GSNumerics

for the APPLE IIGS COMPUTER

from

The information in this manual is subject to change without notice. It does not represent a commitment on the part of Spring Branch Software Inc. The software described in this manual is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of Spring Branch Software Inc.

GSNumerics
TABLE OF CONTENTS

-- A --

Absolute value For any real number a: if **a** is greater than or equal to zero, the absolute value of **a** is **a**. If **a** is less than zero the absolute value of **a** is **-a**.
Actinium A radioactive natural element with atomic number 89.
Aluminum A silver, light metal. The natural element with atomic number 13.
Americium A natural element with atomic number 95.
amu See Atomic Mass Unit
antiderivative a function g, such that g'(x) is equal to f(x), is the antiderivative of f(x).
Antimony A silver-white brittle metal with atomic number 51.
Argon An inert, colorless, odorless gas with atomic number 18.
Arsenic A silvery-white, brittle, very poisonous element with atomic number 33.
Astatine The unstable element with atomic number 85.
Astronomical Unit A unit of length equal to the mean radius of the earth's orbit.
Atomic Number A number representing the relative position of a chemical in the periodic table.
Atomic Mass Unit 1/12 the mass of the Carbon C^{12} atom.
Atomic Weight A number representing the weight of one atom of an element, relative to some standard.

-- B --

Barium A silver-white metal with atomic number 56.
base e logarithm The base e log of a number is the power e would be raised to, if it were to equal the given number. This is referred as the natural logarithm of a number.
base ten logarithm The base 10 log of a number is the power ten would be raised to, if it were to equal the given number.
base two logarithm The base 2 log of a number is the power two would be raised to, if it were to equal the given number.
Berkelium The radioactive chemical element with atomic number 97.
Beryllium The hard, rare, metallic element with atomic number 4.
Best Fit The least square curve with the largest correlation coefficient (absolute value).
Bismuth A grayish-white, hard metallic element with atomic number 83.
Boron A non-metallic chemical element with atomic number 5.
Bromine The chemical element with atomic number 35.
BTU/h A measurement of work. British Thermal Units per hour.

-- C --

Cadmium A blue-white metallic element with atomic number 48.
cal/s See calories/second.
Calcium A soft, silver-white metallic element with atomic number 20.
Californium A radioactive chemical element with atomic number 98.
calories/second A measurement of work.
Carbon A non-metallic element found in all organic compounds. Atomic number is 6.
Celsius A temperature measurement system based on the freezing point of water being equal to zero degrees and the boiling point of water is 100 degrees.
Cerium A gray metallic element with atomic number 58.
Cesium A bluish-gray metallic element with atomic number 55.
Chlorine A greenish-yellow poisonous gaseous chemical element with atomic number 17.
Chromium A white, very hard, crystalline metal with atomic number 24.
cm2 See square meter
cm3 See cubic meter
Cobalt A hard steel-gray ductile metal with atomic number 27.
coefficient matrix The matrix consisting of the coefficients of the linear equations, making up a system of linear equations, whose simultaneous solution is to be found.
complex conjugate A complex number formed from another complex number by changing the sign of it's imaginary part. (i.e. a + ib and a - ib). Complex roots of polynomials will always occur in complex conjugate pairs.
complex number A number consisting of a real part plus an imaginary part, where the imaginary part represents some number multiplied by the square root of a minus one.
Copper A reddish-brown, metallic element with atomic number 29.

correlation coefficient A number returned by a least square regression, indicating the extent to which the generated curve, actually falls on the data. A correlation coefficient of -1 or 1, indicates perfect correlation.
cosecant A trig function equal to the hypotenuse divided by the opposite side.
cosine A trig function equal to the adjacent side divided by the hypotenuse.
cotangent A trig function equal to the opposite side divided by the adjacent side.
cube The result of multiplying a number by itself three times. (i.e. 2*2*2).
cube root The cube root of a number is the number that when cubed is equal to the number. Two is the cube root of eight (i.e. 2 * 2 * 2 = 8).
cubic feet A measurement of volume used in the FPS measurement system.
cubic inch A measurement of volume used in the FPS measurement system.
cubic meter A measurement of volume used in the MKS measurement system.
Curium A chemical element with atomic number 96.

-- D --

Differentiation The procedure used to find a function, whose value when solved for at a specific point, is equal to the slope of the original function at that point.
Dynamic Segments A program object code segment that may be loaded or unloaded from memory, by the memory manager.
Dysprosium A magnetic rare earth element with atomic number 66.

-- E --

Einsteinium A radioactive element with atomic number 99.
electric horsepower a measure of work in an electrical system.
Erbium A metallic rare earth element with atomic number 68.
Europium A rare earth element with atomic weight 63.
Exponential Curve Fit A least square fit to an exponential type curve.
exponential function The function used to take the base of the natural logarithms e to a power.
exponentiation The mathematical process of taking a number to a power.

-- F --

f-p/h See foot pounds/hour
Fahrenheit
feet2 A measure of area used in the FPS system of units.
feet3 A volume unit used in the FPS system of units.
Fermium A radioactive element with atomic number 100.
Fluorine A corrosive greenish-yellow element with atomic number 9.
foot pounds/hour A measure of work.
Francium A metallic element of the alkali group with atomic number 87.

-- G --

Gadolinium A rare earth element with atomic number 64.
Gallium A soft, bluish-white metallic element with atomic number 31.
gallon A volume measure unit used in the FPS system of units.
Germanium A grayish-white metallic element with atomic number 32.
Gold A heavy yellow metallic element with atomic number 79.
Graphing x-y Data The process where pictures of functions are produced by plotting points on a Cartesian Coordinate System.
GSCOL A small disk file that **GSNumerics** uses to keep user screen and graph color settings.

-- H --

Hafnium A metallic element with atomic number 72.
Helium A light, inert, colorless gas element with atomic number 2.
Holmium A metallic element with atomic number 67.
hp(e) See electrical horsepower.
hp(m) See mechanical horsepower.
Hydrogen A colorless, odorless, gaseous element with atomic number 1.

-- I --

Identity Matrix A square matrix, whose diagonal elements all equal one and whose off diagonal elements all equal zero.
in-fix A notation system where the operator is applied between two operands.
inch2 a measure of area in the FPS system of units.

inch3 a measure of volume in the FPS system of units.
Indium A soft, silver-white metallic element with atomic number 49.
Iodine A member of the halogen family with atomic number 53.
Iridium A white, heavy, brittle metallic element with atomic number 77.
Iron A white metallic chemical element with atomic number 26.
iteration The process of solving a problem by doing it over and over, each time coming a little closer to the actual solution.

-- K --

Kelvin A temperature scale measured in degrees centigrade from absolute zero, -273.15 degrees centigrade.
kilogram A measurement of mass in the MKS system of units.
kilometers A measurement of length in the MKS system of units A unit system based on meters (length), kilograms (mass) and seconds (time).
Krypton An inert gaseous element with atomic number 36.

-- L --

Lanthanum A silver metallic rare earth element with atomic number 57.
Lawrencium A radioactive element with atomic number 103.
Lead A heavy, soft, bluish-gray metallic element with atomic number 82.
Least Square A method of regression, minimizing the least square errors.
light year A unit of length used in astronomical measurements equalling the distance light will travel in one year.
Linear Curve A curve whose loci plot a straight line.
Linear regression A process of fitting mathematical curves to x-y data, with the objective of finding a mathematical function that will model the data points.
Lithium A silver-white, light metallic element with atomic number 3.
ln The natural logarithm based on the number **e**.
Log Curve Fit A least square fit to an log type curve.
log The logarithm based on the number 10.
ltwo The logarithm based on the number 2.
Lutetium A metallic, rare earth element with atomic number 71.

-- M --

m2 A measure of area in the MKS system of units.
m3 A measure of volume in the MKS system of units.
Magnesium A light, silver-white metallic element with atomic number 12.
Manganese A grayish-white metallic element with atomic number 25.
mechanical horsepower A measure of work done in a mechanical system.
Mendelevium A radioactive element with atomic number 101.
Mercury A heavy, silver-white metallic element with atomic number 80.
mile2 A measure of area in the FPS system of units.
Modulo division A division process where the integer remainder is returned.
Molybdenum A brittle, silver-white metallic element with atomic number 42.

-- N --

NAN See Not A Number.
NAN(001) Returned when an illegal argument has been passed to a hyperbolic trig function. For instance, ACOSH(0) returns NAN(001).
NAN(002) Returned when an illegal subtraction has been attempted. For example, INF - INF returns -NAN(001).
NAN(004) Returned when an illegal division is attempted. For instance, 0 divided by 0 returns NAN(004).
NAN(008) Returned when an illegal multiplication has been attempted. For instance, 0.0 * INF returns NAN(008).
NAN(034) Returned when an illegal argument has been passed to log function. For instance, log(-5) returns NAN(036).
NAN(036) Returned when an illegal argument has been passed to a hyperbolic trig function. For instance, ACOSH(0) returns NAN(001).
NAN(037) Returned when an illegal argument has been passed to an exponential function. For example, the square root of -4.0 returns NAN(037).
NAN(037) Returned when an illegal argument has been passed to an exponential function. For instance, the square root of -4.0 returns NAN(037).

NAN(061) Returned when an argument that is less than zero is passed to the Factorial Function. For instance, the Factorial of -4.0 returns NAN(061).
NAN(062) Returned when a non-integer argument is passed to the Factorial Function. For instance, the Factorial of 4.56 returns NAN(062).
NAN(063) Returned when an argument that is greater than 1e+12 is passed to **DHMS, HMSD, HADD OR HSUB**. For instance, **DHMS**(1.34e+14) returns NAN(063).
natural logarithm The natural logarithm based on the number **e**.
Neodymium A metallic rare earth element with atomic number 60.
Neon A colorless, odorless, inert gaseous element with atomic number 10.
Neptunium The element with atomic number 93.
Nickel A hard, silver-white metallic element with atomic number 28.
Niobium A metallic element with atomic number 41.
Nitrogen A orderless, colorless gaseous element with atomic number 7.
Nobelium A radioactive element with atomic number 102.
Not A Number The code returned by the program when an illegal math operation has been performed. Stands for Not A Number. A code is returned with the NAN code, indicating the type of math error involved. I.E. NAN(001).

-- O --

Osmium A bluish-white metallic element with atomic number 76.
Oxygen A colorless, odorless, gaseous element with atomic number 8.

-- P --

Palladium A silver-white, metallic element with atomic number 46.
parsec A unit of length measurement used in astronomical calculations. It is approximately equal to 3.26 light years.
parser A program that breaks down, or tears apart user input functions, and solves them according to the rules of mathematics.
Phosphorus A nonmetallic element with atomic number 15.
Platinum A silver-grey metallic element with atomic number 78.
Plutonium A radioactive element with atomic number 94.
polar form A method of expressing a complex number is terms of a magnitude and an angle.
Polonium A radioactive element with atomic number 84.
polynomial A function based on integer powers of x, with real coefficients.
post-fix entry see Reverse Polish Notation.
Potassium A soft, silver-white metallic element with atomic number 19.
Power Curve Fit A least square fit to an power type curve.
Praseodymium A metallic, rare earth element with atomic number 59.
Promethium A metallic, rare earth element with atomic number 61.
Protactinium The radioactive element with atomic number 91.

-- R --

Radium A radioactive, metallic element with atomic number 88.
Radon A radioactive, gaseous element with atomic number 86.
Random function A function provided with the program to provide pseudo random numbers for the user.
rectangular form A method of expressing a complex number where it is written in two parts, a real part and an imaginary part.
Regression A procedure to find a mathematical function that fits a set of x-y data points.
Reverse Polish Notation The method of problem entry used in GSNumerics. The operands are first entered, and then the operator is entered. Also called RPN or post-fix entry.
Rhenium A metallic element with atomic number 75.
Rhodium A hard, gray-white element with atomic number 45.
Romberg method A high order method of numerical integration, normally yielding better results than the lower order approximations.
root of a function The points where a function is exactly equal to zero is a root of the function.
Rubidium A soft, silver-white element with atomic number 37.
Ruthenium A hard, brittle, silver-gray metallic element with atomic number 44.

-- S --

Samarium A metallic, rare earth element with atomic number 62.
SANE The Standard Apple Numerics Environment. A very accurate, high level set of math tools available for use with the Apple IIGS.
Scandium A rare, metallic element with atomic number 21.
secant The trig function represented by the hypotenuse divided by the adjacent side.
Selenium An element of the sulphur group with atomic number 34.
Silicon A non-metallic element with atomic number 14.
Silver A white, precious, metallic element with atomic number 47.
Simpson method A semi-sophisticated method of calculating the area under the curve of a function.
sine A trig function equal to the opposite side divided by the hypotenuse.
slope An measure of how the value of a function is changing at a certain point.
Sodium A silver-white, metallic element with atomic number 11.
square The process of multiplying a number by itself.
square centimeters A measurement of area used in the metric system.
square matrix A matrix that has the same number of rows that it does columns.
square meter A measurement of area used in the metric system.
square mile A measurement of area used in the English system.
square root The square root of a number is that number that is equal to the first number, when it is multiplied by itself. (i.e. 2 * 2 = 4). Two is the square root of four.
Strontium A pale-yellow, metallic element with atomic number 38.
Sulfur A pale-yellow non-metallic element with atomic number 16.
symmetric matrix A matrix whose off diagonal terms $A_{i,j} = A_{j,i}$.

-- T --

tangent The trig function represented by dividing the opposite side by the adjacent side.
Tantalum A rare, steel-blue element with atomic number 73.
Technetium A metallic element with atomic number 43.
Tellurium A rare, non-metallic element with atomic number 52.
Terbium A metallic, rare earth element with atomic number 65.
Thallium A rare, bluish-white metallic element with atomic number 81.
Thorium A rare, grayish, radioactive element with atomic number 90.
Thulium A metallic, rare earth element with atomic number 69.
Tin A soft, silver-white metallic element with atomic number 50.
Titanium A dark-gray, lustrous metallic element with atomic number 22.
Trapezoid method An elementary method of integration, where the area is broken into trapezoids, and they are summed.
Tungsten A metallic element with atomic number 74.

-- U --

Unnihexium A transuranic element with atomic number 106.
Unnilpentium A transuranic element with atomic number 105.
Unnilquadium A transuranic element with atomic number 104.
Unnilseptium A transuranic element with atomic number 107.
Uranium A hard, heavy, radioactive, metallic element with atomic number 92.

-- V --

Vanadium A rare, silver-white, metallic element with atomic number 23.

-- X --

Xenon A heavy, colorless, inert gaseous element with atomic number 54.

-- Y --

Ytterbium A rare, metallic element with atomic number 70.
Yttrium A rare, metallic element with atomic number 39.

-- Z --

Zinc A bluish-white, metallic element with atomic weight 30.
Zirconium A gray metallic element with atomic number 40.

GETTING STARTED

Welcome to **GSNumerics**. Spring Branch Software, Inc. is hopeful that you will gain many benefits from the program and enjoy using it. We suggest you read this chapter carefully, before trying to use the program. It will give you some helpful hints on how to get the maximum benefit from **GSNumerics**.

This chapter gives instructions for running the **GSNumerics** program on 3.5 inch disk and hard disk **Apple IIgs™** systems. **GSNumerics** is designed to run under **GSOS** version 5 with **Express Load**. The program will run under **GSOS 5.0**, but should be run under **GSOS 5.0.2**, for best results. If you don't have **GSOS 5.0.2**, contact your Apple Dealer, who will provide you with a copy.

What You Should Know

To operate the program, the user should be familiar with the operation of the **Apple IIgs™** computer. Before starting you should understand the following concepts:

1. Using the mouse to point, drag, click and double-click.
2. How to start and quit applications from the desktop.
3. How to initialize and copy disks.
4. How to pull down menus and choose menu commands
5. How to use scroll bars, radio buttons, check mark buttons and regular buttons.
6. How to use the cut (**Ó-X**), copy (**Ó-C**) and paste (**Ó-V**) commands.
7. How to use the Standard File dialogs for saving and recalling disk files.

These concepts are explained in the manuals that came with your **Apple IIgs™** and **GSOS System Disk**.

Hardware Required

You will need an **Apple IIgs™** computer with at least one meg of ram memory and at least one 800k disk drive to run **GSNumerics**.

GSNumerics may not run on a system with one meg of ram, if any of the following conditions exist:

1. The user has allocated a large RAM disk.
2. Desk Accessories that use a significant portion of the ram memory are installed.
3. The user has allocated a large ram memory for disk cache operations.

If you have trouble running the program, make sure you have removed Desk Accessories and de-allocated any cache or ram disk memory. You do not need to remove the Control Panel Desk Accessory.

Backup Your GSNumerics Disk

Never work directly with the original **GSNumerics** disk shipped with the program. If you are not working from a hard disk, make a properly labeled backup copy for day to day use. If the backup copy becomes damaged, you can make another from the original. Make sure the original is stored in a safe place.

Operating With One 3.5 Inch Drive

Insert the **GSOS System** disk into the drive and boot the computer. After the computer is booted and the desktop appears, insert the **GSNumerics** backup disk and double click on the **GSNumerics** icon. After a short period of time you will get the following message:

"Please insert the disk: SYSTEM.DISK"

Remove the **GSNumerics** backup disk and insert the **GSOS System** disk, and press the return button. The program will then load the required system tools. After the tools are loaded, you will see the following message:

"Please insert the disk: GSNUMERICS"

Insert the **GSNumerics** backup disk and click on the **OK** button. You are now ready to use the program. If you have limited memory, it will be necessary for you to leave the **GSNumerics** backup diskette in the drive while operating the program. See the section on **Dynamic Segments**, in this chapter, for an explanation of this feature.

Operating With Two 3.5 Inch Drives

Insert the **GSOS System** disk into the boot drive and the **GSNumerics** backup disk in the other drive. Boot the computer and then start the **GSNumerics** program by double clicking on the **GSNumerics** icon.

If you have limited memory, it will be necessary for you to leave the **GSNumerics** backup diskette in the drive while operating the program. See the section on **Dynamic Segments**, in this chapter, for an explanation of this feature.

Hard Disk Installation and Operation

To install **GSNumerics** on a hard disk, create a folder "**GSNumerics**" on the hard disk. After creating the folder, copy the entire contents of the original **GSNumerics** disk into the "**GSNumerics**" folder on the hard disk. To run the program, open the "**GSNumerics**" folder and double click on the **GSNumerics** icon.

Dynamic Segments

GSNumerics is a large program and has been created with "dynamic" segments to allow it to be run with only one meg of ram memory. The segments are called dynamic, because the individual program segments are loaded into ram memory, only when the program needs them. After the program no longer needs a particular program segment, the Memory Manager may remove the segment from memory, to allow memory space for another required program segment. This ability to bring into memory only those program parts required and to unload them from memory when they are not needed, is called dynamic segmentation.

When you run **GSNumerics** on systems with minimum memory, you will notice the segments being read in from the disk, as you select new program options. The time required to load the segments is not objectionable from a user standpoint. In fact, it allows you to run the program without investing in additional ram memory. Since the program must have access to the disk, in order to load segments when needed, you may need to leave the **GSNumerics** backup disk in the drive . If you remove the disk and the program needs to load a segment from it, it will ask you to insert the **GSNumerics** disk.

If your system contains 1.25 meg ram or more, the program will attempt to bring in all segments during the initial program load. If sufficient memory is available to load the entire program, the dynamic segment disk reads are not needed and you can remove the **GSNumerics** backup disk.

Printer Operation

To conserve memory on systems having less that 1.25 meg of memory, the program uses a slightly different printing technique on 1.0 meg systems, than it does on systems having 1.25 meg of memory or greater. If you have limited memory, you will still be able to print the screens and graphs in in either color or black and white. With limited memory you will not be able to select other printing options. The **Print Style** and **Print Job** dialogs will not appear when selecting a screen print, on systems with only 1.0 meg of ram memory.

Demonstration Files

The disk containing the **GSNumerics** program also contains other files. Eight of these files can be used to demonstrate and help the user understand the program. A ninth file is used by the program to set the user screen colors. The demo files on the disk are:

1. **Session.demo** Demonstrates a complete session file save.
2. **Poly1.demo** A sixth order polynomial, with real roots, for polynomial operations.
3. **Poly2.demo** A fourth order polynomial,with real and complex roots, for polynomial operations.
4. **xydata.demo** x-y data file for regression demonstration.
5. **matrix.demo** A matrix session file for matrix operations.
6. **linsys1.demo** 5 x 5 system of real equations for linear systems.
7. **linsys2.demo** 3 x 3 complex system of equations for linear systems.
8. **function.demo** A file of sample user defined functions.

System Speed

GSNumerics does some highly complex mathematical calculations. These calculations are done using the **Standard Apple Numerics Environment (SANE)**. To insure accurate results, extended 80 bit storage is used for all **GSNumerics** floating point variables. This insures the results of lengthy calculations will have minimum errors caused by machine rounding and truncation. The trade-off in using the highly accurate extended variables is some loss of speed. We believe the slower speed is more than compensated for by the very accurate answers produced by the program. These calculations can take some time when computing the solutions to matrices, roots of polynomials, solving systems of equations or graphing functions.

You may find the investment in an accelerator board to be beneficial. The program has been extensively tested using the Applied Engineering **Transwarp GS™** accelerator. The **Transwarp GS™** will dramatically shorten the time involved in many calculations done by the program. We believe **Transwarp GS™** is a good companion to **GSNumerics**.

Product Support

Spring Branch Software, Inc. is committed to providing service to you, our valued customer. It is our intention to help you get the greatest benefits possible from the program. You may contact us for support, if you have questions about using **GSNumerics**. It will be very helpful if you will be prepared prior to contacting us for product support. Before you call, make sure the you have done the following:

1. Read the manual and understand how to use the program.
2. Have the phone near the computer, with the program running.
3. Know the **GSOS** version, **GSNumerics** version and hardware configuration.
4. If you think you have found a bug, detail the steps to produce the bug.

Your comments and suggestions will be appreciated. Product improvements can and will result from our customer suggestions. You may address your comments to:

Spring Branch Software, Inc.
Route #2, Box 268A
Manchester, IA 52057
1-319-927-6537

Thank you for purchasing **GSNumerics**.

THE MAIN SCREEN

When you first open **GSNumerics** the program copyright notice will be shown on the screen. Keying the mouse in the **OK** button on the dialog box or pressing **return** will cause the copyright dialog to go away and will show the first **GSNumerics** screen. This is the main program screen and we will refer to it as the **Scientific Calculator** screen. This screen is used to control other operations of the program. The **Scientific Calculator** screen is shown in Figure 2.1.

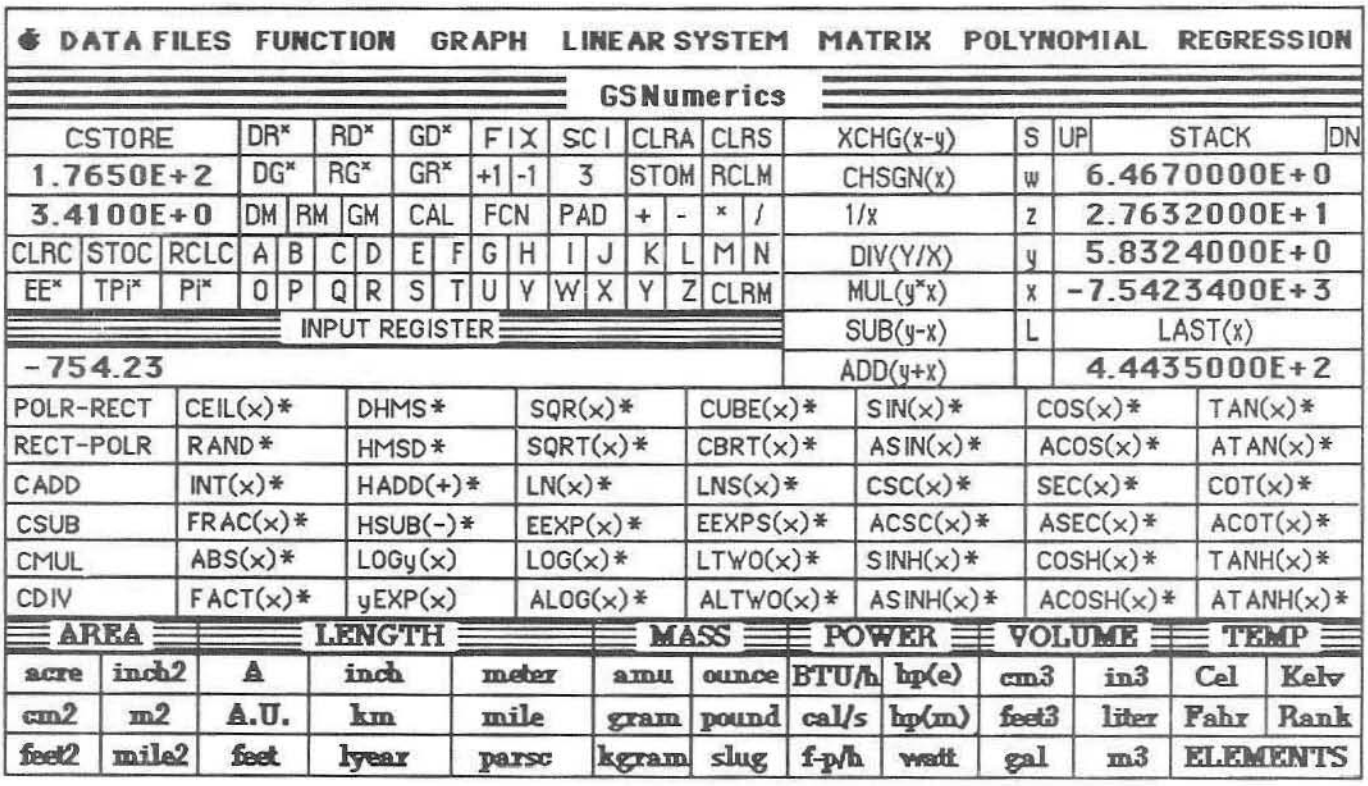

Figure 2.1 Scientific Calculator Screen

The **Scientific Calculator** screen is divided into nineteen rows. The first row is the **Menu Bar** and the second row is the **Title Bar**. The remaining seventeen rows can be divided into three sections, we will refer to as the **Register-Memory** section, **Function** section and the **Conversion-Elements** section .

The **Register-Memory** section contains the buttons that control calculator memory operations, the seven number registers and other buttons, that will be explained later.

The **Function** section has forty-eight buttons for performing different types of math functions.

The **Conversion-Elements** section contains six title bars, thirty-seven conversion buttons and one mode button,when in the **Conversion** mode. The title bars label six different sections used to convert between different unit systems. This section will contain one title bar, one hundred and seven chemical symbol buttons and a mode button, when in the **Element** mode. The user may return either the value of the Atomic Weight of a chemical element or the Atomic Number, when in the **Element** mode. The user may quickly switch back and forth between the **Conversion** mode and the **Element** mode.

Register-Memory Section

The **Register-Memory** section can be more easily understood if it is broken down into nine different areas. An exploded schematic view of the **Register-Memory** section is shown in Figure 2.2. This Figure shows the nine areas and the items each area contains. The area name is written under each area in bold, block letters.

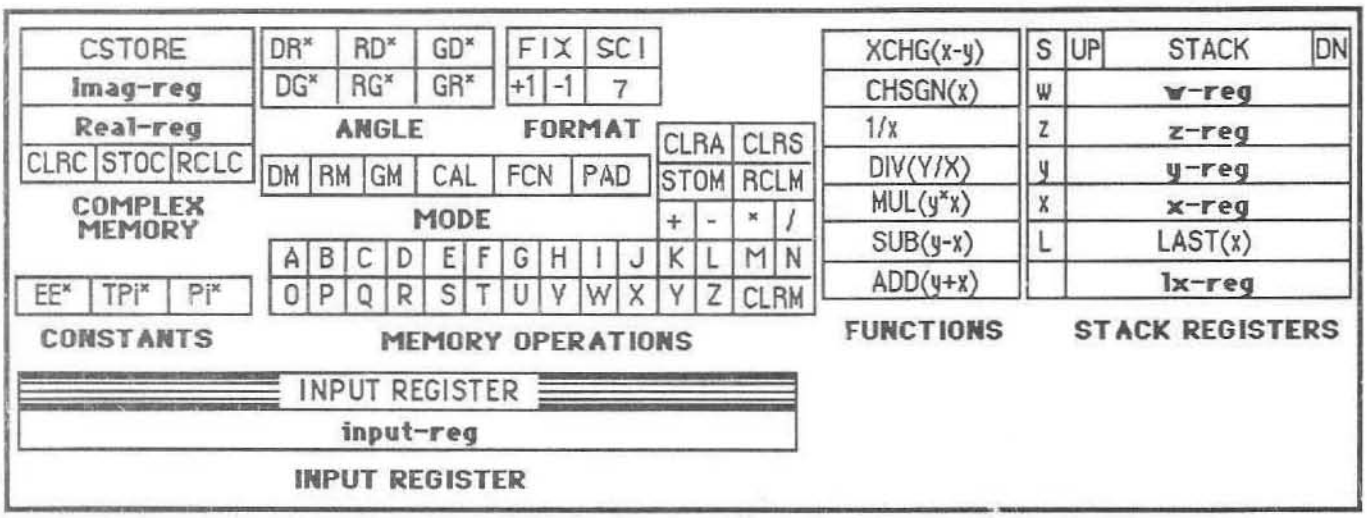

Figure 2.2 Register/Memory Section

FUNCTIONS

The FUNCTIONS area contains seven buttons for performing basic mathematical operations. These buttons are used to exchange the **x-reg** and **y-reg** registers [XCHG(x-y)], change the sign of the **x-reg** [CHSGN(x)], take the reciprocal of the **x-reg** [1/x], divide the **x-reg** into the **y-reg** [DIV(y/x)], multiply the x register by the **y-reg** [MUL(x*y)], subtract the **x-reg** from the **y-reg** [SUB(y-x)], and to add the **x-reg** and **y-reg** [ADD(y+x)]. Double-clicking on the top three buttons, will cause complex number operations to take place. These complex number operations will explained later in this manual. The seven function buttons are brown on a color monitor and dark gray on a monochrome monitor.

INPUT REGISTER

The INPUT REGISTER is where all numbers and formulas are entered when solving problems. The answers to a calculation, shown in the format selected by the user, are shown here after a calculation is completed. Numbers are entered with the number keys, followed by pressing **Return.** We will always refer to this as the **input-reg**. The number in the input register will always be the same number that is in the **x-reg**, immediately after a calculation is initiated. The **input-reg** and **x-reg** numbers will be different when a new number is being input into the calculator.

STACK REGISTER

The STACK REGISTER has four registers for displaying a number input into the calculator or returning the result of calculations. These four registers are the **x-reg, y-reg, z-reg** and **w-reg**. A fifth register will contain the **last x** value. This is the number that was in the **x-reg** prior to entering a new number into the **input-reg** or doing a calculation. There are eight buttons in this section that are used to move numbers around in the stack or to recall any stack register to the **x-reg**. These buttons are used to: roll the stack up [UP], roll the stack down [DN], recall the last x to the **x-reg** [L], recall the **w-reg** to the **x-reg** [w], recall the **z-reg** to the **x-reg** [z], recall the **y-reg** to the **x-reg** [y], recall the **x-reg** to the **x-reg** [x], and to recall the last stack to the stack **x-y-z-w** registers [S]. When doing complex operations, double-clicking will initiate complex operations on the [UP] and [DN] buttons. The buttons in this section are blue on a color monitor.

FORMAT

The FORMAT area contains four brown buttons used to set the format of numbers written to the screen. The format can be set to fixed [FIX] or it can be set to scientific [SCI]. The number of digits after the decimal place can be set by incrementing [+1] or by decrementing [-1]. The number in the decimal place register will change to indicate the number of digits that are shown after the decimal place. This will only change the **input-reg** on the **Scientific Calculator** screen. It also changes the format of program dialog boxes that are explained in later chapters. Some numbers are too large to be expressed in fixed format. The program will automatically convert those numbers to scientific format, before displaying them.

COMPLEX MEMORY

The COMPLEX MEMORY area contains three buttons and two registers. The **real-reg** is used to store the real part and the **imag-reg** is used to store the imaginary part of complex numbers expressed in rectangular format. [CLRC] clears the complex stack, [STOC] stores numbers from the **x-y-z-w** registers into the complex stack and [RCLC] recalls numbers to the **x-y-z-w** registers. Numbers are always entered into the program in rectangular format. The complex stack buttons are brown on a color monitor.

MEMORY OPERATIONS

The MEMORY OPERATIONS area contains blue buttons that control the storage and recall of numbers. Three of the buttons are used to clear memory, twenty-six buttons represent the memory storage locations "**a**" through "**z**", four are to control memory math and two buttons initiate the storage or recall operations. The function of these buttons is to select memory locations "**a**" to "**z**" [A]-[Z], clear all memory locations [CLRA], clear stack memory [CLRS] clear memory "**a**"-"**z**" [CLRM], memory add [+], memory subtract [-], memory multiply [*], memory divide [/] ,memory recall [RCLM] and memory storage [STOM].

ANGLE

The ANGLE area contains six buttons used to convert angles in the **x-reg** between the three types of angular measurement degrees, radians and grads. Buttons used for converting degrees to radians and grads are [DR*] and [DG*]. The buttons used to convert radians are [RD*] and [RG*]. The [GD*] and [GR*] buttons convert from grads to radians and degrees. The angle buttons are brown on a color monitor.

MODE

The MODE area contains six buttons used to control the operating mode of the calculator. Three buttons [DM] , [RM] and [GD*], control the **trig mode** (interpretation of angles input to the trigonometry functions). The three methods of angle input are degrees, radians and grads. Three of the buttons control whether the calculator is in **Scientific Calculator** mode [CAL], **Calculator Function** mode [FCN] or the **Ten Key Calculator** mode [PAD].

CONSTANTS

The CONSTANTS area contains three buttons used to recall three commonly used constants to the **x-reg**, for use in calculations. The three constants are: pi [Pi*] , two pi [TPi*] and the base of the natural logarithms (e) [EE*]. The constant buttons are brown on a color monitor.

Function Section

We will now look at the second second of the scientific calculator, the **CALCULATOR FUNCTION SECTION.** This section contains forty-eight buttons that are use to initiate different functions used with the calculator. These buttons can be subdivided into four basic groups, as show in the schematic drawing of this section if Figure 2.3. The groups contain similar functions and are color coded to enable the user to quickly find the needed function quickly.

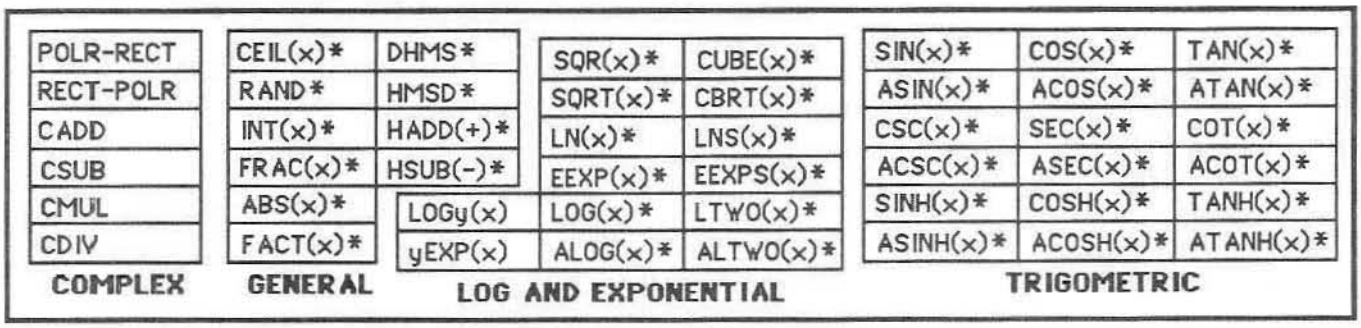

Figure 2.3 Calculator Functions Section

LOG AND EXPONENTIAL

The LOG AND EXPONENTIAL area contains fourteen buttons. Log calculations are available for the natural logarithms **LN(x)***, **LNS(x)***, **EEXP(x)*** and **EEXPS(x)***, for the decimal base ten **LOG(x)*** and **ALOG(x)*** or the base two **LTWO(x)*** and **ALTWO(x)***. Other base calculations can be done using the **LOGy(x)** and **yEXP(x)** functions. Buttons are provided for squaring **SQR(x)***, square roots **SQRT(x)***, cubing **CUBE(x)***, or taking the cube root of a number **CBRT(x)***. This section is brown on a color monitor.

COMPLEX

The COMPLEX area contains six buttons for performing complex number mathematics. They are **POLR-RECT**, **RECT-POLR**, **CADD**, **CSUB**, **CMUL** and **CDIV**. The complex buttons are brown on a color monitor.

GENERAL

The GENERAL area contains ten buttons: **CEIL(x)***, **RAND***, **INT(x)***, **FRAC(x)***, **ABS(x)***, **FACT(x)***, **DHMS***, **HMSD***, **HADD(+)*** and **HSUB(-)***. The general buttons are blue.

TRIGONOMETRY

The TRIGONOMETRY area contains eighteen buttons used in computing trigonometry calculations involving angles. The six basic trig functions sine [SIN(x)*], cosine [COS(x)*], tangent [TAN(x)*], cosecant [CSC(x)*], secant [SEC(x)*] and cotangent [COT(x)*] are available. The inverse trig functions sine^{-1} [ASIN(x)*], cosine^{-1} [ACOS(x)*], tangent^{-1} [ATAN(x)*], cosecant^{-1} [ACSC(x)*], secant^{-1} [ASEC(x)*] and cotangent^{-1} [ACOT(x)*] are provided. Included are the hyperbolic trig functions [SINH(x)*], [COSH(x)*] and [TANH(x)*], along with their inverse functions sinh^{-1} [ASINH(x)*], cosh^{-1} [ACOSH(x)*] and the tanh^{-1} [ATANH(x)*]. This area is brown on a color monitor.

Conversion-Elements Section

This section is a really two sections, depending on whether the user has selected the conversion or the elements mode. The mode of this section can be switched quickly and easily by the user, by merely keying the mouse in the conversion mode box in the bottom right corner of the section. There are five different types of conversions available. They are Area Conversions, Length Conversions, Mass Conversions, Power Conversions, Volume Conversions and Temperature Conversions.

We will start with the conversion section that is shown schematically in Figure 2.4. This section allows the user to convert numbers between different units of measurement. The conversion will always operate on the value in the x-reg.

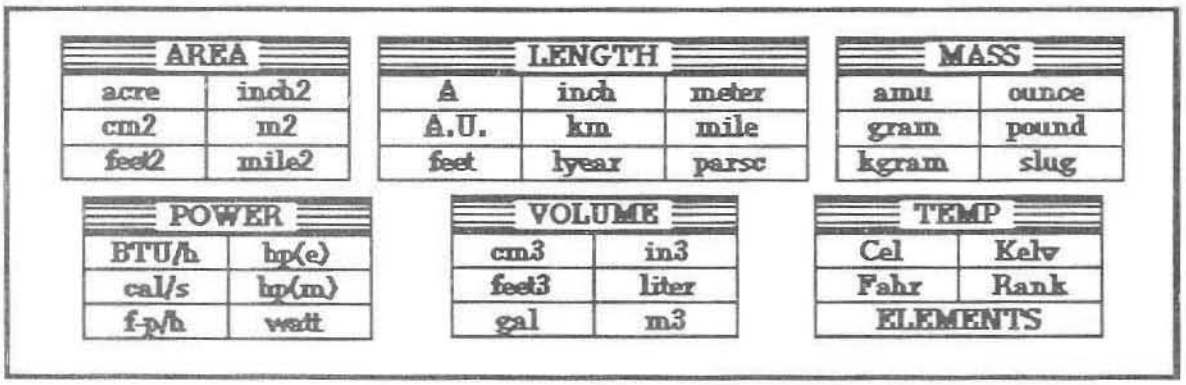

Figure 2.4 Conversion Section

AREA

The AREA conversions allow the user to convert rapidly between acres [acre], square inches [inch2], square centimeters [cm2], square meters [m2], square feet [feet2], and square miles. [mile2] This area is blue on a color monitor.

LENGTH

The LENGTH conversions allow the user to convert rapidly between angstroms [A], inches [inch], meters [meter], astronomical units [A.U.], kilometers [km], miles [mile], feet [feet], light years [lyear], and parsecs [parsc]. The length conversion area is orange on a color monitor

MASS

The MASS conversions allow the user to convert between atomic mass units [amu], ounces [ounce], grams [gram], pounds [pound], kilograms [kgram], and slugs [slug]. The mass conversion area is blue on a color monitor

POWER

The POWER conversions allow the user to convert between B.T.U./h [BTU/h], electric horsepower [hp(e)], calories per second [cal/s], mechanical horsepower [hp(m)], foot-pounds per hour [f-p/h], and watts [watt]. The power conversion area is orange on a color monitor

VOLUME

The VOLUME conversions allow the user to convert rapidly between cubic centimeters [cm3], cubic inches [in3], cubic feet [feet3], liters [liter], gallons [gal], and cubic meters [m3]. The power conversion area is blue on a color monitor

TEMPERATURE

The TEMPERATURE conversions allow the user to convert between Celsius [Cel], Kelvin [Kelv], Fahrenheit [Fahr], and Rankin [Rank]. The conversion screen can be changed to the Elements mode with [ELEMENTS]. The power conversion area is orange on a color monitor

When the user is in the Elements mode a chart of the natural chemical elements is drawn on the screen as shown in Figure 2.5. This screen is used to return either the atomic weight or atomic number of a chemical element to the x-reg.

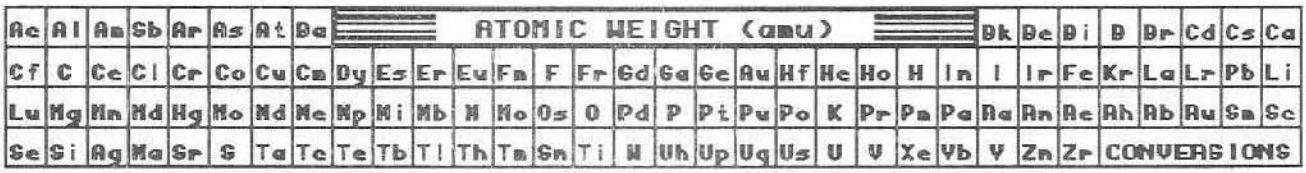

Figure 2.5 Chemical Elements

ELEMENTS

The ELEMENTS screen has one button for each of the chemical elements. Releasing the mouse on an element button will place the Atomic Weight in Atomic Mass Units into the **x-reg.** Double-clicking on an button will return the Atomic Number to the **x-reg.** CONVERSIONS returns to the conversion mode. The elements buttons are arranged in alphabetical order by name

ACTINIUM	Ac	ALUMINUM	Al	AMERICIUM	Am
ANTIMONY	Sb	ARGON	Ar	ARSENIC	As
ASTATINE	At	BARIUM	Ba	BERKELIUM	Bk
BERYLLIUM	Be	BISMUTH	Bi	BORON	B
BROMINE	Br	CADMIUM	Cd	CESIUM	Cs
CALCIUM	Ca	CALIFORNIUM	Cf	CARBON	C
CERIUM	Ce	CHLORINE	Cl	CHROMIUM	Cr
COBALT	Co	COPPER	Cu	CURIUM	Cm
DYSPROSIUM	Dy	EINSTEINIUM	Es	ERBIUM	Er
EUROPIUM	Eu	FERMIUM	Fm	FLUORINE	F
FRANCIUM	Fr	GADOLINIUM	Gd	GALLIUM	Ga
GERMANIUM	Ge	GOLD	Au	HAFNIUM	Hf
HELIUM	He	HOLMIUM	Ho	HYDROGEN	H
INDIUM	In	IODINE	I	IRIDIUM	Ir
IRON	Fe	KRYPTON	Kr	LANTHANUM	La
LAWRENCIUM	Lr	LEAD	Pb	LITHIUM	Li
LUTETIUM	Lu	MAGNESIUM	Mg	MANGANESE	Mn
MENDELEVIUM	Md	MERCURY	Hg	MOLYBDENUM	Mo
NEODYMIUM	Nd	NEON	Ne	NEPTUNIUM	Np
NICKEL	Ni	NIOBIUM	Nb	NITROGEN	N
NOBELIUM	No	OSMIUM	Os	OXYGEN	O
PALLADIUM	Pd	PHOSPHORUS	P	PLATINUM	Pt
PLUTONIUM	Pu	POLONIUM	Po	POTASSIUM	K
PRASEODYMIUM	Pr	PROMETHIUM	Pm	PROTACTINIUM	Pa
RADIUM	Ra	RADON	Rn	RHENIUM	Re
RHODIUM	Rh	RUBIDIUM	Rb	RUTHENIUM	Ru
SAMARIUM	Sm	SCANDIUM	Sc	SELENIUM	Se
SILICON	Si	SILVER	Ag	SODIUM	Na
STRONTIUM	Sr	SULFUR	S	TANTALUM	Ta

TECHNETIUM	Tc	TELLURIUM	Te	TERBIUM	Tb
THALLIUM	Tl	THORIUM	Th	THULIUM	Tm
TIN	Sn	TITANIUM	Ti	TUNGSTEN	W
UNNIHEXIUM	Uh	UNNILPENTIUM	Up	UNNILQUADIUM	Uq
UNNILSEPTIUM	Us	URANIUM	U	VANADIUM	V
XENON	Xe	YTTERBIUM	Yb	YTTRIUM	Y
ZINC	Zn	ZIRCONIUM	Zr		

Menus And Menu Items

The **Apple Menu** shown in Figure 2.6. This menu is used to control screen color, reset the Random Number, draw the **GSNumerics** copyright notice and to draw the System Control Panel.

ó	
About GSNumerics ...óA	Draws the **GSNumerics** copyright dialog on the screen.
Set Screen Colors ...óS	Draws dialog allowing user to set calculator screen colors.
Set Graph Colors ...óO	Draws dialog allowing user to set colors for graphs.
Seed Random...	Draws dialog used to seed random number generator.
Control Panel	Draws system control panel (GSOS 5.0 and later only)

Figure 2.7 Apple Menu

The second menu is the **FILES MENU** shown in Figure 2.8. This menu contains the cut, copy and paste items, Session file recall and Save and Print items. If your system contains less than 1.25 meg of ram, this menu will be slightly different, but you will still be able to print the screen.

FILES	
Cut ... óX	Cut selection and put in scrap.
Copy ... óC	Copy selection and put in scrap
Paste ... óV	Insert scrap contents at selection point
Load Session ...	Initiates loading a previous Session disk file into the program.
Save Session ...	Initiates saving the current Session to a disk file.
Page Setup ...	Draws print page setup dialog (1.25 meg ram and above).
Print Screen ...	Prints screen (systems with 1 meg ram only).
Quit ... óQ	Quits GSNumerics and returns to GSOS desktop.

Figure 2.8 FILES menu

The **FUNCTION MENU** shown in Figure 2.9 is used for non-polynomial function operations. You may enter a new current function, compute the solution for a given x, compute the slope at a given point, find the real roots of a function and compute the area under the function curve, using these menu items.

FUNCTION	
Area ...	Draws Function Area dialog.
Roots ...	Draws Function Roots dialog.
Slope ...	Draws Function Slope dialog.
Solve ...	Draws Solve Function dialog.
Enter Function ... ⌘F	Draws Enter Function dialog. Used to enter current function.

Figure 2.9 FUNCTION menu

The **GRAPH MENU** shown in Figure 2.10 is used to draw graphs of non-polynomial functions, polynomial functions, regression prediction graphs and x-y data graphs. The x-y data graphs may be drawn with a regression curve superimposed over the data. Three items, First Graph, Last Graph and Quit Graph, are disabled until a graph is actually drawn on the screen. The Print items, on the **FILES** menu are used to print graphs to the printer. Parts of non-polynomial function, polynomial and regression prediction graphs may be magnified.

GRAPH	
Function ... ⌘I	Draws Graph Function dialog for drawing function graph. Disabled if no current function present.
Polynomial ... ⌘H	Draws Graph Polynomial dialog for drawing polynomial graph. Disabled if no current polynomial present.
Regression ... ⌘K	Draws Graph Regression dialog for drawing regression graph. Disabled if at least one regression curve has not been computed.
x-y Data ... ⌘J	Draws graph of x-y data with regression curves superimposed. Disabled if no x-y data has been entered by user.
Function B	Changes the graph active function, when two graphs are overlaid in the graphics mode. Disabled if only one function selected.
Stack...	Brings up stack dialog to allow user to do stack operations with the result of a graph calculation. Disabled until a graph calculation has been initiated.
Area	Computes area in graph mode. Disabled unless two points are selected.

Root	Computes root in graph mode. Disabled unless two points are selected.
Slope	Computes slope in graph mode. Disabled unless one point is selected.
First Graph ... ⌘N	Redraws graph using initial values for x domain. Disabled unless a graph is drawn.
Last Graph ... ⌘O	Redraws graph using previous x value for graph domain. Disabled unless a graph is drawn.
Quit Graph ... ⌘W	Quits graph and returns user to Scientific Calculator screen.

Figure 2.10 GRAPH menu

The **LINEAR SYSTEM MENU**, shown in Figure 2.11 is used to solve systems of linear equations. The systems may be either real or complex and may be as large as 10 x 10. Several of the items are disabled until the user has dimensioned the system with the Reset item and entered system data. System files and solutions may be saved to and recalled from disk.files

LINEAR SYSTEM	
Solve ... ⌘E	Draws Solve Linear System dialog. Disabled unless a real or complex system has been entered by the user.
Enter Real System...	Draws Enter Real System dialog. Disabled unless the system has been dimensioned with the Reset menu item.
Enter Complex System ...	Draws Enter Complex System dialog. Disabled unless the system has been dimensioned with the Reset menu item.
Reset ... ⌘M	Draw Reset System dialog. Used to set Linear System dimension.
Load System File ...	Draws Standard File dialog for recalling Linear System file.
Save System File ...	Draws Standard File dialog for saving Linear System file to disk. Disabled unless user has input a real or complex system.

Figure 2.11 LINEAR SYSTEM menu

Figure 2.12 shows the **MATRIX MENU** used for matrix operations. The user can add, subtract, transpose, multiply matrices by a scalar, enter matrix data, compute matrix determinants, multiply and invert matrices, using the Operations item. Matrix files and solutions may be saved and recalled from disk files. The matrix files may either be individual matrix files or Session files, that include all matrix files.

MATRIX	
Operations ...	Draws Matrix Operations dialog used for matrix calculations. Select this item to enter matrix data, do matrix file manipulation or to do matrix operations.
Load Matrix File ...	Draws Standard File dialog for recalling Matrix file from disk. This dialog allows the user to recall Matrix Session files, or individual matrix files from disk files.
Save Matrix File ...	Draws Standard File dialog for saving Matrix file to disk. Disabled unless user has input a matrix. This dialog allows the user to save Matrix Session files, or individual matrix files to disk files.

Figure 2.12 MATRIX menu

The **POLYNOMIAL MENU** show in Figure 2.13 is used for all polynomial operations. The Roots item will be disabled until a current polynomial is entered. Polynomial files and solutions may be saved to and recalled from disk files.

Polynomial operations use special solution algorithms that are specifically designed for polynomials. They are very fast and accurate. Solution time and accuracy of the polynomial solutions will not be as good if polynomials are solved using the **FUNCTION MENU** items.

POLYNOMIAL	
Area ...	Draws Polynomial Area dialog.
Differentiate ...	Draws Differentiate Polynomial dialog.
Binomial Division ...	Draws Binomial Division dialog.
Integrate ...	Draws Integrate Polynomial dialog.
Binomial Multiplication ...	Draws Binomial Multiplication dialog.
Roots ...	Draws Compute Polynomial Roots dialog. Disabled unless a current polynomial has been entered.
Slope ...	Draws Polynomial Slope dialog.
Solve ...	Draws Solve Polynomial dialog.

Enter Polynomial ...	Draws Enter Polynomial dialog. Used to enter current polynomial.
Load Polynomial File ...	Draws Standard File dialog for recalling Polynomial file from disk.
Save Polynomial File ...	Draws Standard File dialog for saving Polynomial file to disk. Disabled unless user has input a Polynomial.

Figure 2.13 POLYNOMIAL menu

Figure 2.14 is the **REGRESSION MENU.** This menu is used to select operations to enter x-y data, sort x-y data compute Linear, Log, Power and Exponential regressions on the x-y data and to save or recall x-y data files and their regression solutions to and from disk files.

REGRESSION	
Solve ... ⌘L	Draws Regression dialog.
Enter x-y file ... ⌘D	Draws Enter x-y Data dialog.
Sort x-y File	Sorts x-y Data, in ascending order on x.
Load x-y File ...	Draws Standard File dialog for recalling x-y Data file from disk.
Save x-y File ...	Draws Standard File dialog for saving x-y Data file to disk. Disabled unless user has input x-y Data.

Figure 2.14 REGRESSION menu

THE SCIENTIFIC CALCULATOR

This chapter will teach you how to enter numbers into the calculator and how to use the calculator to solve problems. We will start by explaining how to move the mouse around the screen and using it to control the calculator buttons.

Selecting and Activating Calculator Buttons

Start by holding the mouse button down and move the cursor (arrow) around on the screen. As the cursor moves around the screen notice the buttons change from their normal color to a black background with colored letters. When the cursor is over a button and the mouse key is down the button is "**selected**". This is always indicated by the inverted (black) background. If the mouse button is released on a "**selected**" button, the button is "**activated**", and the operation controlled by the button is initiated. This is exactly the same as pressing a button on your hand calculator, but with **GSNumerics** you will use the mouse and mouse button to select an operation. To see this more clearly enter the number 123 (use the number keys on the keyboard or the key pad) and press [return]. If you have just started the calculator, you will see the following in the **input-reg**: 1.230000e+2. This is the number 123 written in scientific format, with six digits shown after the decimal point. The calculator always starts with this selection for written format. Now move the cursor over [FIX] and key the mouse button. You have selected the fixed output format by this action. The number in the input-reg will now be written as 123.000000. Notice that the **decimal-reg** is 6, indicating six numbers are to be shown after the decimal point. Now move the mouse to the [-1] button and key the button four times. Each time you key on the [-1] button you decrement the **decimal-register** by one. Watch the number in the **input-reg** change each time you key the [-1] button. After keying it four times, the calculator display format will be fixed with two digits shown after the decimal point, and the **input-reg** will display 123.00.

When requesting the user to enter a number into the **input-reg**, we will use the following convention:

123.345 [return]

This will tell you to enter the number 123.345 into the **input-reg**, using the number pad or number keys on the keyboard, and to press return. Return can be initiated in three ways: 1. Pressing the return key, 2. Pressing the enter key or 3. keying the mouse with the cursor in the **input-reg** label area; It is up to to decide which method you will use. The three ways of entering the number are equivalent.

