E E S

Engineering Equation Solver

for Microsoft Windows
Operating Systems

Commercial and Professional Versions

F-Chart Software
4406 Fox Bluff Rd
Middleton, WI 53562
Phone: 608-836-8531
FAX: 608-836-8536
http://www.fchart.com

EES was compiled with DELPHI 5 by Borland

Registration Number___________________________

ALL CORRESPONDENCE MUST INCLUDE THE REGISTRATION NUMBER

V6.02

E E S

Engineering Equation Solver

for Microsoft Windows
Operating Systems

F-Chart Software
4406 Fox Bluff Rd
Middleton, WI 53562
Phone: 608-836-8531
FAX: 608-836-8536
http://www.fchart.com

Table of Contents

Overview

EES (pronounced 'ease') is an acronym for Engineering Equation Solver. The basic function provided by EES is the solution of a set of algebraic equations. EES can also solve differential equations, equations with complex variables, do optimization, provide linear and non-linear regression and generate publication-quality plots. Versions of EES have been developed for Apple Macintosh computers and for the Windows operating systems. This manual describes the version of EES developed for Microsoft Windows operating systems, including Windows 95/98/2000 and Windows NT 4.

There are two major differences between EES and existing numerical equation-solving programs. First, EES automatically identifies and groups equations which must be solved simultaneously. This feature simplifies the process for the user and ensures that the solver will always operate at optimum efficiency. Second, EES provides many built-in mathematical and thermophysical property functions useful for engineering calculations. For example, the steam tables are implemented such that any thermodynamic property can be obtained from a built-in function call in terms of any two other properties. Similar capability is provided for most organic refrigerants (including some of the new blends), ammonia, methane, carbon dioxide and many other fluids. Air tables are built-in, as are psychrometric functions and JANAF table data for many common gases. Transport properties are also provided for most of these substances.

The library of mathematical and thermophysical property functions in EES is extensive, but it is not possible to anticipate every user's need. EES allows the user to enter his or her own functional relationships in three ways. First, a facility for entering and interpolating tabular data is provided so that tabular data can be directly used in the solution of the equation set. Second, the EES language supports user-written functions and procedure similar to those in Pascal and FORTRAN. EES also provides support for user-written modules, which are self-contained EES programs that can be accessed by other EES programs. The functions, procedures, and modules can be saved as library files which are automatically read in when EES is started. Third, compiled functions and procedures, written in a high-level language such as Pascal, C or FORTRAN, can be dynamically-linked into EES using the dynamic link library capability incorporated into the Windows operating system. These three methods of adding functional relationships provide very powerful means of extending the capabilities of EES.

The motivation for EES rose out of experience in teaching mechanical engineering thermodynamics and heat transfer. To learn the material in these courses, it is necessary for the student to work problems. However, much of the time and effort required to solve problems results from looking up property information and solving the appropriate equations. Once the student is familiar with the use of property tables, further use of the tables does not contribute to the student's grasp of the subject; nor does algebra. The time and effort required to do problems in the conventional manner may actually detract from learning of the subject matter by forcing the student to be concerned with the order in which the equations should be solved (which really does not matter) and by making parametric studies too laborious. Interesting practical problems that may have implicit solutions, such as those involving both thermodynamic and heat transfer considerations, are often not assigned because of their mathematical complexity. EES allows the user to concentrate more on design by freeing him or her from mundane chores.

EES is particularly useful for design problems in which the effects of one or more parameters need to be determined. The program provides this capability with its Parametric Table, which is similar to a spreadsheet. The user identifies the variables that are independent by entering their values in the table cells. EES will calculate the values of the dependent variables in the table. The relationship of the variables in the table can then be displayed in publication-quality plots. EES also provides capability to propagate the uncertainty of experimental data to provide uncertainty estimates of calculated variables. With EES, it is no more difficult to do design problems than it is to solve a problem for a fixed set of independent variables.

EES offers the advantages of a simple set of intuitive commands that a novice can quickly learn to use for solving any algebraic problems. However, the capabilities of this program are extensive and useful to an expert as well. The large data bank of thermodynamic and transport properties built into EES is helpful in solving problems in thermodynamics, fluid mechanics, and heat transfer. EES can be used for many engineering applications; it is ideally suited for instruction in mechanical engineering courses and for the practicing engineer faced with the need for solving practical problems.

The remainder of this manual is organized into seven chapters and five appendices. A new user should read Chapter 1 which illustrates the solution of a simple problem from start to finish. Chapter 2 provides specific information on the various functions and controls in each of the EES windows. Chapter 3 is a reference section that provides detailed information for each menu command. Chapter 4 describes the built-in mathematical and thermophysical property functions and the use of the Lookup Table for entering tabular data. Chapter 5 provides instructions for writing EES functions, procedures and modules and saving them in

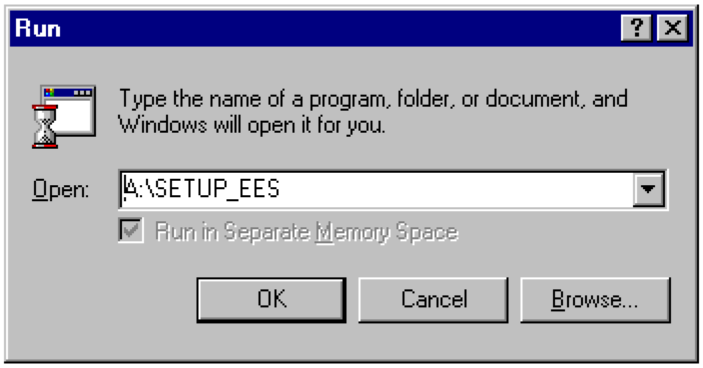
Library files. Chapter 6 describes how compiled functions and procedures, written as Windows dynamic-link library (DLL) routines, can be integrated with EES. Chapter 7 describes a number of advanced features in EES such as the use of string, complex and array variables, the solution of simultaneous differential and algebraic equations, and property plots. Appendix A contains a short list of suggestions. Appendix B describes the numerical methods used by EES. Appendix C shows how additional property data may be incorporated into EES. A number of example problems are provided in the Examples subdirectory included with EES. Appendix D indicates which features are illustrated in the example problems provided with EES.

C H A P T E R 1

Getting Started

Installing EES on your Computer

EES is distributed in a self-installing compressed form in a file called SETUP_EES.exe which may be provided on two floppy disks or on a CD. To install EES, it is necessary execute the SETUP_EES installation program. If you are installing EES from a CD, the installation program will start automatically when the CD is placed in the drive. To install EES from a floppy disk, place the first disk in the drive and select the Run command from the Start menu and then enter A:\SETUP_EES.exe.


Run
Type the name of a program, folder, or document, and Windows will open it for you.
Open:
A:\SETUP_EES
Run in Separate Memory Space
OK
Cancel
Browse...

Here A: is your floppy drive designation. In either case, the installation program will provide a series of prompts which will lead you through the complete installation of the EES program.

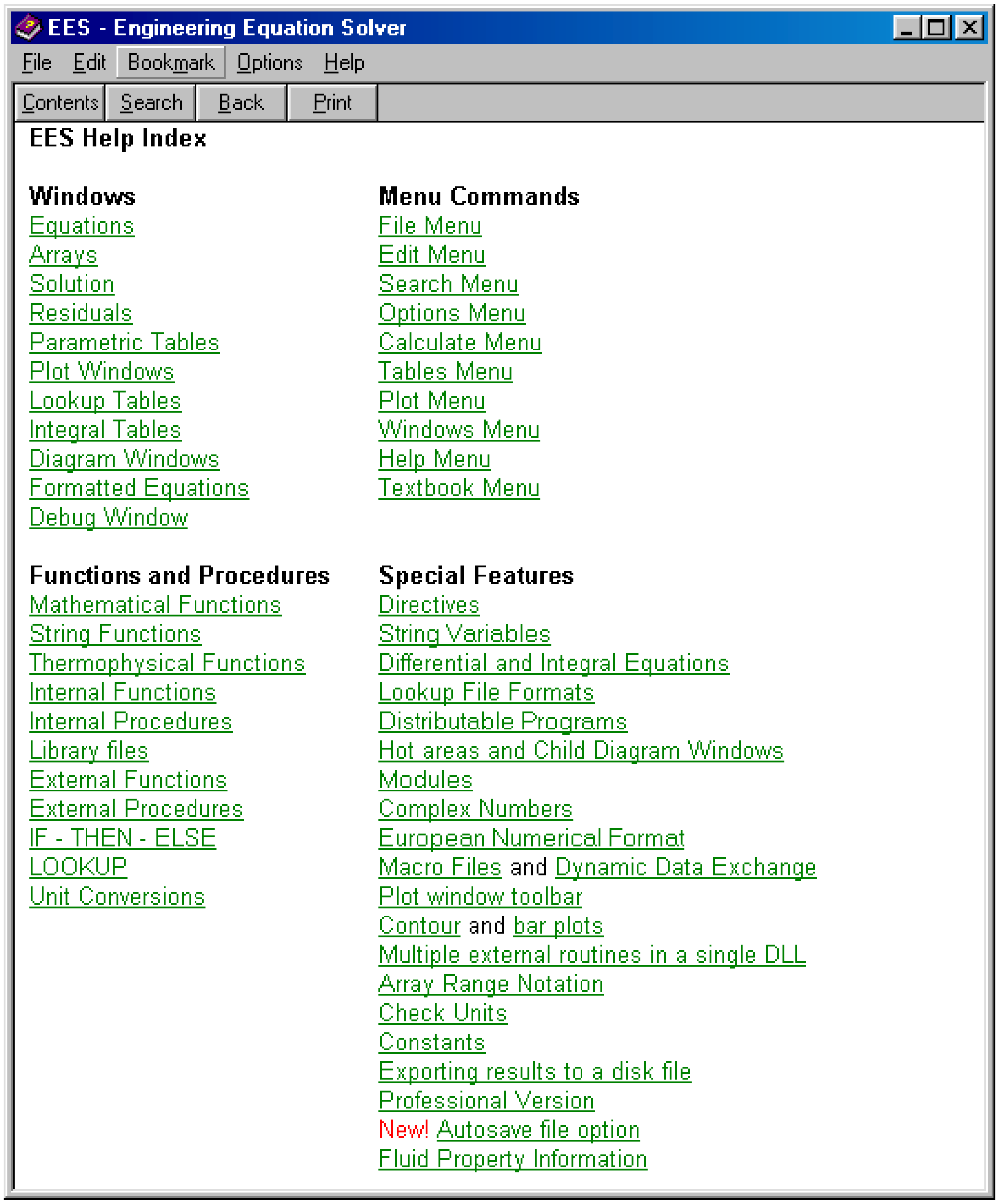
Starting EES

The default installation program will create a directory named C:\EES32 in which the EES files are placed. The EES program icon shown above will identify both the program and EES files. Double-clicking the left mouse button on the EES program or file icon will start the program. If you double-clicked on an EES file, that file will be automatically loaded. Otherwise, EES will load the HELLO.EES file which briefly describes the new features in your version. You can delete or rename the HELLO.EES file if you do not wish to have it appear when the program is started.

Background Information

EES begins by displaying a dialog window that shows registration information, the version number and other information. The version number and registration information will be needed if you request technical support. Click the OK button to dismiss the dialog window.

Detailed help is available at any point in EES. Pressing the F1 key will bring up a Help window relating to the foremost window. Clicking the Contents button will present the Help index shown below. Clicking on an underlined word (shown in green on color monitors) will provide help relating to that subject.


EES - Engineering Equation Solver
File Edit Bookmark Options Help
Contents Search Back Print

EES Help Index

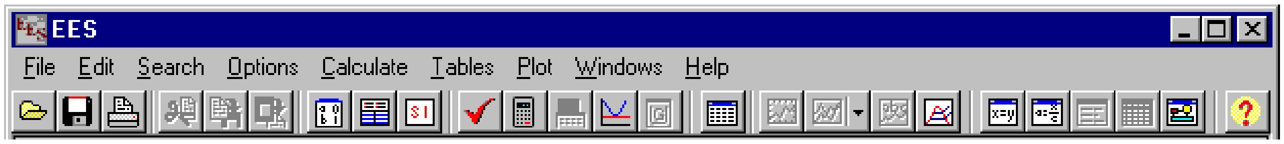
Windows
Equations
Arrays
Solution
Residuals
Parametric Tables
Plot Windows
Lookup Tables
Integral Tables
Diagram Windows
Formatted Equations
Debug Window

Menu Commands
File Menu
Edit Menu
Search Menu
Options Menu
Calculate Menu
Tables Menu
Plot Menu
Windows Menu
Help Menu
Textbook Menu

Functions and Procedures
Mathematical Functions
String Functions
Thermophysical Functions
Internal Functions
Internal Procedures
Library files
External Functions
External Procedures
IF - THEN - ELSE
LOOKUP
Unit Conversions

Special Features
Directives
String Variables
Differential and Integral Equations
Lookup File Formats
Distributable Programs
Hot areas and Child Diagram Windows
Modules
Complex Numbers
European Numerical Format
Macro Files and Dynamic Data Exchange
Plot window toolbar
Contour and bar plots
Multiple external routines in a single DLL
Array Range Notation
Check Units
Constants
Exporting results to a disk file
Professional Version
New! Autosave file option
Fluid Property Information

EES commands are distributed among nine pull-down menus. (A tenth user-defined menu can be placed to the right of the Help menu. See the discussion of the Load Textbook command File menu in Chapter 3.) A brief summary of their functions follows. Detailed descriptions of the commands appear in Chapter 3.



Note the a toolbar is provided below the menu bar. The toolbar contains small buttons which provide rapid access to many of the most frequently used EES menu commands. If you move the cursor over a button and wait for a few second, a few words will appear to explain the function of that button. The toolbar can be hidden, if you wish, with a control in the Preferences dialog (Options menu).

The System menu represented by the EES icon appears above the file menu. The System menu is not part of EES, but rather a feature of the Windows Operating System. It holds commands that allow window moving, resizing, and switching to other applications.

The File menu provides commands for loading, merging and saving work files and libraries, and printing.

The Edit menu provides the editing commands to cut, copy, and paste information.

The Search menu provides Find and Replace commands for use in the Equations window.

The Options menu provides commands for setting the guess values and bounds of variables, the unit system, default information, and program preferences. A command is also provided for displaying information on built-in and user-supplied functions.

The Calculate menu contains the commands to check, format and solve the equation set.

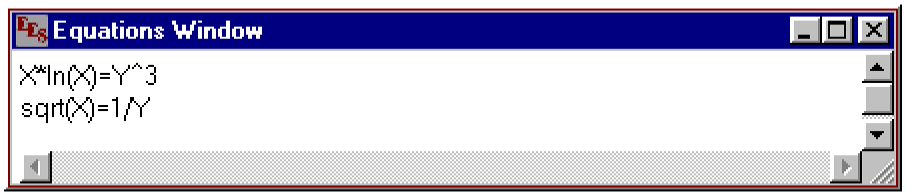
The Tables menu contains commands to set up and alter the contents of the Parametric and Lookup Tables and to do linear regression on the data in these tables. The Parametric Table, similar to a spreadsheet, allows the equation set to be solved repeatedly while varying the values of one or more variables. The Lookup table holds user-supplied data which can be interpolated and used in the solution of the equation set.

The Plot menu provides commands to modify an existing plot or prepare a new plot of data in the Parametric, Lookup, or Array tables. Curve-fitting capability is also provided.

The Windows menu provides a convenient method of bringing any of the EES windows to the front or to organize the windows.

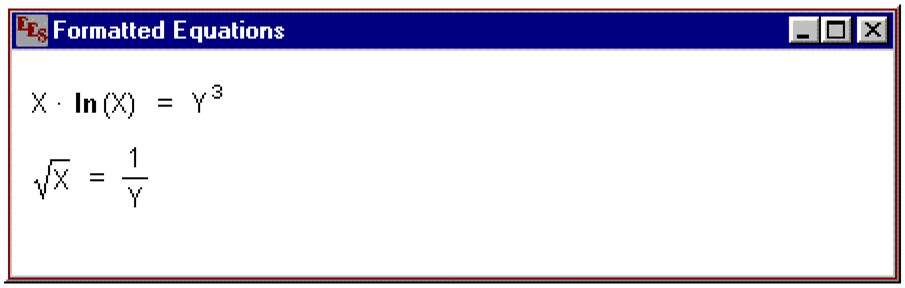
The Help menu provides commands for accessing the online help documentation.

The basic capability provided by EES is the solution of a set of non-linear algebraic equations. To demonstrate this capability, start EES and enter this simple example problem in the Equations window. Note that EES makes no distinction between upper and lower case letters and the ^ sign (or **) is used to signify raising to a power.



```
X*ln(X)=Y^3
sqrt(X)=1/Y
```

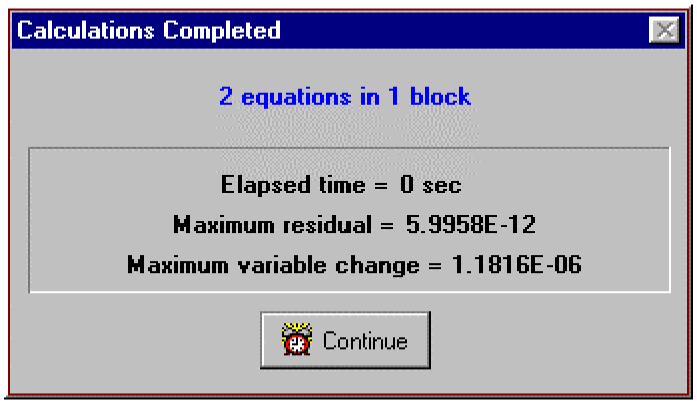
If you wish, you may view the equations in mathematical notation by selecting the Formatted Equations command from the Windows menu.



$$X \cdot \ln(X) = Y^3$$

$$\sqrt{X} = \frac{1}{Y}$$

Select the Solve command from the Calculate menu. A dialog window will appear indicating the progress of the solution. When the calculations are completed, the button changes from Abort to Continue.



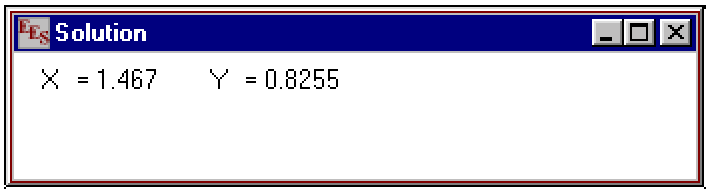
2 equations in 1 block

Elapsed time = 0 sec

Maximum residual = 5.9958E-12

Maximum variable change = 1.1816E-06

Click the Continue button. The solution to this equation set will then be displayed.

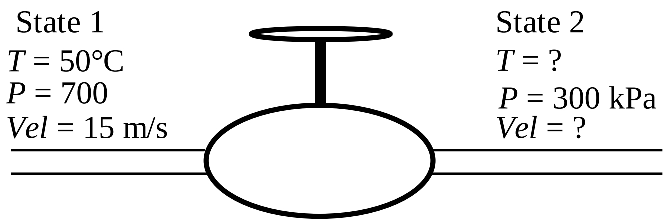


```
X = 1.467     Y = 0.8255
```

An Example Thermodynamics Problem

A simple thermodynamics problem will be set up and solved in this section to illustrate the property function access and equation solving capability of EES. The problem, typical of that which may be encountered in an undergraduate thermodynamics course, is as follows.

Refrigerant-134a enters a valve at 700 kPa, 50°C with a velocity of 15 m/s. At the exit of the valve, the pressure is 300 kPa. The inlet and outlet fluid areas are both 0.0110 m^2. Determine the temperature, mass flow rate and velocity at the valve exit.



To solve this problem, it is necessary to choose a system and then apply mass and energy balances. The system is the valve. The mass flow is steady, so that the mass balance is:

$$\dot{m}_1 = \dot{m}_2 \tag{1}$$

where

$$\dot{m}_1 = A_1\ Vel_1 / v_1 \tag{2}$$

$$\dot{m}_1 = A_2\ Vel_2 / v_2 \tag{3}$$

$\dot{m}$ = mass flowrate [kg/s]
A = cross-sectional area [m^2]
Vel = velocity [m/s]
v = specific volume [m^3/kg]

We know that

$$A_1 = A_2 \tag{4}$$

The valve is assumed to be well-insulated with no moving parts. The heat and work effects are both zero. A steady-state energy balance on the valve is:

$$\dot{\mathrm{m}}_1\left(\mathrm{h}_1+\frac{\mathrm{Vel}_1^2}{2}\right)=\dot{\mathrm{m}}_2\left(\mathrm{h}_2+\frac{\mathrm{Vel}_2^2}{2}\right) \tag{5}$$

where h is the specific enthalpy and $Vel^2/2$ is the specific kinetic energy. In SI units, specific enthalpy normally has units of [kJ/kg] so some units conversions may be needed. EES provides unit conversion capabilities with the CONVERT function as documented in

Chapter 4. In addition, the Check Units command (Calculate menu) can be applied to determine check that all unit conversions have been made and the units in each equation are dimensionally consistent.

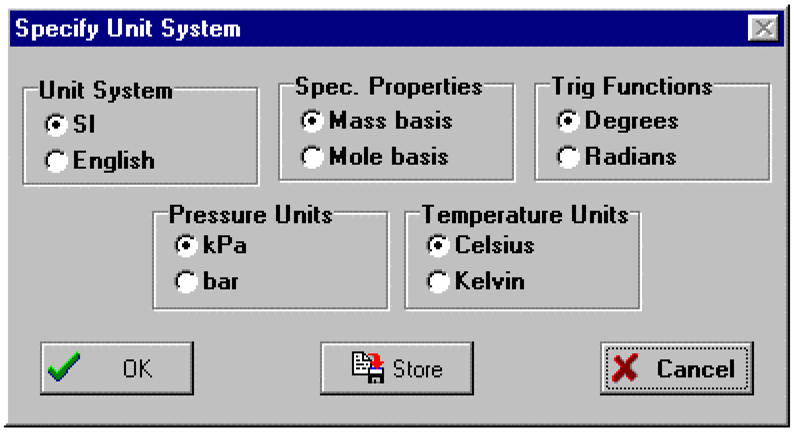
From relationships between the properties of R134a:

$$v_1 = v(T_1, P_1) \tag{6}$$
$$h_1 = h(T_1, P_1) \tag{7}$$
$$v_2 = v(T_2, P_2) \tag{8}$$
$$h_2 = h(T_2, P_2) \tag{9}$$

Ordinarily, the terms containing velocity are neglected, primarily because the kinetic energy effects are usually small and also because these terms make the problem difficult to solve. However, with EES, the computational difficulty is not a factor. The user can solve the problem with the kinetic energy terms and judge their importance.

The values of T_1, P_1, $A_{1,}$ Vel_{11} and P_2 are known. There are nine unknowns: A_2, $\dot{m}_1$, $\dot{m}_2$, Vel_2, h_1, v_1, h_2, v_2, T_2. Since there are 9 equations, the solution to the problem is defined. It is now only necessary to solve the equations. This is where EES can help.

Start EES and select the New command from the File menu. A blank Equations window will appear. Before entering the equations, however, set the unit system for the built-in thermophysical properties functions. To view or change the unit system, select Unit System from the Options menu.



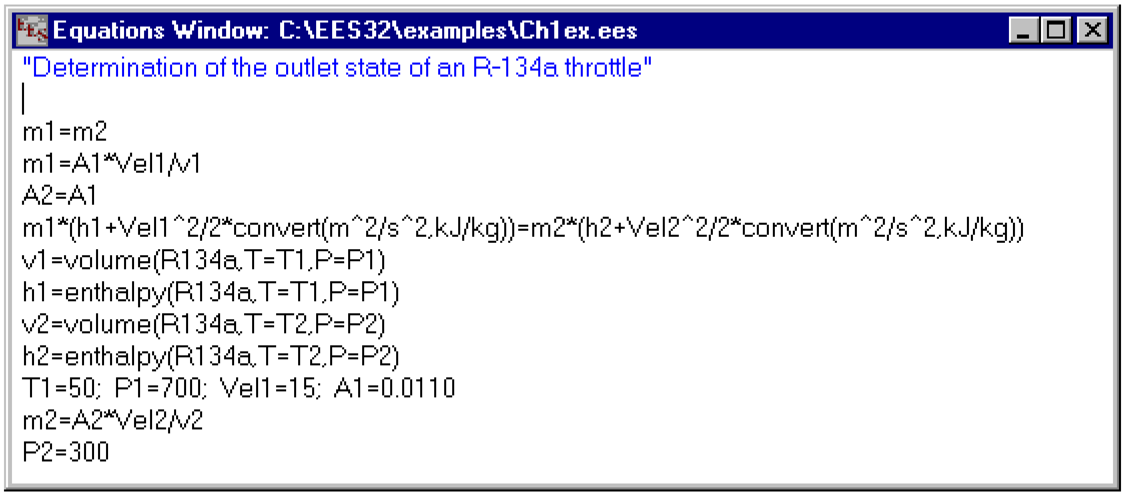
Unit System: [✓ SI] English
Spec. Properties: [✓ Mass basis] Mole basis
Trig Functions: [✓ Degrees] Radians
Pressure Units: [✓ kPa] bar
Temperature Units: [✓ Celsius] Kelvin

EES is initially configured to be in SI units with T in °C, P in kPa, and specific property values in their customary units on a mass basis. These defaults may have been changed during a previous use. Click on the controls to set the units as shown above. Click the OK button (or press the Return key) to accept the unit system settings.

The equations can now be entered into the Equations window. Text is entered in the same manner as for any word processor. Formatting rules are as follows:

1. Upper and lower case letters are not distinguished. EES will (optionally) change the case of all variables to match the manner in which they first appear.
2. Blank lines and spaces may be entered as desired since they are ignored.
3. Comments must be enclosed within braces { } or within quote marks " ". Comments may span as many lines as needed. Comments within braces may be nested in which case only the outermost set of { } are recognized. Comments within quotes will also be displayed in the Formatted Equations window.
4. Variable names must start with a letter and consist of any keyboard characters except ( ) ' | * / + - ^ { } : " or ;. Array variables (Chapter 7) are identified with square braces around the array index or indices, e.g., X[5,3]. String variables (Chapter 7) are identified with a $ as the last character in the variable name. The maximum length of a variable name is 30 characters.
5. Multiple equations may be entered on one line if they are separated by a semi-colon (;)[1]. The maximum line length is 255 characters.
6. The caret symbol ^ or ** is used to indicate raising to a power.
7. The order in which the equations are entered does not matter.
8. The position of knowns and unknowns in the equation does not matter.

After entering the equations for this problem and (optionally) checking the syntax using the Check/Format command in the Calculate menu, the Equations window will appear as shown. Comments are normally displayed in blue on a color monitor. Other formatting options are set with the Preferences command in the Options menu.



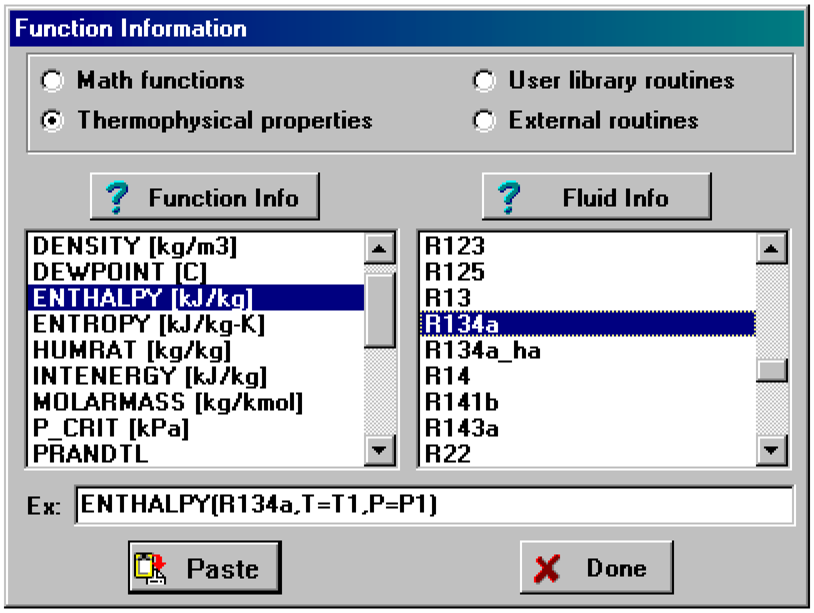
```
"Determination of the outlet state of an R-134a throttle"

m1=m2
m1=A1*Vel1/v1
A2=A1
m1*(h1+Vel1^2/2*convert(m^2/s^2,kJ/kg))=m2*(h2+Vel2^2/2*convert(m^2/s^2,kJ/kg))
v1=volume(R134a,T=T1,P=P1)
h1=enthalpy(R134a,T=T1,P=P1)
v2=volume(R134a,T=T2,P=P2)
h2=enthalpy(R134a,T=T2,P=P2)
T1=50; P1=700; Vel1=15; A1=0.0110
m2=A2*Vel2/v2
P2=300
```

[1] If a comma is selected as the Decimal Symbol in the Windows Regional Settings Control Panel, EES will recognize the comma (rather than a decimal point) as a decimal separator, the semicolon (rather than the comma) as an argument separator, and the colon : (rather than the semicolon) as the equation separator.

Note the use of the **Convert** function in this example to convert the units of the specific kinetic energy [m^2/s^2] to the units used for specific enthalpy [kJ/kg]. The **Convert** function is most useful in these problems. See Chapter 4 for a detailed description of its use.

The thermodynamic property functions, such as **enthalpy** and **volume** require a special format. The first argument of the function is the substance name, R134a in this case. The following arguments are the independent variables preceded by a single identifying letter and an equal sign. Allowable letters are T, P, H, U, S, V, and X, corresponding to temperature, pressure, specific enthalpy, specific internal energy, specific entropy, specific volume, and quality. (For psychrometric functions, additional allowable letters are W, R, D, and B, corresponding to humidity ratio, relative humidity, dewpoint temperature, and wetbulb temperature.)

An easy way to enter functions, without needing to recall the format, is to use the Function Information command in the Options menu. This command will bring up the dialog window shown below. Click on the 'Thermophysical properties' radio button. The list of built-in thermophysical property function will appear on the left with the list of substances on the right. Select the property function by clicking on its name, using the scroll bar, if necessary, to bring it into view. Select a substance in the same manner. An example of the function showing the format will appear in the Example rectangle at the bottom. The information in the rectangle may be changed, if needed. Clicking the Paste button will copy the Example into the Equations window at the cursor position. Additional information is available by clicking the Function Info and Fluid Info buttons.



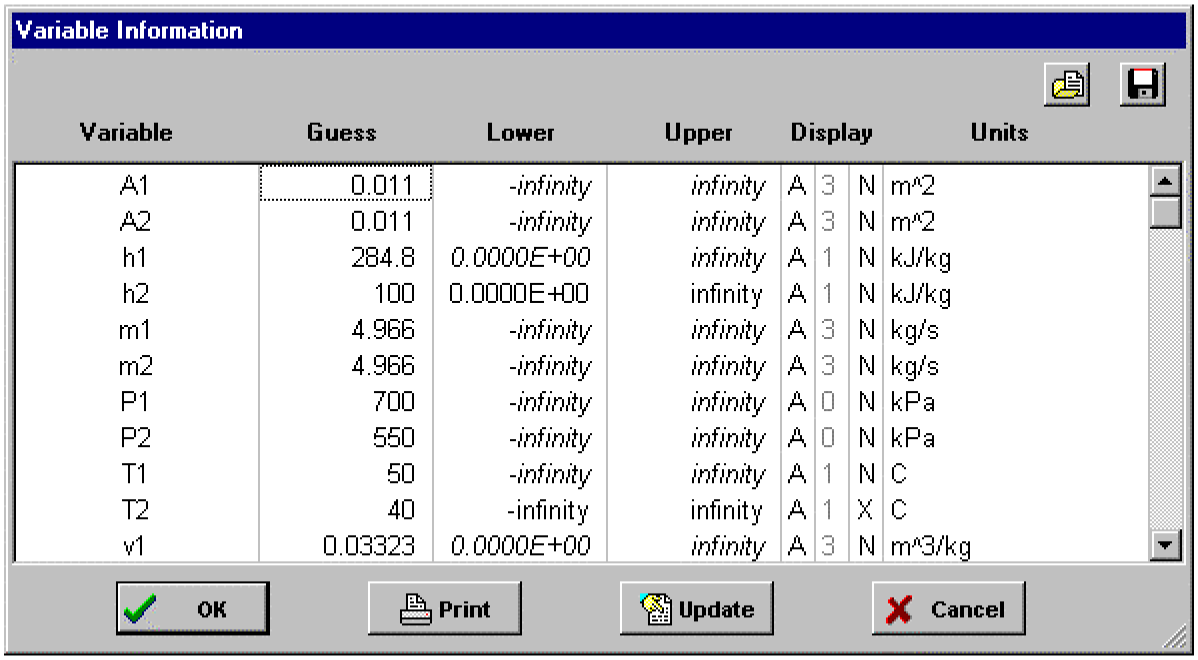
DENSITY [kg/m3]
DEWPOINT [C]
ENTHALPY [kJ/kg]
ENTROPY [kJ/kg-K]
HUMRAT [kg/kg]
INTENERGY [kJ/kg]
MOLARMASS [kg/kmol]
P_CRIT [kPa]
PRANDTL

R123
R125
R13
R134a
R134a_ha
R14
R141b
R143a
R22

Ex: ENTHALPY(R134a,T=T1,P=P1)

It is usually a good idea to set the guess values and (possibly) the lower and upper bounds for the variables before attempting to solve the equations. This is done with the Variable Information command in the Options menu. Before displaying the Variable Information dialog, EES checks syntax and compiles newly entered and/or changed equations, and then solves all equations with one unknown. The Variable Information dialog will then appear.

The Variable Information dialog contains a line for each variable appearing in the Equations window. By default, each variable has a guess value of 1.0 with lower and upper bounds of negative and positive infinity. (The lower and upper bounds are shown in italics if EES has previously calculated the value of the variable. In this case, the Guess value column displays the calculated value. These italicized values may still be edited, which will force EES to recalculate the value of that variable.)

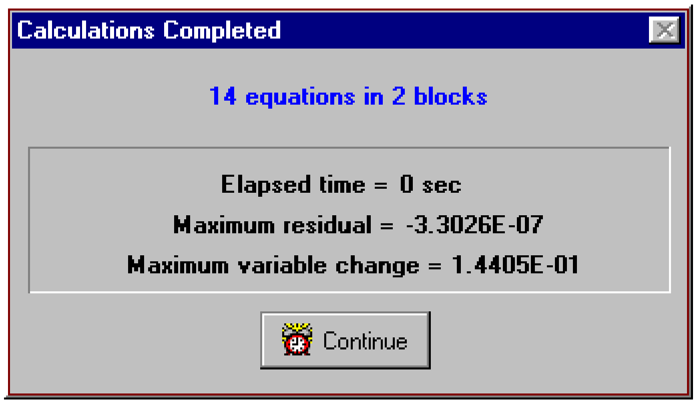


Variable	Guess	Lower	Upper	Display			Units
A1	0.011	*-infinity*	*infinity*	A	3	N	m^2
A2	0.011	*-infinity*	*infinity*	A	3	N	m^2
h1	284.8	*0.0000E+00*	*infinity*	A	1	N	kJ/kg
h2	100	0.0000E+00	infinity	A	1	N	kJ/kg
m1	4.966	*-infinity*	*infinity*	A	3	N	kg/s
m2	4.966	*-infinity*	*infinity*	A	3	N	kg/s
P1	700	*-infinity*	*infinity*	A	0	N	kPa
P2	550	*-infinity*	*infinity*	A	0	N	kPa
T1	50	*-infinity*	*infinity*	A	1	N	C
T2	40	-infinity	infinity	A	1	X	C
v1	0.03323	*0.0000E+00*	*infinity*	A	3	N	m^3/kg

The A in the Display options column indicates that EES will automatically determine the display format for numerical value of the variable when it is displayed in the Solution window. In this case, EES will select an appropriate number of digits, so the digits column to the right of the A is disabled. Automatic formatting is the default. Alternative display options are F (for fixed number of digits to the right of the decimal point) and E (for exponential format). The display and other defaults can easily be changed with the Default Information command in the Options menu, discussed in Chapter 3. The third Display options column controls the hilighting effects such as normal (default), bold, boxed. The units of the variables can be specified, if desired. The units will be displayed with the variable in the Solution window and/or in the Parametric Table. EES does not automatically do unit conversions but it can provide unit conversions using the **Convert** function (Chapter 4) and unit checking with the Check Units command in the Calculate menu. The units information entered here is only for display purposes.

With nonlinear equations, it is sometimes necessary to provide reasonable guess values and bounds in order to determine the desired solution. (It is not necessary for this problem.) The bounds of some variables are known from the physics of the problem. In the example problem, the enthalpy at the outlet, h2, should be reasonably close to the value of h1. Set its guess value to 100 and its lower bound to 0. Set the guess value of the outlet specific volume, v2, to 0.1 and its lower bound to 0. Scroll the variable information list to bring Vel2 into view. The lower bound of Vel2 should also be zero.

To solve the equation set, select the Solve command from the Calculate menu. An information dialog will appear indicating the elapsed time, maximum residual (i.e., the difference between the left-hand side and right-hand side of an equation) and the maximum change in the values of the variables since the last iteration. When the calculations are completed, EES displays the total number of equations in the problem and the number of blocks. A block is a subset of equations that can be solved independently. EES automatically blocks the equation set, whenever possible, to improve the calculation efficiency, as described in Appendix B. When the calculations are completed, the button will change from Abort to Continue.



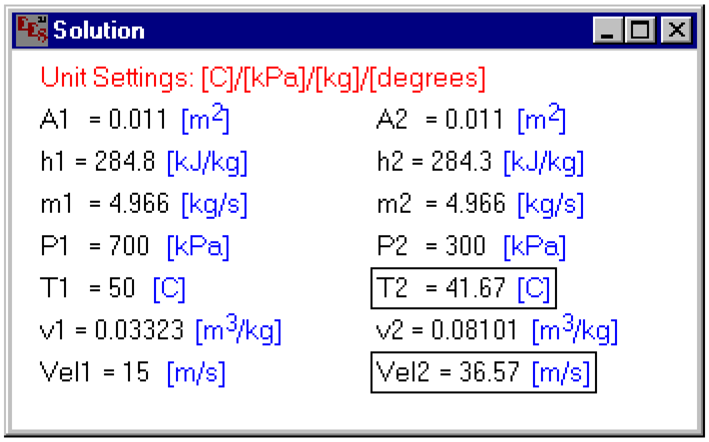
14 equations in 2 blocks

Elapsed time = 0 sec

Maximum residual = -3.3026E-07

Maximum variable change = 1.4405E-01

By default, the calculations are stopped when 100 iterations have occurred, the elapsed time exceeds 60 sec, the maximum residual is less than 10^{-6} or the maximum variable change is less than 10^{-9}. These defaults can be changed with the Stop Criteria command in the Options menu. If the maximum residual is larger than the value set for the stopping criteria, the equations were not correctly solved, possibly because the bounds on one or more variables constrained the solution. Clicking the Continue button will remove the information dialog and display the Solution window shown on the next page. The problem is now completed since the values of T2, m2, and Vel2 are determined.



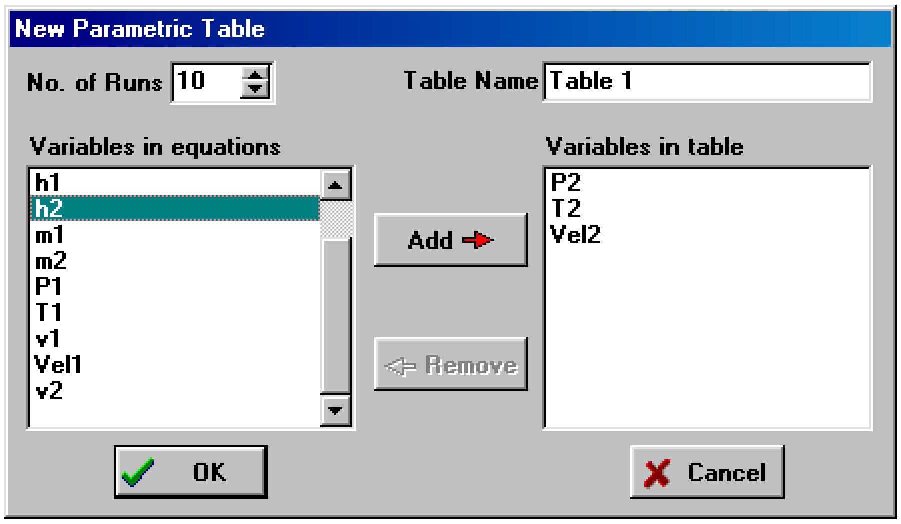
Solution

Unit Settings: [C]/[kPa]/[kg]/[degrees]

A1 = 0.011 [m^2]	A2 = 0.011 [m^2]
h1 = 284.8 [kJ/kg]	h2 = 284.3 [kJ/kg]
m1 = 4.966 [kg/s]	m2 = 4.966 [kg/s]
P1 = 700 [kPa]	P2 = 300 [kPa]
T1 = 50 [C]	T2 = 41.67 [C]
v1 = 0.03323 [m^3/kg]	v2 = 0.08101 [m^3/kg]
Vel1 = 15 [m/s]	Vel2 = 36.57 [m/s]

One of the most useful features of EES is its ability to provide parametric studies. For example, in this problem, it may be of interest to see how the throttle outlet temperature and outlet velocity vary with outlet pressure. A series of calculations can be automated and plotted using the commands in the Tables menu.

Select the New Table command. A dialog will be displayed listing the variables appearing in the Equations window. In this case, we will construct a table containing the variables P2, T2, Vel2, and h2. Click on P2 from the variable list on the left. This will cause P2 to be highlighted and the Add button will become active.



New Parametric Table

No. of Runs 10

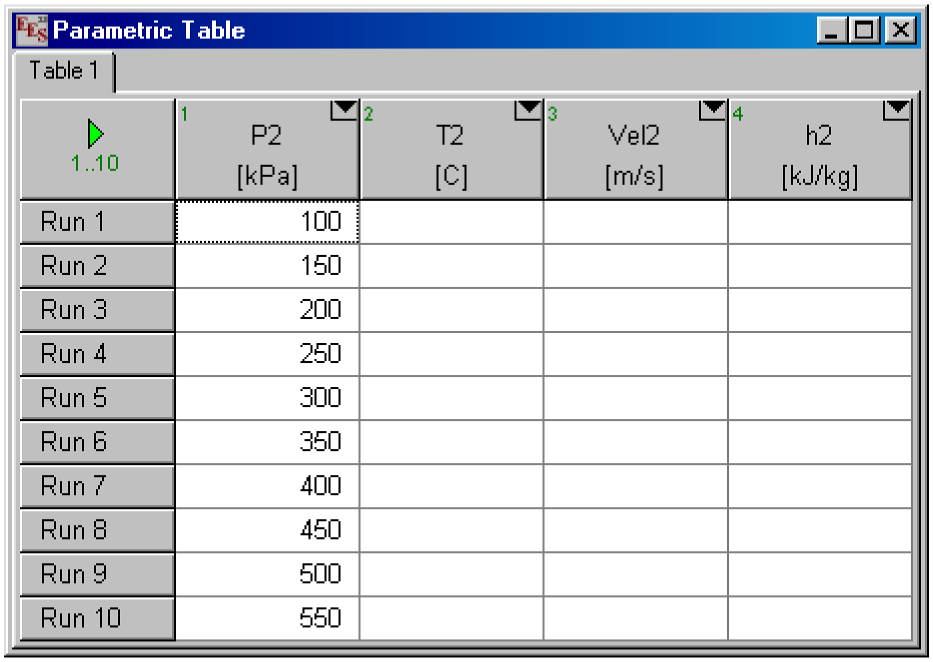
Table Name Table 1

Variables in equations: h1, h2, m1, m2, P1, T1, v1, Vel1, v2

Variables in table: P2, T2, Vel2

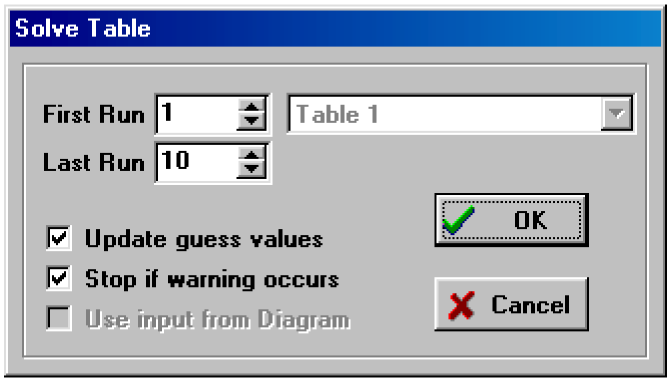
Now click the Add button to move P2 to the list of variables on the right. Repeat for T2, h2, and Vel2, using the scroll bar to bring the variable into view if necessary. (As a short cut, you can double-click on the variable name in the list on the left to move it to the list on the right.). The table setup dialog should now appear as shown above. The default table name is Table 1. This name can be changed at this point or later. Click the OK button to create the table.

The Parametric Table works much like a spreadsheet. You can type numbers directly into the cells. Numbers that you enter are shown in black and produce the same effect as if you set the variable to that value with an equation in the Equations window. Delete the P2 = 300 equation currently in the Equations window or enclose it in comment brackets { }. This equation will not be needed because the value of P2 will be set in the table. Now enter the values of P2 for which T2 is to be determined. Values of 100 to 550 have been chosen for this example. (The values could also be automatically entered using Alter Values in the Tables menu, by right clicking on the table cell that shows P2 and selecting Alter Values from the popup menu, or by using the Alter Values control ▼ at the upper right of each table column header, as explained in Chapter 2.) The Parametric Table should now appear as shown below.



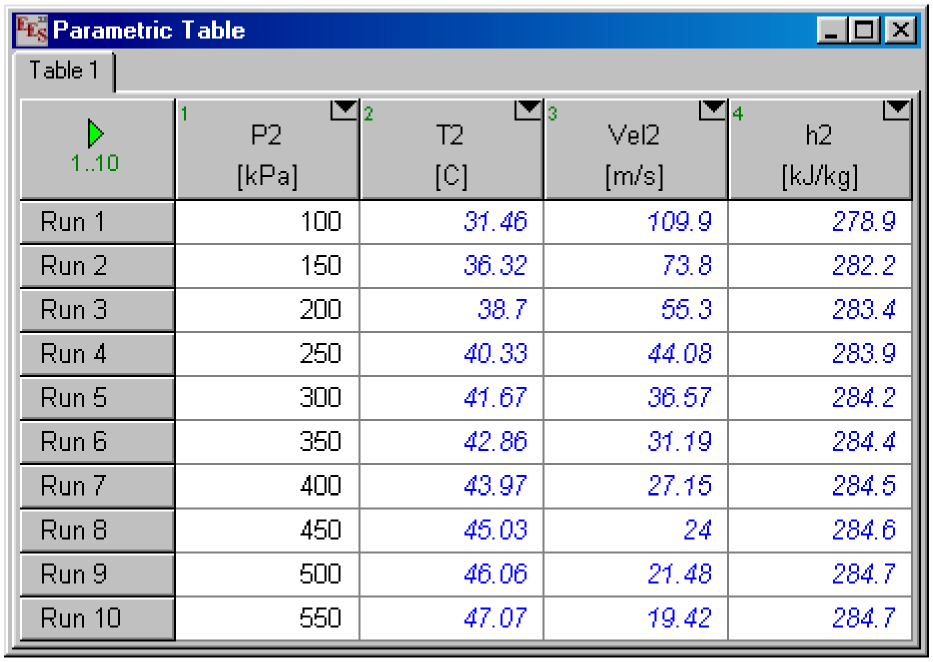
1..10	1 P2 [kPa]	2 T2 [C]	3 Vel2 [m/s]	4 h2 [kJ/kg]
Run 1	100			
Run 2	150			
Run 3	200			
Run 4	250			
Run 5	300			
Run 6	350			
Run 7	400			
Run 8	450			
Run 9	500			
Run 10	550			

Now, select Solve Table from the Calculate menu. The Solve Table dialog window will appear allowing you to choose the runs for which the calculations will be done. If more than one Parametric table was defined, a choice of tables would also be available.



First Run 1
Last Run 10
Table 1
☑ Update guess values
☑ Stop if warning occurs
☐ Use input from Diagram

When the Update Guess Values control is selected, as shown, the solution for the last run will provide guess values for the following run. Click the OK button. A status window will be displayed, indicating the progress of the solution. When the calculations are completed, the values of T2, Vel2, and h2 will be entered into the table. The values calculated by EES will be displayed in blue, bold or italic type depending on the setting made in the Screen Display tab of the Preferences dialog window in the Options menu.

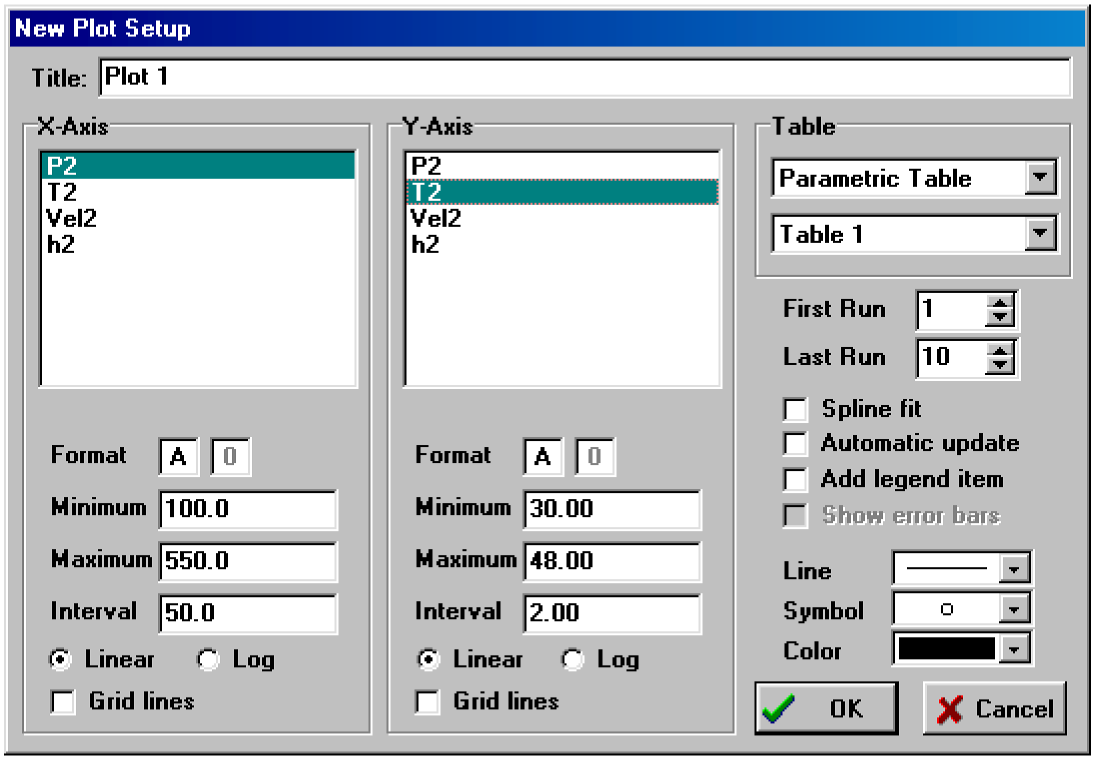


1..10	P2 [kPa]	T2 [C]	Vel2 [m/s]	h2 [kJ/kg]
Run 1	100	31.46	109.9	278.9
Run 2	150	36.32	73.8	282.2
Run 3	200	38.7	55.3	283.4
Run 4	250	40.33	44.08	283.9
Run 5	300	41.67	36.57	284.2
Run 6	350	42.86	31.19	284.4
Run 7	400	43.97	27.15	284.5
Run 8	450	45.03	24	284.6
Run 9	500	46.06	21.48	284.7
Run 10	550	47.07	19.42	284.7

The relationship between variables such as P2 and T2 is now apparent, but it can more clearly be seen with a plot. Select New Plot Window from the Plot menu. The New Plot Window dialog window shown below will appear. Choose P2 to be the x-axis by clicking on P2 in the x-axis list. Click on T2 in the y-axis list. Select the scale limits for P2 and T2, and set the number of divisions for the scale as shown. Grid lines make the plot easier to read. Click on the Grid Lines control for both the x and y axes. When you click the OK button, the plot will be constructed and the plot window will appear as shown below.

Once created, there are a variety of ways in which the appearance of the plot can be changed as described in the Plot Windows section of Chapter 2 and in the Plot menu section of Chapter 3.

This example problem illustrates some of the capabilities of EES. With this example behind you, you should be able to solve many types of problems. However, EES has many more capabilities and features, such as curve-fitting, uncertainty analyses, complex variables, arrays.


New Plot Setup

Title: Plot 1

X-Axis: [✓ P2] T2 Vel2 h2

Y-Axis: P2 [✓ T2] Vel2 h2

Table: Parametric Table; Table 1

First Run 1

Last Run 10

	X-Axis	Y-Axis
Format	A 0	A 0
Minimum	100.0	30.00
Maximum	550.0	48.00
Interval	50.0	2.00

[✓ Linear] Log

Grid lines

Spline fit

Automatic update

Add legend item

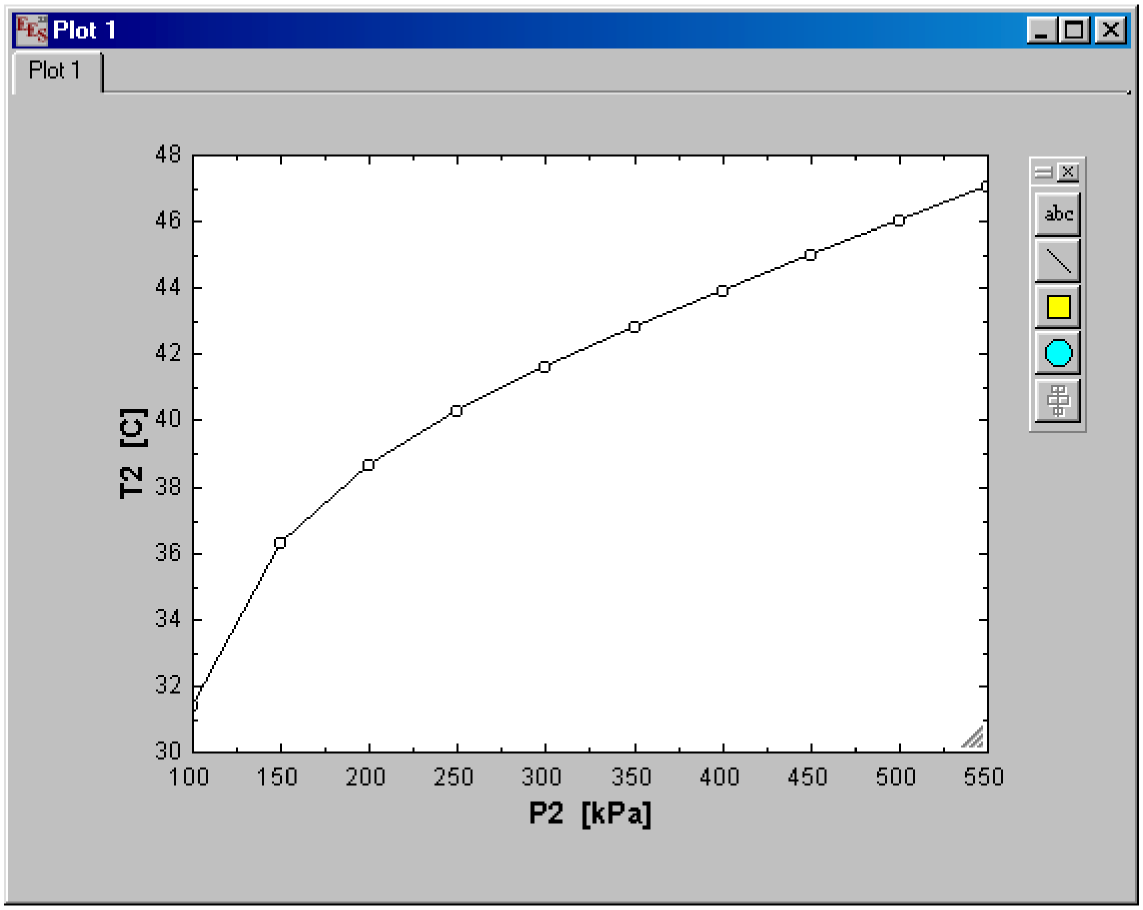
Show error bars

Line

Symbol

Color

OK Cancel

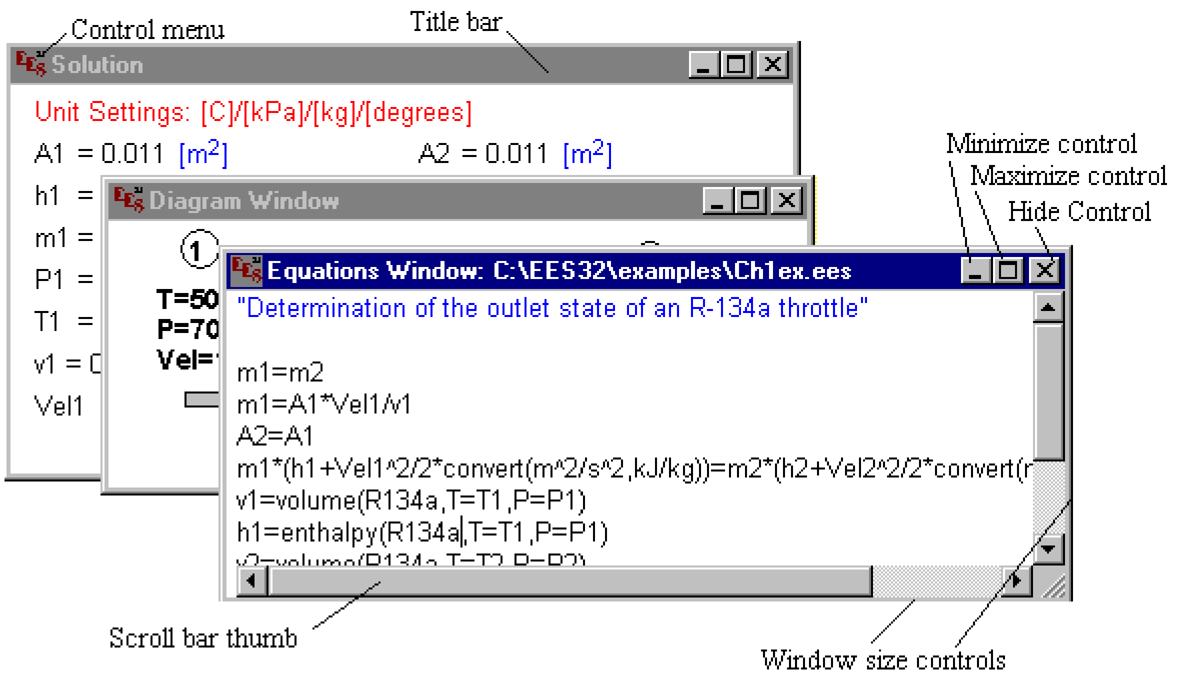


CHAPTER 2

EES Windows

General Information

The information concerning a problem is presented in a series of windows. Equations and comments are entered in the Equations window. After the equations are solved, the values of the variables are presented in the Solution and Arrays windows. The residuals of the equations and the calculation order may be viewed in the Residuals window. Additional windows are provided for the Parametric and Lookup Tables, a diagram and up to 10 plots. There is also a Debug window. A detailed explanation of the capabilities and information for each window type is provided in this section. All of the windows can be open (i.e., visible) at once. The window in front is the active window and it is identified by its highlighted (black) title bar. The figure below shows the appearance of the EES windows in Microsoft NT 4.0/2000. The appearance may be slightly different in other Windows versions.



```
Unit Settings: [C]/[kPa]/[kg]/[degrees]
A1 = 0.011 [m^2]          A2 = 0.011 [m^2]
```

```
"Determination of the outlet state of an R-134a throttle"

m1=m2
m1=A1*Vel1/v1
A2=A1
m1*(h1+Vel1^2/2*convert(m^2/s^2,kJ/kg))=m2*(h2+Vel2^2/2*convert(
v1=volume(R134a,T=T1,P=P1)
h1=enthalpy(R134a,T=T1,P=P1)
```

One difference between EES and most other applications is worth mentioning. The Close control merely hides a window; it does not delete it. Once closed, a window can be reopened (i.e., made visible) by selecting it from the Windows menu.

Every window has a number of controls.

1. To move the window to a different location on the screen, move the cursor to a position on the title bar of the window and then press and hold the left button down while sliding the mouse to a new location.

2. To hide the window, select the Close command (or press Ctrl-F4) from the control menu box at the upper left of the window title bar. A Hide control is also accessible at the upper right of the title bar. You can restore a hidden window by selecting it from the Windows menu. Window information is NOT lost when the window is closed.

3. The Maximize box at the upper right of the window title bar causes the window to be resized so as to fill the entire screen. The Restore box with an up and down arrow will appear below the Maximize box. Click the Restore box (or select Restore form the Control menu box) to return the window to its former size.

4. The size of any window can be adjusted using the window size controls at any border of the window. To change the size of any window, move the cursor to the window border. The cursor will change to a horizontal or vertical double arrow. Then press and hold the left button down while moving the mouse to make the window larger or smaller. Scroll bars will be provided if the window is made too small to accommodate all the information.

5. Double-clicking the left mouse button on the EES icon at the upper left of the title bar will hide that window.

6. Use the Cascade command in the Windows menu to move and resize all open windows.

Equations Window

The Equations window operates very much like a word processor. The equations that EES is to solve are entered in this window. Editing commands, i.e., Cut, Copy, Paste, are located in the Edit menu and can be applied in the usual manner. Clicking the right mouse button on selected text in the Equations window will also editing commands in a popup menu. Additional information relevant to the Equations window follows.

1. Blank lines may be used to make the Equations window more legible.
2. Comments are enclosed in braces {comment} or in quote marks "another comment" and may span multiple lines. Nested comment fields within braces are permitted. Comments within quote marks will appear in the Formatted Equations window. Comments that begin with and exclamation mark ! will appear in a different font or color as determined by the settings made with the Preferences command in the Options menu. EES equations must be less than 255 characters, comments may be of any length. The comments will automatically line break to fill the Equations window if the Wrap Long Lines option in the Preferences Equations tab is selected. The Formatted Equations window will also line break the comments in the display window and in the printed output.
3. Equations may be entered in any order. The order of the equations has no effect on the solution, since EES will block the equations and reorder them for efficient solution as described in Appendix B.
4. The order of mathematical operators used in the equations conform to the rules used in FORTRAN, Basic, C or Pascal. For example, the equation

   $$X = 3 + 4 * 5$$

   will result in X having a value of 23. The caret symbol ^ or ** can be used to indicate raising to a power. Arguments of functions are enclosed in parentheses. EES does not require a variable to appear by itself on the left-hand side of the equation, as does FORTRAN and most other programming languages. The above equation could have been entered as

   $$(X - 3) / 4 = 5$$

5. Upper and lower case letters are not distinguished. EES will (optionally) change the case of all variables to match the manner in which they first appear in the Equations window depending on the settings selected in Preferences dialog in the Options menu. However, this change is made only when an equation is first compiled or modified or when Check/Format command in the Calculate menu is issued.

6. Variable names must start with a letter and consist of any keyboard characters except (‘|)*/+-^{ } ":;. The maximum variable length is 30 characters. String variables hold character information and are identified with a $ as the last character in their names, as in BASIC. Array variables are identified with square braces around the array index or indices, e.g., X[5,3]. The quantity within the braces must be a number, except within the scope of the sum, product or Duplicate commands. As a general rule, variables should not be given names that correspond to those of built-in functions (e.g., **pi**, **sin**, **enthalpy**).

7. As you enter an equation, an unmatched open or close parenthesis will be displayed in bold font.

8. The commercial and educations versions of EES have an upper limit of 6000 variables. The Professional version can have 10,000 variables.

9. Equations are normally entered one per line, terminated by pressing the Return or Enter keys. Multiple equations may be entered on one line if they are separated by a semi-colon[2]. Long equations are accommodated by the provision of a horizontal scroll bar which appears if any of the equations is wider than the window. However, each equation must be less than 255 characters.

10. EES compiles equations into a compact stack-based form. The compiled form is saved in memory so that an equation needs to be compiled only when it is first used or when it is changed. Any error detected during the compilation or solution process will result in an explanatory error message and highlighting of the line in which the problem was discovered.

11. Equations can be imported or exported from/to other applications by using Cut, Copy and Paste commands in the Edit menu. The Merge and Load Library commands in the File menu and the $INCLUDE directive may also be used to import the equations from an existing file. The Merge command will import the equations from an EES or text file and place them in the Equations window at the cursor position. Equations imported with the $INCLUDE directive will not appear in the Equations window.

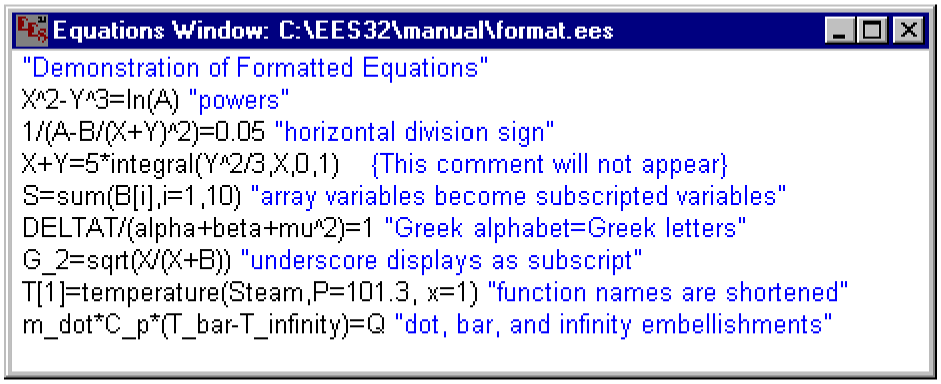
12. Clicking the right mouse button in the Equations window will bring up a pop-up menu that will allow commenting (or uncommenting), cutting, copying or printing of the selected text.

[2] If a comma is selected as the Decimal Symbol in the Windows Regional Settings Control Panel, EES will recognize the comma (rather than a decimal point) as a decimal separator, the semicolon (rather than the comma) as an argument separator, and the vertical bar | (rather than the semicolon) as the equation separator.

