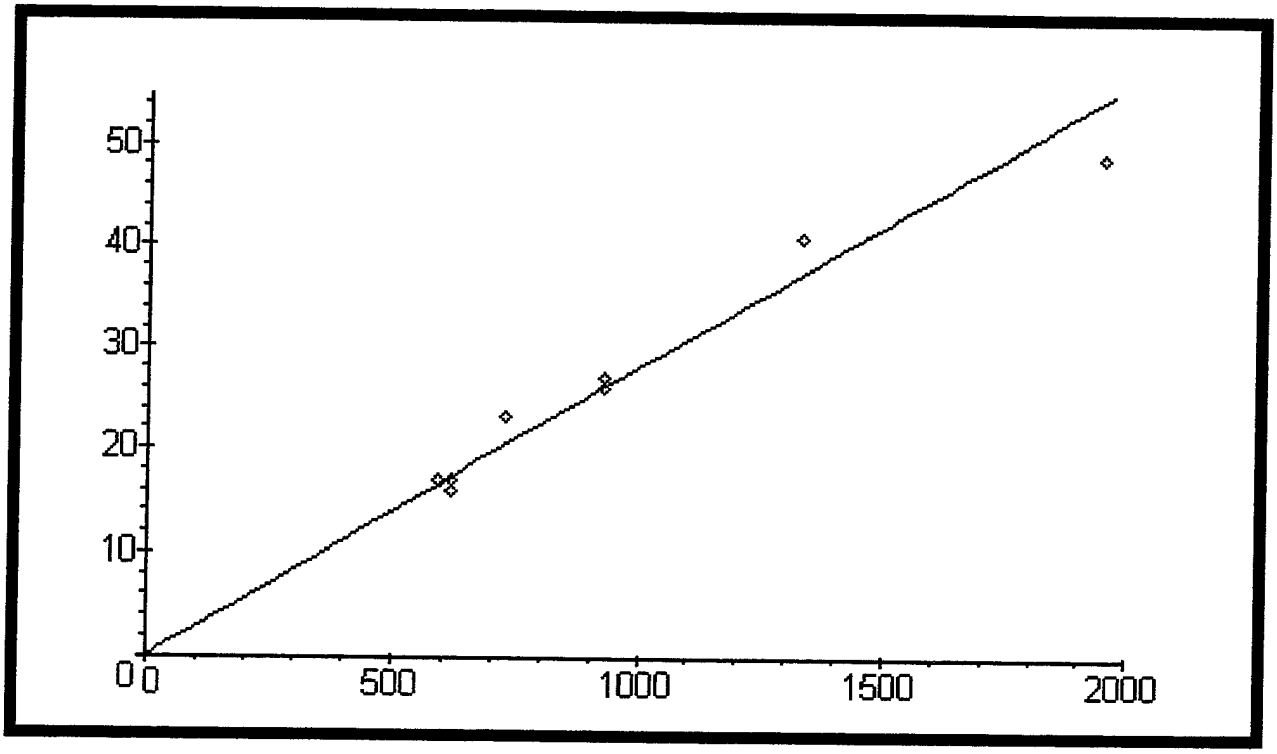

Figure 26. Weight vs. Girth^3.

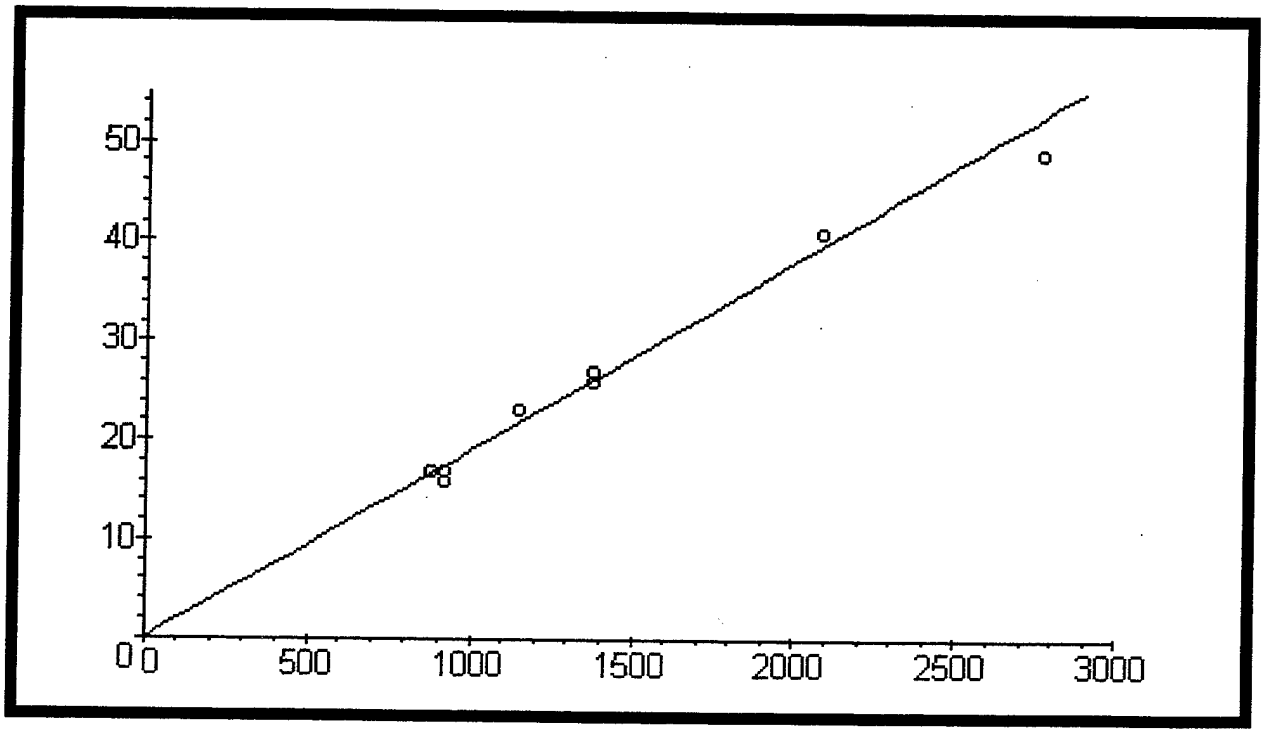

Figure 27. Weight vs. Length*Girth^2.

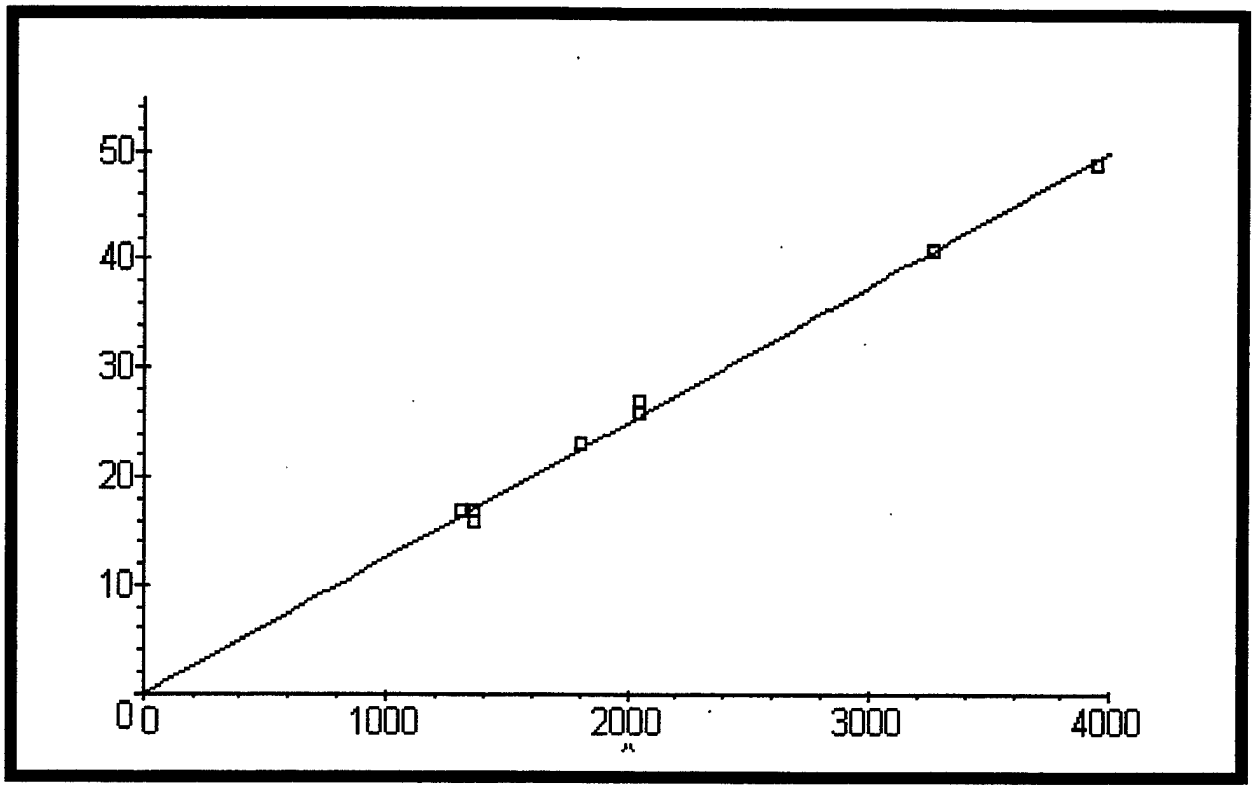

Figure 28. Weight vs. Girth*Length^2.

Step 5: The proportionality constant. Note: the estimates of proportionality will vary, depending on the values chosen to determine the slope. Using the "best" line, pick two points (x_1, y_1) and (x_2, y_2) on this "best" line. The line segment between the two points approximates the hand drawn "best" line. Table 34 presents the commands and Figures 30 through 33 display the output, identical to Figures 25 through 28 with the "best" lines inserted. In each of the four plots, the "best" lines do approximate the data.

Figure 29. Example 3.2.1: The Bass Fishing Derby.

Command	Output
Digits:=3:	
slope1:=(43-8.6)/(5000-1000): evalf(slope1);	0.00860
slope2:=(42-14)/(1500-500): evalf(slope2);	0.0280
slope3:=(45.5-8.5)/(2500-500): evalf(slope3);	0.0185
slope4:=(50-12.5)/(4000-1000): evalf(slope4);	0.0125
plot1:=plot(len3wt,style=point,symbol=cross):	
*plot2:=plot(slope1*x,x=0..6000):*	
display({plot1,plot2});	
plot3:=plot(g3wt,style=point,symbol=diamond):	
*plot4:=plot(slope2*x,x=0..2000):*	
display({plot3,plot4});	
plot5:=plot(lg2wt,style=point,symbol=circle):	
*plot6:=plot(slope3*x,x=0..3000):*	
display({plot5,plot6});	
plot7:=plot(gl2wt,style=point,symbol=box):	
*plot8:=plot(slope4*x,x=0..4000):*	
display({plot7,plot8});	

Table 34. Example 3.2.1: The Bass Fishing Derby.

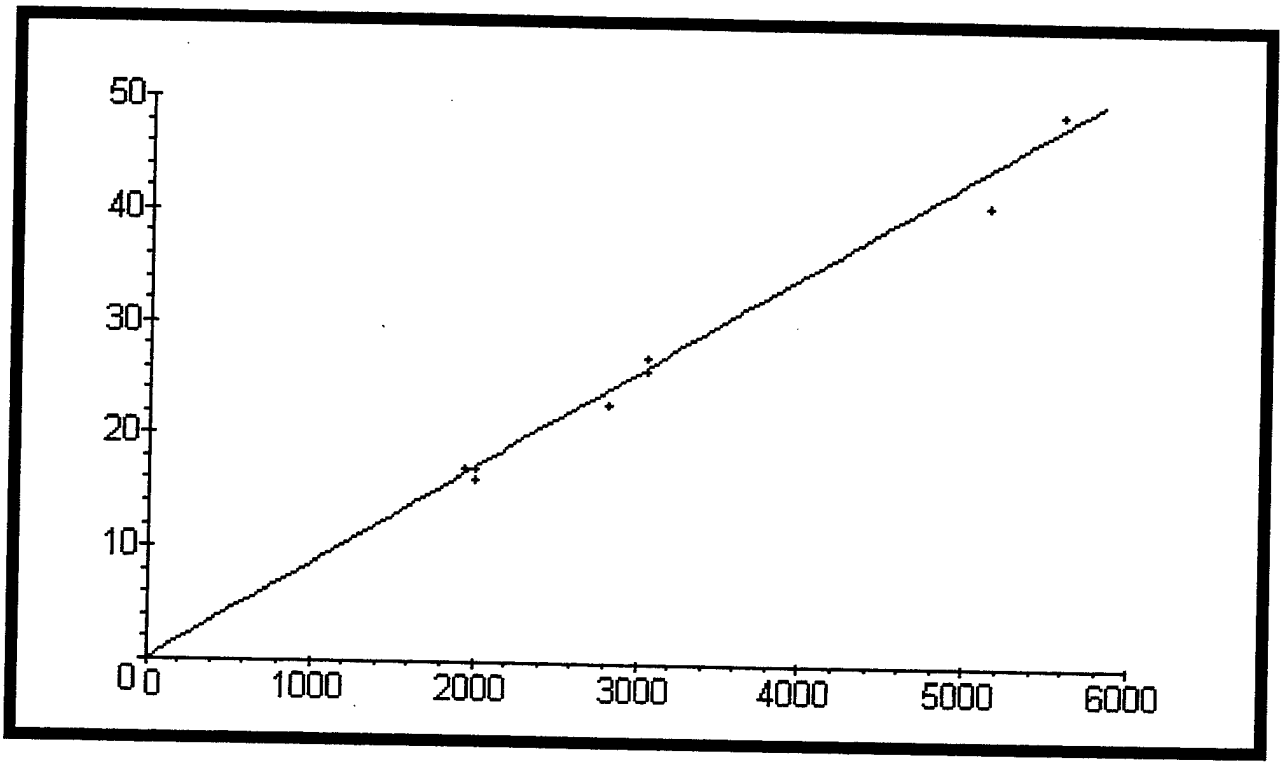

Figure 30. Weight vs. Length^3.

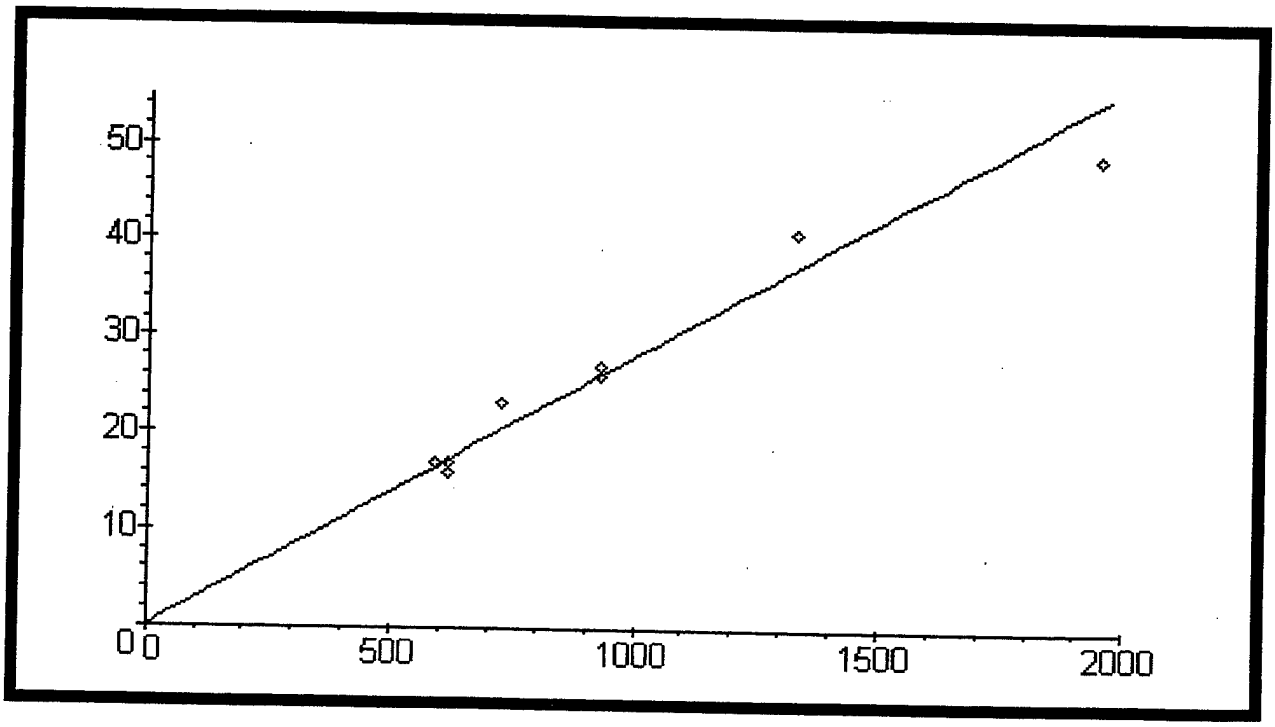

Figure 31. Weight vs. Girth^3.

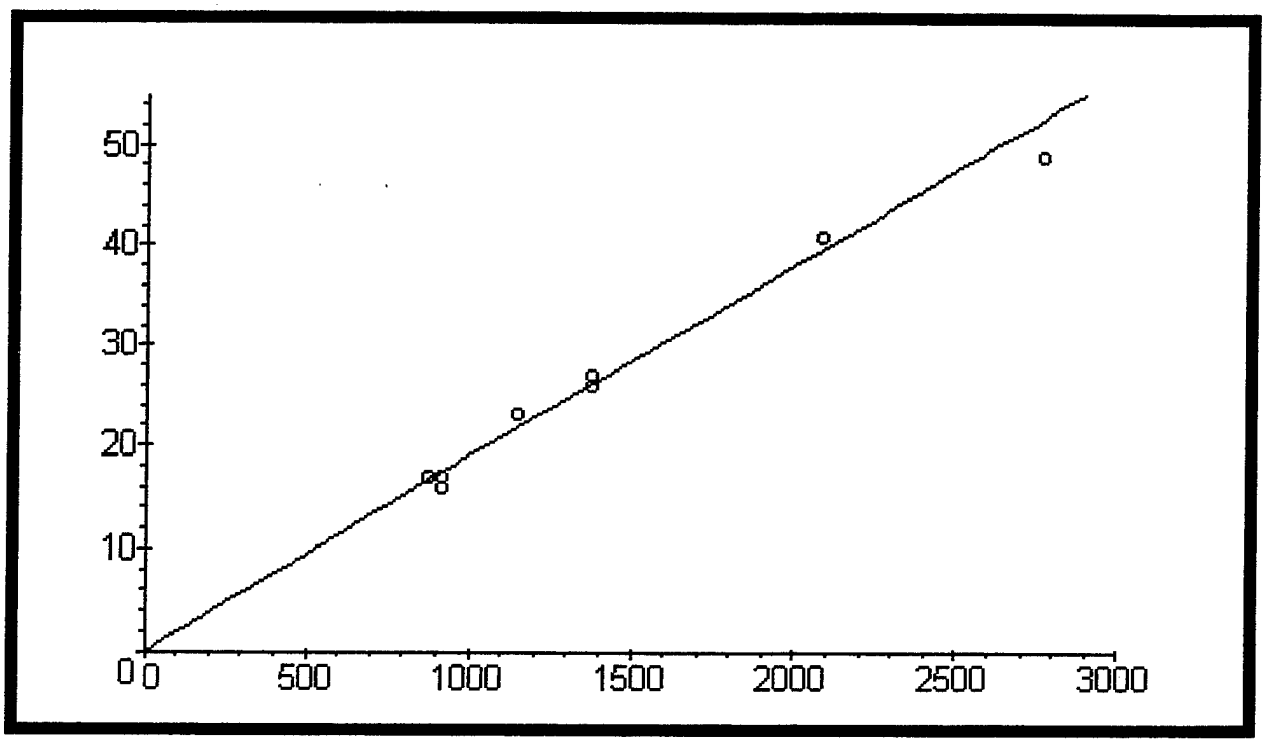

Figure 32. Weight vs. Length*Girth^2.