When you are requested to select and activate a button, the instruction will be shown as follows:
[FIX]

This would tell you to key up the mouse on the [FIX] button on the calculator keyboard. For example, the following would tell you to enter the number 123.456 and change the calculator to fixed format mode:

123.456 (return) [FIX]

A Few Simple Problems

Prior to explaining the calculator registers and **RPN** entry in detail, let's work some simple problems, so you can actually see some problems being worked with the **Scientific Calculator**. We will start by adding two numbers. Before starting the problem, make sure the format mode is fixed and the display is set to two places after the decimal point. For the present, don't pay any attention to the numbers in the **x-reg**, **y-reg**, **z-reg**, **w-reg**, or **lx-reg**. We will explain these later in the chapter.

12.0 (return)
13.6 (return) [ADD(y+x)]
answer: 25.60 displayed in **input-reg**

5.6 (return)
4.15 (return) [MUL(x*y)]
answer: 23.24 displayed in **input-reg**

14.56 (return)
28.6 (return) [SUB(y-x)]
answer: -14.04 displayed in **input-reg**

55.0 (return)
11 (return) [DIV(y/x)]
answer: 5.00 displayed in **input-reg**

See how simple it is to work problems using the **Scientific Calculator**.

The Stack Registers (x-reg, y-reg, z-reg and w-reg)

The stack registers play a very important part in the operation of the **Scientific Calculator**, and it is important for you to understand their operation. The stack, in conjunction with **RPN** entry, will allow you to solve any problem, without ever recording an intermediate operation. We will explain **RPN** entry in the next section. All actions, entering a number or activating a keyboard function, will have one of four actions on the stack registers.

The three actions are 1. Push Stack, 2. Pull Stack , 3. Operate on **x-reg** value or 4. No stack action. These actions are outlined in Figure 3.1.

Push Stack
When a new number is entered into the **input-reg**, or a constant or memory recall is initiated, the stack is "pushed". The **z-reg** value is placed into the **w-reg**, the **y-reg** value is placed into the **z-reg**, the **x-reg** value is placed into the **y-reg**, the **input-reg** value is placed into the **x-reg**. The value that was in the **w-reg**, prior to the stack push is lost.
Pull Stack
When a function of two variables is initiated, the stack registers are "pulled". The result of the function is placed in the **x-reg**, the **z-reg** value is placed in the **y-reg**, the **w-reg** value is place in the **z-reg** and the **w-reg** is unchanged. A function of two variables is any function that uses both the value in the **x-reg** and the value in the **y-reg** in computing it's result.
Operate on x-reg
When a function of one variable is executed, the result of the function is placed in the **x-reg.** The other stack registers are not changed. Conversions are functions of one variable, and only change the **x-reg.**
No stack action
Output formatting changes or mode changes will not change any of the stack registers. Memory storage operations will not change the stack registers.

Figure 3.1 Stack Operations

We will now work some simple problems and observe the action of the stack as the problems are solved. Before starting, make sure the format is Fixed, with two digits after the decimal point and activate the **CLRS** button. This is the **Clear Stack** button and it will clear all of the stack registers and the **input-reg** to zero.

We will first add two numbers (9.5 +8.5). then we will add two other numbers (3.5 + 5.5) and finally, we will divide the result of (9.5 +8.5) by the result of (3.5 +5.5). This simple problem will show you how the stack will help you in complicated calculations. Notice how the stack action keeps the operands in the proper position for the calculation.

$$\frac{(9.5 + 8.5)}{(3.5 + 5.5)}$$

9.5 [return] **x-reg** = 9.5000000e+0 (push)

8.5 [return] **x-reg** = 8.5000000e+0 **y-reg** = 9.5000000e+0 (push)

[ADD(y+x)] This is a function of two variables (the stack will be pulled)

answer in **input-reg** = 18.00 and **x-reg** = 1.8000000e+0. **y-reg** = 0.0000000e+0 (pull)

3.5 [return] **x-reg** = 3.5000000e+0 **y-reg** = 1.8000000e+0 (push)

5.5 [return] **x-reg** = 5.5000000e+0 **y-reg** = 3.5000000e+0 **z-reg** = 1.8000000e+0 (push)

[ADD(y+x)]

answer in **input-reg**=9.00 **x-reg** = 9.0000000e+0 **y-reg**=1.8000000e+0 **z-reg**=0.0000000e+0 (pull)

Notice that the results in the **x-reg** and **y-reg** are exactly in position to perform the division operation using the [DIV(y/x)] button. To complete the solution, all you have to do is:

[DIV(y/x)] This is a function of two variables (the stack will be pulled)
answer in **input-reg**=2.00 **x-reg** = 2.0000000e+0
y-reg=0.0000000e+0 **z-reg**=0.0000000e+0 (pull)

This simple example shows how the stack registers help you in solving complicated problems. If you always work the parts of a problem in the inner-most parenthesis first, and work outward, you will never have to write down or save any intermediate results. Also notice that the stack registers always display their values in scientific notation with seven digits displayed after the decimal point, and that after a calculation, the **x-reg** will always display the same value that is in the **input-reg**.

In the previous problem we only dealt with functions of two variables. As outlined in Figure 3.1 the two variable functions always pull the stack when executed. We will now work a problem that includes a function of two variables, two constant recalls and two functions of one variable. This problem will show you how a constant recall and a function of one variable operate on the stack. Referring to Figure 3.1 you will see that the functions of one variable will not push or pull the stack, and will only change the value of the **x-reg** . The constant recalls will always push the stack.

To demonstrate the effect on the stack of a single variable function, a two variable function and a constant recall, consider the following problem:

$$\frac{2*Pi + \sqrt{e^2-.85}}{Log(98)}$$ Formula 3.1

In this example, there are two single value functions; **Log(98)** and the square root operation $\sqrt{e^2-.85}$ and two constant recalls: **2Pi** and **e** (base of the natural logarithms). The addition of **2*Pi** + $\sqrt{e2-.85}$ and the subtraction e^2**-.85** are, of course, functions of two variables. We will work the problem, by solving the numerator first and by solving the term inside the square root first in the numerator.

	EE*	**x-reg**=2.7182818e+0 (push) - this is a constant recall
	SQR(x)*	**x-reg**=7.3890561e+0 (no action) - function of one variable
.85	return	**x-reg**=8.5000000e-1 **y-reg** = 7.3890561e+0 (push)
	SUB(y-x)	**x-reg**=6.5390561e+0 **y-reg**=0.0000000e+0 (pull) function of two variables
	SQRT(x)*	**x-reg**=2.5571578e+0 (no action) - function of one variable
	TPi*	**x-reg**=6.2831853e+0 **y-reg**=2.5571578e+0 (push) - a constant function
	ADD(y+x)	**x-reg**=8.8403431e+0 **y-reg**=0.0000000e+0 (pull) two variable function This completes the solution of the numerator.
98	return	**x-reg**=9.8000000e+1 **y-reg** = 8.8403431e+0 (push)
	LOG(x)*	**x-reg**=1.9912261e+0 **y-reg** = 8.8403431e+0 (no action)-one variable
	DIV(y/x)	This is a function of two variables (the stack will be pulled)

answer in **input-reg**=4.44 **x-reg** = 4.4396481e+0 **y-reg**=0.0000000e0 **z-reg**=0.0000000e+0 (pull)

From this example you can study the action of the stack, and see how **GSNumerics** will enable you to solve complex problems very easily. You did not have to remember or look up the values of **e** or **2*Pi**, you did not have to write down any intermediate steps and you solved this problem in ten easy steps!

RPN Entry

As you have probably noticed, we have not been entering numbers and math operators in the sequence that is normally used with pocket calculators. For example, addition on most calculators will require you to enter a number, then press the "+" key , enter the second number and then press the "=" key to compute the result. This is called the "**in-fix**" entry method.

In contrast to this, we have performed addition by entering the first number, entering the second number and then keying the ADD(y+x) button. This method is called **Reverse Polish Notation (RPN)** or "**post-fix**" entry. **RPN** is better suited for complex mathematical calculations than "**in-fix**" entry because you can solve any function, no matter how large, without saving intermediate results or having to insert parenthesis around groupings. The only thing you have to remember is to solve the problems from the inside to the outside. By this we mean to solve those terms in the innermost parenthesis first.

It will also make things a little easier if you solve the numerator terms of fractions before solving the denominator. **RPN** is applied to functions of one variable and functions of two variables, as shown in Figure 3.2

ONE VARIABLE FUNCTIONS

Functions of one variable always operate on the value in the **x-reg**. The value is entered, pushing the stack, then the function is initiated. The stack is not pushed or pulled. The result is returned in the **x-reg**. The answer is also written to the **input-reg**, in the format selected by the user.

TWO VARIABLE FUNCTIONS

Functions of two variables always operate on the value in the **x-reg** and the value in the **y-reg**. The values are entered, pushing the stack each time, then the function is executed. The stack is pulled when the function is executed, with the result returned in the **x-reg**. The answer is also written to the **input-reg**, in the format selected by the user.

Figure 3.2 RPN Stack Operation

While **RPN** entry may seem a little different when you first use it, you will find it much easier to use after you work a few problems and get use to it. When you are comfortable with it you will probably never want to go back to a calculator using "**in-fix**" entry!

Just so you understand how to work a problem efficiently using **RPN**, consider the following problem:

$$\frac{1 - (2.3 * \sqrt{3.5\text{-PI}})}{\sin(23.7) + (e * \cos(43.9))} + \sqrt[3]{\tan(45.6)}$$ **Problem 3.3**

First, solve the numerator, working from the inside to the outside:

	DM	Put calculator into degrees mode. Trigonometric inputs are in degrees
3.5	return	**x-reg**=3.5000000e+0
	Pi*	**x-reg**=3.1415927e+0 **y-reg**=3.5000000e+0 (push)
	SUB(y-x)	**y-reg**=3.5840735e-1 (pull)
	SQRT(x)*	**x-reg**=5.9867132e-1 (no action - single variable function)
2.3	return	**x-reg**=2.3000000e+0 **y-reg**=5.9867132e-1 (push)
	MUL(x*y)	**x-reg**=1.3769440e+0 (pull)
1	return	**x-reg**=1.0000000e+0 **y-reg**=1.3769440e+0 (push)

	XCHG(x-y)	**x-reg**=1.3769440e+0 **y-reg**=1.0000000e+0 (exchange **x-reg** and **y-reg**)
	SUB(y-z)	**x-reg**=-3.7694403e-1 (pull)

This completes the solution of the numerator. Now solve the denominator and watch how the numerator value stays on the stack and is ready for computation, when you need it.

43.9	return	**x-reg**=4.3900000e+1 **y-reg**=-3.7694403e-1 (push)
	COS(x)*	**x-reg**=7.2055111e-1 **y-reg**=-3.7694403e-1
	EE*	**x-reg**=2.7182818e+0 **y-reg**=7.2055111e-1 **z-reg**=-3.7694403e-1 (push)
	MUL(x*y)	**x-reg**=1.9586610e+0 **y-reg**=-3.7694403e-1 (pull)
23.7	return	**x-reg**=2.3700000e+1 **y-reg**=1.9586610e+0 **z-reg**=-3.7694403e-1 (push)
	SIN(x)*	**x-reg**=4.0194778e-1 **y-reg**=1.9586610e+0 **z-reg**=-3.7694403e-1
	ADD(y+z)	**x-reg**=2.3606088e+0 **y-reg**=-3.7694403e-1 (pull)

This completes the solution of the denominator. Now finish computation of the first term:

DIV(y/x)	**x-reg**=-1.5968086e-1 (pull)

Now we can compute the second term.

45.6	return	**x-reg**=4.5600000e+1 **y-reg**=-1.5968086e-1 (push)
	TAN(x)*	**x-reg**=1.0211664e+0 **y-reg**=-1.5968086e-1
	CBRT(x)*	**x-reg**=1.0070063e+0 **y-reg**=-1.5968086e-1

Now we will add the first and second terms, completing the solution:

ADD(y+z)	**x-reg**=8.4732540e-1 (pull)

answer in **input-reg**=0.85

Again, you can see how simple it is to solve complicated equations using **GSNumerics**. From this point on, we will discontinue indicating stack pulls and pushes, since you now understand the concept.

Initial Settings

When you first start the **GSNumerics** program, the following initializations are set as shown in Figure 3.3.

1.	The **input-reg** format mode is set to scientific format, with six digits shown after the decimal point.
2.	The **trig mode** mode is set to "radians". With this setting all angles to the trig functions must be input in radians and the output of all inverse trig functions (i.e. **ASIN**, **ATAN**, etc) will be in radian measurement.

Figure 3.3 Program Initial Settings

You may set the format mode to suit your taste. If you are going to work problems using the trig functions and are not going to input angles in radian measurement, you should change the **trig mode** as explained later in this chapter. As an alternative, you can use the **angle conversion** buttons to convert to radians prior to initiating a trig calculation. These buttons are also explained later in the chapter.

The Last Stack Button - Undoing Mistakes

We aren't perfect beings and will sometimes make a mistake. You might enter a wrong number or activate the wrong button. This could cause you a lot of extra work, if you were in the middle of solving a long and complex problem. If you did not have some method of undoing the mistake you would have to start from the beginning and work the problem over. Not a very happy thought! The program provides the [S] button to get you out of these predicaments. This button is called the **Last Stack** button. When you perform any operation with **GSNumerics**, the program saves the stack values in a hidden last stack, prior to doing the operation. If you make a mistake, just key on [S] and the stack will be restored to the values prior to the mistake.

Suppose you were ready to compute the final addition in Problem 3.3, where you were adding the first and second terms. At that point the registers conditions were:

x-reg=1.0070063e+0 **y-reg**=-1.5968086e-1

Now we will add the first and second terms, completing the solution:

[DIV(y/x)] **x-reg**=-1.5856987e-1 (pull) **OOPS!! We hit the wrong button!!**
answer in **input-reg**=-0.16

We can easily correct the mistake by:

[S] **x-reg**=1.0070063e+0 **y-reg**=-1.5968086e-1 (recall **Last Stack**)
[ADD(y+x)] **x-reg**=8.4732540e-1 (pull)
answer in **input-reg**=0.85 (This is the correct answer)

You may recall the **Last Stack** at any time to undo mistakes in number inputs or mistakes caused by hitting the wrong button. The **Last Stack** is always saved for you, automatically, when you initiate any action in the **Scientific Calculator**. There two [S] buttons on the calculator screen. The **Last Stack** is the [S] button above the stack and to the left of the stack label "**STACK**". The other [S] button is used in memory operations and will be explained in a later chapter.

Stack Control Buttons

In addition to the [S] button there are seven other buttons to control stack variables. These buttons and their operations are:

- [UP] This button rolls the stack up, with the **x-reg** value placed in the **y-reg**, the **y-reg** value placed in the **z-reg**, the **z-reg** value placed in the **w-reg** and the **w-reg** value is put into the **x-reg**.
- [DN] This button rolls the stack down, with the **x-reg** value placed in the **w-reg**, the **w-reg** value placed in the **z-reg**, the **z-reg** value placed in the **y-reg** and the **y-reg** value placed in the **x-reg**.
- [x] This button pushes the **x-reg** value into the **x-reg** and pushes the stack, with the **x-reg** value placed in the **y-reg**, the **y-reg** value placed in the **z-reg**, and the **z-reg** value is placed in the **w-reg**. The **w-reg** value is lost.
- [y] This button pushes the **y-reg** value into the **x-reg** and pushes the stack, with the **x-reg** value being placed in the **y-reg**, the **y-reg** value placed in the **z-reg** and the **z-reg** value is placed in the **w-reg**. The **w-reg** value is lost.
- [z] This button pushes the **z-reg** value into the **x-reg** and pushes the stack, with the **x-reg** value being placed in the **y-reg**, the **y-reg** value placed in the **z-reg** and the **z-reg** value is placed in the **w-reg**. The **w-reg** value is lost.
- [w] This button pushes the **w-reg** value into the **x-reg** and pushes the stack, with the **x-reg** value being placed in the **y-reg**, the **y-reg** value placed in the **z-reg** and the **z-reg** value is placed in the **w-reg**. The **w-reg** value is lost.
- [L] This is the **Last x** button. When an operation is performed by the calculator, the current value of the **x-reg** is saved in the **Last x** register prior to the operation being done. The user may push the **Last x** to the **x-reg** with this button.

Calculator Mode Buttons

There are ten mode buttons on the calculator screen. Three of the mode buttons control the type of input and output from the trig functions, three control different operating modes of the calculator and one changes the conversion section to a chart of the Atomic Elements.

The use of CAL, FCN and PAD will be explained later in the chapter. These buttons are used to put the calculator into one of its three operating modes, **Scientific Calculator, Function Entry Calculator** or the **Ten Key Calculator**. The program will always start with the calculator being in the **Scientific Calculator** mode.

The three angle input buttons control the way in which the calculator interprets number inputs to the trig functions. This is called the **Scientific Calculator trig mode**. It is very important that you have the **trig mode** set to the units that you will be entering you trig function angles in. If you don't enter angles, in the same units as the **trig mode** setting, the answers returned by the trigonometry and inverse trigonometry functions will be wrong!! The Scientific Calculator and all other programs computations, using the trigonometric functions, assume the angles are in the **trig mode** form selected on the **Scientific Calculator**. When the program first starts, the **trig mode** will always be set to the **radians mode**.

The last mode button will toggle the **Converstion-Elements** section,back and forth between the **Conversion** mode and the **Elements** mode. This is fully explained, later in this chapter.

DM	When degree mode is selected, all input to the trig functions must be input in degrees. The output of the inverse trig functions will be expressed in degrees.
RM	When radian mode is selected, all input to the trig functions must be input in radians. The output of the inverse trig functions will be expressed in radians.
GM	When grad mode is selected, all input to the trig functions must be input in grads. The output of the inverse trig functions will be expressed in grads.

Figure 3.4 Trig mode settings

It is very important for the user to have the **trig mode** properly set. If you have the **trig mode** set to radians and input angles to the calculator in degrees, the results of any trig calculations will not be correct. As as example, consider the following problems, with the angles being input in degree form.

	DM	Set calculator to degree mode.
30	return	x-reg=3.0000000e+1 (want to solve for the sine of 30 degrees)
	SIN(x)*	x-reg=5.0000000e-1 (This is the correct answer)
	RM	Set calculator to radian mode.
30	return	x-reg=3.0000000e+1 (want to solve for the sine of 30 degrees)
	SIN(x)*	x-reg=-9.8803162e-1 (This is an incorrect answer)

In the second case the calculator actually solved for the sine of 30 radians, not the sine of 30 degrees, as was expected. As another example, suppose you wanted to find the angle, in radians, whose cosine is equal to 0.5. This computation involves the use of the arc cosine (sometimes referred to as the inverse cosine or cosine^{-1}).

	RM	Set calculator to radian mode.
,5	return	x-reg=5.0000000e-1 (want to solve for the arc cosine of 0.5 radians)
	ACOS(x)*	x-reg=1.0471976e+0 (This is the correct answer in radians)
	DM	Set calculator to degree mode.
.5	return	x-reg=5.0000000e-1 (want to solve for the arc cosine 0.5 radians)
	ACOS(x)*	x-reg=6.0000000e+1 (This is an incorrect answer)

In the second problem the calculator actually returned the degree angle whose cosine is equal to 0.5, not the radian angle whose cosine is equal to 0.5 In most cases, problems will be expressed in a definite angular form and the user will know which form to set the calculator angle mode. If you encounter problems, that have mixed angles (i.e. some expressed in degrees and others expressed in grads or radians), the calculator provides angle conversion functions to convert these to the desired mode before initiating a trig calculation. This gives the user complete freedom to mix different angular measurement forms in the same problem.

You must be aware of the form of angles and what form the inverse trig functions will return results. This will become very natural and easy as you gain more experience with using **GSNumerics.**

Angle Conversion Functions

Angular measurement for the trig functions can be input in three forms. The forms and their relationships are shown in Figure 3.5

degrees	1 degree $= \frac{2Pi}{360.0}$ radians $= \frac{400.0}{360.0}$ grads
radians	1 radian $= \frac{360.0}{2Pi}$ degrees $= \frac{400.0}{2Pi}$ grads
grads	1 grad $= \frac{360.0}{400.0}$ degrees $= \frac{2Pi}{400.0}$ radians

Figure 3.5 Angle Conversion Relationships

It is sometimes necessary to convert back and forth between the three forms of angular measurement, and the calculator provides six functions to make these conversions for the user, quickly and accurately. These functions are functions of one variable, and merely make the conversion on the **x-reg**, without doing a stack push or pull. The six angle conversion functions and their operation is explained in Figure 3.6.

DR*	Converts the value in the x-reg from degrees to radians.
DG*	Converts the value in the x-reg from degrees to grads.
GD*	Converts the value in the x-reg from grads to degrees.
GR*	Converts the value in the x-reg from grads to radians.
RD*	Converts the value in the x-reg from radians to degrees.
RG*	Converts the value in the x-reg from radians to grads.

Figure 3.6 Angle Conversion Buttons

To demonstrate the use of these functions, we will input 30.0 degrees into the x-reg and then convert it to radians, then to grads and finally back to degrees. This will let you see how the angle conversion functions work. Notice the stack is not affected by the conversion, and that these functions are functions of one variable, only affecting the **x-reg** value.

30	return	**x-reg**=3.0000000e+1
	DG*	**x-reg**=3.3333333e+1 (30.0 degrees is equal to 33.333 grads)
	GR*	**x-reg**=5.2359878e-1 (33.333 grads is equal to 0.5236 radians)
	RD*	**x-reg**=3.0000000e+1 (0.5236 radians is equal to.30.0 degrees)

You can see that it is very simple to convert between degrees, grads and radians with the scientific calculator.

Clearing The Stack Registers

When starting a new problem, the user may want to reset the stack registers and the **input-reg** to 0.0. This is called clearing the registers. There are two buttons that will clear the stack registers and **input-reg**. The CLRS button (clear stack) will only clear the stack registers and **input-reg**, while the CLRA button (clear all) will clear the stack registers, **input-reg**, all memory storage and the complex storage registers. The memory storage and complex storage registers will be explained in later chapters, so you should use the CLRS to clear the stack registers and **input-reg** for the time being.

Using The General Functions

This section will review the functions in the general functions group of the Calculator Function section. We will explain each function and give some examples of its' use.

CEIL(x)*	The Ceiling of x function is a single variable function, effecting the **x-reg** only. This function returns the integer, greater than or equal to the **x-reg** value, with the sign of the **x-reg** value.

1.5 return CEIL(x)* **x-reg**=2.0000000e+0
-2.3 return CEIL(x)* **x-reg**=-3.0000000e+0
42.2 return CEIL(x)* **x-reg**=4.3000000e+1

RAND*	The Random function returns a random number whose value is between 0.0 and 1.0 to the x-reg. This function acts the same as a constant, and pushes the stack.

RAND* **x-reg**=6.2001508e-1
RAND* **x-reg**=5.9340109e-1

INT(x)*	The Integer x function is a single variable function, effecting the **x-reg** only. This function returns the integer part of the **x-reg** value.

1.5 return INT(x)* **x-reg**=1.0000000e+0
-2.3 return INT(x)* **x-reg**=-2.0000000e+0

FRAC(x)*	The Fractional x function is a single variable function, effecting the **x-reg** only. This function returns the fractional part of the x-reg value.

1.5 return FRAC(x)* x-reg=5.0000000e-1
-2.3 return FRAC(x)* x-reg=-3.0000000e-1
42.2 return FRAC(x)* x-reg=2.0000000e-1

ABS(x)*	The Absolute value of x function is a single variable function, effecting the **x-reg** only. This function returns the absolute value of the **x-reg** value.

1.5 return ABS(x)* x-reg=1.5000000e+0
-2.3 return ABS(x)* x-reg=2.3000000e+0

FACT(x)*	The Factorial of x function is a single variable function, effecting the **x-reg** only. This function returns the Factorial of the value in the **x-reg**, If the **x-reg** value is not an integer with its fractional part equal to 0.0 the Factorial function will return **NAN(062)**, since factorial is only defined for integer values of x. If the **x-reg** value is negative, the function will return **NAN(061)**, since factorial is not defined for negative values of x. The largest factorial that can be computed is for x = 1754. For x values larger than 1754, the function will return **INF**.

8 return FACT(x)* x-reg=4.0320000e+4
32 return FACT(x)* x-reg=2.6313084e+35

DHMS*	The Decimal to Hours-Minutes-Seconds function is a function of one variable and converts the **x-reg** value from time or angles, expressed as a decimal to the hours-minutes-seconds format (**hh.mmss**). Expressed in **hh.mmss** format, the value 12.232155 would be interpreted as 12 hours, 23 minutes, 21.55 seconds. Returns **NAN(061)** for arguments whose absolute value is greater that 1e+12.

1.5 return DHMS* x-reg=1.3000000e+0 (one hour and thirty minutes)
12.345 return DHMS* x-reg=1.2204200e+1 (twelve hours, 20 minutes, 42 seconds)

HMSD*	The Hours-Minutes-Seconds to Decimal function is a function of one variable and converts the **x-reg** value from time or angles, in the hours-minutes-seconds format (**hh.mmss**) to a value expressed as a decimal number. Returns **NAN(061)** for arguments whose absolute value is greater that 1e+12.

1.22042 return HMSD* **x-reg**=1.3678333e+0
5.4030 return HMSD* **x-reg**=5.6750000e+0

HADD(+)*	The Add Hours-Minutes-Seconds function is a function of two variables. This function computes the sum of the time or angle values of the **x-reg** and **y-reg**. The **x-reg** values and **y-reg** values must be expressed in **hh.mmss** format, and the result is expressed in **hh.mmss** format. Returns **NAN(061)** for arguments whose absolute value is greater that 1e+12.

Compute the sum of two hours, 33 minutes and 44 seconds and seven hours, 12 minutes and 16 seconds.

2.3344 return
7.1216 return HADD(+)* **x-reg**=9.4600000e+0 (nine hours, 46 minutes)

HSUB(-)*	The Subtract Hours-Minutes-Seconds function is a function of two variables that computes the difference in the time or angle values of the **x-reg** and **y-reg**. The **x-reg** values, **y-reg** values and result are expressed in **hh.mmss** format. Returns **NAN(061)** for arguments whose absolute value is greater that 1e+12.

Subtract seven hours, twelve minutes and 14 seconds from nine hours and 46 minutes.

9.4600 return
7.1214 return HSUB(-)* **x-reg**=2.3346000e+0

Using The Log and Exponential Functions

We will now review the functions in the Log and Exponential group of the Calculator Functions Section. These functions deal with the problem of raising numbers to powers, and taking the logarithms and inverse logarithms of numbers

LOGy(x)	This functions finds the log of the **x-reg** value to the base of the value in the **y-reg**. Logarithms for negative numbers are not defined in mathematics. This function will return **NAN(036)** for any **x-reg** or **y-reg** value less than zero and **-INF** for an **x-reg** value equal to zero.

Find the log of 34.5 to the base 11.

11 return

34.5 return LOGy(x) **x-reg**=1.4766947e+0

Find the log of 128 to the base 2.

2 return

128 return LOGy(x) **x-reg**=7.0000000

yEXP(x)	This functions raises the **y-reg** value to the power of the value in the **x-reg**. You may use this function to find roots of numbers, raise them to various powers and to find inverse logarithms.

Find $(1.456)^{31}$

1.456 return

31 return yEXP(x) **x-reg**=1.1428852e+5

Find $\sqrt[31]{1.1428852e+5}$ (This is equal to $(1.1428852e+5)^{1/31}$)

1.1428852e+5 return

31 return 1/x **x-reg**=3.2258065e-2 (take the reciprocal of **x-reg**)

yEXP(x) **x-reg**=1.4560000e+0 (just what we expected)

Find the number that has a logarithm of 7.0 in the base 2. (what number is 2 raised to the 7th power)

2 return

7 return yEXP(x) **x-reg**=1.2800000e+2 (again, what we expected)

SQR(x)*	This function squares (raises **x-reg** to the second power) the **x-reg** value. This is a function of one variable, operating on the **x-reg** only.

6.5 return SQR(x)* **x-reg**=4.2250000e+1

-7 return SQR(x)* **x-reg**=4.9000000e+1

SQRT(x)*	This function takes the square root of the **x-reg** value. It is a function of one variable, operating on the **x-reg** only. This function will return **NAN(037)** for negative **x-reg** values.

36 return SQRT(x)* x-reg=6.000000e+1
-9 return SQRT(x)* x-reg=NAN(037)

CUBE(x)*	This function cubes (raises to the third power) the x-reg value. It is a function of a single variable.

3 return CUBE(x)* x-reg=2.7000000e+1
12.45 return CUBE(x)* x-reg=1.9297811e+3

CBRT(x)*	This function takes the cube root of the **x-reg** value. It is a function of a single variable. The function will return **NAN(037)** for negative **x-reg** arguments.

1000 return CBRT(x)* x-reg=1.0000000e+1
64 return CBRT(x)* x-reg=4.0000000e+0

LN(x)*	This function returns the natural logarithm of the **x-reg** value. It is a function of a single variable. The function will return **NAN(036)** for negative **x-reg** arguments and **-INF** for x-reg value equal to **0.0**.

23.4 return LN(x)* x-reg=3.1527360e+0
-5.5 return LN(x)* x-reg=NAN(036)

EEXP(x)*	This function returns **e** (base of the natural logarithms) raised to the power of the **x-reg** value. This is the inverse of the natural logarithm function. It is a single variable function.

Find the number whose natural logarithm is equal to 4.55
4.55 return EEXP(x)* x-reg=9.4632408e+1

LNS(x)*	This function returns the natural logarithm of the **x-reg** value. It is a function of a single variable. It should be used for very small values of x, since it will return a more accurate value for small x than LN(x)*. The function will return **NAN(036)** for negative arguments and **-INF** for **x-reg** values equal to **0.0**.

Find the natural logarithm of 2.32e-5

2.32e-5 return LNS(x)* **x-reg**=2.3199731e-5

EEXPS(x)*	This function returns e (base of the natural logarithms) raised to the power of the **x-reg** value. It should be used for small values of x. This is the inverse of the natural logarithm function for small values of x. It is a single variable function.

Find the number whose natural logarithm is equal to 1.2543e-4

1.25e-4 return EEXPS(x)* **x-reg**=1.2500781e-4

LOG(x)*	This function returns the base 10 logarithm of the **x-reg** value. It is a function of a single variable. The function will return **NAN(036)** for negative **x-reg** arguments and **-INF** for x-reg value equal to **0.0**.

Find the base 10 logarithm of 98.5

98.5 return LOG(x)* **x-reg**=1.9934362e+0

ALOG(x)*	This function returns 10 raised to the power of the **x-reg** value. This is the inverse of the base 10 Log function. It is a function of a single variable.

Find the number whose base 10 logarithm is equal to .9345

.9345 return ALOG(x)* **x-reg**=8.6000307e+0

LTWO(x)*	This function returns the base 2 logarithm of the **x-reg** value. It is a function of a single variable. The function will return **NAN(036)** for negative **x-reg** arguments and **-INF** for x-reg value equal to **0.0**.

Find the base 2 logarithm of 4096

4096 return LTWO(x)* x-reg=1.2000000e+1

ALTWO(x)*	This function returns 2 raised to the power of the **x-reg** value. This is the inverse of the base 2 Log function. It is a function of a single variable.

Find the number whose base 2 logarithm is equal to 16

16 return ALTWO(x)* x-reg=6.5536000e+4

Using The Trigonometry Functions

In this section we will review the trigonometry functions located in the Calculator Functions Section on the screen. These functions contain the functions used for trig calculations and the inverse trig functions. Put the **Scientific Calculator** in degrees mode by keying on DM.

SIN(x)*	The sine function returns the sine of the **x-reg** value. It is a single variable function and interprets the input argument according to the angle mode setting on the calculator. **-INF < x-reg< +INF (input argument domain)** **-1 ≤ sin(x) ≤ +1 (result range)**

Find the sine of 32.5°

32.5 return SIN(x)* x-reg=5.3729961e-1

COS(x)*	The cosine function returns the cosine of the **x-reg** value. It is a single variable function and interprets the input argument according to the angle mode setting on the calculator. **- INF < x-reg < +INF (input argument domain)** **-1 ≤ cos(x) ≤ +1 (result range)**

Find the cosine of 48.0°

48 return COS(x)* x-reg=6.6913061e-1

TAN(x)*	The tangent function returns the tangent of the **x-reg** value. It is a single variable function and interprets the input argument according to the angle mode setting on the calculator. **-INF < x-reg < +INF (input argument domain)** **-INF ≤ tan(x) ≤ +INF (result range)**

Find the tangent of 45°

45 return TAN(x)* **x-reg**=1.0000000e+0

ASIN(x)*	The arc sine function returns the inverse sine of the **x-reg** value. It is a single variable function. Illegal arguments return **NAN(034)**. **-1 ≤ x-reg ≤ +1 (input argument domain)** **-90° ≤ asin(x) ≤ +90° (result range)**

Find the angle whose sine is equal to .743

.743 return ASIN(x)* **x-reg**=4.7987600e+1

ACOS(x)*	The arc cosine function returns the inverse cosine of the **x-reg** value. It is a single variable function. Illegal arguments return **NAN(034)**. **-1 ≤ x-reg ≤ +1 (input argument domain)** **-90° ≤ acos(x) ≤ +90° (result range)**

Find the angle whose cosine is equal to .50

.5 return ACOS(x)* **x-reg**=6.0000000e+1

ATAN(x)*	The arc tangent function returns the inverse tangent of the **x-reg** value. It is a single variable function. **-INF ≤ x-reg ≤ +INF (input argument domain)** **-90° ≤ atan(x) ≤ +90° (result range)**

Find the angle whose tangent is equal to 32.34

32.24 return ATAN(x)* **x-reg**=8.8223405e+1

CSC(x)*	The cosecant function returns the cosecant of the **x-reg** value. It is a single variable function. **-INF ≤ x-reg ≤ +INF (input argument domain)** **-INF ≤ csc(x) ≤ -1 and +1 ≤ csc(x) ≤ +INF (result range)**

Find the cosecant of 45°

45 return CSC(x)* x-reg=1.4142136e+0

SEC(x)*	The secant function returns the secant of the **x-reg** value. It is a single variable function. **-INF ≤ x-reg ≤ +INF (input argument domain)** **-INF ≤ csc(x) ≤ -1 and +1 ≤ csc(x) ≤ +INF (result range)**

Find the secant of 60°

60 return SEC(x)* x-reg=2.000000e+0

COT(x)*	The cotangent function returns the cotangent of the **x-reg** value. It is a single variable function. **-INF ≤ x-reg ≤ +INF (input argument domain)** **-INF ≤ cot(x) ≤ +INF (result range)**

Find the cotangent of 20°

20 return COT(x)* x-reg=2.7474774e+0

ACSC(x)*	The arc cosecant function returns the inverse cosecant of the **x-reg.** It is a single variable function.Illegal arguments return **NAN(034).** **-INF ≤ x-reg ≤ -1 and +1 ≤ x-reg ≤ +INF(input domain)** **0° < acsc(x) ≤ +90° for x ≥ 1 (result range)** **0° < acsc(x) ≤ -90° for x ≤ 1 (result range)**

Find the angle whose cosecant is 2.987

2.987 return ACSC(x)* x-reg=1.9559408e+1

ASEC(x)*	The arc secant function returns the inverse secant of the **x-reg.** It is a single variable function. Illegal arguments return **NAN(034).** **-INF ≤ x-reg ≤ -1 and +1 ≤ x-reg ≤ +INF (input domain)** **0° < acsc(x) ≤ +90° for x ≥ 1 (result range)** **-180° < acsc(x) ≤ -90° for x ≤ 1 (result range)**

Find the angle whose secant is 2.145

2.145 return ASEC(x)* x-reg=6.2212058e+1

ACOT(x)*	The arc cotangent returns the inverse cotangent of the **x-reg.** It is a single variable function. Illegal arguments return **NAN(034).** **-INF ≤ x-reg ≤ +INF (input argument domain)** **-90° ≤ acsc(x) ≤ +90° (result range)**

Find the angle whose cotangent is 4.56

4.56 return ACOT(x)* x-reg=1.2369062e+1

SINH(x)*	The hyperbolic sine function returns the hyperbolic sine of the **x-reg** value. It is a single value function. **-INF ≤ x-reg ≤ +INF (input argument domain)** **-INF ≤ sinh(x) ≤ +INF (result range)**

Find the hyperbolic sine of 121.34°.

121.34 return SINH(x)* x-reg=4.0961928e+0

COSH(x)*	The hyperbolic cosine function returns the hyperbolic cosine of the **x-reg** value. It is a single value function. **-INF ≤ x-reg ≤ +INF (input argument domain)** **1 ≤ cosh(x) ≤ +INF (result range)**

Find the hyperbolic sine of 10.0°.

10 return COSH(x)* x-reg=1.0152696e+0

TANH(x)*	The hyperbolic tangent function returns the hyperbolic tangent of the **x-reg** value. It is a single value function. **-INF ≤ x-reg ≤ +INF (input argument domain)** **-1 ≤ tanh(x) ≤ +1 (result range)**

Find the hyperbolic tangent of 91.0°.

91 return TANH(x)* **x-reg**=9.1988052e-1

ASINH(x)*	The arc hyperbolic sine function returns the inverse hyperbolic sine of the **x-reg** value. It is a single value function. **-INF ≤ x-reg ≤ +INF (input argument domain)** **-INF ≤ asinh(x) ≤ +INF (result range)**

Find the angle whose hyperbolic sine is 156.98

156.98 return ASINH(x)* **x-reg**=3.2940923e+2

ACOSH(x)*	The arc hyperbolic cosine function returns the inverse hyperbolic cosine of the **x-reg** value. It is a single value function. Illegal arguments return **NAN(036).** **1 ≤ x-reg ≤ +INF (input argument domain)** **-INF ≤ acosh(x) ≤ +INF (result range)**

Find the angle whose hyperbolic cosine is 2.476

2.476 return ACOSH(x)* **x-reg**=8.9167383e+1

ATANH(x)*	The arc hyperbolic tangent function returns the inverse hyperbolic tangent of the **x-reg** value. It is a single value function. Illegal arguments return **NAN(036).** **-1 ≤ x-reg ≤ +1 (input argument domain)** **-INF ≤ atanh(x) ≤ +INF (result range)**

Find the angle whose hyperbolic tangent is .539

.539 return ATANH(x)* **x-reg**=3.4534747e+1

Using The Conversion Section

The conversion buttons allow the user to convert the **x-reg** value between different systems of measurement very quickly. The Conversion section is shown in Figure 3.7

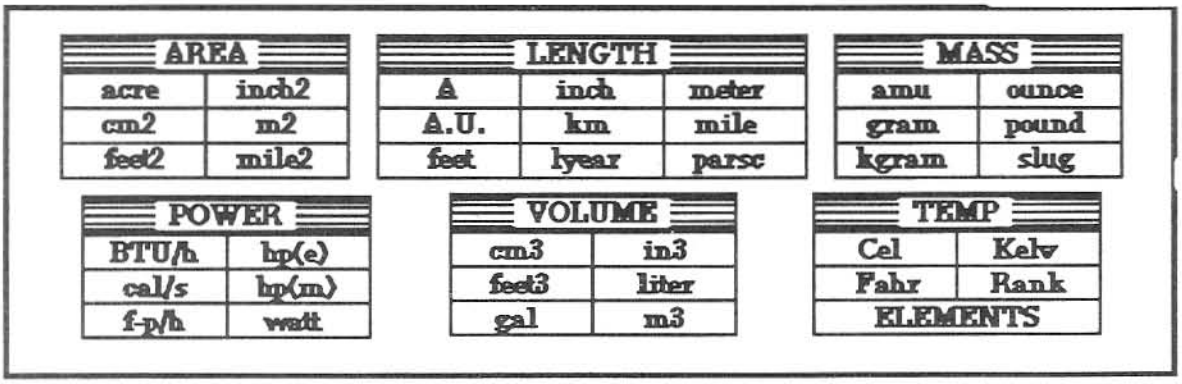

Figure 3.7 Conversion Screen

This section is divided into six sub-sections, each sub-section containing like units of measurement. Conversions are done very easily with **GSNumerics**. The procedure is to activate the conversion "**from**" button and then activate the "**to**" button. When you activate a button in a sub-section all of the buttons on the screen are made inactive, except for the buttons in the sub-section containing the activated button. The activated button will be drawn with a white background, showing that it is activated. After activating the "**from**" button, you then simply activate the "to" button and the conversion will be made on the value in the **x-reg**. At this point the other screen buttons will be reactivated.

If you make a mistake while doing a conversion, there are two ways to correct it, depending on whether you have completed the conversion or have only activated the "**from**" button. If you have completed the conversion, and find that it was wrong, you can reverse the conversion with the Last Stack [S] button. If you have only started the conversion and have only activated the "**from**" button, a second click in the "**from**" button will cancel the conversion. Conversions are a function of one variable and do not have any effect on the stack, except to convert the **x-reg** value.

Convert one square mile to square feet.

1 [return] [mile2] [feet2] **x-reg**=2.7878400e+7 (27,878,400 square feet)

On a very cold day the thermometer indicates -23.4° Celsius. What is the Fahrenheit temperature?

-23.4 [return] [Cel] [Fahr] **x-reg**=-1.0120000e+1 (-10.12° Fahrenheit)

A gas engine generates 346 horsepower. How many electrical watts does it generate?

346 [return] [hp(m)] [watt] x-reg=2.5801217e+5 (258,012.17 electric watts)

Compute the cubic inches of dirt in a square mile of dirt that is 1.5 feet thick?

1 [return] [mile2] [inch2] x-reg=4.0144896e+9 (square inches in a mile)

1.5 [return] [feet] [inch] x-reg=1.8000000e+1 (18 inches deep)

[MUL(x*y)] x-reg=7.2260813e+10 (That's a lot of dirt)

The abbreviations used in the Conversions Section are:

A	Angstrom	**amu**	Atomic Mass Unit
A.U.	Astronomical Unit	**BTU/h**	BTU/hour
cal/s	calories/second	**Cel**	Celsius
cm2	square centimeters	**cm3**	cubic centimeters
Fahr	Fahrenheit	**feet2**	square feet
f-p/h	foot pounds/hour	**gal**	gallon
feet3	cubic feet	**hp(e)**	electric horsepower
hp(m)	mechanical horsepower	**inch2**	square inch
inch3	cubic inch	**Kel**	Kelvin
kgram	kilogram	**km**	kilometers
lyear	light year	**m2**	square meter
m3	cubic meter	**mile2**	square mile
parsc	parsec		

As you can see, it is very easy to make unit conversions with **GSNumerics** . We will move to the Elements screen. You may do this by activating the [ELEMENTS] button.

Using The Elements Section

The elements screen is used to bring the Atomic Weight or Atomic Number of one of the 107 natural elements to the **x-reg**, to be used in calculations. The Atomic Weight is in atomic mass units (**amu**). The Element screen is shown in Figure 3.8.

Ac	Al	Am	Sb	Ar	As	At	Ba	ATOMIC WEIGHT (amu)																	Bk	Be	Bi	B	Br	Cd	Cs	Ca
Cf	C	Ce	Cl	Cr	Co	Cu	Cm	Dy	Es	Er	Eu	Fm	F	Fr	Gd	Ga	Ge	Au	Hf	He	Ho	H	In	I	Ir	Fe	Kr	La	Lr	Pb	Li
Lu	Mg	Mn	Md	Hg	Mo	Nd	Ne	Np	Ni	Nb	N	No	Os	O	Pd	P	Pt	Pu	Po	K	Pr	Pm	Pa	Ra	An	Re	Rh	Rb	Ru	Sm	Sc
Se	Si	Ag	Na	Sr	S	Ta	Tc	Te	Tb	Tl	Th	Tm	Sn	Ti	W	Uh	Up	Uq	Us	U	V	Xe	Yb	Y	Zn	Zr	CONVERSIONS				

Figure 3.8 Elements Screen

To recall an Atomic Weight, you simply activate the button representing the chemical element of interest. If you double-click the button, the Atomic Number will be pushed to the **x-reg**. These buttons are constants and will push the stack when activated. If you make a mistake, simple use the [S] to undo the error.

What is the weight of 1 atom of Cesium? Give the answer in atomic mass units.

[Cs] **x-reg**=1.3290500e+2 (**amu** weight of Cesium)

What is the Atomic Number of Cesium?

[Cs][2] **x-reg**= 5.5000000e+1 (**Atomic Weight** of Cesium = 55). [Cs][2] tells you to double-click on the button. Remember, to get the atomic number you must double-click the element button.

Memory Operations

The ability to store the results of calculations and recall them for later use is very important to the user. **GSNumerics** has twenty-six memory locations for storing and recalling numbers. The buttons used in memory operations are shown in Figure 3.9.

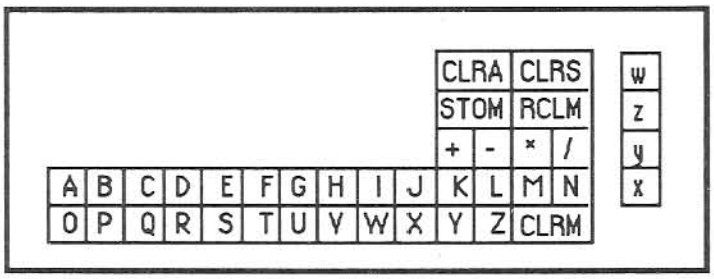

Figure 3.9 Memory Operation Buttons

The calculator memory operations extend beyond simple storage and recall of numbers. The user can store a number to an "**A**" to "**Z**" memory location from the **x-reg, y-reg, z-reg** or **w-reg**. A number stored in a memory location from "**A**" to "**Z**", can be recalled to the **x-reg, y-reg, z-reg** or **w-reg**. In addition to these simple storage/recall operations, the storage or recall can involve storage/recall math using the "+", "-", "*" or "/" operations.

A value stored in memory can be recalled from memory and in the same operation be added to, subtracted from, multiplied by or divided into a value in the **x-reg, y-reg, z-reg** or **w-reg** and the result placed in the **x-reg, y-reg, z-reg** or **w-reg**. The same operations can be applied to memory locations when storing a number from the **x-reg, y-reg, z-reg** or **w-reg** to a specific memory location.

The buttons designating the memory locations "A" to "Z" are the buttons [A] - [Z]. One of the buttons is always colored with a white background. This is the active memory storage location. It is the memory location that the next recall value will be taken from or the next stored value will be stored in, unless the user directs storage in another location as will be explained later in this chapter. The active memory location is always set to [A] when the program is first started. When a memory operation is performed on a specific memory location, the active memory storage is automatically incremented to the next location. When it increments to [Z], it automatically resets to [A].

Two buttons will clear the "A" to "Z" memory locations. They are the [CLRM] and [CLRA] buttons. These buttons set the memory locations to 0.0. The [CLRM] button clears only the "A" to "Z" memory locations, while the [CLRA] clears all storage and registers in the calculator. Both will reset the active memory location to [A]. When you wish to clear only the "A" to "Z" memory locations, use the [CLRM] and you will not reset the stack registers and complex storage locations.

In this section we will demonstrate how to store numbers from the **x-reg**, **y-reg**, **z-reg** or **w-reg** to the"A" to "Z" memory locations and recall numbers from the "A" to "Z" memory locations to the **x-reg**, **y-reg**, **z-reg** or **w-reg**. Prior to starting this demonstration, reset all locations with the [CLRA] button, and input the following:

4 [return] **x-reg**=4.0000000e+0
3 [return] **x-reg**=3.0000000e+0 **y-reg**=4.0000000e+0
2 [return] **x-reg**=2.0000000e+0 **y-reg**=3.0000000e+0 **z-reg**=4.0000000e+0
1 [return] **x-reg**=1.0000000e+0 **y-reg**=2.0000000e+0 **z-reg**=3.0000000e+0
w-reg=4.0000000e+0

We will now store the **x-reg** value in [A], the **y-reg** value in [B], the **z-reg** value in [C] and the **w-reg** value in [D].

[STOM] [STOM] This causes the **x-reg** value to be stored in the active memory location [A] and the active memory location to be increment to the next location which is [B]. Note that [B] now has a white background, indicating it is the active storage location.

[STOM] [y] [y] This causes the **y-reg** value to be stored in the active memory location, which is [B], andincrements the active memory location to [C]. Notice that [C] now has a white background. [y] is the blue button immediately to the left of the **y-reg**.

STOM z z — This causes the **y-reg** value to be stored in the active memory location, which is C, and increments the active memory location to D. Notice that D now has a white background. z is the blue button immediately to the left of the **z-reg**.

STOM w w — This causes the **y-reg** value to be stored in the active memory location, which is D, and increments the active memory location to E. Notice that D now has a white background. w is the blue button immediately to the left of the **w-reg**.

In reviewing the basic memory storage operations, notice that two clicks on the STOM button will always store the **x-reg** value into the active memory location and increment the active memory location. To store either the **y-reg, z-reg** or **w-reg** values into the active memory location, you must click STOM and then click the register button twice. The register buttons are the blue buttons immediately to the left of the registers.

If you wish to cancel the memory operation, before completing it, just click the mouse anywhere on the screen outside of any of the memory buttons shown in Figure 4.1. We now have the following numbers stored in the A through D memory locations:

A = 1.0
B = 2.0
C = 2.0
D = 4.0

We will now recall the stored numbers to the stack. Reset the active memory location to A and the stack registers by keying once on the A and CLRS button. You may set the active memory location to any of the "A" to "Z" locations, at any time by keying the memory location.

RCLM RCLM — **x-reg**=1.0000000e+0 **y-reg**=0.0000000e+0
z-reg=0.0000000e+0 **w-reg**=0.0000000e+0
This operation recalled the active memory value and pushed it to the **x-reg**.It also incremented the active memory location to B.

RCLM RCLM — **x-reg**=2.0000000e+0 **y-reg**=1.0000000e+0
z-reg=0.0000000e+0 **w-reg**=0.0000000e+0
This operation recalled the active memory value and pushed it to the **x-reg**. It also incremented the active memory location to C.

RCLM RCLM — **x-reg**=3.0000000e+0 **y-reg**=2.0000000e+0
z-reg=1.0000000e+0 **w-reg**=0.0000000e+0

This operation recalled the active memory value and pushed it to the **x-reg**. It also incremented the active memory location to [D].

[RCLM] [RCLM] **x-reg**=4.0000000e+0 **y-reg**=3.0000000e+0
z-reg=2.0000000e+0 **w-reg**=1.0000000e+0
This operation recalled the active memory value and pushed it to the **x-reg**. It also incremented the active memory location to [E].

In review, clicking twice on [RCLM] will recall the value stored in the active memory location to the **x-reg**, will push the stack and will advance the active memory location.

Now we will show how to bring the value in the active memory location into the **y-reg**, **z-reg** or **w-reg**. Set the active memory location to "A" by clicking once on [A].

[RCLM] [y] [y] **x-reg**=4.0000000e+0 **y-reg**=1.0000000e+0
z-reg=3.0000000e+0 **w-reg**=2.0000000e+0
This operation recalled the value in the active memory location into the **y-reg**, and pushed the **y-reg** value to the **z-reg** and the **z-reg** value into the **w-reg**. The **x-reg** was not effected. The active memory was incremented to "**B**".

[RCLM] [z] [z] **x-reg**=4.0000000e+0 **y-reg**=1.0000000e+0
z-reg=2.0000000e+0 **w-reg**=3.0000000e+0
This operation recalled the value in the active memory location into the **z-reg**, and pushed the **z-reg** value to the **w-reg.** The active memory was incremented to "C".

[RCLM] [w] [w] **x-reg**=4.0000000e+0 **y-reg**=1.0000000e+0
z-reg=2.0000000e+0 **w-reg**=3.0000000e+0
The last operation recalled the value in the active memory location into the **z-reg**, and pushed the **z-reg** to the **w-reg.** The active memory was incremented to "D".

You would not have much flexibility in memory storage/recall operations if you were always constrained to recalling the active memory location. You can store recall any memory location to any stack register or store any stack register to any memory location, at anytime, as demonstrated in the following:

[STOM] [D] This operation causes the value stored in the **x-reg** to be stored in memory location "D", and the active memory location to be set to "E". A [STOM] followed by a memory location button will always cause the **x-reg** to be saved in the specified memory location and the active memory location to be set to the next location.

[STOM] [z] [K] This operation causes the value stored in the **z-reg** to be stored in memory location "**K**", and the active memory location to be set to "**L**". A [STOM] followed by a stack register button and then a memory location button will always cause the stack location referenced by the stack register button to be saved in the specified memory location and the active memory location to be set to the next location.

[RCLM] [K] This operation causes the value stored in the "**K**" memory location to be recalled to the **x-reg**, pushing the stack. The active memory location is set to "**L**". A [RCLM] followed by a memory location button will recall the value in the referenced memory location to the **x-reg**, pushing the stack in he process.

[RCLM] [y] [L] This sequence causes the value stored in the "**L**" memory location to be recalled to the **y-reg**, pushing the **y-reg**, and **z-reg** values. The active memory location is set to "**M**". A [RCLM] button, followed by a stack register button and a memory location button, will cause the value in the specified memory location to be recalled to the specified stack register, pushing the specified register value and all the registers above.

This completes the basic stack and memory location storage and recall functions. Remember, you can cancel an operation before it is completed by keying of the memory buttons shown in Figure 4.1. You may set the active memory location by keying on the desired memory location button.

Using Memory Math

Suppose you were doing a calculation and wanted to add it to the value in memory location "**A**" and save the total. Memory math will let you do this very quickly and easily. We will first look at the memory save operations available. Prior to starting these examples, reset all locations with the [CLRA] button, and input the following:

4 [return] **x-reg**=4.0000000e+0
3 [return] **x-reg**=3.0000000e+0 **y-reg**=4.0000000e+0
2 [return] **x-reg**=2.0000000e+0 **y-reg**=3.0000000e+0 **z-reg**=4.0000000e+0
1 [return] **x-reg**=1.0000000e+0 **y-reg**=2.0000000e+0 **z-reg**=3.0000000e+0 **wreg**=4.0000000e+0

[STOM] [+] [+] This adds the **x-reg** value to the value in the active memory location "**A**". Memory location "**A**" now contains 1.0000. The active memory location is incremented to "**B**".

STOM + y y This adds the **y-reg** value to the value in the active memory location "**B**". Memory location "**B**" now contains 2.0000. The active memory location is incremented "**C**".

STOM - z z This subtracts the **z-reg** value from the value in the active memory location "**C**". Memory location "**C**" now contains -3.0000. The active memory location is incremented to "**D**".

STOM * w A This multiplies the value in memory location "**A**" by the **w-reg** value and set the active memory location to "**B**". Memory location "**A**" now contains a value of 4.0000. The sequence STOM w * A is equivalent.

To recap the above, a STOM followed by two similar memory math operators will store the **x-reg** into the active memory location, applying the math operator in the process and increment the active memory location. A STOM followed by a memory math operator and two similar stack register buttons will store the value of the specified register into the active memory location, applying the math operator in the process and will increment the active memory location.