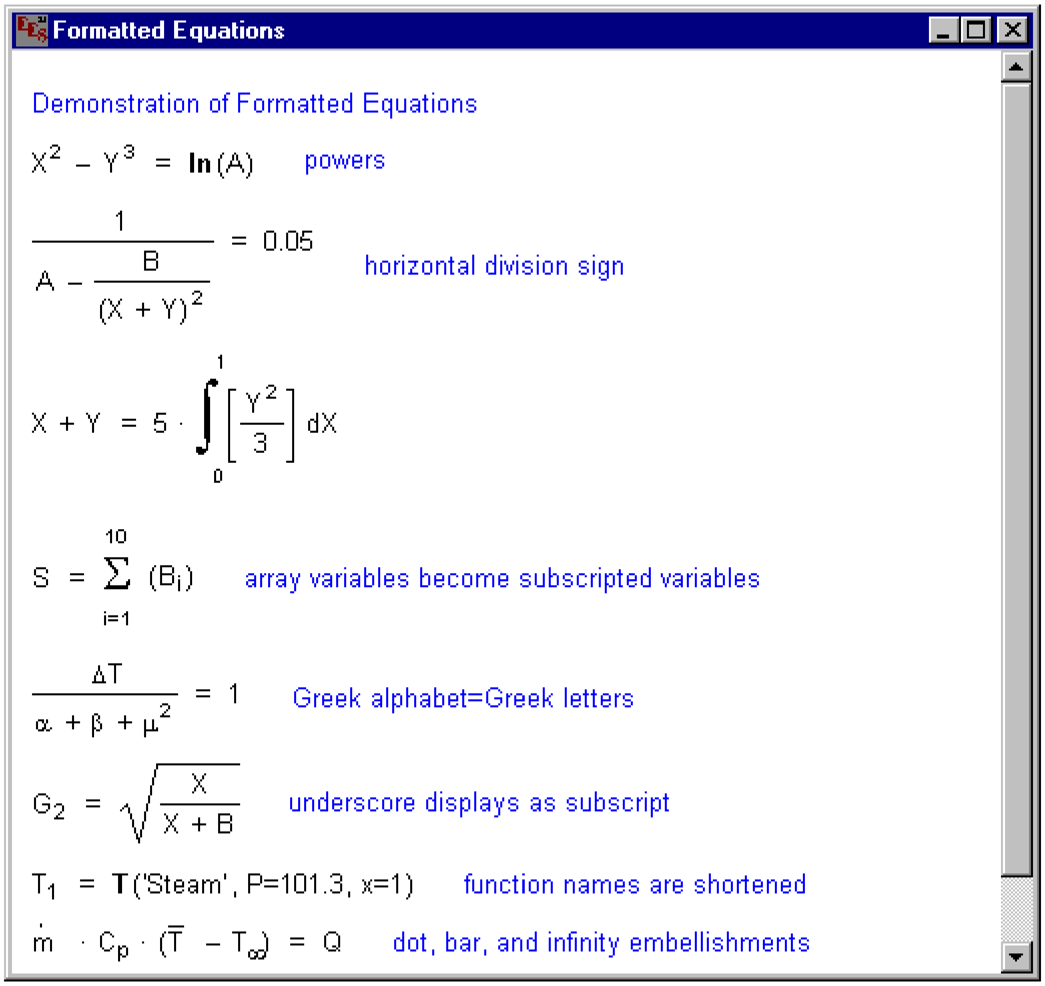
13 If EES is configured to operate in complex mode, all variables as assumed to have real and imaginary components. The complex mode configuration can be changed in the Preferences Dialog (Options menu) or with the $Complex On/Off directive.

Formatted Equations Window

The Formatted Equations window displays the equations entered in the Equations window in an easy-to-read mathematical format as shown in the sample windows below.



```
"Demonstration of Formatted Equations"
X^2-Y^3=ln(A) "powers"
1/(A-B/(X+Y)^2)=0.05 "horizontal division sign"
X+Y=5*integral(Y^2/3,X,0,1)   {This comment will not appear}
S=sum(B[i],i=1,10) "array variables become subscripted variables"
DELTAT/(alpha+beta+mu^2)=1 "Greek alphabet=Greek letters"
G_2=sqrt(X/(X+B)) "underscore displays as subscript"
T[1]=temperature(Steam,P=101.3, x=1) "function names are shortened"
m_dot*C_p*(T_bar-T_infinity)=Q "dot, bar, and infinity embellishments"
```



Demonstration of Formatted Equations

$X^2 - Y^3 = \mathbf{ln}(A)$ powers

$\dfrac{1}{A - \dfrac{B}{(X+Y)^2}} = 0.05$ horizontal division sign

$X + Y = 5 \cdot \displaystyle\int_0^1 \left[\frac{Y^2}{3}\right] dX$

$S = \displaystyle\sum_{i=1}^{10} (B_i)$ array variables become subscripted variables

$\dfrac{\Delta T}{\alpha + \beta + \mu^2} = 1$ Greek alphabet=Greek letters

$G_2 = \sqrt{\dfrac{X}{X+B}}$ underscore displays as subscript

$T_1 = \mathbf{T}(\text{'Steam'}, P=101.3, x=1)$ function names are shortened

$\dot{m} \cdot C_p \cdot (\bar{T} - T_\infty) = Q$ dot, bar, and infinity embellishments

Note that comments appearing in quotes in the Equations window are displayed in the Formatted Equations window but comments in braces are not displayed. An examination of the Formatted Equations Window will reveal a number of EES features to improve the display, in addition to the mathematical notation. Array variables, such as B[1] are (optionally) displayed as subscripted variables. Sums and integrals are represented by their mathematical signs. If a variable name contains an underscore, the underscore will signify the beginning of a subscript, as in variable G_2. However, note that although G[2] and G_2 will display in the same manner in the Formatted Equations Window, they are different variables with different properties. The index of array variables, e.g., G[2], can be used within the scope of Duplicate statements, or with the Sum and Product functions. In addition, the calculated value of G[2] can be displayed in the Arrays Window, as described in more detail in this chapter.

Placing _dot, _bar, or _hat after a variable name places a dot, bar, or hat (^) centered over the name. The _infinity results in a subscript with the infinity symbol (∞). A vertical bar character in a variable name signifies the start of a superscript. For example, G|o will display as G^o. Variables having a name from the Greek alphabet are displayed with the equivalent Greek letter. For example, the variable name beta will display as ß and mu will display as a µ. If the variable name in the Equations window is entered <u>entirely</u> in capital letters, and if the capital Greek letter is distinct from the English alphabet, the capital Greek letter will be used. For example, the variable name GAMMA will be displayed as Γ. The variable JTHETA will be displayed as a J in Symbol font which appears as a theta with a "curly" tail. A special form is provided for variables beginning with DELTA. For example, DELTAT displays as ΔT. Capital BETA looks just like a B, so EES will display the lower case equivalent, i.e., ß. Both the equations and comments will be formatted using these special symbols.

The formatted equations and comments appearing the Formatted Equations window can be moved to other positions if you wish. To move an equation or comment, move the cursor to the item and then press and hold the left mouse button down while sliding the equation or comment to a new location.

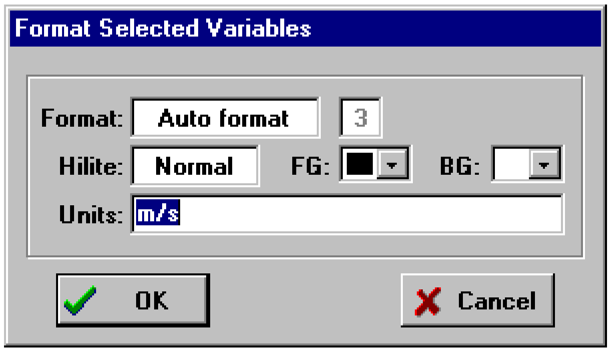
The formatted equations and comments are internally represented as Windows MetaFilePict items or pictures. You can copy one or more equation pictures from this window to other applications (such as a word processor or drawing program). To copy an equation, first select it by clicking the left mouse button anywhere within the equation rectangle. A selected equation or comment will be displayed in inverse video. You may select additional equations. Alternatively, the Select Display command in the Edit menu can be used to select all of the equations and comments which are currently visible in the Formatted Equations window. Copy the selected equations and comments to the clipboard with the Copy command. The equations will be unselected after the copy operation. Comments normally appear in blue text on the Formatted Equations window and they will appear in color when copied to the Clipboard. If you wish to have the comments displayed in black, hold the Shift key down while issuing the Copy command.

The text in the Formatted Equations window cannot be edited. However, clicking the right mouse button on an equation in the Formatted Equation window will bring the Equations window to the front with that equation selected where it can be edited.

Solution Window

The Solution window will automatically appear in front of all other windows after the calculations, initiated with the Solve or Min/Max commands in the Calculate menu, are completed. The values and units of all variables appearing in the Equations window will be shown in alphabetical order using as many columns as can be fit across the window.

The format of the variables and their units can be changed using the Variable Info command in the Options menu, or more simply, directly from the Solution window. Clicking the left mouse button on a variable selects that variable which is then displayed in inverse video. Clicking the left mouse button on a selected variable unselects it. Double-clicking the left mouse button (or clicking the right mouse button) brings up the Format Variable dialog window. The changes made in the Format Variable dialog are applied to ALL selected variables. Pressing the Enter key will also bring up the Format Variable dialog window.


Format Selected Variables

Format: Auto format 3

Hilite: Normal FG: ■ BG: □

Units: m/s

The numerical format (style and digits) and the units of the selected variables can be selected in this dialog window. When configured in Complex mode, an additional formatting option is provided for displaying the variable in rectangular or polar coordinates. The selected variables can also be highlighted (with underlining, bold font, foreground (FG) and background (BG) colors, etc.) or hidden from the Solution window. If a variable is hidden, it can be made visible again with the Display controls in the Variable Info dialog window. Additional information pertaining to the operation of the Solution window follows.

1. The Solution window is accessible only after the calculations are completed. The Solution menu item in the Windows menu will be dimmed when the Solution window is not accessible.

2. The unit settings made with the Unit System command in the Options menu will be displayed at the top of the Solution window if any of the built-in thermophysical property or trigonometric functions is used.

3. The Solution window will normally be cleared and hidden if any change is made in the Equations window. However, there is an option in the Preferences dialog of the Options menu to allow the Solution window to remain visible.

4. The number of columns displayed on the screen can be altered by making the window larger or smaller.

5. If EES is unable to solve the equation set and terminates with an error, the name of the Solution window will be changed to Last Iteration Values and the values of the variables at the last iteration will be displayed in the Solution window.

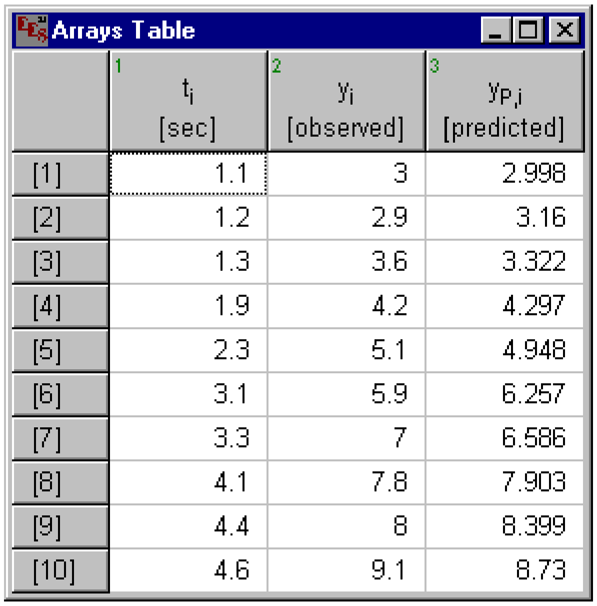
6. When the Solution window is foremost, the Copy command in the Edit menu will appear as Copy Solution. The Copy Solution command will copy the selected variables (shown in inverse video) to the clipboard both as text and as a picture. The text will provide for each variable (selected or not) a line containing the variable name, its value, and its units. The picture will show only those variables which are selected in the same format as they appear in the Solution window. The Select Display command in the Edit menu will select all variables currently visible in the Solution Window. (If you wish to force a black and white picture, hold the Shift key down when you issue the Copy Solution command.) Both the text and the picture can be pasted into another application, such as a word processor. Most word processors will, by default, paste the text. To paste the picture instead of the text, select the Paste Special command and select picture.

7. If the ☒ Display subscripts and Greek symbols option in the General Display tab of the Preferences dialog is selected, EES will display subscripts and superscripts of variable units. For example, m^2 will appear as m^2. An underscore character is used to indicate a subscript so lb_m will appear as lb_m.

Arrays Window

EES allows the use of array variables. EES array variables have the array index in square brackets, e.g., X[5] and Y[6,2]. In most ways, array variables are just like ordinary variables. Each array variable has its own guess value, lower and upper bounds and display format. However, simple arithmetic operations are supported for array indices so array variables can be more convenient in some problems as discussed in Chapter 7.

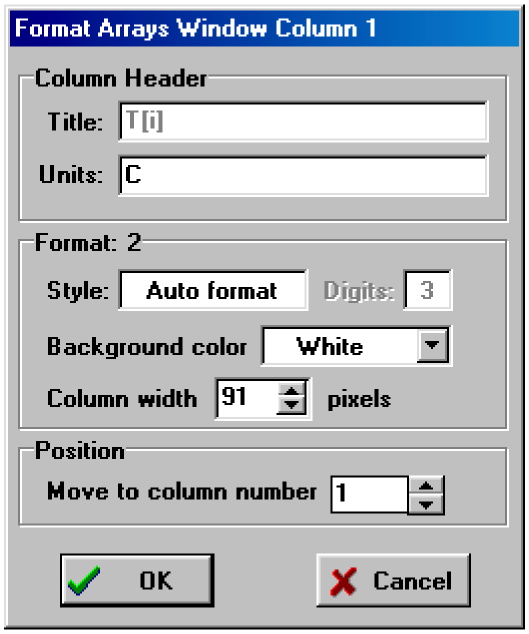
The values of all variables including array variables are normally displayed in the Solution window after calculations are completed. However, array variables may optionally be displayed in a separate Arrays window, rather than in the Solution window. This option is controlled with the ☒ Place array variables in the Arrays window check box in the Preferences dialog (Options tab) in the Options menu. If this option is selected, an Arrays window such as that shown below will automatically be produced after calculations are completed showing all array values used in the problem in alphabetical order with the array index value in the first column.



	1 t_i [sec]	2 y_i [observed]	3 $y_{P,i}$ [predicted]
[1]	1.1	3	2.998
[2]	1.2	2.9	3.16
[3]	1.3	3.6	3.322
[4]	1.9	4.2	4.297
[5]	2.3	5.1	4.948
[6]	3.1	5.9	6.257
[7]	3.3	7	6.586
[8]	4.1	7.8	7.903
[9]	4.4	8	8.399
[10]	4.6	9.1	8.73

The values in the Arrays window may be plotted using the New Plot Window command in the Plot menu. Part or all of the data in the Arrays window can be copied to another application by selecting the range of cells of interest followed by use of the Copy command in the Edit menu. If you wish to include the column name and units along with the numerical information in each column, hold the Ctrl key down while issuing the Copy command.

The format of values in any column of the Arrays window can be changed by clicking the left mouse button on the array name at the top of the column. The following dialog window will appear in which the units, display format and column position can be changed. Note that you can enter a number in the column number field or use the up/down arrows to change its value. If the value you enter is greater than the number of columns in the table, the column will be positioned at the right of the table.


Format Arrays Window Column 1

Column Header

Title: T[i]

Units: C

Format: 2

Style: Auto format Digits: 3

Background color White

Column width 91 pixels

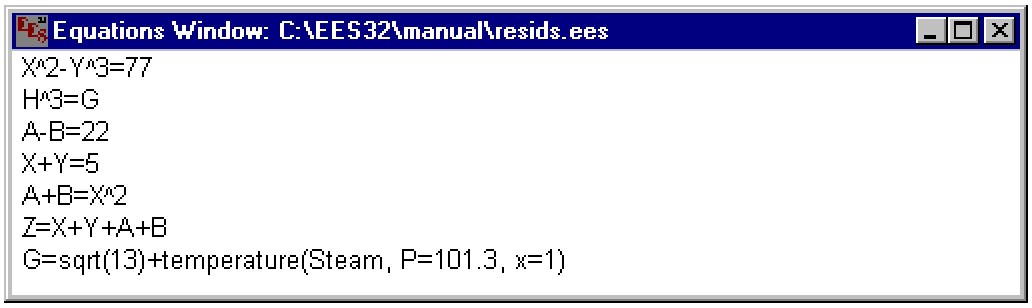
Position

Move to column number 1

Residuals Window

The Residuals window indicates the equation blocking and calculation order used by EES, in addition to the relative and absolute residual values. The absolute residual of an equation is the difference between the values on the left and right hand sides of the equation. The relative residual is the magnitude of the absolute residual divided by the value of left side of the equation.[3] The relative residuals are monitored during iterative calculations to determine when the equations have been solved to the accuracy specified with the Stopping Criteria command in the Options menu.

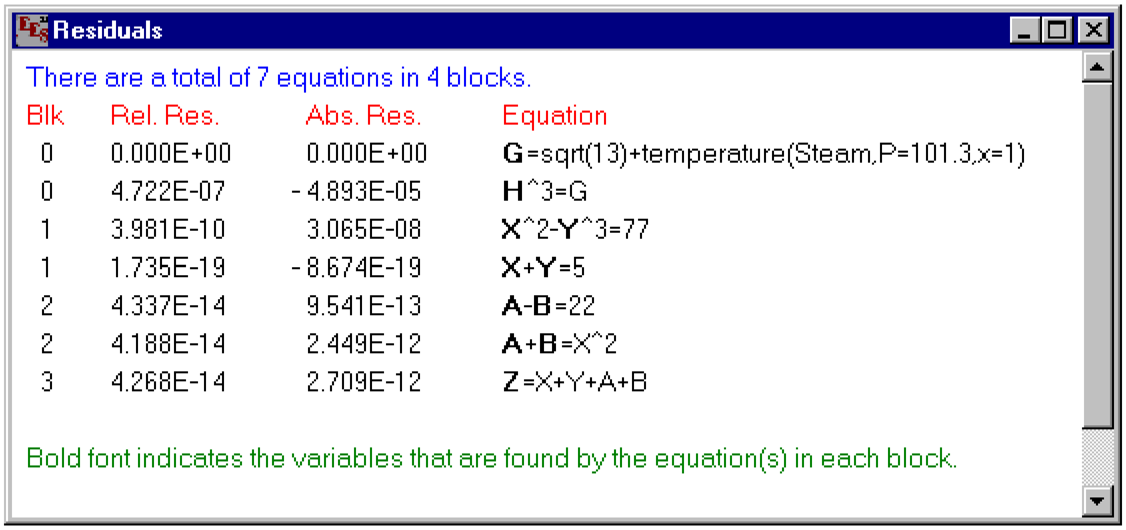
Consider, for example, the following set of six equations and six unknowns.



Equations Window: C:\EES32\manual\resids.ees

```
X^2-Y^3=77
H^3=G
A-B=22
X+Y=5
A+B=X^2
Z=X+Y+A+B
G=sqrt(13)+temperature(Steam, P=101.3, x=1)
```

EES will recognize that these equations can be blocked, i.e., broken into two or more sets, as described in more detail in Appendix B. The blocking information is displayed in the Residuals window.



Residuals

There are a total of 7 equations in 4 blocks.

Blk	Rel. Res.	Abs. Res.	Equation
0	0.000E+00	0.000E+00	**G**=sqrt(13)+temperature(Steam,P=101.3,x=1)
0	4.722E-07	-4.893E-05	**H**^3=G
1	3.981E-10	3.065E-08	**X**^2-**Y**^3=77
1	1.735E-19	-8.674E-19	**X**+**Y**=5
2	4.337E-14	9.541E-13	**A**-**B**=22
2	4.188E-14	2.449E-12	**A**+**B**=X^2
3	4.268E-14	2.709E-12	**Z**=X+Y+A+B

Bold font indicates the variables that are found by the equation(s) in each block.

Variables having values that can be determined directly, i.e., without simultaneously finding the values of other variables, such as G in the example above, are determined first and

[3] If the value of the left hand side of an equation is zero, the absolute and relative residuals assume the same value.

assigned to Block 0[4]. Once G is known, H can be determined. The order in which these individual equations are solved in Block 0 is indicated by the order in which they appear in the Residuals window. After solving all equations in Block 0, EES will simultaneously solve the equations in Block 1, then Block 2, and so on until all equations are solved. The first and third equations in the example above can be solved independently of other equations to determine X and Y and are thereby placed in Block 1. Similarly, the second and fourth equations, which determine A and B, are placed in Block 2. With X, Y, A, and B now known, Z can be determined, so it appears in Block 3. Note that the variable(s) that are determined by the equation(s) in each block are shown in bold font.

The Residuals window will normally be hidden when any change is made in the Equations window. This automatic hiding can be disabled with the Display Options command in the Options menu.

It is possible to display the Residuals window in a debugging situation. If the number of equations is less than the number of unknowns, EES will not be able to solve the equation set, but the Residuals window can be made visible by selecting it from the Windows menu. Normally, the block numbers appear in sequential order. When one or more equations are missing, EES will skip a block number at the point in which it encounters this problem. The equations in the following blocks should be carefully reviewed to determine whether they are correctly and completely entered.

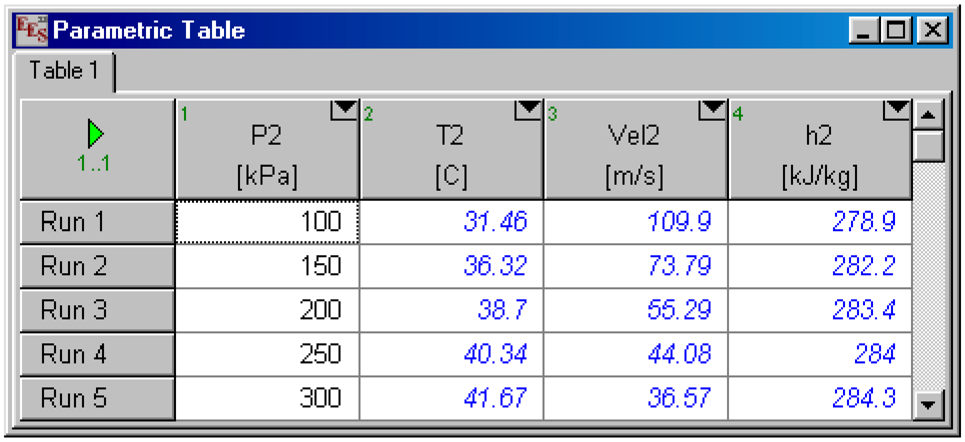
The information in the Residuals window is useful in coaxing a stubborn set of equations to converge. An examination of the residuals will indicate which equations have been solved by EES and which have not. In this way, the block of equations that EES could not solve can be identified. Check these equations to be sure that there is a solution. You may need to change the guess values or bounds for the variables in this block using the Variable Info command in the Options menu.

Doubling-clicking the left mouse button (or clicking the right mouse button) on an equation in the Residuals window will cause the Equations window to be brought to the front with the selected equation highlighted. Use the Find command in the Search menu to help locate the equations.

The entire contents of the Residuals window will be copied as tab-delimited text to the Clipboard if the Copy command is issued when the Residuals window is foremost.

[4] Variables specified in the Diagram Window are identified with a D rather than a block number. See the Diagram Window section. In Complex mode, each equation is shown twice, once for the real part identified with (r) and again for the imaginary component labeled with (i)

Parametric Table Window



1..1	P2 [kPa]	T2 [C]	Vel2 [m/s]	h2 [kJ/kg]
Run 1	100	31.46	109.9	278.9
Run 2	150	36.32	73.79	282.2
Run 3	200	38.7	55.29	283.4
Run 4	250	40.34	44.08	284
Run 5	300	41.67	36.57	284.3

The Parametric Table window contains one or more Parametric Table(s). A Parametric table operates somewhat like a spreadsheet. Numerical values can be entered into any of the cells. Entered values, e.g., the values in the P2 column in the above table, are assumed to be independent variables and are shown in normal type in the font and font size selected with the Preferences command (Options menu). Entering a value in the Parametric Table produces the same effect as setting that variable to the value with an equation in the Equations window. Dependent variables will be determined and displayed in the table in blue, bold type, or italics (depending on the choice made with the Preferences command) when the Solve Table or Min/Max Table command in the Calculate menu is issued.

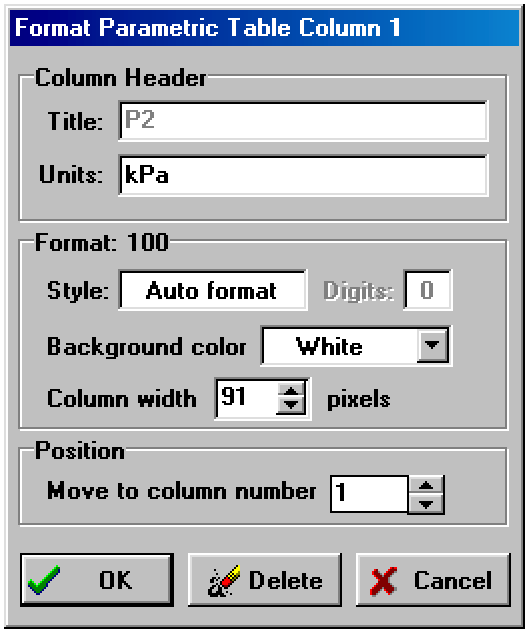
1. A parametric table is created using the New Parametric Table command in the Tables menu. The variables that are to appear in the table are selected from a list of variables currently appearing in the Equations window. Each new table is given a name that appears on a tab at the top of the Parametric Window. When there is more than one table, clicking on the tab brings the corresponding Parametric Table to the front.

2. Each row of the Parametric Table is a separate calculation. The number of rows is selected when the table is generated, but may be altered using the Insert/Delete Runs command in the Tables menu. The maximum number of rows in the Commercial version is 6500. There is no limit to the number of rows in the Professional version.

3. Variables may be added to or deleted from an existing Parametric Table using the Insert/Delete Vars command in the Tables menu. One or more columns can be deleted more simply by right-clicking in the column header and selecting Delete from the pop-up menu.

3. The initial order in which the columns in the Parametric Table appear is determined by the order in which the variables in the table were selected in the New Parametric Table dialog. To change the column number order, click the right mouse button in the column header cell (but not on the alter values control at the upper right). A pop-up menu will

appear with Properties as one of the menu items. Select the Properties menu item. A dialog window will appear as shown below in which the column number can be changed be clicking the up or down arrows to the right of the column number or by directly editing the column number. The display format, units, column width, and column background color can also be entered or changed at this point.

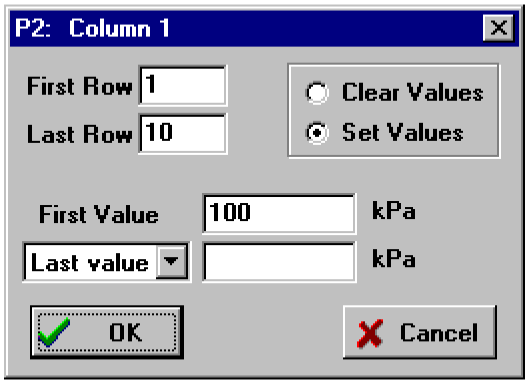

Format Parametric Table Column 1

Column Header
Title: P2
Units: kPa

Format: 100
Style: Auto format Digits: 0
Background color: White
Column width: 91 pixels

Position
Move to column number: 1

5. Values can be automatically entered into the Parametric table using the Alter Values command in the Tables menu. Alternatively, right-clicking in the column cell header and selecting Alter values from the pop-up menu or clicking the mouse on the ▼ control at the upper right of the column header cell will bring up the dialog window shown below which provides the same automatic entry somewhat more conveniently.


P2: Column 1

First Row: 1
Last Row: 10
Clear Values [✓ Set Values]

First Value: 100 kPa
Last value: kPa

6. A green 'go' triangle is displayed in the upper left cell of the Parametric table. Clicking the left mouse button in this triangular area will initiate the foremost Parametric Table calculations for the rows indicated below the triangle. The row range is that selected during the last use of the Solve Table dialog. To select a different range, click the left

mouse button in the first row. Hold the Shift key down and then click the mouse in the last row. Continue to hold the Shift key down while clicking in the green triangle.

7. A Sum row which displays the sum of the values in each column may be hidden or made visible using the 'Include a Sum row in the Parametric table' control provided in the Preferences dialog window (Options tab) in the Options menu.

8. Clicking the left mouse button in the leftmost column of a row in any table will select all of the cells in that row. If the mouse button is held down while the clicking in leftmost column of other rows, these rows will be added to the selection. Rows can also added by holding the Shift key down while clicking in the leftmost column. All of the cells in a column can be selected in a similar manner by clicking the left mouse button in the column header. A right-click in the cell header will bring up the dialog that allows the format, units, and other information in a column to be edited. Right-clicking after selecting a range of cells brings up a popup menu that allows selected printing, copying, and other options.

9 A Parametric Table can be used to solve differential equations or integrals. See Chapter 7 for additional information.

10. The **TableValue** function returns the value of a table cell at a specified row and column.

11. The **TableRun#** function returns the row of the table for which calculations are currently in progress.

12. The TableName$ function returns the name of the Parametric table (as displayed on the tab in the Parametric window) that is currently in use.

13. The independent variables in the Parametric Table may differ from one row to the next. However, when the independent variables are the same in all rows, EES will not have to recalculate the Jacobian and blocking factor information and can thus do the calculations more rapidly.

14. Tabular data may be imported or exported from the Parametric Table via the Clipboard using the Copy and Paste commands in the Edit menu. To copy data from any of the EES tables, click the mouse in the upper left cell. Hold the Shift key down and click in the lower right cell, using the scroll bar as needed. The selected cells will be shown in inverse video. (When the Shift key is released, the upper left cell that has the focus will return to normal display. However, even though it may not displayed in inverse video, the upper left cell is selected and it will be placed on the clipboard with other cells when the Copy command is issued.) Use the Select All command in the Edit menu to select all of the cells in the table. The data are placed on the clipboard with a tab between each number and a carriage return at the end of each row. With this format, the table data will paste directly into a spreadsheet application. If you wish to include the column name

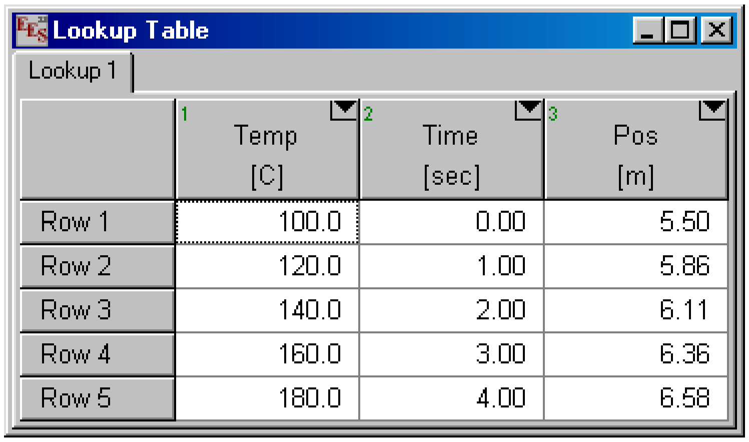
and units along with the numerical information in each column, hold the Ctrl key down while issuing the Copy command. The Copy command is also accessible from a pop-up menu by right-clicking the mouse in a selected cell.

15. Right-clicking in Parametric, Lookup, Arrays, or Integral table windows will bring up a pop-up menu allowing the current selection to be cut, copied, or printed. This option provides a convenient way to print a specified section of a table. Right clicking in the Row column of any table now brings up a menu that allows a back color or border to be associated with the row.

Lookup Table Window

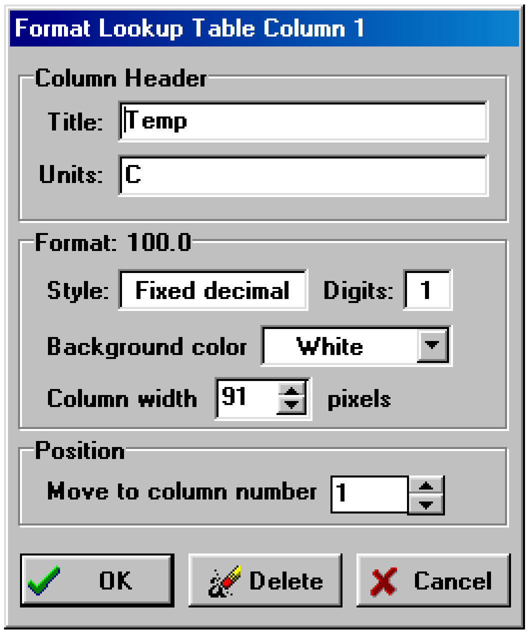
A Lookup Table provides a means of using tabular information in the solution of the equations. A Lookup Table is created using the New Lookup Table command in the Tables menu. There is no limit, other than available memory, on the number of Lookup Tables that can be placed in the Lookup Table Window. Each Lookup table is associated with a table name that appears on the tab at the top of the Lookup Table Window. The number of rows and columns in the table are specified when the table is created and may be altered with Insert/Delete Rows and Insert/Delete Cols commands in the Tables menu or by right-clicking in the row or column headers. All Lookup Tables are saved with other problem information when the EES file is saved. In addition, a Lookup Table may be saved on a disk (separately from the EES file) using the Save Lookup command in the Tables menu. A .LKT filename extension is used to designate EES Lookup files. Lookup files can also be saved in ASCII format with a .TXT or .CSV filename extension. The Lookup table may then be accessed from other EES programs in any of these formats.

The **Interpolate** commands provide linear, quadratic or cubic interpolation or extrapolation of the data in the Lookup Table. See Chapter 4 for details. In addition, the **Lookup**, **LookupCol**, and **LookupRow** functions allow data in a Lookup Table to be linearly interpolated (forwards and backwards) and used in the solution of the equations. The Lookup Table may either reside in the Lookup Table Window or in a previously-saved Lookup File with a .LKT, .TXT, or .CSV filename extension, as explained in more detail in Chapter 4.



	1 Temp [C]	2 Time [sec]	3 Pos [m]
Row 1	100.0	0.00	5.50
Row 2	120.0	1.00	5.86
Row 3	140.0	2.00	6.11
Row 4	160.0	3.00	6.36
Row 5	180.0	4.00	6.58

A sample Lookup Table is shown above. The column number is displayed in small type at the upper left of each column header cell. The column number can be used to specify a specific column when used with the **Lookup** functions. However, the **Lookup** functions will also accept 'ColumnName' in place of the column number where 'ColumnName' is the name of the column shown in the column heading surrounded by single quote marks.[5] The column names are initially **Column1**, **Column2**, etc. but these default names can be changed by clicking the right mouse button in the header cell and selecting Properties from the popup menu that appears.



Format Lookup Table Column 1

Column Header

Title: Temp

Units: C

Format: 100.0

Style: Fixed decimal Digits: 1

Background color: White

Column width: 91 pixels

Position

Move to column number: 1

The column title can be changed and units for the values in the column may be specified. The Format controls allow the data in each column of the table to be displayed in an appropriate numerical format. A control is provided to change the background color for each column. The column width and position may also be changed by either clicking the up or down arrows to the right of the column number or by editing the number directly.

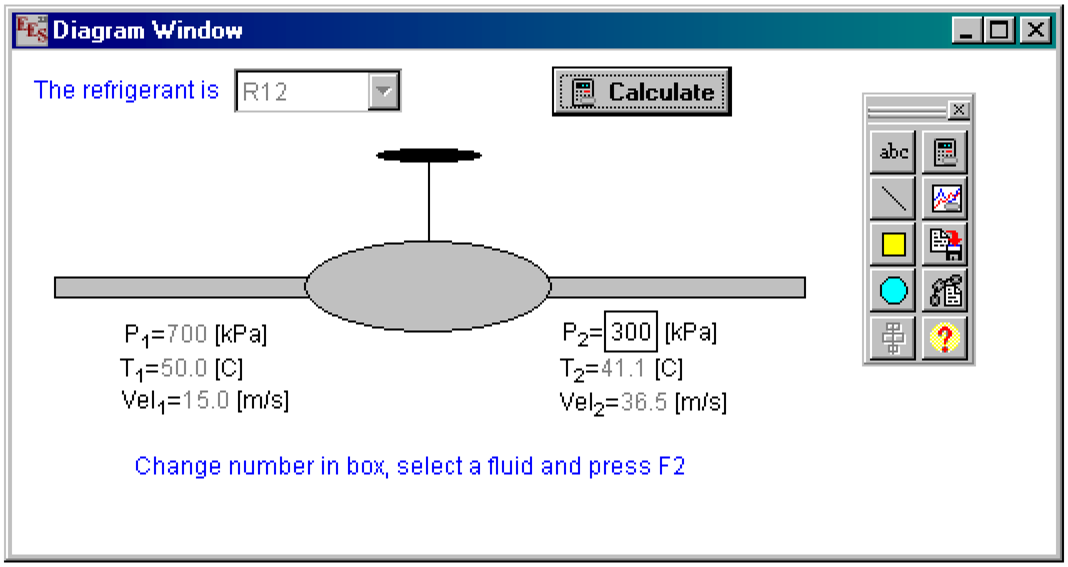
Data can be imported to or exported from the Lookup Table through the Clipboard in the same manner as described for the Parametric Table. Data may be automatically entered into the Lookup table by clicking on the ▾ control at the upper right of the column header cell, as described for the Parametric table. Hold the *Shift* key down when issuing the copy command if you wish to copy the column names and units with the other numerical information on the Clipboard. Data may be interchanged between the Parametric and Lookup Table windows.

One or more Lookup Tables can be deleted from the Lookup Table Window, if desired, with the Delete Lookup Table menu item in the Options menu. Lookup Table files saved with a .LKT, .TXT , or .CSV filename extension can not be deleted from within EES.

[5] EES will also accept #ColumnName in place of the column number.

Diagram Window

The Diagram window can be used in two ways. First, it provides a place to display a diagram (or text) relating to the problem that is being solved. For example, a schematic diagram of a system identifying state point locations can be displayed in the Diagram window to help interpret the equations in the Equations window. Second, the Diagram window can be used for both input and output of information and for report generation. In the Professional version, 'hot areas' can be defined that, when clicked, open additional child Diagram windows. A Calculate button can be placed on Diagram and child windows to conveniently initiate the calculations. Link buttons can be put on the Diagram that display plots or start other programs. Shown below is a diagram with a schematic and input and output variable information superimposed.



P_1=700 [kPa]
T_1=50.0 [C]
Vel_1=15.0 [m/s]

P_2=300 [kPa]
T_2=41.1 [C]
Vel_2=36.5 [m/s]

Change number in box, select a fluid and press F2

Creating the Diagram

There are two ways to place a drawing in the Diagram window. First, a diagram can be created in any drawing program that produces an object drawing such as Microsoft Draw (included in Word for Windows), Corel Draw, Designer, or PowerPoint. Copy the drawing and then Paste it into the Diagram Window. A scanned image in bitmap format can also be used as the diagram provided that it is copied as an picture, rather than a bitmap. An easy way to convert the bitmap into a picture is to paste the bitmap into PowerPoint, PaintBrush or other programs that convert graphical images. Then copy the image from this program into the Diagram window. A second way to create a drawing is to use the drawing tools provided on the Diagram window toolbar. The toolbar provided graphic primitives such as lines/arrows, circles, and rectangles. The characteristics of these graphic objects can be changed by right-clicking or double-clicking the mouse button on the object while the toolbar is visible. The two methods of creating a drawing can be combined. For example, you can position a rectangle on top of a scanned image. The diagram is saved along with all other problem information.

Development and Application Modes

The Diagram window (and child Diagram windows) operate in two modes. When the tool bar is visible, the Diagram window is in development mode. (The tool bar can be made visible or hidden using the Show/Hide Diagram Tool Bar command or the Diagram Window speedbutton.) In the development mode, all objects such as the picture, text, and graphic objects (lines/arrows, rectangles, ellipses) added to the Diagram window, can be moved, modified, or deleted as indicated below. Input variables are disabled in development mode and calculations initiated with the Calculate button are suppressed. When the tool bar is hidden, the Diagram window is in application mode. In application mode, it is not possible to move any of the objects or change their display characteristics. Input variables are enabled and calculations can be initiated using the optional calculate button or the commands in the Calculate menu.

Moving the Diagram

The diagram can be repositioned in the Diagram window in development mode by pressing and holding the left mouse button down anywhere within the diagram rectangle while sliding the diagram to its new location. Any text and graphic objects (e.g., lines, rectangles, buttons, etc.) placed on the Diagram window will move with the diagram itself unless the Ctrl key is depressed.

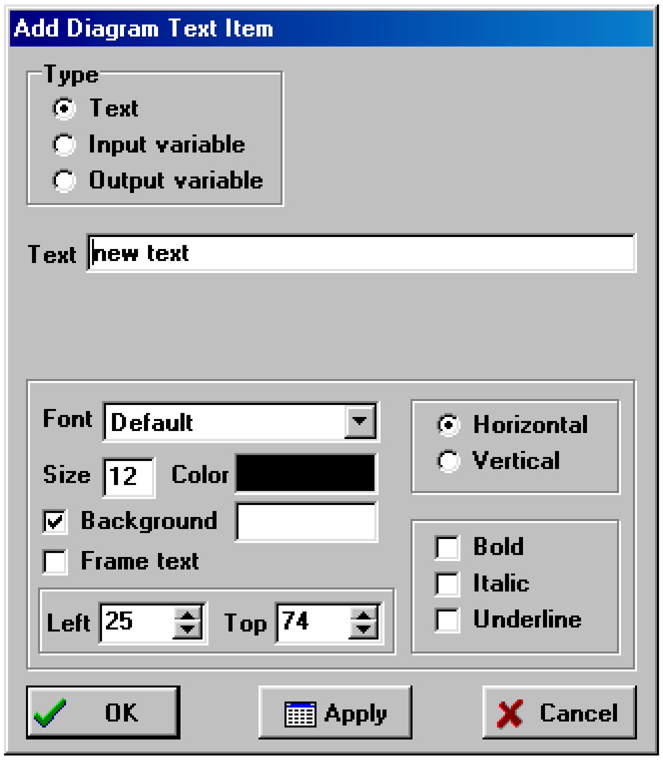
Resizing the Diagram

In development mode, the diagram and its associated text can be resized by double-clicking the left mouse button (or clicking the right mouse button) anywhere within the Diagram window or child Diagram window except on a text, graphic object or button. After a positive confirmation, the diagram will then be resized to fit in the Diagram window. Normally, the diagram and all text and graphic objects in the Diagram window are resized. However, if the Ctrl key is depressed, only the diagram is resized. The diagram and all associated text will be scaled to fit within the Diagram Window by double-clicking the left mouse button (or clicking the right mouse button) anywhere within the Diagram window, except on a text item. The diagram can be made larger or smaller by first changing the size of the Diagram window and then double-clicking to change the size of the diagram itself. The aspect ratio of the diagram is not changed as the diagram is resized.

abc **Adding and Moving Text on the Diagram Window**

The Add Text button on the Diagram window toolbar allows text to be placed anywhere on the Diagram window. Three different types of text may be selected using the radio buttons at the upper left of the diagram window. Selecting the Text radio button will cause the window to appear as shown below, in which the text and its characteristics can be specified. The text will initially appear in a default position within the Diagram window when the dialog is dismissed. It can then be dragged to a new position by pressing and holding the left mouse button down while sliding the text to the desired location. The text or any of its characteristics can later be changed by double-clicking the left mouse button (or by clicking the right mouse button) while the cursor is positioned over the text.

Clicking the Input or Output radio buttons changes the dialog window display so that a list of currently defined variables replaces the text edit box, as shown. Select the variable by clicking on its name in the list. Both Input and Output variables values are displayed on the diagram with the option of also displaying the variable name and unit string. An Output variable displays the value of the selected variable calculated during the previous calculation. An Input variable will be displayed with the value enclosed in a rectangle. This value can be edited and it provides the same function as an equation in the Equations window which sets the variable to a value.


Add Diagram Text Item

Type: [✓ Text] Input variable Output variable

Text: new text

Font: Default

Size: 12

Color: ■

[✓] Background

[ ] Frame text

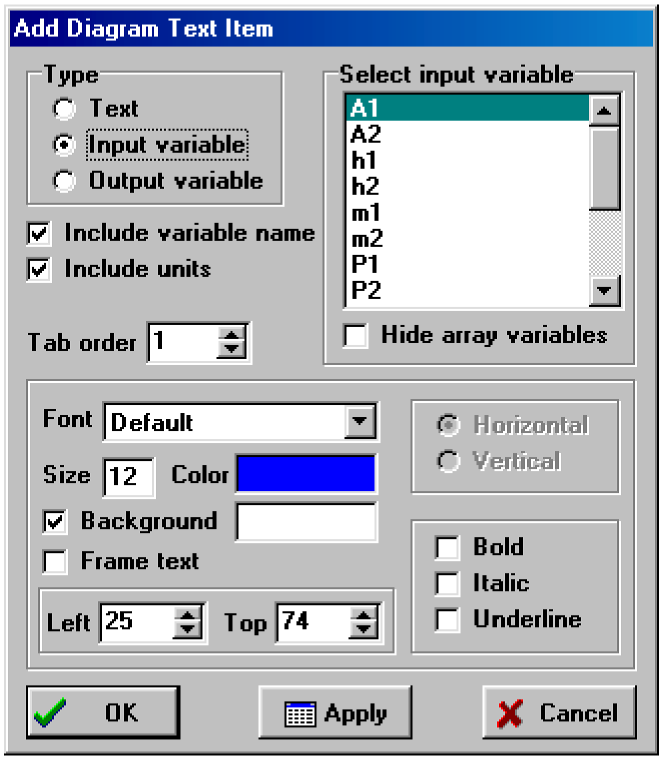
Left: 25 Top: 74

[✓ Horizontal] Vertical

[ ] Bold [ ] Italic [ ] Underline

If a string variable (identified with a $ character as the last character in the variable name) is selected for an Input variable, EES will provide an option of selecting the variable from a pull-down list of string constants that you provide. For example, the Diagram window shown at the start of this section employs a pull-down list of refrigerants. The refrigerant selected by the user is assigned to a string variable which is then provided as the fluid name in the thermodynamic functions.

When the Solve or Min/Max commands (Calculate menu) are issued, EES will first examine the Diagram window to see which variables, if any, are set, provided that the Diagram window is not hidden. A value which is set in the Diagram window cannot also be set in the Equations window. After the calculations are completed, the newly-calculated values of the Output variables will be displayed on the Diagram window. Output values will display in gray-colored text if the value is not currently defined.



Type: Text [✓ Input variable] Output variable

[✓] Include variable name

[✓] Include units

Tab order 1

Select input variable: A1, A2, h1, h2, m1, m2, P1, P2

[ ] Hide array variables

Font Default

Size 12 Color

[✓] Background

[ ] Frame text

Left 25 Top 74

[✓ Horizontal] Vertical

[ ] Bold

[ ] Italic

[ ] Underline

The Diagram window input is ignored if the Diagram window is hidden. The Diagram window can be used with the Parametric table (i.e., the Solve Table command) if the 'Use Input from Diagram' check box is checked in the Solve Table dialog window.

Using Drop-Down Lists to Set One or More EES Variables

When a string variable is selected for input in the Diagram Window, the option of including the units is replaced with an option for providing a list of alternative choices for the string variable in a drop-down list. An Edit button will appear which will allow the existing choices to be viewed and/or changed. The string choices are displayed in a drop-down list box which displays the available choices when the user clicks in the box.

It is possible to use the strings in the drop-down list to set the values of one or more EES variables. This is accomplished by following the string that is to be displayed with characters // and then with one or more EES equations on one line. For example, suppose you wish to use a drop-down list to have a user select the density and thermal conductivity of one of several substances. You could create a drop-down list with the following strings:

Copper // density=8933; conductivity=400
Aluminum // density=2700; conductivity=237
Nickel // density=8900; conductivity=91
Zinc // density=7140; conductivity=116
User Input // density=?; conductivity=?

When an item from the drop-down list is selected or when calculations are initiated, EES will execute the equations to the right of the // characters and update the Diagram Window display. If the Format Display control in the Modify Diagram Text dialog is checked, the drop-down list will only show the characters to the left of // like this.

An interesting extension of this concept is provided in the last string of this list which is 'User Input // density=?; conductivity=?'. In this case, EES interprets the '=?' to mean that that the value of the variable should be entered in an edit box. In order for this to work, the variables density and conductivity must be displayed on the Diagram window as Output variables. When the 'User Input' item is selected, EES will look for these Output variables and change them to Input variables. The general capability afforded by the '=?' notation is that selected variables can be changed from Output to Input or from Input to Output variables on the Diagram Window by a user selection from a drop-down list.

Adding Graphic Items

Lines, rectangles or circles (ellipses) can be drawn on the Diagram window or child diagram windows using the line, rectangle, and circles tools on the toolbar. Holding the Shift key down while drawing a line will cause the line to be drawn horizontal, vertical, or to the closest 45° angle. Holding the Shift key down while drawing rectangles and circles will cause them to be draw with its width equal to its height. The graphic items can be resized or moved in development mode as described next.

Selecting, Modifying and Aligning Text and Graphic Items

One or more text and graphic items in the Diagram window or child diagram windows can be selected by clicking the left mouse button on the item while in development mode. A red dotted box will be displayed around each selected text item. Hold the Shift key down to select more than one item. A selected graphic object will be displayed with its 'handles' (small boxes on each edge) showing. All selected items can be moved at once using the mouse or the arrow keys. The align button in the tool bar can be used to facilitate alignment of the selected items relative to one another. If all of the selected items are text items, pressing the right mouse button will bring up a dialog box in which the font, size, color, and style of the selected text items can be changed as a group. When it first appears, the dialog box displays the characteristics of the first selected text item. Any change made to a text characteristic in the dialog window will be applied to all selected text items. If a text characteristic, such as the font size, is not changed in the dialog window, then the font size of the selected text items will not be changed. Similar capability is provided if all selected items are graphic objects. Clicking anywhere in the Diagram window that is not on a text or graphic object unselects all selected items.

Adding a Calculate Button

A Calculate button can be placed on the Diagram window (but not on a child Diagram window) by selecting the Show/Hide Calculate button in the Diagram window tool bar. By default, the Calculate button caption is "Calculate" and pressing the button provides exactly the same function as selecting Solve from the Calculate menu or pressing F2. The calculate button caption and the action taken when the Calculate button is clicked can be changed by right-clicking the button and providing the desired text while in development mode.

Adding a Plot Window Access Button

A Plot Window access button can be placed on the Diagram window or child Diagram window by selecting the Add Plot Window button in the tool bar. A dialog window will appear after clicking this toolbar button in which the caption and plot window number for the plot window access button can be specified. The plot window is selected by clicking the appropriate radio button. (If a plot window does not exist, its radio button will be disabled.) Clicking OK in this dialog window will create a plot window access button. The plot window access button can be dragged to any location when the toolbar is visible. Right-clicking on the plot window access button when the toolbar is visible will bring up the dialog so that the plot window number or button caption can be changed. When the toolbar is hidden, clicking the plot window access button will bring the specified plot window foremost. If the plot window does not exist, this button will do nothing.

The Copy command can be used to copy the contents of the Diagram window or a child Diagram window to the Clipboard. The information on the clipboard can then be pasted into a word processor or other application. If you hold the Ctrl key depressed during the copy process, the text items added with the Diagram Window tool bar command will not be included with the diagram.

Use the Clear command to delete an existing diagram and associated text.

Creating Hot Areas and Child Diagram Windows *(Professional Version only)*

A 'Hot Area' is a rectangular region on the main Diagram Window that will open a child Diagram Window when the cursor is positioned in the region and the left mouse button is clicked. The child Diagram window includes all of the features of the main Diagram Window, including the ability to have hot areas with its own child windows. To create a hot area, move the cursor to the upper left of the region and drag it to the lower right while holding both the Ctrl and Shift keys down. When the mouse button is let up, a window will appear in which you provide a name for the child Diagram Window. The child Diagram Window will appear whenever you click the mouse in the designated area. You can copy a figure into this window and add text and graphic objects using the commands in the tool bar in the same manner as in the main Diagram window. Note that the tool bar must be visible (i.e., development mode) to move or modify any text or graphic objects.

The characteristics of the child Diagram Window can be changed by clicking the right mouse button in the region while holding the Ctrl and Shift keys down. A dialog window will appear in which the name and the region bounds, and other information can be viewed or changed. The hot area can also be deleted with this dialog.

Hold the Ctrl and Shift keys down to see an outline of all currently defined hot areas.

Saving User Inputs

A Save User Inputs button can be placed on the Diagram window (but not on a child Diagram window) by selecting the Show/Hide Save Inputs button in the Diagram window tool bar. The Save button is needed only for Distributable[6] programs, since regular EES programs can always save all program information with the normal Save command. When the user presses the mouse on the Save button, a file containing the current values of the all EES variables, including the current values of inputs, is written to the disk. The file that is written has the same parent name as the EES file that is executing and a .VAR filename extension. The file is written to the directory that the Distributable program is located in. EES will look for this .VAR file when the Distributable program is started or when one of the five files contained in the Distributable program is selected from the recent file list at the bottom of File menu. If the .VAR file is found, it will be automatically loaded and the input information on the Diagram windows will be updated. The Save button is disabled until calculations have been successfully completed. In this way, it is not possible to write a set of input information that will not successfully calculate. By default, the Save button caption is "Save". The caption can be changed by right-clicking on the button and providing the desired text while in development mode.

Creating Links (Professional Version only)

Link buttons can be placed on the Diagram window or child Diagram window by selecting the Create Link button in the tool bar. A dialog window will appear after clicking this toolbar button in which the caption and link properties can be specified and assigned to a button which will appear on the foremost diagram window. The caption and properties can be changed at a later time by right-clicking on the link button while the Diagram window is in development mode. Five different types of links can be created:

[6] See the Make Distributable Program command in the File Menu section of Chapter 3 for information on Distributable programs.

Link type 1: Start an External Program

This link type allows any program, including a second copy of EES, to be started when the user clicks the link button in the Diagram window while in application-mode. The link button property should be the full filename and optionally, the name of the file that is to be opened when the application starts.   As an example, the following link property will start a second copy of EES and load file TEST.EES in the C:\EES32 directory. When EES is started, it normally checks to see if EES is already running and if so, it displays a warning asking whether the user really wants to launch another copy of EES. This check can be disabled by using the /MULT directive as shown in the example.

```
C:\EES32\EES.exe TEST.EES /MULT
```

EES can communicate with other programs using lookup files. The $SAVELOOKUP and $OPENLOOKUP directives make this process more convenient. In addition, EES will check the date/time stamp of any lookup file it uses and if the file is changed, it will be automatically reloaded.

Link type 2: Open EES File

This link type will close the existing EES file and open the EES file specified as the link button property. EES will check to see that the current EES file is saved, and if not, ask for confirmation before closing the existing file. This option is useful when the results of one EES program provides data for use in the next EES program. For example, information can be calculated and saved in a Lookup Table that is automatically loaded in the next EES program with the $OPENLOOKUP directive.

Link type 3: Open Distributable File

This link type is applicable only to distributable programs created with the Make Distributable Program command. Distributable programs can include up to 5 EES files. As with option 2, this link type allows one EES file to open another, but in the case of a distributable program the EES file must be one of the five included with the distributable program. It is referred to by its number, 1 to 5.

Link type 4: Open Child Diagram Window

This link type is applicable if one or more child Diagram Windows have been created. A list box showing the names of all child Diagram windows is provided. Clicking the button will open the specified child Diagram Window, just as if the user had clicked the mouse within the hot area for the child Diagram window. However, the button can be positioned anywhere in the window.

Link type 5: Play Macro adapted with EES variables

This link type will play a predefined Macro file that has been stored with a .EMF filename extension. The first parameter that must be supplied in the edit box is the macro filename, including the .EMF filename extension. All other parameters are optional. It is possible to provide information to the macro file that allows its capability to be programmed. This is done by providing parameters after the macro file name separated with a space or comma. These parameters can be EES variable names, numerical values, or strings. EES will process the Macro file looking for the character sequence %%N where N is an integer value. If this sequence is found, %%N is replaced with the Nth parameter. For example, suppose the macrofile name doPropPlot.EMF has a line that contains

```
PropPlot %%1 PH 2 %%2 %%3 0 DoQLines
```

If the Play Macro link information is

```
doPropPlot.EMF R$, Thigh, 0
```

EES will replace %%1 with the value of the EES string variable R$, %%2 with the value of EES variable Thigh, and %%3 with 0. The property plot will then be drawn. Note that calculations must be completed to enable the button.

Clicking OK in the Link Properties dialog will create the link button. The button can be dragged to any location while the Diagram Window is in development mode. Right-clicking on the link button while in development mode will bring up the Link Properties dialog so that link properties can be changed or the button can be deleted.

Adding a Help Button

A Help button can be placed on the Diagram window or child Diagram window when the window is in development mode. In this case, the toolbar will be visible. Click on the Show/Hide Help button on the tool bar which is identified with a yellow circle containing a question mark symbol. The Help button will appear along with a small dialog in which you may enter the caption for the Help button and the name of the help file that will be displayed when the user selects the Help button in Application mode. This file can either be an ASCII text file (*.TXT), a Windows help file (*.HLP) or an HTML file (*.HTM). EES will automatically determine the file type independent of the filename extension. The help button can be removed by clicking the Show/Hide Help button in development mode.

Navigating through Child Diagram Window (Professional Version)

The main Diagram window can have 'hot areas' which bring up child Diagram windows when clicked. The child Diagram window can also have 'hot areas' forming grandchildren and so on. Each child Diagram window will have a small 'home' button in the lower right corner. Clicking this button will close the window and bring up the main Diagram window. If the parent of the Diagram window is not the main Diagram window, a second button with a left arrow will also be displayed at the lower right. Clicking this button will close the child Diagram window and bring its parent.

Adding a Print Button

A Print button can be placed on the Diagram window or child Diagram window when the window is in development mode. In this case, the toolbar will be visible. Click on the Show/Hide Print button on the tool bar which is identified with a printer icon. The Print button can be dragged to any location in the window while the toolbar is visible. The caption can be changed or removed by right-clicking on the button. The button can be removed by clicking the Show/Hide Print button on the tool bar. Clicking the left mouse button on the Print button while in application mode will print the contents of the window to the default printer that has been selected with Printer Setup command.

Saving User Inputs

Save and Load Inputs buttons can be placed on the Diagram window (but not on a child Diagram window) by selecting the Show/Hide Save or Load Inputs buttons in the Diagram window tool bar. Save and Load buttons are needed only for Distributable programs, since regular EES programs always save all program information with the normal Save command. This capability is available only in the Professional version.

When the user clicks the Save button, a standard file save dialog will appear in which the name of the a file can be entered. The default name is the same name as the EES file with a filename extension of .VAR. The filename and directory information can be changed but the filename extension must be .VAR in order for EES to recognize the file. If confirmed, a file containing the current values of the all EES variables, including the current values of inputs, is written to the disk. This .VAR file can later be loaded with the Load button. A standard file open dialog will appear in which a .VAR file can be selected. The Diagram windows will be updated with the values of the variables found in this file.

When opening one of the five EES files contained in the Distributable program, EES will look for a .VAR file in the same directory the Distributable program having the same parent name as the EES file. If a .VAR file is found with this parent name, automatically loaded and the input information on the Diagram windows will be updated.

The Save button is disabled until calculations have been successfully completed. In this way, it is not possible to write a set of input information that will not successfully calculate. Both the Save and Load buttons will be disabled if the Diagram window does not contain input variables.

By default, the Save button caption is "Save" and the Load button caption "Load". The captions can be changed by right-clicking on the button and providing the desired text while in development mode.

Plot Windows

Variables which appear in the Parametric, Lookup, Array or Integral tables may be plotted with the New Plot Window or Overlay Plot commands in the Plot menu. In addition, plots of the thermodynamic properties can be generated using the Property Plot command. All plots appear in the Plot Window, which is equipped with tabs at the top of the window to allow easy identification and access to each plot. There is no limit on the number of plot windows that can be constructed (other than available memory), and each may have any number of overlayed plot lines. There are many plotting options such as choice of line type and plot symbol, linear/logarithmic scaling, spline fitting, tick frequency, and grid line control. These options can be set when the plot is first drawn or at a later time using the Plot window controls described below or the Modify Plot and Modify Axes commands in the Plot menu.

A tool bar is provided on the Plot Window with buttons to add text, lines, boxes, and ellipses. One of the button on the tool bar is an alignment button that allows all selected items to be aligned in a specified manner. The visibility of the tool bar is controlled by the Show/Hide Tool Bar menu item in the Plot menu. Right-clicking the mouse within the plot window (but not on any plot item) will also toggle the visibility of the tool bar.

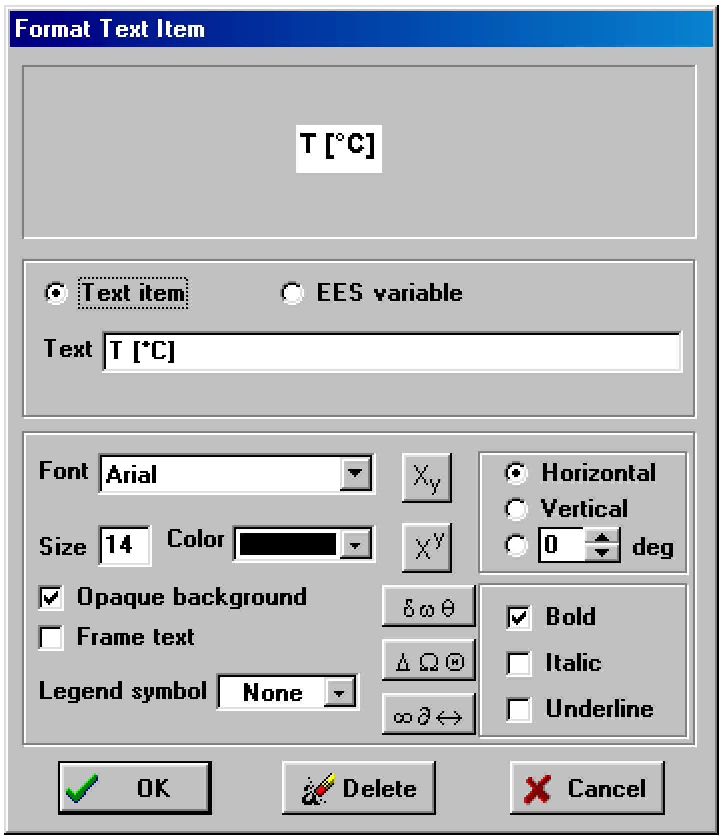
The appearance of the plot can be changed in many ways using the plot menu commands in the Plot menu and controls in the Plot window. The Plot window controls are as follows.

Moving the Plot

The entire plot, including the axis scales and all text items, can be moved to a different location in the Plot window by holding the mouse button down in any location within the plot rectangle (but not on a text item) while sliding the mouse to its new location. An outline of the plot will move with the cursor and the plot will move to the new location when the button is released.

Adding / Changing Text and Text Characteristics

A text item can be added to the plot window by clicking on the text button in the plot window toolbar. When clicked, the Format Text dialog shown below will appear. Two basic types of text items can be created: plain text and EES variables. The choice is made by checking the corresponding radio button at the top of the dialog. Plain text is used to identify fixed information displayed in the plot or to produce a legend. The EES variable option provides the same capability with the difference being that the text can include the name, value and units of a pre-defined EES variable and this information can optionally updated dynamically. The text is entered or edited in the text edit field and it is displayed as it will appear on the plot at the top of the dialog.. Controls are provided to change the font, size, color and style of the text item as well as the text itself. Subscripts and superscripts can be generated using the 'speed' buttons. This is done by selecting the text that is to be subscripted or superscripted and then clicking the button. Three additional speed buttons are provided below the subscript button to facilitate entry of lower or upper case Greek letters and useful symbols. The symbol entry buttons operate as follows. Press and hold the mouse button down on one of the three symbol buttons. A palette of symbols will be displayed to the right of the button. While holding the mouse button depressed, slide the mouse cursor to the symbol palette and position it over the symbol you wish to select. When the mouse button is released, that symbol will be added to the text item at the current insertion point.


Format Text Item

T [°C]

[✓ Text item] EES variable

Text: T [°C]

Font: Arial

Size: 14

Color: ■

[✓ Horizontal] Vertical 0 deg

☑ Opaque background

☐ Frame text

Legend symbol: None

☑ Bold

☐ Italic

☐ Underline

EES allows any horizontal text item to be associated with a plot symbol to facilitate construction of a legend. Clicking in the Legend symbol box will produce a drop-down list containing a descriptor of each existing plot. If a plot is selected, the line type and symbol used for that plot will be displayed just to the left of the text item and it will move when the text item is moved.

When you click on a text item in a plot window, a red dotted box will be drawn around it to indicate that this item is selected. You can select multiple text items by holding the Shift key down as you click on the text items. Double-clicking the left mouse button (or clicking the right mouse button) on a text item will bring up a dialog window in which the characteristics of all selected items can be changed. If only one text item is selected, the Format Text Item dialog window shown above will appear displaying the text and its current characteristics. When more then one text item is selected, the Format Text Item dialog window displays the characteristics of the first selected text item. Any change made to a text characteristic in the dialog window will be applied to all selected text items. If a text characteristic, such as the font size, is not changed in the dialog window, then the font size of the selected text items will not be changed. In this way, you can change just the font without affecting other characteristics such as the size or color. The selected items can be copied or cut using the Copy or Cut commands. The Paste command can then be applied to move them from the clipboard to this plot or to any other EES plot window. The Delete key provides the same function as the Cut command. If you accidentally delete text or lines, use the Paste command to restore them.

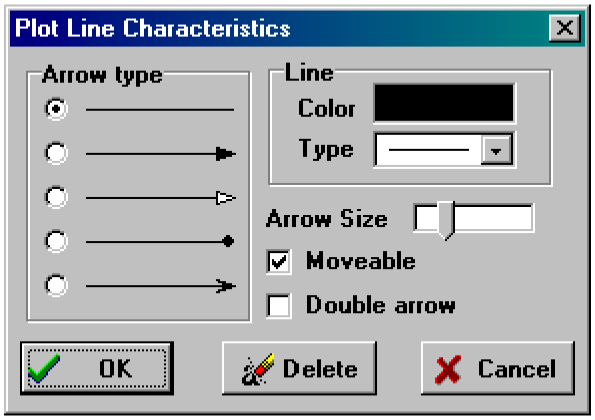
Selected text items can be copied to the Clipboard using the Cut and Copy commands in the Edit menu. The Delete key will cut all selected text items. Once on the Clipboard, the Paste command can be applied to move the text item from the Clipboard to any EES plot window.

Moving Text
Text fields to label the X and Y axes are automatically generated when a plot is constructed. Additional text items can be added to the plot using the Add Text command in the Plot menu. You can move any text item or set of selected text items to any position on the screen by pressing and holding the left mouse button down while the cursor is on the text item and dragging it to its new location. (Use the Shift key to select more than one text item.) Selected text items can also be moved using the arrow keys. Clicking the left mouse button on a text item will select it. Selected text items are displayed within a red dotted rectangle. Each arrow key press moves all selected text one pixel in the arrow direction.

Adding Lines and Arrows, Rectangles and Circles
The Add Line toolbar button in the plot window toolbar allows lines or arrows to be placed anywhere within the Plot window. After clicking on the line button in the toolbar, the cursor will appear as a cross. Press and hold the mouse button down at the position where you wish the line to begin. Hold the mouse button down while moving the mouse to the desired end position of the line. If you hold the Shift key down, the line will be forced to horizontal, vertical, or the nearest 45° angle. Release the mouse button to complete the line construction. Initially, a line will be created of the type that was last chosen. If you wish to change the type, for example to add or change an arrowhead, double-click or right-click on the line. A dialog window will appear in which the arrowhead type, line thickness, and color may be selected. By default, the line can be resized, moved, or rotated. If you wish to prevent these capabilities, deselect the Moveable check box. The current line and following lines created with the Add Line toolbar button will then have the same characteristics as the previous line. Rectangles and circles can be drawn in a similar fashion using the rectangle and circle buttons on the plot window toolbar.