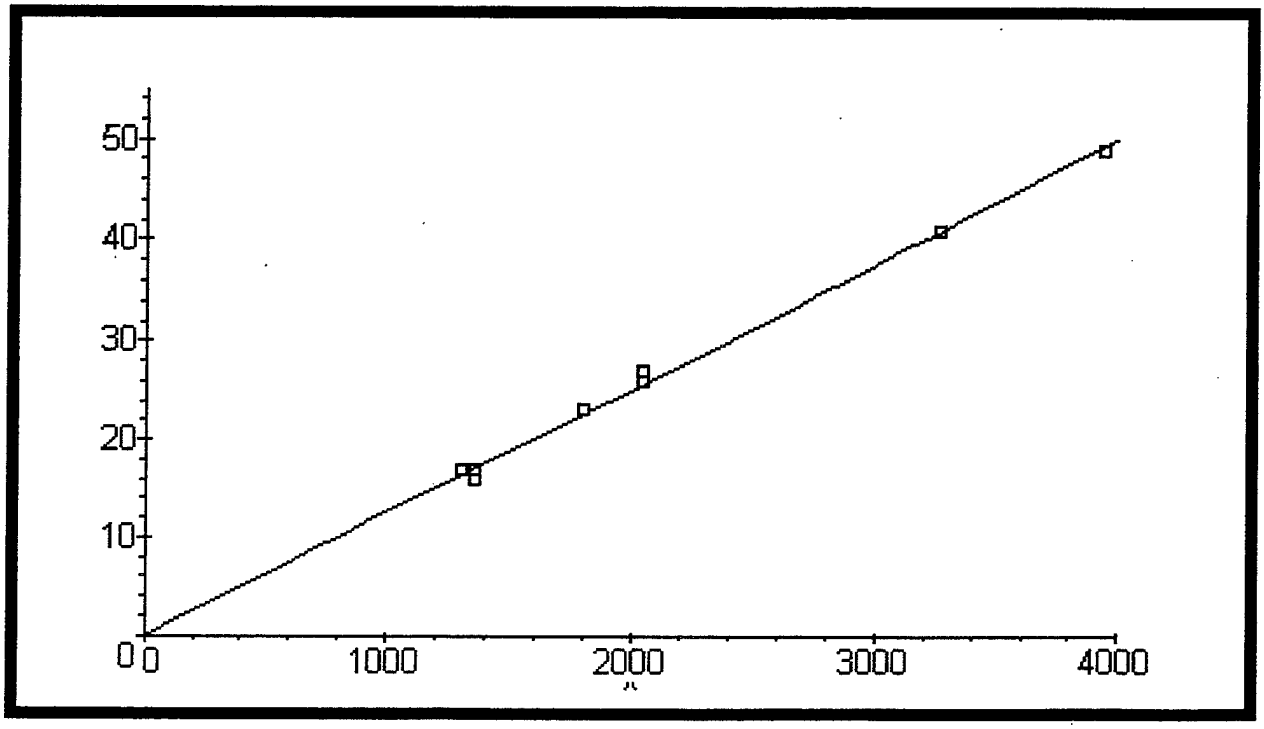

Figure 33. Weight vs. Girth*Length^2.

2. Example 3.2.2: Vehicular Braking Distance

A popular 'Rule of Thumb' often given to students in driver education classes is the 'Two Second Rule' to prescribe a safe following distance. The rule states that if you stay two seconds behind the car in front, you have the correct distance no matter what your speed. Since the amount of time is constant (two seconds), the rule suggests a proportionality between stopping distance and speed, v. To test this rule we pose the following problem: Predict a vehicle's total stopping distance as a function of its speed. In the development of this model (see Giordano and Weir, op.cit., page 64), total stopping distance is calculated as the sum of the reaction distance, d_r, and the braking distance, d_b. The following submodels are hypothesized in that development: (Fox, et al., op.cit., page 52)

- $d_r \propto v$
- $d_b \propto v^2$

These submodels yield that the total stopping distance, d, is represented by an equation of the form $d = k_1 v + k_2 v^2$. These submodels are tested using the data set in Table 35.

v	20	25	30	35	40	45	50	55	60	65	70	75	80
d_r	22	28	33	39	44	50	55	61	66	72	77	83	88
d_b	20	28	40.5	52.5	72	92.5	118	148.5	182	220.5	266	318	376
d	42	56	73.5	91.5	116	142.5	173	209.5	248	292.5	343	401	464

Table 35. The Vehicular Braking Distance Data Points.

Each of the two submodels will be tested as demonstrated in Example 3.2.1. The data must be entered and the hypothesized transformations applied. After plotting the transformed data, calculating the slopes of the "best" lines on each graph will determine the constants of proportionality. These steps are presented in Figures 34 through 40.

Step 1: Input the new data.

```
v:=[20,25,30,35,40,45,50,55,60,65,70,75,80]:
dr:=[22,28,33,39,44,50,55,61,66,72,77,83,88]:
db:=[20,28,40.5,52.5,72,92.5,118,148.5,182,220.5,266,318,376]:
dist:=[42,56,73.5,91.5,116,142.5,173,209.5,248,292.5,343,401,464]:
v2:=map(x->x^2,v):
braking:=array(1..5,1..13,[v,dr,db,dist,v2]);
```

braking:=

20	25	30	35	40	45	50	55	60	65	70	75	80
22	28	33	39	44	50	55	61	66	72	77	83	88
20	28	40.5	52.5	72	92.5	118	148.5	182	220.5	266	318	376
42	56	73.5	91.5	116	142.5	173	209.5	248	292.5	343	401	464
400	625	900	1225	1600	2025	2500	3025	3600	4225	4900	5625	6400

Step 2: Check for trends and identify potential outliers. Define the points to plot from the braking array and plot these points. Three data sets are plotted on one graph, demonstrating no outliers and a non-linear relationship between v and d and between v and d_b.

```
vd:={seq([v[i],dist[i]],i=1..13)}:
vdr:= {seq([v[i],dr[i]],i=1..13)}:
vdb:= {seq([v[i],db[i]],i=1..13)}:
plotvd:=plot(vd,style=point,symbol=diamond):
plotvdr:=plot(vdr,style=point,symbol=circle):
plotvdb:=plot(vdb,style=point,symbol=cross):
t1:=textplot([5,400,`v vs d plotted with diamonds. `], align =RIGHT):
t2:=textplot([5,360,`v vs dr plotted with circles. `], align =RIGHT):
t3:=textplot([5,320,`v vs db plotted with crosses. `], align =RIGHT):
with(plots):
display({plotvd,plotvdr,plotvdb,t1,t2,t3});
```

Figure 34. Vehicular Braking Distance.

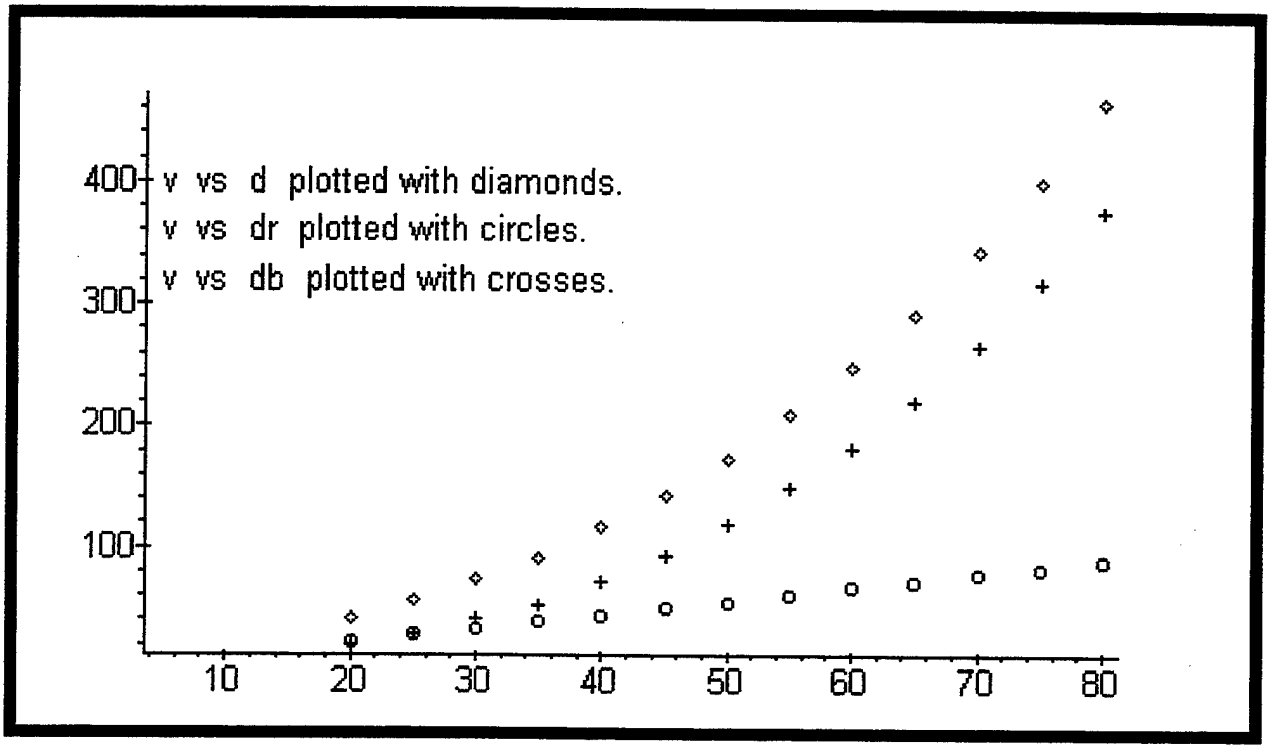

Figure 35. Trends And Outliers Plot.

Step 3: From Figure 35, the reaction distance does appear to be proportional to velocity, and braking distance could be proportional to velocity squared, which supports the derived submodels. The command for this step is found in Step 1, v^2 is in the last line of the braking array.

Step 4: Plot the transformed data to evaluate the submodels. The reaction and braking distances are plotted in Figures 37 and 38. Again, a "best" line has been added to each graph to evaluate the proportionality of the transformations.

```
plot(vdr,style=point,symbol=circle);

datav2db:={seq([v2[i],db[i]],i=1..13)}:
plot(datav2db,style=point, symbol=box,view=[0..6500,0..400]);
```

Figure 36. Vehicular Braking Distance.

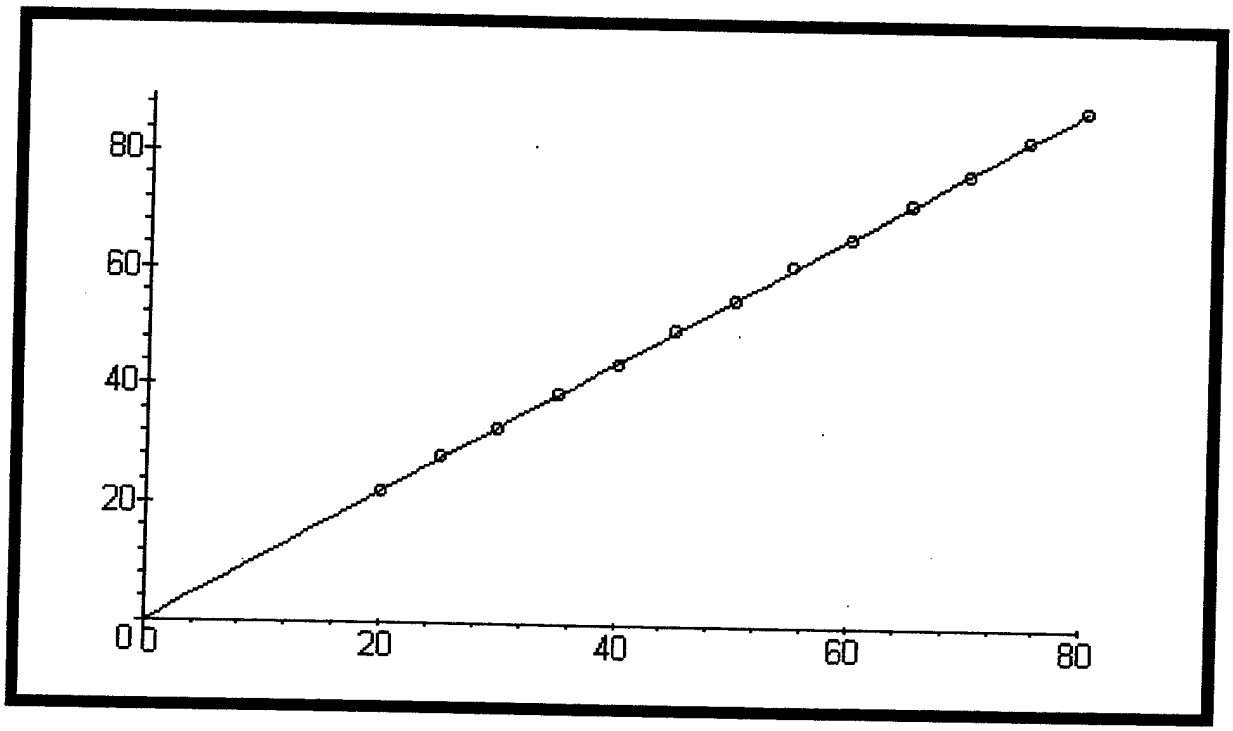

Figure 37. Velocity vs. Reaction Distance.

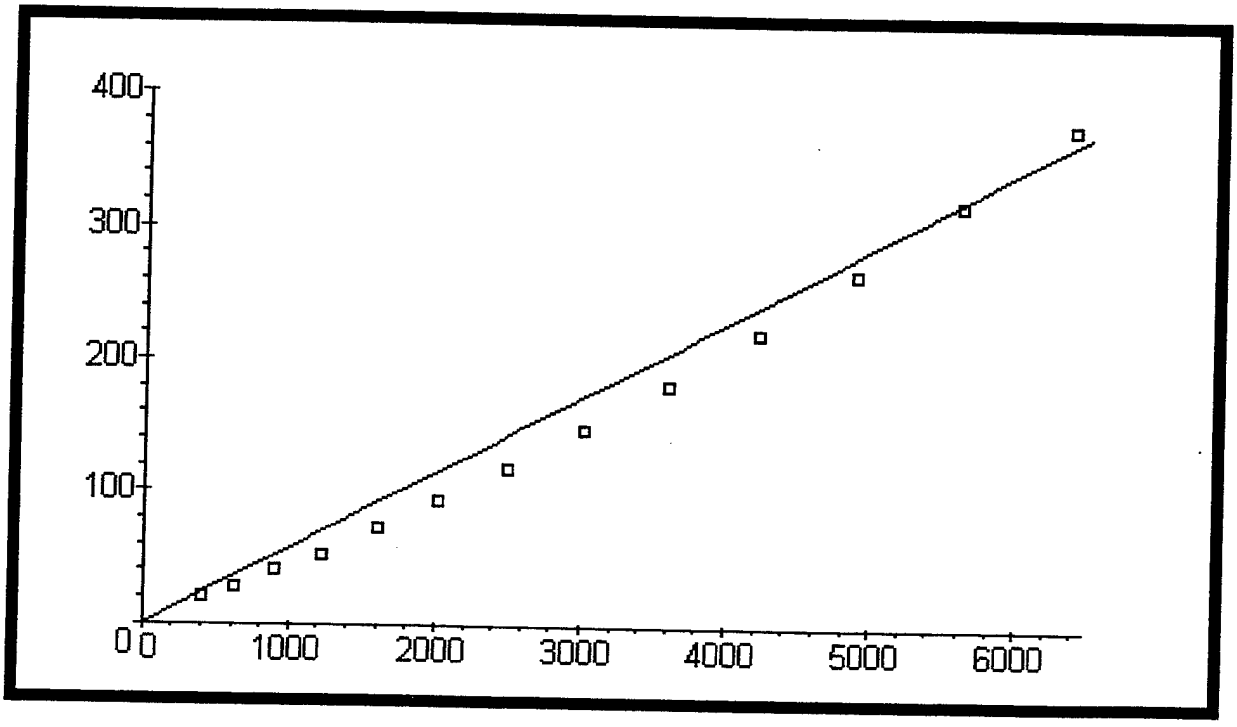

Figure 38. Velocity^2 vs. Braking Distance.

Step 5: Estimate the constants of proportionality. The procedure for selecting the points is identical to the bass fishing derby, select values on the "best" line to calculate the slopes. To evaluate the constants of proportionality, analyze the plots in Figures 37 and 38.

```
slope1:=(88-22)/(80-20);                                    = 1.1
slope2:=(342-57)/(6000-1000);                               = 0.057
plotcontx:=plot(slope1*x+slope2*x^2,x=0..80):
t1:=textplot([5,400,`The data is plotted with diamonds. `], align =RIGHT):
t2:=textplot([5,360,`The model is the solid line.`],align =RIGHT):
display({plotvd,plotcontx,t1,t2});
```

Figure 39. Vehicular Braking Distance.

