EES

Engineering Equation Solver

for Microsoft Windows
Operating Systems*

Commercial and Professional Versions

F-Chart Software
http://fchart.com/
email: info@fchart.com

***EES is designed to operate in the 32 and 64-bit Windows XP/Vista/7/8/10 operating systems**

The author make no guarantee that the program is free from errors or that the results produced with it will be free of errors and assume no responsibility or liability for the accuracy of the program or for the results that may come from its use.

EES was compiled with DELPHI 10 Seattle by Embarcadero

The three-dimensional plotting package is based on a modification of the public domain GLScene package (http://www.glscene.org/).

The genetic optimization method implemented in EES is derived from the public domain Pikaia optimization program (version 1.2, April 2002) written by Paul Charbonneau and Barry Knapp at National Center for Atmospheric Research (NCAR) (http://www.hao.ucar.edu/public/research/si/pikaia/pikaia.html).

Registration Number__________________________

ALL CORRESPONDENCE MUST INCLUDE THE REGISTRATION NUMBER

V10.412

Table of Contents

Table of Contents .... ***iii***

Overview ***1***

Chapter 1: Getting Started ***5***

Installing EES on your Computer 5

Starting EES 5

Background Information 6

An Example Thermodynamics Problem 9

Chapter 2: EES Windows ***23***

General Information 23

Equations Window 25

Formatted Equations Window 29

Solution Window 32

Arrays Window 34

Computational Flow Window 36

Residuals Window 37

Parametric Table Window 39

Integral Table Window 49

Diagram Window 50

Development and Application Modes 51

Moving the Diagram 52

Resizing the Diagram 52

Adding and Moving Text on the Diagram Window 53

Adding Graphic Items 59

Selecting, Modifying and Aligning Text and Graphic Items 61

Group and Ungroup Buttons 62

Adding a Calculate Button 62

Adding a Plot Window Access Button 62

Creating Hot Areas and Child Diagram Windows *(Professional Version only)* 64

Saving User Inputs 64

Creating Links (*Professional Version*) 65

Adding a Help Button 67

Navigating through Child Diagram Window (*Professional Version*) 68

Adding a Print Button 68

Adding an Audio-Visual Item 68

Saving and Loading User Inputs (*Professional Version*) 69

Creating and Using Check Boxes (*Professional Version*) 70

Creating and Using Radio Groups (*Professional Version*) 72

Animation (*Professional Version*) 73
Plot Windows 77
Moving the Plot 77
Adding / Changing Text and Text Characteristics 78
Moving Text 80
Adding Lines and Arrows, Rectangles and Circles 81
Moving Lines and Arrows, Rectangles and Circles 82
Resizing the Plot 83
Modifying the Axis Information 83
Aligning Items on the Plot Window 87
Crosshairs 87
3-D Plot Controls 88
Report Window (Professional Version) 92
Debug Window 94
Incorrect Degrees of Freedom 94
Constrained Solution 95
Unit Checking Report 96
Macro Window 97
Calculator Window 97

Chapter 3: Menu Commands 99
The File Menu 99
The Edit Menu 110
The Search Menu 114
The Options Menu 115
The Calculate Menu 136
The Tables Menu 145
The Plot Menu 154
The Windows Menu 167
The Help Menu 169
The Textbook Menu 171

Chapter 4: Built-in Functions 173
Mathematical Functions 173
CurveFit1D Procedure 189
String Functions 190
Thermophysical Property Functions 194
Incompressible Substances Library 209
Using Lookup Files and Lookup Tables 210
Binary Lookup files (.LKT) 210
ASCII Lookup files (.TXT) 211
ASCII Lookup files (.CSV) and Format Specification Files. (.FMT) 212

ASCII Lookup files (.DAT) 213
Lookup files in EXCEL format (.XLSX) 213

Chapter 5: Functions, Procedures, Modules, and Subprograms 220
EES Functions 221
EES Procedures 224
Single-Line If Then Else Statements 226
Multiple-Line If Then Else Statements 227
GoTo Statements 228
Return Statement 228
Repeat Until Statements 228
CASE Statements 229
Error Procedure 230
Warning Procedure 232
Print Command in Functions and Procedures 233
Modules and SubPrograms 234
Library Files 238

Chapter 6: External Functions and Procedures 241
EES External Functions (.DLF files) 241
The PWF Function – a .DLF external routine in written in DELPHI 243
SUM_C – a .DLF external function written in Microsoft's Visual C++ 247
EES External Procedures (.FDL and .DLP files) 248
External Procedures with the .FDL Format - a FORTRAN Example 249
External Procedures with the .DLP Format - a Pascal Example 251
External Procedures with the .DLP Format - a C++ Example 254
Multiple Files in a Single Dynamic Link Library (.DLL) 256
Help for External Functions and Procedures 260
Library Manager (Professional Version) 261

Chapter 7: Advanced Features 262
String Variables 262
Complex Variables 263
Array Variables 267
The DUPLICATE Command 269
Matrix Capabilities 270
Using the Property Plot 272
Integration and Differential Equations 273
Method 1: Solving Differential Equations with the Table-Based Integral Function 275
Method 2: Use of the $Integral Directive and the Equation-Based Integral Function 279
Method 3: Solving Differential Equations with the TableValue Function 280
Directives 286
$ARRAYS On/Off Directive 286

$CHECKUNITS On/Off/AutoOn/AutoOff 286
$COMMON 287
$COMPLEX On/Off i/j 288
$CONSTANT Name# = ValueSI [UnitsSI] ValueEng [UnitsEng] 288
$COPYTOLOOKUP Directive 289
$DOLAST and $ENDDOLAST Directives 290
$EXPORT Directive 290
$EXPORTPLOT Directive 292
$EXPORTTEXT Directive 292
$HIDEWINDOW Directive 293
$IF/IFNOT Directives 293
$IMPORT Directive 295
$IMPORTTEXT Directive 296
$INCLUDE Directive 296
$INTEGRALAUTOSTEP Directive 297
$INTEGRALTABLE Directive 298
$LOCALVARIABLES On/Off Directive 299
$OPENLOOKUP Directive 300
$PRIVATE Directive 301
$REAL Directive 301
$REFERENCE Directive 302
$REQUIREDOUTPUTS Directive 302
$SAVELOOKUP Directive 303
$SAVETABLE 304
$STOPCRITERIA 305
$SHOWWINDOW WindowName1 WindowName2 305
$SUMROW On/Off Directive 305
$TABWIDTH Directive 306
$TRACE Directive 306
$UNITSYSTEM Directive 306
$VARINFO Directive 307
$WARNINGS On/Off Directive 308
Creating and Using Macro Files (Professional Version) 309
Unit Lists 330
Creating an Application Library 332
EES_App Mobile Application .... **Error! Bookmark not defined.**
EESws - A Web Server for EES Distributable Programs .... **Error! Bookmark not defined.**

Appendix A: Hints for Using EES 335

Appendix B: Numerical Methods Used in EES 338
Solution to Algebraic Equations 338
Blocking Equation Sets 341

Determination of Minimum or Maximum Values 343
Numerical Integration 344
References 346

Appendix C: Adding Property Data to EES 348
Background Information 348
Adding Fluid Properties to EES 349
SAMPLE TESTCO2.IDG File 349
SAMPLE XFLUID.MHE File for pure fluids 351

Appendix D: The 64-Bit Professional Version 357

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, generate publication-quality plots, simplify uncertainty analyses and provide animations. EES has been developed to run under the 32 and 64-bit Microsoft Windows operating systems, i.e., XP and Windows 7/8/ and 10. It can be run in Linux and on the Macintosh using emulation programs.

There are two major differences between EES and existing numerical equation-solving programs. First, EES automatically identifies and groups equations that 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 Procedures similar to those in Pascal and FORTRAN. EES also provides support for user-written routines, which are self-contained EES programs that can be accessed by other EES programs. The Functions, Procedures, Subprograms and Modules can be saved as library files which are automatically read in when EES is started. Third, external 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.

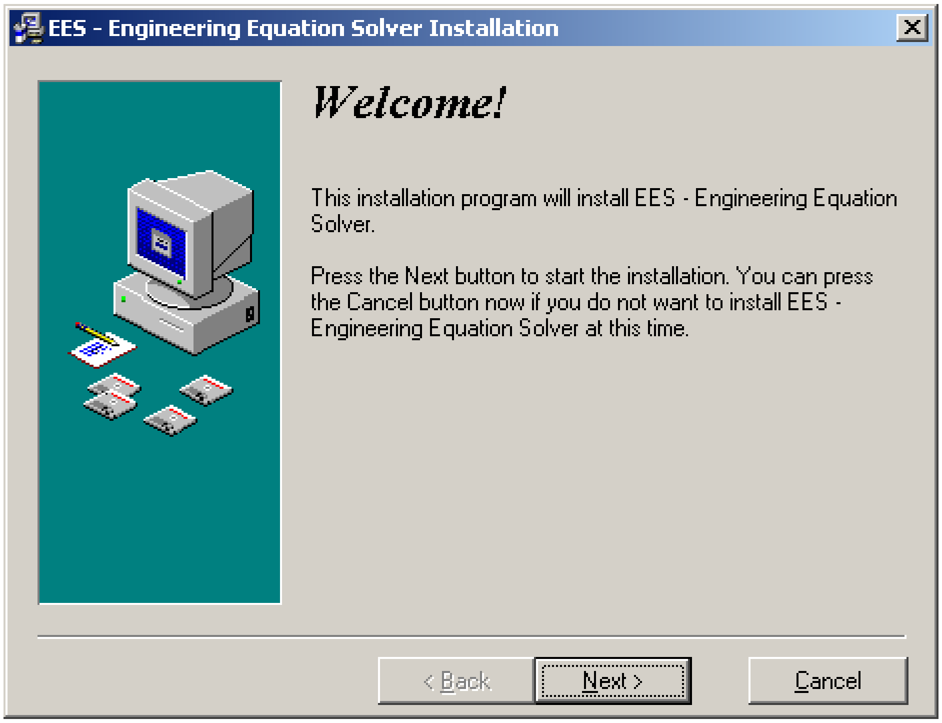
EES offers the advantages of a simple set of intuitive commands that a novice can quickly learn to use for solving any algebraic problems. However, the capabilities of this program are extensive and useful to an expert as well. The large data bank of thermodynamic and transport properties built into 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 three 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. The animation capabilities provided in the Diagram window are described in this chapter. 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, Subprograms and Modules and saving them in Library files. Chapter 6 describes how external 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. The use of directives and macros is also explained. 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.

Chapter 1: Getting Started

Installing EES on your Computer

Non-Academic versions of EES are distributed in a self-installing compressed form in a file called SETUP_EES.exe. To install EES, it is necessary execute the SETUP_EES installation program.



EES - Engineering Equation Solver Installation

Welcome!

This installation program will install EES - Engineering Equation Solver.

Press the Next button to start the installation. You can press the Cancel button now if you do not want to install EES - Engineering Equation Solver at this time.

The installation program will provide a series of prompts which will lead you through the complete installation of the EES program. The default folder for the EES files is C:\EES32.

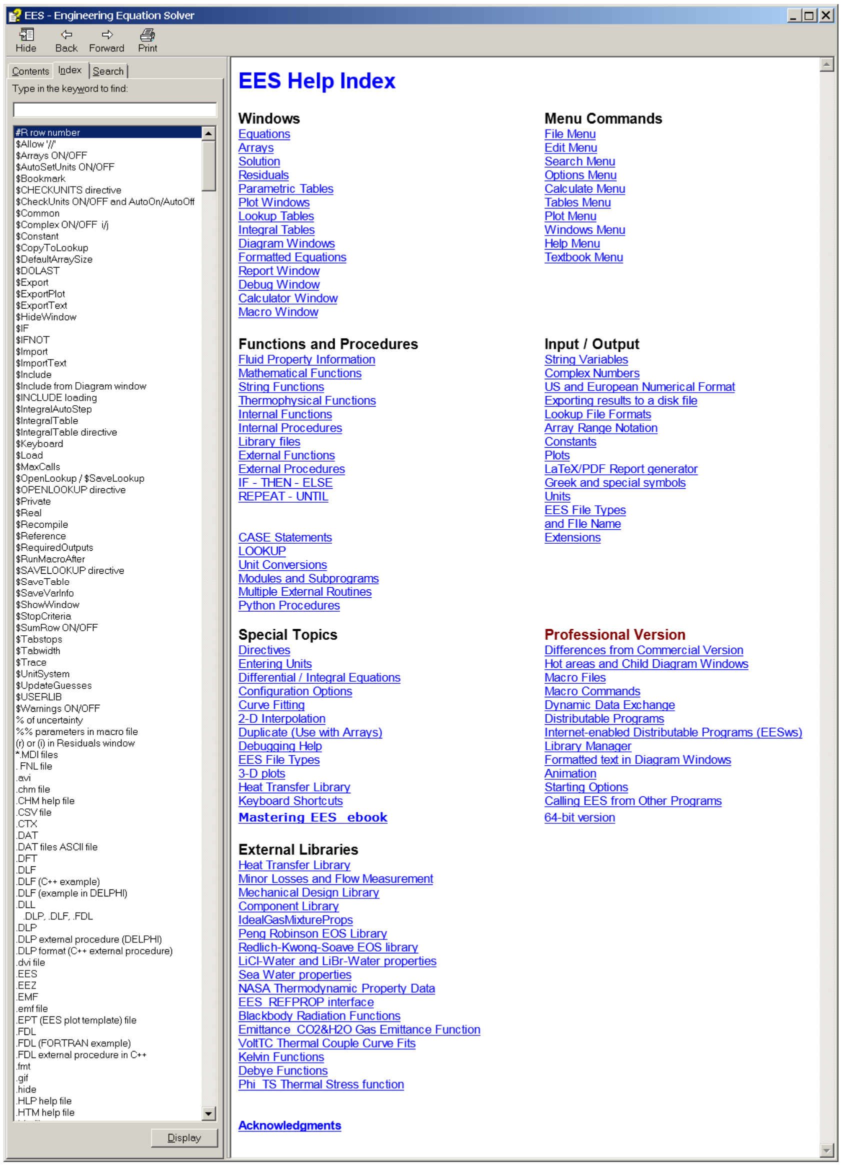
Starting EES

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.

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. If this is your first time running EES, you may wish to specify the default unit system and other default settings. Clicking the Preferences button will take you to the Preferences dialog where the options can be changed and stored. Alternatively, click the Continue button to dismiss the dialog window and start with an empty EES documents.

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 on an underlined word will provide help relating to that subject. The Search tab allows searching for a particular keyword or feature.


EES - Engineering Equation Solver
Hide
Back
Forward
Print
Contents
Index
Search
Type in the keyword to find:
Display

#R row number
$Allow '//'
$Arrays ON/OFF
$AutoSetUnits ON/OFF
$Bookmark
$CHECKUNITS directive
$CheckUnits ON/OFF and AutoOn/AutoOff
$Common
$Complex ON/OFF i/j
$Constant
$CopyToLookup
$DefaultArraySize
$DOLAST
$Export
$ExportPlot
$ExportText
$HideWindow
$IF
$IFNOT
$Import
$ImportText
$Include
$Include from Diagram window
$INCLUDE loading
$IntegralAutoStep
$IntegralTable
$IntegralTable directive
$Keyboard
$Load
$MaxCalls
$OpenLookup / $SaveLookup
$OPENLOOKUP directive
$Private
$Real
$Recompile
$Reference
$RequiredOutputs
$RunMacroAfter
$SAVELOOKUP directive
$SaveTable
$SaveVarInfo
$ShowWindow
$StopCriteria
$SumRow ON/OFF
$Tabstops
$Tabwidth
$Trace
$UnitSystem
$UpdateGuesses
$USERLIB
$Warnings ON/OFF
% of uncertainty
%% parameters in macro file
(r) or (i) in Residuals window
*.MDI files
. FNL file
.avi
.chm file
.CHM help file
.CSV file
.CTX
.DAT
.DAT files ASCII file
.DFT
.DLF
.DLF (C++ example)
.DLF (example in DELPHI)
.DLL
.DLP, .DLF, .FDL
.DLP
.DLP external procedure (DELPHI)
.DLP format (C++ external procedure)
.dvi file
.EES
.EEZ
.EMF
.emf file
.EPT (EES plot template) file
.FDL
.FDL (FORTRAN example)
.FDL external procedure in C++
.fmt
.gif
.hide
.HLP help file
.HTM help file

EES Help Index

Windows
Equations
Arrays
Solution
Residuals
Parametric Tables
Plot Windows
Lookup Tables
Integral Tables
Diagram Windows
Formatted Equations
Report Window
Debug Window
Calculator Window
Macro 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
Fluid Property Information
Mathematical Functions
String Functions
Thermophysical Functions
Internal Functions
Internal Procedures
Library files
External Functions
External Procedures
IF - THEN - ELSE
REPEAT - UNTIL

CASE Statements
LOOKUP
Unit Conversions
Modules and Subprograms
Multiple External Routines
Python Procedures

Input / Output
String Variables
Complex Numbers
US and European Numerical Format
Exporting results to a disk file
Lookup File Formats
Array Range Notation
Constants
Plots
LaTeX/PDF Report generator
Greek and special symbols
Units
EES File Types
and FIle Name
Extensions

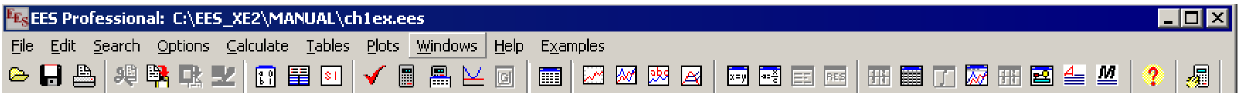
Special Topics
Directives
Entering Units
Differential / Integral Equations
Configuration Options
Curve Fitting
2-D Interpolation
Duplicate (Use with Arrays)
Debugging Help
EES File Types
3-D plots
Heat Transfer Library
Keyboard Shortcuts
Mastering EES ebook

Professional Version
Differences from Commercial Version
Hot areas and Child Diagram Windows
Macro Files
Macro Commands
Dynamic Data Exchange
Distributable Programs
Internet-enabled Distributable Programs (EESws)
Library Manager
Formatted text in Diagram Windows
Animation
Starting Options
Calling EES from Other Programs
64-bit version

External Libraries
Heat Transfer Library
Minor Losses and Flow Measurement
Mechanical Design Library
Component Library
IdealGasMixtureProps
Peng Robinson EOS Library
Redlich-Kwong-Soave EOS library
LiCl-Water and LiBr-Water properties
Sea Water properties
NASA Thermodynamic Property Data
EES REFPROP interface
Blackbody Radiation Functions
Emittance CO2&H2O Gas Emittance Function
VoltTC Thermal Couple Curve Fits
Kelvin Functions
Debye Functions
Phi TS Thermal Stress function

Acknowledgments

EES commands are distributed among nine pull-down menus. (A tenth user-defined menu, used below for Examples, can be placed by the user 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.


EES Professional: C:\EES_XE2\MANUAL\ch1ex.ees
File Edit Search Options Calculate Tables Plots Windows Help Examples

Note that 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 seconds, 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 or Diagram 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.

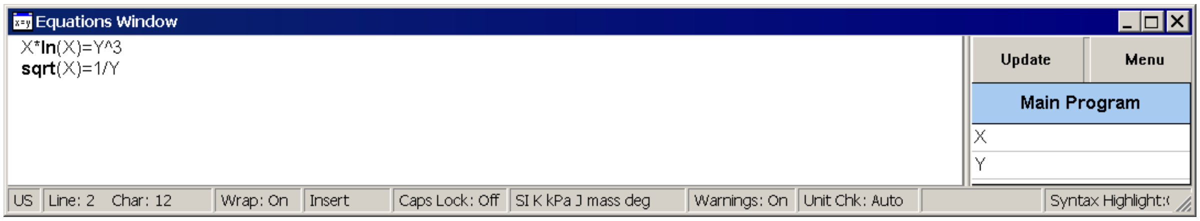
The Tables menu contains commands to set up and alter the contents of the Parametric and Lookup Tables and to do linear regression on the data in these tables. The Parametric Table, similar to a spreadsheet, allows the equation set to be solved repeatedly while varying the values of one or more variables.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

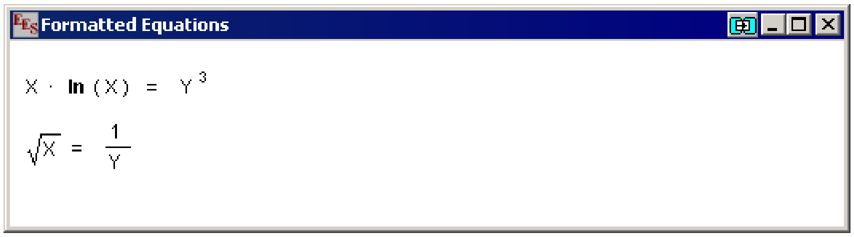
The basic capability provided by EES is the solution of a set of non-linear algebraic equations. To demonstrate this capability, start EES and enter this simple example problem in the Equations window. Note that EES makes no distinction between upper and lower case letters and the ^ sign (or **) is used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



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

```
US | Line: 2   Char: 12 | Wrap: On | Insert | Caps Lock: Off | SI K kPa J mass deg | Warnings: On | Unit Chk: Auto | | Syntax Highlight:
```

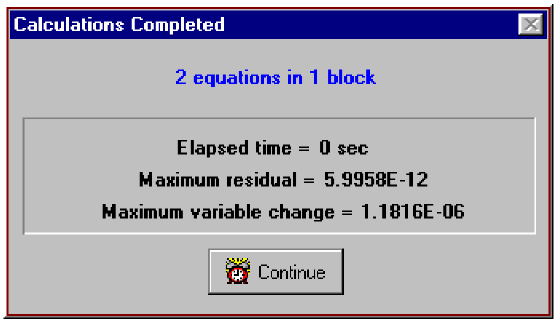
If you wish, you may view the equations in mathematical notation by selecting the Formatted Equations command from the Windows menu.



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

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

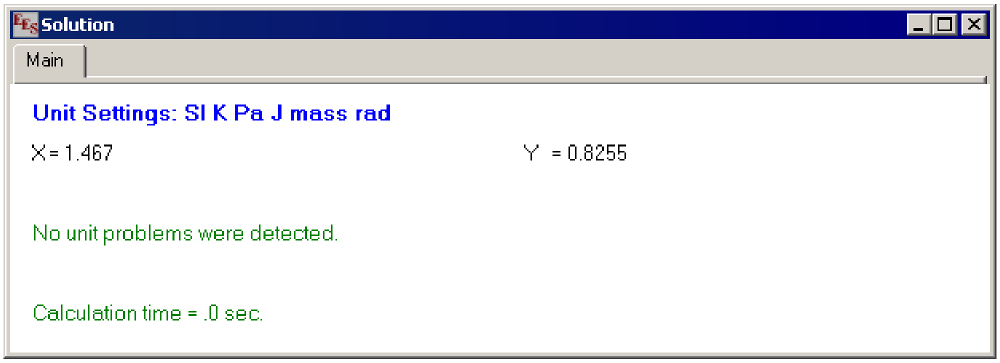
Select the Solve command from the Calculate menu. A dialog window will appear indicating the progress of the solution. When the calculations are completed, the button changes from Abort to Continue.



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.



Unit Settings: SI K Pa J mass rad

X= 1.467 Y = 0.8255

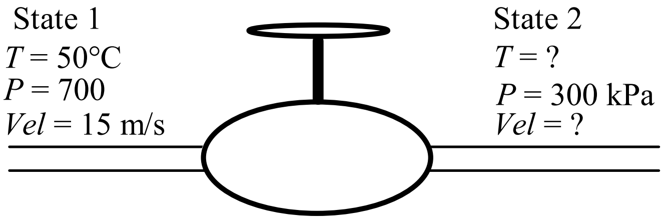
No unit problems were detected.

Calculation time = .0 sec.

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, EES can manually or automatically check that all unit conversions have been made and the units in each equation are dimensionally consistent. The Check Units command (Calculate menu) initiates the unit checking capability manually.

From relationships between the properties of R134a:

$$v_1 = v(T_1, P_1) \tag{6}$$

$$h_1 = h(T_1, P_1) \tag{7}$$

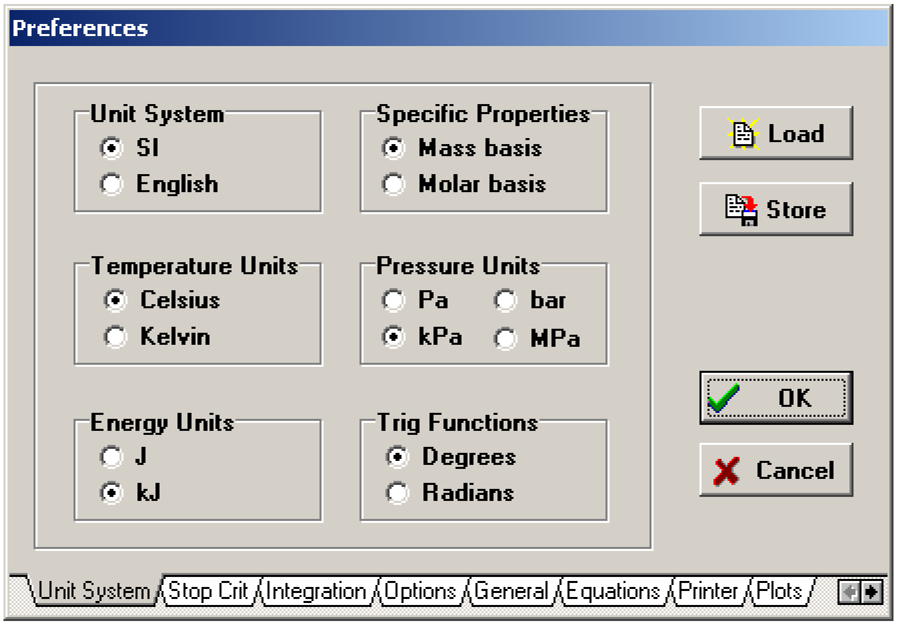
$$v_2 = v(T_2, P_2) \tag{8}$$

$$h_2 = h(T_2, P_2) \tag{9}$$

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

Specific Properties: ● Mass basis ○ Molar basis

Temperature Units: ● Celsius ○ Kelvin

Pressure Units: ○ Pa ○ bar ● kPa ○ MPa

Energy Units: ○ J ● kJ

Trig Functions: ● Degrees ○ Radians

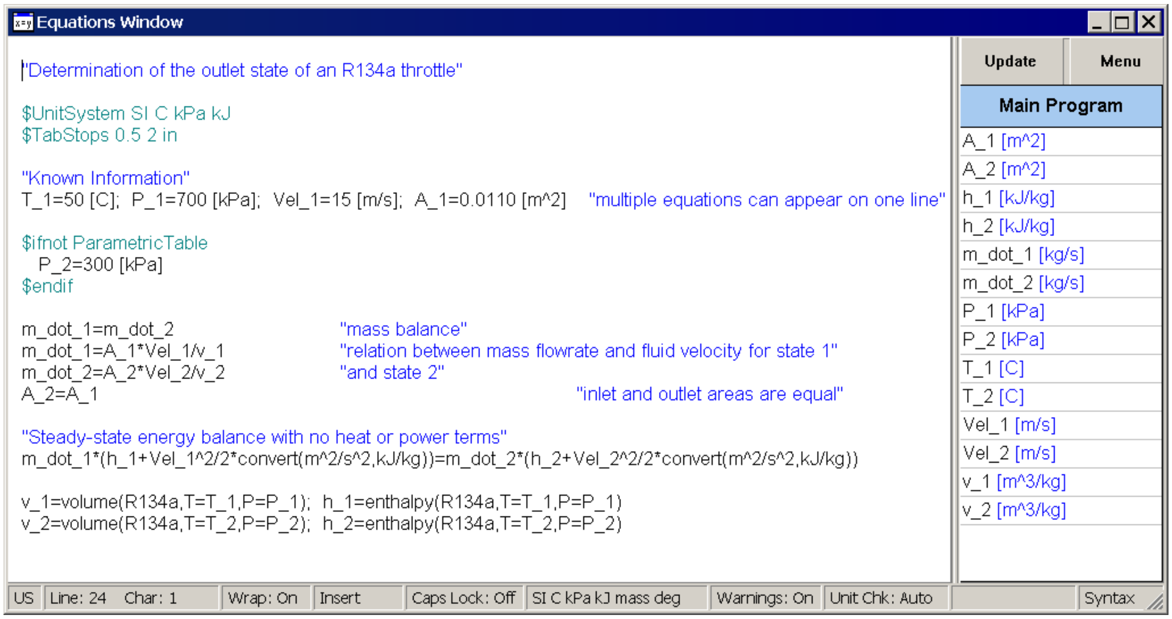
EES is initially configured to be in SI units with T in °C, P in kPa, energy units in kJ and specific property values in their customary units on a mass basis. These defaults may have been changed during a previous use. Click on the controls to set the units as shown above. Click the OK button (or press 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].
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.
9. Units for constants can be entered in braces directly following the comment, e.g.
   g = 9.82 [m/s^2]
10. Underscore characters can be used to produce subscripts on the formatted output. Greek symbol names will be replaced by the Greek characters.

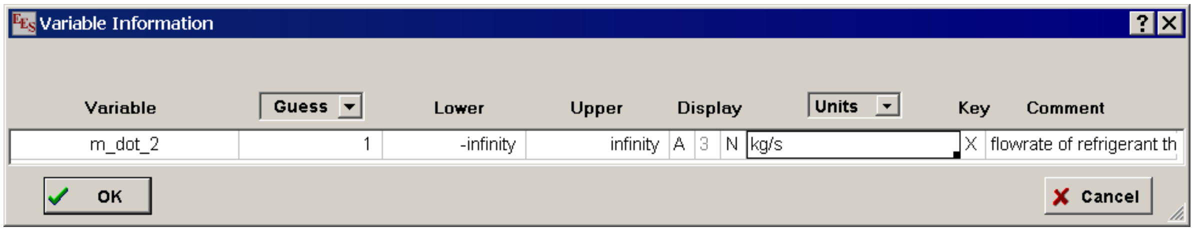
After entering the equations for this problem and (optionally) checking the syntax using the Check/Format command in the Calculate menu, the Equations window will appear as shown. Note that _1 and _2 were used in this example to designate states 1 and 2. Array variables with [1] and [2] could also have been used. Comments are normally displayed in blue on a color monitor. Other formatting options are set with the Preferences command in the Options menu. The $TabStops directive makes it easy to align the comments.

[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.



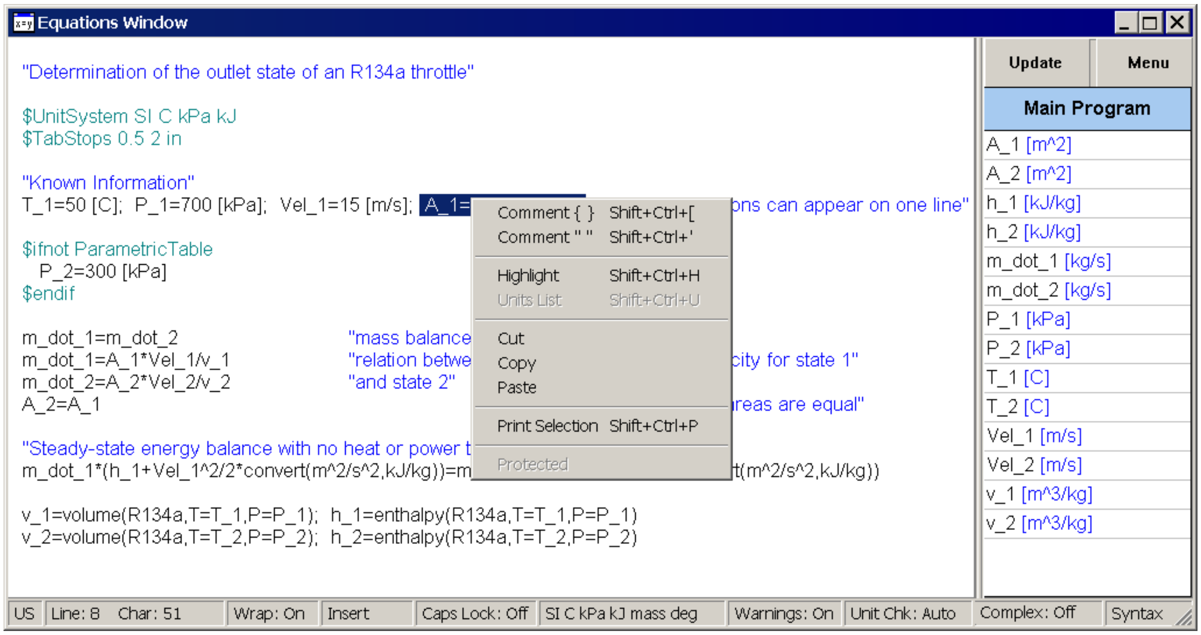
```
"Determination of the outlet state of an R134a throttle"

$UnitSystem SI C kPa kJ
$TabStops 0.5 2 in

"Known Information"
T_1=50 [C];  P_1=700 [kPa];  Vel_1=15 [m/s];  A_1=0.0110 [m^2]    "multiple equations can appear on one line"

$ifnot ParametricTable
   P_2=300 [kPa]
$endif

m_dot_1=m_dot_2                        "mass balance"
m_dot_1=A_1*Vel_1/v_1                  "relation between mass flowrate and fluid velocity for state 1"
m_dot_2=A_2*Vel_2/v_2                  "and state 2"
A_2=A_1                                                    "inlet and outlet areas are equal"

"Steady-state energy balance with no heat or power terms"
m_dot_1*(h_1+Vel_1^2/2*convert(m^2/s^2,kJ/kg))=m_dot_2*(h_2+Vel_2^2/2*convert(m^2/s^2,kJ/kg))

v_1=volume(R134a,T=T_1,P=P_1);  h_1=enthalpy(R134a,T=T_1,P=P_1)
v_2=volume(R134a,T=T_2,P=P_2);  h_2=enthalpy(R134a,T=T_2,P=P_2)
```

Unit consistency is as important as entering the correct equations. EES can check the unit consistency of the equations provided that the units of each variable are known. The units of constants, such as T_1, can be set by following the value with its units in square brackets. The units of variables, such as m_dot_2, can be set in several ways. Perhaps the simplest way is to click on the variable name in the Variable Information panel to the right of the Equations window. The following small dialog window will appear where information about the variable, including its guess values, limits, and units can be set.



Variable	Guess	Lower	Upper	Display			Units	Key	Comment
m_dot_2	1	-infinity	infinity	A	3	N	kg/s	X	flowrate of refrigerant th

Right-clicking the mouse-button in the Equations Window will bring up a pop-up menu with several options. The Comment { } menu item will place commenting braces { } around the selected. If the selection has braces, they will be removed. The Comment " " will use quotes rather than braces to comment/uncomment the selected text. Highlite will either set or clear a colored background around the text. Cut, Copy, Paste and Print Section have their usual meaning.



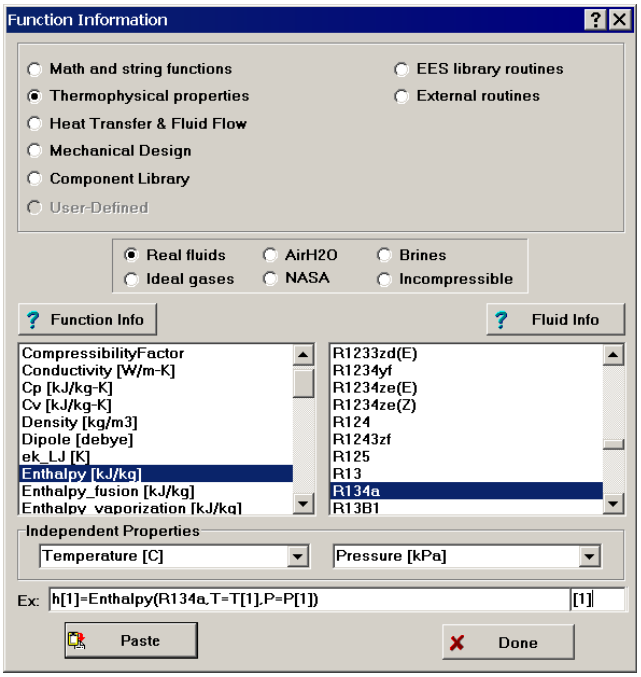
```
"Determination of the outlet state of an R134a throttle"

$UnitSystem SI C kPa kJ
$TabStops 0.5 2 in

"Known Information"
T_1=50 [C];  P_1=700 [kPa];  Vel_1=15 [m/s];  A_1=                    ons can appear on one line"

$ifnot ParametricTable
   P_2=300 [kPa]
$endif

m_dot_1=m_dot_2                   "mass balance
m_dot_1=A_1*Vel_1/v_1             "relation betwe                     city for state 1"
m_dot_2=A_2*Vel_2/v_2             "and state 2"
A_2=A_1                                                               reas are equal"

"Steady-state energy balance with no heat or power t
m_dot_1*(h_1+Vel_1^2/2*convert(m^2/s^2,kJ/kg))=m                      t(m^2/s^2,kJ/kg))

v_1=volume(R134a,T=T_1,P=P_1);  h_1=enthalpy(R134a,T=T_1,P=P_1)
v_2=volume(R134a,T=T_2,P=P_2);  h_2=enthalpy(R134a,T=T_2,P=P_2)
```

```
US | Line: 8   Char: 51 | Wrap: On | Insert | Caps Lock: Off | SI C kPa kJ mass deg | Warnings: On | Unit Chk: Auto | Complex: Off | Syntax
```

Note the use of the **Convert** function in energy balance 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 'Fluid 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.



Function Information

- ○ Math and string functions
- ● Thermophysical properties
- ○ Heat Transfer & Fluid Flow
- ○ Mechanical Design
- ○ Component Library
- ○ User-Defined
- ○ EES library routines
- ○ External routines

● Real fluids ○ AirH2O ○ Brines
○ Ideal gases ○ NASA ○ Incompressible

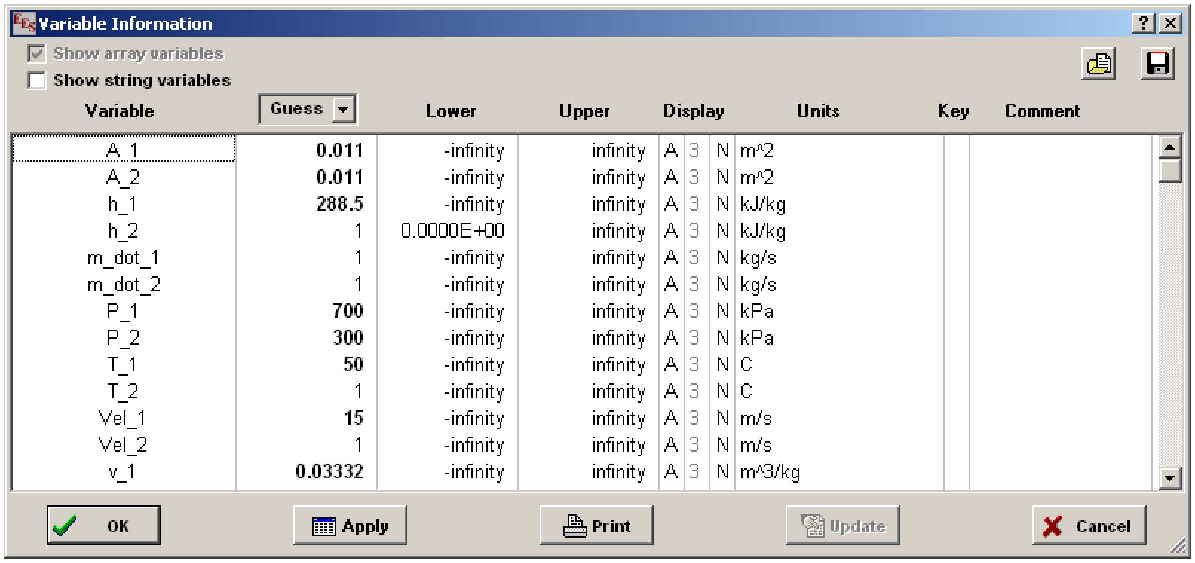
CompressibilityFactor	R1233zd(E)
Conductivity [W/m-K]	R1234yf
Cp [kJ/kg-K]	R1234ze(E)
Cv [kJ/kg-K]	R1234ze(Z)
Density [kg/m3]	R124
Dipole [debye]	R1243zf
ek_LJ [K]	R125
Enthalpy [kJ/kg]	R13
Enthalpy_fusion [kJ/kg]	R134a
Enthalpy_vaporization [kJ/kg]	R13B1

Independent Properties: Temperature [C] Pressure [kPa]

Ex: h[1]=Enthalpy(R134a,T=T[1],P=P[1]) [1]

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.



Variable Information

☑ Show array variables
☐ Show string variables

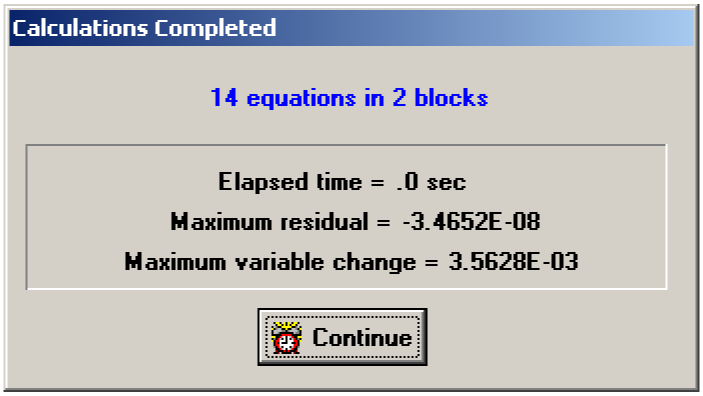
Variable	Guess	Lower	Upper	Display			Units	Key	Comment
A_1	0.011	-infinity	infinity	A	3	N	m^2		
A_2	0.011	-infinity	infinity	A	3	N	m^2		
h_1	288.5	-infinity	infinity	A	3	N	kJ/kg		
h_2	1	0.0000E+00	infinity	A	3	N	kJ/kg		
m_dot_1	1	-infinity	infinity	A	3	N	kg/s		
m_dot_2	1	-infinity	infinity	A	3	N	kg/s		
P_1	700	-infinity	infinity	A	3	N	kPa		
P_2	300	-infinity	infinity	A	3	N	kPa		
T_1	50	-infinity	infinity	A	3	N	C		
T_2	1	-infinity	infinity	A	3	N	C		
Vel_1	15	-infinity	infinity	A	3	N	m/s		
Vel_2	1	-infinity	infinity	A	3	N	m/s		
v_1	0.03332	-infinity	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 N (for N significant figures), F (for fixed number of digits to the right of the decimal point), E (for exponential format), G (for enGineering format) and T (for time 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 formatting effects such as normal (default), bold, boxed.

The units of the variables can be specified in the Variable Information dialog. 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, by default, it is configured to automatically check unit consistency of each equation. The automatic unit checking can be turned off with a control in the Preferences dialog. Unit checking is one the most important capabilities provided by EES. You will rarely want to disable automatic unit checking, but if you do, manual unit checking is provided with the Check Units command in the Calculate menu. In this example, we will not enter any additional units in the Variable Information dialog in order to see the unit checking warnings that EES will raise when the problem is solved. The last two columns allow a variable to be optionally designated as a Key variable and provide a space for a comment.

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, h_2, should be reasonably close to the value of h_1. Set its guess value to 100 and its lower bound to 0. Set the guess value of the outlet specific volume, v_2, to 0.1 and its lower bound to 0. The lower bound of Vel_2 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.



Calculations Completed

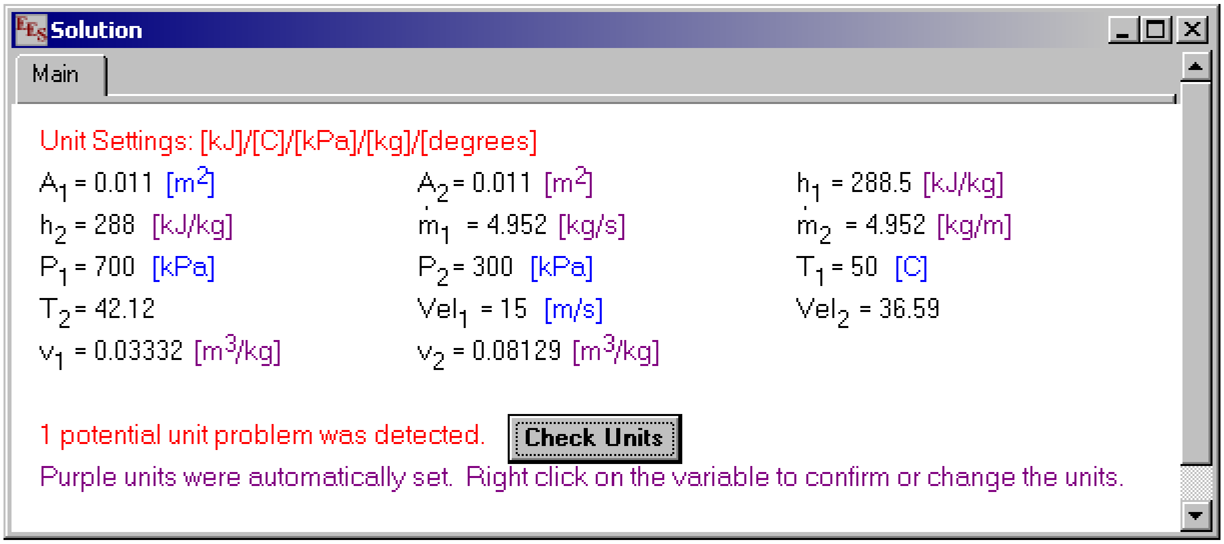
14 equations in 2 blocks

Elapsed time = .0 sec

Maximum residual = -3.4652E-08

Maximum variable change = 3.5628E-03

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. The problem is now completed since the values of T_2, m_dot_2, and Vel_2 are determined. However, note that EES indicates that there are potential unit problems.



Solution

Main

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

A_1 = 0.011 [m^2] A_2 = 0.011 [m^2] h_1 = 288.5 [kJ/kg]

h_2 = 288 [kJ/kg] $\dot{m}_1$ = 4.952 [kg/s] $\dot{m}_2$ = 4.952 [kg/m]

P_1 = 700 [kPa] P_2 = 300 [kPa] T_1 = 50 [C]

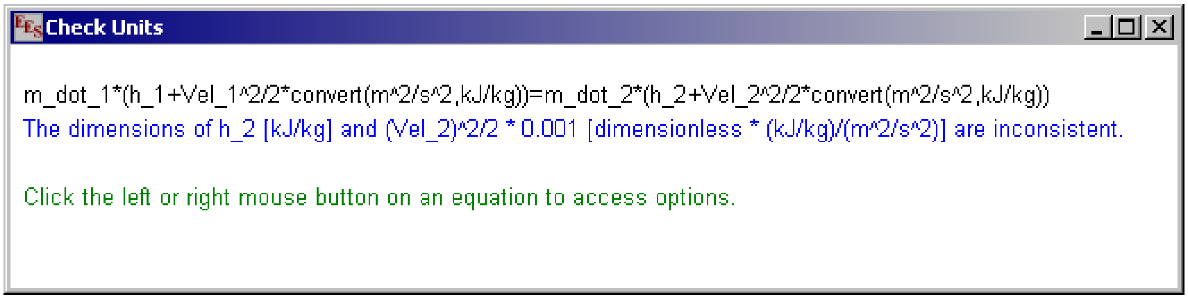
T_2 = 42.12 Vel_1 = 15 [m/s] Vel_2 = 36.59

v_1 = 0.03332 [m^3/kg] v_2 = 0.08129 [m^3/kg]

1 potential unit problem was detected.

Purple units were automatically set. Right click on the variable to confirm or change the units.

EES will display the unit problems if it set to automatically check units in the Preference dialog of the Options menu. Usually, the unit problems arise because the units of some of the variables have not been specified. EES can automatically provide the units for variables that are on the left side of an equal sign and have not been previously specified. (This capability can be disabled in the Preferences dialog.) Variables with units that have been automatically set will be shown in purple font. Right click on the variable with the purple units to accept or change the automatically provided unit. Even with the automatically-supplied units, unit conversions may still be needed. Even when units have been specified, one or more equations may be dimensionally incorrect. Click the Check Units button to see a list of the unit errors for this example.

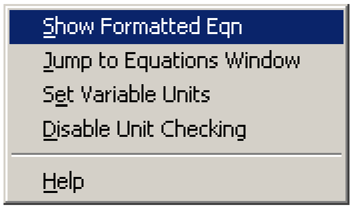


m_dot_1*(h_1+Vel_1^2/2*convert(m^2/s^2,kJ/kg))=m_dot_2*(h_2+Vel_2^2/2*convert(m^2/s^2,kJ/kg))

The dimensions of h_2 [kJ/kg] and (Vel_2)^2/2 * 0.001 [dimensionless * (kJ/kg)/(m^2/s^2)] are inconsistent.

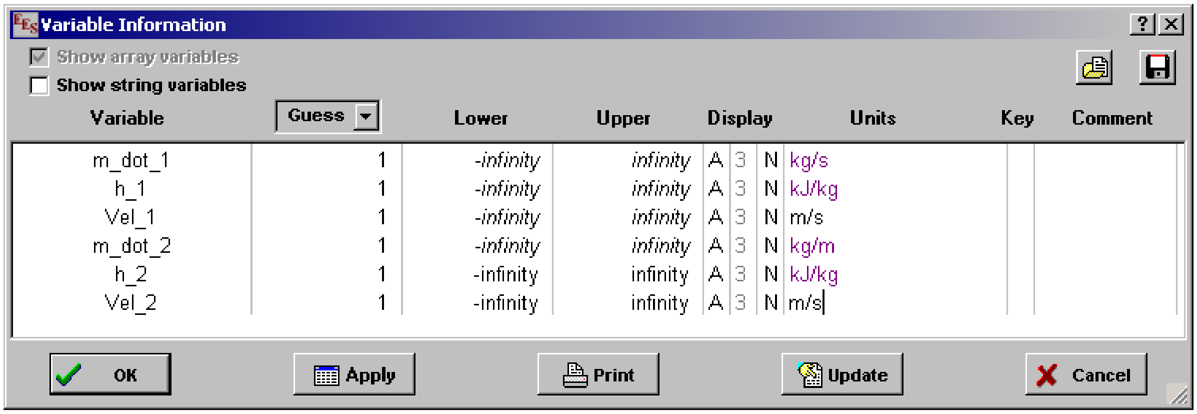
Click the left or right mouse button on an equation to access options.

If the message regarding the unit inconsistency is not clear, click on the equation to bring up the following pop-up menu. Then select Show Formatted Eqn to display the equation in the Formatted Equation window showing the units of variable and constants.



In this example, unit problem warnings occur because we have not yet set the units of variable Vel_2. The units can be set in several ways. Perhaps the easiest is to right click on one or more variables in the Solution Window and enter the units into the dialog that appears. Alternatively, we could left or click on an equation in the Check Units window to bring up the popup menu shown above.

The Set Variable Units menu item will bring up variable information for the selected equation. Enter units of m/s for Vel_2, as shown below. Click on the purple units to change or accept them. If there were other equations with unit warnings, you would repeat the process for the other equations until all of the unit problem warnings are removed.

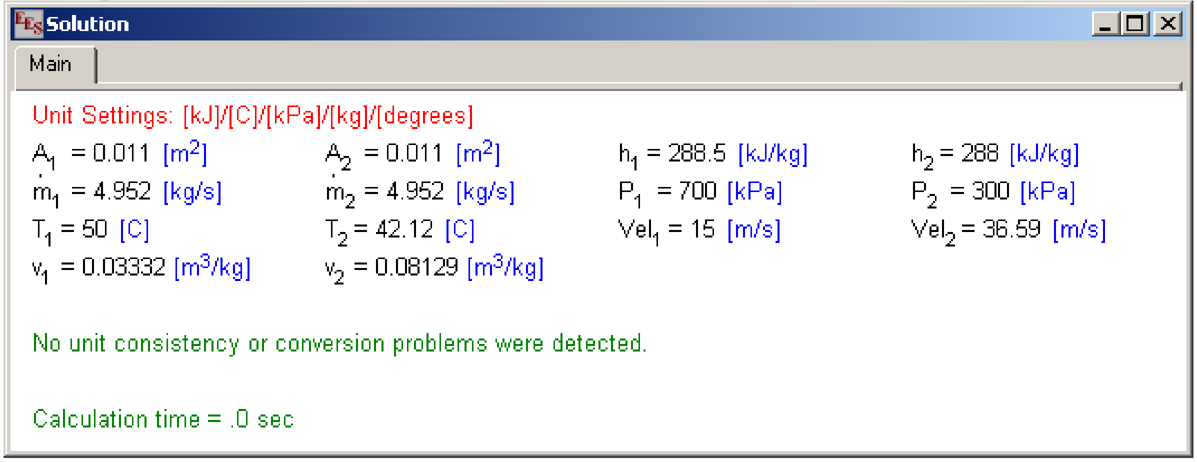


☑ Show array variables

□ Show string variables

Variable	Guess	Lower	Upper	Display			Units	Key	Comment
m_dot_1	1	-infinity	infinity	A	3	N	kg/s		
h_1	1	-infinity	infinity	A	3	N	kJ/kg		
Vel_1	1	-infinity	infinity	A	3	N	m/s		
m_dot_2	1	-infinity	infinity	A	3	N	kg/m		
h_2	1	-infinity	infinity	A	3	N	kJ/kg		
Vel_2	1	-infinity	infinity	A	3	N	m/s		

When all of the units have been specified, the Solution Window will indicate that there are no unit problems.



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

A_1 = 0.011 [m^2] A_2 = 0.011 [m^2] h_1 = 288.5 [kJ/kg] h_2 = 288 [kJ/kg]

$\dot{m}_1$ = 4.952 [kg/s] $\dot{m}_2$ = 4.952 [kg/s] P_1 = 700 [kPa] P_2 = 300 [kPa]

T_1 = 50 [C] T_2 = 42.12 [C] Vel_1 = 15 [m/s] Vel_2 = 36.59 [m/s]

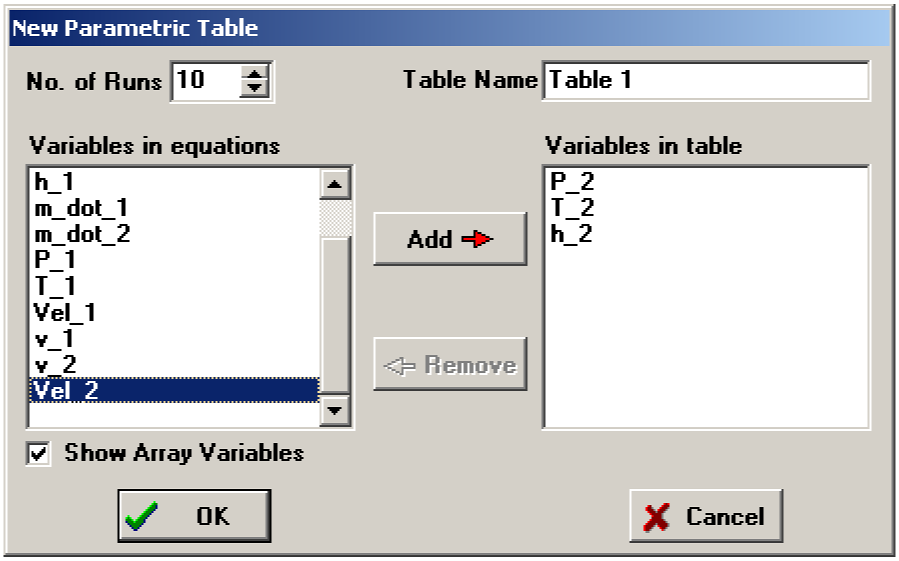
v_1 = 0.03332 [m^3/kg] v_2 = 0.08129 [m^3/kg]

No unit consistency or conversion problems were detected.

Calculation time = .0 sec

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 P_2, T_2, Vel_2, and h_2. Click on P_2 from the variable list on the left. This will cause P_2 to be selected and the Add button will become active.



No. of Runs 10

Table Name Table 1

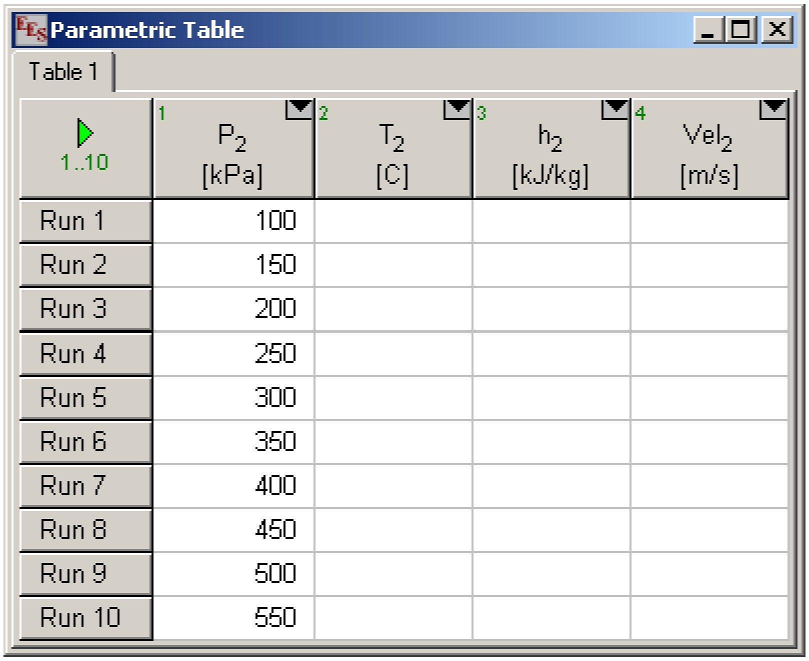
Variables in equations: h_1, m_dot_1, m_dot_2, P_1, T_1, Vel_1, v_1, v_2, Vel_2

Variables in table: P_2, T_2, h_2

☑ Show Array Variables

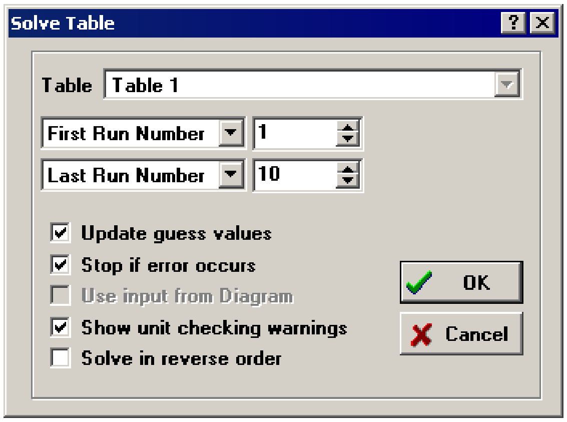
Now click the Add button to move P_2 to the list of variables on the right. Repeat for T_2, h_2, and Vel_2, using the scroll bar to bring the variable into view if necessary. (As a short cut, you can double-click on the variable name in the list on the left to move it to the list on the right.). The 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 P_2 = 300 equation currently in the Equations window or enclose it in comment brackets { }. This equation will not be needed because the value of P_2 will be set in the table. Now enter the values of P_2 for which T_2 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 P_2 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 P_2 [kPa]	2 T_2 [C]	3 h_2 [kJ/kg]	4 Vel_2 [m/s]
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 and the table for which the calculations will be done.



Solve Table

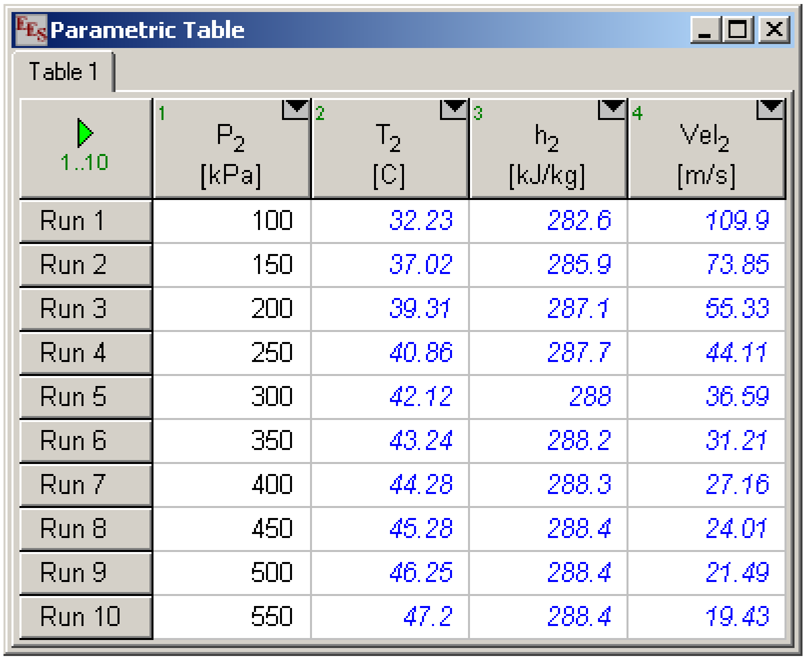
Table: Table 1

First Run Number: 1

Last Run Number: 10

☑ Update guess values
☑ Stop if error occurs
☐ Use input from Diagram
☑ Show unit checking warnings
☐ Solve in reverse order

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 T_2, Vel_2, and h_2 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.



Parametric Table

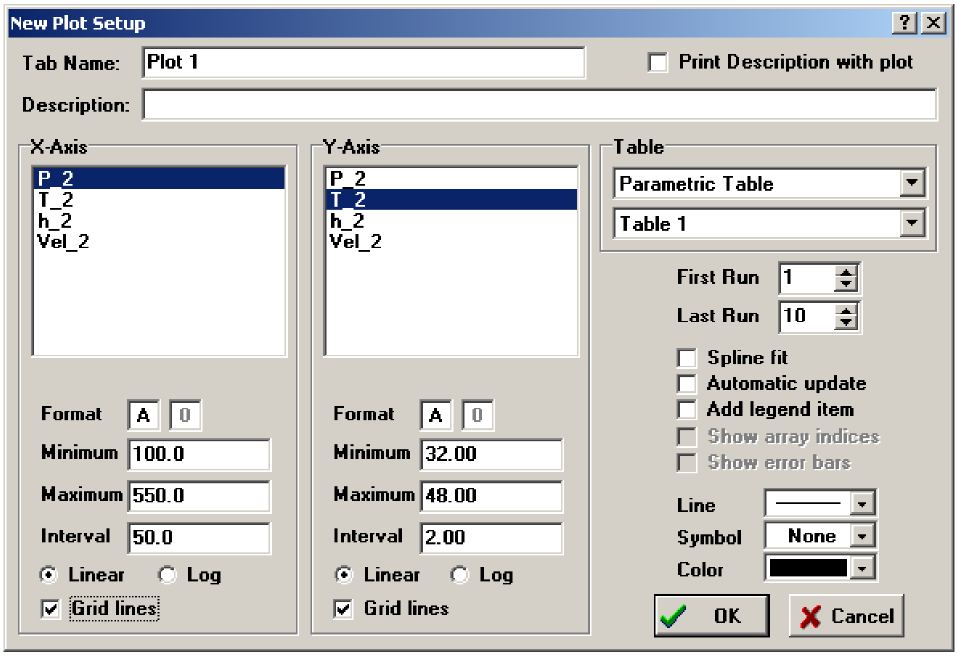
Table 1

1..10	1 P_2 [kPa]	2 T_2 [C]	3 h_2 [kJ/kg]	4 Vel_2 [m/s]
Run 1	100	*32.23*	*282.6*	*109.9*
Run 2	150	*37.02*	*285.9*	*73.85*
Run 3	200	*39.31*	*287.1*	*55.33*
Run 4	250	*40.86*	*287.7*	*44.11*
Run 5	300	*42.12*	*288*	*36.59*
Run 6	350	*43.24*	*288.2*	*31.21*
Run 7	400	*44.28*	*288.3*	*27.16*
Run 8	450	*45.28*	*288.4*	*24.01*
Run 9	500	*46.25*	*288.4*	*21.49*
Run 10	550	*47.2*	*288.4*	*19.43*

The relationship between variables such as P_2 and T_2 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 P_2 to be the x-axis by clicking on P_2 in the x-axis list. Click on T_2 in the y-axis list. Select the scale limits for P_2 and T_2, 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 and animation capabilities. Read on.



New Plot Setup

Tab Name: Plot 1

☐ Print Description with plot

Description:

X-Axis: P_2, T_2, h_2, Vel_2

Format A 0

Minimum 100.0

Maximum 550.0

Interval 50.0

● Linear ○ Log

☑ Grid lines

Y-Axis: P_2, T_2, h_2, Vel_2

Format A 0

Minimum 32.00

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

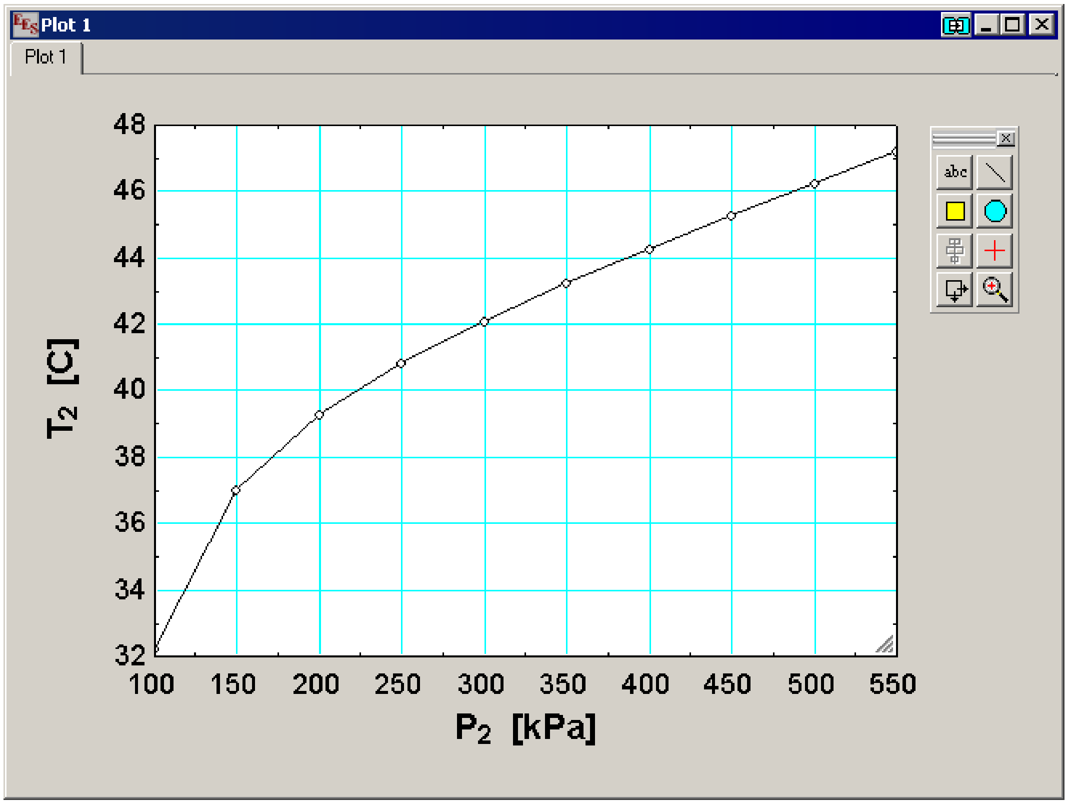
☐ Show array indices

☐ Show error bars

Line

Symbol None

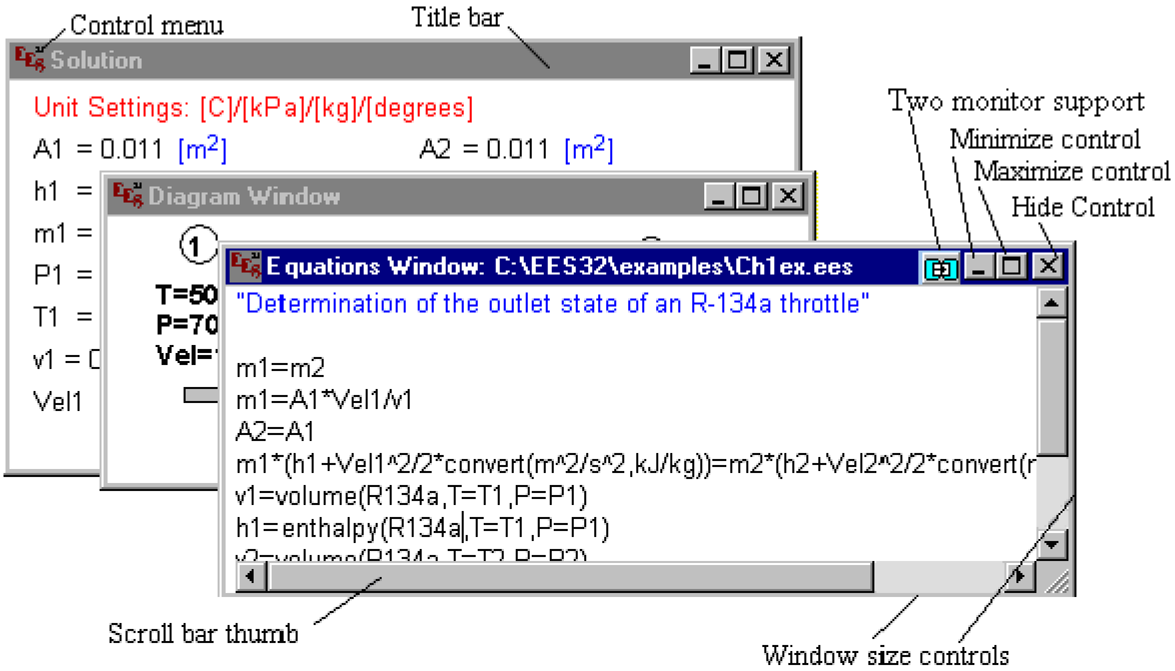
Color


Plot 1
Plot 1
abc
48
46
44
42
40
38
36
34
32
T2 [C]
100
150
200
250
300
350
400
450
500
550
P2 [kPa]

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 title bar. The figure below shows three overlapped EES windows. The appearance may be slightly for different versions of the Windows operating system.



```
Unit Settings: [C]/[kPa]/[kg]/[degrees]
A1 = 0.011 [m^2]                A2 = 0.011 [m^2]
h1 =
m1 =
P1 =
T1 =
v1 = 0
Vel1
```

```
"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(r
v1=volume(R134a,T=T1,P=P1)
h1=enthalpy(R134a,T=T1,P=P1)
v2=volume(R134a,T=T2,P=P2)
```

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.

7. Many of the windows in the Professional version will display the two-monitor display button at the right of the title bar, adjacent to the minimize, maximize and close buttons. Clicking this button will cause the window to be remain in front of all other EES windows and it will also allow the window to be moved outside of the bounding box provided by the main EES window. This display state is very handy when using two monitors as then, the selected window can be dragged to the second monitor. Only one EES window is allowed to be in the 2-monitor display format at any one time. The window will return the window to its normal display state if the two-monitor display button for another window is clicked or if the button is clicked.