Moving Lines and Arrows, Rectangles and Circles
Clicking on a line will cause it to be displayed with small boxes at both ends. Multiple lines can be selected by holding the Shift key down. Double-clicking the left mouse button (or clicking the right mouse button) will cause the Modify Line dialog window to appear in which the characteristics (such as arrow style, arrow size, line thickness, and color) of all selected lines can be changed at one time. Selected lines can be copied to the Clipboard using the Cut and Copy commands in the Edit menu. The Delete key will cut all selected lines. Once on the Clipboard, the Paste command can be applied to move the line from the Clipboard to any EES window of the same type. Rectangles and Circles can be moved and copied in the same manner.

You can move, resize, or rotate the line after it is created. To move the line, press and hold the mouse button down anywhere near the middle of the line while dragging it to its new position. You can also use the arrow keys to move one or more selected lines. When you select a line by clicking on it, small boxes will be displayed at either end of the line. To rotate the line and/or change its length, press and hold the mouse button within either of the two small boxes. Move the end to its new position and release the mouse button. Holding the Shift key down while rotating the line will force it to be horizontal, vertical, or on a 45° angle. Holding the Shift key down while resizing a rectangle or circle will force the width and height to be the same size.

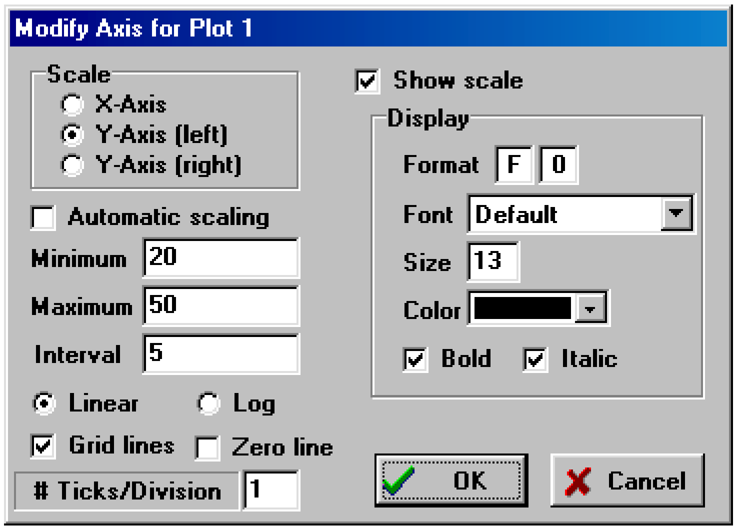


[✓] Moveable
[ ] Double arrow

Resizing the Plot
The size or aspect ratio of the plot can be changed by pressing and holding the left mouse button with the cursor located within the resize area at the lower right corner of the plot rectangle. If the tool bar is visible, the resize area is marked with three 45° lines. The cursor will change from an arrow to the resize indicator when it passes over the resize control. The size of the plot will change as you drag the lower right corner to a new position. When the plot is resized, the size and positions of all text items and lines are proportionally changed. (You can prevent the text font size from changing if you hold the Ctrl key down while resizing the plot.)

Modifying the Axis Information

The axis scaling and appearance can be changed by double-clicking the left mouse button on the abscissa or (left or right) ordinate scales or by selecting the Modify Axes menu item in the Plot menu. Either action will bring up the Modify Axes dialog window. The axis for which the changes are to be made is indicated by the radio-button controls at the upper left. By default automatic axis scaling if off and the scaling of the selected axis is controlled by the minimum, maximum and interval values shown in the edit boxes. These values can be changed as desired. However, if the 'Automatic scaling' checkbox is checked then the axes will be automatically scaled as appropriate to contain plotted data for all lines in this plot window. In this case, the edit boxes for the minimum, maximum and interval values will be disabled. If a new plot line is added or if the data in one or more of the plots are changed such that one or more points are outside the range of the axis scale, the axis scale will automatically change. Scale numbers are placed at the position of each interval, as are Grid lines if selected. Selecting the Zero line causes a vertical (for x-axis) or horizontal (y-axis) line to be drawn at a value of zero. The #Ticks/Division is the number of minor ticks, i.e., the number of tick marks between each interval. If Grid lines is selected, clicking on #Ticks/Division will change it to #Grids/Division allowing grid lines to be placed at points in between the major ticks. If the Show Scale control is selected (as shown), the scale numbers will be displayed. The characteristics of these numbers are controlled by the remaining fields on the right-side of the dialog window.



Modify Axis for Plot 1

Scale: X-Axis [✓ Y-Axis (left)] Y-Axis (right)

Automatic scaling: ☐

Minimum: 20

Maximum: 50

Interval: 5

[✓ Linear] Log

[✓ Grid lines] ☐ Zero line

Ticks/Division: 1

[✓ Show scale]

Display

Format: F 0

Font: Default

Size: 13

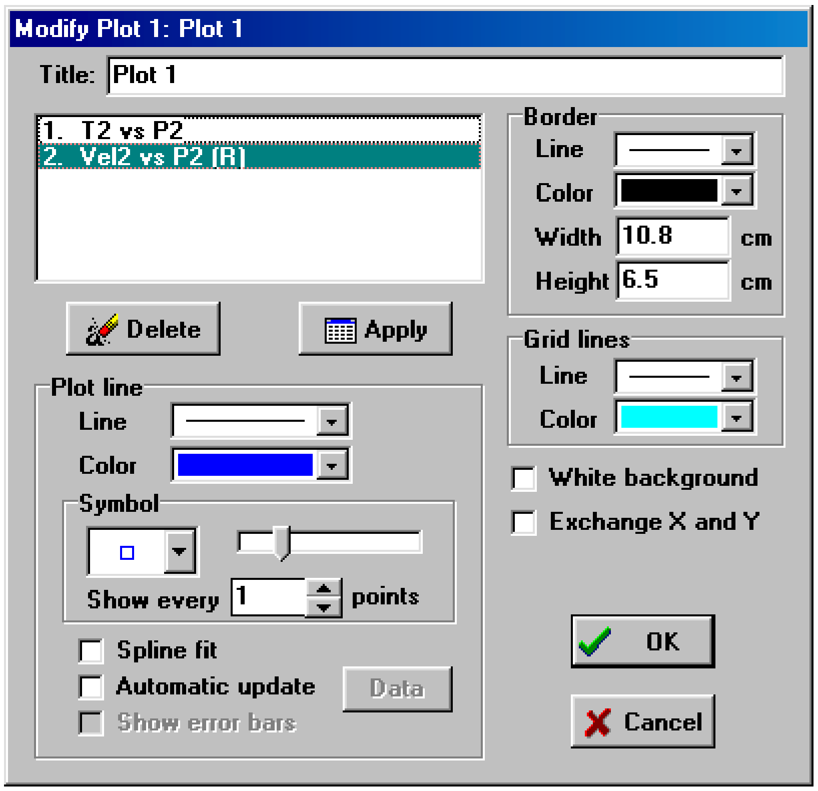
Color: ■

[✓ Bold] [✓ Italic]

OK Cancel

Modifying the Plot Information

The line type, color, plot symbol (or bar type for bar plots), and other information relating to each plot can be viewed or modified by double-clicking the left mouse button anywhere within the plot rectangle (but not on text or a line.) The dialog window shown below will appear. This dialog window can also be made to appear with the Modify Plot menu item in the Plot menu. All current plots in the current plot window will be listed in the rectangle at the upper left in the order in which they were constructed. An (R) to the right of the plot name indicates that the plot uses the right-hand y-axis. Select the plot by clicking on its name.


Modify Plot 1: Plot 1

Title: Plot 1

1. T2 vs P2
2. Vel2 vs P2 (R)

Border
Width 10.8 cm
Height 6.5 cm

Plot line

Symbol
Show every 1 points

- [ ] Spline fit
- [ ] Automatic update
- [ ] Show error bars
- [ ] White background
- [ ] Exchange X and Y

The Spline fit control, if selected, will cause EES to plot the line using cubic splines to produce a smooth curve through the data. Automatic Update sets up a direct link between the plot and the data in the Parametric Table so that the plot will be automatically redrawn if any change is made to the data in the Parametric Table. Clicking the Data button to the right of Automatic Update allows the characteristics of the link (e.g., row range, X and Y axis variables) to be changed. The Show error bars control is enabled only if the data being plotted were obtained with the Uncertainty Propagation Table command in the Calculate menu. Click the Apply button if you wish to view the changes you have made.

Aligning Items on the Plot Window

The Align button on the Plot Window toolbar is enabled when two or more text items, lines, rectangles, or ellipses are selected. These objects are created with the Add Text, Add Line, Add Rectangle, and Add Ellipse toolbar buttons. After selecting this button, a small dialog window will appear showing alignment choices. Click OK to proceed with the alignment of the selected items.

Crosshairs

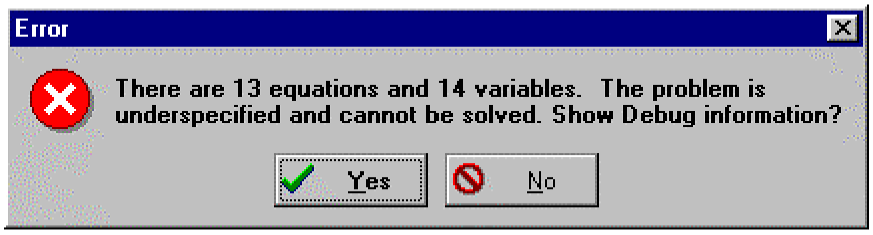
Holding the ***Ctrl*** key down will change the cursor into crosshairs. The coordinates at the intersection of the crosshairs can be viewed in the plot window title bar.

Debug Window

The Debug Window displays diagnostic information of three types: Incorrect Degrees of Freedom, Constrained Solution, and Unit Checking Report. Each of these capabilities is described below.

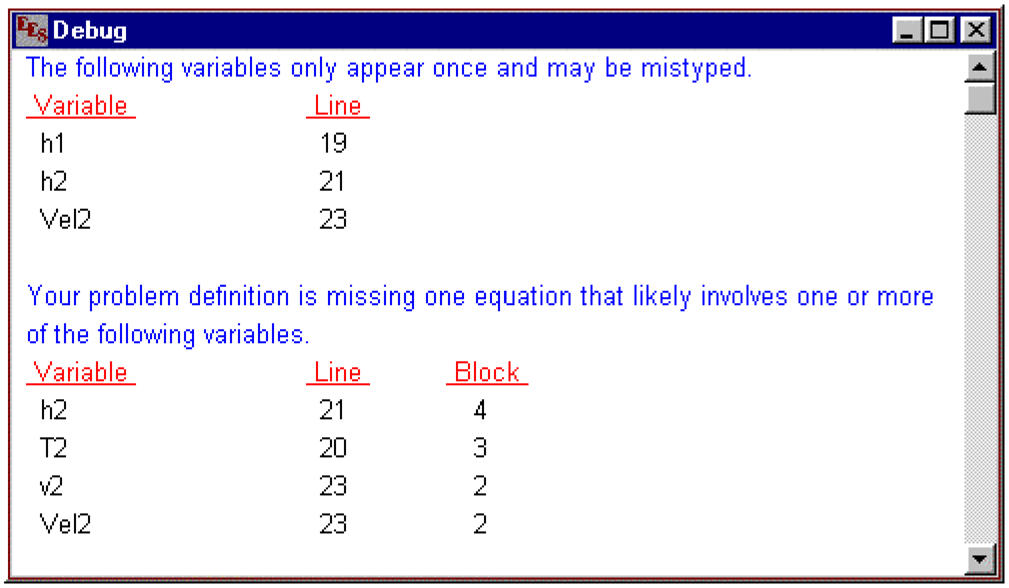
Incorrect Degrees of Freedom

Whenever an attempt is made to solve a set of equations in which the number of equations is not equal to the number of variables, a message box such as that shown below will appear.



There are 13 equations and 14 variables. The problem is underspecified and cannot be solved. Show Debug information?

Clicking the Yes button will bring up the Debug window. For example,



The following variables only appear once and may be mistyped.

Variable	Line
h1	19
h2	21
Vel2	23

Your problem definition is missing one equation that likely involves one or more of the following variables.

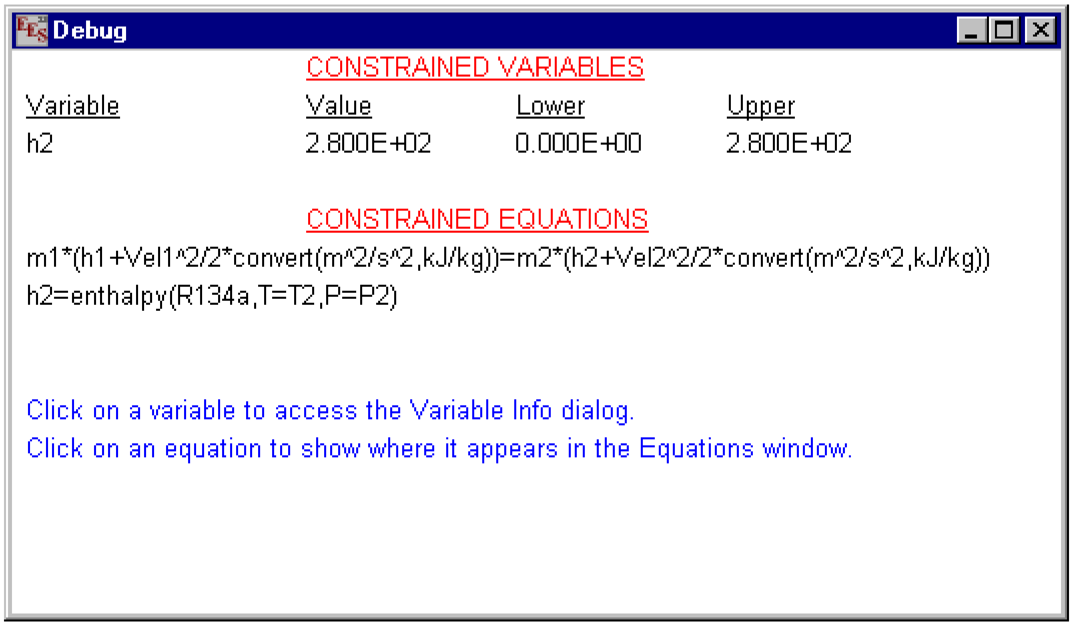
Variable	Line	Block
h2	21	4
T2	20	3
v2	23	2
Vel2	23	2

This window provides two lists of variables. The first list shows all variables which are referenced only once in the Equations Window. These variables are possibly spelled wrong or otherwise not being directly used in the problem, except for informational purposes. The second list shows the variables which are most likely to be involved in any missing or extra equations. The information used to construct this second list is determined by examination of the blocking order of the equations in the Residuals window. You may also find the information in the Residuals window helpful in identifying the problem with your equation set.

Clicking the left mouse button on a variable name in the Debug window will bring the Equations Window to the front with that variable name selected.

Constrained Solution

In some cases, a problem cannot be solved to within the specified tolerances because the lower and/or upper bounds one or more variables have restricted the solution. If this happens, EES will present a dialog box warning of this problem and offer more information. If the user wishes to see the additional information, the Debug window will appear similar to that shown below showing the variables that are constrained and the affected equations. Clicking the left mouse button on a variable name will open the Variable Info dialog window in which the lower or upper bound can be changed. Clicking on an equation will move the cursor to that equation in the Equations window.



CONSTRAINED VARIABLES

Variable	Value	Lower	Upper
h2	2.800E+02	0.000E+00	2.800E+02

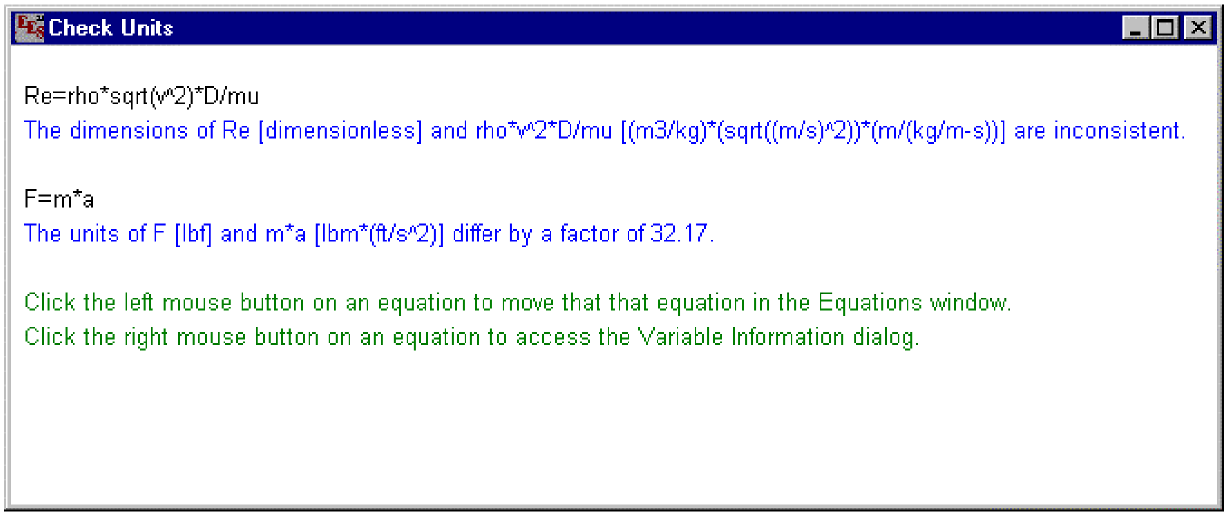
CONSTRAINED EQUATIONS

```
m1*(h1+Vel1^2/2*convert(m^2/s^2,kJ/kg))=m2*(h2+Vel2^2/2*convert(m^2/s^2,kJ/kg))
h2=enthalpy(R134a,T=T2,P=P2)
```

Click on a variable to access the Variable Info dialog.
Click on an equation to show where it appears in the Equations window.

Unit Checking Report

The Check Units command in the Calculate menu will display its finding in the Debug window. Each equation that is found to have a dimensional or unit inconsistency will be displayed (in black) followed by an explanation (in blue). Clicking the left mouse button on the equation will jump the focus to offending equation in the Equations window. Clicking the right mouse button on the equation will bring up an abbreviated form of the Variable Information dialog window showing only the variables in that equation.



```
Re=rho*sqrt(v^2)*D/mu
The dimensions of Re [dimensionless] and rho*v^2*D/mu [(m3/kg)*(sqrt((m/s)^2))*(m/(kg/m-s))] are inconsistent.

F=m*a
The units of F [lbf] and m*a [lbm*(ft/s^2)] differ by a factor of 32.17.

Click the left mouse button on an equation to move that that equation in the Equations window.
Click the right mouse button on an equation to access the Variable Information dialog.
```

The units that are associated with each variable can be entered in a number of ways. A direct way of entering units is to type them directly into the Variable Info dialog (Options menu). It may be somewhat more convenient to enter the units of each variable in the Solution window. Left-clicking will select or unselect a variable. Right-clicking will bring up a dialog window in which the units of all selected variables can be modified at once. Units of constants can be declared directly in the Equations Window by following the equation with a comment with the units provided in comments. For example, the following equation will set the Q_dot to 1000 with units of Btu/hr.

Q_dot = 1000 “[Btu/hr]”

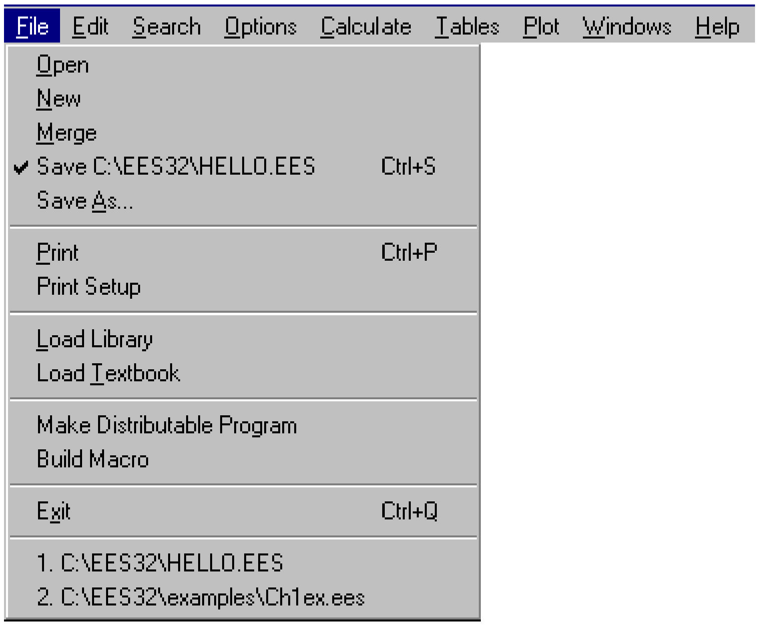
The units of variables in the Parametric and Arrays tables can be changed by right-clicking on the column header. The Default Variable Info command (Options menu) allows the units (and other information) to be set based on the first letter of the variable name. The Professional version allows .var files to be saved from the Variable Info dialog which save the units and all other information. Opening a .var file using the Read Var File button in the Variable Info dialog or using a $Include directive sets the variable information for all variables that have been previously saved.

The units that EES recognizes can be displayed using the Unit Conversion Info command (Options menu). Note that the Check Units command uses the dimensional information contained in the Units.txt file which exists in the directory that the EES program is located. You can add new units or modify information in this file using any text editor.

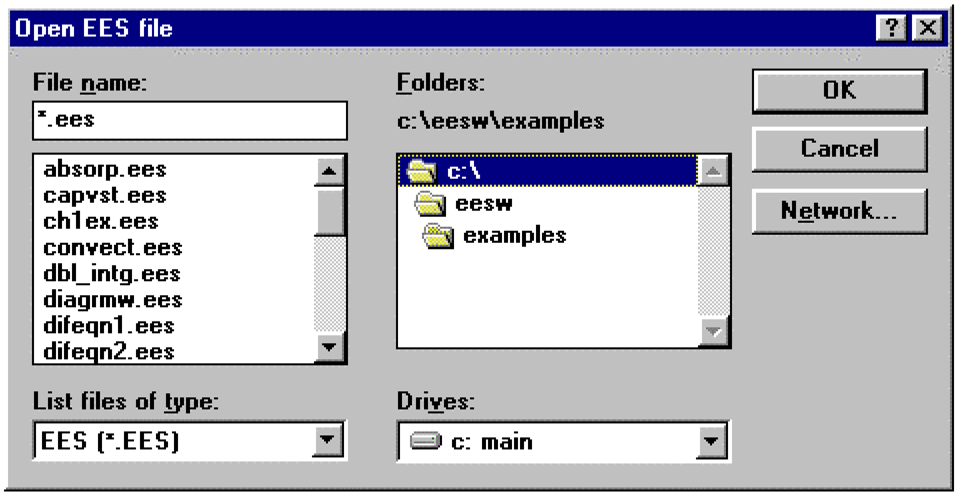
CHAPTER 3

Menu Commands

The File Menu



Open will allow you to access and continue working on any file previously saved with the Save or Save As… commands.



After the confirmation for unsaved work, the dialog window shown above will appear. The current directory is indicated below the text Folders: and EES files in that folder (directory) are shown in the list on the left. To select a file, click on the file name in the list or enter the filename in the File Name: edit box. You can open files in another directory by entering the directory name in File Name: field or by clicking on the folders listed in the Folders list. Clicking on the Drives list displays the available drive designations. Click on the drive name to select it. Choose the OK button to select the file (or directory) displayed in the Filename field.

EES can read four types of files that are identified as **EES file**, **Import file**, **Text file**, and **Library file** formats. The format is selected with the drop-down list at the bottom left of the dialog window. EES files with a .EES filename extension are the norm. Import files with a .XPT extension are files saved by EES with the Export option from a different operating system, such as the Macintosh. Text files with a .TXT filename extension contain ASCII text which is read into the Equations window. Library files are EES files containing one or more functions, procedures or modules which can be automatically loaded at startup, as described in Chapter 5.

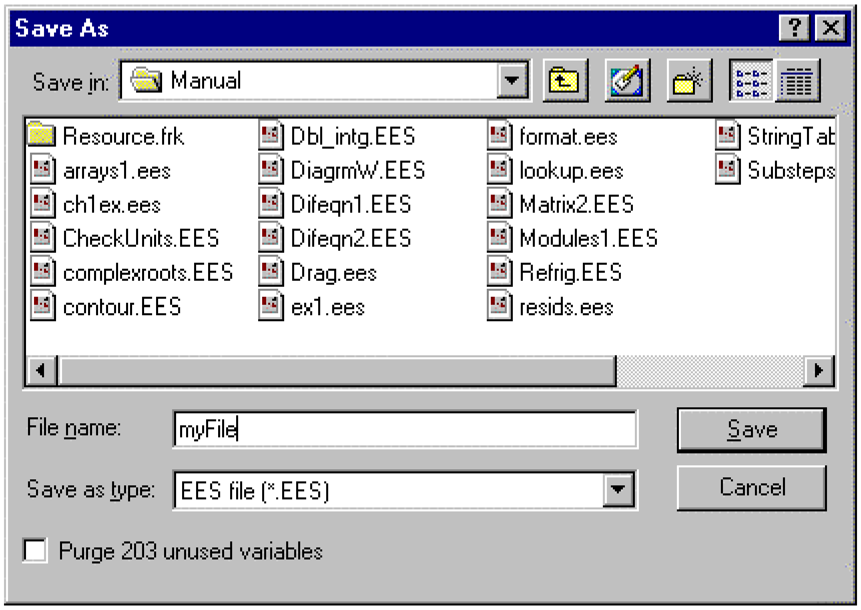
New initiates a new work session. All variables and equations will be cleared. The unit system will be restored to the settings that would be in effect if the program were restarted. If an unsaved problem definition exists, you will be asked if you first wish to save your current problem information.

Merge allows the equations previously saved in an .EES file to be merged with the current contents of the Equations window at the cursor position. The Merge dialog window operates in the same manner as for the Open command. Equations can also be entered from a text file using the $INCLUDE directive. EES functions, procedures and modules can be loaded using the Load Library command or $INCLUDE directive.

Save will save your problem definition with the same file name (which appears after Save in the File menu and in the title bar of the Equations window) as it was last saved. For a new work session which has not yet been named, you will be prompted to supply a file name, just as if the Save As… command were given. All information concerning the problem definition is saved, including the equations, variable information, the tables, the plots, and the size and locations of the windows. By default, the file will be saved in the standard EES file format with a .EES filename extension. If you wish to export the file to a version of EES on a different operating system, use the Export format in the file type available in the Save As… command. A check mark will appear to the left of the word Save in the File menu if the current problem information has been saved on disk. Any change in the problem information will cause the check mark to disappear.

Save As… provides the same function as the Save command except that it will first prompt you to supply a filename in the Save File dialog window. The Save As command allows the problem definition to be saved with another filename or in a form which can be exported to EES versions on other operating systems. Enter the file name of your choice in its place. The 32-bit version of EES supports long file names. The file name may include drive and directory information. However, it is not necessary to enter a filename extension, since EES will supply the extension automatically.

EES recognizes four file types. If EES is displayed in the Type box at the lower left, the extension in the File Name: field will be set to .EES (the norm) and files having this extension will be displayed in the filenames list. The Export file type will apply a .XPT filename extension and save the file in a generic ASCII format which can be transferred to other operating systems such as the Macintosh. The Text file type will apply a .TXT filename extension and save only the text in the Equations window in an ASCII file. The Library type will change the filename extension to be .LIB. Each time EES is started, it opens all of the .LIB files in the USERLIB\ sub-directory and automatically loads the functions, procedures, and modules in these files. These functions can be used exactly like the EES built-in functions. Library files are one of the most powerful features of EES because the user can easily develop special purpose functions. See Chapter 5 for additional information.



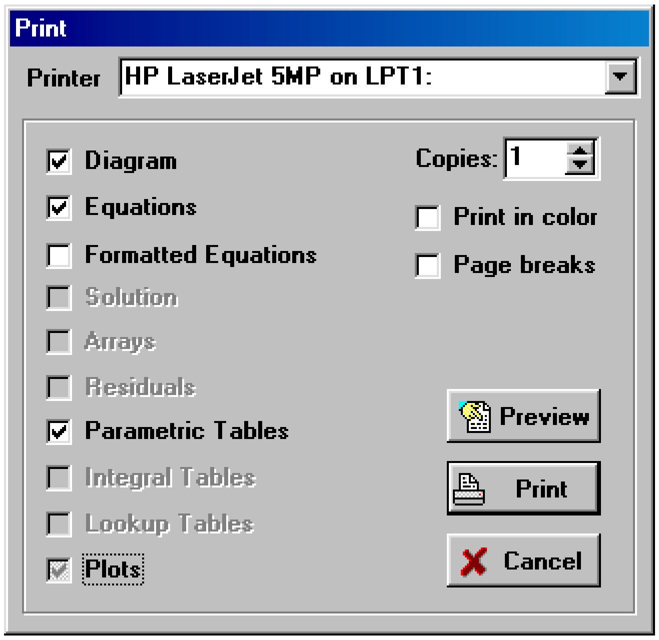
File name: myFile

Save as type: EES file (*.EES)

☐ Purge 203 unused variables

EES normally saves every variable defined during the work session regardless of whether or not it currently is in use when the Save or Save As command is issued. For example, if you define a variable X with the equation X=6 and later decide that the equation should read X1=6, EES will still keep the variable X and all of its characteristics in memory, e.g., guess value, bounds, units, and format. EES can store up to 6000 variables. For very large problems, particularly problems using array variables, EES could run out of space. In cases when there are many variables that are not in use, EES will display a check box in the Save As dialog with the caption "Purge XXX unused variables" as shown above. To save only those variables which are currently in use, click this checkbox. When purging variables, it is advisable to save the file with a new name. In this case, the original file can be recovered if a problem develops.

Print will print any or all of the EES windows to the printer or to a file on the disk. Each window has a small check box preceding its name. If the check box is grayed (as it is for the Arrays window in the Print dialog window shown below) the window is not available for printing. If an X appears in the box, the window will be printed. To place an X in the box or to remove an existing X, click the mouse while the cursor is positioned in the box.


Print

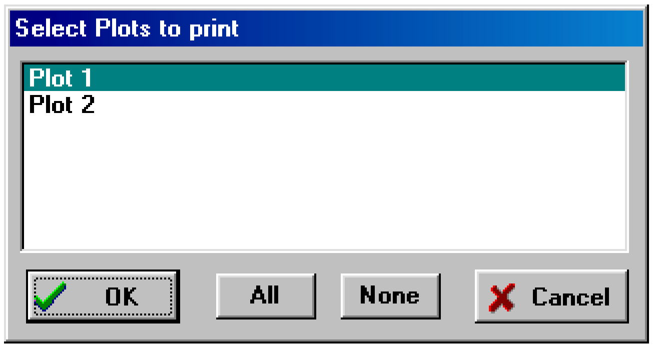
Printer: HP LaserJet 5MP on LPT1:

- [✓] Diagram
- [✓] Equations
- [ ] Formatted Equations
- [ ] Solution
- [ ] Arrays
- [ ] Residuals
- [✓] Parametric Tables
- [ ] Integral Tables
- [ ] Lookup Tables
- [✓] Plots

Copies: 1

- [ ] Print in color
- [ ] Page breaks

Some windows, such as the Plot Window and the Parametric and Lookup Table windows, may contain many sub-windows, accessible by tabs at the top of the window. For these windows, clicking in a check box will bring up a small supplementary window to select the plots or tables that are to be printed.


Select Plots to print

Plot 1
Plot 2

The printed output will be sent to the selected printer. The drop-down box at the top of the dialog allows the printer selection to be changed. It is possible to direct the output to a file, rather than a printer, with the Connect options in the Printers applications. See the

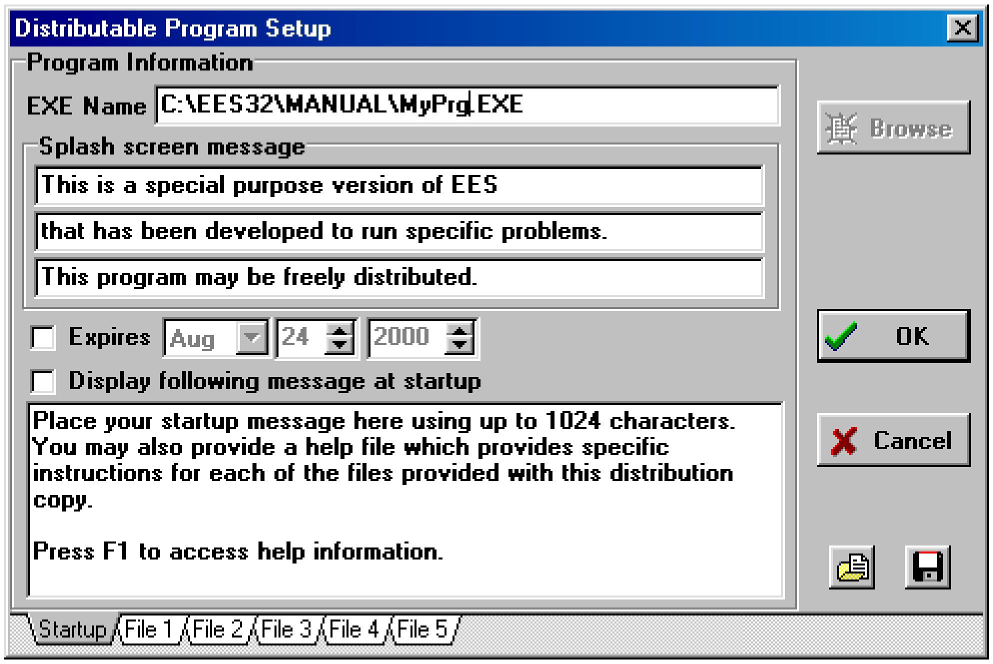
Windows manual for additional information on selecting printers. Printing options such as the font, line spacing, and font size are set in the Preferences dialog window (Options menu). If the “Print in color” check box is selected, EES will attempt to print selected windows using the same colors as appear on the screen. Colored text may not print clearly on some black and white printers. When this control is deselected, all printing will occur in black and white. If the “Page breaks” check box is selected, a forced page break will occur so that the printed output for each window starts on a new page. The Preview button will direct a facsimile of the printed output to the screen.

Printer Setup provides a dialog window that allows the printer to be selected along with printing options, such as the paper size and orientation. The printer selection can also be made in the Print dialog window.

Load Library will bring up the standard open file dialog showing EES Library files (which have a .LIB file name extension) in the file selection box. Library files contain user-supplied functions, procedures, and/or modules which operate as described in Chapter 5. Once loaded, these library files remain in memory until EES is closed. Note that when EES starts, it will preload all of the library and externally-compiled files which are found in the USERLIB\ sub-directory so the Load Library command is not needed for these files. The Load Library can also be used to load external functions and procedures with filename extensions of .DLF, .DLP, and .FDL. See Chapter 6 for additional information. The $INCLUDE directive described in Chapter 7 can also be used to load library files.

Load Textbook reads a user-generated Textbook index file with the filename extension (.TXB) and uses the information in this file to create a Textbook menu at the far right of the menu bar. A textbook index file can also be loaded automatically by placing the textbook index file and associated problem files in a subdirectory within the USERLIB subdirectory The Textbook menu provides convenient access to set of problems, such as those developed for use with a text. The Textbook menu section at the end of this chapter provides instructions for creating and using a Textbook menu.

Make Distributable Program *(Professional version only)* will create a special purpose version of EES which will run one to five pre-selected problems. The following dialog will appear when this menu command is chosen. A special version of the EES program with one to five EES problems and all supporting files are placed in a single executable file when you select the OK button. This executable file can be freely distributed to others. An expiration date can be optionally set which will prevent the program from being used thereafter. This Make Distributable Program command is available only in the Professional version.



EXE Name C:\EES32\MANUAL\MyPrg.EXE

Splash screen message

This is a special purpose version of EES

that has been developed to run specific problems.

This program may be freely distributed.

☐ Expires Aug 24 2000

☐ Display following message at startup

Place your startup message here using up to 1024 characters. You may also provide a help file which provides specific instructions for each of the files provided with this distribution copy.

Press F1 to access help information.

When the distributable program is started, the first file will automatically be loaded by default. However, any one of the five files can be selected as the file that appears at startup by providing /# as a parameter where # is an integer between 1 and 5. For example, entering MyPrg.exe /2 in the Windows Run dialog will start the distributable program and display EES file 2. The other files can be selected from the recently-accessed file list at the bottom of the File menu. The names that appear in the file menu are entered in the third edit field for each file. By default, the name is the filename without the path or filename extension. However, it can be edited to be any name that you wish to see presented in the File menu.

The remaining items in the dialog window are a series of check boxes that control the capabilities the user will have when running the distributable version. If you want you user to have the ability to view the equations window, click the 'Equations Window Visible' check box. You can make this window read-only by clicking the following check box. However, even if the user changes the equations, the changes cannot be saved with the distributable version. The advantage of this feature is that you can freely distribute any program you generate in EES to others.

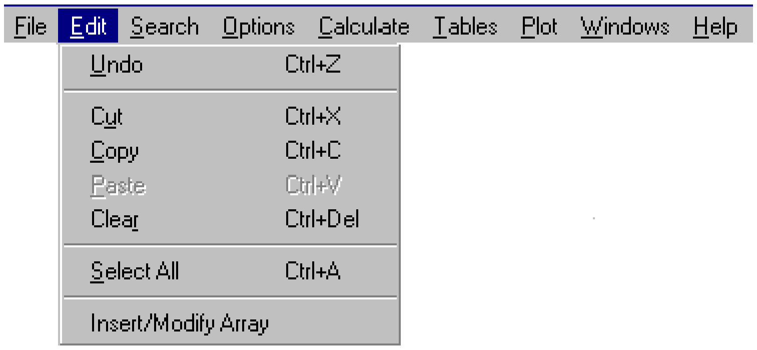
Two small buttons are provided just below the Cancel button in the Make Distributable dialog. These buttons allow saving and loading of scripting files to simplify the process of creating distributable programs. The button on the right saves all of the information in the Make Distributable dialog window into a file having a .MDI (Make Distributable Information) filename extension. The other button will open *.MDI files and fill all of the field in the Make Distributable dialog window using the information in the file.

Build Macro *(Professional version only)* will initiate the process of recording a series of EES instructions that can later be started from the Windows Run command or from a different program to replay all of the instructions in the Macro file. Used in this manner, EES can be directed to solve a set of equations in a specified text or EES file and put the solution into another specified text file without ever appearing on the screen. EES can also run the Macro file on demand by a Dynamic Data Exchange (DDE) command. Additional information concerning Macro files is provided in Chapter 7.

Quit provides a graceful way to exit the program.

The remaining items in the File menu are recently accessed filenames. Selecting any of these filenames opens the file. This list can be disabled in the Preferences dialog. It is recommended that this list be disabled if the program is installed and used on a network.

The Edit Menu



Undo restores the Equations window to the condition it was in before the last editing operation. If Undo is unavailable, the menu will be disabled.

Cut deletes the selected (highlighted) text or selected item. The deleted item is placed on the Clipboard where it can be pasted to another location with the Paste command.

Copy functions in a manner dependent on which window is foremost. Copy will place the selected text from the Equations window on the Clipboard from which it can be restored at the cursor position with the Paste command. When the Parametric, Lookup, or Array Table window is foremost, the Copy command will copy the selected cells (shown in inverse video.) The data copied from the table are stored on the Clipboard in a standard format in which numbers within the same row are separated by a tab and each row ends with a carriage return - line feed. Data in this standard format can be pasted into any location of the Parametric or Lookup tables or into other applications. (Hold the Shift key down when selecting the Copy command if you wish to also copy the column heading and units.)

Copy will move a selected Plot window or the Diagram window graphics into the Clipboard from which it can be pasted into other applications. EES copies the Plot window to the clipboard in two formats: as a picture and as a high-resolution bitmap. Most programs, such as a word processor, provide a Paste Special command in which you can select among the available formats that have been placed on the clipboard. The bitmap requires more memory, but it produces a high quality image when printed. The Picture format produces exactly the image that you see on the screen. If you are intending to print the image that you have copied and you are concerned with the print quality, use the Paste Special command and select the bitmap option.

If you hold the Ctrl key depressed while copying the Diagram window, the text items added with the Add Diagram Text command will not be included. The copy of the Plot or Diagram window is stored in MetaFilePict object format. Copy Solution will place the contents of the Solution window on the Clipboard both as ASCII text with each variable is on a separate line and as a picture of the formatted Solution window. Use the Paste Special in another application, such as a word processor, to select either the text or the picture for pasting. See the description of the Solution Window for more information. Copy will place the entire contents of the Residuals window on the Clipboard as text. Tabs separate the different items on each line of the Residuals windows. The Clipboard contents can be pasted into a word-processor.

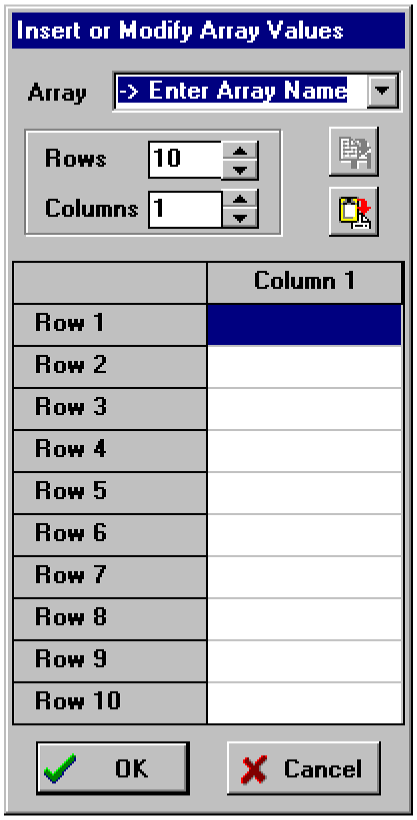
Paste is active for the Equations, Parametric, Lookup, Plot and Diagram windows. Paste moves the text (or graphics for the Diagram window) previously placed on the Clipboard with the Cut or Copy commands in EES or in other applications. When Paste is used in the Parametric or Lookup Table window, the values stored on the clipboard will be copied to the table starting in the cell in which the cursor is currently located. Data can thus be moved between the Parametric and Lookup tables. Copy and Paste can also be used with text items placed on the Plot windows.

Clear removes the selected text without placing a copy on the Clipboard. Clear can also be used to delete the contents of the Diagram window.

Select All will select all of the text in the Equations window, or all of the cells in any of the three tables, depending on which window is foremost when the command is issued. The Select All command is normally followed by Copy which places the selected items on the clipboard. If the Formatted Equations window or Solution window is foremost, the Select All command appears as Select Display. This command will select all currently visible items in the window.

Insert/Modify Array provides an easy way to enter or modify the values in EES Arrays. The dialog window is shown below. The dropdown edit rectangle at the top of the dialog displays the names of all of the arrays that are defined in the Equations window. Select the name of the array for which you wish to enter or modify values. If no arrays are defined, this rectangle will display -> Enter array name. Enter the name of the new array.

The number of columns and rows in the array can be adjusted with the Rows and Columns controls. Scroll bars will appear as needed to allow access to array cells that are not visible in the window. The two speed buttons provide access to Copy and Paste. The top button is the Copy button. It is enabled only if two or more cells in the table are selected. To select a range of cells, click the mouse in the upper left cell. Hold the Shift key down and click in the lower right cell. The Copy button should then be enabled . The Paste button is enabled only if text has been placed on the clipboard. To paste text into this window, click the mouse in the upper left cell for which the paste operation is to start. Pasting will proceed to the right and down the table. The copy and paste capability make it possible to import or export data to and from a spreadsheet.

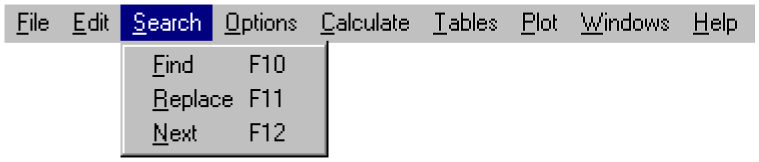


Enter the values of the array elements that you wish to define in the table. When you click the OK button, EES will convert the values that you have entered into EES equations and enter these into the Equations window. The entered will be placed within comments of the form:

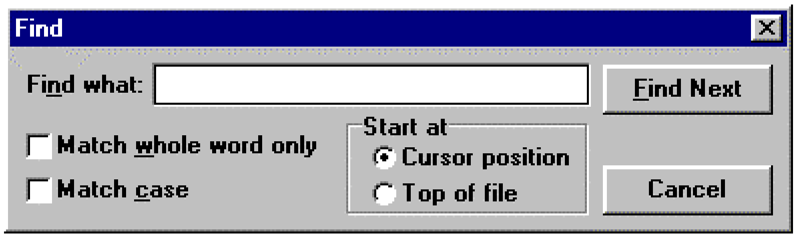
```
{Array A}
A[1]=1
A[2]=2
…
{Array A end}
```

where A is the array name. Do not delete or edit these comments. EES uses these comments to locate the position in the equations window in which the array elements are defined. If you later wish to redefine the array, EES will find the previous array using these comments and overwrite the array with the newly entered values.

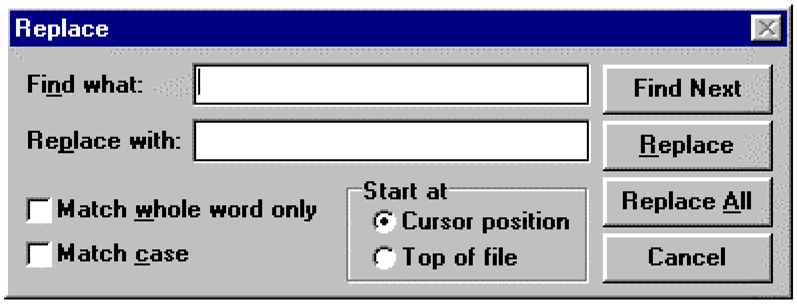
The Search Menu



Find will search the Equations window for the first occurrence of the text entered in the Find what: field. The search is case-insensitive unless the 'Match case' option is selected. If the 'Match whole word only' option is selected, the text will be found only if it is delimited by spaces or mathematical operators. The Cancel button will change to Done after the find process is completed.

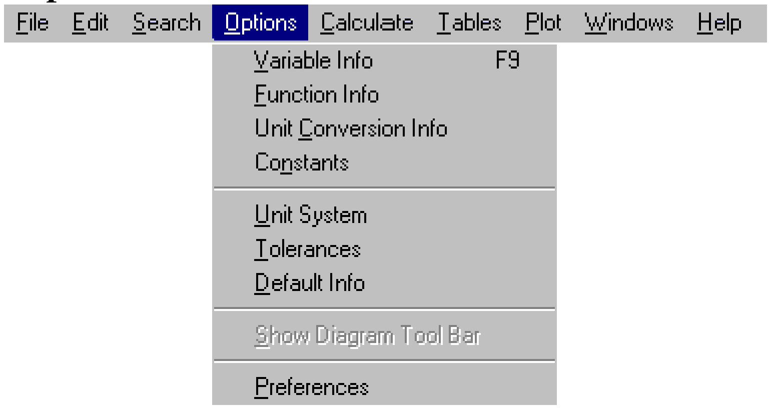


Replace will search the Equations window for the first occurrence of the text in the Find what: field and replace it with the text in the Replace with: field. The search options are as described for the Find command. The Replace All button will replace every occurrence of the search text with the replacement text. The Cancel button will change to Done after the find process is completed.

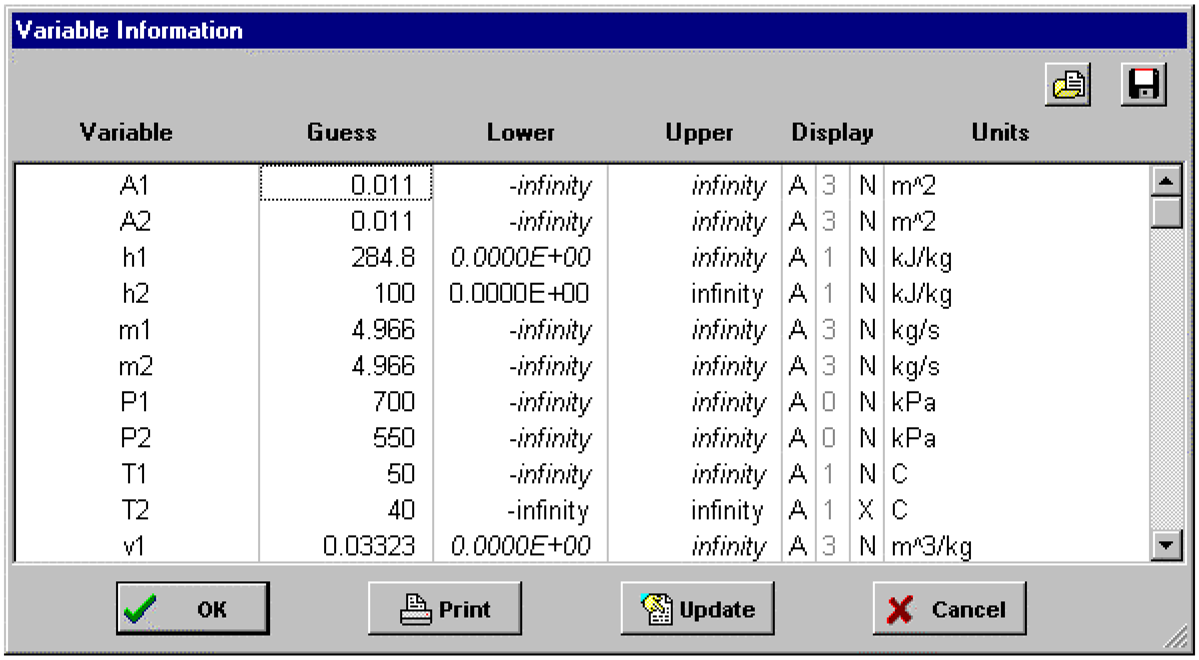


Next will find the next occurrence of the text previously entered with the Find or Replace command. The search options will be remain in effect if they were set in the Find command.

The Options Menu



Variable Info will provide a dialog window, as shown below, in which the guess value, lower and upper bounds, display format, and units of all variables currently appearing in the Equations window can be viewed and changed. These data are initially set to default values. The defaults, selected based on the first letter of the variable name, may be set with the Default Info command.



Variable	Guess	Lower	Upper	Display			Units
A1	0.011	-infinity	infinity	A	3	N	m^2
A2	0.011	-infinity	infinity	A	3	N	m^2
h1	284.8	0.0000E+00	infinity	A	1	N	kJ/kg
h2	100	0.0000E+00	infinity	A	1	N	kJ/kg
m1	4.966	-infinity	infinity	A	3	N	kg/s
m2	4.966	-infinity	infinity	A	3	N	kg/s
P1	700	-infinity	infinity	A	0	N	kPa
P2	550	-infinity	infinity	A	0	N	kPa
T1	50	-infinity	infinity	A	1	N	C
T2	40	-infinity	infinity	A	1	X	C
v1	0.03323	0.0000E+00	infinity	A	3	N	m^3/kg

If the program contains one or more modules (see Chapter 5 for a discussion of modules), a control will be provided at the top of the dialog to select the module or main program. The information for the selected module can then be changed or viewed.

The "Show Array Elements" checkbox control will be visible at the upper left of the dialog window if there any arrays are used in the Equations window. When this control is selected, all array elements appear in the Variable Info dialog and the guess value, bounds, display format, and units can be set for each individual element as for any other variable. However, when the control is not selected, all arrays elements are represented by a single entry. For example, X[ ] represents all array elements having the parent name X. If any of the characteristics for a parent array are changed, that change is applied to ALL array elements. Changing the guess value for X[ ] will result in the new guess value being applied to all array elements in array X. However, other characteristics, such as the bounds and units, would not be affected. In addition, the array name can be changed by editing the name in the first column of the Variable Info dialog.

Use the scroll bar on the right side of the dialog window to bring the variable information into view. Note that the height and width of this dialog window can be changed by selecting an edge and dragging it in the usual Windows manner. All fields, including the variable name, may be changed as required. If the variable name is changed, EES will change every occurrence of the original variable name in the Equations, Parametric table and Diagram windows.

The words, -infinity and infinity can be used to indicate unlimited lower and upper bounds, respectively. The Guess, Lower and Upper value fields will also accept a variable name, as well as a number. When a variable name is provided, EES uses the current value of that variable as the guess value or bound.

EES attempts to solve equations having a single unknown before this display appears. Variables for which the value has been pre-calculated are identified by having the bounds shown in italics. The pre-calculated value appears in the Guess column. These guess values and bounds may still be edited which will then cause EES to recalculate the value.

The display format of a variable in the Solutions or Table window is controlled by the three fields in the Display columns. Clicking in these fields will produce a pop-up menu for the display style, number of significant digits, and hilighting effects.

The units of the variable (or any other desired information) may be entered in the units column. Units are used by EES only for display purposes in the Solution and Parametric Table windows. The Convert function described in Chapter 4 can be used to convert units. Note that the display format and units of each variable can also be changed by clicking on the variable in the Solution Window.

When the OK button is pressed, *all* changes made to the variable information since the dialog window appeared are accepted. The Update button replaces the guess value of each variable with its current value, i.e., the value determined in the last calculation. The same update feature is provided with the Update Guesses command in the Calculate menu. The Print button will direct a copy of the information in this table to the selected printer. The Cancel button will restore all fields to the condition they were in when the Variable Information dialog was first presented and then remove the dialog.

The Variable Info dialog window can be resized by dragging the lower right corner.

Professional Version

Variable information for a module or main EES program can be saved to or loaded from a disk file. The file operations are accomplished using the two small buttons at the upper right of the Variable Information dialog.

Variable information is saved in a file having a .VAR filename extension. However, the information in this file is in tab-delimited ASCII format so that it can be opened, viewed or modified with a word processor or spreadsheet.

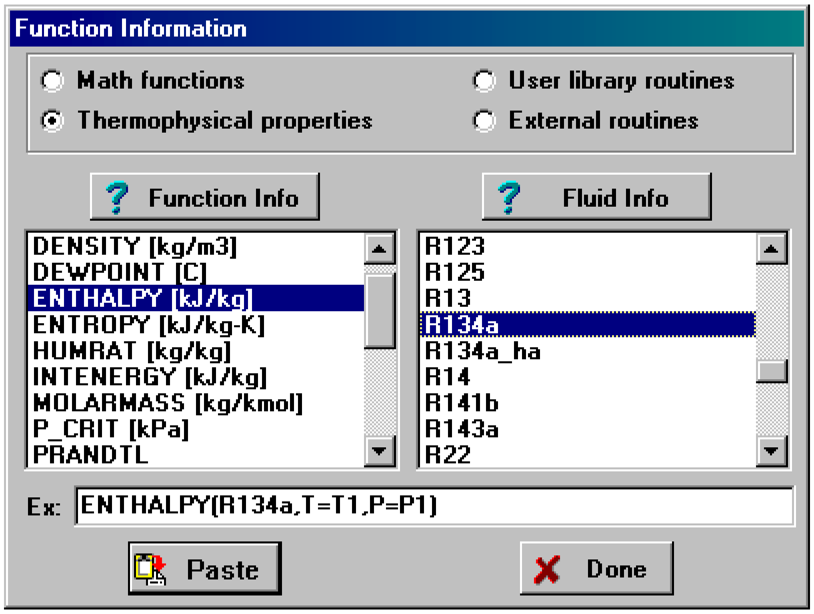
Variable information data can be saved to an existing .VAR file. In this case, the file is updated with the current information for the variables that are in use. Information in the file concerning variables that are not currently in use is not modified or deleted. For example, suppose you have a program that uses variables X, Y, and Z. You save the variable information to an existing file that already has variable information for variables A, B, C, and X. The information for variables A, B, and C is not altered. The variable information for X is updated to the current settings and variable information for Y and Z are appended to the file.

If a program contains one or more modules, a dropdown list appears at the top center of the Variable Info dialog from which the module or main program can be selected. In this case, the Save or Open Variable Information operation will be applied only to the selected module or main program. Variables that are in a module are stored in the variable information file with the module name so that the same variable names can be used in modules and the main program with no confusion.

There are several important applications for variable information files. The first application is to provide different sets of guess values for a problem that has difficulty converging. Each set of guesses can be stored in a separate file and loaded as needed. A more general application is to use a variable information file to store information for variables that you use repeatedly in your problems. For example, if you commonly use variables T1, T2, and T3 in your problems, you can set their guess values, limits and units in the Variable Information dialog and Save the information in a Variable information file. Then, when preparing your next problem, you can open this variable information file and the guess values, limits and units will be automatically updated. Used in this way, the variable information files provide a similar capability to that available with the Default Variable Information command, but in this case, the saved information does not need to relate to the first letter in the variable name.

The information in a .VAR file can be automatically loaded using the $Include directive (Chapter 7).

Function Info will bring up the following dialog window.



DENSITY [kg/m3]
DEWPOINT [C]
ENTHALPY [kJ/kg]
ENTROPY [kJ/kg-K]
HUMRAT [kg/kg]
INTENERGY [kJ/kg]
MOLARMASS [kg/kmol]
P_CRIT [kPa]
PRANDTL

R123
R125
R13
R134a
R134a_ha
R14
R141b
R143a
R22

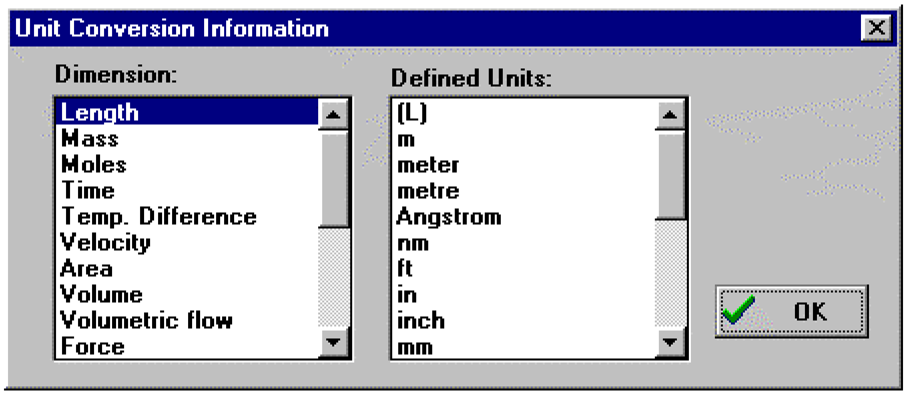
Ex: ENTHALPY(R134a,T=T1,P=P1)

The four buttons at the top of the dialog window indicate which information is to be provided. Math Functions and Thermophysical Props refer to the built-in functions for mathematical and thermophysical property relations, respectively. The User Library button provides a list of the user functions, procedures, and modules loaded from Library files. (See Chapter 5 for additional information on Library files). The External Routines button refers to compiled routines which can be linked to EES as described in Chapter 6. The functions corresponding to the selected button will be displayed in the Function list on the left. To select a function, click on the function name in the scrollable list. Click the **Function Info** button to obtain specific information relating to the function you have selected. The **Fluid Info** button provides information relating to the source and range of applicability of the property correlations.

The units of the thermophysical property function are shown in the function list box. Thermophysical property functions require specification of a substance. The substances for which property data are available are shown in the Substance list on the right. Click on the substance name in the scrollable list to select the substance. 'Ideal gas' will appear above the substance list if the properties of the selected substance are calculated using ideal gas law approximations. 'Real substance' will appear if liquid and vapor states are determined. Substances represented by their chemical formula (e.g., CO2) are modeled as an ideal gas and use JANAF table reference values for enthalpy and entropy. Substances with their name spelled out (e.g., CarbonDioxide) are modeled as real fluids and do not use JANAF table reference values. Air is an exception to this rule. Air is modeled as an ideal gas. Psychrometric functions are applicable only to the substance AirH20. Additional information regarding all built-in functions is provided in Chapter 4.

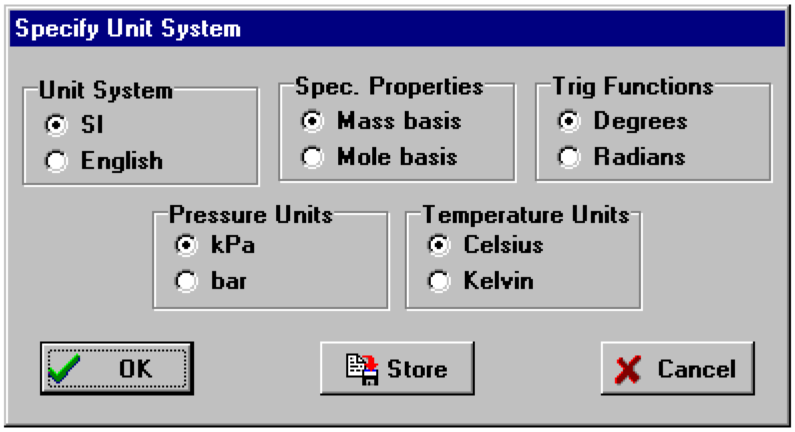
An example of the function with default variables will be shown in the Example rectangle at the bottom. You can edit this information in the usual manner. If you click the Paste button, the contents of the Example rectangle will be pasted into the Equations window at the cursor position.

Unit Conversion Info provides information to support the use of the Convert and ConvertTemp conversion functions. The Convert function has the following format: Convert('From', 'To') where From and To are character strings identifying the unit type such as 'Btu/hr-ft2-R' or 'mph'. (Note that the single quotes marks around the unit identifies are optional.) Many of the unit identifiers are obvious, but not all. The purpose of this command is to list the unit identifiers that have been defined, as shown below.



Click on the dimension in the list at the left. All of the units which have been defined with the selected dimension are listed in the list on the right. Note that only the defined units having the selected dimension are listed. For example, if you click on Area, only Acre and Hectares will be displayed. However, any combination of units having the dimensions indicated at the top of the right list (e.g., L^2 for Area) can be used in the Convert function. You can add additional units if needed. They are stored in the UNITS.TXT file in the main EES directory. Instructions for adding information are provided at the top of the file. This file may be read as a text file and edited in EES.

Unit System provides a dialog window shown below in which the units of the variables used in the built-in mathematical and thermophysical property functions may be set. The unit settings are displayed in the Solution window. The unit system is only needed for the built-in function calls. EES does not provide automatic unit conversion. The units will be changed for the remainder of the work session if the OK button is pressed. The selected units are saved with other problem information when the Save command in the File menu is issued. These units are then restored with the problem using the Open command. If you wish to permanently change the default values, press the Store button.


Specify Unit System

Unit System: [✓ SI] English

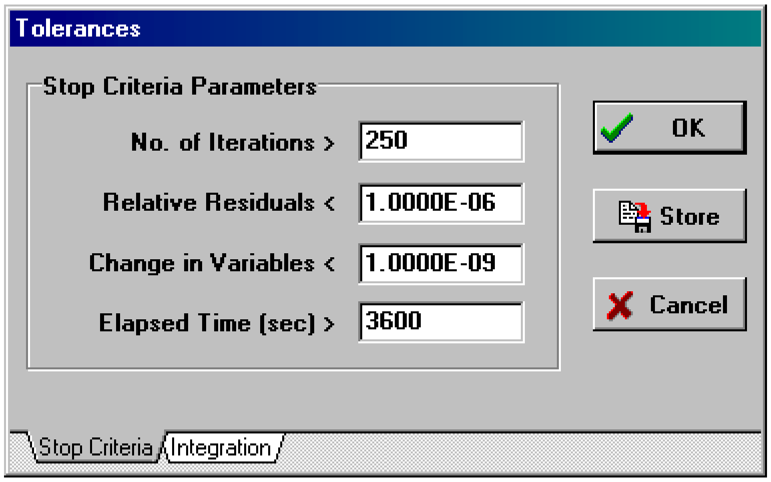
Spec. Properties: [✓ Mass basis] Mole basis

Trig Functions: [✓ Degrees] Radians

Pressure Units: [✓ kPa] bar

Temperature Units: [✓ Celsius] Kelvin

Tolerances will display a dialog with two tabbed windows allowing a specification of Stop Criteria and Integration auto step parameters.


Tolerances

Stop Criteria Parameters

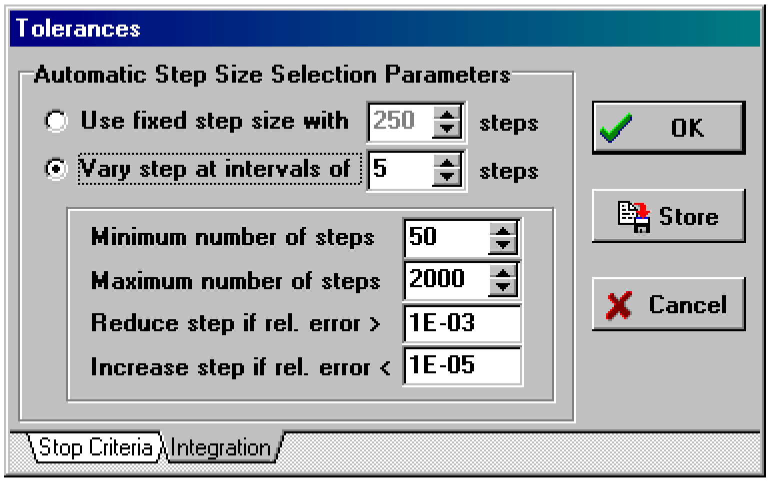
No. of Iterations > 250

Relative Residuals < 1.0000E-06

Change in Variables < 1.0000E-09

Elapsed Time (sec) > 3600

The stop criteria are the number of iterations, the maximum relative residual, the maximum change in a variable value from one iteration to the next and the elapsed time. If any of these criteria is satisfied, the calculations terminate. All calculations in EES are done in extended precision with 21 digits of significance. Loss of precision is unlikely to be a problem even when very small values are set for the maximum residual or variable change. However, small values for these quantities increase the number of iterations required for a solution and therefore the computation time. The stop criteria will be set as displayed for the remainder of the session by pressing the OK button. The stop criteria are saved with other problem information when the Save command in the File menu is issued and restored using the Open command. To change the default stop criteria that EES presents at the start of a new session, press the Store button.