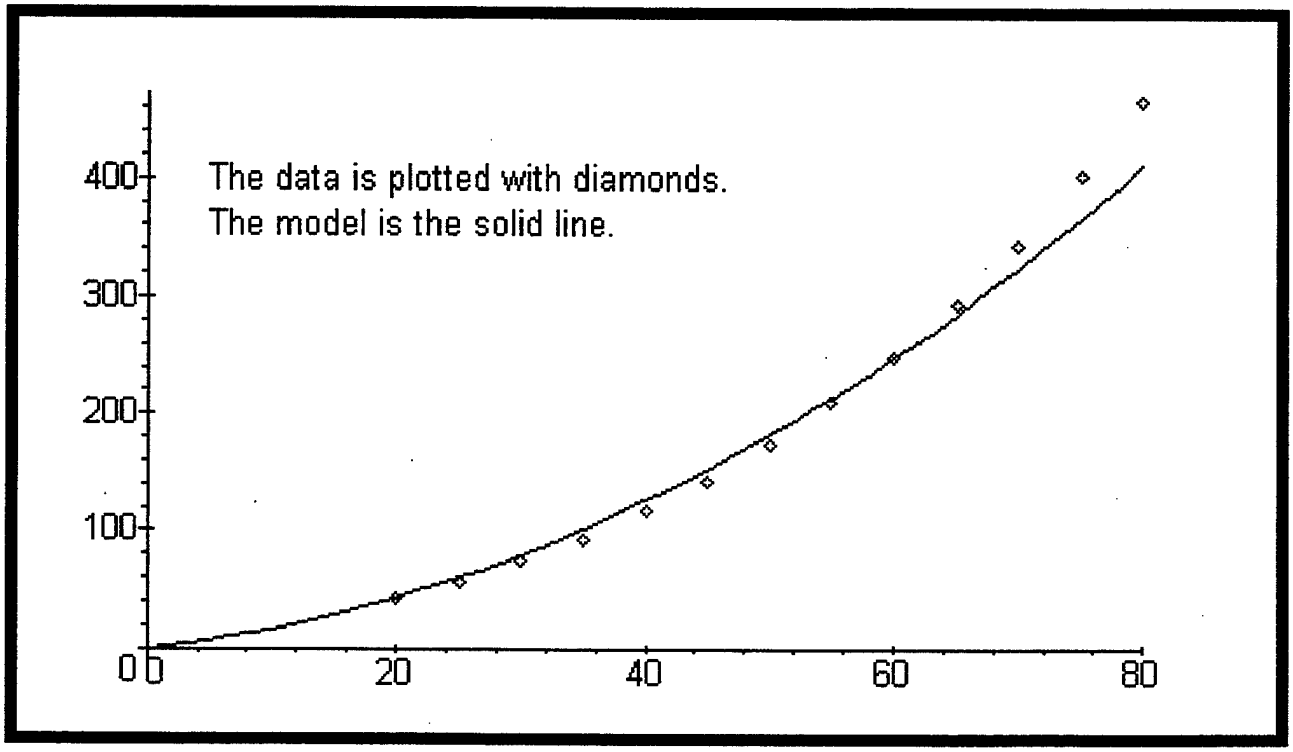

Figure 40. Transformation With Proportionality.

Using the procedure from Section A, and verifying the result graphically in Figure 40, the following model is determined and assessed to be an accurate approximation of the vehicular stopping distance data: $d = 1.1v + 0.057 v^2$.

IV. MODEL FITTING

Chapter III demonstrated Maple's capability to perform various transformations on a data set and to plot the resulting transformed data to assist in determining a proposed proportionality model graphically. In particular, methods to enter data, transform data, obtain a scatterplot, test a proportionality relationship, and estimate the parameters of a model were discussed. This chapter describes how to determine the parameters of a model analytically, according to some criterion of "best fit," and to test the adequacy of the model.

Suppose it is proposed that a parabolic model might best explain a behavior being studied, and you are interested in selecting that member of the family $y = Ax^2$ which best fits the given set of data. Using Maple, y versus x^2 can be plotted and a graphical estimate the slope of the "best" line can be determined, as demonstrated in Chapter III.

This chapter will focus on an analytical method to arrive at an accurate model for a given data set. Again from the family $y = Ax^2$, A can be determined analytically by using a curve-fitting criterion, such as least-squares or Chebyshev, and solving the resulting optimization problem (see Chapter 4, Giordano and Weir, op. cit., for a discussion of model fitting). This chapter presents Maple commands which solve the least-squares optimization problem with analysis of the "goodness of the fit" of the resulting model. (Fox, et al., op.cit., page 66).

A. LEAST-SQUARES CURVE FITTING

The method of least-squares curve fitting is simply the solution to a model such that the sum of the squares of the deviations between the observations and predictions is minimized. In Maple, the *fit* command in the statistical package fits a model curve to a set of data points using the least-squares criterion. Given the set of data points $\{(x_j , y_j): j = 1,2,..., m\}$, the least-squares method will minimize

$$S = \sum_{j=1}^{m} \left[y_j - f\left(x_j\right) \right]^2 \qquad \text{Equation 1.}$$

For example, to fit the model $y = Ax^2$ to a set of data, the least-squares criterion requires the minimization of Equation 2. Note in Equation 2, A is estimated by a.

$$S = \sum_{j=1}^{5} \left[y_j - a x_j^2 \right]^2 \qquad \text{Equation 2.}$$

Minimizing Equation 2 is achieved using the first derivative.

$\frac{ds}{da} = -2 \sum x_j^2 (y_j - a x_j^2) = 0$. Solving for a: $a = (\sum x_j^2 y_j) / (\sum x_j^4)$.

Given the data set in Table 36, a numerical solution can be determined.

x	0.5	1.0	1.5	2.0	2.5
y	0.7	3.4	7.2	12.4	20.1

Table 36. Least-Squares Data Points.

Solving for a: $a = (\sum x_j^2 y_j) / (\sum x_j^4) = (195.0) / (61.1875) = 3.1869$

and the model $y = Ax^2$ becomes $y = 3.1869 x^2$ (Fox, et al., op.cit., pages 66 and 67).

In Maple the *fit* command fits the model type $Y = b + b_1X_1 + b_2X_2 + ... + b_kX_k$ to the data set given in the k specified columns. To fit the quadratic model $y = A_0 + A_1x + A_2x^2$ to a data set of x values, *xv*, and of y values, *yv*, (Fox, et al., op.cit., page 68), the Maple fit command required to solve this example is

*quadraticfit:=fit[leastsquare[[x,y], y = a*x^2 + b*x + c, {a,b,c}]] ([xv,yv]);*

where x and y are command calls which will use the two data sets named in the command (in this case *xv* and *yv*), while *a*, *b* and *c* are the coefficient variables for which a least-squares solution will be fit. Since *fit* is part of the statistics package, the *with(stats):* command must be entered once prior to using the *fit* command. Table 37 and Figure 41 illustrate the procedure.

Command	Output
with(stats):with(plots):Digits:=5:	
xv:=[.5,1,1.5,2,2.5];	xv := [.5, 1, 1.5, 2, 2.5]
yv:=[.7,3.4,7.2,12.4,20.1];	yv := [.7, 3.4, 7.2, 12.4, 20.1]
*xyfit:=fit[leastsquare[[x,y],y=a*x^2+b*x+ c,{a,b,c}]]([xv,yv]);*	xyfit := y = 3.2481 x^2 -.1839 x + .1036
f:=unapply(rhs(xyfit),x);	f:= x→ 3.2481 x^2 - .1839 x +.1036
xy:={seq([xv[i],yv[i]],i=1..5)}:	
plot1:=plot(xy,style=point,symbol=diamond):	
plot2:=plot(f(x),x=0..2.5):	
display({plot1,plot2});	

Table 37. The Least-Squares Command.

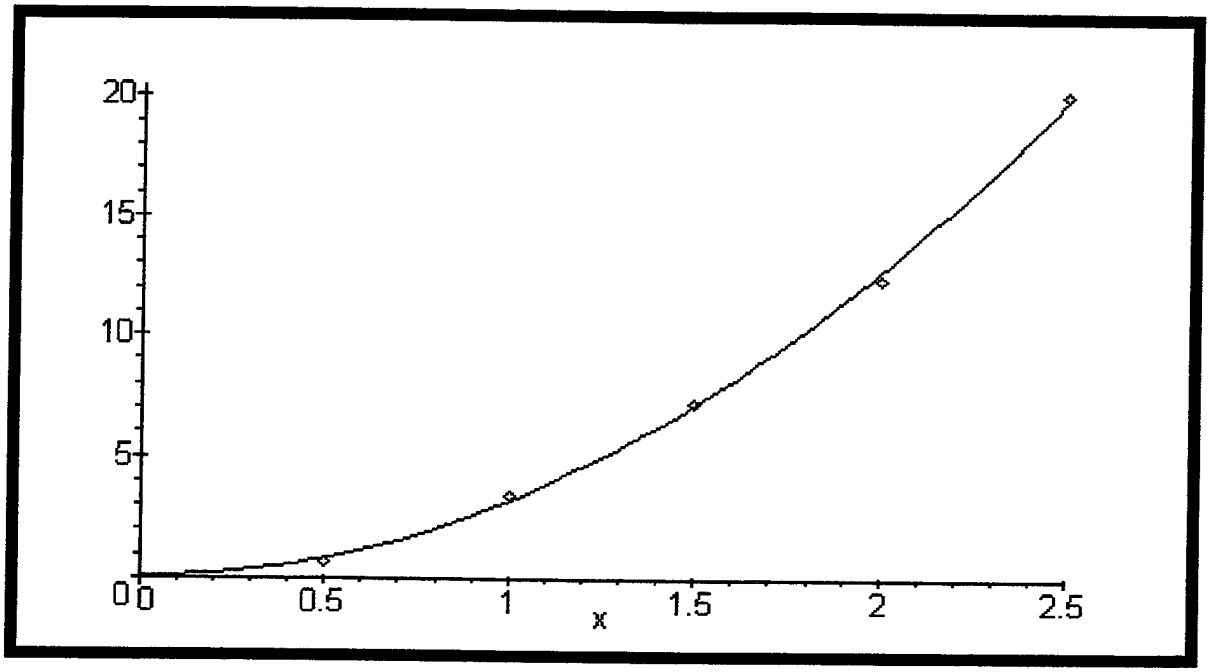

Figure 41. Least-Squares Fit Plotted With The Original Data.

Table 38 and Figure 42 illustrate the *fit* command applied to the model $y = Ax^2$ for the data set in Table 36. As obtained previously, the least-squares model is $y = 3.1869\ x^2$. Since the two data sets used in this example, *xv* and *yv*, have been entered previously, they can be called without re-entering the data. Having previously invoked the *with(plots):* and *with(stats):* commands, they need not be repeated in this continuation example either.

Command	Output
*xyfit:=fit[leastsquare[[x,y],y=a*x^2,{a}]]([xv,yv]);* *f:=unapply(rhs(xyfit),x);* *plot1:=plot(xy,style=point,symbol=diamond):* *plot3:=plot(f(x),x=0..2.5):* *display({plot1,plot3});*	xyfit:= $y = 3.1869\ x^2$ f:= $x \rightarrow 3.1869\ x^2$

Table 38. Least-Squares $y = Ax^2$.

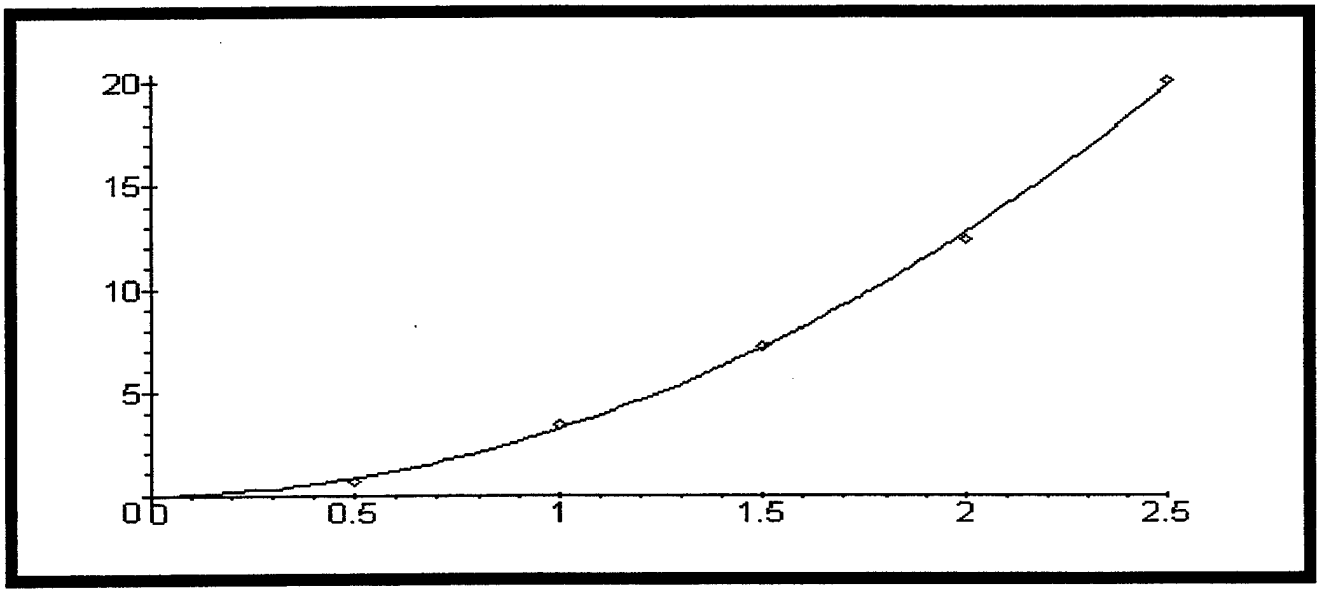

Figure 42. The Least-Squares Ax^2 Fit Plotted With The Data.

1. Example 4.1: A Least-Squares Fit

Table 39 and Figure 43 demonstrate how to determine a constant of proportionality, using the data in Example 3.2.1 of Chapter III. In the Example 3.2.1, the model, $W = .00860\ len^3$, was derived. The next example illustrates the *fit* command analytically fitting the model $W = k\ len^3$ to the same data set.

Command	Output
with(plots): with(stats):Digits:=5:	
len:=[14.5,12.5,17.3,14.5,12.6,17.8,14.1,12.6];	len := [14.5, 12.5, 17.3, 14.5, 12.6, 17.8, 14.1, 12.6]
wt:=[27,17,41,26,17,49,23,16];	wt := [27, 17, 41, 26, 17, 49, 23, 16]
lenwt:={seq([len[i],wt[i]],i=1..8)}:	
*lenwtfit:= fit[leastsquare [[x,y] , y = a*x^3, {a}]] ([len,wt]);*	lenwtfit := y = 0.0084365 x^3
f:=unapply(rhs(lenwtfit),x);	f:= x -> 0.0084365 x^3
plot4:=plot(lenwt,style=point,symbol=diamond):	
plot5:=plot(f(x),x=0..18):	
display({plot4,plot5});	

Table 39. Least-Squares $W = k\ len^3$.

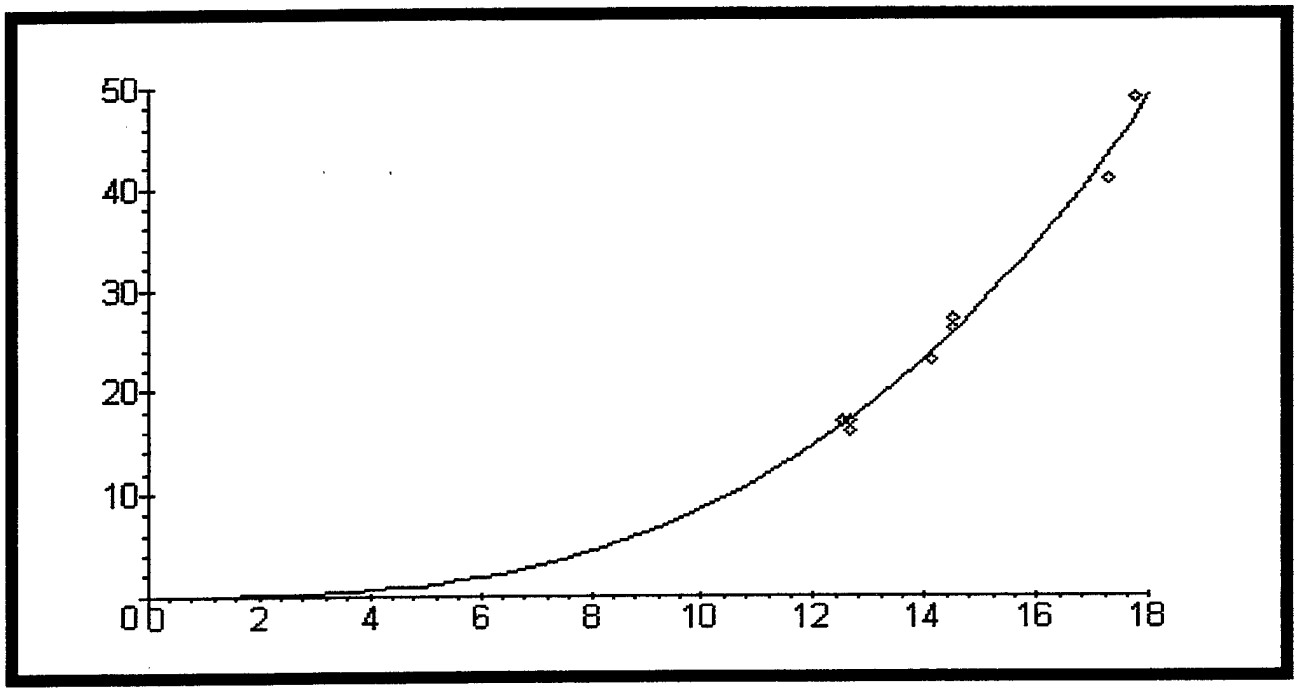

Figure 43. Weight = 0.0084365 *length^3.

The least-squares estimate of the proportionality constant in this model is $k = 0.0084365$. The graphical solution in Figure 43 shows that the model does capture the trend of the data, which corresponds to the solution in Chapter III.

B. PLOTTING THE RESIDUALS FOR A LEAST-SQUARES FIT

> In the previous section you learned how to obtain a least-squares fit of a model, and you plotted the model's predictions on the same graph as the observed data points in order to get a visual indication of how well the model matches the trend of the data. A powerful technique for quickly determining where the model breaks down is to plot the actual deviations or residuals between the observed and predicted values as a function of the independent variable. The deviations should be randomly distributed and contained in a reasonably small band that is commensurate with the accuracy required by the model. Any excessively large residual warrants further investigation of the data point in question to discover the cause of the large deviation. A pattern or trend in the residuals indicates that a predictable effect remains to be modeled, and the nature of the pattern gives clues on how to refine the model, if a refinement is called for. (Fox, et al., op.cit., page 76).

This section explains how to use Maple to compute residuals. After fitting a specified model, the difference between the observed and predicted values can be calculated by using the array manipulations described in Chapter I. In Example 4.1, the relationship between the weight of a bass and its length could be reasonably modeled by the following expression: $W = 0.0084365\ len^3$. The residuals are the differences between the predicted and observed values.