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 bring up 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. 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. Note that the Wrap Long Lines option can also be turned on or off by clicking on Wrap: in the Equations window status bar. The Formatted Equations window will also line break the comments in the display window and in the printed output. Any line in the Equations Window that begins with // will be ignored as if the entire line is a comment under all circumstances. Any characters to the right of // that is not located at the start of a line will be considered to be a comment unless an $Allow // directive is in effect.
3. 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 semicolon[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. Earlier versions of EES required that equations consist of 255 or fewer characters. This limitation no longer exists, but use of shorter equations is recommended as it assists in debugging in clarity
4. 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.
5. The order of mathematical operators used in the equations conforms 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

[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.

(X – 3) / 4 = 5

6. 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.

7. 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**).

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

9. The commercial and educations versions of EES have an upper limit of 6000 variables. The Professional version can have 12,000 variables. The 64-bit Professional version allows 24,000 variables.

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 selecting 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 Load Library command in the File menu and the $INCLUDE directive may also be used to import the equations from an existing file. 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), with either of the two types of comments. The menu also provides options for cutting, highlighting, copying or printing of the selected text. A units list is also available in the pop-up menu to simplify entering the units of constants.

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.

14. A Variable Information panel is displayed to the right of the Equations Window. A vertical slider bar allows setting of the width of the Variable Information panel. The slider can be moved to the right of the window to hide the Variable Information panel.

Click the Update button in the panel to update the Variable Information panel to display the current variables in the Equations window if changes have been made. Click on a variable name to bring up a dialog that allows setting or viewing its properties. By default, the variables that are shown are for the Main program. If you have Functions, Procedures, Modules or Subprograms in the Equations Window, click the Menu button to select the program unit for which variables are to be displaye.d

15. Active hypertext can be entered into the Equations window. Obviously, this should only be done within comments. EES will automatically identify hypertext links that begin with http:\\, https:\\ or file: In addition, a hypertext link that begins with \\EES_ following by the name of an EES window is understood to be a link to that window. Clicking on a link will move the focus to that location, opening another program if necessary. A summary of the recognized links is shown below:

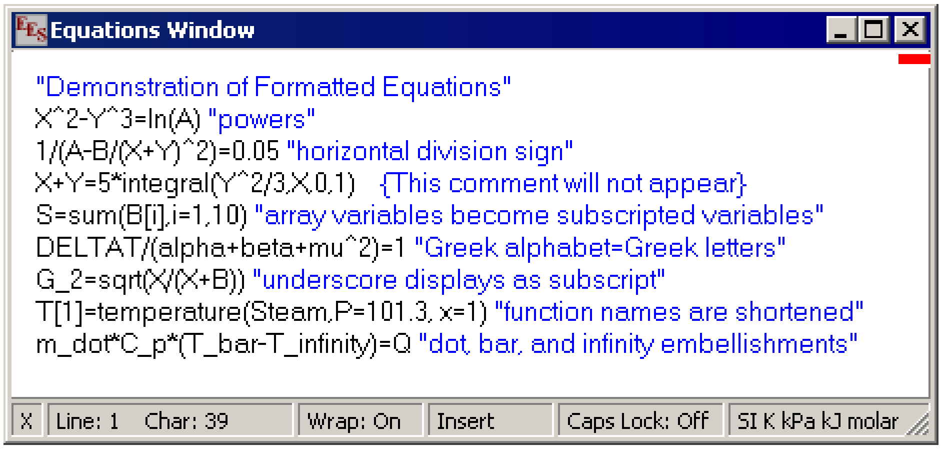
http:\\fchart.com	{open a browser and point it at the web page that follows http:}
https:\\fchart.com	{open a browser and point it at the web page that follows https:}
file:c:\ees32\ees_manual.pdf	{open the local file with a filename that follows file: and start the appropriate application}
\\EES_Solution	{open the Solution window and bring it to the front}
\\EES_Format	{open the Formatted equations window and bring it to the front}
\\EES_Plot	{open the Plot window and bring it to the front}
\\EES_Plot3	{open the Plot window and bring the 3rd plot to the front}
\\EES_Parametric	{open the Parametric table and bring it to the front}
\\EES_Lookup	{open the Lookup table and bring it to the front}
\\EES_Array	{open the Array table and bring it to the front}
\\EES_Integral	{open the Integral table and bring it to the front}
\\EES_Report	{open the Report window and bring it to the front}
\\EES_Diagram	{open the Diagram window and bring it to the front}
\\EES_Residual	{open the Residuals window and bring it to the front}
\\EES_Calculator	{open the EES Calculator window and bring it to the front}
\\EES_Solve	{solve the current set of equations}
\\EES_SolveTable	{apply the Solve Table command in the Calculate menu}
\\EES_MinMax	{apply the MinMax command in the Calculate menu}
\\EES_MinMaxTable	{apply the MinMax Table command in the Calculate menu}

16. Constants representing time may be entered in Time Format. For example, a time of 9 hours, 30 minutes and 10 seconds may be entered as 9:30:10. EES will internally convert this value to hours and associate units of hours with its value.

17. A status bar is provided at the bottom of the Equations window showing the position of the cursor, and settings for word wrap, Caps lock, Insert mode, unit system, warnings, unit checking, and complex math mode. Clicking on any of the settings in the status bar toggles their settings.

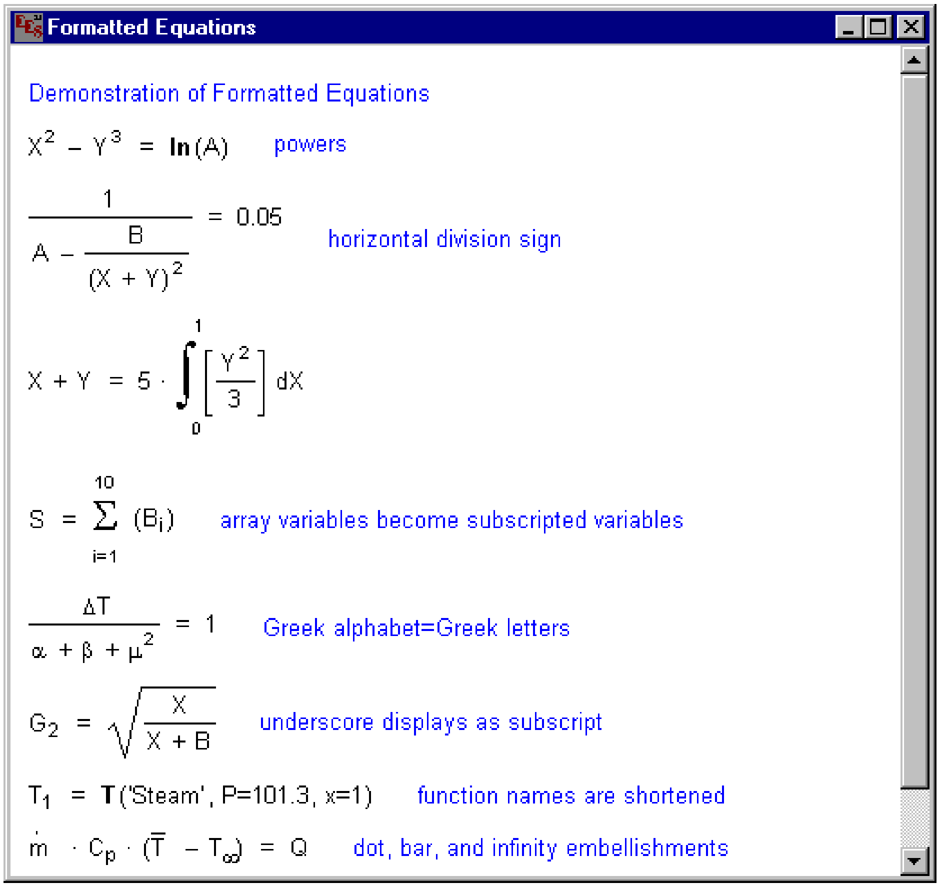
Formatted Equations Window

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"
```

X | Line: 1 Char: 39 | Wrap: On | Insert | Caps Lock: Off | SI K kPa kJ molar



Demonstration of Formatted Equations

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

$$\frac{1}{A - \frac{B}{(X + Y)^2}} = 0.05$$

horizontal division sign

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

$$S = \sum_{i=1}^{10} (B_i)$$

array variables become subscripted variables

$$\frac{\Delta T}{\alpha + \beta + \mu^2} = 1$$

Greek alphabet=Greek letters

$$G_2 = \sqrt{\frac{X}{X + B}}$$

underscore displays as subscript

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

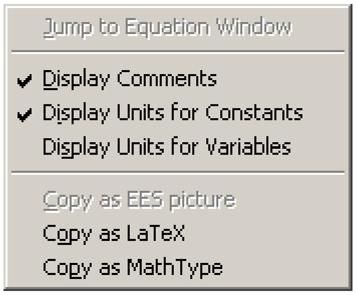
$\dot{m} \cdot C_p \cdot (\bar{T} - T_\infty) = Q$ dot, bar, and infinity embellishments

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.

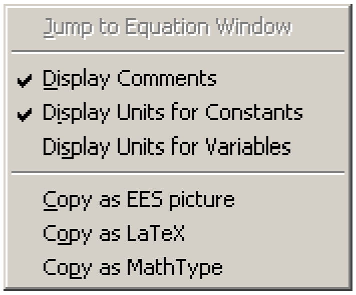
Placing _dot, _ddot, _bar, _hat or _tilde after a variable name places a dot, double dot, bar, hat (^) or tilde (~) centered over the name. The _infinity results in a subscript with the infinity symbol (∞). A vertical bar character in a variable name signifies the start of a superscript. For example, 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 can be displayed with the units of constants and/or variables shown. Displaying the units makes it easier to identify a units inconsistency in the equations. To change the display setting for units of constants and/or variables, right click anywhere in the Formatted Equations window. The popup menu shown below will appear with menu options for displaying units.



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). In addition, the equations can be copied in either LaTeX or MathType compatible formats in the Professional version. 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 All Equations 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. Next, right-click to bring up the following pop-up menu and select on of the three copy commands at the bottom of this menu to place the equation on the Clipboard in the desired format.

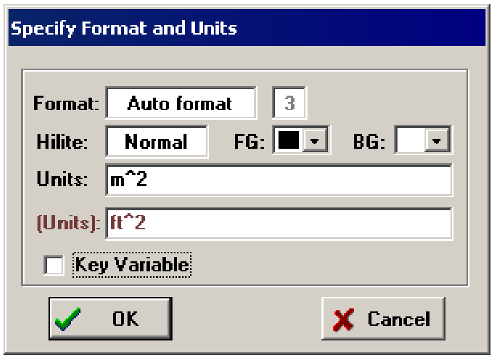


The text in the Formatted Equations window cannot be edited. However, clicking the right mouse button on an equation in the Formatted Equation window will bring up the popup menu shown above. Selecting the first command in this menu will cause the display focus to move to the Equations window with the cursor positioned on the equation that was selected in the Formatted Equations window.

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.


Specify Format and Units

Format: Auto format 3
Hilite: Normal FG: BG:
Units: m^2
(Units): ft^2
□ Key Variable

The numerical format (style and digits), the units, and the alternate set of 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. If any of the variables have been designated to be key variables, a separate tabbed window will display these variables with optional comments. The order in which variables appear is not alphabetical, as in other Solution window displays, but instead by the order in which key variables are designated. The order can be changed by dragging a variable to the line that it should appear. Additional text and separators can be optionally added by right-clicking the mouse at the point where the item it to be inserted, or by pressing the Return key.

4. If an alternate set of units is provided, it must have the same units as the primary set. In this case, the value of the variable will be displayed in both sets of units.

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

6. 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. Variables that were not evaluated are shown in gray.

7. 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.

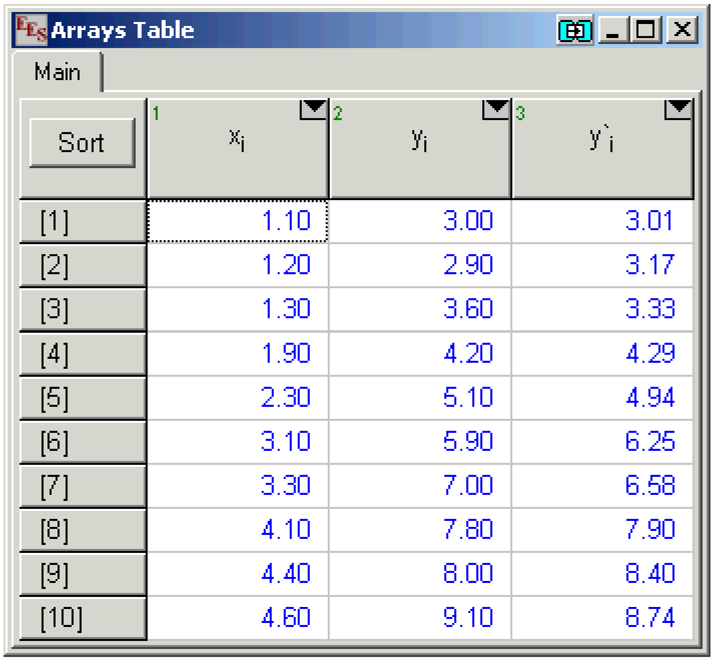
8. If the ☑ Display subscripts and Greek symbols option in the General Display tab of the Preferences dialog is selected, EES will display subscripts and superscripts of variable units. For example, m^2 will appear as m^2. An underscore character is used to indicate a subscript so lb_m will appear as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

9. If the ☑ Show function/procedure/module values option in the General Display tab of the Preferences dialog is selected, EES will display the most recent values of local variables in EES Functions, Procedures, Modules and Subprograms in separate tabbed windows within the Solution window.

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.

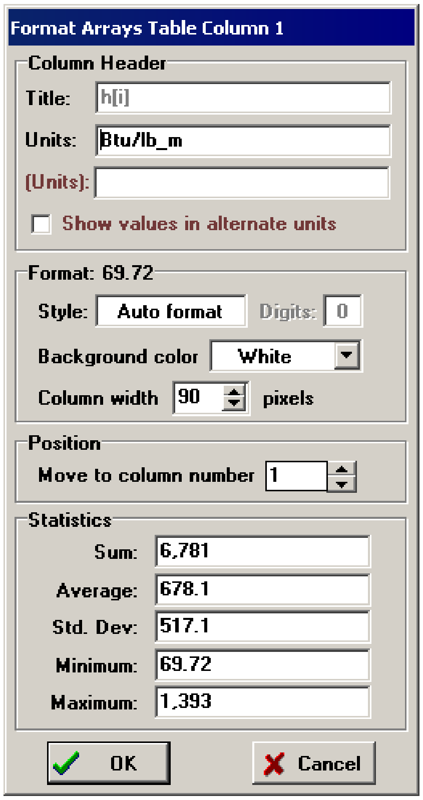
The values of all variables in the Main program including array variables are normally displayed in the Main tab of the Solution window after calculations are completed. However, array variables of the Main program and each function, Procedure, Module and Subprogram may optionally be displayed in a separate Arrays windows, rather than in the Solution window. This option for arrays in the Main program 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 for the Main program after calculations are completed showing all array values used in the problem with the array index value in the first column. This option can also be controlled with the $ARRAYS ON/OFF directive. Note that the array variables in Functions, Procedures, Subprograms, and Modules can also be displayed in separate tabbed windows of the Arrays table window by using the $ARRAYS directive. Clicking the Sort button will arrange the columns in alphabetical order. The positions of the columns can be changed by dragging the column header cell to the desired column position.



	1 x_i	2 y_i	3 $y`_i$
[1]	1.10	3.00	3.01
[2]	1.20	2.90	3.17
[3]	1.30	3.60	3.33
[4]	1.90	4.20	4.29
[5]	2.30	5.10	4.94
[6]	3.10	5.90	6.25
[7]	3.30	7.00	6.58
[8]	4.10	7.80	7.90
[9]	4.40	8.00	8.40
[10]	4.60	9.10	8.74

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 right mouse button on the array name at the top of the column. The following dialog window will appear in which the units, alternate 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. The sum, average, and standard deviation of the numbers in the selected column of the table are also shown in this display.


Format Arrays Table Column 1

Column Header

Title: h[i]

Units: Btu/lb_m

[Units]:

☐ Show values in alternate units

Format: 69.72

Style: Auto format Digits: 0

Background color White

Column width 90 pixels

Position

Move to column number 1

Statistics

Sum: 6,781

Average: 678.1

Std. Dev: 517.1

Minimum: 69.72

Maximum: 1,393

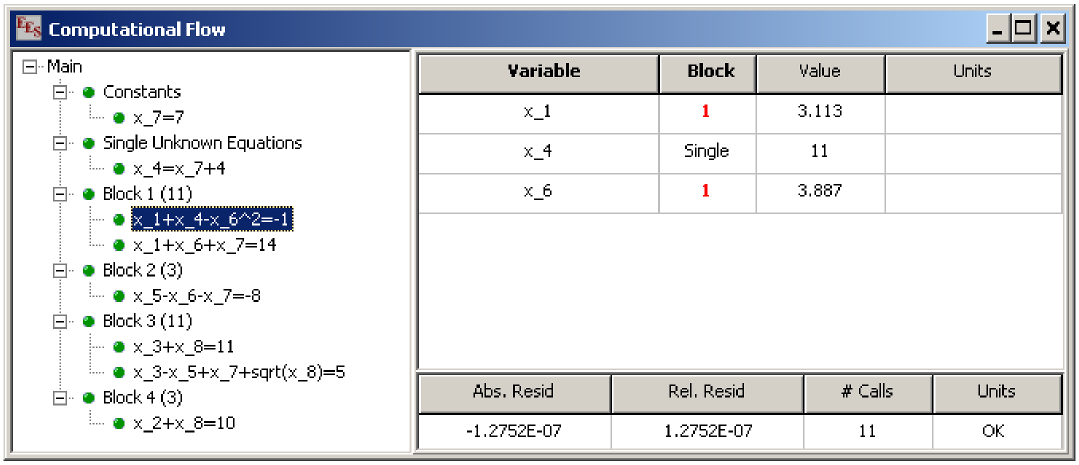
Selected columns in the Arrays Table can be hidden by right-clicking on the column header and selecting Properties from the pop-up menu. In the Properties dialog, set the format Style type to be Hidden.

If alternate units are provided, they must have the same dimensions as the primary set of units. If the “Show values in alternate units” is selected, values will be displayed in both the primary and alternative units.

Computational Flow Window

The Computational Flow Window (available in the Professional version) provides information on how EES solves the system of equations in the Main program and Subprograms and how many iterations are needed for each equation. It also identifies where the value of every variable in the program is set, including variables that are defined in Parametric tables, Diagram windows, Macros and elsewhere. The Computational Flow Window is most useful in the event that the system of equations does not converge to a solution. It is also useful for determining the bottlenecks in the calculation procedure that result in increased computational time. It is better suited for large problems than the Residuals Window which provides similar information in the Commercial version and is discussed in following section. In the Professional version, the Computation Flow menu item is shown in the Windows menu, but the Residuals window menu item is not shown. If you wish to display the Residuals window instead of the Computational Flow window, hold the Ctrl key down while selecting Computational Flow from the Windows menu. See the Mastering EES ebook for a complete discussion of the Computational Flow window.

The Computational Flow window is divided into three panes. The pane on the left shows a tree view of all of information flow. A separate tree structure is provided for the Main program and for every Subprogram in the Equations window, as shown.



```
⊟ Main
   ⊟ ● Constants
        ● x_7=7
   ⊟ ● Single Unknown Equations
        ● x_4=x_7+4
   ⊟ ● Block 1 (11)
        ● x_1+x_4-x_6^2=-1
        ● x_1+x_6+x_7=14
   ⊟ ● Block 2 (3)
        ● x_5-x_6-x_7=-8
   ⊟ ● Block 3 (11)
        ● x_3+x_8=11
        ● x_3-x_5+x_7+sqrt(x_8)=5
   ⊟ ● Block 4 (3)
        ● x_2+x_8=10
```

Variable	Block	Value	Units
x_1	1	3.113	
x_4	Single	11	
x_6	1	3.887	

Abs. Resid	Rel. Resid	# Calls	Units
-1.2752E-07	1.2752E-07	11	OK

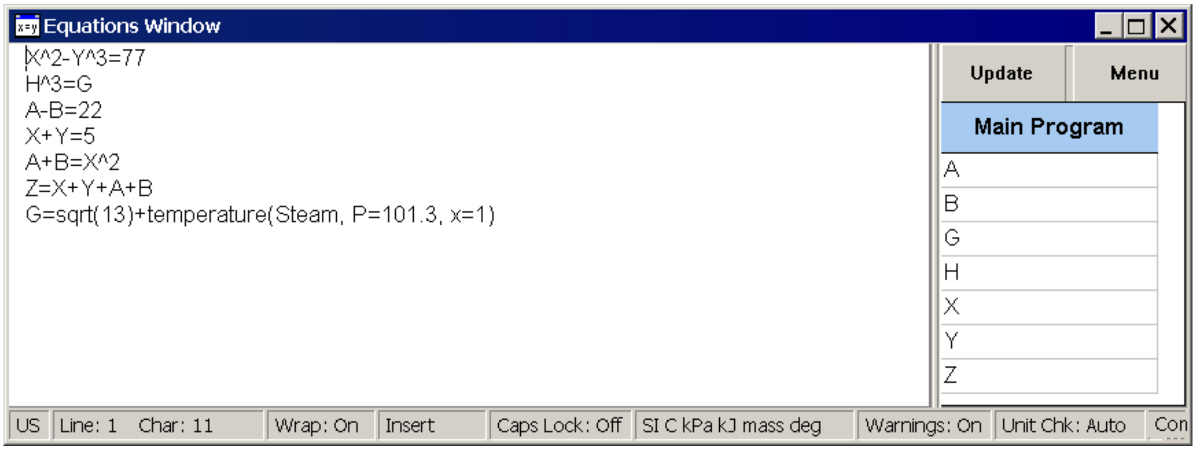
EES reorders and blocks the equations to allow an efficient solution. EES will next look for groups of equations that need be solved simultaneously. Clicking the left mouse button on a block name selects that block. All of the variables that are involved in the equations for the selected block will then be displayed in the top right panel.

Clicking on an equation in the tree view pane will display the variable information for the selected equation, rather than for the selected block. In addition, a third pane will appear at the bottom right of the window showing the absolute and relative residuals for the selected equation.

Residuals Window

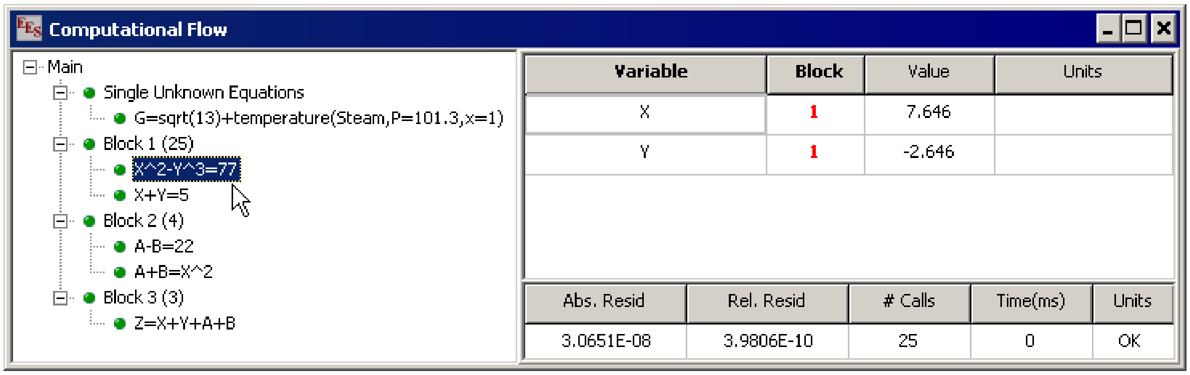
The Residuals window indicates the equation blocking and calculation order used by EES, and the status of the unit checking, in addition to the relative and absolute residual values. The absolute residual of an equation is the difference between the values on the left and right hand sides of the equation. 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.



```
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.



```
Main
  Single Unknown Equations
    G=sqrt(13)+temperature(Steam,P=101.3,x=1)
  Block 1 (25)
    X^2-Y^3=77
    X+Y=5
  Block 2 (4)
    A-B=22
    A+B=X^2
  Block 3 (3)
    Z=X+Y+A+B
```

Variable	Block	Value	Units
X	1	7.646	
Y	1	-2.646	

Abs. Resid	Rel. Resid	# Calls	Time(ms)	Units
3.0651E-08	3.9806E-10	25	0	OK

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 assigned to Block 0. Note the EES has placed a question mark by this equation, since its

[3] If the value of the left hand side of an equation is less than 1E-20, the absolute and relative residuals assume the same value.

unit consistency is suspect. 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.[4] 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.

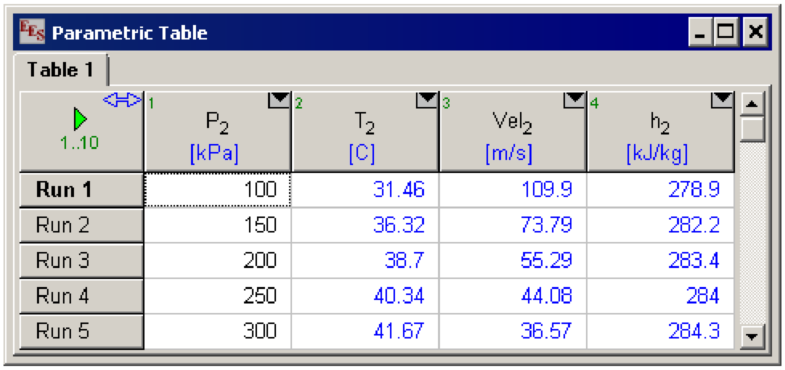
It is possible to display the Residuals window in a debugging situation. If the number of equations is less than the number of unknowns, EES will not be able to solve the equation set, but the Residuals window can be made visible by selecting it from the Windows menu. Normally, the block numbers appear 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 and how many times each equation was evaluated in the process of solving the set. The status bar at the bottom of the Residuals window shows the values of variables in the equation under that mouse cursor. The block number will be displayed in bold font for equations that have residuals greater than the tolerance specified with the Tolerance command (Options menu). 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 equation selected. 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)

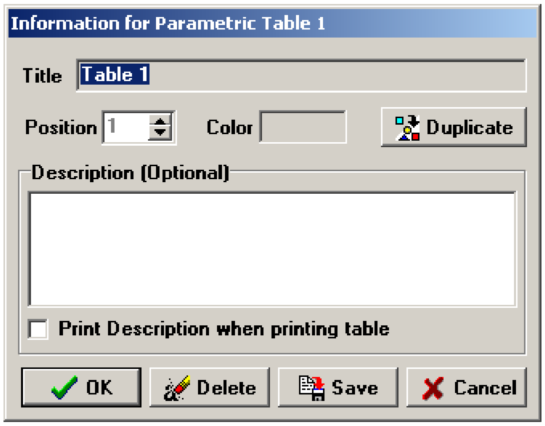
Parametric Table Window



1..10	1 P_2 [kPa]	2 T_2 [C]	3 Vel_2 [m/s]	4 h_2 [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. Clicking the left-mouse button on the tab brings the corresponding Parametric Table to the front. Clicking the right-mouse button on the tab brings up the following dialog with controls that allow the tab position to change, or to duplicate, delete, or save the table to an external file. All tables can be save as .txt or .xls files.



Information for Parametric Table 1

Title: Table 1

Position: 1 Color:

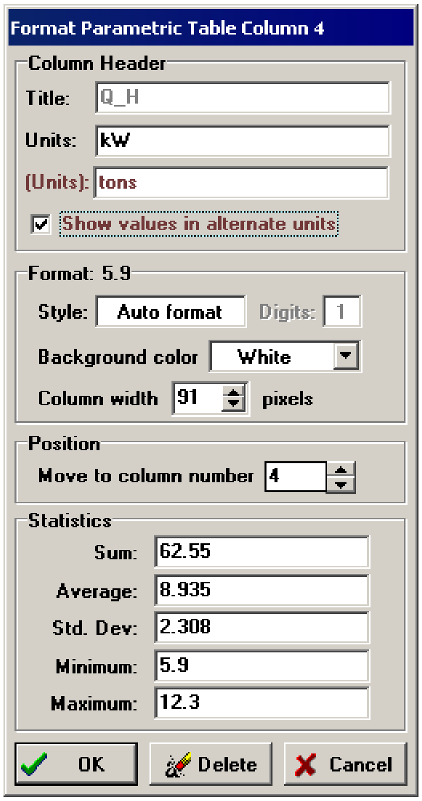
Description (Optional)

☐ Print Description when printing table

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 9000. 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.

4. 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 left mouse button in the column header cell and drag it to the desired column position. Alternatively, 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, alternate units, column width, and column background color can also be entered or changed at this point. Values displayed in alternate units can be plotted in the Professional version.



Format Parametric Table Column 4

Column Header

Title: Q_H

Units: kW

(Units): tons

☑ Show values in alternate units

Format: 5.9

Style: Auto format Digits: 1

Background color: White

Column width: 91 pixels

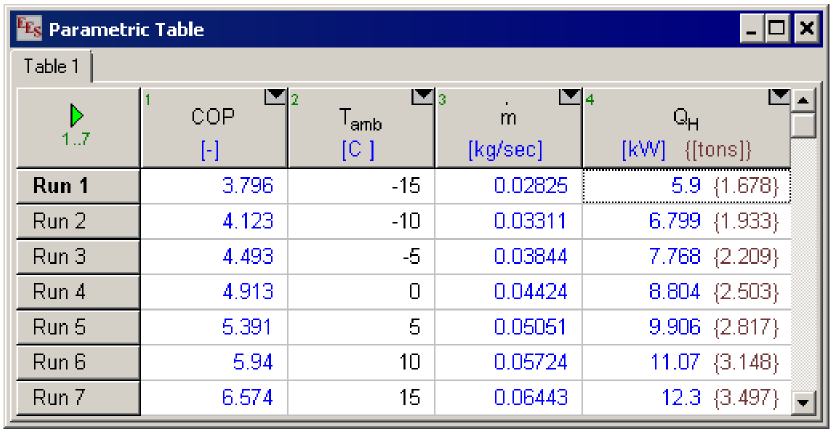
Position

Move to column number: 4

Statistics

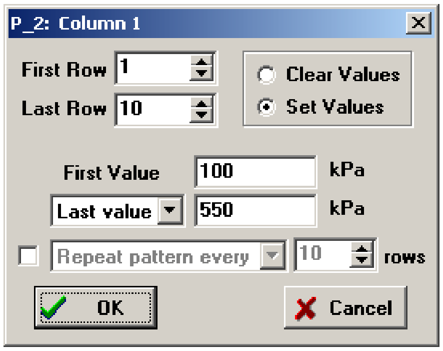
Sum:	62.55
Average:	8.935
Std. Dev:	2.308
Minimum:	5.9
Maximum:	12.3

5. As shown in the figure above, an alternate set of units can be provided for a variable in the Parametric Table or other table types. If the S"how values in alternate units" control box is selected the values in the column will be displayed in both sets of units, as shown.



1..7	1 COP [-]	2 T_{amb} [C]	3 $\dot{m}$ [kg/sec]	4 Q_H [kW] {[tons]}
Run 1	3.796	-15	0.02825	5.9 {1.678}
Run 2	4.123	-10	0.03311	6.799 {1.933}
Run 3	4.493	-5	0.03844	7.768 {2.209}
Run 4	4.913	0	0.04424	8.804 {2.503}
Run 5	5.391	5	0.05051	9.906 {2.817}
Run 6	5.94	10	0.05724	11.07 {3.148}
Run 7	6.574	15	0.06443	12.3 {3.497}

6. Values can be automatically entered into one or more Parametric Table(s) 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 left mouse button 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 in addition to providing additional capabilities.



First Row 1
Last Row 10
○ Clear Values
● Set Values
First Value 100 kPa
Last value 550 kPa
□ Repeat pattern every 10 rows

In some situations, it is necessary to fill a table with a pattern of some type. For example, you may wish to enter 1, 2, 3, 4, and 5 in the first 5 rows of the column and then repeat these values in the next five rows and so on down the table. Alternatively, you may wish to enter 1, 1, 1, 1, 1 in the first 5 rows followed by 2, 2, 2, 2, 2 in the next five rows and so on. The capability to enter patterns of this type is provided with the controls that are above the OK and Cancel buttons. Click the check box to enable the controls. Select Repeat Pattern every to enter a repeating pattern of the form 1, 2, 3, 4, 5. Select Apply Pattern Every to enter a sequence such as 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, etc. The pattern is specified by the first value and the last value / increment / multiplier. The pattern repeats every N rows where N is the value you provide. The pattern is applied to rows indicated with the First Row and Last Row controls.

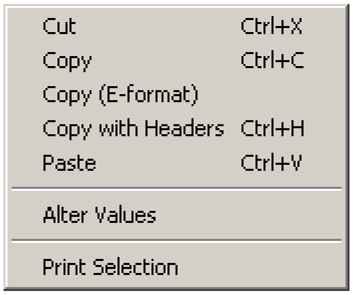
The Professional version will accept a previously defined variable name in place of a number for the specified number or rows and in the Alter Values dialog (accessed from the Table menu). For example, if variable L is set to a contant in the Equations window, L can be entered in place of its value in the First Value and Increment/Last fields.

7. 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 slide the mouse to the last row. Continue to hold the Shift key down while clicking in the green triangle.

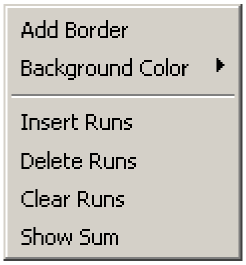
8. A Sum row which displays the sum of the values in each column may be hidden or made visible using the 'Include a Sum row in the Parametric table' control provided in the Preferences dialog window (Options tab) in the Options menu.

9. 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.

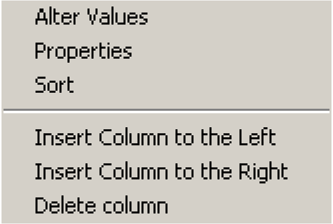
10. Right-clicking in Parametric, Lookup, Arrays, or Integral table windows will bring up the following pop-up menu allowing the current selection to be cut, copied, or printed. This option provides a convenient way to print a specified section of a table. The Copy (E-format) will copy values with 12 significant figures so that they can be pasted to another application with no loss in precision. Alternatively, holding the Shift key down while copying a selection from the Parametric or Lookup tables will cause all numbers to be copied in E-format. EES will automatically retain all of the precision in the copied values when they are copied from any of the EES tables to other EES tables in the same program.



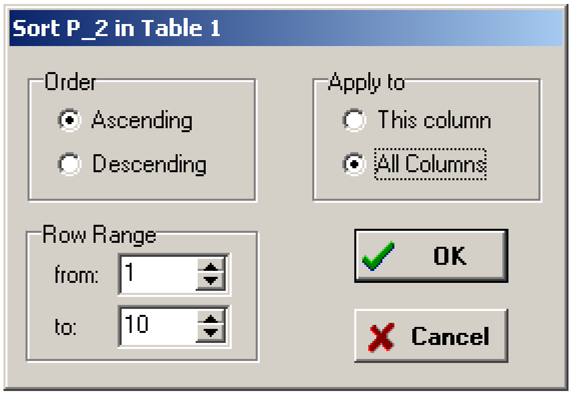
11. Right clicking in the Row column of any table brings up a menu that allows a back color or border to be associated with the row. Also rows and be inserted or deleted from the Parametric and Lookup tables.



12. A right-click in the cell header will bring up the following popup menu.



The Alter Values menu option will allow values to changed as described in item 5 above. The Properties menu item brings up the Format Column dialog that allows the format, units, and other information in a column to be edited, as indicated in item 3 above. The Sort menu option allows the data in the column to be sorted in ascending or descending order. The sort can be applied to a single column or to all columns table.


Sort P_2 in Table 1

Order
● Ascending
○ Descending

Apply to
○ This column
● All Columns

Row Range
from: 1
to: 10

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

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

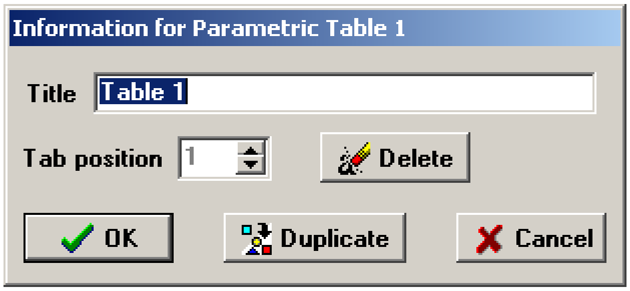
15. The **TableRun#** function returns the row of the table for which calculations are currently in progress.

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

17. 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.

18. 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 be 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.

19. An entire table can be duplicated or deleted by right-clicking on its tab at the top of the table window. A small dialog will appear having a Duplicate and a Delete button. The Delete cannot be undone so be careful.



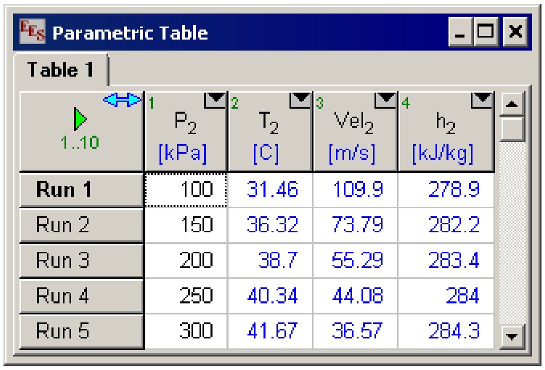
Information for Parametric Table 1

Title: Table 1

Tab position: 1

20. The tabs used to select different table windows can be 'dragged' to a different tab location. This method of reordering the tabs is quicker than the alternative of right-clicking on each tab and specifying the tab position.

21. In the Professional version, all tables are equipped with a control to automatically resize the column widths. The control appears as a left-right arrow icon in the upper left cell of the table. Clicking the icon causes all of the columns to be adjusted to a width that is just large enough to display all of the information the column, as shown below. Clicking the icon again returns the column widths to their original values. If the Ctrl key is depressed when the icon is clicked, the column widths will be set to their default values.



1..10	1 P_2 [kPa]	2 T_2 [C]	3 Vel_2 [m/s]	4 h_2 [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

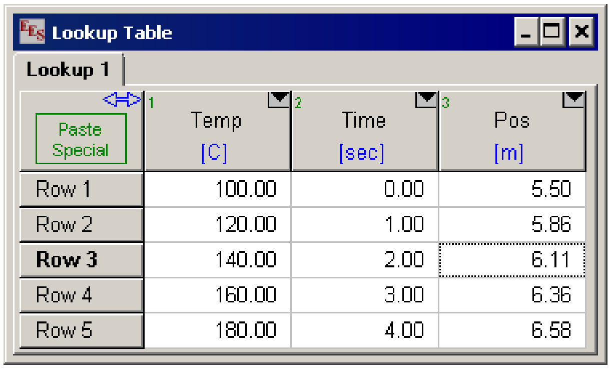
22. Variables can have a base and alternate set of units, and values in both sets of units can be displayed in the Parametric table. In the Professional version, clicking the triangular control will bring up a dialog that will allow values to be entered into the table in the base units even if values are provided in the alternate units, as described in the online help.

23. Hold the Ctrl-V keys down while pasting a row of data into the Parametric table will paste in successive rows of the table.

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. A maximum of 9000 rows is allowed in the Commercial version, but the Professional version has no limit. Both versions allow up to 1500 columns. 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.

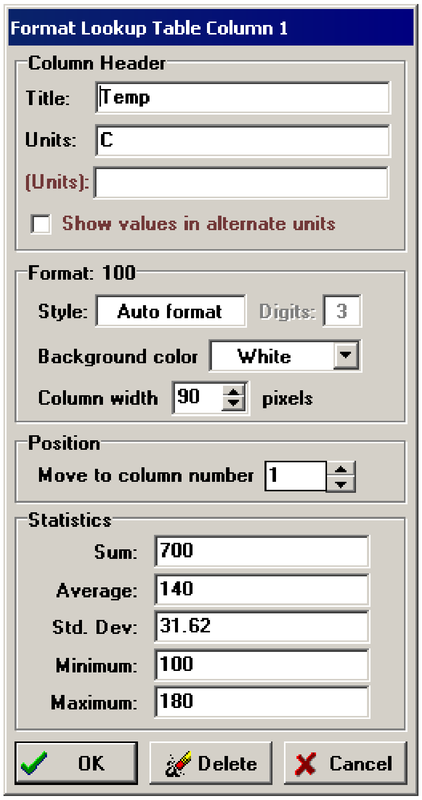


	1 Temp [C]	2 Time [sec]	3 Pos [m]
Row 1	100.00	0.00	5.50
Row 2	120.00	1.00	5.86
Row 3	140.00	2.00	6.11
Row 4	160.00	3.00	6.36
Row 5	180.00	4.00	6.58

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

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

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.



Title: Temp

Units: C

(Units):

□ Show values in alternate units

Format: 100

Style: Auto format Digits: 3

Background color White

Column width 90 pixels

Move to column number 1

Sum: 700

Average: 140

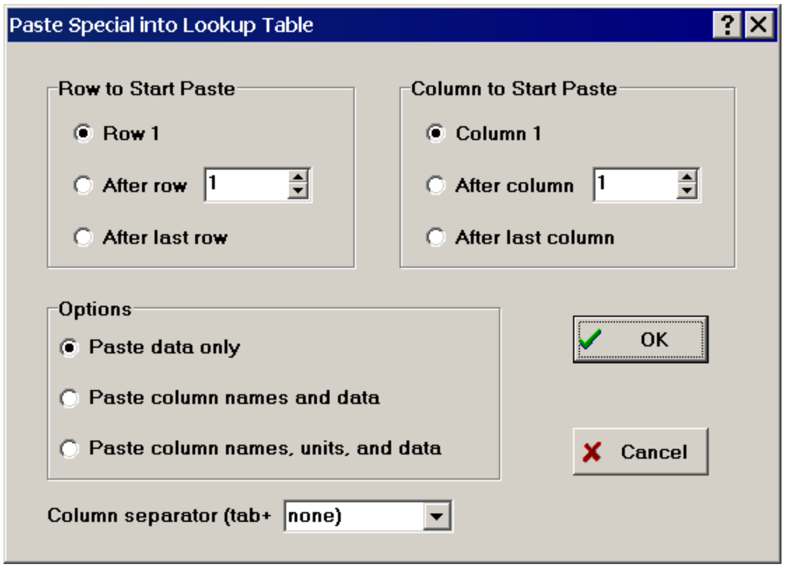
Std. Dev: 31.62

Minimum: 100

Maximum: 180

The column title can be changed and units and alternate units for the values in the column may be specified. Note that if alternate units are provided, they must have the same dimensions as the primary set of units. If the "Show values in alternate units control" is checked, values will be displayed in the table in both sets of units. 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 pasted into a Lookup Table via the Clipboard from another application, such as a spreadsheet. The Paste command will initiate the paste, starting in the row and column in which the cursor is located. However, the names of each column and the units cannot be pasted in this manner. The Paste Special command has been developed to provide this capability. The Paste Special command is accessed by clicking in the Paste Special box in the upper left of the table. Doing so will provide a choice of pasting the values, the values and column names, and the values, column names and units, as shown in the figure below.



Paste Special into Lookup Table

Row to Start Paste

● Row 1

○ After row 1

○ After last row

Column to Start Paste

● Column 1

○ After column 1

○ After last column

Options

● Paste data only

○ Paste column names and data

○ Paste column names, units, and data

Column separator (tab+ none)

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. Values can be entered into a column of a Lookup tables using an equation, similar to capability provided in spreadsheet programs. The equation can refer to values in other columns of the table and to defined EES variables. All of the built-in functions can be employed in the equations. Values in a column are represented in the equations with #n where n is the column number. For example, to convert temperatures in °C in column 1 of the Lookup table to values in K in the selected column, enter convertTemp(C,K,#1).

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. Data in the table can be sorted by right-clicking in the column header for the column that is to control the sort and selecting Sort from the popup menu.

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.

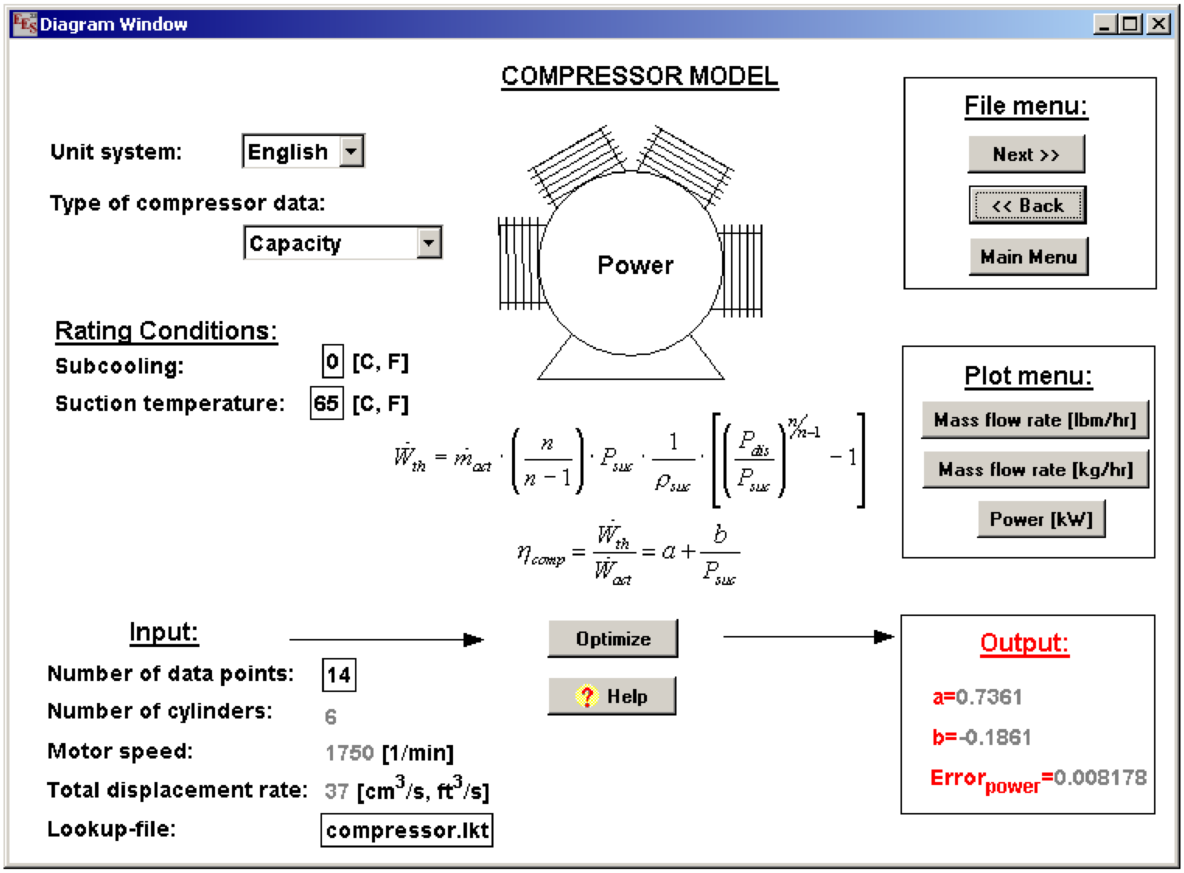
Integral Table Window

An Integral Table is automatically generated during calculations when one or more equation-based Integral functions are in use and the $IntegralTable directive is supplied. The Integral Table holds intermediate values of specified variables that occur during the numerical integration process. The values in this table can be plotted using the New Plot Window and Overlay Plot commands in the Plot mneu. The values can also be printed or copied in the same manner as for data in the Parametric table. The $IntegralTable directive can be supplied in Subprograms and in the Main program. A separate tab in the Integral Table window will be provided for each case.

.

Diagram Window

The Diagram window serves several functions. First, it provides a place to display a graphics and 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 to provide convenient 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, save or load input values, or start other programs. A Help button can be used to access help information from a text file, Windows help file, or web site. Shown below is a typical diagram window. Third, the Diagram window provides the focus for the animation capabilities in EES.



Unit system: English

Type of compressor data: Capacity

Rating Conditions:

Subcooling: 0 [C, F]

Suction temperature: 65 [C, F]

$$\dot{W}_{th} = \dot{m}_{act} \cdot \left(\frac{n}{n-1}\right) \cdot P_{suc} \cdot \frac{1}{\rho_{suc}} \cdot \left[\left(\frac{P_{dis}}{P_{suc}}\right)^{n/n-1} - 1\right]$$

$$\eta_{comp} = \frac{\dot{W}_{th}}{\dot{W}_{act}} = a + \frac{b}{P_{suc}}$$

Input:

Number of data points: 14

Number of cylinders: 6

Motor speed: 1750 [1/min]

Total displacement rate: 37 [cm^3/s, ft^3/s]

Lookup-file: compressor.lkt

Output:

a=0.7361

b=-0.1861

$Error_{power}$=0.008178

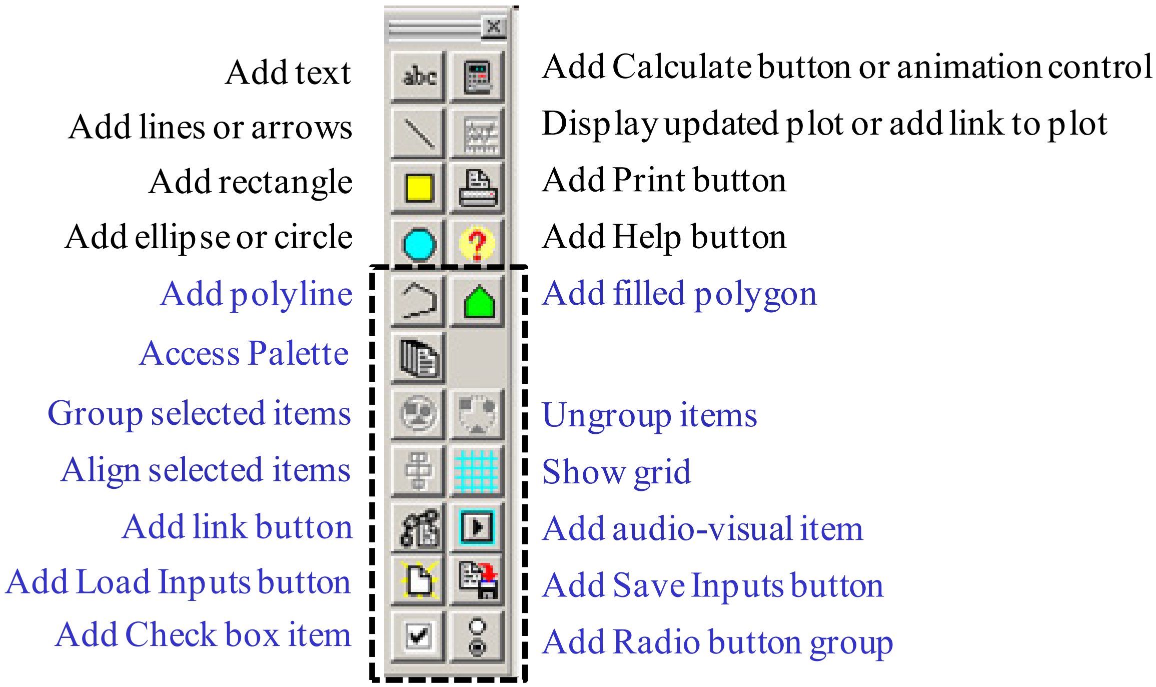
Creating the Diagram

There are two ways to place objects on the Diagram window. First, a graphical or text objects can be created in any drawing program that produces an object drawing such as Corel Draw, Designer, or PowerPoint. Select and copy the item(s) in the drawing application and then use the Paste command to place them 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. Graphic objects generated in either manner are 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 (shown below) 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. A statusbar is visible at the bottom of the Diagram window in development mode. The statusbar shows the cursor position, Diagram window identity and information about the object that the cursor is positioned on. When the tool bar is hidden, the Diagram window is in application mode. The tool bar can be hidden by clicking on the X in its title bar or by selecting the Show/Hide Diagram Tool Bar command. 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.

The tool bar appears as shown below. The items within the dotted lines are only available in the Professional version.



The Copy command in the Edit menu can be used to copy the either the entire contents of the Diagram window or selected items to the Clipboard. The copied information on the clipboard can then be pasted into a word processor or other application, into a different child Diagram window, or into the Diagram window of another instance of EES. In general, Input/Output text items should not be copied into another instance of EES since they contain specific pointers to EES variables that will not be appropriate in the copy.

Use the Select All and Delete command to remove the entire contents of a Diagram window. Right-clicking on the Diagram window will bring up a popup menu that allows the size of the window contents to be changed.

Moving the Diagram

Selected items in the Diagram window can be moved in development mode by pressing and holding the left mouse button down anywhere within a selected item while sliding the mouse. The items can also be moved one pixel at a time with each press of the arrow keys. Use the Select All command in the Edit menu to move the all of the contents of the Diagram Window to a new position.

Resizing the Diagram

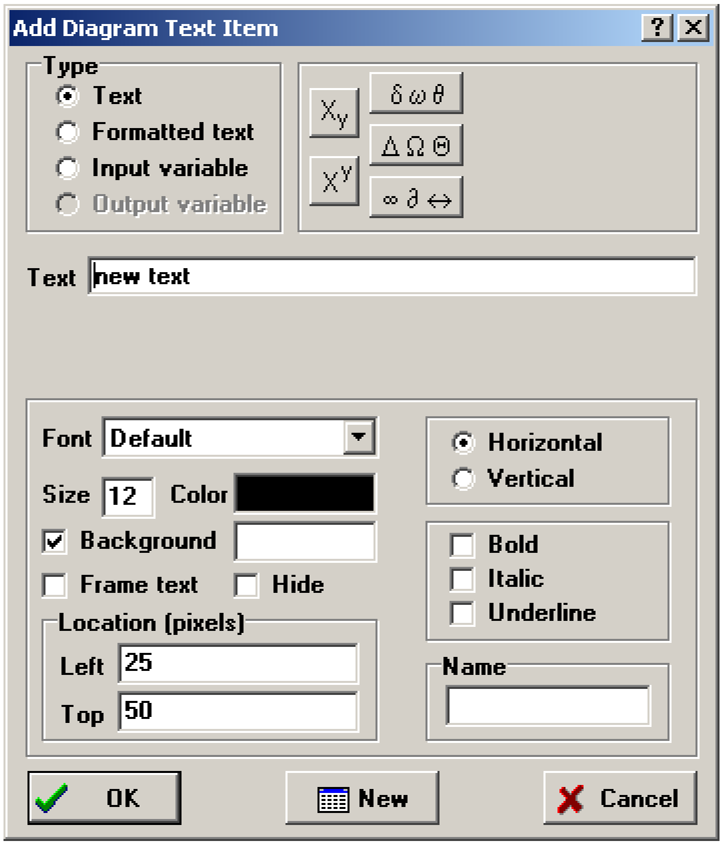
In development mode, the contents of the Diagram window can be resized by double-clicking the left mouse button (or clicking the right mouse button) anywhere within the Diagram window (or child Diagram window) except on a text, graphic object or button. After a positive confirmation, the graphical and text items will be proportionally resized to fit in the Diagram window. The contents of the Diagram window can be made larger or smaller by first changing the size of the Diagram window itself by dragging its lower left corner to a new location and then double-clicking in the window to resize the contents. The aspect ratio of Diagram window information is not changed it is resized. Buttons that are placed on the Diagram window can be resized independently by holding the Ctrl-key down while using the arrow keys. For example, to increase the width of a button, click on the button to select it (in development mode) and then press the right arrow key while holding the Ctrl key down. Each right arrow press increases the width of the button by one pixel. The left arrow key reduces the width. The up and down arrow keys increase and decrease the button height, respectively.

Adding and Moving Text on the Diagram Window abc

The Add Text button on the Diagram window toolbar allows text to be placed anywhere on the Diagram window. Four different types of text may be selected using the radio buttons at the upper left of the Add Text dialog 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 formatting buttons to the right of the text type radio buttons simply entry of subscripts, superscripts, and symbols. 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. Alternatively, selected text items can be moved by pressing the arrow keys. 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.

EES variables in any tabele can be designated to display in date or time format. The Date and Time functions convert a string containing a date or time to the internal numerical representation used in EES. The numerical represenation for dates and times is consistent with other programs, such as Excel so they can be copied and pasted between programs. Dates can be output in 10 formats and any understandable format can be used for input of dates or time. Date and time variables in any EES table can be plotted in the same manner as other data. The dates or times can be oriented at a specified angle with respect to the axis.



Add Diagram Text Item

Type
- ● Text
- ○ Formatted text
- ○ Input variable
- ○ Output variable

Text: new text

Font: Default

Size: 12 Color

☑ Background

□ Frame text □ Hide

Location (pixels)

Left: 25

Top: 50

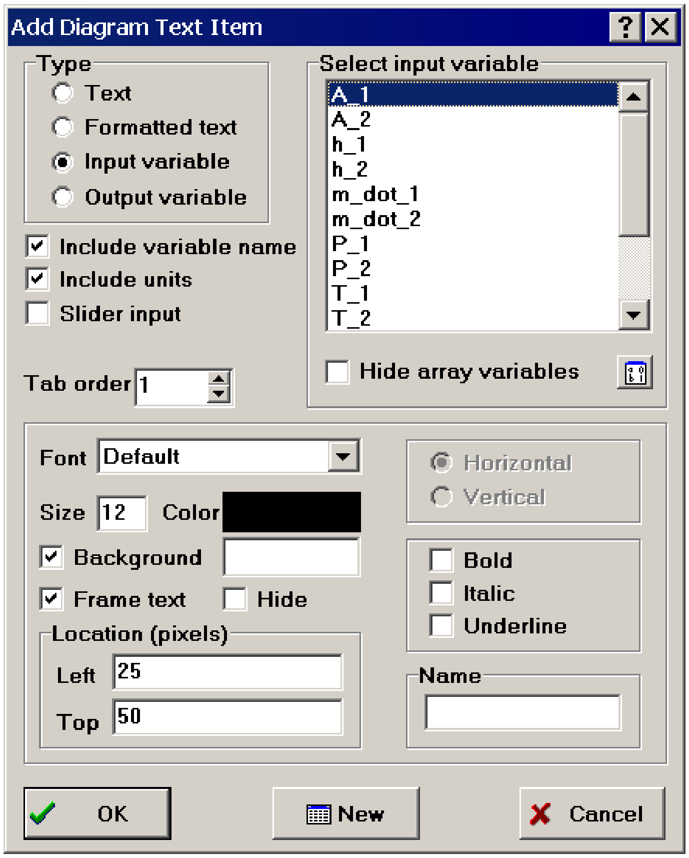
● Horizontal
○ Vertical

□ Bold
□ Italic
□ Underline

Name:

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 the 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 pull-down lists for the unit system and type of compressor data. The selection from the pull-down menu is assigned to the string variable. String variables can be used for many purposes including the names of fluids as described in Chapter 7. String variables can also be used to enter EES equations, as described below.

Assuming that the Diagram window is not hidden, EES will first examine the Diagram window to see which variables, if any, are to be input from the Diagram window when the Solve or Min/Max commands (Calculate menu) are issued.. The Solve Table and Min/Max Table commands provide a check box to indicate whether Diagram window information should be used. 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.



Add Diagram Text Item

Type
○ Text
○ Formatted text
● Input variable
○ Output variable

☑ Include variable name
☑ Include units
□ Slider input

Tab order 1

Select input variable
A_1
A_2
h_1
h_2
m_dot_1
m_dot_2
P_1
P_2
T_1
T_2

□ Hide array variables

Font Default
Size 12 Color
☑ Background
☑ Frame text □ Hide

Location (pixels)
Left 25
Top 50

● Horizontal
○ Vertical

□ Bold
□ Italic
□ Underline

Name

The Formatted text button allows a formatted text item be placed in the Diagram window and in child diagram windows. A formatted text item can consist of one or more lines of formatted text with embedded font, font size, font style, and font color information. The text item can include line breaks and tabs. The text item can be a table, copied from a spreadsheet program. In addition, EES variables can be included in the text so that the current value of the variable is automatically displayed.

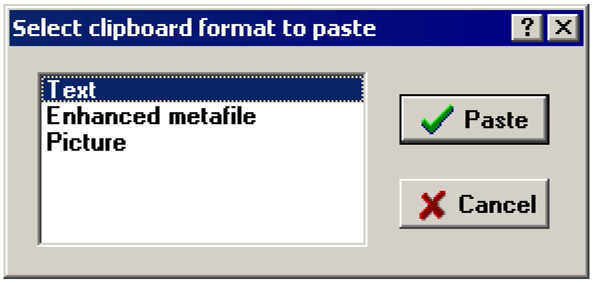
The Professional version of EES provides a field to enter the name of the text item as shown in the above dialogs. Assigning a name to a text item is useful only when the item is to have its attributes determined by the value of EES variables, such as for animation. If a name is provided, EES will create variables and associate them with this text object. For example, if the name of the text item is Area as shown in the above dialog, the following EES variables will be created (if they do not already exist) after pressing the OK button.

Area.left
Area.top
Area.color
Area.Hide

These variables can be set in the Equations window or in Check Boxes, Radio Groups, or Drop-Down lists in the Diagram window. Note that items can be hidden or shown by entering Name.hide or Name.show. A complete equation with an equal sign is not required.

Creating a Formatted Text Item

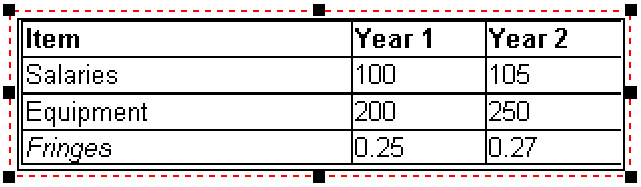
Formatted text items can be created or edited when the Diagram window is in Development mode. There are two ways to create a new item. One way is to click the Text button on the toolbar. This action will bring up the Add Diagram Text Item dialog. Click the Formatted text button at the upper left and enter the text in the large edit box that appears. An easier way is to simply copy the text from any application such as a word processor or spreadsheet program and then use the EES Paste command while the Diagram window is in front. If a formatted text item is selected, the Paste command will add the text to the existing text item. Otherwise a new text item will be created. Often applications place information on the Clipboard in more than one format. For example formatted text copied from WORD may exist on the Clipboard both as a picture and as text. In this case, EES will present the a dialog showing the possible paste formats, as shown below.



Selecting Text will result in just text being pasted. All formatting and pictures will be stripped. The Enhanced Metafile and Picture formats will paste a picture of the information, just as it appears in the application it was copied from. The picture does not allow editing of the text.

Resizing and Moving a Formatted Text Item

Select the formatted text item by clicking on it while in Development mode. A selected item will be displayed with a red dotted line border and 'resize' boxes, as shown for a table item below.

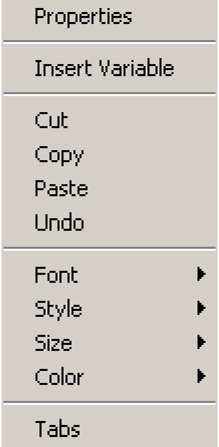


Item	**Year 1**	**Year 2**
Salaries	100	105
Equipment	200	250
Fringes	0.25	0.27

It may be necessary to resize the item when it is first pasted in. To resize, click the left mouse button on one of the eight resize boxes and drag it to a new location. To move the formatted text item to a different location in the window, click anywhere in the region between the red dotted line and the border of the text item and drag the text item to a new location.

Editing a Formatted Text Item

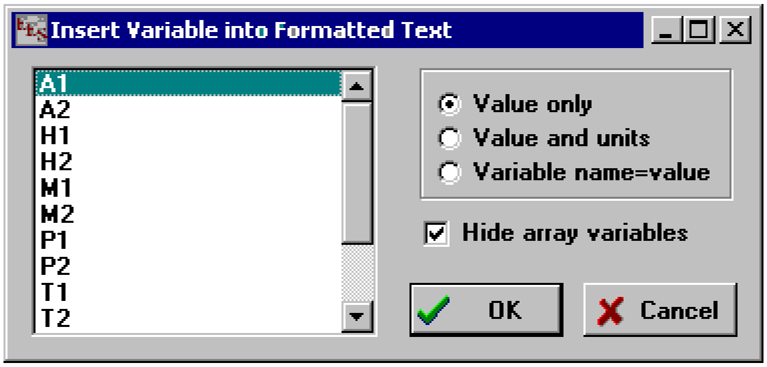
The Diagram window must be in Development mode in order to edit the Formatted text item. Select the item by clicking on it. Click again at the point that you wish to edit. Text can then be edited in the usual manner. Formatted options are available by clicking the right mouse button which will display the following popup menu.



Inserting EES Variable Information into a Formatted Text Item

The calculated value of one or more EES variables, optionally including its units and the EES variable name, can be inserted into a formatted text item while in Development mode. The contents of the text item are automatically updated as the EES variables are changed. This capability is similar to that available for Output variables in the Diagram Window. It differs in that it allows the calculated variable to be placed in a formatted paragraph or table, simplifying the construction of the Diagram window and facilitating report generation.

To insert a variable, place the cursor at the insertion point and click the right mouse button to bring up the menu shown above. Select the Insert Variable command which will display the following dialog.



● Value only
○ Value and units
○ Variable name=value

☑ Hide array variables

Select the EES variable you wish to insert from the list on the left. Typing the first letter of a variable will scroll the list to a variable beginning with that letter. Check the Hide array variables control to eliminate array variables from list. Indicate whether you wish to include the variable name and units with the EES variable by clicking the appropriate radio button. In Development mode, EES will display the inserted variable with a #, followed by 1, 2, or 3 (depending which of the three radio button options was selected) and then the EES variable name. For example, if variable XYZ is inserted with its value and units, it will appear as #2XYZ in the formatted text item. When the Diagram window is in Application mode, the insertion indicator will be replaced with the current EES variable information.

Deleting a Formatted Text Item

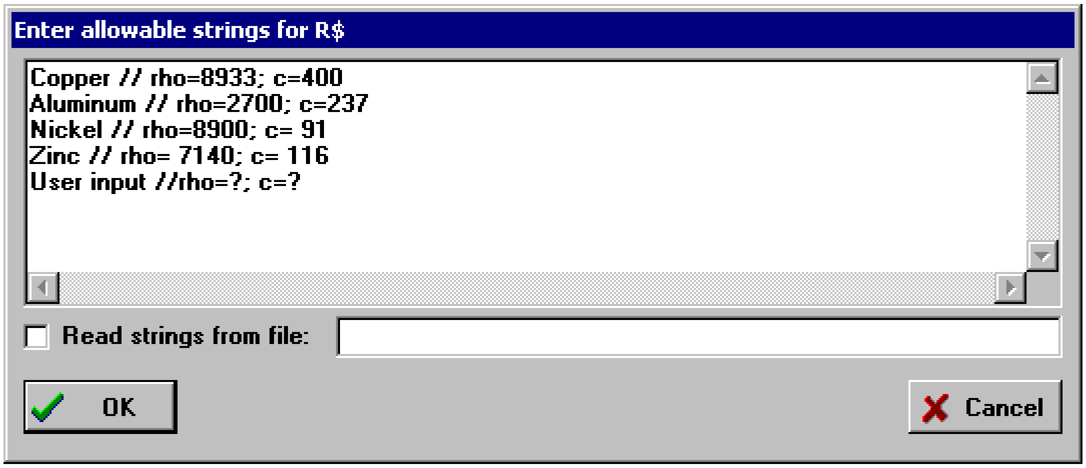
To delete a formatted text item, place the Diagram window in Development mode and select the item by clicking on it. Then press the right mouse button. A pop up menu will appear with Properties as one of the menu item choices. Select the Properties command. A dialog will appear with a Delete button. Clicking the Delete button will remove the formatted text item.

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. The EES equation can also be a $INCLUDE directive which add one or more lines of text to the Equations window. 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. Note that the strings could also be read from a file in the Professional version or set within Check Box or Radio Group controls.



Enter allowable strings for R$

```
Copper // rho=8933; c=400
Aluminum // rho=2700; c=237
Nickel // rho=8900; c= 91
Zinc // rho= 7140; c= 116
User input //rho=?; c=?
```

□ Read strings from file:

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. Copper ▾

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. An example of this capability is provided in the Diagram_In_Out.ees example file.

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 in the next section.

Clicking on the line button in the Diagram window tool bar allows a single line to be drawn. If you want do draw several lines in succession without having to click the line button for each one, hold the right mouse down while clicking the left mouse button on the line button. When you are done drawing lines, click the left mouse button on the toolbar line button or press the Esc key.

To change the characteristics of graphic items, set the Diagram window into Development mode. Then either 1) double-click on the item; 2) right-click on the item and select Properties from the pop-up menu or 3) select the item by clicking on it and then click on the appropriate button (line, ellipse, etc.) in the Diagram tool bar.

In the Professional version, the graphic item can be given a name. The name can then be used to assign attributes to the graphic item with EES variables in the Equation window or Parametric table. For example, the left and top of the item named Bottle could be specified with EES variables Bottle.left and Bottle.top, respectively. Objects return to the location specified in the dialog when the Diagram window is in development mode. In a similar manner, the width, height, angle, fill color and line color of the graphic item can be specified with EES variables, allowing programmatic control over the attributes of a graphic item. Logical uses for this capability are to move a figure (animation) or to show one of several figures while hiding the others based on the selection from a drop-down list.

Adding Polylines, Bezier Curves and Polygons (Professional Version)

Polylines, Bezier curves and Polygons are graphic items that can be drawn on the Plot or Diagram windows. The general method for placing these graphic items is similar to that for adding graphic items as detailed above. However, there are some differences:

Polylines and Bezier Curves

Click the the polyline button on the toolbar. The cursor will be shown as a plus sign. Move the cursor to the location that will be the first point. The graphic item can be moved later, as described below, so clicking the exact position is not a concern. After clicking on the starting point, move the cursor to the second point. A line will be displayed from the last point to the cursor location. Click to position the second point. Continue this process until all of the segments of the polyline are drawn. Double-click (or press the Enter key) to end the drawing process. If you wish to abort the drawing process, press the Esc key.

Once the drawing process is completed, the polyline object can be dragged to a new location and the individual points composing the polyline can be moved. To drag the entire polyline to a new position, press and hold the mouse button down near the center of any one of the line segments. The cursor will change from a pointer to the drag cursor indicator (a pointer attached to a box). While holding the left mouse button down, slide the mouse to a new location. The polyline will move with the cursor. Individual points in the polyline can be moved in a similar manner. Press and hold the mouse down over any one of the vertices of the polyline until the cursor changes to up-down-left-right arrows. While holding the left mouse button down, move the cursor to a new location. The position of the selected vertex will move with the mouse.

The properties of the polyline can be changed when the toolbar is visible by either double-clicking the mouse while it positioned on a line or vertex. Alternatively, position the mouse on a line or vertex and right-click. Select Properties from the pop-up menu. Additional points can be added to the polyline and a selected point can be deleted. To delete a point, move the mouse cursor over a point. It will be displayed as a filled box. Right-click on the point and a pop-up menu will appear. Select the Delete Point menu item. Follow the same procedure to add a point, but select the Add point menu item. The new point will be placed on the line connecting the selected point and the previous point. The new point can then be moved to a new location, if desired.