Tolerances

Automatic Step Size Selection Parameters

( ) Use fixed step size with 250 steps

(●) Vary step at intervals of 5 steps

Minimum number of steps	50
Maximum number of steps	2000
Reduce step if rel. error >	1E-03
Increase step if rel. error <	1E-05

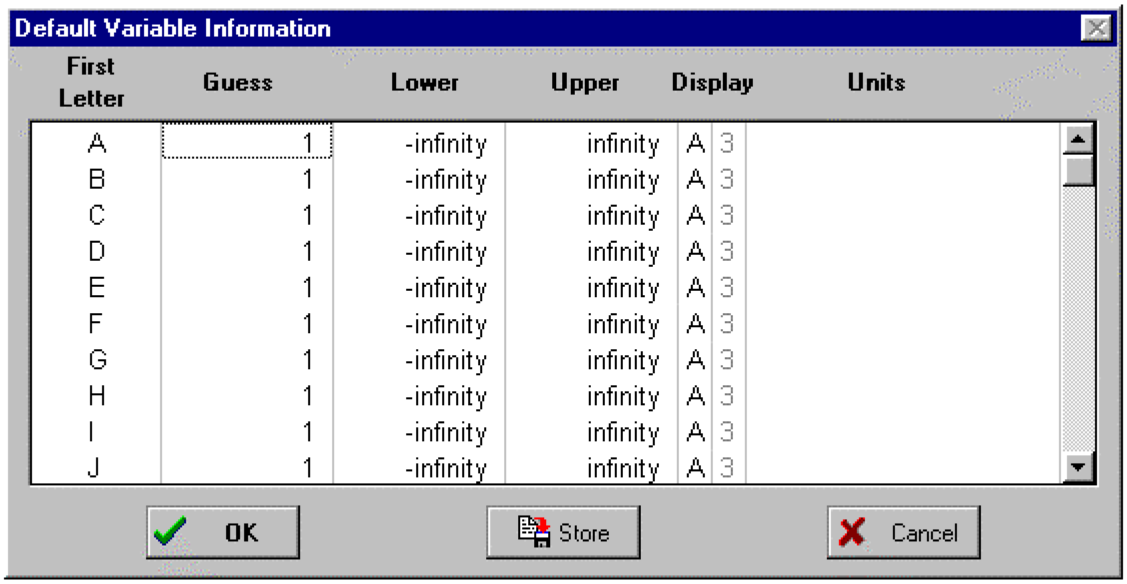
EES uses numerical integration to determine the value of an integral or to solve differential equations. The equation-based Integral function can use either a fixed user-supplied step or an automatic step adjusted to meet some accuracy criteria. The parameters in this tabbed section of the Tolerances dialog only affect the step size that EES automatically selects during numerical integration with the equation-based integral function. The two radio buttons control whether EES will use a fixed or a variable step. If the "Use fixed step size' button is selected, EES will not attempt to adjust the step during calculation, but rather use a fixed step equal to the interval divided by the number of steps indicated by the user. If "Vary step at intervals of" is selected, EES will check the numerical situation of the integration process every N steps where N is input by the user. During this checking process, EES may decrease, increase, or maintain the current step size. The minimum and maximum allowable steps and tolerances which control whether the step size is changed are input by the user. The test that is done is as follows:

Every N steps EES will compare the value of the integral for that step with the value obtained using two half steps. The difference between these two values is the truncation error. The truncation error is normalized by dividing it by the value of the integral for the two half-steps, assuming that it differs from zero. If the normalized truncation error is greater than the value provided for 'Reduce step if rel. error >' the step size if reduced, assuming that it is greater than the minimum allowable step size. If the normalized truncation error is less than the value provided for 'Increase step if rel. error <', the step is increased, provided that the increased step size does not exceed the maximum allowable step. Otherwise, no change is made to the step size.

Note that there is an indirect relation between the Stop Criteria parameters and the integration truncation error parameters. If the Stop Criteria parameters are set to allow convergence with errors larger than that used to reduce the step size, the calculation time and solution accuracy will both be adversely affected.

Default Info provides a means for specifying the default guess values, bounds, display format and units of new or existing variables depending on the first letter in the variable name. There are two ways to use this command. If the problems you do all tend to have the same nomenclature, it is best to set the default variable information and save it by pressing the Store button. The Store button will cause the current default settings to be permanently saved so that these defaults will appear at the start of the program the next time EES is run.

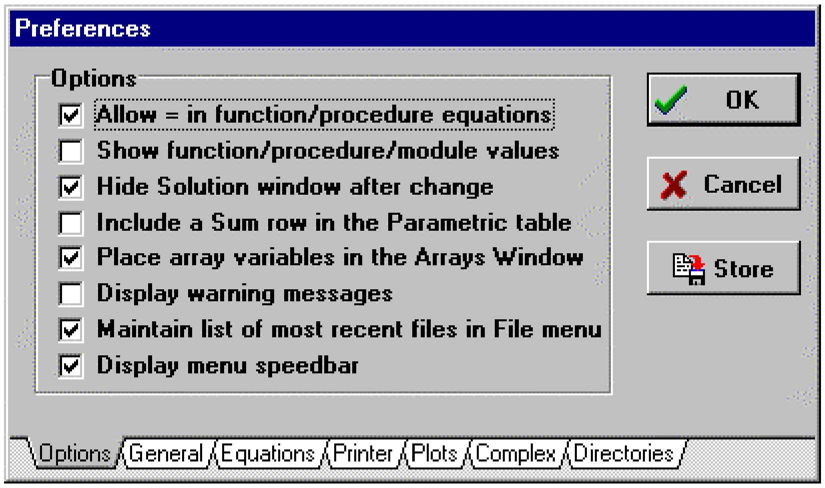
The Default Variable Information command can also be used to selectively change information for existing variables. For example, if you change the units for variables beginning with letter T to [K] and press the OK button, all existing variables beginning with letter T will take on these new units. No other changes to existing variables will be made. Each new variable beginning with letter T will then also take on units of [K]. The OK button sets the current default setting for this problem session only.



First Letter	Guess	Lower	Upper	Display	Units
A	1	-infinity	infinity	A 3	
B	1	-infinity	infinity	A 3	
C	1	-infinity	infinity	A 3	
D	1	-infinity	infinity	A 3	
E	1	-infinity	infinity	A 3	
F	1	-infinity	infinity	A 3	
G	1	-infinity	infinity	A 3	
H	1	-infinity	infinity	A 3	
I	1	-infinity	infinity	A 3	
J	1	-infinity	infinity	A 3	

Show/Hide Diagram Tool bar is enabled when the Diagram window or a child Diagram window (Professional version) is foremost. When enabled, selecting this command will toggle the state of the tool bar for the foremost Diagram window. When the tool bar is visible, the Diagram window is in development mode. Text and graphic objects such as (lines/arrows, rectangles, and ellipses) may be moved, changed or deleted in development mode. An Align button is provided to facilitate alignment of objects relative to one another. When the tool bar is hidden, all text and graphic objects are locked and the window is in application mode. Input is accepted for input variables and calculations can be initiated using the input variables supplied in the Diagram window. The tool bar visibility can also be toggled using the Diagram Window button on the speedbutton bar just below the menu bar. See the Diagram window section in Chapter 2 for more detail.

Preferences provides six tabs for user choices concerning program options, general display options, equations display, printer display, plot window, and complex number options. These options are shown and described below. If the OK button is clicked, the selected preferences remain in effect for the remainder of the work session. The Store button saves the preferences so that they will be in effect at the start of the program the next time EES is run.


Preferences

Options

- ☑ Allow = in function/procedure equations
- ☐ Show function/procedure/module values
- ☑ Hide Solution window after change
- ☐ Include a Sum row in the Parametric table
- ☑ Place array variables in the Arrays Window
- ☐ Display warning messages
- ☑ Maintain list of most recent files in File menu
- ☑ Display menu speedbar

☑ **Allow = in function/procedure equations** will suppress the error message that would normally occur if the assignment symbol (:=) is not used in EES Functions and Procedures. EES Internal Functions and Procedures (Chapter 5) employ assignment statements as in FORTRAN and Pascal, rather than equations as used in the main body of the EES programs. (EES modules use equality statements as in the main body of the EES program and thus cannot use the := assignment statement syntax.) An assignment statement sets the variable identified on the left of the statement to the numerical value on the right. X:=X+1 is a valid assignment statement, but it obviously is not an equality. The := sign is used to signify assignment, but if this control is selected, EES will also accept X=X+1.

☑ **Show function/procedure/module values** will allow the most recent values of local variables in EES functions, procedures and modules to be displayed in the Solution window. Module equations will also appear in the Residuals window. The values of these local variables are ordinarily not of interest but you may wish to know them, particularly for debugging purposes. Note that only the functions, procedures, and modules which appear in the Equations window are affected by this setting. The local values of variables in Functions, procedures, and modules that have been loaded from library files (See Chapter 5) are not displayed.

☑ **Hide Solution Window after change** causes the Solution, Arrays, and Residual windows to be removed from the screen display if a change is made in the Equations window. If this option is not selected and a change is made in the Equations window, the Solution window title will change to Last Solution.

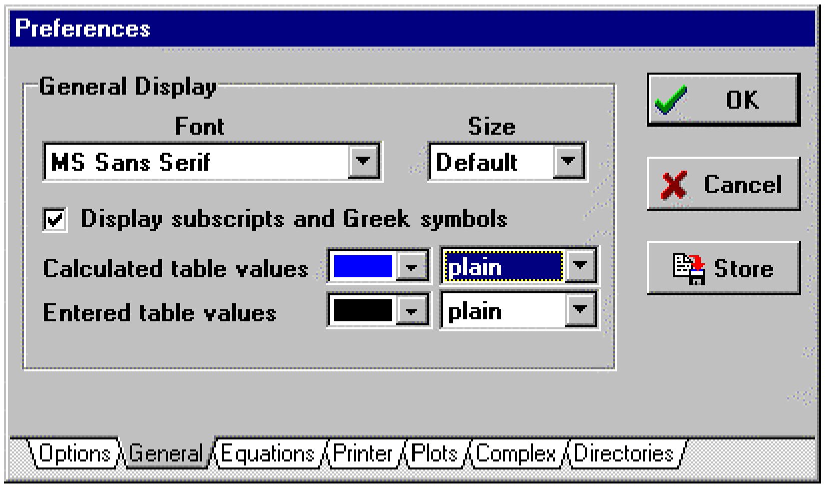
☑ **Include a Sum row in Parametric Table** will result in an extra row being added to the Parametric table which displays the sum of the values in each column.

☑ **Place array variables in the Arrays Window** instructs EES to display all array variables in the Arrays window rather than in the Solution window after calculations are completed. Values in the Arrays window can be plotted and copied just like values in the Parametric and Lookup tables. See the Arrays Window section of Chapter 2 for additional information.

☑ **Display warning messages** will enable or disable warning messages during calculations. Warnings are issued if thermophysical property correlations are applied outside of their range of applicability. Warnings can also be turned on or off using the $Warnings On/Off directive in the Equations window.

☑ **Maintain a list of recent files in the File menu** enables or disables a list of up to 8 recent files at the bottom of the File menu. This list is a convenience that you normally would want to have. The file names are stored in a file entitled EES.FNL in the EES directory. However, if EES is placed on a server where multiple users can access the program, it is best to disable this feature. (This capability is disabled for educational versions.)

☑ **Display menu speedbar** controls the visibility of the toolbar appearing below the menu bar. The toolbar will be hidden if this control is unselected.


Preferences

General Display

Font: MS Sans Serif
Size: Default

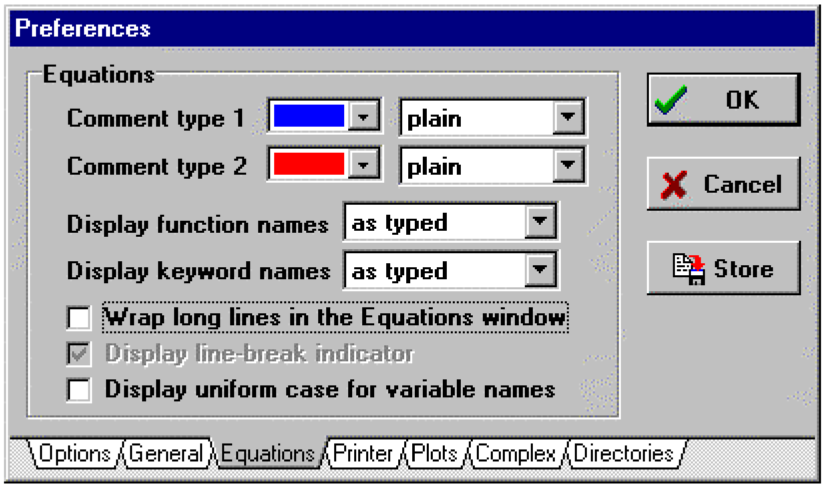
☑ Display subscripts and Greek symbols

Calculated table values 🟦 plain
Entered table values ⬛ plain

The font and font size can be selected from the drop-down lists. The new font and font size will display in all of the EES windows. The printer display is unaffected. The printer font and font size is selected separately in the Printer Display Tab.

Display subscripts and Greek symbols controls the appearance of EES variables in the Solution, Formatted Equations, Parametric Table and Diagram windows. If this option is selected, the underscore will be used to signify the start of a subscript. The following characters in the variable name will be displayed in smaller type and lower case. Additional underscores within the same variable name will be changed into commas. Array variables will also be changed to appear as subscripted variables. Variables which have the name of Greek symbols, such as alpha, beta, and gamma, will appear in Symbol font. If the variable name is entirely in capital letters, the Greek symbol will be shown as a capital letter. A dot, bar, or hat can be positioned over the variable name by adding _dot, _bar, or _hat to the name. For example, X_dot will display as $\dot{X}$. X_infinity will display as X_∞. X_star will display as X*. X_hat will display as $\hat{X}$. The vertical bar character signifies the start of a superscript. For example, G|o will be displayed as G^{o}.

The Calculated Table Values and Entered Table Values pertain to the display of values in the Parametric Table. Entered table values are values which are provided by the user, either directly by typing the value or indirectly through application of the Alter Values dialog. Calculated values are provided by EES during the Solve Table or Min/Max Table commands. Two fields are provides for each category to set the color and the font style of the display. Note that a separate font style can be selected for printing in the Printer Preferences tab.



Equations

Comment type 1 🟦 plain

Comment type 2 🟥 plain

Display function names: as typed

Display keyword names: as typed

☐ Wrap long lines in the Equations window

☑ Display line-break indicator

☐ Display uniform case for variable names

EES can display comments in two different colors and/or styles which are selected by the user with the two drop down controls that follow the comment type label. Comment Type 2 is distinguished from Comment Type 1 by having an exclamation character ! as the first character in the comment. Both Type 1 and Type 2 commands must be enclosed within braces or single quotes. If braces are used, the comment will be displayed in the Equations window but not in the Formatted Equations window. For example,

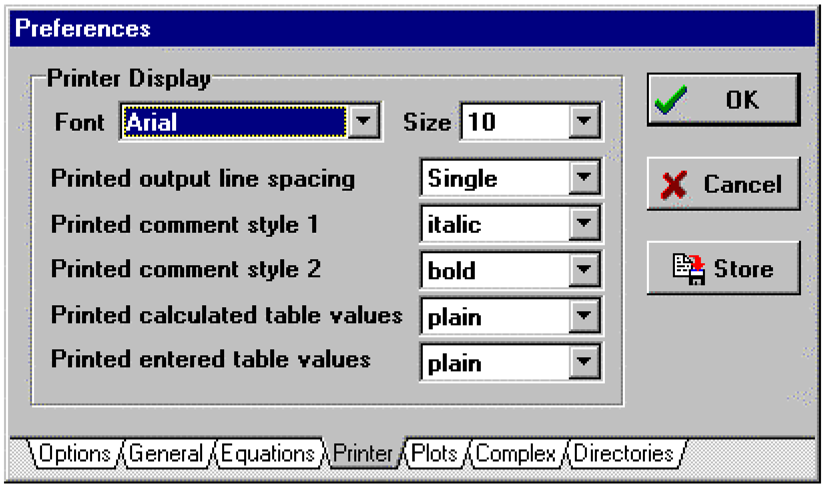
{!This is a Type 2 comment}
"!This is also a Type 2 comment and it will display in the Formatted Equations Window}
"This is a Type 1 comment."

Function names (such as ENTHALPY, SIN, etc.) and keywords (such as FUNCTION, DUPLICATE, fluid names, etc.) can be displayed in upper case, lower case, or as typed, as selected with the drop-down controls.

☑ **Wrap long lines in the Equations window** will hide the horizontal scroll bar. Lines which are too long to be displayed in the Equations window will be broken at an appropriate point and continued on the following line. A red > symbol will be displayed in the left margin of continuation lines if Display line-break indicator is selected.

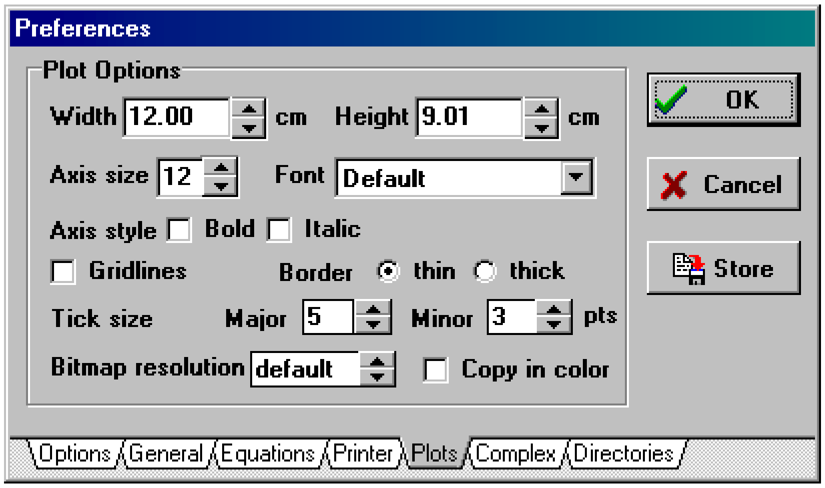
☑ **Display line-break indicator** is applicable only if the Wrap long lines option is selected. This option controls whether line break characters appear in the left margin of continuation lines.

☑ **Display uniform case for variable names** causes each variable to appear with the upper and lower case lettering sequence set in the first occurrence of the variable in the Equations window. If the first occurrence of the variable is changed, the Check/Format command in the Calculate menu will change all other occurrences.



Font: Arial
Size: 10
Printed output line spacing: Single
Printed comment style 1: italic
Printed comment style 2: bold
Printed calculated table values: plain
Printed entered table values: plain

The options in the Printer Tab only affect the appearance of the printed output. Font and size provide choices for the type in which the printed output will appear. Printed output line spacing provides single, space and one-half, or double-space options. Printed comment style 1 and Printed comment style 2 fields allow a font style to be set for each comment type. Comment type 2 is distinguished from comment style 1 by having an exclamation mark (!) as its first character. These controls allow the comments to be printed in a different style than is displayed on the screen. This option is particularly useful when the comments are displayed in color on the screen but are to be printed on a black and white printer. A Print in Color control is provided in the Print dialog window.



Preferences

Plot Options

Width 12.00 cm Height 9.01 cm

Axis size 12 Font Default

Axis style [ ] Bold [ ] Italic

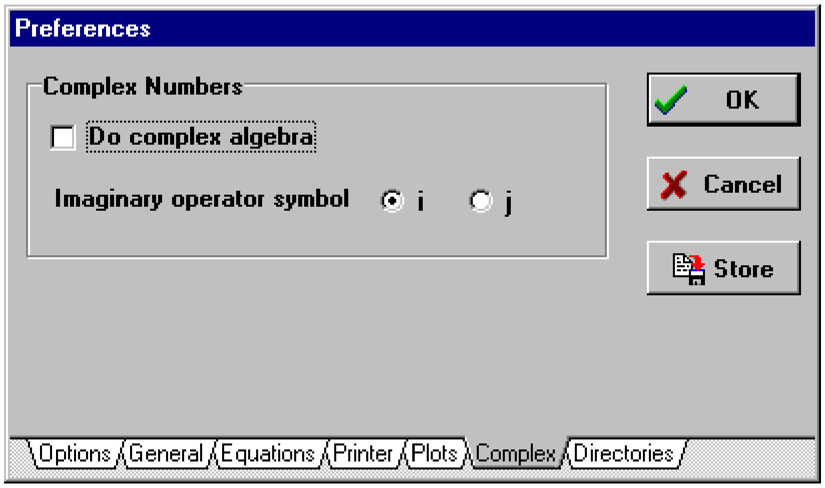
[ ] Gridlines Border [✓ thin] thick

Tick size Major 5 Minor 3 pts

Bitmap resolution default [ ] Copy in color

The Plots tab allows the default setting for the plot width and height, the font, font size, font style, and the major and minor tick sizes for the axis scales to be changed. The plot width and height are entered in point units. Depending on your equipment and video setting, a point is either 1/96 or 1/120 inch. Ticks are the short line segments on the axis scale. Major ticks are placed on the scale at the point where the axis numbers appear whereas minor ticks occur between the axis numbers. Ticks which are drawn into the plot rectangle are represented with positive numbers. The plot can be configured for outdented ticks by specifying negative values for the tick sizes. These default characteristics are applied whenever a new plot is generated.

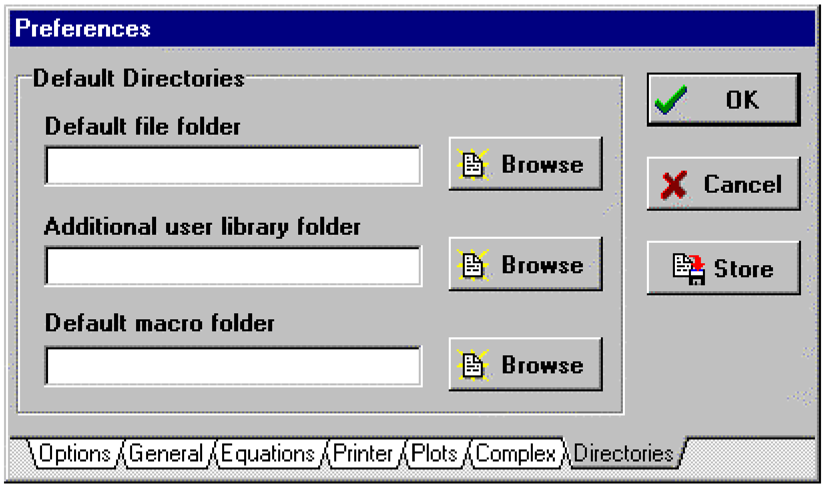
The bitmap resolution and Copy in Color controls affect the resolution, color, and memory requirements of plots copied to the clipboard. If the Copy in Color control is not checked, EES will replace all colors with black before the copy is placed on the clipboard. Use the Paste Special command in an external application such as a word processor, to select the type of paste operation. EES copies a plot to the clipboard as a picture and as a bitmap. The picture format has the advantages of requiring the least memory and, when pasted into a drawing program, it can be modified. However, when printed, the picture format will print at screen resolution and it may appear satisfactory. The Device Independent Bitmap format provides the highest quality image but it requires more memory. The bitmap resolution control affects the memory requirements. The default setting, 300 pixels per inch, seems to provide acceptable results in most cases.



[ ] Do complex algebra

Imaginary operator symbol [✓ i] j

The Complex Tab allows the complex algebra capability in EES to be turned on or off. The complex capability can also be turned on or off with the $COMPLEX ON/OFF directive. The imaginary variable name representing the square root of –1 can be designated to be either i or j depending on the radio button setting.



Default file folder

Additional user library folder

Default macro folder

The Default File Folder specifies the folder that EES will open whenever the Open or Save As... commands are issued. If the directory specified in this field does not exist or if no directory is specified, EES will initially use the directory that the EES program was started in and thereafter, it will use the last accessed directory.

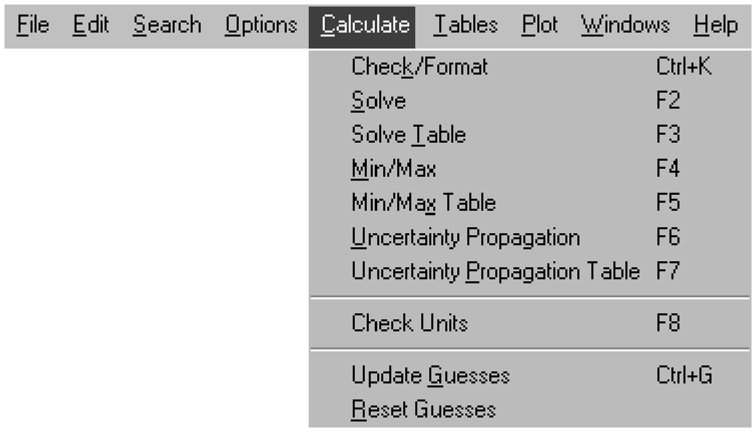
When EES is started, it transparently loads library files from the USERLIB folder located in the directory in which the EES application is located. However, if a valid directory name is provided in the "Alternate user library folder" field, EES will use also open all of the library files in this directory at startup.

The default folder for Macro files is used when the Build Macro command (Professional vesion) is issued. If no folder name is provided, EES will use the directory in which EES is located.

The default directory information is stored in the EES.DIR file. Deleting this file resets the directory information.

In any of the Preferences in any of the tabbed windows are changed, pressing the OK button will set the Preferences for this session only. The Store button will cause the Preferences to be permanently saved so that they will be in effect at the start of the program the next time EES is run.

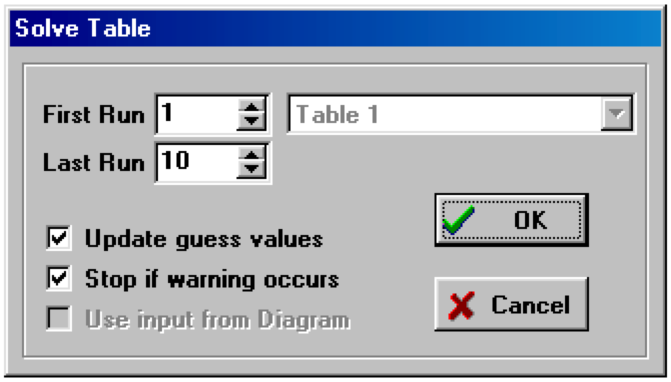
The Calculate Menu



Check/Format will recompile all equations and apply the formatting options selected with the Preferences command in the Options menu. The first syntax error found will be indicated with a message. If no syntax errors are encountered, EES will indicate the number of equations and variables in the Equations window.

Solve will first check the syntax of the equations in the Equations window. If no errors are found and if the number of equations is equal to the number of variables, a solution to the equation set will be attempted. The methods used by EES for solving equations are described in Appendix B. An information dialog window summarizes the progress of the solution. When the calculations are completed, the information dialog indicates the elapsed time, number of blocks, the maximum residual (i.e., difference between the left and right hand sides of an equation), and the maximum change in the value of a variable since the previous iteration. If the number of equations is not equal to the number of variables, EES will provide the option of viewing the Debug window which may help locate a problem. If the Diagram window is being used to enter one or more variable values, it must be open when the Solve command is issued. See Chapter 2 for more information concerning the Debug and Diagram windows.

Solve Table will initiate the calculations using the Parametric Table. (See the description of Parametric menu commands on the following pages for information on the use of the Parametric Table.) The dialog shown below will appear. More than one Parametric tables may be defined in which case the table for which the calculations are to be done must be selected from the drop-down list. One of the choices in the drop-down list is “All Parametric Table”. Choosing this option results in EES doing the calculations for all of the Parametric tables in sequential order.

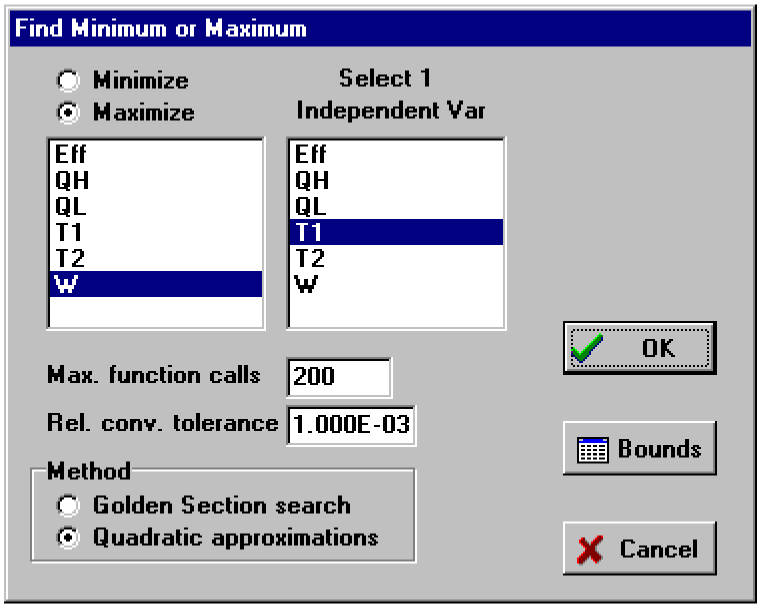


Each row in a Parametric Table is a separate problem. The values of independent variables are shown in normal type. Blank cells (or cells with bold, blue, or italic type from a previous Solve Table command) are dependent variables. The values of these variables will be cleared and the newly calculated values will be entered in the table. If the Update Guess Values control is selected, the guess values for each run will be set to the calculated values from the previous run; otherwise each run will be initiated with the guess values specified with the Variable Info command.

Warnings are issued if a property correlation is applied outside of the range for which it was developed. If the Stop if warning occurs control is selected, EES will terminate the table calculations on the run at which the warning occurred. Otherwise, EES will continue the calculations for the remaining rows. A message will be displayed after all of the table calculations are completed indicating the rows in which a warning occurred.

The “Use input from Diagram” control will be enabled if the Diagram window is open and if it has one or more input variables. If this control is selected, EES will accept input values from the Diagram window, just as if these inputs were specified in the Equations window.

Min/Max is used to find the minimum or maximum of an undetermined variable in an equation set for which there is at least one and ten or fewer degrees of freedom. EES will first check the syntax of the equations in the Equations window. If no errors are found, a dialog window will appear presenting the undetermined variables in two lists.



Minimize [✓ Maximize]

Eff, QH, QL, T1, T2, [✓ W]

Select 1 Independent Var: Eff, QH, QL, [✓ T1], T2, W

Max. function calls: 200

Rel. conv. tolerance: 1.000E-03

Method: Golden Section search [✓ Quadratic approximations]

Click the Minimize or Maximize button above the left list. The variable which is to be minimized/maximized is selected by clicking on its name in the list on the left. The independent variable(s) whose value(s) will be changed in searching for the optimum appear in the list on the right. It is necessary to select as many independent variables as there are degrees of freedom in the Equations window. The number of independent variables which must be selected is indicated above the right-hand list. To select (or unselect) a variable, click on its name in the list.

If there is one degree of freedom, EES will minimize/maximize the selected variable using either a Golden Section search or a recursive quadratic approximation method, depending on the settings of the buttons at the bottom of the dialog window. (See Appendix B for information on the optimization algorithms.) The recursive Quadratic Approximations method is usually faster, but the Golden Section method is more reliable. Multi-dimensional optimization may be done using either Direct Search or a Variable Metric algorithm. The Variable Metric method, which uses numerical derivatives, usually performs much better than the Direct Search method, but it may be confounded if the optimum is constrained to be on a bound.

EES requires finite lower and upper bounds to be set for each independent variable. Careful selection of the bounds and the guess value(s) of the independent variables will improve the likelihood of finding an optimum. You can view or change the bounds and guess value for each selected independent variable by clicking the Bounds button. This will bring up an abbreviated version of the Variable Info dialog containing just the selected independent variables. See the description of the Variable Info command in the Options menu for additional information on setting the bounds.

The maximum number of times in which the equations are solved (i.e., the number of function calls) may be specified, along with the relative tolerance. Calculations will stop if: 1) the relative change in the independent variable(s) between two successive steps is less than the specified tolerance; or 2) the number of steps exceeds the specified maximum. EES will also stop the calculations if the equations cannot be solved with specified value(s) of the independent variables within the tolerance and allowable number of iterations specified with the Stopping Criteria command in the Options menu.

Min/Max Table provides the same capability as the Min/Max command, except that the calculations will be repeated for each row in the Parametric Table. (See the description of Parametric menu commands on the following pages for additional information on the use of the Parametric Table.) As with the Min/Max command, a dialog window will appear in which the variable to be maximized or minimized and the independent variable(s) can be selected. In this case, however, the variable which is to be optimized and all of the independent variables (whose values will be varied in seeking the optimum) must appear in the Parametric Table. The start and stop runs in the Parametric Table for which the calculations will be done may be specified. Values in the Parametric Table which are shown in normal type are fixed and are treated just as if they were set to that value with an equation in the Equations window. The variable which is to be optimized and the independent variable(s) must be the same for each run. If no errors are encountered, the optimum is computed and the values of the remaining columns in the table are entered for each run.

Uncertainty Propagation determines the uncertainty of a selected calculated variable as a function of the uncertainties of one or more measured values upon which it depends. In many experiments, an important quantity is not directly measured but is rather calculated as a function of one or more variables that are directly measured, i.e., Y = f(X1, X2, ...). The measured variables, X1, X2, etc. have a random variability which is referred to as its uncertainty. In EES, that uncertainty is displayed with a ± symbol, e.g., X1 = 300 ± 2.

The purpose of this command is to calculate how the uncertainties in all of the measured variables propagate into the value of the calculated quantity, Y. The method for determining this uncertainty propagation is described in NIST Technical Note 1297 (Taylor B.N. and Kuyatt, C.E., Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results, National Institute of Standards and Technology Technical Note 1297, 1994). Assuming the individual measurements are uncorrelated and random, the uncertainty in the calculated quantity can be estimated as

$$U_Y = \sqrt{\sum_i \left(\frac{\partial Y}{\partial X_i}\right)^2 U_{X_i}^2}$$

where U represents the uncertainty of the variable. After selecting this command, EES will present two lists of variables. Select the variable for which the uncertainty propagation is to be determined from the list on the left. Select one or more measured variables from the list on the right. Note that the variables appearing in the measured variables list must be constants so that their values are set to a numerical constant with an equation in the EES Equations window. To specify the uncertainty associated with the measured variables, click the Set Uncertainties button below the right list. A second dialog window will appear in which the absolute or relative (fraction of the measured value) uncertainties for each selected measured variable can be specified. An uncertainty value for each measured variable must be provided. Click the OK button to set the uncertainties and close the Specify Uncertainties dialog window. Click the OK button Uncertainty Propagation dialog to start the calculations.

After the calculations are completed, EES will display an abbreviated Solution Window containing the calculated and measured variables and their respective uncertainties. The partial derivative of the calculated variable with respect to each measured variable will also be displayed.

Uncertainty Propagation Table provides the same function as the Uncertainty Propagation command, namely the determination of the uncertainty propagation in a calculated variable. The difference is that this command allows the uncertainty calculations to be repeated for one or more measurements by using the Parametric Table. The calculated and measured variables must all be in the Parametric Table before this command is used. After selecting the command, the Uncertainty Propagation dialog window will appear in which the calculated quantity is selected from list of variables on the left and the measured variable(s) are selected from the list on the right. The Parametric Table calculations will proceed after the OK button is selected, just as if the Solve Table command were applied. The value and uncertainty for the calculated variable and each measured variable will be displayed in the Parametric Table after the calculations are completed. The calculated variable can then be plotted with error bars representing the propagated uncertainty using the New Plot Window command.

Check Units will check the dimensional and unit consistency of all equations in the main part of the equations window. The equations in internal functions and procedures do not have units and they are not checked. The results are reported in the Debug window. It is necessary to enter the units of each variable for the checking process to function properly. The units can be entered in the Variable Information dialog window or in the Solution window. The units of a variable set to a constant in the Equations window can also be set with in a comment by enclosing the units in square brackets. For example:

```
P=140 “[kPa]   this line will set P=140 and its units to kPa”
```

The checking algorithm cannot know the units of conversion constants, so it is best to avoid them in your equations. The Convert function should be used instead. For example, suppose you have two variables, L_inch and L_feet, whose units are set to inches and feet, respectively which are used in the following equation.

L_inch=L_feet*12

When the Check Units command is issued, the Debug window will appear and this equation will be flagged as having an error because the units of 12 are not known. However, if the Convert function is employed as shown next, the equation will be accepted with no error.

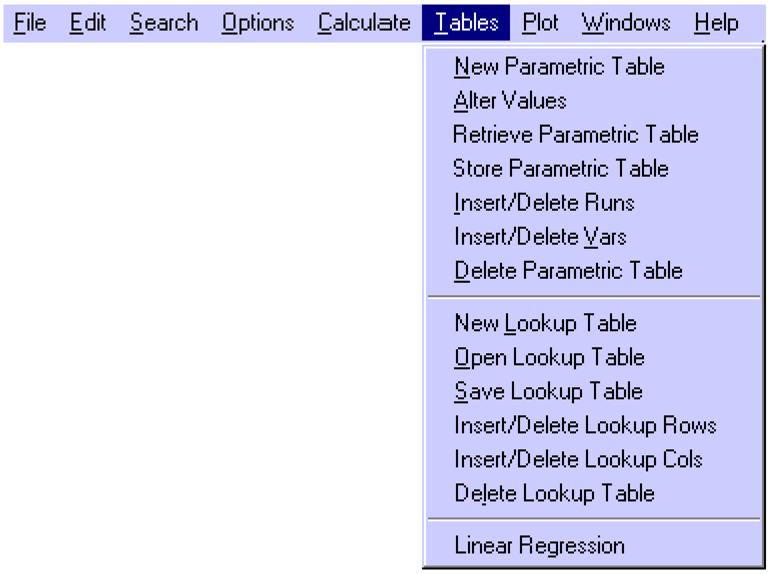
L_inch=L_feet*convert(ft,in)

The Check Units command will display the equation and an explanatory message for each equation that is found to have dimensional consistency or unit consistency errors. If you click the left mouse button on an equation, the focus will jump to that equation in the Equations window. If you click the right mouse button on an equation, an abbreviate form of the Variable Information dialog window will appear showing just the variables that appear in that equation.

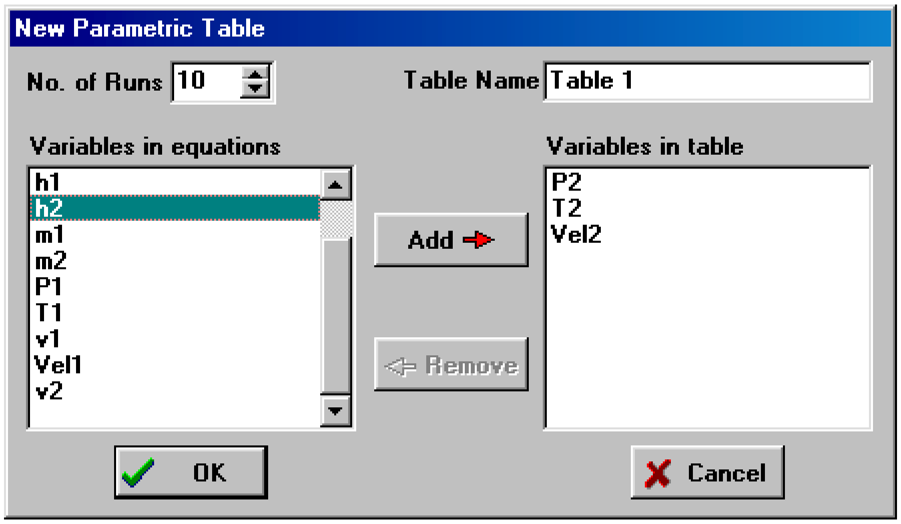
Update Guesses replaces the guess value of each variable in the Equations window with the value determined in the last calculation. This command is accessible after calculations have been successfully completed. Update Guesses improves the computational efficiency of an EES calculation since it ensures that a consistent set of guess values is available for the next calculation. Exactly the same function is provided with the Update button in the Variable Info dialog window, but the Update Guesses command is more accessible.

Reset Guesses replaces the guess value of each variable in the Equations window with the default guess value for that variable. Unless otherwise specified, EES assumes all guess values are 1.0. You can change the default guess values with the Default Info command in the Options menu. You should reset the guess values only if you are experiencing convergence difficulties and you have changed the guess values in an attempt to find a solution.

The Tables Menu



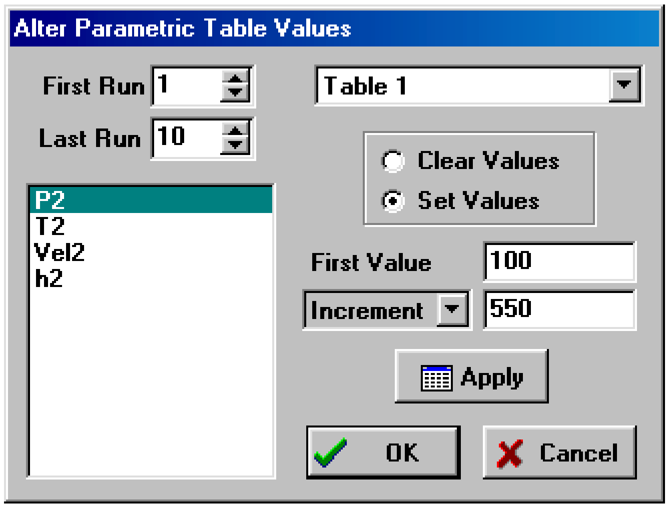
New Parametric Table creates a new Parametric Table in the Parametric Table Window. Parametric Tables are used in EES to automate repetitive calculations, to solve differential equations, and to present data for plotting or curve-fitting. A dialog window will appear in which information must be entered to create the table, as in this example. There is no limit on the number of tables that can be created, other than available memory.



Each table is identified with a table name that is entered when the table is created. The table name is displayed on a tab at the top of the Parametric Table Window. The table name can later be changed by clicking the right mouse button on the tab. The number of runs, which corresponds to rows in the table, is entered in the field at the top. All variables (both independent and dependent) which are to appear in the table are selected from the alphabetical list of variables on the left. To select a variable, click on its name in the list, which will cause it to be highlighted. Multiple names can be selected. Click the Add button to move the highlighted variable names to the list on the right. (As a shortcut, a variable is automatically added to the list on the right when you double-click on its name in the list on the left.) The variables in the right-hand list will appear in the columns of the table in the same order in which they appear in the list. A variable can be removed from the table list by clicking on its name in the right list and then clicking the Remove button or by double-clicking on the variable name. Pressing the OK button will create the Parametric Table, overwriting any existing table.

The Parametric Table operates somewhat like a spreadsheet. Numerical values can be entered in any of the cells. Entered values are assumed to be independent variables and are shown in normal type. Entering a value in a table produces the same effect as setting that variable to the value in the Equations window. Dependent variables will be determined and entered into the table in blue, bold, or italic (depending on the choice made in the Preferences dialog) when the Solve Table, Min/Max Table, or Uncertainty Propagation Table command is issued. If a variable is set in the table, it can not also be set in the Equations window; otherwise the problem will be overspecified. Each row of the table is a separate calculation. The independent variables may differ from one row to the next. However, for every row, the number of independent variables plus the number of equations must equal the total number of variables in the problem.

Alter Values provides an automatic way to enter or clear the values of a variable for multiple runs. This command will bring up the Alter Values dialog, shown below.


Alter Parametric Table Values

First Run 1
Last Run 10
Table 1
Clear Values [✓ Set Values]
P2
T2
Vel2
h2
First Value 100
Increment 550

The runs (i.e., rows) affected are specified at the upper left of the dialog. The variable for which changes are to be made is selected from the list by clicking on its name. The column for this variable will be cleared if the Clear Values control is selected. If Set Values is selected, values for the selected variable will be entered automatically in the table starting with the value in the First Value field. The list box below the First Value controls the manner in which successive values in the table are generated. The choices are Last Value, Increment, and Multiplier. Increment or Multiplier result in successive values in the table being determined by either adding or multiplying, respectively, the preceding table value by the value provided in the box. If Last Value is selected (as shown) the increment will be selected such that the last run has the specified value. The Apply button will change the Parametric table as specified but control will remain in the Alter Table Values dialog window so that additional changes can be made. The OK button accepts and finalizes all of the changes made to the Parametric table.

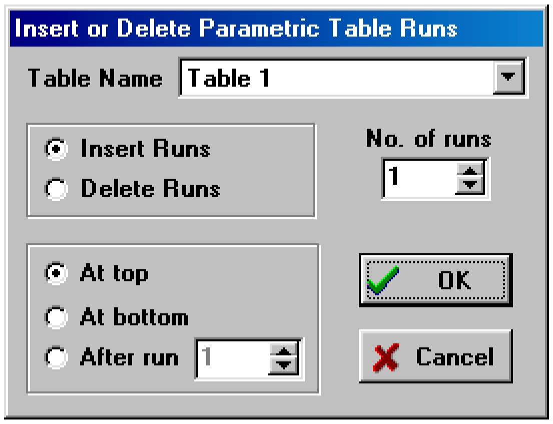
The numerical values entered in a table, either directly or through the Alter Values command, identify independent variables in the equation set; they are shown in normal type. Independent variables are fixed to a constant for each run, just as if there were an equation in the Equations window setting the variable to the constant. Dependent variables are shown in italic, blue or bold depending on the choice made in the Preferences dialog. These values are automatically entered in the table with the Solve Table and Min/Max Table commands. If a value is set in a table, it must not also be set in the Equations window; otherwise an error message will be displayed.

There are other ways of changing the data in the Parametric table. Clicking on the ▼ control at the upper right of each column header cell (or selecting Alter Values from the popup menu that appears when the right-mouse is clicked in the header cell) will bring up a dialog window which operates just like the Alter Values dialog. Also, you can simply type the values directly into the Parametric Table.

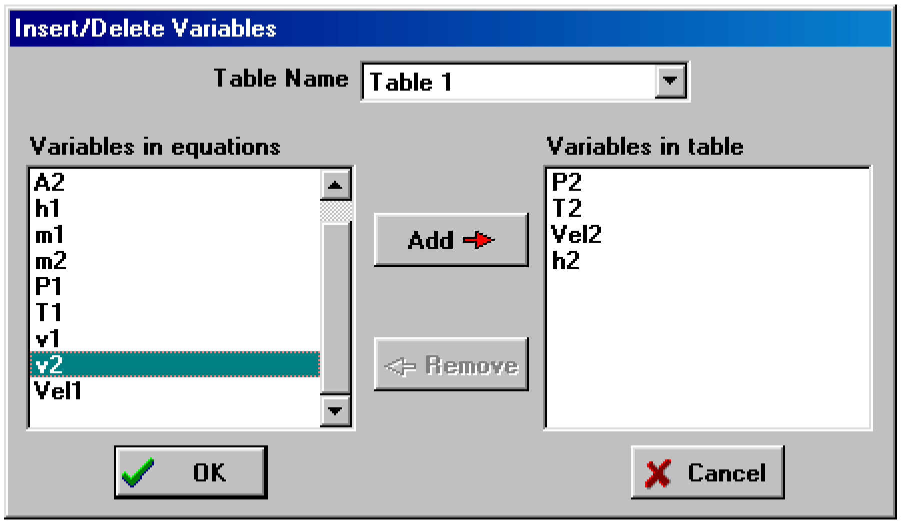
Retrieve Parametric Table will read a specified .PAR file and restore the Parametric table to the same condition it was in when the Store Parametric Table command was issued. See the Store Parametric Table command below for details.

Store Parametric Table saves the current Parametric Table to a binary disk file which, by default, has a .PAR filename extension. All information related to the Parametric Table is saved. The file can later be read to recreate the table with the Retrieve Parametric Table command. The Store and Retrieve Parametric Table commands were originally developed to overcome the limitation of a single Parametric Table in earlier versions of EES. However, EES now allows an unlimited number of Parametric Tables to be defined, so it should no longer necessary to Store and Retrieve Parametric Tables. The Retrieve Parametric Table is provided to allow Parametric Tables saved in older versions of EES to be opened and to allow Parametric Tables from one EES file to be ported to another EES file.

Insert/Delete Runs allows the number of runs in an existing Parametric Table to be changed by inserting or deleting one or more rows in a specified Parametric table at a specified position. Rows in the Parametric Table can also be inserted or deleted by clicking the right mouse button in the first column of the table and selecting Insert Row or Delete Row from the popup menu.



Insert/Delete Variables allows variables in an existing Parametric Table to be added or removed. The following dialog window will appear.



The list on the right shows the variables which currently appear in the Parametric Table. Variables that may be added to the table appear in the list on the left. To add one or more variables to the table, click on the variable name(s) causing them to be highlighted. Click the Add button to move the highlighted variable names to the list on the right. (You can also add a variable by double-clicking on the variable name.) Variables can be deleted from the table by selecting them from the list on the right, followed by clicking on the Remove button. Variables can also be added or deleted from the Parametric Table by clicking the right mouse button in the column header cell following by selecting Insert Column or Delete Column from the popup menu that appears.

Variables will appear in columns of the Parametric Table in the same order as they appear in the list on the right. The order of the variables in this list can be changed by pressing and holding the left mouse button on a variable name while sliding it up or down to a new position in the list. The column order of an existing Parametric table can also be changed by clicking on the column header cell as described in Chapter 2.

Delete Parametric Table will, after a confirmation, delete the Parametric Table and recover the memory it required.

New Lookup creates a table with the specified number of rows and columns in which tabular numerical or string data may be entered. A name must be provided for the Lookup Table. This is the name that must be used with commands that use the Lookup Table. The name will appear on a tab at the top of the Lookup Table Window. The tabular data may be automatically interpolated, differentiated, and used in the solution of the problem using the **Interpolate**, **Differentiate**, **Lookup**, **LookupRow**, and **LookupCol** functions described in Chapter 4. There is no limit (other than available memory) on the number of Lookup Tables that may appear in the Lookup Table Window. In addition, data in the Lookup Window may be saved in a Lookup File (with a .LKT, .TXT. or .CSV filename extension) and the Lookup Files stored in disk files can also be accessed by the **Interpolate**, **Differentiate, Lookup**, **LookupRow**, and **LookupCol** functions. Lookup tables and files provide a great deal of power to EES by allowing any functional relationship between variables which can be represented by tabular information to be entered and used in the solution of the equations.

Open Lookup will read into the Lookup Table Window a Lookup file which was previously stored with the Save Lookup command or as a text file. A name derived from the filename is given to the Lookup Table and this name appears on the tab at the top of the Lookup Table Window. A Lookup file is a two-dimensional table of data that has been stored in a disk file. A name and display format for each column of data may also be stored, depending on the file format. Lookup files can be accessed by the **Differentiate**, **Interpolate**, **Lookup**, **LookupCol**, and **LookupRow** commands. EES recognizes both binary and ASCII forms for Lookup files. Binary files are identified with a .LKT filename extension. ASCII Lookup files usually have a .TXT or. .CSV filename extension. Each file format has advantages and disadvantages. The binary form is read in more quickly and it requires smaller file sizes. The ASCII form is easier to edit and it can be written by spreadsheet or other applications. See Chapter 4, Using Lookup Files and the Lookup Table, for details.

Save Lookup copies the data in the Lookup Window into a Lookup file. Lookup files can be stored either as a binary file with the .LKT filename extension or as an ASCII file with a .TXT or .CSV filename extensions. The Lookup file can be later read with the Open

Lookup Table command or used directly from the disk with the **Interpolate**, **Differentiate, Lookup**, **LookupRow**, and **LookupCol** functions. The information in the Lookup Table Window is also stored with the problem information when the Save or Save As commands are given. See Chapter 4, Using Lookup Files and the Lookup Table, for more information relating to Lookup tables and files.

Insert/Delete Lookup Rows and Insert/Delete Lookup Cols allows one or more rows or columns to be inserted or deleted at a specified position in an existing Lookup Table. Rows and columns can also be inserted by clicking the right mouse button in the header column (for rows) or header row (for columns) followed by selection of Insert or Delete from the table popup menu.

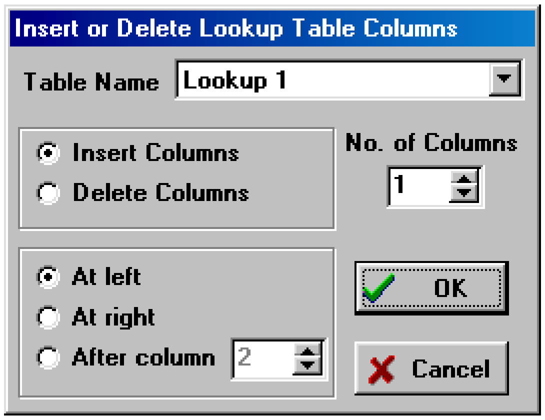

Insert or Delete Lookup Table Columns

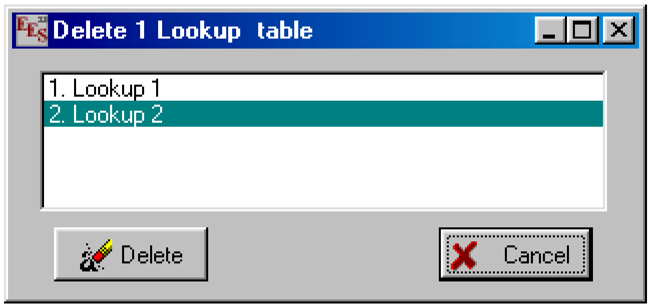
Table Name Lookup 1

[✓ Insert Columns] Delete Columns

No. of Columns 1

[✓ At left] At right After column 2

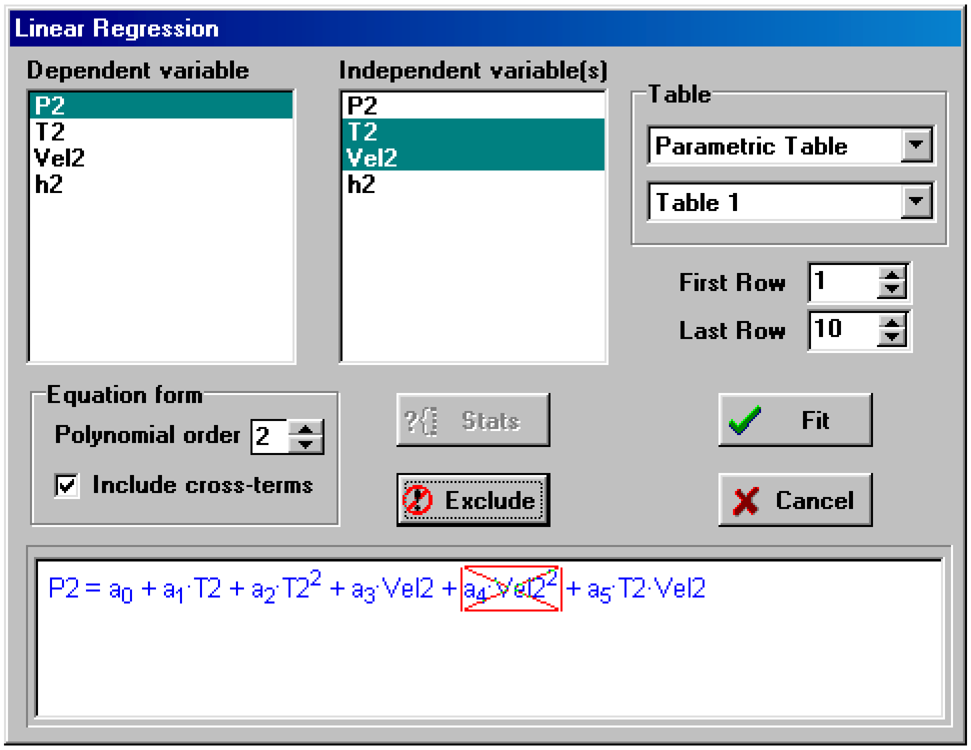
Delete Lookup will delete one or more selected Lookup Tables from the Lookup Table Window and recover the memory it required. There is no Undo for this operation.


Delete 1 Lookup table

1. Lookup 1
2. Lookup 2

Linear Regression provides regression capability for the data in the Parametric, Lookup, or Arrays tables. Note that the Curve Fit command in the Plot menu also provides regression capability but only for one independent variable. With the Linear Regression command, the data in any column can be regressed as a function of the data in up to 6 other columns.

The dialog window shown below appears after the command is chosen. Select the table you wish to operate on from the drop down lists at the upper right and the starting and stopping rows in that table. Specify the dependent variable by clicking on the variable name in the list on the left. The independent variable(s) are selected by clicking on the names in the right list. To de-select an item, click it a second time.



Dependent variable: P2 (selected), T2, Vel2, h2

Independent variable(s): P2, T2 (selected), Vel2 (selected), h2

Table: Parametric Table; Table 1

First Row: 1

Last Row: 10

Equation form

Polynomial order: 2

☑ Include cross-terms

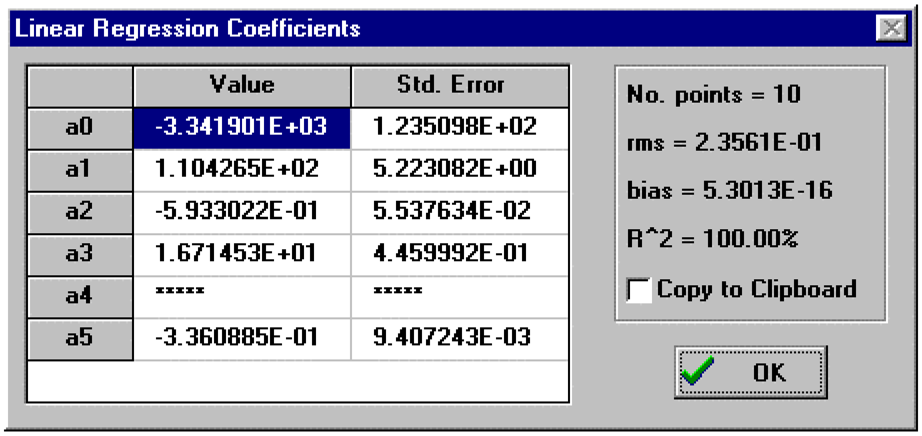
$P2 = a_0 + a_1 \cdot T2 + a_2 \cdot T2^2 + a_3 \cdot Vel2 + \boxed{\cancel{a_4 \cdot Vel2^2}} + a_5 \cdot T2 \cdot Vel2$

The dependent variable will be represented as a linear polynomial function of the independent variables. The order of the polynomial is set between 0 and 6 by clicking on the 'spin button' up or down arrows. If the cross-terms box is selected then terms involving the product of the independent variables will be included in the correlation. As any information relating to the equation form is entered, a representation of the equation to be fit is displayed in the box at the bottom as shown above.

You may exclude some terms from the regression by clicking on the term. This action will display a box around the selected term and enable the Exclude button. Click the Exclude button to remove the term from further consideration. A removed term is displayed within a crossed-out red box as shown above. If you later wish to include an excluded term, click on it. The Exclude button will then be titled Include. Click the Include button.

When the form of the equation is that which you want to fit, click the Fit button. If the fitting process is successful, the fitted equation will appear in the display box. The Stats button will been enabled. Clicking the Stats button will provide a table listing all of the coefficients, their associated standard errors, and other statistics such as the root-mean-square (rms) error, the bias error, and the R^2 value, as shown below. Coefficients which have been excluded will be represented in the table with stars. The coefficients can be copied to the clipboard by checking the Copy to clipboard box.

After a successful fitting process, the Fit button in the Linear Regression dialog window will be changed to Copy and the Cancel button will changed to Done. Either button will dismiss the dialog window. The Copy button will first copy the fitted equation to the clipboard. You can then paste this equation into the EES Equations window or into any other application that accepts text. Note, however, that the Copy process will overwrite any other information in the clipboard, such as the coefficients copied from the Linear Regression Coefficients dialog window.



	Value	Std. Error
a0	-3.341901E+03	1.235098E+02
a1	1.104265E+02	5.223082E+00
a2	-5.933022E-01	5.537634E-02
a3	1.671453E+01	4.459992E-01
a4	*****	*****
a5	-3.360885E-01	9.407243E-03

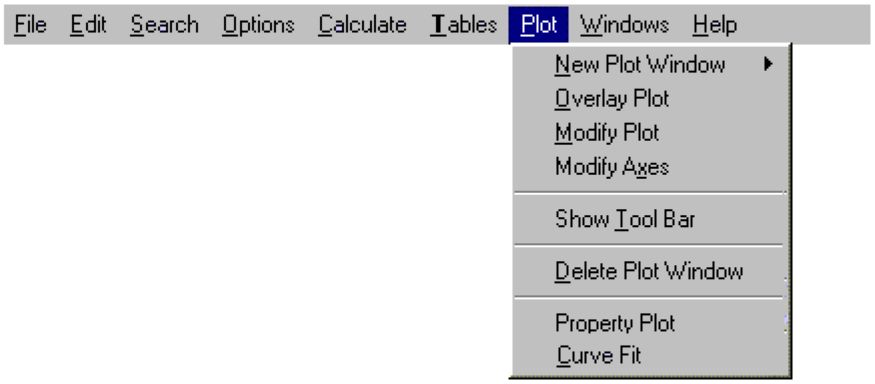
No. points = 10

rms = 2.3561E-01

bias = 5.3013E-16

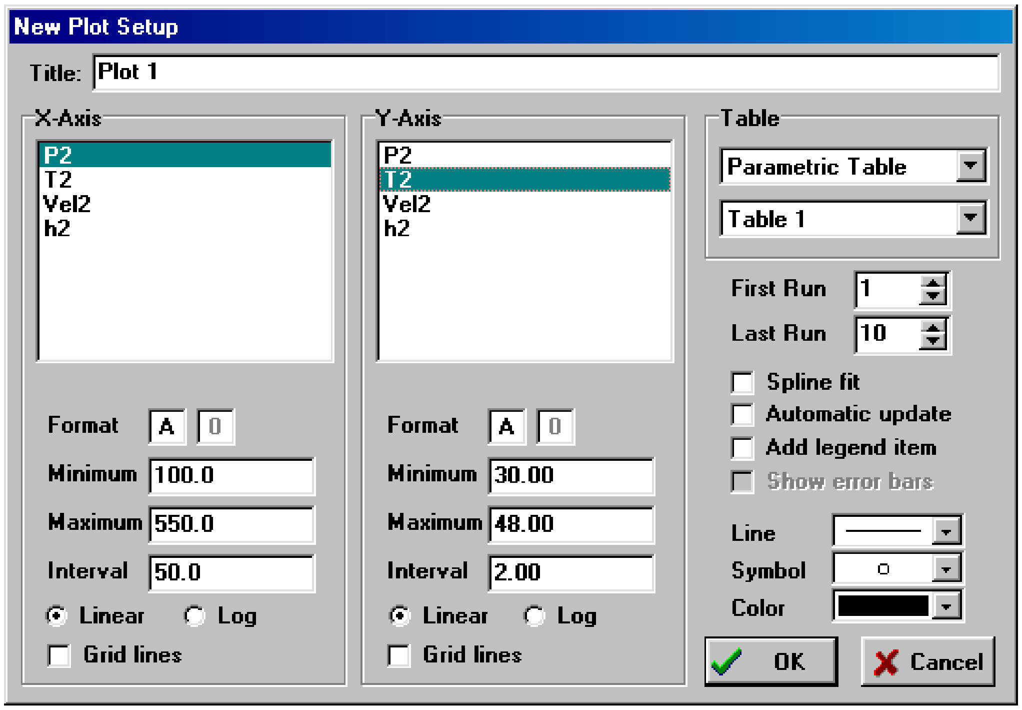
R^2 = 100.00%

The Plot Menu



New Plot Window will display a child menu to generate an X-Y, bar, or contour plot involving one or more variables defined in the Parametric, Lookup, Arrays, or Integrals Tables as a function of any other variable in that table. Use the Overlay Plot command if you wish to plot in an existing plot window. There is no intrinsic limit on the allowable number of plot windows. Each plot created with this command is placed in a separate window of the Plot Window. All plot windows are saved with other program file information when the Save or Save As commands are applied. Once created, the plots can be modified or copied using the Plot Window controls. Each plot window can have of plot overlays that are created with the Overlay Plot command. The information needed to produce the plot is specified in the New Plot Window dialog. All of the information provided in this dialog window can later be changed using Modify Axes and Modify Plot commands and the Plot Window controls described in Chapter 2.

X-Y and Bar Plots


New Plot Setup

Title: Plot 1

X-Axis: P2, T2, Vel2, h2
Format A 0
Minimum 100.0
Maximum 550.0
Interval 50.0
[✓ Linear] Log
Grid lines

Y-Axis: P2, T2, Vel2, h2
Format A 0
Minimum 30.00
Maximum 48.00
Interval 2.00
[✓ Linear] Log
Grid lines

Table: Parametric Table
Table 1
First Run 1
Last Run 10
Spline fit
Automatic update
Add legend item
Show error bars
Line
Symbol ○
Color ■

OK Cancel

To create an X-Y or bar plot, first select the table from which you wish to plot using the drop-down list controls at the upper right of the window. Enter a name for the plot in the edit box at the top of the dialog. This name will appear on the tab at the top of the Plot Window. The name can later be changed by right-clicking on the tab. The variables to be plotted on the X and Y axes are selected from the lists by clicking on their names, using the scroll bar or up/down arrow keys, if necessary to bring the variable names into view. One or more Y-axis variables can be selected. Clicking on an unselected variable name selects that variable, while clicking on a selected variable unselects it. A separate plot line will be generated for each selected Y-axis variable. All selected variables will be plotted with the same axis scale. EES will automatically select appropriate values for the number of display digits, the minimum and maximum axis values, and the interval when a variable is selected. All of these axis formatting variables may be changed.

The two fields to the right of the word Format contain pop-up menus that control the format of the numbers appearing in the scale for each axis. F and E format the numbers with a fixed number of decimal places or exponential notation, respectively. The number in the second field is the number of decimal places (for fixed notation) or significant figures (for exponential notation).

Grid lines will be drawn if the ‘Grid Lines’ checkbox control is selected. The number of grid lines and scale numbers is determined by the specified interval value.

The plot may be formatted in a variety of ways. Clicking the spin box arrows to the right side of the Line list box toggles the display through a list of the available line types. Click on the desired line type or make a selection with the up and down arrow keys. The plot symbol and line color are chosen in a similar manner. If more than one Y-axis variable is selected, the symbol and color list boxes will display 'auto'. With the 'auto' option, the symbol and color for each plot line will be automatically selected by EES so that each plot line has a different symbol and color. This feature can be overridden by simply selecting the symbol and color.

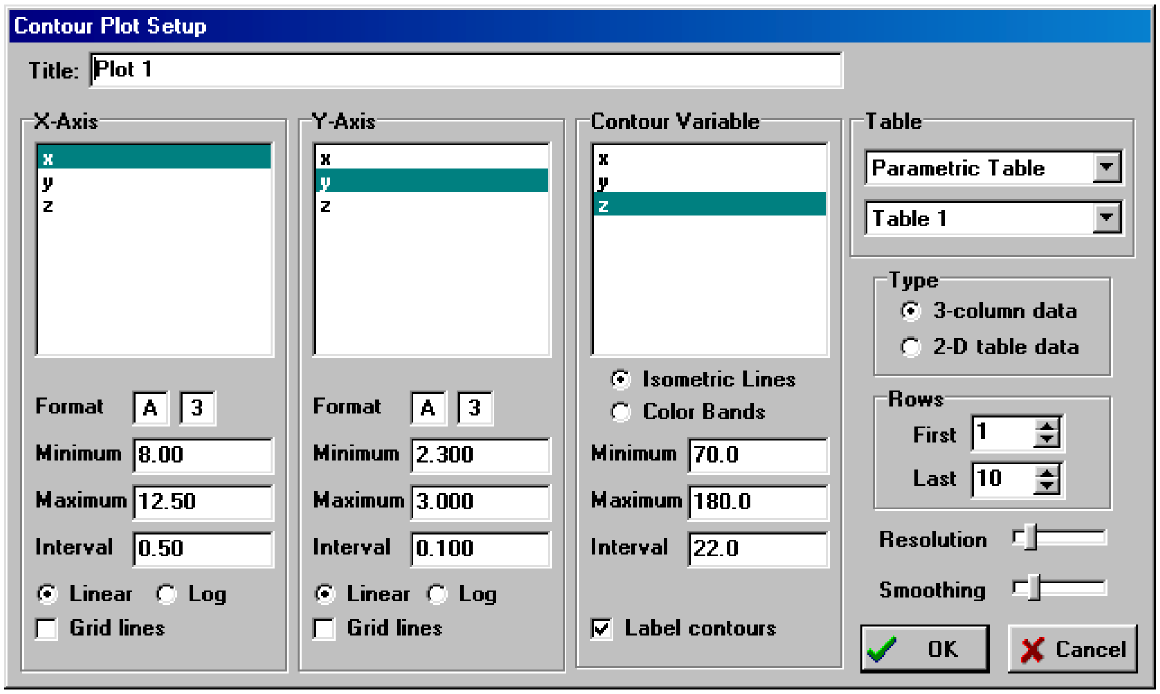
The ‘Spline fit’ control will provide a spline-fit curve through the plotted points.