The observed values are stored in the original data *wt* array, while the predicted values must be calculated using the formula $W = 0.0084365\ len^3$. The residuals are calculated by subtracting each observed value from each predicted value. To analyze the results, the residuals are plotted using the graphical methods of Chapter II. Table 40 illustrates the method, and Figure 44 displays a plot of the residuals.

Command	Output
with(plots): with(stats): with(linalg):Digits:=4: *len:=[14.5,12.5,17.3,14.5,12.6,17.8,14.1,12.6];* *wt:=[27,17,41,26,17,49,23,16];* *lenwtfit:= fit[leastsquare [[x,y] , y = a*x^3, {a}]] ([len,wt]);* *f:=unapply(rhs(lenwtfit),x);* *predict:=map(x->f(x),len);* *resid:=matadd(wt,predict,1,-1);* *wtpred:={seq([len[i],resid[i]],i=1..8)}:* *plot(wtpred,style=point, symbol=diamond);*	len := [14.5, 12.5, 17.3, 14.5, 12.6, 17.8, 14.1, 12.6] wt := [27, 17, 41, 26, 17, 49, 23, 16] lenwtfit := y = 0.008436 x^3 f:= x -> 0.008436 x^3 predict:= [25.61 16.41 43.50 25.61 16.80 47.38 23.55 16.80] resids:= [1.39 .59 -2.50 .39 .20 1.62 -.55 -.80]

Table 40. Plotting Residuals.

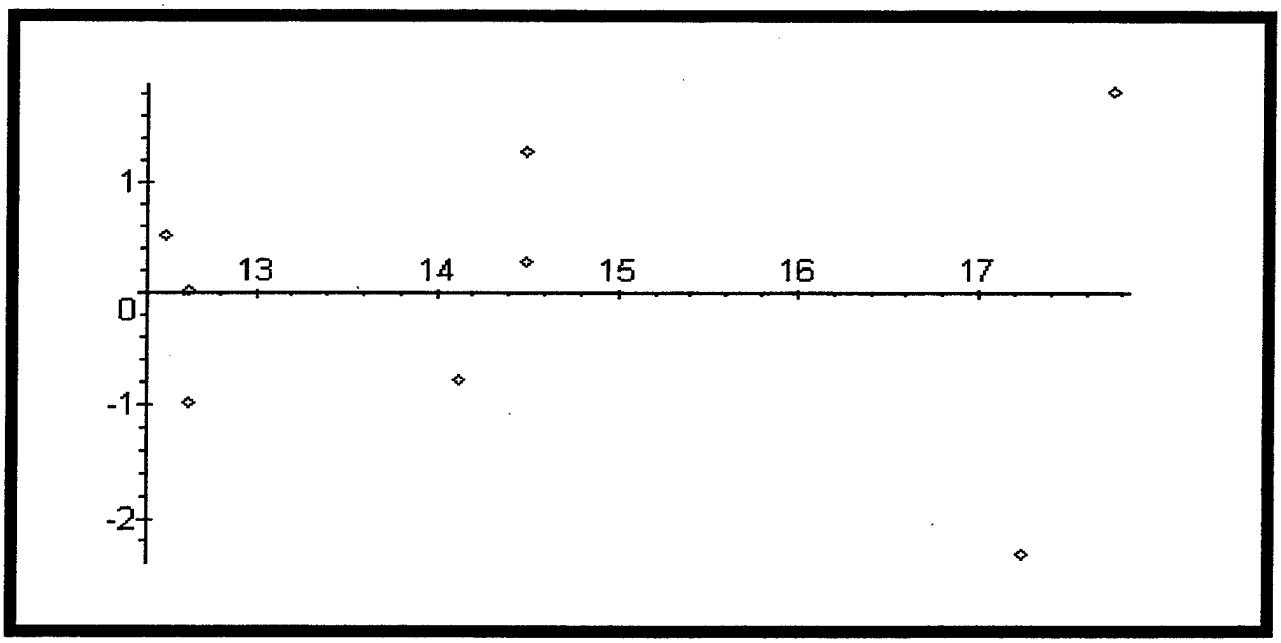

Figure 44. The Bass Derby Fishing Residuals.

Note that the residuals are randomly distributed and contained in a relatively small band about zero. There are no outliers, or unusually large residuals, and there appears to be no pattern in the residuals. Based on these aspects of the plot of the residuals, the model accurately approximates the data.

C. THE STATISTICS PACKAGE

As will be seen when using the Chebyshev criterion, certain statistical functions are required to analyze a model. Finding the largest absolute deviation, the sum of the absolute deviations, and the average absolute deviation are a few of the tasks that are simplified using Maple's statistics package.

Recall that the statistics package requires the *with(stats):* command prior to use of any statistics commands. Having activated the statistics package, the *describe* command is required to access any of the commands listed in Table 41.

count	covariance	kurtosis
mean	median	mode
moment	percentile	quantile
quartile	range	skewness
standarddeviation	variance	coefficientofvariation

Table 41. Commands Available With Describe.

Using the bass derby data set, the commands listed in Table 42 demonstrate finding a variety of statistical values from the *wt* data set.

Command	Output
wt:=[27,17,41,26,17,49,23,16];	wt:= [27 17 41 26 17 49 23 16]
with(stats):	
describe[median](wt): evalf(");	24.5
describe[mean](wt);	27
describe[count](wt);	8
describe[range](wt);	16..49
describe[variance](wt): evalf(");	127.25

Table 42. Example Statistics Commands.

1. The Chebyshev Criterion

As discussed above, the *fit* command determines the parameters of a model such that the sum of the squared residuals is minimized. Other curve-fitting criteria exist although these lead to different optimization problems. For example, the Chebyshev criterion determines the parameters of a model such that the largest absolute deviation is minimized. That is, the Chebyshev criterion determines the parameters of the function type y = f(x) that minimizes the number

$$\text{Maximum } | y_j - f(x_j) |, \; j = 1,2, \ldots, m \qquad \text{Equation 3.}$$

The formulation of a specific problem yields a mathematical program which is either linear or nonlinear. Denote the largest of the residuals that result from a Chebyshev fit by c_{max}. Thus, c_{max} is as small as possible if the Chebyshev criterion is used to determine the model parameters. A bound on c_{max} may be obtained using the results of a least-squares fit as follows:

$$D \leq c_{max} \leq d_{max} \qquad \text{Equation 4.}$$

where $D = ((\Sigma (d_j)^2)/ m)^{0.5}$ and d_{max} is the largest of the residuals d_j resulting when the least-squares criterion is used.... If the difference between D and d_{max} is large, and minimizing the largest absolute deviation is important in a particular application, one may wish to investigate the Chebyshev criterion further. (See Giordano and Weir, op. cit. , for a more thorough discussion of these ideas). (Fox, et al., op.cit., pages 80 and 81)

Table 43 presents the commands and outputs for the bounds for the bass fishing derby model $W = 0.0084365\ len^3$, which was fit with the *fit* command in Chapter IV, Section A. Maple's *sum* command requires a set and will not operate on a list, which requires the residuals to be re-entered.

Command	Output
with(plots): with(stats): with(linalg):Digits:=5: *len:=[14.5,12.5,17.3,14.5,12.6,17.8,14.1,12.6];*	len := [14.5, 12.5, 17.3, 14.5, 12.6, 17.8, 14.1, 12.6]
wt:=[27,17,41,26,17,49,23,16];	wt := [27, 17, 41, 26, 17, 49, 23, 16]
*lenwtfit:= fit[leastsquare [[x,y] , y = a*x^3, {a}]] ([len,wt]);*	lenwtfit := y = 0.0084365 x^3
f:=unapply(rhs(lenwtfit),x): *predict:=map(x->f(x),len):* *resid:=matadd(wt,predict,1,-1):* *residsum:=sum('(resid[k])^2','k=1..8');*	residsum := 12.173
bigd:=((residsum)/8)^(.5);	bigd := 1.2335
resids:=map(x->abs(x),resid): *Resid:=(resids[1],resids[2], resids[3],resids[4], resids[5],resids[6], resids[7],resids[8]):* *dmax:=max(Resid);*	dmax := 2.305

Table 43. The Bounds On c_{max}.

From Table 43, the bounds on c_{max} are

$$1.2335 \leq c_{max} \leq 2.305. \qquad \text{Equation 5.}$$

D. AN ILLUSTRATIVE EXAMPLE

In this section the vehicular stopping distance problem investigated in Example 3.2.2 of Chapter III is re-investigated to illustrate the use of the Maple commands presented thus far in a modeling application. Recall, using proportionality techniques the model $d = 1.1\,v + 0.057\,v^2$ described the data. The model consists of two submodels: $d_r \propto v$ for the reaction distance, d_r, and $d_b \propto v^2$ for the braking distance, d_b.

Table 44 presents the commands for Steps 1 and 2. Note, all the data sets have been previously entered and saved in Example 3.2.2 of Chapter III, so they can be used here without re-entering the data. In the first step, the submodel $d_r \propto v$, uses the data sets *v* and *dr* and fitting a least-squares criterion solves $y = 1.1040\,x$. In the next step, the residuals are plotted to analyze the fit. There is a trend in both sets of residuals which indicates that the model should be refined to address this effect. The routine is repeated to evaluate the second submodel. The submodel $d_b \propto v^2$ uses the data sets *v* and *db* and the least-squares criterion to determine the model $d_b = 0.054209\,v^2$. Thus, the least-squares fit produces the quadratic solution $d = 1.1040\,v + 0.054209\,v^2$. Figures 45 and 46 display the residual plots. Table 45 illustrates the commands used to determine the bounds for c_{max}.

Command	Output
with(stats):with(linalg):with(plots):Digits:=5: *v:=[20,25,30,35,40,45,50,55,60,65,70,75,80]:* *dr:=[22,28,33,39,44,50,55,61,66,72,77,83,88]:* *db:=[20,28,40.5,52.5,72,92.5,118,148.5,182,220.5,266,318,376]:* *vdrfit:=fit[leastsquare [[x,y] , y=a*x,{a}]] ([v,dr]): evalf(");* *f:=unapply(rhs(vdrfit),x):* *predict:=evalf(map(x->f(x),v)):* *resid:=evalf(matadd(dr,predict,1,-1)):* *vpred:={seq([v[i],resid[i]],i=1..13)}:* *plot(vpred,style=point, symbol=diamond);* *dbv2fit:=fit[leastsquare [[x,y] , y = a*x^2, {a}]]([v,db]): evalf(");* *f:=unapply(rhs(dbv2fit),x):* *predict:=evalf(map(x->f(x),v)):* *resid:=evalf(matadd(db,predict,1,-1)):* *vpred:={seq([v[i],resid[i]],i=1..13)}:* *plot(vpred,style=point, symbol=diamond);*	$y = 1.1040\ x$ $y = 0.054209\ x^2$

Table 44. The Vehicular Stopping Distance Residuals.

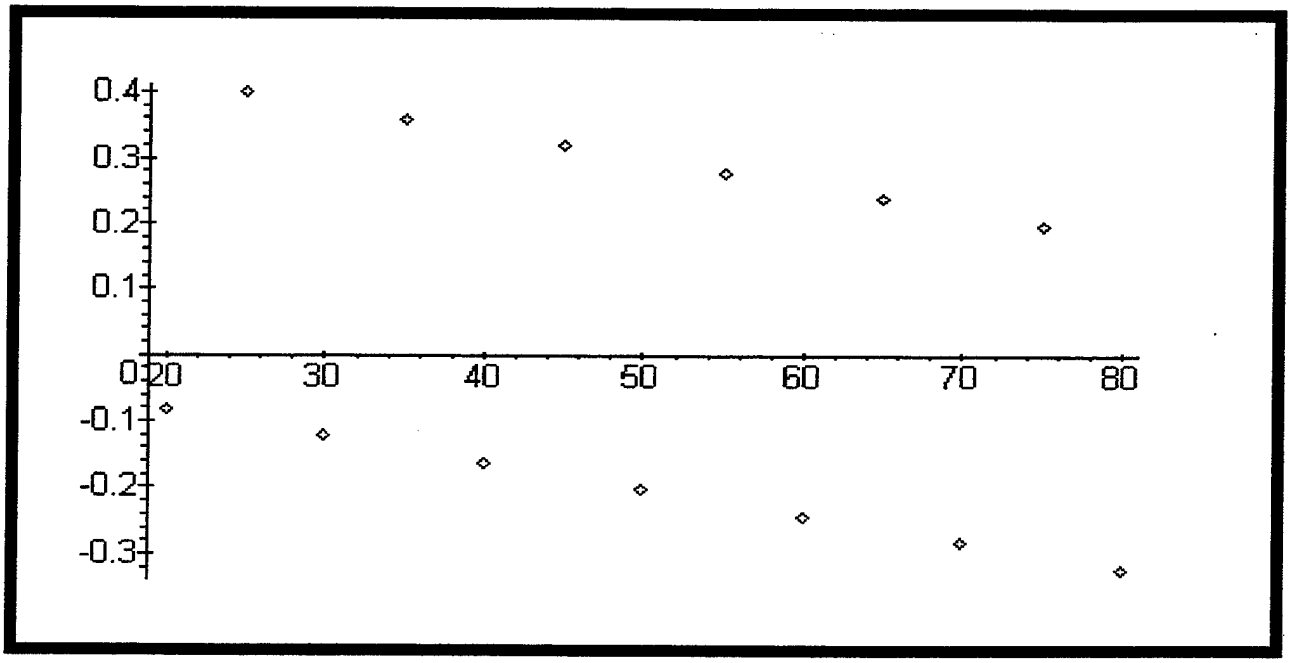

Figure 45. The Reaction Residuals, $y = 1.1040\ x$.

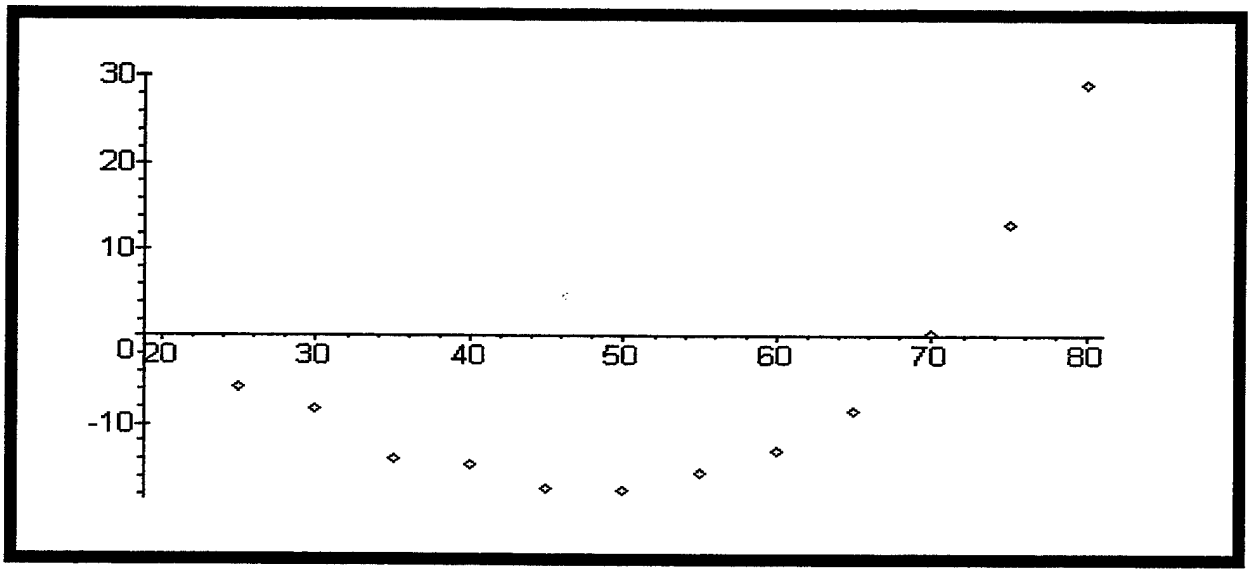

Figure 46. The Braking Residuals, $y = 0.054209\ x^2$.

Command	Output
residsum:=sum('(resid[k])^2','k=1..13'):	
bigd:=((residsum)/13)^(.5);	bigd := 14.203
resids:=map(x->abs(x),resid):	
Resid:=(resids[1],resids[2],resids[3], *resids[4],resids[5],resids[6],* *resids[7],resids[8],resids[9],* *resids[10],resids[11],resids[12],* *resids[13]):*	
dmax:=max(Resid);	dmax := 29.06

Table 45. The Bounds On c_{max}.

Both the residual plots show a trend which indicates that one or both of the submodels are not taking into account factors or relationships affecting the dependent variable. Further investigation reveals that the submodel for braking distance is also inaccurate at high speeds (see Giordano and Weir, op.cit., page 113). From Table 45, the bounds for c_{max} are:

$$14.203 \leq c_{max} \leq 29.06. \qquad \text{Equation 6.}$$

V. EMPIRICAL MODEL CONSTRUCTION

In previous chapters we assumed the modeler has developed a model relating the variables under consideration and explain, in some sense, the observed behavior. If collected data then corroborate the reasonableness of the assumptions underlying the hypothesized relationships, the parameters of the model can be chosen that 'best fit' the model type to the collected data according to some criterion (such as least-squares or the Chebyshev criterion).

In many practical cases the modeler is unable to construct a tractable model form that satisfactorily explains the behavior. Nevertheless, if it is necessary to predict the behavior, the modeler may conduct experiments... to investigate the behavior of the dependent variable(s) for selected values of the independent variable(s) within some range of interest. In essence the modeler desires to construct an empirical model based on the collected data. In this situation the modeler is strongly influenced by the data that have been carefully collected and analyzed, so he or she seeks a curve that captures the trend of the data in order to predict between the collected data points.