<u>Polygons</u>
Polygon graphic items are drawn by clicking on the polygon toolbar button . The drawing process is identical to the process for drawing a polyline, except that the drawing process is concluded by clicking on the first point, which is marked with a circle. Alternative, double-click the mouse buttor or press the Enter key and the cursor will move from the last entered point to the first point to form an enclosed area. The enclosed area will, by default, be shaded in lime color. The characteristics of the polygon can be changed by double-clicking anywhere within the enclosed area or by right clicking and selecting the Properties menu item from the pop-up menu. The line color, line type, fill color, and fill type can be specified. If the Transparent checkbox is selected, the graphic object will be display in a way that allows objects below it to be seen.

The polygon can be moved to a different location by pressing and holding the mouse button down anywhere within the polygon and sliding the mouse to a new location. Individual points can be moved by moving the cursor over the selected point and pressing the mouse button. Hold the mouse button down while moving the cursor to move the point to a new location. Right click on a point to bring up a pop-up menu that allows points to be added or deleted.

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 (or clicking the text button on the Diagram tool bar) 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.

Group and Ungroup Buttons

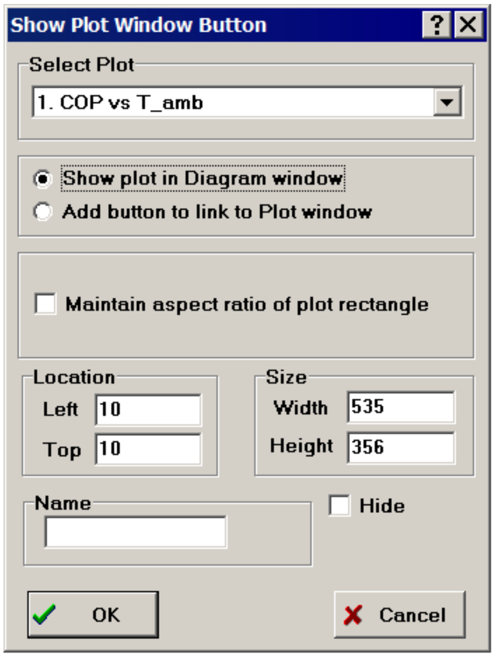
Two or more graphic and/or text items can be grouped in development mode. Groups can consist of graphic items (lines, circles, etc.), text (of all types), buttons, check boxes and radio button groups. When grouped, all items are selected when one item in the group is selected and all items move together. Right-clicking on a group provides access to its attributes, which allow it to be locked or hidden. A group can be given a name that allows its hide attribute to be controlled programmatically. Grouping is provided in the Professional version. HTo create a group, select the items that are to be in group and then click the Group button the toolbar. Alternatively, right click on one of the selected items and select the Group menu command from the pop-up menu that appears. Group information is saved with other file information when the Save command is issued. To ungroup a previously defined group, click on the grouped item and then click the Ungroup button on the toolbar. Alternatively, right click on the grouped item and select the Ungroup menu command from the pop-up menu.

Adding a Calculate Button

One or more Calculate buttons can be placed on the Diagram or (child Diagram) window by selecting the Add 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. The position and size of the button can also be specified in this manner. In the Professional version, the Calculate button can be given a name. The name can be used to dynamically control the left,, top and hide attributes of the Calculate button.

Adding a Plot Window Access Button

A 'live' plot and/or a link button to a particular Plot Window can be placed on the Diagram window or child Diagram window by selecting the Display Plot Window button in the toolbar.



Show Plot Window Button

Select Plot: 1. COP vs T_amb

● Show plot in Diagram window
○ Add button to link to Plot window

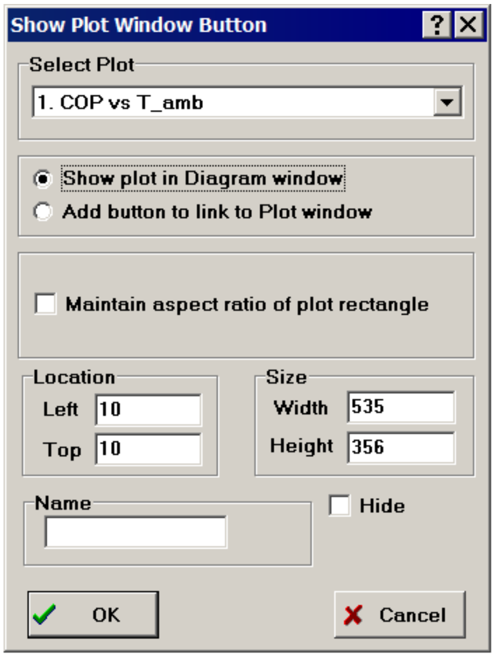
□ Maintain aspect ratio of plot rectangle

Location: Left 10, Top 10

Size: Width 535, Height 356

Name:

□ Hide



Show Plot Window Button

Select Plot: 1. COP vs T_amb

● Show plot in Diagram window
○ Add button to link to Plot window

□ Maintain aspect ratio of plot rectangle

Location: Left 10, Top 10

Size: Width 535, Height 356

Name:

□ Hide

The dialog window shown on the left will appear after clicking this toolbar button. The plot that is to be displayed (or linked to) is selected from the drop down list at the top of the dialog. The names appearing in this list are the names on the tabs of the Plot window. The dialog window provides two radio buttons. If the first radio button is selected (as shown on in the Dialog window on the right), the selected plot will be displayed in the Diagram or child Diagram window. The location and size of the plot can be specified using the edit controls at the bottom of the dialog. In addition, the plot can be dragged to a desired location and resized in the same manner as any other graphic items while in the Development mode. If the Maintain aspect ratio check box is selected, the height to width ratio of the plot will be maintained when the plot is resized.

If the Add button to link to Plot window radio button is selected (as shown in the left dialog), a button will be placed on the Diagram window. Clicking this button in Application mode will open and display the selected plot window. If the plot window does not exist, this button will do nothing. The location and size of the plot window access button can be specified in the plot window button dialog. Alternatively, the plot window access button can be dragged to any location when then toolbar is visible. Finer positioning is available by pressing the left, right, up, or down arrow keys which move the button one pixel for each press. The width and height of the button can be changed by holding the Ctrl key down while pressing arrow keys. The right arrow increases the button width by one pixel. Left arrow decreases the width. Down and up arrows affect the height.

Right-clicking on the plot or the plot window access button when the toolbar is visible (i.e., in Development mode) will bring up the dialog so that the characteristics of the plot access button (name, glyph, or position) or the plot position and size can be changed. The Professional version allows a picture that is stored in a bitmap (.bmp) or JPEG (.jpg) file to be displayed on the button. The picture option is selected with the drop-down control below the caption. A standard file selection dialog box to facilitate selection of the picture file will appear when the picture option is chosen. The Professional version allows a Name to be associated with the plot button (left figure) or plot display (right figure). For example, the name has been set to plotbutton1 in the above figures. If a name is provided, EES will generate variables name.left, name.to, and name.hide. These variables can be set in an EES program or in the Diagram window. One example of this capability is to show or hide the plot or plot button by setting name.hide=true# or name.hide=false#.

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.

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

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. Use the arrow keys to reposition the button and the Ctrl arrow keys to resize the button.

Creating Links (*Professional Version*)

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 that will appear on the foremost Diagram window. An option is provided for a transparent button which can be placed anywhere in the Diagram or child Diagram windows. 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. Six different types of links can be created:

<u>Link type 1: Start an External Program</u>

This link type allows any program, including another instance of EES, to be started when the user clicks the link button in the Diagram window while in application-mode. The link button property can be the full filename and optionally, the name of the file that is to be opened when the application starts. Alternatively, just the name of the file that is to be opened is needed if the application information is provided in the operating system registry, as usually the case. As an example, the following link property will start a second copy of EES and load file TEST.EES.

C:\EES32\TEST.EES

EES can communicate with other programs using Lookup files (Chapter 4). 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.

<u>Link type 2: Open EES File</u>

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 100 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 Diagram window, so it no necessary to click on the hot area for the child Diagram 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.

Link type 6: Open a Lookfile File
A Link button can be placed on the Diagram window that, when clicked, will display the familiar Open File dialog. A user can then select a Lookup file having a filename extension of .LKT, .TXT, or .CSV and that Lookup file will be opened. After the selection has been

made, an EES string variable having the same name as the caption of the Link button is created and its value is set to the name of the Lookup table that has been read in. EES does not allow spaces in a variable name so any spaces in the caption are set to underscores. Also, the maximum length of the variable is 29. The last character of the EES variable name is set to $ to indicate that it is a string variable. As an example, if the caption of the Link button is "Open Lookup File", clicking the button will read the file, create variable Open_Lookup_File$ and set the value of this variable to the name of the Lookup table that was created. If the Lookup table already exists, it will not be re-read.

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 or moved with the arrow keys. 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. In this dialog, the Link button can be given a name. The name is used to dynamically control the button attributes. For example, if the button name is Link1, EES will create variables Link1.Hide, Link1.left, and Link1.top. These variables can be set in the EES program or in other parts of the Diagram window, such as in the code that accompanies Check Boxes, Radio Groups, or Drop-down lists. The button size can be set in the Link button dialog. Alternatively, the Ctrl-arrow keys allow adjustment of the height and width of the button.

Adding a Help Button

One or more Help buttons 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 Add Help button icon 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. In the Professional version, the Help button can be given a name. The name is used to dynamically control the button attributes. For example, if the button name is Help1, EES will create variables Help1.Hide, Help1.left, and Help1.top. These variables can be set in the EES program or in other parts of the Diagram window, such as in the code that accompanies Check Boxes, Radio Groups, or Drop-down lists. The help button can be removed by right-clicking on the button in Development model and then clicking the Delete button. If the file name provided for a Help button for the Diagram window begins with http://, EES will start up the default browser program when the button is clicked and point it at the URL provided as the file name. With this capability, Help in a Diagram window can reside on a website.

Navigating through Child Diagram Window (*Professional Version*)

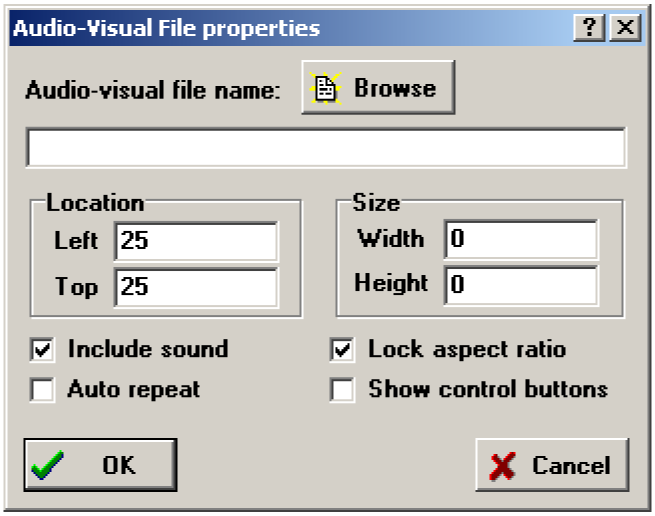
The main Diagram window can have 'hot areas' which bring up child Diagram windows when clicked. The child Diagram window can also have 'hot areas' forming grandchildren and so on. Each child Diagram window will have a small 'home' button which, by default, is initially located in the lower right corner of the window. 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. The location and size of these navigation buttons can be changed using the arrow and Ctrl-arrow keys, respectively. A caption can be added to either button by right-clicking on the button in development mode.

Adding a Print Button

One or more Print buttons 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 ‘Add Print button’ icon 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 location of the button can also be changed with the arrow keys. The caption and button location and size can be changed by right-clicking on the button to bring up the Print Button dialog. In the Professional version, the Print button can be given a name and variables name.left, name.top, and name.hide are then created. These variables can be set in the Equations window or in other Diagram window controls. The Print button can be configured in two ways. It will either print the Diagram window (with no confirmation) or it will bring up the Print dialog as if the Print command in the File menu were selected.

Adding an Audio-Visual Item

Audio-visual items such as movie files (having an .avi, .mpg, or .wav file name extension) or sounds (having a .wav filename extension) can be placed on the Diagram window or child Diagram windows by selecting the Add Audio-Visual item button in the toolbar. The following dialog window will appear after clicking this toolbar button.



Audio-visual file name:

Location
Left 25
Top 25

Size
Width 0
Height 0

☑ Include sound
☐ Auto repeat
☑ Lock aspect ratio
☐ Show control buttons

The Browse button facilitates selection of the audio-visual file. Any file that following the MDI interface should be acceptable, including files that have a .avi or .wav file name extension. If a movie file is selected, EES will automatically fill in the Width and Height of the movie. These fields can be changed. If the Lock aspect ratio button is selected, changing either width or height will result in an automatic change in the height or width to ensure that the original aspect ratio remains unchanged.

If the Show control buttons is checked, a control bar ▶ ‖ □ |▶ ◀| will be placed below the audio-visual item. The bar has the familiar play, pause, stop, step, previous control buttons. If the control buttons are not selected for display, then the movie is controlled by the left and right mouse buttons. To play, click the left mouse button within the item rectangle while in Application mode. To pause a playing item, click the left mouse button within the item rectangle. Clicking the right-mouse button stops a playing move and resets it to the beginning.

The audio-visual control information can be viewed or changed by right-clicking in the item rectangle while in Development mode.

Saving and Loading User Inputs (*Professional Version*)

Save and Load Inputs buttons can be placed on the Diagram window (but not on a child Diagram window) by selecting the “Add Save Inputs” or “Add Load Inputs” button 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 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.

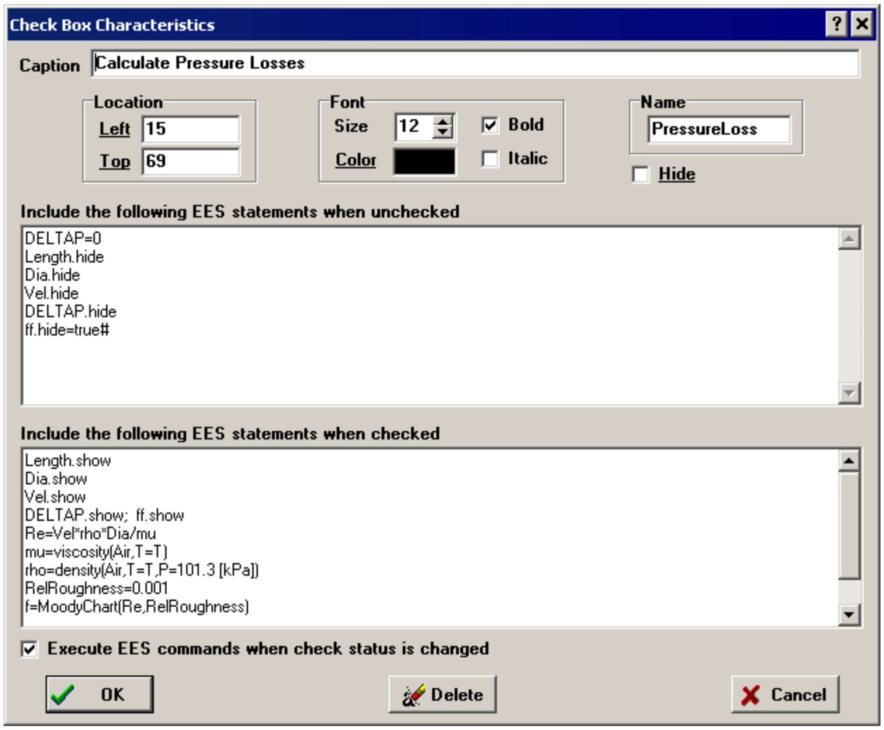
When opening one of the five EES files contained in the Distributable program, EES will look for a .VAR file in the same directory the Distributable program having the same parent name as the EES file. If a .VAR file is found with this parent name, automatically loaded and the input information on the 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. A dialog will appear in which the caption, size, location and name of the button can be specified. If the button is given a name, the button attributes can be controlled by assigning name.hide, name.left and/or name.top with EES variables in the Equations window or in Check Box, Radio Button or drop-down Diagram window controls.

Creating and Using Check Boxes (*Professional Version*) ☑

A check box can be placed on the Diagram window using the Add Check Box button on the Diagram window toolbar (Professional Version). A check box allows the equation set that EES solves to be modified by the check box setting. After selecting the Add Check Box button on the toolbar, the setup checkbox dialog will appear that allows the caption text, location on the Diagram window, and font to be specified. The location can be later changed by dragging the check box to the desired location while the Diagram window is in development mode.



Caption: Calculate Pressure Losses

Location — Left: 15; Top: 69

Font — Size: 12; Color; ☑ Bold; □ Italic

Name: PressureLoss

□ Hide

Include the following EES statements when unchecked

```
DELTAP=0
Length.hide
Dia.hide
Vel.hide
DELTAP.hide
ff.hide=true#
```

Include the following EES statements when checked

```
Length.show
Dia.show
Vel.show
DELTAP.show;  ff.show
Re=Vel*rho*Dia/mu
mu=viscosity(Air,T=T)
rho=density(Air,T=T,P=101.3 [kPa])
RelRoughness=0.001
f=MoodyChart(Re,RelRoughness)
```

☑ Execute EES commands when check status is changed

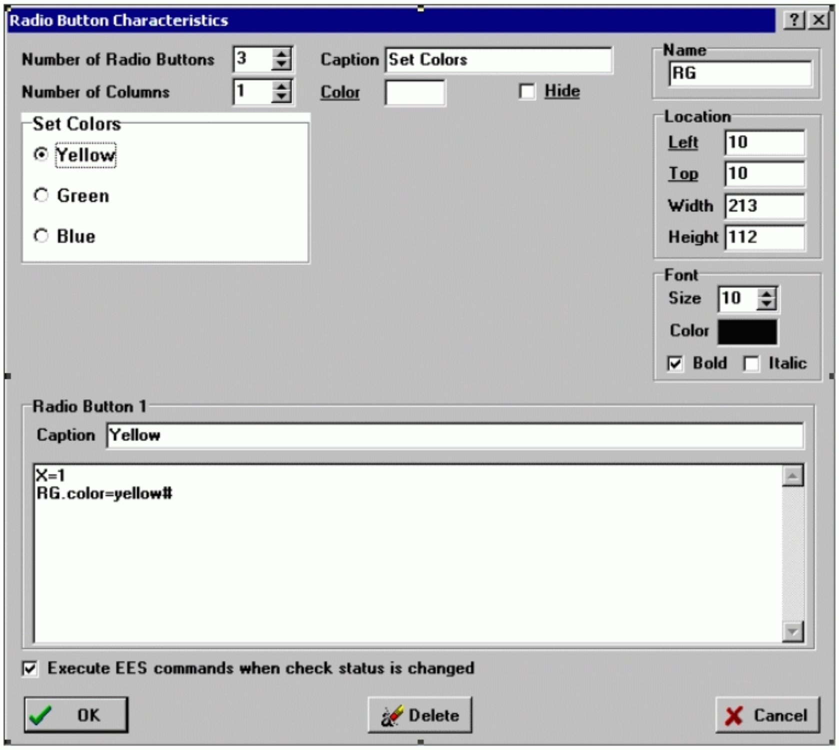
The checkbox can be given a name. If a name is provided, the attributes shown with underlined text can be set with EES variables. For example, the left position of the check box in the example below could be specified by setting the value of PressureLoss.left. In the following example, several input and output text items have been defined and been given names. These text items are either hidden or shown depending on the check box setting. The check box dialog provides two windows in which EES commands (comment, equations and directives) can be entered, just as if they were entered in the Equations window. If the check box is unchecked, the commands in the first window are included with other commands in the Equations window when calculations are initiated. If the check box is checked, the commands in the second window are included.

If "Execute EES commands when check status is changed" is selected, EES will attempt to execute the commands in the unchecked or checked window, depending on the setting of the check box, directly after the change is made to the check box. This capability is most useful when the check box setting affects the display in the Diagram window in some way. In the example below, some text inputs and outputs are immediately hidden if the check box to calculate pressure losses is not selected. Graphical objects and buttons can be hidden or positioned in this manner as well.

To change the information in a check box, set the Diagram window into Development mode, and then either double-click on the check box or click on it (to select it) and click the right mouse button or the Check box tool on the Diagram tool bar.

Creating and Using Radio Groups (*Professional Version*)

A radio button group is placed on the Diagram window using the 'Add Radio Group' button on the Diagram Window Toolbar (Professional version). A radio button group provides up to 8 radio buttons. Each button is associated with a set of EES commands that are enabled when the corresponding radio button is selected. The following dialog window will appear after clicking the toolbar button.



Radio Button Characteristics

Number of Radio Buttons: 3
Number of Columns: 1
Caption: Set Colors
Color
□ Hide

Name: RG

Location
Left: 10
Top: 10
Width: 213
Height: 112

Font
Size: 10
Color
☑ Bold □ Italic

Set Colors
● Yellow
○ Green
○ Blue

Radio Button 1
Caption: Yellow

```
X=1
RG.color=yellow#
```

☑ Execute EES commands when check status is changed

The number of radio buttons is set with the spin control, or by entering the value. The number of buttons can be between 2 and 8. The number of columns controls how the buttons are positioned. The number of columns can be between 1 and 4. An optional group caption can be entered for the radio button group. The background color, width, height, location, and font characteristics can also be specified. Changing any of these controls will cause the group box in the dialog window to be redrawn as it will appear in the Diagram window. Each button is associated with EES commands. The caption and EES commands for the currently selected button appear at the bottom of the dialog. Both the caption and the EES commands for this button can be modified.

The radio group can be given a name. If a name is provided, the characteristics identified with underlined text, e.g., left, top, color, and hide, can be set with EES variables. In the example above in which the name is RG, setting EES variable RG.Color would change the background color of the radio button group.

If "Execute EES commands when check status is changed" is selected, EES will attempt to execute the commands associated with the selected button directly after it is selected. This capability is most useful when the check box setting affects the display in the Diagram window in some way. The frivolous example above sets the background color of the radio group to different colors depending on the selected button. A more useful application of the radio button capability would be to control the position and visibility of objects, text, and buttons on the Diagram window. For example, a Calculate button can be hidden by setting the EES variable, Name.Hide=true# where Name is the name given to the Calculate button in the Calculate Button Characteristics dialog.

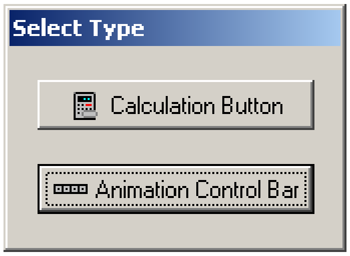
After clicking the OK button the radio button group will appear in the Diagram window. In Development mode, the radio button group can be dragged to a new position by pressing and holding the mouse on the group border will moving the group to a new location. If the mouse is pressed on the lower right corner of the radio button group the cursor will change to a resize arrow cursor. Dragging the mouse at this point will allow the width and height to be changed. Right-clicking (or double-clicking the left mouse button) on the border of the radio button group in Development mode brings up the Radio Button Characteristics dialog so changes can be made. Alternatively, click the radio button group to select it and then click the radio group button on the Diagram tool bar.

Animation (*Professional Version*)

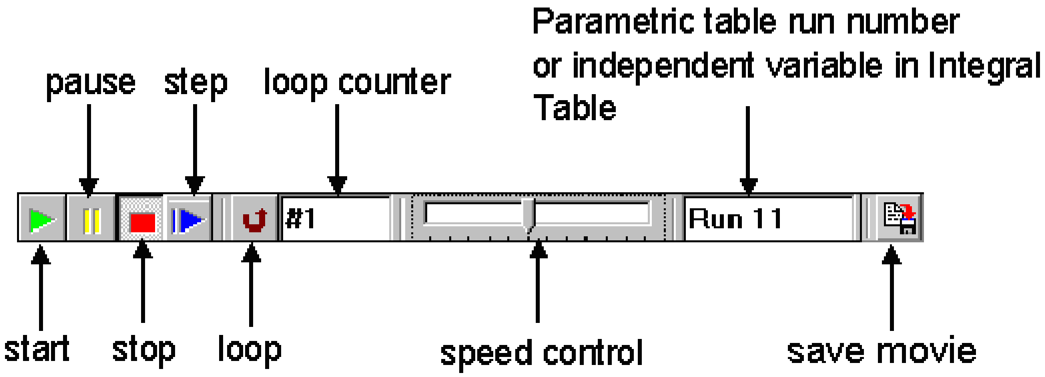
The Professional version of EES provides powerful capabilities to animate drawing and plots in the Diagram and Plot windows. Unlike many other animation programs that provide capability to create animated drawings, EES allows the animation to be controlled by calculated values. The animation in the Diagram window can thus be accurately displayed and directly coupled to information in the plots. Animations can only be created by the Professional version, but once created, they can be displayed in any version of EES, including distributable programs.

Creating an animation with EES is very simple. The first step is to draw the object in a the Diagram (or child diagram Window). Alternatively, the object can be created in an external program such as Power Point and pasted into the Diagram window. The object must be given a name in the name field of the object properties dialog. Then equations are added into the Equations window to specify the position, size, or color of the object or text. The position and size are specified in pixels with 0,0 being the upper left corner of the window. EES provides functions DiagramHeight#, DiagramWidth#, and PixelsperInch# to allow the position and size to be computed in other units and converted to the pixels needed in the Diagram window.

The final step is to add an Animation control bar to the Diagram window. This is done by selecting the Add Calculate button in the Diagram Window Toolbar when the Diagram window is in Development mode. The following dialog will appear.



Click the Animation Control Bar button to place an animation control bar on the Diagram window.


Parametric table run number or independent variable in Integral Table
pause
step
loop counter
#1
Run 11
start
stop
loop
speed control
save movie

The functions of the controls on the animation bar are indicated above. The bar can be dragged to a desired location by pressing and holding the mouse button down anywhere on the first four buttons and sliding the mouse to a new location. The Save movie button will allow each frame of the animation to be saved as a .jpg, .bmp, or .emf file.

As a simple example, we will animate the position of a ball in the Diagram window. Bring the Diagram window to the front and set it into Development mode. Click the circle tool on the toolbar and create a red ball. Enter the name of the object to be Ball, as shown.

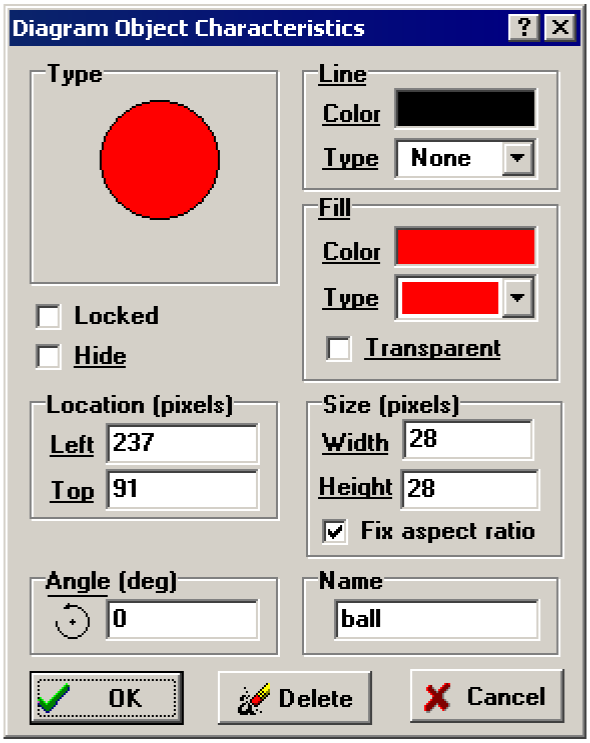


Diagram Object Characteristics

Type

□ Locked
□ Hide

Location (pixels)
Left 237
Top 91

Angle (deg)
0

Line
Color ■
Type None

Fill
Color 🟥
Type 🟥
□ Transparent

Size (pixels)
Width 28
Height 28
☑ Fix aspect ratio

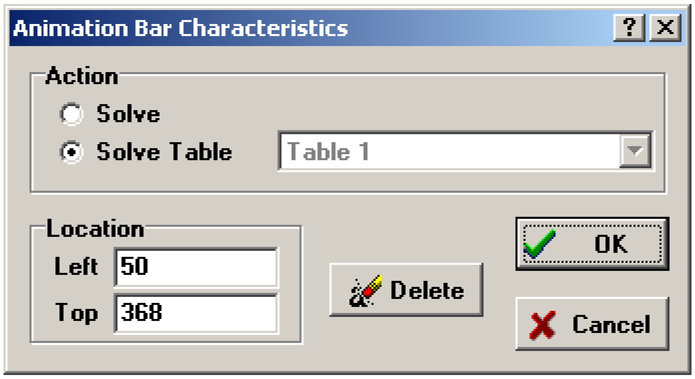
Name
ball

In the Equations window, enter the following equations.

ball.left=X
ball.top=150 + 100*sin(X)

Create a Parametric table with columns X, ball.left, and ball.top and 360 rows. Enter values for X between 0 and 360.

Click the Add Calculate button and select the Animation bar. Right-click on the start button in the animation bar to bring up the animation characteristics dialog. Select the Solve Table button.



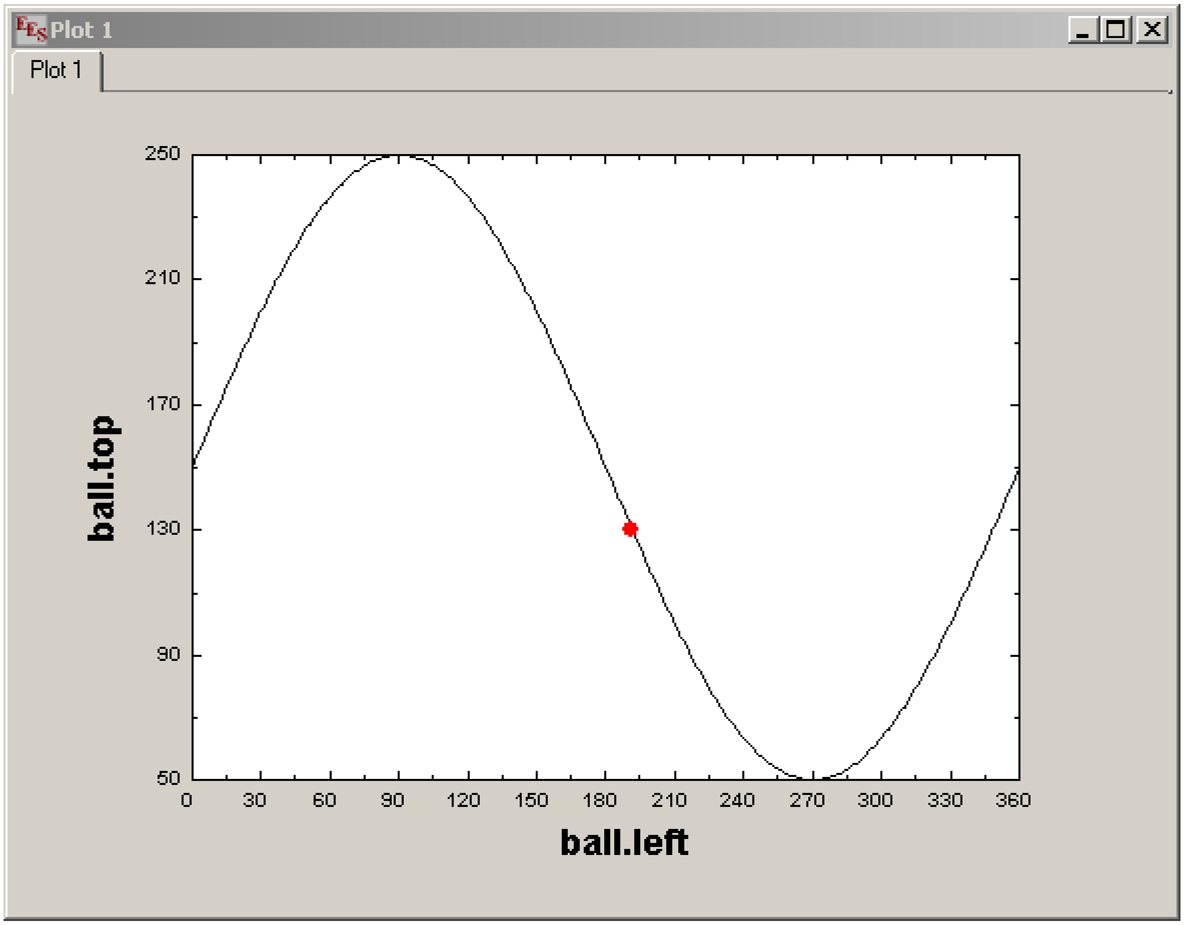
Animation Bar Characteristics

Action
○ Solve
● Solve Table Table 1

Location
Left 50
Top 368

Hide the Diagram toolbar so that the Diagram window is now in application mode. Click the start button and watch the show. Text and lines can also be animated in this way.

The animation can be coupled to a plot. Crate a new plot with the x-variable set to Ball.left and the y-variable set to Ball.top. Right click on the plot and select the Automatic Update checkbox. Start the animation and watch the plot. Its EESy to create animations.



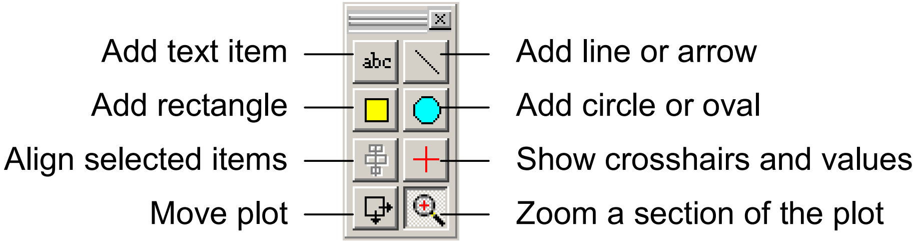
Animations generated in the Diagram window (Professional version) can be saved in a sequence of .EMF files that can be assembled into a stand-alone movie with Microsoft Movie Maker or other software. This capability is enabled by clicking the Save Movie button on the animation control bar.

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. The New Plot Window command allows 2-D and 3-D plots to be created. In addition, 2-D 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 tabbed plot windows that can be constructed (other than available memory), and each 2-D plot may have any number of overlaid plot lines. A hint appears when the mouse is moved over that tab on a plot or table window. The hint normally contains the descriptive text that can be added when you right-click on the tab. If no descriptive text has been added, the hint for the plot window is the names of the variables that are plotted on the X and Y axes. The hints make it easier to find the plot or table when many have been created.

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, shown below, is provided on 2-D plots with buttons to add text, lines, boxes, and ellipses. An alignment button on the tool bar allows 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 a 2-D plot can be changed in many ways using the plot menu commands in the Plot menu and controls in the Plot window.

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 by clicking the move plot button in the plot window toolbar or by holding the Ctrl key down. Either action is then followed by pressing the mouse button within the plot rectangle and dragging the mouse to move the plot to the desired location in the window.

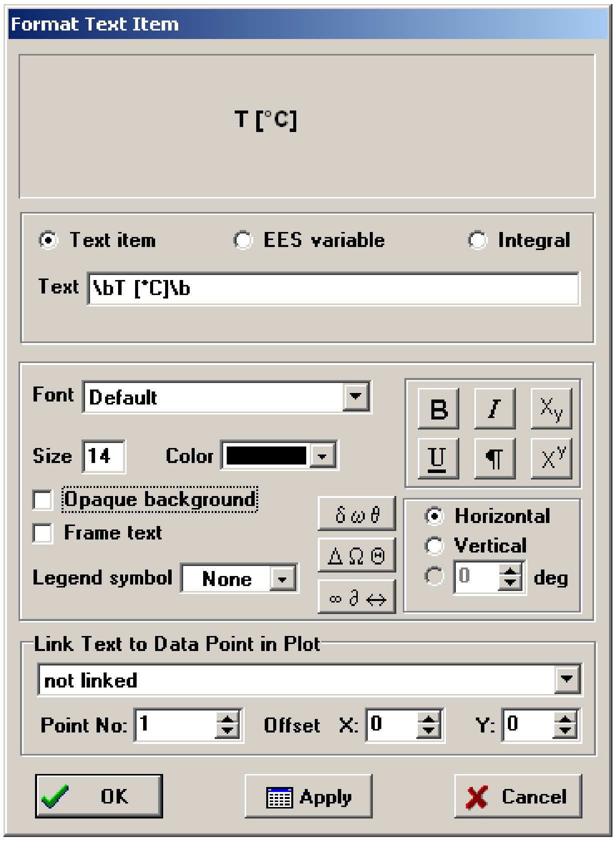
Adding / Changing Text and Text Characteristics abc

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. Three basic types of text items can be created: plain text, EES variables, and integral. 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. If the Integral button is selected, EES will create a text item holding the integral of a selected line on the plot, with the option of displaying its units. If Auto Update is selected for the plot and the values being plotted change, the integral will also change accordingly.

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. For example, to set some or all of the text to have a bold style, select the text and the click the **B** button. The ***I*** and **U** buttons set italic and underline styles. The ¶ button places a linebreak. Subscripts and superscripts can be generated using the appropriate buttons after selecting the text that is to be subscripted or superscripted. 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.

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.

Labels can be associated with plotted points for X-Y and bar plots. The labels are initially positioned to the right of the point and move with the point if the axes are scaled. The relative position of the label to the point is maintained if the label is manually moved. In the Commercial version, the labels that can be automatically generated are the numbers corresponding to the run or row numbers (for the Parametric and Lookup tables) and the subscript index number (for Arrays tables). The Professional version extends this capability by allowing the labels to be any strings that are displayed in a column of the table. The point label option is selected when a new plot or plot overlay is created by checking the 'Add point labels' check box and selecting the string variable column from the menu below the check box control.



● Text item ○ EES variable ○ Integral

Text: \bT [°C]\b

Font: Default

Size: 14 Color: ■

□ Opaque background

□ Frame text

Legend symbol: None

● Horizontal ○ Vertical ○ 0 deg

Link Text to Data Point in Plot: not linked

Point No: 1 Offset X: 0 Y: 0

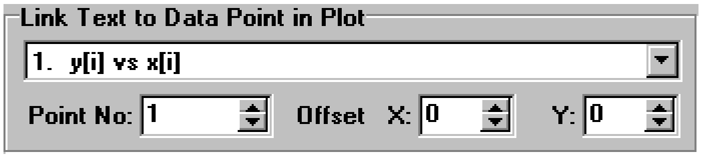
When you click on a text item in a plot window when the tool bar is visible, a red dotted box will be drawn around it to indicate that this item is selected. You can select multiple text items by holding the Shift key down as you click on the text items. Alternative, a rectangular area within the plot window can be selected when the tool bar is visible by pressing the mouse at the upper left corner and dragging it to the lower right corner. All text and graphic objects within this rectangle will be selected when the mouse button is released. 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.

Labels can be associated with plotted points for X-Y and bar plots. The labels are initially positioned to the right of the point and move with the point if the axes are scaled. The relative position of the label to the point is maintained if the label is manually moved. In the Commercial version, the labels that can be automatically generated are the numbers corresponding to the run or row numbers (for the Parametric and Lookup tables) and the subscript index number (for Arrays tables). The Professional version extends this capability by allowing the labels to be any strings that are displayed in a column of the table. The point label option is selected when a new plot or plot overlay is created by checking the 'Add point labels' check box and selecting the string variable column from the menu below the check box control.

The Professional version of EES will also show the following information in the Format Text Item dialog.



Link Text to Data Point in Plot

1. y[i] vs x[i]

Point No: 1 Offset X: 0 Y: 0

These controls can be used to position a text item relative to data point in a selected plot. If the data change and Automatic Update is selected for the plot, the text item will move with the data. This option is particularly convenient when plotting arrays. Selecting Show Array Indices in the New Plot Setup or Overlay Plot dialogs will automatically generate a text item for each plotted array point containing the number of the data point for identification.

Note that the text item is initially placed just to the upper right of the data point. However, the text item can be moved to a different location by clicking on it to select it and then dragging it with the mouse or using the arrow keys. Alternatively, the X and Y offset values (in pixels) can be set. (These offset values are referenced to the default value that EES uses when it first places the test to the upper right of the data point.) If the text item is linked to a data point, its poistion relative to the data point will be maintained, even if the data point is subsequently moved.

Moving Text

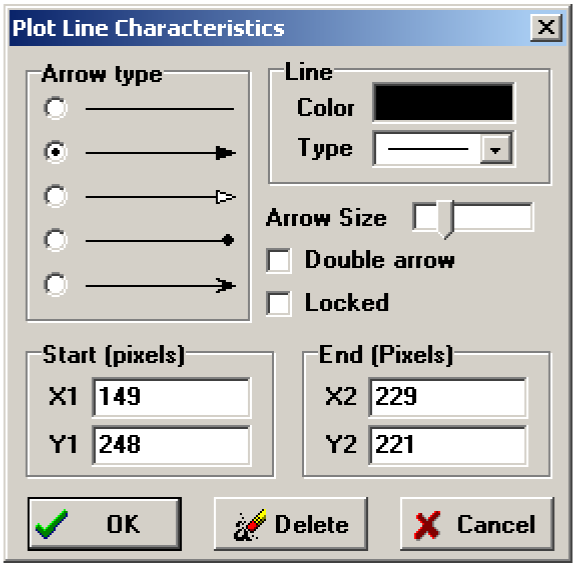
Text fields to label the X and Y axes are automatically generated when a plot is constructed. Additional text items can be added to the plot using the Add Text command in the Plot menu. You can move any text item or set of selected text items to any position on the screen by pressing and holding the 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 Graphics and Text from the Clipboard

Graphic and text items that have been copied to the Clipboard from other applications in the Windows metafile format can be pasted into the Plot Window using the Paste command in the Edit menu when the plot window tool bar is visible. After an item has been pasted, it can be moved and sized, just like any of the other items in the Plot window. With this capability, it is possible to, for example, place equations from Mathtype in the Plot Window or place a small version of another plot within a Plot. The easiest way to place a plot within a plot is to copy the Plot Window to the Clipboard and then Paste it into PowerPoint or Word. Rescale it and then paste it back into the EES Plot window. Two or more instances of EES may be open at the same time. In this case, graphic and text items be copied from one instance and pasted into another. International characters can be entered into text items that appear in the Plot Windows. See the online help for details.

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 drawn at an angle of 0°, 30°, 45°, 60° or 90°. 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.



Plot Line Characteristics

Arrow type: ○ ● ○ ○ ○

Line Color: ■

Line Type: ———

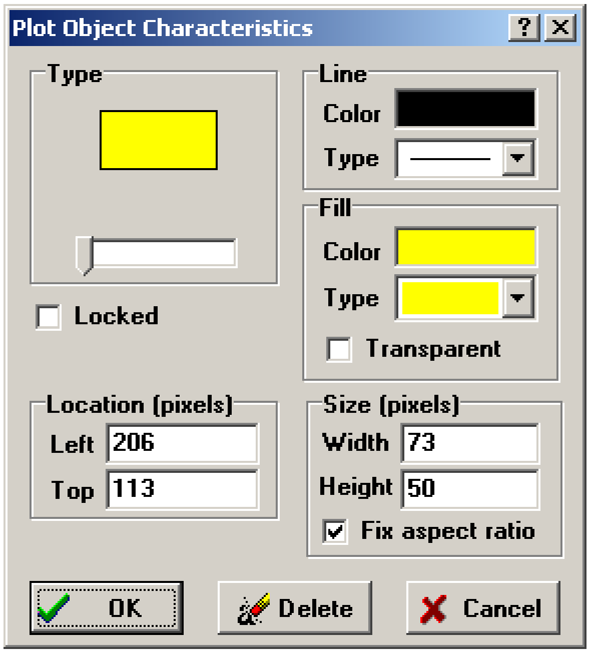
Arrow Size

□ Double arrow

□ Locked

Start (pixels) X1 149 Y1 248

End (Pixels) X2 229 Y2 221



Plot Object Characteristics

Type

□ Locked

Line Color: ■

Line Type: ———

Fill Color: 🟨

Fill Type: 🟨

□ Transparent

Location (pixels) Left 206 Top 113

Size (pixels) Width 73 Height 50

☑ Fix aspect ratio

Moving Lines and Arrows, Rectangles and Circles

Clicking on a line when the plot window tool bar is visible will cause it to be displayed with small boxes at both ends. Multiple lines can be selected by holding the Shift key down or by using the mouse to draw a selection rectangle around the objects. 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. The position and length of the line can also be controlled by entering values for the Start and End position in the plot line dialog.

Zooming In on an Area in the Plot 🔍

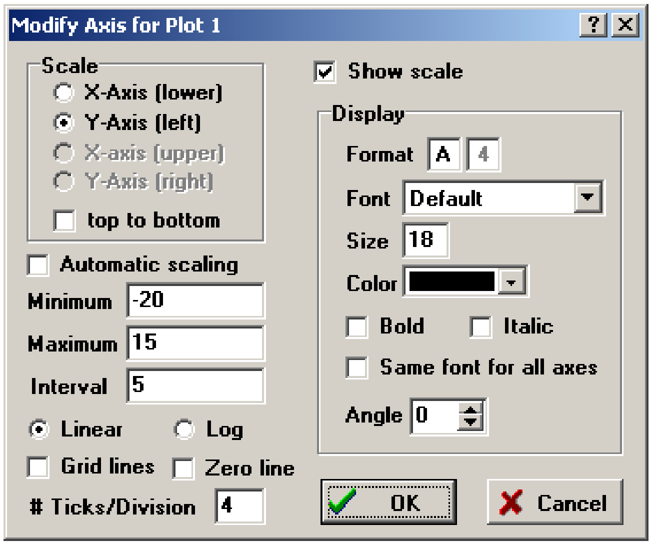
A rectangular area within the plot window can be selected by pressing the mouse at the upper left corner and dragging it to the lower right corner. If the Shift key is held down during this process, the selected rectangle will have the same aspect ratio as the plot rectangle. If the plot window toolbar is visible, all text and graphic objects within this rectangle will be selected when the mouse button is released. If the plot window toolbar is not visible or if the Zoom Selection button on the plot tool bar has been pressed, the plot information within selected rectangle will be drawn with the enlarged scale in a new plot window. The axes and any other information in this zoomed plot can be adjusted in the same manner as for any other plot window.

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. Enter 4 ticks/division to break the major division into 5 pieces. 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 (lower)
● Y-Axis (left)
○ X-axis (upper)
○ Y-Axis (right)
□ top to bottom

□ Automatic scaling
Minimum -20
Maximum 15
Interval 5
● Linear ○ Log
□ Grid lines □ Zero line
Ticks/Division 4

☑ Show scale

Display
Format A 4
Font Default
Size 18
Color
□ Bold □ Italic
□ Same font for all axes
Angle 0

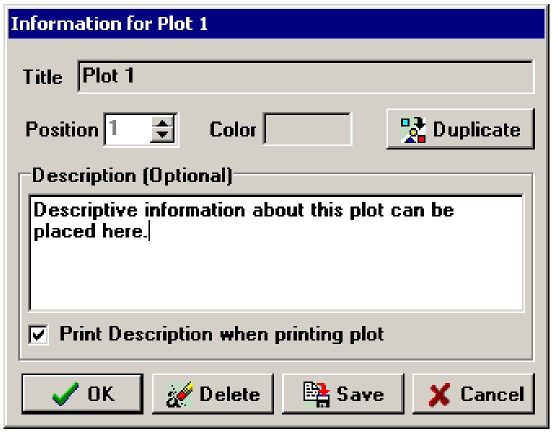
The X and Y-axis scaling can also be changed without using the Modify Axis dialog. As the mouse cursor passes over an axis of the plot, its appearance will change to a left-right or up-down arrow. Holding the left mouse button down at this point will allow the axis to be dragged to a new location.

Duplicating or Deleting a Plot Window

An entire plot window can be duplicated or deleted by right-clicking on its tab at the top of the plot window. A small dialog will appear having a Duplicate and a Delete button. The duplicate plot provides a convenient means of creating a plot template so that all of your plots have the same appearance. The Delete cannot be undone so be careful.

Saving the Plot

All plots are automatically saved with the EES file. However, in some circumstances, you may wish to save a file just containing the plot. Right-clicking the plot tab at the top of the plot window will bring up the dialog window below. Clicking the Save button in will allow you to save the specified plot as a TIFF (.TIF) file, bitmap (.BMP), JPEG (.JPG), Windows Metafile (.WMF), or Windows Enhanced Metafile (.EMF).



Information for Plot 1

Title: Plot 1

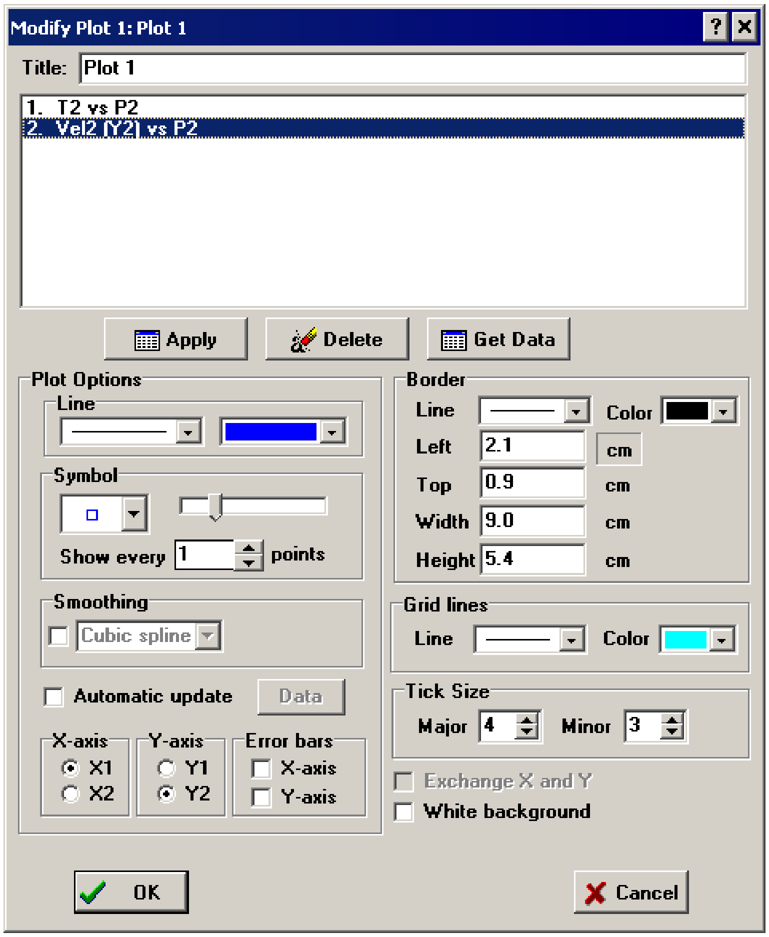
Position: 1

Description (Optional): Descriptive information about this plot can be placed here.

☑ Print Description when printing plot

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. Select the plot by clicking on its name.



Title: Plot 1

1. T2 vs P2
2. Vel2 [Y2] vs P2

Show every 1 points

□ Cubic spline

□ Automatic update

X-axis ● X1 ○ X2

Y-axis ○ Y1 ● Y2

Error bars: □ X-axis □ Y-axis

Border: Left 2.1 cm, Top 0.9 cm, Width 9.0 cm, Height 5.4 cm

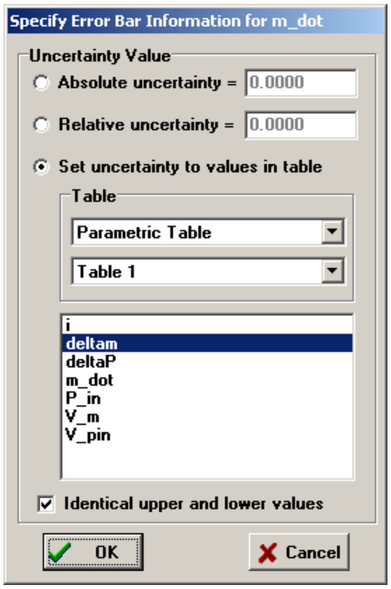
Tick Size: Major 4, Minor 3

□ Exchange X and Y

□ White background

EES allows plots to be created that use the right-hand y-axis (Y2) and a top x-axis (X2) scale. The scales in use for the selected plot are shown at the lower left of the dialog in radio button controls that allow changes to be made. 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 error bars control allows error bars to be placed in the horizontal, vertical or both directions. Error bar information can come from one of two sources. If calculations are initiated with the Uncertainty Propagation command, the error bar information is assumed to be the calculated or specified uncertainty values for the variables plotted on the X and Y axes. If the Uncertainty Propagation Table command is not used, clicking the X-Axis or Y-axis check box in the Error bars box will bring up the Specify Error Bars dialog in which the data needed to draw symmetric or unsymmetrical error bars is entered. Click the Apply button if you wish to view the changes you have made.

When there is more than one plot line within a plot window, the plots are drawn in the order displayed in the plot line list in the Modify Plot dialog. The order can be changed by selecting a plot line and dragging it to the desired location with the right mouse button. When the order is changed, reordered will be displayed below the left corner of the plot line list. The reordering is not actually done until the Apply or OK buttons are clicked.



Specify Error Bar Information for m_dot

Uncertainty Value

○ Absolute uncertainty = 0.0000

○ Relative uncertainty = 0.0000

● Set uncertainty to values in table

Table

Parametric Table

Table 1

i
deltam
deltaP
m_dot
P_in
V_m
V_pin

☑ Identical upper and lower values

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

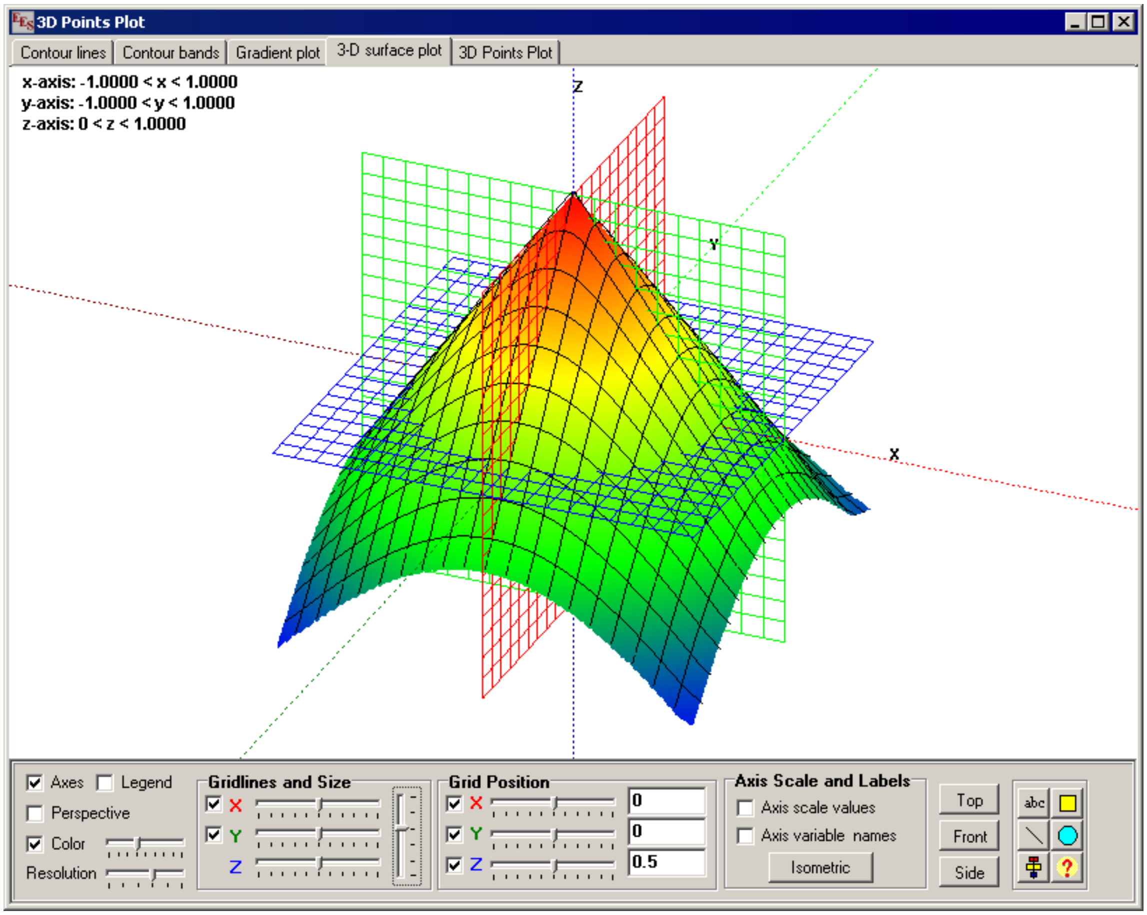
Clicking the crosshairs button on the plot window toolbar or holding the ***Shift*** and ***Ctrl*** keys down will change the cursor into crosshairs. The coordinates at the intersection of the crosshairs can be viewed in the status bar at the bottom of the plot window. If the plot was generated using the Properly Plot command in the Plot menu, the values of all thermodynamic properties corresponding to the state fixed by the crosshairs will be displayed in the plot window title bar. In the Professional version, the crosshairs can be used to identify the location of a plotted point in a table. Position the crosshairs over a plotted symbol and press the right mouse button. The status bar will show the coordinates of the point, the table that the data originated in and the row in that table. Move the crosshairs out of the plot rectangle to exit the crosshairs display.

Displaying Plot Windows

Pressing the left and right arrow keys in the Plot Window displays the previous or next plot, respectively, provided a text or graphic item is not selected. If an item is selected, the arrow keys move the item by one pixel.

3-D Plot Controls

The three-dimensional plotting package is based on a modification of the public domain GLScene package (http://www.glscene.org/). An example 3-D plot window is shown below. The image is generated with the New Plot|X-Y-Z plot menu command. The image in the plot window can be rotated around any axis by clicking the mouse on the image and dragging it to a new location. The image can be colored or gray scale. In either case, the color can be used to represent the values of any variable. If no specification is made, the colors are chosen relative to the z-axis variable. The 3-D plotting controls allow points as well as surfaces to be plotted on rotatable coordinates.



x-axis: -1.0000 < x < 1.0000
y-axis: -1.0000 < y < 1.0000
z-axis: 0 < z < 1.0000

☑ Axes ☐ Legend
☐ Perspective
☑ Color
Resolution

Gridlines and Size: ☑ X ☑ Y Z

Grid Position: ☑ X 0 ☑ Y 0 ☑ Z 0.5

☐ Axis scale values
☐ Axis variable names

General Controls

The 3-D plot control panel, shown at the bottom of the 3-D plot window, can be made visible or hidden by double-clicking or right-clicking the mouse in the plot window. The Show/Hide Toolbar command in the Plot menu also provides this function.

The Axes checkbox has 3 positions. Clicking in the box rotates through the positions. If the box is checked, X, Y, and Z axis lines are drawn through the plot origin with red, green, and blue dotted lines, respectively. If the box is checked with a white background, axis labels will be displayed to help identify the rotational position. If the box is unchecked, the axis lines are not drawn.

If the Perspective checkbox is checked, a perspective projection of the 3-D surface is displayed. If the box is not checked, orthogonal projection is employed. Perspective projection appears more correct to the eye but orthogonal projection is easier interpret.

The Color checkbox has three positions. If the box is unchecked, the surface is displayed with with shades of gray. The color can be controlled with the slider to the right of the color checkbox. If the box is checked once, the value of Z (or the value of an optional color variable) is represented on the surface in color ranging from blue to red. If the box is checked a second time, the colors are changed to be full-spectrum as shown in the figure above. The optional color variable is chosen in the X-Y-Z Plot Setup. If a variable is not chosen, the color variable is Z itself. A legend indicating the values associated with the colors if the Legend box is clicked.

The Resolution slider controls the size of the polygons used to form the surface plot. High resolution (slider bar to the right) results in a crisp plot, but rendering and rotating may be sluggish. You may want to experiment with this control to determine a suitable compromise between display speed and quality.

Scale

Slider controls are provided to separately scale the X, Y and Z axes. Moving the slider to the right increases the scale factor for the selected axis. The slider can be moved with the mouse or with the left/right arrow keys. The check boxes for X and Y control the visibility of surface grid lines. The vertical slider scales the X, Y, and Z directions uniformly. Moving the slider down increases the apparent size of the plot. This slider can also be controlled with the up/down arrow keys. The lower and upper limits of the plotting range can be changed for the X and Y axes by right-clicking on the slider control.

Grid Position

The option of displaying grids in each of the orthogonal planes is controlled with the checkboxes. For example, the grid in the YZ plane will be visible if the checkbox preceding X is checked. The grid can be moved to any position within the range of values chosen in the X-Y-Z Plot Setup using the mouse or the left/right arrow keys. The current location of the grid is shown in the box to the right of the slider. A value can be directly entered into this box which will cause the grid to move to the specified value.

Axis Scale and Labels

If the Axis scale values check box is checked, the lower and upper limits of the scales for the X, Y, and Z axes will be displayed. If the Axis variable names check box is checked, the names of the variables plotted on the X, Y and Z axes will be displayed.

Orientation Buttons

Clicking the Top button will rotate the display so that the view is of the X-Y plane looking down the positive Z-axis towards the origin. The Front button presents a Y-Z plane viewed along the positive X axis. The Side button displays the X-Z plane from the positive Y axis. Clicking the Isometric button will orient the axes in an isometric manner that best shows the axis scale values and axis names.

Tool Bar

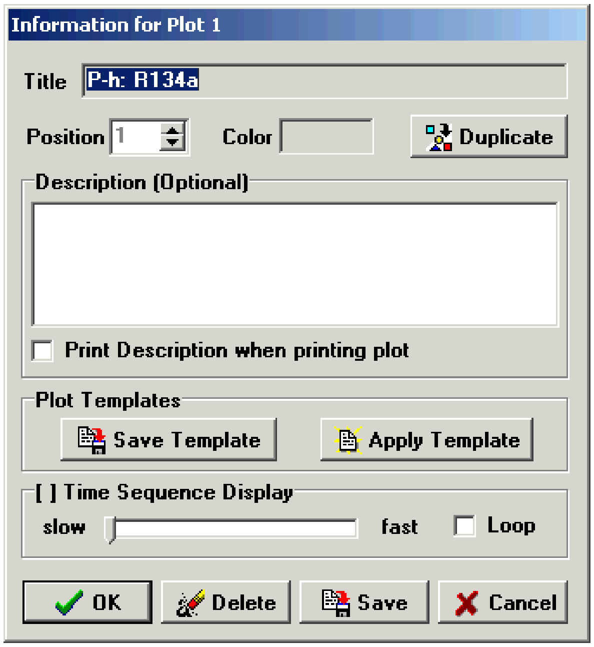
The tool bar provide buttons to place text, lines, boxes, and circles on the plot in the same manner as for 2-D plot windows. An align button is also provided. The help button displays this page.

Position Indication

Holding the Shift and Ctrl keys down will cause the 3-D coordinates to be displayed in the window caption, provided that the X and Y-axis grids are visible. The information displayed is the current locations of the X and Y grids, followed by the value of Z on the surface for the given X and Y. If the color corresponds to a variable other than Z, its value on the surface at the specified X and Y is also shown. If the X and Y grids are moved using the mouse while the Shift and Ctrl keys are held down, the information in the window caption will be automatically updated.

Saving and Loading Plot Templates

The Professional version allows plot templates to be saved and applied. A plot template contains information that affects the visible display of a plot. The characteristics of an existing plot can be set as desired and then these characteristics can be be saved in a .EPT (EES plot template) file. The characteristics can be transferred to other plots by selecting the .EPT file and applying it. The Save Plot and Apply Plot Template buttons are accessed by right-clicking the mouse on the tab at the top of the plot window. That will bring up the following dialog. Click the Save Template or Apply Template button to initiate the operation.



Information for Plot 1

Title P-h: R134a

Position 1

□ Print Description when printing plot

The .EPT file is a text file that can be viewed and edited by the user. The Save Template saves the characteristics for the current file, but these characteristics can be edited with any standard text editor. Each item in the file is annotated with a comment. Setting a value in the .EPT file to -99 disables the change in that characteristic. Open an .EPT file for further explanation.

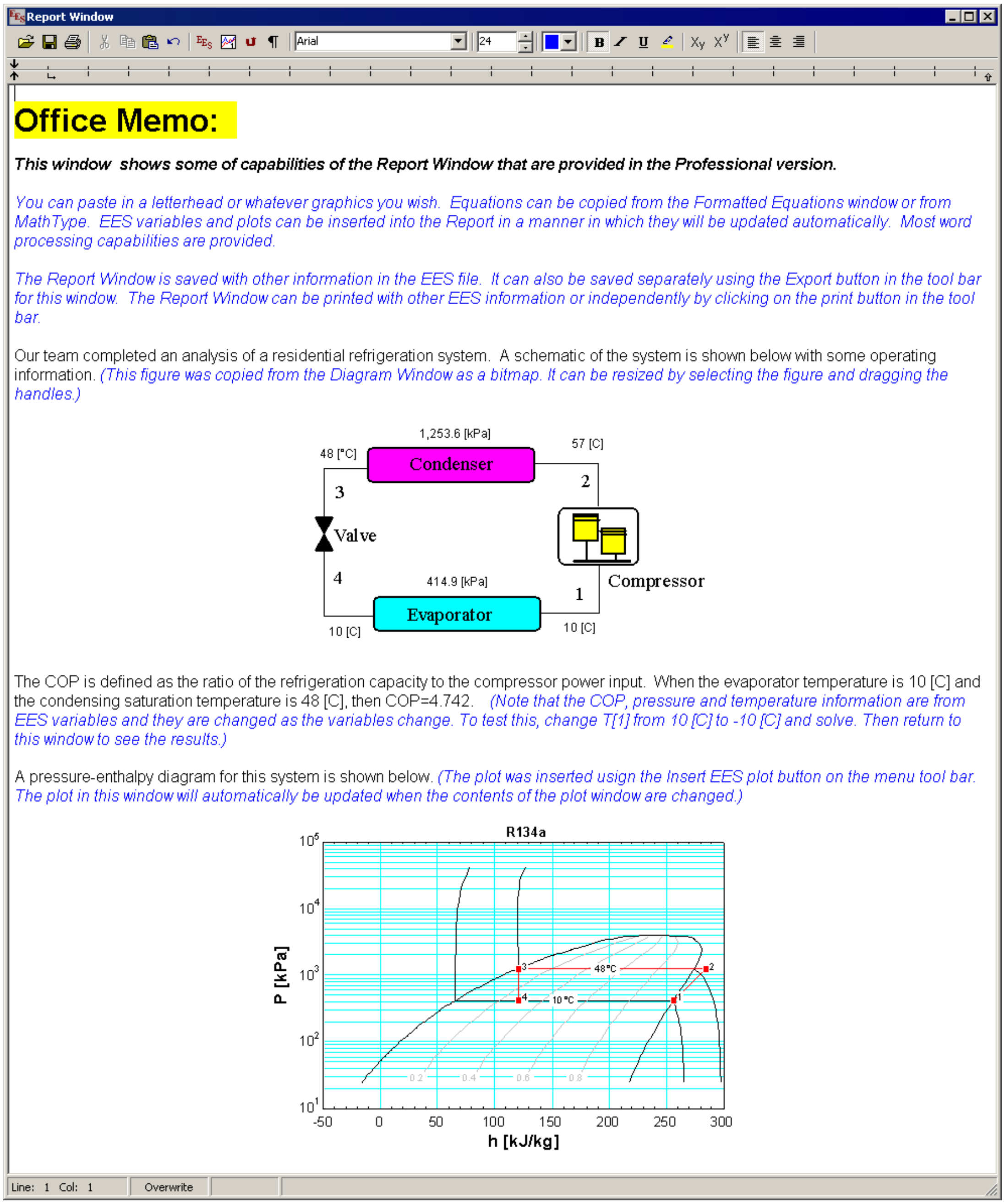
The tabs used to select different plot windows can be 'dragged' to a different tab location. This method of reordering the tabs is quicker than the alternative of right-clicking on each tab and specifying the tab position in the above dialog.

Plot Thumbnails

The Plot Thumbnails window in the Professional version shows a display of all of the plots in an EES file and makes it easy to find and select the desired plot. The operation of this window has changed. The array keys are used to select a plot in the Plot Thumbnails window. The selected plot is displayed with a yellow border. Pressing the Enter key or clicking the right mouse button on the plot will cause the Plot Thumbnails window to close and the Plot Window to display the selected plot.

Report Window (Professional Version)

The Report Window operates just like a typical word processor offering the usual choice of fonts, font sizes, text style, color, highlighting, tabs, left and right indents, etc. Graphics can be pasted into the window at the cursor position. The graphics can be equations copied from the Formatted Equations window, a plot from the Plot window, or any graphics in the Diagram window. Graphics can also be pasted from other applications. In addition, the Report Window can include EES variables that are automatically updated as their values change. A sample Report Window is shown below.



Office Memo:

This window shows some of capabilities of the Report Window that are provided in the Professional version.

You can paste in a letterhead or whatever graphics you wish. Equations can be copied from the Formatted Equations window or from MathType. EES variables and plots can be inserted into the Report in a manner in which they will be updated automatically. Most word processing capabilities are provided.

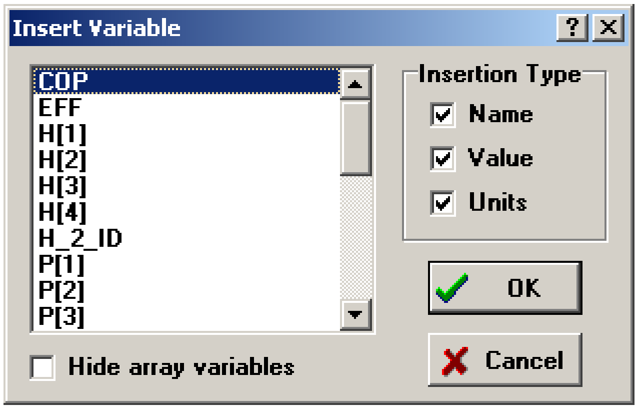
The Report Window is saved with other information in the EES file. It can also be saved separately using the Export button in the tool bar for this window. The Report Window can be printed with other EES information or independently by clicking on the print button in the tool bar.

Our team completed an analysis of a residential refrigeration system. A schematic of the system is shown below with some operating information. *(This figure was copied from the Diagram Window as a bitmap. It can be resized by selecting the figure and dragging the handles.)*

The COP is defined as the ratio of the refrigeration capacity to the compressor power input. When the evaporator temperature is 10 [C] and the condensing saturation temperature is 48 [C], then COP=4.742. *(Note that the COP, pressure and temperature information are from EES variables and they are changed as the variables change. To test this, change T[1] from 10 [C] to -10 [C] and solve. Then return to this window to see the results.)*

A pressure-enthalpy diagram for this system is shown below. *(The plot was inserted usign the Insert EES plot button on the menu tool bar. The plot in this window will automatically be updated when the contents of the plot window are changed.)*

To include the name, value, and/or units of an EES variable, click on the EES button in the tool bar EES or, alternatively, click the right mouse button and select Insert Variable from the popup menu. Either action will bring up a dialog showing all current EES variables.



COP
EFF
H[1]
H[2]
H[3]
H[4]
H_2_ID
P[1]
P[2]
P[3]

☑ Name
☑ Value
☑ Units

☐ Hide array variables

Select the variable you wish to insert from the list on the list and then indicate with the check boxes the information about this variable that is to be inserted in the Report Window. Click OK. The information inserted in this manner is automatically updated in the Report Window when changes value of the variable is changed.