When the ‘Automatic update’ control is selected, the plot will be generated using the current data in the Parametric Table, rather than the data which existed when the plot was first drawn. The plot will be updated as the data in the table are changed. The Data button allows the number of rows and the X and Y axis variables to be viewed or changed.

If the ‘Add legend item’ is selected, a text item having the name of the Y-axis variable will be placed at the upper left corner of the plot, preceded by the line and symbol type used for the plot. The legend item text can be moved, changed, or deleted just as any plot window text item, as described in the Plot Window section of Chapter 2.

The "Show error bars" control is accessible only if one or both of the variables selected for plotting have associated uncertainty values specified with the Uncertainty Propagation Table commands.

Contour Plots


Contour Plot Setup

Title: Plot 1

X-Axis: [✓ x] y z
Format A 3
Minimum 8.00
Maximum 12.50
Interval 0.50
[✓ Linear] Log
Grid lines

Y-Axis: x [✓ y] z
Format A 3
Minimum 2.300
Maximum 3.000
Interval 0.100
[✓ Linear] Log
Grid lines

Contour Variable: x y [✓ z]
[✓ Isometric Lines] Color Bands
Minimum 70.0
Maximum 180.0
Interval 22.0
[✓ Label contours]

Table: Parametric Table; Table 1

Type: [✓ 3-column data] 2-D table data

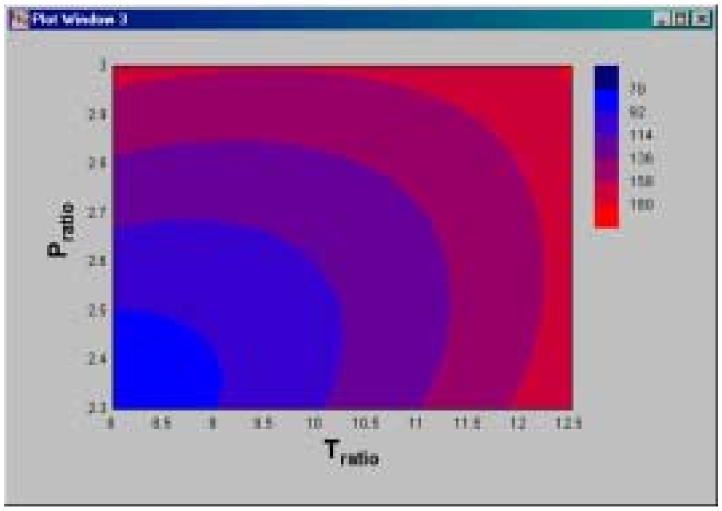
Rows: First 1, Last 10

Resolution

Smoothing

OK Cancel

The Contour Plot option generates lines or color bands indicating the path of a fixed value of the contour variable (Z) in X-Y space. A contour plot requires three-dimensional data for construction. The data may be provided in either of two ways from the Parametric, Lookup or Arrays table windows. If the 3-column data radio button is selected, EES will expect that three dimensional data to be provided in three columns from one of the table windows. The X, Y, and Z variables are then selected from the three lists. If the 2-D table data radio button is selected, EES will assume the column numbers in the table are proportional to the X-coordinate and the rows are proportional to the Y-coordinate. The Z-values will be read from the table. Controls are provided to allow a subset of rows and/or columns in the table to be plotted. The X and Y axis scaling is, by default, set to the number of columns and rows in the table, but it can be changed to provide scaling.



To prepare the contour plot, the selected data must be interpolated or extrapolated as necessary to provide values of the contour variable Z over the entire range of X and Y at relatively fine intervals. A number of methods were investigated for this task. The method which was found to be most successful is the multiquadric radial basis function. With this approach, the selected data are first scaled and then fitted to a function of the form

$$Z(x, y) = \sum_{j=1}^{N} w_i \sqrt{(x_j - x)^2 + (y_j - y)^2 + \sigma^2}$$

where N is the number of data points, the w values are weighting factors, and s is a relative smoothing parameter. The relative (scaled) value of s is chosen with the Smoothing slider which has a range between 0 and 5. Larger values of s cause the contour lines to be more damped. This parameter can significantly change the appearance of the contour plot, particularly in regions in which extrapolation is necessary. You may need to experiment with different values of this parameter to produce a satisfactory plot. The default value of 1 seems to work well in most problems.

The weights are found by linear least square fitting. There is a practical upper limit to the number of points that can be fit and this upper limit is controlled by the Resolution slider. The range of the Resolution slider is 100 to 1024. If more data points are provided than are specified by the Resolution slider, EES will select a subset of the data using a clustering algorithm. Even so, the contour plot algorithm involves extensive calculations and you may have to wait a while for the plot to be prepared, depending on the number of data points provided and the speed of your computer.

The contour plot may be produced using isometric lines or color bands representing constant values of Z. The number of lines or bands is determined as the difference between the maximum and minimum values divided by the interval value. Each isometric line is a separate plot overlay, whereas the color band plot is simply one plot. A maximum of 25 isometric lines and 250 colors is allowed. Note that selecting a large number of colors (>100) results in a plot that has nearly continuous colors ranging from blue (low values) to red (high values). The resulting plot can be quite pretty, but the time required to display the plot will increase with the number of colors.

A Label Contours check box control is provided for isometric contour plots. In this case, EES will generate a text item containing the value for each contour line and place it on the line. These text items can be moved or changed just as any other Plot window text items. If a banded color plot is chosen, an Include Legend check box is provided. If the Include Legend check box is selected, a color legend will be displayed to the right of the plot relating colors to the values of Z. Note that, on a color band contour plot, you can read the value of Z directly in the Plot Window title bar at any X-Y position by holding the ***Ctrl*** key down and moving the mouse.

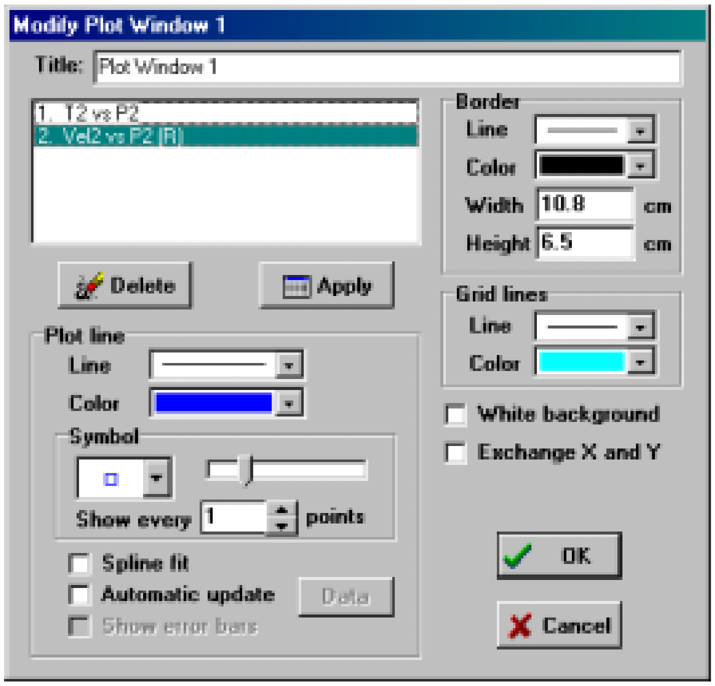
An X-Y or bar plot can be overlayed on the contour plot using the Overlay Plot command.

Overlay Plot allows a new plot curve to be drawn over existing plots. The use of this command is identical to that for the New Plot command described above except that it does not first clear the Plot window. All overlaid plots must share the same x-axis scaling. If the y-axis scale specified for a plot overlay differs from that of the first plot, a control will be displayed to select the existing scale on the left-hand y-axis or a new scale on the right-hand y-axis. Following plot overlays may then use either the left or right y-axis scale, depending on the choice of the axis selection control.

Modify Plot allows the characteristics of existing plot curves to be changed by manipulation of information in the following dialog window. This command can also be invoked by double-clicking the mouse button within the plot rectangle.

The plot for which changes are to be made is selected from the list at the upper left. The plots appear in this list in the order in which they were generated. An (R) following the plot descriptor indicates that this plot uses the right-hand y-axis scale.

The line type, symbol, symbol size, symbol frequency, and color of the plot curve can be changed using the drop-down lists at the lower left. The ‘Spline fit’ and ‘Automatic update’ options may be changed. (See the New Plot Window command for a description of these options.)



Modify Plot Window 1

Title: Plot Window 1

1. T2 vs P2
2. Vel2 vs P2 (R)

Border
Line
Color
Width 10.8 cm
Height 6.5 cm

Grid lines
Line
Color

Delete Apply

Plot line
Line
Color
Symbol
Show every 1 points

Spline fit
Automatic update
Show error bars

Data

White background
Exchange X and Y

OK
Cancel

Controls are provided to change the plot window title, the size and characteristics of the plot border and the grid lines. Click the Apply button after changing the plot line characteristics and before selected another plot. The plot window will immediately show any changes.

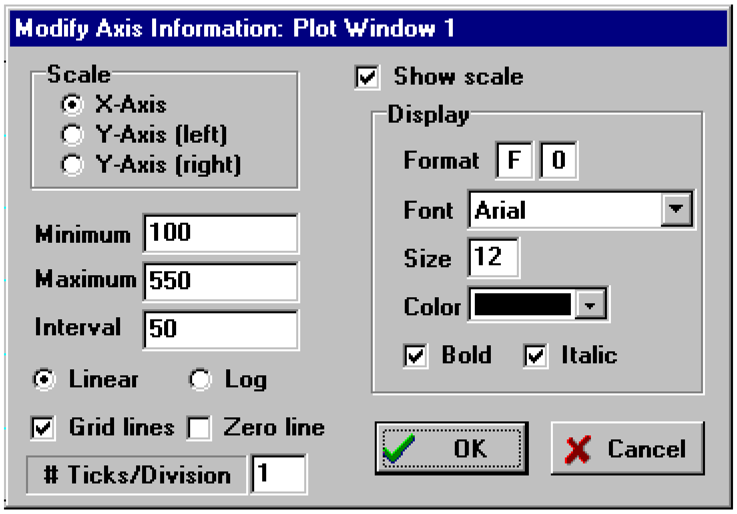
A single plot curve may be selectively deleted (leaving all other plots intact) using the Delete button. Legend text for the plot is also deleted. The Delete Plot Window command described below will delete an entire plot window, including all overlays.

Modify Axes allows the appearance of the axes of an existing plot to be changed. This command can also be invoked by double-clicking the mouse on the axis scale for which changes are to be made. The dialog window shown below will appear. The axis for which changes are to be made is selected with the radio-button controls at the upper left. The current minimum, maximum, and interval values for the selected axis are shown. These values can be changed and the plot will be redrawn, scaled with the new values.

The #Ticks/Division is the number of minor tick marks in each interval. If selected, Grid lines are normally placed at each major tick mark. However, the #Ticks/Division toggles to #Grids/Division if you click on this control. Grid lines can be placed at positions between the major ticks by setting the No. Grids/Division to a value greater than 0.

The display format, font, font size, font style, and color of the scale numbers can be changed using the drop-down menus appearing on the right of the dialog window. These fields will be hidden if the Show Scale checkbox is not selected, in which case a scale will not be drawn.

Clicking the Apply button applies the changes so that they can be viewed in the Plot window. The OK button accepts the changes and exits the dialog. The Cancel button will restore the plot to the condition it was in before this command was issued.



Scale: [✓ X-Axis] Y-Axis (left) Y-Axis (right)
Minimum 100
Maximum 550
Interval 50
[✓ Linear] Log
[✓ Grid lines] Zero line
Ticks/Division 1
[✓ Show scale]
Format F 0
Font Arial
Size 12
Color ■
[✓ Bold] [✓ Italic]

Show/Hide Toolbar controls the visibility of the tool bar that is provided with each plot window to add new text or drawing items to the plot window or to manipulate these items. The toolbar is ordinarily displayed when the plot window is created. It can be hidden by clicking in the small box with the cross at the upper right of the toolbar. The Show Toolbar command displays the toolbar if it was previously hidden. If the toolbar is visible, this command will have a Hide Toolbar caption and if selected, it will hide the toolbar. The toolbar contains buttons to add text, add lines, add rectangles, add ellipses and to align selected items.

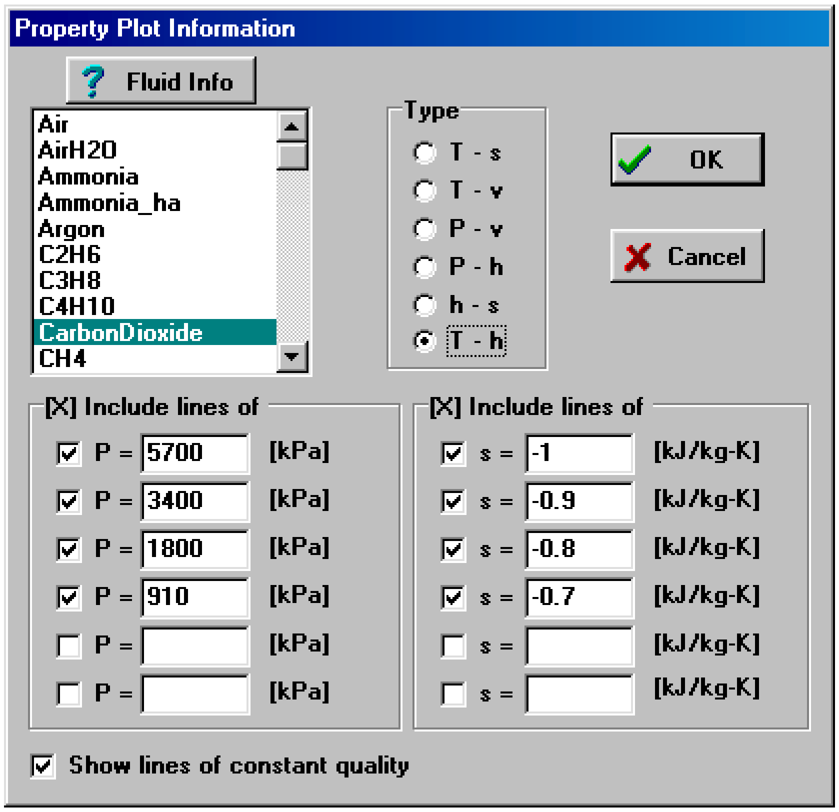
Delete Plot Window will delete the entire contents of the selected Plot Window. Use the Delete button in the Modify Plot dialog window if you wish to delete only one of several overlayed plots.

Property Plot creates a new plot window with thermodynamic property data for a selected substance. Once created, additional property data or thermodynamic cycle state points can be superimposed on the plot using the Overlay Plot command. Also, the plot characteristics and axis scales can be modified in the usual manner with the Modify Axes and Modify Plot commands.

Select the substance from the list at the left. The substance type (real fluid or ideal gas) is shown above the list. The general rule is that the substance is modeled as a real fluid if its name is spelled out (e.g., Oxygen) and as an ideal gas if its name is a chemical formula (e.g., O2). Air is the exception to this rule.

For all substances except AIRH2O (psychrometric air-water mixtures), there are buttons to allow specification of the axes for the property plot. The AIRH2O substance provides a field in which the total pressure can be specified.

Controls are provided to allow specification of up to six isobars, isotherms or isentropes. A line with the value shown in a box will be superimposed on the plot. Suggested values are provided. If you do not wish to display this line, click on the check box preceding the value.



Air
AirH2O
Ammonia
Ammonia_ha
Argon
C2H6
C3H8
C4H10
CarbonDioxide
CH4

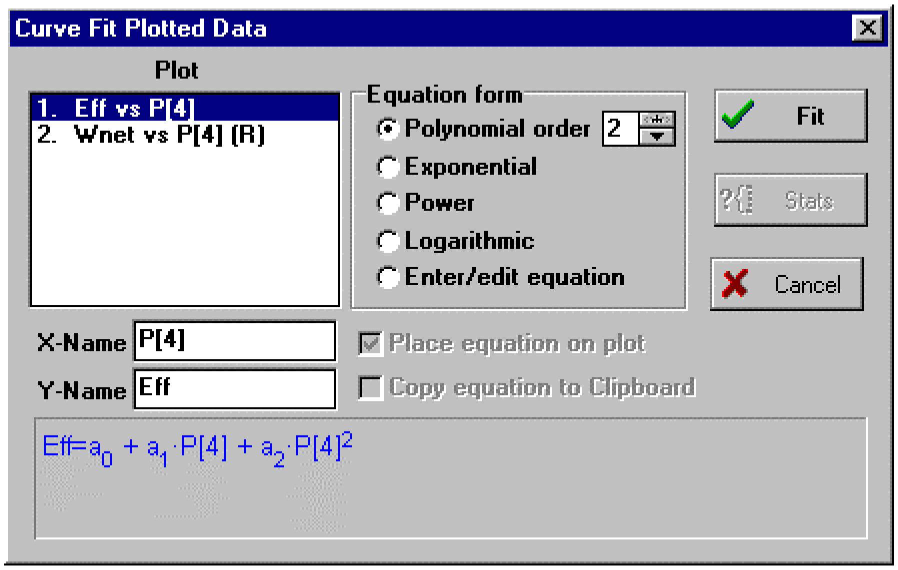
Type: ( ) T - s ( ) T - v ( ) P - v ( ) P - h ( ) h - s [● T - h]

[X] Include lines of			[X] Include lines of		
☑	P = 5700	[kPa]	☑	s = -1	[kJ/kg-K]
☑	P = 3400	[kPa]	☑	s = -0.9	[kJ/kg-K]
☑	P = 1800	[kPa]	☑	s = -0.8	[kJ/kg-K]
☑	P = 910	[kPa]	☑	s = -0.7	[kJ/kg-K]
☐	P =	[kPa]	☐	s =	[kJ/kg-K]
☐	P =	[kPa]	☐	s =	[kJ/kg-K]

☑ Show lines of constant quality

Curve Fit will find the best fit of a smooth curve through a previously plotted set of data points using unweighted least squares. (The Curve fit dialog provides a fit with a single independent variable. The Linear Regression command in the table menu allows a variable to be fitted with as many as 6 independent variables.) The dialog window shown below will appear.

Chose the data to be fitted from the list of plots at the left. Note that data can be plotted from the Parametric Table, the Lookup Table, or the Arrays Table with the New Plot or Overlay Plot commands. Select the form of the curve fit by clicking the appropriate radio button. A sample of the equation form will appear in blue in the box at the bottom of the dialog window. The first four buttons correspond to commonly used equation forms for which linear least squares can be used to determine the unknown coefficients. The Enter/edit equation button allows you to enter any equation form or to edit a previously entered equation. The equation you enter may be linear or non-linear in the unknown parameters. You will prompted to supply guess values and bounds for the unknown parameters.



1. Eff vs P[4]
2. Wnet vs P[4] (R)

Equation form: [✓ Polynomial order 2] Exponential, Power, Logarithmic, Enter/edit equation

X-Name: P[4]

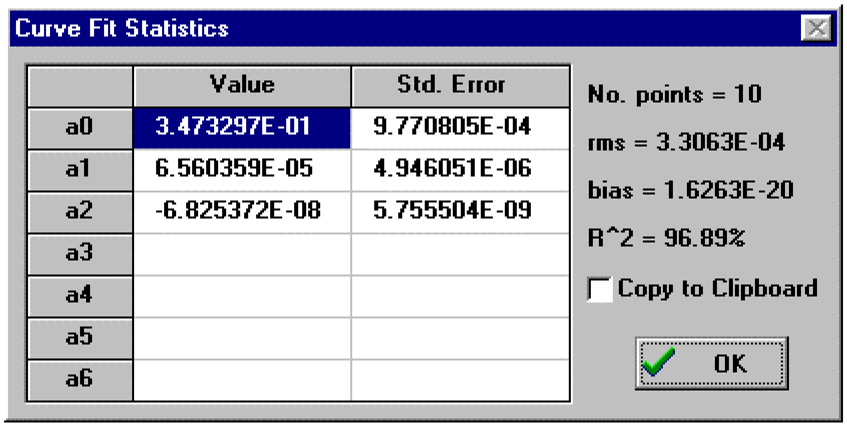
Y-Name: Eff

[✓ Place equation on plot]

Copy equation to Clipboard

$Eff=a_0 + a_1 \cdot P[4] + a_2 \cdot P[4]^2$

Click the Fit button (or press the Enter key). The fitted equation will be displayed in the box at the bottom of the dialog window. A Stats button will appear. Clicking the Stats button will display the following statistical information relating to the curve fit.



	Value	Std. Error
a0	3.473297E-01	9.770805E-04
a1	6.560359E-05	4.946051E-06
a2	-6.825372E-08	5.755504E-09
a3		
a4		
a5		
a6		

No. points = 10

rms = 3.3063E-04

bias = 1.6263E-20

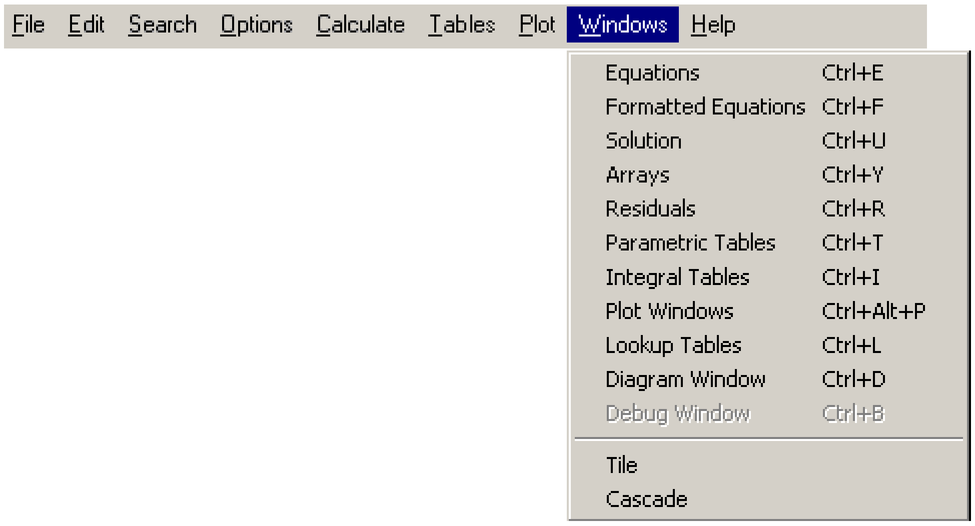
R^2 = 96.89%

Copy to Clipboard

Std. Error is the standard error of the curve-fitted parameter value; rms is the root mean square error of the fit; bias is the bias error of the fit. R^2 is the ratio of the sum of squares due to regression to the sum of square about the mean of the data.

The Fit button will now have changed into the Plot button. Click the Plot button if you wish to have the curve fit equation overlayed on your plot. If the Plot Legend check box is selected, a legend containing the equation will be created and displayed on the plot. The curve fit equation will be copied to the clipboard if the To Clipboard checkbox is selected when either the Plot or Cancel button is selected.

The Windows Menu



Equations causes the Equations window to become the active window by bringing it to the front of all other windows and making it visible if it were hidden previously.

Formatted Equations first checks the syntax of the equations and then brings the Formatted Equations window to the front displaying the contents of the Equations window in mathematical format.

Solution, Arrays and Residuals cause the Solution, Arrays, and Residual windows, respectively, to be moved to the front of all other windows. These windows are normally viewed after the Solve, Min/Max, or Uncertainty Propagation command has been successfully completed. Any change made to the Equations window will remove these windows from the screen if the Hide Solution after Change Option in the Preferences dialog (Options tab) is selected. If EES is unable to solve the equation set and terminates with an error, the name of the Solution window will be changed to Last Iteration Values and the values of the variables at the last iteration will be displayed in the Solution window; the residuals for the last iteration will be displayed in the Residuals window.

Plot Windows will bring the Plot Window to the front of all other windows. Each plot is identified by a title that appears on the tab displayed at the top of the Plot Window. This title is entered when the plot is first created. The title can be modified by right-clicking on the tab or by entering the modified title in the Modify Plot dialog. The tab position can be modified by right-clicking on the tab. There is no limit (other than memory limitations) to the number of plot windows that can be created and an unlimited number of overlayed plots may be drawn in each plot window using the Overlay Plot command. The menu item will be grayed if no plots have been created. The graphics in any of the plot windows can be copied to the Clipboard by selecting Copy from the Edit menu.

Parametric Table and **Lookup Table** bring the Parametric and Lookup Table windows, respectively, to the front of all other windows and make it the active window. There may be one or more tables in each of these window. Click on the tab at the top of the window to access the desired table. The Parametric and Lookup Table windows may be hidden by choosing close from the Windows control menu or by pressing Ctrl-F4.

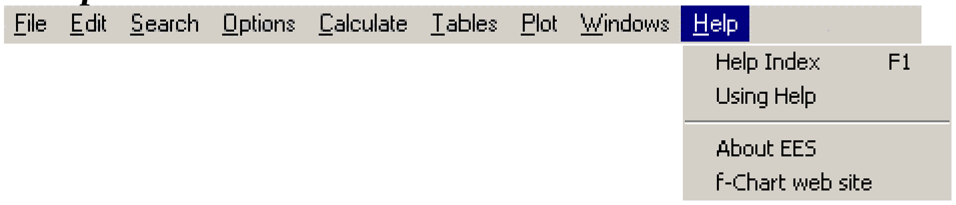
Diagram Window will bring the Diagram Window or a child Diagram Window to the front of all other windows. If one or more child Diagram Windows have been created, the Diagram Window menu option will display a 'fly-out' menu listing all of the Diagram and child Diagram windows. A diagram can be entered into EES from a drawing program or it can be created in EES using the Diagram Window drawing tools.

Debug Window will generally be disabled. This window is available only after the Solve command is attempted with the number of equations not equal to the number of variables. In this case, an option is presented to display the Debug Window and this menu item becomes enabled. Any change made to the Equations window will clear the Debug window and disable this menu item.

Tile arranges all open windows to fill the screen so that a portion of each can be viewed.

Cascade arranges the currently visible windows so that the title bar of each is shown.

The Help Menu



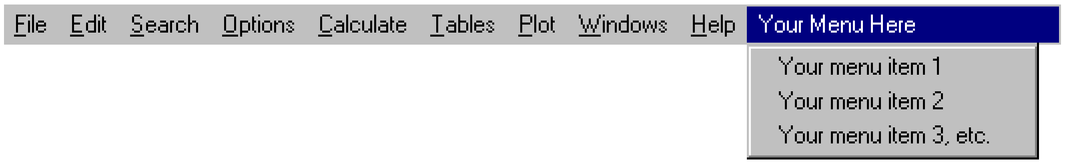
Help Index will activate the Help processor which provides specific information on the use of EES. The Help processor will open to the EES Information index which lists the subjects for which help is available. Clicking on the subject opens the Help window to information for that subject. Help can also be accessed by pressing the F1 key which will bring up help information specific to the window or dialog which is foremost. The on-line help provides most of the information contained in this manual.

Using Help shows information provided by the Windows Help processor on how to use the features in the Help program.

About EES will bring up the EES header window. This window registration information and indicates the version of your EES program. This information will be needed in any correspondence with F-Chart Software.

f-Chart web site will open your default browser program and set the URL to the f-Chart web site. The f-Chart web site has a "goodies' section with free examples and support programs for EES. The program developers can be contacted by e-mail through the web site.

The Textbook Menu



The Textbook menu is a user-defined menu that is designed to allow easy access to EES files. It has been used to provide a convenient means to access EES problems associated with a textbook, and thus its name. This menu can be created either by opening a textbook index file with the Load Textbook command (File Menu) or by placing the textbook index file in the USERLIB subdirectory.

A textbook index file is an ASCII file identified with the filename extension (.TXB). When EES reads a textbook index file, it creates the Textbook menu at the far right of the menu bar, as shown above. The format of the textbook index file is quite simple. The menu shown above was created with the following textbook index file.

```
Your Menu Here
1
Textbook information line 1
Textbook information line 2
Textbook information line 3
Reserved
>Your menu item 1
Descriptive problem name1 | FileName1.EES | HelpFile1.HLP | NO.BMP
Descriptive problem name2 | FileName2.EES | HelpFile2.HLP | NO.BMP
Descriptive problem name3 | FileName3.EES | HelpFile3.HLP | NO.BMP
>Your menu item 2
Descriptive problem name4 | FileName4.EES | HelpFile4.HLP | NO.BMP
Descriptive problem name5 | FileName5.EES | HelpFile5.HLP | NO.BMP
Descriptive problem name6 | FileName6.EES | NO.HLP | NO.BMP
>Your menu item 3, etc.
etc | FileName1.EES | HelpFile1.HLP | NO.BMP
```

The first line in the file is the menu title. This title is the name of the menu which will appear in the menu bar to the right of the Help menu. The following line is a version number used internally by EES. EES currently ignores this line but a 1 should be provided, as shown. The following three lines provide information about the textbook or problem set. This information will be displayed whenever any menu item is selected from the Textbook menu. A fourth line containing the word "reserved" is provided for possible future use. EES currently ignores the information on this line, but it must be provided. The following lines contain a menu item name (preceded by an identifying > character) and then one or more problem descriptions. The menu item name is the name that will appear in the Textbook menu list.

Each problem description line contains four pieces of information, separated by a | character. The first item is a descriptive name for the problem, which may be up to 128 characters. The second item is the filename for the EES program file. This filename may be partially or fully qualified with directory information, e.g., C:\myBook\Chapter1\Problem1.EES. However, in most cases, directory information is not necessary and it should not be included. The third item is an optional help file which is to be associated with this file. The help file can be an ASCII text file, a Windows HLP file produced by a Help formatting program, or an HTML file readable by a browser program. By convention, the help files have a .HLP or .HTM filename extension. If no help file is available, enter NO.HLP for this field. The final item is the filename for a figure associated with the problem. EES does not currently use the figure, but a figure name must be provided. Use NO.BMP as a placeholder.

The textbook index file should be placed in the same location (i.e., floppy disk, subdirectory, or folder) with all of the referenced EES program and help files. When the user selects a command from the Textbook menu, a dialog window will appear showing a list of the descriptive names of the problems for that menu item. The user can then select a name from the list and the file associated with that problem will be opened.

C H A P T E R 4

Built-in Functions

EES has a large library of built-in mathematical functions. Many of these (e.g., Bessel, hyperbolic, error functions, etc.) are particularly useful for engineering applications. EES also provides a function to convert units and functions that help manipulate complex numbers. The major feature that distinguishes EES from other equation solving programs, however, is its extensive library of built-in functions for thermophysical properties. Thermodynamic and transport properties of steam, R22, R134a, R407C, air, ammonia, carbon dioxide, and many others are implemented in a manner such that any independent set can be used to determine the remaining unknown properties. The first two sections of this chapter provide reference information for the built–in mathematical and thermophysical functions. EES also provides a Lookup Table which allows tabular data to be entered and used in the solution of the equation set. The third section provides information on the use of the Lookup Table. Much of the information provided in this chapter can also be obtained from within the program using the Info button in Function Info dialog.

Mathematical Functions

The mathematical functions built into EES are listed below in alphabetical order. (The functions which operate on the Lookup Table are described in the Using the Lookup Table section at the end of this chapter.) All of the functions (except **pi** and **tableRun#**) require one or more arguments which must be enclosed in parentheses and separated with commas. The argument may be a numerical value, a variable name, or an algebraic expression involving values and variables.

abs(X**)** returns the absolute value of the argument. In complex mode, **abs** returns the magnitude of the complex argument. (See also **Magnitude**(X));

angle(X**)**, **angleDeg(**X**)** and **angleRad(**X**)** all return the angle (also called amplitude or argument) of complex variable X. Representing X as X_r + i*X_i, this function returns arctan(X_i/X_r). Angle will return the angle in either degrees or radians depending on the Trig Function setting in the Unit System dialog. AngleDeg will always return the angle in degrees and AngleRad will always return the angle in radians. All three functions return the angle in the correct quadrant of the complex plane. Note that the Angle functions are used to extract the angle of a complex number variable or expression but they cannot be used for assigning the angle of a complex number. For example, the equation Angle(X)=4 will produce an error.

arcCos(X**)** returns the angle which has a cosine equal to the value of the argument. The units of the angle (degrees or radians) will depend on the unit choice made for trigonometric functions with the Unit System command.

arcCosh(X**)** returns the value which has a hyperbolic cosine equal to the value of the argument.

arcSin(X**)** returns the angle which has a sine equal to the value of the argument. The units of the angle (degrees or radians) will depend on the unit choice made for trigonometric functions with the Unit System command.

arcSinh(X**)** returns the value which has a hyperbolic sine equal to the value of the argument.

arcTan(X**)** returns the angle which has a tangent equal to the value of the argument. The units of the angle (degrees or radians) will depend on the unit choice made for trigonometric functions with the Unit System command.

arcTanh(X**)** returns the value which has a hyperbolic tangent equal to the value of the argument.

average(Arg1, Arg2, Arg3, ... **)** will return the average value of the arguments. The number of arguments must be between 1 and 1000. An array or subset of an array can be provided as an argument list by using array range notation, e.g., X[1..50].

bessel_I0(X**)** returns the value of zeroth-order Modified Bessel function of the first kind for argument value X where $-3.75 \le X <$ infinity.

bessel_I1(X**)** returns the value of first-order Modified Bessel function of the first kind for argument value X where $-3.75 \le X <$infinity.

bessel_J0(X**)** returns the value of zeroth-order Bessel function of the first kind for argument value X where $-3 \le X <$infinity.

bessel_J1(X**)** returns the value of first-order Bessel function of the first kind for argument value X where $-3 \le X <$infinity.

bessel_K0(X**)** returns the value of zeroth-order Modified Bessel function of the second kind for argument value X where $0 \le X <$infinity.

bessel_K1(X**)** returns the value of first-order Modified Bessel function of the second kind for argument value X where $0 \le X <$infinity.

bessel_Y0(X**)** returns the value of zeroth-order Bessel function of the second kind for argument value X where 0< X <infinity.

bessel_Y1(X**)** returns the value of first-order Bessel function of the second kind for argument value X where 0< X <infinity.

cis(X**)** is a complex mode function that returns cos(X)+i*sin(X). The required units (degrees or radians) of the angle is controlled by the unit choice made for trigonometric functions with the Unit System command. However, you can append deg or rad to the angle to override the Unit System setting. For example, V=3*cis(20deg) will set the value of complex variable V to have a magnitude of 3 and an angle of 20 degrees, regardless of the Unit System setting.

conj(X**)** returns the complex conjugate of a complex variable X. Representing X as X_r + i*X_i, this function returns X_r - i*X_i. Note the this function returns a complex result. The EES equation, Y=conj(X), will set the real part of Y (Y_r) to the real part of X and the imaginary part of Y to negative of the imaginary part (Y_i).

convert('From', 'To') returns the conversion factor needed to convert units from the unit designation specified in the 'From' string to that specified in the 'To' string. The single quote marks are optional. For example, FI = **convert**(ft^2, in^2) will set FI to a value of 144 because 1 square foot is 144 square inches. Combination of units and multiple unit terms may be entered. In a combination of units, such as Btu/hr-ft^2-R, the individual units are separated with dash (i.e., minus), a star, a dot (character Alt-250) or division symbols. Only one division symbol may be used in any one term. All units to the right of the division symbol are assumed to be in the denominator (i.e., raised to a negative power.) The ^ symbol is optional so ft2 and ft^2 are equivalent. The **convert** function will accept multiple unit terms if each term is enclosed within parentheses. Terms are separated with an optional * symbol or with a / symbol, as in the example below.

P =15* **Convert**((lbm/ft3)*(ft)/(s^2/ft), kPa)

The defined unit symbols can be displayed with the Unit Conversion Info command in the Options menu. If you find that a unit you need is not defined, you can enter it by editing the UNITS.TXT file in the EES directory.

ConvertTemp('C', 'F', T) converts temperatures from one scale to another. Four scales are supported: Celsius (C), Kelvin (K), Farenheit (F), and Rankine (R). The first two parameters are string constants or string variables which must be C, K, F, or R. Both upper and lower case letters are permitted. The single quotes surrounding the string constants are not required. The third parameter is a temperature in the scale indicated by the first parameter. The function returns the temperature in the scale indicated by the second parameter. Example: TF=convertTemp('C', 'F', 100) sets TF to 212.

cos(X**)** will return the cosine of the angle provided as the argument. The required units (degrees or radians) of the angle is controlled by the unit choice made for trigonometric functions with the Unit System command.

cosh(X**)** will return the hyperbolic cosine of the value provided as the argument.

differentiate('Filename', 'ColName1', 'ColName2', ColName2=Value**)** returns the derivative determined from two columns of tabular data based on cubic interpolation. See the ***Using Lookup files and the Lookup Table*** section of this chapter for more information and examples.

erf(X**)** returns the Gaussian Error function of X.

erfc(X**)** returns the complement of the Gaussian Error function of X which is 1-erf(X).

exp(X**)** will return the value e raised to the power of the argument X.

if (A, B, X, Y, Z**)** allows conditional assignment statements in the Equations window. If A<B; the function will return a value equal to the value supplied for X; if A=B, the function will return the value of Y; if A>B, the function will return the value of Z. In some problems, use of the **if** function may cause numerical oscillation. It is preferable to use the *if then else, repeat until* and *goto* statements in a function or procedure for conditional assignments. See Chapter 5 for additional information.

imag(X**)** returns the imaginary part of a complex variable X. Representing X as X_r + i*X_i, this function returns X_i. The **Imag** function cannot be used for assigning the imaginary part of a complex number. For example, the equation Imag(X)=4 will produce an error. Instead you should just enter X=4*i which will set X to 0 + 4*i. If you wish to only set the imaginary part of X, you can enter X_i=4.0

integral(Integrand, VarName**)** or **integral**(Integrand, VarName, Start, Stop, Step) returns the integral of the expression represented by Integrand with respect to the variable VarName, i.e., ∫(Integrand) d(VarName). There are two basic forms of the integral function which differ in their reliance on the Parametric table. If the values of Start, Stop, and Step are not provided, the **integral** function is used only in conjunction with the Parametric Table. In this case, VarName must be a legal variable name which has values defined in one of the columns of the Parametric table and Integrand can be a variable or any algebraic expression involving VarName and other variables or values. If Start, Stop, and (optionally) Step are provided, EES will numerically integrate all equations involving variable VarName, setting the value of VarName to values between Start and Stop as appropriate. If Step is not provided, EES will internally chose a step size using an automatic stepsize adjustment algorithm. The **integral** function can be used to solve initial value differential equations. See Chapter 7 for additional information.

integralValue(t,’X’) returns a value from the Integral Table that is created using the $IntegralTable directive. In that sense, the IntegralValue function is similar to the TableValue function which retrieves data from the Parametric Table and the Lookup and Interpolate functions which retrieve data from the Lookup Table window or Lookup files. t is the value of the independent integration variable for which the value of X is to be returned. X is the name of a variable that has been included in the Integral Table. (The single quotes around the variable name are optional.) The value provided for t must be less than or equal to the current value of the independent integration variable. Values of X corresponding to values of t that are not included in the integration range may not be properly defined.

interpolate('Filename', 'ColName1', 'ColName2', ColName2=Value**)** returns an interpolated or extrapolated value from tabular data in the Lookup table, a Lookup file (if filename is provided), or the Parametric table using cubic interpolation. See the ***Using Lookup files and the Lookup Table*** section of this chapter for more information and examples.

interpolate1('Filename', 'ColName1', 'ColName2', ColName2=Value**)** provides the same function as the **interpolate** command except that it uses linear interpolation.

interpolate2('Filename', 'ColName1', 'ColName2', ColName2=Value**)** provides the same function as the **interpolate** command except that it uses quadratic interpolation.

lookup(‘Filename’, Row, Column**)** returns the value in the Lookup Table or Lookup file at the specified row and column. Filename is optional. See ***Using Lookup files and the Lookup Table*** section of this chapter for more information and examples.

lookup$ function operates just like the **lookup** function except that it returns a string rather than a numerical value. As in the **lookup** function the, **lookup$** function can have two or three arguments. If three arguments are provided, the first is a string that provides the name of a Lookup table stored on disk. The next argument is a numerical value or expression that provides the row in the table. This row value should be an integer. Interpolation between rows is not allowed, as it is in the **lookup** function. The final parameter indicates the column. The column can be indicated by a numerical value or expression which provides the column number or by the name of the column provided in a string constant or string variable. In order to accept string information, the format style of the column in the Lookup table must be set to STRING. To change the format style, click in the column header and make the change in the Format Table dialog window.

lookupCol('Filename', Row, Value**)** uses the data in the specified row of the Lookup Table or Lookup file to determine the column which corresponds to the value supplied as the second argument. Filename is optional. See the ***Using Lookup files and the Lookup Table*** section of this chapter for more information and examples.

lookupRow('Filename', Col, Value**)** uses the data in the specified column of the Lookup Table or Lookup file to determine the row which corresponds to the value supplied as the second argument. Filename is optional. The column may be indicated in several ways. First, it may be entered as an integer value. Alternatively, it may be entered by providing the column name as a string constant or a string variable. An older format which is still accepted is to supply the column title preceded with the # symbol. The function will return the row in the Lookup Table corresponding to the value supplied as the second argument. The row value returned will not necessarily be an integer. Interpolation between rows will be provided as needed. See the ***Using Lookup files and the Lookup Table*** section of this chapter for more information and examples.

Lookup$Row operates just like the **LookupRow**function except that it its final argument is a string rather than a value. The function returns the row in the table in which this string exists. As in the **LookupRow**function the, **Lookup$Row** function can have two or three arguments. If three arguments are provided, the first is a string that provides the name of a Lookup table stored on disk. The next argument is the column in the table. The final argument is a string constant or string variable holding the string that will be searched in the table. Note. In order to accept string information, the format style of the column in the Lookup table must be set to STRING. To change the format style, click in the column header and make the change in the Format Table dialog window.

ln(X**)** will return the natural logarithm of the argument.

log10(X**)** will return the base 10 logarithm of the argument.

magnitude(X**)** returns the magnitude (also called modulus or absolute value) of a complex variable X. In complex mode, the **abs** function also returns the magnitude. Representing X as X_r + i*X_i, this function returns sqrt(X_r^2+X_i^2). The **magnitude** function cannot be used for assigning the magnitude of a complex number. For example, the equation magnitude(X)=4 will produce an error.

max(X1, X2, X3, …**)** will return the value of the largest of its arguments. The number of arguments must be greater or equal to 1.

min(X1, X2, X3, …**)** will return the value of the smallest of its arguments. The number of arguments must be greater or equal to 1.

ntableruns('name') takes one string argument which must be 'Parametric', 'Lookup', 'Arrays', or the name of a Lookup file stored on disk. In this last case, the name may be provided as a string constant or string variable. The function returns the number of rows in the specified table. If the specified table does not exist, the function returns zero. For example, **n=ntableruns('LOOKUP')** returns the number of rows in the Lookup Table.

pi is a reserved variable name which has the value of 3.1415927.

product(Arg, Series_info**)** returns the product of a series of terms. Arg can be any algebraic expression. Series_info provides the name of the product index variable and the lower and upper limits which must be integers or variables which have been previously set to integer values. **product** (j, j=1,4) will return 1*2*3*4 or 24, which is 4 factorial. The **product** function is most useful when used with array variables, e.g., X[j]. For example, the product of the square of all 10 elements in the vector X can be obtained as **product** (X[j]*X[j], j=1,10).

real(X**)** returns the real part of a complex variable X. Representing X as X_r + i*X_i, this function returns X_r. The **real** cannot be used for assigning the real part of a complex number. For example, the equation real(X)=4 will produce an error. Instead you should just enter X=4 which will set X to 4 + i*0. If you wish to only set the real part of variable X, you can enter X_r=4.

random(A,B**)** returns a uniformly distributed number in the range between A and B. This function can be used only within EES Functions and Procedures

round(X**)** will return a value equal to the nearest integer value of the argument.

sin(X**)** will return the sine of the angle provided as the argument. The required units (degrees or radians) of the angle is controlled by the unit choice made for trigonometric functions with the Unit System command.

sinh(X**)** will return the hyperbolic sine of the value provided as the argument.

sqrt(X**)** will return the square root of the value provided as the argument which must be greater than or equal to zero.

step(X**)** will return a value of 1 if the argument is greater than or equal to zero; otherwise the Step function will return zero. The **step** function can be used to provide conditional assignments, similar to the if function. The **step** and if functions are provided to maintain compatibility with earlier versions. Conditional assignments are more easily and clearly implemented with the IF THEN ELSE statement in functions or procedures as described in Chapter 5.

There are two forms for the **sum** function. EES determines which format is in use by context.

sum(Arg, Series_info**)** returns the sum of a series of terms, i.e., ΣArg. Arg can be any algebraic expression. Series_info provides the name of the summation index variable and the lower and upper limits. These limits must be integers or variables which have been previously set to integer values. The function is best explained by examples. **sum**(j, j=1,4) will return 1+2+3+4 or 10. The **sum** function is most useful when used with array variables, e.g., X[j]. For example, the scalar product of two vectors, X and Y, each with 10 elements can be obtained as **sum**(X[j]*Y[j], j=1,10). See Chapter 7 for information on how the **sum** function can be used with array variables to manipulate matrices.

sum(Arg1, Arg2, ...ArgN**)** returns the sum of the arguments. Array range notation is particularly convenient for the list form. For example, SUM(X[1..100]) returns the sum of the 100 elements in the X array.

tableName$ returns the name of Parametric Table that is currently being used in the calculations. Parametric table names are seen on the tabs at the top of the Parametric Table Window. The name of a Parametric Table can be changed by right-clicking on the tab. This function has no arguments and it should be used only when calculations are initiated with the Solve Table or Min/Max Table command in the Calculate menu.

tableRun# returns the Parametric Table run number, i.e., the current row in the Parametric Table or zero, if the Parametric Table is not being used in the calculations. This function should only be used with the Solve Table or Min/Max Table command in the Calculate menu.

tableValue(Row, Column**)** or **tableValue(**Row, 'VariableName'**)** returns the value stored in a specified row and column of the Parametric Table. The column number may be either entered directly as an integer number or indirectly by supplying the variable name for the desired column, enclosed by the single quotes, e.g., TableValue(6,'ABC')[7]. An error message will be generated if the row or column (or corresponding variable name) does not exist in the Parametric Table or if the referenced cell does not have a value. The

[7] For compatibility with earlier versions, EES will also accept the #symbol preceding the variable name in place of enclosing it within single quotes, e.g., TableValue(6, #ABC).

tableValue function is useful in the solution of some 'marching-solution' type problems in which the current value of a variable depends on its value in previous calculations.

tan(X**)** will return the tangent of the angle provided as the argument. The required units (degrees or radians) of the angle is controlled by the unit choice made for trigonometric functions with the Unit System command.

tanh(X**)** will return the hyperbolic tangent of the value provided as the argument.

trunc(X**)** will return a value equal to the integer value corresponding to the argument rounded toward zero.

UnitSystem(‘Unittype’**)** is a function which allows an EES program to know what unit settings have been selected with the Unit System command. This function takes one argument which must be placed within single quote marks. Legal arguments are 'SI', 'Eng', 'Mass', 'Molar', 'Deg', 'Rad', 'kPa', 'bar', 'psia', 'atm', 'C', 'K', 'F', and 'R'. The function returns either 1 (for true) or 0 (for false). As an example, the following assignment statement in an EES function or procedure,

```
g:=unitsystem('SI') + 32.2*unitsystem('Eng')
```

will set g equal to 1 if the user has selected the SI unit system and g equal to 32.2 if the user has selected the English unit system.

String Functions

EES supports both numerical and string variables. String variable are identified by a $ as the last character in the variable name, e.g., Fluid$. String constants are enclosed in single quote marks. The following functions operate on string constants and variables.

Concat$ accepts two arguments both of which must be either a string constant or a string variable. The function returns a single string that concatenates the two strings.
Example: R$=CONCAT$('R','22') {R$ will be set to 'R22'}

Copy$ creates a string that is a substring of the string expression provided as the first argument. The second argument indicates the character position at which the substring starts, and the third parameter is the substring length. If this length exceeds the length of the string expression, it is set to the length of the string expression.
Example: Neat$=COPY$('This is neat', 9, 255) {sets Neat$ to 'neat'}

**LowerCase$** accepts one string variable or string constant argument. It returns a string in which all letters in the string are converted to lowercase. It is not necessary to change the case of fluid names or file names as the functions that use this information are case insensitive. Example: E$=LowerCase$('EESy') {sets E$ to ‘eesy’}

String$ accepts one argument which can be a numerical constant, variable, or expression. The function returns a string representing the numerical value of the argument. The function uses Auto format to set the format of the value before converting it to a string. The StringVal function provides the inverse operation, converting a string to numerical value.
Example: R$=CONCAT$('R',String$(10+12)) {sets R$ ti 'R22'}

StringPos accepts two arguments both of which must be either be a string constant or string variable. The function returns the position of the first string within the second. If the first string is not contained within the second, the function returns 0. This function is case-sensitive. You may wish to use the LowerCase$ or UpperCase$ functions.
Example: p=StringPos(‘y’, ‘EESy does it’) {Sets p to 4}

StringVal returns the numerical value formed by the characters of a string constant or string variable. This function provides the inverse operation of the String$ function. This function can be applied with a string variable dropdown list on the Diagram window when it is desired to allow the user to select from a pre-determined set of numerical values.
Example: R$='22'; V=StringVal(R$) {Sets V to 22}

**Uppercase$** accepts one string variable or string constant argument. It returns a string in which all letters in the string are converted to uppercase. It is not necessary to change the case of fluid names or file names as the functions that use this information are case insensitive. Example: E$=UpperCase$('ees') {Sets E$ to 'EES'}

Thermophysical Property Functions

The first argument of all built-in thermophysical property functions is the name of the substance. The substance names recognized by EES are[8]:

Recognized Substance Names for Property Functions

Ideal Gas	Real Gas		
Air	Ammonia	R11	R404A
AirH2O [+]	Ammonia_ha[*]	R12	R407C
C2H6	Argon*	R13	R410A
C3H8	CarbonDioxide[*]	R14	R500
C4H10	Ethane[*]	R22	R502
CH4	Helium[*]	R22_ha[*]	R600
CO	Isobutane[*]	R23*	R600a
CO2	Methane	R32[*]	R717
H2	Methana_ha	R114a	R718
H2O	Oxygen[*]	R123	R744
N2	n-Butane	R134a	
NO2	n-Butane_ha[*]	R134a_ha[*]	
O2	Neon[*]	R141b*	
SO2	Nitrogen[*]	R152a	
	Propane	R290	
	Propane_ha[*]		
	Steam		
	Steam_IAPWS[#]		
	Steam_NBS[*]		
	Water		

[+] AirH2O provides psychrometric relations
[*] Fluid is represented with a high-accuracy equation of state. Refer to the online help for the specific publication the describes the equation of state.
[#] Steam_IAPWS implements high accuracy thermodynamic properties of water substance with the 1995 Formulation for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use, issued by The International Association for the Properties of Water and Steam (IAPWS). This correlation replaced the 1984 formulation of Haar, Gallagher, and Kell (NBS/NRC Steam Tables, Hemisphere Publishing Co., 1984) which is implemented in substance Steam_NBS. The new formulation is based on the correlations of Saul and Wagner (J. Phys. Chem. Ref. Data, 16, 893, 1987) with modifications to adjust to the International Temperature Scale of 1990. The modifications are described by Wagner and Pruss (J. Phys. Chem. Ref. Data, 22, 783, 1993). This correlation provides accurate results for temperatures between 273.15 K and 1273.15 K at pressures up to 1000 MPa. The formulation allows extrapolation of properties to 5000 K. Steam_IAPWS is available only in the Professional version.

[8] Your version of EES may have additional fluids. Property data may be added as described in Appendix C.

Additional substances may be provided in external files in the USERLIB subdirectory with a user-supplied property file having a .MHE filename extension, as described in Appendix C. Also provided in the USERLIB subdirectory are the external routines providing thermodynamic property data for lithium bromide-water mixtures (H_LIBR, T_LIBR, V_LIBR, Q_LIBR, P_LIBR, X_LIBR), ammonia-water mixtures (NH3H2O) and specific heat, enthalpy and entropy for hundreds of additional substances with JANAF table references (JANAF). Documentation for these routines is provided through the Function Info command in the Options menu. Click the External routines button at the top right and then select the external routine name from the list. Clicking the Function Info button will provide the documentation.

It may appear from the above list that some substances, e.g., *N2* and *Nitrogen*, *CO2*, *R134a*, and*R134a_ha*, *H2O*, *Steam*, *Water* and *Steam_NBS*, are duplicated, but this is not quite true. Whenever a chemical symbol notation (e.g., *N2*, *CO2*, *CH4* etc.) is used, the substance is modeled as an ideal gas and the enthalpy and entropy values are based on JANAF table references. The JANAF table reference for enthalpy is based on the elements having an enthalpy value of 0 at 298K (537R). The entropy of these substances is based on the Third Law of Thermodynamics. Whenever the substance name is spelled out (e.g., *Steam* (or *Water)*, *Nitrogen*, *R12*, *CarbonDioxide*, *Methane*, etc.) the substance is modeled as a real fluid with subcooled, saturated, and superheated phases. Exceptions to this rule occur for *Air* and *AirH2O*, both of which are modeled as ideal gases. *AirH2O* is the notation for air-water vapor mixtures, i.e., psychrometrics.

The property keywords *Water* and *Steam* are treated identically. Either keyword provides access to approximate water property functions based on empirical correlations which have been developed for rapid calculations. The *Steam/Water* property correlations assume the fluid is incompressible in the subcooled region; This assumption is not accurate for pressures above 350 atm and for states near the critical point. The *Steam_NBS* keyword uses property correlations published by Harr, Gallagher, and Kell (Hemisphere, 1984). These property correlations are accurate over a large range of conditions. However, they require considerably more computing effort than the Steam/Water relations. Steam_IAPWS provides the 1995 Formulation for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use which supercedes the formulations provided in Steam_NBS.

Fluids identified in the above table with * following their name are implemented using a high accuracy equation of state. In some cases, property data for the same fluids may also be implemented using a simpler equation of state that is less accurate near the critical region and in the subcooled regime. When both the high accuracy and simpler formulations appear for the same fluid, _ha is appended to the fluid name for the high accuracy formulation. Information concerning the source of the property data and the range of applicability is provided when the Fluid Info button in the Function Info dialog.

Many of the thermodynamic functions can take alternate sets of arguments. For example, the **enthalpy** function for steam can be accessed with temperature and pressure as arguments; alternatively, the same function could be accessed with entropy and quality as arguments. In general, any independent set of arguments can be supplied for thermodynamic functions.

All arguments in thermophysical property functions, aside from the substance name, are identified by a single case-insensitive letter followed by an equal sign. The value or algebraic expression representing the value of the argument follows the equal sign. The letters which are recognized in function arguments and their meaning are as follows:

Property Indicators
for Use in Thermophysical Functions

B = Wetbulb Temperature	**T** = Temperature
D = Dewpoint Temperature	**U** = Specific Internal Energy
H = Specific Enthalpy	**V** = Specific Volume
P = Pressure	**W** = Humidity Ratio
R = Relative Humidity	**X** = Quality
S = Specific Entropy	

Arguments must be separated with commas and may be in any order, provided that the substance name is first, as in the examples shown below. EES will display the function name in the format selected for Functions in the Display Options dialog window. The substance name is an EES keyword and it will be displayed in the format selected for Keywords in the Display Options dialog window

EES does not require the argument to a function to have a known value. For example:

h1 = enthalpy(STEAM, T=T1, P=P1)

will return the value of h1 corresponding to known temperature and pressure, T1 and P1. If, however, the value of h1 is known, but T1 is unknown, the same equation will return the appropriate value of the temperature. Alternatively, the temperature could be found by:

T1 = temperature(STEAM, h = h1, P=P1)

The latter method is preferable in that the iterative calculations implemented for thermodynamic properties are less likely to have convergence difficulty.

The built-in thermophysical property functions are listed below in alphabetical order. The units, which depend on the choices made with the Unit System command in the Options menu, are shown in brackets. One or more examples, showing the allowable formats for the function, are also given.

Conductivity [W/m-K, Btu/hr-ft-R] returns the thermal conductivity of the specified substance. For ideal gas substances, the Conductivity function takes one parameter in addition to the fluid name and this parameter must be temperature. For AirH2O (moist air), the temperature, pressure, and humidity ratio (or relative humidity) must be supplied as arguments. For real fluids, any two independent properties can be supplied, provided that the state does not consist of two phases.

Examples: k1 = conductivity (AIR, T=200)
k2 = conductivity (AMMONIA, T=100, P=200)
k3 = conductivity (STEAM_NBS, T=100, x=1)
k4 = conductivity (AIRH2O, T=80, P=14.7,R=0.5)

Density [kg/m^3, kgmole/m^3, lb/ft^3, lbmole/ft^3] returns the density of a specified substance. Two arguments are required for all pure substances; three are needed for moist air.

Example: d1 = **Density**(AIR, T=300, P=100)
d2 = **Density** (Steam, h=850, P=400)
d3 = **Density** (AirH2O, T=70, R=0.5, P=14.7)

DewPoint [°F, °C, R, K] returns the dewpoint temperature for air-water gas mixtures. This function can be used only with AIRH2O as the substance name. Three arguments follow the substance name in any order: temperature, total pressure, and relative humidity (or humidity ratio or wetbulb temperature).

Example: D1 = dewpoint(AIRH2O, T=70, P=14.7, w=0.010)
D2 = dewpoint(AIRH2O, T=70, P=14.7, R=0.5)
D3 = dewpoint(AIRH2O, T=70, P=14.7, B=50)

Enthalpy [kJ/kg, kJ/kgmole, Btu/lb Btu/lbmole] returns the specific enthalpy of a specified substance. The exact form of the enthalpy function depends on the substance and independent variable(s) selected. Substances which obey the ideal gas law, such as air, require a single argument, (temperature or internal energy), in addition to the substance name whereas real fluid substances, e.g., STEAM and CARBONDIOXIDE, will always require two independent variables. For AIRH2O, three arguments are required.

Example: h1 = enthalpy(AIR, T=300)
h2 = enthalpy(STEAM, T=900, P=300)
h3 = enthalpy(AIRH2O, T=70, P=14.7, R=0.50)

Entropy [kJ/kg-K, kJ/kgmole-K, Btu/lb-R, Btu/lbmole-R] returns the specific entropy of a specified substance. For all pure substances, the entropy function always requires two arguments, in addition to the substance name. For AIRH2O, three arguments are required.

Example: s1 = entropy(O2, T=400, P=100)
s2 = entropy(AIRH2O, T=70, P=14.7, R=0.50)

HumRat [dimensionless] returns the humidity ratio (defined as the mass of water vapor per mass of dry air) for air-water gas mixtures. This function is applicable only to the substance AIRH2O. The function requires three arguments which must include pressure and any two remaining independent variables such as temperature, relative humidity, enthalpy, or dew point.

Example: w1 = humRat(AIRH2O, T=70, P=14.7, R=0.50)
w2 = humRat(AIRH2O, T=70, P=14.7, h=25)

IntEnergy [kJ/kg, kJ/kgmole, Btu/lb, Btu/lbmole] returns the specific internal energy of a specified substance. The exact form of the **IntEnergy** function depends on the substance and independent variable(s) selected. Substances which obey the ideal gas law, such as air, require a single argument (temperature or enthalpy) whereas real fluid pure substances, like steam, will always require two arguments in addition to the substance name. AIRH2O requires three additional arguments.

Example: u1 = intEnergy(AIR, T=300)
u2 = intEnergy(STEAM, T=1320, P=300)
u3 = intEnergy(AIRH2O, T=70, P=14.7, R=0.50)

IsentropicIndex returns the dimensionless ratio of the constant pressure and constant volume specific heats. **IsentropicIndex** requires temperature as an input for ideal gas substances. Any two independent properties can be provided for real gases. EES does not provide a built-in function for the constant volume specific heat, C_v. However, it does provide a built-in function for the constant pressure specific heat, C_p, which is called SPECHEAT. C_v can be found with the **SpecHeat** and **IsentropicIndex** functions.

Example: k_1=IsentropicIndex(Air,T=100)
k_2=IsentropicIndex(Steam,T=100,P=50)

IsIdealGas is a very simple function that requires only one parameter and that is the name of the fluid. The function returns either 1 (meaning true) or 0 (meaning false). If the function returns true, the fluid is considered to behave according to the ideal gas law. The convention used in EES is that fluid names that are chemical symbols, e.g., O2, N2, and CO2, are modelled with the ideal gas law whereas fluid names that are spelled out, e.g., Oxygen, Nitrogen, and CarbonDioxide, are considered to be real fluids with subcooled, saturated and vapor phases. Air is an exception to this rule. The need for the **IsIdealGas**

function arises because the number of arguments may differ is the fluid is represented by the ideal gas law. Enthalpy, for example, is determined by temperature alone for O2, but temperature and another property such as pressure are needed to determine the enthalpy of Oxygen. If you are trying to write a general function to return a property of a fluid that is specified in a string variable, you may not know how many parameters it requires unless you know if the fluid is represented by the ideal gas law. That is the purpose of this function. Use an IF THEN ELSE statement testing on the value of **IsIdealGas** to branch to the correct form of the property function you wish to determine.

Example: B=IsIdealGas(O2)
C=IsIdealGas(R$)

MolarMass returns the molar mass (often called molecular weight) of the fluid provided as the parameter.

Example: M_CO2 =MolarMass(CarbonDioxide)

Pressure [kPa, bar, psia, atm] returns the pressure of a specified substance. The argument list for the pressure function always requires the substance name followed by two arguments, each item separated by commas. The pressure function is not implemented for AIRH2O; however, an unknown pressure can still be determined using any of the functions which are applicable to moist air and which take pressure as an argument.

Example: P1 = pressure(STEAM, h=1450, T=900)

P_Crit [kPa, bar, psia, atm] returns the critical pressure of the specified fluid. The fluid may be a fluid name or a string variable. Critical property information is not available for ideal gas substances.

Example: Pc=P_Crit(R134a) "returns the critical pressure of R134a"

Prandtl returns the dimensionless Prandtl number for the specified fluid defined as $\Pr = \frac{\mu c_p}{k}$ where μ is the viscosity, c_p is the specific heat, and k is the thermal conductivity. The Prandtl number requires temperature as an input for ideal gas substances and temperature and pressure for real substances.

Example: Pr_1=Prandtl(Air,T=100)
Pr_2=Prandtl(Steam,T=100,P=50)

Quality [dimensionless] returns the quality (vapor mass fraction) for substances modeled as real fluids such as WATER and R12. Two independent arguments are required. Temperature and pressure are not independent for saturated states. If the state of the substance is found to be subcooled, the quality is returned as –100. If it is superheated, 100 is returned.

Example: x1 = quality(R12, h=50, T=80)

Relhum [dimensionless] returns the relative humidity as a fractional number for air-water gas mixtures. There are three arguments to this function, in addition to the substance name, AIRH2O. The three arguments are temperature, total pressure and any two remaining independent variables such as temperature, wetbulb, enthalpy, dew point, or humidity ratio.

Example: R1 = relhum(AIRH2O, T=70, P=14.7, w=0.01)
R2 = relhum(AIRH2O, T=70, P=14.7, h=25)
R3 = relhum(AIRH2O, T=70, P=14.7, B=55)

Specheat [kJ/kg-K, kJ/kgmole-K, Btu/lb-R, Btu/lbmole-R] returns the constant pressure specific heat of the specified substance. For pure substances which obey the ideal gas law, the specific heat function has temperature as its only other argument in addition to the substance name. The temperature and pressure must both be provided as arguments for substances modeled as real fluids. The specific heat of the liquid or vapor may be returned, depending on the temperature and pressure values provided.

Example: Cp1 = specheat(AIR, T=350)
Cp2 = specheat (AMMONIA, T=100, P=30)

SurfaceTension [N/m, lbf/ft] returns the surface tension at the liquid-vapor interface of a saturated fluid. This function requires only one argument, in addition to the fluid name, and that is the temperature.

Example: sigma=surfacetension(Water, T=400)

Temperature [°C, K, °F, R] returns the temperature of the substance. The exact form of the function depends on the substance and argument(s) selected. Substances which are assumed to obey the ideal gas law, such as air, may require one or two arguments whereas pure real fluid substances, like STEAM, will always require two arguments.

Example: T1 = temperature(AIR, h=300)
T2 = temperature(AIR, s=1.75, P=100)

T_Crit [°C, K, °F, R] returns the critical temperature of the specified fluid. Critical property information is not available for ideal gas substances.

Example: Tc=T_Crit(R134a)

Volume [m^3/kg, m^3/kgmole, ft^3/lb, ft^3/lbmole] returns the specific volume of a specified substance. Two arguments are required for all pure substances; three are needed for moist air.

Example: v1 = **Volume**(AIR, T=300, P=100)
v2 = **Volume**(Steam, h=850, P=400)

v3 = **Volume**(AirH2O, T=70, R=0.5, P=14,7)

V_Crit [m^3/kg, m^3/kgmole, ft^3/lb, ft^3/lbmole] returns the critical specific volume of the specified fluid. Critical property information is not available for ideal gas substances.

Example: vc=V_Crit(R134a) "returns the critical volume of R134a"

Wetbulb [°C, K, °F, R] returns the wetbulb temperature for air-water gas mixtures. This function is applicable only to the substance AIRH2O. There are three arguments to this function, in addition to the substance name. The three arguments are temperature (or enthalpy), total pressure, and relative humidity (or humidity ratio or dewpoint).

Example: B1 = wetbulb(AIRH2O, T=70, P=14.7, w=0.01)
B2 = wetbulb(AIRH2O, h=25, P=14.7, w=0.01)
B3 = wetbulb (AIRH2O, h=25, P=14.7, D=30)

Viscosity [N-sec/m^2, lb_m/ft-hr] returns the dynamic viscosity of the specified substance. For ideal gas substances, the Viscosity function takes one parameter in addition to the fluid name and this parameter must be temperature. For AirH2O (moist air), the temperature, pressure, and humidity ratio (or relative humidity) must be supplied as arguments. For real fluids, the Viscosity function takes two parameters. Any two independent properties can be supplied, provided that the state does not consist of two phases.

Example: v1 = viscosity(AIR, T=300)
v2 = viscosity(R134a, T=40, x=1)
v3 = viscosity(STEAM_NBS, T=100, v=0.335)
v4 = viscosity(AIRH2O, T=80, P=14.7, R=0.5)

Using Lookup Files and Lookup Tables

A Lookup file is a two-dimensional set of data with a specified number of rows and columns. Lookup files provide a means to enter functional relationships with tabular data and to use these relationships in the solution of the equations. Lookup files can be stored in a disk file. In addition, one or more Lookup Tables can exist in the Lookup Table Window where they can be viewed and changed. The six menu commands which pertain to the Lookup Table Window appear at the bottom of the Options menu and are summarized here.

New Lookup Table creates a new empty Lookup Table with a specified number of rows and columns in the Lookup Table Window. A name must be provided for the Lookup Table. This is the name that must be used with commands that use the Lookup Table, including the Lookup, Lookup$, LookupRow, Interpolate, and Differentiate commands. The name will appear on a tab at the top of the Lookup Table Window.

The Open Lookup Table command will read the data in a Lookup file stored on disk into a Lookup Table in the Lookup Table Window. There are three Lookup File Formats, and all three can be opened with the command. In addition to being read into the Lookup Table, Lookup files can be accessed directly with Interpolate, Differentiate, Lookup, LookupCol, and LookupRow functions. Lookup Tables must have a name to be used with functions that operate on the Lookup Table data. The name that is given to the Lookup Table that is read in is the filename without the drive and filename extension. The name appears on a tab at the top of the Lookup Table Window. You can change the name by right-clicking on the tab.