A STOM followed by a memory math operator, a stack register button and a memory location button will store the specified stack register value into the specified memory location, applying the memory math operator and setting the active memory location to the memory location after the one specified. A STOM followed by a stack register button, a memory math operator and a memory location button is equivalent. Now we will examine the recall of memory values to the stack registers using memory math. Prior to starting this put the **x-reg**, **y-reg**, **z-reg**, and **w-reg** values into memory locations "**A**" through "**D**", and set the active memory location to "**A**", as show below:

STOM STOM	Puts **x-reg** = 1.0 value into memory location "A"
STOM y y	Puts **x-reg** = 2.0 value into memory location "B"
STOM z z	Puts **x-reg** = 3.0 value into memory location "C"
STOM w w	Puts **x-reg** = 4.0 value into memory location "D"
A	Sets active memory location to "A"

Now we can demonstrate the memory recall operations with math operators.

RCLM + + Recalls the value in the active memory location "**A**" and adds it to the **x-reg**, putting the result into the **x-reg** and increments the active memory location. The **x-reg** will now contain the value 2.000.

RCLM + y y — Recalls the value in the active memory location "B" and adds it to they-reg. The sum is put into the **y-reg**. The active memory location is incremented to "C". The stack is not pushed or pulled. The **y-reg** not contains 4.0000.

RCLM * w C — Recalls the value in memory location C and multiplies it by the value in the **w-reg**, placing the result into the **w-reg**. The active memory location is incremented to "B". The w and C operations can be reversed with the same result. The **w-reg** now contains the value 12.00.

In review, a RCLM followed by two similar memory math operators will recall the active memory location to the **x-reg** and advance the active memory location. The RCLM followed by a memory math operator and two similar stack register buttons will recall the active memory location to the indicated stack register performing the indicated math operation in the process. The active memory location will be advanced. Finally a RCLM followed by a memory math operator, a stack register button and then a memory location button will recall the selected memory location to the selected stack register, applying the selected math operator in the process. A RCLM followed by a stack register button, then a memory math operator and finally a memory location button is equivalent.

As you can see the memory recall operations with a memory math operator operate the same as the memory storage operations with a memory math operator used. You must be aware of the fact that a memory recall math operation does not push the stack like a simple memory recall operation.

Direct Function Entry

You have now learned how to solve problems and use the memory operations available with the **Scientific Calculator**. Up to this point, all problems have been solved using **RPN** and the individual buttons on the calculator screen. This has been a step by step procedure and you have observed the stack registers and now know how they operate. In this chapter we will show you how to work problems much more quickly and efficiently using the **Direct Function Entry** available in **GSNumerics**.

Direct Function Entry allows you to type an equation into the **input-reg**, press the return button and have the calculator analyze the equation and solve it. This is called parsing, or breaking apart, a mathematical statement. The program contains a powerful mathematical function parser to handle these operations for you. To demonstrate the **Direct Function Entry**, consider the problem 3.1. We will solve it with direct function entry by typing the following into the **input-reg** and pressing return:

(tpi+(ee^2-.85)^.5)/log(98) return **x-reg**=4.4396481e+0

The answer 4.4396481e+0 is pushed onto the stack and the answer is show in the **input-reg** with the format you have selected. The ten steps show in solving problem 3.1 have been replaced by entering one formula directly into the **input-reg** and pressing the return button! You will find that working problems with **Direct Function Entry** will speed up you calculations and will tend to lessen mistakes, since you are doing fewer actions.

When you pressed return, the parser solved the function by doing the following steps:

1. Called the constant **EE** (base of the natural logarithms) and squared it.
2. Subtracted 0.85 from the previous quantity and took the square root of the result.
3. Recalled the constant **tpi** (2 Pi) and added it to the quantity computed in step #2.
4. Computed the base 10 logarithm of 98.0 and divided it into the previous quantity.
5. Wrote the result to the **x-reg** and pushed the stack.

As you can see, this is a much faster way of solving problems, than entering them one step at a time. The program did exactly what you did in problem 3.1, but did it instantly when you entered the function. You can see the advantage of using **direct function entry** with the **Scientific Calculator**.

As you have used the **Scientific Calculator** you may have noticed many of the buttons have an asterisk following the label on the button. For instance, the LOG(x)* button has an asterisk following the label and the CDIV button does not have an asterisk. This tells you LOG(x)* can be used in **Direct Function Entry** and CDIV is not valid for **Direct Function Entry**.

^	Exponentiation (power)
-	Minus or Unary Minus
*	Multiplication
+	Addition
%	Modulo division
/	Division
ABS(X)*	Absolute Value
ACOS(X)*	Arc Cosine
ACOSH(X)*	Arc Hyperbolic Cosine
ACOT(X)*	Arc Cotangent
ACSC(X)*	Arc Cosecant
ALOG(X)*	Inverse Log Base 10

EE*	Natural Log Base Constant
EEXP(X)*	EE Exponentiation
EEXPS(X)*	Exponentiation (small)
FACT(X)*	Factorial
FRAC(X)*	Fractional Part
GD*	Grads to Degrees
GR*	Grads to Radians
HADD(+)*	Add Hours Format
HMSD*	Hours to Decimal Format
HSUB(-)*	Subtract Hours Format
INT(X)*	Integer Part
LN(X)*	Natural Log

ALTWO(X)*	Inverse Log Base 2
ASEC(X)*	Arc Secant
ASIN(X)*	Arc Sine
ASINH(X)*	Arc Hyperbolic Sine
ATAN(X)*	Arc Tangent
ATANH(X)*	Arc Hyperbolic Tangent
CBRT(X)*	Cube Root
CEIL(X)*	Ceiling
COS(X)*	Cosine
COSH(X)*	Hyperbolic Cosine
COT(X)*	Cotangent
CSC(X)*	Cosecant
CUBE(X)*	cube
DG*	Degrees to Grads
DHMS*	Decimal to Hours Format
DR*	Degrees to Radians

LNS(X)*	Natural Log (small)
LOG(X)*	Log Base 10
LTWO(X)*	Log Base 2
PI*	Pi Constant
RAND*	Random Number
RD*	Radians to Degrees
RG*	Radians to Grads
SEC(X)*	Secant
SIN(X)*	Sine
SINH(X)*	Hyperbolic Sine
SQR(X)*	Square
SQRT(X)*	Square Root
TAN(X)*	Tangent
TANH(X)*	Hyperbolic Tangent
TPI*	2Pi Constant

Figure 3.10 Valid Direct Function Entry Functions

Table 5.1 gives a complete list of functions and operators that are valid in **Direct Function Entry** equations. The functions are not case sensitive. In other words, **SIN(32.3)** will be interpreted by the parser in the same way that **sin(32.3)** is. We will work some simple examples to let you get use to **Direct Function Entry.**

Confirm the trig identity $\mathbf{Sin^2(x) + Cos^2(x) = 1}$ for x=12.45? **Scientific Calculator trig mode** in degrees.

sin(12.45)^2 + cos(12.45)^2 return **x-reg**=1.0000000e+0

The previous problem could also be entered in the following ways:

SQR(SIN(12.45))+SQR(COS(12.45)) return **x-reg**=1.0000000e+0

sin(12.45)*SiN(12.45)+cos(12.45)*COS(12.45) return **x-reg**=1.0000000e+0

One runner ran a marathon in 4.62 hours and another ran the marathon in 4.55 hours. Determine their average time of running the marathon, expressed in hh.mmss format?

dhms((4.62+4.55)/2) return **x-reg**=4.3506000e+0

Their average time was four hours, thirty-five minutes and six seconds.

Find the sum of 3!, 4! and 5!? 3! is the math symbol for 3 factorial (3! = 3* 2* 1).

fact(3)+fact(4)+fact(5) return **x-reg**=1.5000000e+2

Find the sum of .324 * Pi radians and 3.45°? Express the answer in grads.

dg(rd(.324*pi)+3.45) return **x-reg**=6.8633333e+1

In this example the parser first computed .324 * Pi and the converted it to degrees. This value was added to 3.45°. Finally, this term was converted from degrees to grads.

Compute the result of 15 divided by 4, using modulo division.

15%4 return **x-reg**=3.0000000e+0

Modulo division returns the remainder of a division. In this case 4 will go into 15 three times, with a remainder of 3. 15 divided by 5, using modulo division would equal 0, since there is no remainder. Try it and see.

15%5 return **x-reg**=0.0000000e+0

Multiply the fractional part of pi divided by 2 times the integer part of the tangent of 56 degrees. Make sure the calculator is in degrees mode, before doing this calculation.

frac(pi/2)*int(tan(56)) return **x-reg**=5.7079633e-1

Generate a random integer between 0 and 1000.

int(rand(x)*1000) return **x-reg**=9.9500000e+2

Operator Hierarchy In Direct Function Entry

When entering functions for parsing by the **Scientific Calculator**, you need to be aware of the order in which each operator will be evaluated. This is called the hierarchy of the operators. Understanding the hierarchy of the operators is critical to your use of Direct Function Entry. Figure 3.11 lists the hierarchy of the direct function entry operators.

1.	()	Items in parenthesis have highest hierarchy.
2.	**FUNCTION** ()	Functions (i.e. sin(x)) have next highest hierarchy.
3.	^ -	Exponentiation (i.e. 4^3) and unary minus have the next highest hierarchy and are solved left to right.
4.	* / %	Multiplication, division and modulo division have the next highest hierarchy and are solved left to right.
5.	+ -	Addition and subtraction have the lowest hierarchy and are solved left to right.

Figure 3.11 Direct Function Hierarchy

As an example, consider the following function:

-3^2 [return] **x-reg**=9.0000000e+0

This function evaluated to 9.0 because the unary operator '-' was applied to 3, prior to the number being squared. In this case the hierarchy of operations was: 1. Take the unary minus of 3, which is -3, and 2. Square this number for the final result. The unary minus has the same hierarchy as the exponentiation operator '^'. If the exponentiation operator had a higher hierarchy than the unary minus, the result would have been -9.0, since the square would have been done first, then the unary minus would have been taken. Now consider the following:

2+3*5-2*4+1 [return] **x-reg**=1.0000000e1

Why should this equal 10.0. Understanding how a function is parsed is critical to obtaining correct answers. If you meant to add 2 and 3, multiply them by 5, then subtract 2, then multiply by 4 and then add 1, you would have been surprised by the answer, because that is not what happened. To solve the equation that way it would have had to be written as follows:

((2+3)*5-2)*4+1 [return] **x-reg**=9.3000000e+1 (a big difference between 9 and 93)

These two examples again show how operator hierarchy enters into the picture when you solve equations using direct function entry. In the first example, the multiplication terms 3*5=15 and 2*4=8 were first solved by the parser, then it computed 2+15-8+1=9. The reason for this is that the '*' operator has a

higher hierarchy than either the '-' or '+' operator, and the parser will always solve the higher hierarchy parts first. After doing the two multiplication terms, the remaining terms only contained '-' and '+' operators. These two operators have the same hierarchy and the parser will always solve operators with equal hierarchy from left-to-right.

In the second case, the parser solved the problem by first computing 2+3=5, then 5*5 =25, then it computed 25-2=23, did the multiplication 4*23=92 and ended with 93+1=94. The reason for this is that parenthesis have a a higher hierarchy than any of the other operators, and the parser will faithfully obey the rule of working problems from the inside-out. This means that it will work the parts contained in parenthesis before applying any other operators. **GSNumerics** will not automatically solve problems for you. You must understand how the problem is to be set-up.

Consider a problem containing only multiplication and division operators, and how the parser solves such an equation:

36/6*9*3/5 return **x-reg**=3.2400000e+1

Since the operators '*' and '/' have the same hierarchy, the parser just started from the left and applied each operator in left-to-right order. The procedure was 1. 36/6=6. 2. 6*9=54, 3. 54*3=162 and finally 4. 162/5=32.4. Again, when the parser encounters operators with like hierarchy, it will always faithfully work the problem from left to right, applying the operators as it comes to them. If we really wanted to divide 36 by 6*9 and multiply this result by 3/5, we could have used parenthesis to override the standard hierarchy, as follows:

36/(6*9)*(3/5) return **x-reg**=4.0000000e-1

This time the parser solved the quantity in parenthesis 6*9=54 first and then computed 36/54=.66666. The term 3/5=.6 was then computed, since the parenthesis overide the '*', and finally the .6*.66666=.4 was computed. If you work a few problems, you will quickly learn how to write a direct function, considering the hierarchy of the operators. If you are in doubt, you can always force a particular part of a function to be solved first by enclosing it in parenthesis. You will find that the hierarchy of the operators is natural and easy to get used to, with a little practise.

Memory With Direct Function Entry

In the previous chapter we learned how to store/recall values from memory and how to use the memory math operators to include a mathematical operation in the store/recall operation. Since we don't want to lose the ability to do storage/recall operations when we are operating in direct function entry, this section will explain how to do the memory operations in **Direct Function Entry**.

To store the result of a direct function entry calculation in a specific "**A**" to "**Z**" register, you merely prefix the formula with the character of the memory location and an '=' sign. This will store the result in the specified register and will not effect the stack. When the function is executed, the active memory location will automatically be set to the next memory location after the one specified in the calculation. This will be a reminder to you of the last memory location you used, to store a result. Consider the following :

a=sin(32.4)	return	This stores the result of the function into memory location "**A**", sets the active memory location to "**B**" and does not change the stack. The **input-reg** will show the value of the **x-reg** after the calculation.
m=45.67	return	The value 45.67 is stored in the "**M**" memory location, the active memory location is set to "**N**". The stack is not changed.

To recall the value in a specific register, you merely reference the memory location in the direct function entry formula. The following examples will demonstrate this concept. For demonstration purposes put the following values in the registers indicated.

a=3	return	Puts 3.0 into the "**A**" register
b=5	return	Puts 5.0 into the "**B**" register
m=6	return	Puts 6.0 into the "**M**" register

Now we will demonstrate memory recall with direct function entry. The following simple problems will demonstrate this:

a	return	Recall the value in "**A**" to the **x-reg** and pushes the stack.
d=a*m	return	Recalls "**A**" value and "**M**" value and multiplies them, putting the result into the **"D"** memory location.
sin(B)^a	return	Recall "**B**" value and takes sine, then recalls "**A**" value and raises sin(B) to t he power of the value stored in "**B**". This is the same as typing sin(5)^3. This will push the value of 6.6204579e-4, if the calculator is set to degrees mode.

a=a^2	return	This operation will replace a=3 with a=3*3=9. The active memory register will be set to "**B**". The stack is not pushed.
b=b+2*b	return	Puts 15.0 into "**B**" and advances active memory register to "**C**".

As you can see, it is very simple to save numbers to and recall numbers from the memory storage locations, "**A**" to "**Z**", when using direct function entry. This is a very important and powerful part of **GSNumerics**, that you will use many times as you learn how to use the program.

Solving Large Problems

You are limited to solving problems that have no more than 49 characters, because the **input-reg** is only long enough to accommodate 49 characters. This will not limit your ability to solve these large problems quickly and easily. The key is to break the problems into smaller sections, and store the results.

$$\frac{tpi*\sin(32.4)}{2*\log(23)} + \sqrt[3]{\sin(23.5)*\cos(45.6)} - \sqrt[5]{3.5-\sec(45)*\tan(2.6)*\ln(23)}$$

Just solve this one piece at at time, save the results in memory and then make a final calculation, using the stored values as shown below:

a=tpi*sin(32.4)/(2*log(23))	return	Solve first section and store in "**A**"
b=cbrt(sin(23.5)*cos(45.6))	return	Solve next section and store in "**B**"
c=(3.5-sec(45)*tan(2.6)*ln(23))^(1/5)	return	Solve last section and store in "**C**"
a+b-c	return	Push result=6.2001193e-1 to **x-reg**.

You can make the equation shorter by setting the terms in parenthesis equal to a memory location and using the memory variable in the function. The equation would have to be very long, if you need to go to this extent to get a solution. An example is shown below, using the previous problem.

f=tpi*sin(32.4)	return	
g=2*log(23)	return	
a=f/g	return	This puts the result of the first term in memory "**A**".

Some Practice Problems

Before leaving this section we will solve some additional problems.Given the regression formula **y = 3x - 15**, solve for **x=4**, **x=7.5** and **x=3.5**. (form y = a*x + b)

a=3	return	The easiest way is to store constants first.

b=-15	return	Store the b constant and then solve for each x.
a*4+b	return	Pushes -3.0000000e+0 into the **x-reg**.
a*7.5+b	return	Pushes 7.5000000e+0 into the **x-reg**.
a*3.5+b	return	Pushes -4.5000000e+0 into the **x-reg**. Storing the constants in "**A**" and "**B**" , speeded things up.

Compute the total cost of two dozen eggs costing $0.98 per dozen, 24 cans of soft drink costing $0.35 each, 2 loaves of bread costing $0.77 per loaf and one can of coffee costing $6.85. The coffee and soft drink purchases have a 5.0% state tax.

2*.98+2*.77+1.05*(24*.35+6.85) return — Total cost, pushed to **x-reg** is 1.9512500e+1, or $19.51. Notice how we needed to use the parenthesis to total 24*.35 + 6.85 prior to multiplying by the sales tax multiplier.

A new home will have five rooms whose measurements are as follows: kitchen 10'x12', living room 15'x23', bedroom 12'x12', bedroom, 11'x13', bathroom 7'x9'. The cost of building the home will be $48.95 per square foot. Compute the total cost of the completed house.

48.95*(10*12+15*23+12*12+11*13+7*9) return — Total cost of home is 3.9894250e+4 pushed to the **x-reg**. ($39,894.25)

The following formula computes the volume of the Frustum of a Right Cone:

$$\text{Volume} = \frac{pi*v}{3}*(R_1^2+R_1*R_2+R_2^2)$$

Compute the volume given the following: $R_1 = 4.6$, $R_2 = 2.3$ and $v = 7$.

pi*7/3*(4.6^2+4.6*2.3+2.3^2) return — Volume equal to 2.7144408e+2 pushed to **x-reg**. 271.44 in **input-reg**.

Solve the polynomial $-x^4+3.5x^3-2x^2-3x+4.5$ for $x = 1.1$.

x=1.1 return — Put x value in as a constant

-(x^4)+3.5*x^3-2*x^2-3*x+4.5 return — Notice the parenthesis around the first term. If you don't do this, the unary minus will be applied first, resulting in taking -1.1 to the fourth power. This would be the same as having a +1 coefficient on the fourth power term, and you would actually be solving the polynomial $x^4+3.5x^3-2x^2-3x+4.5$. The answer, pushed to the **x-reg** is 1.9744000e+0.

Verify the trig identity: **sin(a+b) + sin(a-b) = 2*sin(a)*cos(b)** **for a=32.4° and b=55.1°.**
If this is true then: **sin(a+b) + sin(a-b) - 2*sin(a)*cos(b) = 0.0**

a=32.4	return	Input first constant
b=55.1	return	Input second constant
sin(a+b) + sin(a-b) - 2*sin(a)*cos(b)	return	Answer: 0.0000000e+0 pushed to x-reg.

This completes our review of direct function entry with **GSNumerics.** As you solve more problems, you will find it is very easy to solve large formulas using direct function entry. Remember, the problem must be set up right, or you will get incorrect answers. The hierarchy is absolutely important: 1. Parenthesis, 2. Functions 3. Unary minus and exponeniation, 4. Multiplication, Division and Modulo Division and 5. Addition and Subtraction.

Complex Numbers

Many mathematical problems require the use of complex numbers. Complex numbers consist of two parts, a real number and a pure imaginary number. An imaginary number represents the square root of a negative number. The definition of a pure imaginary number is shown in Figure 3.12. Complex numbers are a very important part of the study of mathematics, and are used in many fields of science and engineering, especially in the field of Electrical Engineering. They are very easy to use, if you will only take the time to understand the basic concepts.

Pure Imaginary Number

For any real number **a**, with **a < 0**:

$$\sqrt{a} = \sqrt{-1} * \sqrt{a} = ia \quad \textbf{Where } i = \sqrt{-1}$$

"**i**" represents the square root of -1, which is an imaginary number.

Figure 13.12 Definition of an Pure Imaginary Number

As you can see imaginary numbers were created to handle the case of taking the square root of a negative number, since either a positive number or a negative number results in a positive number when it is squared. We can add, subtract, multiply and divide pure imaginary numbers, just like other numbers, as long as we observe one special rule in doing multiplication. The multiplication of two pure complex numbers will always result in the negative of the product of the real numbers. The rules for complex number addition, subtraction and division are shown in Figure 3.13.

For any two pure imaginary numbers **ia** and **ib**:

$ia + ib = i * (a+b)$

$ia - ib = i * (a-b)$

$ia * ib = i * i * a * b = \sqrt{-1} * \sqrt{-1} * a * b = -1 * (a * b) = -(a * b)$

$$\frac{ia}{ib} = \frac{i}{i} * \frac{a}{b} = \frac{\sqrt{-1}}{\sqrt{-1}} * \frac{a}{b} = \frac{a}{b}$$

"**i**" represents the square root of -1, which is an imaginary number.

Figure 3.13 Mathematic operations on Pure Imaginary Number

Most imaginary number calculations involve a combination of pure imaginary numbers and real numbers, forming a complex number. The definition of a complex number is given in Figure 3.14

A **complex number** is any number that can be expressed in the form:

a + ib

Where **a** and **b** are real numbers and $i = \sqrt{-1}$. **a** is called the real part of the complex number and **b** is referred to as the imaginary part of the number.

Figure 3.14 Definition of Complex Number

Examples of complex numbers are:

3.5 + i27.5

4.8 + i0

-10 + i7

0 -i4

Before covering complex number math, we will review the two forms of writing complex numbers.

Polar and Rectangular Forms

There are two commonly used ways of expressing complex numbers: Rectangular and Polar form.Up to this point we have expressed complex numbers in the rectangular format a + ib. Figure 13.15 shows the conversion of a rectangular form complex number to a polar form complex number.

The complex number **a + ib** is expressed in the polar form **r∠ θ** as follows:

The magnitude $\mathbf{r} = \sqrt{\mathbf{a}^2 + \mathbf{b}^2}$

The angle θ = **arc tangent**$(\frac{\mathbf{b}}{\mathbf{a}})$

The polar format expresses a complex number in terms of a **magnitude (r)** and an **angle** (θ).

Figure 3.15 Rectangular to Polar conversion

To convert a number from polar format to rectangular format, follow the steps shown in Figure 3.16.

The complex number **r∠ θ** is expressed in rectangular form **a + ib** as follows:

The real part **a = r * cosine**(θ)

The imaginary part **b = r * sine**(θ)

a + ib = r * cosine(θ) **+ i(r * sine**(θ))

The rectangular format expresses a complex number in terms of a **real part (a)** and an **imaginary part (b)**.

FIGURE 3.16 Polar to Rectangular conversion

A graphical comparison of the same complex number, graphed in rectangular form and polar form is shown in Figure 3.17.

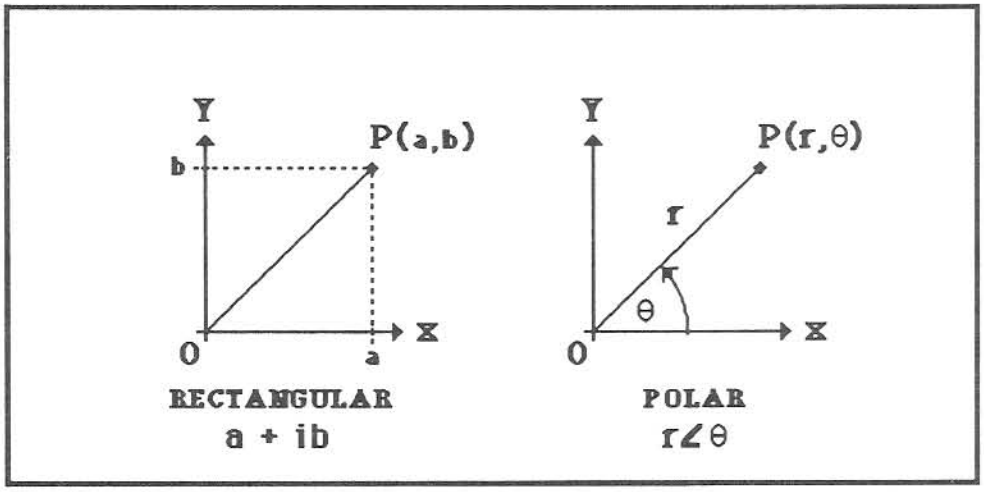

FIGURE 3.17 Graphs of Rectangular and Polar forms of complex numbers

You can see from examining Figure 3.17 that a complex number expressed in rectangular form is exactly the same number that it is when it is expressed in polar form. The main reason for expressing the same complex number in two different ways is to make the addition, subtraction, multiplication and division of complex numbers easier. When adding or subtracting two complex numbers, it is easier to express them in rectangular form. For multiplying or dividing, the polar form is easier.

Before explaining the use of **GSNumerics** for solving complex number problems, we will explain and review how to add, subtract, divide and multiply complex numbers. The procedures are shown in Figure 3.18.

Addition

$$(a + ib) + (c+id) = (a + c) + i(b + d)$$

Subtraction

$$(a + ib) - (c+id) = (a - c) + i(b - d)$$

Multiplication

$$(a + ib) * (c+id) = r_1 \angle \theta_1 * r_2 \angle \theta_2 = (r_1 * r_2) \angle (\theta_1 + \theta_2)$$

Division

$$\frac{(a + ib)}{(c + id)} = \frac{r_1 \angle \theta_1}{r_2 \angle \theta_2} = \frac{r_1}{r_2} \angle (\theta_1 - \theta_2)$$

FIGURE 3.18 Mathematic operations on Complex Numbers

The main reason that a lot of people don't like to work complex number problems is the amount of work involved in converting back and forth between rectangular and polar forms, depending on whether you are ready to add, subtract, multiply or divide two complex numbers. This is very time consuming and creates more opportunities for mistakes. Consider the following complex number problem:

$$\frac{(5 \angle 1.5) + (7 \angle 2.3)}{(3 \angle 1.8) + (1 \angle 0.3)}$$ **The angles are expressed in radians.**

To solve this problem you would first have to convert the two complex numbers in the numerator to their rectangular form, add the real and imaginary parts, convert the sum to polar form and save the result. You would then convert the two complex numbers in the denominator to their rectangular form, add the real and imaginary parts, and convert to polar format. At this point you would divide the magnitude of the numerator term by the magnitude of the denominator term and subtract the angle of the denominator from

the angle of the numerator. The answer would then be in polar form. As you are about to see **GSNumerics** lets you solve complex problems without all of this work!

Entering Complex Numbers

Complex numbers are always entered in two parts. When a rectangular form complex number is entered, the **real part** is pushed to the **x-reg** and the **imaginary part** is pushed to the **y-reg**. In other words, two stack pushes are involved with complex number entry. First you enter the **imaginary part**, and it is pushed onto the **x-reg**, then you enter the **real part** and it is pushed onto the **x-reg**. If you enter a complex number in polar format, the **angle** must be entered first, with the **magnitude** being the second entry. This results in the **angle** being in the **y-reg** and the **magnitude** will being in the **x-reg**. You will always enter complex numbers using the **Complex Operators**, explained later in this chapter.

When doing complex math **GSNumerics** always expects the numbers to be in rectangular format. It is **very important** that you convert a number entered in polar format to rectangular format, prior to doing any complex number calculations. **If the numbers are left in polar format and complex number calculations are done, the answers will be wrong!!!** You may convert a polar form complex number in the x-reg (magnitude) and the y-reg (angle) to a rectangular form complex number with the **POLR-RECT** function. You may convert a rectangular form complex number in the x-reg (real part) and y-reg (imaginary part) to a polar form complex number with the **RECT-POLR** function.

At this point, we have outlined a procedure for entering complex numbers. For a rectangular format number you would enter the imaginary part first, and then enter the real part. This requires two keyboard operations and pressing **return** twice, for a total of four steps. To input a polar form complex number, you would first enter the angle, then enter the magnitude and convert to rectangular form with **POLR-RECT**. This requires five steps. Although you can enter complex numbers with this procedure, it is lengthy and you will be prone to making mistakes. For this reason **GSNumerics** provides the complex entry operators '**r[**' and '**p[**', to make it easier to enter complex numbers.

The rectangular complex entry operator **r[** takes a keyboard entry **r[a,b** and pushes it onto the stack in the proper manner, with **b** being pushed into the **y-reg** and **a** being pushed in the **x-reg**. This allows you to enter a rectangular form complex number in one operation.

The rectangular complex entry operator **p[** takes a keyboard entry **p[r,θ**, with **r** being the **magnitude** and θ being the **angle**, converts it to the rectangular form, and pushes the **imaginary** part to the **y-reg** and **real part** to the **x-reg**.

With the automatic conversion from polar form to rectangular form being handled by the polar operator, you can enter a polar form complex number in one operation and have it entered in the proper form for complex math operations.

In order for the complex stack to operate properly, you must use the complex operators, **p[r,θ** and **r[a,b** , for all complex number entry into the **Scientific Calculator**. Just putting the complex numbers into the **x-reg** and **y-reg**, even if they are properly converted to rectangular form, will not push the complex stack. These operators have three purposes: 1. To make your job easier, when you are entering complex numbers, 2. To make the automatic polar to rectangular conversion for you, and 3. to properly push the complex stack, when a complex number is entered from the keyboard. You should routinely use the two complex number operators, when you are entering complex numbers into the **Scientific Calculator**.

Polar Entry Operator

The polar entry operator **p[r,θ** firsts makes the following conversion:

Real = p * cos(θ)

Imag = p * sin(θ)

It then pushes **Imag** to the **y-reg** and **Real** to the **x-reg**. The previous **x-reg** is pushed to the **z-reg** and the previous **y-reg** is pushed to the **w-reg**

Rectangular Entry Operator

The rectangular operator **r[a,b** pushes **b** to the **y-reg** and **a** to the **x-reg**.The previous **x-reg** is pushed to the **z-reg** and the previous **y-reg** is pushed to the **w-reg**

Figure 3.19 Complex number entry operators

Now we will give some examples of entering complex numbers. Since the only reason you will probably want to enter numbers, is to perform complex number math, we will always enter them so they are in the rectangular form.

Prior to starting this exercise, clear the stack registers by keying on **CLRS** and put the calculator in degrees mode with **DM**.

Enter the complex number **5∠35.6°** using the polar entry operator.

p[5,35.6 **return** **x-reg**=4.0655038e+0 **y-reg**=2.9106149e+0 (converts to ectangular form).

Enter the complex number **3 + i4,** using the rectangular entry operator.

r[3,4 (return) **x-reg**=3.0000000e+0 **y-reg**=4.0000000e+0
z-reg=4.0655038e+0 **w-reg**=2.9106149e+0
Notice that the complex number that was in the **x-reg** and **y-reg** has been pushed to the **z-reg** and **w-reg**.

Enter the complex number **0 + i**, using the polar entry operator.

p[1,90 (return) **x-reg**=0.0000000e+0 **y-reg**=1.0000000r+0 (converts to rectangular form).
z-reg=3.0000000e+0 **w-reg**=4.0000000e+0
Notice that the complex number that was in the **x-reg** and **y-reg** has been pushed to the **z-reg** and **w-reg**.

The Complex Number Stack

The only difference between complex number arithmetic and real number arithmetic is that you have to deal with twice as many numbers with complex arithmetic. Each number has two parts, that have to be treated separately when adding, subtracting, multiplying or dividing. The problems are not harder, they just involve more numbers.

When we worked problems involving only real numbers we had four stack registers: **x-reg, y-reg, z-reg** and **w-reg,** to work with. Using **RPN** entry in conjunction with the four stack registers, we found that we could always work a problem, no matter how long it was, without any problem. How can we work long complex number problems with only four stack registers? The answer is we can't. We need twice as many stack registers since a complex numbers always consist of two parts. There are in fact eight stack registers in **GSNumerics,** but four of them are hidden. The hidden registers are the **a.reg, b.reg, c.reg** and **d.reg**. Think of these registers as being tacked to the top of the **w-reg**, in the order listed. When a complex number is entered or recalled from complex storage, the stack is pushed twice. When a complex function of two variables is executed, the stack is pulled twice.

You may roll the complex stack up and down by double clicking on the **UP** or **DN** buttons. A double click will always be indicated by **DN** [2]. The superscript 2 in brackets tells you to double click on the indicated button. A double click is two clicks in a short time, as opposed to one click. Clear the stack registers with **CLRC** and set the calculator to degrees mode with **DM**.

Enter the following complex numbers, using the appropriate complex entry operator:

3-i4 2∠24° 4∠55° 5+i2.1

r[3,-4	**return**	**x-reg**=3.0000000e+0 **y-reg**=-4.0000000e+0
		z-reg=0.0000000e+0 **w-reg**=0.0000000e+0
		a-reg=0.0000000e+0 **b-reg**=0.0000000e+0 (not visible)
		c-reg=0.0000000e+0 **d-reg**=0.0000000e+0 (not visible)
p[2,24	**return**	**x-reg**=1.8270909e+0 **y-reg**=8.1347329e-1
		z-reg=3.0000000e+0 **w-reg**=-4.0000000e+0
		a-reg=0.0000000e+0 **b-reg**=0.0000000e+0 (not visible)
		c-reg=0.0000000e+0 **d-reg**=0.0000000e+0 (not visible)
p[4,55	**return**	**x-reg**=2.2943057e+0 **y-reg**=3.2766082e+0
		z-reg=1.8270909e+0 **w-reg**=8.1347329e-1
		a-reg=3.0000000e+0 **b-reg**=-4.0000000e+0 (not visible)
		c-reg=0.0000000e+0 **d-reg**=0.0000000e+0 (not visible)
r[5,2.1	**return**	**x-reg**=5.0000000e+0 **y-reg**=2.1000000e+0
		z-reg=2.2943057e+0 **w-reg**=3.2766082e+0
		a-reg=1.8270909e+0 **b-reg**=8.1347329e-1 (not visible)
		c-reg=3.0000000e+0 **d-reg**=-4.0000000e+0 (not visible)

The complex entry operators push the complex numbers onto the stack in pairs and actually push complex numbers to two hidden stacks. You can confirm the numbers are there by double clicking on the **UP** or **DN** button. Let's try it and see what happens.

DN [2]	**x-reg**=2.2943057e+0 **y-reg**=3.2766082e+0
	z-reg=1.8270909e+0 **w-reg**=8.1347329e-1
DN [2]	**x-reg**=1.8270909e+0 **y-reg**=8.1347329e-1
	z-reg=3.0000000e+0 **w-reg**=-4.0000000e+0
DN [2]	**x-reg**=3.0000000e+0 **y-reg**=-4.0000000e+0
	z-reg=5.0000000e+0 **w-reg**=2.1000000e+0
DN [2]	**x-reg**=5.0000000e+0 **y-reg**=2.1000000e+0
	z-reg=2.2943057e+0 **w-reg**=3.2766082e+0

See, four double clicks on **DN** brought us right back to where we were. Double clicking the **UP** button four times would have the same result, but it would roll the complex number stack in the opposite direction.

You should be aware that real number operations do not have any effect on the hidden complex number registers **a.reg, b.reg, c.reg** and **d.reg**. If you have two complex numbers stored in these registers, they will stay there, until you initiate a complex number operation or clear the stack. You can store two complex numbers in the hidden registers, do a series of real number operations, and then recall the complex numbers with [UP] [2] or [DN] [2]. The hidden registers,**a.reg, b.reg, c.reg** and **d.reg**, are saved automatically into the last stack and the [S] button can be used to recover from mistakes, just like it was used in real number operations.

You can enter complex numbers one number at a time, as if they were real numbers. You would do this by entering the imaginary part first, then you would enter the real part. If you were entering a polar form complex number, you would enter the angle first, then enter the magnitude and finally convert to rectangular format with [POLR-RECT]. This is not recommended as the complex stack operations will not be initiated!!! **Always use the complex entry operators!!!**

Complex Number Operations

There are fourteen buttons on the Scientific Calculator that can be used in complex number operations. These buttons and their function is shown in the following table.

[POLR-RECT]	Converts a polar form complex number in the **x-reg** and **y-reg** into a rectangular form complex number, putting the result in the **x-reg** and **y-reg**. Does not push or pull the complex stack.
[RECT-POLR]	Converts a rectangular form complex number in the **x-reg** and **y-reg** into a polar complex number, putting the result in the **x-reg** and **y-reg**. Does not push or pull the complex stack.
[1/x] [2]	A double click will take the reciprocal of the rectangular form complex number in the **x-reg** and **y-reg**, placing the result in the **x-reg** and **y-reg**. Does not push or pull the complex stack.
[CHSGN(x)] [2]	A double click will take the negative of the rectangular form complex number in the **x-reg** and **y-reg**, placing the result in the **x-reg** and **y-reg**. Does not push or pull the complex stack.
[XCHG(x-y)] [2]	A double click will exchange two complex numbers, one in the **x-reg** and **y-reg**, the other in the **z-reg** and **w-reg**. Does not push or pull the complex stack.

SQR(z)* [2]	A double click will square the rectangular form complex number in the **x-reg** and **y-reg**, placing the result in the **x-reg** and **y-reg**. Does not push or pull the complex stack.
SQRT(z)* [2]	A double click will take the square root of the rectangular form complex number in the **x-reg** and **y-reg**, placing the result in the **x-reg** and **y-reg**. Does not push or pull the complex stack.
CADD	Adds two rectangular form complex numbers, one in the **x-reg** and **y-reg**, the other in the **z-reg** and **w-reg**, returning the result in the **x-reg** and **y-reg**. This is a function of two complex variables and will do a complex pull on the stack, bringing the **a-reg** to the **z-reg**, the **b-reg** to the **w-reg**, the **c-reg** to the **a-reg** and the **d-reg** to the **b-reg**.
CSUB	Subtracts a rectangular form complex number in the **x-reg** and **y-reg** from a rectangular form complex number in the **z-reg** and **w-reg**, returning the result in the **x-reg** and **y-reg**. This is a function of two complex variables and will do a complex pull on the stack, bringing the **a-reg** to the **z-reg**, the **b-reg** to the **w-reg**, the **c-reg** to the **a-reg** and the **d-reg** to the **b-reg**.
CMUL	Multiplies two rectangular form complex numbers, one in the **x-reg** and **y-reg**, the other in the **z-reg** and **w-reg**, returning the result in the **x-reg** and **y-reg**. This is a function of two complex variables and will do a complex pull on the stack, bringing the **a-reg** to the **z-reg**, the **b-reg** to the **w-reg**, the **c-reg** to the **a-reg** and the **d-reg** to the **b-reg**.
CDIV	Divides a rectangular form complex number in the **x-reg** and **y-reg** into a rectangular form complex number in the **z-reg** and **w-reg**, returning the result in the **x-reg** and **y-reg**. This is a function of two complex variables and will do a complex pull on the stack, bringing the **a-reg** to the **z-reg**, the **b-reg** to the **w-reg**, the **c-reg** to the **a-reg** and the **d-reg** to the **b-reg**.
STOC	Initiates a complex storage operation from either the **x-reg** and **y-reg**, or from the **z-reg** and **w-reg**, to the **CSTORE** registers.
RCLC	Initiates a complex recall operation to either the **x-reg** and **y-reg**, or the **z-reg** and **w-reg**, to the **CSTORE** registers.
CLRC	Clears the **CSTORE** registers, and **a.reg**, **b.reg**, **c.reg** and **d.reg**, setting their values to 0.0;
UP [2]	Performs a complex number roll up on the complex number stack.
DN [2]	Performs a complex number roll down on the complex number stack.

Figure 3.20 Complex Button Functions

Complex Number Arithmetic

We will now solve some complex number problems, to show you how the complex number operators and the complex number stack, will allow you to solve very involved complex number problems easily. Make sure the calculator is in degrees mode, clear the complex number stack and the stack registers. As we work through these problems, pay particular attention to the operation of the stack, as you input the complex numbers and initiate the complex operators.

Add the complex numbers **3.5∠46.5°** and **2.6∠13.45°**.

p[3.5,46.5	return	**x-reg**=2.4092410e+0 **y-reg**=2.5388103e+0
		z-reg=0.0000000e+0 **w-reg**=0.0000000e+0
p[2.6,13.45	return	**x-reg**=2.5286905e+0 **y-reg**=6.0475148e-1
		z-reg=2.4092410e+0 **w-reg**=2.5388103e+0
	CADD	**x-reg**=4.9379315e+0 **y-reg**=3.1435618e+0
		z-reg=0.0000000e+0 **w-reg**=0.0000000e+0

Convert the previous answer to polar form and then convert back to rectangular form?

RECT-POLR	**x-reg**=5.8536440e+0 **y-reg**=3.2481446e+1
	z-reg=0.0000000e+0 **w-reg**=0.0000000e+0
	This is approximately **5.854∠32.48°**.
POLR-RECT	**x-reg**=4.9379315e+0 **y-reg**=3.1435618e+0
	z-reg=0.0000000e+0 **w-reg**=0.0000000e+0

Notice how the complex number operators **p[a,b** and **r[a,b** take care of converting the number to rectangular form and how they initiate a complex push to the complex stack registers. For these reasons, always use the complex operators **p[a,b** and **r[a,b** to enter complex numbers. If you don't, you will lose the complex stack, since entering the individual parts of a complex number individually into the **x-reg** and **y-reg**, does not initiate a complex push on the stack.

Solve the complex problem:

$$\frac{(3 - i4 * 6 + i7)}{(4 - i * 2 + i2)}$$

r[3,-4	return	**x-reg**=3.0000000e+0 **y-reg**=-4.0000000e+0
		z-reg=4.9379315e+0 **w-reg**=3.1435618e+0
r[6,7	return	**x-reg**=6.0000000e+0 **y-reg**=7.0000000e+0
		z-reg=3.0000000e+0 **w-reg**=-4.0000000e+0

	CMUL	x-reg=4.6000000e+1 y-reg=-3.0000000e+0
		z-reg=4.9379315e+0 w-reg=3.1435618e+0
r[4,-1	return	x-reg=4.0000000e+0 y-reg=-1.0000000e+0
		z-reg=4.6000000e+1 w-reg=-3.0000000e+0
r[2,2	return	x-reg=2.0000000e+0 y-reg=2.0000000e+0
		z-reg=4.0000000e+0 w-reg=-1.0000000e+0
	CMUL	x-reg=1.0000000e+1 y-reg=6.0000000e+0
		z-reg=4.6000000e+1 w-reg=-3.0000000e+0
	CDIV	x-reg=3.2500000e+0 y-reg=-2.2500000e+0
		z-reg=4.9379315e+0 w-reg=3.1435618e+0

This problem was solved by making four keyboard entries and keying on three complex operator buttons, a total of seven steps. If you were to work this by hand, you would make eight keyboard entries, convert two numbers from rectangular to polar format and two multiplications plus one divide. Our last problem will be a very long and complicated complex number problem. This will give you practice in solving multiple term problems, and will help you to understand the process.

Prior to starting work on the next problem, make sure the **trig mode** is in degrees and clear all registers with CLRA. This button will clear all registers, including the complex registers **a.reg, b.reg, c.reg** and **d.reg,** setting them equal to 0.0. As you work the problem, observe the action of the registers.

It is important to work complex number problems from the inside out, just like we worked real numbers problems. You will generally work the numerators first, starting with the computations inside the parenthesis. We demonstrate this with the following problem. This is a very long problem. It is included so you will thoroughly understand the operation of the complex stack and complex entry operators. Solve the following problem:

$$\frac{(2 - i3 * (7\angle 60^\circ + 8\angle 42^\circ) + 3 + i2.4 * (1.3\angle 35.4^\circ - 16\angle 23^\circ))}{(-1 - i1 * (3.9\angle 10.4^\circ + 1.3\angle -8.5^\circ))} + (6\angle 12^\circ + 70^\circ)$$

p[7,60	return	x-reg=3.5000000e+0 y-reg=6.0621778e+0
		z-reg=0.0000000e+0 w-reg=0.0000000e+0
p[8,42	return	x-reg=5.9451586e+0 y-reg=5.3530449e+0
		z-reg=3.5000000e+0 w-reg=6.0621778e+0
	CADD	x-reg=9.4451586e+0 y-reg=1.1415223e+1
		z-reg=0.0000000e+0 w-reg=0.0000000e+0

r[2,-3	return	x-reg=2.0000000e+0 y-reg=-3.0000000e+0
		z-reg=9.4451586e+0 w-reg=1.1415223e+1
	CMUL	x-reg=5.3135985e+1 y-reg=-5.5050305e+0
		z-reg=0.0000000e+0 w-reg=0.0000000e+0
p[1.3,35.4	return	x-reg=1.0596661e+0 y-reg=7.5306552e-1
		z-reg=5.3135985e+1 w-reg=-5.5050305e+0
p[16,23	return	x-reg=1.4728078e+1 y-reg=6.2516981e+0
		z-reg=1.0596661e+0 w-reg=7.5306552e-1
	CSUB	x-reg=-1.3668412e+1 y-reg=-5.4986325e+0
		z-reg=5.3135985e+1 w-reg=-5.5050305e+0
r[3,2.4	return	x-reg=3.0000000e+0 y-reg=2.4000000e+0
		z-reg=-1.3668412e+1 w-reg=-5.4986325e+0
	CMUL	x-reg=-2.7808516e+1 y-reg=-4.9300085e+1
		z-reg=5.3135985e+1 w-reg=-5.5050305e+0
	CADD	x-reg=2.5327469e+1 y-reg=-5.4805116e+1
		z-reg=0.0000000e+0 w-reg=0.0000000e+0
p[3.9,10.4	return	x-reg=3.8359287e+0 y-reg=7.0402467e-1
		z-reg=2.5327469e+1 w-reg=-5.4805116e+1
p[1.3,-8.5	return	x-reg=1.2857206e+0 y-reg=-1.9215223e-1
		z-reg=3.8359287e+0 w-reg=7.0402467e-1
	CADD	x-reg=5.1216494e+0 y-reg=5.1187243e-1
		z-reg=2.5327469e+1 w-reg=-5.4805116e+1
r[-1,-1	return	x-reg=-1.0000000e+0 y-reg=-1.0000000e+0
		z-reg=5.1216494e+0 w-reg=5.1187243e-1
	CMUL	x-reg=-4.6097769e+0 y-reg=-5.6335218e-0
		z-reg=2.5327469e+1 w-reg=-5.4805116e+1
	CDIV	x-reg=3.6234028e+0 y-reg=7.4607943e+0
		z-reg=0.0000000e+0 w-reg=0.0000000e+0
p[6,12	return	x-reg=5.8688856e+0 y-reg=1.2474701e+0
		z-reg=3.6234028e+0 w-reg=7.4607943e+0
p[7,0	return	x-reg=7.0000000e+0 y-reg=0.0000000e+0
		z-reg=5.8688856e+0 w-reg=1.2474701e+0
	CADD	x-reg=1.2868886e+1 y-reg=1.2474701e+0
		z-reg=3.6234028e+0 w-reg=7.4607943e+0
	CADD	x-reg=1.6492288e+1 y-reg=8.7082644e+0
		z-reg=0.0000000e+0 w-reg=0.0000000e+0

The answer is approximately **1.87∠27.84°** in polar format. This problem was worked with eleven keyboard entry steps and ten mouse clicks on complex operator buttons, for a total of twenty-one steps. If you work this problem manually, you will find it will take a lot of steps to get the solution. You will also have a lot more opportunity to make a mistake. We will leave it as an exercise for the reader to work the problem manually, to see if **GSNumerics** actually saved you time!!!

Complex Number Memory Operations

The complex number storage and recall operations are very similar to the real number operations discussed previously. The main difference is that there is only one memory location to store complex numbers into and they can only be recalled to two memory locations. The memory locations for complex storage are called **real-reg** and the **imag-reg**. They are visible as the two registers under the **CSTORE** heading on the Scientific Calculator screen. Complex numbers can only be recalled to the **x-reg** and **y-reg**, or to the **z-reg** and **w-reg**. If a complex number is recalled to the **x-reg** and **y-reg**, a complex stack push occurs. If a complex number is recalled to the **z-reg** and **w-reg**, the stack push does not effect the values in the **x-reg** and **y-reg**, as only the **z-reg** and **w-reg** and the hidden registers above the **z-reg** and **w-reg** are pushed.

The following procedure is used to store complex numbers in the **CSTORE real-reg** and **imag-reg**.

STOC **STOC** — This combination stores the complex number in the **x-reg** and **y-reg** into the **real-reg** and **imag-reg**. The x-reg is stored into the **real-reg** and the **y-reg** is stored into the **imag-reg**.

STOC **+** **+** — This combination does a complex addition on the complex number in the **x-reg** and **y-reg** and the complex number in the **real-reg** and **imag-reg**. The result is stored in the **real-reg** and the **imag-reg**. **STOC** **+** **y** is equivalent.

STOC **z** **z** — This combination stores the complex number in the **z-reg** and **w-reg** into the **real-reg** and **imag-reg**. The **x-reg** is stored into the **real-reg** and the **y-reg** is stored into the **imag-reg**. The combination **STOC** **w** **w** is equivalent and will give the same result.

STOC **/** **z** — This combination divides the complex number in the **z-reg** and **w-reg** into the complex number in the **real-reg** and **imag-reg**. The real part is stored into the **real-reg** and the imaginary part is stored into the **imag-reg**. **STOC** **z** **/**, **STOC** **/** **w**, and the combination **STOC** **w** **/** are equivalent

The rules for storing complex numbers are simple. [STOC] [STOC] will store the complex number in the **x-reg** and **z-reg** into the **CSTORE** registers. [STOC] [y] [y] and [STOC] [x] [x] are equivalent to [STOC] [STOC]. The combination [STOC] [z] [z] or [STOC] [w] [w] will store the complex number in the **z-reg** and **w-reg** into the **CSTORE** registers. [STOC] followed by two similar math operators will perform the indicated math operation on the complex numbers in the **x-reg** and **y-reg** and the complex numbers in the **real-reg** and **imag-reg**, storing the result into the **real-reg** and **imag-reg**. Finally, [STOC] followed by a math operator and a [x], [y], [z] or [w] button will perform the indicated math operation on the **CSTORE registers** and the **x-reg** and **y-reg** or the **z-reg** and **w-reg**, depending on which button was initiated, and store the result into the **real-reg** and **imag-reg**. To recall complex numbers from the **CSTORE real-reg** and **imag-reg**, use the following procedure. This procedure is very similar to the procedure used to store complex numbers in the the complex memory registers.

[RCLC] [RCLC] — This combination recalls the complex number in the **real-reg** and **imag-reg** into the **x-reg** and **y-reg**. A complex push is done on the stack. The **real-reg** is stored into the **x-reg** and the **imag-reg** is stored into the **y-reg**.

[RCLC] [+] [+] — This combination does a complex addition on the complex number in the **x-reg** and **y-reg** and the complex number in the **real-reg** and **imag-reg**. The result is stored in the **x-reg** and the **y-reg**. [RCLC] [+] [y] is equivalent. The stack is not pushed or pulled.

[RCLC] [z] [z] — This combination recalls the complex number in the **real-reg** and **imag-reg** into the **z-reg** and **w-reg**. A complex push is done on the hidden complex registers above the **z-reg** and **w-reg** only. The combination [RCLC] [w] [w] is equivalent and will give the same result.

[RCLC] [/] [z]. — This combination divides the complex number in the **real-reg** and **imag-reg** into the complex number in the **z-reg** and **w-reg**. [RCLC] [z] [/], [RCLC] [/] [w], and the combination [RCLC] [w] [/] are equivalent

From these examples you can see the simple rules for recalling numbers from the **CSTORE** registers. [RCLC] [RCLC] will recall the complex number in the **real-reg** and **imag-reg** into the **STACK x-reg** and **y-reg** registers. [RCLC] [y] [y] and [RCLC] [x] [x] are equivalent to [RCLC] [RCLC]. The combination [STOC] [z] [z] or [RCLC] [w] [w] recalls the complex number in the **real-reg** and **imag-reg** into the **STACK z-reg** and **w-reg** registers. [RCLC] followed by two similar math operators will

perform the indicated math operation on the complex numbers in the **x-reg** and **y-reg** and the complex numbers in the **real-reg** and **imag-reg**, storing the result in the **STACK x-reg** and **y-reg** registers. Finally, [RCLC] followed by a math operator and a [x], [y], [z] or [w] button will perform the indicated math operation on the indicated **STACK registers** with the **real-reg** and **imag-reg** values. The result will be stored in the **STACK x-reg** and **y-reg** registers or the **STACK z-reg** and **w-reg**, depending on whether the button following the math operator is a [x], [y], [z] or [w] button.

Simple recalls from the **CSTORE** registers will push the **STACK**. If the **real-reg** and **imag-reg** values are recalled to the **x-reg** and **y-reg**, a complex stack push is done. If the **real-reg** and **imag-reg** values are recalled to the **z-reg** and **w-reg**, a complex stack push is done on the hidden registers above the z-reg and w-reg.

Calculator Function Mode

The Calculator Function mode is activated by clicking on the [FCN] button. You will use this mode to do the following:

1. Work with large functions. The input register is expanded to fill the entire width of the screen, allowing you to enter functions of up to 80 characters in length.
2. Save and recall user defined function files to and from disk files.
3. Save and recall user defined functions to and from calculator memory.

When you are in the **Calculator Function** mode, all buttons on the **Scientific Calculator** screen are disabled, except for [DM], [RM], [GM], [CAL], [FCN], [PAD], [FIX], [SCI], [+1] and [-1]. This allows you to to set the **trig mode**, adjust the output format of the numbers and change to the **Scientific Calculator** mode with the [CAL] button or the **Ten Key Calculator** mode with the [PAD] button.

Formulas entered and solved in the **Calculator Function** mode are not passed to the **STACK**. The **STACK** is not involved in these operations. Results that are to be used in **STACK** calculations must be saved in an [A] to [Z] memory location and recalled, as explained previously in this chapter. Functions are entered and solved using the **Direct Function Entry** method. For example, to solve the function **log(12.3) * 1.2$^{1.4}$**, use the following procedure:

log(12.3)*1.2^1.4	[return]	The answer 1.4068 is returned in the Function Input Register.
a=log(12.3)*1.2^1.4	[return]	This will store the result in [A] for later use.

The top portion of the **Calculator Function** mode screen is shown in FIGURE 3.21.