To contrast explicative and empirical models, consider the data shown in Figure 47. If the modeler's assumptions lead to the expectation of a quadratic explicative model, a parabola would be fit to the data points, as illustrated in Figure 48. However, if the modeler has no reason to expect a model of a particular type, a smooth curve may be passed through the data points to serve as an empirical model, as illustrated in Figure 49. (Fox, et al., op.cit., page 93)

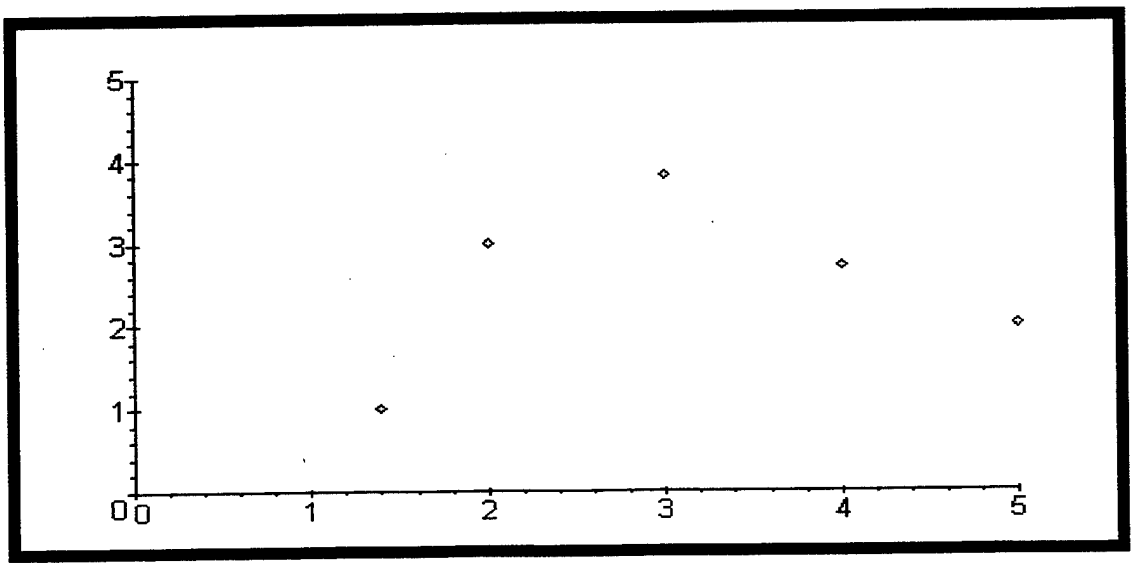

Figure 47. Random Data.

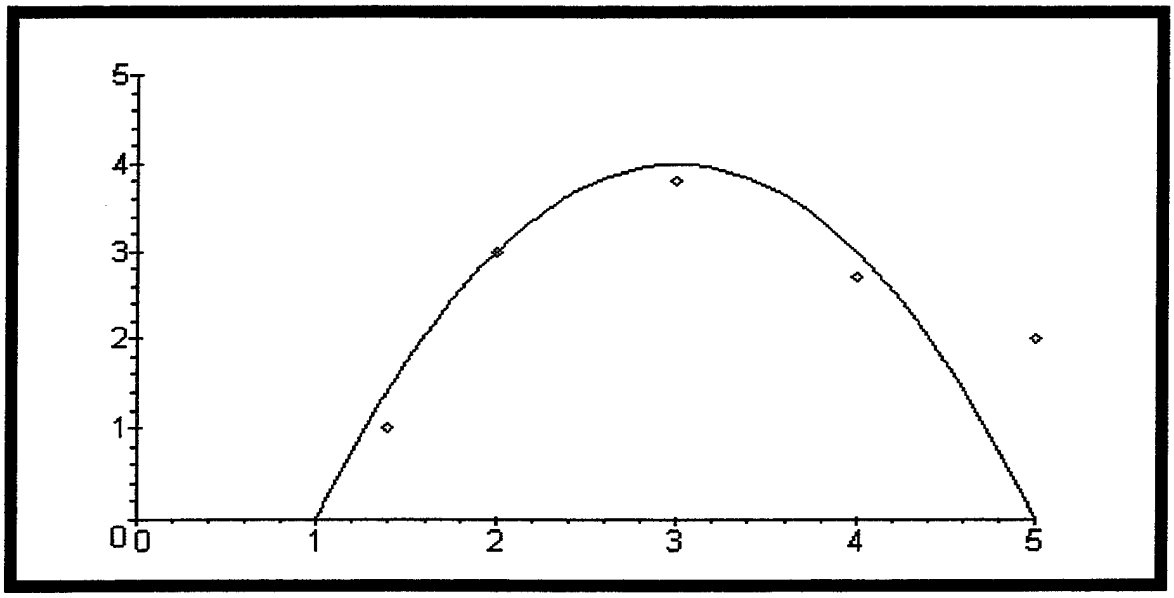

Figure 48. A Quadratic Fit.

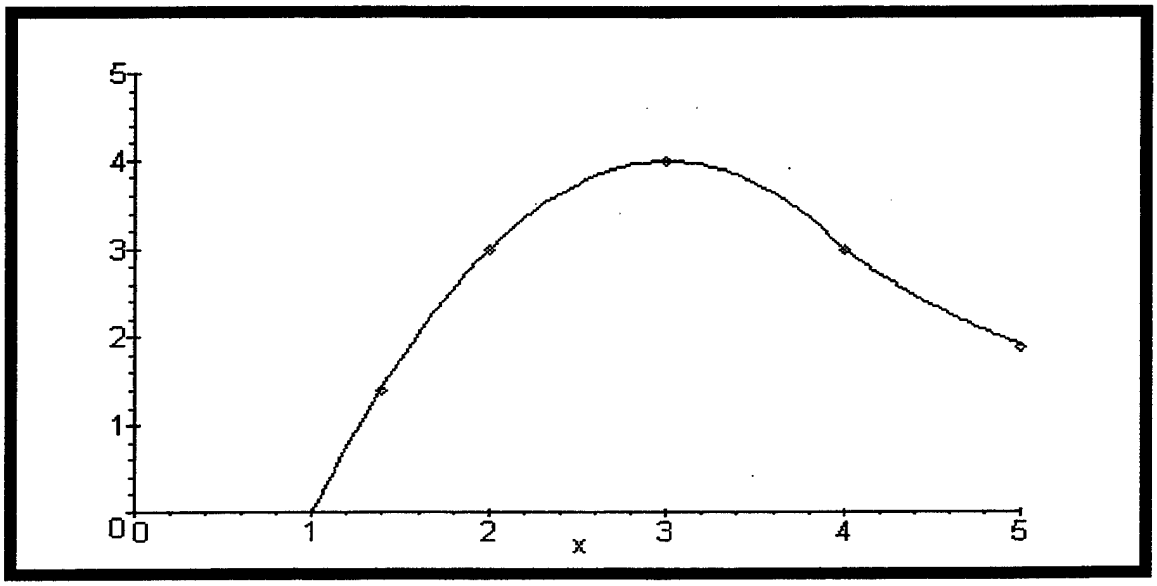

Figure 49. A Smooth Curve.

This chapter describes the construction of empirical models. While there are many techniques to assist in model construction, Table 46 presents a logical procedure for determining an appropriate empirical model.

1.	Examine a scatterplot of the data for trends.
2.	Examine outliers and replace/discard them.
3.	Find a one-term model and, if possible, fit the chosen one-term model, if the fit is adequate, end.
4.	Otherwise, attempt another one-term model or consider a polynomial model.
5.	From a divided difference table, fit a low-order polynomial; if adequate, end.
6.	Otherwise, construct a cubic spline.

Table 46. Empirical Model Process.

First examine if a trend in the data is discernible. If so, begin with the simplest technique available (a one-term model) and then proceed through the above steps until an empirical model is developed that satisfies the requirements of the particular application. As a general rule, we select the simplest model which captures the trend of the data. For a more detailed discussion of empirical model construction, see Chapter 6 of Giordano and Weir, op.cit.

A. ONE-TERM MODELS

> We commence our study of empirical model building with the presentation of simple one-term models. These models have several advantages. One obvious and powerful advantage is their simplicity. Another is that a one-term model may capture the trend of the data better than any other empirical model (such as a polynomial). This feature is particularly important in situations when the modeler intends to make a prediction by extrapolating for values outside the intervals of the observations; for instance, when the modeler intends to predict a future value based on historical information.
>
> When constructing an empirical model, always begin with a careful analysis of the collected data. Investigate whether the data suggest the existence of a trend. Are there data points that fail to follow the trend? If such 'outliers' do exist, you may wish to discard them. Or if they were obtained experimentally, you may choose to repeat the experiment as a check for a data collection error. (Fox, et al., op.cit., page 96)

If a trend is observed to exist, the use of one-term models is investigated by matching of the trend's concavity using the "Ladder of Transformations." For a more detailed discussion of the "Ladder of Transformations," see Chapter 6 of Giordano and Weir, op.cit.

1. Example 5.1: The Bass Fishing Derby

> Consider the Bass Fishing Derby discussed previously in Example 3.2.1 in Chapter III. In Example 5.1, the scatterplot of the original data suggests a trend which is concave up. Thus we select the z^2 transformation and plot W vs len^2. Since the plot is reasonably linear, we fit the model analytically (using least-squares) and plot the residuals. Since there is an evident trend in the residuals, we investigate W vs len^3 in a similar manner. The two resulting models are: $W = 0.13108\ len^2$ and $W = 0.00844\ len^3$. Note that the residuals of W vs len^3 are more random about the origin, indicating that the second model is the better one. (Fox, et al., op.cit., page 98)

The commands are presented in Tables 47 through 49 and the plots are located in Figures 50 through 56.

Commands
len:=[14.5,12.5,17.25,14.5,12.625,17.75,14.125,12.625]:
wt:=[27,17,41,26,17,49,23,16]:
lw:={seq([len[i],wt[i]],i=1..8)}:
plot(lw,style=point,view=[0..20,0..50]);
len2:=map(x->x^2,len):
l2w:= {seq([len2[i],wt[i]],i=1..8)}:
plot(l2w,style=point,view=[0..400,0..50]);

Table 47. The Bass Fishing Derby.

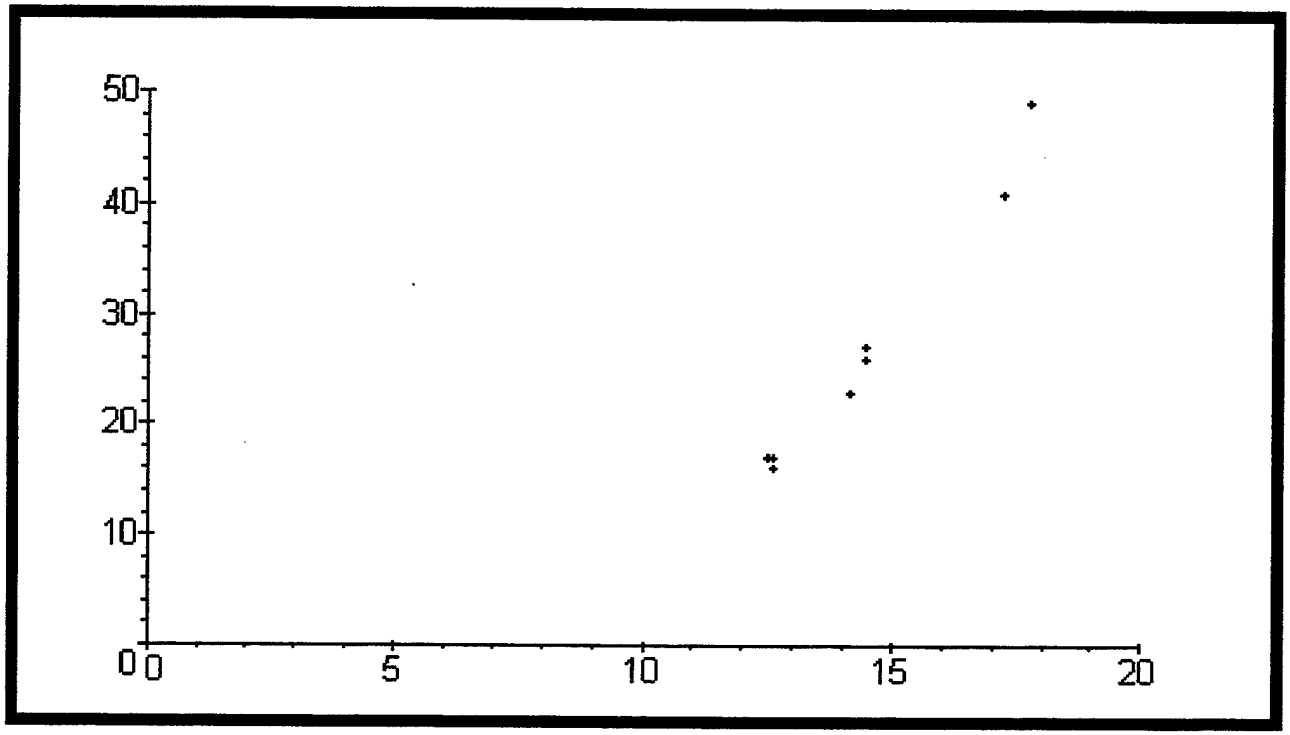

Figure 50. Length vs. Weight.

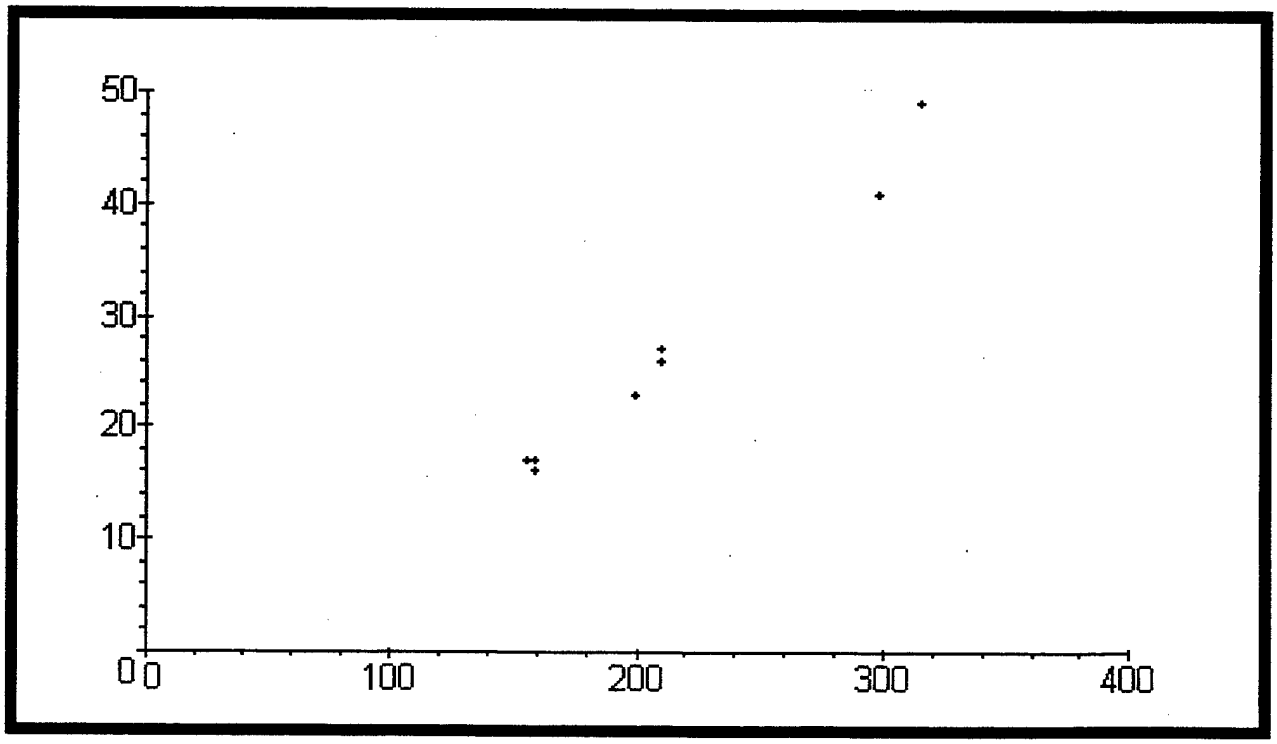

Figure 51. Length^2 vs. Weight.

Command	Output
with(plots): with(stats): with(linalg):Digits:=5: *len:=[14.5,12.5,17.3,14.5,12.6,17.8,14.1,12.6]:* *wt:=[27,17,41,26,17,49,23,16]:* *fit1:=fit[leastsquare[[x,y],y=a*x^2,{a}]]([len,wt]): evalf(");* *f:=unapply(rhs(fit1),x):* *predict:=map(x->f(x),len):* *resid:=matadd(wt,predict,1,-1):* *wtpred:={seq([len[i],resid[i]],i=1..8)}:* *plot(wtpred,style=point, symbol=diamond);* *plot1:=plot(lw,style=point,view=[0..20,0..50],color=black):* *plot2:=plot(f(x),x=0..20):* *display({plot1,plot2});*	$y = 0.13108\ x^2$

Table 48. The Bass Fishing Derby.

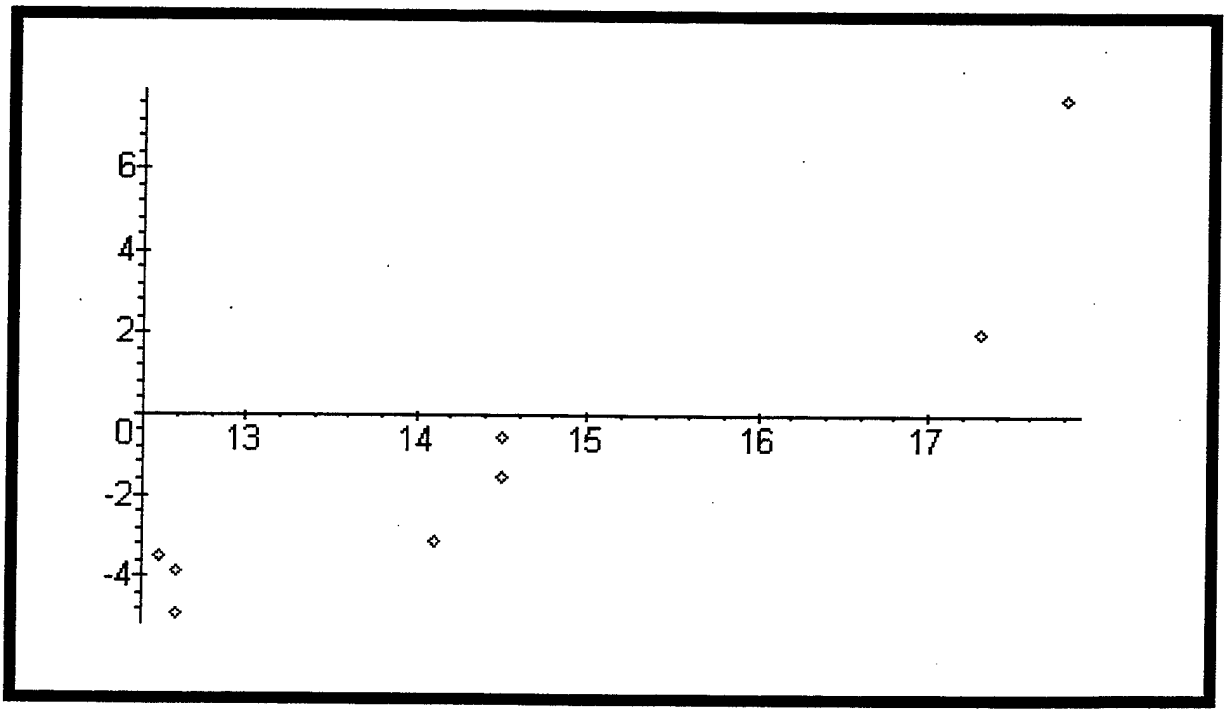

Figure 52. The Second Order Bass Residuals.