Binary Lookup files (.LKT)

Binary Lookup files store all of the information that appears in a Lookup Table window in a binary file on disk, including the data, and the column name, units, and display format for each file type. A binary (.LKT) Lookup file is created using the Save Lookup Table command. Once created, the Lookup file can be opened into the Lookup Table window using the Open Lookup Table command. Binary files require less disk storage space and the can be opened and saved more quickly by EES. However, they cannot be created, edited or viewed by any application other than EES.

ASCII Lookup files (.TXT)

There are several variations for the ASCII Lookup file format. In the basic form, the first line of the file contains the number of rows and columns in the table. Each following line provides the data for one row with the value for each column separated by one or more spaces or a tab character. The basic form does not provide a means of specifying the names, units, or display format for the data. EES assigns the names "COLUMN1', 'COLUMN2', etc. and these column names should be used when the file is used with the

Interpolate or **Differentiate** commands. Automatic formatting is used to display the data if the file is read into the Lookup Table with the Open Lookup Table command. The following example shows the ASCII data needed for a Lookup file with five rows and three columns.

```
5  3
1     11     111
2     22     222
3     33     333
4     44     444
5     55     555
```

If a negative number is provided in the file for the number of rows, EES will determine the number of rows of data in the file. If the number of columns is a negative number, EES will expect to find the format specification (e.g., A3, F3 or E4) followed by one space and then the column heading and units for each column. The units are enclosed in square brackets. The following lines contain the data for each row, separated by one or more spaces or a tab. The example below would create a table with 2 rows and 3 columns. The columns would be formatted with E4, F0, and F3 format specifications and the column names will be ColA, ColB, and ColC.

```
2   -3
E4 ColA [Btu]
F0 ColB
F3 Col
1.23E-12      2       4.56
2.34E-11      4       7.89
```

In addition to being read into the Lookup Table, Lookup files in either the binary or ASCII formats can be accessed directly with **Interpolate**, **Differentiate**, **Lookup**, **LookupCol**, and **LookupRow** functions. These functions are documented below.

ASCII Lookup files (.CSV)
The .CSV file provides only data without any information concerning the column names or data format. The number of values in the first row of the file determines the number of columns in the table. The number of rows in the table is equal to the number of rows of data. Values on each row are separated with the list separator character. Each row ends with a linefeed - carriage return. The .CSV format is necessary when you wish to export the data to another application, such as a spreadsheet.

Save Lookup saves the foremost Lookup Table in the Lookup Table Window as a Lookup file on the disk. Lookup files can be accessed with the Lookup functions described below. The

Lookup file can be saved as a binary file with a .LKT filename extension or as an ASCII text file. The ASCII format allows the data to be exported to another application. Note that the Lookup Tables in the Lookup Table Window are also saved with other problem information when the Save command is issued. It is not necessary to save a Lookup Table separately unless it is to be used program.

Insert/Delete Lookup Rows will allow one or more rows to be added or removed from an the specified Lookup Table in the Lookup Table Window. Select the name of the Lookup Table for which the change is to be made from the drop-down list at the top of the dialog. Note that rows in a Lookup Table can be delete more simply by clicking in the row header (in the leftmost column) to select the row followed by pressing the Delete key or selecting the Delete command in the Edit menu.

Insert/Delete Lookup Cols will allow one or more columns to be added or removed from an existing Lookup Table in the Lookup Table Window.. Select the name of the Lookup Table for which the change is to be made from the drop-down list at the top of the dialog. Note that columns in a Lookup Table can be delete more simply by clicking in the column header to select the column followed by pressing the Delete key or selecting the Delete command in the Edit menu.

Delete Lookup Table will present a dialog that shows all Lookup Tables in the Lookup Table Window. Select the Lookup Tables that you wish to delete by clicking on their names in the list. Selected files will be deleted when the OK button is clicked.

Data in the Lookup Table can be accessed with the **Differentiate**, **Interpolate**, **Lookup**, **Lookup$**, **LookupRow**, **Lookup$Row**, and **LookupCol** functions. These functions may either operate on data in the Lookup Table window or in a Lookup file on disk. In the former case, the name of the Lookup Table that is accessed is provided as the first argument as a string constant (surrounded with single quotes) or string variable (identified by the $ character at the end of the variable name). In the latter case, the first argument of the function is the Lookup filename as it is stored on disk. The file name can be supplied as a string constant or as a string variable. Chapter 7 provides details on the use of string variables. The filename extension can either be .LKT (for binary lookup files) or .TXT or .CSV (for ASCII lookup files). If a filename extension is not provided, EES will assume that the file format to be binary (i.e., EES will automatically append the .LKT extension).

Differentiate(*TableName*, 'ColName1', 'ColName2', ColName2=Value**)** returns the derivative determined from two columns of tabular data based on cubic interpolation. These data can be in the Lookup table, a Lookup file, or the Parametric table. *TableName* is a string constant or string variable that provides that provides the name of the Lookup table in the

Lookup Window or the name of an existing Lookup file stored on disk. The string constant must be enclosed within single quote marks. Lookup table names appear on the tabs at the top of the Lookup Table Window. If the name of a disk file is supplied, it must be the name of an existing Lookup file having having a .LKT, .TXT. or .CSV filename extension. If the *TableName* parameter is not supplied, the **Differentiate** function will be applied to date in the foremost Lookup Table.

ColName1 and ColName2 are the column header names. The single quotes enclosing the column header names are optional. These columns names can also be supplied with string variables. The final parameter is of the form ColName2=Value where the text to the left of the equal sign can be either of the column header names (ColName1 or ColName2) specified with the two previous parameters. Value is a numerical value or expression. EES will return an estimate of the derivative d(ColName1)/d(ColName2) at a point fixed by the specified value of either ColName1 or ColName2. The **Differentiate** function can also be made to operate on data in the Parametric table if Filename is set to ‘Parametric’. In this case, the values in the Parametric table must already exist, such as those entered from the keyboard. Values which are to be calculated when the Solve Table command is issued cannot be used with the **Differentiate** command.

Examples: dXdY=Differentiate(‘Lookup 1’,'X', 'Y', Y=2.34)
{returns the derivative dX/dY at a value of Y=2.34 using data in Lookup Table l}
Y=Differentiate('C:myFile',T,X,T=100) {returns the derivative dT/dX at a value of T=100 using data from Lookup file myFile.LKT on drive C}

Interpolate('*TableName*, 'ColName1', 'ColName2', ColName2=Value**)** returns an interpolated or extrapolated value from tabular data in the Lookup table, a Lookup file, or the Parametric table using cubic interpolation. *TableName* is a string constant or string variable that provides that provides the name of the Lookup table in the Lookup Window or the name of an existing Lookup file stored on disk. The string constant must be enclosed within single quote marks. Lookup table names appear on the tabs at the top of the Lookup Table Window. If the name of a disk file is supplied, it must be the name of an existing Lookup file having with a .LKT, .TXT. or .CSV filename extension. If the *TableName* parameter is not supplied, the **Interpolate** function will be applied to date in the foremost Lookup Table.

ColName1 and ColName2 are the column header names. These single quotes enclosing the column header names are optional. The column header names can alternatively be entered as string variables. The final parameter is of the form ColName2=Value where the text to the left of the equal sign can be either of the column header names (ColName1 or ColName2) specified with the two previous parameters. Value is a numerical value or expression. EES will return the interpolated value from the data in column ColName1 corresponding to the specified value of ColName2. If the value of ColName1 is supplied, EES will return the interpolated value of ColName2. If Filename is 'Parametric', the **interpolate** command will be applied to the existing Parametric table. In this case, the values in the Parametric table must already exist, such as those entered from the keyboard. Values which are to be calculated when the Solve Table command is issued cannot be used with the **interpolate** command.

Examples: Z=interpolate(‘Lookup 1’, ‘Col1’, ‘Col2’, Col1=2.3)
{returns a value from the Lookup Table in table Lookup 1 from column Col2 of the Lookup table corresponding to a value in Col1 equal to 2.3 using cubic interpolation. Note that the quotes are optional.}
X= Interpolate(C:\myData.LKT,X,Y,Y=4.5) {returns a value from column X in the lookup table called myData.LKT on drive C: corresponding to a value in column Y equal to 4.5 using cubic interpolation.}

interpolate1('*Filename*', 'ColName1', 'ColName2', ColName2=Value**)** provides the same function as the **interpolate** command except that it uses linear interpolation.

interpolate2('*Filename*', 'ColName1', 'ColName2', ColName2=Value**)** provides the same function as the **interpolate** command except that it uses quadratic interpolation.

Lookup(*TableName*, Row, Column**)** returns the value in a Lookup Table or Lookup file at the specified row and column. *TableName* is a string constant or string variable that provides that provides the name of the Lookup table in the Lookup Window or the name of an existing Lookup file stored on disk. The string constant must be enclosed within single quote marks. Lookup table names appear on the tabs at the top of the Lookup Table Window. If the name of a disk file is supplied, it must be the name of an existing Lookup file having with a .LKT, .TXT. or .CSV filename extension. Note that the column (last argument) can be specified either by providing a numerical value (or expression) for the column number or by providing the column name as a string constant (enclosed in single quotes) or as a string variable. An older format in which the column name is preceded with the # symbol is still supported. The row and column arguments need not be integers. The value returned will be interpolated between rows and columns as needed. For example, Lookup(‘Lookup 1’,2.5, 3) will return a value which is midway between the values on the second and third rows in the third column. If the specified row or column is less than 1, the value in the first row or column will be returned. Similarly, if the row or column is greater than the number of rows and columns in the lookup table, the value in the last row or column will be returned. The **Lookup** function can be used with the **LookupCol** and **LookupRow** functions to provide interpolated values of user-supplied tabular information. However, the **Interpolate** commands are usually more convenient for this purpose.

Examples:

X=Lookup('Lookup 1',1,2) { Set X to the value in row 1, column 2 in the Lookup Table named 'Lookup 1'}

X=Lookup('Lookup 1',1,'X') { Set X to the value in row 1 of the column in the Lookup table which is named X. }

X=Lookup('C:\abc\ CopperK.LKT',R,'T') {Set X to the value in row R and the column which is named T in Lookup file C:\ abc\ CopperK.LKT}

Lookup$ operates just like the **Lookup** function except that it returns a string rather than a numerical value. As in the **Lookup** function the first argument is the name of the table in the Lookup Table window or a name of a Lookup file stored on disk.. The next argument is a numerical value or expression that provides the row in the table. This row value should be an integer. Interpolation between rows is not allowed, as it is in the **Lookup** function. The final parameter indicates the column. The column can be indicated by a numerical value or expression which provides the column number or by the name of the column provided in a string constant or string variable. Note. In order to accept string information, the format style of the column in the Lookup table must be set to STRING. To change the format style, click in the column header and make the change in the Format Table dialog window.

Example:

R$=Lookup$('Lookup 1',1,2) {Column 2 must be set to STRING format}

LookupCol(*'Filename'*, Row, Value**)** uses the data in the specified row of the Lookup Table or Lookup file to determine the column which corresponds to the value supplied as the second argument. The column value returned may not be an integer. Interpolation between columns will be provided as needed. The purpose of the **LookupCol** function is to provide a means of relating tabular information in different rows of the Lookup Table or Lookup file.

Examples:

C=LookupCol('Lookup 1',2, 100)

{Set C to the column number in row 2 of table Lookup 1 which has a value of 100}

C=LookupCol('C:\abc\ CopperK.LKT', R, X) {Set C to the column number in row R of \Lookup file C:\abc\ CopperK.LKT having the value X}

LookupRow(*Tablename***,**Column, Value**)** uses the data in the specified column of the Lookup Table or Lookup file to determine the row corresponding to the value supplied as the second argument. *TableName* is a string constant or string variable that provides that provides the name of the Lookup table in the Lookup Window or the name of an existing Lookup file stored on disk. The string constant must be enclosed within single quote marks.

Lookup table names appear on the tabs at the top of the Lookup Table Window. If the name of a disk file is supplied, it must be the name of an existing Lookup file having with a .LKT, .TXT. or .CSV filename extension. Note that the last argument which indicates the column in the table can be indicated either by supplying its numerical value or by providing the column name as a string constant (enclosed in single quotes) or as a string variable. An older format in which the column name is preceded by the # symbol is also accepted. The row value returned may not be an integer. Interpolation between rows will be provided as needed. The purpose of the **LookupRow** function is to provide a means of relating tabular information in different columns of the Lookup Table.

Examples:

R=LookupRow('Lookup 1', 2, 100)

{Set R to the row number in column 2 of the Lookup table which has a value of 100}

R=LookupRow('C:\abc\ CopperK.LKT', C, X) {Set R to the row number in column C Lookup file C:\abc\ CopperK.LKT which has the value X}

Lookup$Row operates just like the **LookupRow** function except that it its final argument is a string rather than a value. The function returns the row in the table in which this string exists. As in the **LookupRow**function the, **Lookup$Row** function can have two or three arguments. If three arguments are provided, the first is a string that provides the name of a Lookup table stored on disk. The next argument is the column in the table. The final argument is a string constant or string variable holding the string that will be searched in the table. Note. In order to accept string information, the format style of the column in the Lookup table must be set to STRING. To change the format style, click in the column header and make the change in the Format Table dialog window.

When a new Lookup Table is created, the columns are initially named **Column1**, **Column2**, etc. These default names and the table display format can be changed by clicking the right mouse button in the header cell and selecting Properties from the popup menu. See the Lookup Window section of Chapter 2.

Information can be copied to or from the Lookup Table via the Clipboard. In this way, data may be transferred between the Lookup Table and the Parametric Table or between other applications such as a spreadsheet program. To select a rectangular group of cells in the Table, click the left mouse in the upper left cell. Hold the Shift key down and then click in the lower right cell. Selected cells will be displayed in inverse video. Use the Select All command in the Edit menu to select all of the cells in the Lookup table. Next, use the Copy command in the Edit menu to copy a selected range of table cells *to* the Clipboard. Hold the Ctrl key depressed if you wish to copy the column header name and units, in addition to the selected data. Data may be copied *from* the Clipboard by clicking the upper-left cell into which the data are to be pasted, followed by the Paste command. The data in the Clipboard will be pasted into the Lookup Table, starting from the selected cell.

The $OpenLookup and $SaveLookup Directives

The $OPENLOOKUP directive (available in the Professional version) opens a file having the specified name and reads that file into the Lookup Table. The file can be have .CSV, .TXT or .LKT Lookup file format. The filename should include the filename extension. The filename may be a string constant (enclosed within single quotes) or a string variable that has been previously assigned to the filename, as in the following examples.

```
$OPENLOOKUP 'C:\EES32\myTable.lkt'
$OPENLOOKUP FILE$  {File$ has been previously set to a valid lookup file name}
$OPENLOOKUP ? {Bring up a dialog window to choose the lookup file}
```

When EES encounters this directive, it will first check to see if it has already opened a Lookup table with this filename. If the Lookup table has already been opened and the time/date information for the disk file has not changed, no action will be taken.

If a single question mark (?) is provided as a file name, EES will present a Windows open file dialog and use then substitute the selected file name for the ? after opening the file. If two question marks are provided, EES will open the selected file, but not make the substitution so that the open file dialog is presented during every execution.

The $SAVELOOKUP directive will save a specified Lookup table into a disk file after calculations are completed. The format is:

$SAVELOOKUP *LookupTableName*   'C:\EES32\myTable.lkt'

LookupTableName is the name of the Lookup Table (as seen on the tabs in the Lookup Table Window) that is to be saved. *LookupTableName* can be a string constant contained in single quotes or a string variable. The last parameter is the name of the file. This name must be a string constant or a string variable.

Lookup tables are automatically saved and opened when an EES file is saved and opened. It is not necessary to use the $OPENLOOKUP and $SAVELOOKUP directives for this purpose. The directives are useful for chaining EES programs, i.e., when the an EES program writes information into the Lookup Table using the Lookup command in a function or procedure so that a following EES program can use that information.

The Lookup command can be used to save calculated results in the Lookup Table. Several EES programs can be chained in sequential operation by saving the Lookup table as a Lookup file in one EES program and loading that Lookup file in the next EES program. Link buttons on the Diagram window facilitate the chaining process.

CHAPTER 5

EES Functions, Procedures and Modules

Most high-level programming languages allow the user to write subprograms. EES also offers this capability in a variety of ways. An EES subprogram is a function, procedure, or module written within EES. A function is a subprogram that accepts one or more inputs and returns a single result. A procedure can return one or more results. A module is similar to a procedure in that it can return one or more results. However, it differs from a procedure in that it employs equalities rather than assignment statements, as explained below. EES can access both internal subprograms that have been written within EES and external subprograms written in Pascal, C, C++, FORTRAN, or any compiled language. The development of external subprograms is described in Chapter 6. Both internal and external subprograms can be stored in the USERLIB\ subdirectory from which they are automatically loaded when EES is started.

EES subprograms offers a number of advantages. First, they make it easier to formulate the solution for a complicated system by breaking the problem up into a number of smaller parts. Programs which rely on subprograms are easier to understand. Second, subprograms can be saved in a library file and reused in other EES programs. Third, EES functions and procedures (but not modules) allow use of *if then else*, *repeat until* and *goto* statements. The statements appearing in functions and procedures differ from those in the main body of EES in that they are assignment statements, similar to those used in most high-level programming languages, rather than equality statements. They are executed in the order in which they appear. Modules employ equality statements just as those used in the main body of an EES program. EES reorders the equality statements as needed to efficiently solve the equations. The combination of both statement types offers a great deal of flexibility in the manner in which a problem can be formulated in EES.

There are several ways to access subprograms. The Merge command in the File menu can be used to import EES subprograms from one EES file into another. In addition, EES also allows subprograms to be saved in a library file. Library files are EES files containing one or more functions, procedures, and/or modules which have been saved with a .lib filename extension using the Save As command. Subprograms stored in library files that reside in the USERLIB\ subdirectory are automatically and transparently loaded when EES starts. Library files can also be loaded with the Load Library command in the File menu and with the $INCLUDE directive. Functions, procedures and modules in library files act just like EES internal functions. They can even provide help when requested. The steps necessary for creating library files is described at the end of this chapter.

EES Functions

EES provides the capability for the user to write functions directly in the Equations window using the EES equation processor. EES functions are similar to those in Pascal. The rules for these functions are as follows:

1. The user functions must appear at the top of the Equations window, before any modules or any equations in the main body of the EES program.
2. User functions begin with the keyword FUNCTION. The function name and arguments, enclosed in parentheses and separated by commas, follow on the same line.
3. The function is terminated by the keyword END.
4. The equations appearing in EES functions and procedures are fundamentally different from those appearing in the main body of EES. The equations in functions and procedures are more properly called assignment statements, similar to those used in FORTRAN and Pascal. An assignment statement sets the variable identified on the left of the statement to the numerical value on the right. X:=X+1 is a valid assignment statement but it obviously cannot be an equality, as assumed for all equations in the main body of EES. The := sign (rather than the = sign) is used to signify assignment. However, EES will accept an equal sign in assignment statements if the ☑ **Allow = in Functions/Procedures** control in the Display Options dialog window (Options menu) is selected.
5. EES normally processes the assignment statements in a function or procedure in the order they appear. However, *If Then Else*, *Repeat Until* and *goto* statements may be used in functions and procedures to alter the calculation order. The format of these logic control statements is described below.
6. Functions are called simply by using their name in an equation. The arguments must follow the name, enclosed in parentheses. The function must be called with the same number of arguments appearing in the FUNCTION statement.
7. Equations in user functions may call any of the built-in functions. In addition, they may call any previously-defined user function or procedure or any functions or procedures previously loaded as Library files. Recursive functions which call themselves are, however, not allowed. Functions may not call modules.
8. All variables used in the function body are local to the function except those variables defined in the scope of the $COMMON directive. The function returns the value to which its name is assigned.
9. Functions always operate in real mode regardless of the setting for complex algebra.

Functions can be used to implement an analytical relationship between two or more variables. For example, the specific availability of a flowing stream, often called ψ, is

$$\psi = (h - h_o) - T_o (s - s_o) + V^2/2 + g z$$

where

h and s are specific enthalpy and entropy, respectively
h_o and s_o are specific enthalpy and entropy at the ‘dead’ state condition, T_o and P_o
V is the velocity
g is gravitational acceleration
z is elevation, relative to a selected zero point

Once the temperature and pressure of the dead state are selected, h_o and s_o are constants. A user function for the availability of steam, with T_o=530 R and P_o=1 atm, could be implemented by placing the following statements at the top of the Equations window. A reference to psi(T1, P1, V1, Z1) from an equation would return the specific availability of steam in Btu/lb_m for the chosen ‘dead’ state.

```
FUNCTION psi(T, P, V, Z)
    To := 530       “R    dead state temperature”
    ho := 38.05    “Btu/lbm    specific enthalpy at dead state conditions”
    so := 0.0745  “Btu/lbm-R    specific entropy at dead state conditions”
    h := enthalpy(STEAM, T=T, P=P)
    s := entropy(STEAM, T=T, P=P)
    g = 32.17     “ft/s^2  gravitational acceleration
    psi := (h-ho)- To * (s – so) + (V^2 / 2 + g * Z) * Convert(ft^2/s^2, Btu/lbm)
END
```

Functions can also be used to change the name of any built-in function and/or to shorten the argument list. For example, the following function changes the name of **humrat**, the built-in function for humidity ratio, to **w**, eliminates the need to specify the substance AIRH2O as an argument, and sets the total pressure to 100 kPa in each case.

```
FUNCTION w(T,RH)
    w := humrat(AIRH2O, T=T, P=100, R=RH);
END
```

The two example functions both employ EES internal property functions and as a result, they depend on the EES unit settings to be properly set. Using the **UnitSystem** function (Chapter 4) and the IF THEN ELSE statements documented below, it is possible to write general functions that will operate correctly with any unit settings.

EES Procedures

EES procedures are very much like EES functions, except that they allow multiple outputs. The format of a Procedure is:

```
PROCEDURE test(A,B,C : X,Y)
...
...
X :=...
Y :=...
END
```

Procedures must be placed at the top of the Equations window, before any of the modules or equations in the main body of an EES program. The procedure name, TEST, in the example above, can be any valid EES variable name. The argument list consists of a list of inputs and a list of outputs separated by a colon. In the example above, A, B, and C are inputs and X and Y are outputs. Each procedure must have at least one input and one output. Each output variable must be defined by an equation with the output variables name on the left of the assignment sign. An END statement closes the procedure.

To use the procedure, place a CALL statement anywhere within your equations. The CALL statement appears as

```
...
CALL test(1,2,3 : X,Y)
...
```

The numbers of inputs and outputs in the CALL statement argument list must exactly match the PROCEDURE declaration statement. The arguments may be constants, string variables, numerical variables, or algebraic expressions. Additional arguments can be passed between the main body of an EES program and a procedure using the $COMMON directive. EES will evaluate the outputs using the input variables supplied in the argument list. Procedures may also call other functions and procedures, provided that they are defined previously. Procedures may not call modules.

The equations within a procedure differ from ordinary EES equations in modules or in the main body of an EES program. First, all variables except for the inputs and outputs are local to the procedure. Second, the equations are really assignment statements, rather than equalities, and to make this distinction clear, the assignment symbol (:=) is used in place of the equal sign. You may override this convention by enabling the ☑ **Allow = in Functions/Procedures** control in the Preferences dialog window (Options menu). Third, *if then else*, *repeat until* and *goto* statements may be used. The format of these flow control statements is described in the next section.

Implicit equations can not be directly solved in a procedure or function, as they are in modules and in the main equation body. Using the *If Then Else*, *Repeat Until* and *goto* statements, it is possible to program your own iterative loop. However, it is also possible to have EES solve implicit equations within a procedure. For example, consider the following two non-linear equations.

```
X^3 + Y^2 = 66
X/Y = 1.23456
```

To solve for X and Y in a procedure, subtract the right-hand side from the left hand side of each equation and set them to residuals, R1 and R2, respectively. Now use EES to solve for X and Y such that the residuals are 0. Here's a program which does this. However, it should be noted that implicit equations could be solved more directly and efficiently using a module, as described later in this chapter.

```
PROCEDURE Solve(X,Y:R1,R2)
    R1:=X^3+Y^2-66
    R2:=X/Y-1.23456
END

CALL Solve(X,Y:0,0)     {X = 3.834, Y = 3.106 when executed}
```

Procedures offer a number of advantages for the EES user. Commonly-used procedures may be saved separately and merged into the Equations window with the Merge command in the File menu. Alternatively, the procedure could be saved as a library file so that it is loaded automatically when EES is started. Procedures can be selectively loaded with the Load Library command in the Options menu or with the $INCLUDE directive. For example, the equations describing a turbine can be entered once and saved. Each time a turbine calculation is needed, the CALL Turbine statement can be used to determine the turbine work and outlet state variables.

EES supports both internal and externally-compiled procedures. Internal procedures are entered directly at the top of the Equations window, as described in this section. Compiled procedures are written in a high-level language such as C, Pascal, or FORTRAN and called from EES. The CALL statement for both types of procedures is identical. See Chapter 6 for a detailed description of writing and using compiled functions and procedures.

Single-Line If Then Else Statements

EES functions and procedures support several types of conditional statements. These conditional statements can *not* be used in modules or in the main body of an EES program. The most common conditional is the *If Then Else* statement. Both single-line and multiple-line formats are allowed for *If Then Else* statements. The single-line format has the following form.

If (Conditional Test) *Then* Statement 1 *Else* Statement 2

The conditional test yields a *true* or *false* result. The format is very similar to that used in Pascal. Recognized operators are =, <, >, <=, >=, and <> (for not equal). The parenthesis around the conditional test are optional. Note that string variables (see Chapter 7) can be used in the condition test. The *Then* keyword and Statement 1 are required. Statement 1 can be either an assignment or a *GoTo* statement. The *Else* keyword and Statement 2 are optional. In the single-line format, the entire *If Then Else* statement must be placed on one line with 255 or fewer characters. The following example function uses *If Then Else* statements to return the minimum of its three arguments.[9]

```
Function MIN3(x,y,z)    { returns smallest of the three values}
   If (x<y) Then m:=x Else m:=y
   If (m>z) Then m:=z
   MIN3:=m
End

Y = MIN3(5,4,6)      { Y will be set to 4 when this statement executes}
```

The AND and OR logical operators can also be used in the conditional test of an *If Then Else* statement. EES processes the logical operations from left to right unless parentheses are supplied to change the parsing order. Note that the parentheses around the (x>0) and (y<>3) are required in the following example to override the left to right logical processing and produce the desired logical effect.

```
If (x>y) or ((x<0) and (y<>3)) Then z:=x/y Else z:=x
```

[9] Note the the built-in MIN function accepts any number of arguments so this function would not be needed.

Multiple-Line If Then Else Statements

The multiple-line *If Then Else* statement allows a group of statements to be executed conditionally. This conditional statement can be used in functions and procedures, but not in modules or the main body of an EES program. The format is as follows:

```
If (Conditional Test) Then
    Statement
    Statement
    ...
Else
    Statement
    Statement
    ...
EndIf
```

The *If* keyword, the conditional test, and *Then* keyword must be on the same line. The parentheses around the conditional test are optional. The statements which are to be executed if the conditional test is true appear on following lines. These statements may include additional *If Then Else* statements so as to have nested conditionals. An *Else* (or *EndIf*) keyword terminates this first group of statements. The *Else* keyword should appear on a line by itself, followed by the statements which execute if the conditional test is false. The *EndIf* keyword, which terminates the multiple-line *If Then Else* statement, is required and it must appear on a line by itself. The format is illustrated in the following example. Indentation is used to make the logic flow more clear. However EES ignores the blank spaces. Also upper and lower case are treated equally.

```
Function IFTest(X, Y)
    If (X<Y) and (Y<>0) Then
     A:=X/Y
     B:=X*Y
      If (X<0) Then          { nested If statement}
       A:=-A;  B:=-B
     EndIf
    Else
     A:=X*Y
     B:=X/Y
    EndIf
    IFTest:=A+B
End
```

```
G=IFTest(-3,4) { G will be set to 12.75 when this statement executes}
```

GoTo Statements

EES will normally process the assignment statements in a function or procedure in the order they appear starting with the first statement. However, the flow control can be altered using *GoTo* statements. The format of a *GoTo* statement is simply

```
GoTo #
```

where # is a statement label number which must be an integer number between 1 and 30000. Statement labels precede an assignment statement separated with a colon (:). The *GoTo* statement must be used with *If Then Else* statements to be useful. The following function illustrates the use of *GoTo* and *If Then Else* statements in the calculation of the factorial of a value supplied as the argument.

```
Function FACTORIAL(N)
    F:=1
    i:=1
10: i:=i+1
    F:=F*i
    If (i<N) Then GoTo 10
    FACTORIAL:=F
End

Y= FACTORIAL(5)   { Y will be set to 120 when this statement executes}
```

Repeat Until Statements

Looping within functions and procedures can be implemented with *If Then Else* and *GoTo* statements described above, but it is generally more convenient and readable to use a Repeat Until construct. The *Repeat Until* statement has the following format. Note that *Repeat Until* statements can only be used in functions and procedures.

```
Repeat
    Statement
    Statement
    ...
Until (Conditional Test)
```

The conditional test yields a *true* or *false* result using one of the following operators: =, <, >, <=, >=, and <> (for not equal). The format is identical to that used in Pascal. Here is the same Factorial example presented in the previous section implemented with a Repeat Until construct.

```
Function Factorial(N)
    F:=1
    Repeat
     F:=F*N
     N:=N-1;
    Until (N=1)
    Factorial:=F
End
```

Y= **FACTORIAL**(5) { Y will be set to 120 when this statement executes}

Error Procedure

The Error procedure allows the user to halt calculations if a value supplied to a function or procedure is out of range. The format of the Error procedure is

Call Error('error message',X) or *Call* Error(X)

where 'error message' is an optional character string enclosed within single quotes and X is the value of the parameter which caused the error. If the error message string is not provided, EES will generate the following error message when it executes the ERROR procedure.

Calculations have been halted because a parameter is out of range. The value of the parameter is XXX.

The value of X supplied to the Error procedure replaces XXX. If an error string is provided, EES will display that string, inserting the value of X in place of the characters XXX. If a formatting option, such as F1 or E4 follows the XXX, as in the example below, the value of X will be accordingly formatted, otherwise a default format will be applied. The ERROR procedure will most likely be used with an IF - THEN - ELSE statement as in the following example.

```
Function abc(X,Y)
    if (x<=0) then CALL ERROR('X must be greater than 0.  A value of XXXE4 was supplied.', X)
    abc:=Y/X
end

g:=abc(-3,4)
```

When this function is called, the following message will be displayed and calculations will stop: *X must be greater than 0. A value of -3.000E0 was supplied.*

Modules

Modules can be considered to be stand-alone EES subprograms that can be called from the main EES program. The format of a Module is similar to that for an internal Procedure. The Module is supplied with inputs and it calculates outputs. The formal format of the Module statement uses a colon in the argument list. The number of arguments provided to the left of the colon is the number of degrees of freedom in the module, or stated another way, the number of values that must be supplied to have the number of equations equal to the number of unknowns within the module. An example of a Module statement is

```
MODULE Testme(A, B : X, Y)
```

In this case, EES understands that there are two inputs (A and B) and two outputs (X and Y). However, EES Modules employ equalities rather than assignment statements as used in Procedures. In most cases, it does not matter what variables are specified as inputs as long as the appropriate number are specified. As a consequence, the colon which separates inputs from outputs is irrelevant and it can be replaced with a comma (or semi-colon for the European numerical format). The following form for the Module statement is equivalent to the format shown above.

```
MODULE Testme(A, B, X, Y)
```

A module is accessed with a CALL statement. For example, the following statement would access the Testme module.

```
CALL Testme(77,1.5, X,Y)
```

Note that if a colon is used to separate the inputs and output in the MODULE statement, it must also be used in the CALL statement. Similarly if a colon is not used in the MODULE statement, it should not be used in the CALL statement.

When EES encounters a CALL statement, it transparently grafts the equations in the module to the equations in the main program. The steps necessary for this process are as follows. First, every variable in the module, including the inputs and outputs in the MODULE statement, is renamed with a unique qualifier that EES can recognize. Then EES adds one equation for each input and output which sets the value of the parameter in the calling program to the value of the value in the module. Finally, all of the equations in the module, with their renamed variables, are merged into the EES program at the point at which it is called. If the module is called a second time, the process is repeated, but with a different qualifier for the variable names in the module. The net effect is that a copy of all of the equations in the module are merged into the main EES program each time a CALL statement is encountered. EES currently allows up to 6000 equations (10,000 for the Professional version) so rather large problems can be developed. EES then uses its efficient blocking techniques to reorganize the equations for optimal solution. As a result of this reorganization, the equations in the module may not necessarily be called in sequence. In fact, this is rarely the case. You can view the equations in the order that EES has rearranged them in the Residuals window. Equations from a module are identified with the module name followed by a backslash and then the call index number. For example, the following equation in the Residuals window

Turbine\2: h2=h1+Q/m

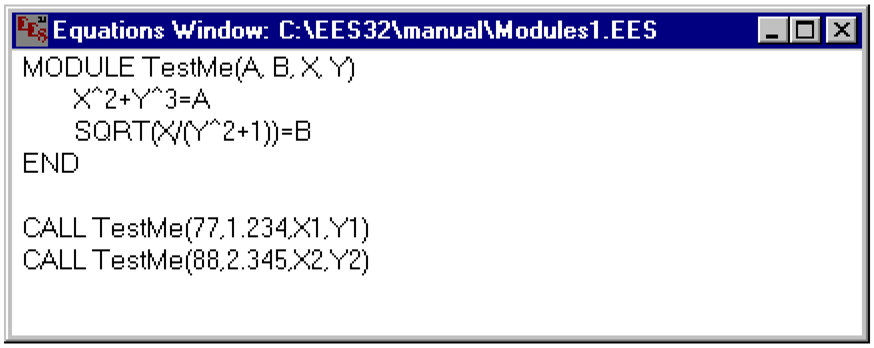
would indicate that the equation h2=h1+Q/m originated from the second call to the Turbine module.

The local values of variables in modules are normally not displayed in the Solution window. However, you can view these local solutions for modules which appear in your equations window by selecting the 'Show function/procedure/module variables' control in the Options tab of the Preferences dialog.

Variables values are normally passed to the module through the argument list. However, the $COMMON directive (Chapter 7) may be used to set the values of variables that are defined in the main program.

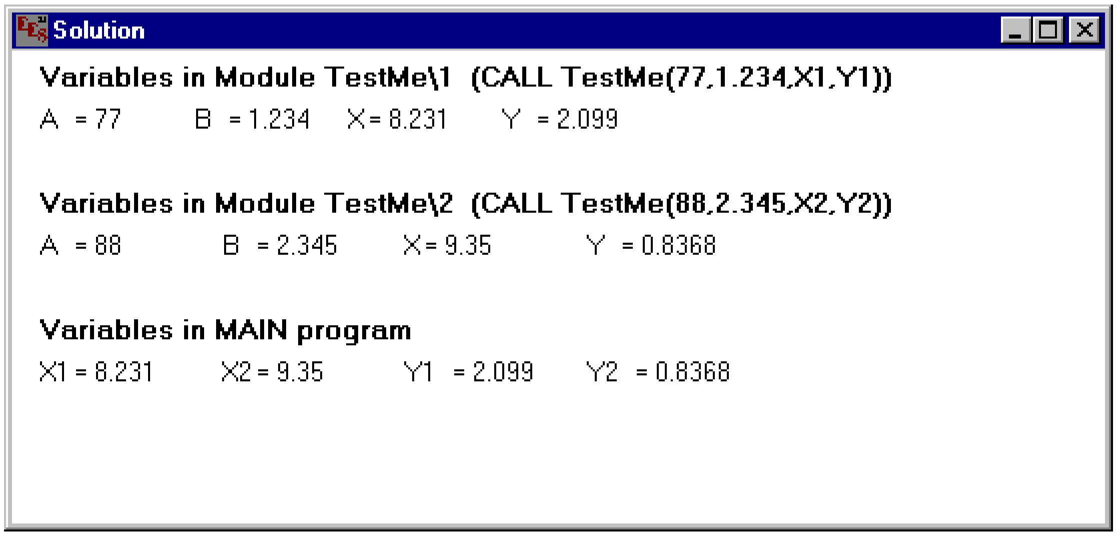
All variables defined within the module assume the same guess value, lower and upper bounds, and formatting information. The variable information is accessible with the Variable Info command. The local variables in a module are always real regardless of the Complex Numbers setting.

The Module is terminated with an END statement. A CALL statement is used to call the module, just as for the procedure. The important difference between a procedure and a module is that the module is composed of equality statements whereas the procedure is composed of assignment statements. Consequently, a module cannot support logic constructs such as IF THEN ELSE, but it can provide iterative solutions to implicit equations, when needed, just as in the main part of EES. Here is an example.



```
MODULE TestMe(A, B, X, Y)
    X^2+Y^3=A
    SQRT(X/(Y^2+1))=B
END

CALL TestMe(77,1.234,X1,Y1)
CALL TestMe(88,2.345,X2,Y2)
```

When **Solve** command is issued, EES will merge the equations in the Testme module twice into the main set of equations then solve the entire set. The Solution window will then appear as



```
Variables in Module TestMe\1  (CALL TestMe(77,1.234,X1,Y1))
A  = 77        B  = 1.234    X = 8.231     Y  = 2.099

Variables in Module TestMe\2  (CALL TestMe(88,2.345,X2,Y2))
A  = 88           B  = 2.345       X = 9.35          Y  = 0.8368

Variables in MAIN program
X1 = 8.231       X2 = 9.35          Y1  = 2.099      Y2  = 0.8368
```

Note that the local variables for each call to the Testme module will be shown only if the ‘Show function/procedure/module values’ control in the Options tab of the Preferences dialog is selected.

Modules may be stored as library files, just like internal functions and procedures. (Library files are described in the following section.) The library files can be automatically loaded if they are placed in the USERLIB subdirectory. Alternatively, a $INCLUDE directive can be used to transparently load a library file. Help can be included within the module using the same syntax as used in procedures or it can be supplied with a separate ASCII or Windows help file having the same filename as the library file with a .HLP filename extension.

Modules can significantly increase the capabilities of your EES programming.

Library Files

EES allows files containing one or more functions, procedures or modules (subprograms) to be saved as Library files. A Library file has a .LIB filename extension. When EES starts, it will automatically load all of the functions, procedures, and modules in the library files which reside in the USERLIB\ subdirectory. Library files can also be loaded manually with the Load Library command in the File menu and with the $INCLUDE directive. Library subprograms will not displayed in the Equations window. They are used just like EES built-in functions. To create a Library file, enter one or more functions, procedures, and/or modules into the Equations window. Compile the equations using Check, Solve or Solve Table. Then save the file with a .LIB filename extension using the Save As command.

Subprograms in library files can provide help information in the Function Info dialog window, just like the built-in functions. There are several ways to provide the help information. The simplest way is to include the help text as a comment within the EES library file. In this case, the first character after the opening comment brace must be a $, followed by the name of the function, procedure or module followed by a carriage return. The lines following, up to the closing comment brace, are the help text that will be displayed when the user selects the Info button in the Function Info dialog window as shown in the example below. Alternatively, a separate help file can be supplied with the same name as the library file and a filename extension of .hlp or .htm. The .hlp file can contain ASCII text or it can be a Windows Help file. EES will be able to recognize the file type from its contents. The .htm file should be designed to be read by a browser.

The Library file concept is among the most powerful features of EES because it allows the user to easily write customized subprograms for personal use or for use by others. The following example uses a library file to provide a fourth-order Runge-Kutta numerical integration function in EES. The Runge-Kutta algorithm is used to numerically solve a differential equation of the form:

$$\frac{\mathrm{d}Y}{\mathrm{d}X} = f(X,Y)$$

where *f(X,Y)* is any function involving the dependent variable *Y* and independent variable *X*. *Y* must have a known initial value, Y0, corresponding to the initial value of *X*.

The Runge-Kutta algorithm has been implemented as a general purpose library function called RK4. RK4 requires 4 parameters: the initial value of X (LowX), the final value of X (HighX), the step size (StepX), and the value of Y at X=LowX (Y0). The function returns the value of Y at X=HighX. The RK4 function calls another function, fRK4(X,Y), to provide the value of dY/dX for given X and Y values. A dummy fRK4 function is provided in the RK4.LIB file as a placeholder. In an actual application, the user overrides the dummy fRK4 function by entering another fRK4 function in the EES Equations window. The RK4 and fRK4 functions have been saved in a library file called RK4.LIB in the USERLIB\ sub-directory. EES will load these functions when it starts. If you were to open the RK4.LIB file in EES, you would see the following statements. Note how the functions provide help text as a comment with the $filename key.

```
FUNCTION fRK4(X,Y)
{$fRK4
fRK4 is a user-supplied function to evaluate dY/dX.  This
function is used with the RK4 function to solve differential
equations with the Runge-Kutta method.  Enter a fRK4(X,Y)
function in the Equations window to evaluate dY/dY for your
problem.  See the RK4 function for additional information.}
   fRK4:=(Y+X)^2
END

FUNCTION RK4(LowX,HighX,StepX,Y0)
{$RK4
RK4 is a general purpose function which solves a first-order
differential equation of the form dY/dX=fRK4(X,Y) using the
Runge-Kutta 4th order algorithm.  The RK4 function calls function
fRK4(X,Y) supplied by the user to evaluate dY/dX at specified values
of X and Y.  The user must supply the fRK4 function.

RK4 requires four input parameters.  LowX is the initial
value of independent variable X.  HighX is the final value
of independent variable X and StepX is the step size.  Y0 is
the value of Y when X is equal to LowX.}

   X := LowX
   Y := Y0;
   Tol := 0.1*StepX
10:
   IF (X>HighX-Tol) THEN GOTO 20
   k1 := fRK4(X,Y)*StepX
   k2 := StepX*fRK4(X+0.5*StepX,Y+0.5*k1)
   k3 := StepX*fRK4(X+0.5*StepX,Y+0.5*k2)
   k4 := StepX*fRK4(X+StepX,Y+k3)
   Y := Y+k1/6+(k2+k3)/3+k4/6
   X := X+StepX
   GOTO 10;
20:
   RK4:=Y
END
```

Suppose you wish to numerically solve the equation $\int_0^2 X^2 dx$ using the RK4 function.

You provide a function fRK4 to evaluate the integrand, which is X^2 in this case. Your function overrides the fRK4 function in the RK4 library file. Assuming RK4 was in the USERLIB\ sub-directory when EES was started, all that would be needed is:

```
FUNCTION fRK4(X,Y)
    fRK4:=X^2
END
V=RK4(0,2,0.1,0)
```

When you solve this problem, EES will display V=2.667 in the Solution window.

Note: The RK4 function solves an integral or differential equation using the Runge-Kutta algorithm. This algorithm works fine for differential equations it fails when there are combined algebraic and differential equations that must be solved. The built-in Integral function is more efficient for solving differential equations and integrals than the Runge-Kutta method described in the above example.

$COMMON Directive

The $COMMON directive provides a means for passing information from the main program to internal functions, procedures, and modules. Use of $COMMON provides an alternative to passing values as arguments. This directive is similar in concept to the COMMON statement in FORTRAN. It differs in that information flow is one-way. Variable values can be passed from the main program to the function or procedure. However, the function or procedure may not assign or alter these values.

The $COMMON directive must directly follow the FUNCTION, PROCEDURE, or MODULE declaration on a line by itself. Variables appearing in the $COMMON statement are separated with commas, as in the following example.

```
FUNCTION  TESTCOMMON(X)
$COMMON B,C,D {variables B,C, and D are from the main program}
     TESTCOMMON:=X+B+C+D
END
B=4;  C=5;  D=6
G=TESTCOMMON(3)
```

$COMMON should only be used with functions, procedures, and modules appearing in the Equations window. It should not be used with library functions.

$INCLUDE Directive

The $INCLUDE directive provides an automatic method for loading a library file or ASCII text file containing EES equations. The format is:

$INCLUDE FILENAME

FILENAME is the filename including the filename extension which can be one of .TXT, .LIB, .FDL, .DLF, or .DLP. The filename should also include the complete path name, e.g., C:\EES32\myDefn.TXT. However, if a path name is not provided, EES will look in the current directory. If EES is unable to find the file, it will provide the opportunity to browse so that you can locate it. The $INCLUDE statement must be on a line by itself, starting in column 1. It is best to place the $INCLUDE directives at the top of the Equations Window to ensure that the directive is processed before compilation of the equations is initiated.

.TXT Files

If the filename extension is .TXT, EES expects FILENAME.TXT to be an ASCII text file containing EES equations. Syntax errors can not be identified in the library file so care should be taken to ensure the equations are correct before saving the library file. EES will include these equations with others in the Equations window during compilation. However, the equations and the variables associated with these equations will be hidden. Nested use of the $INCLUDE directive is not supported so that the text file must not include any $INCLUDE statements.

Equations can also be entered from a file with the Merge command in the File menu. The difference between the Merge command and the $INCLUDE directive is that equations entered with the Merge command are placed directly into the Equations window as if they were typed and they remain visible. The text entered with the $INCLUDE directive will be hidden. Note that the speed of editing in the Equations window is reduced as the size of the text in the Equations window increases. Use of the $INCLUDE directive for very large problems can eliminate this problem on slower machines.

Library Files

If the filename extension is .LIB, .FDL, .DLF, or .DLP, EES will expect the file to be a library file of a type corresponding to the filename extension. EES internal functions, procedures. and modules are recognized with the .LIB extension, whereas external functions use a .DLF extension and external procedures use either the .FDL or .DLP extension. EES will automatically load the referenced library file if it is not already loaded. Note that library files can also be loaded automatically when EES is started by placing them in the USERLIB subdirectory or by applying the Load Library command. Help files having the same name as the library file and a filename extension of .HLP or .HTM will also be loaded automatically.

$EXPORT Directive

The $Export directive provides a simple way of writing selected variables to an ASCII file. This file can then be read by the Open Lookup Table command in EES or by another application, such as a spreadsheet program. If the filename extension is .CSV (comma-separated values), the data will be written as a CSV ASCII text file that is easily recognized by spreadsheet applications. In this format, each value is separated by a list separator character. If the filename extension is .TXT, EES will assume that the data should be written with header information in the EES Lookup file format that can be read directly by the Open Lookup Table command. The header information will include the name of the variable, its units, and its display format. The format of the $Export Directive is

```
$Export /A 'FileName', Var1, Var2, X[1..5], ...
```

The /A is optional. If present, it indicates that the values should be appended to the existing file, if it exits.

FileName can be either a string constant (within single quotes) or a string variable containing the name of the file that the values will be written to.

Var1, Var2, X[1..5] represent variables that are used in the main section (not in a Module) of your EES program. Note that array range notation is supported.

The $Export directive can be placed anywhere in the Equations Window. Multiple $Export directives are permitted, and they will be evaluated in the order that they occur in the Equations Window.

The Lookup command can be used to write (as well as read) values in the Lookup Table. Saving the Lookup table as a .CSV file also provides export capability, but the $Export directive is much more convenient.

Example:

```
T$='C:\temp\junk.csv'
A=5
B=A^2
C=A^3
$Export T$ A,B,C
```

After solving, file C:\temp\junk.csv will be written with the values

```
5.00000000E+00,2.50000000E+01,1.25000000E+02
```

$IMPORT Directive

The $Import directive provides a simple way of reading selected variables from an ASCII file. The data could be provided from a file written using the $Export directive or from another application. The combination of $EXPORT and $IMPORT directives provides a convenient way to transfer information from one EES program to another.

The format of the $IMPORT Directive is

$Import *FileName*, Var1, Var2, X[1..5], S$...

FileName can be either a string constant (within single quotes) or a string variable (ending with a $) holding the name of the file that the values will be read from.

Var1, Var2, X[1..5] represent variables that are used in your EES program. Note that array range notation is supported as are string variables.

The $IMPORT directive can be placed anywhere in the Equations Window. Multiple $IMPORT directives are permitted, and they will be evaluated in the order that they occur in the Equations Window.

Example: File text.csv contains the following data:

```
1.2,  3.4  5
'string 1'  'string 2'  6
```

```
$IMPORT 'text.csv'  X[1..3], A$, B$, C
```

After solving, variables X[1], X[2], X[3], A$, B$, and C will be set to the values in the file.

C H A P T E R 6

Compiled Functions and Procedures

EES has an extensive library of built-in functions, but it is not possible to anticipate the needs of all users. A remarkable feature of EES is that the user can add (and later remove) functions and procedures written in any compiled language, such as Pascal, C, C++ or FORTRAN. These compiled routines may have any number of arguments. Functions return a single value whereas procedures may return multiple values. The compiled routines are used in exactly the same manner as internal EES subprograms. This capability gives EES unlimited flexibility and it is among its most powerful features.

Compiled functions and procedures are written as dynamic link library (DLL) routines under the Windows operating system[10]. Compiled functions are identified with a .DLF. There are two formats for compiled procedures identified by .DLP and .FDL filename extensions. When EES is started, it examines the files in the EES USERLIB\ subdirectory. Any files having a .DLF, .DLP, or .FDL filename extension are assumed to be compiled functions or procedures and they are automatically loaded. External routines can also be loaded using the Load Library command in the File menu or with the $INCLUDE directive. The function name to be referenced in EES equations is the filename (without the extension).

Compiled functions and procedures can (optionally) be setup to work with the Function Info command (Options menu) so that they provide an example and detailed help when it is requested. The following sections of this chapter provide detailed information and examples of compiled functions and procedures.

EES Compiled Functions (.DLF files)

Compiled functions can be written in C, C++, Pascal, or any language which can produce a dynamic link library (DLL). The function statement header, however, must have a specific format. To avoid having to set a fixed upper limit on the number of inputs, the input information to a compiled function is implemented as a linked list. The linked list record or structure consists of an extended precision value and a pointer to the next input. The last input

[10] Note that there are both 16 and 32-bit DLLs in the Windows Operating System and they are not interchangeable. The 16-bit version of EES can only use 16-bit DLLs and the 32-bit version can only use 32-bit DLLs. The 32-bit version of EES can be used with Windows 95 or NT, but not with Window 3.1. The 16-bit version will run under any Windows operating system, but it executes more slowly than the 32-bit version. Instructions for preparing both DLL types are provided in this chapter. If you are not sure as to whether you are using the 16-bit or 32-bit version of EES, start the program and examine the startup screen. The 32-bit version of EES will display (32-bit) on the same line as the version number and in the main window title.

points to nil. Some compiled languages, such as FORTRAN 77 do not support pointers so .DLF compiled functions cannot be written in these languages The .FDL format described in this chapter should be used in this case.

The compiled function should check that the number of inputs supplied in the linked list is equal to the number the function expects. (The PWF function example in the next section shows how this checking can be done.) Although the values of the inputs may be changed in the function, these changes are local and will be disregarded by EES. Only the function result will be used by EES. A skeleton listing of a compiled function written in Borland's Delphi 1.0 (16-bit) or Delphi 3.0 (32-bit) is as follows:

```
library XTRNFUNC;
{$N+}

type
      ParamRecPtr = ^ParamRec;
      ParamRec = record  { defines structure of the linked list of inputs }
              Value: extended;
               next: ParamRecPtr;
      end;

function FuncName (var S:string; Mode:integer; Inputs:ParamRecPtr): extended; export; stdCall; [11]
     begin
              ...
              FuncName:=Value; { Funcname must be extended precision }
     end;

exports FuncName;

begin
end.
```

The major concern is the function header. To be recognized by EES, the function name, called FuncName in the above example, must be the same as the filename. The function statement has three arguments.

S is a standard 255-character Pascal string. The first character contains the actual length of the string. S can be used for both input and output. If the first parameter provided in the EES function is a string (within single quotes), EES will pass this string to the external routine. If an error is encountered, S should be set in the external routine to an appropriate error message. If the length of S is not zero, EES will terminate calculations and display S as an error message.

[11] The **stdCall** keyword is required for Delphi 3.0 (32-bit). It is not needed for Delphi 1.0 (16-bit)

Mode is an integer set by EES. If Mode=–1, then EES is requesting that the function return in S an example of the function call. If Mode>=0, then the function should simply return the function value. Currently, EES does not use the return value of Mode.

Inputs is a pointer to the head of a linked list of input values supplied by EES. Each input consists of a value (extended precision) and a pointer to the next input, as indicated by the ParamRec structure. The function may have one or more inputs. The next field of the last input will be a null pointer (nil). The function should count the inputs to be sure that the number supplied is as expected and issue an error message in S if this is not the case.

A skeleton listing of a compiled function written in Borland's C++ follows:

```
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#define EXAMPLE=-1;
// Use extern "C" to prevent C++ name mangling
extern "C"
{
 long double far pascal _stdcall _export FUNCNAME
      (char* S, int Mode, struct ParamRec *FirstInput);
}
int far pascal LibMain  //DLL entry point for initiation
  (HINSTANCE hInstance, WORD wDataSeg, WORD cbHeapSize, LPSTR lpstrCmdLine)
 {
      if (cbHeapSize) UnlockData(0);
      return TRUE;
 }
int far pascal WEP(int nParam)  //Windows exit procedure - not needed in Borland C++
 {
       return TRUE;
 }
struct ParamRec
{
      long double               value;
      struct ParamRec           *next;
};
long double far _export _stdcall pascal FUNCNAME(char* S, int Mode, struct ParamRec *FirstInput)
  {
      ….
      ….
      return (v);
}
```

Note that the pascal keyword must be provided to ensure that the calling parameters are in an order that will be understood by EES.

The PWF Compiled Function

EES does not have any internal economic functions. An economic function called the present worth factor (PWF)[12] has been added as a compiled function. PWF is the present worth of a series of N future payments which inflate at rate *i* per period accounting for the time value of money with a market discount rate per period of *d*. The equation for PWF is

$$PWF(N,i,d) = \sum_{j=1}^{N} \frac{(1+i)^{j-1}}{(1+d)^{j}} = \begin{cases} \frac{1}{d-i}\; 1-\left(\frac{1+i}{1+d}\right)^{N} & \text{if } i \neq d \\ \frac{N}{1+i} & \text{if } i = d \end{cases}$$

where

- N is the number of periods (e.g., years)
- i is the interest rate per period, expressed as a fraction
- d is the market discount rate per period, expressed as a fraction

A compiled function, called PWF, has been written to do this economic calculation. This function is stored in the PWF.DLF file on the EES disk. EES treats this compiled function just like any of its internal functions. Shown on the following pages is the complete listing for the PWF compiled function written in Borland's Delphi 3.0 (32-bit version).

Four other compiled functions are included with EES. These functions implement a generalized equation of state using Redlich-Kwong-Soave equation[13].

Compressibility(Tr, Pr, w**)** returns the compressibility of a gas, i.e., the ratio of the specific volume of the gas to the specific volume of an ideal gas at the same conditions. Tr is the reduced temperature, Pr is the reduced pressure, and w is the acentric factor. The third parameter is optional.

EnthalpyDep(Tr, Pr, w**)** and **EntropyDep(**Tr, Pr, w**)** return the dimensionless enthalpy and entropy departures, respectively. The dimensionless enthalpy departure is defined as (h[ideal]-h)/(R Tc). h[ideal]-h is the difference in enthalpy between an ideal gas and a real gas at the same temperature and pressure, R is the gas constant and Tc is the critical temperature. The dimensionless entropy departure is similarly defined as (s[ideal]-s)/(R).

FugCoef(Tr, Pr, w**)** returns the dimensionless fugacity coefficient (fugacity / pressure).

Do you need a function which EES doesn't include? Write your own. It's EESY!

[12] Duffie, J.A. and Beckman, W.A., *Solar Engineering of Thermal Processes,* 2nd edition, J. Wiley and Sons, 1992, Chapter 11.
[13] G. Soave, Chem. Eng. Science, Vol. 27, pp. 1197-1203, 1972

Listing of the PWF Compiled Function in Borland's Delphi 3.0

```
library PWFP;
uses
  SysUtils, Classes;
{$N+}

const doExample = -1;

type
  ParamRecPtr=^ParamRec;
  ParamRec=record
    Value:extended;
    next:ParamRecPtr;
  end;

 function CountValues (P: ParamRecPtr): integer;
  var  N: integer;
  begin
    N := 0;
    while (P <> nil) do begin
       N := N + 1;
       P := P^.next
    end;
    CountValues := N;
  end; {CountValues}

function PWF(var S:Shortstring; Mode:integer; Inputs:ParamRecPtr):extended; export; stdcall;
  var  P: ParamRecPtr; V: extended;

 function CountValues (P: ParamRecPtr): integer;
  var
   N: integer;
  begin
    N := 0;
    while (P <> nil) do begin
       N := N + 1;
       P := P^.next
     end;
     CountValues := N;
  end; {CountValues}

 function PWFCalc: extended;
  var
    Periods, NArgs: integer;
    interest, discount: extended;
  begin
    PWFCalc:=0; {in case of error exit}
    S := '';
    P := Inputs;
    Periods := round(P^.value);
    if (Periods < 1) then begin
       S := 'The number of periods for the PWF function must be >0.';
       exit;
    end;
    P := P^.next;
    interest := P^.value;
    if (interest >= 1) or (interest < 0) then begin
      S := 'The interest rate is a fraction and must be between 0 and 1.';
      exit;
    end;
    P := P^.next;
    discount := P^.value;
    if (discount >= 1) or (discount < 0) then begin
      S := 'The discount rate is a fraction and must be between 0 and 1.';
      exit;
    end;
    if (interest <> discount) then
      PWFCalc := 1 / (discount - interest) * (1 - exp(Periods * ln((1 + interest) / (1 + discount))))
    else
      PWFCalc := Periods / (1 + interest);
  end; {PWF}

 begin
   PWF:=1;
   if (Mode = doExample) then begin
     S := 'PWF(Periods,Interest,Discount)';
     exit;
   end;
   if (CountValues(Inputs)<>3) then
     S := 'Wrong number of arguments for PWF function.'
   else begin
     PWF:=PWFCalc;
   end;
 end; {PWF}

 exports  PWF;

begin
  {no initiation code needed}
end.
```

When this Pascal code is compiled with the Borland Delphi 3.0, a dynamically-linked library routine is created. The compiler automatically generates a .DLL filename extension for the compiled code. EES must distinguish compiled functions from compiled procedures. It does this by the filename extension. Compiled functions must have a .DLF filename extension. Rename the compiled file so that it has a .DLF extension.

Access the external PWF function by a statement of the following form in your EES program.

P = PWF(Periods,Interest,Discount)

EES Compiled Procedures (.FDL and .DLP files)

EES compiled procedures are very similar to EES compiled functions. In either case, the user supplies the function or procedure in compiled form as a Windows dynamically linked library routine. The major difference between functions and procedures is that procedures can return one or more values, whereas a function returns a single value. Procedures are useful, for example, for thermodynamic property evaluations where multiple properties (e.g., volume, enthalpy, entropy, etc.) are to be determined given one set of independent variables (e.g., temperature and pressure).

External procedures are written as 32-bit dynamic link libraries (DLL's). There are two formats for external procedures and they are identified to EES by their filename extension. The two formats differ in the manner in which EES exchanges information with the external routine. The .FDL format passes inputs and outputs in double precision floating point arrays. The .DLP format passes inputs and outputs as linked lists (as in the .DLF functions) so there is no limit to the number of inputs and outputs. EES identifies the format by the filename extension which must be .FDL or .DLP. External procedures using arrays to hold the inputs and outputs must have a .FDL filename extension. This is the usual case when providing DLLs written in FORTRAN. C, C++, and Pascal procedures can use either format.

Compiled procedures are accessed from EES with the CALL statement which has the following format,

CALL procname(‘text’, A, B : X, Y, Z)

where

procname is the name of the procedure

‘text’ is an (optional) text string that will be passed to the procedure. This text can either be a string constant enclosed in single quote marks or a string variable. (See Chapter 7).

A and B are inputs. There may be one or more inputs, separated by commas, appearing to the left of the colon. Inputs may be numerical constants, EES variable names, or algebraic expressions. String variables cannot be supplied as inputs.

X, Y, and Z are outputs determined by the procedure. There should be one or more outputs to the right of the colon, separated by commas. Outputs must be EES numerical variable names. String variables cannot be provided for outputs.

Note that the CALL statement used to access compiled functions is identical in format to the CALL statement used for internal EES Procedures.

The following two sections describe the .FDL and .DLP external procedure formats and provide a simple example which should serve as a model.

Compiled Procedures with the .FDL Format - a FORTRAN Example

The .FDL format is illustrated by the following FORTRAN subroutine fragments. The code may differs slightly depending on which compiler is being used.

32-bit .FDL library using the Digital Visual FORTRAN 6.0 compiler

```
SUBROUTINE MYPROC(S,MODE,NINPUTS,INPUTS,NOUTPUTS,OUTPUTS)
!DEC$ATTRIBUTES ALIAS:'MYPROC' :: MYPROC
!DEC$ATTRIBUTES DLLEXPORT :: MYPROC
INTEGER(4) MODE, NINPUTS, NOUTPUTS
REAL(8) INPUTS(50), OUTPUTS(50)
CHARACTER(255) S
…
OUTPUTS(1)=…
…
RETURN
END
```

S is a null-terminated C-style character string containing 255 characters. If the first parameter in the EES Call statement is a text string (within single quotes), EES will pass this string to the external program in S. When EES calls the subroutine with `MODE` =-1, it is asking for an example of the calling sequence of this procedure from EES to be placed in S so that it can be displayed in the Function Info Dialog window. S is also used to return user-supplied error messages if necessary. If an error is detected in the subroutine, `MODE` should be set to a value greater than 0 to signal EES to terminate calculations. If S is defined, it will be displayed in the EES error message. In normal operation, `MODE` =0 and S need not be defined.

`NINPUTS` and `NOUTPUTS` are the number of inputs and outputs provided by EES. The routine should check to see if these agree with the expected number of inputs and outputs and return an error condition (`MODE`>0) if this is not the case. `INPUTS` and `OUTPUTS` are arrays of double precision (REAL*8) values. There is no inherent limit on the number of elements of these arrays, as EES will allocate memory as necessary. Up to 1000 variables can be passed in the argument list of external functions and procedures using array element notation, i.e., X[1..1000]. EES will supply the values in the `INPUTS` array. Results calculated by the subroutine are placed in the `OUTPUTS`.

The external program must be compiled and linked as a dynamic link library (DLL) routine. The compiling procedure differs among different languages and compilers.

Creating a 32-bit DLL with Digital Visual Fortran 6.0 is most easily done within the Microsoft Developer Studio environment. A new project workspace is selected as a Dynamic Link Library. The FORTRAN sources file(s) are inserted into the workspace and compiled with the standard options. Note that the two !DEC$ATTRIBUTES directives should be included in the main program, as noted above. The output filename in the Link settings should be set to

MYPROC.FDL where MYPROC is the name that will be used in the EES Call statement. Alternatively, the default filename MYPROC.DLL should be changed to MYPROC.FDL after building the dynamic link library project.

The simple FORTRAN program listed below provides the product, dividend, sum, and difference of two input values. This program should provide a model for writing external EES procedures in FORTRAN.

Listing of the FORTRAN MDASF Program

```
       SUBROUTINE MDASF(S,MODE,NINPUTS,INPUTS,NOUTPUTS,OUTPUTS)
C.  The following two lines are specific to Microsoft Power Station 4.0
         !MS$ATTRIBUTES ALIAS:'MDASF' :: MDASF
         !MS$ATTRIBUTES DLLEXPORT :: MDASF
C.  Replace INTEGER(4) with INTEGER*2 for a 16 bit DLL in the following line
       INTEGER(4) MODE, NINPUTS, NOUTPUTS
       REAL(8) INPUTS(25), OUTPUTS(25)
       CHARACTER(255) S
C.
       IF (MODE.EQ.-1) GOTO 900
       IF (NINPUTS.NE.2) GOTO 100
       IF (NOUTPUTS.NE.4) GOTO 200
C.  DO CALCULATIONS
       X=INPUTS(1)
       Y=INPUTS(2)
       IF (ABS(Y).LE.1E-9) GOTO 300
       OUTPUTS(1)=X*Y
       OUTPUTS(2)=X/Y
       OUTPUTS(3)=X+Y
       OUTPUTS(4)=X-Y
       MODE=0
       S=''C
       RETURN
100    CONTINUE
C.  ERROR:  THE NUMBER OF INPUTS ISN'T WHAT THIS SUBROUTINE EXPECTS
C.  NOTE: SET MODE>0 IF AN ERROR IS DETECTED.  IF S IS EQUAL TO A
C.  NULL STRING, THEN EES WILL DISPLAY THE MODE NUMBER IN AN ERROR
C.  MESSAGE.  IF S IS DEFINED, EES WILL DISPLAY THE STRING IN THE
C.  ERROR MESSAGE.  THE C AT THE END OF THE STRING INDICATES C-STYLE
C.  S='MDASF REQUIRES 2 INPUTS'C
       MODE=1
       RETURN
200    CONTINUE
       S='MDASF EXPECTS TO PROVIDE 4 OUTPUTS'C
       MODE=2
       RETURN
300    CONTINUE
       S='DIVISION BY ZERO IN MDASF'C
       MODE=3
       RETURN
900    CONTINUE
C.  PROVIDE AN EXAMPLE OF THE CALLING FORMAT WHEN MODE=-1
       S='CALL MDASF(X,Y:A,B,C,D)'C
       RETURN
       END
```

Compiled Procedures with the .DLP Format - a Pascal Example

The .FDL format described in the previous section was illustrated with FORTRAN, but it can be implemented in any compiled language. The .DLP calling format described in this section uses linked lists for inputs and outputs, and thereby is not suitable for use with FORTRAN. There is essentially no difference in efficiency between the two formats. Both are provided for backward compatibility and complete flexibility.

Compiled procedures using the .DLP format are very similar to compiled functions (.DLF files) described previously. The only difference is that a procedure must have, in addition to a linked list of input values, a linked list of output values. The calling sequence for a compiled Pascal procedure with the .DLP format has the following form

```
procedure procname (var S: string; Mode: integer; Inputs, Outputs: ParamRecPtr);
```

S, Mode, and Inputs are identical to their counterparts for the EES compiled functions. Outputs is a linked list of extended values which provides the results of the calculations to EES in the order in which they appear in the CALL statement.

Shown on the following page is a complete listing of an EES compiled procedure, called MDAS (an acronym for MyDearAuntSally) which provides the product, dividend, sum, and difference of two input values. (This is the same program used in the .FDL example.) The code checks to make sure that the number of inputs and outputs provided in the CALL statement are what the routine expects before it does the calculations and sets S to an error message if this is not the case.