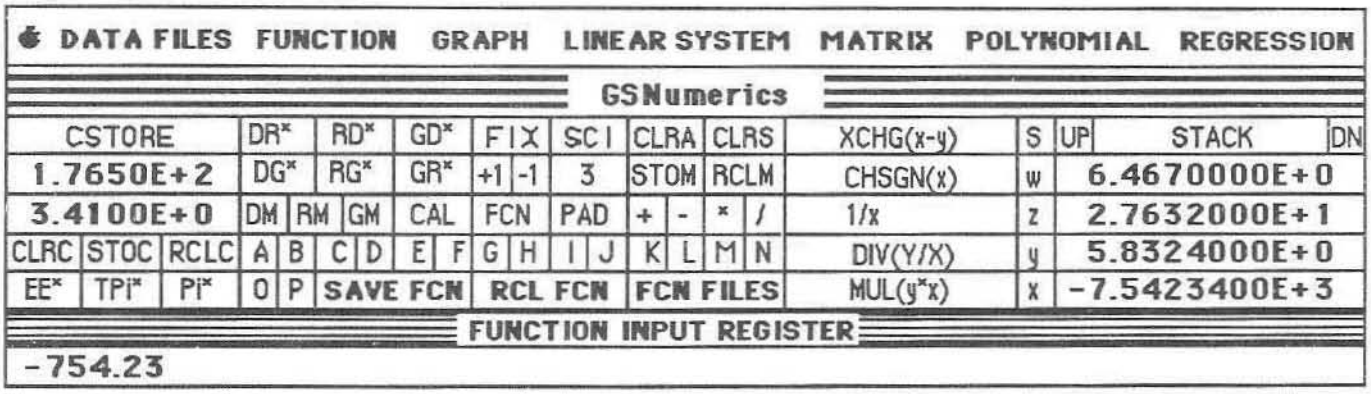

FIGURE 3.21 Calculator Function mode screen

Notice the **INPUT REGISTER** has been changed to the **FUNCTION INPUT REGISTER** and is much wider. You will also notice three new buttons **SAVE FCN**, **RCL FCN** and **FCN FILES**, located directly above the **FUNCTION INPUT REGISTER**. These buttons are used to save and recall functions to memory, and to save and recall files containing user defined functions, to and from disk storage.

We will save some functions to memory, to demonstrate the process. Suppose you needed to do conversions for resistors hooked in a delta configuration to resistors in a wye configuration and from wye configuration to a delta configuration. These are called wye-delta transformations and delta-wye transformations. The formulas for each wye-delta transformation is:

$$R_a = \frac{R_1R_2 + R_1R_3 + R_2R_3}{R_3} \quad R_b = \frac{R_1R_2 + R_1R_3 + R_2R_3}{R_2} \quad R_a = \frac{R_1R_2 + R_1R_3 + R_2R_3}{R_1}$$

The delta-wye transformations are:

$$R_1 = \frac{R_aR_b}{R_a+R_b+R_c} \quad R_2 = \frac{R_aR_c}{R_a+R_b+R_c} \quad R_3 = \frac{R_bR_b}{R_a+R_b+R_c}$$

We can all agree that it would not be much fun typing each of these equations, if you needed to make a large number of transformations. We can save a lot of time, and cut down on the chances for error, by entering them once and saving them for later use. To do this click on the **SAVE FCN** button. When you activate this button, the **RCL FCN** button will change to **CANCEL**, and the **Save Function** dialog, shown in FIGURE 13.21, will appear.

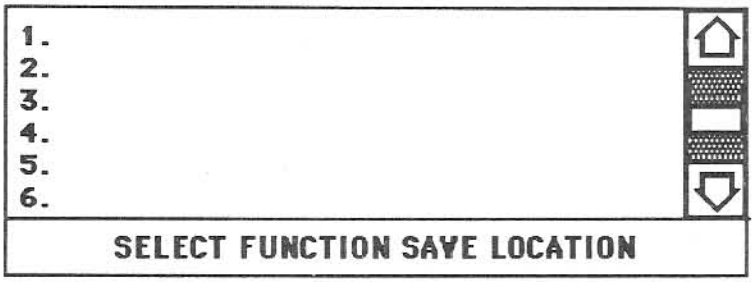

FIGURE 3.21 Save Function dialog

You first select a function location, where you wish to save the function. Merely click to the right of the label "**1.**", and the line will invert. A second click on the inverted line, will bring up a dialog asking you to "**Enter function you wish to save**". At this point we will enter the first wye-delta transformation. We will save the **Ra**, **Rb** and **Rc** values in memory [A], [B] and [C] and will assign memory locations [R], [S] and [T] for the resistance values. Enter the function, in the **FUNCTION INPUT REGISTER**, as follows:

	[SAVE FCN]	Double-click on line #1.
a=(r*s+r*t+s*t)/t	[return]	This enters the function and a new dialog appears, asking you to name the function. Type "**WYE-DELTA-RA (R-S-T)**" and press [return]. This will remind you that it is the formula for the wye-delta transformation, solving for **Ra**, and that the wye values are represented by the variables **r**, **s** and **t**.
	[SAVE FCN]	Notice the function "**WYE-DELTA-RA (R-S-T)**" is on line one. Double-click on line #2.
b=(r*s+r*t+s*t)/s	[return]	Enter the function and press [return]. Name this function "**WYE-DELTA-RB (R-S-T)**" and press [return].
	[SAVE FCN]	Notice the function "**WYE-DELTA-RB (R-S-T)**" is on line one. Double-click on line #3.
c=(r*s+r*t+s*t)/r	[return]	This enters the third wye-delta function. Name the function "**WYE-DELTA-RC (R-S-T)**" and press [return].

You can enter the delta-wye conversions in exactly the same manner. It will be easier to use the variables r,s and t for the delta resistances and a,b and c for the wye resistances, like we used them in the above example. When you are ready to recall and use the functions, set the memory locations to the appropriate values, the wye resistances in the above examples, prior to recalling the stored function. You may store up to thirty functions in the memory file. Use the vertical scroll bar to scroll through the thirty locations.

We wish to find the delta configuration resistors R_a, R_b and R_c, given that the wye configuration R_1= **3.2 omhs,** R_2 **= 5.2 ohms** and R_3 **= 6.1 ohms.** Input the wye values to memory locations r,s and t:

r=3.2	return	Enter R_1 value.
s=5.2	return	Enter R_2 value.
t=6.1	return	Enter R_3 value.

To use the function, click on RCL FCN and the **Select Function for Recall** dialog, shown in FIGURE 13.22 will appear.

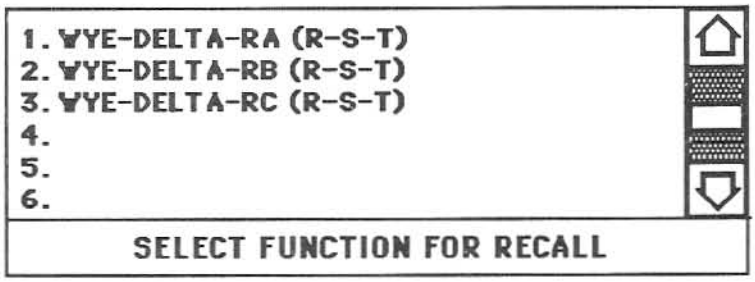

Figure 3.22 Select Function for Recall dialog

We will first solve for R_a. Click twice on the function "**WYE-DELTA-RA (R-S-T)**" to recall the function. This brings the function to the **FUNCTION INPUT REGISTER.** To solve the function, do the following:

return **Ra** is equal to 11.1279 ohms. This value is also stored in A.

Now solve for R_b by clicking on RCL FCN and selecting "**WYE-DELTA-RB (R-S-T)**".

return R_b is equal to 13.0538 ohms. This value is also stored in B.

Now solve for R_c by clicking on RCL FCN and selecting "**WYE-DELTA-RC (R-S-T)**".

return R_c is equal to 21.2125 ohms. This value is also stored in C.

The stored values can be used in the **Scientific Calculator** mode, by recalling them to the stack or using them in memory operations. Notice how easy it is to solve the functions, after you have entered and saved them in the **Calculator Function** mode.

User Defined Function File

You may save user defined functions to a disk file and recall them at a later time. This will allow you to define regularly used formulas, group them into related categories and call them up when needed. We will now save the functions file you have created.

Start by clicking on the **FCN FILES** button. This will change the screen to the new screen shown in Figure 13.23.

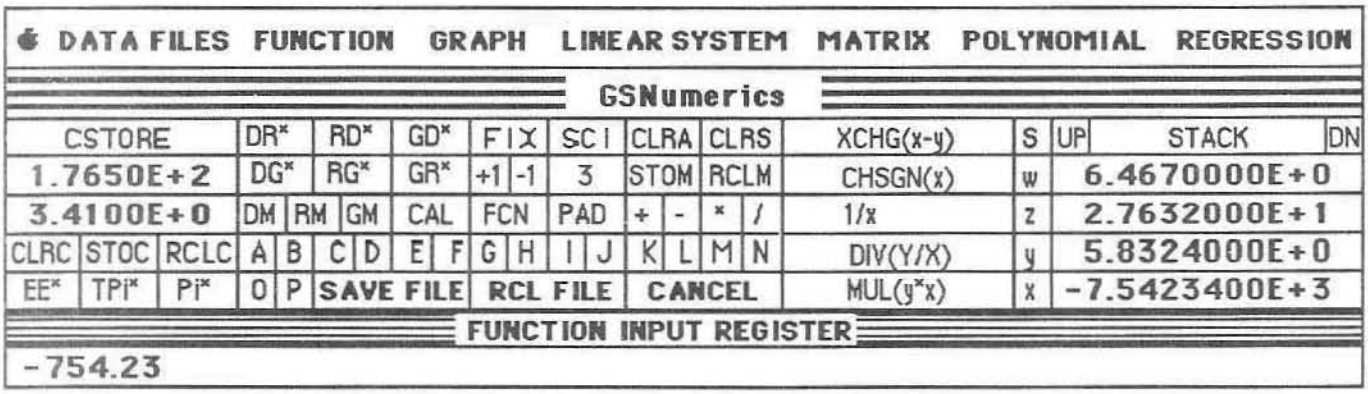

Figure 3.23 User Function File Operations screen

To save the file click on the **SAVE FILE** button and the Standard File dialog will appear. Select the proper disk and folder and type in the name you want to give the file. Then save the file.

You may recall the functions by first putting the **Scientific Calculator** in the **Calculator Function** mode by clicking on the **FCN** button. You will then click on the **FCN FILES** button and the **RCL FILE** button. This will bring up the Standard File dialog for recalling files.
Select the disk, folder and function file you wish to recall and recall it. The formulas will then be in memory and can be used as demonstrated above.
You may leave the **Calculator Function** mode and return to the **Scientific Calculator** mode by clicking on the **CAL** button, or you may go directly to the **Ten Key Calculator** mode by clicking on the **PAD** button.

Ten Key Calculator Mode

This mode allows the user to use the keypad as a ten key calculator, operating with **RPN** entry. You select the **Ten Key Calculator** mode by clicking on the **PAD** button from either the **Scientific Calculator** mode or the **Calculator Function** mode.

When operating in the mode, the only buttons enabled on the screen are memory storage, recall and math operators for real numbers and the CAL, FCN, PAD, FIX, SCI, +1, S and -1 buttons. All other buttons are disabled. You may use the mouse to initiate memory operations, as explained earlier in the chapter.

Using the **Ten Key Calculator** is very simple. First key in a number, then enter the number by pressing on the enter key or initiate an operation by pressing on either the +, −, *, / or = key on the number pad. The = key is the unary minus and is used to change the sign of the **x-reg** . The **Ten Key Calculator** operates just like the **Scientific Calculator**, except you use the key pad buttons.

Do the following calculation, using the **Ten Key Calculator**: $\frac{3.24 + 12.5}{-4.5 * 3.5}$

3.24	enter	Enter the first numerator number 3.24.
12.5	+	Do the first addition. Notice that the + entered the number and initiated the addition. **x-reg** = 15.74.
4.5	=	Enter the first denominator number 4.5. Apply the unary operator = to make it a negative number. Note the stack was pushed.
3.5	* /	Enter the second denominator number 3.5 and do the multiplication. Then do the division. The answer is -0.999 and is in the **x-reg** and **INPUT REGISTER**.

Ten Key Calculator stack operations are real number operations. The operators +, −, * and / buttons pull the stack, enter pushes the stack and = is a function of one variable and does not push or pull the stack, unless it is also used to enter the number. You may undo mistakes with the S button. Each time a calculation is made, the last stack is saved.

You may leave the **Calculator Function** mode and return to the **Scientific Calculator** mode by clicking on the CAL button, or you may go directly to the **Calculator Function** mode by clicking on the FCN button.

Calculator Screen Colors

You may set the screen colors of the **Scientific Calculator** to suit your own taste. You do this by selecting the **Set Screen Colors** item under the menu. This will bring up the **Set Screen Color** dialog.

This dialog contains three horizontal scroll bars, labelled "**RED Level=**", "**GREEN Level=**" and "**BLUE Level=**". Color labels show the intensity level, (hexidecimal 0 to hexidecimal 15) for the red, green and blue colors. The dialog also contains four radio buttons labelled **Color 1, Color 2, Color 3** and **Color 4**. These represent the four colors the user can set. **Color 1** controls the color of the buttons, in the upper part of the calculator screen, that are colored light blue when the program is first started. **Color 2** controls the color of the upper screen colors normally colored light brown. **Color 3** controls the buttons normally blue on the **Conversion-Elements** section and **Color 4** controls the orange buttons in the **Conversion-Elements** section.

There are three buttons on the dialog [Color], [Mono] and [OK]. Clicking on the [Color] button will reset the screen colors to the standard color screen of light blue, light brown and orange. Clicking on the [Mono] button will reset the calculator colors to a monochrome (black and white) mode of grey levels, for use with a monochrome monitor. When you have selected the desired colors, clicking on the [OK] button will return you to the **Scientific Calculator**.

To set a screen color, click on the radio button controlling the color you wish to set. After you do this, the color that will be changed will blink for a short time, showing which colors you have selected. You can then "mix" you own color for the selection by using the scroll bars to add or subtract different levels of red, green and blue. You will see the colors on the screen change, as you change the red green and blue levels. After you have set the colors, click on the [OK] button. The program will save your colors to a disk file named "**GSCOL**". This file will be read when you restart the program and you screen colors will be automatically set for you. If the file "GSCOL" is missing from your disk, the program will create it for you and set the colors to the standard default blue, brown and orange.

Seeding The Random Number Generator

The random number generator will generator a repeatable sequence of pseudo random numbers, if the initial "seed" value is started at the same value. This seed is set with the **Seed Random** item from the menu. This will bring up the **Set Random Seed** dialog. This dialog has one edit line and button labelled [OK]. When you first start **GSNumerics**, the random number is seeded with 1,915,998. The number may be seeded with any number, whose value falls within the following range:

$$1 \leq \text{random seed} \leq 2{,}147{,}483{,}646$$

After entering the new seed, click on the [OK] button and you will be returned to the **Scientific Calculator** screen. The reason you may want to seed the random number generator is to regenerate a sequence of numbers, or to start a new sequence. The random seqence will always be the same, for a

specific random number seed. This allows you to repeat the sequence. If you seed with an entirely new number, the random sequence will be different. Use this when you want a totally fresh run of random numbers. If you seed with a number outside the range given above, the program will automatically reset the seed to 1,915,998.

Special Operators

Special operators are special entries to the **INPUT REGISTER** that initiate special actions of the program. The special operators and the chapters they are explained in, are listed in Figure 3.24.

1.	**p[**	Polar operator. **SCIENTIFIC CALCULATOR**
2.	**r[**	Rectangular operator. **SCIENTIFIC CALCULATOR**
3.	**fi[**	Function Input operator. **FUNCTION**
4.	**fg[**	Function Graph operator. **FUNCTION**
5.	**pr[**	Polynomial Reset operator. **POLYNOMIAL**
6.	**pc[**	Polynomial Coefficient operator. **POLYNOMIAL**
7.	**pi[**	Polynomial Input operator. **POLYNOMIAL**
8.	**pg[**	Polynomial Graph operator. **POLYNOMIAL**
9.	**ling[**	Linear Graph operator. **LINEAR REGRESSION**
10.	**logg[**	Log Graph operator. **LINEAR REGRESSION**
11.	**expg[**	Exponential Graph operator. **LINEAR REGRESSION**
12.	**powg[**	Power Graph operator. **LINEAR REGRESSION**
13.	**datag[**	Data Graph operator. **LINEAR REGRESSION**
14.	**ling[**	Linear Prediction operator. **LINEAR REGRESSION**
15.	**logg[**	Log Prediction operator. **LINEAR REGRESSION**
16.	**expg[**	Exponential Prediction operator. **LINEAR REGRESSION**
17.	**powg[**	Power Prediction operator. **LINEAR REGRESSION**

Figure 3.24 **Gsnumerics** Special Operators

You have already seen two of the operators, the **Polar Entry** operator and the **Rectangular Entry** operator. Special operators are used to quickly initiate actions, bypassing dialogs that are normally used to initiate the actions. Refer to the indicated chapters to find an explanation and the use of the Special Operators.

POLYNOMIAL OPERATIONS

Polynomials are an important class of functions. Many practical uses are found for polynomials in engineering and science. Understanding polynomials is very important to a complete understanding of mathematics.

A polynomial is a function of the form:

$$y(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x^1 + a_0 x^0 \qquad \text{Formula 4.1}$$

Where the **a** terms are the polynomial coefficients and **n** is the order of the polynomial. Examples of polynomial functions are:

$y(x) = x^2 - 3x + 1$	Order two polynomial	Formula 4.2
$y(x) = 4x^5 - 3x^2 - 1$	Order five polynomial	Formula 4.3
$y(x) = x + 5$	Order one polynomial	Formula 4.4

The definition of a polynomial given in Formula 4.1 can also be expressed as the product of **n** binomial factors, where **n** is the order of the polynomial. A binomial is a polynomial of order one, as shown in Formula 4.4. Expressed as a product of binomials, the definition of a polynomial is:

$$y(x) = (x - r_n)(x - r_{n-1}) \ldots (x - r_1) \qquad \text{Formula 4.5}$$

Where the **r** terms are the roots and the $(x - r_n)$ terms are the factors. The roots may be either real or complex, and the number of roots will always exactly equal the order of the polynomial. In other words, a second order polynomial will always have exactly two roots, and a sixth order polynomial will always have exactly six roots. Examples of polynomials expressed in both forms are:

$$y(x) = x^2 + 2x - 3 = (x + 3)(x - 1) \qquad \text{Formula 4.6}$$

$$y(x) = x^3 + 2x^2 - x - 2 = (x + 1)(x - 1)(x + 2) \qquad \text{Formula 4.7}$$

$$y(x) = x^2 - 2x + 2 = (x - 1 + i)(x - 1 + i) \qquad \text{Formula 4.8}$$

The roots of the polynomial given in Formula 4.6 are -3 and 1, the roots of Figure 4.7 are -1, +1, and -2, and the roots of the third polynomial (Formula 4.8) are the complex conjugate pair 1 - i and 1 + i. Complex roots of polynomials will always appear in complex conjugate pairs. You will never have a single complex root of a polynomial.

GSNumerics allows the user to perform the following operations involving polynomials:

1. Enter polynomials of up to the tenth order.
2. Solve a polynomial for a specific x value.
3. Compute the slope of a polynomial at a specific point.
4. Compute the area under a polynomial curve between two specific points.
5. Differentiate a polynomial, returning a new polynomial that is the derivative of original polynomial.
6. Integrate a polynomial, returning a new polynomial that is the integral of the original polynomial.
7. Analytically find all of the real and complex roots of a polynomial.
8. Create polynomials with roots at predetermined locations.
9. Multiply a polynomial by a binomial term, creating a new, higher order polynomial containing the binomial factor.
10. Divide a polynomial by a binomial term, creating a new, lower order polynomial with the binomial term removed, and returning the division remainder.
11. Pass the roots, area, solution or slope to the stack, using stack math operations.
12. Graph a polynomial over a given x range, magnify selected areas of the graph and find the real roots graphically.

Entering Polynomials

Polynomials are entered by selecting the **Enter Polynomial** menu item from the **POLYNOMIAL** menu. The dialog screen shown in figure 4.1 will then appear:

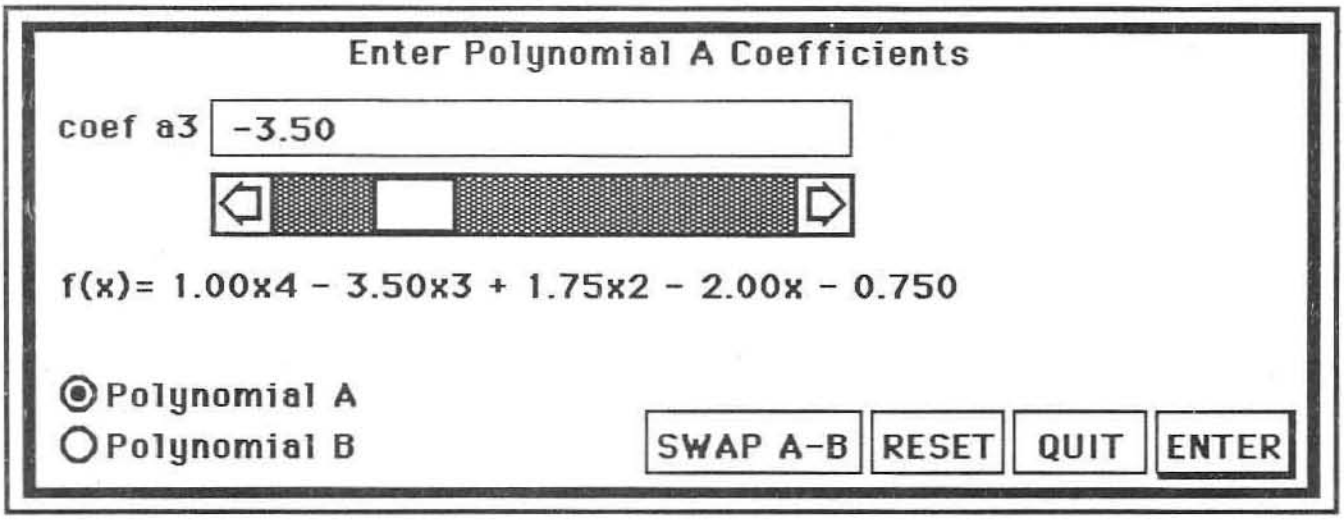

Figure 4.1 Polynomial Entry dialog

This dialog allows the user the option of resetting the polynomial coefficients to zero, using the RESET button, quitting the entry screen by keying on the QUIT button or entering a polynomial coefficient, using the ENTER button or return key. The scroll bar can be used to review and edit the polynomial coefficients. The order of the current polynomial is displayed on the dialog screen.

When entering a new polynomial you should first reset the polynomial coefficients and polynomial order to zero using the RESET button. When you are returned to the Polynomial Entry Dialog the coefficient label to the left of the dialog edit line will be set to "**coef a0**", indicating you are ready to input the polynomial constant coefficient a_0. Polynomials are entered starting with the lowest order term and moving to the highest order term. If a coefficient of one or more of the lower terms is zero, it must be entered as zero. After the first order (a_1) term is entered, the Polynomial Order will be incremented, showing the order of the current polynomial and the coefficient label will increment, to the next term to be entered. When you have entered the highest order term, you may return to the calculator screen, by keying on the QUIT button.

Three types of entry are possible, when entering values into the edit line:

1. Entering a number, followed by the ENTER button or return key, will enter the number as the polynomial coefficient and advance the coefficient label and polynomial order.
2. Entering a valid function, using direct function entry, will set the polynomial coefficient to the value of the function and advance the polynomial coefficient label and polynomial order.
3. Entering a memory assignment with direct function entry (i.e. a = 2.35 or c = sin(b)/cos(d)) will not advance the polynomial label or set the polynomial coefficient to the value of the memory assignment. The coefficient can be set to the result by referencing the memory location, using direct function entry. As an example, suppose you wanted to enter a polynomial coefficientmade up of the following function:

$$\sin(2 * f) * \cos(c)/\log(23.9) + \tan(2 * f)^2/(34 * g) \qquad \text{Formula 4.9}$$

You could enter the coefficient by first entering **a = sin(2 * x) * cos(c)/log(23.9)**, then entering **a + tan(2 * f)^2/(24 * g)**. Since the last term is not a memory assignment the result will be entered as a coefficient and the coefficient label and polynomial order will both be incremented.

The following examples will demonstrate entering polynomials. Notice that a zero coefficient must be entered as zero, to insure the order of the polynomial is correct.

Enter the polynomial $y(x) = x^2 - 2x + 1$

	RESET	Resets all polynomial coefficients and order to 0.0.
1	ENTER	Enter the a_0 coefficient.
-2	ENTER	Enter the a_1 coefficient
1	ENTER	Enter the a_2 coefficient. Notice that the polynomial order is shown as 2. You could have pressed the return key instead of keying on the ENTER button.

Enter the polynomial $y(x) = x^5 + 3x^3 - 4x$ Notice the a_4, a_2 and a_0 terms are 0.0.

	RESET	Resets all polynomial coefficients and order to 0.0.
0	ENTER	You must enter the 0.0 for coefficient a_0.
-4	ENTER	Enter the a_1 coefficient.
0	ENTER	You must enter the 0.0 for coefficient a_2.
3	ENTER	Enter the a_3 coefficient.
0	ENTER	You must enter the 0.0 for coefficient a_4.
1	ENTER	Enter the a_5 coefficient. Notice the polynomial order is five.

After entering a polynomial you may exit the **Enter Polynomial Coefficients** dialog by keying on the QUIT button. You will be returned to the screen that called the dialog.

Solving Polynomials

GSNumerics can quickly solve polynomials for a specific x value. This is done by using the **Solve Polynomial** dialog, that is selected from the **POLYNOMIAL** menu by keying on the **Solve** menu item. The **Solve Polynomial** dialog is shown in figure 4.2. This basic dialog will be used to do all polynomial operations except for the **Enter Polynomial** dialog and the **Solve for Polynomial Roots** dialog. As you cursor through the dialog, the dialog items will change, depending on which polynomial function you have seleced. You may toggle to the other polynomial operations screens, by clicking on the MODE button found on the screens. Clicking on the MODE button will bring up the next polynomial operations screen. Clicking on the QUIT button, will return you to the **Scientific Calculator** main screen.

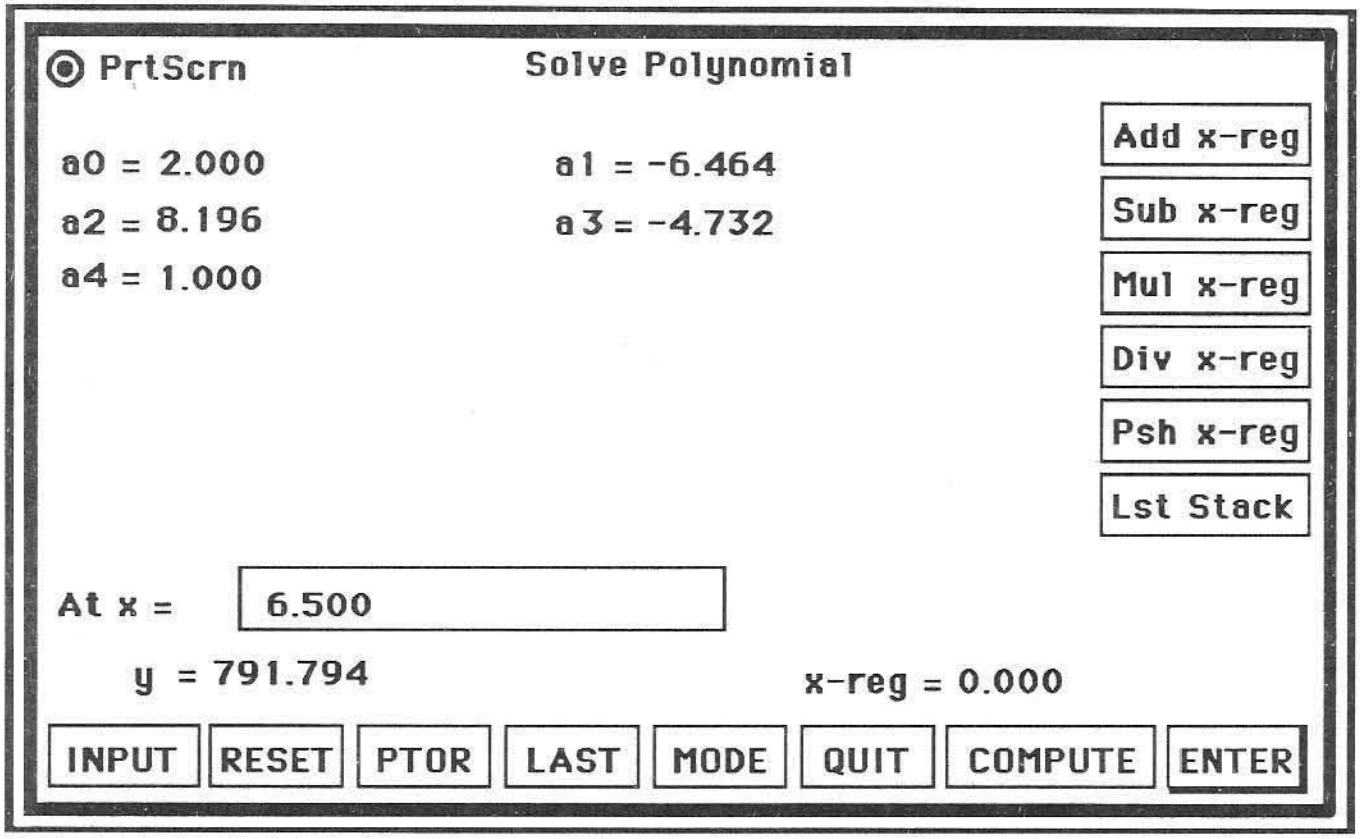

Figure 4.2 Solve Polynomial dialog

The dialog box contains fourteen buttons, an edit line and one radio button. The coefficients of the current polynomial will be shown in the dialog box. The current polynomial in figure 4.2 is:

$$y(x) = x^4 - 4.732x^3 + 8.196x^2 - 6.464x + 2 \qquad \text{Formula 4.10}$$

The value of the current **x-reg**, from the main calculator screen is also shown (**x-reg = 0.000**). This is used in conjunction with the six vertical buttons to pass the result of a polynomial solution to the stack, for use in other calculations. The edit line is used to input the value of x, where the solution of the polynomial is required, and the solution is shown under the edit line (**y = 791.794**). Notice that the format of the numbers will always be the format selected on the **GSNumerics** calculator screen, in this example it is fixed format with three digits after the decimal. If you wish to change the format of the numbers on a dialog screen, simply return to the **Scientific Calculator** screen, set the desired format, and return to the screen of interest. The radio button, in the upper left corner of the screen, labelled **PrtScreen** is used to print the dialog screen to the printer. It can be selected at any time and will print the dialog box, to the printer, in black and white mode.

Four of the horizontal row of buttons, across the bottom of the **Solve Polynomial** dialog, will be active. The other eight will be dimmed and inactive when the **Solve Polynomial** dialog is selected. The active buttons are explained in the following:

INPUT This button is used to call the **Input Polynomial** dialog if the user wishes to enter a new polynomial, or edit the present polynomial.

ENTER This button enters an x value typed into the edit line labelled **At x =**, and solves the polynomial at that value of x and displays the solution after **y =**. This button will be dimmed and inactive if a polynomial has not been entered.

QUIT Quits the **Solve Polynomial** dialog and returns the user to the main **Scientific Calculator** screen.

MODE Calls the next polynomial operation screen (**Integrate Polynomial** dialog). By keying on this button, the user may call up any of the polynomial operation screens, without returning to the main menu. Just continue to key on this button, until the desired dialog is called.

To demonstrate the solution of polynomials, we will solve the polynomial entered in the previous section :

$$y(x) = x^5 + 3x^3 - 4x \qquad \text{Formula 4.11}$$

If you do not have the polynomial entered, please enter it as shown in the previous section.

Solve the polynomial for **x = 2**:

2 **ENTER** The correct answer is **y = 48.0000.**

Solve the polynomial for **x = -.55**:

-.55 **ENTER** The correct answer is **y = 1.6505.**

Solve the polynomial for **x = 1**:

1 **ENTER** The correct answer is **y = 0.0000.**

As you can see, it is very easy to solve polynomials, even complicated polynomials, for any value of x using the **Solve Polynomial** dialog.

You may use the vertical buttons to pass the polynomial solution to the stack. This enables you to use the solution of the polynomial, in computations requiring other calculations. In fact, you can push the solution onto the stack, add the solution to the **x-reg**, subtract the solution from the **x-reg**, multiply the **x-reg** by the solution or divide the **x-reg** by the polynomial solution. If you make an error, you may recall the last stack to correct the error. The stack buttons will be dimmed and inactive until you actually input an x value and solve the current polynomial. The **Lst Stack** button will be inactive until you do a stack operation. The stack buttons are:

Add x-reg	Adds the solution of the polynomial to the current x-reg value, and places the result into the x-reg.
Sub x-reg	Subtracts the solution of the polynomial from the current x-reg value, and places the result into the x-reg.
Mul x-reg	Multiplies the current x-reg value by the polynomial solution and places the result into the x-reg.
Div x-reg	Divides the current x-reg value by the polynomial solution and places the result into the x-reg.
Psh x-reg	Pushes the solution of the polynomial to the stack.
Lst Stack	Recalls the Last Stack to the stack registers. Used to correct errors.

Keying on the **MODE** button will bring up the **Integrate Polynomial** dialog. You may move to any Polynomial Operation dialog, without returning to the menu, by multiple keying on this button.

Computing the Slope of a Polynomial

Many problems involve finding the slope of a polynomial at a certain value of x. For those of you who have had calculus, you will recognize this as the problem of taking the derivative of the polynomial and solving the derivative at the desired value of x. The derivative of y(x) is usually denoted as y'(x) read as "y prime of x". The derivative of a polynomial is easily computed, using the following rule:

$$y'(x) = (n)a_n x^{n-1} + (n-1)a_{n-1}x^{n-2} + \ldots + (1)a_1 \qquad \text{Formula 4.12}$$

This may look difficult to you non-calculus users, but it is really very simple. For instance, consider taking the derivative of $y(x) = x^4 - 2x^3 + 2x - 1$. You merely follow the rule expressed in formula 4.12 as follows:

$$y'(x) = (4)(1)x^{(4-1)} - (3)(2)x^{(3-1)} + (1)(2)x^{(1-1)} \qquad \text{Formula 4.12}$$

$$y'(x) = 4x^3 - 6x^2 + 2x^0 = 4x^3 - 6x^2 + 2 \qquad \text{Formula 4.13}$$

The slope of the function $y(x) = x^4 - 2x^3 + 2x - 1$ at $x = 2$ is merely the derivative solved for $x = 2$. Therefore, the slope of the function at $x = 2$ is:

$$y'(2) = 4(2)^3 - 6(2)^2 + 2 = 10 \qquad \text{Formula 4.14}$$

Many computer algorithms use iteration techniques to approximate the slope of functions. **GSNumerics** actually takes the derivative of the function, using the method outlined above, and solves the derivative at the required x value. Therefore, for polynomial computations, the slope is exact, within computational limits of the computer.

To compute the slope of a polynomial select the **Slope** menu item from the **POLYNOMIAL** menu. The **Polynomial Slope** dialog show in figure 4.3 will appear on the screen. You may also select this screen from other Polynomial Operations dialogs by keying on the MODE button, until the **Polynomial Slope** dialog appears.

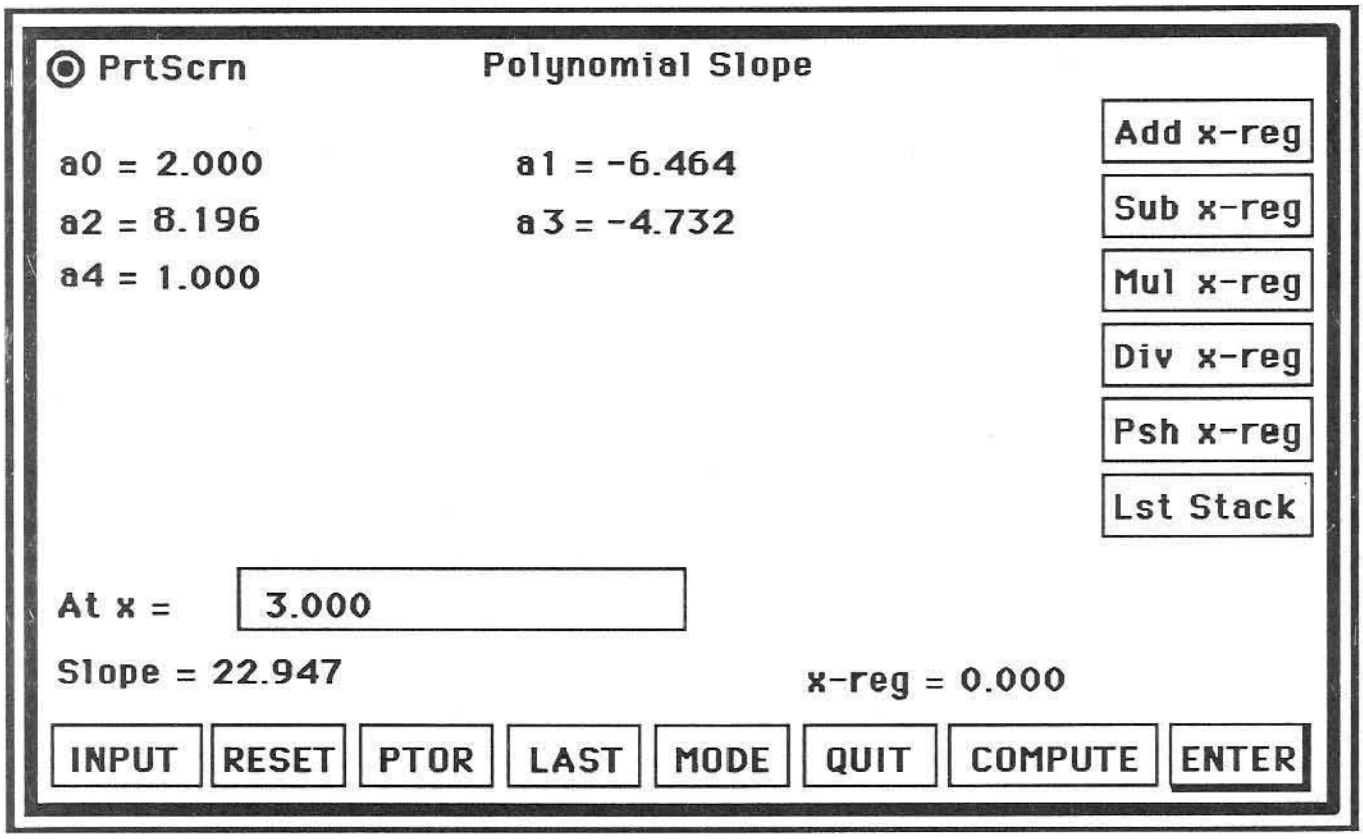

Figure 4.3 Polynomial Slope dialog

This screen is very similar to the **Solve Polynomial** dialog and the buttons perform the same functions. The active buttons are exactly like those explained under the previous section. To compute the slope of the active polynomial, you merely input the x value where you want to compute the slope and the value of the slope at that point will be computed and displayed after the label **Slope =**. To demonstrate the computation of the slope of a polynomial, we will solve the following polynomial that was entered in the **Entering Polynomial** section :

$$y(x) = x^5 + 3x^3 - 4x \qquad \text{Formula 4.15}$$

If you do not have the polynomial entered, please enter it as shown in the **Entering Polynomials** section.

Compute the slope $y(x) = x^5 + 3x^3 - 4x$ of at x = 1:
1 **ENTER** The slope at x = 1 is **y = 10.0000.**

Compute the slope $y(x) = x^5 + 3x^3 - 4x$ of at x = -0.75:
-0.75 **ENTER** The slope at x = -0.75 is **y = 2.6445.**

Compute the slope $y(x) = x^5 + 3x^3 - 4x$ of at x = Pi:
Pi **ENTER** The slope at x = Pi is **y = 571.8719.**

Compute the slope $y(x) = x^5 + 3x^3 - 4x$ of at x = Log(12.3):
Log(12.3) **ENTER** The slope at x = Log(12.3) is **y = 13.7465.**

You may perform the stack operations outlined in the **Solve Polynomial** section. The only difference is that you will be adding, subtracting, multiplying, dividing or pushing the slope of the polynomial to the stack instead of the solution. This will enable you to compute the slope of a polynomial and pass the result to the stack to use in other computations. Keying on the **MODE** button will bring up the **Solve Polynomial** dialog.

To compute the slope of any polynomial, you merely input the polynomial using the **Input Polynomial** dialog, bring up the Polynomial Slope dialog (either from the **POLYNOMIAL** menu or with the **MODE** button) and enter the value of x at the point where you want the slope computed.

Computing the Area under a Polynomial Curve

The third basic operation on polynomials is computing the area under the polynomial curve. This is expressed mathematically as:

$$\text{area} = \int_a^b f(x)dx$$

This is read as "the integral of f, with respect to x, from a to b". In non-calculus terms this tells us to compute the area under the function f(x) from x = a to x = b. To compute the area under a polynomial curve, you would first compute the "antiderivative" (a function whose derivative is equal to f(x)), and then solve the "antiderivative" at x = a and at x = b. You would then subtract the solution at x = a from the solution at x = b. The result would be the area under the polynomial curve from x = a to x = b. The "antiderivative" of a function is commonly called the "integral" of the function. The process of finding the "antiderivative" or "integral" is called integration.

We computed the derivative of $y(x) = x^4 - 2x^3 + 2x - 1$ in the previous section. The derivative was determined to be $4x^3 - 6x^2 + 2$. Therefore, $x^4 - 2x^3 + 2x - 1$ is an "antiderivative" of the function $4x^3 - 6x^2 + 2$. Therefore, the area under the curve $4x^3 - 6x^2 + 2$ from x = 1 to x =2 of is computed as follows:

$$\int_1^2 (4x^3 - 6x^2 + 2)\,dx = ((2)^4 - 2(2)^3 + 2(2)) - ((1)^4 - 2(1)^3 + 2(1)) = 3 \quad \text{Formula 4.16}$$

As you can see, computing the area under the curve of a polynomial can be tedious and lends itself to errors. The above computation computes the area exactly. Some computer algorithms compute the area under curves using iteration techniques. **GSNumerics** actually integrates the polynomial and solves for the area as show in Formula 4.16. Therefore, the answers are exact, within the computational limits of the computer.

GSNumerics will quickly, easily and accurately compute the area under a polynomial curve using the **Polynomial Area** dialog. You may select this dialog by selecting the "**Area**" menu item under the **POLYNOMIAL** menu, or by keying on the **MODE** button from any Polynomial Operation dialog, until the **Polynomial Area** dialog appears. The **Polynomial Area** dialog is shown in Figure 4.4.

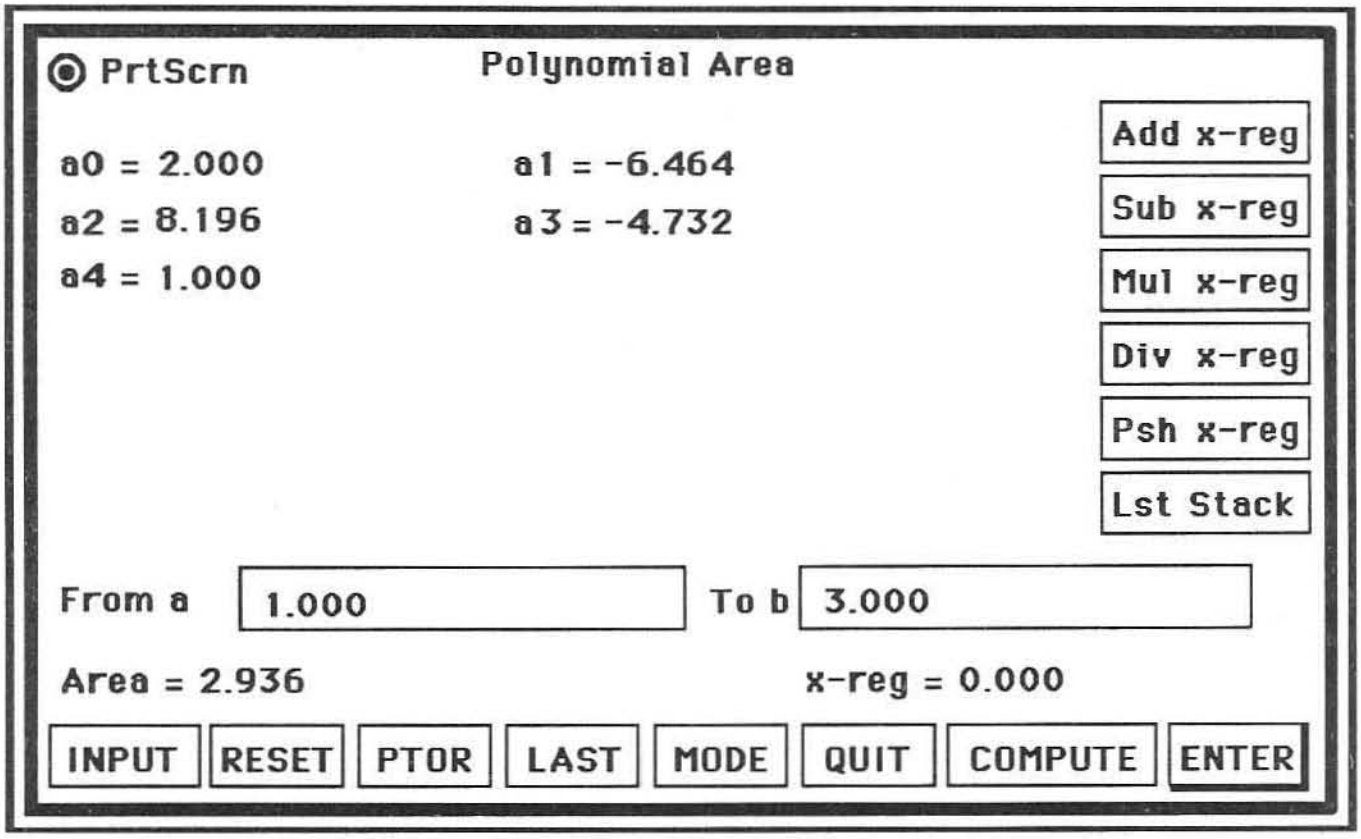

Figure 4.4 Polynomial Area dialog

The **Polynomial Area** dialog is similar to the **Solve Polynomial** and **Polynomial Slope** dialog boxes, except for the addition of another edit line. If a polynomial has not been entered, the ENTER and COMPUTE buttons will be drawn undimmed and active. To compute the area, you first type the x = a value in the "**From a**" edit line and key on ENTER. You then type the x = b value into the "**To b**" edit line and key on the ENTER button. Finally, you click on the COMPUTE button and the area is computed and shown after the label "**Area =**". To demonstrate we will use Formula 4.15. If you do not have the formula entered, please enter it using the **Input Polynomial** dialog.

Compute the area under the polynomial curve $y(x) = x^5 + 3x^3 - 4x$ from x = 5 to x = 6.

5	ENTER	Enter the first integration limit.
6	ENTER COMPUTE	Enter the second integration limit, then compute the area.

The area under the curve from x = 5 to x = 6 is 5653.0833. This is show to the right of the label "**Area =**", beneath the "**From a**" label.

Compute the area under the polynomial curve $y(x) = x^5 + 3x^3 - 4x$ from x = 0 to x = 1.5.

0	ENTER	Enter the first integration limit.
1.5	ENTER COMPUTE	Enter the second integration limit, then compute the area.
		The area under the curve from x = 5 to x = 6 is 1.1953.

Compute the area under the polynomial curve $y(x) = x^5 + 3x^3 - 4x$ from x = -.0124 to x = -0.0225.

-0.0124	ENTER	Enter the first integration limit.
-0.0225	ENTER COMPUTE	Enter the second integration limit, then compute the area.
		The area under the curve from x = -0.0124 to x = -0.0225 is -0.0007.

If you have solved for polynomial curve areas by hand, you will appreciate the ease with which you can make these computations with **GSNumerics**. You may pass the resultant area to the stack, using the stack buttons, as outlined in the previous sections.

Polynomial Integration

The process of integrating or finding the "antiderivative" of a polynomial was discussed briefly in the previous section. You will find it necessary to integrate polynomials in many scientific and engineering applications.

Integrating polynomials is really a very easy and straight forward problem. This is especially true for polynomials of lower order with integer coefficients. The difficulty comes in when the polynomials are of high order and have non-integer coefficients. It is very easy to make an error in the integration procedure. **GSNumerics** will integrate polynomials up to order nine quickly and precisely. You will only need to make sure you enter the coefficients accurately, using the Enter Polynomial dialog.

Since the program is limited to operating on polynomials of maximum order ten, you can't integrate an order ten polynomial, since the result would be a polynomial of order eleven.

You can activate the **Integrate Polynomial** dialog box by selecting the **Integrate** item from the **POLYNOMIAL** menu. You can also use the MODE button on any Polynomial Dialog screen to move to the **Integrate Polynomial** dialog.

The **Integrate Polynomial** dialog is shown in figure 4.5.

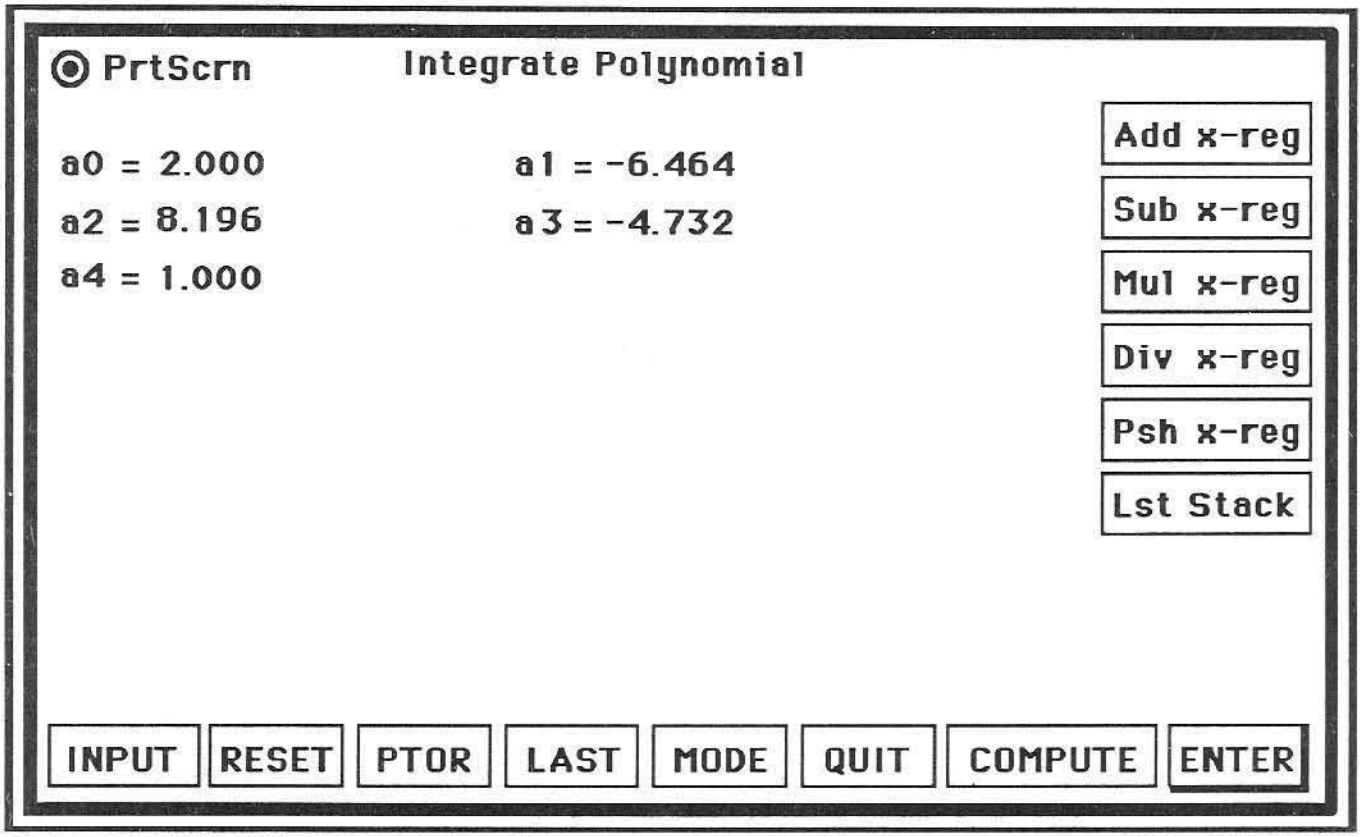

Figure 4.5 Integrate Polynomial dialog

The operation of the **Integrate Polynomial** dialog is very simple. When you bring up the dialog, four buttons are active. They are the INPUT , MODE, COMPUTE and QUIT buttons. If you do not have a polynomial entered, the COMPUTE button will be inactive. You may enter a new polynomial at any time by keying on the INPUT button. To integrate the current polynomial just key on the COMPUTE button.

After you have computed the polynomial integral, the LAST button will be activated, allowing you wish to replace the integrated polynomial with the unintegrated polynomial. This allows you to undo an error if you mistakenly integrate a polynomial, or to get the integral of a polynomial and then return to the original polynomial. The LAST button is used the same way you learned to recall the Last Stack, to recover from errors. You may also find this handy if you wish to compute the integral of a polynomial, but want to return to the original polynomial for further computations.

To demonstrate the **Integrate Polynomial** dialog, input the following polynomials by keying on the button to bring up the **Input Polynomial** dialog, and following the procedure for entering polynomials outlined earlier in the chapter:

Enter and integrate the polynomial $y(x) = x^2 + 3x - 2$

COMPUTE The integral is $y(x) = 0.333x^3 + 1.5x^2 - 2x + 0.$

Enter and integrate the polynomial $f(x) = x^5 - 4x^3 - 2x^2 + 3$

COMPUTE The integral is $y(x) = 0.166x^6 - x^4 - 0.666x^3 + 3x + 0.$

Enter and integrate the polynomial $f(x) = x^9 - 6.6x^3 - 2x + 3$

COMPUTE The integral is $y(x) = 0.1x^{10} - 1.65x^4 - 1.00x^2 + 3x + 0.$

Enter and integrate the polynomial $f(x) = x^8 - 2.1x^7 - 2.9x^6 + 2$

COMPUTE The integral is $y(x) = 0.111x^9 - 0.263x^8 - 0.414x^7 + 2x + 0.$

Enter and integrate the polynomial $f(x) = .341x^5 - 1.54x^4 - 2.75x^3 + 3.12x^2 + .987x - 0.265$

COMPUTE The integral is $y(x) = 0.057x^6 - 0.308x^5 - 0.688x^4 + 1.040x^3 + 0.494x^2 - 0.265x$

Notice the a_0 term will always evaluate to zero when integrating a polynomial. Commonly this is replaced by an unknown constant '**c**' in the indefinite integral. You should include the '**c**' when writing the integral on paper. If you have integrated many large polynomials, with non-integer coefficients, you will appreciate how fast **GSNumerics** will do the job.

Polynomial Differentiation

Differentiation of a polynomial is the problem of finding the polynomial whose "antiderivative" or integral is known. The rules for differentiating a polynomial were given earlier in this chapter under Polynomial Slope computations. The **Differentiate Polynomial** dialog is show in Figure 4.6. You may bring up this dialog by selecting the **Differentiate** item from the **POLYNOMIAL** menu, or by using the MODE button from another polynomial operation dialog.