Note that the Report Window provides buttons on the tool bar to Import and Export text. The Import button will insert the contents of a selected .RTF or .TXT file at the cursor position. The Export file will export the entire file to a specified .RTF or .TXT file. The .RTF files can include graphics. For example, you can include a graphical letter head in an .RTF file and import it whenever you wish to prepare a formatted report. The report can be printed, by clicking the print button.

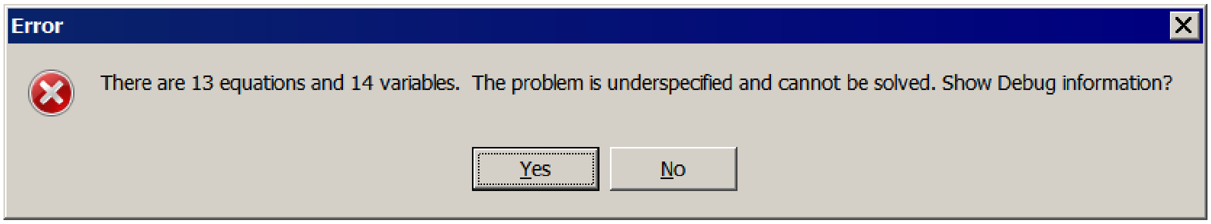
The ¶ button on the toolbar shows or hides the coding that EES uses to identify variables. In some cases, the hidden text will interfere with the ability to add text. If this problem occurs, click this button and you should then be able to enter text anywhere. Exposing the coding also helps to identify where in the text the EES variables have been placed. Click the text again to hide the coding.

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.

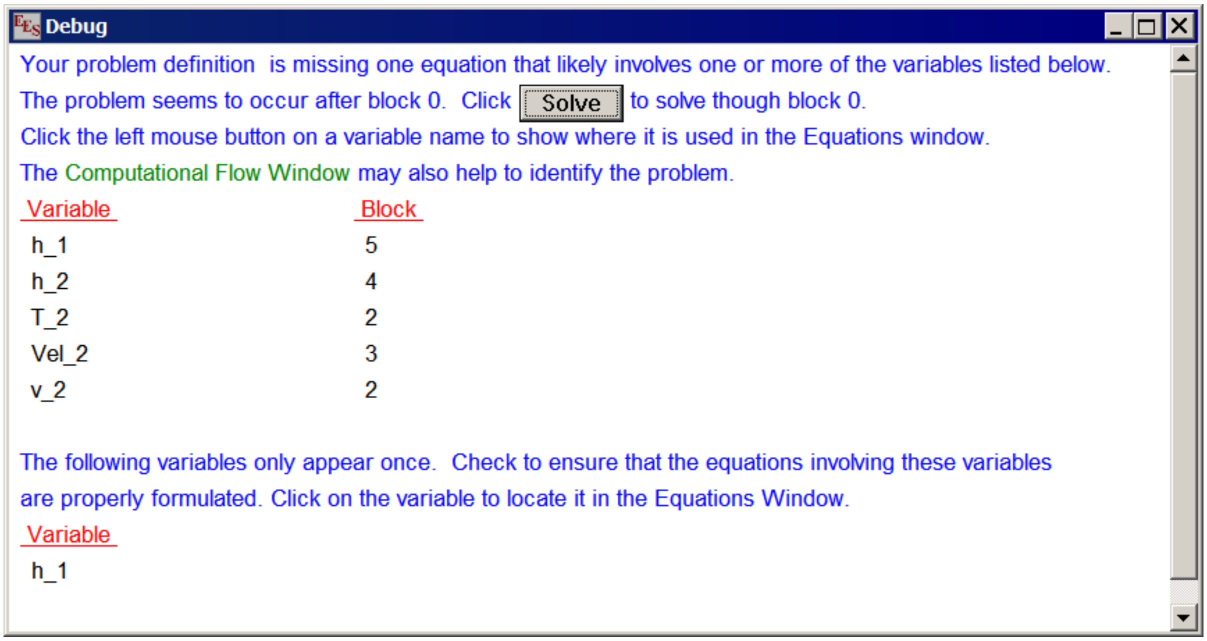
Incorrect Degrees of Freedom

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,



Your problem definition is missing one equation that likely involves one or more of the variables listed below.
The problem seems to occur after block 0. Click Solve to solve though block 0.
Click the left mouse button on a variable name to show where it is used in the Equations window.
The Computational Flow Window may also help to identify the problem.

Variable	Block
h_1	5
h_2	4
T_2	2
Vel_2	3
v_2	2

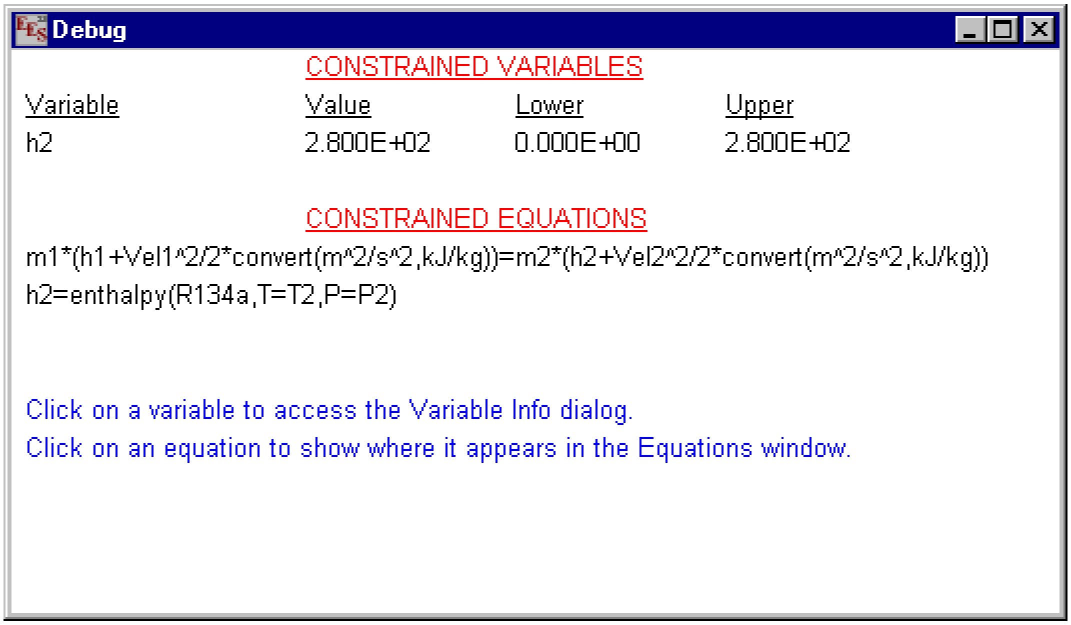
The following variables only appear once. Check to ensure that the equations involving these variables are properly formulated. Click on the variable to locate it in the Equations Window.

Variable
h_1

This window provides two lists of variables. Clicking the left mouse button on a variable name in either list will highlight the variable in the Equations Window. The first 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 or Computational Flow windows helpful in identifying the problem with your equation set. The second 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. Clicking the Solve button will result in EES solving as far as it can with the current equations.

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

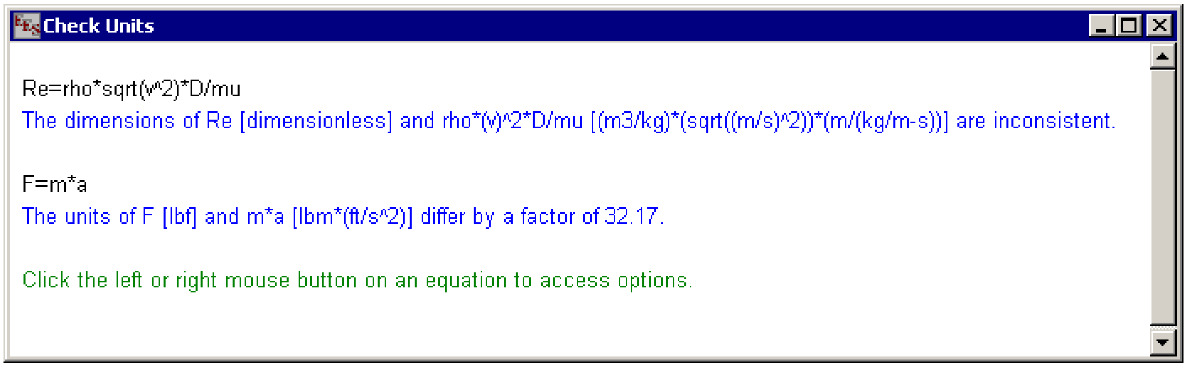
CONSTRAINED EQUATIONS

```
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 or right mouse button will bring up a popup menu. The pop-up menu offers the options of showing the formatted equation with units displayed for variables and constants, jumping the focus to offending equation in the Equations window or bringing up an abbreviated form of the Variable Information dialog window showing only the variables in that equation. In addition, the unit checking for the selected equation can be disabled from this menu.



```
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 or right mouse button on an equation to access options.
```

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. Clicking the mouse button on an variable will toggle its selection state. Right-clicking will open a dialog in which the units of all selected variables can be set. Units of constants can be declared directly in the Equations Window by following the numerical value with the units provided in brackets. For example, the following equation will set the Q_dot to 1000 with units of Btu/hr.

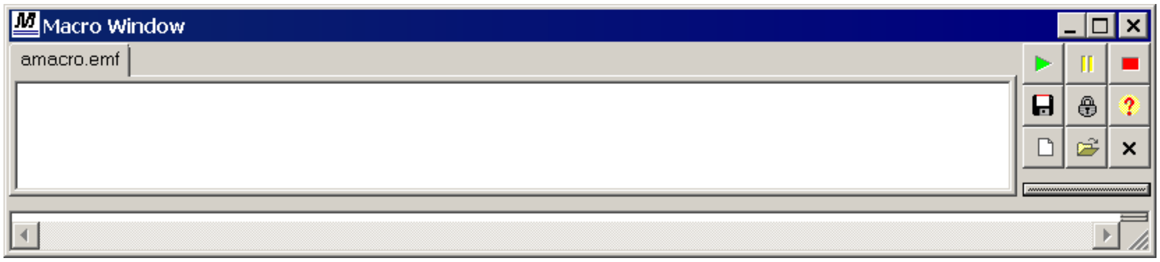
Q_dot = 1000 [Btu/hr]

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.

Macro Window

The Macro window is only available in the Professional version. This window provides capability to automatically record or manually edit macro commands, which are similar in some respects to the MATLAB Command window. Macros can be 'played' allowing an easy way to repeat a sequence of instructions. Each set of macro commands is contained in its own tabbed window, as shown below. See Creating and Using Macro Files (Professional Version) in Chapter 7 for more information.

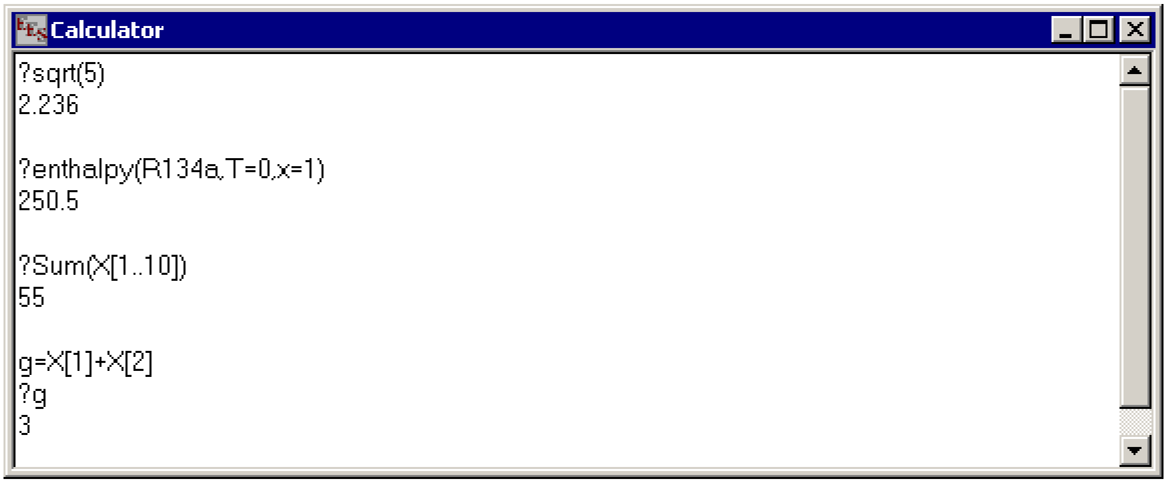


Calculator Window

The Calculator window allows single-line expressions to be evaluated. For example, if you enter:

?sqrt(5)

and then press the Enter key, EES will evaluate the expression and place the result on the following line, as shown.



```
?sqrt(5)
2.236

?enthalpy(R134a,T=0,x=1)
250.5

?Sum(X[1..10])
55

g=X[1]+X[2]
?g
3
```

The expression you enter can involve EES variables that have been previously defined (and evaluated) in the Equations window. Any of the built-in functions can be used, as well. Note that the value returned in the Calculator window is the current value of the variable. If you have not initiated the Solve or SolveTable commands, the value may not be current.

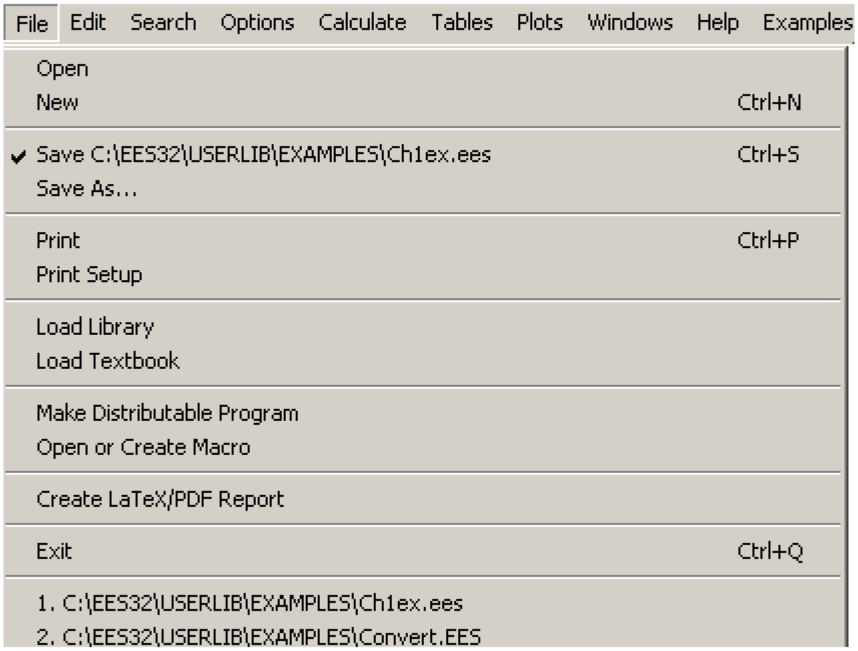
New variables may be defined in the Calculator window, provided that they are not already defined in the Equations window. For example, in the above example, variable x is set to the enthalpy of R134a. If any of the thermodynamic property functions are used, the unit system in effect is that which is defined with the Unit System command.

Be sure to press the Enter key after entering each line in order to process the command. The contents of the Calculator window are deleted when the window is closed. If you need to save the information in this window, select it and copy it to another window or another application.

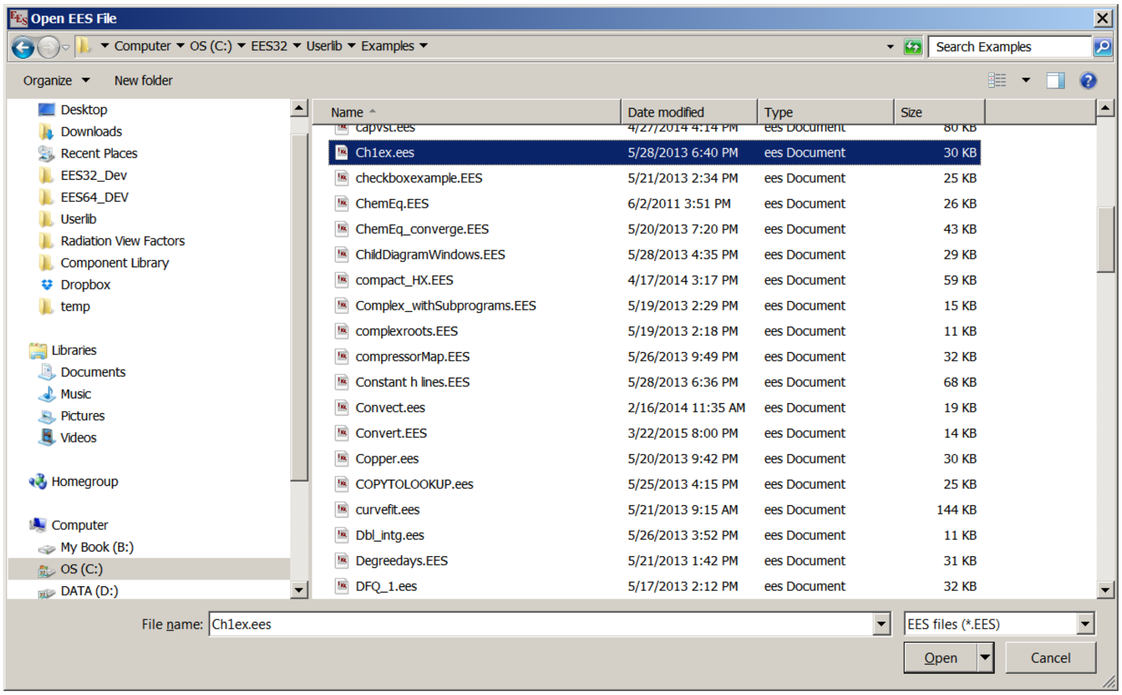
The Calculator window recognizes the following commands.
 % {insert the last calculated value in place of the % character}
 Clear {clear the contents of the Calculator window}
 Solve {initiates the same action as the Solve command in the Calculate menu}
 Exit {close the Calculator window}

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.



Name	Date modified	Type	Size
capvst.ees	4/27/2014 4:14 PM	ees Document	80 KB
Ch1ex.ees	5/28/2013 6:40 PM	ees Document	30 KB
checkboxexample.EES	5/21/2013 2:34 PM	ees Document	25 KB
ChemEq.EES	6/2/2011 3:51 PM	ees Document	26 KB
ChemEq_converge.EES	5/20/2013 7:20 PM	ees Document	43 KB
ChildDiagramWindows.EES	5/28/2013 4:35 PM	ees Document	29 KB
compact_HX.EES	4/17/2014 3:17 PM	ees Document	59 KB
Complex_withSubprograms.EES	5/19/2013 2:29 PM	ees Document	15 KB
complexroots.EES	5/19/2013 2:18 PM	ees Document	11 KB
compressorMap.EES	5/26/2013 9:49 PM	ees Document	32 KB
Constant h lines.EES	5/28/2013 6:36 PM	ees Document	68 KB
Convect.ees	2/16/2014 11:35 AM	ees Document	19 KB
Convert.EES	3/22/2015 8:00 PM	ees Document	14 KB
Copper.ees	5/20/2013 9:42 PM	ees Document	30 KB
COPYTOLOOKUP.ees	5/25/2013 4:15 PM	ees Document	25 KB
curvefit.ees	5/21/2013 9:15 AM	ees Document	144 KB
Dbl_intg.ees	5/26/2013 3:52 PM	ees Document	11 KB
Degreedays.EES	5/21/2013 1:42 PM	ees Document	31 KB
DFQ_1.ees	5/17/2013 2:12 PM	ees Document	32 KB

File name: Ch1ex.ees

EES files (*.EES)

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.

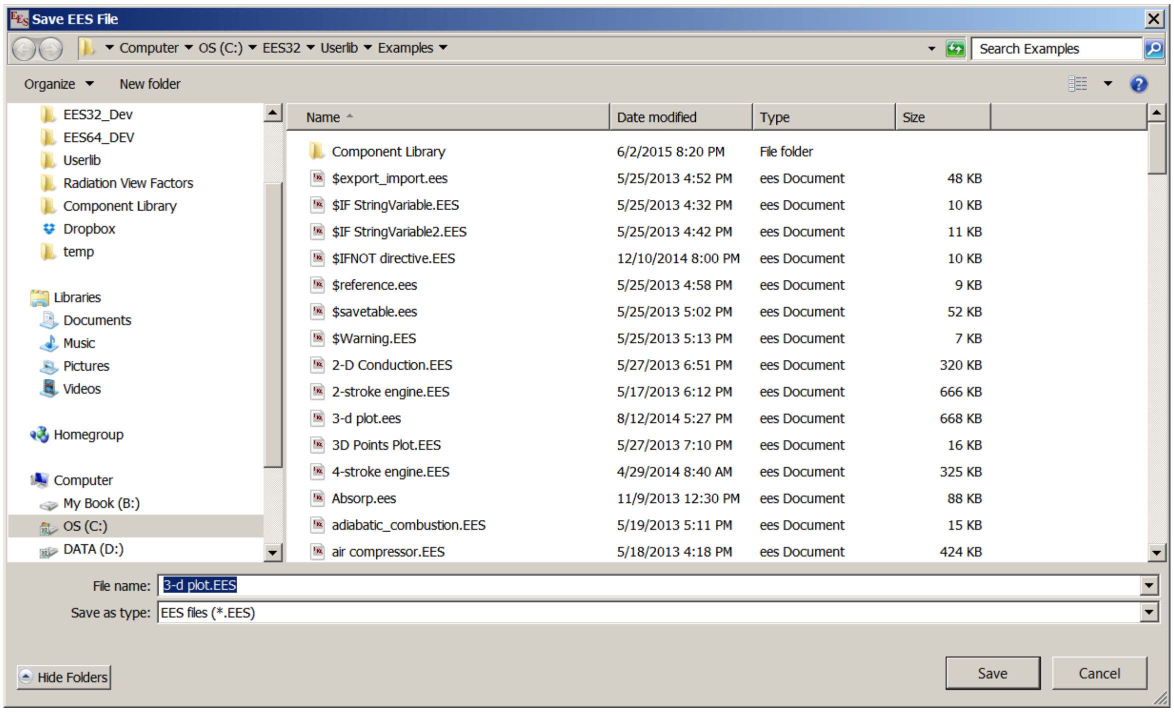
EES can read four types of files that are identified as **EES file**, **Text file**, **Library file, and Compressed EES file** formats. The format is selected with the drop-down list at the bottom left of the dialog window. EES files with a .EES filename extension (or .EES64 for the 64-bit version) are the norm. Text files with a .TXT filename extension contain ASCII text which is read into the Equations window. Library files (.LIB or .LIB64) are EES files containing one or more functions, procedures or modules which can be automatically loaded at startup, as described in Chapter 5. Compressed EES files have a .EEZ or .EEZ64 filename extension. These files can only be written in the Professional version, but they can be read by any version. The compressed files use the zip compression techniques and they can be much smaller than the corresponding .EES file when the EES file contains a lot of data or employs graphics in the Diagram window.

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.

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. Note that EES will check your equations before saving the file and any error in the file will be flagged before the save operation is initiated.

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 can save four distinct 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 (.EES64 for the 64-bit version) and files having this extension will be displayed in the filenames list. 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 (or .LIB64) files located within in the USERLIB\ (or USERLIB64\) 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. The Compressed EES file with .EEZ (or .EEZ64) filename extension contains exactly the same information as an .EES file, but it is compressed and therefore is often a much smaller file than the .EES file.



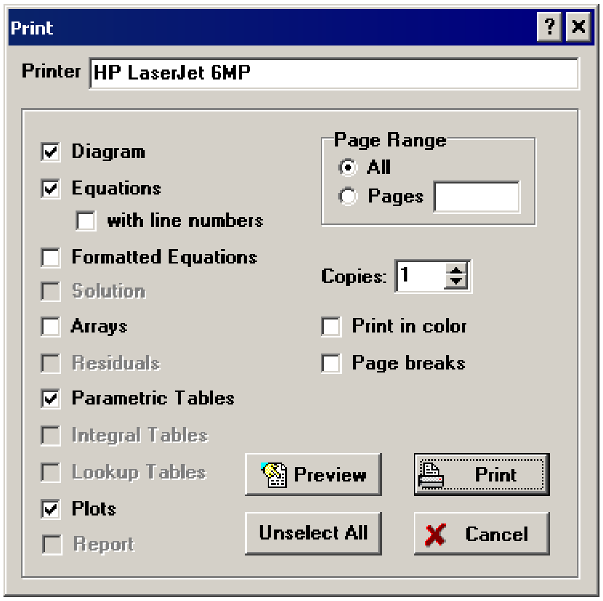
Name	Date modified	Type	Size
Component Library	6/2/2015 8:20 PM	File folder	
$export_import.ees	5/25/2013 4:52 PM	ees Document	48 KB
$IF StringVariable.EES	5/25/2013 4:32 PM	ees Document	10 KB
$IF StringVariable2.EES	5/25/2013 4:42 PM	ees Document	11 KB
$IFNOT directive.EES	12/10/2014 8:00 PM	ees Document	10 KB
$reference.ees	5/25/2013 4:58 PM	ees Document	9 KB
$savetable.ees	5/25/2013 5:02 PM	ees Document	52 KB
$Warning.EES	5/25/2013 5:13 PM	ees Document	7 KB
2-D Conduction.EES	5/27/2013 6:51 PM	ees Document	320 KB
2-stroke engine.EES	5/17/2013 6:12 PM	ees Document	666 KB
3-d plot.ees	8/12/2014 5:27 PM	ees Document	668 KB
3D Points Plot.EES	5/27/2013 7:10 PM	ees Document	16 KB
4-stroke engine.EES	4/29/2014 8:40 AM	ees Document	325 KB
Absorp.ees	11/9/2013 12:30 PM	ees Document	88 KB
adiabatic_combustion.EES	5/19/2013 5:11 PM	ees Document	15 KB
air compressor.EES	5/18/2013 4:18 PM	ees Document	424 KB

File name: 3-d plot.EES

Save as type: EES files (*.EES)

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.

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. All windows that are currently visible will be checked. The Unselect All button unchecks all of the entries. If a check box is disabled (as it is for Arrays in the Print dialog window shown below) the window is not available for printing. If a check appears in the box, the window will be printed. To check the box or to remove an existing check, click the mouse while the cursor is positioned in the box.


Print

Printer: HP LaserJet 6MP

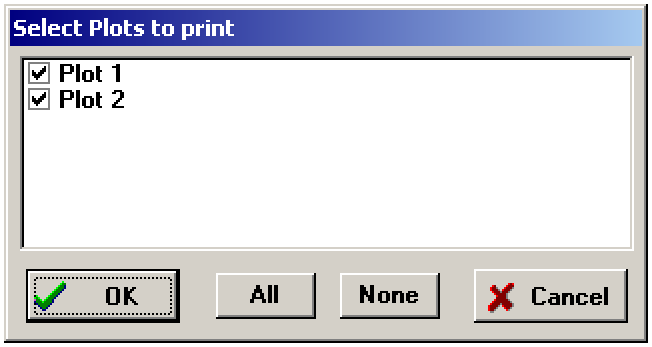
☑ Diagram
☑ Equations
□ with line numbers
□ Formatted Equations
□ Solution
□ Arrays
□ Residuals
☑ Parametric Tables
□ Integral Tables
□ Lookup Tables
☑ Plots
□ Report

Page Range
● All
○ Pages

Copies: 1

□ Print in color
□ Page breaks

Some windows, such as the Plot Window and the 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

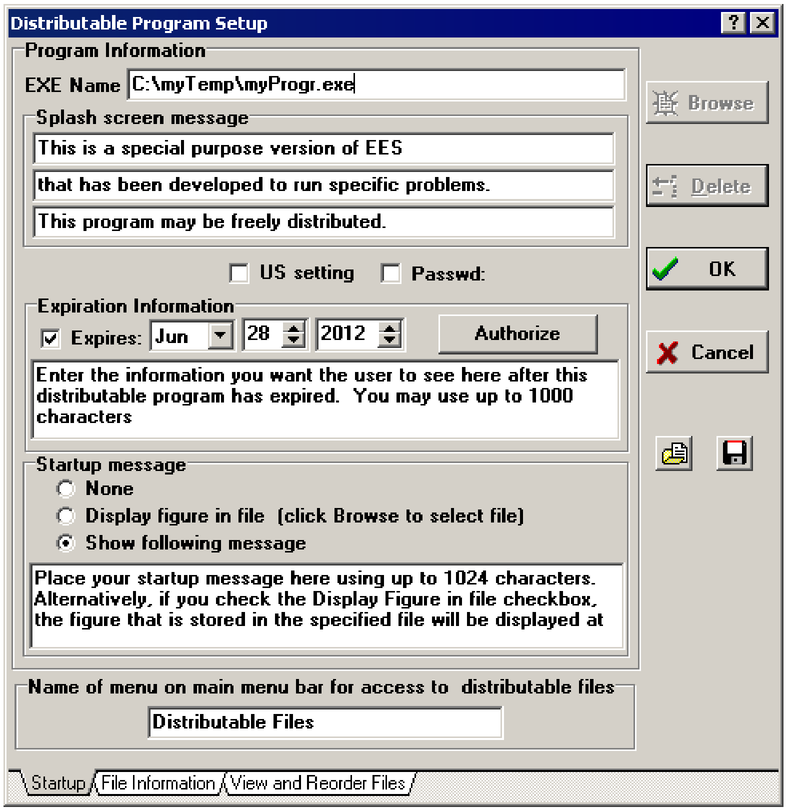
Windows manual for additional information on selecting printers. Printing options such as the font, line spacing, and font size are set in the Preferences dialog window (Options menu). If the "Print in color" check box is selected, EES will attempt to print selected windows using the same colors as appear 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 a standard open director dialog EES will load all recognized types of external programs (having file name extensions of .LIB, .FDL, .DLF, .DLP, .DLL (or their 64-bit equivalents for the 64-bit version. This command is only available in the Professional version. 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 will create a special purpose version of EES which will run up to 100 pre-selected problems. The following dialog will appear when this menu command is chosen. A special version of the EES program with the selected EES problems and all supporting files are placed in an executable file when you select the OK button. If the Create Zip file option is selected, the executable file and any library files that you select to include with it are compressed into a single .zip file. The executable file can be freely distributed to others. An expiration date can be optionally set which will prevent the program from being used thereafter. The executable file can be password protected. This Make Distributable Program command is available only in the Professional version.



Program Information

EXE Name C:\myTemp\myProgr.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.

□ US setting □ Passwd:

Expiration Information

☑ Expires: Jun 28 2012

Enter the information you want the user to see here after this distributable program has expired. You may use up to 1000 characters

Startup message

○ None

○ Display figure in file (click Browse to select file)

● Show following message

Place your startup message here using up to 1024 characters. Alternatively, if you check the Display Figure in file checkbox, the figure that is stored in the specified file will be displayed at

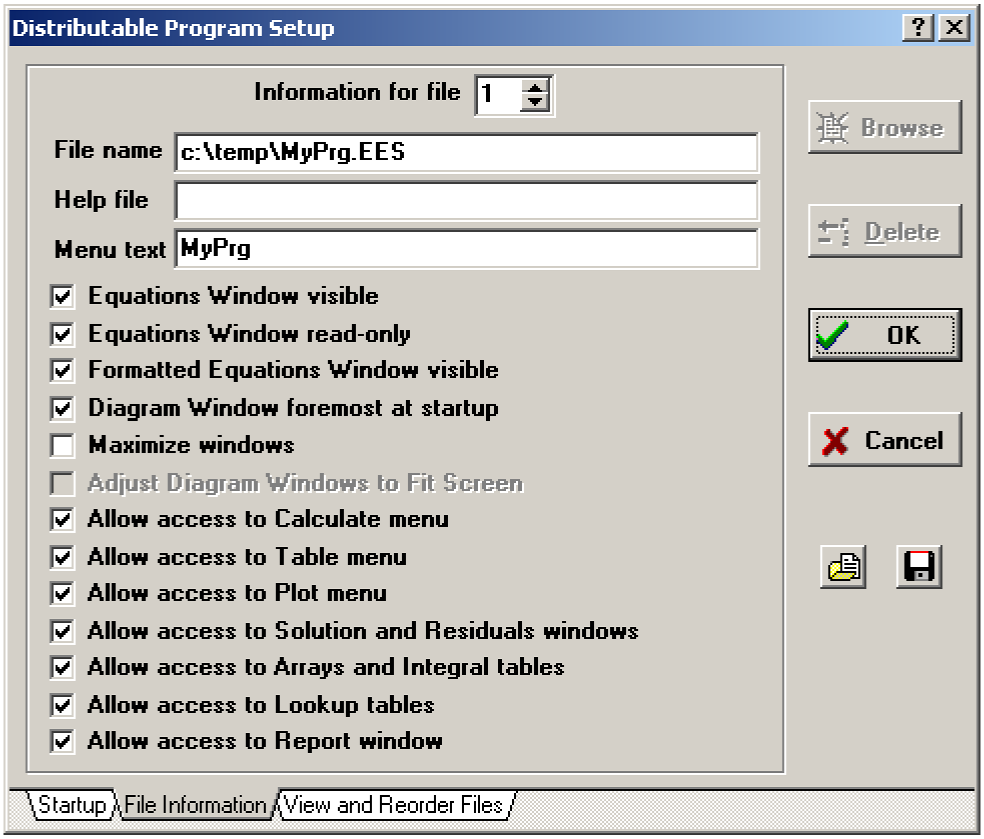
Name of menu on main menu bar for access to distributable files

Distributable Files

When the distributable program is started, the first file will automatically be loaded by default. However, any one of the files can be selected as the file that appears at startup by providing /# as a parameter where # is an integer between 1 and 100. For example, entering MyPrg.exe /2 in the Windows Run dialog will start the distributable program and display EES file 2. A menu with the file names is provided at the rightmost position of the menu bar. The name of the menu is entered at the bottom of the dialog. Any of the other files can be selected from this menu.

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.

Clicking on the File Information tab displays the following dialog with specific information for each EES file that is to be included in the distributable program. An optional help file can be provided for each EES file. The help is accessed from the Help menu. The name of the file as it appears in the menu can also be specified.



Distributable Program Setup

Information for file 1

File name c:\temp\MyPrg.EES

Help file

Menu text MyPrg

☑ Equations Window visible
☑ Equations Window read-only
☑ Formatted Equations Window visible
☑ Diagram Window foremost at startup
☐ Maximize windows
☐ Adjust Diagram Windows to Fit Screen
☑ Allow access to Calculate menu
☑ Allow access to Table menu
☑ Allow access to Plot menu
☑ Allow access to Solution and Residuals windows
☑ Allow access to Arrays and Integral tables
☑ Allow access to Lookup tables
☑ Allow access to Report window

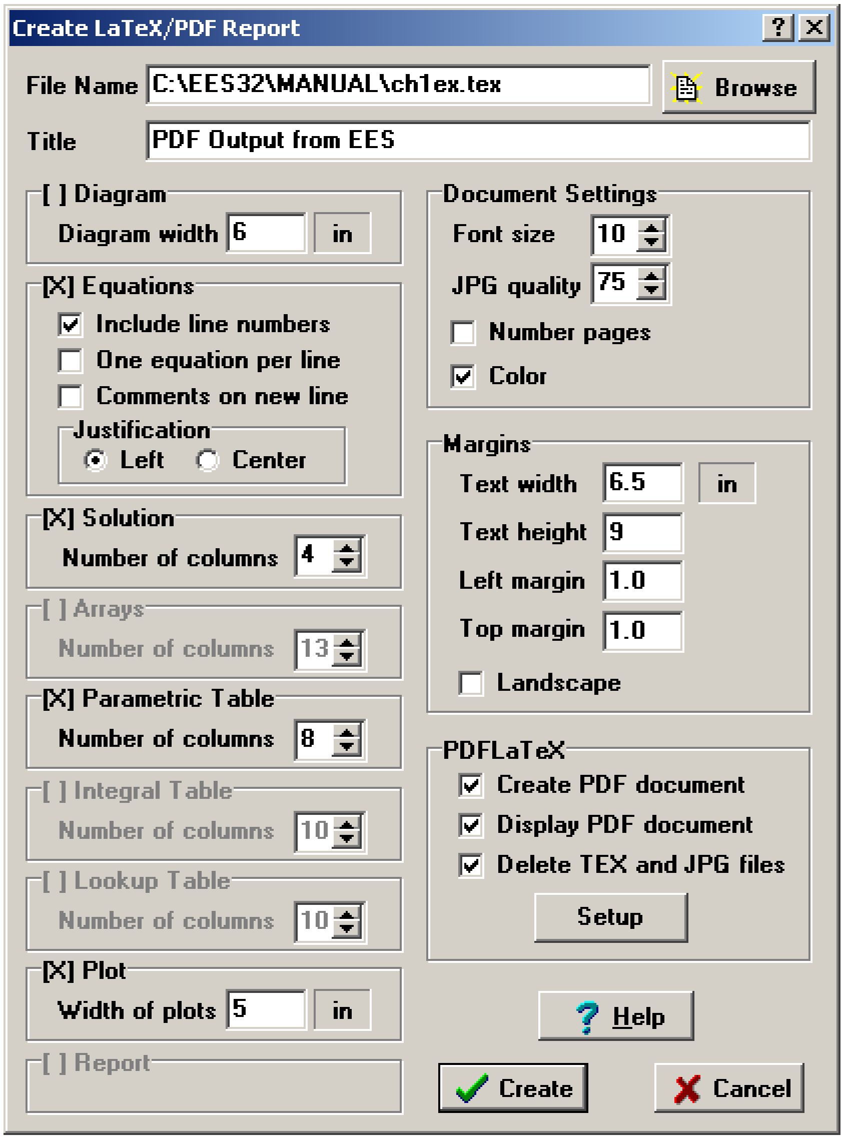
The remaining items in for each EES file are a series of check boxes that control the capabilities the user will have when running the distributable version. If you want you user to have the ability to view the equations window, click the 'Equations Window Visible' check box. You can make this window 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.

Open or Create 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.

Create Latex/PDF Report generates a report including the Diagram, equations, solution, tables, and plots. In this respect, this command is similar to the Print command. However, the Create LaTeX/PDF Report command does not directly print but instead creates a TeX document in LaTeX2e. The TeX document is an ASCII file with a .tex filename extension that must be processed by a LaTeX2e compiler. LaTeX2e produces a

.dvi (device independent) output file that can be viewed and printed by various utilities. However, the PDFLaTeX accessory that is included with the TeX compiler can also produce a .pdf (portable document interface) file that can be viewed and printed with Adobe Acrobat Reader. To make use of the output generated by this command you will need to install the LaTeX2e compiler and Adobe Acrobat Reader, if they are not already installed on your computer. Both of these programs are available at no cost.

The recommended installation of Latex2E is MiKTeX. The MiKTeX homepage is http://miktext.org. The following installation directions are applicable to MiKTeX version 2.9 and earlier. Download the distribution zip file in one step by either clicking on the appropriate link in the MiKTeX home page or entering the following URL into your browser program: https://miktex.org/howto/download-miktex. Instructions for downloading the MikTeX files are provided on this webpage. The downloading is facilitated by the MikTeX Setup Wizard. Click the Basic MikTeX Installer option. This installation program will install MiKTeX into the directory that you specify. EES must know the directory path of the PDFLaTeX.exe program that is provided with MiKTeX so make a note of the directory that you are installing the MiKTex package into. To configure EES to work with the MikTeX package, start EES and select the Create LaTeX/PDF Report command from the File menu. Click the Setup button in the PDF LaTeX box and enter the directory information for the PDFLaTeX.exe program. The directory should be C:\Program Files (86)\MiKTeX 2.9\miktex\2.9\bin.


Create LaTeX/PDF Report

File Name C:\EES32\MANUAL\ch1ex.tex

Title PDF Output from EES

[ ] Diagram
Diagram width 6 in

[X] Equations
☑ Include line numbers
□ One equation per line
□ Comments on new line
Justification
● Left ○ Center

[X] Solution
Number of columns 4

[ ] Arrays
Number of columns 13

[X] Parametric Table
Number of columns 8

[ ] Integral Table
Number of columns 10

[ ] Lookup Table
Number of columns 10

[X] Plot
Width of plots 5 in

[ ] Report

Document Settings
Font size 10
JPG quality 75
□ Number pages
☑ Color

Margins
Text width 6.5 in
Text height 9
Left margin 1.0
Top margin 1.0
□ Landscape

PDFLaTeX
☑ Create PDF document
☑ Display PDF document
☑ Delete TEX and JPG files

A dialog shown above will appear after selecting the Create LaTeX/PDF Report command. At the top of the dialog is an edit field for a filename. By default, this filename is set to the name of the EES file but with a .tex filename extension. The filename can be changed, but is should have a .tex filename extension if it is to be recognized by the pdflatex application. The Browse button to the right of the filename should make it easy to select an alternative location for the file. The Title field is by default filled with the EES filename and the date. This title, which can be changed to whatever you wish, will be displayed centered at the top of report. On the left side of the dialog are controls to select the windows that will be included in the report. Each window is represented with a group box with one or more options. If the group box is displayed in grayed text, the corresponding window is not available. The title of each group box begins with a left bracket character. If the window is to be included in the report, the character following the left bracket must be an X or a number. Click on the group name to change the selection. The Parametric Table, Lookup Table, Integral Table and Plot windows, have tabs with the possibility of accessing more than one window. If there are multiple windows, a dialog will appear when you click on the group name to allow selection of the tables or plots that are to be included in the report. In this case, the number enclosed in the brackets in the group name is the number of tables or plots that are to be included. If there is only one possibility, such as for the Equations, Solution, and Arrays windows, an X, rather than a number will be enclosed within the brackets if the window is to be included.

One limitation of the LaTeX report is that it will not automatically fit the contents to the width of the paper. For example, long equations may end up running off the right margin. Each group box provides controls to deal with this situation. Long equations are the major problem. You can control this situation somewhat by modifying the equations you enter in the Equations window. For example, a long equation can be broken into two or more shorter equations by introducing some new variables. Also, EES allows you to break an equation so that it continues on the following line if the last character on the line is a &. (Note: the total length of an EES equation must be less than 255 characters, even if it is broken into multiple lines with the & character.) The LaTeX report will also attempt to break the equation at the & symbol, but depending on the formatting, it may produce a LaTeX error. If you are familiar with LaTeX code, you can of course modify the .TeX file as needed. The check boxes in the Equations group box allow multiple equations that are entered on a single line in EES to appear on separate lines in the report and comments that appear on the same line as the equation to be be placed on a separate line. Both of these options help to reduce the long line problem. Other tools available to deal with long lines are the font size and margins that are specified on the right side of the dialog.

The Solution and table group boxes each have a field to specify the number of columns that will be used to display the data in the report. The .TeX processor does not wrap the information if it is too wide for the page, but you can control this problem quite well by specifying the number of columns. The width of the Diagram and plot windows can be directly entered. The figures will be scaled to the width you specify while maintaining the existing aspect ration.

The Diagram and Plots that are to be included in the LaTeX document are stored in separate files having a .jpg filename extension in the same directory as the .tex file. These .jpg files have same parent name as the .tex file with the addition of _D for the diagram and _P1, _P2, etc. for the plots. When creating a .jpg file, a decision must be made to balance the file size with file quality. The user has control over this decision with the JPG quality input box. The allowable values are 1 to 99. Low values result in small .jpg file sizes (which result in small .pdf file sizes) but sacrifice graphic quality.

Large values result in high quality graphics, but at an expense of great file sizes. The default value, 75, seems to provide a reasonable balance.

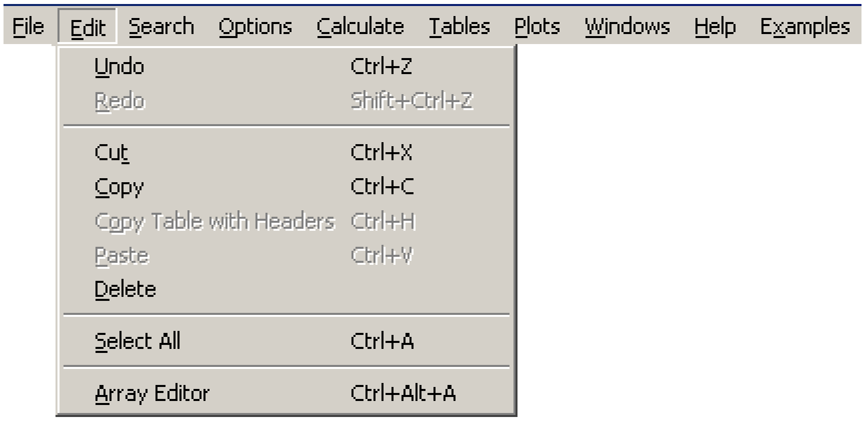
The PDFLaTeX group box on the right side of the dialog contains three checkboxes. If the Create PDF document checkbox is selected, EES will attempt to start the PDFLatex.exe application to compile that .tex file that it has just saved. If the Display PDF document checkbox is selected, EES will attempt to start Abode Acrobat to display the .pdf file. If the Delete TEX document checkbox is selected, the LaTeX file will be deleted leaving only the .PDF file. EES must know where the PDFLaTeX.exe application resides. The default location is C:\textmf\miktex\bin\pdflatex.exe. You can specify or change the default location by clicking on the Setup button.

When you click the Create button, EES will write the information in the dialog to a small file called EES_TeX.ini so that it is available as the default the next time you enter this dialog. The file is written in the Default File Folder specified in the Preferences dialog. If no folder is specified, the file will be placed in the same directory as the EES application. Also, note that EES will save your file when starting the Create LaTeX/PDF report command with a filename created by appending a ~ to the current name. Should any problems occur that cause EES to stop, your file can be recovered.

Exit 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. The Equations window supports 8 levels of Undo. If Undo is unavailable, the menu will be disabled.

Cut deletes the selected 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.

Copy Table with Headers is only enabled when the Parametric, Lookup, Arrays, or Integral table window is the foremost window. In this case, both the Copy Table and Copy Table with Headers commands will be visible. The Copy Table command will copy to the Clipboard the selected cells in the table as tab-delimited text. This text can be directly pasted in a spreadsheet application. The Copy Table with Headers command will copy the Header and Units of each column in addition to the selected data. Use the Select All command to select all of the cells in the table before issuing the Copy Table or Copy Table with Headers command.

Copy Diagram is only enabled when the Diagram window is the foremost window. The contents of Diagram window are placed on the clipboard as a bitmap and as an extended metafile.

Save Diagram is only visible when the Diagram window is the foremost window and the Diagram window tool bar is hidden. The Diagram can be saved as a .jpg or .bmp file.

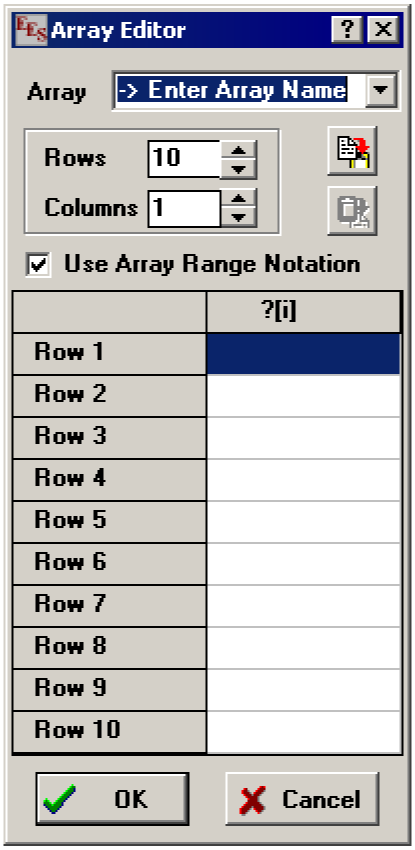
Paste is active for the Equations, Parametric, Lookup, Plot and Diagram windows. Paste moves the text (or graphics for the Diagram window) previously placed on the Clipboard with the Cut or Copy commands in EES or in other applications. When Paste is used in the Parametric or Lookup Table window, the 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.

Array Editor 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.



Array -> Enter Array Name

Rows 10

Columns 1

☑ Use Array Range Notation

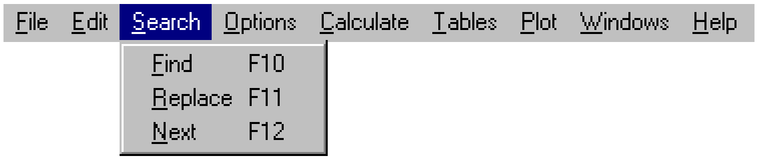
	?[i]
Row 1	
Row 2	
Row 3	
Row 4	
Row 5	
Row 6	
Row 7	
Row 8	
Row 9	
Row 10	

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:

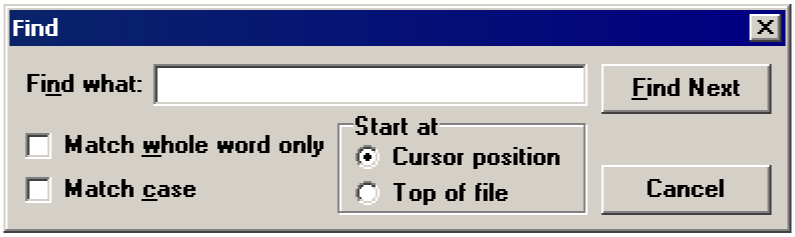
```
{Array A}
A[1]=1
A[2]=2
…
{Array A end}
```

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.



Find what:

□ Match whole word only

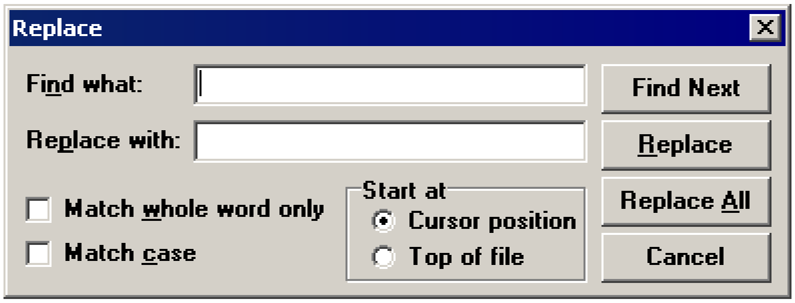
□ Match case

Start at

● Cursor position

○ Top of file

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.



Find what:

Replace with:

□ Match whole word only

□ Match case

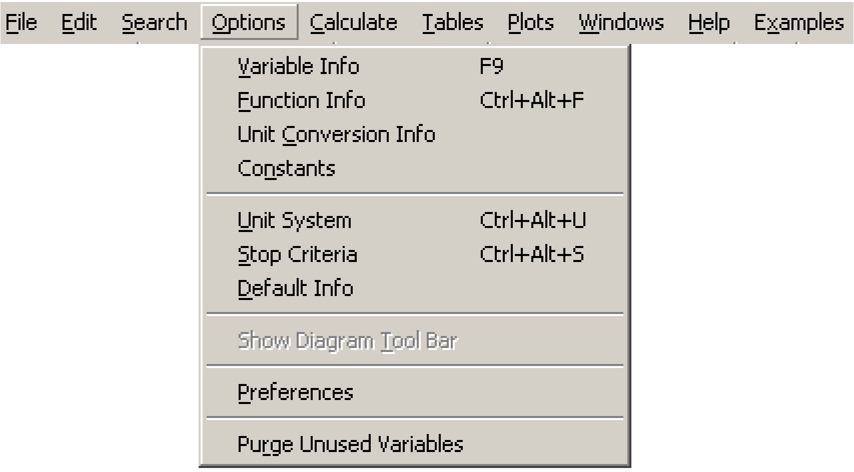
Start at

● Cursor position

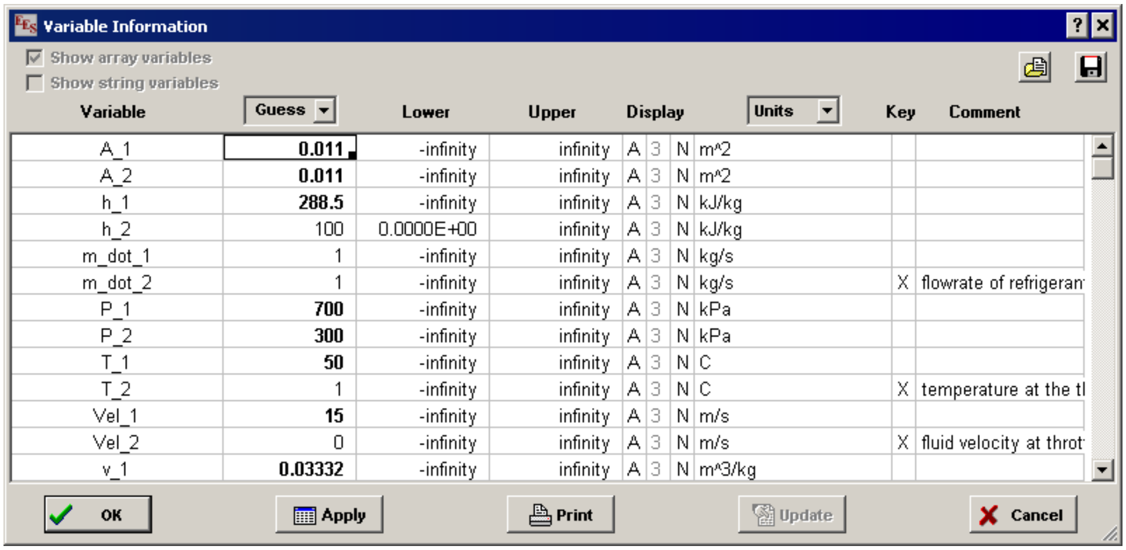
○ Top of file

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, units, and alternative 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.



☑ Show array variables
☐ Show string variables

Variable	Guess	Lower	Upper	Display			Units	Key	Comment
A_1	0.011	-infinity	infinity	A	3	N	m^2		
A_2	0.011	-infinity	infinity	A	3	N	m^2		
h_1	288.5	-infinity	infinity	A	3	N	kJ/kg		
h_2	100	0.0000E+00	infinity	A	3	N	kJ/kg		
m_dot_1	1	-infinity	infinity	A	3	N	kg/s		
m_dot_2	1	-infinity	infinity	A	3	N	kg/s	X	flowrate of refrigeran
P_1	700	-infinity	infinity	A	3	N	kPa		
P_2	300	-infinity	infinity	A	3	N	kPa		
T_1	50	-infinity	infinity	A	3	N	C		
T_2	1	-infinity	infinity	A	3	N	C	X	temperature at the tl
Vel_1	15	-infinity	infinity	A	3	N	m/s		
Vel_2	0	-infinity	infinity	A	3	N	m/s	X	fluid velocity at throt
v_1	0.03332	-infinity	infinity	A	3	N	m^3/kg		

If the program contains one or more subroutine, i.e., Functions, Procedures, Modules or Subprograms) (see Chapter 5 for a discussion of subroutines), a control will be provided at the top of the dialog to select the subroutine or main program. The information for the selected routine can then be changed or viewed.

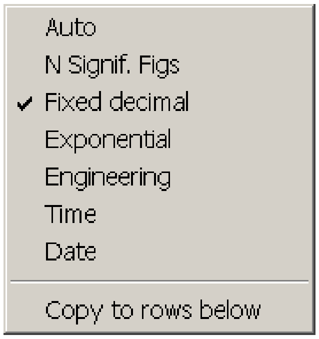
The "Show Array Elements" checkbox control will be visible at the upper left of the dialog window if there are any arrays are used in the Equations window. When this control is selected, all array elements appear in the Variable Info dialog and the guess value, bounds, display format, and units can be 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 control above the second column allows either the guess values or current values of the variables to be displayed the second column. 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.

Before this dialog window appears, EES will attempt to solve equations that have a single unknown. If the value of a variable has been pre-calculated, its pre-calculated value will be displayed instead of the its guess value in the guess value column and the value will be shown in bold font. These bold values can be changed and doing so will change the guess values. In some cases, that will affect the calculated result for these variables.

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 highlighting effects. The poup for for display style offer the following display choices.

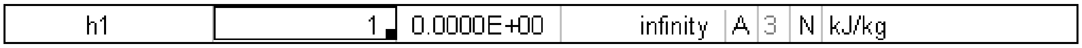


The units of the variable may be entered in the units column. Click the Units drop-down box at the top of the units column. If you select {Units}, the units column will display the alternate set of units that have been provided for each variable. The primary and alternate units can be displayed with the variable value by only the primary set of units is used for unit checking. A comment can be embedded within a unit string by surrounding the comment with either braces ({}} or backslash (\) characters. The comment will be displayed with the unit but it will be ignored by the Convert function when doing unit conversions. The length of the unit string, including the comment, is limited to 30 characters.. 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.

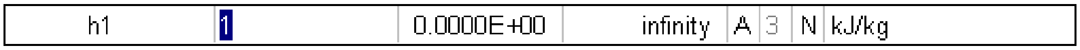
The Variable Info dialog window can be resized by dragging the lower right corner.

Selection Mode



h1	1	0.0000E+00	infinity	A	3	N	kJ/kg

Edit Mode



h1	1	0.0000E+00	infinity	A	3	N	kJ/kg

As shown in the figure above, there are two display modes. In Selection Mode, the active cell is outlined and there is a small box at the lower right of the cell. The contents of the cell can be copied to other cells above or below it by pressing the mouse button while the cursor is positioned within the small box and dragging the cursor to the desired location. (The cursor will change to an up/down arrow when it is positioned over the small box.) While in Selection mode, Ctrl-C will copy the contents of the active cell to the clipboard and Ctrl-V will paste the contents of the clipboard to the active cell. Selection and Edit modes are applicable to the Guess, Lower, Upper, and Units columns. Clicking the mouse within the selected cell causes a change to Edit mode. In Edit mode, the outline and small box will not be displayed. Instead the contents of the cell will initially be selected. The usual editing procedures can be applied to modify the contents of the cell. Pressing F2 causes the display to toggle between Edit Mode and Selection Mode. Alternatively, clicking on any other cell or in the header or footer of the dialog will change the display to Selection Mode.

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.

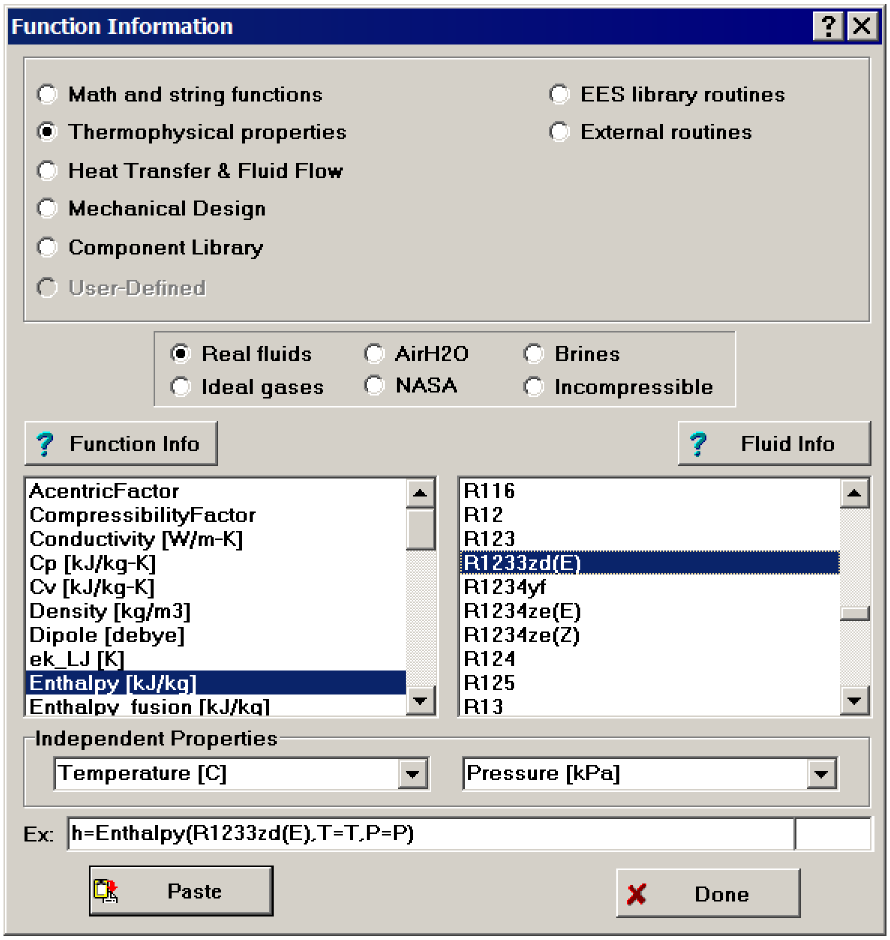
If a program contains one or more modules, a dropdown list appears at the top center of the Variable Info dialog from which the module or main program can be selected. In this case, the Save or Open Variable Information operation will be applied only to the selected module or main program. Variables 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).

The guess value and limits of variables in the main body of EES and in Subprograms can be specified with equations that can involve existing variables and numerical constants. For example, the guess value for variable X can be set to (X_min+X_max)/2 provided that variables X_min and X_max have been defined.

Function Info will bring up the following dialog window.



Function Information

○ Math and string functions
● Thermophysical properties
○ Heat Transfer & Fluid Flow
○ Mechanical Design
○ Component Library
○ User-Defined
○ EES library routines
○ External routines

● Real fluids ○ AirH2O ○ Brines
○ Ideal gases ○ NASA ○ Incompressible

AcentricFactor
CompressibilityFactor
Conductivity [W/m-K]
Cp [kJ/kg-K]
Cv [kJ/kg-K]
Density [kg/m3]
Dipole [debye]
ek_LJ [K]
Enthalpy [kJ/kg]
Enthalpy_fusion [kJ/kg]

R116
R12
R123
R1233zd(E)
R1234yf
R1234ze(E)
R1234ze(Z)
R124
R125
R13

Independent Properties: Temperature [C], Pressure [kPa]

Ex: h=Enthalpy(R1233zd(E),T=T,P=P)

The eight buttons at the top of the dialog window indicate which information is to be provided. Math functions and Thermophysical properties refer to the built-in mathematical functions and the thermodynamic and transport property functions for fluids and solids (See Chapter 4). Thermophysical properties are subclassified as Real fluids, Ideal gases, AirH2O (Psychrometrics), NASA fluids, Brines and Incompressible fluids. The EES library functions button provides a list of the user functions, procedures, subprograms and modules that are written in EES and loaded from Library files. (See Chapter 5 for additional information on Library files). The External routines button refers to external routines which can be linked to EES as described in Chapter 6. The remaining buttons provides access to application libraries. Information on how to add content to this control is provided in Chapter 7. There are three additional buttons for the Heat Transfer & Fluid Flow Library, the Mechanical Design Library, and the Component Library[7]. The functions corresponding to the selected button will be displayed in the

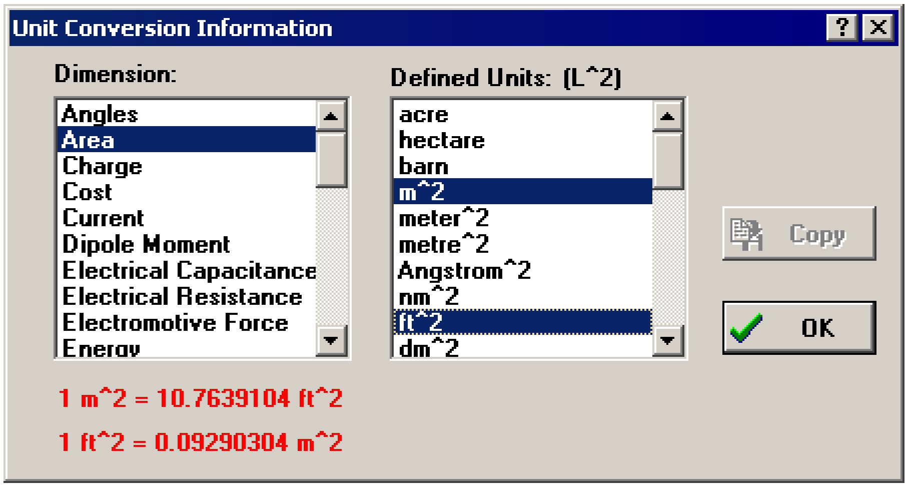
[7] Note that the Component Library is not automatically loaded at startup even though it resides in the Userlib folder. To load the Component Library, add a $LOAD 'Component Library' directive.

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 for the selected type are shown in the Substance list on the right. Click on the substance name in the scrollable list to select the substance. 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.

Thermophysical functions may require between 0 and 3 independent properties depending on the choice of function and substance. If the function requires independent properties, a box will appear with drop-down menus listing the alternatives available for the selected function-substance combination. The example box will display a sample use of the function with the selected independent properties. Click on Ex: to select all of the text in the Example edit control. The small index edit box at the right of the example box provides a convenient way to customize the display in the example box. The text in the edit box is appended to each thermodynamic variable. For example, if the text in the index edit box above were changed from _1 to [1], the example box would then display h[1]=ENTHALPY(Steam,T=T[1],x=x[1]). You can also edit the information in the Example text box if you wish. Click the Paste button to copy the text in the Example box into the EES Equation 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.

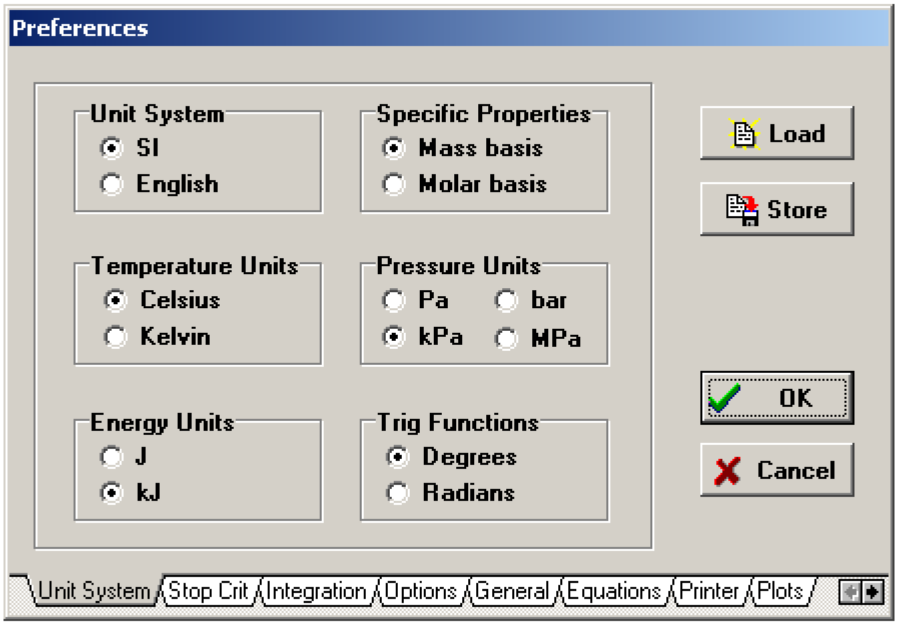


1 m^2 = 10.7639104 ft^2

1 ft^2 = 0.09290304 m^2

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 along with some unit combinations that have the same dimension. For example, if you click on Area, a list of units that have dimensions of area will be displayed. If you click on one of the Defined Units in the list on the right, that Copy button becomes enabled and the units can be copied by the Clipboard by clicking the Copy button. Click on any two units in the list of the right to see the conversion and inverse conversion factors relating them.. 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 Notepad.

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 but it can optionally check the unit consistency of each equation. 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.



Preferences

Unit System
● SI
○ English

Specific Properties
● Mass basis
○ Molar basis

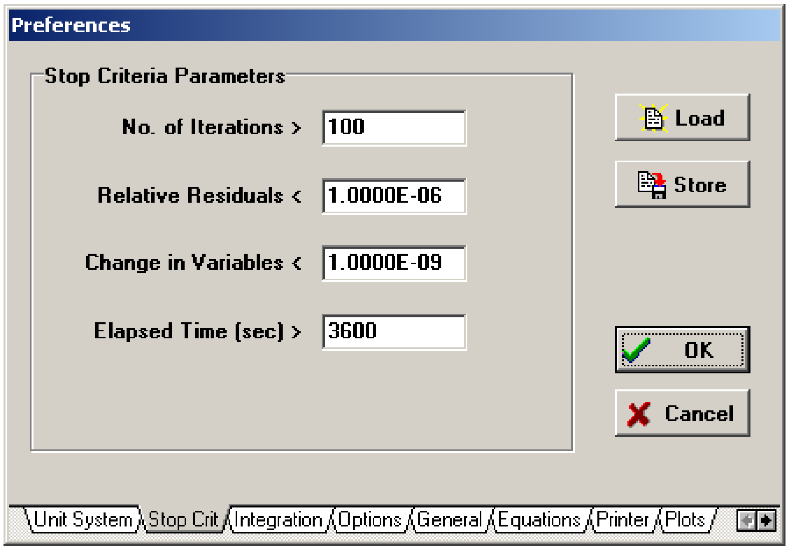
Temperature Units
● Celsius
○ Kelvin

Pressure Units
○ Pa ○ bar
● kPa ○ MPa

Energy Units
○ J
● kJ

Trig Functions
● Degrees
○ Radians

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



Preferences

Stop Criteria Parameters

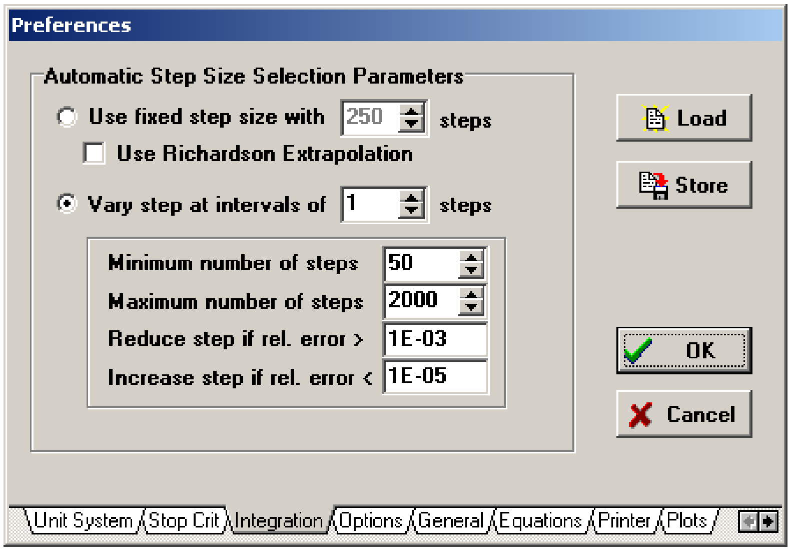
No. of Iterations > 100

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.



Preferences

Automatic Step Size Selection Parameters

○ Use fixed step size with 250 steps

□ Use Richardson Extrapolation

● Vary step at intervals of 1 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

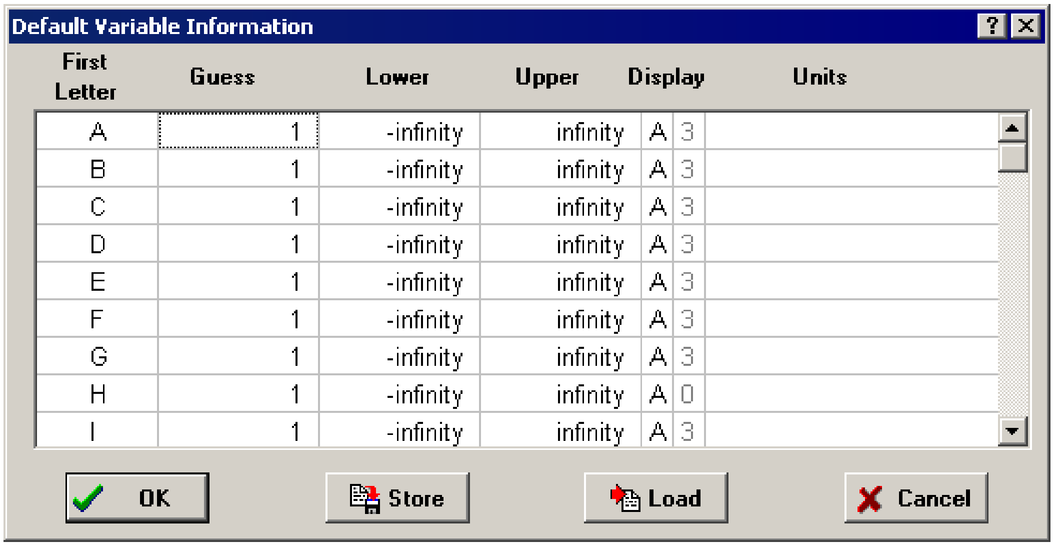
EES uses numerical integration to determine the value of an integral or to solve differential equations. The equation-based Integral function can use either a fixed user-supplied step or an automatic step adjusted to meet some accuracy criteria. The parameters in this tabbed section of the Tolerances 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.

Richardson extrapolation is provided as an option when the equation-based Integral function is used with a fixed step size. A fixed step size can either be specified in one of two ways. The step size can be specified (as the fifth parameter) in the equation-based integral function. Alternatively, if a step size is not provided in the function, EES will use the step size setting indicated in the Integration tab of the Tolerances dialog. Richardson extrapolation can be employed if the 'Use fixed step size' radio button is selected. Richardson extrapolation is a clever technique to reduce the truncation error associated with numerical integration. An explanation of this technique can be found in Numerical Methods for Engineers by S.C. Chapra and R.P. Canale, McGraw-Hill, 1985.

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.

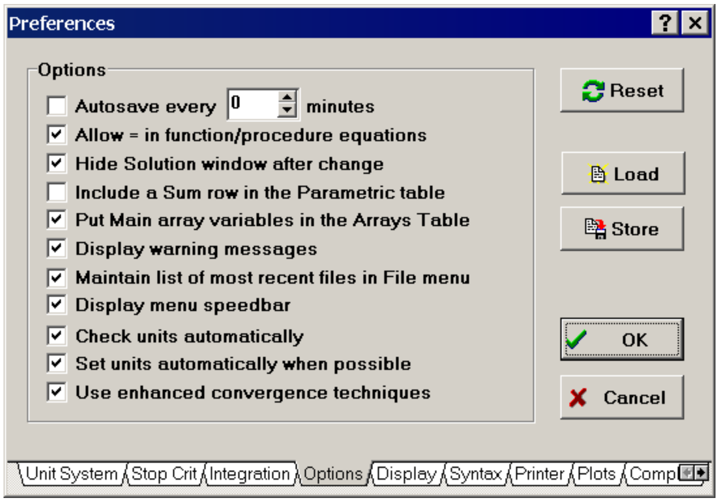
The Default Variable Information command can also be used to selectively change information for existing variables. For example, if you change the units for variables beginning with letter T to [K] and press the OK button, all existing variables beginning with letter T will take on these new units. No 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	0	
I	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 tabs for user choices concerning program options, general display options, equations and syntax display, printer output, plot window options, complex number options, directories and sounds. 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. The Reset button restores all choices to their default (as-shipped) values.



Options

- □ Autosave every 0 minutes
- ☑ Allow = in function/procedure equations
- ☑ Hide Solution window after change
- □ Include a Sum row in the Parametric table
- ☑ Put Main array variables in the Arrays Table
- ☑ Display warning messages
- ☑ Maintain list of most recent files in File menu
- ☑ Display menu speedbar
- ☑ Check units automatically
- ☑ Set units automatically when possible
- ☑ Use enhanced convergence techniques

☑ **Autosave every XXX minutes** instructs EES to save all information at regular intervals. The information is saved to a temporary .EES file that has the same name as the open EES file, but with a tilde (~) preceding the filename. For example, file C:\EES32\ABC.EES will be autosaved as C:\EES32\~ABC.EES. Under normal operation, the autosave file is deleted when a new file is created or opened or when EES is closed at the end of the worksession, so do not depend upon it for permanent storage. You should always save any file that you intend to use at a later time using the Save or Save As commands. However, if EES quits unexpectedly, the last saved copy of the backup file should still be available in your directory.

☑ **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. Only the Functions, Procedures, Modules, and Subprograms which appear in the Equations window are affected by this setting. The local values of variables in Functions, Procedures, Modules, and Subprograms 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. If the table is currently displaying results from the Uncertainty table calculation, the sum row will display the uncertainty of the summed values, i.e., the square root of the sum of the squared uncertainty values from each row.

☑ **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 or by clicking the Warnings On/Off field in the Equations window status bar.

☑ **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.)

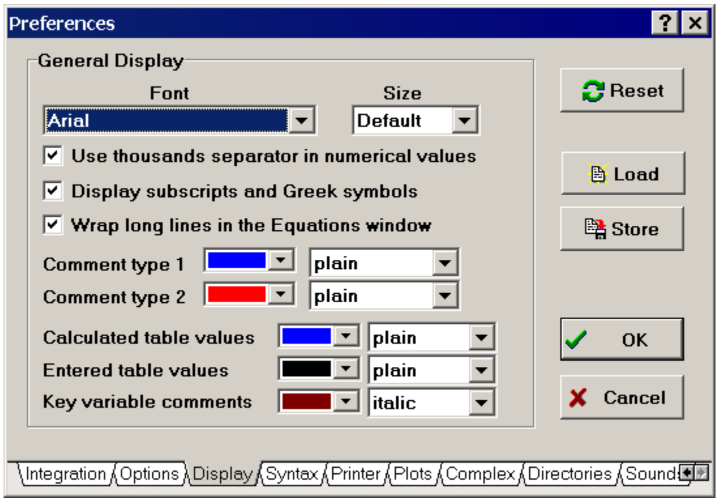
☑ **Display menu speedbar** controls the visibility of the toolbar appearing below the menu bar. The toolbar will be hidden if this control is unselected.

☑ **Check Units automatically** will, if enabled, check the unit consistency of each equation when the Solution Window or Residuals window is displayed. If the Show function/procedure/module values option in this dialog is enabled, the unit consistency for equations in internal functions, procedures, modules and subprograms will also be reported. This option can also be turned on or off with the $CHECKUNITS AutoON/AutoOff directive.

☑ **Check units automatically** will, if enabled, check the unit consistency of each equation when the Solution Window or Residuals window is displayed. If the Show function/procedure/module values option in this dialog is enabled, the unit consistency for equations in internal functions, procedures, modules and subprograms will also be reported. This option can also be turned on or off with the $CHECKUNITS directive.

☑ **Set units automatically** when possible will, if enabled, cause EES to enter the units of variables for which the units have not been previously specified. Units will be set only when a variable is by itself on the left hand side of an equal sign and a unit warning would otherwise occur. Note that the same effect can be accomplished using the $AutoSetUnits directive.

☑ **Use enhanced convergence techniques** will, if checked, algebraically rearrange equations into equivalent forms that converge more readily, reducing the need to provide guess values.



Preferences

General Display

Font	Size
Arial	Default

☑ Use thousands separator in numerical values

☑ Display subscripts and Greek symbols

☑ Wrap long lines in the Equations window

Comment type 1	🟦	plain
Comment type 2	🟥	plain
Calculated table values	🟦	plain
Entered table values	⬛	plain
Key variable comments	🟫	italic

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.

The Use thousands separator will display values with a thousands separator. However, the thousands separator is not used to input values.

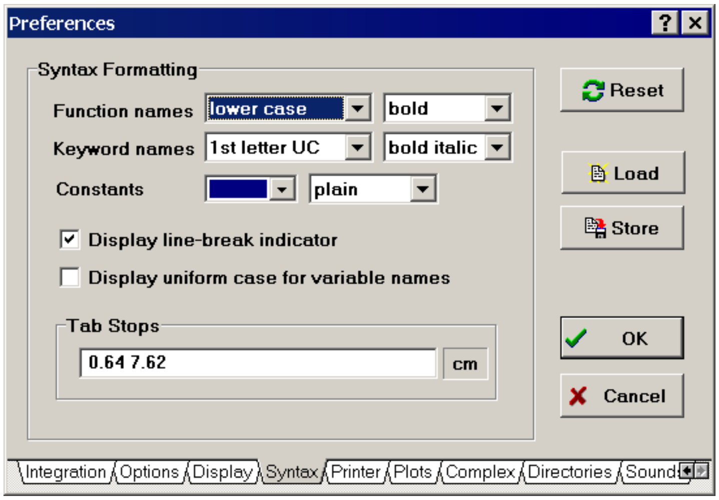
Display subscripts and Greek symbols controls the appearance of EES variables in the Solution, Formatted Equations, Parametric Table and Diagram windows. If this option is selected, the underscore will be used to signify the start of a subscript. The following characters in the variable name will be 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, _ddot, _bar, _hat or _tilde 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}.

Wrap long lines in the Equations Window will hide the horizontal scroll bar and wrap lines.

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."

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. The color of Key variable comments can be selected. They only appear in the Key tab of the Solutions window.



Preferences

Syntax Formatting

Function names	lower case	bold
Keyword names	1st letter UC	bold italic
Constants	■	plain

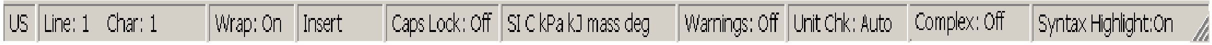
☑ Display line-break indicator

□ Display uniform case for variable names

Tab Stops

0.64 7.62 cm

The Syntax tab provides syntax highlighting of functions, keywords and constants in the Equations window. Function names (such as ENTHALPY, SIN, etc.) and keywords (such as FUNCTION, DUPLICATE, fluid names, etc.) can be displayed in with different fonts and cases, as selected with the drop-down controls. To turn off syntax highlighting, click the Syntax Highlight On/Off at the far right of the Equations Window status bar or click the Equations window button located in the button bar below the main menu.

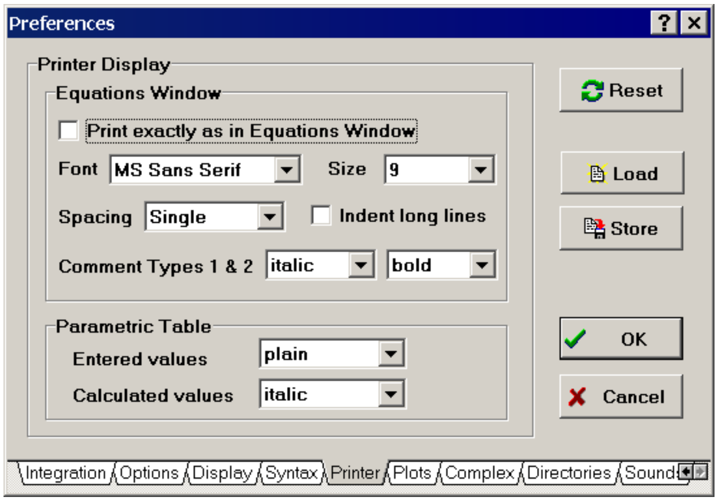


US | Line: 1 Char: 1 | Wrap: On | Insert | Caps Lock: Off | SI C kPa kJ mass deg | Warnings: Off | Unit Chk: Auto | Complex: Off | Syntax Highlight:On

☑ **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.

The Tab Stops edit field allows up to five tab stops to be set for use in the Equations window. These tab setting can also be specified with the $TABSTOPS directive which overrides this setting in the Preferences dialog.



Preferences

Printer Display

Equations Window

□ Print exactly as in Equations Window

Font: MS Sans Serif Size: 9

Spacing: Single □ Indent long lines

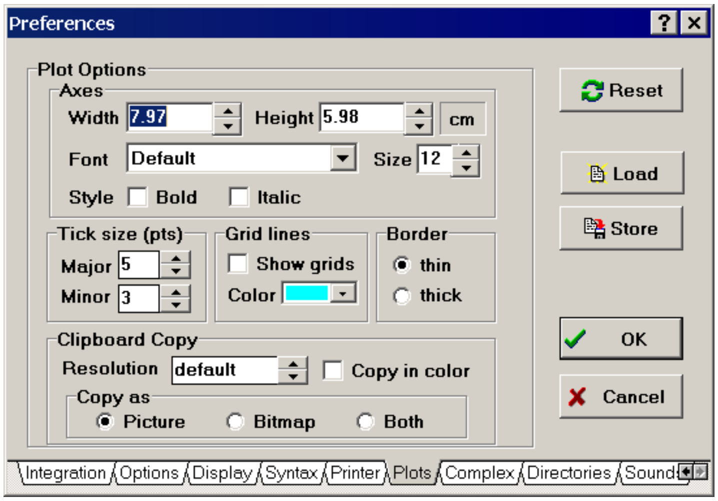
Comment Types 1 & 2: italic, bold

Parametric Table

Entered values: plain

Calculated values: italic

The options in the Printer Tab affect the appearance of the printed output. If the Print exactly as in Equations window check box is selected, the Equations window will be printed as it appears on the screen, including any highlighting that may have been applied. Otherwise, the print output can be customized for the printer. 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

Axes

Width 7.97 Height 5.98 cm

Font Default Size 12

Style □ Bold □ Italic

Tick size (pts)

Major 5

Minor 3

Grid lines

□ Show grids

Color

Border

● thin

○ thick

Clipboard Copy

Resolution default □ Copy in color

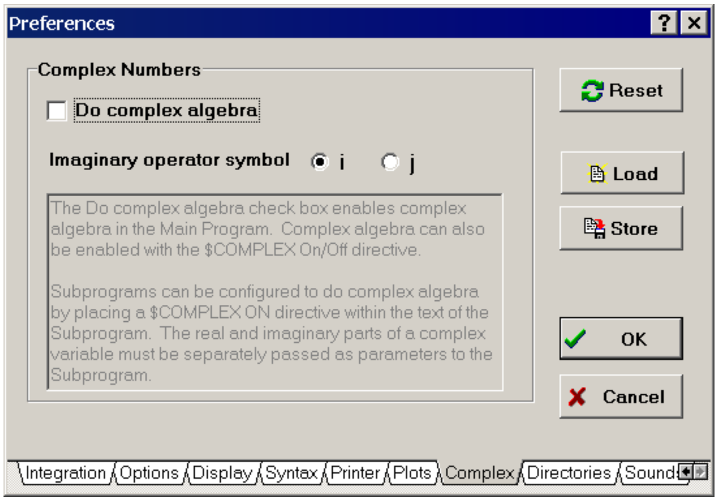
Copy as

● Picture ○ Bitmap ○ Both

The **Plot Options** 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 controls in the Clipboard Copy box affect the manner in which plots from EES are transmitted to other applications. The plot will be copied to the clipboard as a picture (enhanced metafile), as a bitmap, or as both a picture and bitmap depending on the setting of the radio buttons in the Copy as box. The resolution of the copy may be specified as 100, 300, 600, or 1200 dots per inch. The default value is 300 dots per inch. Higher resolutions generally produce a better image, but in the case of the bitmap copy, they result in much greater memory utilization particularly if the copy is done in color. After a plot has been copied to the Clipboard, it can be pasted into another application using the application's Paste or Paste Special command. If the plot is copied as both a picture and bitmap, the paste special command provides a choice of picture, picture (enhanced), bitmap, device independent bitmap. Selecting Picture will paste the enhanced metafile. The device independent bitmap will paste the bitmap. You may want to test both of these options to see which provides the best choice for a high quality image in your application.

The Copy in Color check box controls whether the Copy command will reproduce the plot in color or in black and white.



Preferences

Complex Numbers

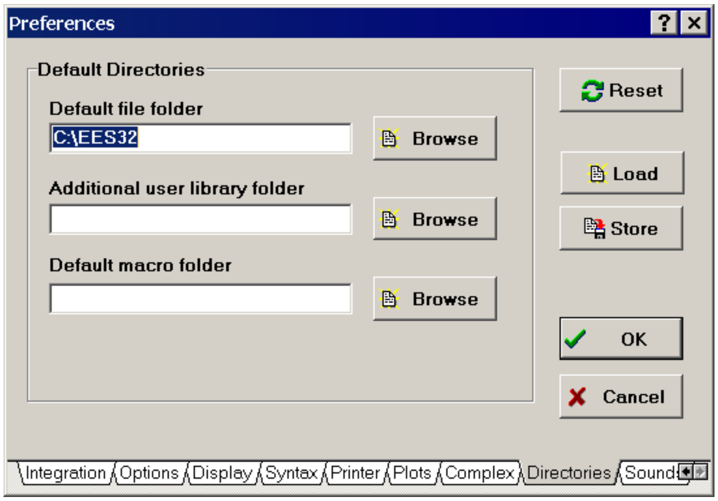
□ Do complex algebra

Imaginary operator symbol ● i ○ j

The Do complex algebra check box enables complex algebra in the Main Program. Complex algebra can also be enabled with the $COMPLEX On/Off directive.

Subprograms can be configured to do complex algebra by placing a $COMPLEX ON directive within the text of the Subprogram. The real and imaginary parts of a complex variable must be separately passed as parameters to the Subprogram.

The **Complex Numbers** 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. Complex mode can also be turned on or off but clicking on the Complex field in the status bar of the Equations window.



Preferences

Default Directories

Default file folder: C:\EES32

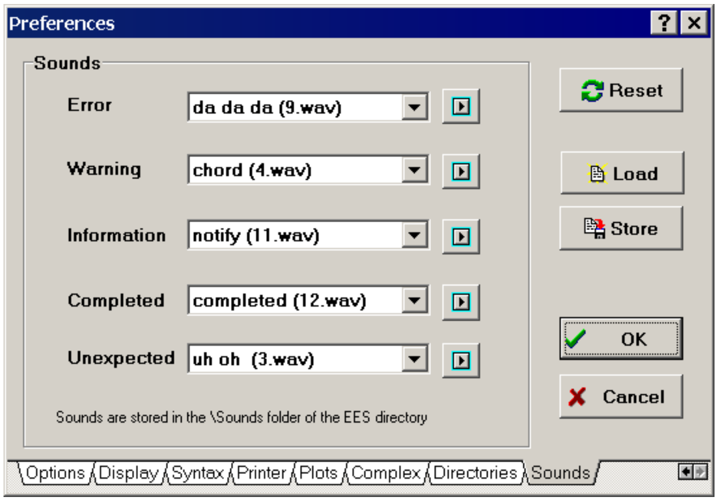
Additional user library folder:

Default macro folder:

The **Default File Folder** tab specifies the folder that EES will use for the Constants.txt, Units.txt and the units list (*.unt) files. If this field is left blank, the default file folder will be the same folder that the EES application is located in. The Constants.txt, Units.txt and default .unt files found in the specified folder will be opened and read, replacing the information that may have been read earlier. Also, EES will use the Default file folder whenever the Open or Save As... command is issued. If the directory specified in this field does not exist or if no directory is specified, EES will initially use the file folder that EES is located in and thereafter, it will use the last accessed folder. The Default file folder provides a method of allowing each use to have his or her own personal settings although EES may reside on a write-protected network. Note that the Default directory information is stored in the EES.prf file. This file must be loaded in order to determine the default directory information. EES will automatically load the EES.prf file found in the same directory as the EES application. An existing .PRF file can be loaded from any directory with the Load button in the Preferences dialog or with a $INCLUDE directive.

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 **Additional user library folder** field, EES will use also open all of the library files in this directory at startup.

The **Default Macro folder** holds files 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.



Preferences

Sounds

Error: da da da (9.wav)

Warning: chord (4.wav)

Information: notify (11.wav)

Completed: completed (12.wav)

Unexpected: uh oh (3.wav)

Sounds are stored in the \Sounds folder of the EES directory

The **Sounds** tab allows the sounds that EES generates for different situations to be customized. Click the arrow to the right of the sound file name to play the sound.

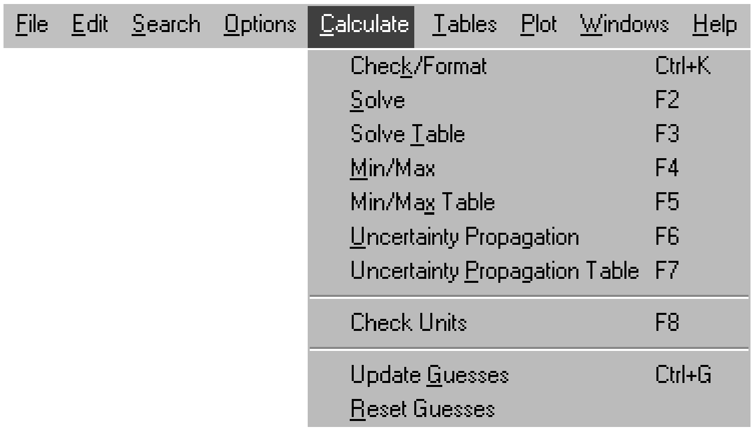
In any of the Preferences in any of the tabbed windows are changed, pressing the OK button will set the Preferences for this session only. The Store button will cause the Preferences to be permanently saved in a user-specified file. The usual name and location for this file is EES.prf in the same directory as the EES application. EES will automatically load the EES.prf file in the when it starts so that the default settings will be in effect at the start of the program the next time EES is run. However, if EES is running off of a write-protected network or if there are multiple users on the same computer, it is better to store the default setting in a personal location. These default settings can be loaded at a later time using the **Load** button in the Preferences dialog.

Note: If EES is started from a short-cut, it is possible to specify the starting directory by right-clicking on the short-cut and entering the starting directory in the short-cut properties. If EES finds the EES.PRF file that contains the user preferences in this starting directory, it will use it rather than the EES.PRF found in the EES directory. This provides a way for each user to have unique preferences.

Purge Unused Variables provides a way to remove unused variables from memory. 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, format, and key variables comment. The Commercial version of EES can store up to 6,000 variables. The Professional version allows 12,000 variables. The 64-bit Professional version allows 24,000 variables. Consequently, a small number of unused variables normally are not a concern. Where they can be a concern is when arrays are no longer being used, since an array can include a large number of individual variables. For very large problems, particularly problems using array variables, EES could run out of variable space.

The Purge Unused Variables command will examine the EES program and determine which variables are not in use. In addition, this command will determine if there are any Subprograms that are not in use. If either unused variables or Subprograms are detected they will be deleted (after a confirmation) and the memory that they required will be recovered. The purging process is relatively complicated from a programming standpoint. Before using this command, it is a good idea to save a backup copy of your file, just in case a problem occurs during the purging process.

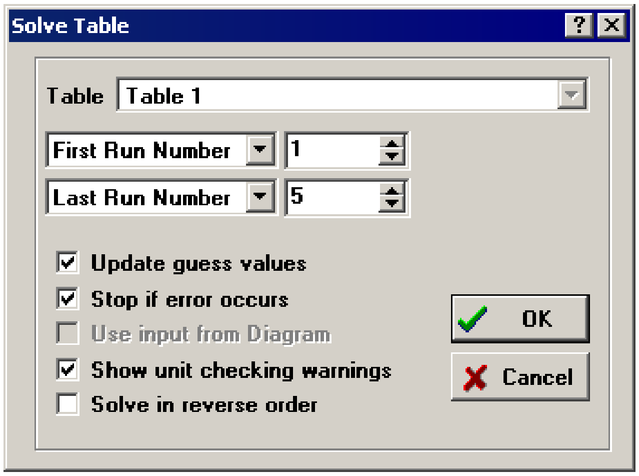
The Calculate Menu



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 Table 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 Tables”. Choosing this option results in EES doing the calculations for all of the Parametric tables in sequential order.



Solve Table

Table: Table 1

First Run Number: 1

Last Run Number: 5

☑ Update guess values

☑ Stop if error occurs

☐ Use input from Diagram

☑ Show unit checking warnings

☐ Solve in reverse 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.

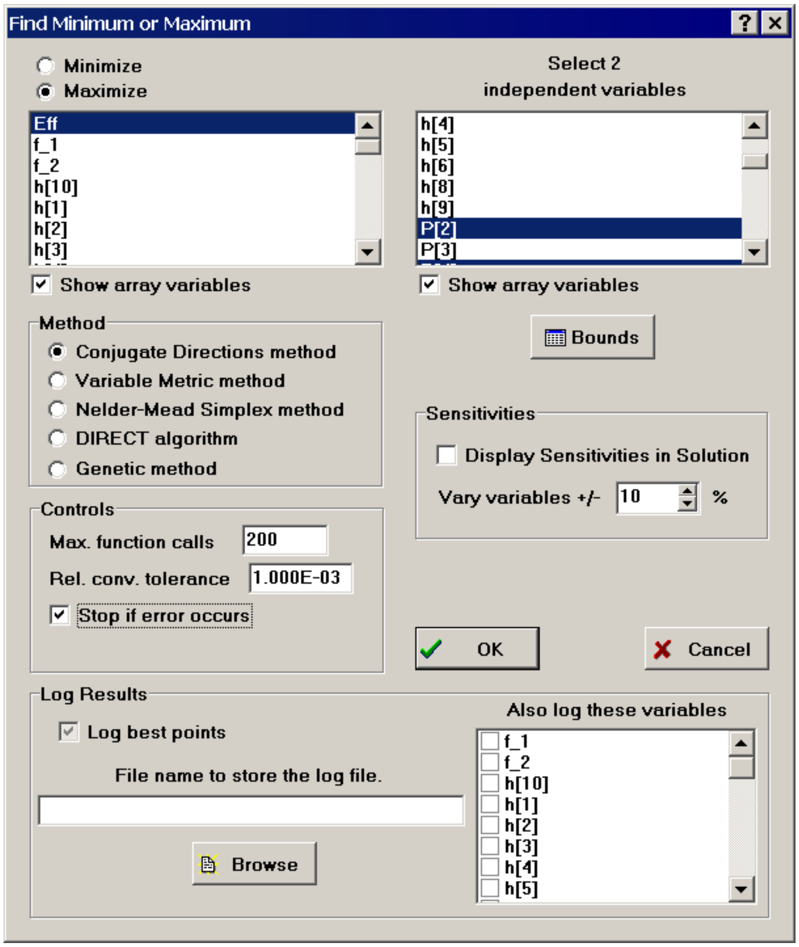
If the **Stop if error occurs** control is selected, EES will terminate the table calculations on the run at which the error occurred. Otherwise, EES will continue the calculations for the remaining rows. A message will be displayed after the table calculations are completed indicating the rows in which a error 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.

EES will automatically check units if the Check Units Automatically control in the Options tab of the Preferences dialog is selected. A warning dialog will appear if any unit problems are found. The **Show unit checking warnings** control allows this warning dialog to be disabled.

In the Professional version, the start and stop runs for the Parametric table calculations can be set to EES variables. Clicking in the First Run box will allow a selection of First Run Number or First Run Variable. If First Run Number is selected, a number corresponding to the first row in the Parametric table for which calculations are to be done is entered. If First Run Variable is selected, a drop down list appears from which an EES variable can be selected.

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 up to 20 (Commercial Version) or 80 (Professional Version) 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. 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.



Find Minimum or Maximum

○ Minimize
● Maximize

Eff
f_1
f_2
h[10]
h[1]
h[2]
h[3]

☑ Show array variables

Select 2
independent variables

h[4]
h[5]
h[6]
h[8]
h[9]
P[2]
P[3]

☑ Show array variables

Method
● Conjugate Directions method
○ Variable Metric method
○ Nelder-Mead Simplex method
○ DIRECT algorithm
○ Genetic method

Controls
Max. function calls 200
Rel. conv. tolerance 1.000E-03
☑ Stop if error occurs

Sensitivities
☐ Display Sensitivities in Solution
Vary variables +/- 10 %

Log Results
☑ Log best points
File name to store the log file.

Also log these variables
☐ f_1
☐ f_2
☐ h[10]
☐ h[1]
☐ h[2]
☐ h[3]
☐ h[4]
☐ h[5]

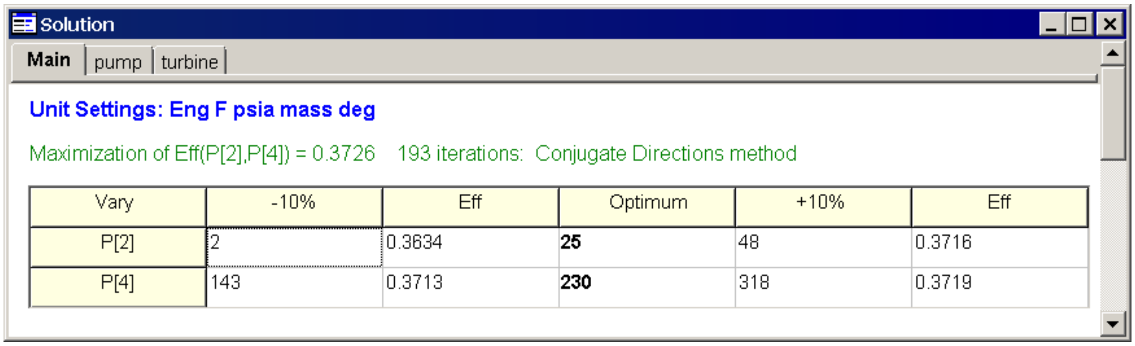
If there is one degree of freedom, EES will minimize/maximize the selected variable using either a Golden Section search or a recursive quadratic approximation method, depending on the settings of the buttons at the bottom of the dialog window. (See Appendix B for information on the optimization algorithms.) The recursive Quadratic Approximations method is usually faster, but the Golden Section method is more reliable. Multi-dimensional optimization may be done using Conjugate Directions method or the

Variable Metric algorithm. The Professional version provides three additional optimization algorithms: the Nelder-Mead, the Direct and the Genetic method. The Conjugate Directions 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. The Nelder Mead simplex method does not use derivatives and is effective for some optimization problems. The Genetic method nearly always finds the global optimum but it is slow. The Direct algorithm is also able to find a global optimum and it is usually much more efficient than the Genetic method.

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.

The Professional version allows the optimization results to be written to a file. The values of the dependent variable and the independent variables are output to a specified text file for each iteration. In addition, selected variables can also be written to the text file. The Professional version provides the option to display a Sensitivity table in the Solution window (see below) that shows how the optimum depends on each independent variable. Click the Display Sensitivities in Solution check box to enable this capability.



Unit Settings: Eng F psia mass deg

Maximization of Eff(P[2],P[4]) = 0.3726   193 iterations: Conjugate Directions method

Vary	-10%	Eff	Optimum	+10%	Eff
P[2]	2	0.3634	**25**	48	0.3716
P[4]	143	0.3713	**230**	318	0.3719

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 Table 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 up to 12 (50 for the Professional version) calculated variables as a function of the uncertainties of up to 30 (200 for the Professional version) 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.

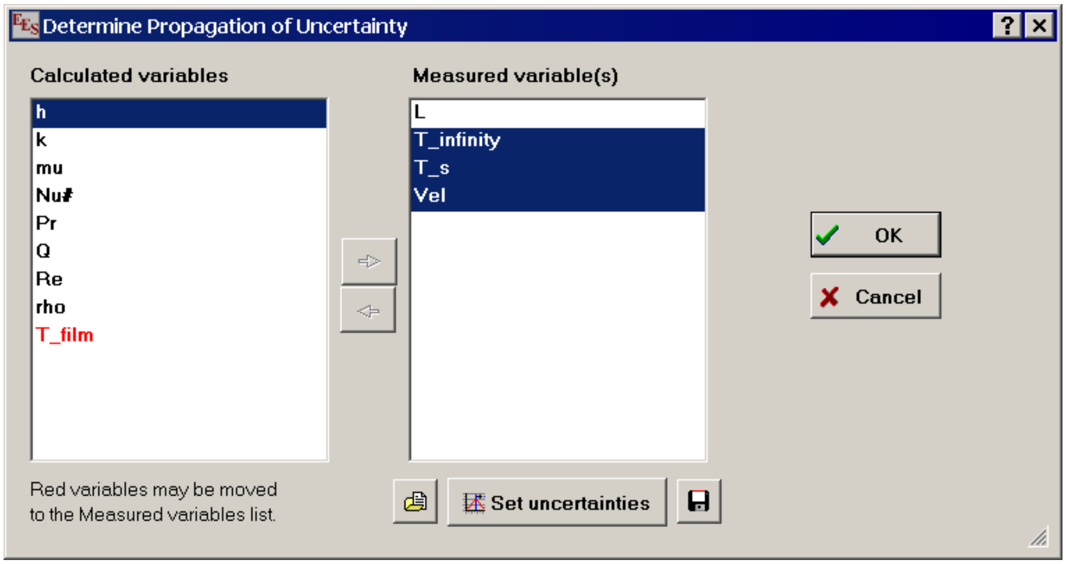
The purpose of this command is to calculate how the uncertainties in all of the measured variables propagate into the value of the calculated quantity, Y. The method for determining this uncertainty propagation is described in NIST Technical Note 1297 (Taylor B.N. and Kuyatt, C.E., Guidelines for Evaluating 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

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

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.

Note that the variables initially appearing in the Measured variables list are constants, i.e., values that are independent of the values of other variables. However, some variables (shown in red) that initially appear in the Calculated variable list can be moved to or from the Measured variables list by selecting the variables and clicking the left or right arrow buttons between the lists as seen in the figure below.

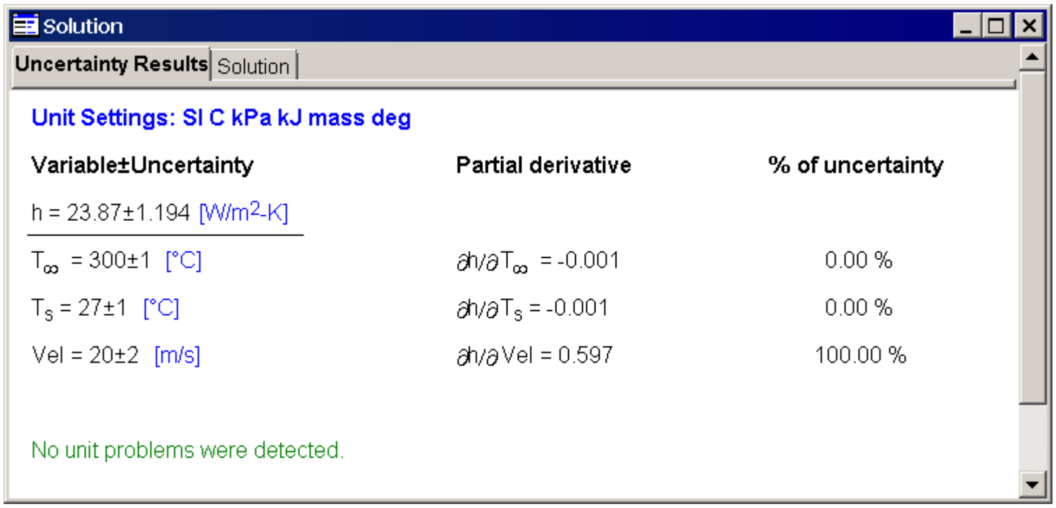
The Uncertainty Propagation dialog allows multi-selection of sequential variables by clicking the first variable, pressing the Shift key, and then clicking the last variable.



Red variables may be moved to the Measured variables list.

In the Professional version, the choices made for the calculated and measured variables and their associated relative or absolute uncertainties can be saved by clicking the Save Uncertainty Information button in the Uncertainty Propagation or Uncertainty Propagation Table dialog located to the left of the Set uncertainties button. These choices can later be restored by clicking the Open Uncertainty Information button in these dialogs located to the right of the Set uncertainties button.. Saving the uncertainty information allows one EES file to solve several different uncertainty problems without having to reenter the uncertainty settings. The Save Uncertainty Information saves a text file having an .UNC file name extension.

After the calculations are completed, EES will display a Solution window with two tabs, as shown below. The Uncertainty Results tab displays 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. In addition the % of the total uncertainty in the calculated variable resulting from the uncertainty in each measured variable will also be displayed. This % is calculated as the ratio of the square of the product of the partial derivative and uncertainty of each measured variable to the square of the uncertainty in the calculated variable. The Solution tab shows the Solution window as it would normally appear without the uncertainty calculations.



Unit Settings: SI C kPa kJ mass deg

Variable±Uncertainty	Partial derivative	% of uncertainty
h = 23.87±1.194 [W/m2-K]		
T_∞ = 300±1 [°C]	$\partial h/\partial T_\infty$ = -0.001	0.00 %
T_S = 27±1 [°C]	$\partial h/\partial T_S$ = -0.001	0.00 %
Vel = 20±2 [m/s]	$\partial h/\partial$Vel = 0.597	100.00 %

No unit problems were detected.

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. NOTE: The calculated and measured variables must all be in the Parametric Table. 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. Click the Set Uncertainties button to set the uncertainty value associated with each measured variables. Note that the uncertainty value can be provided as a variable in the Parametric table. The Parametric Table calculations will proceed after the OK button is selected. 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. If the Sum row in the table has been made visible with the 'Show Sum Row in Parametric table'option in the Preferences dialog or with the $SUMROW ON directive, the sum row will display the uncertainty of the summed values which is the square root of the sum of the squared uncertainty values from each row.

Check Units will check the dimensional and unit consistency of all equations in the Equations window. The unit checking will be extended to equations in internal functions, procedures, Modules and Subprograms if the Show Function/Procedure/Module values option in the Options tab of the Preferences dialog is selected or if a $LOCAL ON directive is provided in the Equations window. 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 Info dialog window or in the Solution window. In addition, selecting the variable in the Equations window and right-clicking the mouse will bring up a pop-up menu with a Variable Info menu item that allows the units of the selected variable to be entered. The units of a constants in the Equations window can be set by following the value with the units enclosed in square brackets. For example:

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

Unit consistency can be obtained by specifying the units of a constant or by using the Convert function. 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. This problem can be corrected by either entering the units for 12