Example Compiled Procedure (.DLP) in Borland's Delphi

```
library MDASP;

uses
  SysUtils, Classes;
{$N+}
const Example = -1;
type
  ParamRecPtr=^ParamRec;
  ParamRec=record
    Value:extended;
    next:ParamRecPtr;
  end;

 function CountValues (P: ParamRecPtr): integer;
  var  N: integer;
  begin
    N := 0;
    while (P <> nil) do begin
      N := N + 1;
      P := P^.next
    end;
    CountValues := N;
  end; {CountValues}

procedure MDAS(var S:Shortstring; Mode:integer;
      Inputs,Outputs:ParamRecPtr); export; stdcall;

 procedure MyDearAuntSally;
  var
    P1, P2: extended;
    P: ParamRecPtr;
  begin
    P := Inputs;
    P1 := P^.Value;
     P := P^.next;
   P2 := P^.value;
    P := Outputs;
    P^.Value := P1 * P2;
    P := P^.next;
    P^.Value := P1 / P2;
    P := P^.next;
    P^.Value := P1 + P2;
    P := P^.next;
    P^.Value := P1 - P2;
  end; {doCall}

 begin {MDAS}
   if (Mode = -1) then
     S := 'CALL MDAS(In1,In2:Out1,Out2,Out3,Out4)'
   else begin
    if (CountValues(Inputs) <> 2) then begin
       S := 'Wrong number of inputs for MDAS.';
       exit;
    end;
    if (CountValues(Outputs) <> 4) then begin
       S := 'Wrong number of outputs for MDAS.';
       exit;
    end;
     MyDearAuntSally;
     S:='';
    end;
 end; {MDAS}

 exports
   MDAS;

begin
  {no initiation code needed}
end.
```

Multiple Files in a Single Dynamic Link Library (.DLL)

EES recognizes three different types of externally compiled files. The three types are:

DLF - dynamically-linked function
DLP - dynamically-linked procedure
FDL - dynamically-linked procedure with calling sequence accessible from FORTRAN

Originally, only one external routine could reside in a file. The filename extension (.DLF, .DLP, or .FDL) identified the type of externally compiled file and the name of the external routine had to be the same name as filename (without the extension).

However, it is also possible to place one or more external routines in a single file and this single file can contain all three types of external routines. The external file can have any name but it must have a .DLL filename extension. If this file is placed in the USERLIB subdirectory, EES will automatically load all the external routines in the file at startup.

EES must know the names of the external routines in the .DLL file and their type (DLF, DLP, or FDL) because the calling arguments differ for each type. The mechanism for telling EES the names and types of the external routines is to provide three short routines in the DLL file with names DLFNames, DLPNames, and FDLNames that do nothing but return the calling names of each routine type in the DLL file. DLFName, DLPNames, and FDLNames must be exported in the DLL. They have one argument which is a character string. The character string is filled with the names of the routines of each type that are contained in the DLL file. A comma separates each file name. A zero length string is used to indicate that there are no files of that type. A short example in DELPHI 5 code follows.

```
library MYEXTRNLS  {This DLL file contains two DLF functions and one DLP procedure}
uses
  SysUtils;
const  doExample = -1;
{*********************************************************************}
type
  ParamRecPtr = ^ParamRec;
  ParamRec = record
               Value : Extended;
               Next  : ParamRecPtr;
   end;
{*********************************************************************}
{There are 2 functions; names are separated with commas}


procedure DLFNames(Names : PChar); export; stdcall;
begin
  StrCopy(Names,'myFunc1, myFunc2');
end;
```

```
{**********************************************************************}
{There is one DLP procedure}
procedure DLPNames(Names : PChar); export; stdcall;
begin
  StrCopy(Names,'myDLP');
end;

{**********************************************************************}
{no FDL procedures so return a null string}
procedure FDLNames(Names : PChar); export; stdcall;

begin
  StrCopy(Names,'');
end;

{**********************************************************************}
function myFunc1 (var PString: ShortString; Mode: integer;
       Inputs: ParamRecPtr): extended; export; stdCall;
 begin
    {Code for myFunc1}
    ...
    ...
end; {myFunc1}

{**********************************************************************}
function myFunc2 (var PString: ShortString; Mode: integer;
       Inputs: ParamRecPtr): extended; export; stdCall;

 begin
    {Code for myFunc2}
    ...
    ...
end; {myFunc2}


procedure myDLP(var PString:Shortstring; Mode:integer;
        Inputs,Outputs:ParamRecPtr); export; stdCall;
 begin
    {Code for myDLP}
    ...
    ...
end; {myDLP}

{**********************************************************************}
 exports
   DLFNames,
   DLPNames,
   FDLNames,
   myFunc1,
   myFunc2,
   myDLP;

begin
end.
```

Help for Compiled Functions and Procedures

The Function Info dialog (Options menu) has an INFO button which, when used, provides help explaining the use of the selected function. When the user clicks the INFO button, EES will look for a file with the name of the compiled routine and a .HLP extension. This file can be an ASCII text file, a Windows .HLP file or an HTML file. This help file will be displayed if the file is found in the directory in which the external library file resides; otherwise a message will appear which states the help is not available for this item.

If an ASCII file is provided, it should be formatted so that each paragraph ends with a carriage return. Long lines which do not fit within the Help window will be broken and word-wrapped as needed. Blank lines and spaces can be used to make the text more clear.

Note that the Windows .HLP and HTML (.HTM) files allow figures and formatting options to be used so it is a better way of providing help. Either of these help file formats can be composed using any of the commercial Help generating programs.

CHAPTER 7

Advanced Features

The advanced features in EES allow the program to work with string, complex, and array variables and solve simultaneous algebraic and differential equations. The commands and functions which implement these features are described in this chapter and illustrated with examples.

String Variables

EES provides both numerical and string variables types. A string variable holds character string information. A string variable is identified to EES with a variable name that ends with the $ character, as in the BASIC language. The variable name must begin with a letter and consist of 30 or fewer characters, including the $ character.

String variables can be set to string constants. A string constant is a set of up to 255 characters enclosed within single quote marks, e.g.

A$='carbon dioxide'

String variables can be set equal to other string variables, e.g.,

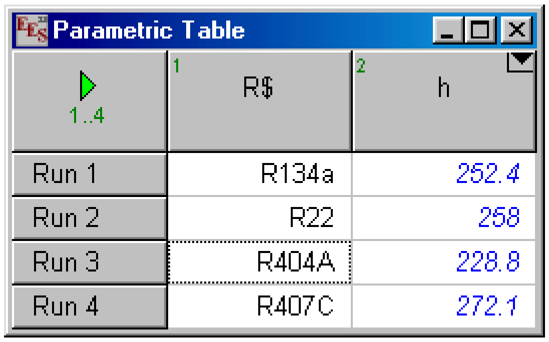
B$=A$

String variables may be passed as arguments to internal functions, procedures and modules or to external functions and procedures as described in Chapter 6.

In general, string variables can be used in EES equations anywhere in which character information is provided. For example, the name of a fluid provided to a thermophysical property function may be a string variable, e.g.,

h=enthalpy(R$,T=T,P=P)

String variables may be used with the Parametric table. In the example below, a Parametric table is used to tabulate the values of specific enthalpy for four refrigerants at 0°C, 100 kPa. Note that the single quote marks which are normally used when specifying a string constant should not be used when entering the strings in the Parametric table.



1..4	R$	h
Run 1	R134a	252.4
Run 2	R22	258
Run 3	R404A	228.8
Run 4	R407C	272.1

A string variable may be used to hold the name of a Lookup file or the name of a column for use with the **Interpolate** or **Lookup** commands, e.g.,

m=Interpolate(File$,Col1$,Col2$,Col1$=x)
k=Lookup(File$,Row,Col$)

String variables can be used to specify the units of other variables. For example:
U$='kJ/kg'
h=15 "[U$]"
The comment to the right of the 15 sets the units of variable h to U$. The units could also be set in the Variable Info dialog or by clicking on h in the Solution window. In any case, EES will recognize that U$ is a string variable and set the units of h to the string that is assigned to U$.

String variables may be supplied to the Equations Window using the Diagram Window, either with an edit box or with a pull-down list of alternatives. See the Diagram Window section in Chapter 2 for more information on this capability.

Complex Variables

EES will solve equations involving complex variables of the form a+b*i if the ‘Do Complex Algebra’ control in the Complex tab of the Preferences dialog is checked. The imaginary number operator may be set to either i or j in the Preferences dialog although i is used in the following discussion.

When set in complex mode, every (non-string) EES variable is represented internally as two variables corresponding to the real and imaginary components of a complex number. The real part is internally identified by appending _r to the variable name. The imaginary part has an appended _i. (You should not use _r or _i at the end of a variable name unless you are specifically referring to the real or imaginary component.) If, for example, you enter the equation

X=Y

EES will automatically create variables X_r, X_i, Y_r and Y_i corresponding to the real and imaginary components of the variables. You will normally not have to refer to the renamed variables, although they will appear with these names in the Variable Info and New Parametric Table dialogs, as well as in column headers of the Parametric Table. However, you can set the value of a real or imaginary part of a complex number by entering the real or imaginary variable name on the left of an equation in the Equations window. For example, the following equation will set the imaginary part of variable omega to 0.

omega_i=0

Any later attempt to set the imaginary part of omega will cause EES to display an error message. If, for example, you also enter the equation, omega = 3 in the Equations window, EES will present an error message when you attempt to solve because the omega=3 equation sets both the real and imaginary parts of omega and the imaginary part would have already been set. You could, of course, enter omega_r=3.

Complex numbers can be entered in either rectangular or polar form. In rectangular form, the complex number is entered using the imaginary number operator (i or j) with a multiplication symbol (*) separating the imaginary number operator from variables or constants. A complex constant can be entered in polar form by entering the magnitude (also called absolute value) of the number and the angle separated with the < symbol. The < character will be displayed in the Solution and Formatted Equations windows as ∠. The angle can be entered in either degrees or radians. If no designation is provided, the angle is assumed to be in the same units as indicated for trigonometric functions in the Unit System dialog. However, you can ensure that the angle you enter is in degrees or radians regardless of the unit system setting by appending deg or rad to the number (with no spaces). For example, the value of Y can be set to the same complex constant in any one of the following three ways.

Y=2 + 3 * i

Y=3.606 < 56.31deg

Y=3.606 <0.9828rad

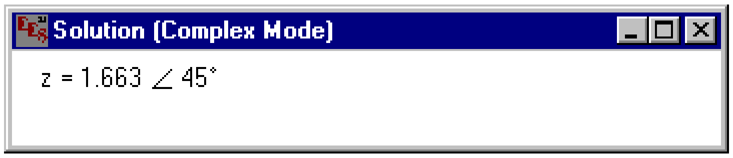
The use of deg or rad to indicate the units of the angle is strongly recommended for three reasons. First, the value of the constant you enter will not be changed by the unit system setting. Second, you can enter complex variables in polar form with the angle in degrees yet do all calculations with the unit setting in radians which is more efficient. Third, if you designate the angle to be in degrees, the degree sign will be shown with the angle in the Formatted Equations window. The output display of the number in the Solution Window can be set for degrees or radians (in polar notation) independent of how the number is entered. The display is changed by clicking the right mouse button on the value and selecting the display options from the Format Variable dialog.

Internally, EES creates two equations for each equation that is entered in the Equations window. One equation is used to equate the real parts of the variables whereas the second equation equates the imaginary parts. The actual set of equations used in complex mode is most clearly seen by viewing the Residuals window which displays the residual and blocking order for each equation. Equations used for the real and imaginary parts are identified with (r) and (i), respectively.

When configured in complex mode, some EES functions, such as **Min** and **Max**, are not accessible. However, most of the built-in functions (including the thermophysical property functions) have been modified to work with complex numbers. For example, the **sin**, **cos**, **ln**, **exp**, and **tanh** functions will accept and return appropriate complex numbers. User-written functions, procedures, and external routines can be used but they will accept and return real numbers only. (Modules are currently not supported in complex mode.) Only the real part of a complex variable will be placed in the argument list of internal or external functions, procedures, and modules.

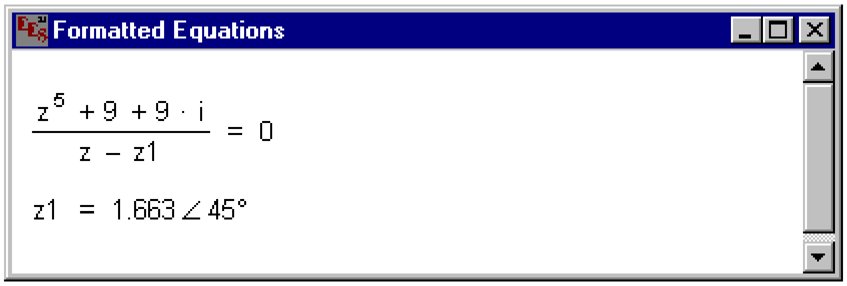
There are a few built-in functions that operate only in complex number mode. They are **Real**, **Imag**, **Cis**, **Magnitude**, **Angle**, **AngleDeg**, **AngleRad**, and **Conj**. These functions all take one (complex number) argument and set the real part of the complex variable to the selected value.

One limitation of the manner in which EES implements complex numbers is that EES can only return one solution, although two or more solutions may exist. There is, however, a simple way to coax EES into providing multiple solutions. Consider the problem of determining the five roots to the complex equation $z^5 + 9 + 9\,i = 0$. Entering this equation into the Equations window and solving with the default guess values will produce a solution of z=1.176+1.176 i. Setting the Solution window display to polar (degree) coordinates will produce the following solution.



z = 1.663 ∠ 45°

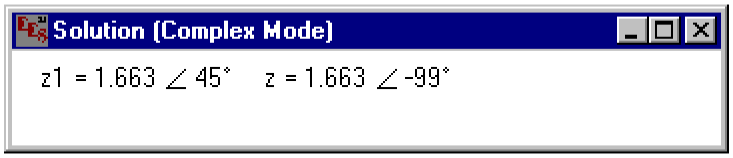
This is a correct solution to the equation, but there are four others. To find a different solution (without changing guess values), divide the equation by the difference between z and the value of this root. The Formatted Equations window will appear as shown.



$$\frac{z^5 + 9 + 9 \cdot i}{z - z1} = 0$$

$z1 = 1.663 \angle 45°$

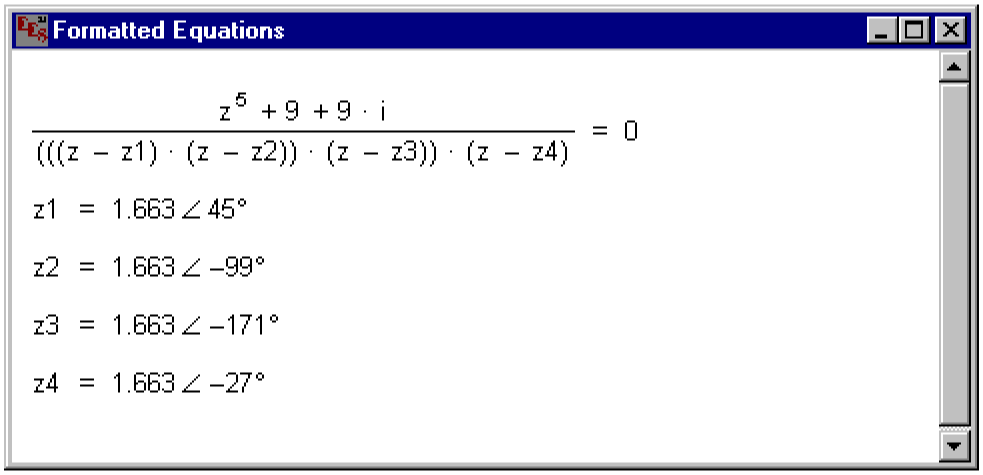
Now, a second solution will be found.



$z1 = 1.663 \angle 45°$ $z = 1.663 \angle -99°$

This process can be repeated to find the third, fourth, and fifth roots.

The Formatted Equations window and solution for this last step are shown below.



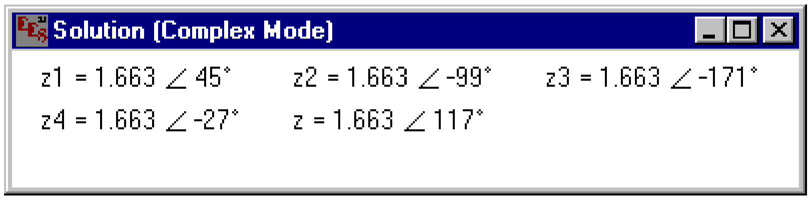
$$\frac{z^5 + 9 + 9 \cdot i}{(((z - z1) \cdot (z - z2)) \cdot (z - z3)) \cdot (z - z4)} = 0$$

$z1 = 1.663 \angle 45°$

$z2 = 1.663 \angle -99°$

$z3 = 1.663 \angle -171°$

$z4 = 1.663 \angle -27°$



$z1 = 1.663 \angle 45°$ $z2 = 1.663 \angle -99°$ $z3 = 1.663 \angle -171°$

$z4 = 1.663 \angle -27°$ $z = 1.663 \angle 117°$

Array Variables

EES recognizes an array variable by placing the array index within square brackets, e.g. X[5]. Multi-dimensional array variables may also be used with the indices separated by commas, e.g., Z[1,2,3]. The special requirements pertaining to array variables are:

1. An array index may be an integer number, an EES variable which has been previously set to a constant value, the **TableRun#** function, or an algebraic expression involving these quantities with operators +, –, *, and /. Index arithmetic is done left to right with no operator precedence. For example, X[2*3+1] is a valid array variable which EES will transform into X[7]. X[1+2*3] will be transformed into X[9]. The index variable for the DUPLICATE command or the **sum** or **product** functions can also be used in an expression for the array index as shown below.
2. Valid index values range between –32760 and +32760, including zero.
3. The right bracket must be the last character in the variable name.
4. The total length of the variable name, including the brackets and the integer value of the index, must not exceed 30 characters.

EES treats array variables in a very different manner than FORTRAN or Pascal. In EES, each array variable, such as X[99], is a unique variable name. As such, X[99] appears to EES just like any other variable such as ZZZ. The guess value and bounds (along with other information) may be specified for X[99] with the Variable Info command, just as for any other variable. It is legal (but not good practice) to have EES variables names of X, X[1], X[2,3] all within the same equation set. The fact that X[99] appears in the Equations window does not cause EES to reserve memory for 99 elements. Memory is allocated only for the variables which appear in the equations.

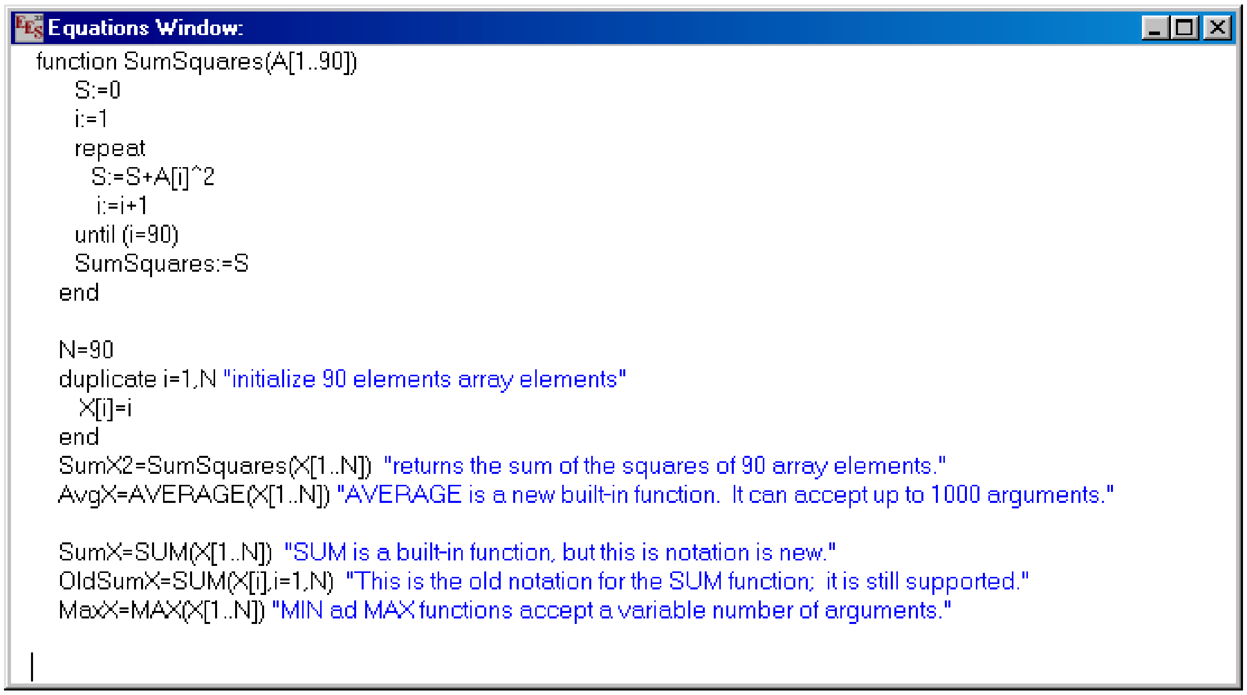
Array variables can be useful in several ways. They provide a means of grouping variables of similar type. For example, the temperatures at each state in a system can be written as T[1], T[2], etc. Array variables can be plotted. For example, the temperature and entropy of each state in a thermodynamic cycle can be overlaid on a T-s property diagram. See Property Plot in the Plot Menu section of Chapter 3 for details. Finally, array variables can be used with the DUPLICATE command and the **sum** and **product** functions to provide matrix capability and thereby significantly reduce the amount of typing needed to specify some problems.

Array Range Notation

Array range notation is a shorthand notation to facilitate passing of array variables to internal and external Functions and Procedures. A range of array variables can be indicated by separating the first array index value from the last index value by two decimal points. For example, X[1..5] can be used in place of X[1], X[2], X[3], X[4], X[5] as the argument list to a function. This shorthand notation is supported for two dimensional array variables as well, e.g., Z[2,1..3]. EES variables may be used in the index range (e.g., X[N..M]) provided that their values have been previously set in assignment statements. The notation can be used in the arguments of function calls and CALL statements, in Function and Procedure statements and in $Common directives.

Array range notation can be convenient when array variables are being used as arguments to internal and external functions. However, the major purpose of this notation is to allow long argument lists to be passed. A long argument list is not possible in any other way since EES statements must be 255 or fewer characters. A maximum of 1000 arguments may be passed to a function or procedure using array range notation.

The following example illustrates array range notation. this capability.



```
function SumSquares(A[1..90])
    S:=0
    i:=1
    repeat
      S:=S+A[i]^2
      i:=i+1
    until (i=90)
    SumSquares:=S
  end

  N=90
  duplicate i=1,N "initialize 90 elements array elements"
    X[i]=i
  end
  SumX2=SumSquares(X[1..N])  "returns the sum of the squares of 90 array elements."
  AvgX=AVERAGE(X[1..N]) "AVERAGE is a new built-in function.  It can accept up to 1000 arguments."

  SumX=SUM(X[1..N])  "SUM is a built-in function, but this is notation is new."
  OldSumX=SUM(X[i],i=1,N)  "This is the old notation for the SUM function;  it is still supported."
  MaxX=MAX(X[1..N]) "MIN ad MAX functions accept a variable number of arguments."
```

The DUPLICATE Command

The DUPLICATE command provides a shorthand way of entering equations into EES. The equations which are to be duplicated are enclosed between the DUPLICATE and END command words. DUPLICATE is useful only when used with array variables. For example, the following statements:

```
N=5
X[1]=1
DUPLICATE j=2,N
    X[j]=X[j–1]+j
END
```

are equivalent to:

```
X[1]=1
X[2]=X[1]+2
X[3]=X[2]+3
X[4]=X[3]+4
X[5]=X[4]+5
```

Note that, within the scope of the DUPLICATE command, the DUPLICATE index variable (j in the example above) can be used in an algebraic expression for the array index. The DUPLICATE index is not an EES variable, but rather just a temporary placeholder for the integers applied in the DUPLICATE command.

The special format requirements pertaining to the DUPLICATE command are as follows:

1. The DUPLICATE command must be on its own line in the Equations window or separated from other equations with a semicolon.
2. The lower and upper limits specified for the index variable in the DUPLICATE command must be integers, EES variables previously assigned to constant values, or the **TableRun#** function.
3. DUPLICATE commands may be nested within one another as deep as desired. However, each DUPLICATE command must use a different index variable name and each must be terminated with an END command. The lower or upper limit of an internal DUPLICATE may be the index value of an external DUPLICATE, e.g.,

   ```
   DUPLICATE i=1,5;  DUPLICATE j=i,6;  X[i,j] = i*j;  END;  END
   ```

4. The END command terminates the last opened DUPLICATE command.

Matrix Capabilities

Many engineering problems can be formulated into a linear system of algebraic equations of the form

$$[\mathbf{A}]\ [X] = [B]$$

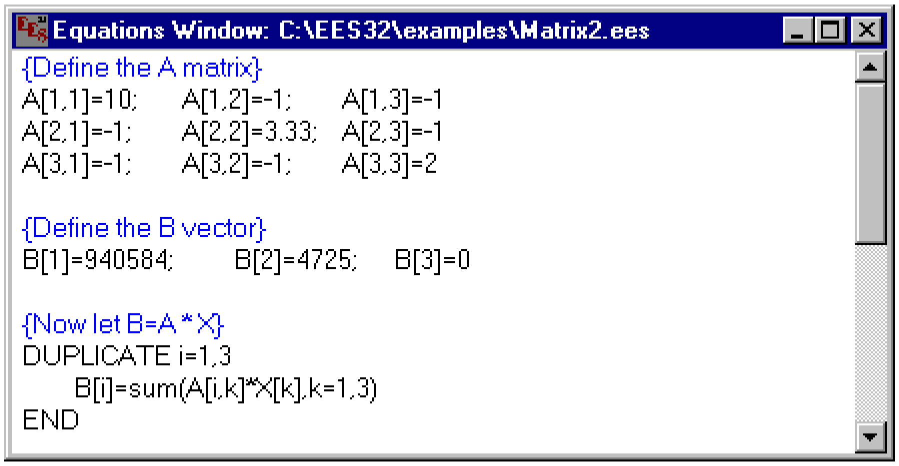
where [**A**] is a square matrix of coefficients, and [X] and [B] are vectors. Ordinarily, the matrix equation is solved to determine the elements in the vector [X] for known [**A**] and [B]. In this case,

$$[X] = [\mathbf{A}]^{-1}\ [B]$$

EES can directly solve the equations represented by [**A**] [X] = [B] by entering each equation directly into the Equations window in any format or order. However, a more elegant and convenient method for solving these equations in EES is to make use of the matrix capability. EES can solve matrix equations, formulated with array variables, by using the DUPLICATE command and the **sum** function. For example, consider the following radiation heat transfer problem in which [A] and [B] are given below, and the radiosity vector, [X] is to be determined.[14]

$$[A] = \begin{bmatrix} 10 & -1 & -1 \\ -1 & 3.33 & -1 \\ -1 & -1 & 2 \end{bmatrix} \qquad [B] = \begin{bmatrix} 940584 \\ 4725 \\ 0 \end{bmatrix}$$

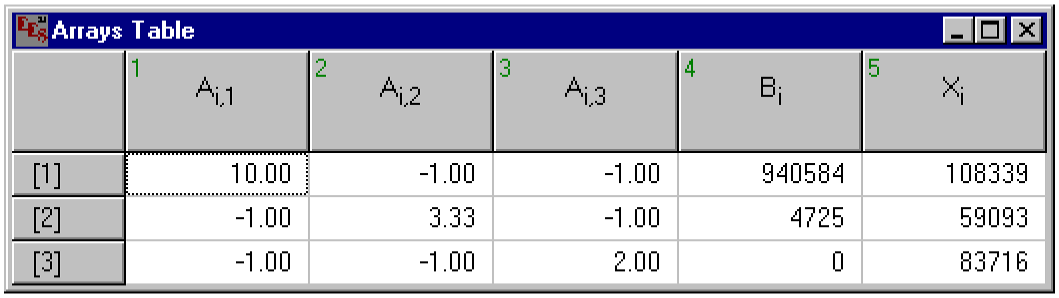
The equations required in EES to solve this problem are as follows:



```
{Define the A matrix}
A[1,1]=10;    A[1,2]=-1;    A[1,3]=-1
A[2,1]=-1;    A[2,2]=3.33;  A[2,3]=-1
A[3,1]=-1;    A[3,2]=-1;    A[3,3]=2

{Define the B vector}
B[1]=940584;    B[2]=4725;   B[3]=0

{Now let B=A*X}
DUPLICATE i=1,3
     B[i]=sum(A[i,k]*X[k],k=1,3)
END
```

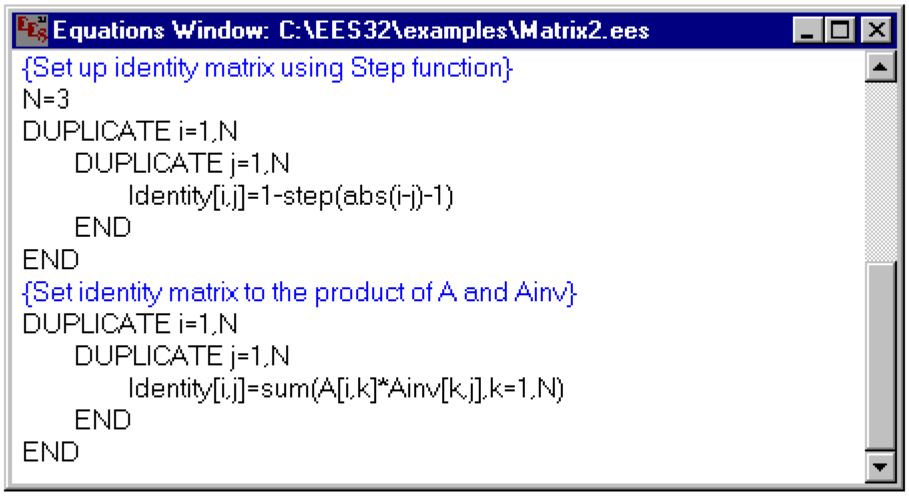
The calculated elements in the X array will appear in the Arrays window.

[14] Incropera, F.P. and DeWitt, D.P., *Fundamentals of Heat and Mass Transfer*, 2nd edition, John Wiley and Sons, 1985, Chapter 13


Arrays Table

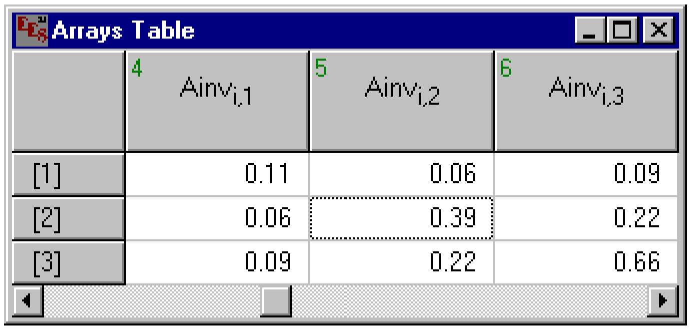
	1 $A_{i,1}$	2 $A_{i,2}$	3 $A_{i,3}$	4 B_i	5 X_i
[1]	10.00	-1.00	-1.00	940584	108339
[2]	-1.00	3.33	-1.00	4725	59093
[3]	-1.00	-1.00	2.00	0	83716

Note that it was not necessary to determine the inverse of [**A**] to obtain the solution. EES calculates the inverse matrix internally, as needed to solve these and any other simultaneous equations. However, the inverse matrix [**A**]$^{-1}$ can be determined by setting the matrix product [**A**] [**A**]$^{-1}$ equal to the identity matrix in the following manner.


Equations Window: C:\EES32\examples\Matrix2.ees

```
{Set up identity matrix using Step function}
N=3
DUPLICATE i=1,N
     DUPLICATE j=1,N
          Identity[i,j]=1-step(abs(i-j)-1)
     END
END
{Set identity matrix to the product of A and Ainv}
DUPLICATE i=1,N
     DUPLICATE j=1,N
          Identity[i,j]=sum(A[i,k]*Ainv[k,j],k=1,N)
     END
END
```

The inverse matrix Ainv will appear in columns of the Arrays window.


Arrays Table

	4 $Ainv_{i,1}$	5 $Ainv_{i,2}$	6 $Ainv_{i,3}$
[1]	0.11	0.06	0.09
[2]	0.06	0.39	0.22
[3]	0.09	0.22	0.66

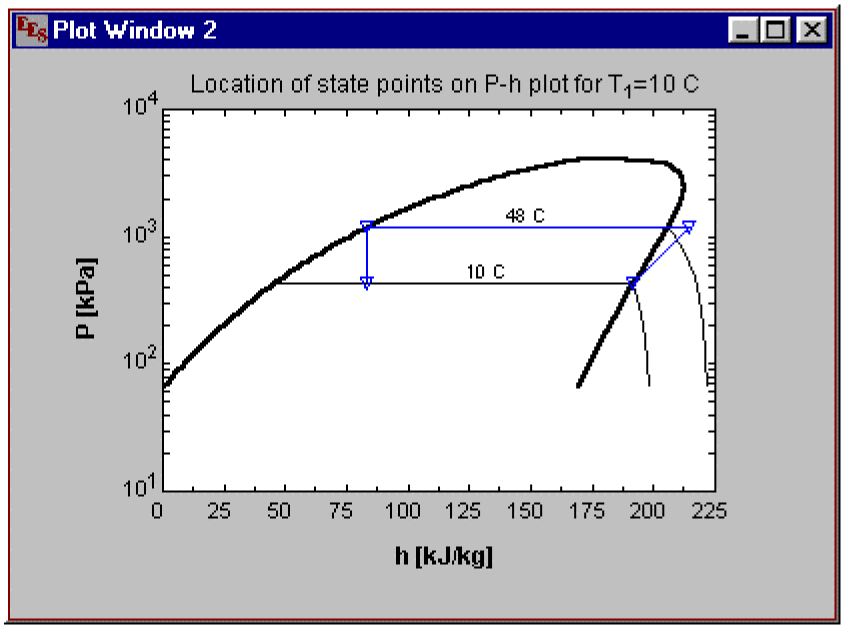
The two examples above provide a general procedure for determining the product of a matrix and a vector or the product of two matrices. Using the DUPLICATE command along with array variables in EES is no more efficient than the alternative of entering each equation separately with non-array variable names; however, the matrix capabilities in EES can significantly reduce the amount of typing required to enter the problem and, more importantly, make the equations easier to follow.

Using the Property Plot

The Property Plot menu item in the Plot menu generates T-s, T-v, P-v, or P-h diagrams for any of the fluids in the EES data base. A psychrometric chart is generated if substance AirH2O is selected. The property plot is placed in one of the EES plot windows.

Additional property data or thermodynamic state point information can be superimposed on the property plots using the Overlay Plot command in the Plot menu. Overlays can be plotted if array variables are used for the thermodynamic variables. Another benefit of using array variables is that the state property data then appear in the Arrays Table in a convenient tabular form.

The P-h plot below shows the state points for simple refrigeration cycle operating between an evaporator temperature of 10°C and a condensing temperature of 48°C with a compressor isentropic efficiency of 0.70. The plot was prepared by first producing a P-h plot for R12 with isotherms at 10°C and 48°C with the Property Plot command and then overlaying the P[i] and h[i] arrays (plotting from the Arrays table) for the four state points in a refrigeration cycle analysis. The equations can be found in file REFRIG.EES in the Examples subdirectory.



Integration and Differential Equations

The INTEGRAL function is used to evaluate an integral and in the solution of differential equations. The format of the **Integral** function is:

$$\int_{t1}^{t2} f\,dt = \textbf{Integral}(f, t)$$

There are two basic forms of the integral function which differ in their reliance on the Parametric table.

Table-based Integral function

The table-based Integral function uses the Parametric table to provide the limits and step size of the integration variable. The format of this function is **Integral**(f, t). This form of the Integral function can be used only in conjunction with the Parametric Table. The integration variable, t, must be a legal variable name which has values defined in one of the columns of the Parametric table. The limits on the integration variable, t_1 and t_2 are the first and last values specified for variable t. The integrand, f, can be a variable or any implicit or explicit algebraic expression involving variables, values, and the integration variable, t.

Equation-based Integral function

The Equation-based integral function serves the same purpose as the table-based integral function but it does not require the use of the Parametric table. The format for the equation-based integral function is

F = INTEGRAL(f, t, t1, t2, tStep)
or

F = INTEGRAL(f, t, t1, t2) {automatic step size}

t1 and t2 are the lower and upper limits of the integration variable. These limits may be specified with a constant or an EES expression. However, the limits cannot be a function of the integration variable t or any other variable which changes during the course of the integration.

tStep is the increment EES will use for the integration variable while numerically evaluating the integral between the specified limits. tStep cannot vary during the course of the integration. Note that specification of tStep is optional. If tStep is not provided, EES will select the stepsize using an automatic stepsize adjustment algorithm.

EES uses a second-order predictor-corrector algorithm for evaluating the integral. This algorithm is designed for solving combined algebraic and differential equations which result when the integrand is a complex function of other variables. The algorithm is especially suited to stiff equations.

EES uses the **Integral** function to solve initial value differential equations. Any first-order differential equation can be transformed into an appropriate form by integrating both sides. For example, the differential equation, dy/dx = f(x,y) can be equivalently written as

$$y = y_0 + \int f(x,y)\,dx$$

where y_0 is the initial value of y. This equation can be solved using either the table-based or equation-based forms of the **Integral** function. The table-based form would be entered into the EES Equations Window as

y = y0 + INTEGRAL(fxy, x)

where fxy is an EES variable or expression. To solve the equation it is necessary to create a Parametric table containing a column for variable x. Values of x are entered into the Parametric table with the value in first row corresponding to the lower limit of x and the value in the last row corresponding to the upper limit. The step size is determined by the difference between the value of x in successive rows and need not be a fixed value. The integral is evaluated when the **Solve Table** command is applied.

The equation-based form would appear in the EES Equations window as

y = y_0 + INTEGRAL(fxy, x, low, high)

y_0 and fxy are as defined above. Low and high are the lower and upper limits for x. A Parametric table is not required for use with this form of the **Integral** command. Since the stepsize is not specified, EES will use automatic stepsize selection.

<u>Solving First-Order Initial Value Differential Equations</u>
Initial-value differential equations can be solved in a number of ways with EES. Chapter 5 describes a Library function included with EES in the USERLIB subdirectory which implements a 4th order Runge-Kutta algorithm. This method can only be used if the derivative can be expressed explicitly as a function of the dependent and independent variables. This section demonstrates two ways of solving simultaneous algebraic and differential equations using the **Integral** function or the **TableValue** function in conjunction with the Parametric Table.

Method 1: Solving Differential Equations with the Table-Based Integral Function
Consider the problem of determining the time–temperature history of a sphere of radius r=5 mm, initially at a uniform temperature of 400°C. The sphere is exposed to 20°C air with a convection coefficient of h=10 W/m^2-K. The thermophysical properties of the sphere material are:

ρ = density = 3000 kg/m^3
k = thermal conductivity = 20 W/m-K
c = specific heat = 1000 J/kg-K

Calculation of the Biot number will indicate that the sphere can be treated as a lumped system, and therefore it may be assumed to be at a uniform temperature at any instant of time.[15] The relation between the sphere temperature and time is given by an energy balance on the sphere, which results in the following differential equation

$$-h\,A\,(T-T_\infty)=\rho\,c\,V\,\frac{dT}{dt}$$

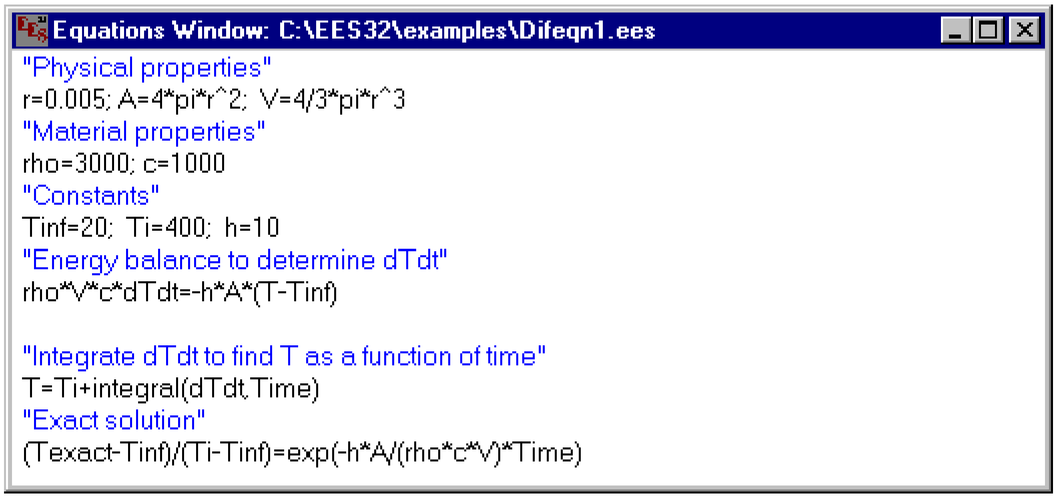
where

h	is the convective heat transfer coefficient
T	is the uniform temperature of the sphere at any time
T_∞	is the temperature of the air stream = 20°C
A	is the surface area of the sphere = 4 π r^2
V	is the volume of the sphere = 4/3 π r^3
t	is time

This differential equation has the following analytic solution which can be used to check the accuracy of the numerical solution provided by EES.

$$\frac{T-T_\infty}{T_i-T_\infty}=\exp\left(\frac{-h\,A}{\rho\,c\,V}\,t\right)$$

To solve the differential equation numerically in EES, enter the following equations.



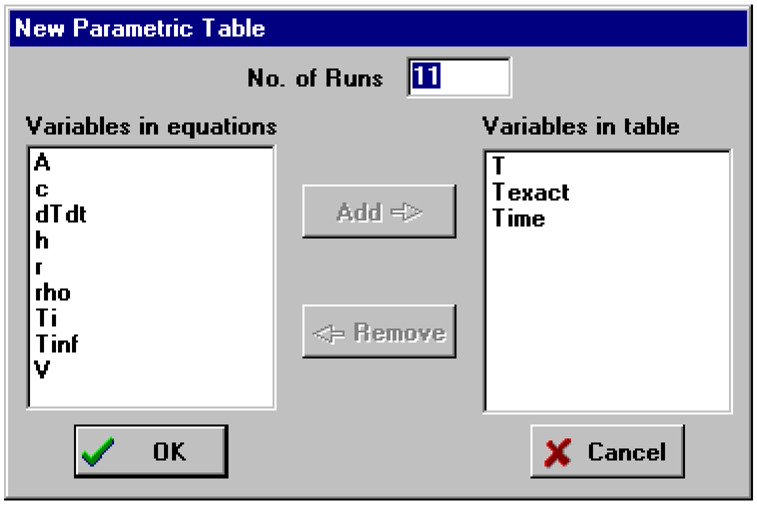
```
"Physical properties"
r=0.005; A=4*pi*r^2;  V=4/3*pi*r^3
"Material properties"
rho=3000; c=1000
"Constants"
Tinf=20;  Ti=400;  h=10
"Energy balance to determine dTdt"
rho*V*c*dTdt=-h*A*(T-Tinf)

"Integrate dTdt to find T as a function of time"
T=Ti+integral(dTdt,Time)
"Exact solution"
(Texact-Tinf)/(Ti-Tinf)=exp(-h*A/(rho*c*V)*Time)
```

Next, a Parametric Table is generated with the New Table command in the Parametrics menu. (Note that, if the start and stop times were provided as the third and fourth parameters for the **Integral** function, it would not be necessary to use the Parametric table. Intermediate results for times between the start and stop times could be directed to the Integral Table using the $IntegralTable directive as shown in Method 2.)

[15] Incropera, F.P. and DeWitt, D.P., *Fundamentals of Heat and Mass Transfer*, 2nd edition, John Wiley and Sons, 1985, Chapter 5

Select T, Time, and Texact as the three variables to include in the table. Enter 11 runs which will allow a time–temperature history for 1000 seconds starting at 0 with 100 second intervals. The New Table dialog window should now appear as shown below.

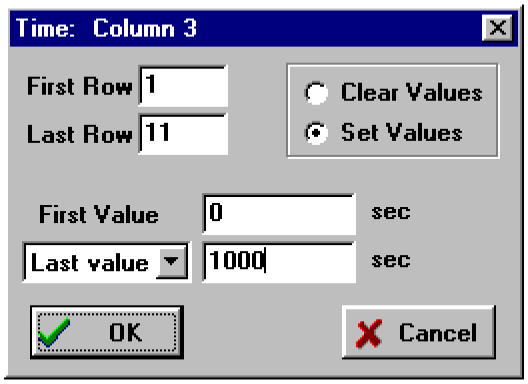
Click the OK button. It is next necessary to enter the values of Time in the table for which the temperature T is to be calculated. A timestep of 100 seconds has been chosen. With a fixed timestep, the values of Time can be most easily entered by clicking on the ▼ control at the upper right of the Time column header cell. Enter 0 for the First Value. Set the drop-down list control to 'Last Value' and enter 1000 for the last value as shown.


New Parametric Table

No. of Runs: 11

Variables in equations: A, c, dTdt, h, r, rho, Ti, Tinf, V

Variables in table: T, Texact, Time


Time: Column 3

First Row: 1

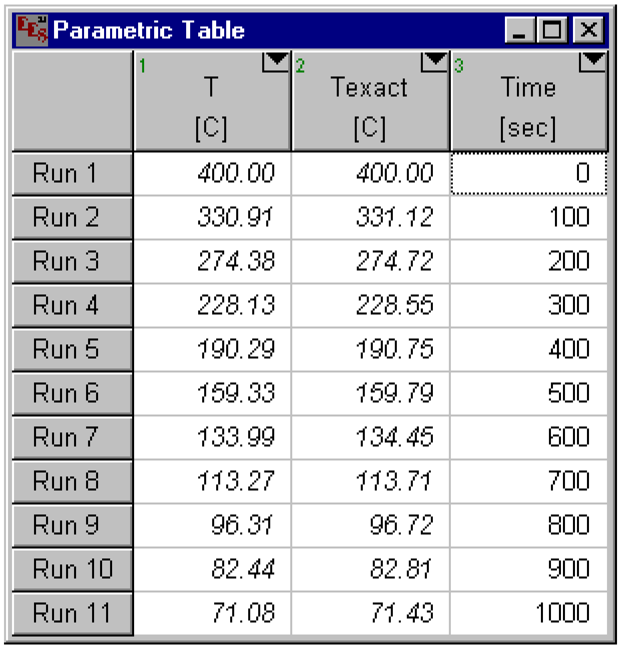
Last Row: 11

( ) Clear Values [✓ Set Values]

First Value: 0 sec

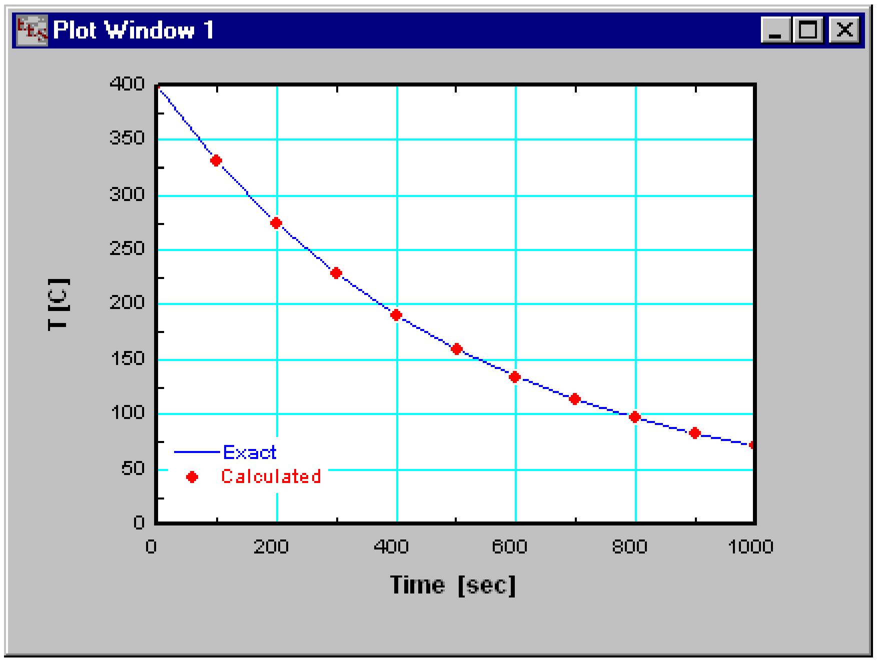
Last value: 1000 sec

The value of Time from 0 to 1000 will be automatically entered into the table when you click the OK button and displayed in normal type. Now select Solve Table from the Calculate menu to calculate the numerical and analytical values of temperature corresponding to each value of Time in the table. When the calculations are completed, the Parametric Table window will display the solutions. Calculated values are shown in bold. (The format of calculated values in the Parametric Table can be set using the Preferences command in the Options menu.) The plot shows that the numerically determined temperature agrees closely with the exact analytic solution.



Parametric Table

	1 T [C]	2 Texact [C]	3 Time [sec]
Run 1	*400.00*	*400.00*	0
Run 2	*330.91*	*331.12*	100
Run 3	*274.38*	*274.72*	200
Run 4	*228.13*	*228.55*	300
Run 5	*190.29*	*190.75*	400
Run 6	*159.33*	*159.79*	500
Run 7	*133.99*	*134.45*	600
Run 8	*113.27*	*113.71*	700
Run 9	*96.31*	*96.72*	800
Run 10	*82.44*	*82.81*	900
Run 11	*71.08*	*71.43*	1000



Method 2: Use of the $Integral Directive and the Equation-Based Integral Function
In this section, we solve the same first-order differential equation described in Method 1 using the equation-based Integral function in conjunction with the $IntegralTable directive. The equation-based Integral function does not employ the Parametric table to specify the values of the integration variable, as does the table-based Integral function. Rather, the lower and upper limits and optionally, the step size of the integration variable is specified as arguments within the Integral function, e.g.,

F = Integral (Integrand, VarName, LowerLimit, UpperLimit, StepSize)

The last argument in the Integral command is the integration step size. If a step size is not provided, EES will use automatic step size. Normally, EES does not keep intermediate values of any variable used during the numerical integration. Such information is needed if you with to know the trajectory or path of variables as well as their values at a specified time. The $IntegralTable directive instructs EES to store the intermediate values in an Integral Table. The values can then can be plotted, printed, or copied to another application. The format of the $IntegralTable directive is:

$IntegralTable Time:100, T, Texact

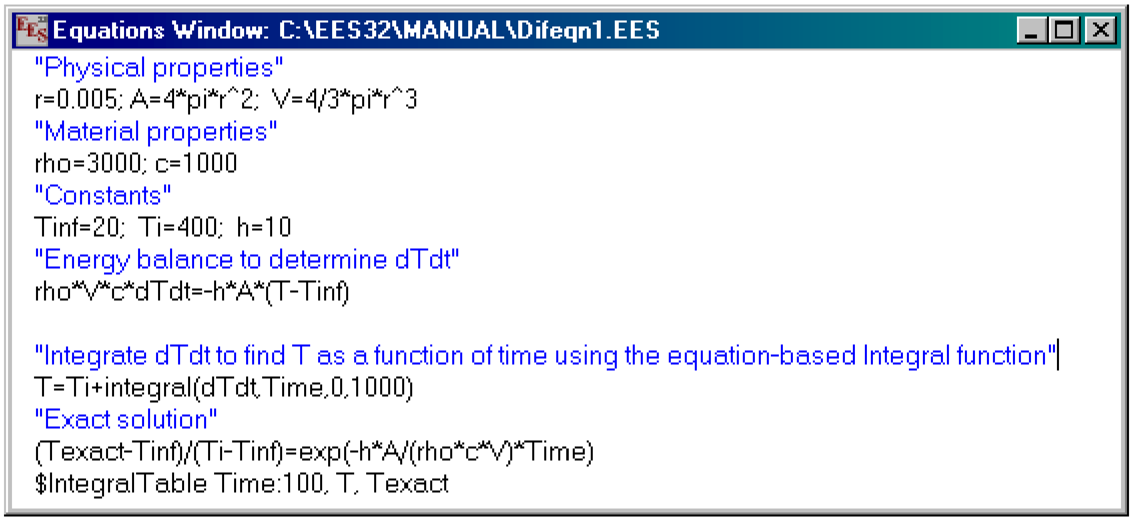
where Time is the integration variable, but it of course can have any legal EES variable name. Time is used here as an example. This variable must appear in one or more equation-based Integral functions within the Equations window. The first column in the Integral Table will hold values of this variable. The colon and number following the integration variable are optional. If a number is supplied, this number will be taken to be the output step size and integration variables will be reported in the Integral Table at the specified step size. If a step size is provided, it must be a number - not a variable. If a step size is not specified, EES will select a step size. Note that the step size used to report integration results is totally independent of the step size of used in the numerical integration. If the numerical integration step size and output step size are not factors, linear regression will be used to determine the integrated quantities at the specified steps.

The remaining parameters in the $IntegralTable directive are variables in the EES program that you wish to see as a function of the integration variable. T and Texact have been selected here. A separate column will be created in the Integral Table for each specified variable. Array range notation, e.g., X[1..5] can be used. The variables must be separated by a space or list delimiter.

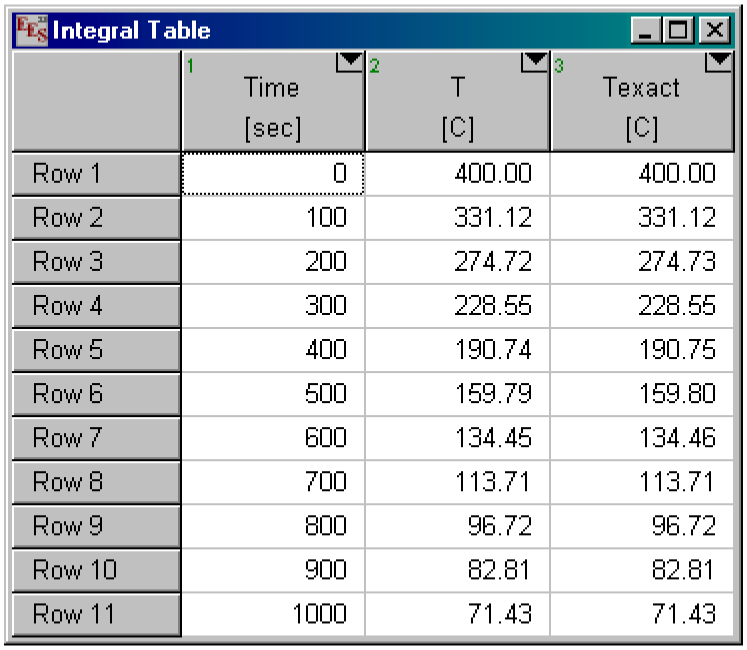
When calculations are initiated, EES will first check to ensure that all variables in the $IntegralTable command are used in the equations. If no errors are found, an Integral Table will be created and filled with intermediate values results from the numerical integration.

Values from this table can then be plotted, printed, and copied, in exactly the same manner as for other tables.

When the $IntegralTable directive is used with the Solve Table command, an Integral Table is produced for each run (i.e., row) of the Parametric Table. Only one Integral Table can be viewed at one time. The run number control in the upper left cell of the Integral Table indicates the Parametric Table run number for which data are currently displayed. This run number may be changed by clicking on the up/down arrows or by directly entering the run number. The data in the table will be automatically updated when the run number is changed. Shown below are the Equations window and the Integral Table that is produced when the equations are solved.



```
"Physical properties"
r=0.005; A=4*pi*r^2;  V=4/3*pi*r^3
"Material properties"
rho=3000; c=1000
"Constants"
Tinf=20;  Ti=400;  h=10
"Energy balance to determine dTdt"
rho*V*c*dTdt=-h*A*(T-Tinf)

"Integrate dTdt to find T as a function of time using the equation-based Integral function"
T=Ti+integral(dTdt,Time,0,1000)
"Exact solution"
(Texact-Tinf)/(Ti-Tinf)=exp(-h*A/(rho*c*V)*Time)
$IntegralTable Time:100, T, Texact
```



	1 Time [sec]	2 T [C]	3 Texact [C]
Row 1	0	400.00	400.00
Row 2	100	331.12	331.12
Row 3	200	274.72	274.73
Row 4	300	228.55	228.55
Row 5	400	190.74	190.75
Row 6	500	159.79	159.80
Row 7	600	134.45	134.46
Row 8	700	113.71	113.71
Row 9	800	96.72	96.72
Row 10	900	82.81	82.81
Row 11	1000	71.43	71.43

Method 3: Solving Differential Equations with the TableValue Function

In this section, we solve the same first-order differential equation described in Method 1.

$$-h\,A\,(T - T_{\infty}) = \rho\,c\,V\,\frac{dT}{dt}$$

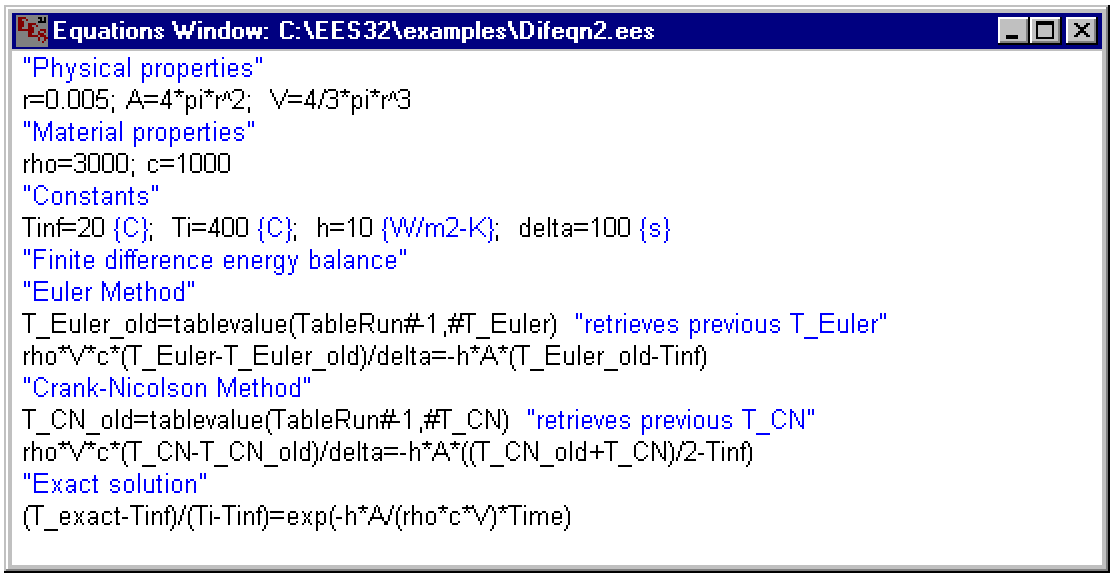
The differential is approximated as

$$\frac{dT}{dt} \simeq \frac{T^{new} - T^{old}}{\Delta t}$$

T^{new} is the temperature at the current time which is to be calculated. T^{old} is the temperature at the previous time which can be found in the previous row of the Parametric table using the **TableValue** function. The **TableValue** function returns the value in the Parametric Table at a specified row and column, as described in Chapter 4. With this function, it is possible to access the values of variables calculated in previous runs during the Solve Table calculations. Δt is the timestep which, in the equations shown below, is the difference of the current and previous values of the variable Time.

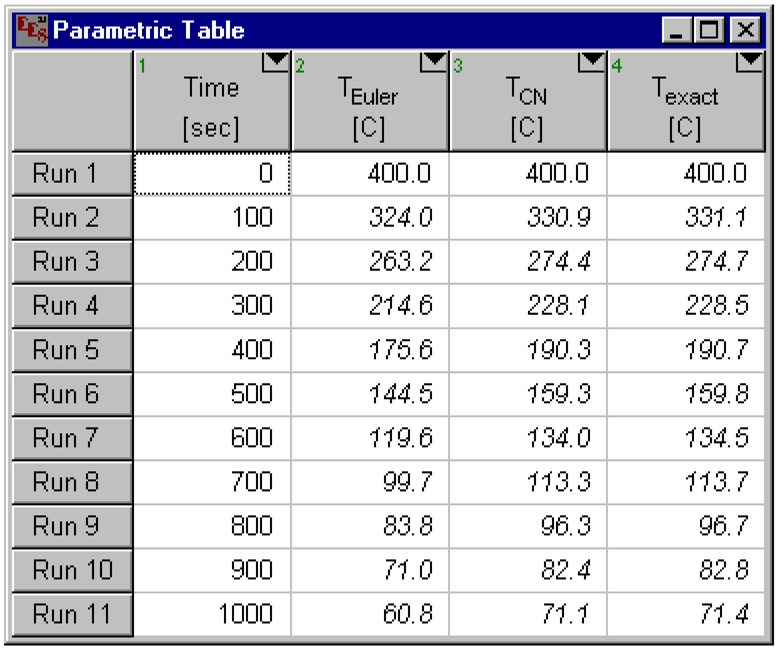
Both an explicit method (Euler's method) and an implicit method (Crank-Nicolson) are used to solve this first-order differential equation and compared with the exact solution. In the Euler method, only previous temperatures are used to evaluate the right-hand side of the differential equation. In the Crank-Nicolson method, the average of the previous and current temperatures is used. The Crank-Nicolson method is implicit because the current temperature is not as yet determined. The implicit method is no more difficult to implement since EES is designed to solve implicit equations. Shown below is a listing of all of the equations needed to solve this problem.

Most of the equations are identical to those used for Method 1. T_Euler is the temperature calculated by Euler’s method. T_CN is the temperature calculated by the Crank-Nicolson method. (In the Formatted Equations and Solution windows, these variables will display as T_{Euler} and T_{CN}, respectively.) To proceed, a Parametric table must be defined, as in Method 1. The values of T_Euler, T_CN and T_exact in the first row of the table, corresponding to Time=0, are the initial conditions and their values (400°C) must be entered. Then, the Solve Table command is used to complete the table, with <u>calculations starting at Run 2</u>. Note the use of the TableRun# function to return the row number in the Parametric Table for which calculations are currently being done, and the TableValue function which returns the value at a specified row and column in the Parametric Table.



```
Equations Window: C:\EES32\examples\Difeqn2.ees
"Physical properties"
r=0.005; A=4*pi*r^2;  V=4/3*pi*r^3
"Material properties"
rho=3000; c=1000
"Constants"
Tinf=20 {C};  Ti=400 {C};  h=10 {W/m2-K};  delta=100 {s}
"Finite difference energy balance"
"Euler Method"
T_Euler_old=tablevalue(TableRun#-1,#T_Euler)  "retrieves previous T_Euler"
rho*V*c*(T_Euler-T_Euler_old)/delta=-h*A*(T_Euler_old-Tinf)
"Crank-Nicolson Method"
T_CN_old=tablevalue(TableRun#-1,#T_CN)  "retrieves previous T_CN"
rho*V*c*(T_CN-T_CN_old)/delta=-h*A*((T_CN_old+T_CN)/2-Tinf)
"Exact solution"
(T_exact-Tinf)/(Ti-Tinf)=exp(-h*A/(rho*c*V)*Time)
```

Shown below is the completed table with the numerical and analytical solutions. Calculated values are shown in italics. It is evident that Euler's method does not provide as accurate a solution as that obtained with the **Integral** function (Methods 1 and 2) or the Crank-Nicolson method. Improved accuracy could be obtained by reducing the time step, but this would require additional computational effort and storage space (which is not significant here).

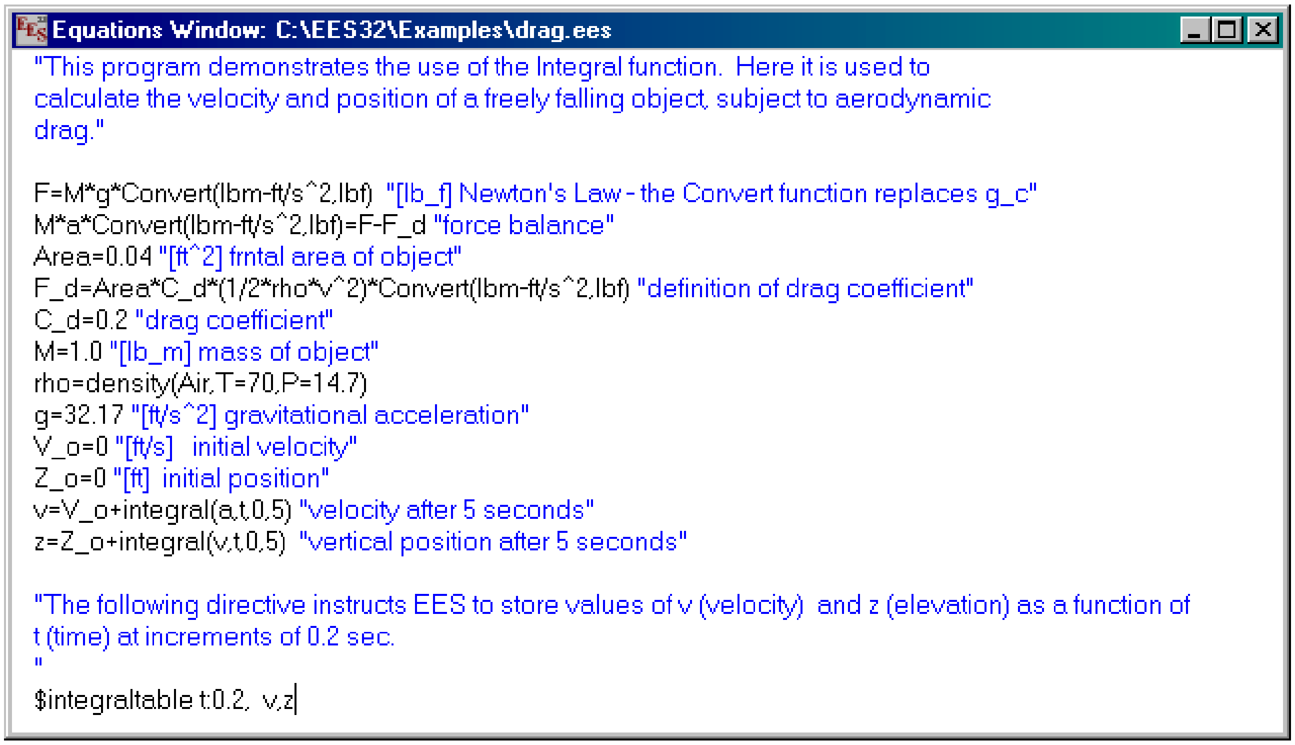


Parametric Table

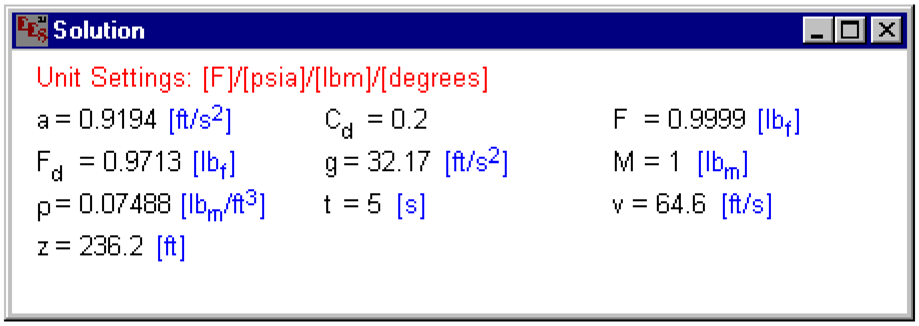
	1 Time [sec]	2 T_{Euler} [C]	3 T_{CN} [C]	4 T_{exact} [C]
Run 1	0	400.0	400.0	400.0
Run 2	100	*324.0*	*330.9*	*331.1*
Run 3	200	*263.2*	*274.4*	*274.7*
Run 4	300	*214.6*	*228.1*	*228.5*
Run 5	400	*175.6*	*190.3*	*190.7*
Run 6	500	*144.5*	*159.3*	*159.8*
Run 7	600	*119.6*	*134.0*	*134.5*
Run 8	700	*99.7*	*113.3*	*113.7*
Run 9	800	*83.8*	*96.3*	*96.7*
Run 10	900	*71.0*	*82.4*	*82.8*
Run 11	1000	*60.8*	*71.1*	*71.4*

Solving Second and Higher Order Differential Equations

Higher order differential equations can also be solved by repeated use of the **Integral** function. Shown below is an EES program which solves a second-order differential equation to calculate the velocity and position of a freely falling object, subject to aerodynamic drag. The Solution Window appears after the Solve command (F2) is issued. The Integral Table that is produced shows how the velocity and position of the object vary with time.



```
"This program demonstrates the use of the Integral function.  Here it is used to
calculate the velocity and position of a freely falling object, subject to aerodynamic
drag."

F=M*g*Convert(lbm-ft/s^2,lbf)  "[lb_f] Newton's Law - the Convert function replaces g_c"
M*a*Convert(lbm-ft/s^2,lbf)=F-F_d "force balance"
Area=0.04 "[ft^2] frntal area of object"
F_d=Area*C_d*(1/2*rho*V^2)*Convert(lbm-ft/s^2,lbf) "definition of drag coefficient"
C_d=0.2 "drag coefficient"
M=1.0 "[lb_m] mass of object"
rho=density(Air,T=70,P=14.7)
g=32.17 "[ft/s^2] gravitational acceleration"
V_o=0 "[ft/s]  initial velocity"
Z_o=0 "[ft]  initial position"
v=V_o+integral(a,t,0,5) "velocity after 5 seconds"
z=Z_o+integral(v,t,0,5)  "vertical position after 5 seconds"

"The following directive instructs EES to store values of v (velocity)  and z (elevation) as a function of
t (time) at increments of 0.2 sec.
"
$integraltable t:0.2,  v,z
```



Unit Settings: [F]/[psia]/[lbm]/[degrees]

$a = 0.9194$ [ft/s^2] $C_d = 0.2$ $F = 0.9999$ [lb$_f$]

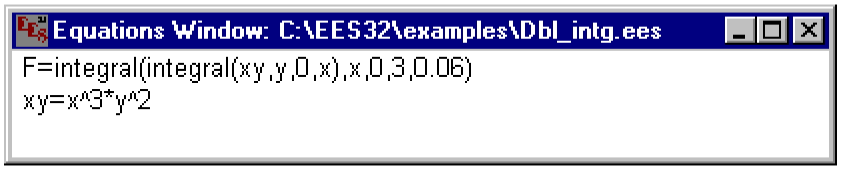
$F_d = 0.9713$ [lb$_f$] $g = 32.17$ [ft/s^2] $M = 1$ [lb$_m$]

$\rho = 0.07488$ [lb$_m$/ft^3] $t = 5$ [s] $v = 64.6$ [ft/s]

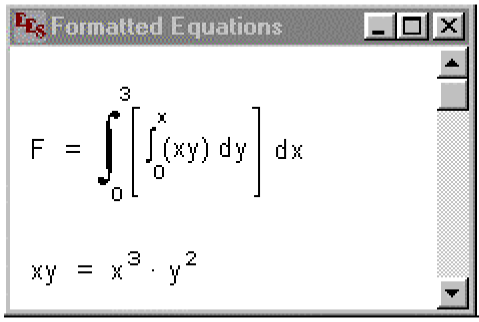
$z = 236.2$ [ft]

Multiple-Variable Integration

Multiple integration is provided by nesting calls to the **Integral** function. Up to six levels can be nested. The following example performs a numerical double integration using the equation-based **Integral** function.