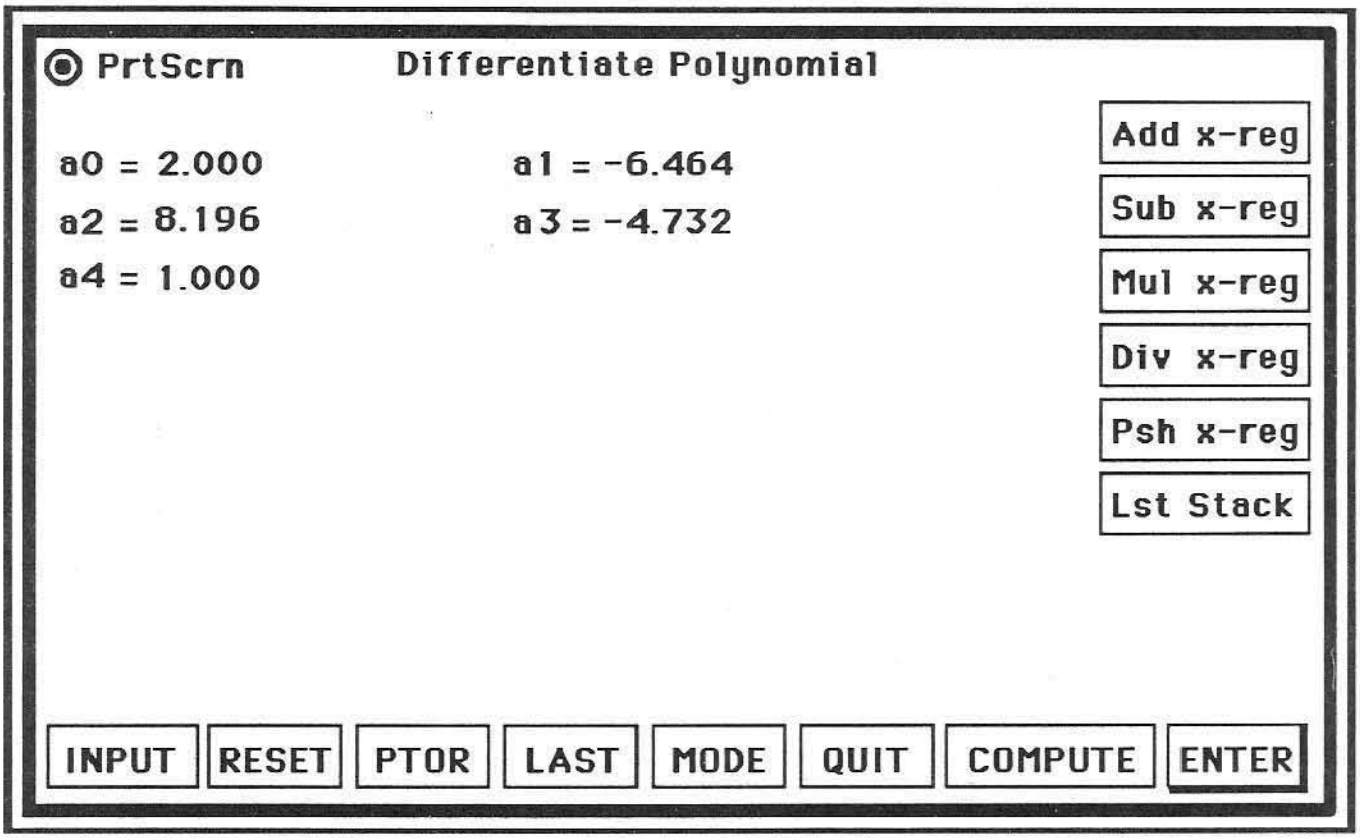

Figure 4.6 Differentiate Polynomial dialog

When you bring up the dialog, four buttons are active. They are the INPUT, MODE, COMPUTE and QUIT buttons. If you do not have a polynomial entered, the COMPUTE button will be inactive. You may enter a new polynomial at any time by keying on the INPUT button. To differentiate the current polynomial just click on the COMPUTE button. After you have computed the polynomial derivative, the LAST button will be activated, allowing you to replace the computed derivative with the original polynomial. This allows you to undo an error if you mistakenly differentiate a polynomial, or want to get the derivative of a polynomial and then return to the original polynomial.

To demonstrate the **Differentiate Polynomial** dialog, input the following polynomials by keying on the button to bring up the **Input Polynomial** dialog, and following the procedure for entering polynomials outlined earlier in the chapter:

Enter and differentiate the polynomial $y(x) = x^3 - 4x^2 - 2x + 3$

COMPUTE The derivative is $y(x) = 3x^2 - 8x - 2$.

Enter and differentiate the polynomial $f(x) = 2x^5 - 2.5x^3 - 2.8x$

COMPUTE The derivative is $y(x) = 10x^4 - 7.5x^2 - 2.8$.

Enter and differentiate the polynomial $f(x) = x^8 - .715x^7 - .365x^6 - .765x^4 + 2.564x^2 + 3$

COMPUTE The derivative is $y(x) = 8x^7 - 5.005x^6 - 2.190x^5 - 3.060x^3 + 5.128x$

Multiplying Polynomial by Binomials

As was shown in Formula 4.5 all polynomials can be expressed as the product of binomial terms. To be specific, a polynomial of order **n**, can always be expressed as the product of **n** binomial terms, the roots of which are the r_n values of the individual binomials. Therefore, if we wish to construct a polynomial with roots at know locations, we multiply the **n** individual binomial terms, containing the roots as the r_n values, and we will end up with an n^{th} order polynomial with roots at the r_n locations. We may also multiply any polynomial , by a binomial term and create a new polynomial of order **n+1**, containing the root in the binomial term. The binomial terms are called the factors of the polynomial.

If the binomial term is complex, we will end up with a **n+2** order polynomial, containing the complex root and it's complex conjugate as roots. You can't add only one complex root, since they will always occur in complex conjugate pairs. As an example, suppose you want to find a polynomial that contains roots at 2, 3, and -4. The procedure to follow in building this polynomial is as follows:

$$y(x) = (x - 2)(x - 3)(x + 4) = x^3 - x^2 - 14x + 24 \qquad \text{Formula 4.17}$$

If you solve the above polynomial at x = 2, x = 3, or x = -4, you will see that they are indeed roots of the polynomial. Suppose you wanted to add a complex root at 1 - i. Since complex roots always occur in pairs in polynomials with real coefficients, we would have to do two binomial multiplications:

$$y(x) = (x - 1\text{-}i)(x - 1+i)(x^3 - x^2 - 14^x + 24) = x^5 - 3x^4 - 10x^3 + 50x^2 - 76x + 48$$

We now have an order five polynomial with three real roots and a complex conjugate pair of roots. Construction of polynomials and adding roots to existing polynomials is done using the **Binomial Multiplication** dialog shown in Figure 4.7.

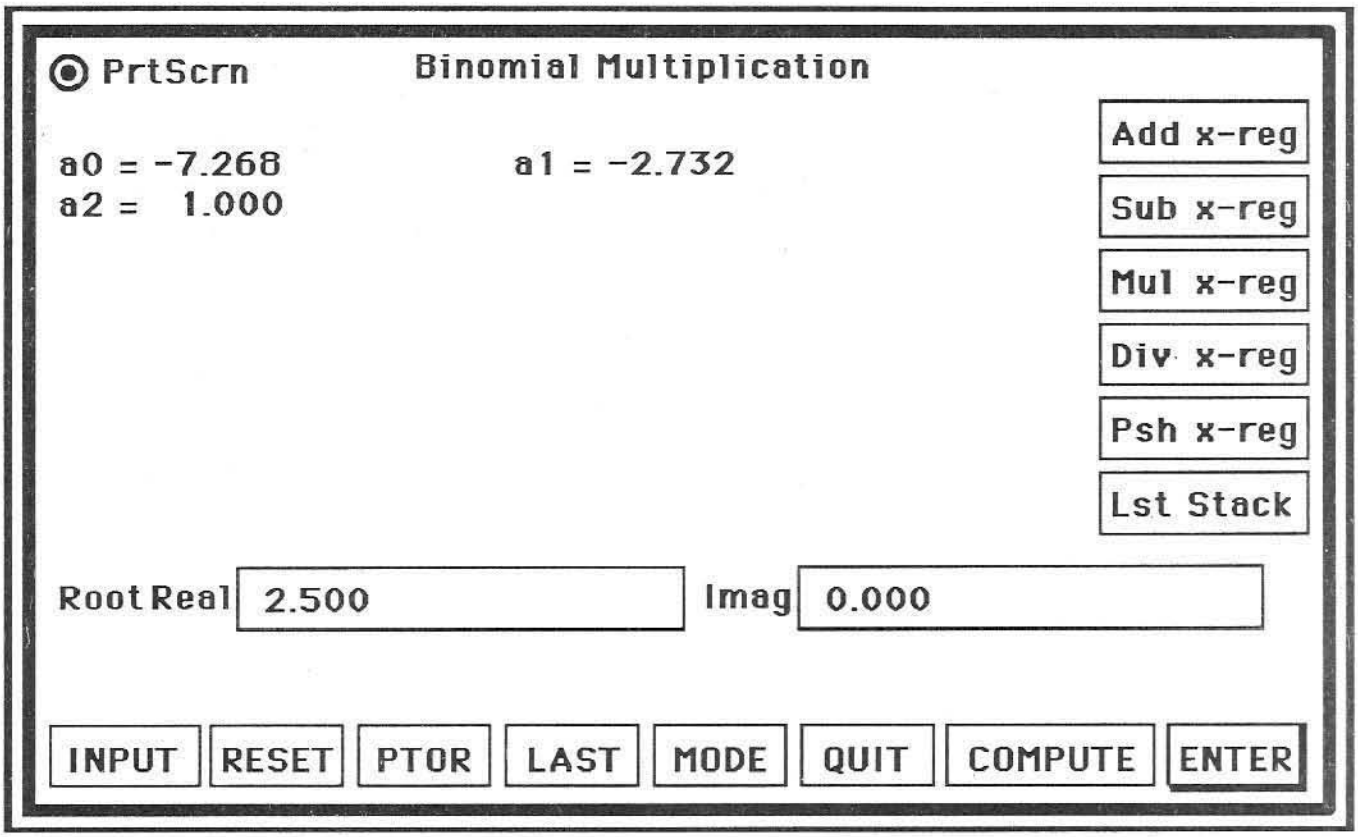

Figure 4.7 Binomial Multiplication

The **Binomial Multiplication** dialog can be invoked by selecting the **Binomial Multiplication** item from the **POLYNOMIAL** menu, or by using the **MODE** button from another polynomial dialog. When you bring up the dialog, seven buttons are active. They are the **INPUT**, **MODE**, **COMPUTE**, **QUIT**, **RESET**, **PTOR** and **ENTER** buttons. After you have computed a binomial multiplication, the **LAST** button will be activated, allowing you to replace the computed polynomial with the original polynomial. This allows you to undo an error if you mistakenly multiply a polynomial, or want to multiply a polynomial and return to the original polynomial. The **PTOR** button allows you to convert complex roots input in polar format to rectangular format, prior to doing a multiplication. The **Binomial Multiplication** routine will not compute properly unless the complex roots are input in rectangular format. The **RESET** button allows you to reset the polynomial coefficients to zero, without going through the **ENTER** sequence.

To multiple monomials you first input the root on the two edit lines in the dialog. If the second edit line labelled "**Imag**" is equal to zero, the program will do a real root multiplication. If the "**Imag**" edit line contains a non-zero value, the multiplication will be a complex multiplication.

When doing a complex multiplication you need only to input one of the complex conjugate roots. The program will supply the other root and do the multiplication for both roots. In this case, the order of the resulting polynomial will be increased by two. Since **GSNumerics** is limited to polynomials no bigger than order ten, you can't multiply an order nine polynomial by a complex root, since this would result in an order eleven polynomial. You also can't mulitply a tenth order polynomial, since this would also esult in an eleventh order polynomial.

If you initiate a multiplication with all polynomial coefficients equal to zero, the program assumes you are building a new polynomial and will set the constant term to 1.0 prior to doing the multiplication. This enables you to reset the polynomial coefficients to zero and build an order one polynomial, without having to enter an a_0 coefficient equal to 1.0 with the **Input Polynomial** dialog.

To demonstrate the Binomial Multiplication, we will some build polynomials with the following examples:

Build a polynomial with real roots at -1, -2, and -3

	RESET	
-1	ENTER COMPUTE	Creates binomial $y(x) = x + 1$
-2	ENTER COMPUTE	Creates polynomial $y(x) = x^2 + 3x + 2$
-3	ENTER COMPUTE	Creates polynomial $y(x) = x^3 + 6x^2 + 11x + 6$. This is a third order polynomial containing roots at the desired locations -1, -2 and -3.

Build a polynomial with real roots at 1 and 2 and a pair of complex roots a 3 - i and 3 + i

	RESET	
1	ENTER COMPUTE	Creates binomial $y(x) = x - 1$
2	ENTER COMPUTE	Creates polynomial $y(x) = x^2 - 3x + 2$
3	ENTER	Input the real part in the Real Root edit line
-1	ENTER COMPUTE	Input the imaginary part in the Imag edit line. This creates a fourth order polynomial containing the desired roots. $y(x) = x^4 - 9x^3 + 30x^2 - 42x + 20$.

Given the polynomial $x^3 - 3.4x^2 + 2.5x - .5$, add real roots at 0.5 and 0.75 to the polynomial. You must first enter the given polynomial using the **Input Polynomial** dialog.

.5	ENTER COMPUTE	Add root at 0.5. $y(x) = x^4 - 3.9x^3 + 4.2x^2 - 1.75x + 0.25$
.75	ENTER COMPUTE	Now add the second root a 0.75. $y(x) = x5 - 4.65x^4 + 7.125x^3 - 4.90x^2 + 1.562x - 0.188$

Notice how the program supplied the constant 1.0 to the polynomial when you used the RESET button to set all coefficients to zero.

As you can see it is easy to add roots to an existing polynomial or to construct polynomials with roots at know locations using the **Binomial Multiplication** dialog. These are very important concepts. Electrical engineers use polynomials and their root locations to build electronic filters of different types. These are used in building your stereo receivers and television sets. Also, automatic feedback control systems, used to control aircraft, space vehicles and other machinery items depend heavily on polynomial roots for their design. Having the tools to quickly build polynomials with roots at know locations, will allow you to study polynomials graphically, and learn more about their characteristics.

Division of Polynomials by Binomials

The division of a polynomial by a binomial can be used to determine if a specific value of x is a root of a polynomial or to reduce a polynomial to one of lower order. If the binomial divisor is not a factor of the polynomial, the binomial division will return a remainder. If the divisor is a factor of the polynomial, the remainder will be equal to zero. Binomial Division and Multiplication, give the user complete control of constructing and decomposing polynomials into their individual parts.

You can call the Binomial Division dialog by selecting **Binomial Division** item from the **POLYNOMIAL** menu, or by using the MODE button from another Polynomial Dialog. The **Binomial Division** dialog is shown in Figure 4.8.

When you bring up the dialog, six buttons are active. They are the INPUT, MODE, COMPUTE, QUIT, PTOR and ENTER buttons. After you have computed a binomial division, the LAST button will be activated, allowing you to replace the computed polynomial with the original polynomial. This allows you to undo an error if you mistakenly divide a polynomial, or want to divide a polynomial without disturbing the original polynomial. The PTOR button allows you to convert complex roots input in polar format to rectangular format, prior to doing a division. The Binomial Division routine will not compute properly unless the complex roots are input in rectangular format.

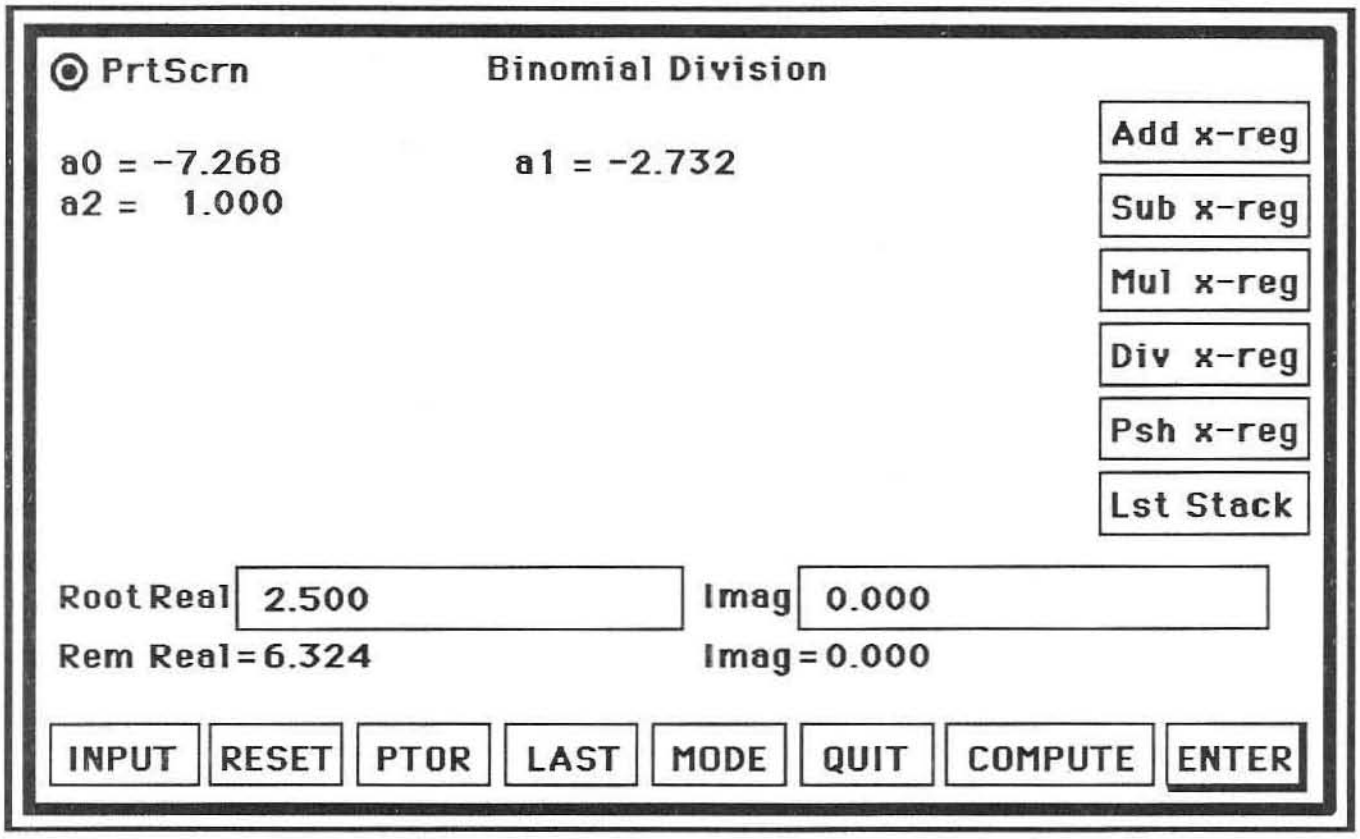

Figure 4.8 Binomial Division dialog

To divide monomials you first input the root on the two edit lines in the dialog. If the second edit line labelled "**Imag**" is equal to zero, the program will do a real root division. If the "**Imag**" edit line contains a non-zero value, the division will be a complex division. When doing a complex division you need only to input one of the complex conjugate roots. The program will supply the other root and do the division for both roots. In this case, the order of the resulting polynomial will be decreased by two.

The polynomial $y(x) = x^5 - 10x^4 + 43.75x^3 - 102.5x^2 + 124x - 60$ contains a pair of complex roots at 2 - i2 and 2 + i2. Reduce the polynomial to a third order polynomial and determine if 1.5 is a real root of the polynomial.

You must first enter the polynomial using the **Enter Polynomial** dialog. Do that now, then:

2	ENTER	Input the real part in the **Real Root** edit line
2	ENTER COMPUTE	Input the imaginary part in the **Imag** edit line. This creates a third order polynomial. The remainder is zero, confirming 2 + i2 and 2 - i2 are roots.

$y(x) = x^3 - 6x^2 + 11.75x - 7.5$.

1.5 **ENTER** **COMPUTE** This returns the polynomial $y(x) = x^2 - 4.5x + 5$. The remainder is zero. Therefore, 1.5 is a real root of the polynomial.

Divide $y(x) = x^3 - 2.5x^2 + 3.1x - .75$ by the binomial $x + 1$. Note that division by $x + 1$ is the same as looking for a root of -1. Is it a root? If not, what is the remainder?

You must first enter the polynomial using the **Enter Polynomial** dialog. Do that now, then:

-1 **ENTER** **COMPUTE** Returns $y(x) = x^2 - 3.5x + 6.6$ with a real remainder of -7.35. Therefore, -1 is not a root of the polynomial.

A bit of caution is in order here. Computers do not compute numbers exactly. In fact, some numbers can't be represented exactly in the digital form used by a computer. Even though a factor may indeed be a root of a polynomial, the program may return some small value as a remainder. This is due to internal rounding and truncation errors. This will occur most often with higher order polynomials, and especially when dividing by complex factors. If you obtain a remainder that is very small, say less than 1e-12, you can be sure that the factor is very close to a root if not actually a root. For most practical purposes, you can consider it to be a root of the polynomial. It is not always possible to determine the exact values for the roots of a polynomial. **GSNumerics** gives you three ways to look for polynomial roots. Binomial division is the first tool. In the next section we will discuss the **GSNumerics** Polynomial Rootfinder. This is a better tool to determine the roots of a polynomial. In a later section, we will show you how to look for polynomial roots graphically. Mathematics is not always exact. Consider the fact that pi is a never ending, non-repeating number, that does not have an exact value.

The Polynomial Rootfinder

The Polynomial Rootfinder will find all real and complex roots and compute the error in the solutions, by substituting the computed roots back into the current polynomial and displaying the result. To use the Polynomial Rootfinder, select the **Roots** menu item from the **POLYNOMIAL** menu. This menu item will only be active when a polynomial has been entered by using the **Input Polynomial** dialog or by constructing a polynomial with the **Binomial Multiplication** dialog.

The Polynomial Rootfinder will find all of the real and complex roots of a polynomial up to order ten. The **Polynomial Rootfinder** dialog is shown in Figure 4.9.

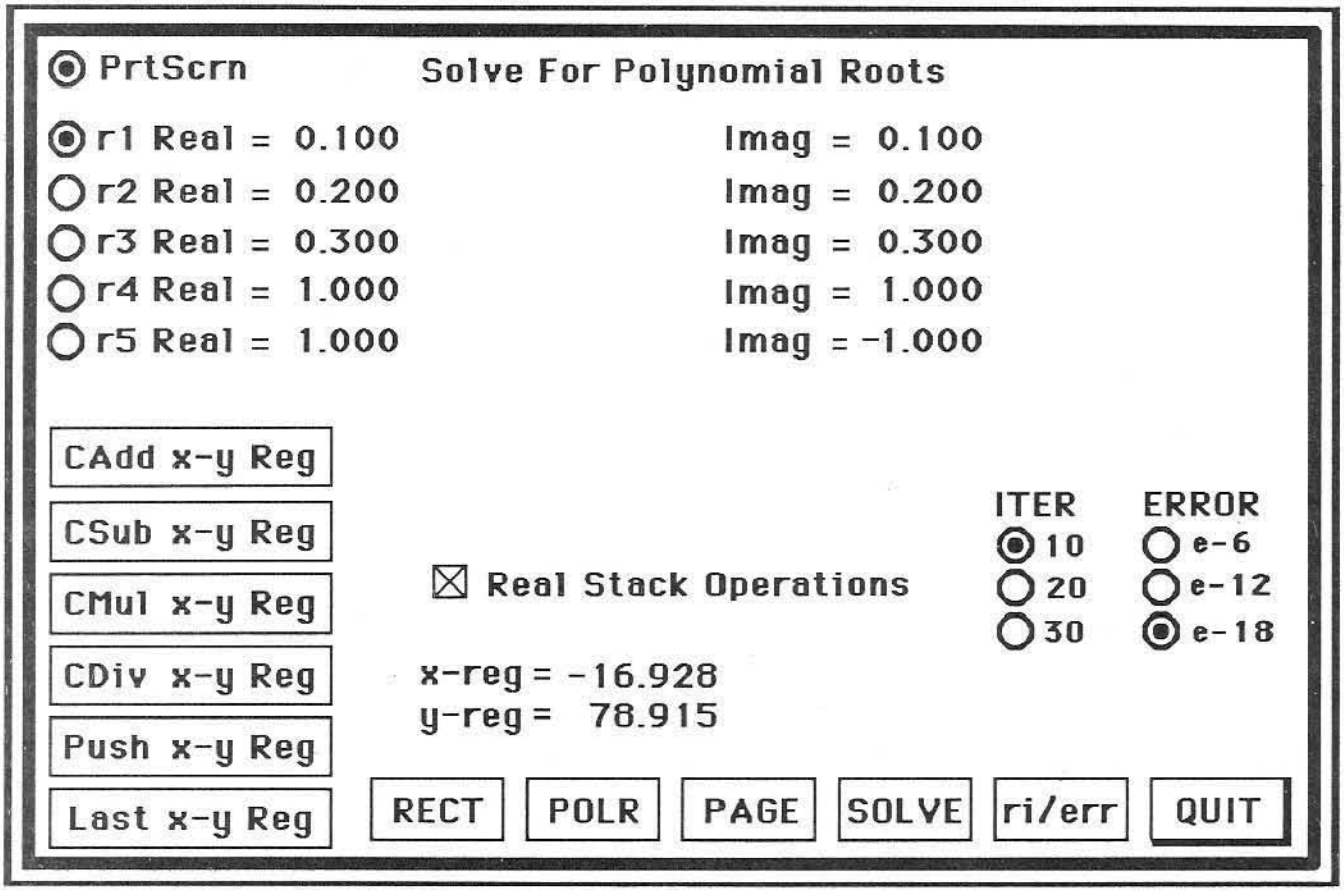

FIGURE 4.9 Solve For Polynomial Roots dialog

The Polynomial Rootfinder operates on the current polynomial, that is the polynomial entered with the **Input Polynomial** dialog or a polynomial created with the **Binomial Multiplication** dialog. To use the Polynomial Rootfinder, you must first enter a polynomial, using one of these two options. If you have not entered a polynomial, the **Roots** menu item, on the **Polynomial** menu will be disabled and you will not be able to select the **Roots** dialog.

The dialog contains two sets of three radio buttons labelled "**ITER**" and "**ERROR**". The Rootfinder uses an iteration algorithm to compute the roots, and these radio buttons set the number of iterations the Rootfinder will go through before quitting and the level of accuracy desired in the result. For instance, if you select the "**ITER**" radio button "**10**" and the "**ERROR**" button "**e-18**", the Rootfinder will iterate until either all root solutions solve the polynomial with an error not exceeding 1e-18 or ten iterations have been completed. It is recommended that you use the default values of ten iterations and error value of 1e-18. You may speed up the process somewhat by selecting an error level of 1e-6, but the solutions will not be as accurate. You may try twenty or thirty iterations. This will only improve the error in a small number of cases, and may significantly increase the computation time.

The following buttons are used to select operations on the **Roots** dialog:

SOLVE This button initiates the polynomial roots solution. **GSNumerics** contains a flag that is set when the current polynomial roots have been solved with the Polynomial Rootfinder. If the roots have been solved, this button will be disabled, except when you first enter the dialog from the menu. Changing either the "**ITER**" or "**ERROR**" will reset the solved flag to "notsolved" and enables this button.

ri/Err This button changes the display output from the polynomial roots to a display of the root errors. If the roots errors have been selected, the button will change the display back to the polynomial roots. The root errors are always displayed in polar format, since the magnitude of the error is the best indication of how accurate the roots were computed. This button is disabled, unless the polynomial roots have been solved.

PAGE Only five roots or errors are displayed on the screen at one time. If the order of the current polynomial is greater than five, this button will be available. It is used to select the second page, where roots (or errors) six through ten are displayed. Keying on this button will toggle between page one and page two. It is disabled, if the polynomial roots have not been solved. It will not be present on the screen, if the order of the polynomial is less than six.

POLR Changes the root display to polar format. Dimmed and disabled when the roots are displayed in polar format.

RECT Changes the root display to rectangular format. Dimmed and disabled when the roots are displayed in rectangular format.

The stack operation buttons perform the same function as previously described, except the user can select the option of only performing real stack operations by clicking on the check box labelled "**Real Stack Operations**". When this option is selected, a stack operation will only involve the real part of the polynomial root, and the operation will not be a complex operation. If this option is not selected, the operation will be complex and will involve both the real and imaginary parts of the polynomial root.

The specific root to be added, subtracted, multiplied or divided is determined by the radio buttons to the immediate left of the label "**r1 Real=**", "**r2 Real=**", etc. If the radio button to the left of "**r5 Real=**" is selected, the real and complex part of root number five will be used in any complex stack operation, or only the real part of root number five will be used in stack operations, if the "**Real Stack Operations**" option is selected.

We will work a few problems, finding all of the real and complex roots of polynomials, using the Polynomial Root Finder.

Find the all real and complex roots of the polynomial $y(x) = x^3 - 3.6x^2 - 6.25x + 9.9$. Also determine the error in each root solution. You must first enter the polynomial using the **Input Polynomial** dialog, then select the **Polynomial Roots** dialog.

SOLVE This initiates the root solution. After a few seconds the rootfinder will have completed it's task and the roots will be shown as follows:

r1 Real = 1.100 Imag = 0.000
r2 Real = -2.000 Imag = 0.000
r3 Real = 4.500 Imag = 0.000

The roots are 1.1, -2.0 and 4.5. All three roots are real.

ri/Err The root solutions are replaced by the errors for each root. The errors are always displayed in polar format.

e1 Mag = 0.000 Ang = 0.000
e2 Mag = 0.000 Ang = 0.000
e3 Mag = 0.000 Ang = 0.000

You return to the roots display by keying on the **ri/Err** button.

Find all roots of the polynomial $y(x) = x^6 - 5.60x^5 + 11.11x^4 - 11.356x^3 + 4.51x^2 - 0.708x + 0.036$. Enter the polynomial using the **Input Polynomial** dialog.

SOLVE This initiates the root solution. After a few seconds the rootfinder will have completed it's task and the roots will be displayed as follows:

r1 Real = 0.1000 Imag = 0.000
r2 Real = 0.2000 Imag = 0.000
r3 Real = 0.3000 Imag = 0.000
r4 Real = 1.000 Imag = -1.000
r5 Real = 1.000 Imag = 1.000

PAGE You must page to see root six.

r6 Real = 3.000 Imag = 0.000

The real roots are at 0.100, 0.200, 0.300 and 3.000. There is a complex conjugate pair of roots at 1 + i and 1 - i.

Special Polynomial Operators

The following operators, when input into the Scientific Calculator Input Register, will perform the following polynomial operations, without bringing up the appropriate Polynomial dialog:

PR[	Resets all polynomial coefficients to zero. **Caution:** You will not be provided with an Alert dialog box asking if you are sure you want to zero the coefficients.
PC[n,c	Sets an individual polynomial coefficient **n** to a value **c**. For example **PC[3,4.5**, sets the current polynomial coefficient a_3 to the value 4.5.
PI[c_n, . .,c_0	Enters a new polynomial, replacing the current polynomial. For example, the string **PI[1,-3.2,4.5** will enter the new polynomial $y(x) = x^2 - 3.2x + 4.5$ as the current polynomial. Zero coefficients must be entered, but may be entered a two commas. The string **PI[1,,,,3** will enter the polynomial $x^4 + 3$. The following string is an equivalent entry **PI[1,0,0,0,3.**
PG[l,u,c1,c2	Initiates a polynomial graph of the current polynomial from $x_1 = l$ to $x_2 = u$, with graph clip limits (optional) of **c1** and **c2**. For example, the string **PG[-2,2** will graph the current polynomial from x = -2 to x = 2. The string **PG[0,4,.1,-.1** will graph the current polynomial from x = 0 to x = 4, with graph clip limits set to 0.1 and -0.1. An equivalent entry would be **PG[,4,.1,-.1.** The program will supply the implied zero.

The special operators provide a fast method of performing some routine tasks on polynomials, bypassing the polynomial dialog screens.

Polynomial File Operations

GSNumerics provides file operations to allow you to store polynomials and their solved roots to a data file on disk. You may recall the individual files, at anytime, for further use. In addition, if the user saves a **Session**, the polynomial coefficients and roots (if solved) will be recalled with the **Session Recall** menu item.

To save a polynomial file, select the **Save Polynomial File** from the **Polynomial** menu. When the **Standard File** dialog comes up, enter the name under which you want to save the current polynomial, and press **Save**. The current polynomial will be saved to the disk file with the name you selected.
To recall a polynomial, select the **Load Polynomial File** from the **Polynomial menu**. When the **Standard File** dialog appears, select the desired polynomial file, by double clicking on the name, or by selecting it and keying on **Open**. The selected polynomial file will be loaded into the program. **Caution**:

recalling a polynomial file will replace the current polynomial with the polynomial in the file. The current polynomial will be lost.

FUNCTIONS

The **Function Operations** dialogs are used to do computations on non-polynomial functions. Although, you may enter polynomials and do polynomial computations with the **Function Operations** dialog, the computations will not be as accurate and will take longer, than if done with the **Polynomial Operations** dialogs.

GSNumerics can analyze any valid function y(x), that is input according to the rules outlined in the **Direct Function Entry Section** of Chapter 2 (**SCIENTIFIC CALCULATOR**). Examples of valid functions are:

1. **sin(x)/cos(x)**
2. **log(a)^3.2*int(acos(.978))**
3. **d+e/v*sin(pi/3)**

When analyzing functions, you should be aware of the fact that the function parser replaces the value in the [X] memory location with the input independent variable x, and the [X] memory location value will be lost. For example, if you solve a particular function for x = 1.23, the [X] memory location will be assigned the value of x = 1.23 by the function parser. Function operations rely heavily on the [A] to [Z] memory locations. Many functions will be entered in terms of constants **a** through **z**, with the constants referred to in the functions. **Caution:** don't assign a constant to the memory location [X], since it will be changed by the parser. If you do this, the solution will not be as you expected, in fact, it will be in error. Other memory location values are left untouched during function operations.

Five operations are available with the **Function Operations** dialogs:

1. Input a new function. This becomes the current function.
2. Solve the current function for a specific x value.
3. Compute the slope of the current function at a specific value of x.
4. Find a real root of the current function, between two specific x values.
5. Compute the area under the function curve, between two specific x values, using either the Trapezoidal, Simpson or Romberg method of integration.

You may select a specific **Function Operations** dialog, by selecting the item from the **FUNCTION** menu.

Entering Functions

To enter a new function select the item from the **FUNCTION** menu. This will bring up the **Function Entry** dialog shown in FIGURE 5.1.

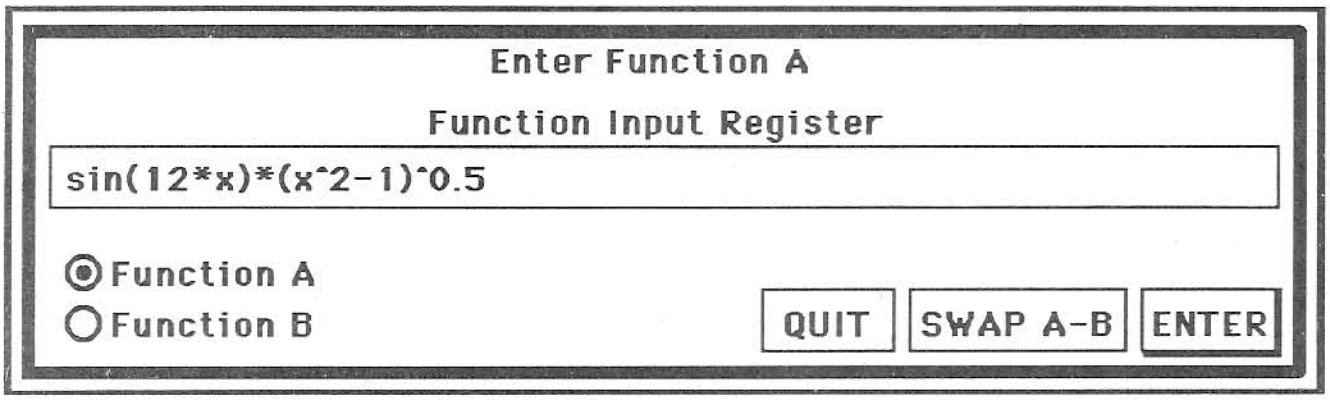

FIGURE 5.1 Function Entry dialog

The **Function Entry** dialog contains one edit line and one ENTER button. When the dialog first appears, the current function is displayed in the edit line. If no function has been entered, the edit line will be blank. To enter a new function you merely type in the function and click on the ENTER button. When you click on the ENTER button, the program checks the function for validity. If it is a valid function, adhering to the rules of **Direct Function Entry**, the **Function Entry** dialog will close returning you to the calling screen. You may use the **Cut** (Ó-x), **Copy** (Ó-c) and **Paste** (Ó-v) operations to edit a function. You may highlight and select areas of the function by **dragging** the mouse.

If the entered function is not valid, you will be returned to the **Function Entry** dialog after clicking on the ENTER button, and the function will drawn in reverse highlight. This will alert you to the fact that the entered function was not a valid function. The system will sound a "beep" to alert you that an invalid function has been entered. Remember that functions must be entered exactly as they are shown on the **Scientific Calculator** screen and only those functions with an asterisk (*) following them, are valid in **Direct Function Entry**. They are not case sensitive, SIN(X) will get the same result as sin(x).

You may use the cut, copy and paste operations to make changes to function that has already been entered. Of you don't know how to use the cut, copy and paste, you will find this explained in the manual that came with your Apple IIGS.

Solving a Function for x

The basic function operation is solving a function for a given x. You may solve a function for any x by first selecting the **Solve Function** item from the **FUNCTION** menu. This will bring up the **Solve Function** dialog shown in FIGURE 5.2.

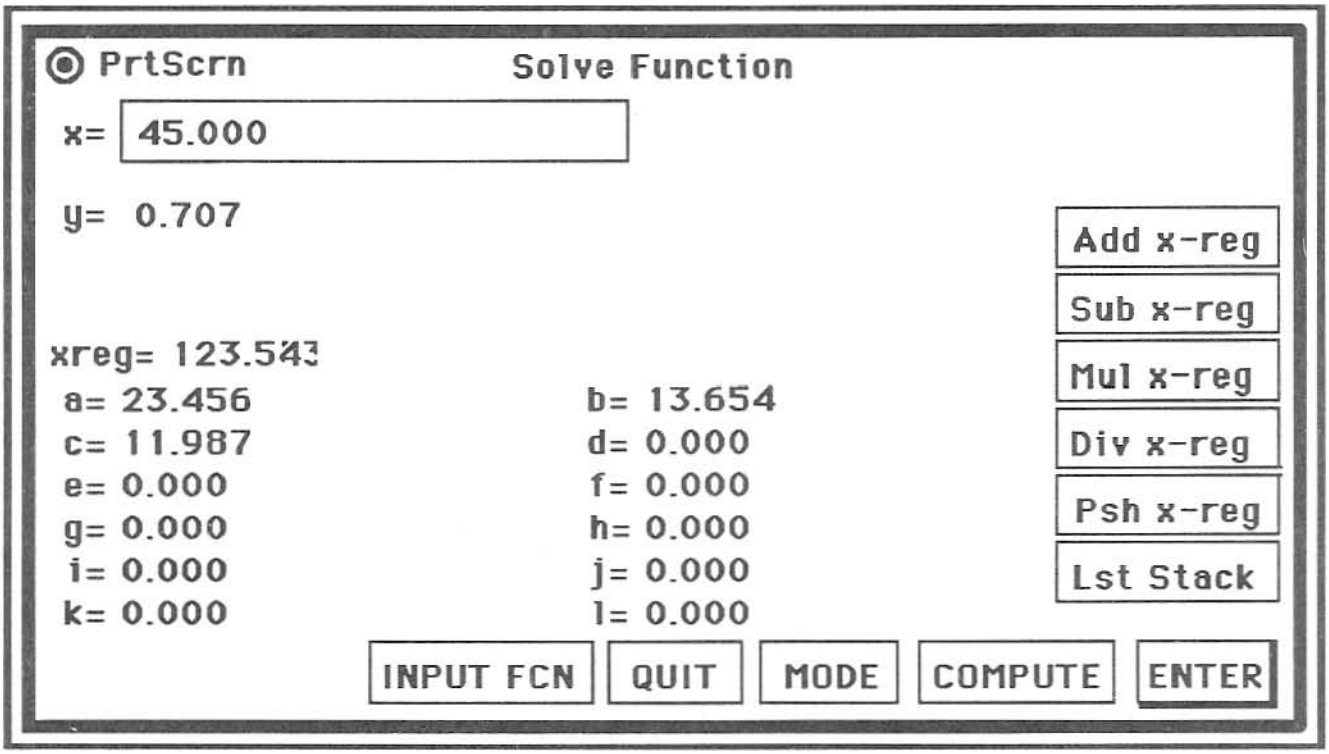

FIGURE 5.2 Solve Function dialog

This dialog contains one edit line and eleven buttons. It also shows the current contents of the A to L memory locations. This is to assist you in using the memory constants **a-l** in function operations. The contents of the **Scientific Calculator** x-reg are show to the right of the label "**x-reg=**" and there is a "**y=**" label where the solution will be shown.

The current function will be shown in the window title bar. Clicking on the **INPUT FCN** button will bring up the **Function Entry** dialog, allowing you to edit the current function or input a new current function. The **MODE** button is used to advance to the next **Function Operations** dialog, without having to return to the main menu. The **QUIT** button is used to quit the **Function Operations** dialog and return to the **Scientific Calculator** screen.

The **Add x-reg**, **Sub x-reg**, **Mul x-reg**, **Div x-reg**, **Psh x-reg** and **Lst Stack** buttons are used to do stack operations with the results obtained when solving the current function. This enables you to use the **Function Operations** dialogs to obtain solutions to problems and use them in larger problems using the **Scientific Calculator**. The label "**xreg=**" shows the current value of the **x-reg**. This will be updated, to the correct value as you do stack math operations, using the stack buttons. You can undo errors with the **Lst Stack** button. To solve the current function you input the value of x, where you wish to solve the function, and press the **return** button or click on the **ENTER** button. The solution is then displayed to the right of the label "**y=**".

The following edit line entries are valid:

1. Entering a number, followed by the **ENTER** button or **return** key, will enter the number as the x and solve the equation.
2. Entering a function, using direct function entry, will set the value of x to the function value, and solve the current function at that x.
3. Entering a memory assignment with direct function entry (i.e. a = 2.35 or c = sin(b)/cos(d)) assign the result of the function to the referenced memory location, and will not solve the function. This allows you to change the value of the constants assigned to the current function.

We will now enter a function and solve it at various x values.

	INPUT FCN	This will bring up the **Function Entry** dialog.
log(x)*x^2	**ENTER**	Enter function and return to the **Solve Function** dialog. The current function is now **y(x) = log(x)*x²**. It is displayed in the window title bar.
33.2	**ENTER**	Solves for **x = 33.200** and displays the answer **y = 1676.659**.
c=2.4	**ENTER**	Sets memory location **c** to a value of 2.4.
c	**ENTER**	Solves for **x = 2.400** and displays the answer **y = 2.190**.
3*pi	**ENTER**	Solves the function **x = 3*pi = 9.425** then solves the current function for **x = 9.425** and displays the answer **y = 86.541**.
	INPUT FCN	This will bring up the **Function Entry** dialog.
fact(x)/x	**ENTER**	Enter function and return to the **Solve Function** dialog. The current function is now **y(x) = fact(x)/x**. It is displayed in the window title bar.
5	**ENTER**	Solves for **x = 5.000** and displays the answer **y = 24.000**.
23	**ENTER**	Solves for **x = 23.000** and displays the answer **y = 1.124e+21**.

int(ln(98654)/10) **ENTER** Solves the function **x = int(ln(98654)/10)** then solves the current function for **x** and displays the answer **y = 1.000.**

It is very easy to solve function for various values of x using the **Solve Function** dialog.

Solving for the Slope of a Function

To solve for the slope of the current function you will use the **Function Slope** dialog, shown in FIGURE 5.3. You may select this dialog by selecting the **Slope** item under the **FUNCTION** menu, or by clicking on the button from another **Function Operations** dialog until the **Compute Function Slope** dialog appears.

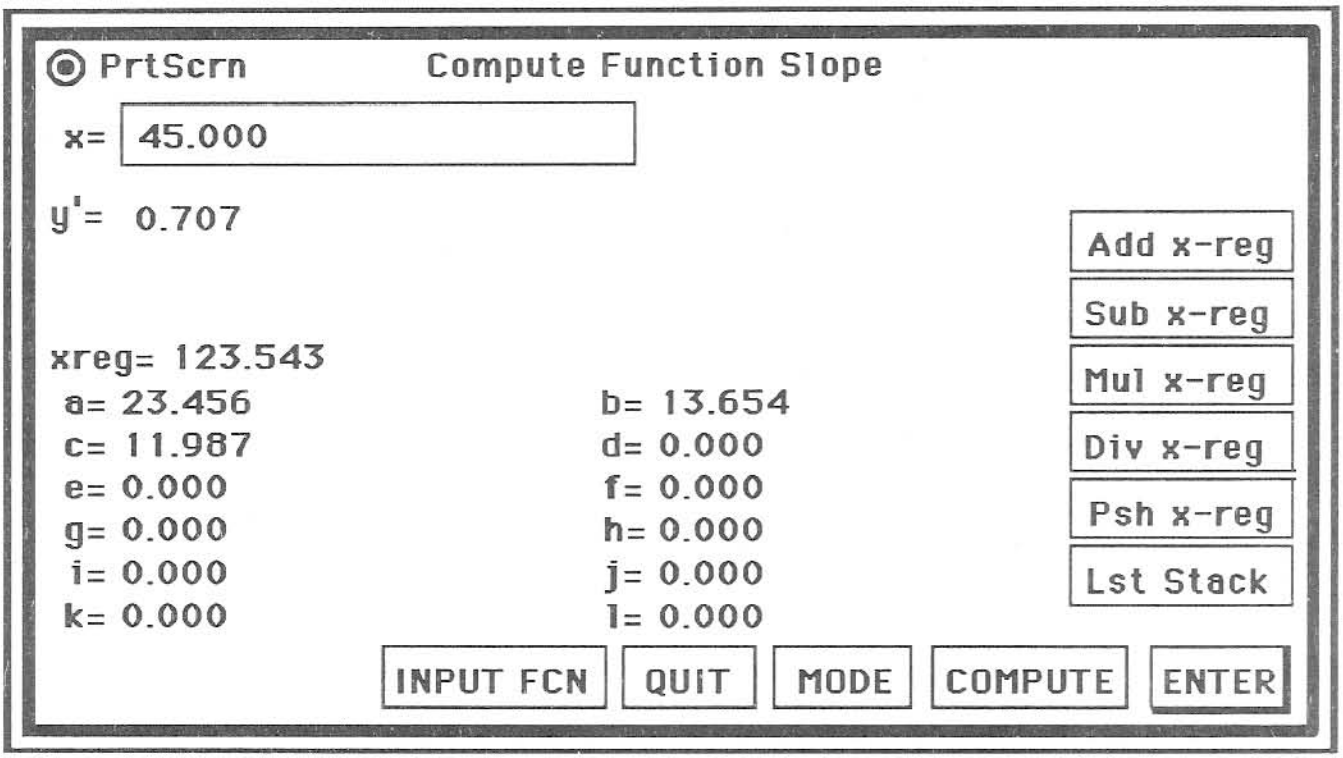

FIGURE 5.3 Compute Function Slope dialog

This dialog operates exactly like the **Solve Function** dialog, except it computes the slope at the input x value, rather than solving the function for x. The slope is shown to the right of the label "**y'**". This is read as "**y prime of x=**".

When solving for the slope of functions involving trigonometric functions you must input the angle in the same units as the calculator **Angle Mode** setting on the **Scientific Calculator**. In other words, if the function contains trigonometric functions and you have the calulator set to **Degrees Mode**, you must input the angles in degrees.

CAUTION:

Computation of the slope of general functions using digital computers is difficult. You must be careful if the function slope is steep (i.e. it's absolute value evaluates to a large number), since the answer returned by this computation may not be accurate in such cases. If the slope returned by this computation is large, you should check it by other means, prior to using the computed slope in other calculations.

We will now input some functions, and solve for the slope at different points.

	INPUT FCN	This will bring up the **Function Entry** dialog. You will enter the function as explained in the first part of this chapter.
2*cbrt(x)	ENTER	Enter function and return to the **Solve Function** dialog. The current function is now **y(x) = 2*x^1/3**. It is displayed in the window title bar.
33.2	ENTER	Solves for **x = 33.200** and displays the answer **y = 0.065**.
	INPUT FCN	
ln(x)	ENTER	Enter function and return to the **Solve Function** dialog. The current function is now **y(x) = ln(x)**. It is displayed on the window title bar.
3	ENTER	Solves for **x = 3.000** and displays the answer **y = 0.3333**.
6.5	ENTER	Solves for **x = 6.500** and displays the answer **y = 0.1538**.
	INPUT FCN	
2*sin(x)*cos(x)	ENTER	Enter function and return to the **Solve Function** dialog. The current function is now **y(x) = 2*sin(x)*cos(x)**. It is displayed in the window title bar.
35	ENTER	Compute the slope at **x = 35.000** degrees. The slope is **y' = 0.6840**.
pi	ENTER	The slope is **y' = 2.000**.

	INPUT FCN	
tan(x)	ENTER	Enter function and return to the **Solve Function** dialog. The current function is now **y(x) = tan(x)**. It is displayed on the window title bar.
56.3	ENTER	Compute the slope at **x = 56.300** degrees. The slope is **y' = 3.2483.**
45.0	ENTER	Compute the slope at **x = 45.0** degrees. The slope is **y' = 2.000.**
89.95	ENTER	The computed slope is **y' = 1323834.015.** Better check this one. Look at the large slope. The actual slope of tan(x) at x = 89.95^0 is y' = 1313122.873!!!

Computing the Real Roots of a Function

To compute the real roots of a function, select the **Roots** item under the **FUNCTION** menu. This will bring up the **Compute Function Roots** dialog shown in FIGURE 5.4.

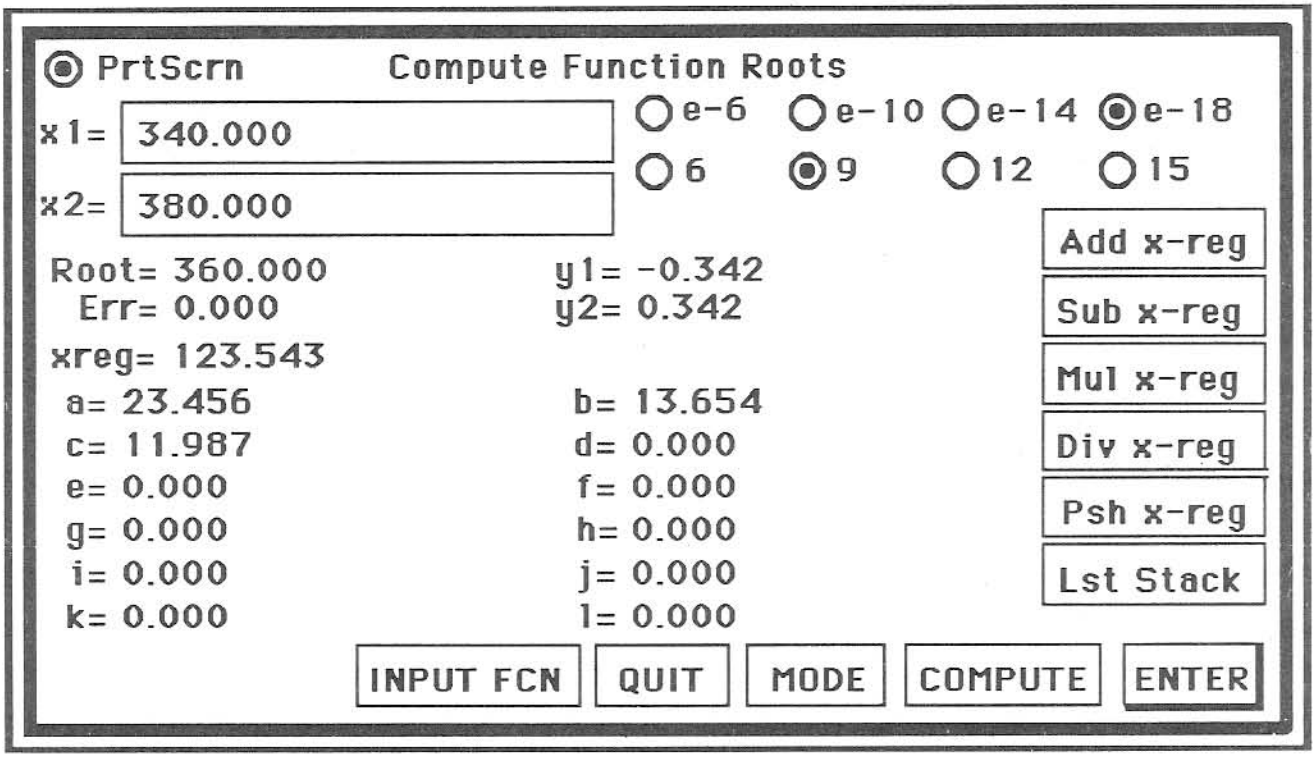

FIGURE 5.4 Compute Function Roots dialog

The **Compute Function Roots** dialog has the eleven buttons contained on the previous dialogs, plus eight radio buttons. The radio buttons are are divided into two groups. The first group of four buttons is labelled "**e-6**", "**e-10**", "**e-14**", and "**e-18**". The second group of four radio buttons are labelled "**6**", "**9**", "**12**", and "**15**". Computation of function roots is an iteration algorithm and these buttons set the number of iterations the program will go through in computing the solution and the amount of error desired in the solution. The iteration process will end when either the number of iterations selected have been done or the root has been computed to a point the error is less than the error set with the radio button. For instance, suppose the user has selected the "**e-6**" error limit and the "**9**" iteration button. When the current function solved with the computed root substituted for x is less than 1e-6, the iteration will terminate. If the number of iterations reaches the limit of 9 before the current function is solved to a value not exceeding 1e-6, the iteration will end. Most roots are best found using the initial settings of "**e-18**" and "**9**" iterations. Selection of 12 or 15 iterations, may give a slightly better answer, but will extend the time required to compute the root. You can speed up the process by selecting 6 iterations, if the answer is not required to be real precise.

In computing roots, the user must supply two initial guesses that straddle a root. These guesses are entered in the two edit lines labelled "**x1=**" and "**y1=**". The two labels "**y1=**" and "**y2=**" will show the solution of the current function when solved for the initial guesses x_1 and x_2. The function value will be computed at these points when the guesses are entered with the return key or by clicking on the **ENTER** button. If the guesses do not straddle a root, you will be warned and computation of the root will not continue.