```
L_inch=L_feet*12 [in/ft]
```

or by using the Convert function.

```
L_inch=L_feet*convert(ft,in)
```

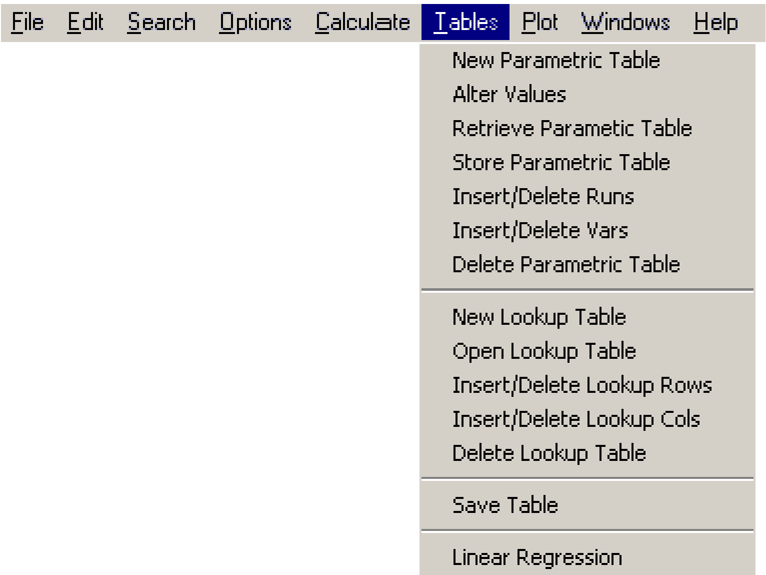
The Check Units command will display the equation and an explanatory message for each equation that is found to have dimensional consistency or unit consistency errors. If you click the left mouse button on an equation, the focus will jump to that equation in the Equations window. If you click the right 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.

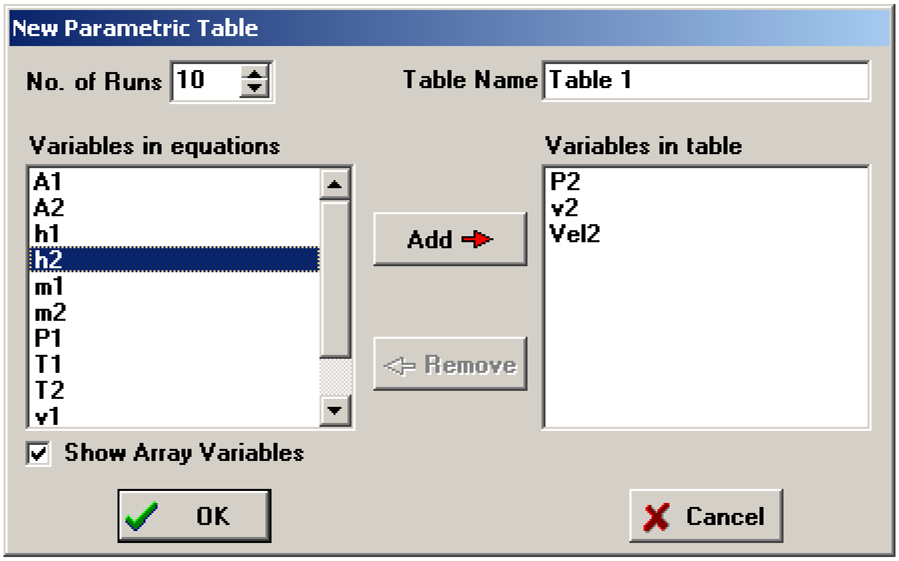
Reset Guesses replaces the guess value of each variable in the Equations window with the default guess value for that variable. Unless otherwise specified, EES assumes all guess values are 1.0. You can change the default guess values with the Default Info command in the Options menu. You should reset the guess values only if you are experiencing convergence difficulties and you have changed the guess values in an attempt to find a solution. Reset Guesses will not change the lower and upper limits.

Reset Limits replaces the lower and upper limits of each variable in the Equations window with the default limits for that variable.

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.


New Parametric Table

No. of Runs 10

Table Name Table 1

Variables in equations: A1, A2, h1, h2, m1, m2, P1, T1, T2, v1

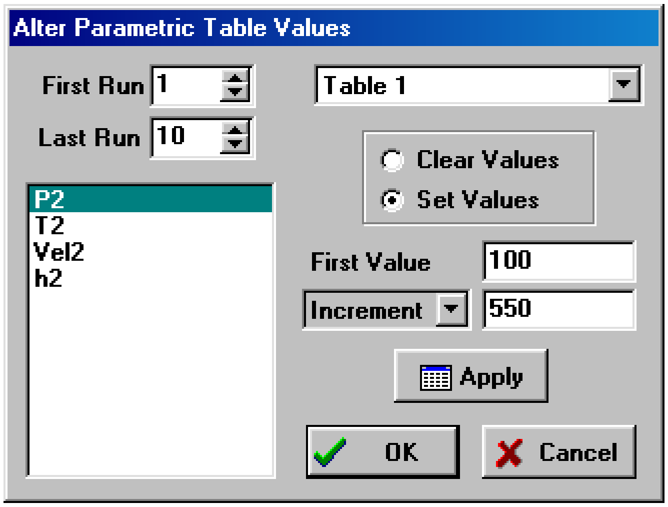
Variables in table: P2, v2, Vel2

☑ Show Array Variables

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 selected 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 in one or more tables. 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

When there is more than one Parametric table, the Alter Values command initially shows data for the Parametric table that is formost. If there is more than one Parametric Table, clicking on the table name will allow other Parametric tables to be chosen. It is possible to have the Alter Values dialog operate on more than one table. To do so, click on the table name or the down arrow at the right of the table name. The list of available Parametric tables will be displayed. Hold the Shift key down while clicking on the table names in this dropdown list to select or unselect the tables for which the changes will be applied.

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 .TXT or .PAR (binary) 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 text (.txt) or binary (.par) file. 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. However, these commands make it possible to import/export Parametric tables to other EES programs or to external programs such as EXCEL.

Insert/Delete Runs allows the number of runs in an existing Parametric Table (or for selected tables) 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.

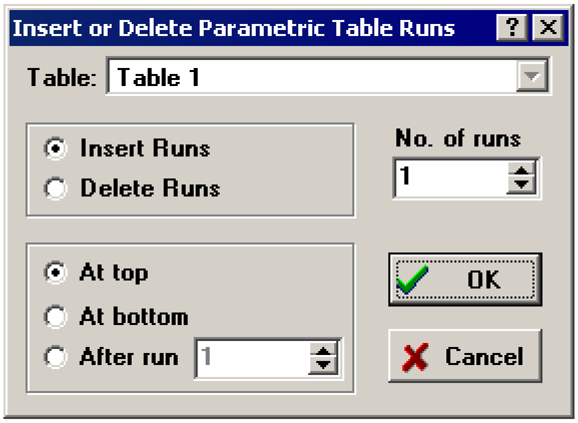


Table: Table 1

● Insert Runs
○ Delete Runs

No. of runs: 1

● At top
○ At bottom
○ After run 1

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

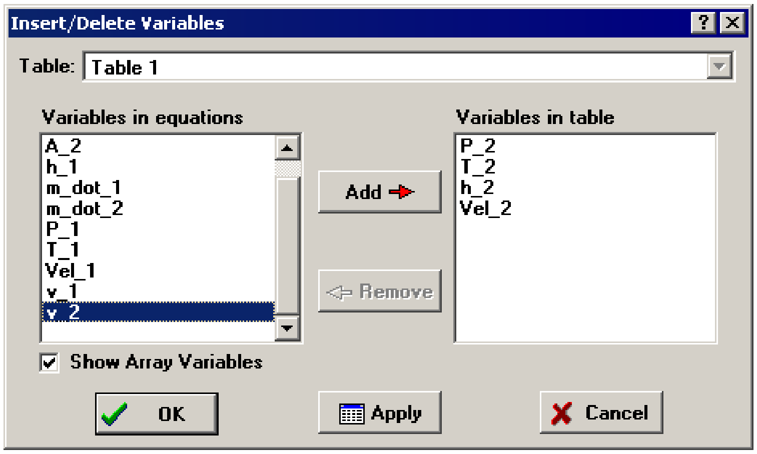


Table: Table 1

Variables in equations: A_2, h_1, m_dot_1, m_dot_2, P_1, T_1, Vel_1, v_1, v_2

Variables in table: P_2, T_2, h_2, Vel_2

☑ Show Array Variables

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 selected. Click the

Add button to move the selected 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.

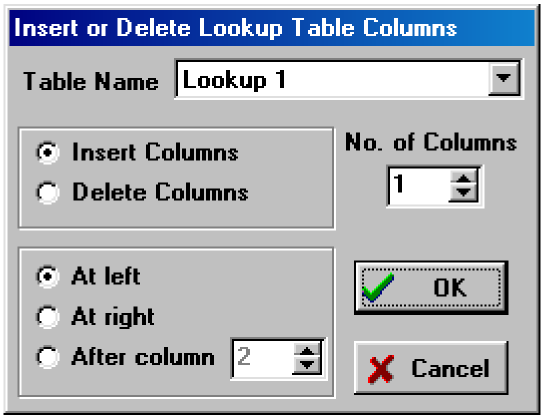
Variables will appear in columns of the Parametric Table in the same order as they appear in the list on the right. The order of the variables in this list can be changed by pressing and holding the left mouse button on a variable name while sliding it up or down to a new position in the list. The column 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 an .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. EES can also read files written by EXCEL having an .xlsx file name extension. See Chapter 4, Using Lookup Files and the Lookup Table, for details.

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

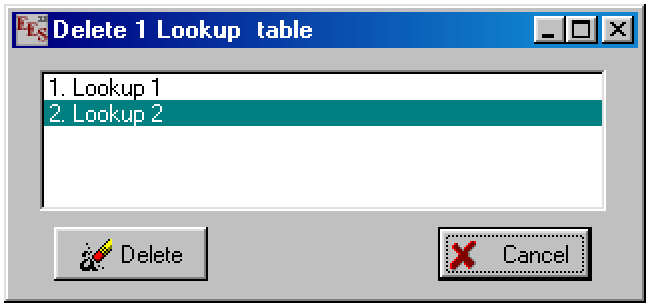
Table Name Lookup 1

● 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

Save Table provides the ability to save any EES table (Parametric, Lookup, Arrays or Integral) to a file on disk. There are three possible file formats and any one of the three formats can be applied with this command. In addition, the Save Table command allows to be saved with the rows and columns transposed. The three file formats are identified with the filename extension which can be *.txt, *.csv or *.lkt.

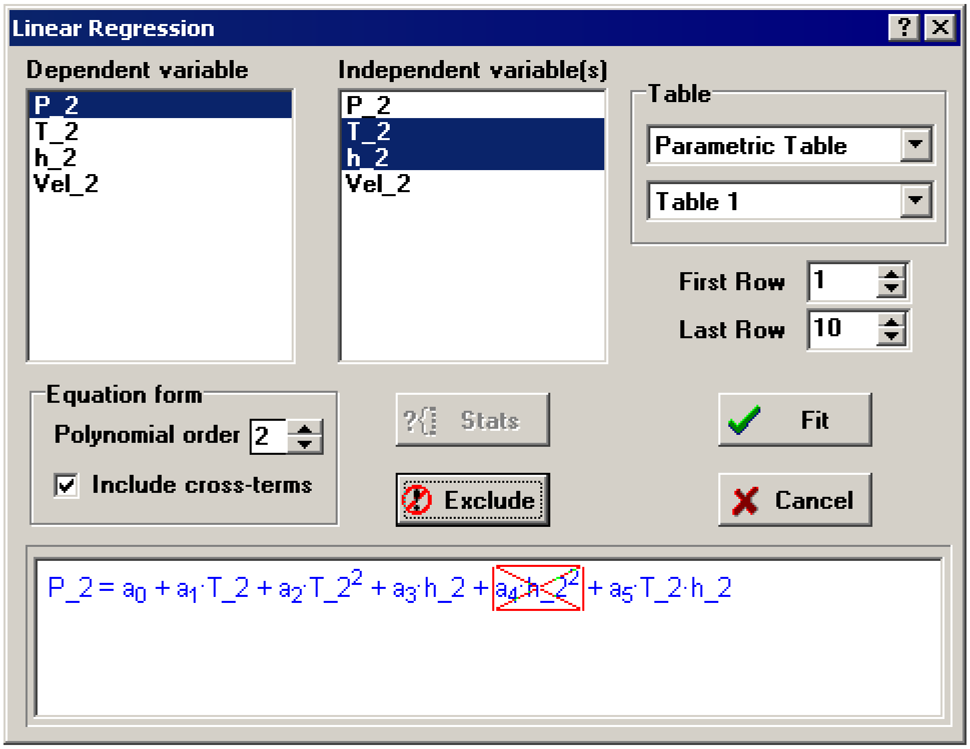
The .txt file saves the data in a .txt file with tabs between each value and a carriage return at the end of a row of data. Most spreadsheet programs can read data in this format. If the Include column header information checkbox is checked, the name and unit information of each column in the table will be written to the file. If the Use EES Lookup file text format checkbox is selected, the .txt file will be written in a Lookup table format that allows the column names and units, format information and the data to be read into an EES Lookup table using the Open Lookup Table command. If the Transpose rows and columns checkbox is selected, the rows and columns in the table will be interchanged before the data are stored in the disk file. Transposed data cannot be saved using the .lkt format.

The .CSV format saves only the data (no column names or formatting information) as a comma-delimited ASCII file. (Note semicolons are used for European format.) The .CSV format is convenient when you wish to export the data to another application, such as a spreadsheet.

The binary format (.LKT) saves the data in the table in the EES binary Lookup Table format. All information in the file including the column heading, units, and display format are saved. The file can then be read into another EES program with the Open Lookup Table command but it is not readable by external programs.

Data stored in a Lookup File on disk can be directly accessed with Interpolate, Differentiate, LOOKUP, LOOKUPCOL and LOOKUPROW functions. Note that data can also be moved from any EES table to another table or another application with the Copy and Paste commands.

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 9 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.



Linear Regression

Dependent variable

P_2
T_2
h_2
Vel_2

Independent variable(s)

P_2
T_2
h_2
Vel_2

Table

Parametric Table

Table 1

First Row 1

Last Row 10

Equation form

Polynomial order 2

☑ Include cross-terms

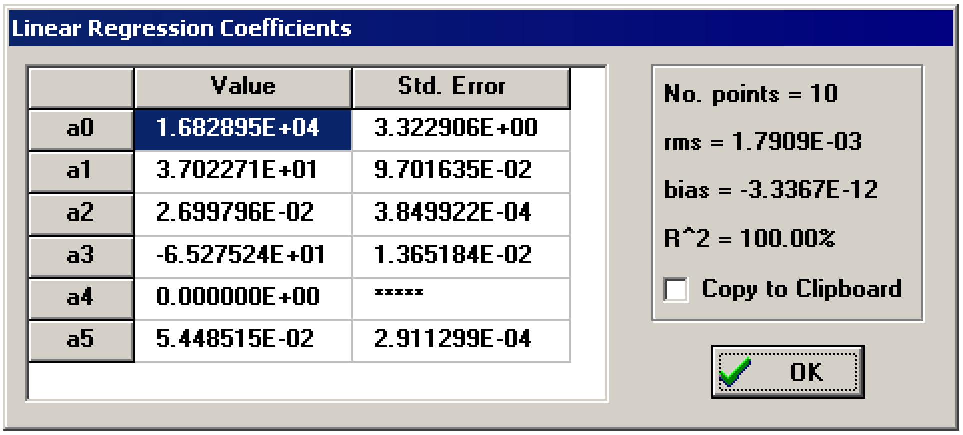
$P\_2 = a_0 + a_1 \cdot T\_2 + a_2 \cdot T\_2^2 + a_3 \cdot h\_2 + \cancel{a_4 \cdot h\_2^2} + a_5 \cdot T\_2 \cdot h\_2$

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 9 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 Exclude button will become the Plot button. Clicking the Plot button will display a plot of the predicted values of the dependent variable versus the values in the table. 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	1.682895E+04	3.322906E+00
a1	3.702271E+01	9.701635E-02
a2	2.699796E-02	3.849922E-04
a3	-6.527524E+01	1.365184E-02
a4	0.000000E+00	*****
a5	5.448515E-02	2.911299E-04

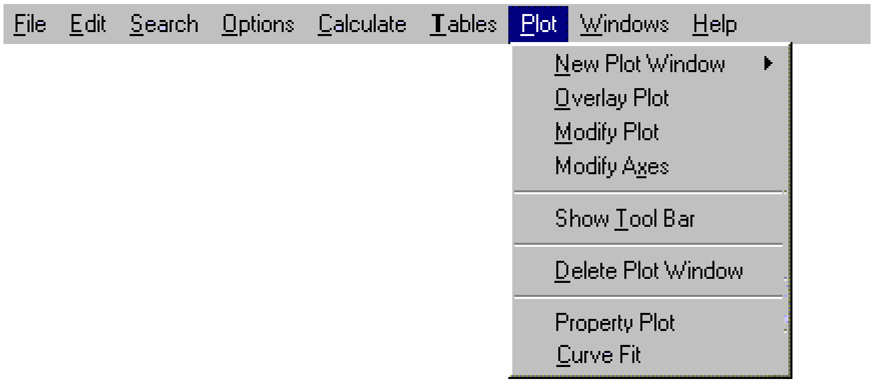
No. points = 10

rms = 1.7909E-03

bias = -3.3367E-12

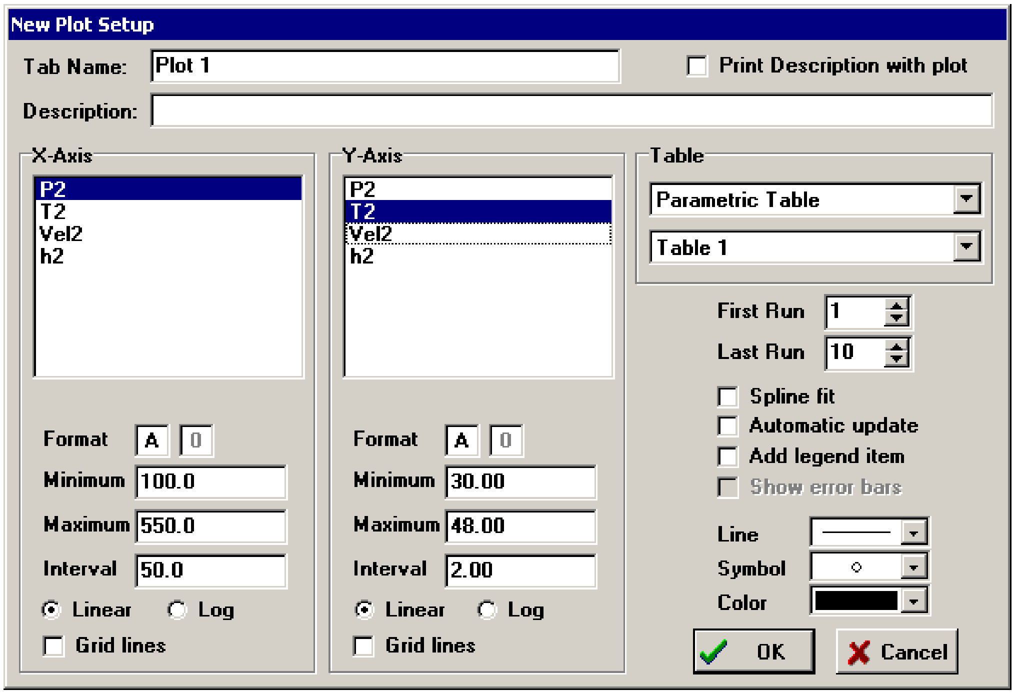
R^2 = 100.00%

The Plot Menu



New Plot Window will display a child menu to generate an X-Y, bar, or contour involving two 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. The plot windows 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

Tab Name: Plot 1 ☐ Print Description with plot

Description:

X-Axis: P2, T2, Vel2, h2

Y-Axis: P2, T2, Vel2, h2

Table: Parametric Table; Table 1

	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	● Linear ○ Log
	☐ Grid lines	☐ Grid lines

First Run 1

Last Run 10

☐ Spline fit
☐ Automatic update
☐ Add legend item
☐ Show error bars

Line ———
Symbol ○
Color ■

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 based on the selected row range 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.

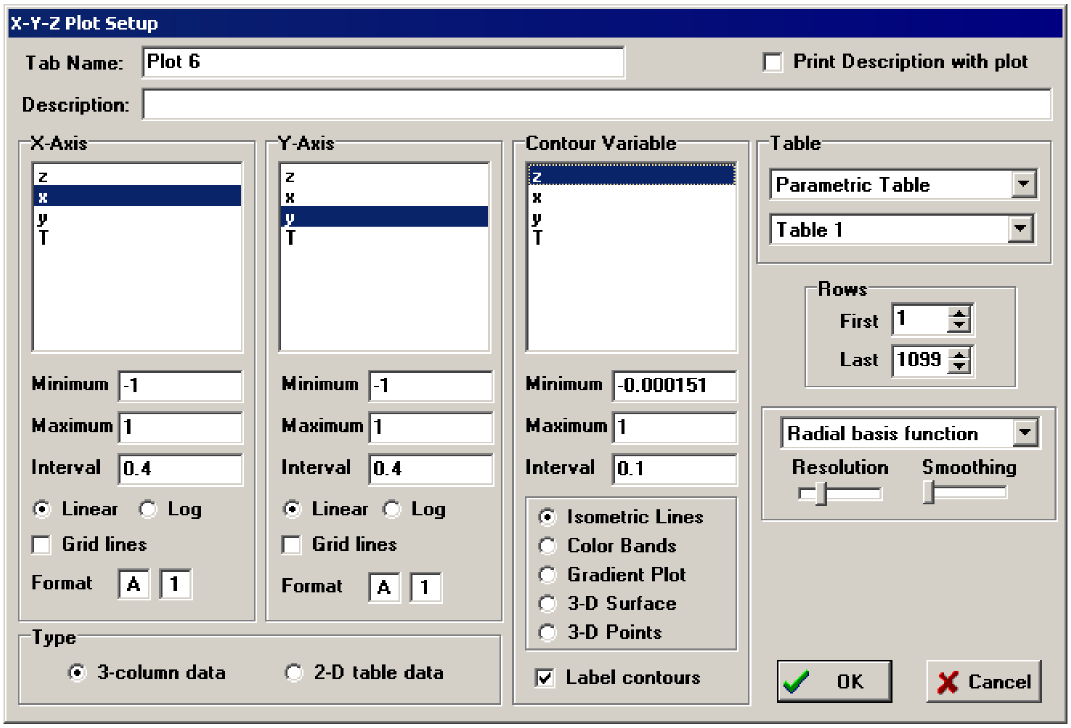
When the ‘Automatic update’ control is selected, the plot will be generated using the current data in the Parametric Table, rather than the data which existed when the plot was first drawn. The plot will be updated as the data in the table are changed. The Data button allows the number of rows and the 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. By default, error bars are generated in both the X and Y directions. The X and Y error bars can be individually controlled with the Modify Plot command.

Side-by-side bar plots can be selected from the New Plot Setup dialog when two or more bar plots are selected for plotting. The Modify Plot command provides a control for the width of the bar plots.

X-Y-Z Plots



X-Y-Z Plot Setup

Tab Name: Plot 6

□ Print Description with plot

Description:

X-Axis: z, x, y, T
Minimum -1
Maximum 1
Interval 0.4
● Linear ○ Log
□ Grid lines
Format A 1

Y-Axis: z, x, y, T
Minimum -1
Maximum 1
Interval 0.4
● Linear ○ Log
□ Grid lines
Format A 1

Contour Variable: z, x, y, T
Minimum -0.000151
Maximum 1
Interval 0.1
● Isometric Lines
○ Color Bands
○ Gradient Plot
○ 3-D Surface
○ 3-D Points
☑ Label contours

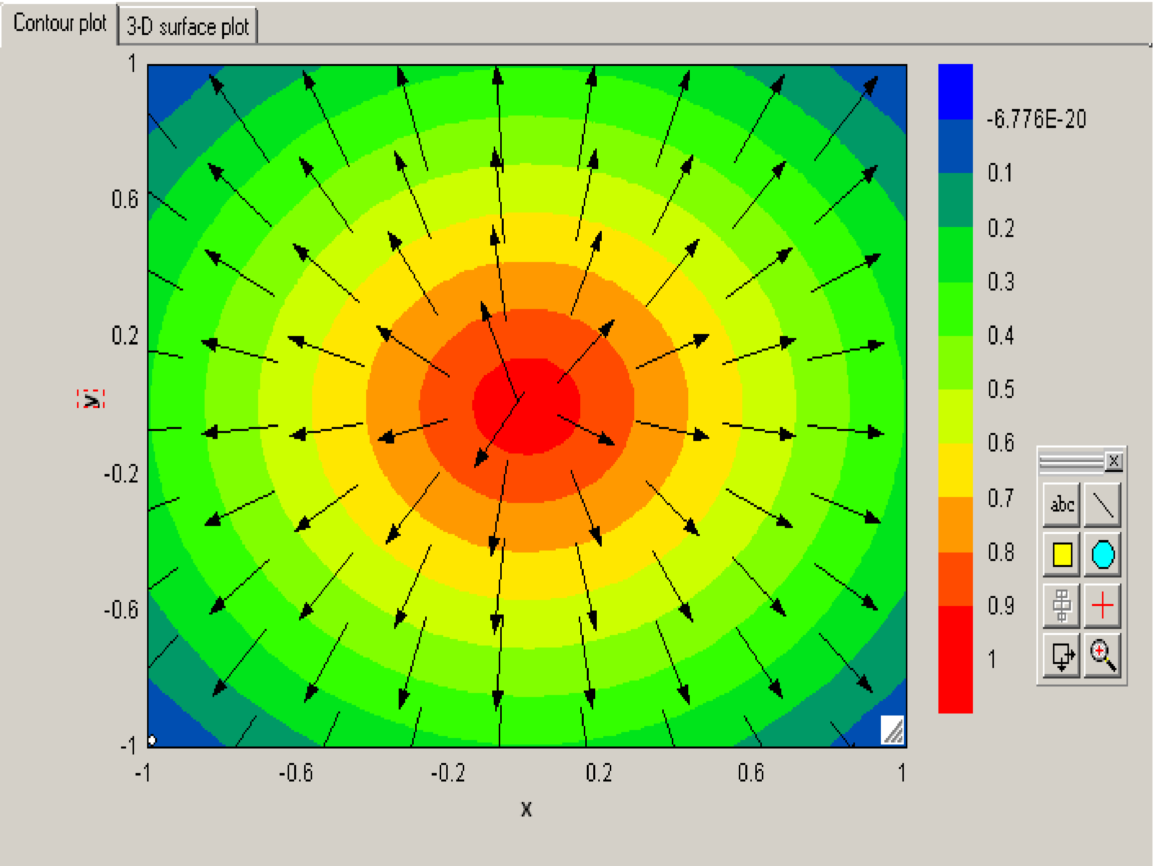
Table: Parametric Table
Table 1
Rows
First 1
Last 1099

Radial basis function
Resolution
Smoothing

Type
● 3-column data ○ 2-D table data

The X-Y-Z plot option generates contours lines and contour color bands indicating the path of a fixed value of the contour variable (Z) in X-Y space. Gradient arrows may be optionally drawn on the contour plot. In addition, a 3-D plot option is provided in the EES3D.exe application. An X-Y-Z plot requires three-dimensional data for construction. The data may be provided in either of two ways from the Parametric, Lookup, Arrays or Integral table windows. If the 3-column data radio button is selected, EES will expect the 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.



A contour plot is shown above with superimposed gradient arrows. See the Plot Window section of Chapter 3 for an example of a 3-D plot. To prepare a 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. Two methods are provided: Bi-Quadratic Polynomials and Radial Basis Functions.

When Radial Basis Function is chosen, the data are first scaled and then fitted to a function of the form

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

where N is the number of data points, the w values are weighting factors, and s is a relative smoothing parameter which, based on experimentation, is best set to zero for numerically-generated data. However smoothing may be needed for experimental data or for sparse data sets. Moving the Smoothing slider bar to the right increases s which increases the smoothing effect.

The weights are found by solving the linear system of N equations that force the above equation to perfectly represent the provided data. Once the weights are determined the equation is used to interpolate or extrapolate the values of Z to particular values of x and y. The computations needed to determine the weights are O(N cubed) so there is a practical upper limit to the number of points (N) that can be accommodated. This upper limit is controlled by the Resolution slider. The range of the Resolution slider is 400 to 1600. 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. At full resolution, it is necessary to solve a 1600 x 1600 system so you may have to wait a while for the plot to be prepared depending on the speed of your computer.

When Bi-Quadratic Polynomial is chosed, the data are fitted to a second-order polynomial of the form: $Z\left(X,Y\right)=a_o+a_1X+a_2X^2+a_3Y+a_4Y^2+a_5XY$

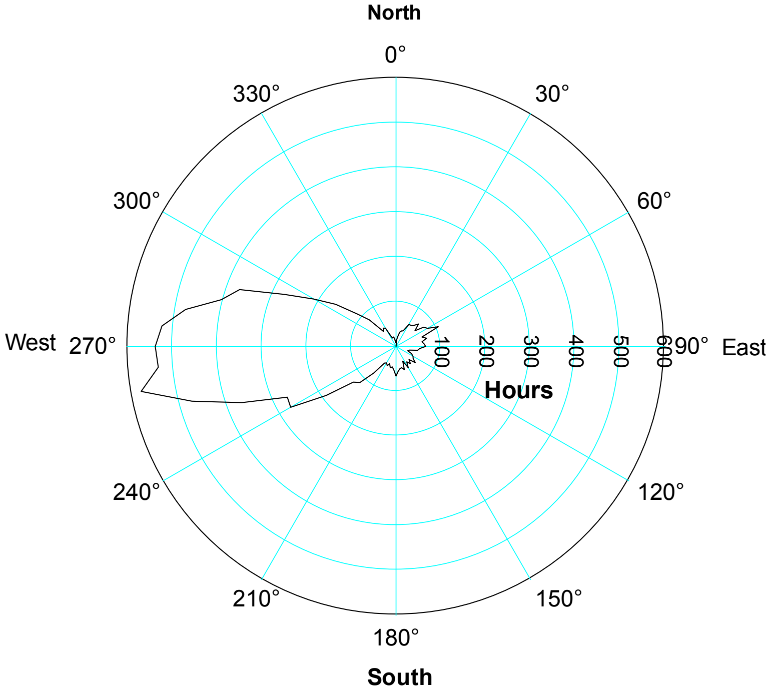
A separate set of the a coefficients is fitted for each point in the grid using weighted linear least squares. The number of grid points, controlled by the setting of the Resolution slider, ranges from 2,500 to 10,000.

Contour plots may be produced using isometric lines or color bands representing constant values of Z. The number of lines or bands is determined by 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 and Shift keys down and moving the mouse. An X-Y or bar plot can be overlaid on 2 two-dimensional contour plot using the Overlay Plot command.

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 selecting the cross-hairs tool from the tool menu and moving the mouse.

Polar Plot

The New Plot Window | Polar Plot command is available in the Professional version of EES. It allows variables defined in the Parametric Table, the Lookup Table, the Arrays Table or the Integral Table to be plotted as a function of any other variable in the table. The plot is drawn in a new tabbed plot window, as shown below. The information needed to produce the plot is specified in the New Polar Plot Setup dialog. All of the information specified in this dialog window can be modified later using the Modify Axes and Modify Plot commands. See the online help for details.

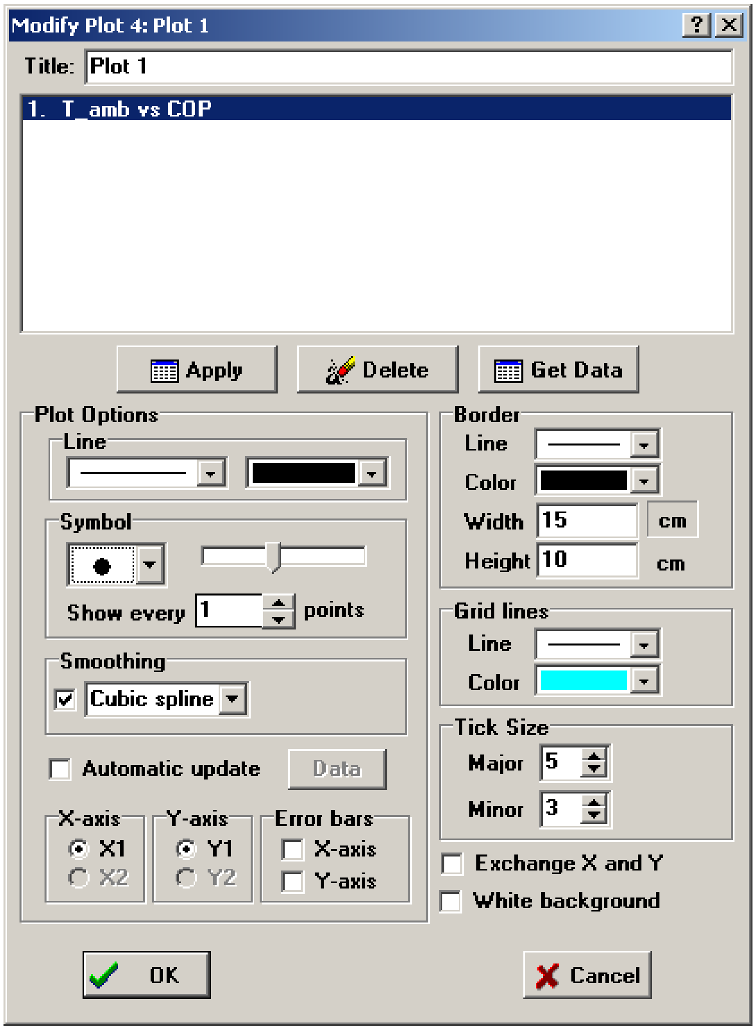


Overlay Plot allows a new plot curve to be drawn over existing plots. The use of this command is identical to that for the New Plot command described above except that it does not first clear the Plot window. Controls are provided to create or select an existing top x-axis or right y-axis.

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. A (Y2) following a variable name indicates that it is plotted using the right Y-axis. Similarly (X2) indicates that the plot uses the X-axis at the top of the plot. The X-axis and Y-axis scales employed for the selected plot are indicated with the radio buttons at the lower left of the dialog. The axis scales can be changed by clicking on these buttons.

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. 'Automatic update' options may be changed. (See the New Plot Window command for a description of this options.



Title: Plot 1

1. T_amb vs COP

Show every 1 points

☑ Cubic spline

□ Automatic update

X-axis ● X1 ○ X2

Y-axis ● Y1 ○ Y2

Error bars □ X-axis □ Y-axis

Width 15 cm

Height 10 cm

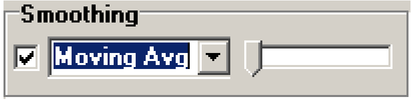
Major 5

Minor 3

□ Exchange X and Y

□ White background

To enable smoothing, click the check box within the Smoothing group. There are three smoothing options, accessible with a menu box control. The first option is a cubic spline fit that smoothly tries to direct the line through all points. The second option is a polynomial fit. The order of the polynomial is increased by moving the slider bar to the right. The third option is a moving average that is intended to reduce scatter when experimental data are plotted. A slider will appear as shown below. Moving the slider to the right increases the number of data points that are used to compute the moving average and thus reduces the scatter in the data. The plot is updated as the slider is moved so the effects of a change can be viewed.



Controls are provided to change the plot window title, the size and characteristics of the plot border, the grid lines and the major and minor ticks. Negative tick sizes are displayed outside of the plot rectangle. 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.

Note that multiple plots can be selected by holding the Shift key down while clicking (or dragging) the mouse in the plot line list. In this case, the plot line parameters will be displayed for the last selection. If a change is made to a parameter, that change will be applied to all selected plots when the Apply or OK button is clicked. This capability makes it easy to, for example, change the size of the symbols for all plots in the plot window at one time. Other parameters can be changed in a similar manner. However, the data sources (accessed with the Data button) are disabled when more than one plot is selected. It is possible to change the range (First Row and Last Row) of all selected plots at one time provided that they were all plotted from the same table.

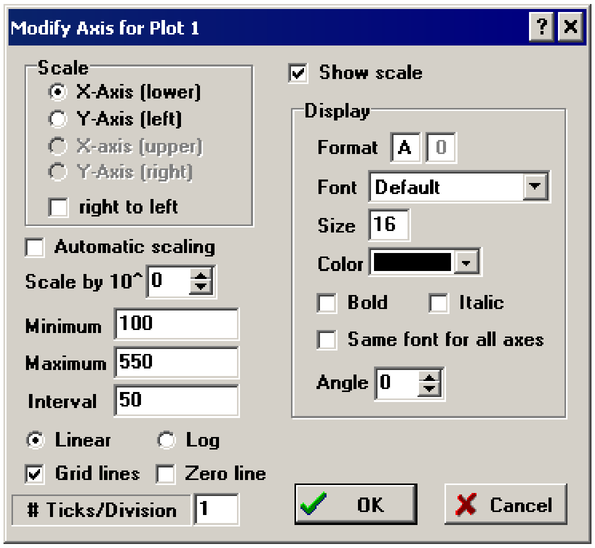
Clicking the Get Data button will allow the recover the data used to generate the plot. The data will be saved to a Lookup table having a name entered in the prompt that appears when the button is clicked.

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. EES will automatically select the number of minor tick marks for new X-Y plots. 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. The axis values can be scaled by a factor of 10^n, where n is an integer provided by the user. The Professional version also allows the angle of the scale numbers to be specified.

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 (lower)
○ Y-Axis (left)
○ X-axis (upper)
○ Y-Axis (right)
□ right to left

□ Automatic scaling
Scale by 10^ 0
Minimum 100
Maximum 550
Interval 50
● Linear ○ Log
☑ Grid lines □ Zero line
Ticks/Division 1

☑ Show scale
Display
Format A 0
Font Default
Size 16
Color ■
□ Bold □ Italic
□ Same font for all axes
Angle 0

Note that the 2nd axis scales (initially located at the top of the plot for the X axis and to the right of the plot for the Y-axis) can be moved to other locations by holding the Ctrl key down while pressing the arrow keys. The up/down arrows control the position of the X2 scale and the left/right arrows control the position of the Y2 scale.

Show/Hide Toolbar controls the visibility of the tool bar that is provided with each 2-D 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. On 3-D plots, this command toggles the visibility of the 3-D control panel.

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.

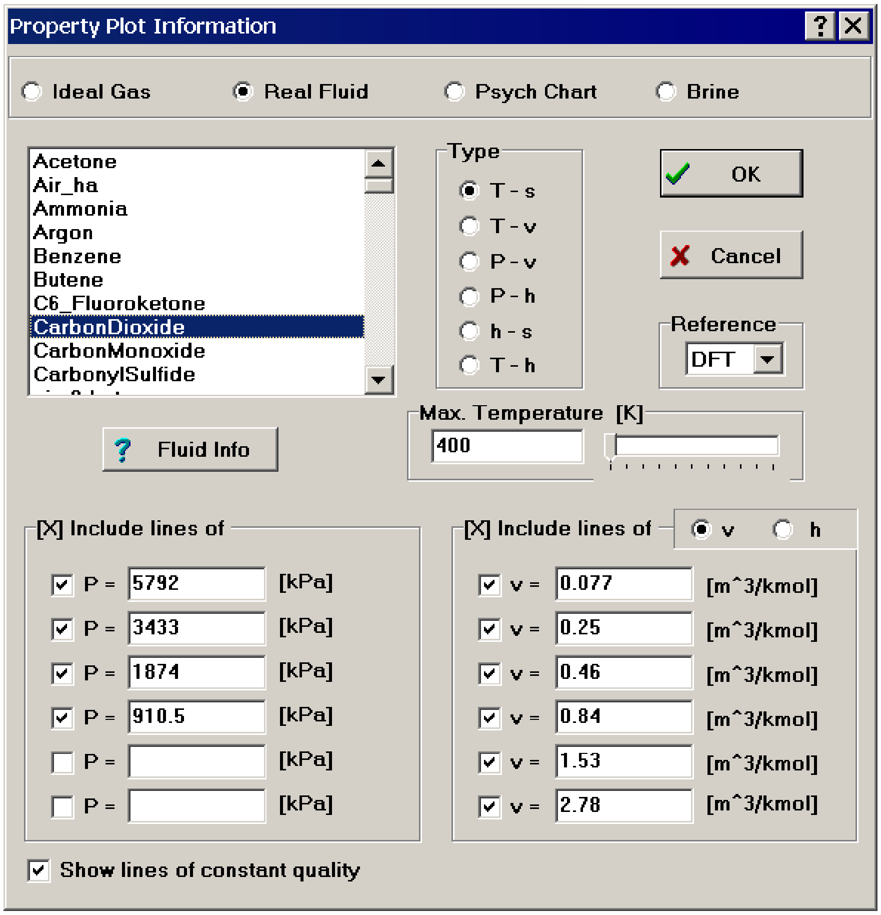
Property Plot creates a new plot window with thermodynamic property data for a selected fluid. The fluid can be an ideal gas, real fluid. air-water mixture or a brine. Once created, additional property data or thermodynamic cycle state points can be superimposed on the plot using the Overlay Plot command. 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 fields in which the total pressure and temperature limits 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. For some fluids, a choice of reference state is available for the values of enthalpy and entropy. See the $Reference directive Chapter 7 for more information.

The Maximum Temperature control allows the maximum temperature in T-S, T-V and T-h plots to be selected from a range between the default and the maximum temperature for which the correlation is valid.



Property Plot Information

○ Ideal Gas ● Real Fluid ○ Psych Chart ○ Brine

Acetone
Air_ha
Ammonia
Argon
Benzene
Butene
C6_Fluoroketone
CarbonDioxide
CarbonMonoxide
CarbonylSulfide

Type
● T - s
○ T - v
○ P - v
○ P - h
○ h - s
○ T - h

Reference: DFT

Max. Temperature [K]: 400

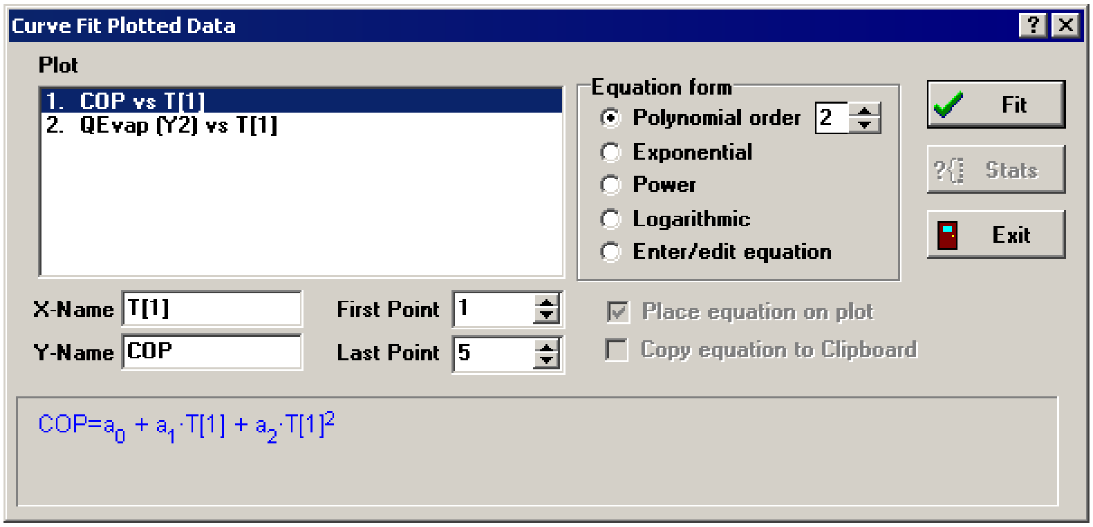
[X] Include lines of
☑ P = 5792 [kPa]
☑ P = 3433 [kPa]
☑ P = 1874 [kPa]
☑ P = 910.5 [kPa]
□ P = [kPa]
□ P = [kPa]

[X] Include lines of ● v ○ h
☑ v = 0.077 [m^3/kmol]
☑ v = 0.25 [m^3/kmol]
☑ v = 0.46 [m^3/kmol]
☑ v = 0.84 [m^3/kmol]
☑ v = 1.53 [m^3/kmol]
☑ v = 2.78 [m^3/kmol]

☑ 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.

Choose the data to be fitted from the list of plots at the left. Note that data can be plotted from the Parametric, Lookup, Integral or 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 be prompted to supply guess values and bounds for the unknown parameters.



Curve Fit Plotted Data

Plot

1. COP vs T[1]
2. QEvap (Y2) vs T[1]

Equation form

- ● Polynomial order 2
- ○ Exponential
- ○ Power
- ○ Logarithmic
- ○ Enter/edit equation

X-Name T[1]

Y-Name COP

First Point 1

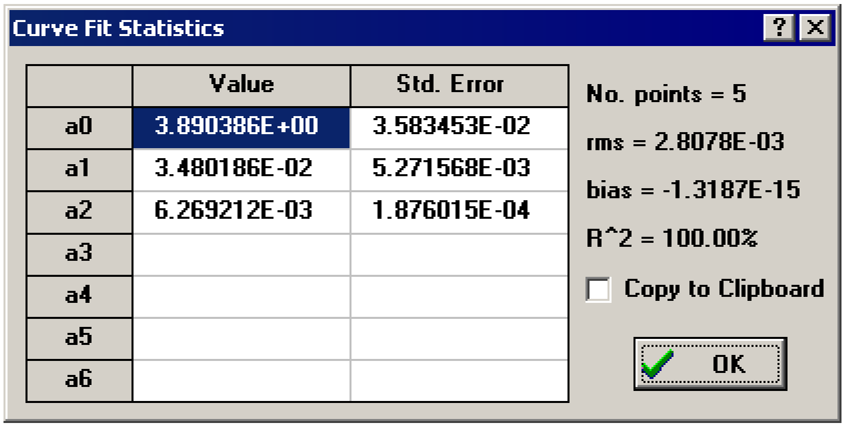
Last Point 5

☑ Place equation on plot

□ Copy equation to Clipboard

$COP=a_0 + a_1 \cdot T[1] + a_2 \cdot T[1]^2$

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.



Curve Fit Statistics

	Value	Std. Error
a0	3.890386E+00	3.583453E-02
a1	3.480186E-02	5.271568E-03
a2	6.269212E-03	1.876015E-04
a3		
a4		
a5		
a6		

No. points = 5

rms = 2.8078E-03

bias = -1.3187E-15

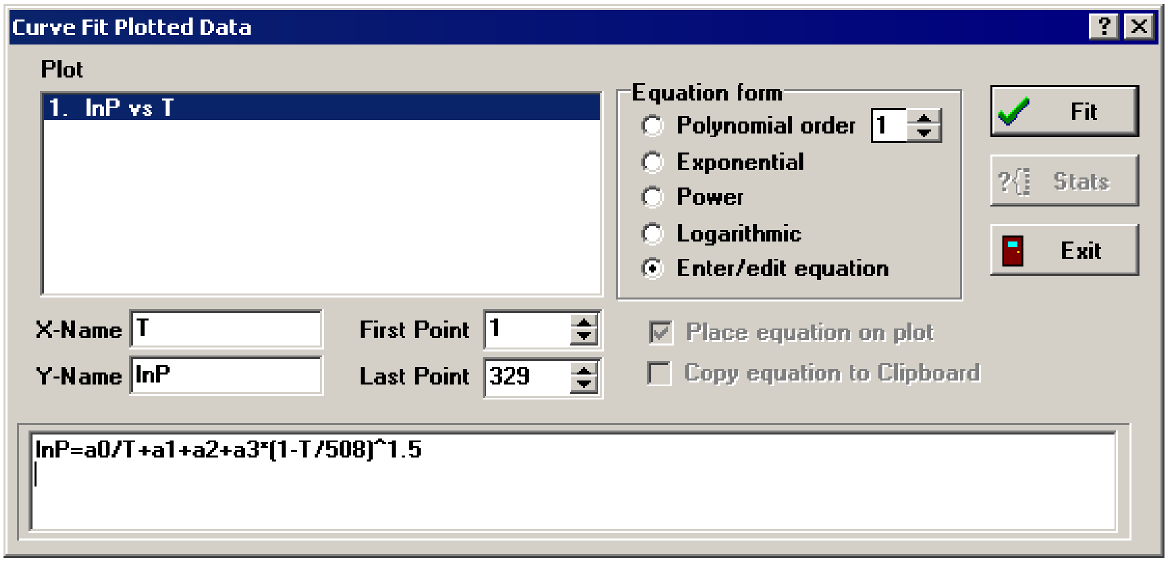
R^2 = 100.00%

□ 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 Enter/Edit Equation option allows the user to enter a curve fit equation of the form - YName=f(X-Name) with up to 7 unknown coefficients. X-Name and -YName are the names of the X and Y variables appearing in the edit fields just above the equation. By default, X-Name and Y-Name are set to the names appearing in the plot axis labels. However, other names may be entered. The coefficients must be entered as a0, a1, a2, ... a6. Upper case or lower case letters may be used. It is not necessary to use the coefficients in any order. The equation must begin with Y-Name= and involve X-Name and at least one coefficient. An example is shown in the figure below.



Plot

1. lnP vs T

Equation form

○ Polynomial order 1
○ Exponential
○ Power
○ Logarithmic
● Enter/edit equation

X-Name T

Y-Name lnP

First Point 1

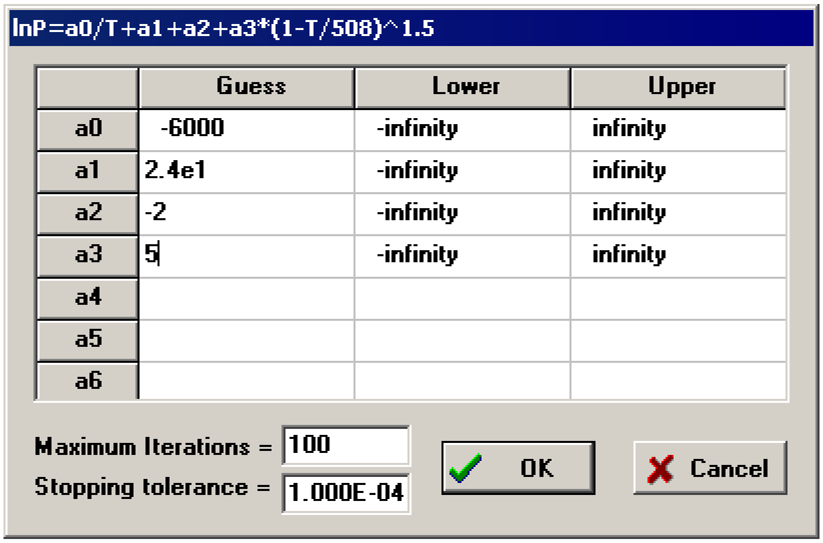
Last Point 329

☑ Place equation on plot

□ Copy equation to Clipboard