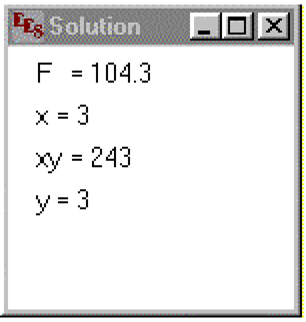

Equations Window: C:\EES32\examples\Dbl_intg.ees

```
F=integral(integral(xy,y,0,x),x,0,3,0.06)
xy=x^3*y^2
```

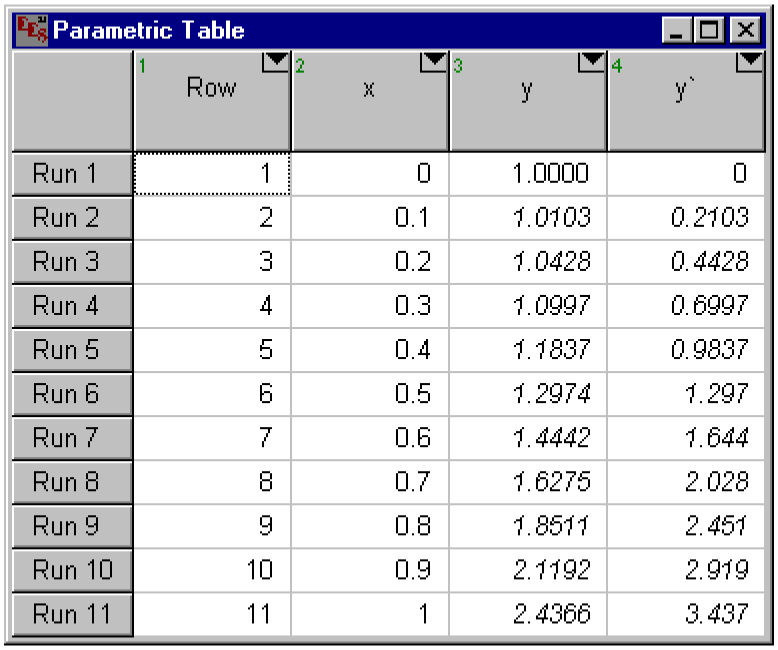

Formatted Equations

$$F = \int_0^3 \left[\int_0^x (xy)\, dy \right] dx$$

$$xy = x^3 \cdot y^2$$


Solution

F = 104.3
x = 3
xy = 243
y = 3


Parametric Table

	1 Row	2 x	3 y	4 y`
Run 1	1	0	1.0000	0
Run 2	2	0.1	1.0103	0.2103
Run 3	3	0.2	1.0428	0.4428
Run 4	4	0.3	1.0997	0.6997
Run 5	5	0.4	1.1837	0.9837
Run 6	6	0.5	1.2974	1.297
Run 7	7	0.6	1.4442	1.644
Run 8	8	0.7	1.6275	2.028
Run 9	9	0.8	1.8511	2.451
Run 10	10	0.9	2.1192	2.919
Run 11	11	1	2.4366	3.437

Creating and Using Macro Files

A macro is a set of instructions to EES that are read from an ASCII file. The filename extension for a macro file is .EMF, which signifies EES Macro File. The instructions allow EES to open a file, solve the equations, create and solve a table, store calculated results in a file, print, and other functions. In fact, many of the capabilities provided with the menu commands in EES can be implemented with a macro instruction. In addition, EES can directly communicate with Microsoft WORD with the WORD macro commands.

Using the macro file capability, EES can be run from the Windows command line or it can be controlled from another program, either by running EES with the macro file name or by dynamic data exchange (DDE) with the calling program.

To run an EES macro from the command line or from another application, the macro file name is placed after the EES executable file name. For example, to start EES and have it run the commands in file myMac.emf stored in the C:\EES32 directory, you would enter:

```
C:\EES32\EES.EXE  C:\EES32\myMac.emf
```

The .emf filename extension is required. Otherwise EES will consider the file to be a normal EES file. When EES is started with this command, it will remain hidden while it opens the macro file and executes each of the instructions in that file. Provided that no errors are encountered, all of the instructions will be executed in the order in which they appear and then EES will quit. EES writes an ASCII log file called EESMacro.LOG that indicates what instructions it has completed and any error messages. This log file is placed in the same directory as the EES executable.

Creating an EES Macro File

The easiest way to create an EES Macro file is to use the Build Macro command in the File menu. After selecting this command, EES will prompt the user to provide a name for the macro file. Then a small macro file window will appear at the bottom of the screen. If the filename you provided already exists, it will be read into macro file window. Otherwise it will be blank. Now as you select commands form the EES menus, the macro equivalents of the commands will be automatically entered into the macro file window. For example, when you apply the Open command, EES will add a line to the window of the form.

```
OPEN C:\myfile.EES
```

The macro command window is editable, so that you can modify the commands that appear in this window or delete them if you wish. Most of the EES menu commands will produce a macro instruction. Some commands, such as those in the Windows menu, are not applicable for a macro file since EES is usually not visible while running the macro commands. Others, such as those which are used to modify plots, are not supported at this time. The Build Macro command name is changed to Close Macro after you open a macro command window. When you are done entering instructions, select the Close Macro command to close the macro file.

The major purpose of the macro capability is to allow an external program to use the EES solving engine. Using the macro file, EES can be instructed to open an existing EES or .TXT file and solve the equations in that file. EES macro files provide instructions to save the contents of the Solution and Parametric Table windows so that the solution can be returned to the program that called EES.

Running an EES Macro File by Dynamic Data Exchange

The easiest way to run an EES macro file from another application is to simply start EES and provide the macro file name (with an .EMF filename extension) as a parameter on the command line. However, the disadvantage of this method is that EES quits when the macro file commands have been executed. If it is necessary to call EES repeatedly, this method becomes very inefficient as EES must be repeatedly loaded into memory. An alternative approach is to use dynamic data exchange (DDE) messages.

EES must be running to receive a DDE message. This is not really a problem as EES can be started from a remote application using the Windows CreateProcess command. However, when EES starts, it normally displays its splash screen and waits for the user to click the OK button. To avoid having EES display the splash screen and wait for user input, supply /HIDE as a parameter on the command line, i.e., start EES with the following command.

```
C:\EES32\EES.EXE /HIDE
```

After receiving this command EES will start in a minimized mode. Its icon will be visible in the Windows task bar, but the program will otherwise not be visible on the screen.

The calling program must next send EES a series of messages according to the DDE message protocol. The first message that must be sent is the wm_DDE_initiate message. This message is normally sent using the Windows SendMessage command. The SendMessage command requires four parameters. The first should be a broadcast message to all running applications. The second must be wm_DDE_initiate. The third is the handle of the calling application. EES will use this handle to send message back to the calling application. The final parameter is a global atom referring to 'EES', so that EES knows to act on this initiate message. Shown below is a code fragment in Delphi 4 that sends the wm_DDE_initiate message to EES.

```
procedure TForm1.doInitiate(Sender: TObject);
  var theApp, theTopic:Atom;
       lParam:longInt;
begin
  theApp:=GlobalAddAtom('EES');
  theTopic:=GlobalAddAtom('');
  lParam:=MakeLong(theApp,theTopic);
  SendMessage(HWND(-1),wm_DDE_Initiate,Handle,lParam);
  GlobalDeleteAtom(theApp);
  GlobalDeleteAtom(theTopic);
end;
```

EES will respond to the wm_DDE_Initiate message by sending the application a wm_DDE_ack message. Included in this message is the handle to the EES application, here called EESWindowHandle, that the calling program must provide in following messages. A code segment that receives the wm_DDE_Ack message may appear as shown below.

```
procedure TForm1.GetACK(var theMessage:TMessage);
  var AppName,TopicName:charString;
      S:shortString;
 begin
    EESWindowHandle:=theMessage.wParam;
 end;
```

After the wm_DDE_initiate message has been received by EES, the calling application can next send a wm_DDE_Request message to play a macro. The information sent to EES includes a global atom referring to the string 'PLAY' and a character string containing the DOS filename of the macro file. The complete file name should be sent, including the directory information and the .EMF filename extension. The code fragment below shows how this is done.

```
procedure TForm1.PlayMacro(Sender: TObject);
  var cfile,command:atom;
       lParam:longint;
       CSTR:charString;
begin
  StrCopy(CStr,'c:\ees32\myMacro.emf');
  cfile:=GlobalAddAtom(CStr);
  Command:=GlobalAddAtom('PLAY');
  lParam:=MakeLong(cfile,Command);
  PostMessage(EESWindowHandle,wm_DDE_REQUEST,Handle,lParam);
```

```
end;
```

After receiving this message, EES will open the macro file and execute the instructions therein. It will then post a WM_DDE_ACK message to inform the calling program that the calculations are completed. When EES executes a macro file, it writes results to a specified text file. The calling application can open this text file to retrieve the results.

The only other message EES will accept is the quit command which terminates EES. Here's how it can be sent.

```
procedure TForm1.QuitClick(Sender: TObject);
  var cformat,command:atom;
      lParam:longint;
begin
  cformat:=GlobalAddAtom('');
  Command:=GlobalAddAtom('QUIT');
  lParam:=MakeLong(cformat,Command);
  PostMessage(EESWindowHandle,wm_DDE_REQUEST,Handle,lParam);
end;
```

AlterValues P2 Rows=1..10 First=100 Last=550
{Enter values into the Parametric Table for the column holding variable P2 setting the values of rows 1 to 10 to values starting at 100 and ending at 550. You can specify Inc=50 instead of providing Last=550}

Macro Command List

AlterValues 'Table 1' P2 Rows=1..10 First=100 Last=550
{Enter values into the Parametric Table named 'Table 1' for the column holding variable P2 setting the values of rows 1 to 10 to values starting at 100 and ending at 550. You can specify Inc=50 instead of providing Last=550}

Copy ArraysTable R1 C1:R10 C4 {Copy the specified range of the Arrays table to the clipboard. If no range is provided, the entire table is copied}

Copy IntegralTable R1 C1:R10 C4 {Copy the specified range of the Integral table to the clipboard. If no range is provided, the entire table is copied}

Copy EquationWindow L3 C1:L5 C9 {Copy the characters in the Equations WIndow starting with character 1 on line 3 through character 9 on line 5}

Copy LookupTable 'Lookup 1' R1 C1:R10 C4 {Copy the specified range of the Lookup table with name 'Lookup 1' to the clipboard. If no range is provided, the entire table is copied}

Copy ParametricTable 'Table 1' R1 C1:R10 C4 {Copy the specified range of the Parametric table named 'Table 1' to the clipboard. If no range is provided, the entire table is copied}

Copy PlotWindow 3 {Copies the graphics in Plot Window 3 in both metafile bitmap formats}

Copy SolutionWindow {Copies the entire contents of the Solution window in ASCII format}

DeletePlot 1 {Delete Plot Window 1.}

LoadLib C:\EES32\myLib.lib {Load a library file called C:\EES32\myLib.lib}

Lookup['Lookup 1', 2,3]=10 {Set the value in row 2 column 3 of the Lookup table having name 'Lookup 1' to 10}

ModifyAxis X Plot=1 Min=100 Max=600 Int=100 Linear Grids=on Showscale=off Size=10 Style=BoldItalic Format=A3
{Change the scaling of the X axis of plot window 1 to the specified minimum, maximum and interval in linear or log coordinates. The scale numbers are shown in bold italic with fontsize 10 using automatic format. The gridlines and scale can be turned on or off. Use Y for the left Y axis and Y2 for the right Y axis}

ModifyPlot 1 Width=12 cm Height=15 cm
{Change the size of plot rectangle for the specified plot window. If the plot window number is not provided, the command is applied to the foremost plot window. Size can be specified in cm or in}
New {Clear the workspace and bring up an empty Equations window}

NewLookup 'Lookup 1' Rows=24 Cols=2 {creates a new Lookup table having name 'Lookup 1' with 24 rows and 2 columns}

NewPlot Table=PAR2 X=P2 Y=T2 Rows=1..10 Line=1 Symbol=1 Color=Red
{Create a new plot window and Plot T2 vs P2 using the first 10 rows of data from table in Tab 2 of the Parametric table window}

NewTable 'Table 1' Rows=10 X Y Z   Create a new Parametric table having name 'Table 1' with 10 rows and columns for variables X, Y and Z}

Open C:\EES32\Examples\Regen.ees   {open the specified file}

OpenLookup C:\EES32\myTable.lkt   {open the specified Lookup file}

OverlayPlot 1 Table=Par2 X=P[4] Y=Eff Rows=1..10 Line=1 Symbol=4 Color=Blue Right

{overlay a plot of Eff vs. P[4] from data in rows 1..10 of the table in Tab 2 of the Parametric table.   Use the right Y-scale. To plot other tables, use Look for Lookup table, Arr for Arrays table or Int for Integral table.}

Parametric['Table 1',1,4]=20 {Set the value in row 1 column 4 of the Parametric table named 'Table 1' to 20}

Paste Lookup 'Lookup 1' R2 C3   {Paste the contents of the clipboard into the Lookup table named 'Lookup 1' starting at row 2 column 3.}

Paste Parametric 'Table 1' R2 C3   {Paste the contents of the clipboard into the Parametric table named 'Table 1' starting at row 2 column 3.}

Print Diag Equ For Arr Par2 Look3 Plot1
{Print the Diagram, Equations, FormattedEqns, Arrays, the Parametric table in the second tab of the Parametric Table, the Lookup Table in the 3rd tab of the Lookup Table and Plot 1 windows}

PrintSetup PRINTER='\\SEL\Sandy' Orientation=PORTRAIT Copies=1

{Set the printer specifications}
PropPlot Steam TS 4 11000 5300 2100 660 6 0.074 0.94 3.37 12 45 155 DoQLines
{Create a TS property plot for Steam. Include 4 constant pressure lines and 6 constant entropy lines at the specified values and draw lines of constant quality}
ResetGuesses   {Reset the guess values of all variables to their default values}

SolveTable 'Table 1' Rows=1..10   {Solve rows 1 through 10 of the Parametric Table having the name 'Table 1'. If no range is specified, all rows are solved.}

SaveArrays myArrays.txt   {Save the contents of the Arrays table to a file called myArrays.txt.

SaveLookup 'Lookup 1' myTable.txt {Save the contents of the Lookup table having the tab name 'Lookup 1' to a file called mytable.txt.

SaveSolution mySoln.txt /Append {Save the contents of the Solution window to an existing file called mySoln.txt. The /Append option will allow writing to the bottom of the file without deleting the existing information.}

SaveTable 'Table 1' myTable.txt /Append {Save the contents of the table named 'Table 1' in the Parametric Table Window to an existing file called myTable.txt. The /Append option will allow writing to the bottom of the file without deleting the existing information.}

Show Equations {Bring the Equations window to the front. Any window, e.g., PlotWindow 2, ParametricTable,DiagramWindow, etc. can be shown in this manner}

Solve {Solve as if the Solve command has been issued}

StopCrit It=100 Time=3600 Res= 1.0E-0006 Var= 1.0E-0006
{Set the Stop Criteria properties as indicated}

Units C kPa MASS DEG {Set the Units as indicated. Mixed units, e.g., C and psia, are not supported.}

UpdateGuesses {Update the guess values of all variables to the last set of calculated values}

VarInfo P2 Lower=200 Upper=500 Guess=300
{Set the lower bound, upper bound, and guess value for variable P as specified}

WORD.FileNew {Start WORD and create a new empty document.}

WORD.FileOpen('FileName'){Start WORD and open the specified Word file. It is necessary to provide the complete file name.}

WORD.FileSaveAs('FileName') {Save the current WORD document with the specified filename.}

WORD.Hide {Hide the open WORD document.}

WORD.Insert('any text here') {Insert the text contained with single quotes into the WORD document at the current position.}

WORD.Paste {Paste the current contents of the Clipboard into WORD.}

WORD.PasteSpecial(formatType) {Paste the current contents of the Clipboard into WORD in the specfied format. The formatType can be TEXT, PICTURE, BITMAP, DEVICE INDEPENDENT BITMAP, and ENHANCED METAFILE.}

WORD.Quit {Close the communication with WORD.}

WORD.Show {Make the open WORD document visible.}

Appendix A

Hints for Using EES

1. The Variable Information command in the Options menu produces an alphabetical list of all variables appearing in the Equations window. Check this list to make sure that you have not misspelled a variable name.

2. The Residuals window provides an indication of the accuracy in which each non-trivial equation in the Equations window was solved *and* the order in which the equations are solved. An examination of the residuals indicates which equations were not solved when EES indicates that a solution could not be found.

3. If your equations do not converge, it may be that the guess values are poor. In this case, the problem can often be solved by entering equations which set guess values for one or more of the unknown variables and modifying the equations as needed to ensure an equal number of variables and equations. If a solution is then obtained, use Update Guesses in the Calculate menu to set the guess values of all variables to their current values. Then return the Equations window to its original form and solve again.

4. If EES is unable to solve your set of nonlinear equations, try exchanging some independent and dependent variables to produce an equation set which is easier to solve. For example, EES may not be able to solve the following heat exchanger equations to determine NTU with the default guess values and bounds.

   ```
   Eff=.9
   Cmax = 432
   Cmin = 251
   eff = (1 - exp(-NTU * (1 - (Cmin/Cmax)))) / (1 - (Cmin/Cmax) * exp(-NTU *
       (1 - (Cmin/Cmax))))
   ```

   However, the equations would be easily solved if the value of NTU were specified in place of Eff.

   ```
   NTU = 5
   Cmax = 432
   Cmin = 251
   eff = (1 - exp(-NTU * (1 - (Cmin/Cmax)))) / (1 - (Cmin/Cmax) * exp(-NTU *
       (1 - (Cmin/Cmax))))
   ```

   A few trials will indicate that NTU must be between 3 and 5 for Eff = 0.9. Setting a guess value for NTU of 4 allows EES to quickly determine the final value of 3.729.

5. A sure way to solve difficult problems with EES is to add an additional variable so that the problem has one more degree of freedom. Then, use the Parametric Table to vary the values of one of the implicit variables in order to find the solution in which the additional variable has a value of zero. For example, consider the following radiation calculation in which the value of T is to be determined. The first three equations must be solved simultaneously and they are non-linear because T is raised to the fourth power. EES may have trouble determining the solution, depending upon the guess values.

```
QL = AL*Sigma*(T^4 - TL^4)
QB = AH*Sigma*(TH^4 - T^4)
QL = QB
Sigma=0.1718E-8
AL=.5;  AH=1;  TL=300;  TH=1000
```

   Alternatively, add a variable, Delta, such that

```
QL = AL*Sigma*(T^4 - TL^4)
QB = AH*Sigma*(TH^4 - T^4)+Delta
QL = QB
Sigma=0.1718E-8
AL=.5;  AH=1;  TL=300;  TH=1000
```

   Now, set up a Parametric Table containing variables T and Delta. Use the Alter Values command to set a range of values of T and Solve Table to calculate the corresponding values of Delta. The value(s) of T for which Delta is zero constitute a solution to the equation set. The New Plot Window command is handy for visualizing the relationship between T and Delta. If the values of Delta do not cross zero, there is no solution to the equation set for the range of T values investigated. Once a reasonable value of T is found, it can be entered as an equation in the Equations window and an approximate solution corresponding to this value of T can be found. The final step is to use the Update Guesses command in the Calculate menu, set Delta=0 and remove the equation that set the value of T. EES will now quickly converge to the correct solution. This is perhaps the most useful method of solving a difficult set of non-linear equations.

6. The Store button in the Default Info dialog can be particularly convenient if you have a customary set of nomenclature for your variable names. For example, if variables beginning with letter T often designate temperatures, set the bounds, display format and units for letter T and then Store the default information. You can later use the Load button to restore this set of default variable information. EES will then always set this information for you in following problems.

9. The arrow keys help some users move about more quickly in the Equations, Parametric and Lookup Tables. In the Equations window, the up and down arrows move the cursor up and

down one line; left and right arrow move the cursor left and right one character. Home and End move to the start and end of the current line. In the tables, the arrow keys move to the next cell in the direction of the arrow. The Return and Tab keys produce the same effects as the down arrow and right arrow keys, respectively.

8. Use the Tab key in the Equations window to set off the equations for improved readability.

9. Only the high-accuracy thermodynamic property correlations, e.g., Steam_NBS, CarbonDioxide, Ammonia_ha, Methane_ha, etc. are specifically applicable in the compressed (sub-cooled) liquid range. All other real thermodynamic property relations assume that subcooled liquid is incompressible and properties are taken to be those for saturated liquid. Thus, in the subcooled region, $v(T,P)=v(T,P_{sat})$, $u(T,P)=u(T,P_{sat})$ and $s(T,P)=s(T,P_{sat})$. To calculate the ideal work of a pump, for example, recall that h2-h1 = -Wpump = $\int v\, dP$ = v*(P2-P1), since for an incompressible substance, v is independent of P.

10. The Arrays window can be quite useful for organizing the property information in a thermodynamics problem having multiple states. Use array variables, such as T[1], P[1], and h[1] (rather than T1, P1, and h1) for the properties at each state. The state properties will appear in a neat table in the Arrays window, rather than jumbled together in the Solutions window. Be sure the Use Arrays window option in the Display Options dialog has been selected.

11. Considerable effort has been expended to design EES so that it does not unexpectedly quit under any circumstances. However, this may still happen. In this case, EES will try to save your work in a file called EESERROR before it terminates. You can restart EES and load the EESERROR file so none of your work is lost.

12. Use the $INCLUDE directive to load commonly used constants, unit conversions, or other equations into the Equations window. You won't see them, but they're there, available for use. You can also load library files with the $INCLUDE directive.

13. If you write an EES Library Function which calls any of the built-in thermodynamic or trigonometric functions, use the UnitSystem command to determine the current unit system settings. Then you can use If Then Else statements to ensure that the arguments provided to the thermodynamic or trigonometric functions have the correct values.

14. The background color option for columns in the Parametric Table is useful if you are preparing an EES program in which others will be entering data into the columns, as in a spreadsheet. Set the background color for the columns in which data are to be entered, to distinguish them from the columns in which calculated results will appear.

15. If you are working in Complex Mode, use the $COMPLEX ON directive at the top of the Equations window. Its more convenient than changing the Complex Mode setting in the Preferences dialog.
16. To enter µm, hold the Alt-Key down and type 230 on the numeric keypad. Let the Alt key up and the µ should appear. Then enter m. Other useful characters are Alt-248 which displays the degree symbol ° and Alt-250 which appears as a dot (·) and is used to represent multiplication.

Appendix B

Numerical Methods Used in EES

EES uses a variant of Newton's method [1-4] to solve systems of non-linear algebraic equations. The Jacobian matrix needed in Newton's method is evaluated numerically at each iteration. Sparse matrix techniques [5-7] are employed to improve calculation efficiency and permit rather large problems to be solved in the limited memory of a microcomputer. The efficiency and convergence properties of the solution method are further improved by the step-size alteration and implementation of the Tarjan [8] blocking algorithm which breaks the problems into a number of smaller problems which are easier to solve. Several algorithms are implemented to determine the minimum or maximum value of a specified variable [9,10]. Presented below is a summary of these methods, intended to provide users with a better understanding of the processes EES uses in obtaining its solutions.

Solution to Algebraic Equations

Consider the following equation in one unknown:

$$x^3 - 3.5\,x^2 + 2\,x = 10$$

To apply Newton's method to the solution of this equation, it is best to rewrite the equation in terms of a residual, ε, where

$$\varepsilon = x^3 - 3.5\,x^2 + 2\,x - 10$$

The function described by this equation is shown in Figure 1. There is only one real solution (i.e., value of x for which $\varepsilon = 0$) in the range illustrated at x = 3.69193.

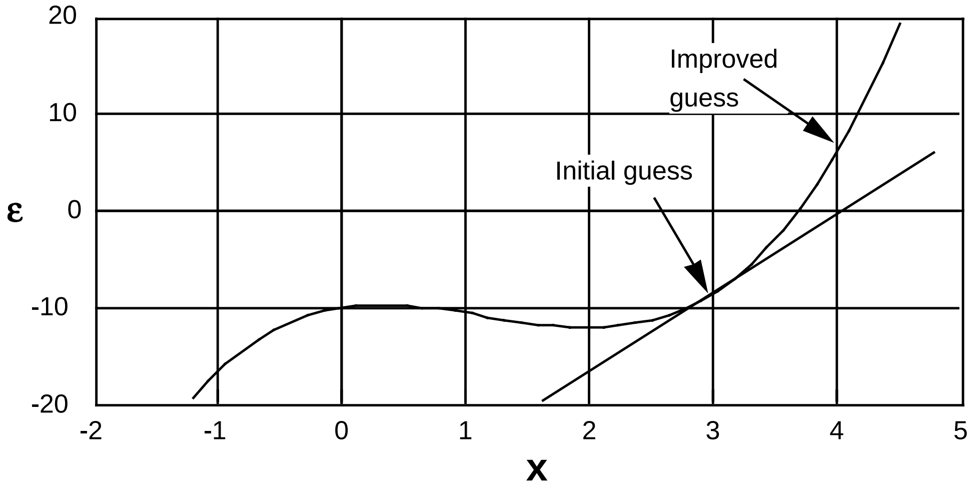


Figure 1: Residual of $x^3 - 3.5\,x^2 + 2\,x = 10$ as a function of x

Newton's method requires an estimate of the total derivative of the residual, J. For this equation, the derivative is:

$$J = \frac{d\varepsilon}{dx} = 3\,x^2 - 7\,x + 2$$

To solve the equation, Newton's method proceeds as follows:

1. An initial guess is made for x (e.g., 3).
2. The value of ε is evaluated using the guess value for x. With x = 3, ε = -8.5.
3. The derivative J is evaluated. With x = 3, J = 8.
4. The change to the guess value for x, i.e., Δx, is calculated by solving J Δx = ε. In this example, Δx is -1.0625.
5. A (usually) better value for x is then obtained as x - Δx. In the example, the improved value for x is 4.0625 (which results in ε = 7.4084).

Steps 2 to 5 are repeated until the absolute value of ε or Δx becomes smaller than the specified tolerances in the Stop Criteria dialog. The method, when it converges, converges quite quickly. However, a bad initial guess can cause the method to diverge or to converge quite slowly. Try, for example, an initial guess of 2 and see what happens.

Newton's method can be extended for solving simultaneous non-linear equations. In this case, the concept of "derivative" generalizes into the concept of "Jacobian matrix." Consider the following two simultaneous equations in two unknowns:

$$x_1^2 + x_2^2 - 18 = 0$$

$$x_1 - x_2 = 0$$

The equations can be rewritten in terms of residuals ε_1 and ε_2:

$$\varepsilon_1 = x_1^2 + x_2^2 - 18 = 0$$

$$\varepsilon_2 = x_1 - x_2 = 0$$

The Jacobian for this matrix is a 2 by 2 matrix. The first row contains the derivatives of the first equation with respect to each variable. In the example above, the derivative of ε_1 with respect to x_2 is 2 x_2. The Jacobian matrix for this example is:

$$J = \begin{bmatrix} 2\cdot x_1 & 2\cdot x_2 \\ 1 & -1 \end{bmatrix}$$

Newton's method as stated above applies to both linear and nonlinear sets of equations. If the equations are linear, convergence is assured in one iteration, even if a "wrong" initial guess was made. Non-linear equations require iterative calculations. Consider the following initial guess:

$$x = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

The values of ε and J for this initial guess are: $\varepsilon = \begin{bmatrix} -10 \\ 0 \end{bmatrix}$ $J = \begin{bmatrix} 4 & 4 \\ 1 & -1 \end{bmatrix}$

Improved values for the x vector are obtained by solving the following matrix problem involving the Jacobian and the residual vector.

$$\begin{bmatrix} 4 & 4 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} -10 \\ 0 \end{bmatrix}$$

Solving this linear equation results in:

$$\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} -1.25 \\ -1.25 \end{bmatrix}$$

Improved estimates of the x_1 and x_2 are obtained by subtracting Δx_1 and Δx_2, respectively, from the guess values.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3.25 \\ 3.25 \end{bmatrix}$$

The correct solution to this problem is $x_1 = x_2 = 3.0$. The calculated values of x_1 and x_2 are closer to the correct solution than were the guess values. The calculations are now repeated using the most recently calculated values of x_1 and x_2 as the guess values. This process is repeated until convergence is obtained.

The Jacobian matrix plays a key role in the solution of algebraic equations. The Jacobian matrix can be obtained symbolically or numerically. Symbolic evaluation of the Jacobian is more accurate, but requires more processing. Accuracy of the Jacobian, however, does not necessarily lead to more accuracy in the solution, only to (sometimes) fewer iterations. EES evaluates the Jacobian numerically. Because EES does all calculations with 96 bit precision (about 20 decimal places), numeric evaluation of the Jacobian rarely results in convergence problems from loss of precision.

In most equation sets, many of the elements of the Jacobian matrix are zero. A matrix with many zero elements is called a sparse matrix. Special ordering and processing techniques make handling of sparse matrices quite efficient. In fact, without sparse matrix techniques the number of simultaneous equations which could be solved by EES would be substantially less than 6000, (10,000 in the Professional version) the current limit implemented in EES. Further references on sparsity and on how to handle sparse matrices are available in [5, 6]. A collection of routines that are designed to handle very large sparse matrices are described in [7].

Newton's method does not always work, particularly if a "bad" initial guess for the x vector is supplied. The solution obtained after applying the correction Δx to the previous x vector should be more correct (i.e., result in a smaller maximum residual) than the solution obtained before the correction. EES always checks for this condition. If this is not true, EES will halve the step Δx and evaluate the residuals again. If this does not improve the solution, the step is halved again (up to 20 times). If the resultant solution is still not better than the solution prior to the correction, EES will reevaluate the Jacobian and try again until one of the stopping criteria forces the calculations to stop. Step halving is very helpful when a bad initial guess is provided. Figure 2 illustrates the process for the solution of the single equation in the first example, starting from a guess of x=2.5. In this case, step halving works quite well.

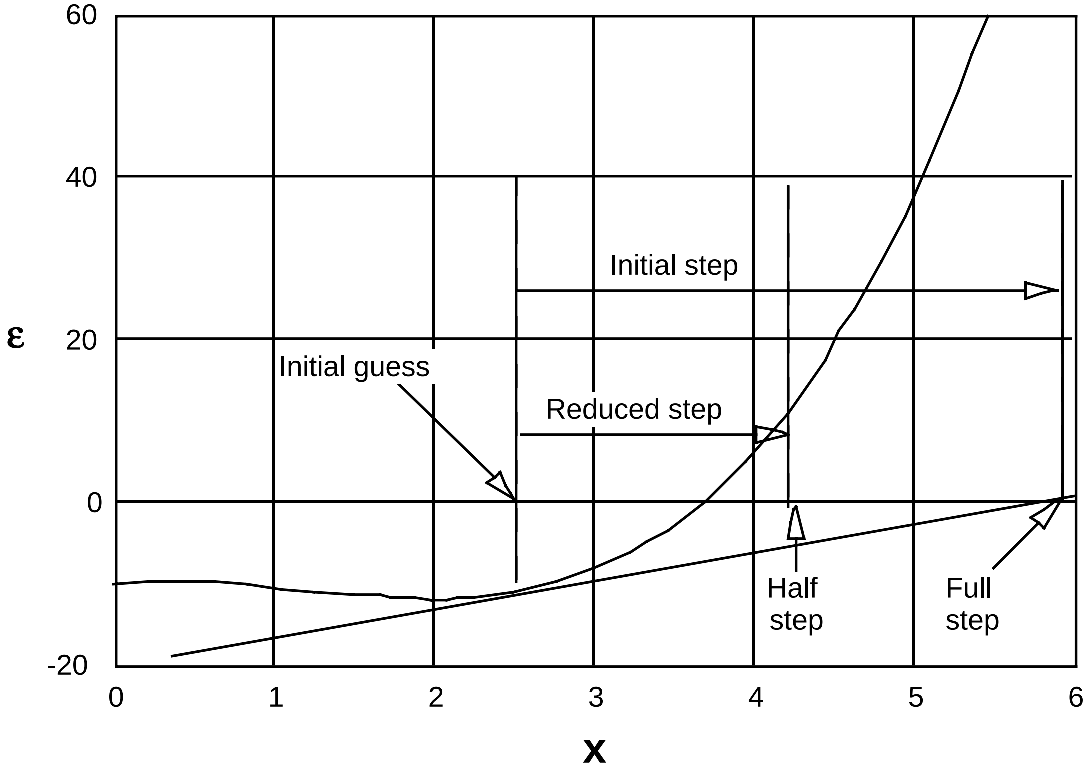


Figure 2: Use step-halving for improving the convergence

Blocking Equation Sets

Even though you may have what looks like a set of simultaneous equations, it is often possible to solve these equations in groups (sometimes one at the time) rather than all together as one set. Solving equations in groups makes Newton's method work more reliably. For this reason, EES organizes the equations into groups (or blocks) before solving. Consider, for example, the following set of equations:

$$\begin{aligned} x_1 + 2\,x_2 + 3\,x_3 &= 11 \\ 5\,x_3 &= 10 \\ 3\,x_2 + 2\,x_3 &= 7 \end{aligned}$$

These equations can be solved as one simultaneous set. However, they can be more easily solved if they are reordered and blocked. It is better to re-order first. EES automatically recognizes that equation 2 can be solved directly for x_3. Once this is done, equation 3 can be solved for x_2. Finally, equation 1 can be solved for x_1. This results in three blocks of equations, each with one equation and one variable which are directly solved. Because the equations in this example are linear and they can be totally uncoupled, the process looks trivial. Things can get a little more interesting if the blocks are a little less obvious. Consider the following example with 8 linear equations in 8 unknowns:

$$\begin{array}{llllllllll}
 & & x_3 & & & & & + x_8 & = 11 \\
 & & & & & & x_7 & & = 7 \\
 & & & & x_5 & - x_6 & - x_7 & & = -8 \\
x_1 & & & + x_4 & & - x_6 & & & = -1 \\
 & x_2 & & & & & & + x_8 & = 10 \\
 & & x_3 & & - x_5 & & & + x_8 & = 6 \\
 & & & x_4 & & & & & = 4 \\
x_1 & & & & & + x_6 & + x_7 & & = 14
\end{array}$$

These equations and variables can be re-numbered and blocked. Each block is solved in turn. In the case above, blocking allows the equations to be solved in 6 blocks as follows:

Block 1: Equation 7
$x_4 = 4$

Block 2: Equation 2
$x_7 = 7$

Block 3: Equations 4 and 8

$x_1 + x_4 - x_6 = -1$	From here	$x_1 = 1$
$x_1 + x_6 + x_7 = 14$	and:	$x_6 = 6$

Block 4: Equation 3

$x_5 - x_6 - x_7 = -8$	From here:	$x_5 = 5$

Block 5: Equations 1 and 6

$x_3 + x_8 = 11$	From here:	$x_3 = 3$
$x_3 - x_5 + x_8 = 6$	and:	$x_8 = 8$

Block 6: Equation 5:

$x_2 + x_8 = 10$	From here:	$x_2 = 2$

The first two blocks contain a single equation with a single variable. These blocks simply define constants. EES will recognize that equations that depend from the start on a single variable are in reality parameter or constant definitions. These parameters are determined before any solution of the remaining equations takes place. No lower and upper limits on the guesses are needed for parameters, since the values of these parameters are determined immediately. The solution of the remaining equations is now very simple, although it did not appear trivial at the beginning of the process.

Grouping of equations is useful when the equations are linear, but it is not essential. When the equations are nonlinear, grouping of equations is nearly indispensable, otherwise later groups of equations begin iterating with totally incorrect values of earlier variables. The result is often divergence. EES is able to recognize groups of equations prior to solution by inspecting the Jacobian matrix using the Tarjan [8] algorithm. See reference [6] for more details on this algorithm.

Determination of Minimum or Maximum Values

EES has the capability to find the minimum or maximum (i.e., optimum) value of a variable when there is one to ten degrees of freedom (i.e., the number of variables minus the number of equations). For problems with a single degree of freedom, EES can use either of two basic algorithms to find a minimum or maximum: a recursive quadratic approximation known as Brent's method or a Golden Section search [9]. The user specifies the method, the variable to be optimized and an independent variable whose value will be manipulated between specified lower and upper bounds. When there are two or more degrees of freedom, EES uses Brent's method repeatedly to determine the minimum or maximum along a particular direction. The direction is determined by a direct search algorithm known as Powell's method or by the conjugate gradient method [9,10].

The recursive quadratic approximation algorithm proceeds by determining the value of the variable which is to be optimized for three different values of the independent variable. A quadratic function is fit through these three points. Then the quadratic function is differentiated analytically to locate an estimate of the extremum point. If the relationship between the variable which is being optimized and the independent variable is truly quadratic, the optimum is found directly. If this is not the case, the algorithm will use the newly obtained estimate of the optimum point and two (of the three) points which are closest to it to repeat the quadratic fit. This process is continued until the convergence criteria set for the minimization/maximization process are satisfied.

The Golden Section search method is a region-elimination method in which the lower and upper bounds for the independent variable specified by the user are moved closer to each other with each iteration. The region between the bounds is broken into two sections, as shown in Figure 3. The value of the dependent variable is determined in each section. The bounds for the section which contains the smaller (for minimization) or larger (for maximization) dependent variable replace the interval bounds for the next iteration. Each iteration reduces the distance between the two bounds by a factor of (1-τ) where τ =0.61803 is known as the Golden Section ratio.

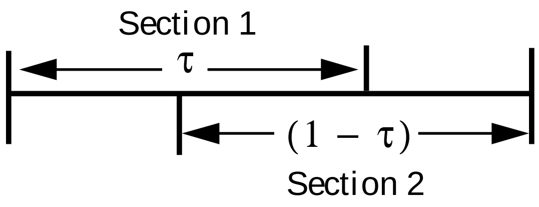


Figure 3: Region Elimination using the Golden Section Method

Numerical Integration

EES integrates functions and solves differential equations using a variant of the trapezoid rule along with a predictor-corrector algorithm. In explaining this method, it is helpful to compare the numerical scheme with the manner one would use to graphically determine the value of an integral.

Consider the problem of graphically estimating the integral of the function

$$f = 5 -5 X + 10 X^2$$

for X between 0 and 1. In graphical integration, a plot of f versus X would be prepared. The abscissa of the plot would be divided into a number of sections as shown below. The area under the curve in each section is estimated as the area of a rectangle with its base equal to the width of the section and its height equal to the average ordinate value in the section. For example, the ordinate values at 0 and 0.2 in the plot below are 5 and 4.4, respectively. The area of the first section is then 0.2 * (5+4.4)/2 or 0.94. The estimate integral value between 0 and 1 is the sum of the areas of the 5 sections. The accuracy of this method improves as the number of sections is increased.

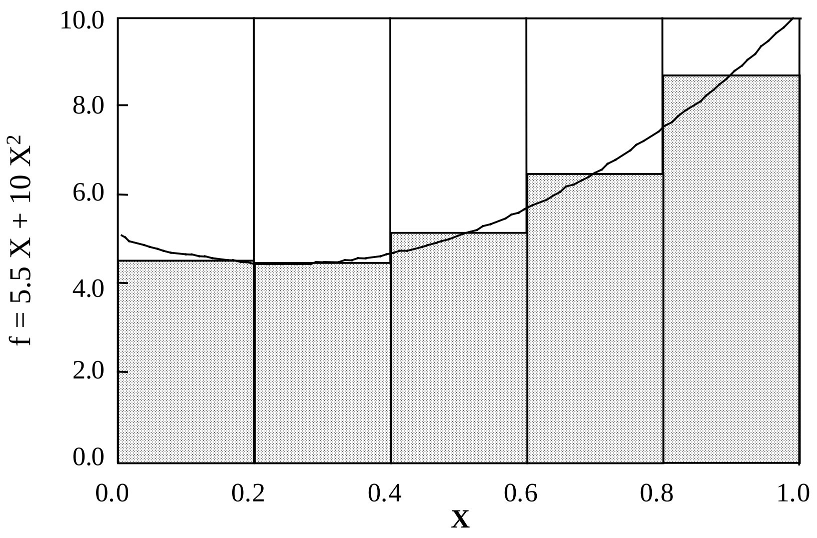


Figure 4: Numerical approximation of an integral

Integration in EES takes place in a manner quite analogous to the graphical representation. The abscissa variable, X, in the example above, is placed in the Parametric Table. The values of X entered into the table identify the width of each section. EES does not require each section to have the same width. The numerical value of the function, f, which is to be integrated, is evaluated at each value of X and supplied to EES through the Integrate function, e.g., Integral(f,X,0,1).

In some situations, such as in the solution of differential equations, the value of f may not be explicitly known at a particular value of X. The value of f may depend upon the solution to non-linear algebraic equations which have not yet converged. Further, the value of f may depend upon the value of the integral up to that point. In this case, iteration is needed. EES will repeatedly evaluate the section area using the latest estimate of f at the current value of X until convergence is obtained. The procedure in which the estimate of the integral made on the first calculation is corrected with later information is referred to as a predictor-corrector algorithm.

References

1. A. W. Al-Khafaji and J. R. Tooley, ***Numerical Methods in Engineering Practice***, Holt, Rinehart and Winston, 1986, pp. 190 & ff.

2. C. F. Gerald and P. O. Wheatley, ***Applied Numerical Analysis***, Addison-Wesley 1984, pp. 135 & ff.

3. J. H. Ferziger, ***Numerical Methods for Engineering Application***, Wiley-Interscience 1981, Appendix B.

4. F. S. Acton, ***Numerical Methods that Usually Work***, Harper and Row 1970.

5. I. S. Duff, A. M. Erisman and J. K. Reid, ***Direct Methods for Sparse Matrices***, 1986 Oxford Science Publications, Clarendon Press.

6. S. Pissanetsky, "Sparse Matrix Technology," Academic Press 1984.

7. F. L. Alvarado, "The Sparse Matrix Manipulation System," ***Report ECE-89-1***, Department of Electrical and Computer Engineering, The University of Wisconsin, Madison, Wisconsin, January 1989.

8. Tarjan, R. "Depth-First Search and Linear Graph Algorithms," ***SIAM J. Comput.*** 1, 146-160, (1972)

9. Powell's Method of Successive Quadratic Approximations, Reklaitis, Ravindran and Radsdell, ***Engineering Optimization***, John Wiley, New York, (1983)

10. W. H. Press, B. P. Flannery and S. A. Teukolsky, and Vetterling, W.T., ***Numerical Recipes in Pascal***, Cambridge University Press, Chapter 10, (1989)

Appendix C

Adding Property Data to EES

Background Information

EES uses an equation of state approach rather than internal tabular data to calculate the properties of fluids. For some substances and conditions, the ideal gas law is applicable. EES employs a naming convention to distinguish ideal gas and real fluid substances. Substances which are represented by their chemical symbol (e.g., N2) are modeled with the ideal gas law whereas substances for which the name is spelled out (e.g., Nitrogen) are considered to be real fluids. (Air and AirH2O are exceptions to this naming convention.)

Ideal gas substances rely on JANAF table data [Stull, 1971] to provide the enthalpy of formation and absolute entropy at a reference state of 298 K, 1 atm. Specific heat correlations for these gases and the ideal gas law are used to calculate the thermodynamic properties at conditions other than the reference state. A number of ideal gas substances are built into EES. The external JANAF program provides thermodynamic property information for hundreds of additional substances. Additional ideal gas fluid data can be added with .IDG files in the USERLIB folder, as explained below.

Real fluids properties are implemented with several different equations of state. Early versions of EES used the Martin-Hou [1955] equation of state (or variations of it) for all real fluids except water. The Martin-Hou property data base is still supported in EES. However, this equation of state is unable to provide accurate results for states near the critical point or at very high pressures. It is also unable to provide properties in the subcooled region. For this reason, a high accuracy equation of state has been implemented in the form of the Fundamental Equation of State (Tillner-Roth [1998]). The Fundamental Equation of State provides highly accurate properties in all regimes. In some cases, properties for a fluid, e.g., carbon dioxide, are implemented with both the Martin-Hou and the Fundamental Equation of State. In this case, the letters *ha are appended to fluid name, e.g.,* R134a_ha.

Several equations of state are provided for water, the most accurate and computationally intensive being the equation of state published by Harr, Gallagher, and Kell [1984]. Ice properties rely upon correlations developed by Hyland and Wexler [1983].

Thermodynamic property relations are used to determine enthalpy, internal energy and entropy values based upon the equation of state and additional correlations for liquid density, vapor pressure, and zero-pressure specific heat as a function of temperature. A modification to the

Martin-Hou equation of state proposed by Bivens et al. [1996] allows this equation equation of state to be applied for mixtures, such as the R400 refrigerant blends.

Viscosity and thermal conductivity of liquids and low-pressure gases are correlated with fluid specific relations. Many rely on polynomials in temperature. Temperature alone determines the transport properties for ideal gases. For real fluids, the effect of pressure on the gas transport properties is estimated using correlations from Reid et al. [1977] or included in the fluid specific relations. For example, the transport properties for fluid CarbonDioxide use the transport properties of Vesovic et al. [1990]. The source of all data implemented in EES can be viewed using the Fluid Info button of the Function Info dialog in the Options menu.

Adding Fluid Properties to EES

EES has been designed to allow additional fluids to be added to the property data base. Currently, it is possible to add property information for ideal gas fluids and for fluids represented by the Martin-Hou [1949] equation of state. Fluids represented by the Fundamental Equation of State cannot be added by the user.

To add property information, the user must supply the necessary parameters for the thermodynamic and transport property correlations. The parameters are placed in an ASCII text file which must be located in the EES\USERLIB subdirectory. EES will load all fluid files found in the EES\USERLIB subdirectory at startup. The additional fluids will appear in every way identical to the built-in fluids. The following sections describe the format required for the property data files.

Ideal Gas files

Ideal gas files must have a .IDG filename extension. An equation of state is not needed since it is assumed that the fluid obeys the ideal gas equation of state. However, particular attention must be paid to the reference states if the gas is to be used in calculations involving chemical reactions. The enthalpy of formation and Third-law entropy values at 298 K and 1 bar (or 1 atm) must be supplied. An example file providing the parameters for CO_2 is provided below. The properties of ideal gas fluid can be entered by adapting the file format to the new fluid.

SAMPLE TESTCO2.IDG File

```
TestCO2
44.01       {Molar mass of fluid
100.0       {Tn Normalizing value in K}
250         {Lower temperature limit of Cp correlation in K}
1500        {Upper temperature limit of Cp correlation in K}
-3.7357 0   {a0, b0  Cp=sum(a[i]*(T/Tn)^b[i], i=0,9 in kJ/kgmole-K}
30.529  0.5 {a1, b1}
-4.1034 1.0 {a2, b2}
0.02420 2.0 {a3, b3}
0 0         {a4, b4}
0 0         {a5, b5}
0 0         {a6, b6}
0 0         {a7, b7}
0 0         {a8, b8}
0 0         {a9, b9}
298.15      {TRef in K}
100         {Pref in kPa}
-393520     {hform - enthalpy of formation in kJ/kgmole at TRef}
213.685     {s0 - Third law entropy in kJ/kgmole-K at Tref and PRef}
0           {reserved - set to 0}
0           {reserved - set to 0}
200         {Lower temperature limit of gas phase viscosity correlation
in K}
```

```
1000         {Upper temperature limit of gas phase viscosity correlation
in K}
-8.09519E-7 {v0  Viscosity = sum(v[i]*T^(i-1)) for i=0 to 5 in Pa/m^2}
6.039533E-8 {v1}
-2.8249E-11 {v2}
9.84378E-15 {v3}
-1.4732E-18 {v4}
0           {v5}
200    {Lower  temperature  limit  of  gas  phase  thermal  conductivity
correlation in K}
1000  {Upper  temperature  limit  of  gas  phase  thermal  conductivity
correlation in K}
-1.1582E-3  {t0  Thermal Conductivity = sum(t[i]*T^(i-1)) for i=0 to 5
in W/m-K}
3.9174E-5   {t1}
8.2396E-8   {t2}
-5.3105E-11 {t3}
3.1368E-16  {t4}
0           {t5}
0           {Terminator - set to 0}
```

Real Fluid Files Represented by the Martin-Hou Equation of State

A pure real fluid is identified with a .MHE (for Martin-Hou Equation) filename extension. A sample file named XFLUID.MHE is listed on the following pages illustrating the required file format. (The sample file contains the parameters used for n-butane.) The file consists of 75 lines. The first line provides the name of the fluid which EES will recognize in the property function statements. For example, the first line in the sample file contains UserFluid. The enthalpy for this substance would then be obtained as follows.

h=**Enthalpy**(UserFluid,T=T1, P=P1)

The fluid name will appear in alphabetical order with other fluid names in the Function Information dialog window. The following 74 lines each contain one number. A comment follows on the same line (after one or more spaces) to identify the number. The forms of all of the correlations except the pressure-volume-temperature relation are indicated in the XFLUID.MHE file. Pressure, volume and temperature are related by the Martin-Hou equation of state in the following form. A method for obtaining the coefficients is described by Martin and Hou, [1955].

Martin-Hou Equation of State (parameters in lines 18-36)

$$P = \frac{R\,T}{v-b} + \frac{A_2 + B_2T + C_2e^{-\beta T/T_c}}{\left(v-b\right)^2} + \frac{A_3 + B_3T + C_3e^{-\beta T/T_c}}{\left(v-b\right)^3}$$
$$+ \frac{A_4 + B_4T + C_4e^{-\beta T/T_c}}{\left(v-b\right)^4} + \frac{A_5 + B_5T + C_5e^{-\beta T/T_c}}{\left(v-b\right)^5} + \frac{A_6 + B_6T + C_6e^{-\beta T/T_c}}{e^{\alpha v}(1 + C'e^{\alpha v})}$$

where

P [=] psia, T [=] R, and v [=] ft^3/lb$_m$

You may need to curve fit tabular property data or data obtained form a correlation in a different form to obtain the appropriate parameters. Most of the correlations are linear with respect to the parameters so that they can be determined by linear regression. A parameter set which improves upon the fit resulting from the Martin and Hou method can be determined by non-linear regression. EES can be used to do these regressions.

SAMPLE XFLUID.MHE File for pure fluids

```
UserFluid
58.1           { molecular weight}
0              { not used}
12.84149       { a}  Liquid
Density=a+b*Tz^(1/3)+c*Tz^(2/3)+d*Tz+e*Tz^(4/3)+f*sqrt(Tz)+g*(Tz)^2}
33.02582       { b}     where Tz=(1-T/Tc) and Liquid
Density[=]lbm/ft3
-2.53317       { c}
-0.07982       { d}
9.89109        { e}
0              { f}
0              { g}
-6481.15338    { a}   Vapor pressure fit: lnP=a/T+b+cT+d(1-
T/Tc)^1.5+eT^2
15.31880       { b}      where T[=]R and P[=]psia
-0.0006874     { c}
4.28739        { d}
0              { e}
0              { not used}
0.184697       { Gas constant in psia-ft3/lbm-R}
1.5259e-2      { b}   Constants for Martin-Hou EOS/English_units
-20.589        { A2}
9.6163e-3      { B2}
-314.538       { C2}
0.935527       { A3}
-3.4550e-4     { B3}
19.0974        { C3}
-1.9478e-2     { A4}
0              { B4}
0              { C4}
```

```
0              { A5}
2.9368e-7      { B5}
-5.1463e-3     { C5}
0              { A6}
0              { B6}
0              { C6}
5.475          { Beta}
0              { alpha}
0              { C'}
-7.39053E-3    { a}    Cv(0 pressure) = a + b T + c T^2 + d T^3 +
e/T^2
6.4925e-4      { b}            where T[=]R and Cv[=]Btu/lb-R
9.0466e-8      { c}
-1.1273e-10    { d}
5.2005e3       { e}
124.19551      { href offset}
0.0956305      { sref offset}
550.6          { Pc [=] psia}
765.3          { Tc [=] R}
0.07064        { vc [=] ft3/lbm}
0              { not used}
0              { not used}
2              { Viscosity correlation type: set to 2: do not change}
260            { Lower limit of gas viscosity correlation in K}
535            { Upper limit of gas viscosity correlation in K}
-3.790619e6    { A}     GasViscosity*1E12=A+B*T+C*T^2+D*T^3
5.42356586e4   { B}     where T[=]K and GasViscosity[=]N-s/m2
-7.09216279e1  { C}
5.33070354e-2  { D}
115            { Lower limit of liquid viscosity correlation in K}
235            { Upper limit of liquid viscosity correlation in K}
2.79677345e3   { A}    Liquid Viscosity*1E6=A+B*T+C*T^2+D*T^3
-2.05162697e1  { B}     where T[=]K and Liquid Viscosity[=]N-s/m2
5.3698529e-2   { C}
-4.88512807e-5 { D}
2              { Conductivity correlation type: set to 2: do not
change}
250            { Lower limit of gas conductivity correlation in K}
535            { Upper limit of gas conductivity correlation in K}
7.5931e-3      { A}    GasConductivity=A+B*T+C*T^2+D*T^3
-6.3846e-5     { B}     where T[=]K and GasConductivity[=]W/m-K
3.95367e-7     { C}
-2.9508e-10    { D}
115            { Lower limit of liquid conductivity correlation in K}
235            { Upper limit of liquid conductivity correlation in K}
2.776919161e-1 { A}    LiquidConductivity=A+B*T+C*T^2+D*T^3
-8.45278149e-4 { B}     where T[=]K and LiquidConductivity[=]W/m-K
1.57860101e-6  { C}
-1.8381151e-9  { D}
0              { not used: terminator}
```

Fluid Properties for Blends

The Martin-Hou equation of state can be adapted for mixtures as proposed by Bivens et. al. The major modifications needed to make this pure component equation of state applicable to blends is to provide separate correlations for the bubble and dew point vapor pressures and a correlation for the enthalpy of vaporization, since the equation of state can not provide this information. Shown below is a listing of the R410A.MHE file that is used to provide property data for R410A, along with an explanation of each line in the file.

```
R410A
72.584         {molecular weight Bivens and Yokozeki}
400            {Indicator for blend}
30.5148        {a} Liquid density = a+b*Tz^(1/3)+c*Tz^(2/3)+d*Tz
60.5637        {b}       +e*Tz^(4/3)+f*sqrt(Tz)+g*(Tz)^2}
-5.39377       {c} where Tz=(1-T/Tc) and Liquid Density[=]lbm/ft3
55.5360815     {d}
-21.88425      {e}
0              {f}
0              {g}
-5.9789E+03  -5.9940E+03  {a} Bubble and Dew Pt Vapor pressure fit:
24.06932  24.04507        {b}     lnP=a/T+b+cT+d(1-T/Tc)^1.5+eT^2
-2.1192E-02  -2.1084E-02  {c}     where T[=]R and P[=]psia fit
-5.5841E-01  -4.4382E-01  {d}
1.3718E-05  1.3668E-05    {e}
0  0                      {not used}
0.1478         {Gas constant in psia-ft3/lbm-R}
0.006976       {b}  Constants for Martin-Hou EOS/English_units from Bivens
-6.40764E+00   {A2}
3.40372E-03    {B2}
-2.34220E+02   {C2}
1.41972E-01    {A3}
4.84456E-06    {B3}
9.13546E+00    {C3}
-4.13400E-03   {A4}
0              {B4}
0              {C4}
-9.54645E-05   {A5}
1.17310E-07    {B5}
2.45370E-02    {C5}
0              {A6}
0              {B6}
0              {C6}
5.75           {Beta}
0              {alpha}
0              {C'}
0.036582       {a}   Cv(0 pressure) = a + b T + c T^2 + d T^3 + e/T^2
```

```
2.808787E-4    {b}           where T[=]R and Cv[=]Btu/lb-R from Bivens
-7.264730E-8   {c}
2.6612670E-12  {d}
0              {e}
65.831547      {href offset}
-0.082942      {sref offset}
714.5          {Pc [=] psia}
621.5          {Tc [=] R}
0.03276        {vc [=] ft3/lbm}
0              {not used}
7              {# of coefficients which follow - used for blends}
1              {DeltaH Correlation type}
0.5541498      {Xo}
87.50197       {A} DeltaH_vap=A+B*X+C*X^2+D*X^3+E*X^4 Bivens
185.3407       {B}   where X =(1-T/Tc)^.333-X0, T in R and enthalpy in Btu/lb
13.75282       {C}
0              {D}
0              {E}
2              {Viscosity correlation type: set to 2: do not change}
200            {Lower limit of gas viscosity correlation in K}
500            {Upper limit of gas viscosity correlation in K}
-1.300419E6    {A}    GasViscosity*1E12=A+B*T+C*T^2+D*T^3
5.39552e4      {B}    where T[=]K and GasViscosity[=]N-s/m2
-1.550729e1    {C}
0              {D}
-999           {Lower limit of liquid viscosity correlation in K}
-999           {Upper limit of liquid viscosity correlation in K}
0              {A}   Liquid Viscosity*1E6=A+B*T+C*T^2+D*T^3
0              {B}    where T[=]K and Liquid Viscosity[=]N-s/m2
0              {C}
0              {D}
2              {Conductivity correlation type: set to 2: do not change}
200            {Lower limit of gas conductivity correlation in K}
500            {Upper limit of gas conductivity correlation in K}
-8.643088e-3   {A}   GasConductivity=A+B*T+C*T^2+D*T^3
7.652083e-5    {B}    where T[=]K and GasConductivity[=]W/m-K
2.144608e-9    {C}
0              {D}
-999           {Lower limit of liquid conductivity correlation in K}
-999           {Upper limit of liquid conductivity correlation in K}
0              {A}   LiquidConductivity=A+B*T+C*T^2+D*T^3
0              {B}    where T[=]K and LiquidConductivity[=]W/m-K
0              {C}
0              {D}
0 {terminator}
{The forms of the correlations and in some cases the coefficients have been
adapted from D.B. Bivens and A. Yokozeki, "Thermodynamics and Performance
Potential of R-410a," 1996 Intl. Conference on Ozone Protection Technologies
Oct, 21-23, Washington, DC.}
```

References

ASHRAE Handbook of Fundamentals, (1989, 1993, 1997), American Society of Heating, Refrigerating and Air Conditioning Engineers, Atlanta, GA

ASHRAE, **Thermophysical Properties of Refrigerants**, American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Atlanta, GA, (1976)

D.B. Bivens and A. Yokozeki, "Thermodynamics and Performance Potential of R-410a," 1996 Intl. Conference on Ozone Protection Technologies Oct, 21-23, Washington, DC.

Downing, R.C. and Knight, B.W., "Computer Program for Calculating Properties for the "FREON" Refrigerants," DuPont Technical Bulletin RT-52, (1971); Downing, R.C., "Refrigerant Equations", ASHRAE Transactions, Paper No. 2313, Vol. 80, pt.2, pp. 158-169, (1974)

Gallagher, J., McLinden, M, Morrison, G., and Huber, M., REFPROP - NIST Thermodynamic Properties of Refrigerants and Refrigerant Mixtures, Versions 4, 5, and 6, NIST Standard Reference Database 23, NIST, Gaithersburg. MD 20899, (1989)

Harr, L. Gallagher, J.S., and Kell, G.S (Hemisphere, 1984). **NBS/NRC Steam Tables**, Hemisphere Publishing Company, Washington, (1984)

Howell, J.R., and Buckius, R.O., **Fundamentals of Engineering Thermodynamics**, McGraw-Hill, New York, (1987)

Hyland and Wexler, "Formulations for the Thermodynamic Properties of the Saturated Phases of H_2O from 173.15 K to 473.15 K, ASHRAE Transactions, Part 2A,Paper 2793 (RP-216), (1983)

Keenan, J.H., Chao, J., and Kaye, J., **Gas Tables**, Second Edition, John Wiley, New York, (1980)

Keenan, J.H. et al., **Steam Tables**, John Wiley, New York, (1969)

Irvine, T.F. Jr., and Liley, P.E., **Steam and Gas Tables with Computer Equations**, Academic Press Inc., (1984)

Martin, J.J. and Hou, Y.C., "Development of an Equation of State for Gases," A.I.Ch.E Journal, 1:142, (1955)

McLinden, M.O. et al., "Measurement and Formulation of the Thermodynamic Properties of Refrigerants 134a and 123, ASHRAE Trans., Vol. 95, No. 2, (1989)

Reid, R.C.Prausnitz, J.M. and Sherwood, T.K., **The Properties of Gases and Liquids**, McGraw-Hill, 3rd edition, (1977)

Shankland, I.R., Basu, R.S., and Wilson, D.P., "Thermal Conductivity and Viscosity of a New Stratospherically Sate Refrigerant - 1,1,1,2 Tetrafluoroethane (R-134a), published in **CFCs:**

Time of Transition, American Society of Heating, Refrigeration and Air-Conditioning Engineers, Inc., (1989)

Shankland, I.R., "Transport Properties of CFC Alternatives", AIChE Spring Meeting, Symposium on Global Climate Change and Refrigerant Properties, Orlando, FL, March, (1990)

Stull, D.R., and Prophet, H., **JANAF Thermochemical Tables**, Second Edition, U.S. National Bureau of Standards, Washington, (1971)

Reiner Tillner-Roth, "Fundamental Equations of State", Shaker, Verlag, Aachan, 1998.

Van Wylen, G.J., and Sonntag, R.E., **Fundamentals of Classical Thermodynamics**, Third Edition, John Wiley, New York, (1986)

Vesovic et al., The Transport Properties of Carbon Dioxide, J. Phys. Chem Ref, Data, Vol. 19, No. 3, 1990.

Wilson, D.P. and Basu, R.S., "Thermodynamic Properties of a New Stratospherically Safe Working Fluid - Refrigerant 134a", paper presented at the ASHRAE meeting, Ottawa, Ontario, Canada, June, (1988), published in **CFCs: Time of Transition**, American Society of Heating, Refrigeration and Air-Conditioning Engineers, Inc., (1989)

Example Problem Information

The EXAMPLES subdirectory within the EES directory contains many worked-out example problems. Each example problem illustrates one or more EES features, as indicated in the information below.

Feature	**EES Example Files**
Arrays	MATRIX.EES, MATRIX2.EES, RANKINE.EES, REFRIG.EES, REGEN.EES
Complex numbers	COMPLEXROOTS.EES
Comments	HEATEX.EES
Curve-fitting	COPPER.EES
Diagram window	DIAGRMW.EES, DIAGRAM_IN_OUT.EES, STMPROPS.EES
Differential equations	DRAG.EES, RK4_TEST.EES, DIFEQN1.EES, DIFEQN2.EES
Differentiate function	COPPER.EES
DUPLICATE command	MATRIX.EES, MATRIX2.EES, NLINRG.EES, REGEN.EES
Formatted Equations	HEATEX.EES, DRAG.EES
Functions, user-written	CONVECT.EES, MOODY.EES, RK4_TEST.EES
Greek symbols	HEATEX.EES, NLINRG.EES
Integration	DBL_INTEG.EES, DIFEQN1.EES, DIFEQN2.EES, DRAG.EES, RK4_TEST.EES
Interpolate function	COPPER.EES
JANAF table	FLAMET.EES, JANAF.EES
LOOKUP table	NLINRG.EES, COPPER.EES
Minimize or maximize	MAXPOWER.EES, NLINRG.EES, RANKINE.EES
Modules	MOODY.EES
Overlay Plot	CH1EX.EES RANKINE.EES
Parametric table	CAPVST.EES, CH1EX.EES, DIFEQN1.EES, FLAMET.EES, SUBSTEPS.EES
Plotting	CAPVST.EES, DIFEQN2
Procedures, user-written	REGEN.EES
Procedures, external	ABSORP.EES
Properties, thermodynamic	REFRIG.EES, CATVST.EES, CH1EX.EES, FLAMET.EES, REGEN.EES,STMPROPS.EES
Property Plot	RANKINE.EES, REFRIG.EES
Psychrometric functions	SUPERMKT.EES
Regression	NLINRG.EES
Subscripted variables	MATRIX.EES, MATRIX2.EES, HEATEX.EES
SUM function	MATRIX.EES, MATRIX2.EES, NLINRG.EES
Systems of equations	HEATEX.EES, CH1EX.EES
TABLEVALUE	DIFEQN2.EES
Transport properties	CONVECT.EES
Unit conversion	DRAG.EES