A root is straddled when one solution of the current function is less than zero and the other solution of the function is greater than zero. If the guesses do not straddle a root, the root finder will alert the user with the message "**Guesses must straddle a function root y1 * y2 < 0.0**" and beep the computer speaker.

After entering two guesses, that straddle a root, clicking on the **COMPUTE** key will start the root finder computation. After the root is computed, the root will be shown to the right of the label "**Root=**", and the error will be shown to the right of the label "**Err**". Remember, the error is the value of the current function solved with x equal to the computed root. If the root computation is exact, the error will be zero.

The exact location of roots is not always know. The **Graph Function** dialog can be used in conjunction with the **Compute Function Roots** dialog to find roots very quickly. The function is first graphed and a **Graph Range Selection** is made with the cursor on each side of a root. The range upper and lower values are passed to the **Compute Function Roots** dialog automatically. When you leave the graph

graph dialog and return to the **Compute Function Roots** dialog, these values will be in the "**x1=**" and "**x2=**" edit lines. You may ten click **COMPUTE** to compute the root. This allows you to visually located the approximate position of roots with the graph dialog and then compute them exactly with the roots dialog. We will now compute the roots of several functions.

Find a root of the function **y(x) = 2*sin(12*x)+(1-x²)** between **1.1⁰** and **1.4⁰**. Make sure the **trig mode** is set to degrees (**DM**) before doing this calculation.

2*sin(12*x)+(1-x^2)	**ENTER**	Enter function and return to the **Compute Function Roots** dialog.
1.1	**ENTER**	Enter **x1 = 1.100⁰. y1 = 0.247.**
1.4	**ENTER**	Enter **x2 = 1.400⁰. y2 = -0.382.** We know a root is straddled since **y1** > 0.000 and **y2** < 0.000.
	COMPUTE	**Root = 1.228⁰. Err = 0.000.** The function has a real root located at **x = 1.228⁰.**

Does the function **y(x) = 2*sin(12*x)+(1-x²)** have a root between **0.700** and **1.200**? Enter the function with using the **Function Entry** dialog.

-1	**ENTER**	Enter **x1 = 0.700. y1 = -0.8022.**
-3	**ENTER**	Enter **x2 = 1.200. y2 = 0.0574.** We know a root is not straddled since **y1** > 0.000 and **y2** > 0.000.

Does the function **y(x) = 2*sin(12*x)+(1-x²)** have a root between **0⁰** and **-10⁰**?

-0	**ENTER**	Enter **x1 = 0.000⁰. y1 = 1.000.**
-10	**ENTER**	Enter **x2 = -10.000⁰. y2 = -100.732.** There is a root since y1 * y2 < 0.000.
	COMPUTE	**Root = -0.813⁰. Err = 0.000.**

Does the function **y(x) = 2*log(x) * eˣ** have a root between **0.1** and **5.0**?

-0	**ENTER**	Enter **x1 = 0.100. y1 = -2.2103.**
-10	**ENTER**	Enter **x2 = 5.000. y2 = 207.4727.**
	COMPUTE	**Root = 1.000. Err = 0.000.**

Computing the Area under a Function Curve

The fourth type of function calculation is computing the area under a function curve, between two points x_1 and x_2. This type of problem is encountered quite frequently in engineering and science. Computing areas under curves is called integrating the function f(x) from a to b, and is shown in math terms as:

$$\text{area} = \int_a^b f(x)dx$$

This is read as "the integral of f, with respect to x, from a to b". Computation of the integral is done using the **Compute Function Area** dialog. You may select this dialog by selecting the **Area** item under the FUNCTION menu or by clicking on the MODE button, when you are in another **Function Operations** dialog. The area computation is always done on the current function, that is entered using the **Function Entry** dialog, as demonstrated earlier in this chapter.

The **Compute Function Area** dialog is shown in Figure 5.5. This dialog is very similar to the **Compute Function Roots** dialog, covered in the previous section.

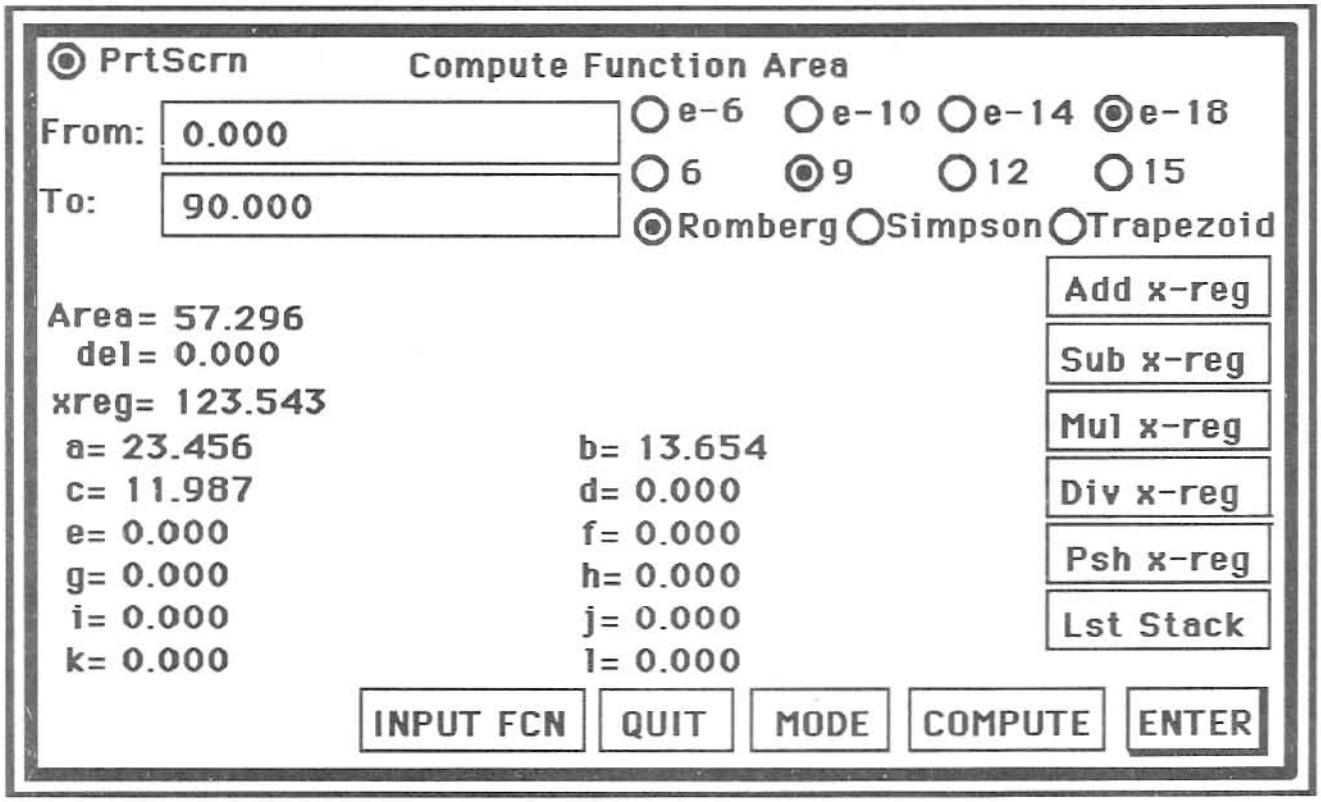

FIGURE 5.5 Compute Function Area dialog

There are three addition radio buttons on this dialog. They are labelled "**Romberg**", "**Simpson**" and "**Trapezoid**". These are three "orders" of integration, with Trapezoid being the lowest order and Romberg being the "highest" order. Normally, the higher order integration method will yield the best results. The "lower" order types are somewhat faster, and may yield good results on smooth functions, while the "higher" order integration is slower and will yield better results on functions whose graphs are not really smooth. The three types of integration are included in **GSNumerics** to allow the user to investigate the results when using different types of integration. When first studying integration, it is not unusual to do Trapezoid and Simpson integrations "by hand", to demonstrate the idea. You will normally use the Romberg integration, when solving for areas, since the answer will normally be more accurate.

As explained in the previous section, integration is an iteration process. We suggest you use the default settings of "**e-18**" and "**9**", since this is a good trade-off for speed and accuracy. Selecting "**12**" or "**15**" iterations will lead to excess computation times and is not necessary in most cases. You may speed up the process by selecting "**6**" iterations and the error limit "**e-6**", but the answers will not be as good. These decisions are up to the user, depending on what type problem is being solved and how accurate the answers need to be.

The area under the curve is displayed to the right of the label "**Area=**". Integration is a technique of dividing the area into small pieces and summing these pieces. The "**del =**", means "**delta area**" and tells you how big the last small area added to the total area was. Each iteration will add a smaller delta to the total. Thus, "**del =**" is an indication of how close the computation is to the actual area. If the area is exact, the "**del =**" term will be zero.

To compute an area you enter the lower limit of integration "**a**" in the first edit line labeled "**From**" and the upper limit of integration "**b**" in the second edit line labeled "**To**", then click on the **COMPUTE** button. After a short time for computation, the area will be displayed by the label "**Area =**" with the delta shown by the "**del =**" label. To demonstrate the use of the **Compute Function Area** dialog, we will compute some function areas.

Compute the area under the curve $2*\sin(12*x)+(1-x^2)$ from a = 0.00 to b = 1.5. If the function is not entered, enter it using the **Function Entry** dialog. Make sure the calculator **trigmode** is set to degrees.

0.0	**ENTER**	Enter the lower limit of integration **a = 0.000**.
1.5	**ENTER**	Enter the upper limit of integration **b = 1.500**.
	COMPUTE	The result is Area =3.0924.

Compute the area from a = 1.5 to b = 0.0 for the same function.

1.5	ENTER	Enter the lower limit of integration **a = 1.5**.
0.0	ENTER	Enter the upper limit of integration **b = 0.0**.
	COMPUTE	The result is Area = **-3.0924**. The area is negative, since the limits were reversed. We would expect this to be the result of reversing the integration limits.

Compute the area of the of y(x) = **log(x)+e^{x}sin(3x)** from x = 1 to x = 6.

	INPUT FCN	
log(x)+ee^x*sin(3*x)	ENTER	Enter function and return to the **Compute Function Roots** dialog.
1.1	ENTER	Enter lower limit of integration a = 1.000.
1.4	ENTER	Enter upper limit of integration b = 6.000.
	COMPUTE	This one takes several seconds to do the computation. The area under the curve from **1.000** to **6.000** is **Area = 106.788**.

Special Function Operators

The following operators, when input into the Scientific Calculator Input Register, will perform the following polynomial operations, without bringing up the appropriate Function dialog:

FI[(fcn)	Inputs a new current function. For example **F[sin(2*x)/ln(x)**, will replace the current function with the function **sin(2*x)/ln(x)**.
FG[l,u,c1,c2	Initiates a function graph of the current function from x_1 = **l** to x_2 = **u**, with graph clip limits (optional) of **c1** and **c2**. For example, the string **FG[-2,2** will graph the current function from x = -2 to x = 2. The string **FG[0,4,.1,-.1** will graph the current function from x = 0 to x = 4, with graph clip limits set to 0.1 and -0.1. An equivalent entry would be **FG[,4,.1,-.1**. The program will supply the implied zero.

The special operators provide a fast method for performing routine polynomial tasks. By using the special operators, you may enter a new function or graph the existing function, without being required to bring up the appropriate dialog.

Function File Operations

GSNumerics provides file operations to allow you to save the current function to a data file on disk, when a **Session Save** is done. The saved function will be recalled as the current function when you do a **Session Recall**. The **Session Save** and **Session Recall** operations are initiated by selecting them under the **FILES** menu.

MATRICES

A rectangular array of numbers enclosed by a pair of brackets is called a matrix. The order of a matrix is given by the number of rows and columns the matrix contains. The general form of a matrix is shown in Figure 6.1, for a matrix with **m** rows and **n** columns. This would be an order **m** by **n** matrix.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Figure 6.1 Matrix definition

A **square matrix** is a matrix that has the same number of rows and columns. A **symmetric matrix** is a square matrix whose off diagonal, symmetric elements are equal (**i.e.** $a_{12} = a_{21}$, $a_{13} = a_{31}$, $a_{24} = a_{42}$). Two off diagonal elements are symmetric, if their subscripts are the same when reversed, and they contain the same numerical value. The diagonal of a square matrix is the elements a_{11}, a_{22}, a_{33},...,a_{mn}. The square matrix, whose off diagonal elements are all equal to zero, and whose diagonal elements are equal to one is called the **Identity matrix**, normally designated by **I**. Several examples of matrices are shown in Figure 6.2

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 5 & 2 \\ 7 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 7 & 1 \\ 4.5 & 8 & 3 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 1 & 3 & 5 \\ 3 & 7 & 1 \\ 5 & 1 & 3 \end{bmatrix} \quad \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 6.2 Types of Matrices

In Figure 6.2 matrix **A** is a **3 x 2 matrix**, matrix **B** is a **3 x 3 square matrix**, matrix **C** is a **3 x 3 symmetrical matrix** and matrix **I** is a **3 x 3 identity** matrix. A matrix may have complex elements, in which case it is called a **complex matrix**. Figure 6.3 shows a **3 x 3 complex matrix**.

$$\begin{bmatrix} 1.0 + i3.4 & 3.0 - i2.1 & 5.2 + i6.1 \\ 6.2 - i6.0 & -7.0 + i6.2 & 1.0 - i0.0 \\ 4.5 + i1.7 & 8.0 - i12.0 & 3.0 + i2.0 \end{bmatrix}$$

Figure 6.3 Complex Matrix

GSNumerics provides the operations for either real or complex matrices shown in Figure 6.4. Matrix operations that involve two matrices such as Addition, Subtraction and Multiplication, operate on two user input matrices **A** and **B**, with the result of all matrix operations, being placed in the result matrix **R**. The user may swap the elements of the input matrices **A** and **B**, or copy the elements of the result matrix **R** to either matrix **A** or **B**.

Input Matrix	Input a real or complex matrix
Matrix Addition	Matrix addition of two matrices (**A** + **B**). The number of rows in **A** must equal the number of rows in **B** and the columns of **A** must equal the columns in **B**.
Matrix Subtraction	Matrix subtraction of two matrices (**A** - **B**). The number of rows in **A** must equal the number of rows in **B** and the columns of **A** must equal the columns in **B**.
Matrix Multiplication	Matrix multiplication of two matrices (**A** * **B**). The number of columns in **A** must equal the number of rows in **B**. The resultant matrix **R** will have the same number of rows as **A** and the same number of columns as **B**.
Compute Transpose	The transpose of an **m** x **n** matrix will be an **n** x **m** matrix.
Scalar Multiplication	Multiplies each element of a matrix by a real or complex scalar constant.
Compute Determinate	Computes the determinate of a **square matrix**.
Matrix Inversion	Inverts a **square matrix** and computes the matrix determinate.
Matrix Memory	Save user input matrices or result matrices in either of two matrix memory locations for later recall and use.
Stack Operations	Permits the user to pass the determinate of matrices to the stack, using the math operators: add, subtract, multiply, divide and push.
Matrix Files	Allows the user to save matrices and results of matrix operations to disk files, and to recall these files for later use.
View Matrix	Allows the user to view the contents of matrices, with complex matrix elements displayed in either rectangular or polar format.

Figure 6.4

To initiate matrix operations select the **Operations** item from the **MATRIX** menu. The **Matrix Operations** dialog is shown in Figure 6.5.

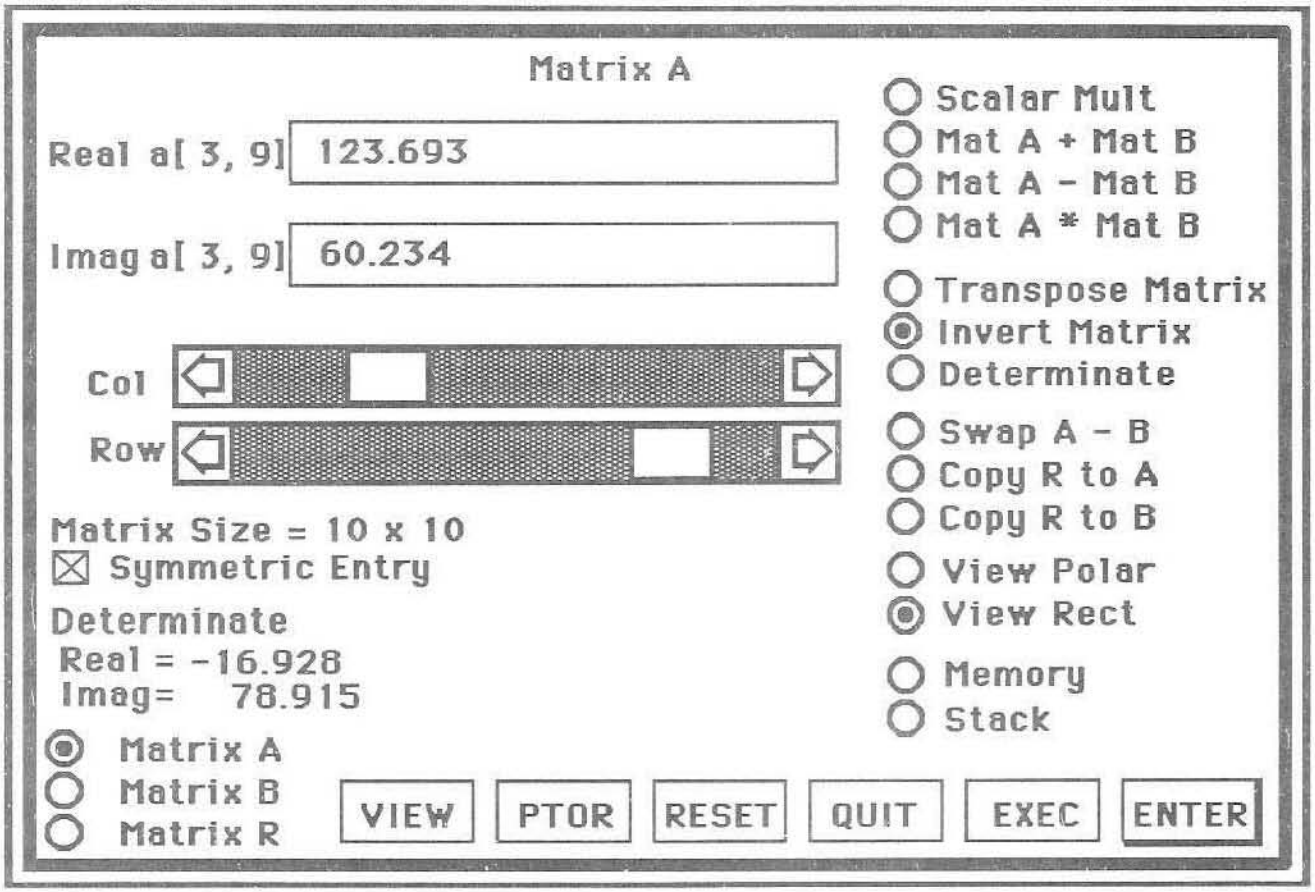

Figure 6.5 Matrix Operations dialog

Matrix operations are done using three matrices. Matrix **A** and Matrix **B** are user matrices and are used to input the user matrices. Matrix **R** is the result matrix and will contain the result of matrix operations. Matrix **R** can't be edited or changed by the user. Matrices **A** and **B** are the operands of matrix operations involving one or two matrices. The label at the top center of the screen tells the user which matrix is currently being represented on the screen. The current matrix in Figure 6.5 is Matrix **A**. The current matrix is selected by keying on one of the radio buttons in the left, lower corner of the dialog. These radio buttons are labelled: "**Matrix A**", "**Matrix B**", and "**Matrix R**". If the result matrix **R** is empty, the "**Matrix R**" button will be disabled.

Matrices **A** and **B** can be in either of two modes, the **entry mode** or the **view mode**. Matrix **R** will always be in the **view mode**, when it is shown on the screen. If **A** or **B** is in the **entry mode**, either one or two edit lines will be on the screen, depending on whether the matrix has been defined as a real or a complex matrix. If they are in the **view mode**, the edit lines will not be on the screen. If a matrix is complex and is in the **view mode**, the two radio buttons labelled "**View Polar**" and "**View Rect**" will be enabled.

The **Matrix Operations** dialog contains two scroll bars for scrolling through the current matrix. The top bar scrolls the **matrix columns** and the bottom bar scrolls the **matrix rows.** The matrix size is always shown, on the left side of the screen, under the row scroll bar.

Beneath the matrix size label is a check box control labelled "**Symmetric Entry**". This is used to enter symmetric matrices. If this control is selected the user needs to only enter the off diagonal terms of a matrix one time, and the program automatically enters the corresponding symmetric entry. For example, with the control selected (indicated by a check in the control), entry of a number in element $\mathbf{a_{12}}$, will automatically enter the same number in element $\mathbf{a_{21}}$, entry of a number in element $\mathbf{a_{13}}$, will automatically enter the same number in element $\mathbf{a_{31}}$, etc. This can save considerable time and cut down the chance for error when entering symmetric matrices. The "**Symmetric Entry**" check box must not be selected when non-symmetric matrices are entered. This control is disabled if the current matrix is not a square matrix, since only a square matrix can be symmetric.

The determinate of the current matrix is shown under the symmetric entry control. If the current matrix is a complex matrix the determinate will have two parts, a real part and an imaginary part. Real matrices will only display a real part. If the determinate has not been computed, this determinate will display "**Not Computed**" for the determinate values. The determinate display is labelled "**Determinate**" and the individual parts are labelled "**Real =**" and "**Imag =**".

Real Matrix Data Entry

To enter data into the elements the user will first select either the **A** matrix or **B** matrix. This is done by keying on either the "**Matrix A**" or "**Matrix B**" radio buttons in the lower, left of the screen. The user then clicks on the RESET button to bring up the **Reset Matrix** dialog, shown in Figure 6.6.

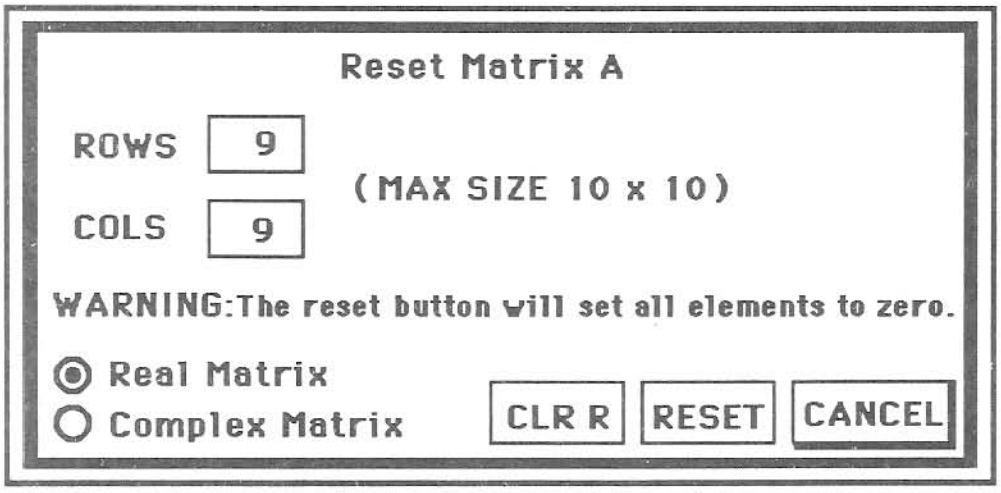

Figure 6.6 Reset Matrix dialog

The **Reset Matrix** dialog contains two edit lines to input the row and column size of the new matrix. Two radio buttons labelled "**Real Matrix**" and "**Complex Matrix**" let the user select the type of matrix to be entered. After setting the proper row and column sizes and selecting the type of matrix, using the radio buttons, clicking on the RESET button will zero all matrix elements, set the row and columns, set the matrix type and return the user to the **Matrix Operations** dialog. The operation may be cancelled by clicking on the CANCEL button. The CLR R button allows the user to clear the elements of the result matrix **R** to zero, if desired. The label at the top of the screen tells the user which matrix is being reset. You are now ready to enter element values.

Three types of entry are possible when entering values into the matrix edit lines:

1. Entering a number, followed by the ENTER button or return key, will enter the number as the matrix element and advance the matrix element indicator.
2. Entering a valid function, using direct function entry, will set the matrix element to the value of the function and advance the matrix element indicator.
3. Entering a memory assignment with direct function entry (i.e. a = 2.35 or c = sin(b)/cos(d)) will not advance the matrix element indicator or enter the number as the matrix element value. This allows the user to build a single matrix element that is a combination of several functions. The matrix element can be set to the result by referencing the memory location, using direct function entry.

To demonstrate how to enter a new matrix, we will enter the following matrix into the **A** matrix. Select the A matrix with the "**Matrix A**" radio button.

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 7 & 1 \\ 4.5 & 8 & 3 \end{bmatrix}$$

	RESET	This brings up the **Reset Matrix** dialog box.
3	TAB	Enter the number of **Rows**. Then use the "**TAB**" key to move to the **Cols** edit line, or click the mouse in the **Cols** edit line.
3		Enter the number of columns. Click on the "**Real Matrix**" Radio button to set type of matrix to real.
	RESET	Return to **Matrix Operations** dialog.

The matrix is not symmetric, so we can't use the "**Symmetric Entry**" mode.

1	ENTER	Enter element a_{11}.
3	ENTER	Enter element a_{12}.
5	ENTER	Enter element a_{13}.
2	ENTER	Enter element a_{21}.
7	ENTER	Enter element a_{22}.
1	ENTER	Enter element a_{23}.
4.5	ENTER	Enter element a_{31}.
8	ENTER	Enter element a_{32}.
3	ENTER	Enter element a_{33}.

This completes the entry of the matrix elements. Notice that the following radio buttons, located on the right side of the **Matrix Operations** dialog are now enabled: "**Scalar Mult**", "**Transpose Matrix**", "**Invert Matrix**", "**Determinate**", "**Swap A - B**", and "**Memory**". These are all valid operations for a single, square matrix. If the matrix was not a square matrix, the "**Invert Matrix**" and "**Determinate**" radio buttons would be disabled. **GSNumerics** will only activate those buttons that are valid operations for the matrix or matrices entered.

Now enter the following matrix into matrix **B**. Select matrix **B** by clicking on the radio button "**Matrix B**". Notice that this is a symmetric matrix ($a_{12} = a_{21}$, $a_{13} = a_{31}$, and $a_{23} = a_{32}$).

$$\begin{bmatrix} 2 & 1 & 6 \\ 1 & 8 & 4 \\ 6 & 4 & 5 \end{bmatrix}$$

	RESET	This brings up the **Reset Matrix** dialog box.
3	TAB	Enter the number of **Rows**. Then use the "**TAB**" key to move to the **Cols** edit line, or click the mouse in the **Cols** edit line.
3		Enter the number of columns. Click on the "**Real Matrix**" Radio button to set type of matrix to real.
	RESET	Return to **Matrix Operations** dialog.

Since you had selected matrix **B**, the **Reset Matrix** dialog title was "**RESET MATRIX B**", and the operations only changed matrix **B**.

We know the matrix is symmetric, so select symmetric entry by clicking on the "**Symmetric Entry**" radio button.

2	ENTER	Enter element a_{11}.
1	ENTER	Enter element a_{12}.
6	ENTER	Enter element a_{13}.
8	ENTER	Enter element a_{22}. The program put a_{21} in for you.
4	ENTER	Enter element a_{23}.
5	ENTER	Enter element a_{33}. The program put $_{a31}$ and $_{a32}$ in for you, since you selected symmetric entry. This saved you three entry steps.

You will see that three additional radio buttons have been activated on the right had side of the **Matrix Operations** dialog. The additional activated buttons are: "**Mat A + Mat B**", "**Mat A - Mat B**" and "**Mat A * Mat B**". These are valid operations for two matrices of the same size. Notice how GSNumerics will only allow valid operations to be performed on matrices. If a particular operation is invalid, the corresponding radio button will be disabled, and you will not be able to select it.

Matrix Memory Operations

We may save either the **A**, **B** or **R** matrices in two memory matrices **MM1** and **MM2**. They can be recalled for use at a later time. We will now save the two matrices **A** and **B**, using the **Matrix Memory Operations** dialog and recall them later in this chapter. The **Matrix Memory Operations** dialog is shown in Figure 6.7.

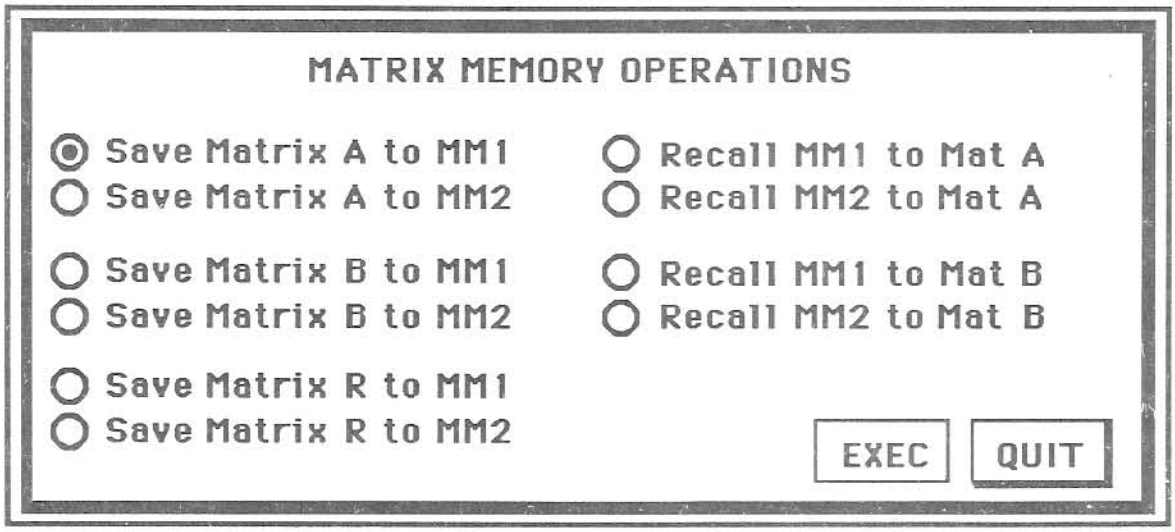

Figure 6.7 Matrix Memory Operations dialog

You can bring up the **Matrix Memory Operations** dialog by clicking on the radio button labelled "**Memory**", located at the right of the screen, near the bottom. When the screen comes up you will see that only the "**Save Matrix A to MM1**", "**Save Matrix A to MM2**", "**Save Matrix B to MM1**", and "**Save Matrix B to MM2**" radio buttons are enabled. The buttons will only be enabled if the matrix or matrix memory location contains data.

Save matrix **A** into **MM1** by clicking on the "**Save Matrix A to MM1**" radio button and then clicking on the **EXEC** button. Save matrix **B** into **MM2** by clicking on the "**Save Matrix B to MM2**" button and then clicking on the **EXEC** button. This will enable the "**Recall MM1 to A**", "**Recall MM2 to A**", "**Recall MM1 to B**" and "**Recall MM2 to B**" radio buttons, since these memory matrices now contain data. Return to the **Matrix Operations** dialog by clicking on **QUIT** .

Complex Matrix Data Entry

Entering complex matrices is as easy as entering real matrices. The only difference is that you must enter two numbers for each element, the real part and the imaginary part. Complex matrix elements must be entered in rectangular form. If you enter elements in polar form, matrix operations will not compute correctly. If the matrix elements you are going to enter are in polar form, use the **PTOR** button to convert them to rectangular form, after entering the elements. The conversion will assume the **trig mode** set on the **Scientific Calculator Screen**. If the polar format is expressed in degrees, make sure you have selected the degrees mode on the **Scientific Calculator Screen**.

Now enter the following complex matrix into matrix **A**. Select matrix **A** by clicking on the radio button "**Matrix A**". Notice that this is a symmetric matrix ($a_{12} = a_{21}$, $a_{13} = a_{31}$, and $a_{23} = a_{32}$).

$$\begin{bmatrix} 1\ \angle 30^0 & -2 - i3 & 2 + i1 \\ -2 - i3 & 5 - i5 & 4\ \angle 37.5^0 \\ 2 + i1 & 4\ \angle 37.5^0 & 3 - i3 \end{bmatrix}$$

	RESET	This brings up the **Reset Matrix** dialog box.
3	**TAB**	Enter the number of **Rows**. Then use the "**TAB**" key to move to the **Cols** edit line, or click the mouse in the **Cols** edit line.
3		Enter the number of columns. Click on the "**Complex Matrix**" Radio button to set type of matrix to complex.
	RESET	Return to **Matrix Operations** dialog.

We know the matrix is symmetric, so select symmetric entry by clicking on the "**Symmetric Entry**" check button. Also, make sure the **trig mode** is in degrees mode, since the matrix contains complex elements expressed in polar degrees form, that must be converted to rectangular form.

1	ENTER	Enter element a_{11} polar magnitude, then enter the polar angle.
30	PTOR	Converts the polar form to rectangular form and enters element.
-2	ENTER	Enter element a_{12} real.
-3	ENTER	Enter element a_{12} imaginary.
2	ENTER	Enter element a_{13} real.
1	ENTER	Enter element a_{13} imaginary.
5	ENTER	Enter element a_{22} real.
-5	ENTER	Enter element a_{22} imaginary.
4	ENTER	Enter element a_{23} polar magnitude, then enter the polar angle.
37.5	PTOR	Converts the polar form to rectangular form and enters element.
3	ENTER	Enter element a_{33} real.
-3	ENTER	Enter element a_{33} imaginary.

The symmetric entry feature saved you six data entry steps.

Matrix View Mode

The matrix **view mode** is used to view matrices only. You can't edit matrices or call matrix functions when in the **view mode**. To view a function in **view mode**, you simply click on the VIEW button. You may return to the entry mode by again clicking on the VIEW button. If the radio button labelled "**Matrix R**" is enabled, indicating the result matrix **R** contains matrix data, clicking on the "**Matrix R**" button will put you in the **view mode**. If you click on the VIEW button while viewing the **R** matrix, you will be returned to the matrix you were in when you called the **view mode**. When in this mode you cursor through the matrix using the scroll bars. The **view mode** allows you to see complex matrices in polar format. If the **view mode** matrix is complex the two radio buttons "**View Polar**" and "**View Rect**" are enabled.

To see the **view mode** in action, set the current matrix to **A** and click on VIEW. Now click on "**View Polar**" Using the scroll bars, you will be able to review the matrix elements, of the complex matrix we input in the previous section, in polar format. Return to the **entry mode** by clicking on the VIEW button.

Matrix Addition

To demonstrate matrix math operations, we will use the **A** and **B** matrices you input and saved in the previous section and saved in **MM1** and **MM2**. Please recall these functions now, by clicking the "**Memory**" radio button. When the **Matrix Memory Operations** dialog comes up, click on "**Recall MM1 to A**" followed by a click on the EXEC and then click on "**Recall MM2 to B**" followed by a click on the EXEC button. Return to the **Matrix Operations** dialog by clicking on the QUIT button. This puts the original **A** matrix data back into the **A** matrix and the original **B** matrix data back into the **B** matrix.

To add the **A** and **B** matrix click on "**Mat A + Mat B**" then click on the EXEC button. After a short pause for calculation the "**Matrix R**" radio button will be highlighted, signifying that a solution is stored in the **R** matrix. Click on the "**Matrix R**" radio button and the **view mode** will be automatically invoked. You may now confirm the following, by inspecting the **R** matrix elements, using the scroll bars to cursor through the elements.

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 7 & 1 \\ 4.5 & 8 & 3 \end{bmatrix} + \begin{bmatrix} 2 & 1 & 6 \\ 1 & 8 & 4 \\ 6 & 4 & 5 \end{bmatrix} = \mathbf{R} = \begin{bmatrix} 3 & 4 & 11 \\ 3 & 15 & 5 \\ 10.5 & 12 & 8 \end{bmatrix}$$

Return to the entry mode by clicking on the VIEW button.

Matrix Subtraction

To subtract matrix **B** from matrix **A** you simple click on the "**Mat A - Mat B**" button and then click on the EXEC button. Do this and confirm the following by looking at the of elements matrix **R** in the **view mode**.

$$\mathbf{A} - \mathbf{B} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 7 & 1 \\ 4.5 & 8 & 3 \end{bmatrix} - \begin{bmatrix} 2 & 1 & 6 \\ 1 & 8 & 4 \\ 6 & 4 & 5 \end{bmatrix} = \mathbf{R} = \begin{bmatrix} -1 & 2 & -1 \\ 1 & -1 & -3 \\ -1.5 & 4 & -2 \end{bmatrix}$$

Notice that matrix math operations do not change either the **A** matrix or the **B** matrix. If you want to save the result matrix **R**, you can use the **Matrix Memory Operations** dialog to save it in **MM1** or **MM2**. You may also place the **R** matrix in either matrix **A** or matrix **B** by using the "**Copy R to A**" or "**Copy R to B**" options. The "**Swap A and B**" option will exchange the **A** and **B** matrices. After choosing one of these options, you must click on EXEC to execute the option.

Matrix Multiplication

To multiply matrix **B** by matrix **A** click on the "**Mat A * Mat B**" button and then click on the **EXEC** button. Do this and confirm the following by looking at the elements of matrix **R** in the **view mode**.

$$\mathbf{A * B} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 7 & 1 \\ 4.5 & 8 & 3 \end{bmatrix} * \begin{bmatrix} 2 & 1 & 6 \\ 1 & 8 & 4 \\ 6 & 4 & 5 \end{bmatrix} = \mathbf{R} = \begin{bmatrix} 35 & 45 & 43 \\ 17 & 62 & 45 \\ 35 & 80.5 & 74 \end{bmatrix}$$

Matrix Transpose

To compute the transpose of a matrix select the desired matrix by clicking on the "**Matrix A**" or "**Matrix B**" button, click on the "**Transpose Matrix**" radio button and click on **EXEC**. Compute the transpose of matrix **A**. First click on the "**Matrix A**" radio button to make **A** the active matrix. Compute the transpose of **A** and confirm the following by viewing the **R** matrix.

$$\mathbf{A}^{\mathbf{T}} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 7 & 1 \\ 4.5 & 8 & 3 \end{bmatrix}^{\mathbf{T}} = \mathbf{R} = \begin{bmatrix} 1 & 2 & 4.5 \\ 3 & 7 & 8 \\ 5 & 1 & 3 \end{bmatrix}$$

Matrix Inversion and Determinants

To invert a matrix click the "**Invert Matrix**" radio button and click the **EXEC** button. The current matrix will be inverted and the result placed in the **R** matrix. Invert the A matrix and confirm the following by viewing the **R** matrix.

$$\mathbf{A}^{-1} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 7 & 1 \\ 4.5 & 8 & 3 \end{bmatrix}^{-1} = \mathbf{R} = \begin{bmatrix} -0.1884 & -0.4493 & 0.4638 \\ 0.0217 & 0.2826 & -0.1304 \\ 0.2246 & -0.0797 & -0.0145 \end{bmatrix}$$

Notice the matrix determinant is now shown as **Real = -69.0000**. You could also compute the determinant directly by clicking on the "**Determinant**" radio button and the clicking on **EXEC**. This will compute the determinant more quickly, especially on large matrices. You get the determinant automatically when you compute the matrix inverse.

Now confirm the following $\mathbf{A * A^{-1} = I}$. A matrix multiplied by its inverse is equal to the identity matrix. First copy the **R** matrix to **B** using the "**Copy R to B**" button followed by the **EXEC** button. This puts the inverse calculated in step one above into the **B** matrix.

Do the multiplication by clicking on the "**Mat A * Mat B**" radio button followed by [EXEC]. Review the resulting **R** matrix in the **view mode.**

$$\mathbf{A} * \mathbf{A}^{-1} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 7 & 1 \\ 4.5 & 8 & 3 \end{bmatrix} * \begin{bmatrix} -0.1884 & -0.4493 & 0.4638 \\ 0.0217 & 0.2826 & -0.1304 \\ 0.2246 & -0.0797 & -0.0145 \end{bmatrix} = \mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix Scalar Multiplication

You may multiply a matrix by a real or complex scalar, by using the **Scalar Multiplication** dialog. You may bring up the **Scalar Multiplication** dialog by first clicking on the "**Scalar Mult**" radio button, then clicking on the [EXEC] button. The **Scalar Multiplication** dialog is shown in Figure 6.7.

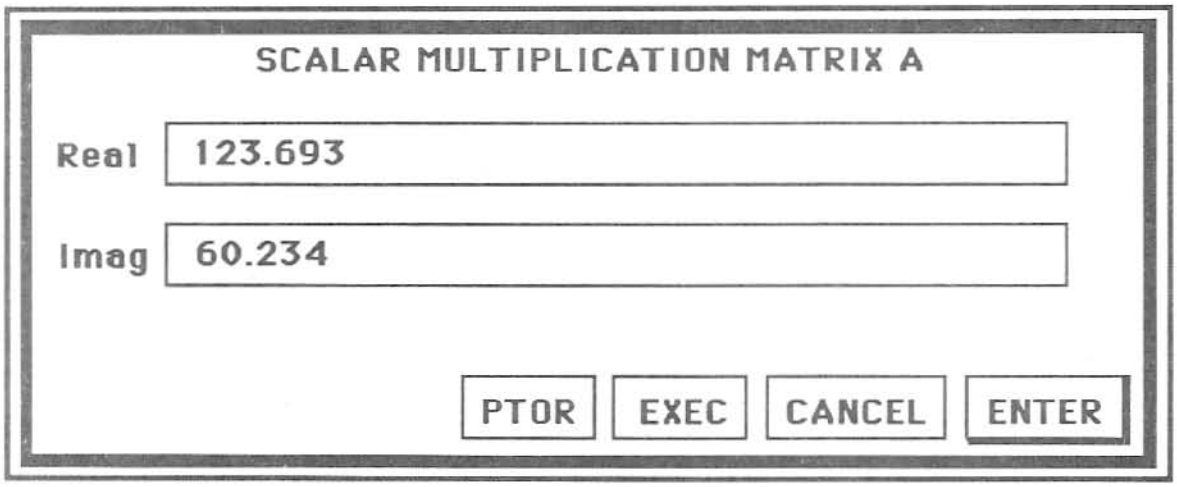

Figure 6.7 Matrix Scalar Multiplication dialog

To multiply a matrix by a scalar, enter the rectangular components of the scalar into the edit lines and then click the button. If the Imaginary part of the scalar is zero, the program will do a real scalar multiplication. If the Imaginary part of the scalar is non-zero, the program will do a complex scalar multiplication. The result is placed into the **R** matrix. Multiplying a real matrix by a complex scalar will result in a complex matrix result. Scalars must be input in rectangular format. Use the [PTOR] button to convert scalars input in polar format. Any matrix operation, combining a real and a complex matrix, will result in the **R** matrix being complex. All matrices are stored as complex matrices. When you designate a matrix as real, the program simply puts 0.0 in each of the complex elements, and ignores them in real operations.

To demonstrate scalar multiplication, do the following scalar multiplication:

$$2 \angle 35^{0} * A = \begin{bmatrix} 1.6383 + i1.1472 & 4.9149 + i3.4415 & 8.1915 + i5.7358 \\ 3.2766 + i2.2943 & 11.4681 + i8.0301 & 1.6383 + i1.1472 \\ 7.3724 + i5.1622 & 13.1064 + i9.1772 & 4.9149 + i3.4415 \end{bmatrix}$$

To do this multiplication select the **Scalar Multiplication** dialog from the **A** matrix entry screen by clicking "**Const Mult**". Then do the following:

2	ENTER	Enter the scalar magnitude.
35	PTOR	Enter the scalar angle and convert to rectangular. This enters the rectangular \form of the scalar.
	EXEC	This initiates the scalar multiplication and returns you to the entry screen.

The resultant matrix **R** can be examined using the **view mode** to confirm the above multiplication.

Matrix Stack Operations

You may do stack operations with the determinants of matrices. If the active matrix determinant has been computed, the "**Stack**" radio button will be activated. Clicking on the "**Stack** " button will bring up the **Matrix Stack Operation** dialog shown in Figure 6.8.

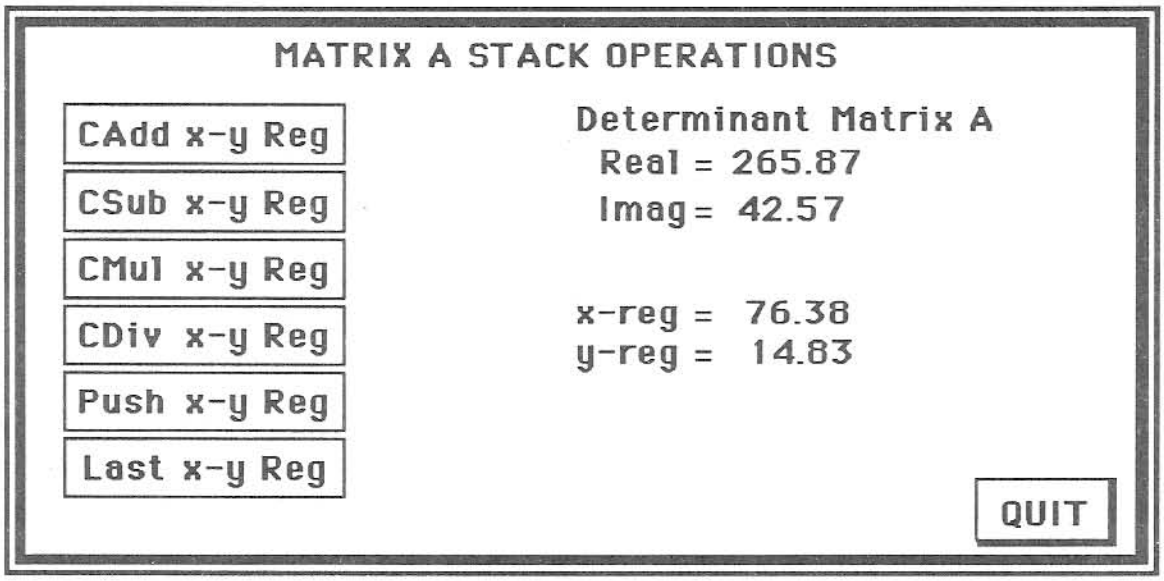

Figure 6.8 Matrix Stack Operations dialog

If the current matrix is a complex matrix, the **x-reg** and **y-reg** values will be shown, along with the Real and the Imaginary parts of the Determinant. The stack operations will be complex operations and will perform complex pushes to the stack. If the current matrix is a real matrix, the x-reg and Real part of the determinant will be shown and all stack operations will be real.

Matrix File Operations

You may save and recall individual matrices or all matrices (**A, B, R, MM1, MM2**) to disk files. The process of saving all matrix files at one time is called a **Matrix Session Save**. All matrices are also saved and recalled when you use the **Session Save/Recall** menu items in the **FILE** menu. When a matrix is saved, the determinant is saved, if it has been computed. To save matrix files, use the **Save Matrix Files** dialog shown in Figure 6.9. You may call this dialog by selecting the **Save Matrix File** menu item in the **MATRIX** menu. This menu item will be disabled if none of the matrix storage locations contains a matrix.

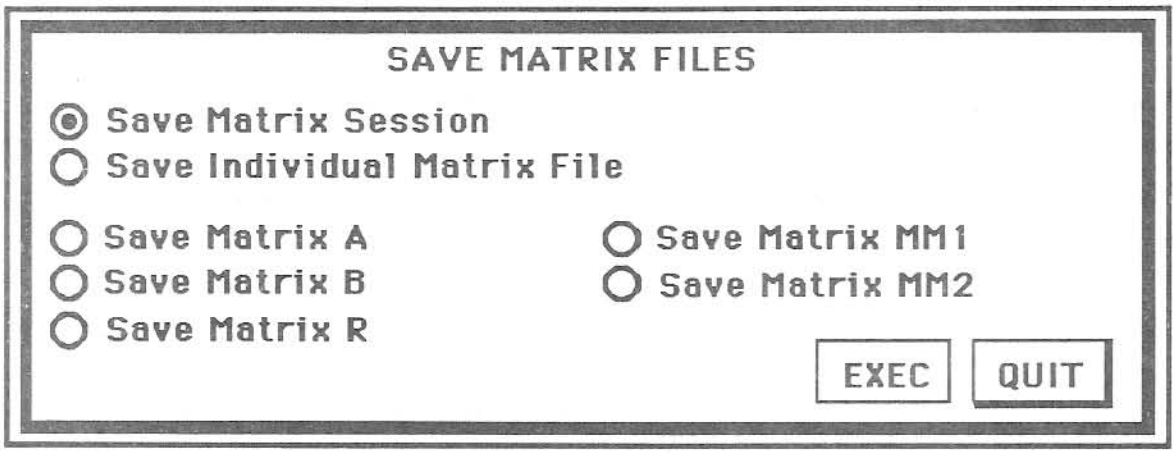

Figure 6.9 Save Matrix Files dialog

To save a Matrix Session you click on the "**Save Matrix Session**" radio button and click on the **EXEC** button. The **Standard File Save** dialog box will then appear, allowing you to name the file and complete the save.

To save an individual matrix file click on the "**Save Individual Matrix File**" button. The individual matrix buttons will then be enabled, if they contain a matrix. You then select the matrix you wish to save by clicking on the appropriate radio button (i.e. "**Save Matrix MM1**" would save the **MM1** matrix) and then clicking on the **EXEC** button. The **Standard File Save** dialog box will then appear, allowing you to name the file and complete the save.

To recall matrix select the "**Load Matrix Files**" menu item from the **MATRIX** menu. The **Load Matrix Files** dialog will then appear. This dialog is shown in Figure 6.10. As you can see, this dialog operates exactly like the **Save Matrix Files** dialog dialog, except you are recalling stored matrices to the program, when using the **Load Matrix Files** dialog.

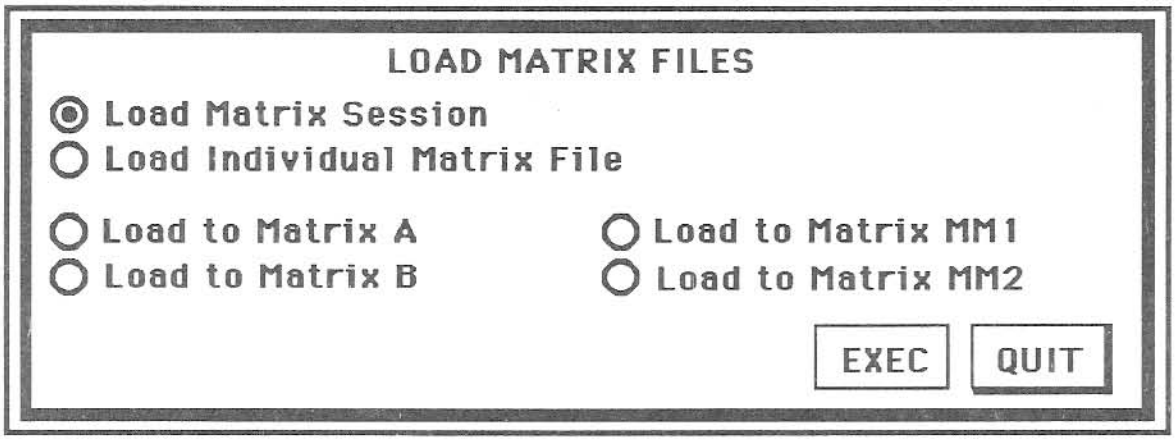

Figure 6.10 Load Matrix Files dialog

To load a Matrix Session click on the "**Load Matrix Session**" radio button and then click the EXEC button. The **Standard File Recall** dialog will then appear, allowing you to select the particular Matrix Session file to be recalled and to complete the load operation.

To load an individual matrix file click on the "**Load Individual Matrix File**" and click on the radio button indicating where you wish the recalled file to be placed (i.e. clicking on "**Load to Matrix B**" will cause the file to be loaded into matrix **B**"). A final click on the EXEC button will cause the **Standard File Recall** dialog to appear, allowing you to select the particular matrix file to be recalled.

LINEAR SYSTEMS

As we progress in math we find that many practical problems reduce themselves to finding the simultaneous solution to a system of two or more linear equations. Generally these systems have more than one unknown or independent variable. The general form of a system of linear equations is shown in Figure 7.1:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_n \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_n \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\
a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n
\end{aligned}
$$

Figure 7.1 System of Linear Equations

where the **a** terms are the coefficients, the **b** terms are the constants and the **x** terms are the unknowns. This system would be called a system of **n** equations in **n** unknowns. The problem is to find the values of $x_1, x_2, \ldots, x_n$ that solve each of the equations simultaneously. Systems of linear equations are best solved by putting them in the matrix form and using matrix operations to find the solutions. The equations of Figure 7.1 are shown in matrix form in Figure 7.2:

$$
\begin{bmatrix}
a_{11} & a_{12} & \ldots & a_{1n} \\
a_{21} & a_{22} & \ldots & a_{2n} \\
\ldots & \ldots & \ldots & \ldots \\
a_{m1} & a_{m2} & \ldots & a_{mn}
\end{bmatrix}
\begin{bmatrix}
x_1 \\ x_2 \\ \ldots \\ x_3
\end{bmatrix}
=
\begin{bmatrix}
b_1 \\ b_2 \\ \ldots \\ b_3
\end{bmatrix}
$$

Figure 7.2 Matrix Form ofLinear System

where the matrix containing the a_{ij} terms is called the coefficient matrix, the matrix containing the x_i terms is called the solution matrix and the matrix containing the b_i terms is called the constant matrix.

Systems of linear equations may be either real or complex. Real systems will have real a_{ij}, x_i and b_i terms. Complex systems of equations will have complex a_{ij}, x_i and b_i terms. **GSNumerics** can solve systems of linear equations, either real or complex, of up to ten equations in ten unknowns. The program will also compute the errors in the solutions by substituting the **xi** terms back into the equations and finding the difference between the constant term and the computed solution, for each equation in the system.

Consider the following system of four equations in four unknowns:

$$\begin{aligned} 3x_1 - 6x_2 + x_3 + 2x_n &= 13 \\ 5x_1 + 3x_2 - x_3 + 4x_n &= 2 \\ 2x_1 + 4x_2 - 3x_3 + 2x_n &= 0 \\ 3x_1 + 3x_2 - 7x_3 + 6x_n &= -2 \end{aligned}$$

In matrix form this system is expressed as:

$$\begin{bmatrix} 3 & -6 & 1 & 2 \\ 5 & 3 & -1 & 4 \\ 2 & 4 & -3 & 2 \\ 3 & 3 & -7 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 13 \\ 2 \\ 0 \\ -2 \end{bmatrix}$$

The following complex system:

$$\begin{aligned} (1 + i3)x_1 - (2 + i2)x_2 + (2 - i2)x_3 &= 4 - i2 \\ (4 + i1)x_1 - (1 - i3)x_2 + (2 + i6)x_3 &= 1 - i7 \\ (1 + i5)x_1 - (9 + i4)x_2 + (3 - i1)x_3 &= 5 + i2 \end{aligned}$$

is written in matrix form form as follows:

$$\begin{bmatrix} 1 + i3 & -2 - i2 & 2 - i2 \\ 4 + i1 & -1 + i3 & 2 + i6 \\ 1 + i5 & -9 - i4 & 3 - i1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 - i2 \\ 1 - i7 \\ 5 + i2 \end{bmatrix}$$

The x_i terms will be complex in the complex matrix. Complex matrices are identical to real matrices except that it takes two values to represent each number in a complex matrix. Solution of a set of complex equations involves a larger number of operations, since all operations must be done using complex arithmetic.