```
lnP=a0/T+a1+a2+a3*(1-T/508)^1.5
```

Note that the equation you enter may be either linear or nonlinear in the coefficients. EES will employ a nonlinear least squares method to determine the coefficients. The parameter estimation algorithm requires that you supply a guess value for each of the parameters. The Curve Fit Guess Values window will appear after you click Fit.



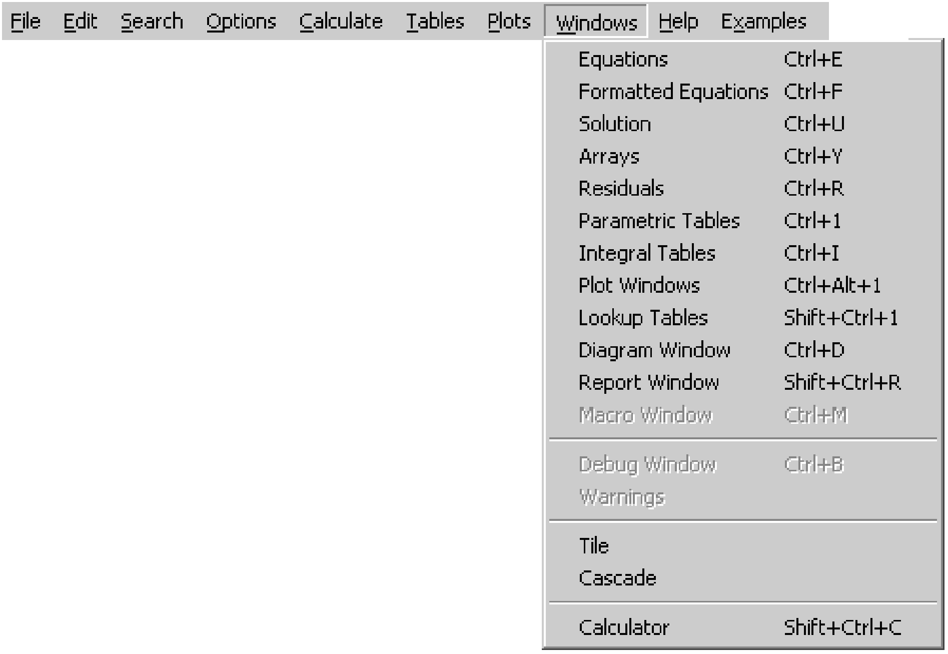
lnP=a0/T+a1+a2+a3*(1-T/508)^1.5

	Guess	Lower	Upper
a0	-6000	-infinity	infinity
a1	2.4e1	-infinity	infinity
a2	-2	-infinity	infinity
a3	5	-infinity	infinity
a4			
a5			
a6			

Maximum Iterations = 100

Stopping tolerance = 1.000E-04

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 windows. 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.

Report Window provides access to the Report Window which operates like a word processor but with the capability to substitute the current values of EES variables.

Macro Window provides access to the Macro window.

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.

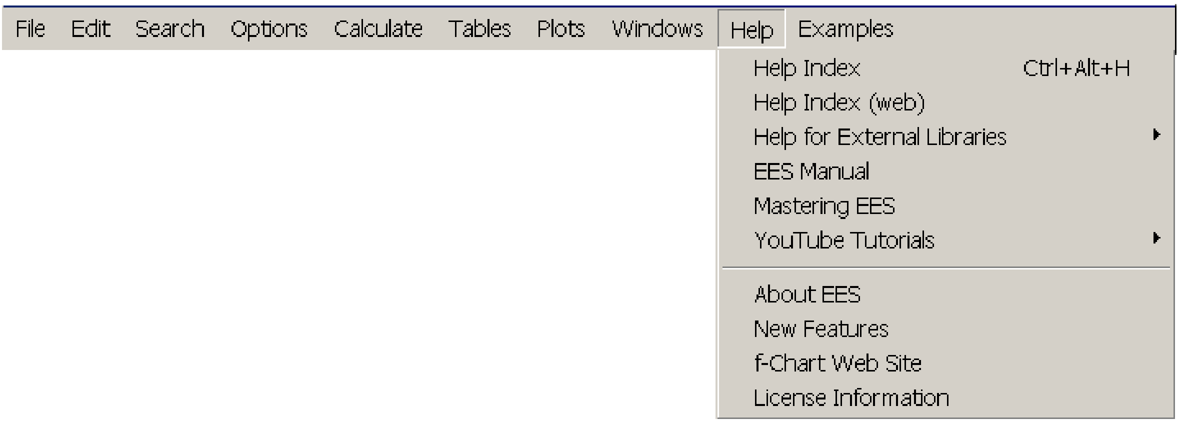
Warning will bring up a window showing warning messages that were issued during the preceding calculation. Warning will appear only if the Display Warning Messages setting in the Options Tab of the Preferences dialog is selected or if the $Warnings On directive appears in the Equations window. If no warning messages exist, the menu item will be disabled. Note that the generation of warning during calculations may reduce the calculation speed. Turn warnings off in calculation-intensive problems.

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 sizes of each window are maintained if possible.

Calculator activates the Calculator window.

The Help Menu



Help Index will activate the built-in 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.

Help Index (Web) will access the help files http://fchart.com using your browser rather than the built-in Help processor.

Help for External Libraries provides access to .HLP or .HTM files created by a user to document external routines written in EES itself or in a compiled language such as C++, Pascal, or FORTRAN. External functions and procedures that are placed in the \USERLIB folder are automatically loaded at startup. External routines can have a separate help file having an .HLP or .HTM filename extension. Because these routines are not truly part of EES, the help files cannot be included within the EES help file. However, access to the help file for any external routine is provided in the Function Information dialog by selecting the desired function and clicking the Function Info button or directly with the Help for External Libraries command.

EES Manual will start Abode Acrobat and display the electronic version of this manual which is in file ees_manual.pdf. Note that ees_manual.pdf must be in the same folder as the EES application.

Mastering EES will check to see if the Mastering EES version you own is up to date and offer the option of downloading a newer version if not.

YouTube Tutorials will display a submenu with access to short YouTube videos on how to use EES.

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.

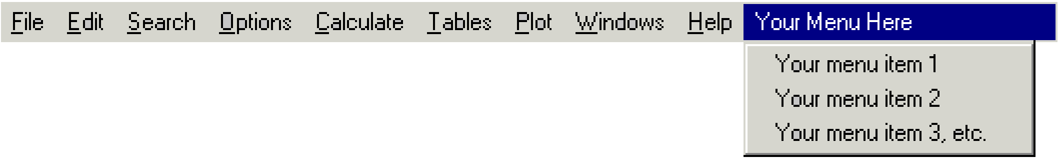
New Features shows a list of new features in EES ordered by version number and date.

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.

License Information provides informaton about the EES program and the expiration date of Instant Update and Technical Service (IUTS). If the IUTS subscription is currenbt, a link to the download the lateste version of EES from the F-Chart website is provided.

The Help menu may also contain a menu item to provide specific help for a problem that has been accessed with the Textbook menu, as described in the next section.

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. The combination of >- will place a break line in the menu. Note that a second level of menu nesting is supported. If a menu item name is preceded with >>, this menu item will become a 'flyout' menu which is displayed when the parent menu item is chosen.

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. The help file, if provided, can be accessed from an additional menu item that is placed in the Help menu. 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.

EES currently uses the Textbook menu to display examples. The Examples menu provides convenient access to a number of instructive EES programs that may be of use to a new user. You can remove this menu by moving the Examples folder out of the USERLIB folder or be deleting the Example.TXB file.

Chapter 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].

avgLookup ('TableName', 'ColumnName', RowStart, RowStop) returns the average of all or selected cells in a specified column of a specified Lookup table or Lookup file on disk. RowStart and RowStop are optional. If they are not provided, all rows in the Lookup table will be averaged. 'TableName' is a string constant or string variable that contains the name of the Lookup table. The name of the Lookup table is shown on its tab at the top of the table. 'ColumnName' is a integer value, string constant or string variable the identifies the column. If an integer value is provided, it is taken to be the column number in the Lookup table. If a string constant is provided, it is assumed to be the name of the variable for the column in the Lookup table. The single quotes that normally identify a string constant are not required. If a string variable is provided, it must contain the name of one of the columns in the Lookup table.

avgParametric('TableName', 'ColumnName', RowStart, RowStop) provides the same column average capability as described for avgLookup, but for a Parametric table.

beep # can only be used in internal functions or procedures. # is an optional sound indicator that must be an integer between 1 and 5. The function produces a sound that may be useful for alerts or debugging.

besseli(j, x**)** returns the value of jth-order Modified Bessel function of the first kind for argument value x where x>0. If j=0, this function automatically calls the bessel_I0 function.

besselj(j, x**)** returns the value of jth-order Bessel function of the first kind for argument value x where x>0. If j=0, this function automatically calls the bessel_J0 function.

besselk(j, x**)** returns the value of jth-order Modified Bessel function of the second kind for argument value x where x>0. If j=0, this function automatically calls the bessel_K0 function.

bessely(j, x**)** returns the value of jth-order Bessel function of the second kind for argument value x where x>0. If j=0, this function automatically calls the bessel_Y0 function

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

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

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

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

bessel_K1(X**)** returns the value of first-order Modified Bessel function of the second kind for argument value X where 0≤ 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.

ceil(x) returns the lowest integer greater than or equal to X.

chi_square(x, k) returns the value of the cumulative distribution function *F(x,k)* for the argument value x and k degrees of freedom.

$$F(x,k)=\frac{\int_0^{x/2} t^{k/2-1}e^{-t}dt}{\int_0^{\infty} t^{k/2-1}e^{-t}dt}=\frac{\gamma\left(\frac{k}{2},\frac{x}{2}\right)}{\Gamma\left(\frac{k}{2}\right)}$$

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. Combinations 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.

diagramheight#('Name') and **diagramwidth#**('Name') return the height and width in pixels of the Diagram window or child diagram window that has the specified name. The main diagram window is called 'Main'. If the specified name can not be found, this function returns the height of the main diagram window. The height of the Diagram window can be converted into inches using the PixelsPerInch# function. Knowing the height of the Diagram window is useful when moving object such as during animation.

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. A warning message will be generated if the command attempts to extrapolate data. Warnings will not be visible unless the Display Warning Messages control in the Options tab of the Preferences dialog is checked.

differentiate1 is used exactly like the **differentiate** function. The only difference is that differentiate1 uses linear interpolation whereas the differentiate function uses cubic interpolation.

differentiate2 is used exactly like the **differentiate** function. The only difference is that differentiate1 uses quadratic interpolation whereas the differentiate function uses cubic interpolation.

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.

factorial(x) results the factorial of argument X.

fileexists(T$) function returns 1 if file T$ is found or 0 otherwise. This function is most useful in IF-THEN-ELSE statements and in macros.

floor(X**)** will return a value equal to the largest integer value that is less than or equal to the value of X. Note that FLOOR differs from TRUNC in that FLOOR when X is less than zero. For example TRUNC(-2.5) = -2 whereas FLOOR(-2.5) = -3.

gamma_(X**)** returns the Γ (GAMMA) function for the argument value X. The Gamma function is defined as: $\Gamma(x)=\int_0^\infty t^{x-1}e^{-t}dt$. The Gamma function is approximated as a series expansion as described by Abramowitz and Stegun, "Handbook of Mathematical Functions", Dover Publications, 10th printing, 1972.

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 choose 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 the main program. 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.

interpolate2D('Filename', 'X', 'Y', ‘Z’, X=value1, Y=value2, N**)** returns an interpolated or extrapolated value as a function of two independent variables from tabular data. A minimum of 8 data points are required. These data can be in a Lookup table in the Lookup Table Window or a Lookup file (stored on disk). This function extends the capability of the Interpolate function which returns a value as a function of one independent variable.

'Table Name' is a string constant or string variable 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 either the binary (.LKT) or ASCII (.TXT, .CSV) Lookup file Formats.

X, Y, and Z are the names of three different columns in the specified table. These arguments can be string constants or string variables. Since there is no ambiguity, the single quotes that are normally required for string constants are optional.

The next two parameters identify the two independent variables and set their values. The format is ColumnName=Value where the text to the left of the equal sign must be one of the three column names specified in the second, third and fourth parameters. Note that if a string variable was provided for the column name, then the same string variable must be used for this parameter. Value1 and value2 are numerical values or expressions.

The last parameter, N, is optional. It serves two purposes. First, it indicates which of two different interpolation algorithrms are used. If N < 0, then a bi-quadratic method is used in the interpolation. If N > 0 then a multi-quadric radial basis function interpolation method is used. The absolute value of N is the maximum number of data points used in the interpolation process. The two-dimensional interpolation algorithm is computationally intensive for large values of N. If the data table contains more than N points, EES will use the N points in the table that are nearest to the point for which the interpolate is requested in normalized coordinates. If the data table contains fewer than N points, N is set to the number of points in the table. If N is set to be less than 8, EES will reset it to 8 and issue a warning message. If N is not provided, a value of -16 is assumed so that the default case is a bi-quadratic method with 16 points.

Values for two of the three specified columns in the data table are specified in the calling parameters. EES will return the interpolated value from the data as a function of these independent variables using interpolation or extrapolation, as required using a multi-quadric radial basis function interpolation method.

(Note: use a semicolon instead of a comma as the list separator when using the European numerical format).

interpolate2DM ('Table Name', X, Y) provides two-dimensional linear interpolation for data in a Lookup table or Lookup file that are arranged in a row and column matrix format. Note that the INTERPOLATE2D function also provides two-dimensonal interpolation, but for this function, the data must be arranged in three columns and a gridding method is applied to the data. The **interpolate2DM** function has the following format.

'Table Name' is a string constant or string variable 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 either the binary (.LKT) or ASCII (.TXT, .CSV) Lookup file Formats.

X is the value of the independent variable whose values appear in the first row of the table. The columns used to determine the interpolated value will be identified by this value.

Y is the value of the independent variable whose values appear in the first column of the table. The rows used to determine the interpolated value will be identified by this value.

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.

LOOP# is useful only when animation is employed. The animation control in the Diagram window allows the calculations to be restarted after they are completed thereby providing a continuous display. Loop# is a counter indicating the number of times the calculations have been started. Under some circumstances, this information can be used in the Equations window. For example, it is possible to solve a dynamic steady-state problem in which the end condition is the starting condition using the animation control. In this case, the initial conditions are found by reading the value at the end of the Parametric Table, but this value does not exist until the table has completed one loop. The Loop# variable can be used in a Function to test for this condition. Loop# has no arguments and it should only be used with animation. Otherwise the function returns 0.

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.

maxLookup('TableName', 'ColumnName', rowStart, rowStop**)** returns the maximum of all or selected cells in a specified column of a specified Lookup table or Lookup file on disk. RowStart and RowStop are optional.

maxParametric('TableName', 'ColumnName', rowStart, rowStop**)** returns the maximum of all or selected cells in a specified column of the Parametric table. RowStart and RowStop are optional.

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.

minLookup('TableName', 'ColumnName', rowStart, rowStop**)** returns the minimum of all or selected cells in a specified column of a specified Lookup table or Lookup file on disk. RowStart and RowStop are optional.

minParametric('TableName', 'ColumnName', rowStart, rowStop**)** returns the minimum of all or selected cells in a specified column of the Parametric table. RowStart and RowStop are optional.

mod(Arg1, Arg2) returns the modulus of Arg1 with respect to Arg2. The modulus is the remainder when Arg1 is divided by Arg2. If Arg2 is 0, an error will result.

nArrayColumns(0**)** takes one numerical argument which should be set to 0. The argument is used to ensure that nArrayColumns is a function name and not a variable name. The function returns the number of column in the Arrays window. If the Arrays table does not exist, the function call will return 0.

nArrayRows(0**)** takes one numerical argument which should be set to 0. The argument is used to ensure that nArrayRows is a function name and not a variable name. The function returns the number of rows in the Arrays window. If the Arrays table does not exist, the function call will return 0.

nFileRows(F$**)** takes one argument which must be either a string constant or a string variable that contains the name of a text file stored on the disk. The function returns the number of rows in that file. If the file does not exist or it cannot be read, the function returns 0.

nLookupColumns('name') returns the number of columns in a Lookup table or Lookup file. The argument to this function can be a string variable.

nLookupRows('name') takes one argument that must be either a string constant or a string variable. The string contains the name of a Lookup table (as in appears on a tab in the Lookup TableWindow) or the name of a Lookup file stored on disk. The function returns the number of rows in the specified table. If the specified table does not exist, the function call will result in an error message.

Example: n=nlookuprows('LOOKUP 1') "number of rows in the Lookup 1"

nParametricColumns('name') returns the number of columns in a Parametric table. The argument to this function is the name of the Parametric table and it can be a string variable or string constant.

nParametricRows('name') takes one argument that must be either a string constant or a string variable. The string contains the name of a Parametric table. The function returns the number of rows in the specified table. If the specified table does not exist, the function call will result 0.

Example: NC=NParametricRows('Table 1') "number of columns in the Parametric table 'Table 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.

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.

pixelsperinch# returns the number of pixels per inch used to display images on the screen. This function takes no argument. It is most useful in placing text and graphic objects on the Diagram window in particular locations, such as needed during animation.

probability(x1, x2, mu, sigma) returns the probability (as a fraction less than 1) that a value x will lie between x1 and x2 in a normal distribution having average mu and standard deviation sigma. The mathematical description is

$$\text{probability}(x_1, x_2, \mu, \sigma) = \int_{z_1}^{z_2} \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-z^2}{2}\right) dz$$

$$\text{where } z_1 = \frac{x_1 - \mu}{\sigma} \qquad \text{and } z_2 = \frac{x_2 - \mu}{\sigma}$$

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).

randg(mean, StdDev, [Seed]**)** returns a randomly-selected value from a Gaussian distribution having the specified mean and standard deviation. The Seed is optional, but if a non-zero integer is supplied, the random number for a specified seed will always have the same value.

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

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.

rgb(red,green,blue) returns the value of color formed by providing the red, green, and blue color component values. The component values can range from 0 to 255. The RGB function is used to specify the color of objects or text items in the Diagram window that have been associated with EES variables. For example, if an object has been created in the Diagram window and given the name Ball, then Ball.fillColor=RGB(0,255,0) will set its color to be green.

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

shortfilename$(**aStringVar$)** accepts one argument which can be a string constant or string variable that is expected to be a Windows style filename including the path and filename name extension. The function returns a string that is set to the filename with the path information removed.

sign(X) will return the sign (+1 or -1) 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.

StdDev(Arg1, Arg2, Arg3, ...) will return the standard deviation of the arguments. The number of arguments must be between 2 and 2000. An array or subset of an array can be provided as an argument list by using array range notation, e.g., X[1..50].

stddevlookup('TableName', 'ColumnName', RowStart, RowStop**)** returns the standard deviation of all or selected cells in a specified column of a specified Lookup table or Lookup disk file. 'TableName' is a string constant or string variable that contains the name of the Lookup table. The name of the Lookup table is shown on its tab at the top of the table. 'ColumnName' is a integer value, string constant or string variable the identifies the column. If an integer value is provided, it is taken to be the column number in the Lookup table. If a string constant is provided, it is assumed to be the name of the variable for the column in the Lookup table. The single quotes that normally identify a string constant are not required. If a string variable is provided, it must contain the name of one of the columns in the Lookup table. RowStart and RowStop are integers or EES variables specifying the starting and stopping rows for which the average will be performed. If these two parameters are not provided, the standard deviation will be determined using all of the data in the selected column.

stddevparametric ('TableName', 'ColumnName', RowStart, RowStop**)** returns the standard deviation of all or selected cells in a specified column of the Parametric table. 'TableName' is a string constant or string variable that contains the name of the Parametric table. The name of the Parametric table is shown on its tab at the top of the table. "ColumnName' is a integer value, string constant or string variable the identifies the column. If an integer value is provided, it is taken to be the column number in the Parametric table. If a string constant is provided, it is assumed to be the name of the variable for the column in the Parametric table. The single quotes that normally identify a string constant are not required. If a string variable is provided, it must contain the name of one of the variables in the Parametric table. RowStart and RowStop are integers or EES variables specifying the starting and stopping rows for which the sum will be performed. If these two parameters are not provided, the standard deviation will be determined using all of the data in the selected column.

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.

sumLookup ('TableName', 'ColumnName', RowStart, RowStop) returns the sum of all or selected cells in a specified column of the Lookup table. RowStart and RowStop are optional. If they are not provided, all rows in the Lookup table will be summed. 'TableName' is a string constant or string variable that contains the name of the Lookup table. The name of the Lookup table is shown on its tab at the top of the table.

'ColumnName' is a integer value, string constant or string variable the identifies the column. If an integer value is provided, it is taken to be the column number in the Lookup table. If a string constant is provided, it is assumed to be the name of the variable for the column in the Lookup table. The single quotes that normally identify a string constant are not required. If a string variable is provided, it must contain the name of one of the columns in the Lookup table.

sumParametric ('TableName', 'ColumnName', RowStart, RowStop) provides the same capability as described above for sumLookup, but for the Parametric table.

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 only 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’)[8]. 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 **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.

TimeStamp(T$) function accepts one parameter, which can be a string constant or string variable providing the name of a file. The function returns an integer value which is a

[8] 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).

coded representation of the date and time that the file was last modified. The larger this number is, the more recent the file. If the specified file is not found, the function returns -1.

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

Uncertaintyof(X**)** returns the assigned or calculated uncertainty of the variable name that is provided as the argument to the function. The returned value will be zero if uncertainty information has not been specified for the specified variable or if the calculations are not initiated with the Uncertainty Propagation or Uncertainty Propagation Table commands.

Uncert.ParDif(X,Y) returns the partial derivative of variable X with respect to variable Y.

Uncert.Percent(X,Y) returns the percent of the total uncertainty in variable X that can be attributed to variable Y.

UnitsOF$ returns a string that is the units of the EES variable that is provided as a parameter to the function. One use of this function is to provide help to users (e.g., in the Diagram Window) to inform them of the units of a variable for which they are expect to provide a value.

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.

CurveFit1D Procedure

CURVEFIT1D is a built-in procedure that provides the parameters and statistics resulting from a linear regression curve fit to data provided in array variables. This procedure provides the same capability as the Curve Fit menu item but it may be more convenient in that the data do not first have to be plotted and the curve fit parameters are returned so that they can be used in following equations. The CURVEFIT1D procedure has the following format:

Call CURVEFIT1D(FitType, X[1..n], Y[1..n]: a[1..m], RMS, Bias, R|2, a_stderr[1..m])

FitType is a string constant or string variable that can be any of the following:

'LINEAR'		{fit Y[i]=a[0]+a[1]*X[i]}
'POLYNOMIAL1'	or 'POLY1'	{same as linear}
'POLYNOMIAL2'	or 'POLY2'	{fit Y[i]=a[0]+a[1]*X[i]+a[2]*X[i]^2}
'POLYNOMIAL3'	or 'POLY3'	{fit Y[i]=a[0]+a[1]*X[i]+a[2]*X[i]^2+a[3]*X[i]^3}
'POLYNOMIAL4'	or 'POLY4'	{fit $Y[i]=\sum_{j=0}^{4} a[j]*X[i]^j$ $Y[i]=\sum_{j=0}^{6} a[j]*X[i]^j$
'POLYNOMIAL5'	or 'POLY5'	{fit $Y[i]=\sum_{j=0}^{5} a[j]*X[i]^j$
'POLYNOMIAL6'	or 'POLY6'	{fit $Y[i]=\sum_{j=0}^{6} a[j]*X[i]^j$
'EXPONENTIAL'	or 'EXP'	{fit Y[i]=a[0]*exp(a[1]*X[i])}
'POWER'		{fit Y[i]=a[0]*X[i]^a[1]}
'LOGARITHMIC'	or 'LOG'	{fit Y[i]=a[0]+a[1]*ln(X[i])}

X[1..n] is an array of values for the independent variable and Y[1..n] is an array of values for the dependent variables. n can be a numerical constant or a previously defined EES variable. The number of data points in the X and Y arrays must be greater than the number of parameters that are being determined and less than 1,000.

a[1..m] is the array of curve fit parameters returned by the CURVEFIT1D procedures. Note that m is equal to 2 for all cases except for the POLYNOMIAL fit types. For those type, m is equal to one greater than the polynomial order. It is not necessary to use array range notation for the curve fit parameters. For example, a[1..2] can be replaced with a0, a1.

<u>All following outputs are optional.</u>

RMS is root mean square error defined as: sqrt(1/n*sum[(Y-Y`)^2] where n is the number of data points, and Y` is the estimated value of Y.

BIAS is the bias error define as: 1/n*sum[(Y-Y`)].

R|2 is the correlation coefficient between Y and X

a_stderr[1..m] is the standard error of the curve fit parameters defined as the square root of the estimated variance of the parameter

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.

ChangeFileExt$ accepts two string variables or constants. The first parameter is a Windows style file name and the second is a replacement file name extension, including the decimal point (.). The function returns the filename with the replacement file name extension.
Example: NewFile$=ChangeFileExt$('myFile.dat','.txt');

Chr$ requires one parameter (x) which must be between 0 and 255. The function returns a string consisting of one character having an ASCII code as given by x. This function is useful for providing characters that cannot be entered on the keyboard or would not be accepted by EES. Some common uses are:

CHR$(13)	{returns carriage return}
CHR$(39)	{returns a single quote}
CHR$(123)	{returns { }

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'}

Note that the concat operator || can be sued to concantenate string more simply than the Concat$ function, since multiple strings can be concatenated, e.g.,
R$='R'||String$(10+12)||' used to be a common refrigerant.'

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'}

Date$ returns the current date. The function has no arguments. The format of the date is controlled by the settings in the Regional Options Control Panel of your Windows Operating System.
Example: T$=concat$('Today is ',DATE$)

EESFILEDIR$** and **EESPROGRAMDIR$ return the directory of the current EES file and the EES application, respectively. The string returned from these functions can be used with the Concat$ function to create a string variable with a completely specified path.

EESFILENAME$ returns the complete file name of the currently open EES file.

FLUIDTYPE$ function takes a single argument, which is the name of a fluid. It returns a string that is one of the following: 'Ideal', 'Real', 'Brine', or 'SL fluid'. This string function can be used to identify the type of fluid and thus the parameters is requires. It is most useful for library routines.

**LOOKUPCOLNAME$**('TableName',ColNo) returns or sets the name of a specified column in a Lookup table. 'TableName' is the name of a Lookup table or a Lookup file. It may be a string constant or string variable. ColumnNo can be a number or expression that evaluates to the column number in the table. If the column is not found, the function will return '??'. The name of an existing column in a Lookup table can only be changed within an internal Function or Procedure. To change the column name, place the LOOKUPCOLNAME$ function call on the left hand side of an equation. Note that the both the column name and the units can be set. The units are assumed to be to the right of the \ character.

**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 or optionally two arguments. The first argument is a numerical constant, variable, or expression. The function returns a string representing the numerical value of the argument. The second argument provides a format specification for the conversion. If a format specification is not provided, 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$(10+12)) {sets R$ to 'R22'}
X$=String$(9.1234,’F1’) {sets X$ to 9.1}

StringLen returns the number of characters in the string constant or string variable provided as the argument. 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$='12345'; V=StringLen(R$) “V is set to value 5”

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}

Time$ returns the current time. The function has no arguments. The format of the string returned by this function is controlled by the settings in the Regional Options Control Panel of your Windows Operating System.
Example: T$=concat$('The time is now ',TIME$)

TimeStamp$(T$) function accepts one parameter, which is a string constant or string variable providing the name of a file. The function returns a string that displays the date and time that the file was last modified. If the file cannot be found, the function returns 'File not found';

Trim$ removes any leading or trailing spaces in the string provided as the argument, returning a string without these spaces.
Example: G$=' abc '; H$=Trim$(G$)

UnitSystem$ provides string information relating to the current settings in the Unit System dialog window. The function takes one character-string argument that must be surrounded by single quote marks. Legal arguments are 'Temperature', 'Pressure', 'Mass', 'Energy, 'Entropy', 'Volume', and 'Trig', The function will return a string that represents the current unit setting. Note that the units of energy, entropy and volume are returned for specific properties. Possible return strings for each of the arguments are as follows:

Temperature:	F, C, R, K
Pressure:	psia, atm, Pa, kPa MPa, bar
Energy:	Btu/lb_m, Btu/lbmol, J/kg, J/kmol, kJ/kg, kJ/kmol
Entropy:	Btu/lb_m-R, Btu/lbmol-R, J/kg-K, J/kmol-K, kJ/kg-K, kJ/kmol-K
Volume:	ft^3/lb_m, ft^3/lbmol, m^3/kg, m^3/kmol
Trig:	radians, degrees
Viscosity:	lb_m/ft-hr kg/m-s
VolExpCoef	1/R, 1/K
YieldStress:	ksi Mpa
YoungsModulus:	psia GPa
Length:	ft, m
SoundSpeed:	ft/s m/s
SurfaceTension	lb_f/ft N/m

Example: T$=UnitSystem$('Temperature') {T$ holds the temperature unit setting}

**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. EES provides built-in thermophysical property data for the fluids listed in the tables below. Separate tables are shown for real gases, ideal gases and brines. In addition property data are provided for ammonia-water, lithium bromide-water and other mixtures. All names in EES are case-insensitive. Additional fluid property data can be added as described in Appendix C.

Names of Built-In Pure or Pseudo-Pure Fluids in EES*

Air_ha	Krypton	R32	R500
Acetone	MD4M	R40	R502
Ammonia	MDM	R41	R507A
Argon	Methane	R113	R508B
Benzene	Methanol	R114	R513
Butene	MM	R115	R514
CarbonMonoxide	m-Xylene	R116	R600
CarbonDioxide	OrthoHydrogen	R123	R600a
CarbonylSulfide	Oxygen	R124	R717
Cis-2-Butene	n-Butane	R125	R718[#]
Cyclohexane	n-Decane	R131B	R744
Cyclopentane	n-Dodecane	R134a	R1216
D4	n-Heptane	R141b	R1225ye(Z)
D5	n-Hexane	R142b	R1233zd(E)
Deuterium	n-Nonane	R143a	R1234yf
DeuteriumOxude	n-Octane	R143m	R1234ze(E)
DimethylCarbonate	n-Pentane	R152a	R1234ze(Z)
DiethylEther	n-Undecane	R161	R1243zf
DimethyEther	Neon	R218	RC318
Ethane	Neopentane	R236ea	RE245cb2
Ethanol	Nitrogen	R236fa	RE245fa2
Ethylbenzene	NitrousOxide	R245fa	SES36
Fluorine	Novec649	R290	SF6
Helium	o-Xylene	R365mfc	Steam[#]
HFE7000	Parahydrogen	R404A	Steam_IAPWS[#]
HFE7200	Propane	R407C	Steam_NBS
HFE7500	Propylene	R410A	SulfurDioxide
HFO1336mzz(Z0	p-Xylene	R423A	SulfurHexafluoride
Hydrogen	R11	R448A	Toluene
HydrogenChloride	R12	R449A	Trans-2-Butene
HydrogenSulfide	R13	R450A	Water[#]
Ice	R13I1	R452A	Xenon
Isobutane	R14	R452B	
Isobutene	R22	R454C	
Isohexane	R23		

[*] Most fluids are represented with a high-accuracy equation of state. Refer to the online help for the specific reference for 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. Using Steam, Water, R718 or Steam_IAPWS will all access property data using Steam_IAPWS.

Names of Built-In Ideal Gas Substances in EES[#]

Air	**C6H14**
AirH2O [+]	**C8H18**
Ar	**CO**
CH3OH	**CO2**
CH4	**H2**
C2H2	**H2O**
C2H4	**He**
C2H5OH	**N2**
C2H6	**NO2**
C3H8	**O2**
CH3OH	**SO2**
C4H10	
C5H12	

[+] AirH2O provides psychrometric relations

[#] Note that, in addition to the list of built-in ideal gas substances shown above, the property functions will recognize and operate with any of the 1262 ideal gases provided in the NASA ideal gas data base compiled by McBride et al.[9] The list of these gases is displayed when you click the NASA button in the Function Information dialog..

[9] Bonnie J. McBride, Michael J. Zehe, and Sanford Gordon, "NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species", NASA/TP-2002-211556, Sept. 2002

Names of Built-In Brine Substances in EES[$]

CACL2	Calcium Chloride-Water
EA	Ethylene Alcohol-Water
EG	Ethylene Glycol-Water
GLYC	Glycerol-Water
K2CO3	Potassium Carbonate-Water
KAC	Potassium Acetate-Water
KFO	Potassium Formate-Water
LICL	Lithium Chloride-Water
MA	Methyl Alcohol-Water
NACL	Sodium Chloride-Water
NH3W	Ammonia-Water
PG	Propylene Glycol-Water

[$] Brine properties are provided given the temperature and mass concentration in %.

The correlations used in EES for brine properties were obtained from Properties of Secondary Working Fluids for Indirect Systems, IIF/IIR, Melinder, 2010, http://www.iifiir.org/en/details.php?id=1177

Note that brine properties in older versions of EES were provided with the BrineProp2 external procedure. The brine property data in EES is based on newer correlations than used in BrineProp2. However, BrineProp2 is provided with EES for backward compatibility

NH3H2O provides the thermodynamic properties of ammonia-water mixtures in subcooled, saturated and superheated conditions based on the mixture equation of state described in:

Ibrahim, O.M., Klein, S.A.,
"Thermodynamic Properties of Ammonia-Water Mixtures,"
ASHRAE Trans.: Symposia, 21, 2, 1495 (1993).

Since NH3H2O is a mixture, its implementation differs somewhat from that for the pure substances. First, it requires 3 independent properties to fix the state. Second, there is a new property function called MassFraction that returns the mass fraction of ammonia. Third, property designators X and Q are used for NH3H2O, which differs from their use for real fluids. These properties are identified by a single case-insensitive letter followed by an equal sign.

Property functions available for fluid NH3H2O

Conductivity	Density	Enthalpy
Entropy	IntEnergy	MassFraction
Pressure	Quality	SurfaceTension
Temperature	Viscosity	Volume

Property designators that are applicable to NH3H2O and their meanings

H=specific enthalpy	P=pressure
Q=quality	S=specific entropy
T=temperature	U=specific internal energy
V=specific volume	X=mass fraction

The input properties and properties returned by the functions will be in the unit system set with the $UnitSystem directive or Unit System menu command.

Example:

```
$UnitSystem SI C kPa
T=10 [C]                      "temperature"
P=1000 [kPa]                  "pressure"
x=0.5                         "mass fraction"
h=enthalpy(NH3H2O,T=T,P=P,x=x)                "specific enthalpy"
Q=quality(NH3H2O,T=T,P=P,x=x)                  "quality"
k=conductivity(NH3H2O,T=T,P=P,x=x)             "thermal conductivity of liquid"
mu=viscosity(NH3H2O,T=T,P=P,x=x)               "viscosity of liquid"
sigma=surfaceTension(NH3H2O,T=T,x=x,Q=0)    "surface tension"

{Solution
h=-194.8 [kJ/kg]
k=0.4791 [W/m-K]
mu=0.001294 [kg/m-s]
Q=-0.001
sigma=0.05055 [N/m]}
```

Newer and more accurate correlations for ammonia-water mixture properties are provided in the NIST REFPROP (http://www.nist.gov/srd/nist23.cfm) program. Note that f-Chart Software provides an interface between EES and REFPROP so that you can continue to use EES while accessing the NIST REFPROP data base. See http://fchart.com/ees/ees-refprop.php for more information.

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 external library routines providing thermodynamic property data for lithium bromide-water mixtures (LiBrH2O.LIB), lithium chloride-water mixtures (LiClH2O.lib), ammonia-water mixtures (NH3H2O), brine property data (BRINEPROP2.LIB), the Peng-Robinson equation of state (Peng_Robineson.DLL) and others. Also, the specific heat, enthalpy and entropy for hundreds of additional substances with NASA function reference compiled by McBride et al. Documentation for these routines is provided through the Function Info command in the Options menu. 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 *Steam, Water, R718, and Steam_NBS* are treated identically. All three keywords use property correlations for water substance published by Harr, Gallagher, and Kell (Hemisphere, 1984). These property correlations are accurate over a large range of conditions. In addition, property keyword *Ice* provides the same properties as *Steam*, *Steam_NB*S, Water and R718 except that it will return the state corresponding to a lower temperature when two solutions are possible, as in the following example.

```
T_ice=temperature(Ice,v=1.088e-3,x=0)              {T_ice=-16.85 [C]}
T_steam=temperature(Steam,v=1.088e-3,x=0)          {T_steam=147.8 [C]}
```

The *Steam_IAPWS* keyword 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.*

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	only applicable for AIRH2O
C	Concentration (%)	only applicable for Brines
D	Dewpoint Temperature	only applicable for AIRH2O
H	Specific Enthalpy	
P	Pressure	
R	Relative Humidity	only applicable for AIRH2O
S	Specific Entropy	
T	Temperature	
U	Specific Internal Energy	
V	Specific Volume	
W	Humidity Ratio	only applicable for AIRH2O

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.

AcentricFactor [dimensionless] returns the acentric factor, a thermodynamic property of a fluid that is defined by

$\omega = -1.0 - \log10(P_{sat}/P_{crit})$

where

P_{sat} is the vapor pressure of the fluid at a temperature equal to 0.7 * T_{crit}
P_{crit} is the critical pressure
T_{crit} is the critical temperature [R or K]

CompressibilityFactor [-] returns the the compressibility factor of a fluid, which his often represented as Z. The compressibility factor is the ratio of the specific volume of the fluid to the specific volume it would have at the same temperature and pressure if the fluid obeyed the ideal gas law.

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 [C])
          k2 = conductivity (AMMONIA, T=100 [C], P=200 [kPa])
          k3 = conductivity (STEAM_NBS, T=100 [C, x=1)
          k4 = conductivity (AIRH2O, T=80 [F], P=14.7 [psia],R=0.5)
```

CP and **CV** [kJ/kg-K, kJ/kgmole-K, Btu/lb-R, Btu/lbmole-R] return the constant pressure and constant volume specific heats of the specified substance, respectively. 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. Two independent properties, e.g., temperature and pressure, must both be provided as arguments for substances modeled as real fluids. Note that, at saturation, pressure and temperature are not independent. If the constant volume specific heat of saturated liquid or vapor is required, it is necessary to provide the quality or some other quality-dependent property. The specific heat of a two-phase mixture with a quality other than 0 or 1 is infinite and a value cannot be returned for this state. For AirH2O, three arguments are required for the CP function. One of these arguments must be total pressure (P). The remaining two can be any of the following: temperature (T), enthalpy (H), relative humidity (R), humidity ratio (W), wetbulb (B), or dewpoint (D). X (for quality) is not allowed. Note also that the specific heat returned for substance AirH2O (psychrometrics) is per unit mass of dry air.

```
Example: Cp1 = CP(AIR, T=350 [K])
         Cv2 = CV (AMMONIA, T=100 [F], P=30 [psia])
         CP3 = CP(AIRH2O,T=25 [C],P=101.3 [kPa],R=0.50)
```

Debye_T [°F, °C, R, K] is approximately equal to the temperature corresponding to a material's highest normal mode of vibration. This property is provided for some incompressible substances. The Debye temperature can be used with the functions DebyeC and DebyeU in order to implement the Debye model of vibrational specific heat and internal energy. Functions DebyeC and DebyeU are documented in the Debye external library. Information about these functions can be obtained from the Function Information dialog when the EES Library Routines button is selected.

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 [K], P=100 [kPa])
d2 = **Density** (Steam, h=850 [Btu/lb_m], P=400 [psia])
d3 = **Density** (AirH2O, T=70 [F], R=0.5, P=14.7 [psia])
d4 = Density (PG,T=50 [C], C=50 [%])

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 [F], P=14.7 [psia], w=0.010)
D2 = dewpoint(AIRH2O, T=70 [F], P=14.7 [psia], R=0.5)
D3 = dewpoint(AIRH2O, T=70 [F], P=14.7 [psia], B=50)

ek_LJ [K, R] returns the Lennard-Jones energy potential (often called epsilon) divided by Boltzmann's constant (k). The units of the returned value will be K in SI units and R in English units.

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 [K])
h2 = enthalpy(STEAM, T=900 [K], P=300 [kPa])
h3 = enthalpy(AIRH2O, T=70 [F], P=14.7 [psia], R=0.50)

Enthalpy_Formation [kJ/kg, kJ/kmol, J/kg, J/kmol, Btu/lbm Btu/lbmol] returns the specific enthalpy of the specified substance at a reference temperature of 25°C (77°F). This enthalpy value is based on a reference state in which elements in their natural state at the reference temperature have zero enthalpy. The units of the value returned by this function depend on the user settings of the unit system. This function is implemented for ideal gases, the NASA gases and organic incompressible fluids. Only one argument is required and that is the name of the substance. The function is not applicable for real fluids or brines. The value of the enthalpy of formation depends on the phase of the substance. The values returned by this function assume that that the phase is what it would be at the reference temperature and standard atmospheric pressure.

Examples: h_ref_CO2=enthalpy_formation(CO2)
h_ref_H2O=enthalpy_formation(H2O) {this result is for liquid water}

ENTHALPY_FUSION [kJ/kg, kJ/kmole, Btu/lbm Btu/lbmole] returns the difference between the specific enthalpy of the liquid and solid phases at the normal melting temperature, i.e., the temperature on the solid-liquid phase boundary corresponding to atmospheric pressure. If the triple-point pressure is above atmospheric pressure, as it is for carbon dioxide, the enthalpy change of fusion corresponds to the triple-point temperature rather than the normal melting point. The normal melting temperature is not provided as an EES function, but the effect of pressure on the melting temperature is small so that the temperature returned by the T_triple property function provides a good estimate of the melting temperature.

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 [K], P=100 [kPa])
s2 = entropy(AIRH2O, T=70 [F], P=14.7 [psia], R=0.50)

FreezingPt [F,C, R, K] returns the freezing point temperature of a brine. The function requires the name of the brine and the concentration. For consistency, it will accept temperature, although temperature is not needed.

Example: FP=freezingPt(EG,C=20 [%])

Fugacity [kPa, bar, psia, atm] returns the fugacity of the specified pure fluid for the given conditions. For an ideal gas fluid, f is identical to P.

HenryConstant_Water [Pa, kPa, bar, MPa, psia, atm] returns Henry's Law constant for a specified fluid in a water solvent. The function accepts one argument in addition to the name of the fluid and that must be temperature in the units set within EES. Henry's Law constant information is provided for most of the pure fluids in the real-fluids data base.

HigherHeatingValue [kJ/kg, kJ/kmol, J/kg, J/kmol, Btu/lbm Btu/lbmol] returns the specific higher heating value of the specified substance at a reference temperature of 25°C (77°F). The units of the value returned by this function depend on the user settings of the unit system. This function is implemented for ideal gases, the NASA gases (that contain only carbon, hydrogen, and oxygen) and organic incompressible fluids. The HigherHeatingValue function requires only one argument and that is the name of the substance. The function is not applicable for real fluids or brines.

Examples: HHV_CH4=HigherHeatingValue(CH4)
HHV_propanol=HigherHeatingValue('n-Propanol')

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 [F], P=14.7 [psia], R=0.50)
w2 = humRat(AIRH2O, T=70 [F], P=14.7 [psia], 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 [K])
u2 = intEnergy(STEAM, T=1320 [K], P=300 [kPa])
u3 = intEnergy(AIRH2O, T=70 [F], P=14.7 [psia], R=0.50)

IntK [W/m, Btu/hr-ft] returns the integrated thermal conductivity which is equal to the integral of conductivity with respect to temperature from some reference temperature. The integrated thermal conductivity is useful for problems in which conductivity is a strong function of temperature. IntK information is provided by selected incompressible substances. Click on IntK in the Function Information dialog to see a list of the substances for which data are available.

IsentropicExponent returns the isentropic exponent of a fluid, defined as $-\frac{v}{P}\left.\frac{\partial p}{\partial v}\right|_s$. For an ideal gas, the isentropic exponent is equal to the ratio of c_p/c_v.

IsIdealGas is a very simple function that requires only one parameter and that is the name of the fluid. It returns 0 for a real fluid, 1 for an ideal gas, -1 for a fluid in the incompressible fluids library, and -2 for a brine. The convention used in EES is that fluid names that are chemical symbols, e.g., O2, N2, and CO2, are modeled 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. The IsIdealGas function will return -1 if the argument is one of the fluids in the incompressible fluids library. Specifically, the fluid must appear in the Fluids.txt file and property data for density, specific heat, viscosity and thermal conductivity must be provided.

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

IsothermalCompress returns the isothermal compressibility of real and ideal gas fluids.

Example: K_T=IsothermalCompress(Steam,T=T,P=P)

KinematicViscosity [m^2/sec or ft^2/hr] returns the kinematic viscosity for the specified fluid defined as the ratio of the dynamic visocity to the density.

Example: nu_1=KinematicViscosity(Air,T=100 [C], P=100 [kPa])

LowerHeatingValue [kJ/kg, kJ/kmol, J/kg, J/kmol, Btu/lbm Btu/lbmol] returns the specific lower heating value of the specified substance at a reference temperature of 25°C (77°F). The units of the value returned by this function depend on the user settings of the unit system. This function is implemented for ideal gases, the NASA gases (that contain only carbon, hydrogen, and oxygen) and organic incompressible fluids. The LowerHeatingValue function requires only one argument and that is the name of the substance. The function is not applicable for real fluids or brines.

Examples: LHV_CH4=LowerHeatingValue(CH4)
LHV_propanol=LowerHeatingValue('n-Propanol')

MolarMass [kg/kmol, lb_m/lbmol] returns the molar mass (often called molecular weight) of the fluid provided as the parameter.

Example: M_CO2 =MolarMass(CarbonDioxide)

Phase$ returns a stringwith the phase (superheated, saturated, liquid, ideal gas) of the specified fluid at the stated condition.

Example: P$=PHASE$(Water, T=100 [C], P=50 [kPa])

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.

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

P_sat [kPa, bar, psia, atm] returns the saturation vapor pressure of a specified pure substance at the specified temperature. This function accepts only one argument, in addition to the fluid name, and that is the temperature. The T= must precede the value, variable, or expression providing the temperature. The temperature must be provided in the units specified in the Unit Information dialog. The fluid may be a fluid name or a string variable.

Example: PS=P_sat(R134a, T=25 [C])

Prandtl returns the dimensionless Prandtl number for the specified fluid defined as $\mathrm{Pr} = \frac{\mu c_p}{k}$ 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 [C])
Pr_2=Prandtl(Steam,T=100 [C],P=50 [kPa])

RealThermoProps is a built-in procedure (not function) that provides a coding shortcut when determining two or more thermodynamic properties at a specified state. This procedure will return temperature, pressure, specific volume, specific enthalpy, specific entropy, specific internal energy and quality (in that order) for a specified fluid and state. The procedure requires as inputs, two arguments with property identifiers, in addition to the substance name. Note that this procedure can only be used for real fluids. It is not applicable for ideal gases, brines or incompressible substances.

Example: Call realthermoprops(R134a, T=25 [C], P=100 [kPa] : T, P ,v ,h ,s ,u ,x)

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 [Btu/lb_m], T=80 [F])

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 [F], P=14.7 [psia], w=0.01)
R2 = relhum(AIRH2O, T=70 [F], P=14.7 [psia], h=25 [Btu/lb_m])
R3 = relhum(AIRH2O, T=70 [F], P=14.7 [psia], B=55 [F])

Sigma_LJ [m, ft] returns the Lennard-Jones length potential (often called sigma).

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 [K])
Cp2 = specheat (AMMONIA, T=100 [F], P=30 [psia])

SoundSpeed [m/s, ft/s] function returns the speed of sound c with units [m/s, ft/s] through the fluid. The speed of sound is defined as:

$$c = \sqrt{\left.\frac{\partial P}{\partial \rho}\right|_s}$$

but for an ideal gas, this simplifies to:

$$c = \sqrt{R\,T\,\frac{c_p}{c_v}}$$

Examples:
C1 = SoundSpeed(Air, T=300 [K])
C2 = SoundSpeed(R134a_ha,T=300 [K],P=100 [kPa])

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 [K])

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 [kJ/kg])
T2 = temperature(AIR, s=1.75 [kJ/kg-K], P=100 [kPa])

ThermalDiffusivity [m^2/sec or ft^2/hr] returns the thermal diffusivity of the substance.

Example: alpha=ThermalDiffusivity(Air,T=100 [C], P=100 [kPa])

T_Crit [°C, K, °F, R] returns the critical temperature of the specified fluid.

Example: Tc=T_Crit(R134a)

T_sat [°C, K, °F, R] returns the saturation temperature of a specified substance at the specified pressure. This function accepts only one argument, in addition to the fluid name, and that is the vapor pressure. The P= must precede the value, variable, or expression providing the pressure. The pressure must be provided in the units specified in the Unit Information dialog. The fluid may be a fluid name or a string variable.

Example: TS=T_sat(R134a, P=100 [kPa])

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 [K], P=100 [kPa])
v2 = **Volume**(Steam, h=850 [kJ/kg], P=400 [kPa])
v3 = **Volume**(AirH2O, T=70 [F], R=0.5, P=14.7 [psia])

V_Crit [m^3/kg, m^3/kgmol, ft^3/lb, ft^3/lbmol] returns the critical specific volume of the specified fluid.