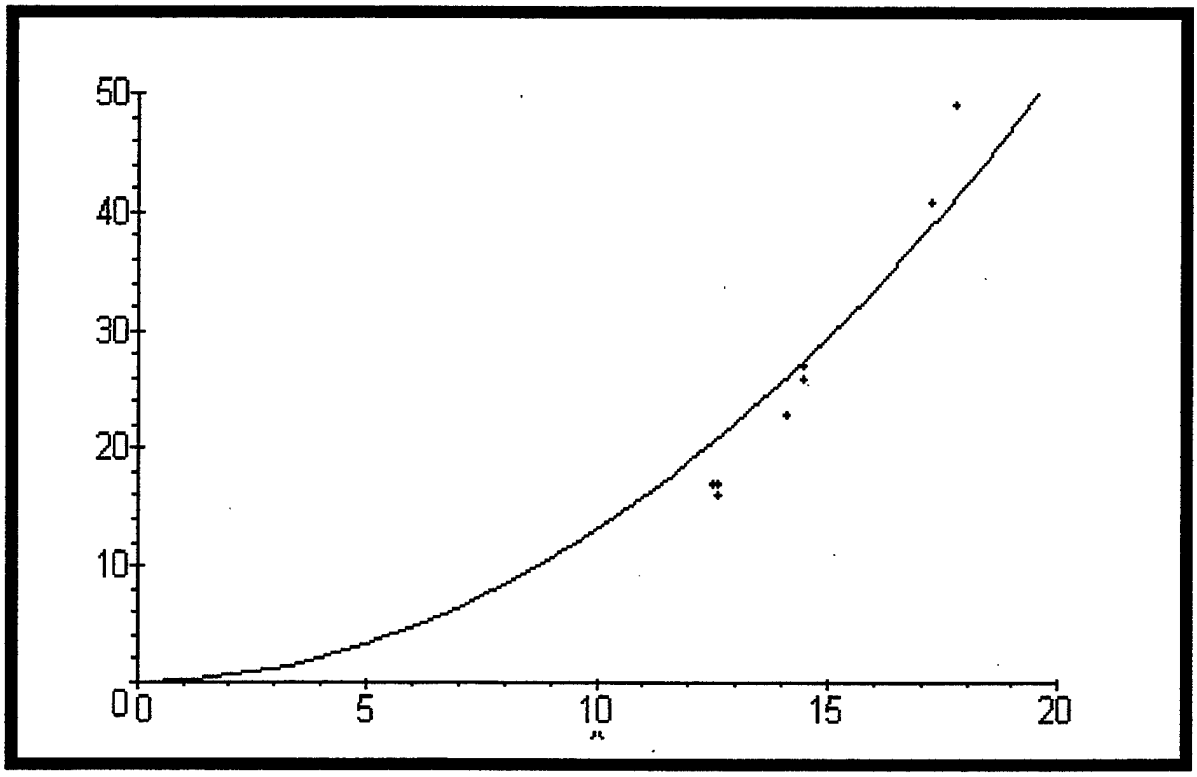

Figure 53. The Graph Of $y = 0.13108\ x^2$ Superimposed Over The Original Data.

Command	Output
len3:=map(x->x^3,len): *l3w:= {seq([len3[i],wt[i]],i=1..8)}:* *plot(l3w,style=point,view=[0..5600,0..50]);* *fit2:=fit[leastsquare[[x,y],y=a*x^3,{a}]]([len,wt]):evalf('');* *f2:=unapply(rhs(fit2),x):* *predict2:=map(x->f2(x),len):* *resid2:=matadd(wt,predict2,1,-1):* *wtpred2:={seq([len[i],resid2[i]],i=1..8)}:* *plot(wtpred2,style=point, symbol=diamond);* *plot3:=plot(.0084365*x^3,x=0..20):* *display({plot1,plot3});*	$y = 0.0084365\ x^3$

Table 49. The Bass Fishing Derby.

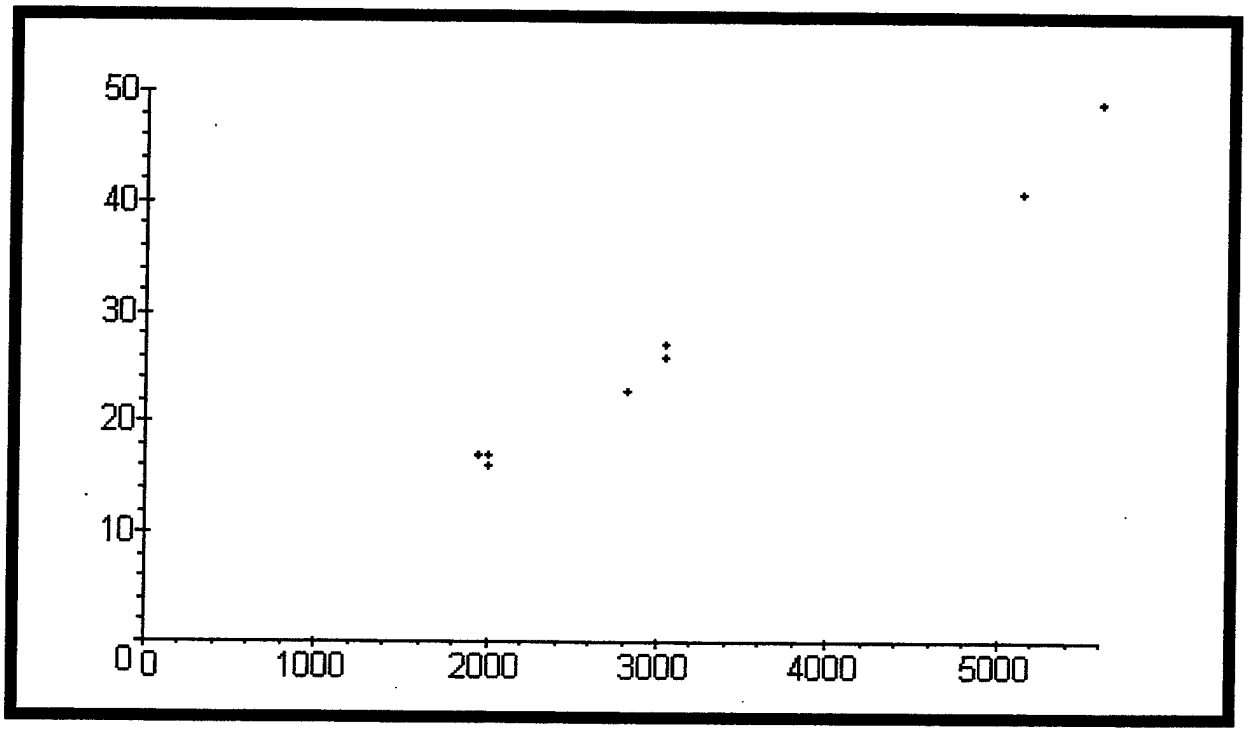

Figure 54. Length^3 vs. Weight.

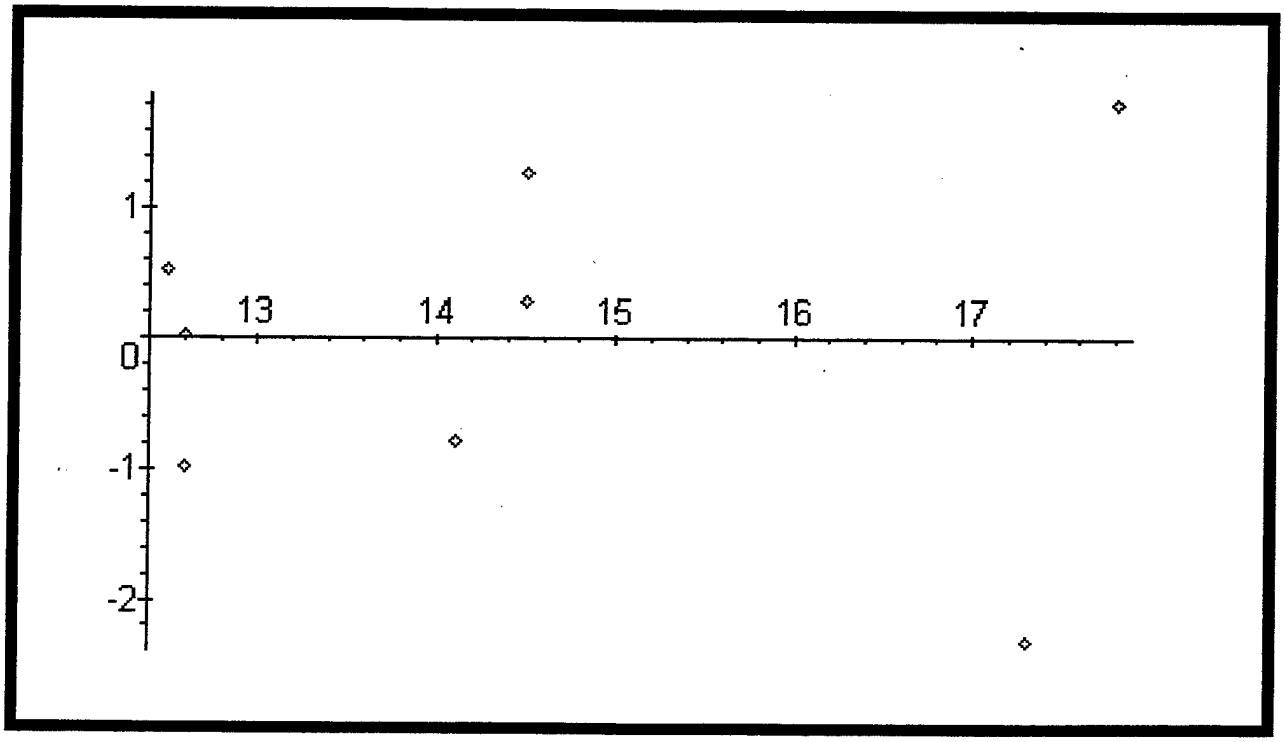

Figure 55. The Third Order Bass Residuals.

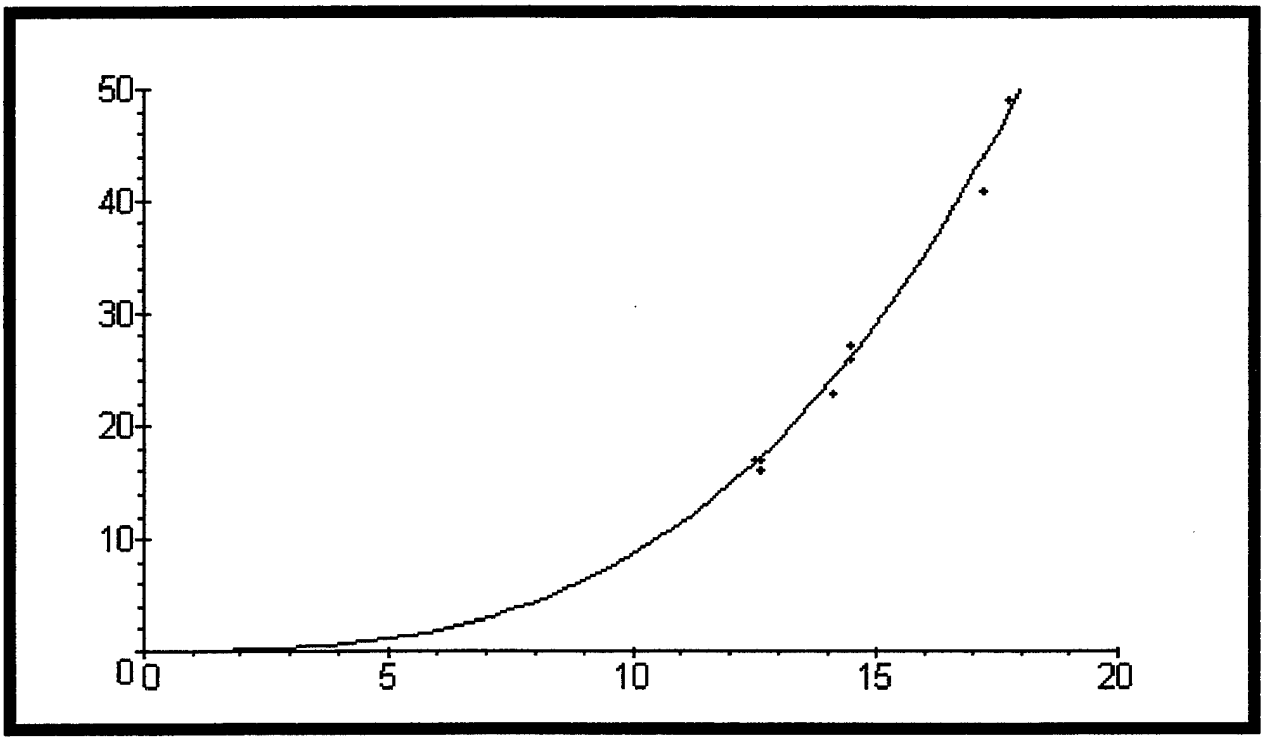

Figure 56. The Graph Of $y = 0.0084365\ x^3$ Superimposed Over The Original Data.

Figure 52 displays the residuals of the W vs len^2 model. The residuals are in three clusters and show an increasing trend. Figure 55 displays the W vs len^3 model, and more randomness of the residuals is displayed in this graph than in Figure 52. As a result, the W vs len^3 model is selected as the more accurate model in capturing the data.

B. FITTING AN N-1 ORDER POLYNOMIAL TO N DATA POINTS

> Because of the inherent simplicity, one-term models facilitate model analysis including sensitivity analysis, optimization, estimation of rates of change and area under a curve applications, and so forth. However, precisely because of this simplicity, one-term models are not likely to capture the trend of an arbitrary data set. In many cases models with more than one term must be considered. The remainder of this chapter considers one type of multiterm model; namely, the polynomial. Since polynomials are easy to integrate and differentiate, they are especially popular to use.... Consider passing a quadratic $P_2(x) = a + b\ x + c\ x^2$ through the following data points in Table 50.

x	1	2	3
y	5	8	25

Table 50. Data Points For A Polynomial.

Requiring that $P_2(x_i) = y_i$ yields the following system of equations in Table 51.

a +	1 b +	1 c	=	5
a +	2 b +	2^2 c	=	8
a +	3 b +	3^2 c	=	25

Table 51. The System Of Equations.

To solve the above system conveniently using Maple, we consider the linear system in the form of the matrix equation $AX = B$, which has the solution $X = A^{-1} B$, provided that A is invertible. Thus, the above system can be written in matrix form as

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \bullet \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \\ 25 \end{bmatrix} \quad \text{where } X = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \qquad \text{Equation 7.}$$

and, since the coefficient matrix is invertible, the solution is

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 3 & -3 & 1 \\ -2.5 & 4 & -1.5 \\ 0.5 & -1 & 0.5 \end{bmatrix} \bullet \begin{bmatrix} 5 \\ 8 \\ 25 \end{bmatrix} = \begin{bmatrix} 16 \\ -18 \\ 7 \end{bmatrix} \qquad \text{Equation 8.}$$

or $P_2(x) = 16 - 18x + 7x^2$.

In general, the requirement that an n-1 degree polynomial pass through n distinct data points... yields a system of n linear algebraic equations in n unknowns. It is important to realize that large systems of equations can be difficult to solve with great accuracy, and small round-off errors in computer arithmetic can cause large oscillations to occur due to the presence of the higher-order terms. (Fox, et al., op.cit., pages 107 and 108)

The process of solving for a, b and c by inverting A and multiplying it by B is computed by Maple in one step with the *gaussjord* command. Additionally, an in-depth discussion of the causes and effects of these oscillations is presented in Chapter 6 of Giordano and Weir, op.cit pages 184 through 186.

1. Example 5.2: An N-1 Degree Polynomial

The Maple linear algebra package is capable of in-depth matrix manipulations. Specifically, Maple quickly solves the type of problem discussed above with one command. Reviewing that earlier example:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \bullet \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \\ 25 \end{bmatrix} \qquad \text{Equation 9.}$$

Table 52 presents the Maple commands which determine the values a, b and c.

Command	Output
with(linalg): *B:=array([[1,1,1,5],[1,2,4,8],[1,3,9,25]]);*	$B := \begin{bmatrix} 1 & 1 & 1 & 5 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 25 \end{bmatrix}$
gaussjord(B);	$\begin{bmatrix} 1 & 0 & 0 & 16 \\ 0 & 1 & 0 & -18 \\ 0 & 0 & 1 & 7 \end{bmatrix}$

Table 52. Gauss-Jordan Elimination.

providing the coefficients for the solution of $P_2(x) = 16 - 18x + 7x^2$. The graphical presentation demonstrates the fit of the solution, presented in Table 53 and Figure 57.

Commands
with(linalg):
x:=[1,2,3]:
y:=[5,8,25]:
xy:={seq([x[i],y[i]],i=1..3)}:
with(plots):
plot1:=plot(xy,style=point,symbol=circle):
*plot2:=plot(16-18*z+7*z^2,z=0..5):*
display({plot1,plot2});

Table 53. Plotting A Quadratic Curve.

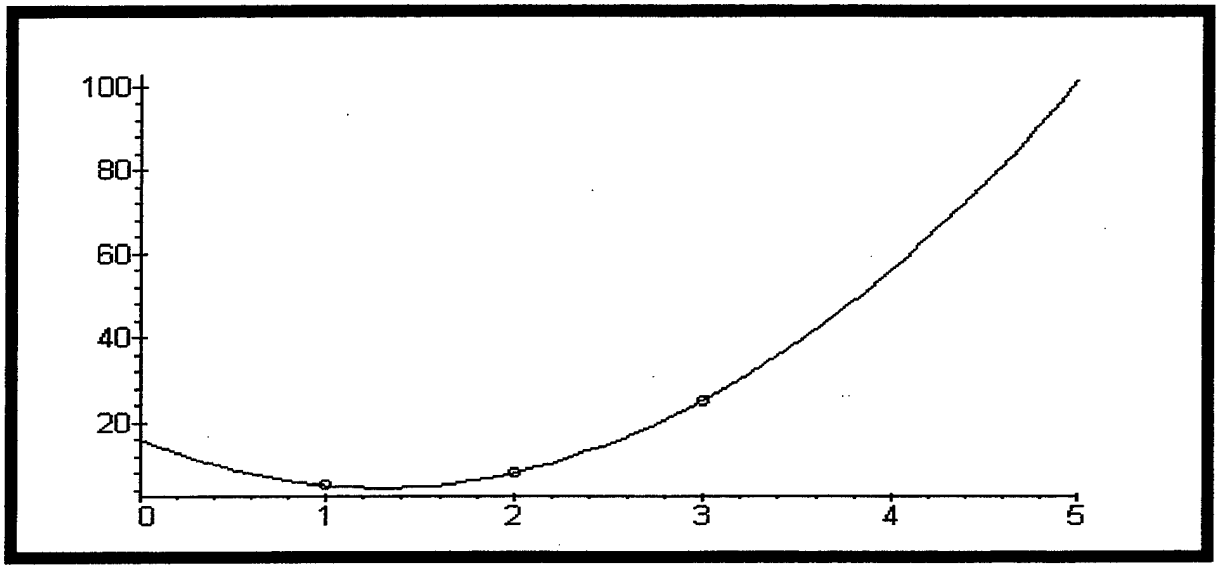

Figure 57. The Quadratic Curve Passes Through The Points.