All complex number data must be entered in rectangular format. The solutions will be incorrect if you fail to convert numbers entered in polar form to rectangular form, prior to doing calculations.

Entering Real Linear Systems Data

To enter a system of linear equations you must first tell the program the size of the system to be solved. You do this by selecting the **Reset** item from the **LINEAR SYSTEMS** menu. This will bring up the **Set System Size** dialog. This dialog is very simple and contains one edit line labeled **System Size** and an OK button. Enter the size of the system and click on the OK button and you will be returned to the **Scientific Calculator Screen.** To enter a system with real number elements, select the **Enter Real System** item on the **LINEAR SYSTEMS** menu. This will bring up the data entry screen used to enter the matrix data. The **Real Matrix Data Entry** is shown in Figure 7.3.

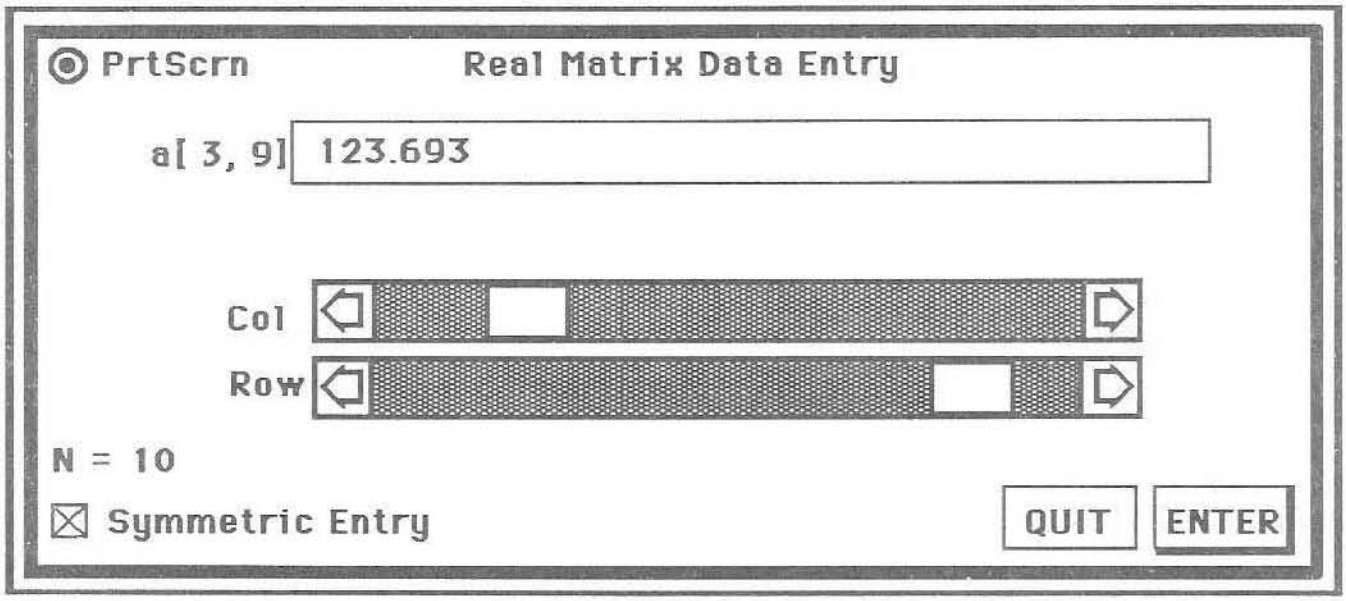

Figure 7.3 Real System Data Entry dialog

This dialog screen contains one edit line, labelled **a[3,9]**. This edit line is used to input the coefficients and constants. The scroll bars labelled **Col** and **Row** are used to cursor through the matrix when reviewing or editing individual elements. The size of the system is shown with the the label **N = 10**. A check mark control, labelled **Symmetric Entry,** is used to assist in the entry process when the coefficient matrix is symmetrical. Using **Symmetric Entry** will enable you to enter symmetric matrices more quickly and with less chance of making an error in the entry process. After a number is entered, you may press the return key or click on the ENTER button to enter the number. The QUIT button is used to return to the calculator screen.

After the last coefficient matrix element for each row has been entered the edit line label will change to **b[i]**. This will indicate you are to input the constant coefficient for that row. It will revert back to indicate a coefficient entry for the next row (i.e. **a[i+1,1]**), after the constant has been entered.

Three types of entry are possible when entering values into the matrix edit lines:

1. Entering a number, followed by the ENTER button or return key, will enter the number as the matrix element and advance the matrix element indicator.
2. Entering a valid function, using direct function entry, will set the matrix element to the value of the function and advance the matrix element indicator.
3. Entering a memory assignment with direct function entry (i.e. a = 2.35 or c = sin(b)/cos(d)) will not advance the matrix element indicator or enter the number as the matrix element value. This allows the user to build a single matrix element that is a combination of several functions. The matrix element can be set to the result by referencing the memory location, using direct function entry.

We will enter the following real system of three equations in three unknowns.

$$\begin{aligned} 3x_1 - 6x_2 + x_3 &= 13 \\ 5x_1 + 3x_2 - x_3 &= 2 \\ 2x_1 + 4x_2 - 3x_3 &= 0 \end{aligned}$$

Prior to starting the input process select the **Reset** item from the **LINEAR SYSTEM** menu and set the system size to **3**. Then select the **Real Matrix Data Entry** dialog from the **LINEAR SYSTEMS MENU** by selecting the **Enter Real System** menu item.

3	ENTER	Enter the coefficient **a[1,1]**.
-6	ENTER	Enter the coefficient **a[1,2]**.
1	ENTER	Enter the coefficient **a[1,3]**.
13	ENTER	Enter the constant **b[1]**. Notice the line edit label changed, indicating you are to enter the constant term.
5	ENTER	Enter the coefficient **a[2,1]**. The next row coefficient entry.
3	ENTER	Enter the coefficient **a[2,2]**.
-1	ENTER	Enter the coefficient **a[2,3]**.
2	ENTER	Enter the constant **b[2]**. The row two constant entry.
2	ENTER	Enter the coefficient **a[3,1]**.
4	ENTER	Enter the coefficient **a[3,2]**.
-3	ENTER	Enter the coefficient **a[3,3]**.
0	ENTER	Enter the constant **b[3]**. The third row constant entry.

Solving Systems of Real Equations

We will now solve the **3 x 3** real system input in the previous section. You first leave the **Real Matrix Data Entry** dialog by clicking on QUIT, and select the **Solve** item from the **LINEAR SYSTEM** menu. The **Solve System of Real Equations** dialog, shown in Figure 7.4, will appear.

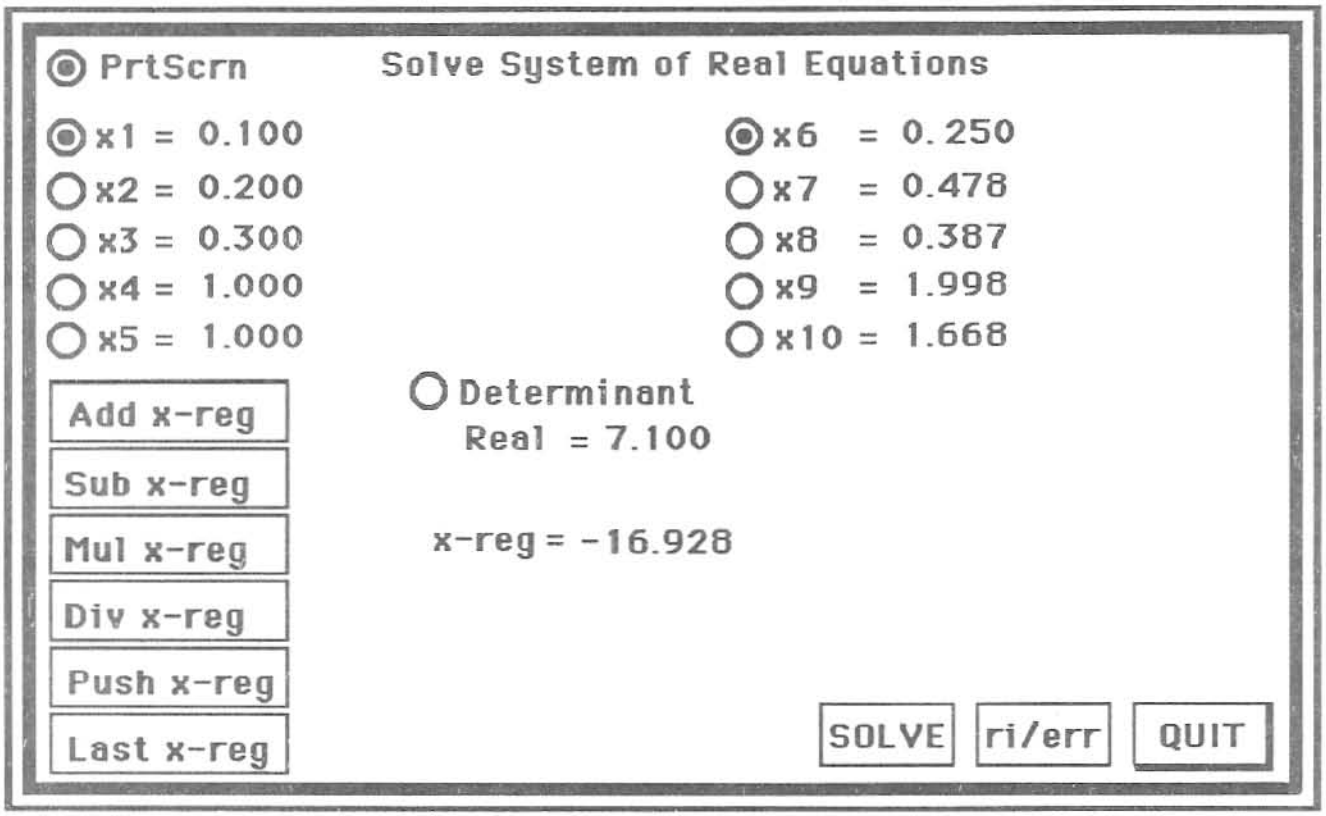

Figure 7.4 Real System Solution Dialog

The dialog box shown in Figure 7.4 represents the situation where the system size is **10 x 10**. For the **3 x 3** system we just entered, the x_4 through x_{10} labels and will not be drawn on the dialog, since there will only be three solutions to a **3 x 3** matrix. The **10 x 10** example is drawn for illustration purposes only. The SOLVE button is used to initiate the solution of the system. After the solutions have been computed you may see the error in the solutions, by clicking on the ri/Err button. The labels x_1 - x_n will be changed to e_1 - e_n, when you are viewing the errors. You may change back to a display of the solutions, by again clicking on the ri/Err button. To quit the dialog click on the QUIT button.
The Add x-reg, Sub x-reg, Mul x-reg, Div x-reg, Psh x-reg and Lst Stack buttons are used to perform stack operations with the solutions and determinant of the coefficient matrix. These buttons are used in conjunction with the radio buttons to the left of the x_1 - x_n labels.

When you bring up the **Solve System of Real Equations** dialog for the **3 x 3** matrix you previously entered, the labels x_1, x_2, and x_3 are drawn on the screen. You can now solve this system by clicking on the [SOLVE] button. After a short pause for calculation, the solutions will be shown as follows:

$$x_1 = 1.177$$
$$x_2 = -1.861$$
$$x_3 = -1.696$$

The value of the determinant of the coefficient matrix is shown as -79.000. Key on the [ri/Err] button and the errors in the solutions will be displayed. The errors represent the solutions of the equations, solved with the computed values substituted for each x_i, subtracted from the actual input constants. You may display the solutions by clicking on the [ri/Err] button.

Entering Complex Linear System Data

The **Complex Matrix Data Entry** dialog, shown in Figure 7.5 is used to enter complex systems.

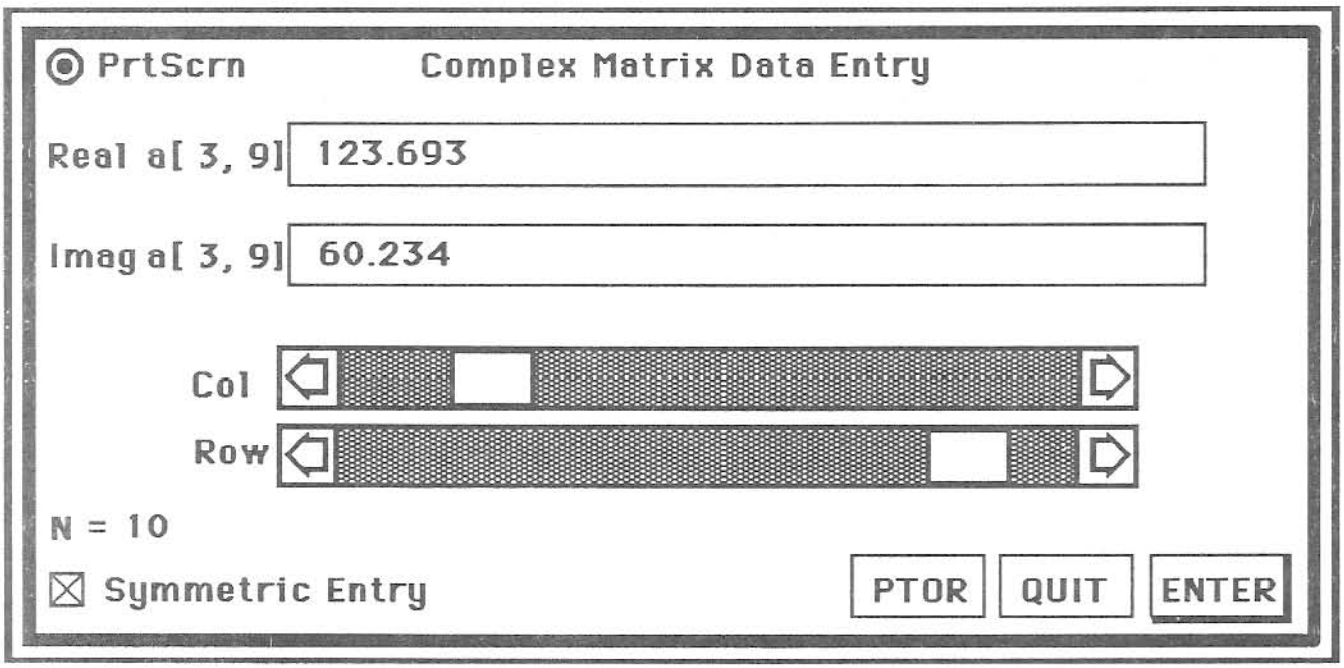

Figure 7.5 Complex System Data Entry dialog

To enter a system of complex equations, you first reset the system size as previously explained. After the matrix size has been set, select the **Enter Complex System** item from the **LINEAR SYSTEM** menu. This will bring up the dialog shown in Figure 7.5.

The only difference between the **Complex Matrix Data Entry** dialog and the **Real Matrix Data Entry** dialog is the second edit line **Imag** and the **PTOR** button. These are not present on the **Real Matrix Data Entry** dialog. After resetting the size of the **Linear System** matrix with the **Set System Size** dialog, you may enter either a real or a complex system. After starting entry of a real system, you must reset the matrix prior to entering complex data. Likewise, if you have started entry of a complex system, you must reset the matrix with the **Set System Size** dialog, before entering a real system. The **Set System Size** dialog will reset all matrix elements to zero. If the Linear System matrix contains elements, a warning dialog will appear when you call the **Set System Size** dialog, advising you that you are about to lose the current Linear Systems data.

After you have entered the last column of a row the **Real a** and **Imag a** labels will change to **Real b** and **Imag b**, indicating you are ready to enter the constant value for that particular row. After you have entered the constant value, the matrix indicator will be automatically incremented to column one of the next row, and the labels will change back to **Real a** and **Imag a**, indicating that you are ready to start entering the coefficient elements for the new row.

Complex numbers must be entered in rectangular format. If you are entering data in polar format, you may convert to rectangular form with the **PTOR** button. If you are entering the complex angle in degrees, you must be sure that the **Scientific Calculator trig mode** is set to the degrees mode. Likewise, if grads or radians are being converted, the trig mode must be set accordingly. If you enter complex numbers in polar form and fail to convert them to rectangular form with the **PTOR** button, the solutions to the system will not be correct. In this instance, the program will interpret the magnitude as the real part of the coefficient and the angle as the imaginary part of the coefficient. Make sure you do not initiate a solution without converting polar form numbers to rectangular form.

We will now enter and solve the following system of complex linear equations:

$$(1 + i3)x_1 - (2 + i2)x_2 + (2 - i2)x_3 = 4 - i2$$
$$(4 + i1)x_1 - (1 - i3)x_2 + (2 + i6)x_3 = 1 - i7$$
$$(1 + i5)x_1 - (9 + i4)x_2 + (3 - i1)x_3 = 5 + i2$$

Prior to starting make sure the calculator **trig mode** is set to the degrees mode and reset the linear systems matrix to a **3 x 3** matrix using the **Reset** menu item. The matrix is not symmetrical, so make sure the **Symmetric Entry** check box is off.

1	ENTER	Enter the real part of coefficient **a[1,1]**.
3	ENTER	Enter the imaginary part of coefficient **a[1,1]**.
-2	ENTER	Enter the real part of coefficient **a[1,2]**.
-2	ENTER	Enter the imaginary part of coefficient **a[1,2]**.
2	ENTER	Enter the real part of coefficient **a[1,3]**.
-2	ENTER	Enter the imaginary part of coefficient **a[1,3]**.
4	ENTER	Enter the real part of constant **b[1]**.
-2	ENTER	Enter the imaginary part of coefficient **b[1]**.
4	ENTER	Enter the real part of coefficient **a[2,1]**.
1	ENTER	Enter the imaginary part of coefficient **a[2,1]**.
-1	ENTER	Enter the real part of coefficient **a[2,2]**.
3	ENTER	Enter the imaginary part of coefficient **a[2,2]**.
2	ENTER	Enter the real part of coefficient **a[2,3]**.
6	ENTER	Enter the imaginary part of coefficient **a[2,3]**.
1	ENTER	Enter the real part of constant **b[2]**.
-7	ENTER	Enter the imaginary part of coefficient **b[2]**.
1	ENTER	Enter the real part of coefficient **a[3,1]**.
5	ENTER	Enter the imaginary part of coefficient **a[3,1]**.
-9	ENTER	Enter the real part of coefficient **a[3,2]**.
-4	ENTER	Enter the imaginary part of coefficient **a[3,2]**.
3	ENTER	Enter the real part of coefficient **a[3,3]**.
-1	ENTER	Enter the imaginary part of coefficient **a[3,3]**.
5	ENTER	Enter the real part of constant **b[3]**.
2	ENTER	Enter the imaginary part of coefficient **b[3]**.

For this **3 x 3** complex matrix, we must make 24 number entries. Two for each of the 9 coefficients and two for each of the constants. If the matrix was symmetrical, we could have reduced this to 18 number entries. This will be demonstrated in the next example. When you input a matrix, always inspect the elements to determine if it is symmetrical. Symmetric entry can save you a lot of time, and will greatly lessen your chances of making an error when you input the element values.

This completes the entry of our **3 x 3** system of complex equations. Leave the **Complex Matrix Data Entry** dialog by clicking on the QUIT button. Don't save it to a disk file. We will now solve the **3 x 3** complex system of equations.

Solving Systems of Complex Equations

Select the **Solve** item from the **LINEAR SYSTEM** menu. The **Solve System of Complex Equations** dialog, shown in Figure 7.6, will appear:

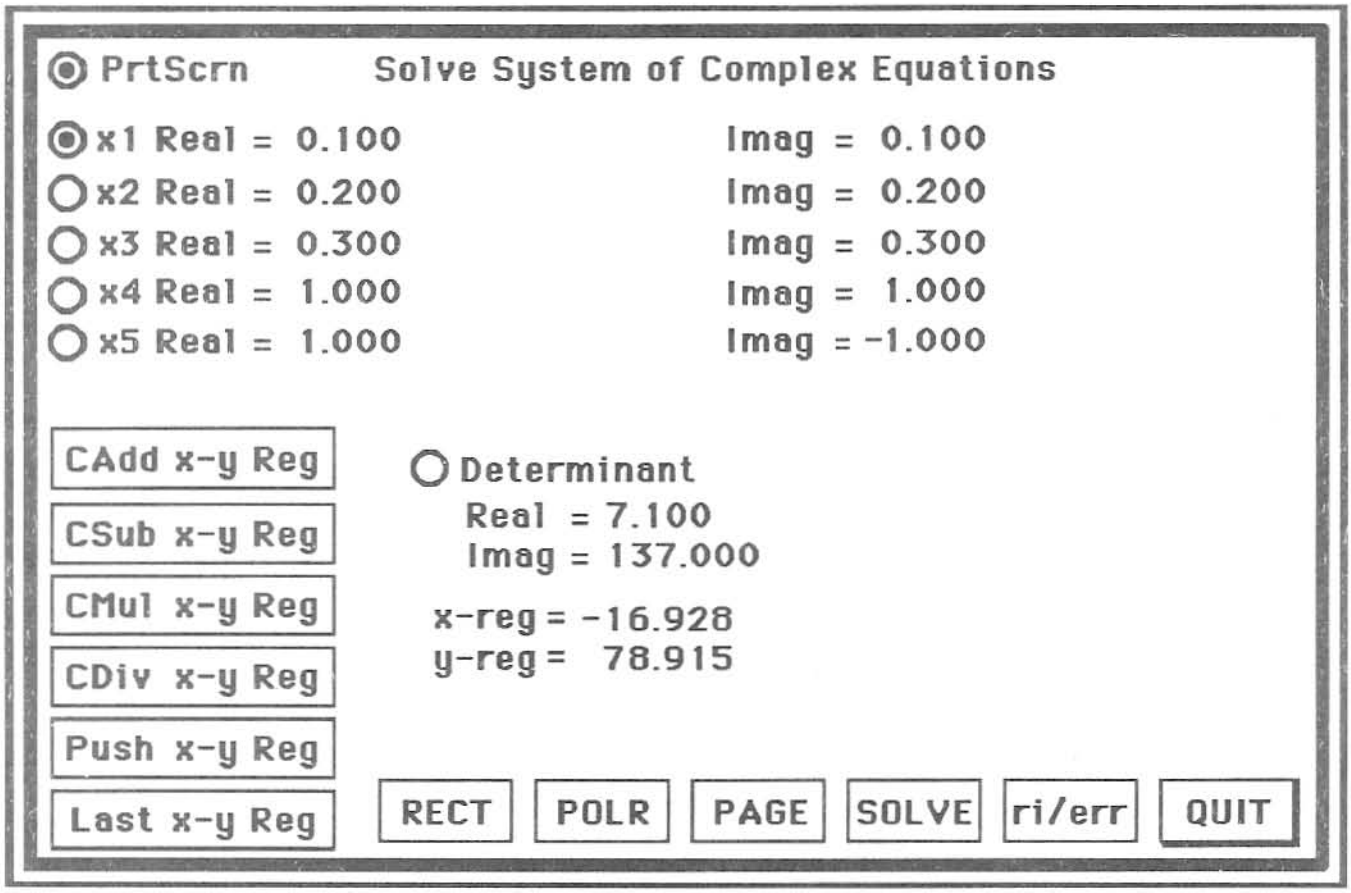

Figure 7.6 Complex System Solution Dialog

For demonstration purposes, Figure 7.6 is the **Solve System of Complex Equations** dialog for a **10 x 10** system.

The dialog is similar to the real system solution screen. The only differences are the addition of three buttons, and the result values now display both a real and imaginary part. The stack operation buttons indicate complex operations, operating on both the **x-reg** and **y-reg**. Both **x-reg** and **y-reg** values are displayed on the dialog.

If the system size is greater than **5 x 5**, the solutions x_6 - x_{10} are shown by clicking on the **PAGE** button. Only five solutions are shown at one time. The solutions may be displayed in either polar or rectangular format, by selecting either the **RECT** or **POLR** button.

If you display the solutions in polar form, by selecting the **POLR** button, the values are not changed. Stack operations will still be done in rectangular format. The **POLR** button only determines how the solutions are displayed to the user.

When you call the **Solve System of Complex Equations** dialog for the **3 x 3** system you input earlier, only the x_1, x_2, and x_3 solution labels will be displayed. We will now solve the **3 x 3** system. After bringing up the dialog, click the **SOLVE** button. After a few seconds, the solutions will be displayed on the screen as follows:

$$x_1 = -0.492 - i2.018$$
$$x_2 = 0.193 - i0.745$$
$$x_3 = -0.019 + i0.176$$

The determinant of the coefficient matrix is displayed as -184.000 + i88. Click on the **POLR** button to display the solutions in polar format. The solutions in polar format are:

$$x_1 = 2.077 \angle -103.691^\circ \quad \text{(polar form)}$$
$$x_2 = 0.769 \angle -75.463^\circ \quad \text{(polar form)}$$
$$x_3 = 0.177 \angle 96.120^\circ \quad \text{(polar form)}$$

and the determinant is displayed in polar form as $203.961 \angle 154.440^\circ$.

The solution errors will always be displayed in polar format, as the magnitude of the error is the best indicator of the accuracy of the solution.

A Practical Example

We will now input and solve a **3 x 3** complex matrix that represents a practical problem found at the university level. Although this is a typical electrical engineering problem, it would be encountered by other engineering students and physics majors, in introductory electricity courses.

Electrical engineers deal with the variable **i** as the current variable in circuits having time variant (alternating current) voltage sources. To keep from confusing the problem by mixing the imaginary number variable **i**, with the alternating current variable **i**, it is common, in solving problems dealing with electricity, to use **j** as the variable to indicate the imaginary square root of -1. We will use this convention, in solving this problem. In other words, 3 + **j**4 is exactly equal to 3 + **i**4.

We will solve the circuit in Figure 7.4, in this example.

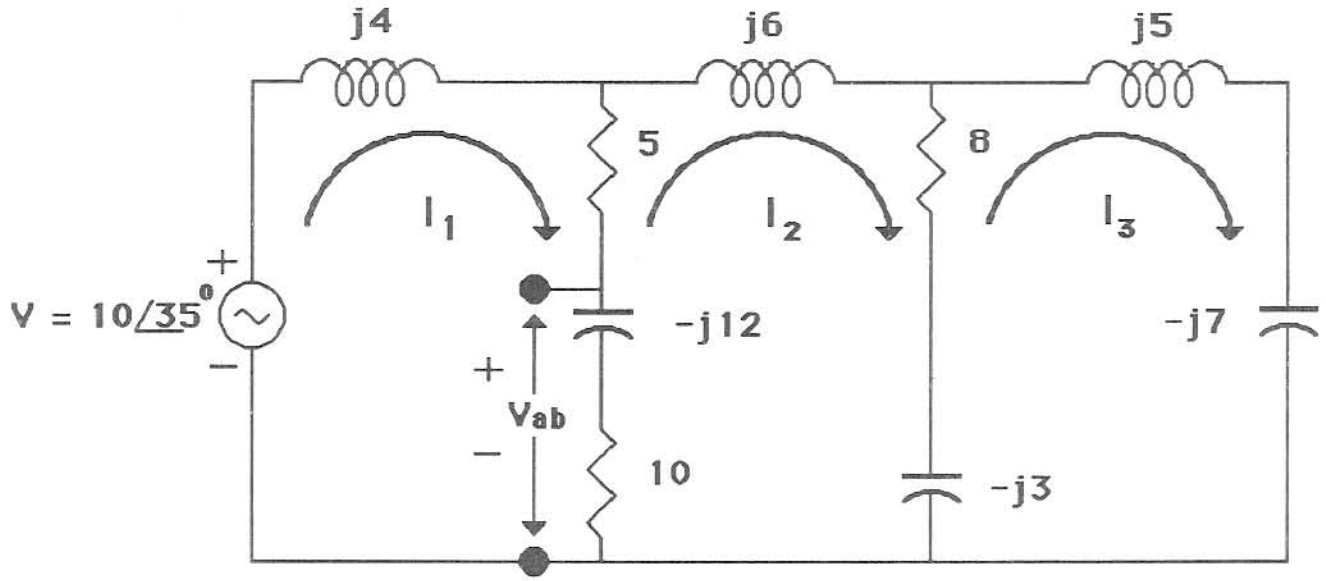

Figure 7.4 Three Mesh Complex Circuit

For those not familiar with electrical circuits, this is a three mesh circuit, driven by a complex voltage source of **10 ∠35°**. The circuit elements consist of inductors, capacitors and resistors. The resistors are the real terms in the coefficient matrix and the inductors and capacitors are the imaginary terms. The problem is to solve for the currents $\mathbf{I_1}$, $\mathbf{I_2}$, and $\mathbf{I_3}$, and to solve for the complex voltage $\mathbf{V_{ab}}$, shown on the diagram. Finding the voltage $\mathbf{V_{ab}}$ will demonstrate the use of the stack operations. Prior to starting this problem, put the calculator **trig mode** in the degrees mode.

Reset the linear systems matrix to 3 by using the **Reset** menu item, then select the **Enter Complex System** item from the **LINEAR SYSTEMS** menu. Linear circuits, such as this one always generate symmetric coefficient matrices. Click on the **Symmetric Entry** radio button, to put the entry dialog into the Symmetric Entry mode.

By using direct function entry, we can enter the matrix by inspection. Again for those not familiar with circuit analysis, the diagonal elements $\mathbf{a_{11}}$, $\mathbf{a_{22}}$, and $\mathbf{a_{33}}$ are the sum of the complex impedances in mesh 1, mesh 2, and mesh 3 passed through by $\mathbf{I_1}$,$\mathbf{I_2}$, and $\mathbf{I_3}$. The off diagonal elements are the sum of the complex impedances in mesh 1 passed through by $\mathbf{I_2}$ ($\mathbf{a_{13}}$), and $\mathbf{I_3}$ ($\mathbf{a_{13}}$), the sum of the complex impedances in mesh 2 passed through by $\mathbf{I_1}$ ($\mathbf{a_{21}}$), and $\mathbf{I_3}$ ($\mathbf{a_{23}}$), and the sum of the complex impedances in mesh 3 passed through by $\mathbf{I_1}$ ($\mathbf{a_{13}}$) and $\mathbf{I_2}$ ($\mathbf{a_{32}}$). For this type circuit $\mathbf{a_{12} = a_{21}}$, $\mathbf{a_{13} = a_{31}}$ **and** $\mathbf{a_{23} = a_{32}}$, therefore the matrix is symmetric. A complex impedance is the sum of the real

values of the resistance plus the sum of the imaginary values of the capacitive and inductive reactance. There will only be one non-zero value in the constant matrix. That value will be $\mathbf{b_{11}}$ and will be the value of the voltage source $\mathbf{V = 10 \angle 35^o}$. Input the linear system matrix as follows:

5+10	ENTER	The $\mathbf{I_1}$ resistive elements in mesh 1 ($\mathbf{a_{11}}$).
4-12	ENTER	The $\mathbf{I_1}$ reactive elements in mesh 1 ($\mathbf{a_{11}}$).
-(5+10)	ENTER	The $\mathbf{I_2}$ resistive elements for mesh 1 ($\mathbf{a_{12}}$).
-(-12)	ENTER	The $\mathbf{I_2}$ reactive elements for mesh 1 ($\mathbf{a_{12}}$).
	ENTER	The $\mathbf{I_3}$ resistive elements for mesh 1 ($\mathbf{a_{13}}$).
	ENTER	The $\mathbf{I_3}$ reactive elements for mesh 1 ($\mathbf{a_{13}}$).
10	ENTER	The magnitude of the voltage **V b[1]**.
35	PTOR	Enter angle of the voltage **V b[1]** and convert to rectangular form.
5+8+10	ENTER	The $\mathbf{I_2}$ resistive elements in mesh 2 ($\mathbf{a_{22}}$).
-12+6-3	ENTER	The $\mathbf{I_2}$ reactive elements in mesh 2 ($\mathbf{a_{22}}$).
-8	ENTER	The $\mathbf{I_3}$ resistive elements for mesh 2 ($\mathbf{a_{23}}$).
-(-3)	ENTER	The $\mathbf{I_3}$ reactive elements for mesh 2 ($\mathbf{a_{23}}$).
	ENTER	Enter the real part of constant **b[2]**.
	ENTER	Enter the imaginary part of coefficient **b[2]**.
8	ENTER	The $\mathbf{I_3}$ resistive elements in mesh 3 ($\mathbf{a_{33}}$).
-3-7+5	ENTER	The $\mathbf{I_3}$ reactive elements in mesh 3 ($\mathbf{a_{33}}$).
	ENTER	Enter the real part of constant **b[3]**.
	ENTER	Enter the imaginary part of coefficient **b[3]**.

Review the matrix with the scroll bar, and you will see the matrix values are:

$$\mathbf{(15 - j8)x_1 + (-15 + j12)x_2 + (0 - j0)x_3 = 8.192 + j5.736}$$
$$\mathbf{(-15 + j12)x_1 + (23 - j9)x_2 + (-8 + j3)x_3 = 0 - j0}$$
$$\mathbf{(0 + j0)x_1 + (-8 + j3)x_2 + (8 - j5)x_3 = 0 + j0}$$

Notice the symmetry of the off-diagonal elements $\mathbf{a_{12}}$, $\mathbf{a_{13}}$, $\mathbf{a_{21}}$, $\mathbf{a_{23}}$, $\mathbf{a_{31}}$ and $\mathbf{a_{32}}$. This symmetry allows us to use **Symmetric Entry** and save time in entering the elements. By using **Symmetric Entry**, when a matrix is symmetrical, you will lessen the chances of making a data input error, since you will be entering fewer numbers.

We can now solve for the currents I_1, I_2, and I_3, by leaving the dialog by keying on the **QUIT** button. We then select the **Solve** menu item from The **LINEAR SYSTEMS** menu, to compute the solutions of the currents.

After bringing up the dialog, click the **SOLVE** button. The solutions are:

$I_1 = 0.805 - j0.832$ amps
$I_2 = 0.689 - j1.093$ amps
$I_3 = 0.808 - j0.846$ amps

The determinant of the coefficient matrix is displayed as 1300.000 + j528.000. In polar form:

$I_1 = 1.157 \angle -45.959°$ (polar form)
$I_2 = 1.292 \angle -57.770°$ (polar form)
$I_3 = 1.170 \angle -46.321°$ (polar form)

The second part of the problem is to determine the value of the voltage V_{ab}. The voltage across the resistor and capacitor is computed using Ohms law. The voltage is equal to the current through the resistor and capacitor multiplied by the value of the complex impedance. By inspection, we know the current though the elements is $I_1 - I_2$, so the voltage would be $I_1 - I_2$ multiplied by the impedance 10 - j12. ($V_{ab} = (I_1 - I_2)$ **(10 - j12)**). We will use the two stack operation buttons **CSub x-y reg** and **Push x-y reg** to compute the value of $I_1 - I_2$. Push the value of I_1 to the stack by clicking on the radio button to the left of the label **x1 Real =** and then click on the button **Push x-y reg**. This pushes the value of **x1** (I_1) to the **x-reg** and **y-reg**. Then click on the radio button to the left of **x2 Real =** and click on the **CSub x-y reg** button . This does a complex subtraction of the value of x_2 (I_2) from the complex numbers in the **x-reg** and **y-reg** (I_1). The value of $I_1 - I_2$ is now shown by the labels **x-reg** and **y-reg** as 0.116 +j0.260. Now leave the dialog and return to the **Scientific Calculator** screen by clicking on the **QUIT** button. The solution is completed by entering the value of the impedance 10 - j12 into the calculator and doing a complex multiplication. This is done as follows:

r[10,-12	**ENTER**	Push the impedance value to the x-reg and y-reg.
	CMUL	The voltage is shown in the **x-reg** and **y-reg** and is $V_{ab} = 4.284 + j1.215 = 4.453 \angle 15.831°$ (polar form). This demonstrates using the stack operations to solve problems.

We should point out that direct function entry can be very useful in entering problems of this type. Suppose the complex impedance elements are given in terms of inductance (Henries) and capacitance (Farads). Knowing the frequency of the circuit, you would compute the complex impedances using the following formulas:

$$Z_l = 2 * pi * f * L \quad \text{and} \quad Z_c = \frac{-1}{2 * pi * f * c}$$

Given L = 0.002 Henries, c = 1.5e-6 Farads and the frequency is 1.67e6 Hertz you may enter these directly by setting the constant **2 * pi * f** equal to some memory location, say w by the following:

w=tpi*1.67e6	ENTER	Store the omega term (**2 * pi * f**)in memory location w. A memory operation will not advance the matrix element.

Then the complex term of a circuit containing the two elements given above can be enter directly with direct function entry as follows:

0.002*w-1/(1.5e-6*w)	ENTER	This will compute the complex term, put it into the matrix element and advance the matrix to the next element.

Linear System File Operations

You may save and recall linear systems to disk files by using the **Save System File** and **Load System File** items from the **LINEAR SYSTEMS** menu. They will also be saved and recalled with the **Session Save** and **Session Recall** items under the **FILES** menu. If the linear system has been solved, the solutions and errors will be saved with the file, and you will now have to recompute it when you call it back. When you select either of the above file operations, the **Standard File** dialogs will come up. You must name the file you wish to save, and select the file you want to recall with the mouse.

LINEAR REGRESSION

Linear regression is used to find a curve y = f(x) that best fits a series of n (x,y) data points. There are many practical uses for regression, and it is often used to:

1. Determine if a mathematical relationship exists between the independent variable x and the dependent variable y.
2. Prediction of y given a value of x.
3. To determine the reasons for a relationship between x and y.

In computing regression curves, we may try to fit any of the following types of curves to the (x,y) data, and determine which type of curve provides the best fit to the data. The type curves that the program will fit are:

1. $y = bx + a$	**Linear Curve Fit**
2. $y = b\ln(x) + a$	**Log Curve Fit ($x > 0$)**
3. $y = ae^{bx}$	**Exponential Curve Fit ($y > 0$)**
4. $y = ax^b$	**Power Curve Fit ($x > 0$ and $y > 0$)**

The general form of each type of regression curve is shown in Figure 8.1:

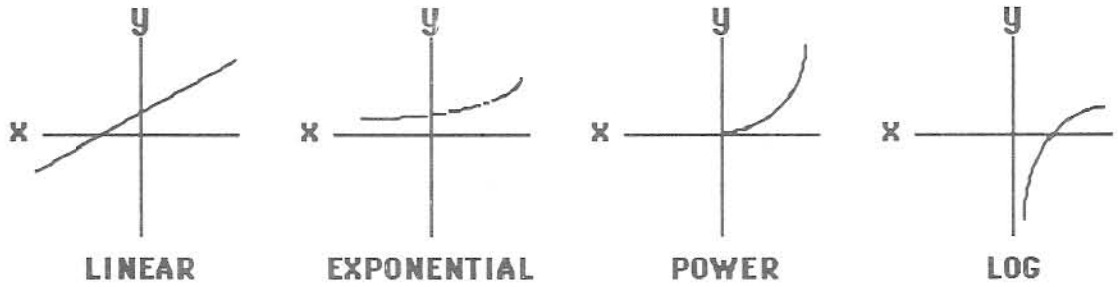

Figure 8.1 Types of Regression Curves

As can be seen from Figure 8.1, the four type curves allow fitting of regression to (x,y) data that has different characteristics than that of a straight line. Many relationships, between variables, do not correspond to a straight line. The ability to fit the Log curves, Exponential curves and Power curves to x-y data, in addition to the straight line, greatly increases your power to model data. The program will, if selected by the user, compute all four types of curves on x-y data, and return the type of curve whose correlation coefficient is highest. The curve with the highest correlation coefficient, fits the curve with the least error, as measured by the "Least Square" technique of regression.

GSNumerics will solve for the constants **a** and **b**, for each type of curve, using the "Least Square" technique. It also computes the correlation coefficient **r** for each type of curve. The correlation coefficient is a measure of the validity of using the particular regression curve to model the data. Data that is perfectly predicted by the "Least Square" technique will have a correlation coefficient of either +1 or -1. The sign of the correlation coefficient is determined By the slope of the constant **b**, in the formula. Completely unrelated data will return a correlation coefficient of 0. Values of r will normally fall between -1 and 1, depending on how well the data is related. The absolute value of the correlation coefficient is the indicator of how well the regression curve fits the actual x-y data. In other words, a regression with a correlation coefficient of -0.95 is a better fit than a regression with a correlation coefficient of 0.8.

GSNumerics allows you to do the following operations on x-y data:

1. Input x-y data pairs.
2. Perform any of the four types of linear regression on the x-y data.
3. Perform all four types of linear regression and let the program select the curve with the highest correlation coefficient.
4. Solve the computed curve for y or solve the computed curve for x.
5. Pass the selected regression to the **Function Operations** dialog, to allow for computation of the slope, at a given x value or to compute the area under the curve of the regression formula.
6. Graph the actual x-y data and superimpose a regression curve, computed at the data points, over the x-y data graph, using the **Graph x-y Data** dialog.
7. Graph a computed regression curve over any range, using the **Graph Linear Reg** dialog.
8. Sort x-y data pairs in ascending order of the x value.
9. Save and recall x-y data and regression solutions to disk files.

The first step in computing regression curves is to input the x-y data into the program. The x-y data is a set of ordered pairs (x,y), with x being the independent variable and y being the dependent variable.

It is not necessary to enter the x-y data in ascending or deceasing order of the x value. **GSNumerics** allows you to sort the data, after it has been entered, so don't worry about the order when entering the data. The order of the x-y pairs will not change the computation of the regression curves.

To input x-y data select the **Enter x-y Data** item from the **Regression MENU**. The **x-y Data Entry** dialog is shown in Figure 8.2.

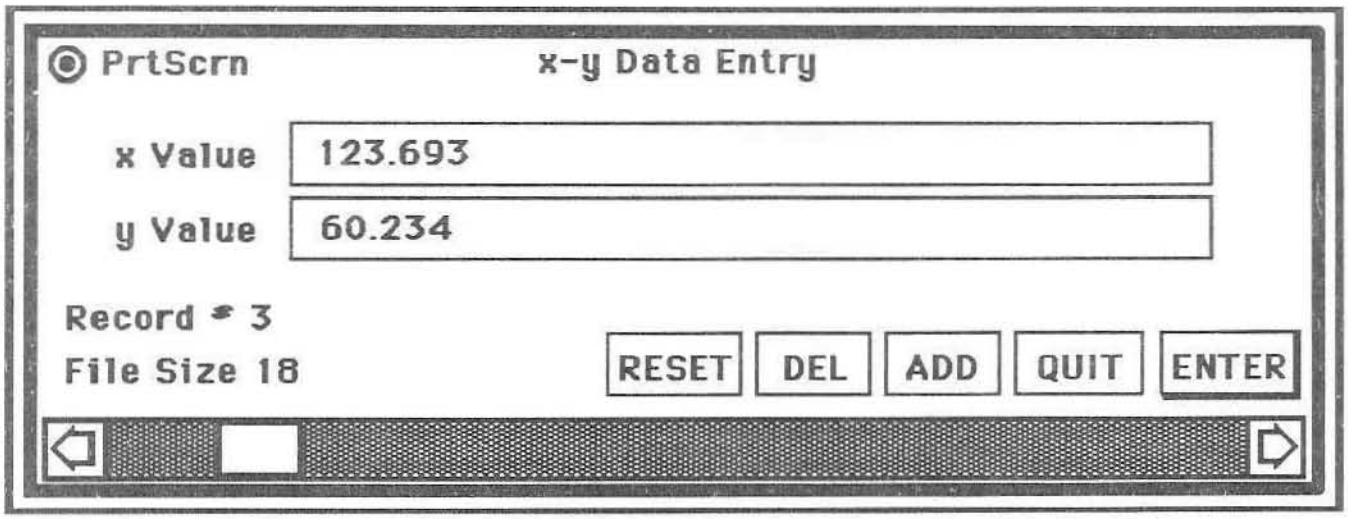

Figure 8.2 x-y Data Entry dialog

The data entry dialog contains two edit lines, used to input the x values and the y values. A scroll bar allows you to scroll through the data, to review the values or to edit values. The size of the x-y file is shown to the right of the label **File Size**. The **File Size** is the number of x-y data pairs that have been input. This number is incremented by one, each time the y value of an x-y pair is entered. The value to the right of the label **Record #**, indicates the record number whose x-y values are shown in the two edit lines.

The **x-y Data Entry** dialog contains five buttons. The RESET button will reset all x-y data values to zero, set the **File Size** to zero and set the input to **Record # 1**. This is used when you wish to input new data. The ENTER button is used to enter an x or y data value. If the entry is a y value, the **Record** label will be incremented, and if the entry was a new record the **File Size** will be incremented by one.

You may delete an x-y record input in error by using the DEL button. This button removes the record from the data base, and decrements the **File Size** by one.

You may also add a new record anywhere in the data file by keying on the ADD button. This button shifts all records above the selected record to the right one record, increases the file size by one and zeros the current x-y edit lines for new input. You may accomplish the same thing by adding the new record to the end of the file and sorting the x-y data file, when you have finished the data entry.

The QUIT button is used to quit the **x-y Data Entry** dialog and return to the main calculator screen. We will now demonstrate how to input x-y data points to be used in the regression computations.

Entering x-y Data

You may enter up to 300 x-y data points into the program. Data entry into the x and y edit lines can be done in three ways:

1. Entering a number, followed by the ENTER button or return key, will enter the number as the x or y data value.
2. Entering a function, using direct function entry, will set the value to the function value.
3. Entering a memory assignment with direct function entry (i.e. a = 2.35 or c = sin(b)/cos(d)) will not set the x or y value to the value of the memory assignment. This allows the user to build a single x or y data point, that is a combination of several functions. The data value can be set to the result by referencing the memory location, using direct function entry.

You will seldom use the second or third method of data entry for x-y data regression, since most regressions are computed on data observations obtained experimentally.

To demonstrate the use of the Regression calculations, we will enter the following x-y data and then solve the data for the "best fit" regression curve:

A coach clocked various boys in the 100 yard dash and has gathered the following data. He wants to find a function that will predict how fast a boy can run the 100 yard dash, given his age. The data is the average times for boys in the given age group.

Age (yrs)	9	10	11	12	13	14	15	16	17	18
Time(sec)	15.6	14.2	13.72	13.6	13.3	12.9	12.4	12.2	11.98	11.67

After bringing up the **x-y Data Entry** dialog, enter the data as follows:

	RESET	Reset the x-y data to zero to input new x-y data. This reset all x-y data points to zero and clears the way for new data entry.
9	ENTER	Input first age group.
15.6	ENTER	Enter average time for age group nine years.
10	ENTER	Input second age group.
14.2	ENTER	Enter average time for age group ten years.
11	ENTER	Input third age group.
13.72	ENTER	Enter average time for age group eleven years.

12	ENTER	Input fourth age group.
13.6	ENTER	Enter average time for age group twelve years.
13	ENTER	Input fifth age group.
13.3	ENTER	Enter average time for age group thirteen years.
14	ENTER	Input sixth age group.
12.9	ENTER	Enter average time for age group fourteen years.
15	ENTER	Input seventh age group.
12.4	ENTER	Enter average time for age group fifteen years.
16	ENTER	Input eighth age group.
12.2	ENTER	Enter average time for age group sixteen years.
17	ENTER	Input ninth age group.
11.98	ENTER	Enter average time for age group seventeen years.
18	ENTER	Input tenth age group.
11.67	ENTER	Enter average time for age group eighteen years.

This completes the x-y data entry. Notice how the **Record** number and **File Size** were increased by one each time you entered the y value. When entering a new record the **Record** number will always be one greater than the **File Size**, since the new record is not yet entered. The **File Size** will now indicate **10**, showing that you have entered ten x-y data records into the file. You may use the scroll bar to inspect your entries, to make sure that they are correct. You may delete a record with the DEL button, if you entered too many records by mistake . Leave the **x-y Data Entry** dialog by clicking on the QUIT button.

This will bring up a dialog asking you if you want to save the x-y data to a file. Click on the NO button to return to the main calculator screen. If you wanted to save the x-y data entered, you could click on the YES button. This would bring up the **Standard Save File** dialog and allow you to save the data to a disk file. This is a very good idea, if you have entered a large amount of data. In the event, you turned off the computer in error or have a power failure, you will not lose the data you input. We only entered ten data points in this example, and it is not critical.

x-y Data Regression Solutions

When you have completed the x-y data entry for a linear regression problem, you are ready to compute the actual regression equations. You must leave the **x-y Data Entry** and select the **Regression** dialog to complete the regression solutions. To compute the regression equations select the **Solve** menu item from the **REGRESSION MENU**. This will bring up the **Regression** dialog, shown in Figure 8.3.

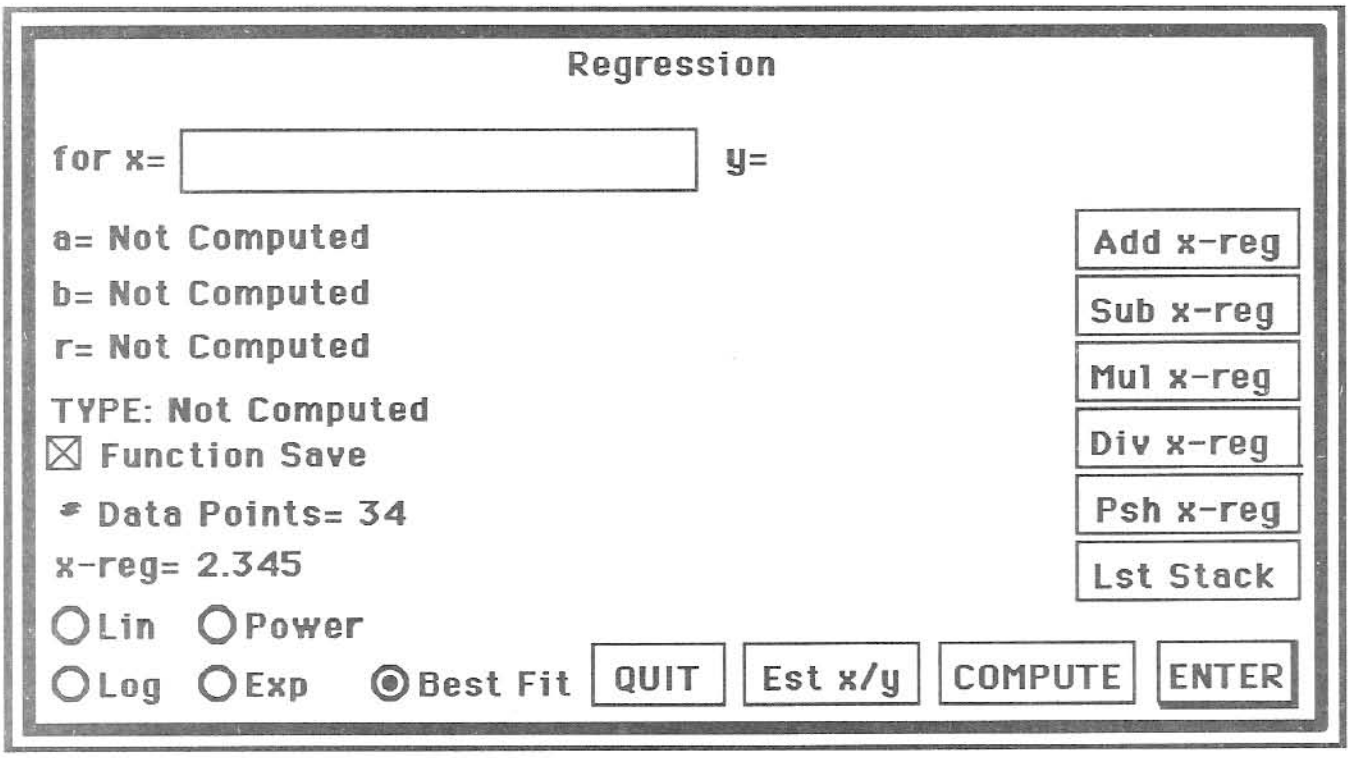

Figure 8.3 Regression Solution dialog

The **Regression** dialog contains an edit line that is used to input x value and y values, to get the solution of a regression curve at a specific x or y value. Five radio buttons are used to select the type of regression you want to compute or to select a desired computed regression. If, after entering new x-y data, you wish to compute all types of curves and let the program tell you which is the "best fit", select the "**Best Fit**" radio button, before computing the regression curve. You may also compute a specific type of regression by selecting either the "**Lin**" , "**Log**", "**Power**", or "**Exp**" radio button, before computing the regression.

The label "**a=**" will display the **a** constant for the regression displayed. Likewise, the "**b=**" and "**r=**" labels will display the **b** constant and the regression coefficient **r**. The "**TYPE:**" label will display the form of the selected regression (i.e. **y = bx + a** for a linear type). These labels will display "**Not Computed**", if the selected type of regression curve has not been computed. The current value of the **x-reg** is shown to the right of the label "**x-reg=**" and the number of data points is shown by the label "**# Data Points=**".

The **Regression** dialog has ten buttons. The **Add x-reg**, **Sub x-reg**, **Mul x-reg**, **Div x-reg**, **Psh x-reg** and **Lst Stack** buttons are used to do stack operations with the results obtained when solving a regression formula for a specific x or y value. This will be explained later.

After entering new x-y data and bring up the **Regression** dialog, you will initiate computation of the regressions by clicking on the [COMPUTE] button. After you have done the regression computation,you may enter a number in the edit line and click on [ENTER] to compute the value of the selected regression formula at a specific value of x. This will be shown by the label "**y=**". If you click on the [Est x/y] button, the label "**for x=**" will change to "**for y=**" and the label "**y=**" will change to "**x=**". A value entered into the edit line will now solve the regression for x, given a specific value of y. You may change back and forth from solving for y or solving for x with the [Est x/y] button. You can return to the main calculator screen by clicking on the [QUIT] button.

The last control on the **Regression** dialog is a square check control labelled "**Function Save**". If you click on this button with the dialog in the solve for y mode, indicated by the label "**for x=**", you will save the **a** constant in mem [A], the **b** constant in memory [B] and the regression formula in the function memory. If the dialog is in the solve for x mode, indicated by the label "**for y=**", you will save the **a** constant in mem [A], the **b** constant in mem [B], and the inverse function (solving the regression formula for y) in the function memory. You may then compute the slope or area under the curve by selecting the appropriate item under the **FUNCTION** menu.