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 [F], P=14.7 [psia], w=0.01)
B2 = wetbulb(AIRH2O, h=25 [Btu/lb_m], P=14.7 [psia], w=0.01)
B3 = wetbulb (AIRH2O, h=25 [Btu/lb_m], P=14.7 [psia], D=30 [F])

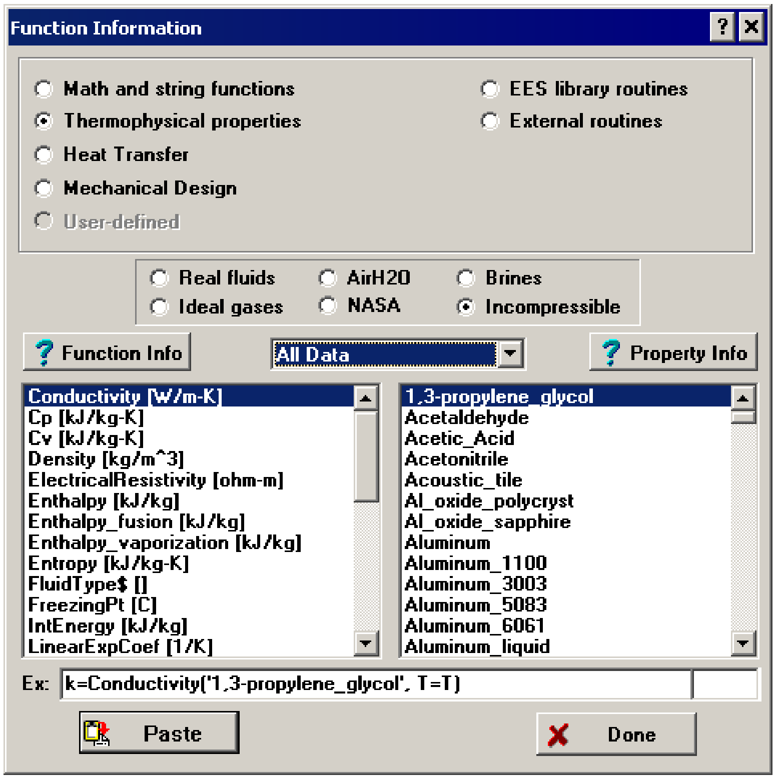
Viscosity [N-sec/m^2, lb_m/ft-hr] returns the dynamic viscosity of the specified substance. For ideal gas substances, the Viscosity function takes one parameter in addition to the fluid name and this parameter must be temperature. For AirH2O (moist air), the temperature, pressure, and humidity ratio (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 [K])
          v2 = viscosity(R134a, T=40 [C], x=1)
          v3 = viscosity(STEAM_NBS, T=100 [C], v=0.335 [m^3/kg])
          v4 = viscosity(AIRH2O, T=80 [F], P=14.7 [psia], R=0.5)
```

Procedure **IdealGasMixtureProps** calculates the molar mass, specific enthalpy and entropy, viscosity, and thermal conductivity for a mixture composed of up to 20 ideal gases in the EES ideal gas data base.

Incompressible Substances Library

Clicking the Incompressible button in the Function Information dialog with the Thermophysical properties button checked changes the display to appear as follows.



○ Math and string functions
● Thermophysical properties
○ Heat Transfer
○ Mechanical Design
○ User-defined
○ EES library routines
○ External routines

○ Real fluids ○ AirH2O ○ Brines
○ Ideal gases ○ NASA ● Incompressible

All Data

Conductivity [W/m-K]
Cp [kJ/kg-K]
Cv [kJ/kg-K]
Density [kg/m^3]
ElectricalResistivity [ohm-m]
Enthalpy [kJ/kg]
Enthalpy_fusion [kJ/kg]
Enthalpy_vaporization [kJ/kg]
Entropy [kJ/kg-K]
FluidType$ []
FreezingPt [C]
IntEnergy [kJ/kg]
LinearExpCoef [1/K]

1,3-propylene_glycol
Acetaldehyde
Acetic_Acid
Acetonitrile
Acoustic_tile
Al_oxide_polycryst
Al_oxide_sapphire
Aluminum
Aluminum_1100
Aluminum_3003
Aluminum_5083
Aluminum_6061
Aluminum_liquid

Ex: k=Conductivity('1,3-propylene_glycol', T=T)

The list on the left shows the property functions that are applicable for these incompressible substances. Functions that have been added for incompressible fluids are ElectricalResistivity, Enthalpy_vaporization, LinearExpCoef, NormalBoilingPt, PoissonsRatio, TotalThermalExp, UltimateStress, YieldStress, and YoungsModulus. Note that only some of these functions are available for all of the substances.

On the right are names of substances to which these functions can be applied. The drop-down list box in the center allows a selection criterion to be applied to the list of substances. The selection criteria are: All data, building materials, heat transfer fluids, insulation, liquid metals, metals, miscellaneous, molten salts and organic liquids.

An example indicating the format of the function use appears in the examples box. Note that Temperature can be a numerical value, and EES variable, or an algebraic expression that evaluates to the desired temperature in the scale selected in the Unit System dialog. Details such as the source of the data and the applicable range of the correlation can be obtained by clicking the Function Info or Info buttons.

The substance name can be supplied as a string constant (within single quotes) or string variable. (The quotes are optional) Incompressible property functions require only one numerical parameter and that is the temperature. For example, the following function returns the thermal conductivity for copper.

k=conductivity(Copper,T=300 [K])

The property data for incompressible fluids are stored in EES Lookup files having a .LKT filename extension. The library and data files are in the UserLib/EES_System/Incompressible folder. You can add additional property data and other functions if you wish as explained in the online help.

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 Tables 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 five Lookup File Formats, and all three can be opened with this 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.

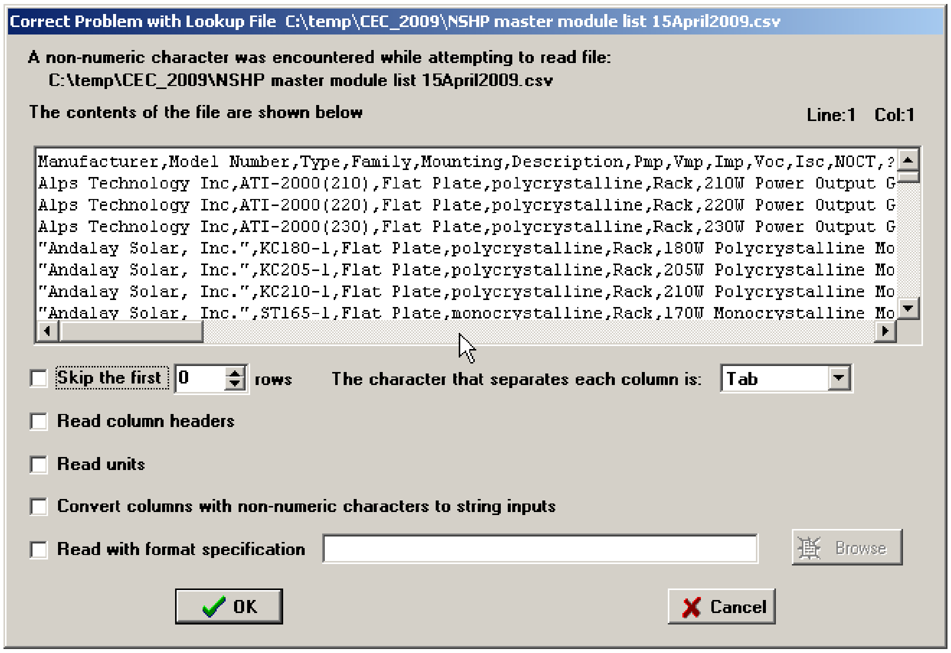
```
5  3
1     11    111
2     22    222
3     33    333
4     44    444
5     55    555
```

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**, **Interpolate2D**, **Lookup**, **LookupCol**, and **LookupRow** functions. These functions are documented below.

ASCII Lookup files (.CSV) and Format Specification Files. (.FMT)
The .CSV file provides only data without any information concerning the column names or data format. Values in the same row are separated with comma, or semicolon character. The number of values in the first non-skipped 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 excluding the number of rows that skipped. 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. Note that when EES tries to open a .CSV file and encounters character data, it will bring up the following dialog which allows options on reading the data.



Correct Problem with Lookup File C:\temp\CEC_2009\NSHP master module list 15April2009.csv

A non-numeric character was encountered while attempting to read file:
C:\temp\CEC_2009\NSHP master module list 15April2009.csv

The contents of the file are shown below Line:1 Col:1

```
Manufacturer,Model Number,Type,Family,Mounting,Description,Pmp,Vmp,Imp,Voc,Isc,NOCT,?
Alps Technology Inc,ATI-2000(210),Flat Plate,polycrystalline,Rack,210W Power Output G
Alps Technology Inc,ATI-2000(220),Flat Plate,polycrystalline,Rack,220W Power Output G
Alps Technology Inc,ATI-2000(230),Flat Plate,polycrystalline,Rack,230W Power Output G
"Andalay Solar, Inc.",KC180-1,Flat Plate,polycrystalline,Rack,180W Polycrystalline Mo
"Andalay Solar, Inc.",KC205-1,Flat Plate,polycrystalline,Rack,205W Polycrystalline Mo
"Andalay Solar, Inc.",KC210-1,Flat Plate,polycrystalline,Rack,210W Polycrystalline Mo
"Andalay Solar, Inc.",ST165-1,Flat Plate,monocrystalline,Rack,170W Monocrystalline Mo
```

☐ Skip the first 0 rows The character that separates each column is: Tab

☐ Read column headers

☐ Read units

☐ Convert columns with non-numeric characters to string inputs

☐ Read with format specification

One of the options is to read the data with a format specification provided in a Format file (.fmt). A Format file can be provided with the $OPENLOOKUP directive or the OPENLOOKUP macro commands to read a LOOKUP file into a LOOKUP table in a specified manner. The .fmt file provides the column display format and optionally, the name and units of the data that are to read into the LOOKUP table. An example .fmt file is:

```
//this is a format specification for .tm2 files
skip 1
2-3   F0     year
4-5   F0     Month
6-7   F0     Day
8-9   F0     Hour
```

10-13 A	G_o	W-hr/m^2
18-21 A	G	W-hr/m^2
24-27 A	DNI	W-hr/m^2
30-33 A	G_D	W-hr/m^2
68-71 A	T_a*10	C
74-77 A	T_dp*10	C
80-82 A	RH	%
85-88 A	P_a	millibar
96-98 A	Wind	m/s

Any row that begins with // is understood to be a comment and is not further considered by EES. The first (non-comment) rows can optionally be any of the following specifications:

Skip n //skip the specified number of rows (n) when reading the file

ReadHeaders //consider the first non-skipped row to be the text in the header

ReadUnits //consider the row after the headers to be the units that appear in the header

Separator Tab // Legal separators are: space tab, comma, or any single character such as ;

FixLFCR //needed only when the text file to be opened was written on a Unix system

AllowStrings //automatically set column in the Lookup table to be string format if the data file contains a non-numerical character in the first row of data

The remaining rows are all of the same form. Each row has 4 fields separated by the Tab character. The first field provides the range of columns that will be read with a hyphen (-) delimiting the start and end columns. The second field provides a format specification which directs EES on how to display the data. The following specifications are valid

Fn display the data as fixed decimal showing n figures after the decimal point
En display the data as floating decimal with n significant figures
A display the data and let EES automatically select the display format
S read the data as string input

The third and fourth fields are optional. If present, they provide the name in the column header and the units.

ASCII Lookup files (.DAT)

Like the .CSV formal, the .DATA file provides only data without any information concerning the column names or data format. Values in the same row can be separated with a Tab character.

Lookup files in EXCEL format (.XLSX)

EES can read and write Lookup files that have a .XLSX file name extension used by EXCEL.

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 with another 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 deleted 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**, **Interpolate2D 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 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. A row range can be optionally provided as the last two paremeters.

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, [Row1, RowN]**)** 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 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 *TableName* 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. A row range can be optionally provided as the last two paremeters.

Examples: Z=interpolate(‘Lookup 1’, ‘Col1’, ‘Col2’, Col1=2.3,2,10)
{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 with data in rows 2 through 10. 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.}

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 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 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.

Chapter 5: Functions, Procedures, Modules, and Subprograms

Most high-level programming languages allow the user to write subroutines. EES also offers this capability in a variety of ways. An EES subroutine is a Function, Procedure, Module or Subprogram written within EES. A Function is a subroutine 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. The equations in a Module are grafted into the main EES routine. A Subprogram differs from a Module in that it forms a stand-alone EES program that can be called from the main EES program or from a subroutine. EES can access both internal subroutines that have been written within EES and external subroutines written in Pascal, C, C++, FORTRAN, or any compiled language. The development of external subrouitnes is described in Chapter 6. Both internal and external subroutines can be stored in the USERLIB\ subdirectory from which they are automatically loaded when EES is started.

EES subroutines offer 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 subroutines are easier to understand. Second, subroutines 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 subroutines. EES also allows subroutines 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. Subroutines 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 of the 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*, *goto* and *Return* 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 user function or procedure or any functions or procedures previously loaded as Library files. Recursive functions which call themselves are, however, not allowed. A functions may not call a Modules, but it can call a Subprogram.

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.

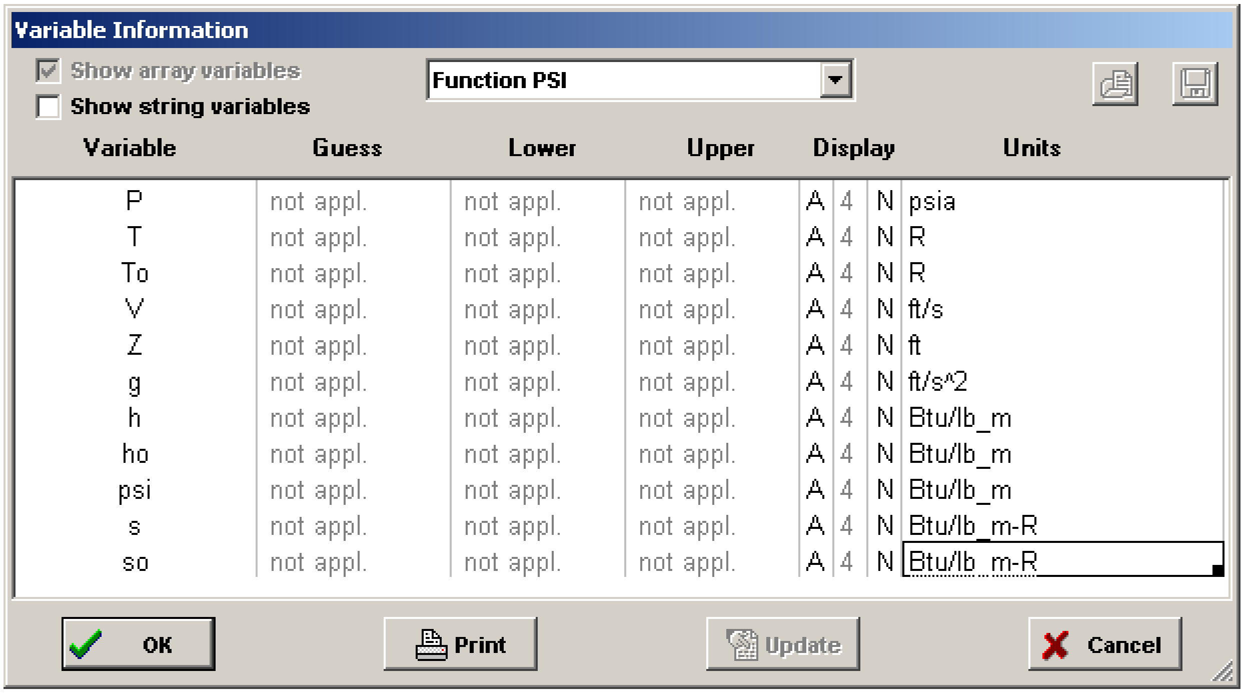
```
FUNCTION psi(T, P, V, Z)
    To := 530 [R]                 “dead state temperature”
    ho := 38.05 [Btu/lbm]         “specific enthalpy at dead state conditions”
    so := 0.0745 [Btu/lbm-R]      “specific entropy at dead state conditions”
    h := enthalpy(STEAM, T=T, P=P)
    s := entropy(STEAM, 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) “[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.

Note that the units for variables in a Function/Procedure/Module/Subprogram can be set in the Equations window by selecting the variable and right-clicking the mouse to select the Variable Info menu item or with the Variable Information menu command (Options menu). The Variable Information dialog for Function psi would appear as shown below. The Guess value, and Lower and Upper bounds are not applicable for functions since variables in Functions and Procedures are set with assignment statements.



☑ Show array variables
□ Show string variables

Function PSI

Variable	Guess	Lower	Upper	Display			Units
P	not appl.	not appl.	not appl.	A	4	N	psia
T	not appl.	not appl.	not appl.	A	4	N	R
To	not appl.	not appl.	not appl.	A	4	N	R
V	not appl.	not appl.	not appl.	A	4	N	ft/s
Z	not appl.	not appl.	not appl.	A	4	N	ft
g	not appl.	not appl.	not appl.	A	4	N	ft/s^2
h	not appl.	not appl.	not appl.	A	4	N	Btu/lb_m
ho	not appl.	not appl.	not appl.	A	4	N	Btu/lb_m
psi	not appl.	not appl.	not appl.	A	4	N	Btu/lb_m
s	not appl.	not appl.	not appl.	A	4	N	Btu/lb_m-R
so	not appl.	not appl.	not appl.	A	4	N	Btu/lb_m-R

The units of variables in Functions and Procedures can also be set in the Solutions window.

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)
...
```

Note that is it not necessary to evaluate all the outputs a call to EES internal or external procedures. For example, if variable Y is not of interest, the call could be written as shown next. List separators (commas for the US system semicolons for the EU system must still be supplied.

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

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. Procedures can be saved as a library file so that they are 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. External 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 external 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.[10]

```
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
```

[10] 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}
```

Return Statement

The Return statement can only be used within Functions and Procedures. When EES encounters a Return statement, it will exit the Function or Procedure and control will resume at the point where the Function or Procedure was called. The Return statement is used in logic constructions with the If Then Else or Repeat-Until statement.

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. Note that Duplicate-End structures that are entered into a function or procedure are automatically converted into Repeat-Until constructs. 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}
```

CASE Statements

CASE statements provide logical operations and they are similar to IF-THEN-ELSE statements in that that provide logical decision branching. Like IF-THEN-ELSE and REPEAT-UNTIL statements, CASE statements can only be used in EES Functions or Procedures. CASE statements somewhat more convenient to use when the logic control must select from multiple options.

The CASE statement must be followed on the same line with the CASE decision variable, which may be an EES numerical (e.g., X) or string variable (e.g., X$). The CASE options are identified with CASE labels followed by a double colon (::). If the CASE decision variable is an EES numerical variable, the CASE labels must be integers. If the CASE decision variable is a string variable, the Case labels must be string constants surrounded by single quotes. An equation is normally placed on the same line after the CASE label. Additional equations may follow and these equations will be executed if the Case variable matches the CASE label. An ELSE label may optionally be used (without quotes) with either the numerical or string versions. Control will transfer to the first equation after the ELSE keyword if the CASE variable does not match any of the CASE labels. An ENDCASE keyword is used to finalize the CASE statement.

CASE statements may not be used in a nested manner. Simple examples of the CASE statement follow to illustrate the syntax.

```
FUNCTION TESTCASE(V)
  CASE V
    1:: X=V+41
```

```
       Y=SQRT(X)
       TESTCASE=X+Y
    3:: TESTCASE=V^2
    5:: GOTO 3
  else:: Call Error('The CASE value must be 1, 3, or 5.  A value of XXXF0 was provided.',V)
  ENDCASE
  Return
  3:  TESTCASE=3
END

G=TESTCASE(1)

PROCEDURE MDAS(OP$,X,Y: R)
  CASE OP$
     'Multiply'::  R=X*Y
     'Divide'::    If (Y<>0) Then
                     R=X/Y
                   Else
                      Call Error('Attempt to divide by zero.')
                   Endif
     'Add'::       R=X+Y
     'Subtract'::  R=X-Y
     ELSE::        R=0
  Endcase
End

Call Mdas('Divide',3,4:R)
```

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.*

Warning Procedure

The WARNING procedure can be used only within an internal function or procedure. The WARNING procedure generates a warning message which is placed in the warning message que and displayed when calculations are completed. Note that the message que is displayed only if the 'Display Warning Messages' checkbox in the Options tab of the Preferences dialog is selected, or if a $Warnings On directive is provided. The WARNING procedure is similar to the ERROR procedure which halts calculations when some user-specified error condition is encountered. The WARNING procedure has the following formats:

CALL WARNING(X)

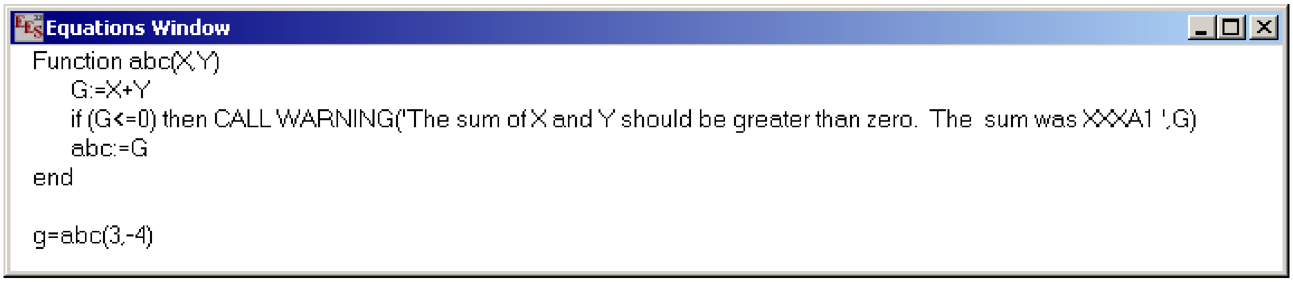
CALL WARNING('My warning message XXXF1',X)

CALL ***WARNING***('My warning message')

The warning string is optional, but if it is provided, it must be placed within single quote. If a warning string is provided, the variable X is optional. X is the value of the parameter which initiates the warning. If a warning string is not provided, EES will generate the following generic warning message. "A warning message was issued due to the value XXX in XXX." Here XXX is the parameter value provided in the WARNING call statement and YYY is the function or procedure name in which the Call WARNING statement appears.

If a warning 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 A3, F1 or E4 follows the XXX, as in the example above, the value of X will be accordingly formatted, otherwise a default format will be applied. If no format is supplied***, automatic format is assumed.*** Note: To insert a string, rather than a numerical value, use $ as the formatting option ***following the XXX, for example:*** CALL WARNING('My error string is XXX$',X$)

The WARNING procedure will most likely be used with an IF - THEN - ELSE statement as in the following example.



```
Function abc(X,Y)
     G:=X+Y
     if (G<=0) then CALL WARNING('The sum of X and Y should be greater than zero.  The  sum was XXXA1 ',G)
     abc:=G
end

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

Print Command in Functions and Procedures

The Print command can be used with the assignment statements in Internal Functions and Procedures to output intermediate results. The format of the Print command is

Print "filename" x, y, z

Filename is the name of the text file into which the values of the designated variables will be written. Filename can be a string constant or a string variable.

x, y, and z are names of variables within the Function or Procedure for which the values will be written to the designated file. The Print command will allow one or more more variable names to be specified. The values of the variables appearing in the Print command must be defined before the Print command is executed.

The Print command will write to file each time the Print command is executed, creating a new row in the file output. The first column of each row will show the name of the function or procedure followed by the number of times the function or procedure has been called since the start of the calculations in parentheses. See the example below.

The file will be cleared at the start of the calculations unless the Print command is followed by the /A option. In this case, the new information will be appended to the existing file; e.g.,
Print/A "filename", x, y, z

Example: The following function will output values of i, x, and y in text file 'text.txt'.

```
function test(x,y)
    t=x+y
    F$='text.txt'
    i=0
    repeat
        i=i+1
        print  F$, i,x,y
    until (i>=t)
    test=t
end

g=test(2,3)
```

Listing of file Text.txt after running the above program.

```
"test(1)"  1 2 3
"test(1)"  2 2 3
"test(1)"  3 2 3
"test(1)"  4 2 3
"test(1)"  5 2 3
```

Modules and SubPrograms

Modules and Subprograms can be considered to be stand-alone EES programs that can be called from the main EES program. The format of a Module / Subprogram is similar to that for an internal Procedure. The Module / Subprogram is supplied with inputs and it calculates outputs. Like a Procedure, the Module statement can (optionally) use a colon in the argument list to separate inputs from outputs. The number of arguments provided to the left of the colon is the number of degrees of freedom in the Module / Subprogram, or stated another way, the number of values that must be supplied to have the number of equations equal to the number of unknown. Examples of Module / Subprogram statements with a colon in the argument list are:

MODULE Testme(A, B : X, Y)   or   SUBPROGRAM 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, the equations in EES Modules / Subprograms are 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 unnecesary and it can be replaced with a comma (or semi-colon for the European numerical format). In fact, this is the preferred way to use Modules and Subprograms. The following Module / Subprogram statements are equivalent to the format shown above.

MODULE Testme(A, B, X, Y)   or   SUBPROGRAM Testme(A, B, X, Y)

A Module / Subprogram is accessed with a CALL statement. For example, the following statement would access the Testme Module or Subprogram.

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

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

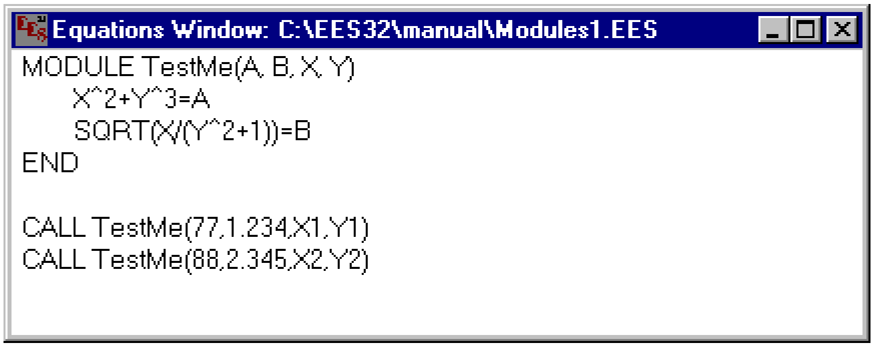
At this point, you must be curious as to the difference between a Module and a Subprogram. The difference is as follows: When EES encounters a CALL statement to a SUBPROGRAM, it opens a new workspace and it then solves the equations in the SUBPROGRAM, using blocking and iteration as necessary. The calculated variables appearing as parameters in the SUBPROGRAM statement are returned to the main program.

When EES encounters a CALL statement to a MODULE, it transparently grafts the equations in the module into the equations in the main program. The steps necessary for this process are as follows. First, every variable in the module, including each input and output 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. Note that the guess values and limits for the EES variables appearing in the CALL statement in the main program override the specified guess values and limits for the variables in the argument list of the MODULE statement. 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 is merged into the main EES program each time a CALL statement is encountered. EES currently allows up to 6000 equations (10,000 in 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

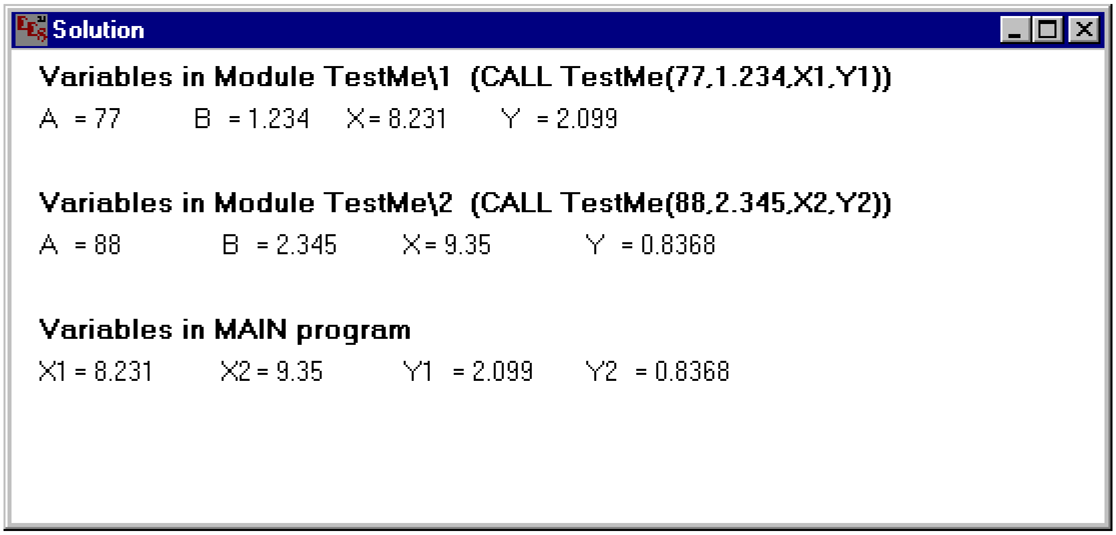
indicates that the equation h2=h1+Q/m originated from the second call to the Turbine module.

The Module / Subprogram is terminated with an END statement. A CALL statement is used to call the Module / Subprogram, just as for a Procedure. The important difference between a Procedure and a Module / Subprogram is that the Module / Subprogram is composed of equality statements whereas the Procedure is composed of assignment statements. Consequently, a Module / Subprogram cannot support logic constructs such as IF THEN ELSE, but it can provide iterative solutions to implicit equations, when needed, and order independent equation input, just as in the main part of EES. The following example uses a Module to solve an implicit set of two equations and two unknowns.



```
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)
```

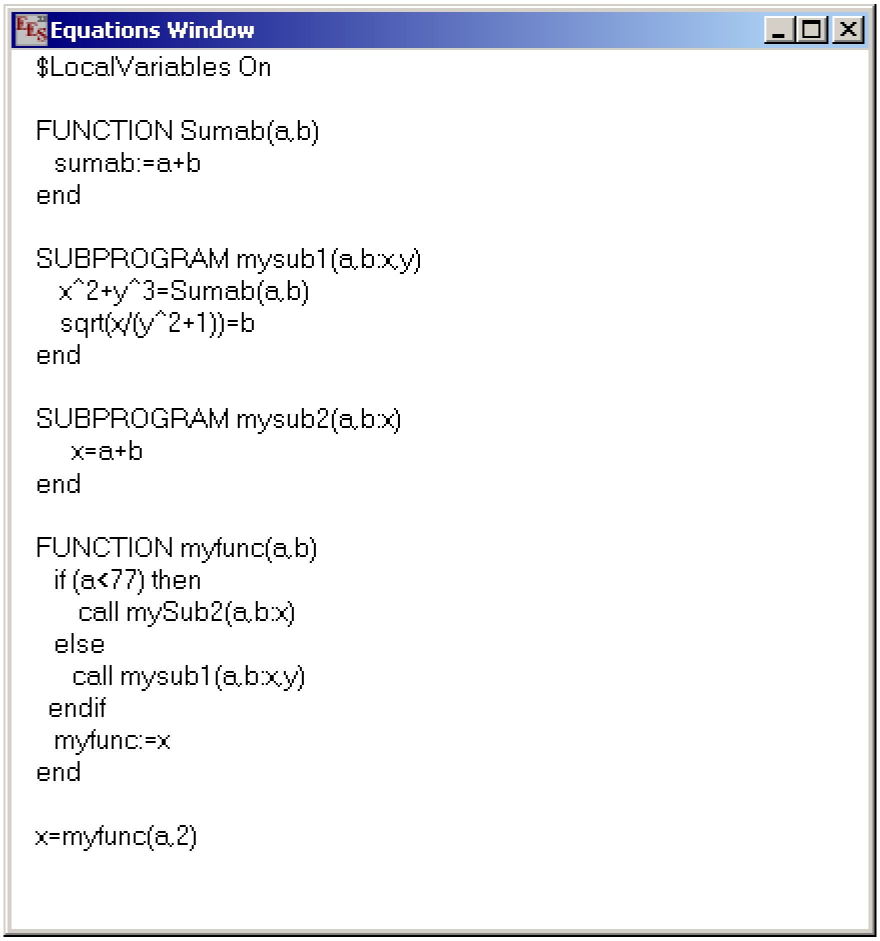
The local values of variables in Modules / Subprograms are normally not displayed in the Solution window. However, you can view these local solutions for Modules / Subprograms that appear in the Equations window by selecting the 'Show function/procedure/module variables' control in the Options tab of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



```
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
```

So should you use a Module or a Subprogram? Limited tests thus far indicate that a Module tends to solve a problem more efficiently than a Subprogram, although you may wish to test both forms to determine which is better for your application. A major advantage of a Subprogram is that it can be called from a Function or Procedure. Conditional statements, such as the IF THEN ELSE, GOTO, and REPEAT - UNTIL are only supported in Functions and Procedures. If your application requires integration with conditional statements, you will necessarily have to use a Subprogram. Shown below is a short program which calls a Function that uses an IF THEN ELSE statement to call one of two Subprograms, one of which calls another function. Subprograms can call and be called by Functions and Procedures and that is their most significant difference from Modules.

If automatic unit checking is enabled, EES will check the unit consistency of all equations in Subprograms and Modules. It will further check to see that the units variables provided in the Call statement match those in the Module or Subprogram statement.



```
$LocalVariables On

FUNCTION Sumab(a,b)
  sumab:=a+b
end

SUBPROGRAM mysub1(a,b:x,y)
   x^2+y^3=Sumab(a,b)
   sqrt(x/(y^2+1))=b
end

SUBPROGRAM mysub2(a,b:x)
    x=a+b
end

FUNCTION myfunc(a,b)
  if (a<77) then
     call mySub2(a,b:x)
  else
    call mysub1(a,b:x,y)
  endif
  myfunc:=x
end

x=myfunc(a,2)
```

Variables values are normally passed to a Module / Subprogram through the argument list. The $COMMON directive may be used to pass variables that are defined in the main program to a Module / Subprogram. Variable information, including guess values and lower/upper limits for Module / Subprogram variables is accessible with the Variable Info command. The local variables in a Module / Subprogram are always real regardless of the Complex Numbers setting.

Modules / Subprograms may be stored as library files, just like Internal Functions and Procedures. 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 / Subprogram (using the same syntax as used in Procedures) or it can be supplied with a separate help file having the same filename as the library file with a .HLP or .HTM filename extension.

Modules and Subprograms 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 that reside in the USERLIB\ subdirectory or in subdirectories therein. Files within shortcut folders that are eplaced in the USERLIB directory will also be automatically loaded. 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.

The order in which library files are loaded can be an issue if one library file calls another. Note that library files in the USERLIB\EES_System folder are loaded before any other library files. Library files that are called by other library files and thus need to be loaded first can be placed in this folder.

Chapter 6: External 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 external routines may have any number of arguments. Functions return a single value whereas procedures may return multiple values. The external 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. External functions and procedures can provide the units of their inputs and output so that EES can check units when they are called. They can also issue error messages (which stop the calculations) or warning messages that are displayed after the calculations are completed.

External functions and procedures are written as dynamic link library (DLL) routines under the Windows operating system. External functions are identified with a .DLF. There are two formats for external procedures identified by .DLP and .FDL filename extensions. Files having a .DLL filename extension can hold one or more functions and procedures. When EES is started, it examines the files in the \USERLIB\ subdirectory. Any files having a .DLF, .DLP, .FDL, or .DLL filename extension are assumed to be external functions or procedures and they are automatically loaded unless otherwise directed by the Library Manager described later in this chapter. 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).

External 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 in this chapter provide detailed information and examples of external functions and procedures.

EES External Functions (.DLF files)

External functions can be written in C, C++, Pascal, or any language that 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 external function is implemented as a linked list. The linked list record or structure consists of a double precision value and a pointer to the next input. The last input points to nil. FORTRAN 77 does not support pointers so .DLF external functions cannot be written in this language. The .FDL format described in this chapter should be used in this case.

The external function should provide an example of the calling statement and check that the number of inputs supplied in the linked list is equal to the number the function expects. The PWF and SUM_C function examples in the following subsections show how this checking can be done in Pascal and C, respectively. 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 external function written in Borland's Delphi 5 is as follows:

```
library XTRNFUNC;
type
     charstring=array[0..255] of char;
     ParamRecPtr = ^ParamRec;
     ParamRec = record  { defines structure of the linked list of inputs }
           Value: double;
            next: ParamRecPtr;
     end;

function FuncName (var S:charstring; var Mode:integer; Inputs:ParamRecPtr):double; export;
stdCall;
    begin
           ...
           FuncName:=Value; { Funcname must be double 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 256-character C string terminated with an ASCII 0. 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.

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 (double 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 number of external functions are included with standard version of EES. For example, the following four functions implement a generalized equation of state using Redlich-Kwong-Soave equation[11].

Compress(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).

Another useful function that is provided as an external function for EES is the PWF or Present Worth Factor function described in the following subsection.

The PWF Function – a .DLF external routine in written in DELPHI

EES does not have any internal economic functions. An economic function called the present worth factor (PWF)[12] has been added as a external 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

$$PWF(N,i,d) = \sum_{j=1}^{N} \frac{\left(1+i\right)^{j-1}}{\left(1+d\right)^{j}} = \begin{cases} \dfrac{1}{d-i}\ 1-\left(\dfrac{1+i}{1+d}\right)^{N} & \mathit{if}\ i \neq d \\ \dfrac{N}{1+i} & \mathit{if}\ i = d \end{cases}$$

where

N is the number of periods (e.g., years)

[11] G. Soave, Chem. Eng. Science, Vol. 27, pp. 1197-1203, 1972

[12] Duffie, J.A. and Beckman, W.A., *Solar Engineering of Thermal Processes*, 2nd edition, J. Wiley and Sons, 1992, Chapter 11.

i is the interest rate per period, expressed as a fraction
d is the market discount rate per period, expressed as a fraction

A external 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 external function just like any of its internal functions. Shown on the following pages is the complete listing for the PWF external function written in Borland's Delphi 3.0 (32-bit version).

```
library PWFP;

const doExample = -1;

type
  CharString = array[0..255] of char;
  ParamRecPtr=^ParamRec;
  ParamRec=record
    Value:double;
    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:CharString;  var Mode:integer;
          Inputs:ParamRecPtr):double; export; stdcall;
  var
    P: ParamRecPtr;
    V: double;

 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: double;
 var
   NArgs: integer;
   interest, discount, periods: double;
 begin
   PWFCalc:=0; {in case of error exit}
   S := '';
   P := Inputs;
   Periods := P^.value;
   if (Periods < 0) 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 external functions from external Procedures. It does this by the filename extension. External functions must have a .DLF filename extension. Rename the external 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)
```

SUM_C – a .DLF external function written in Microsoft's Visual C++

The SUM_C function listed below returns the sum of its two arguments. There is of course no need for a function of this type in EES. However, this simple function clearly demonstrates how to write a .DLF external function for EES in C++.

```
#include <windows.h>
// Structure for handling EES calling syntax
struct EesParamRec {
  double value;
  struct EesParamRec *next;
};

// Tell C++ to use the "C" style calling conventions rather than the C++ mangled names
  extern "C"  {

__declspec (dllexport) double SUM_C(char s[256], int& mode, struct EesParamRec *input_rec)
{
      double In1, In2, sum_res;
      int NInputs;

      if (mode==-1) {
            strcpy(s,"a = SUM_C(b,c)"); // return example call string
            return 0;
      }
      if (mode==-2) {
            strcpy(s," "); // return a string with expected units of the inputs, separated by commas or spaces
            return 0;
      }
      if (mode==-3 {
            strcpy(s," "); // return a string with expected units of the outputs, separated by commas or spaces
            return 0;
      }

// Check the number of inputs
    NInputs=0;
    EesParamRec * aninput_rec=input_rec;
      while (aninput_rec!= 0)
      {
            aninput_rec= aninput_rec->next;
            NInputs++;
      };
      if (NInputs!=2) {
            strcpy(s,"SUM_C expects two inputs");
            return 0;
      }

      In1= input_rec->value;
      input_rec=input_rec->next;
      In2=input_rec->value;
      sum_res = In1+In2;

      return sum_res;
}
};
```

EES External Procedures (.FDL and .DLP files)

EES external procedures are very similar to EES external functions. In either case, the user supplies the function or procedure in external 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 dynamic link libraries (DLL's) in the Windows operating system. 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.

External 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.

The CALL statement used to access external functions is identical in format to the CALL statement used for internal EES Procedures. Note that is it not necessary to evaluate all of the outputs. If only variable Y is of interest, the call to the external procedure could be entered as:

CALL procname(‘text’, A, B : , Y,)

The following three sections describe the .FDL and .DLP external procedure formats and provide examples in FORTRAN, DELPHI, and C++ which should serve as a model.

External 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. A skeleton of the code is shown below in the Digital Visual FORTRAN environment.

```
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.

Note that the two !DEC$ATTRIBUTES directives are compiler specific. The default filename MYPROC.DLL should be changed to MYPROC.FDL after building the dynamic link library project.

The simple FORTRAN program listed is compiled with the GCC Fortran compiler. It provides the product, dividend, sum, and difference of two input values. This program should provide a model for writing external EES procedures in FORTRAN.

```
SUBROUTINE MDASF(S,LS,MODE,NINPUTS,INPUTS,NOUTPUTS,OUTPUTS)
USE iso_c_binding, only: c_null_char !necessary to append a null character
IMPLICIT NONE
!GCC$ ATTRIBUTES STDCALL :: MDASF       ! STDCALL calling convention for FDL
INTEGER MODE, LS
integer, intent(in) ::  NINPUTS, NOUTPUTS
double precision, intent(in) :: INPUTS(ninputs)
double precision, intent(out) :: OUTPUTS(noutputs)
double precision :: x, y
CHARACTER(len=255) :: S
IF (MODE.EQ.-1) THEN
   S='CALL MDAS(X,Y:A,B,C,D)'//C_null_char
   RETURN
ENDIF
IF (MODE.EQ.-2) THEN
   S='m,m'//C_null_char
   RETURN
ENDIF
IF (MODE.EQ.-3) THEN
   S='m^2,,m,m'//C_null_char
   RETURN
ENDIF
IF (MODE.LT.0) THEN
   S=''//C_null_char
   RETURN
ENDIF
IF (NINPUTS.NE.2) THEN
   S='MDAS requires two input.'//C_null_char
   RETURN
ENDIF
IF (NOUTPUTS.NE.4) THEN
   S='MDAS EXPECTS TO PROVIDE 4 OUTPUTS'//C_null_char
   RETURN
ENDIF
X=INPUTS(1)
Y=INPUTS(2)
```

```
IF (ABS(Y).LE.1E-9) THEN
  S='DIVISION BY ZERO IN MDAS'//C_null_char
  RETURN
ENDIF
OUTPUTS(1)=X*Y
OUTPUTS(2)=X/Y
OUTPUTS(3)=X+Y
OUTPUTS(4)=X-Y
!S=''//C_null_char
RETURN
END
```

Note that it is necessary to include a .def file so that the compiler can process the parameters. The .def file for this procedure is as follows:

```
EXPORTS
     mdasf_@32 @ 1
     MDASF = mdasf_@32
```

External 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.

External procedures using the .DLP format are very similar to external 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 external 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 external functions. Outputs is a linked list of double 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 external 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.

```
library MDASP;

const Example = -1;
```

```
type
  CharString = array[0..255] of char;
  ParamRecPtr=^ParamRec;
  ParamRec=record
    Value:double;
    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:CharString; var Mode:integer;
      Inputs,Outputs:ParamRecPtr); export; stdcall;

 procedure MyDearAuntSally;
  var
   P1, P2: double;
   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 if (Mode = -2) then
     S := ' ' {return units of inputs}
  else if (Mode = 3) then
     S := ' ' {return units of outputs}
  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.
```

External Procedures with the .DLP Format - a C++ Example

The .DLP format for external procedures can be written in C++. Listed here is the MDAS_C procedure which provides exactly the same capabilities as those provided above in FORTRAN and Pascal.

```
#include <windows.h>
// Structure for handling EES calling syntax
struct EesParamRec {
 double value;
 struct EesParamRec *next;
};

// Use the "C" style calling conventions rather than the C++ mangled names
extern "C"
 __declspec (dllexport) void MDAS_C(char s[256], int& mode, struct EesParamRec *input_rec, struct
EesParamRec *output_rec)
{
  double v, v1, v2;
  int NInputs, NOutputs;
  EesParamRec *outputs, *inputs;

  if (mode==-1) {
    strcpy(s,"CALL MDAS_C(X,Y : M, D, A, S)");
    return;
  }
  if (mode==-2) {
    strcpy(s," ");   //return string with units of inputs

    return;
  }
  if (mode==-3) {
    strcpy(s," ");  //return string with units of outputs
    return;
  }

  NInputs=0;
  inputs=input_rec;
  while (inputs!=NULL) {
      NInputs++;
      inputs=intputs->next;
  }
  if (NInputs!=2)) {
    strcpy(s,"MDAS_C expects two inputs");
    mode=1;
    return ;
  };
  outputs=output_rec;
  NOutputs=0;
  while (outputs!=NULL) {
     NOutputs++;
       outputs=outputs->next;
  }
  if (NOutputs!=4) {
   strcpy(s,"MDAS_C requires 4 outputs");
   mode=2;
   return ;
  }
  strcpy(s,"");
  inputs=input_rec;
  v1=inputs->value;
  inputs=inputs->next;
  v2=inputs->value;

  v=v1*v2;
  outputs=output_rec;
  outputs->value=v;
  outputs=outputs->next;
  if (v2==0) {
   strcpy(s,"attempt to divide by zero in MDAS_C");
     mode=3;
     return;
  }
  v=v1/v2;
  outputs->value=v;
  outputs=outputs->next;
  outputs->value=v1+v2;
  outputs=outputs->next;
  outputs->value=v1-v2;
  mode=0;
};
```

EES is also able to call Python scripts as EES Procedures. An example of the call statement is:

CALL PYTHON('filename.pyw', 'optional string', In1, In2, ... : Out1, Out2,...)

The Python script can be designed to return the units of the inputs and outputs so that EES can use its unit checking algorithms with the Python script. An example program is available in the Examples folder. See the online help for details.

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 provided that the Library Manager does not direct otherwise. (The Library Manager is described below.)

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
  CharString = array[0..255] of char;
  ParamRecPtr = ^ParamRec;
  ParamRec = record
             Value : Double;
             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 S: CharString; var Mode: integer;
       Inputs: ParamRecPtr): double; export; stdCall;
 begin
    {Code for myFunc1}
    ...
    ...
end; {myFunc1}

{***********************************************************************}
function myFunc2 (var S: CharString; var Mode: integer;
       Inputs: ParamRecPtr): double; export; stdCall;
 begin
    {Code for myFunc2}
    ...
    ...
end; {myFunc2}

procedure myDLP(var S:CharString; var Mode:integer;
        Inputs,Outputs:ParamRecPtr); export; stdCall;
 begin
    {Code for myDLP}
    ...
    ...
end; {myDLP}

{***********************************************************************}
 exports
   DLFNames,
   DLPNames,
   FDLNames,
   myFunc1,
   myFunc2,
   myDLP;

begin
end.
```

```
#include <windows.h>
#include <math.h>

// Defines the entry point for the dll
BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
              )
{
    return TRUE;
}

// Tell EES which functions and procedures are exported in the library :
// List of DLF format functions
__declspec(dllexport) void DLFNames(char* Names)
{
   strcpy(Names,"myFunc1,myFunc2");
}

// List of DLP format procedures
__declspec(dllexport) void DLPNames(char* Names)
{
   strcpy(Names,"myDLP");
}

// List of FDL format procedures
__declspec(dllexport) void FDLNames(char* Names)
{
   strcpy(Names,"myFDL");
}


// DLF functions implementation
__declspec(dllexport) double myFunc1(char s[256], int& mode, EES_PARAM_REC *rec_in)
{Code for myFunc1}
    ...
    ...
   return;
}

__declspec(dllexport) double myFunc2(char s[256], int& mode, EES_PARAM_REC *rec_in)
{Code for myFunc2}
    ...
    ...
   return;
}
// DLP procedure implementation
__declspec(dllexport) void myDLP(char s[256], int& mode, EES_PARAM_REC *rec_in,
EES_PARAM_REC *rec_out)
{Code for myDLP}
    ...
    ...
       return;
   }

// FDL procedures implementation (FORTRAN style)
// Note : fortran is always passing the length of the string after
// a string parameter. FORTRAN is always expecting parameters to be
// passed by reference
//
__declspec(dllexport) void myFDL(char s[256], int& clen, int& mode,
   int& NInputs, double inputs[25], int& NOutputs, double outputs[25])
{
{Code for myDLP}
    ...
    ...
      return;
   }
```

Help for External 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 external routine. This file can be an ASCII text file (with a .TXT extention, a portable document file (.PDF), 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 .PDF format and the Windows .HLP and HTML (.HTM) files allow figures and formatting options to be used so it is a better way of providing help. These help file formats can be composed using any of the commercial Help generating programs.

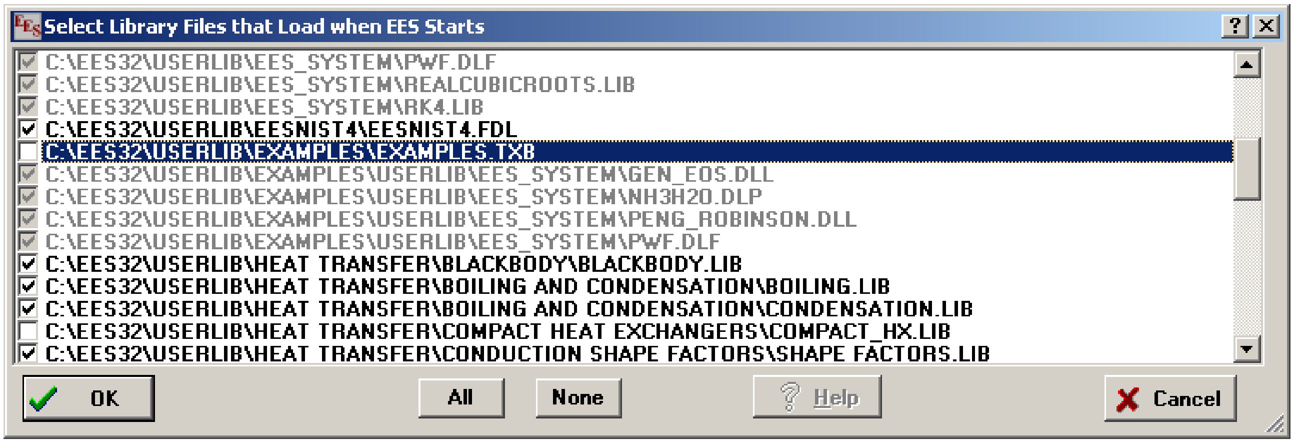
A single library file can hold many functions. For example, a (.DLL) library file written in EES can have any number of functions and procedures. It is possible to design a Windows .HLP file such that EES will jump the appropriate topic when the user clicks the Func Info button. This is done by providing a file containing help context information. The help context file has the same name as the library file but with a .CTX filename extension. EES will look for the .CTX file when help is requested. If a .CTX file is found, EES will open the file to find the context number for the topic that has the name of the function for which help is requested. The Help file will be then opened directly to that topic. If the topic is not found or if the .CTX file is not provided, the Help file will open to the first topic. Each line in the .CTX file contains two items: the context number (as it is used in the Help program) and a string which contains the function name. Shown below is the context file for the external files in the LiBr.DLL.

```
1 LiBr_External_Routines
2 H_LIBR
3 T_LIBR
4 P_LIBR
5 V_LIBR
6 X_LIBR
7 Q_LIBR
8 VISC_LIBR
9 COND_VISC
```

Library Manager (Professional Version)

Normally, all files placed in the USERLIB folder (within the folder that the EES.exe program resides) are automatically loaded when EES is started. These files can be library files with filename extensions .LIB, .FDL, .DLF, .DLP, or .DLL. Textbook menu files having a .TXB filename extension are also automatically loaded, but only one .TXB file can be active at a time. The user can control which files are loaded at startup by moving files into and out of the USERLIB folder. However, a more convenient alternative is to use the Library Manager provided in the Professional version of EES.

The Library Manager is accessed by clicking the Auto Load button in the Function Information dialog. This button is visible when any of the radio buttons at the top right of the Function Information dialog are selected. Clicking Auto-Load will bring up a dialog having the appearance shown in the figure below. The Auto Load button is visible if either the EES Library Routines or the External Routines radio buttons is selected. Each file in the USERLIB folder appears in the list. The check box to the left of the list controls whether or not the file is automatically loaded when EES is started. If the checkbox is gray, the function is located in the EES_System directory and it cannot be unloaded in the Library Manager. Changing the loading status of any file has an immediate effect. If a file is not currently loaded, clicking the check box will automatically load the file and also ensure that it is loaded the next time EES is started. Unchecking a box will remove a loaded file from memory and prevent it from loading the next time EES is started. Information is written by the Library Manager into file LibraryManager.txt in the USERLIB folder. Deleting this file will cause all files to be automatically loaded.



☑ C:\EES32\USERLIB\EES_SYSTEM\PWF.DLF
☑ C:\EES32\USERLIB\EES_SYSTEM\REALCUBICROOTS.LIB
☑ C:\EES32\USERLIB\EES_SYSTEM\RK4.LIB
☑ C:\EES32\USERLIB\EESNIST4\EESNIST4.FDL
☐ C:\EES32\USERLIB\EXAMPLES\EXAMPLES.TXB
☑ C:\EES32\USERLIB\EXAMPLES\USERLIB\EES_SYSTEM\GEN_EOS.DLL
☑ C:\EES32\USERLIB\EXAMPLES\USERLIB\EES_SYSTEM\NH3H2O.DLP
☑ C:\EES32\USERLIB\EXAMPLES\USERLIB\EES_SYSTEM\PENG_ROBINSON.DLL
☑ C:\EES32\USERLIB\EXAMPLES\USERLIB\EES_SYSTEM\PWF.DLF
☑ C:\EES32\USERLIB\HEAT TRANSFER\BLACKBODY\BLACKBODY.LIB
☑ C:\EES32\USERLIB\HEAT TRANSFER\BOILING AND CONDENSATION\BOILING.LIB
☑ C:\EES32\USERLIB\HEAT TRANSFER\BOILING AND CONDENSATION\CONDENSATION.LIB
☐ C:\EES32\USERLIB\HEAT TRANSFER\COMPACT HEAT EXCHANGERS\COMPACT_HX.LIB
☑ C:\EES32\USERLIB\HEAT TRANSFER\CONDUCTION SHAPE FACTORS\SHAPE FACTORS.LIB

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.,

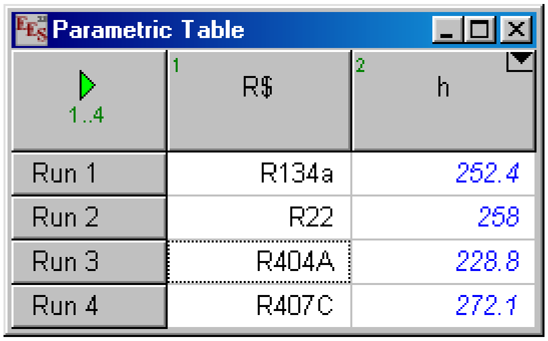
```
B$=A$
```

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	1 R$	2 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

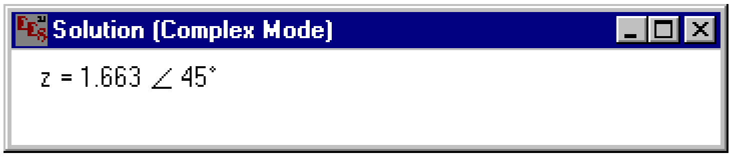
The use of deg or rad to indicate the units of the angle is strongly recommended for three reasons. First, the value of the constant you enter will not be changed by the unit system setting. Second, you can enter complex variables in polar form with the angle in degrees yet do all calculations with the 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.

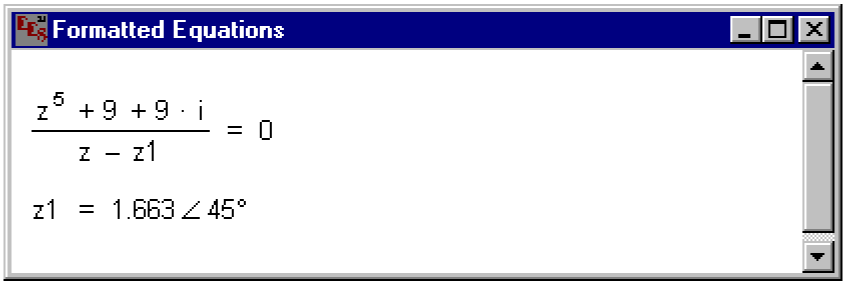
There are a few built-in functions that operate only in complex number mode. They are **Real**, **Imag**, **Cis**, **Magnitude**, **Angle**, **AngleDeg**, **AngleRad**, and **Conj**. These functions all take one (complex number) argument and set the real part of the complex variable to the selected value.

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°

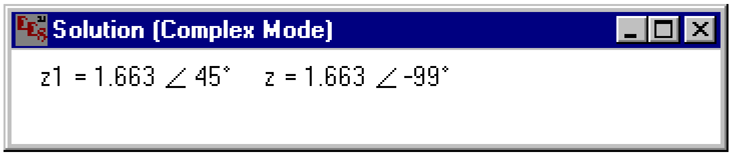
This is a correct solution to the equation, but there are four others. To find a different solution (without changing guess values), divide the equation by the difference between z and the value of this root. The Formatted Equations window will appear as shown.



$$\frac{z^5 + 9 + 9 \cdot i}{z - z1} = 0$$

$z1 = 1.663 \angle 45°$

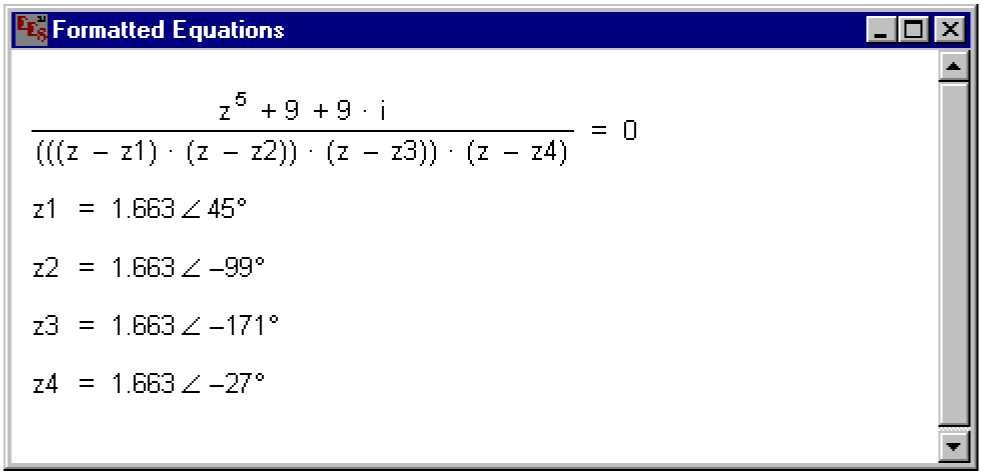
Now, a second solution will be found.



$z1 = 1.663 \angle 45°$ $z = 1.663 \angle -99°$

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.



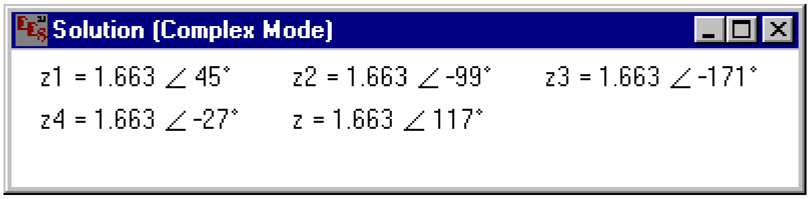
$$\frac{z^5 + 9 + 9 \cdot i}{(((z - z1) \cdot (z - z2)) \cdot (z - z3)) \cdot (z - z4)} = 0$$

$z1 = 1.663 \angle 45°$

$z2 = 1.663 \angle -99°$

$z3 = 1.663 \angle -171°$

$z4 = 1.663 \angle -27°$



$z1 = 1.663 \angle 45°$ $z2 = 1.663 \angle -99°$ $z3 = 1.663 \angle -171°$

$z4 = 1.663 \angle -27°$ $z = 1.663 \angle 117°$

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.
5. Array values can be entered with a short-hand array range notation (see below), e.g.,
   X[1..4]=[11, 22, A*B, Z^2]

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.

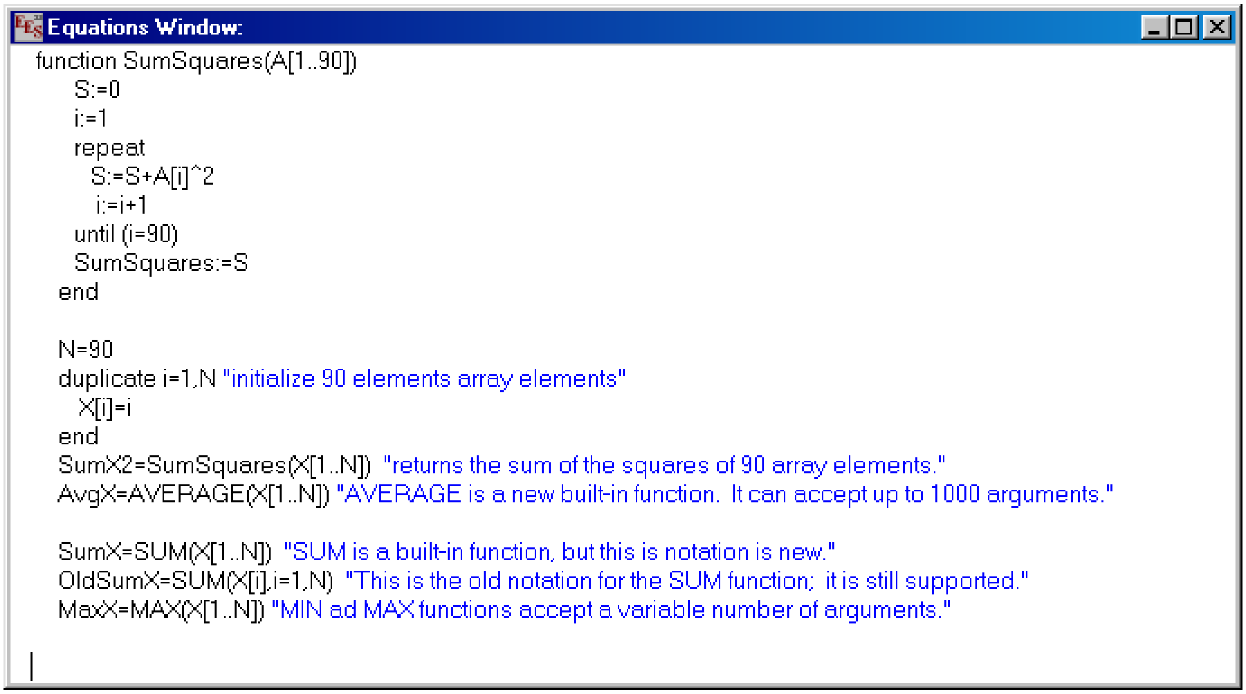
Array Range Notation

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 2000 arguments may be passed to a function or procedure using array range notation.

Array range notation can be used with variable limits, provided that the variable is previously specified. For example, X[1..n] can be used if n is set to a value in a statement before the X[1..n] is encountered. When Array range notation is used in the header statement of an internal Function or Procedure, a problem occurs since the value of variable, cannot possibly be known. In this case, EES provides a default maximum dimension which currently is 100. If you wish to have a larger maximum, you can specifically provide a number for the maximum. This capability is only applicable for one-dimensional arrays. The calling program can provide any number of elements for the array provided it is less than the maximum.

The following example illustrates array range notation.



```
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 loop limits can be any of the following:
 i) a numerical quantity or expression
 ii) an EES variable that has been set to a value ahead of the Duplicate statement
 iii) an EES variable that is set to a value in the Parametric tableWindow_Parametric
 iv) the TableRun#_TABLERUN_ keyword.

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

$$[\mathbf{A}]\,[X] = [B]$$

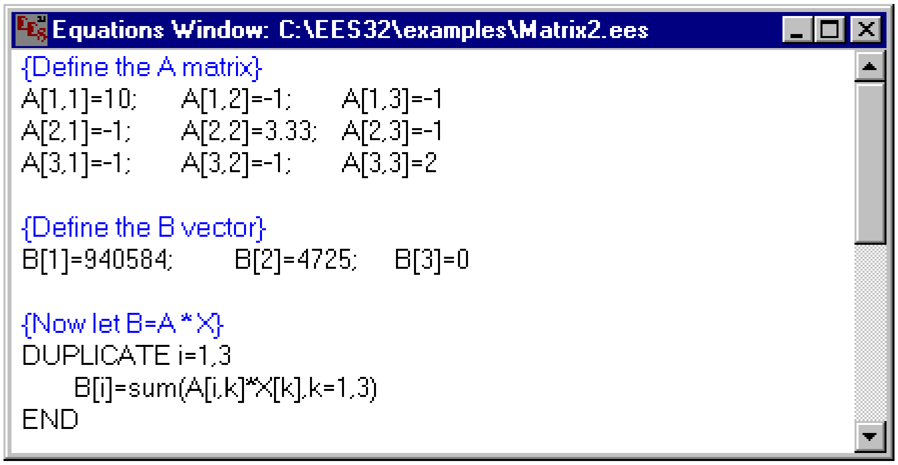
where [**A**] is a square matrix of coefficients, and [X] and [B] are vectors. Ordinarily, the matrix equation is solved to determine the elements in the vector [X] for known [**A**] and [B]. In this case,

$$[X] = [\mathbf{A}]^{-1}\,[B]$$

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.[13]

$$[A] = \begin{bmatrix} 10 & -1 & -1 \\ -1 & 3.33 & -1 \\ -1 & -1 & 2 \end{bmatrix} \qquad [B] = \begin{bmatrix} 940584 \\ 4725 \\ 0 \end{bmatrix}$$

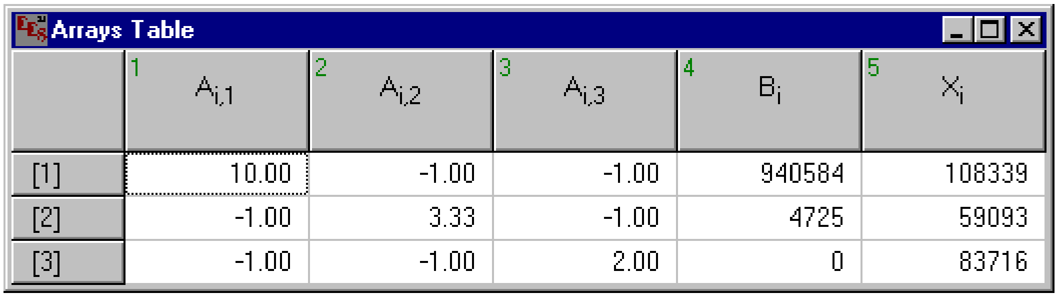
The equations required in EES to solve this problem are as follows:



```
{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
```

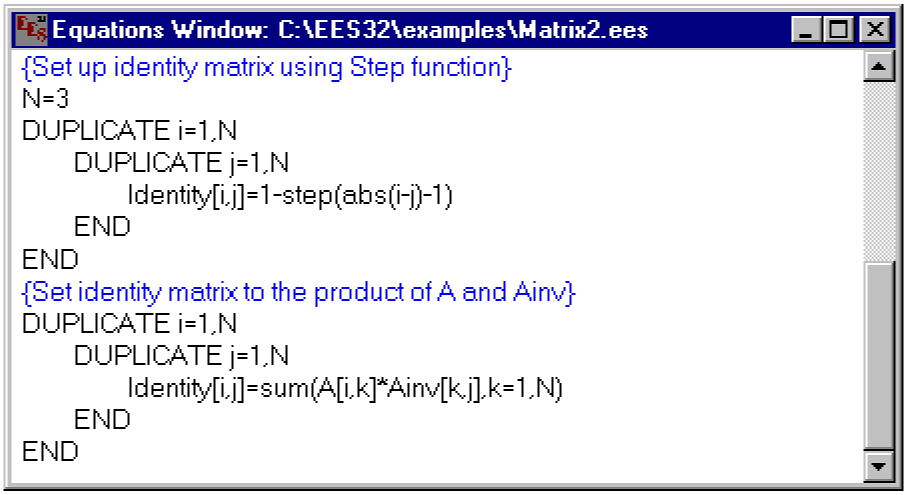
The calculated elements in the X array will appear in the Arrays window.

[13] Incropera, F.P. and DeWitt, D.P., *Fundamentals of Heat and Mass Transfer*, 2nd edition, John Wiley and Sons, 1985, Chapter 13



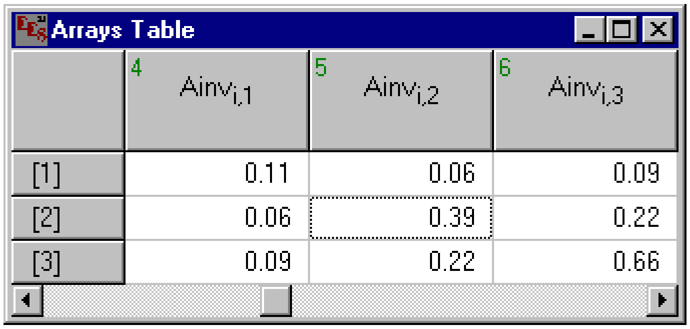
	1 $A_{i,1}$	2 $A_{i,2}$	3 $A_{i,3}$	4 B_i	5 X_i
[1]	10.00	-1.00	-1.00	940584	108339
[2]	-1.00	3.33	-1.00	4725	59093
[3]	-1.00	-1.00	2.00	0	83716

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 $[\mathbf{A}]^{-1}$ can be determined by setting the matrix product [**A**] $[\mathbf{A}]^{-1}$ equal to the identity matrix in the following manner.



```
{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.



	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

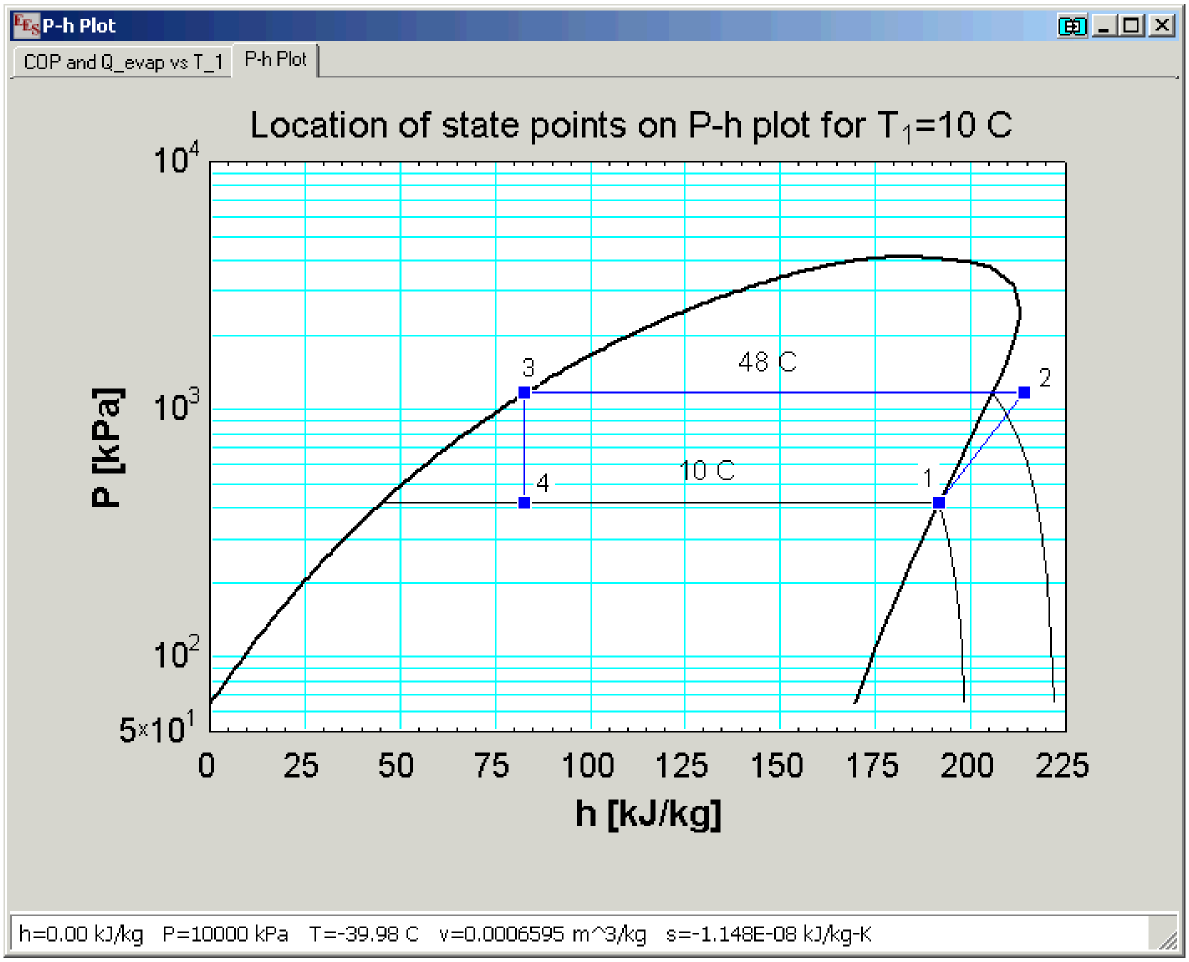
The two examples above provide a general procedure for determining the product of a matrix and a vector or the product of two matrices. Using the DUPLICATE command along with array variables in EES is no more efficient than the alternative of entering each equation separately with non-array variable 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 with the Show Array Index option enabled) for the four state points in a refrigeration cycle analysis. The equations can be found in file REFRIG.EES in the Examples subdirectory.



h=0.00 kJ/kg P=10000 kPa T=-39.98 C v=0.0006595 m^3/kg s=-1.148E-08 kJ/kg-K

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 can vary during the course of the integration. This capability allows, for example, the user to provide a function in which the step-size is small at the start of the integration but increases as the integration proceeds. Note that specification of tStep is optional. If tStep is not provided, EES will select the stepsize using an automatic stepsize adjustment algorithm as specified in the Integration tab of the Preferences dialog.

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.

Solving First-Order Initial Value Differential Equations

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.[14] 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

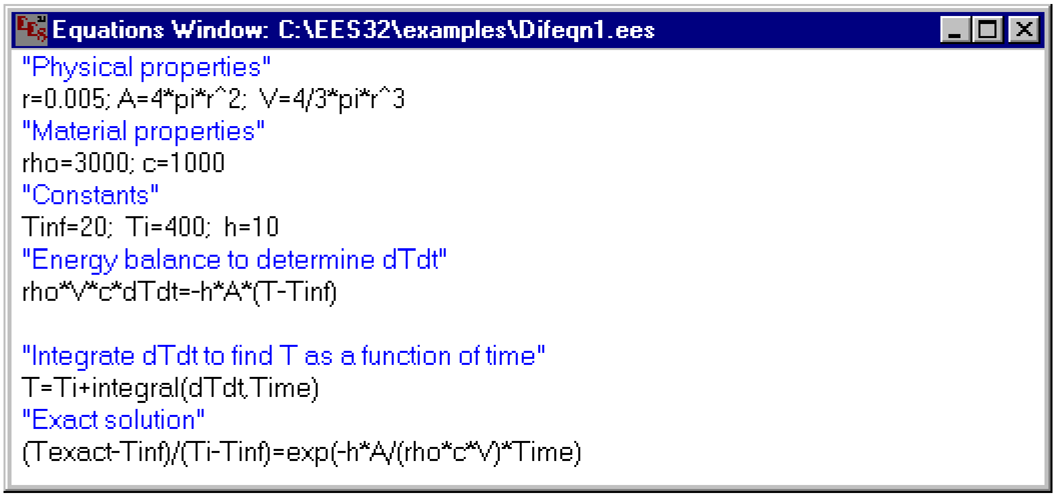
h	is the convective heat transfer coefficient
T	is the uniform temperature of the sphere at any time
T_∞	is the temperature of the air stream = 20°C
A	is the surface area of the sphere = 4 π r^2
V	is the volume of the sphere = 4/3 π r^3
t	is time

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.

[14] Incropera, F.P. and DeWitt, D.P., *Fundamentals of Heat and Mass Transfer*, 2nd edition, John Wiley and Sons, 1985, Chapter 5



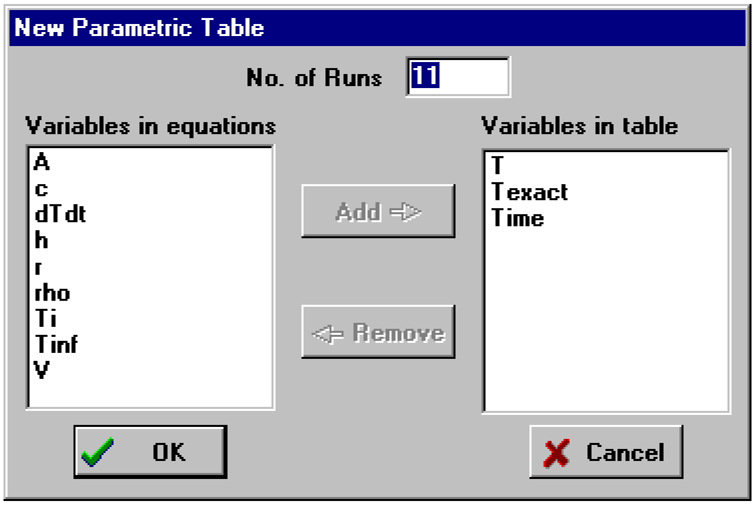
Equations Window: C:\EES32\examples\Difeqn1.ees

```
"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; 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.)