C. POLYNOMIAL SMOOTHING

Smoothing with low-order polynomials is an attempt to retain the advantages of polynomials as empirical models while at the same time reducing the tendencies of higher-order polynomials to snake and oscillate. The idea of smoothing is illustrated graphically in Figure 58.

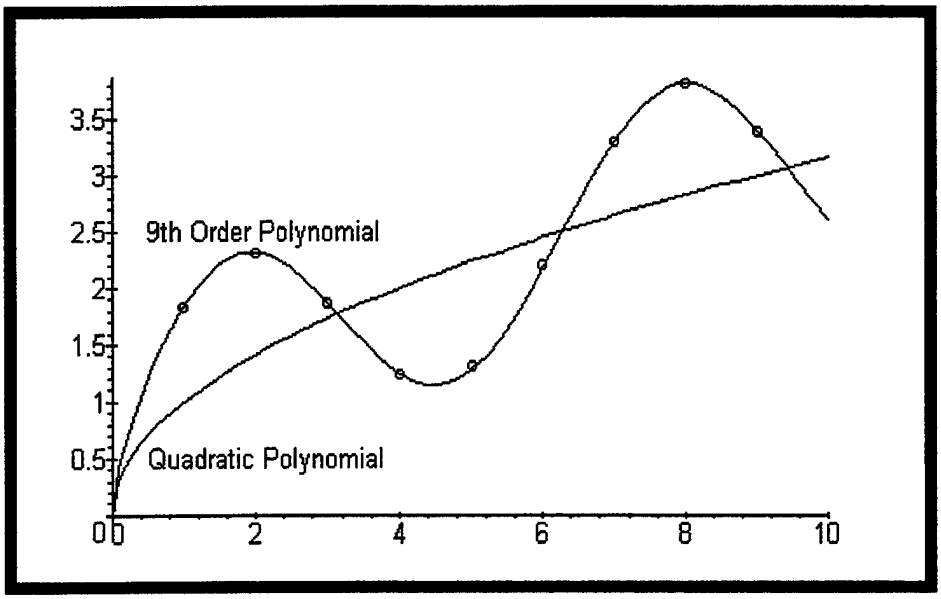

Figure 58. Polynomial Smoothing.

Rather than forcing a polynomial to pass through all the data points, a low-order polynomial is chosen and fit to the data. This choice normally results in a situation where the number of data points exceeds the number of constants necessary to determine the polynomial. Smoothing with polynomials requires two decisions:

- Is a low-order polynomial appropriate? If so, what should the order be?
- What are the parameters of the model according to some criterion of 'best fit,' such as the least-squares criterion? (Fox, et al., op.cit., page 114)

1. Example 5.3: Fitting A 5th-Order Polynomial Using Least-Squares

Given a set of data points, see Table 54, an analyst decides to attempt to fit a curve to the data using a high order polynomial. Table 55 displays the commands required and Figure 59 displays the polynomial curve superimposed on the data.

x	1	2	3	4	5	6
y	305	266	135	-16	125	1230

Table 54. The Data Points For Example 5.3.

Command	Output
with(stats):with(plots): *xdata:=[1,2,3,4,5,6]:* *ydata:=[305,266,135,-16,125,1230]:* *xyfit:=fit[leastsquare[[x,y],y=a*x^5+b*x^4+c x^3+d*x^2+e*x+g,{a,b,c,d,e,g}]]([xdata, ydata]);* *f:=unapply(rhs(xyfit),x):* *xy:={seq([xdata[i],ydata[i]],i=1..6)}:* *plot1:=plot(xy,style=point,symbol=diamond):* *plot2:=plot(f(x),x=0..6):* *display({plot1,plot2});*	xyfit:= $y = x^5 - 5x^4 - 3x^3 + 7x^2 + 5x + 300$

Table 55. The Commands For Example 5.3.

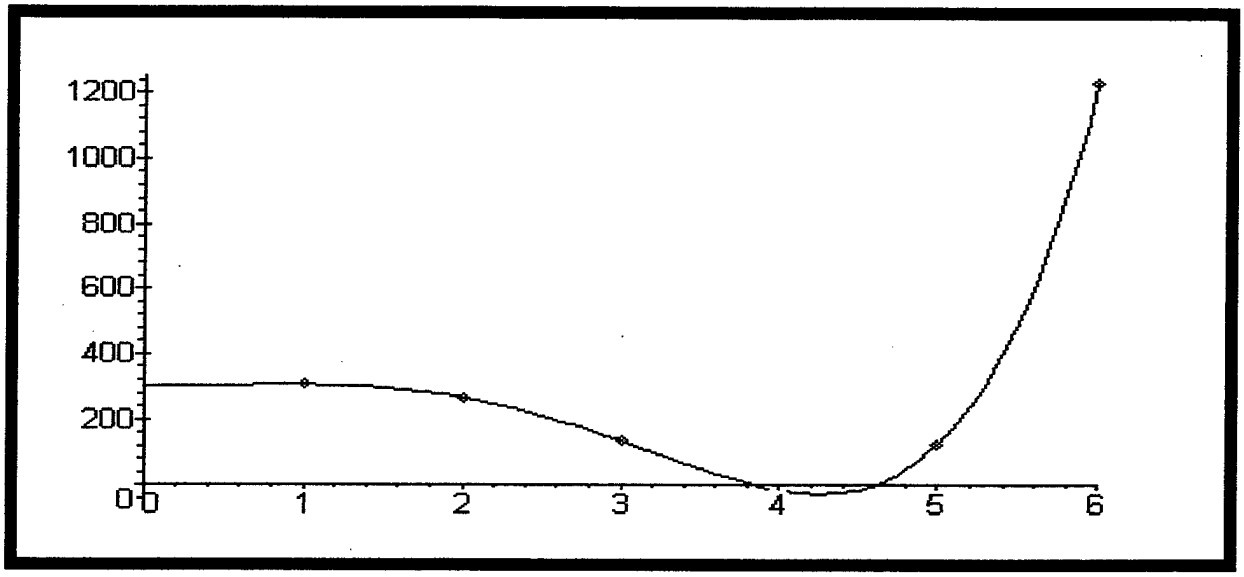

Figure 59. The Plot For Example 5.3.

D. DIVIDED DIFFERENCE TABLES AND CUBIC SPLINES

Thus far in this chapter, methods for approximating trends in data have been explored using one-term models when appropriate otherwise using polynomial models. When deciding to use a polynomial model, two questions must be answered: is a polynomial appropriate and what order polynomial would best describe the data? Section 6.3 of Giordano and Weir, op.cit., offers a detailed discussion of how to answer these

questions using the derivative and divided difference tables. This chapter describes the Maple commands required to develop a divided difference table and the technique of a solution using the cubic spline.

1. The Divided Difference Table

The divided difference table is composed of two columns of data and successive columns of divided differences which approximate derivatives. A few new functions are introduced in this section: the *for*, *if*, *proc*, *save* and *read* commands. The *for* statement is a repetition statement which executes a command a specific number of times. The *if* statement is a conditional statement which executes a command when a condition is met. *Proc* is the command to create a procedure. A procedure is an expression or group of expressions which are assigned to a name, in this case, *newproc*. The procedure will evaluate a parameter or set of parameters as defined in the command statement, inside the parenthesis immediately following *proc*. The local command designates local variables which will have no value outside of the procedure. Once a name, *newproc*, has been assigned a procedure, then it may be invoked by calling *newproc(arguments)*, where the procedure will be applied to specific arguments. Table 56 demonstrates applications of all of these commands and their results. The first *for* loop prints the integers from 1 to 3. The second *for* loop creates a matrix with values 2,4,6, and 8. The *if* statement evaluates if the first value, 2 is less than the third value, 6, since this is true, it replaces the fourth value, 8, with the value 17. The first procedure will return two times the square root of the argument of interest. The second procedure will divide the product of two arguments by the sum of the same two arguments. The *proc* commands is presented in two different

formats, both are correct. Once a procedure has been created it can be saved and used at some later point by reading the saved procedure. The *save* command will save the procedure, while the *read* command can be used at any later time to use the procedure without re-typing the procedure.

Command	Output
for j from 1 to 3 do print(j) od;	1 2 3
with(linalg):	
amatrix:=matrix(1,4);	amatrix:= array(1..1,1..4,[ ])
*for j from 1 to 4 do amatrix[1,j]:=2*j: od:*	
evalm(amatrix);	$\begin{bmatrix} 2 & 4 & 6 & 8 \end{bmatrix}$
if amatrix[1,1]<amatrix[1,3] then amatrix[1,4]:=17: fi: evalm(amatrix);	$\begin{bmatrix} 2 & 4 & 6 & 17 \end{bmatrix}$
newproc1:=proc(x) *local twosqrt;* *twosqrt:=2*x^(.5);* *RETURN (evalf(twosqrt))* *end;*	newproc1 := proc(x) local twosqrt; twosqrt:= 2*x^.5; RETURN(evalf(twosqrt)) end
newproc1(14);	7.483314774
newproc1(121);	22
newproc2:=proc(x,y) local pos; *pos:= x*y/(x+y); RETURN(pos) end;*	newproc2 := proc(x,y) local pos; pos := x*y/(x+y); RETURN(pos) end
newproc2(.5,2);	0.4
newproc2(3,6);	2
save newproc2, `TestProcedure1.m`: *read(`TestProcedure1.m`);*	

Table 56. The *For, If, Proc, Save* And *Read* Commands.

a. Example 5.4.1: Generating A Divided Difference Table

Using a hypothetical set of data from Giordano and Weir, op.cit., page 191, Table 57 demonstrates the creation of a divided difference table using Maple

commands in a step by step format. Notice in Table 57 the *evalm(xym);* command displays the matrix. The command has been used three times to demonstrate the evolution of the matrix, the three matrices have been collocated in Figure 60 for ease of presentation. The first matrix is the empty *xym* matrix while the second matrix is the *xym* matrix with the *xdata* and *ydata*. The third matrix is the finished divided difference table.

Command	Output
with(stats): with(linalg): *xdata:=[0,2,4,6,8];* *ydata:=[0,4,16,36,64];* *lengthdata:=describe[count](ydata);* *xym:=matrix(5,5);* *for j from 1 to 5 do for k from 1 to 5 do* *xym[j,k]:=0: od: od:* *evalm(xym);* *for j from 1 to 5 do xym[j,1]:=xdata[j]: od:* *for j from 1 to 5 do xym[j,2]:=ydata[j]: od:* *evalm(xym);* *for j from 2 to 4 do for k from j to 5 do* *xym[k,j+1]:=(xym[k,j]-xym[k-1,j])/* *(xym[k,1]-xym[k-(j-1),1]):* *if abs(xym[k,j+1])<10^(-5) then* *xym[k,j+1]:=0* *fi: od: od:* *evalm(xym);*	xdata:= [0, 2, 4, 6, 8] ydata:= [0, 4, 16, 36, 64] lengthdata:=5 xym:= array(1..5,1..5,[ ])

Table 57. A Divided Difference Table.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 \\ 4 & 16 & 0 & 0 & 0 \\ 6 & 36 & 0 & 0 & 0 \\ 8 & 64 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 2 & 0 & 0 \\ 4 & 16 & 6 & 1 & 0 \\ 6 & 36 & 10 & 1 & 0 \\ 8 & 64 & 14 & 1 & 0 \end{bmatrix}$$

Figure 60. The Evolution Of The Divided Difference Table.

Having examined each step of the creation of the divided difference table in Table 57, Table 58 demonstrates the divided difference table procedure, *ddproc*, which will be applied to the previous example and to Example 5.4.2. Table 59 presents the final solution of the previous example using the *ddproc* procedure.

Commands
ddproc:=proc(x,y) local len, xym, j, k; *len:=describe[count](y):* *xym:=matrix(len,len):* *for j from 1 to len do for k from 1 to len do xym[j,k]:=0: od:od:* *for j from 1 to len do xym[j,1]:=x[j]:od: for j from 1 to len do xym[j,2]:=y[j]:od:* *for j from 2 to (len-1) do for k from j to len do* *xym[k,j+1]:=(xym[k,j]-xym[k-1,j])/(xym[k,1]-xym[k-(j-1),1]):* *if abs(xym[k,j+1])<10^(-5) then xym[k,j+1]:=0: fi: od: od:* *Digits:=3:* *RETURN (evalf(evalm(xym)))* *end:*

Table 58. The Divided Difference Table Procedure.

Command	Output
with(stats): with(linalg): *xdata:=[0,2,4,6,8]:* *ydata:=[0,4,16,36,64]:* *ddproc(xdata,ydata);*	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 2 & 0 & 0 \\ 4 & 16 & 6 & 1 & 0 \\ 6 & 36 & 10 & 1 & 0 \\ 8 & 64 & 14 & 1 & 0 \end{bmatrix}$

Table 59. An Application Of The Divided Difference Table Procedure.

b. Example 5.4.2: Vehicular Stopping Distance

The divided difference table assists in determining what order polynomial should be used to approximate a set of data. This can be demonstrated using the vehicular stopping distance example, previously solved using the quadratic $d = 1.104\ v + 0.0541\ v^2$ in Section D of Chapter IV. Table 60 and Figure 61 display this example.

Commands
with(stats): with(linalg):
dx:= [20,25,30,35,40,45,50,55,60,65,70,75,80]:
dy:= [42,56,73.5,91.5,116,142.5,173,209.5,248,292.5,343,401,464]:
ddproc(dx,dy);

Table 60. The Commands For The Divided Difference Table.

20	42	0	0	0	0	0	0	0	0	0	0	0
25	56	2.8	0	0	0	0	0	0	0	0	0	0
30	73.5	3.5	.07	0	0	0	0	0	0	0	0	0
35	91.5	3.6	.01	−.004	0	0	0	0	0	0	0	0
40	116	4.9	.13	.008	.0006	0	0	0	0	0	0	0
45	143	5.3	.04	−.006	−.0007	−.000052	0	0	0	0	0	0
50	173	6.1	.08	.0027	.00043	.000045	0	0	0	0	0	0
55	210	7.3	.12	.0027	0	−.000017	0	0	0	0	0	0
60	248	7.7	.04	−.005	−.0004	−.000016	0	0	0	0	0	0
65	292	8.9	.12	.005	.00053	.000037	0	0	0	0	0	0
70	343	10.1	.12	0	−.00027	−.000032	0	0	0	0	0	0
75	401	11.6	.15	.002	.0001	.000015	0	0	0	0	0	0
80	464	12.6	.10	−.003	−.00027	−.000015	0	0	0	0	0	0

Figure 61. The Vehicular Stopping Distance Divided Difference Table.

From Figure 61, all of the divided differences above the first divided difference are significantly small in comparison to the data. However, the third, four and fifth divided differences all demonstrate a fluctuation between positive and negative values, which suggests measurement errors or other errors which would not be desired in a high order polynomial. This eliminates all possibilities but the quadratic polynomial to

approximate the data. Table 61 demonstrates the Maple solution of the quadratic using the least-squares criterion, for an in-depth discussion of this example see Giordano and Weir, op.cit., page 195-196.

Command	Output
with(stats): with(linalg):	
*vdfit:= fit[leastsquare [[x,y] , y = a*x^2+b*x+c, {a,b,c}]] ([dx,dy]);*	vdfit:= y = .0886 x^2 - 1.970x + 50.06
f:=unapply(rhs(vdfit),x):	
predict:=evalf(map(x->f(x),dx)):	
resid:=evalf(matadd(dy,predict,1,-1)):	
residsum:=sum('(resid[k])^2','k=1..13'):	
bigd:=((residsum)/13)^(.5);	bigd:= 2.6866
resids:=(abs(resid[1]),abs(resid[2]),abs(resid[3]), abs(resid[4]),abs(resid[5]),abs(resid[6]), abs(resid[7]),abs(resid[8]), abs(resid[9]), abs(resid[10]),abs(resid[11]),abs(resid[12]), abs(resid[13])):	
dmax:=max(resids);	dmax := 4.5660

Table 61. The Vehicular Stopping Distance Bounds For c_{max}.

From Table 61, $d = 50.06 - 1.970\,v + 0.0886\,v^2$ approximates the data much better than the previous quadratic $d = 1.104\,v + 0.0542\,v^2$ as shown by the bounds for c_{max}. Below the bounds for c_{max}, as computed using the original quadratic equation, followed by the bounds for c_{max}, computed using the new quadratic equation, demonstrate a definitive improvement using the new equation:

$$14.203 \leq c_{max} \leq 29.06 \qquad \text{Equation 10.}$$

$$2.6866 \leq c_{max} \leq 4.5660. \qquad \text{Equation 11.}$$

By computing the divided difference table, the order of the polynomial is accurately predicted. Using the least-squares technique in Chapter IV, the polynomial is determined and the residuals demonstrate the improvement over the techniques demonstrated in Chapter IV.

2. The Cubic Spline

> In this section we introduce cubic spline interpolation as an alternative method for constructing empirical models. By using different cubic polynomials between successive pairs of data points and by connecting the cubics together in a smooth fashion, we can capture the trend of the data, regardless of the underlying relationships. Simultaneously we will reduce the tendency towards oscillation and the sensitivity of the coefficients to changes in the data. (Fox, et al., op.cit., pages 131 and 132)

Maple has a spline command, which may be used only after issuing the command *readlib(spline):* which activates the spline command. By reviewing the Example 5.2 in Section B, the cubic spline method can be applied to the data with the commands presented in Table 62, with output in Table 63 and Figure 62. The *spline* command produces the group of polynomials which define the curve. The command *s:=unapply(",x);* assigns to *s* the value of the curve at a particular point, which is displayed with two arbitrary points: s(1.4859) and s(2.3).