We will now compute the regressions for the data entered previously. After bringing up the Regression dialog, select the radio button "**Best Fit**" and click on the [COMPUTE] button. This causes the program to compute each of the four type of regression and display the "best fit" type. After a few seconds the computation will be completed and the following will be shown on the screen:

a=34.547 b=-0.376 r=-0.984
TYPE: y= a * x exp[b] Power radio button highlighted

This tells you the best fit to the x-y data is the Power function $y = 34.547\ x^{[-0.376]}$, with a correlation coefficient **r = -0.984**. The negative correlation coefficients is expected, since the time to run the 100 yard dash goes down as the age of the boys increases. The absolute value of **r** is very close to 1.0, indicating the power curve is a very good model.

By clicking on the other radio buttons, you can see the correlation coefficients and constants for the other types of curves:

Linear	$y = -0.379\ x + 18.271$	$r = -0.964$
Logarithmic	$y = -5.022\ \ln[26.109]$	$r = -0.978$
Exponential	$y = 19.253\ e^{[-0.028\ x]}$	$r = -0.974$

Predicting x and y Values

Now suppose a new boy comes to town who is interested in running the 100 yard dash. He is 14 and one-half years old. Can you make a prediction of his time in the 100 yard dash? Make sure the dialog shows the "**for x=**" and "**y=**" labels, then input the following:

14.5	ENTER	The solution **y = 12.650** is a good predictor of his time, if he is an average runner.

Suppose you knew a boy could run the 100 yard dash in 11.1 seconds. Make a prediction as to how old he is:

	Est x/y	Change to "**for y=**" mode. Solve for x.
11.1	ENTER	x = **20.534** years old.

Before you do a lot of regressions on stock prices and then invest all of your "hard earned" money in the stock market, be aware that predictions from regressions can be very misleading. For instance, if you great grandfather is 98 years old, predict his time in the 100 yard dash, using our regression formula?

	Est x/y	Change to "**for x=**" mode. Solve for x.
98	ENTER	x = **6.171** seconds!!! I bet my "hard earned" money he can't do it!

Another example of how regression predictions can get you into trouble, would be to compute the time it would take a six months old child to "run" the 100 yard dash.

0.5	ENTER	x = **44.824** seconds!!! Pretty good time for a baby who can't walk!

Even though you may have a very high correlation coefficient, be very careful of predictions outside the range of the low and high x values you used in computation of the regression curves. Using the results of Regression analysis, without being aware of the "pitfalls", can lead to serious errors in analyzing data. Regression is an extremely powerful tool for analyzing data, but, you must know what you are doing and and how to interpret the results. Regression is a part of mathematics known as Statistics. Statistics covers a very large range of subjects used to analyze numbers and used to predict the errors in using the results of number analysis. To use Regression analysis to its fullest, you must use some of the more advanced concepts of Statistics, to assist you in knowing how much confidence you can put into regression predictions.

Slope and Areas of Regression Curves

When you have computed the regression curves for a set of x-y data, you may be interested in computing the slope of the curve or computing the area under the curve between two points. This can be very useful in analyzing x-y data.

GSNumerics allows you to pass the regression to the **Function Operations** dialog to make these computations. This is done by clicking on the "**Function Save**" button. If the **Regression** dialog is in the solve for y mode (i.e. **for x=**), the regression curves will be placed in the function storage used by the **Function Operations** dialog, the constant **a** will be placed in mem [A], and the constant be will be placed in mem [B]. Conversely, if the dialog is in the solve for x mode (i.e. **for y=**) the current regression formula will be placed in the function storage written to solve for x, given a specific y value.

The formulas will be passed as follows, depending on which type regression is selected when the "**Function Save**" button is activated:

Solve for y mode

1. **b * x + a**	**Linear Curve Fit**
2. **b * ln(x) + a**	**Log Curve Fit (x > 0)**
3. **a * eexp(b * x)**	**Exponential Curve Fit (y > 0)**
4. **a * x ^ b**	**Power Curve Fit (x > 0 and y >0)**

Solve for x mode

1. **(x - a)/b**	**Linear Curve Fit (b not equal 0)**
2. **eexp((x-a)/b)**	**Log Curve Fit (b not equal 0)**
3. **ln(x / a)/b**	**Exponential Curve Fit (b not equal to 0)**
4. **eexp(ln(x / a)/b)**	**Power Curve Fit (a or b not equal to 0)**

Notice that the **solve for x mode** formula will be passed as a function of **x**, since the **Function Operations** dialogs always assumes x as the independent variable. Remember, the [A] mem and [B] mem will contain the **a** and **b** constants, so don't change them before calculations are made.

Stack Operations

You may pass the result of solving a regression formula for x or y to the **Scientific Calculator** stack by using the stack operation buttons. These buttons perform the following operations on the stack **Scientific Calculator** stack registers:

Add x-reg	Adds the x or y solution to the current **x-reg** value, and places the result into the **x-reg.**
Sub x-reg	Subtracts the x or y solution from the current **x-reg** value, and places the result into the **x-reg.**
Mul x-reg	Multiplies the current **x-reg** value by the x or y solution and places the result into the **x-reg.**
Div x-reg	Divides the current **x-reg** value by the x or y solution and places the result into the **x-reg.**
Psh x-reg	Pushes the x or y solution to the stack.
Lst Stack	Recalls the Last Stack to the stack registers. Used to correct errors.

Special Regression Operators

The following operators, when input into the **Scientific Calculator Input Register**, will perform the following polynomial operations, without bringing up the appropriate Regression dialog:

DATAG[c	Graphs the current x-y data. The c is optional and if included will cause the x-y data points to be connected.
LING[c	Graphs the current x-y data with the Linear Regression curve superimposed over the x-y points. The c is optional and if included will cause the x-y data points to be connected.
LOGG[c	Graphs the current x-y data with the Log Regression curve superimposed over the x-y points. The c is optional and if included will cause the x-y data points to be connected.
EXPG[c	Graphs the current x-y data with the Exponential Regression curve superimposed over the x-y points. The c is optional and if included will cause the x-y data points to be connected.
POWG[c	Graphs the current x-y data with the Power Regression curve superimposed over the x-y points. The c is optional and if included will cause the x-y data points to be connected.
LINGR[a,b	Graphs the Linear Regression prediction curve from $x_1 = a$ to $x_2 = b$.
LOGGR[a,b	Graphs the Log Regression prediction curve from $x_1 = a$ to $x_2 = b$.
EXPGR[a,b	Graphs the Exponential Regression prediction curve from $x_1 = a$ to $x2 = b$.
POWGR[a,b	Graphs the Power Regression prediction curve from $x_1 = a$ to $x_2 = b$.

The special operators provide a fast method for performing routine graphing tasks.

Regression File Operations

GSNumerics provides file operations to allow you to store x-y data and the regressions to a data file on disk. You may recall the individual files, at anytime, for further use. In addition, if the user saves a **Session**, the x-y data and the regressions will be recalled with the **Session Recall** menu item.

To save an x-y data file, select the **Save x-y File** from the **Regression** menu. When the **Standard File** dialog comes up, enter the name under which you want to save the current x-y data, and press **Save**. The current x-y data will be saved to the disk file with the name you selected.

To recall a polynomial, select the **Load x-y File** from the **Regression** menu. When the **Standard File** dialog appears, select the desired x-y data file, by double clicking on the name, or by selecting it and keying on **Open**. The selected x-y data file will be loaded into the program. **Caution**: recalling an x-y data file will replace the current x-y data with the x-y data in the file. The current x-y data will be lost.

GRAPH

The visual representation of mathematical functions through graphing, is a powerful way to analyze functions and to observe the characteristics of different classes of functions. **GSNumerics** provides for graphing polynomials, non-polynomial functions, user input x-y data and regression curves. The graphs are quickly drawn and the x and y axis are automatically scaled. The user may "magnify" any part of a graph, with simple mouse movements, allowing more detailed inspection of individual parts or graphical extraction of real roots of functions. The graphic routines will indicate ranges where a particular function is undefined. Graphs can be printed to the printer in color or black and white modes. Two functions may be drawn on the graph at once (overlaying), to assist in comparing two functions.

The graphic routines are divided into two types. The first type includes polynomials, non-polynomial functions and regression predictions. The second type includes x-y data and regression curves corresponding to the range and domain of the input x-y data. The regression curves can be superimposed over the actual x-y data.

Graphic routines are initiated by selecting the appropriate item under the **GRAPH** menu. This menu contains the following items:

1. **Function**	Used to graph the current function.
2. **Polynomial**	Used to graph the current polynomial.
3. **Regression**	Graphs a computed regression curve over any user defined domain of x values. Used for predictions from regression formulas.
4. **x-y Data**	Graphs user input x-y data and Linear, Log, Exponential and Power regression curves, over the range and domain of the x-y data. Regression curves can be superimposed over the x-y data, to more clearly show the relationship between the regression and the actual data.
5. **Function B**	Selects second function (or polynomial), if two graphs are overlaid in the graphics mode.
6. **Stack**	Brings up Stack dialog, enabling user to do stack operations with solutions generated in graphics mode.
7. **Area**	Computes the area between two selected points on the graph.
8. **Root**	Computes the root between two selected points on the graph.
9. **Slope**	Computes the slope of a selected point on the graph.
6. **Last Graph**	Redraws the last graph drawn prior to a magnification of a graph part.
7. **Quit Graph**	Returns the program to the **Scientific Calculator** screen.

Drawing a Function Graph

We will demonstrate the graphic feature by graphing the function **f(x) = sin(2*x)*cos(4*x)**. Prior to graphing this function please set the calculator **trig mode** to the radians mode (**RM**). Please enter this function, using the **Enter Function** dialog found under the **FUNCTION** menu. If you are not familiar with entering functions, this is explained fully in the **FUNCTION** chapter. After leaving the **Enter Function** dialog, select the **Function** item on the **GRAPH** menu. After selecting this item, the **Graph Function Dialog**, shown in Figure 9.1 will appear.

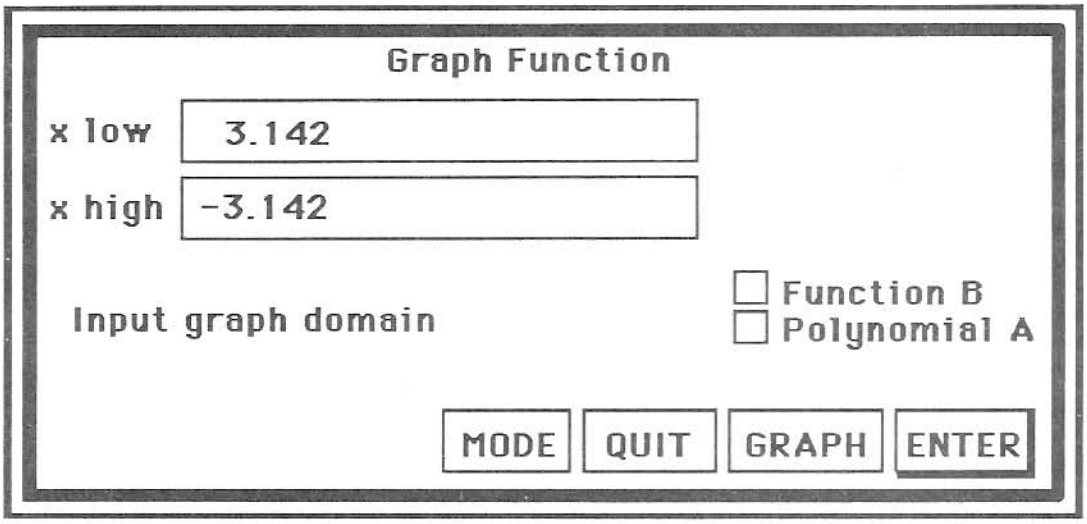

Figure 9.1 Graph Function dialog

This dialog is used to input domain of x values that you wish to graph. We will graph the function from **-pi** to **pi**. Enter the range as follows:

-pi	ENTER	Enter the low x value -pi.
pi	ENTER	Enter the high x value pi. You have now entered the range of x values to be graphed and are ready to graph the function.
	GRAPH	Clicking on this button will initiate the graph process. The computer will show a note "**Computing Graphic Parameters, Please Wait**" and a flashing red rectangle. During this time, the computer is computing the lets you know that it is constructing the graph, and all is proceeding well.

The time it takes to draw a graph depends on the function. Polynomials, using special polynomial algorithms will graph quickly. Large non-polynomial functions will take some time, as they must be parsed at each of the 300 data points on the graph.

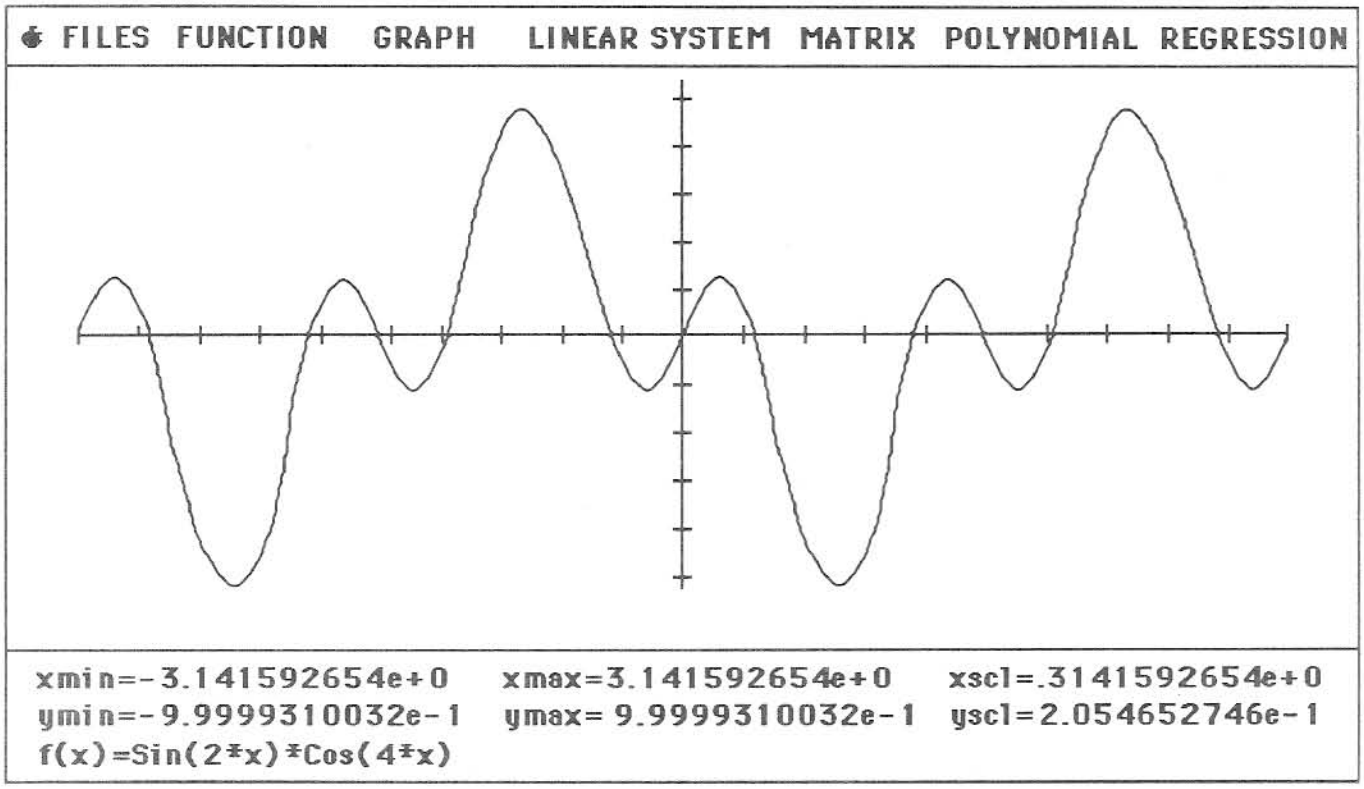

Figure 9.2 Function Graph of f(x) = sin(2x) Cos(4x)

The graph is divided into areas, the **Graph Area** and the **Display Area**. The **Graph Area** is the upper part of the screen, containing the x and y axis and the actual graph. The **Display Area** is the smaller window at the bottom, containing details about the graph.

The **Display Area** contains the following labels:

xmin	This shows the smallest x value for the graph. This corresponds to the left side of the x axis.
xmax	This shows the largest x value for the graph and corresponds to the right side of the x axis.
ymin	This is the value of the minimum y value plotted on the graph. In this case it is -1 and corresponds to the bottom tips of the graph, where the slope is zero.
ymax	This is the value of the maximum y value plotted on the graph, in this case -1. It corresponds to the top tips of the graph where the slope is zero.
xscl	Indicates the value of the range between each vertical bar on the x axis. The x axis will always be divided into twenty parts. In this case the x range is from -pi to pi, a range of 2pi and the scale is one-tenth pi per division.

yscl This is the value of the range between each of the horizontal lines on the y axis. Every y axis will be divided into ten vertical divisions. Since the y range of this graph is 2.0, from -1 to 1, the y scale is two tenths per division.

f(x)= The function being graphed will always be shown here.

The **xmax**, **xmin**, **ymax**, **ymin**, **xscl** and **yscl** will assist you in seeing the value of the graphed function, at different x and y points.

Determining the Value of Specific Points on a Graph

If you wish a more precise determination of the value at a specific point on the graph, put the cursor on a point on the graph whose value you wish to determine and click the mouse button. Notice the **Display Area** changes to show the following information about the point you selected on the graph.

x1 This is the x value of the graph where the mouse was clicked.

y1 This is the y value of the graph where the mouse was clicked.

For example, if you clicked the mouse at the first point where the graph crosses the x axis to the right of the x = 0 and y =0 point, you will see the x value at this point is shown to be 1.167691876 and the y value of the graph at this point is -3.002821728e-2. This is an approximation of the root of the function at this point, since the graph is crossing the x axis. Cancel the action by clicking in the **Display Area**. When you click in the display area, the original **Display Area** information will appear.

The graph is made up of 300 points. Each point represents an increment of the domain of 2pi divided by 300 or 0.02094 radians. This is the smallest change in x you will be able to graph on the initial graph. In other words, the definition of this graph is limited to discrete increments of 0.02094 radians for **x**. In the next section, we will show you how to magnify any part of the graph. When you do this, the discrete increments will get smaller and you will be able to determine the value of points on the graph more precisely. In fact, you can continue to magnify a point until the discrete increments of x will be as small as 1e-18, allowing you to very precisely determine the y value.

In review, to determine the value of a point on the graph, click the mouse on the point and the value of both x and y will be shown in the display area. Clicking the mouse in the **Display Area** cancels this action. Look at several points on the graph by clicking the mouse. Don't forget to cancel each action, prior to looking at the next point, by clicking in the **Display Area**. If you forget to cancel the action, you will initiate the graph magnification procedure, explained in the next section. You can cancel this by two clicks of the mouse, in the **Display Area**.

Determining the Area between Two Points on a Graph

To compute the area under the graph between two points, you first click on the first point and then click the mouse on the second point. Remember, you can cancel a selection by clicking in the **Display Area**. After the second point is selected, the **Display Area**. will show the following:

x1	The x value of the first point selected.
y1	The y value of the first point selected.
x2	The x value of the second point selected.
y2	The y value of the second point selected.
dx	The x range of the selection from x1 to x2 (x2 - x1).
dy	The change in y for the selection from x1 to x2 (y2 - y1).

After making the second selection, the graph will be redrawn with the color blue, over the range selected and a blue line will be drawn from the x-axis to the y1 and y2 points. This clearly shows the area you wish to compute. To compute the area select the **Area Item** from the **GRAPH MENU**. If you are graphing a non-polynomial function the program will do a **Romberg Area Calculation**, using the present error and iteration settings. This is explained in Chapter 5. If you are graphing a polynomial function the area will be calculated using the polynomial algorithm.

After a short time the following will be displayed in the **Display Area**:

Area	The area from point x1 to point x2.
Del	The last delta increment. This is an indication of the accuracy of the computation.
	Press mouse button to continue!

In review, to determine the area under the curve between two points on the on the graph, click the mouse first on x1 and then on x2. Clicking the mouse in the **Display Area** cancels either action. Select the **Area Item** from the **GRAPH MENU**. After a short time, the area will be displayed. Then click the mouse button to continue.

The result of the area calculated can be used in stack operations by selecting the **Stack Item** under the **GRAPH MENU**. This will be explained later in this chapter.

Finding a Function Root between Points on a Graph (analytically)

To find the root of a function between two points, you first click on the first point and then click the mouse on the second point. Remember, you can cancel a selection by clicking in the **Display Area**.
After the second point is selected, the **Display Area**. will show the following:

x1	The x value of the first point selected.
y1	The y value of the first point selected.
x2	The x value of the second point selected.
y2	The y value of the second point selected.
dx	The x range of the selection from x1 to x2 (x2 - x1).
dy	The change in y for the selection from x1 to x2 (y2 - y1).

After making the second selection, the graph will be redrawn with the color blue, over the range selected and a blue line will be drawn from the x-axis to the y1 and y2 points. These two points must straddle the root you wish to compute. To compute the area select the **Root Item** from the **GRAPH MENU**.

After a short time the following will be displayed in the **Display Area**:

Root	The x value of the root between the two selected points.
Err	The result of the function solved with x equal to the computed root. If the root computation was exact, this will be 0.0.
	Press mouse button to continue!

If the selected points did not straddle a root, you will get the following message:

Selected points must straddle a function root!!

In review, to compute a root between two points on the on the graph, click the mouse first on x1 and then on x2. Clicking the mouse in the **Display Area** cancels either action. Select the **Root Item** from the **GRAPH MENU**. After a short time, the area will be displayed. Then click the mouse button to continue.

The result of the root calculation can be used in stack operations by selecting the **Stack Item** under the **GRAPH MENU**. This will be explained later in this chapter.

Finding the Slope of a Function at a Point on a Graph

To find the slope of a function at a point, you first click on the point. Remember, you can cancel a selection by clicking in the **Display Area**. After the point is selected, the **Display Area**. will show the following:

x 1	This is the x value of the graph where the mouse was clicked.
y 1	This is the y value of the graph where the mouse was clicked.

After making the selection, a blue line will be drawn from the x-axis to the y. To compute the slope select the **Slope Item** from the **GRAPH MENU**.

After a short time the following will be displayed in the **Display Area**:

Slope	The x value of the root between the two selected points. Read the section on computing the slope of a non-polynomial function in Chapter 4. If the absolute value of the slope is large, be careful, as the answer may not be correct. If the function is a polynomial, the slope will be correct for all values, within the computational limits of the computer.

In review, to compute the slope at a point on the on the graph, click the mouse on the point. Clicking the mouse in the **Display Area** cancels either action. Select the **Slope Item** from the **GRAPH MENU**. After a short time, the slope will be displayed. Then click the mouse button to continue.

The result of the slope calculation can be used in stack operations by selecting the **Stack Item** under the **GRAPH MENU**. This will be explained later in this chapter.

Graph Stack Operations

You may use the results of graphic computations in stack operations by selecting the **Stack Item** from the **GRAPH MENU**. This will not be activated until you have made a selection on a part of the graph as explained previously. The Graphics Stack Operations Dialog (Figure 9.3) will come up. You then select the value you wish to use in stack operations, by keying the mouse on the proper radio button. You may pass the last x or last y selected, the computed area, the computed root or the computed slope. Operations include addition to the **x-reg**, subtraction from the **x-reg**, multiplying the **x-reg** by the selected value, dividing the **x-reg** by the selected value, pushing the selected value onto the stack, or recalling the Last Stack if an error is made.

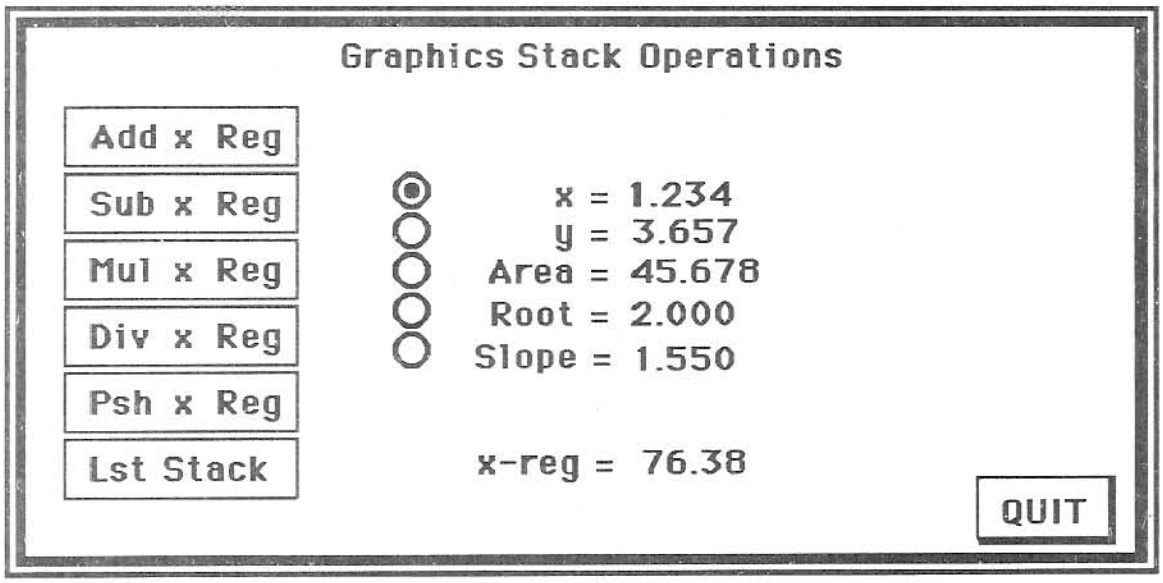

Figure 9.3 Graphics Stack Dialog

Magnifying Parts of a Graph

You can magnify or "explode" any part of a graph very easily. Select the low x point by clicking the mouse button on the graph at the low x point. This is exactly the same thing that you did in the previous section. Now, instead of cancelling the procedure, select another point, the high x point for the graph with the mouse, and click the mouse button. The **Display Area** will now show the following:

x1	The x value of the first point selected.
y1	The y value of the first point selected.
x2	The x value of the second point selected.
y2	The y value of the second point selected.
dx	The x range of the selection from x1 to x2 (x2 - x1).
dy	The change in y for the selection from x1 to x2 (y2 - y1).

When you make the second selection, the graph color will change to light blue on a color monitor or a different shade of gray on a black and white monitor, over the range where the new graph will be drawn. A light blue, vertical line will also be drawn from the **x1** point to the graph and the **x2** point to the graph, showing exactly the x range involved. You may cancel the selection by clicking twice in the **Display Area** of the graph. This will return you to the original display, and redraw the original graph. You may cancel any graph selection by keying the mouse in the **Display Area** of the graph.

After selecting a range, a single click, in the **Graph Area**, will cause the program to redraw the graph, using the **x1** and **x2** values as the new values for the x range. To see how this works, select two points on the graph, by clicking the mouse on each point. Then click the mouse in the **Graph Area** and watch the program magnify the graph. You can continue to magnify an area by selecting new points and clicking in the **Graph Area** again. You may redraw the original graph, at any time, by selecting the "**First Graph**" item on the **GRAPH** menu.

Finding Function Roots Graphically

To find a real root of a function, you merely select a point on each side of the root and magnify the graph. You can continue to magnify the area where the function crosses the x axis, and determine the root very precisely. The **dx** value will tell you how close you are to a root. For instance if the **dx** value is very small, say the value is 1e-15, you know you have isolated the root location very closely. It is possible to make an error and select **x1** and **x2** values that do not straddle a root. If you do this, select the **Last Graph** item from the **GRAPH** menu, and it will redraw the previous graph, allowing you to undo the error.

We will demonstrate finding a root of the current function graphically. Before we do this, select the **First Graph** item from the **GRAPH** menu, to redraw the original graph from **-pi** to **pi**. We will find the first root to the left of the point where the x and y axis intersect.

We start by picking two points, one on each side of the root, as close to possible to the root. Make sure that **y1** and **y2** have different signs, or you will not be straddling the root. After making the selection, the **Display Area** will show the following:

```
x1 =  -4.307869525e-1    x2 =  -3.887589571e-1
y1 =   1.151680049e-1    y2 =  -1.105573208e-2
dx =   4.202799537e-2    dy =   1.262237370e-1
```

Your values may be slightly different, depending on where you selected the two points. Now key the mouse in the **Graph Area** to redraw the graph. The new graph will be similar to the graph in Figure 9.4. Notice how we are starting to isolate the root. We now know the point is somewhere between the **x1** value of **-4.307869525e-1** and the **x2** value of **-3.8875892571e-1**. The **dx** value tells us we are within **4.202799537e-2** of the actual root.

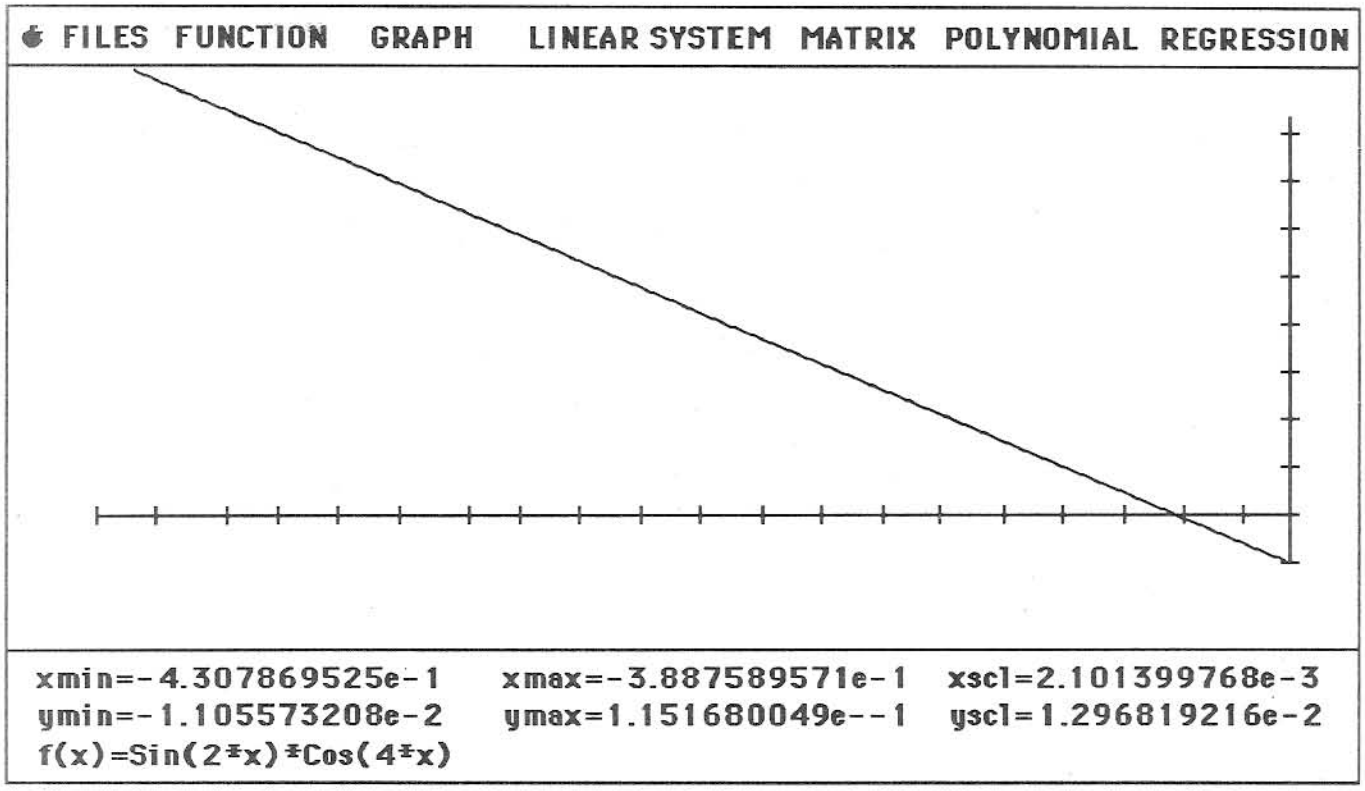

Figure 9.4 Graph Magnified around a real root

Now select two points, close to and on each side of the root on the magnified graph. After making the selection, the **Display Area** will show the following:

```
x1 = -3.931167747e-1   x2 = -3.926946891e-1
y1 = 1.181266968e-3    y2 = -1.242393117e-5
dx = 4.216855722e-4    dy = 1.193690889e-3
```

Again, your numbers may not be exactly the same as these, depending on what points you have selected on the graph. Now we know we are within the **dx** value of **4.216855722e-4** of the actual root and that the root is between the **x1** and **x2** values. You can magnify the graph again by clicking in the **Graph Area** with the mouse. Each time you expand the graph, you will isolate the root more precisely. We will leave it as an exercise to continue to magnify the graph and determine the root more accurately using the graphic technique.

You don't have to find the roots by continual magnification of the area around the root. When you select two points on each side of the root, the two x values are stored by the program, and passed to the **Function Roots** dialog as **x1** and **x2**. When you bring up the **Function Roots** dialog, these numbers will be be displayed in the edit lines, and all you have to do is compute the root.

Graphing Undefined Function Ranges

The graph of **log(x)** is not defined for values of x equal to or less than zero, since log(x) is not defined for those ranges. The function **y(x) = sin(x)*$\sqrt{x^2-1}$** is not defined where **x^2 - 1** is less than zero. When **GSNumerics** graphs a function over an undefined range of x values, it will plot a thick blue line, along the x axis, where the function is not defined. This will alert the user, that the function is not defined over that range of x. To demonstrate this, enter the function **log(x)**, using the **Function Entry** dialog, and then graph it from **x = -.5** to **x = 10**.

Since the graphic function divides the x range into three hundred discrete points, it may not show an undefined individual point. For instance, graph the function **f(x) =1/x** from **x= -0.1** to **x = 0.1**. Notice that the graph did not mark the point **x = 0**, as an undefined point. This would only occur if one of the three hundred calculations happened to fall exactly on 0.0. You can see that the singularity is obvious, as the graph obviously goes toward positive infinity in one direction and toward negative infinity in the other direction.

Using Graph Clip Limits

It is sometimes helpful to set maximum and minimum limits on the y values that will be graphed. We will demonstrate this by graphing the **y(x) = sec(x)**. Input (**Enter Function** dialog) the function and graph it over the x range of **-pi** to **pi**. Make sure the **trig mode** is set to radians, before graphing the function. As you can see, the graph hardly looks like a graph of the secant function. Now regraph the function, but this time click on the **MODE** button found on the **Graph Function** dialog and set a minimum clip of **-5** and a maximum clip of **5**. Then click on the **MODE** button and enter the limits of **-pi** to **pi**. Graph by clicking on the **GRAPH** button. The **MODE** button toggles the dialog back and forth between setting the x range and the clip limits. Setting clips of **-5** and **5** tells the program to not graph any value of the function greater than the maximum clip of **5** or less than the minimum clip of **-5**. Now regraph the secant function. The function now looks like the secant function you see in text books! Try the same thing with the function **y(x) = tan(x)**, without clip limits and with clip limits of **-5** and **5**. Notice how the clip limits let you graph the "classic" tangent function, you are probably used to seeing.

When you graphed these functions, without clip limits being set, the large values of f(x) where the functions are approaching infinity, dominate the automatic scaling of the graph function. The small values are completely dominated by the large values, when the scaling is done. Clip limits can be used to search for roots over large ranges of x. Just set small clip limits, say -0.0001 and 0.0001, before graphing the function. This will prevent large values of the function from dominating the automatic scaling of the graph, and allow you to see the areas of the function, where is is crossing the x axis.

Graphing Polynomials and Regression Predictions

Polynomials are graphed exactly like functions, except the graph generated is the graph of the current polynomial, and the graph is initiated by selecting the **Polynomial** item on the **GRAPH** menu. This will bring up the **Graph Polynomial** dialog (Figure 9.5), that is exactly like the **Graph Function** dialog. The procedure is the same. If you have not entered a polynomial using either **Polynomial Entry** or **Binomial Multiplication**, the **Graph Polynomial** menu item will be disabled.

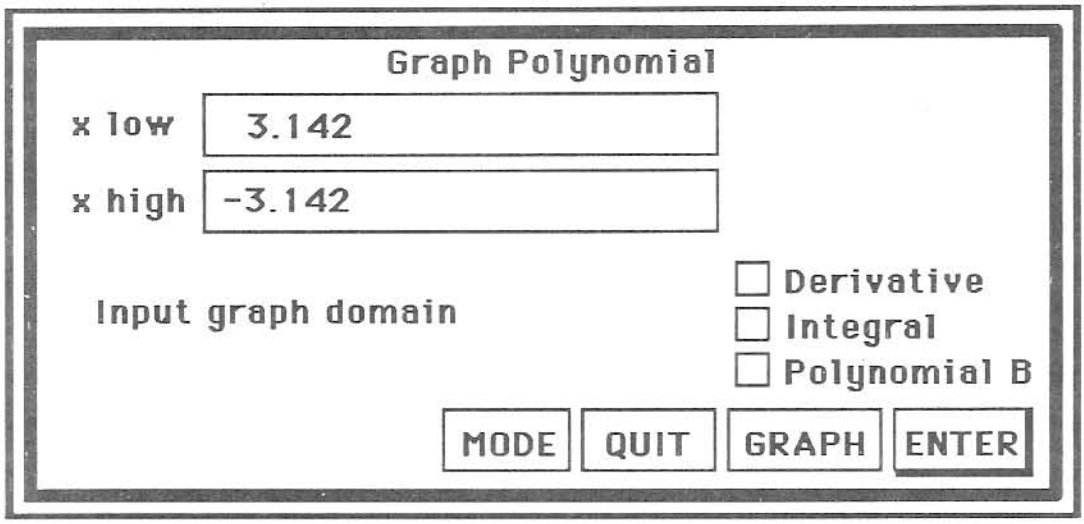

Figure 9.6 Graph Regression dialog

Selecting the **Regression** item on the **GRAPH** menu will bring up the **Graph Regression** shown in Figure 9.6. You must have entered x-y data and computed at least one type regression curve to select the **Graph Regression** item. If you have not computed, at least one regression curve, this menu item will be disabled.This dialog has four radio buttons labelled "**Lin**", "**Log**", "**Exp**", and "**Pow**". When doing a regression graph, you must tell the program which type of regression curve you wish to graph. If you have not computed a particular type of regression curve, the button for that type curve will be dimmed and disabled. The dialog comes up with the Linear regression curve selected.

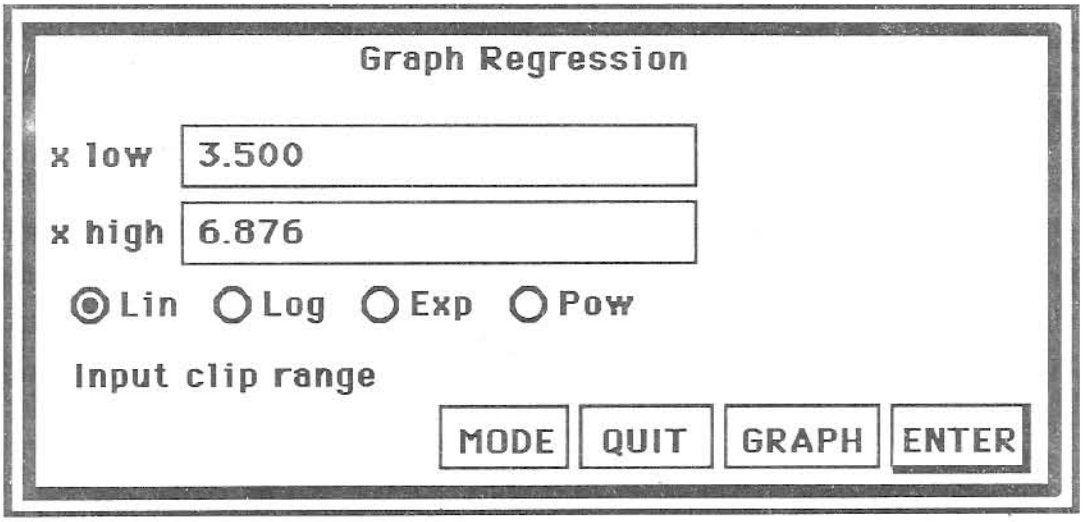

Figure 9.6 Graph Regression dialog

Except for the radio buttons used to select the type of curve to graph, graphing regression curves is exactly like graphing functions or polynomials. Remember, the menu items used to initiate graphs of polynomials, functions and regressions, will be disabled until a current function or current polynomial is entered, or a regression is done on x-y data.

Graphing x-y Data

The **Graph x-y Data** dialog, show in Figure 9.5, is used to graph x-y data. This dialog is selected from the **GRAPH MENU**, by selecting the **x-y Data** menu item. You will not be able to select this item unless you have input x-y data, using the **x-y Data Entry** dialog. If no x-y data has been entered the **x-y Data** menu item will be disabled.

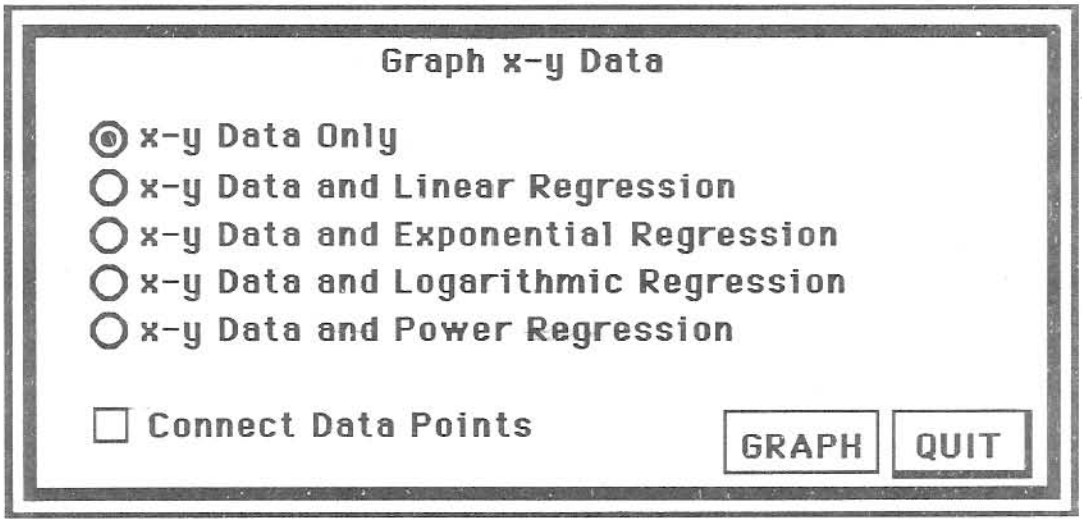

Figure 9.5 Graph x-y Data dialog

The dialog contains five radio buttons and a check box. The user has the option of:

1. Graphing the x-y data points only.
2. Graphing the x-y data with the Linear Regression curve superimposed over the data.
3. Graphing the x-y data with the Exponential Regression curve superimposed over the data.
4. Graphing the x-y data with the Logarithmic Regression curve superimposed over the data.
5. Graphing the x-y data with the Power Regression curve superimposed over the data.
6. Plotting the x-y data with the data points connected with lines, or plotting the individual points without connecting lines.

To graph the x-y data, select the combination of curves you wish to plot by clicking on the appropriate radio button with the mouse. If you want the data curves connected, click in the square check box. The check box will then display an "x". You can turn this off by a second click in the check box. Finally, click on the **GRAPH** button to draw the graph.

If the graph is drawn back and forth across the screen, instead of being drawn from left to right, the data points are not sorted in ascending order of x. This can be corrected by selecting the **Sort x-y Data** item from the **REGRESSION** menu. This will sort the x-y data, ascending on x, keeping the y values with the proper x values, and the graph will then draw properly.

You can't magnify the x-y data graphs. They only show the actual x-y data and the regression curves plotted at the x value of the actual data. To look at larger x ranges of the computed regression curves, use the **Regression** item on the **GRAPH** menu.

Graph Overlays (Graphing two functions simultaneously)

You may overlay two graphs in the graphic mode. The following overlays are possible:

1. Function B overlaying Function A.
2. Polynomial A overlaying Function A.
3. Derivative of Polynomial A overlaying Polynomial A.
4. Integral of Polynomial A overlaying Polynomial A.
5. Polynomial B overlaying Polynomial A.

This feature allows you to compare two functions or to compute the common solutions between two functions graphically. For instance, this feature allows you to see the relationship between the roots of a function derivative and the functions local maximum and minimum points.

To demonstrate this feature, enter the function **sin(2x)*cos(4*x)** as Function A, using the **Enter Function Item** on the **FUNCTION MENU.** Then enter the polynomial **.3183098862x** as Polynomial A. using the **Enter Polynomial Item** on the **POLYNOMIAL MENU.** Make sure the calculator is in the **Radians Mode.** After entering the function and polynomial, select the **Function Item** from the **GRAPH MENU.** The dialog show in Figure 9.7 will appear:

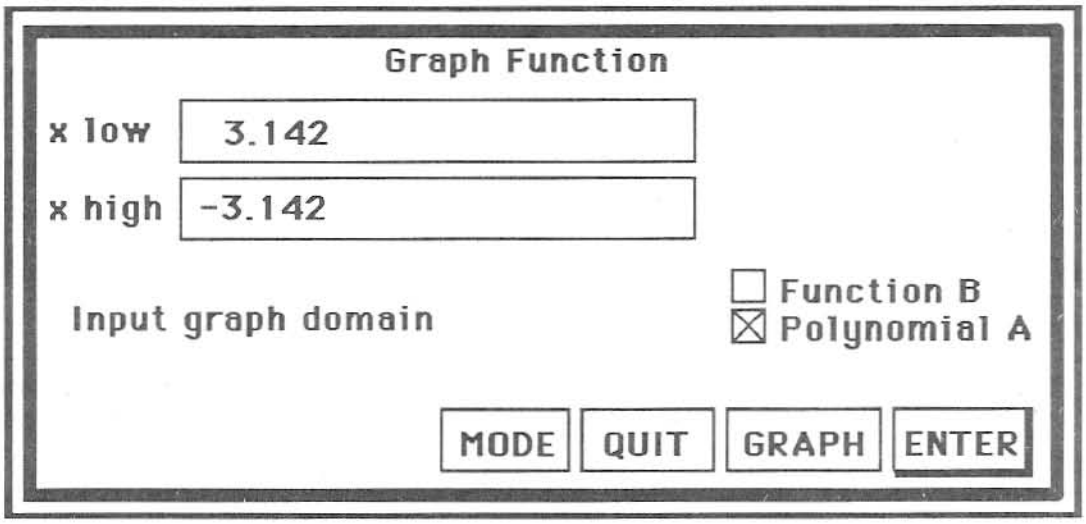

Figure 9.7 Graph Function dialog

Enter **-pi** for **x low** and **pi** for **x high**. Select the polynomial overlay by keying on the square check box to the left of the label Polynomial A. If you haven't entered a Function B or have not entered Polynomial A, the respective check box will not be shown on the dialog box.

If you were overlaying polynomials by selecting the **Polynomial Item** from the **GRAPH MENU**, the label Polynomial B will not be drawn unless a Polynomial B had been entered, using the Enter Polynomial Item on the **POLYNOMIAL GRAPH.**

We have now told the program to draw the graph of **f(x) = sin(2*x)*cos(4*x)** from **-pi** to **pi**, and to draw the polynomial **f(x) = .3183098862x** over the trigonometric function. The program will scale the graph automatically, using the maximum value of the function or the polynomial, depending on which is the greatest and the minimum value of the function or the polynomial, depending on which is the smallest. Initiate the graph by clicking on **GRAPH**.

Graphing overlays will take longer than graphing single functions, as the program must compute both functions at three hundred point over the selected domain. After a short time the graph shown in Figure 9.8 will be drawn.

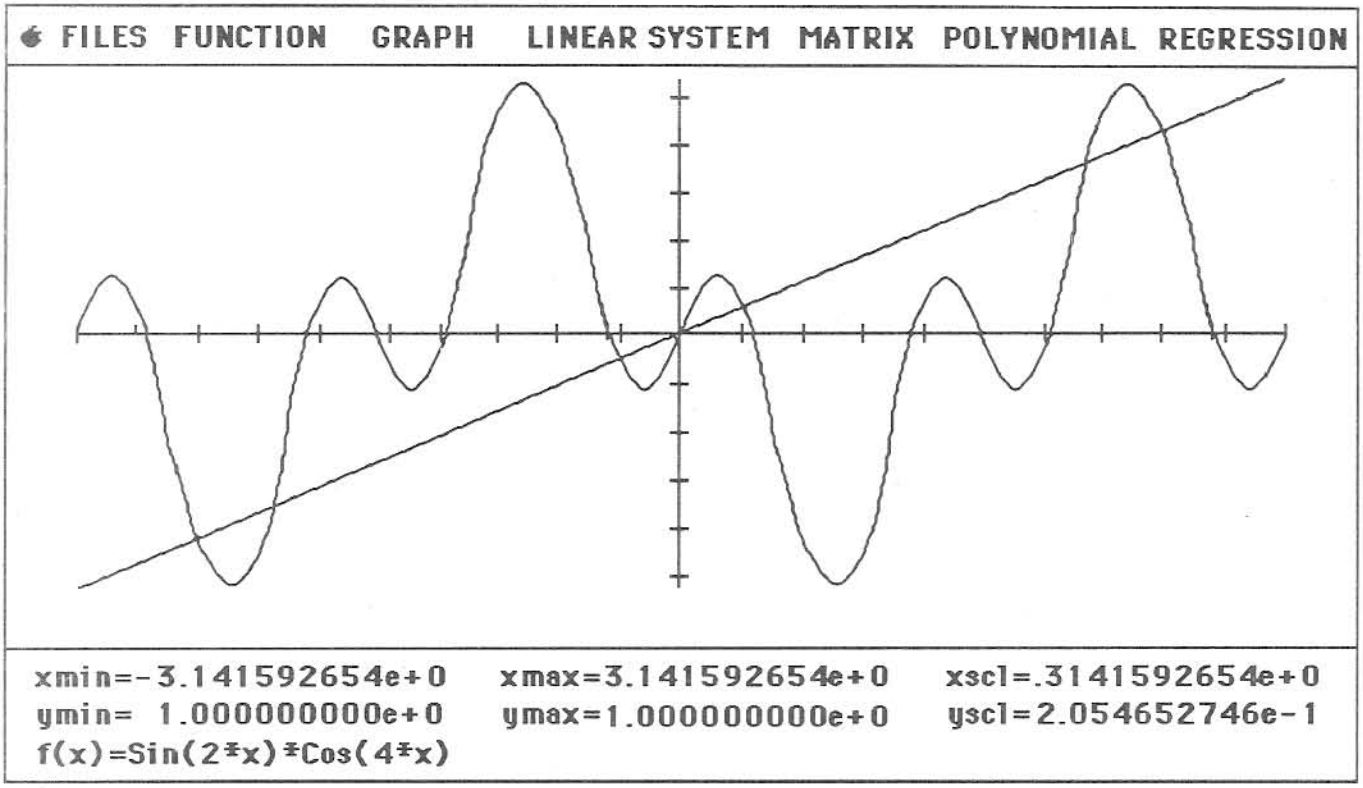

Figure 9.8 Graph Overlay

When the graph is drawn the function f(x) = sin(2*x)*cos(4*x) will be drawn in red. This indicates the active function. The polynomial f(x) = **.3183098862x** will be drawn in black, indicating it is the overlay function.

You may magnify the graph, find a y value for a given x, find the area between two points on the active function, find a root of the active function or find the slope of the active function by using the methods explained earlier in this chapter. The menu functions will always find the values for the active function. Remember, the active function is drawn in red.

You may change the active function by selecting the **Polynomial A Item** under the **GRAPH MENU**. When you do this the polynomial **f(x) = .3183098862x** will be redrawn in red and the function **f(x) = sin(2*x)*cos(4*x)** will be redrawn in black. The label in the **Graph Area** will change from **f(x) = sin(2*x)*cos(4*x)** to **f(x) = .3183098862x** showing the new active function. The **Polynomial A Item** under the **GRAPH MENU** will also change to **Function A**, indicating you may now select **Function A** as the active function. You may return to **f(x) = sin(2*x)*cos(4*x)** as the active function by selecting **Function A**.

Find the common solutions of Functions Graphically

You may use the graph overlays to find the common solution (intersection) of a function and the overlay function, by magnifying the graph in the area around their points of intersection. As an example, will find the common solution of **f(x) = sin(2*x)*cos(4*x)** and **f(x) = .3183098862x** at their intersection near the point **x = 2**. Make **f(x) = sin(2*x)*cos(4*x)** the active function by selecting **Function A** from the **GRAPH MENU**. Then select a point on each side of the intersection of the two functions. After selecting the points, click in the **Graph Area** to redraw the graph. This is the same procedure used to isolate function roots, explained earlier in this chapter. After the graph is redrawn around the area where the functions intersect, you may magnify the graph again, by selecting a point on each side of the graph and clicking in the **Graph Area** to redraw the graph. You may continue this procedure until the intersection point has been isolated as close as you desire. After a few magnifications, you will find that the common solution to the two functions, near **x = 2** is very close to the following:

x = 2.172155084
y = 0.6914184376

You may pass one or both of these solutions to the **x-reg** by using the **Stack Item** under the **GRAPH MENU**. After leaving the graph, store the value of x for the common solutions in the **a** memory location and solve **f(x) = sin(2*x)*cos(4*x)** for **a** using the **Solve Function Dialog** and solve the function **f(x) = .3183098862x** for **a** using the **Solve Polynomial Dialog**. You will confirm that both functions have the same solution : **y = 0.6914184376** at this value of x.

Setting Graph Colors

You may set the screen colors of the graphs to suit your own taste. You do this by selecting the **Set Graph Colors** item under the menu. This will bring up the **Set Graph Color** dialog.

This dialog contains three horizontal scroll bars, labelled "**RED Level=**", "**GREEN Level=**" and "**BLUE Level=**". These color labels show the intensity level, (hexidecimal 0 to hexidecimal 15) for the red, green and blue colors. You may select from 4096 color combinations, by mixing the three numbers. The dialog also contains two radio buttons labelled **Color 1** and **Color 2**. These represent the two colors the user can set. **Color 1** controls the color computed graphs and the superimposed, computed regression curves. **Color 2** controls the color of the x-y data points, graph selection ranges and the undefined graph ranges.

There are three buttons on the dialog [Color], [Mono] and [OK]. Clicking on the [Color] button will reset the graph colors to the standard color screen of light blue and red. Clicking on the [Mono] button will reset the graph colors to a monochrome (black and white) mode of grey levels, for use with a monochrome monitor. When you have selected the desired colors, clicking on the [OK] button will return you to the **Scientific Calculator**.

To set a graph color, click on the radio button controlling the color you wish to set. After you do this, the color that will be changed will blink for a short time, showing which colors you have selected. You can then "mix" your own color for the selection by using the scroll bars to add or subtract different levels of red, green and blue. You will see the colors on the small graphs change, as you change the red green and blue levels. After you have set the colors, click on the [OK] button. The program will save your colors to a disk file named "**GSCOL**". This file will be read when you restart **GSNumerics** and your personal graph colors will be automatically set for you.