Command	Output
readlib(spline):	
dx:=[1,2,3]:dy:=[5,8,25]:	
spline(dx,dy,x,cubic);	$\begin{cases} 2+10x-10.5x^2+3.5x^3 & x<2 \\ 58-74x+31.5x^2-3.5x^3 & otherwise \end{cases}$
s:=unapply(",x);	s:= s→ piecewise
	$\left(x<2,\quad 2+10x-10.5x^2+3.5x^3,\ 58-74x+31.5x^2-3.5x^3\right)$
dxy:={seq([dx[i],dy[i]],i=1..3)};	dxy := {[1, 5], [2, 8], [3, 25]}
plot1:=plot(s(x),x=0..4):	
plot2:=plot(dxy,style=point, symbol=box):	
with(plots):	
display({plot1,plot2});	
s(1.4859);	5.16
s(2.3);	11.85

Table 62. The Spline Command.

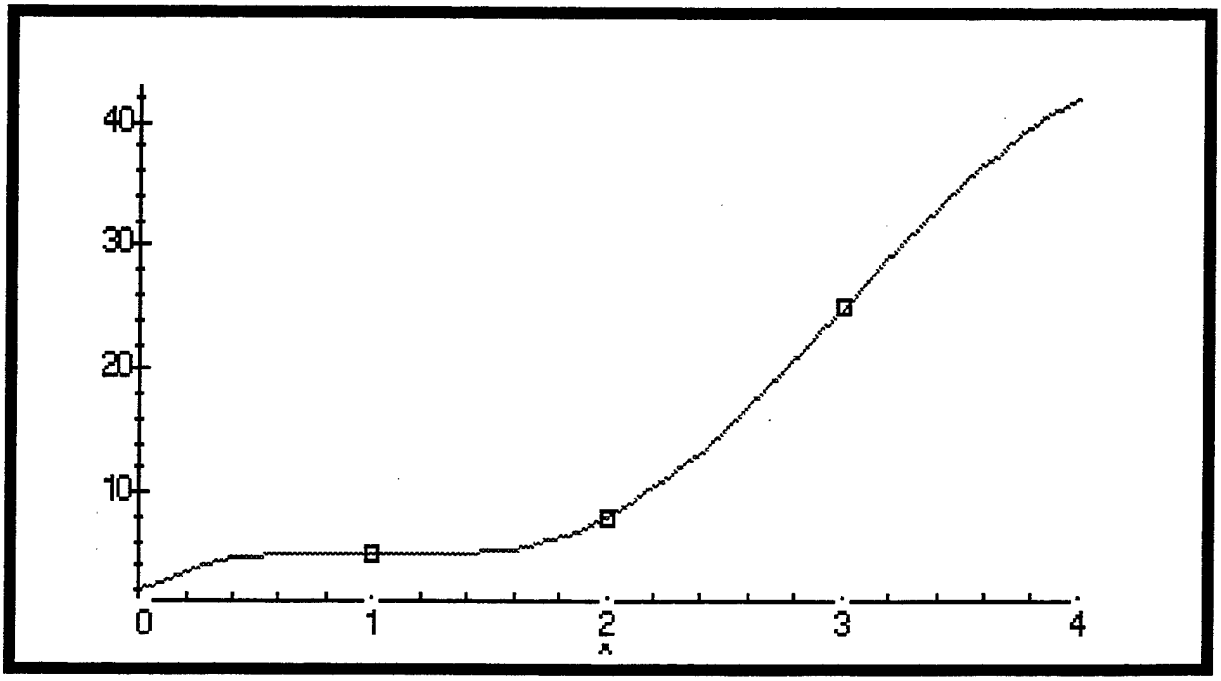

Figure 62. The Spline Plot.

The solution provides the polynomials which describe each segment of the curve. By using the spline procedure, the value of the spline, *s*, can be determined at any point on the curve defined by Table 63.

Interval	Model
$1 \le x \le 2$	$S_1(x) = 2 + 10x - 10.5x^2 + 3.5x^3$
$2 \le x \le 3$	$S_2(x) = 58 - 74x + 31.5x^2 - 3.5x^3$

Table 63. The Curves Of The Spline.

a. Example 5.4.3: The Cost Of The Postage Stamp

Between 1917 and 1996 the cost of the postage stamp has changed on over a dozen occasions. An analyst would like to be able to model the cost of the stamp and more importantly predict its future cost. By using the cubic spline, this section will present the modeling process to answer a real-world question. Table 64 presents the cost of the postage stamp at the beginning of every decade since 1920. Table 65 and Figure 63 display the commands and the results.

year	1920	1930	1940	1950	1960	1970	1980	1990
cost	2	2	3	3	4	6	15	25

Table 64. The Postage Stamp Data Points.

Commands
year:=[1920,1930,1940,1950,1960,1970,1980,1990]:
cost:=[2,2,3,3,4,6,15,25]:
readlib(spline):
spline(year,cost,x,cubic):
s:=unapply(",x):
yc:={seq([year[i],cost[i]],i=1..8)}:
plot1:=plot(s(x),x=1920..2010):
plot2:=plot(yc,style=point, symbol=box):
with(plots):
display({plot1,plot2});

Table 65. The Commands To Determine The Cost Of A Postage Stamp.

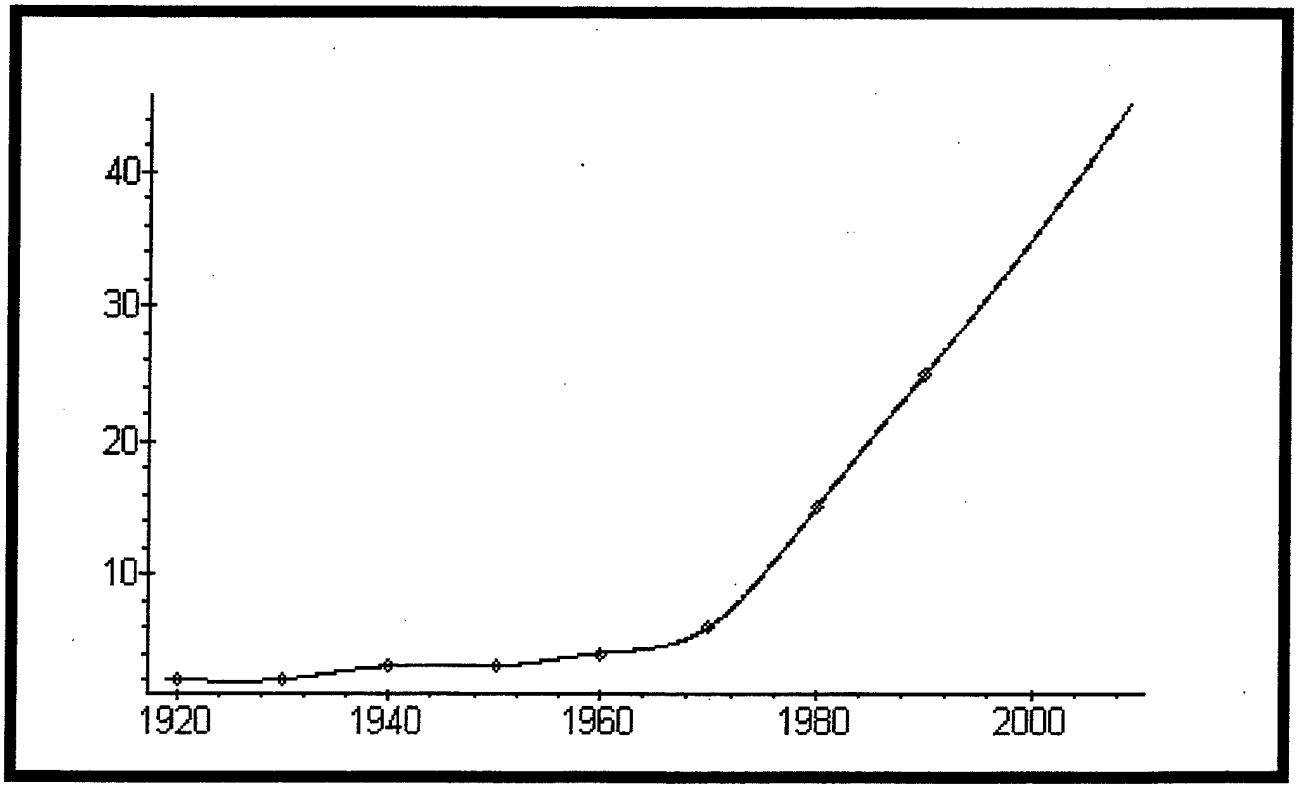

Figure 63. The Cost Of A Postage Stamp From 1920 To 2010.

Figure 63 displays the spline curve superimposed over the original data points. Estimations of the value of the function can be taken from the graph or by using the function *s* as shown in Table 66. Notice that by the year 2001 the spline approximates the cost of the postage stamp to be $0.36, a reasonable estimation. Note, the cubic spline is dependent on the data points, approximations outside of the endpoints can act erratically. An approximation of the cost of the postage stamp in 1871 is -$0.39, the government would pay each of us to use the mail system, an unlikely event.

Command	Output
s(1997);	32
s(2001);	36
s(2002);	37
s(2050);	132
s(1871);	-39

Table 66. The Approximation Of The Cost Of A Postage Stamp.

APPENDIX. MAPLE COMMANDS

A complete list of Maple commands used or described in this manuscript.

Data Entry Commands

:=	The assignment operator, used as a command <lhs> := <rhs>; where the lhs is the name and the rhs is the expression whose value the lhs assumes. ie) a:=17; a:=17 a:=x^2; $a:=x^2$ x:=8; x:=8 a; a:=64
:	statement separator, suppresses output
;	statement separator, displays output on screen
[ ]	list notation ie) a:=[2,5,19,4]; a:=[2,5,19,4] b:=a[2]; b:=5
{ }	set notation, orders data ie) a:={2,2,5,19,4} a:={2,4,5,19}
$	operator for forming an expression sequence ie) c:='i'$'i'=1..3 c:=1,2,3 c:='i^2'$'i'=1..3 c:=1,4,9
importdata	reads statistical data from a file importdata(filename,# of columns) ie) importdata(bass,2) reads bass file of 2 columns of data
readdata	reads raw data from a file readdata(filename, # of columns) ie) readdata(bass, 2) reads bass file of 2 columns of data

Package Control Commands

Digits — Defines the number of digits in the floats, how many significant digits will be used and presented in calculations. Default is ten digits.

```
ie)  Digits:=4;        Digits:=4
     evalf(1/2);       .5000
```

evalf( ); — evaluates using floating-point arithmetic, converts fractional solutions to decimal solutions.

```
ie)  evalf(1/2);       .5000000000
```

with( ): — defines the names of functions from a library package indicated inside the brackets.

```
ie)  with(linalg):     linalg package
     with(plots):      plots package
     with(stats):      stats package
```

Plotting Commands

plot(f(x), hr); — plots a function, f(x), over a horizontal range, hr

```
ie)  plot(sin(x),x=0..8);
```

plot(data); — plots data points

```
ie)  lenwt:=[17,25,12,18,10,15]:
     plot(lenwt,style=point);
```

seq — assigns independent x and y coordinates to a sequence of pairs, to assist in plotting data points

```
ie)  x:=[1,2,3,4,5]:
     y:=[8,5,2,1,0]:
     xy:={seq([x[i],y[i]],i =1..5)};
     plot(xy,style=point);
```

textplot — adds text into a plot

```
ie)  with(plots):
     t:=textplot(2,5,`Place this text at 2,5`):
     p:=plot(lenwt,style=point);
     display({t,p});
```

display — displays a list of plot structures

```
ie)  p1:=plot(lenwt,style=point):
     p2:= plot(sin(x),x=0..8):
     display({p1,p2});
```

Plotting Options

scaling=	CONSTRAINED or UNCONSTRAINED
axes=	FRAME, BOXED, NORMAL, or NONE.
coords=	polar
numpoints=	minimum number of points to be generated
resolution=	n
color=	specify the color of the curves
xtickmarks=	n
style=	POINT, LINE or PATCH
discont=	s
title=	`the title`
thickness=	0, 1, 2, or 3.
linestyle=	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
symbol=	BOX, CROSS, CIRCLE, POINT, or DIAMOND.
font=	Font for text objects in the plot
titlefont=	Font for the title of the plot
axesfont=	Font for the labels on the tick marks of the axes
labelfont=	Font for the labels on the axes of the plot
view=	[xmin..xmax,ymin..ymax]

Statistics/Linear Algebra

matadd — matrix addition, adds a multiple of one matrix to a multiple of another matrix

ie)
```
a:=matrix(3,2,[1,3,5,2,6,18]):
b:=matrix(3,2,[1,2,3,4,5,6]):
matadd(a,b,2,3);
```

$$\begin{bmatrix} 5 & 12 \\ 16 & 16 \\ 27 & 54 \end{bmatrix}$$

array — creates an array

ie) `array(1..4,[1,4,2,2]);` [1 4 2 2]

describe

A subpackage of the statistics package, required when using any of the following commands:

coefficientofvariation	count
countmissing	variance
covariance	decile
geometricmean	range
harmonicmean	kurtosis
linearcorrelation	mode
standarddeviation	mean
quadraticmean	median
meandeviation	moment
percentile	skewness
quartile	quantile

ie)
```
a:=[4,45,229]:
describe[count](a);     3
describe[median](a);    45
describe[kurtosis](a);  3/2
```

evalm( )

evaluates a matrix operation

ie)
```
a:=matrix(3,2,[1,3,5,2,6,18]):
b:=matrix(3,2,[1,2,3,4,5,6]):
```

evalm(a+2*b);

$$\begin{bmatrix} 3 & 7 \\ 11 & 10 \\ 16 & 30 \end{bmatrix}$$

fit(leastsquare)

fits a curve to data using the least-square method

ie)
```
xdata:=[1,2,3]:
ydata:=[4,45,229]:
xyfit:=fit[leastsquare[[x,y], y = a*x^2+b*x+c,
    {a,b,c}]] ([xdata,ydata]);
evalf(");                    y= 71.5x^2-173.5x+106
```

gaussjord

conducts Gauss-Jordan elimination on a matrix

ie)
```
a:=matrix(3,2,[1,3,5,2,6,18]):
```

gaussjord(a);

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

inverse — computes the inverse of a matrix

ie) a:=matrix(2,2,[1,3,5,2]):

inverse(a); $\begin{bmatrix} -.15 & .23 \\ .38 & -.08 \end{bmatrix}$

map — changes a variable into a function

ie) xdata:=[1,2,3]:

a:=map(x->x^2,xdata); a:=[1, 4, 9]

matrix — creates a two dimensional array

ie) a:=matrix(2,2,[1,3,5,2]); a:=$\begin{bmatrix} 1 & 3 \\ 5 & 2 \end{bmatrix}$

multiply — matrix-matrix multiplication

ie) a:=matrix(2,2,[1,3,5,2]):

b:=matrix(3,2,[1,2,3,4,5,6]):

multiply(b,a) $\begin{bmatrix} 11 & 7 \\ 23 & 17 \\ 35 & 27 \end{bmatrix}$

op — extracts operands from an expression

ie) a:=[8,32,2]:

op(2,a); 32

readlib( ) — read a library file of a specified name

ie) readlib(spline):

scalarmul — multiply a vector by an expression

ie) a:=[8,32,2]:

b:=scalarmul(a,2); [16 64 4]

spline — computes a natural spline

ie) a:=[1,2,4]:

b:=[17,4,44]:

spline(a,b,cubic); If($x<2$, $30-2x-16.5x^2+5.5x^3$, $96-101x+33x^2-2.75x^3$)

subsop — substitutes operands into an expression

ie) a:=[8,32,2]:

subsop(2=4,a); [8,4,2]

sum — calculates the sum of an expression

ie) a:=[4,45,229]:

sum('a[k]','k=1..3'); 278

unapply — returns an operator from an expression, assists in evaluating a least-squares fit following the fit command

ie)
```
xdata:=[1,2,3]:
ydata:=[4,45,229]:
xyfit:=fit[leastsquare[[x,y], y = a*x^2+b*x+c,
    {a,b,c}]] ([xdata,ydata]):
f:=unapply(rhs(xyfit),x):
f(1.8);                    25.36
```

Conditional and Repetitive Statements

for — a repetitive statement which will carry out a command while some condition is met

ie)
```
totalK:=0:
for K from 1 by 6 to 30 do
        totalK:=totalK + K:
od:
totalK;                          65
```

if — a conditional statement which carries out a command when a condition is met

ie)
```
soln:=0:
xdata:=3:
ydata:=5:
if xdata < ydata
        then soln:=xdata*ydata:
        else soln:=xdata/ydata:
fi;                              15
```

Algebraic Operations And Functions

a + b	addition
iquo(a,b)	quotient
a - b	subtraction
irem(a,b)	remainder
a * b	multiplication
isqrt(n)	square root
a / b	division
a ^ b	exponentiation
a ** b	exponentiation
signum(n)	sign of a number
n !	factorial
abs(n)	absolute value
min(a,b)	minimum
max(a,b)	maximum
abs	absolute value of real or complex argument
argument	argument of a complex number or expression
binomial	binomial coefficients: binomial(n,r) = n!/(r!*(n-r)!)
ceil	ceil(x) = smallest integer greater than or equal to x
exp	the exponential function: exp(x) = sum(x^i/i!,i=0..infinity)
factorial	the factorial function factorial(n) = n!
floor	floor(x) = greatest integer less than or equal to x
ln	natural logarithm (logarithm with base E = 2.71828...)
log	logarithm to arbitrary base
log10	log to the base 10
max, min	maximum/minimum of a list of real values
RootOf	function for expressing roots of algebraic expressions
round	round(x) = nearest integer to x (round(.5) = 1)
sqrt	square root
trunc	trunc(x) = nearest integer from x in the direction of 0

LIST OF REFERENCES

1. Giordano, F.R. and Weir, M.D., *A First Course in Mathematical Modeling*, Brooks/Cole Publishing Company, Belmont, California, 1985.

2. Fox, W.P., Giordano, F.R., Maddox, S.L, Weir, M.D., *Mathematical Modeling With Minitab*, Brooks/Cole Publishing Company, Belmont, California, 1987.

3. Char, B.W., *Maple V Library Reference Manual*, Waterloo Maple Software, Philadelphia, Pennsylvania, 1991.

4. Char, B.W., *Maple V Language Reference Manual*, Waterloo Maple Software, Philadelphia, Pennsylvania, 1991.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center 8725 John J. Kingman Road, Ste 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, California 93943-5101	2
3.	Maurice D. Weir, Code MA/Wc Naval Postgraduate School Monterey, California 93943-5216	2
4.	Richard Franke, Code MA/Fe Naval Postgraduate School Monterey, California 93943-5216	1
5.	Carlos Borges, Code MA/Bc Naval Postgraduate School Monterey, California 93943-5216	1
6.	Frank Giordano, Code MA/Giordano Naval Postgraduate School Monterey, California 93943-5216	1
7.	Gordon Beauchamp 647 Black Matt Rd. Douglassville, PA 19518	1
8.	Robert Beauchamp 253 Worden St. Portsmouth, RI 02871	2