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.

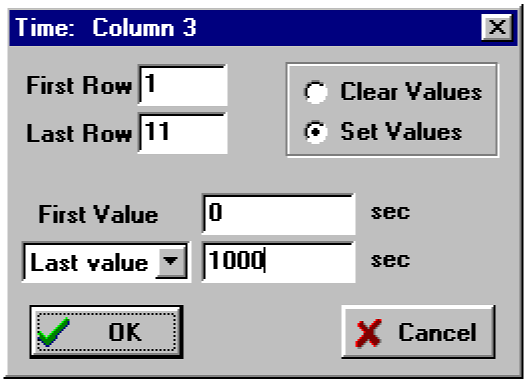


New Parametric Table

No. of Runs 11

Variables in equations: A, c, dTdt, h, r, rho, Ti, Tinf, V

Variables in table: T, Texact, Time



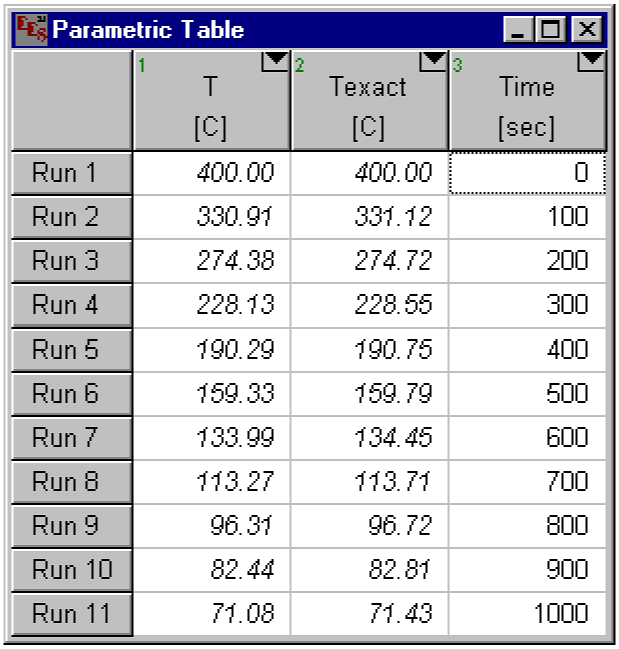
Time: Column 3

First Row 1
Last Row 11

○ Clear Values
● Set Values

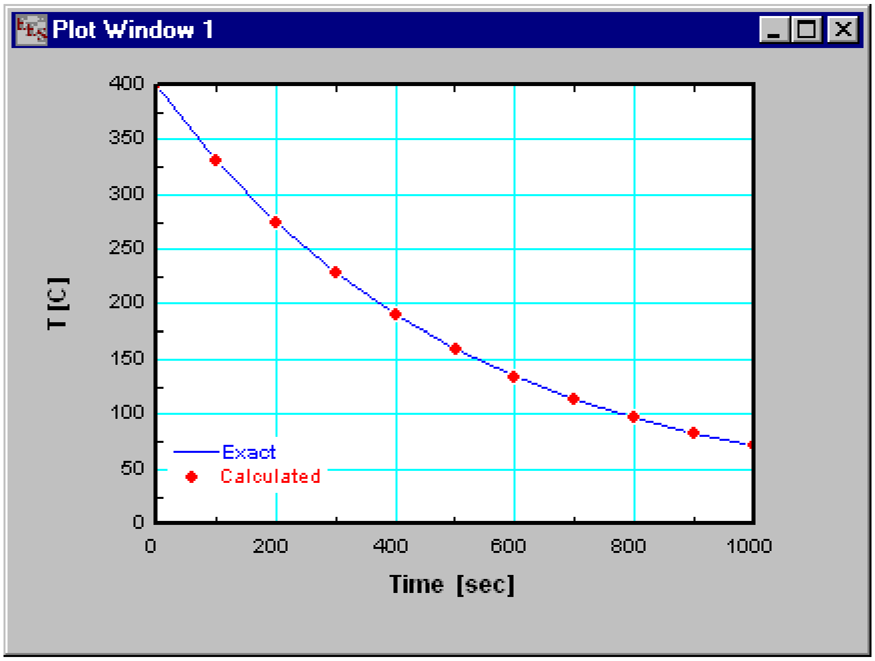
First Value 0 sec
Last value 1000 sec

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


Plot Window 1
400
350
300
250
200
150
100
50
0
T [C]
0
200
400
600
800
1000
Time [sec]
Exact
Calculated

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 or if the step size is 0, 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

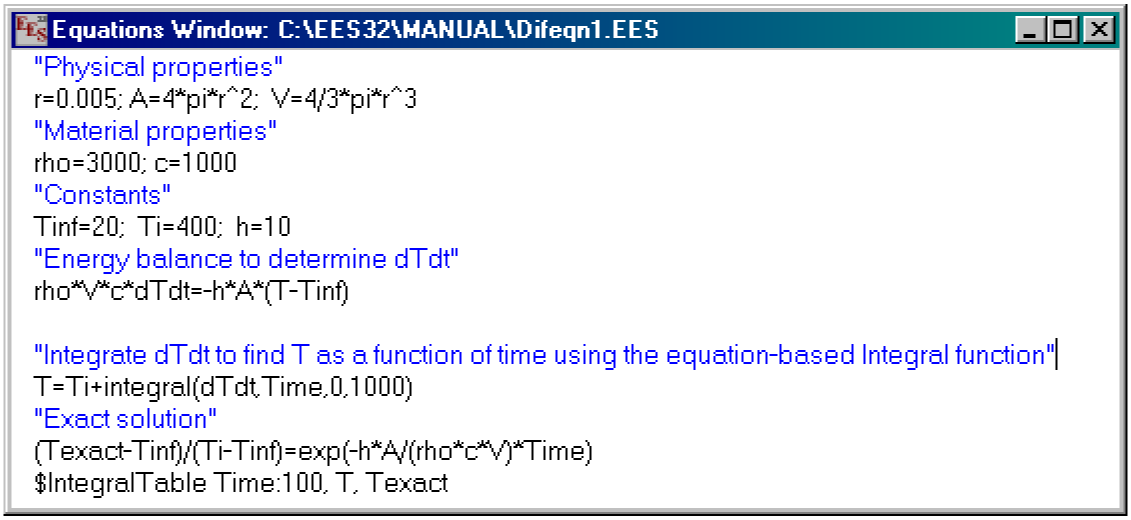
where Time is the integration variable, but it of course can have any legal EES variable name. Time is used here as an example. This variable must appear in one or more equation-based Integral functions within the Equations window. The first column in the Integral Table will hold values of this variable. The colon and number (or variable) 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.

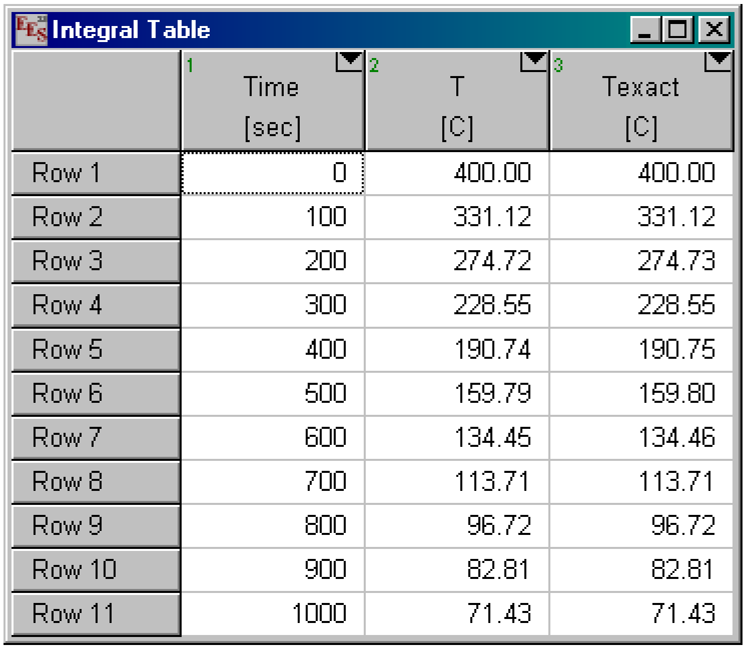
When calculations are initiated, EES will first check to ensure that all variables in the $IntegralTable command are used in the equations. If no errors are found, an Integral Table will be created and filled with intermediate values results from the numerical integration.

Values from this table can then be plotted, printed, and copied, in exactly the same manner as for other tables. The $IntegralTable directive can be supplied in Subprograms and in the Main program. A separate tab in the Integral Table window will be provided for each case.

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.

$$-h\ A\ (T - T_{\infty}) = \rho\ c\ V\,\frac{dT}{dt}$$

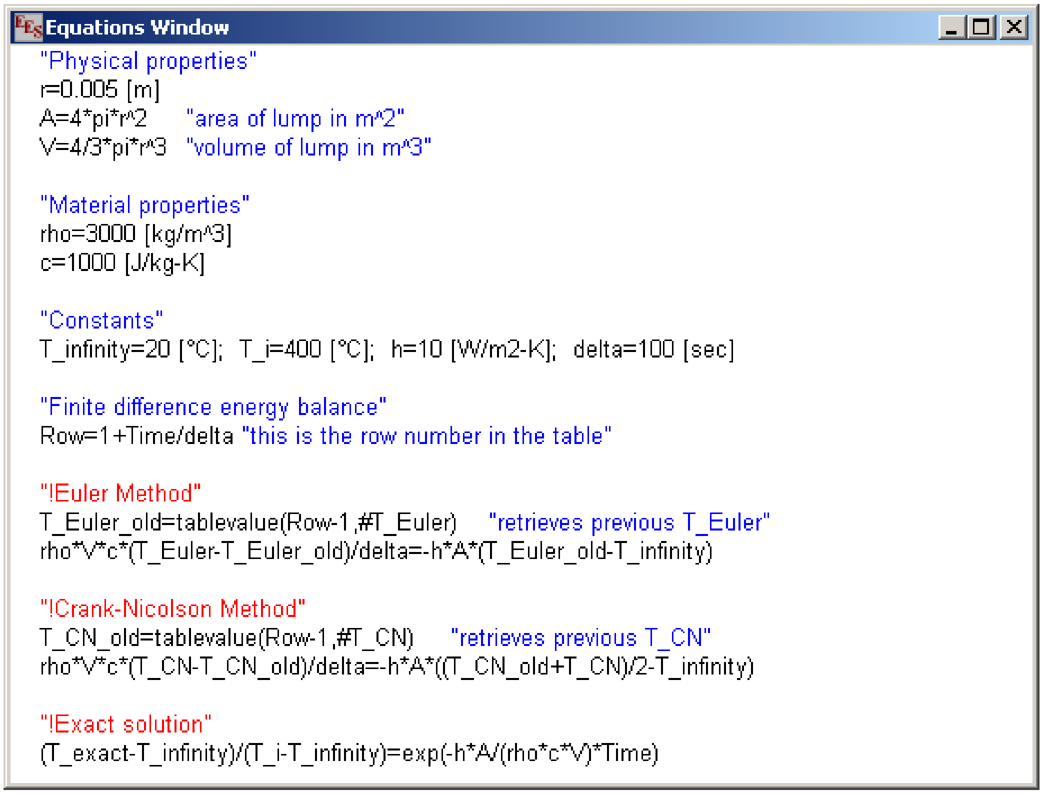
The differential is approximated as

$$\frac{dT}{dt} \approx \frac{T^{new} - T^{old}}{\Delta t}$$

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.

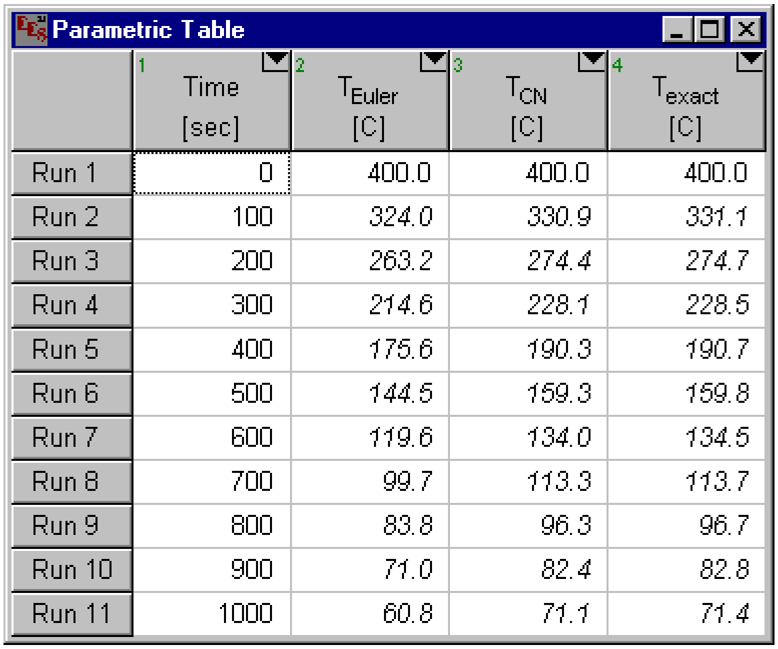
Both an explicit method (Euler's method) and an implicit method (Crank-Nicolson) are used to solve this first-order differential equation and compared with the exact solution. In the Euler method, only previous temperatures are used to evaluate the right-hand side of the differential equation. In the 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 calculations starting at Run 2. 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.



```
"Physical properties"
r=0.005 [m]
A=4*pi*r^2      "area of lump in m^2"
V=4/3*pi*r^3   "volume of lump in m^3"

"Material properties"
rho=3000 [kg/m^3]
c=1000 [J/kg-K]

"Constants"
T_infinity=20 [°C];  T_i=400 [°C];  h=10 [W/m2-K];  delta=100 [sec]

"Finite difference energy balance"
Row=1+Time/delta "this is the row number in the table"

"!Euler Method"
T_Euler_old=tablevalue(Row-1,#T_Euler)     "retrieves previous T_Euler"
rho*V*c*(T_Euler-T_Euler_old)/delta=-h*A*(T_Euler_old-T_infinity)

"!Crank-Nicolson Method"
T_CN_old=tablevalue(Row-1,#T_CN)     "retrieves previous T_CN"
rho*V*c*(T_CN-T_CN_old)/delta=-h*A*((T_CN_old+T_CN)/2-T_infinity)

"!Exact solution"
(T_exact-T_infinity)/(T_i-T_infinity)=exp(-h*A/(rho*c*V)*Time)
```

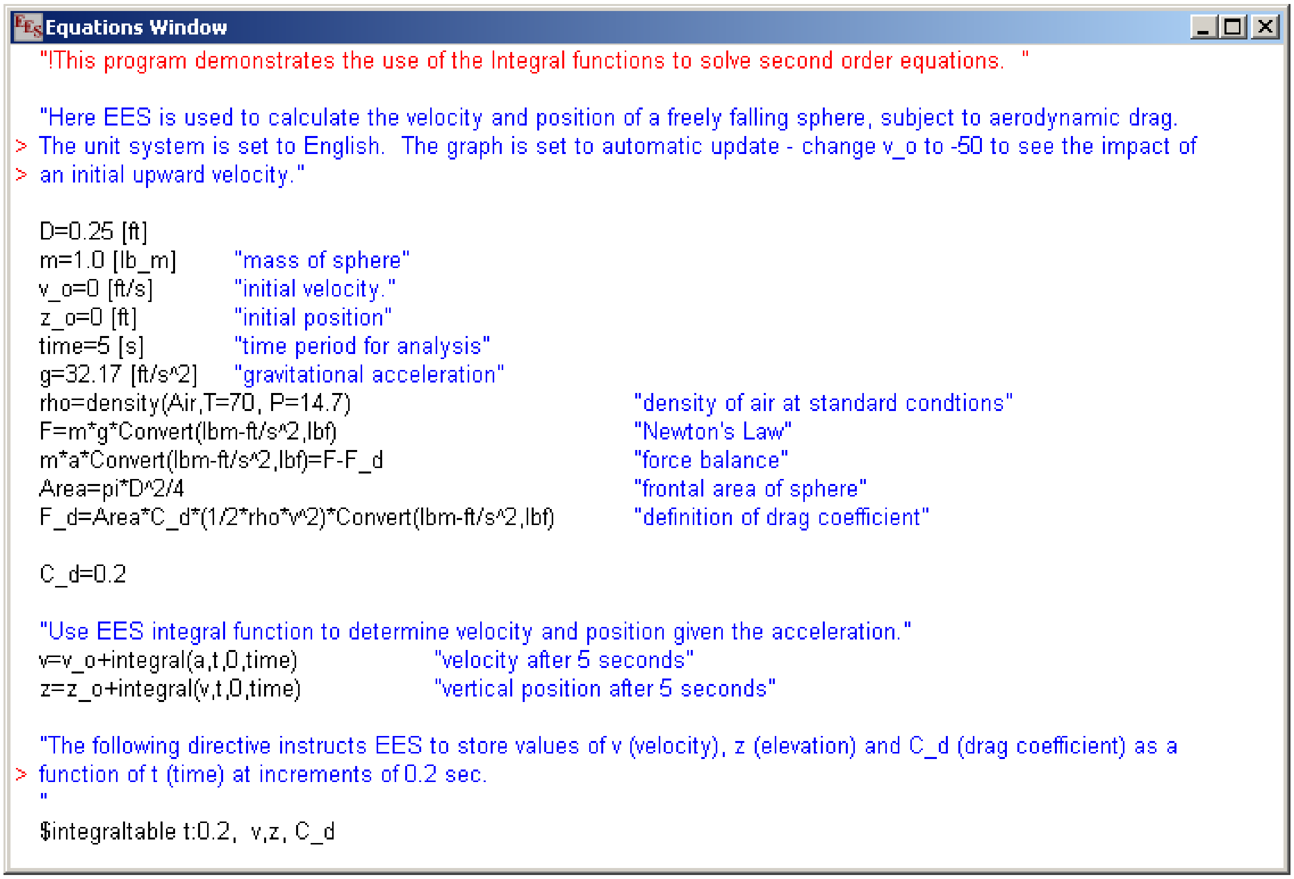
Shown below is the completed table with the numerical and analytical solutions. Calculated values are shown in italics. It is evident that Euler's method does not provide as accurate a solution as that obtained with the **Integral** function (Methods 1 and 2) or the Crank-Nicolson method. Improved accuracy could be obtained by reducing the time step, but this would require additional computational effort and storage space (which is not significant here).



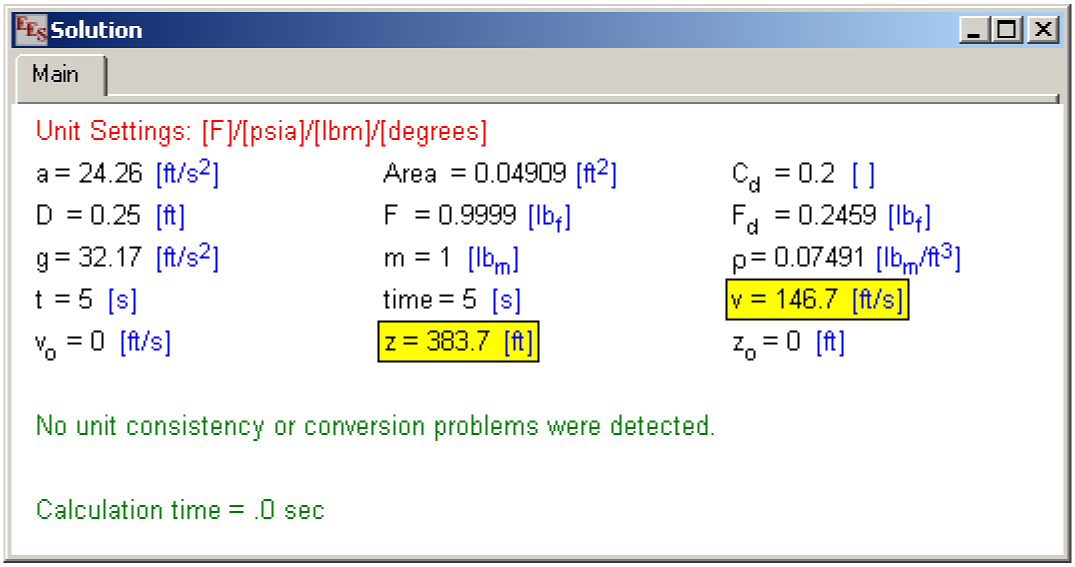
	1 Time [sec]	2 T_{Euler} [C]	3 T_{CN} [C]	4 T_{exact} [C]
Run 1	0	400.0	400.0	400.0
Run 2	100	*324.0*	*330.9*	*331.1*
Run 3	200	*263.2*	*274.4*	*274.7*
Run 4	300	*214.6*	*228.1*	*228.5*
Run 5	400	*175.6*	*190.3*	*190.7*
Run 6	500	*144.5*	*159.3*	*159.8*
Run 7	600	*119.6*	*134.0*	*134.5*
Run 8	700	*99.7*	*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 functions to solve second order equations.  "

  "Here EES is used to calculate the velocity and position of a freely falling sphere, subject to aerodynamic drag.
> The unit system is set to English.  The graph is set to automatic update - change v_o to -50 to see the impact of
> an initial upward velocity."

  D=0.25 [ft]
  m=1.0 [lb_m]       "mass of sphere"
  v_o=0 [ft/s]       "initial velocity."
  z_o=0 [ft]         "initial position"
  time=5 [s]         "time period for analysis"
  g=32.17 [ft/s^2]   "gravitational acceleration"
  rho=density(Air,T=70, P=14.7)                       "density of air at standard condtions"
  F=m*g*Convert(lbm-ft/s^2,lbf)                       "Newton's Law"
  m*a*Convert(lbm-ft/s^2,lbf)=F-F_d                   "force balance"
  Area=pi*D^2/4                                       "frontal area of sphere"
  F_d=Area*C_d*(1/2*rho*v^2)*Convert(lbm-ft/s^2,lbf)  "definition of drag coefficient"

  C_d=0.2

  "Use EES integral function to determine velocity and position given the acceleration."
  v=v_o+integral(a,t,0,time)             "velocity after 5 seconds"
  z=z_o+integral(v,t,0,time)             "vertical position after 5 seconds"

  "The following directive instructs EES to store values of v (velocity), z (elevation) and C_d (drag coefficient) as a
> function of t (time) at increments of 0.2 sec.
  "
  $integraltable t:0.2,  v,z, C_d
```



Unit Settings: [F]/[psia]/[lbm]/[degrees]

$a = 24.26$ [ft/s^2] $Area = 0.04909$ [ft^2] $C_d = 0.2$ [ ]

$D = 0.25$ [ft] $F = 0.9999$ [lb$_f$] $F_d = 0.2459$ [lb$_f$]

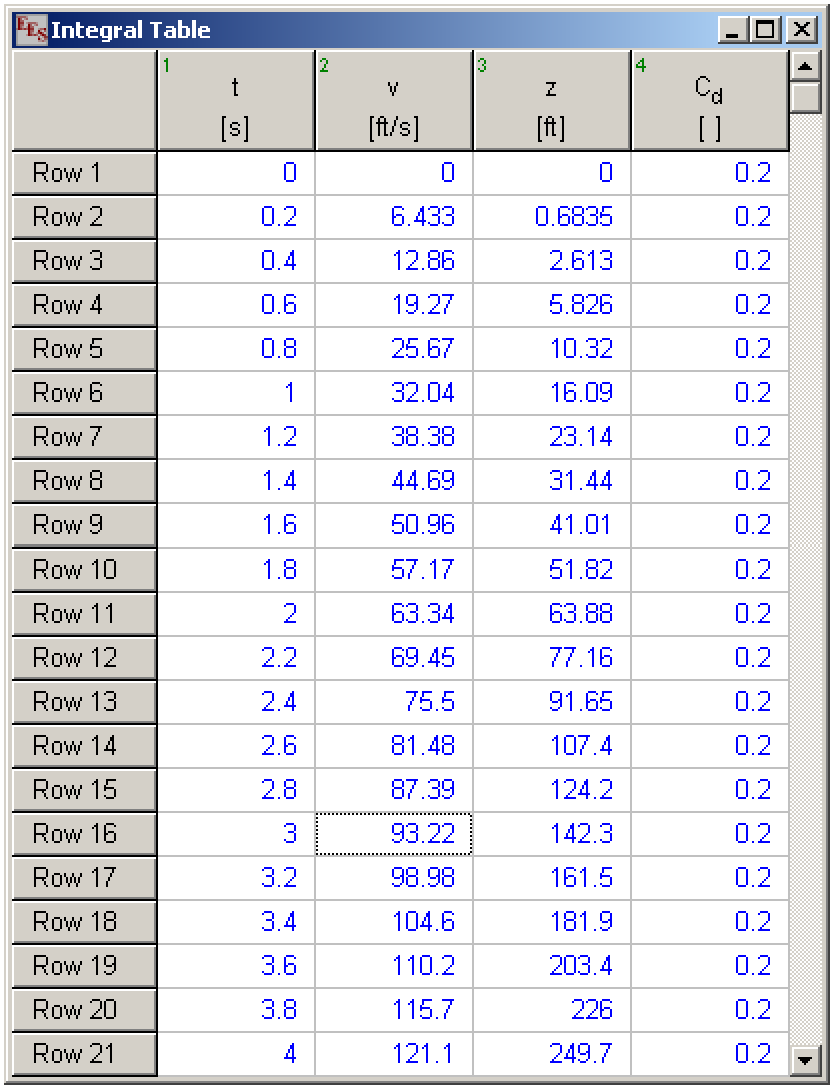
$g = 32.17$ [ft/s^2] $m = 1$ [lb$_m$] $\rho = 0.07491$ [lb$_m$/ft^3]

$t = 5$ [s] $time = 5$ [s] $v = 146.7$ [ft/s]

$v_o = 0$ [ft/s] $z = 383.7$ [ft] $z_o = 0$ [ft]

No unit consistency or conversion problems were detected.

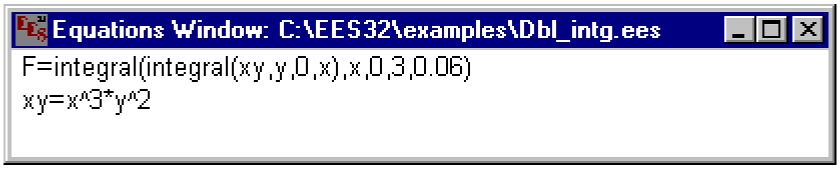
Calculation time = .0 sec



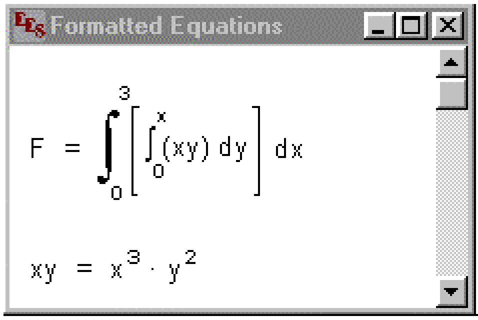
	1 t [s]	2 v [ft/s]	3 z [ft]	4 C_d [ ]
Row 1	0	0	0	0.2
Row 2	0.2	6.433	0.6835	0.2
Row 3	0.4	12.86	2.613	0.2
Row 4	0.6	19.27	5.826	0.2
Row 5	0.8	25.67	10.32	0.2
Row 6	1	32.04	16.09	0.2
Row 7	1.2	38.38	23.14	0.2
Row 8	1.4	44.69	31.44	0.2
Row 9	1.6	50.96	41.01	0.2
Row 10	1.8	57.17	51.82	0.2
Row 11	2	63.34	63.88	0.2
Row 12	2.2	69.45	77.16	0.2
Row 13	2.4	75.5	91.65	0.2
Row 14	2.6	81.48	107.4	0.2
Row 15	2.8	87.39	124.2	0.2
Row 16	3	93.22	142.3	0.2
Row 17	3.2	98.98	161.5	0.2
Row 18	3.4	104.6	181.9	0.2
Row 19	3.6	110.2	203.4	0.2
Row 20	3.8	115.7	226	0.2
Row 21	4	121.1	249.7	0.2

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.

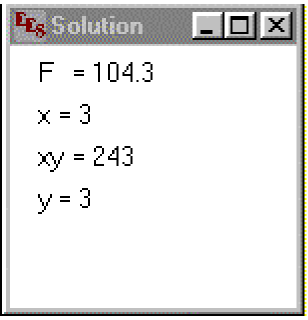


```
F=integral(integral(xy,y,0,x),x,0,3,0.06)
xy=x^3*y^2
```



$$F = \int_0^3 \left[\int_0^x (xy)\, dy \right] dx$$

$$xy = x^3 \cdot y^2$$

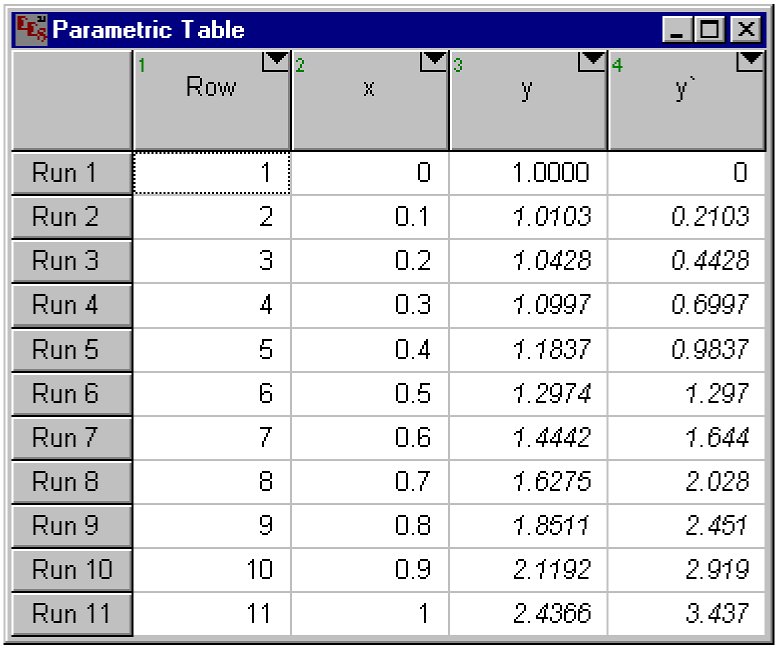


F = 104.3

x = 3

xy = 243

y = 3



	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

Directives

Directives are special EES commands that are placed in the Equations window. Directives are distinguished from equations by the $ that precedes each directive name. The directives are listed alphabetically with a short description of their functions.

$ARRAYS On/Off Directive

Starting with version 8.770, array variables may be displayed in the Arrays table window regardless of whether they appear in the Main program, a Function, a Procedure, a Subprogram or a Module. The default behavior for array variables in the Main program is controlled by an option in the Preferences dialog. As shipped, EES is configured to display array variables in the Main program in the Main tab of the Arrays table window, rather than in the Solution window, but the arrays in Functions, Procedures, Modules and Subprograms are not displayed. This default behavior can be changed by placing a $Arrays ON directive within the Main program, Function, Procedure, Subprogram or Module. In this case, the array variables will appear in a separate tab window of the Arrays table window and they will not appear in the Solution window. A $Arrays Off directive reverses the behavior so that the array variables appear in the Solution window, but not in the Arrays table window.

$BOOKMARK Name

The $Bookmark directive must be on a line by itself with the $ character in the first column. Name is any name that you wish to associate with the text that follows the $Bookmark directive. The Name that you supply will be sorted in alphabetical order and placed in the lower section of the Search menu and in the popup menu that appears when you press the right mouse button within the Equations or Formatted Equations window. Selecting the Name from either of these menus will adjust the vertical scrolling of the Equations or Formatted Equations window so that the information following the selected $Bookmark directive appears near the top of the window.

$CHECKUNITS On/Off/AutoOn/AutoOff

When EES encounters a $CHECKUNITS Off directive, equations that follow will not be subject to unit checking. $CHECKUNITS On restores the unit checking to the equations that follow. Automatic unit checking is controlled with the Check Units Automatically option is set in the General Tab of the Preferences dialog. When automatic unit checking is enabled, the unit consistency of all equations will be checked each time the Solve or Solve Table command is applied. The results of the unit checking will be summarized in the Equations window. Also, when automatic unit checking is on, the Residuals window will display a Units column indicating OK, ? or X for the status of the unit checking of each equation. OK indicates that the units appear to be correct. ? indicates that a problem was found. X indicates that the equation was not checked for unit consistency. The X will appear if the equation follows a

$CHECKUNITS Off directive or if the unit checking was disabled for this equation by right-clicking on it in the Residuals window. $CHECKUNITS AutoOn/AutoOff overrides the setting in the Preferences dialog. To ensure that automatic unit checking is on, enter $CHECKUNITS AutoOn at the top of the EES file.

$CLEARLOOKUP 'LOOKUPTABLENAME' /C2..5 /R1..10

The $ClearLookup directive (Professional version) will clear data from the specified Lookup table with the option to specify the rows and/or columns that are to be cleared. 'LookupTableName' is a string constant or string variable that provides the name of the Lookup table as it appears on the tab of the Lookup Table window. The /C and /R are optional parameters. /C specify a range of columns, e.g., 1..10 that will be cleared. A string variable can be used to provide the range, e.g., /CC$ where C$ contains the range. If the range is not specified, then all columns in the table will be cleared for the specified range of columns. /R specify a range of rows, e.g., 2..5 that will be cleared. A string variable can be used to provide the range, e.g., /RR$ where R$ contains the range. If the range is not specified, then all rows in the table will be cleared for the specified range of columns. The $ClearLookup directive is executed while the equations are being compiled, before the calculations are initiated.

$CLEARPARAMETRIC 'ParametricTableName' /A /R1..10

The $ClearParametric (Professional version) will clear data from the specified Parametric table with the option to specify the rows that are to be cleared. An option is provided to clear just the calculated data cells or all cells. 'ParametricTableName' is a string constant or string variable that provides the name of the Parametric table as it appears on the tab of the Parametric Table window. The /A and /R are optional parameters. /A indicates that all data (including specified cells) will be cleared. If this option is not set, only the calculated data cells are cleared. /R specify a range of rows, e.g., 2..5 that will be cleared. If the range is not specified, then all rows in the table will be cleared. Note that EES automatically clears calculated data from the rows that are solved when the Solve Table, MinMax Table, or Uncertainty Table commands are issued. It is not necessary to use this directive unless data are to be cleared from rows that are not within the calculated range.

$COMMON

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, SUPROGRAM, 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. Problems with calculation order can order if variables that are not set to constants are used in $COMMON.

$COMPLEX On/Off i/j

EES provides an option in the Complex tab of the Preferences dialog to enable or disable complex algebra. The $COMPLEX On or Off directive provides a means of setting the option for complex numbers in the Equations Window. The directive also optionally allows whether i or j is to be used to represent the square root of -1. Subprograms can be configured to operate with complex variables by placing a $COMPLEX ON directive within the text of the Subprogram. The real and imaginary parts of complex variables that are passed to the Subprogram must be passed as separate parameters. The status bar at the bottom of the Equations window will display the complex mode with a red background when it is engaged.

$CONSTANT Name# = ValueSI [UnitsSI] ValueEng [UnitsEng]

The $CONSTANT directive allows the specification of a constant. A constant is similar in some respects to an EES variable in that it has a value and units, but its value cannot be changed. In addition, it is a global (rather than a local) variable in the respect that all EES functions, procedures, subprograms, modules, as well as the main program have access to constants. A constant is recognized by EES by the last character in its name which must be the # character. Constants are normally specified in the Constants.txt file that is read in when EES is started. These constants are permanent constants that are available to all EES files. The difference between a constant that is defined in the Constants.txt file and one that is defined with the $CONSTANT directive is that the latter is only available to the EES file that it is defined within. It is, in this sense, a temporary constant. The $CONSTANT directive in no way changes the Constants.txt file. The format, shown above is as follows:

Name# is the name of the constant. The $CONSTANT directive will allow a constant that has been previously defined in the Constants.txt file to be redefined, but you generally do not want to do this. ValueSI is a numerical value that will be the value of this constant when EES is configured to operate in SI units. ValueSI must be a numerical value and not an EES variable. UnitsSI is the units for the constant that will be used when EES is configured in SI units. Note that the units must be enclosed in square braces. The units specification is optional.

ValueEng is a numerical value that will be the value of this constant when EES is configured to operate in English units. ValueEngI must be a numerical value and not an EES variable. If ValueEng is not provided, it is assumed that the constant has the same value in either unit system. UnitsEng is the units for the constant that will be used when EES is configured in English units. Note that the units must be enclosed in square braces. The units specification is optional.

Multiple constants can be defined, but each requires a separate $CONSTANT directive. The $CONSTANT directive should appear before the constant is used, preferably at the very top of the EES file. The CONSTANT directive will accept simple expressions involving numerical values and previously set constants, e.g.:

```
$CONSTANT N#=100
$CONSTANT N1#=N#-1
```

Note that string constants can be defined. String contants must have a name that ends with characters #$, e.g.,

```
$CONSTANT R#$='R134a'
```

$CONVERTEESREFPROPUNITS /Mass

This directive is only used in conjuction with the EES_REFPROP interface. If a $ConvertEESREFRPOPUnits directive is placed in the Equations window, EES will automatically convert all inputs to and outputs from the EES_REFPROP interface so that they correspond to the unit system set in EES. Specific properties on a molar basis are also converted to a mass basis if EES is set to return properties on a mass basis. The /Mass flag is optional. Normally, the composition of a mixture is specified in terms of mole fractions. Also, the composition of liquid and/or vapor phases calculated by REFPROP are returned as mole fractions. However, when the /Mass flag is provided, all composition indices will be understood to be mass fractions, rather than mole fractions. Note that this opton is only useful when the mixture is specified by using the + symbol between the names of the pure components

$COPYTOLOOKUP Directive

$CopytoLookup provides a simple way of move selected variables in the main program or in Functions, Procedures, or Subprograms to a specified column in a specified Lookup table. This directive provides a simpler and faster alternative to using the Lookup command on the left side of an equation in a Function or Procedure as explained in the use of the Lookup function. $CopyToLookup directive is only implemented in the Professional version.

$DEFAULTARRAYSIZE 200

This directive sets the default size of an array that is specified by an indeterminate variable. Without this directive, the default array size is 100. This directive is only needed when passing an unspecified array, e.g., X[1..N] as a parameter to a Function or Procedure.

$DIAGRAMUPDATE ON / OFF

The $DiagramUpdate directive is useful for some large EES programs that contain a Diagram window. When controls in the Diagram window, such as dropdown lists, radio buttons, check boxes, and sliders, are changed, the Diagram window may recompile the equations in the Equations window depending on whether the Execute EES commands .. check box is checked for these controls. Large programs with many equations and directives may require signficant time to recompile, which can cause an unacceptable delay in the response to the change made in the Diagram window. This situation can be remedied with $DiagramUpdate directives. $DiagramUpdate Off and On directives can be placed around Functions, Procedures, Modules, and/or Subprograms in the Equations window that do not need to be recompiled during the Diagram window updating process. $DiagramUpdate Off will cause all following equations to be ignored until a $DiagramUpdate On directive is found.

$DOLAST and $ENDDOLAST Directives

Equations that are placed between the $DOLAST and $ENDDOLAST directives are compiled and executed after all of the other calculations are completed. The capability to control the order in which equations are executed is seldom needed since EES automatically orders the equations to provide the most efficient calculation. However, EES attempts to compile all equations before the calculations begin. Some equations, such as those involving a Duplicate statement cannot be compiled unless the limits are known apriori. In some cases, information such as the limits can be calculated and then used in further calculations. The $DOLAST directive is useful in these situations. Another applications of the $DOLAST directive would be to set the limits of an Integral function or to control calculation order with Functions that are using $COMMON. Note that each $DOLAST directive must be matched with a $ENDDOLAST directive. More than one set of these directives can be used but they cannot be nested within each other. $DOLAST directives can only be used within the main program, i.e., not within Modules or Subprograms.

$EXPORT Directive

The $Export directive provides a simple way of writing selected variables to an ASCII file. Numerical or string constants may also be include with the variables that are written to a 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  /D  /L  /T  /C  /Q  /U /V  'FileName', Var1, Var2, X[1..5], ...
```

/A is optional. If present, it indicates that the values should be appended to the existing file, if it exits.

/D is optional. If present, it indicates that the exported values are formatted to appear just as they do in the Solutions window.

/L is optional. If present, a carriage return/line feed will not be entered at the end of the list.

/T is optional. Normally, results are exported for each run in the Parametric table. If the /T is present, only the results for the last run in the Parametric table will be exported. This option is ignored if the calculations do not involve the Parametric table.

/C is optional. If present, the file (or clipboard) is cleared before exporting.

/Q is optional. If present, the single quotes that normally surround string information will not be included.

/U is optional. If present, it indicates that the units of the exported values are exported with the values. Units are enclosed in brackets and are separated with one space from the values.

/V is optional. If present, it indicates that the variable names of the exported values are exported with the values. The variable name precedes the value and is separated with a tab. Use the /N option to force each variable to be written on a separate line.

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. If filename is specified to be 'CLIPBOARD', the output will be placed as delimited text on the clipboard. A maximum of 32760 characters can be written to the clipboard. Filename can have a .DAT and .TAB filename extension, in addition to the existing .CSV and .TXT options. .DAT files separate values with one space and .TAB files separate value with a tab character.

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 format of the numerical value for each variable in the $EXPORT statement can be followed with a format specification that controls how it is written to the file. Appending :A (e.g., Var1:A) causes the value to appear in the same format as seen in the Solution window,. :Fn and :En format for fixed decimal or exponential format with n digits where n is an integer between 0 and 9.

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. $EXPORT directives can be placed in Subprograms or in

Functions. The designated variables in the Function or Subprogram will be exported to the specified file after calculations are completed.

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
```

$EXPORTPLOT Directive

The $ExportPlot plot provides a simple way to save a specified plot to a file. The plot is saved after calculations are completed. The format of the $ExportPlot directive is

```
$ExportPlot FileName PlotName /R=N
```

FileName is the name of the file that the plot will be saved to. The format of the graphics file is determined by the file name extension, which can be .bmp, .JPG, .TIF, .GIF, or .EMF. These file name extensions correspond to graphics files of type Bitmap, JPEG (Joint Photographic Experts Group) , Tagged Information File Format (TIFF) , Graphics Interchange Format (GIF), and Enhanced Meta file (EMF), respectively. Filename can be a string constant (enclosed in single quotes) or a string variable that has been previously set to the name of the file. The path name can be included, but if it is not, the file will be saved to the current directory. PlotName is the name of the plot as it appears on the tab of the plot window. PlotName can be a string constant (within single quotes) or a string variable that has been set to the name of the plot. Alternatively, PlotName can be an integer, e.g., 2. In this case EES will set PlotName to be the name of the plot in with this integer position in the plot window. The command will be ignored if the plot window does not exist. The /R option controls the resolution of the plot. This parameter is optional. If it is supplied, N must be an integer between 0 and 4. The values of N have the following meaning.

0 the plot will be saved with the default resolution.
1 the plot will be saved at the screen resolution (100 pixels per inch)
2 the plot will be saved at 300 pixels per inch
3 the plot will be saved at 600 pixels per inch
4 the plot will be saved at 1200 pixels per inch.

Example:

```
$ExportPlot 'C:\temp\myplot.jpg' 1
```

$EXPORTTEXT Directive

The $ExportText directive provides a simple way of writing selected variables to an ASCII file.

Example:

```
    $ExportText /A  'FileName'   S1$, S2$, S$[1..4]
```

$HIDEWINDOW Directive

$HIDEWINDOW Windowname hides the specified window where Windowname can be any of the following keywords:

```
    Equations    Formatted    Plot         Diagram      Solution     Arrays
    Parametric   Lookup       Residuals    Integral     Report       Warnings
```

$IF/IFNOT Directives

The $IF and $IFNOT directives allow conditional exclusion of EES equations that are entered in the Equations window. The format of these directives is

```
$IF Condition
...
...
$ELSE
...
...
$ENDIF
```

These directives test the condition that follows the $IF/IFNOT keyword during equation compilation. The condition can be a simple expression, e.g., T$='K' or a boolean expression using the OR and AND operators. Parentheses can be added to control the boolean order of the evaluation. If the expression is true (for the $IF directive) or not true (for the $IFNOT directive), the equations that follow the $IF/IFNOT directive up to the $ELSE (if present) or the $ENDIF keyword are compiled and included with other equations in the EES Equations window. If the expression is false, the equations are ignored. Multiple $IF/IFNOT directives can be used and they may be nested.

Recognized condition keywords are:

INTEGRALTABLE

This condition is true if a $INTEGRALTABLE directive appears in the EES file. The $INTEGRALTABLE directive can appear above or below this conditional test as EES will initiate a 2-pass compilation if a test for the existence of an Integral Table is required.

DIAGRAMWINDOW

This condition is true if the main Diagram Window is displayed on the screen. In this case, any input variables that are defined in the Diagram Window will be used while compiling the equations. If the Diagram Window is hidden (by selecting the close button at the upper right corner of the window), any input variables defined in the Diagram window are ignored and the DIAGRAMWINDOW condition is false.

MACRO

This condition is true if calculations have been initiated with the Run or Play Macro commands.

MIN/MAX

This condition is true if calculations have been initiated with the Min/Max command.

MIN/MAXTABLE

This condition is true if calculations have been initiated with the Min/Max Table command.

MIN/MAXTABLE='Table Name'

This condition is true if calculations have been initiated with the Min/Max Table command operating on the Parametric table named ‘Table Name’ which must be a string constant enclosed within single quotes. String variable names are not accepted.

PARAMETRICTABLE

This condition returns true if calculations have been initiated with the Solve Table, Min/Max Table or Uncertainty Propagation Table commands. It otherwise returns false.

PARAMETRICTABLE='Table Name'

The condition returns true if calculations initiated with the Solve Table, Min/Max Table or Uncertainty Propagation Table commands are applied to the Parametric Table named 'Table Name'. The table name should be a string constant enclosed in single quotes. String variable names are not accepted.

TABLERUN# = Value or TABLERUN# < Value or TABLERUN# > Value

The condition returns true if calculations are initiated with the Solve Table, Min/Max Table or Uncertainty Propagation Table commands and the run (or row) in the Parametric Table that is currently being solved is equal to Value. Value must be an integer number. When this directive is used, EES must recompile the equations in the Equations window after every run in the Parametric Table, so the calculation speed is reduced.

UNITSYSTEM('XX')

'XX' can be any of the following unit specifications: 'SI', 'Eng', 'Mass', 'Molar', 'Deg', 'Rad', 'kPa', 'bar', 'psia', 'atm', 'C', 'K', 'F', 'R', 'Pa', 'MPa', and 'KJ'. The UNITSYSTEM condition returns true if the unit specification provided as the argument with the UNITSYSTEM condition is selected in the Unit System dialog.

StringVariable$='String constant'

StringVariable$ is any EES string variable that is defined before the $IF directive. The string variable can be defined in the Equations window, Parametric Table or the Diagram window. The text to the right of the equal sign is a string constant. Single quotes around the string constant are optional. Upper and lower case letters are not distinguished when testing equivalence between a string variable and string constant. A dropdown list of strings is the most convenient means of setting a string variable in the Diagram window. A common use of this directive is to set variables that are used with the Solve command but not with the Solve Table command.

UNCERTAINTY

This condition is true if calculations have been initiated with the Uncertainty Propagation command.

UNCERTAINTYTABLE

This condition is true if calculations have been initiated with the Uncertainty Propagation Table command.

UNCERTAINTYTABLE='Table Name'

The condition returns true if calculations are initiated with the Uncertainty Propagation Table command are applied to the Parametric Table named 'Table Name'. The table name should be a string constant enclosed in single quotes. String variable names are not accepted.

Examples:

```
$IFNOT  ParametricTable
  Row=1 {This equation is used only if a Parametric Table is not used}
$endif
x=Row^2

$IF UNITSYSTEM('F')
    TR=(T-32)/1.8  {This equation converts the temperature from F to C}
$ELSE
    TR=T
$ENDIF


X$='abc'
$IF X$='abc
   y=3 {This equation is executed only if X$='abc'}
$ELSE
   y=5 {This equation is executed if X$ is not equal to ‘abc’}
$ENDIF
```

$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

$Import *FileName* //skiplines=10 Var1, Var2, X[1..5], S$...

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. If *FileName* is specified to be 'CLIPBOARD', the values of the variables will be read from the clipboard in text format.

The //skiplines=n is optional. If provided, EES will skip n lines before reading the data.

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.

$IMPORTTEXT Directive
The $ImportText directive provides a simple way of reading string information form a text file or the clipboard into EES String variables.

$INCLUDE Directive
The $INCLUDE directive provides an automatic method for loading a library file, ASCII text file containing EES equations, an EES macro file (.EMF) or a .VAR file that holds variable information. The format is:

$INCLUDE FILENAME

FILENAME is a text string or EES text variable that provides the filename including the filename extension which can be one of.TXT, .LIB, .FDL, .DLF, .DLP, .DLL, .EMF, .UNT, or .VAR. 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. Note that FILENAME can use $USERLIB to refer to the USERLIB directory, e.g.,

$INCLUDE $USERLIB\myfunc.lib.

<u>.TXT Files</u>

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; the text file must not include $INCLUDE statements.

Equations can also be entered from a file with the $INCLUDE directive. 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.

Preference Files (.PRF)
A specified user preferences file will be automatically loaded.

EES Macro Files (.EMF)
If the filename extension is .EMF, EES will look for an EES Macro file and automatically load it into the EES Macro Command window. Note that multiple Macro files can be included by providing a $Include directive for each file.

$INPUT X\Enter value for X
The $Input directive (provided in the Professional version) provides a prompt to enter the value of an EES variable. X is an EES variable. The EES variable may be a string variable, e.g., X$. When solving, the $INPUT directive will display a dialog box prompting the user to enter a value. Entering a value for X is equivalent to providing an equation in the Main section of the Equations window setting the value of X. An optional text prompt may be included by appending it to the variable name separated by a backslash (\).

$INTEGRALAUTOSTEP Directive
$IntegralAutoStep provides an alternative to selecting the integration auto step size parameters from the Preferences dialog. The $IntegralAutoStep directive is processed when the equations are compiled and the specifications made with the directive will override any previous settings. The format of the $IntegralAutoStep directive is

```
$IntegralAutoStep  Fixed=250  Richardson=Off
    or
$IntegralAutoStep  Vary=1  Min=50  Max=2000  Reduce=1e-3  Increase=1e-5
```

Integration with automatic step size is assumed when the equation-based Integral command does not include a step size or when the specified step size is zero. In either case the step size will be chosen so as to provide a fixed number of steps or according to an automatic step-size adjustment algorithm. The first form of the $IntegralAutoStep shown above specifies the fixed number of number of steps and optionally allows the Richardson extrapolation to be turned on or off.

The second form of the $IntegralAutoStep directive instructs EES to use an automatic step size adjustment algorithm that is applied every N steps where N is the number specified with the Vary keyword. The minimum and maximum number of steps can be specified with the Min and Max keywords. The step size is reduced if the relative error tolerance increased above the value provided with the Reduce keyword. The step size is increased when the relative error falls below the value provided by the Increase keyword. All of the keywords, with the exception of the Vary keyword are optional. Note that parameters must be provided with numerical constants. EES variables cannot be used to provide the integration parameter values.

$INTEGRALTABLE Directive

The $IntegralTable directive is used in conjunction with the equation-based Integral function. 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)
```

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 t:0.1,  x,y,z
```

where t is the integration variable. It of course can have any legal EES variable name. t 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 (or variable name) 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. The step size may be a variable name, rather than a number, provided that the variable has been set to a constant value preceding the $IntegralTable directive. If a step size is not specified, EES will output results for each calculation set. In this case, the step size may vary from row to row in the table if automatic step size is used. 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.

x, y, z ... are variables in the EES program. Algebraic equations involving variables are not accepted. 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 (comma or semicolon)

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. Starting in version 7.110, the Integral Table is saved with when the EES file is saved. If an Integral Table exists when calculations are initiated, it will be deleted if a new Integral Table is created. Starting in version 9.125, separae Integral tables can be created for the Main program and for integrals within Subprogram.

$KEYBOARD US/EU Directive

The $Keyboard US/EU allows you to change the system keyboard setting while using EES. For example, suppose you system is set to use the comma as the decimal separator. Entering $Keyboard US will change the setting in EES to use US setting, which assumes a decimal point is the decimal separator and a comma is a list separator.

$LOAD 'Component Library'

To speed the startup process, the Component Library is not automatically loaded at startup even though it resides in the Userlib folder. To load the Component Library, add a $LOAD 'Component Library' directive.

$LOCALVARIABLES On/Off Directive

$LOCALVARIABLES ON/OFF controls whether the values of local variables used in Functions, Procedures, Modules and Subprograms will be displayed in the Solution and Residuals windows. If ON, the local values of all Functions, Procedures, Modules, and Subprograms that appear in the Equations window will be displayed in the Solution window. (Functions, Procedures, Modules and/or Subprograms that have been loaded from a library file will not be displayed.) This directive overrides the Show function/procedure/module value setting in the Options tab of the Preferences dialog. EES only changes the first 5 letters of the keyword, so it is acceptable to use $LOCAL ON/OFF.

$MAXCALLS

Functions and Procedures can employ logic statements such as If-Then-Else and Repeat-Until. In some cases, use of these logic statements results in a repeated sequence of on/off decisions and convergence thereby becomes impossible. The $MaxCalls directive will eliminate such problems by placing an upper limit on the number of times the equations within the Function or Procedure are evaluated. The $MAXCALLS directive is only applicable when placed within a Function or Procedure. The format is:

$MaxCalls=100

where the number provided to the right of the equal sign is the maximum number of times the equations in the Function or Procedure will be called. If the calculations are done in a Parametric table, this number is the maximum number of times the equations in the Function or Procedure will be called for any row in the Parametric table. The number provided must be greater than 2. If the number of calls to the Function or Procedure exceeds the specified limit, the Function / Procedure will return the same results as it did for the last valid call. This capability effectively prevents the logic statements from providing alternating values. Use of the $MaxCalls directive is not recommended unless a problem relating to the use of logic statements has been identified.

$OPENLOOKUP Directive

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 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}
$OPENLOOKUP ?.csv {Bring up a dialog window showing only .csv files}
```

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 $OPENLOOKUP directive can have an optional second parameter which must be a string variable name. The name of the Lookup file that is opened is assigned to this variable. For example:

$OpenLookup ?? myLookupFile$

will bring up a standard file input dialog from which the file name can be selected. That Lookup file will be opened and copied into the Lookup Table and the string variable myLookupFile$ will be set to the name of the file.

The /FORMAT specification is optional. If present, EES will look for the specified format file with a .fmt filename specification and it will read the Lookup file according to the format specification.

The /RENAME specification is optional. Normally, EES will associate the name on the tab of the Lookup window with the name of the file. However the /RENAME specification provides the option to change the table name as desired using either a string constant or predefined string variable.

$PRIVATE Directive

Library files can contain EES Functions, Procedures, Modules and/or Subprograms. These library files can be loaded manually with the Load Library command or automatically by placing the library file in the USERLIB folder. The name of each Function, Procedure, Module, and Subprogram in the loaded library files can normally be seen with the Function Information dialog by selecting the EES library routines radio button. Help information, if available, can be accessed for the routine and the EES code can be viewed. However, in some cases, you do not want a routine to be displayed in the Function Information listing. It may be intended for private use or to only support other routines or perhaps you just don't want others to view the code. Including the $PRIVATE directive will remove the routine from the Function Info display.

$REAL Directive

The $REAL X, Y, Z directive indicates to EES that the variables that follow (e.g., X, Y, Z) are to be considered to be real i.e., their imaginary component is zero. This directive is only useful when operating in the complex mode with a Parametric table. In this case, EES will not require both the real and imaginary components of the variable to be set in the table. The $REAL command should be placed at the top of the Equations window directly following the $Complex ON directive. The variables can be separated by a space or a list separator.

$RECOMPILE

The $Recompile directive forces the Equations window to be recompiled before each run in the Parametric table. This capability is rarely needed and it slows execution speed. However in some circumstances, it is needed such as when the equations change from one run to the next.

$REFERENCE Directive

The $REFERENCE directive allows the default reference state for a specified fluid to be changed. The format for this directive is

$REFERENCE Fluid ReferenceID

where

Fluid is a string constant or string variable that provides the name of the fluid, e.g., R134a. (This directive may be used with the EES_REFPROP interface.)

ReferenceID is one of the following:

DFT - the reference is reset to the default value. See the fluid property information for the specified fluid to determine the default reference state

NBP - the values of specific enthalpy and specific entropy are each set to 0 for saturated liquid at the normal boiling point. Note that this option is not applicable to fluids for which the critical pressure is less than one atmosphere. In this case, the reference will be reset to the default value.

ASH - the values of specific enthalpy and specific entropy are each set to 0 for saturated liquid at -40°C (-40°F). This is the ASHRAE standard reference. Note that this option is not applicable to fluids for which the critical temperature is less than -40°C. In this case, the reference will be reset to the default value.

IIR - the value of specific enthalpy is set to 200 kJ/kg and the value of specific entropy is set to 1.0 kJ/kg-K for saturated liquid at 0°C (273.15 K). This is the standard reference state for the International Institute of Referigeration. Note that this option is not applicable to fluids for which the critical temperature is less than 0°C. In this case, the reference will be reset to the default value.

$REQUIREDOUTPUTS Directive

The $REQUIREDOUTPUTS allows the number of outputs in the Call statement for a Procedure to be less than the number of outputs in the Procedure header statement. This directive must be placed after the Procedure statement but before the END statement for that procedure. If this directive appears in other places or if it indicates that more outputs are required than in the Procedure header statement, it will be ignored. This directive is most useful for library files.

$RUNMACROAFTER /A provides an easy way to embed macro commands within the Equations window, as an option to using the Macro window to play the macro commands.

There are a large number of macro commands that can be used to automatically produce plots, export data, start programs or other activities. As implied in its name, the $RunMacroAfter directive will play the specified macro commands after the calculations have been completed. The directive can be used with the Solve, MinMax and Uncertainty Propagation commands and their related commands that use the Parametric table. If the Parametric table is involved, the $RunMacroAfter will start operation after calculations for all of the rows in the Parametric table have compled unless the option /A flag is provided in which case the macro is run after each row in the Parametric Table. There are two formats for the $RunMacroAfter directive. The first format requires the name of an existing macro file (having an .emf file name extension) to be provided directly following the $RunMacroAfter keyword, .e.g,

```
$RunMacroAfter C:\Temp\MyMacro.emf
```

The second format provides the macro commands in the Equations window between the $RunMacroAfter directive and a required $EndMacro directive, e.g.,

```
$UnitSystem SI C kPa kJ mass
P1=6000 [kPa]
P2=4000
P3=2000
P4=1000
v1=1 [m^3/kg]
v2=5
$RunMacroAfter
   DeletePlot 1
   PropPlot Steam TS 4 P1 P2 P3 P4  2 v1  v2 DoQLines
$EndMacro
```

Note that this macro uses variables defined in the EES program to specify lines of constant pressure and volume in an automatically generated property plot. There are many other ways in which the $RunMacroAfter directive can be employed.

$SAVELOOKUP Directive

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 (identified with a $ as its last character). The last parameter is the name of the file. This name must be a string constant, a string variable that has been set to a valid file name or a ? or ??. If a ? is provided, EES will display a standard save file dialog from which the name of the file can be selected. That file name will then overwrite the ? so that the save file dialog does not reappear. For example,

$SAVELOOKUP 'LookupTableName' ?

will save the data in the lookup table name 'LookupTableName' into a file that is selected from the save file dialog. The ? will be overwritten with the selected filename. If ?? is provided, it will not be overwritten and the save file dialog will appear after every calculation.

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.

$SAVETABLE

The $SAVETABLE directive allows an automatic way to save any table to a .LKT, .CSV or .TXT file. Options are provided to save the column header information, save in EES format, append to existing file, and transpose rows and columns. The $SAVELOOKUP provides many of the same capabilities, but only for Lookup Tables. The $SAVETABLE directive can be used for any table, including Integral tables. Optional parameters are as follows: These parameters are not applicable for .LKT files.

/A append to existing file
/C1..10 save columns 1 through 10
/E save in EES format so that the file can be read directly by the EES Open Lookup Table command (applicable only for .TXT files)
/F Normally, results are exported for each run in the Parametric table. If the /F is present, only the results for the last run in the Parametric table will be exported. This option is ignored if the calculations do not involve the Parametric table.
/N include column name and unit information
/R2..6 save rows 2 through 6
/T transpose rows and columns (can not be used with /E option)
/Q do not surround strings with single quotes

$SAVEVARINFO

The $SAVEVARINFO directive can be used in the Professional version to save variable information (e.g, guess values, limits, units, and format) to a .VAR file. This information can be retrieved with a $INCLUDE directive, or with the Load buttons in the Variable Information dialog and the Diagram window. This directive provides an automatic way to save the variable information in a Distributable program.

$STOPCRITERIA

The $STOPCRITERIA directive provides an alternative way to set the stop criteria that control the results of iterative calculations. The format of the $StopCriteria directive is:

$StopCriteria Iterations=200 Residuals=1e-5 Variables =1e-9 Time=3000

Note that it is not necessary to enter all four parameters and that they can be entered in any order.

$SHOWWINDOW WindowName1 WindowName2

After EES successfully solves, it displays a window with calculated results. By default, the Solution window is brought to the front after the Solve, Min/Max or Uncertainty Propagation commands are completed provided that the calculations were not initiated with a Calculate button in the Diagram Window. The Solve Table, Min/Max Table and Uncertainty Propagation Table commands display the Parametric table after the calculations are completed. In some cases, it would be better to display a different window, e.g., the Arrays Window or the Diagram Window. The $ShowWindow directive provides the capability to display a selected window after calculations are successfully completed. WindowName1 can be any of the following:

Equations Formatted Plot Diagram Solution Arrays
Parametric Lookup Residuals Integral Report None
Warnings

If WindowName1 is Solution, Plot, Parametric, or Lookup, then WindowName2 can be the name of any of the tabs that appear in the window.

$SUMROW On/Off Directive

EES provides an option in the Preferences dialog to display a sum row in the Parametric table. This option and other preferences are not saved with a EES program but rather with the user preferences. In some situations, it is desirable for the EES program to control whether the sum row is to be used. This can be done simply with a $SUMROW On or $SUMROW off directive which in the Equations window.

$SYNTAX OFF/ON Keywords(case, style) FuncNames(case, style)

The $Syntax directive provides an alternative to using the Syntax Highlighting page of the Preferences dialog to set the case and style for function names and keywords. It also provides a convenient method to set the syntax hightlighting preferences for each individual file. Case can be one of following: (ASTYPED, LOWERCASE, UPPERCASE, 1STLETTERUC) and style can be one of the following: (PLAIN, ITALIC, BOLD, UNDERLINE, BOLDITALIC)

$TABSTOPS Directive

$TABSTOPS sets one to ten tab stops that are used in the Equations window. The format of this directive is

```
$TABSTOPS   TabStop1 TabStop2 ... TabStop5  Units
```

where TabStop1 through 5 must be numerical values, separated by one or more spaces or list delimeters. The tab stop values should be entered in increasing numerical order. Units must be either in or cm. If no units are specified, in is assumed if the Unit System is set to English units and cm is assumed for SI units. Example:

```
$TABSTOPS   0.5   2  3 in
```

will set the tabstops at 0.5, 2 and 3 inches.

Tabstops can also be set in the Equations tab of the Preferences dialog. However, if a $TABSTOPS directive appears in the Equations window, the setting made with this directive override the default settings appearing in the Preferences dialog.

$TABWIDTH Directive

$TABWIDTH provides exactly the same capability as the $TABSTOPS directive.

$TRACE Directive

$TRACE allows the values of selected variables to be recorded during the process in which the equations are iteratively solved. The format is

```
$Trace x, y, z
```

where x, y, and z are variables for which an iteration to iteration trace is needed.

$UNITSYSTEM Directive

The unit system is needed only when the built-in thermophysical (e.g., enthalpy) or trigonometric (e.g., sine) functions are used. Normally, the unit system is specified with the Unit System menu command. However, the unit system can also be specified with the $UnitSystem directive which has the following format:

```
$UnitSystem SI[or ENG]  MASS[or MOLE]  DEG[or RAD]  KPA[or PSIA]  BAR[or ATM]
C[or F] K[or R]
```

The unit specifications can be made in any order and only those which are to be changed need to be provided. Note it is not possible to specify a mixed unit system. That is, if the unit system is SI, specifying F for the temperature will result in temperature units of C, not F. Similarly specifying psia for pressure will result in pressure units of kPa.

Since the unit system specifications are saved with an EES file, there is no need to provide a $UnitSystem directive in an EES file. However, the unit system is not saved with text files that may be read into EES, possibly under the control of a macro command file. In this case, the $UnitSystem directive may be need to ensure that the unit system is configured as needed.

A $UnitSystem directive can be specificed for for Functions, Procedures, and Subprograms (but not Modules) so that the unit settings are different than those in the Main program. If a Function, Procedure, or Subprogram with specified unit settings calls another Function, Procedure, or Subprogram for which specific unit setting have not been specified, the unit system of the calling routine will remain in effect. If a $UnitSystem directive appears in a Function, Procedure or Subprogram, the unit setting will be displayed in the Solution window tab corresponding to this program unit. Unit consistency checking will be conducted using the specified unit settings in the program unit. Note that the units of local variables in Functions, Procedures, and Subprograms must be set using the Variable Information dialog.

$VARINFO Directive

The $VarInfo directive allows variable information (e.g., the guess value, limits, units, display format, etc.) to be entered from the Equations window, rather than from the Variable Information dialog. This directive is very useful when it is necessary to enter commands to EES from a macro or from a text file, as for example, when using the internet EES_App. A separate directive is needed for each variable. The format of the VarInfo directive is as follows:

$VarInfo VariableName **Guess**=V1 **Lower**=V2 **Upper**=V3 **Units**='XXXX' **AltUnits**='YYYY **Display**=F1 **Style**=Bold **Key**='Key variable comment'

VariableName must be the first parameter. The variable name does not need to previously defined, so the location of the $VarInfo directive is not important. However, it is best to place the $VarInfo directives before the first equation in the main section of the Equations window. If the variable name is an array, e.g., T[], the variable information specifications are applied to all variables in the array. Following VariableName, the remaining part of the directive consists of one or more keywords (shown in bold above) followed by an equal sign and a specification. It is only necessary to provide the keyword=specification terms that you are interested in changing and they can be provided in any order. The **Units** and **AltUnits** specifications can be a string constant (within single quotes) or a string variable. The **Key** specification indicates that the variable should be considered to be a key variable. The comment that appears in the Key variable tab of the Solution window is provided after the equal sign in quotes.

The $VarInfo directive can be used for variables in Functions and Procedures as well. Functions and Procedures use assignment statements, rather than equalities, so the Guess, Lower and Upper keywords are not applicable. However, the Units, Display and Style for variables in Functions and Keywords can be set with this directive.

Example:

Calculate the specific enthalpy of steam in variable h and set its units to 'kJ/kg' and its alternate units to Btu/lbm.

```
$UnitSystem SI C kPa kJ mass
T=100 [C]
P=50 [kPa]
$VarInfo h Units='kJ/kg'  AltUnits='Btu/lbm'
h=enthalpy(Steam,T=T,P=P)
```

$WARNINGS On/Off Directive

$Warnings ON/OFF directives can be used in the Equations window to specifically control which equations can or can not issue warning messages. The Warnings control in the status bar of the Equations window controls whether the Warning Information dialog window automatically appears after calculations are completed when there are warnings. The Warning Information dialog window can be manually displayed by selecting Warnings from the Windows menu.

Running EES from an External Program

EES (Professional version) can be started and made to run an EES file or an EES macro file from the Run command line (or from another program such as MATLAB) by providing the name of the file and /Solve as command line parameters. For example, the command line C:\EES32\EES.EXE myEESFile.EES /solve will start EES in a hidden state, open myEESFile.EES, solve this file and then quit. For this capability to be useful, the EES file should write output to a file using the $EXPORT or $SAVETABLE directives.

Creating and Using Macro Files (Professional Version)

A macro is a set of instructions to EES that are read from an ASCII file or from the EES Macro window. EES Macros are similar in function to the capabilitiess of the MATLAB Command window. The filename extension for an EES 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 many other functions. In fact, almost all of the capabilities provided with the menu commands in EES can be implemented with a macro instruction. In addition, there are many other capabilities that only be implemented with an EES macro. EES variables can be assigned a value in a macro files. For example, the Macro command, X$='GUESS' sets the string variable X$ just as it would if this statement appeared in the Equations window. 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
```

If EES is launched from the command line with the /NOSPLASH parameter, the splash (or startup) screen will be skipped, as in the following example:

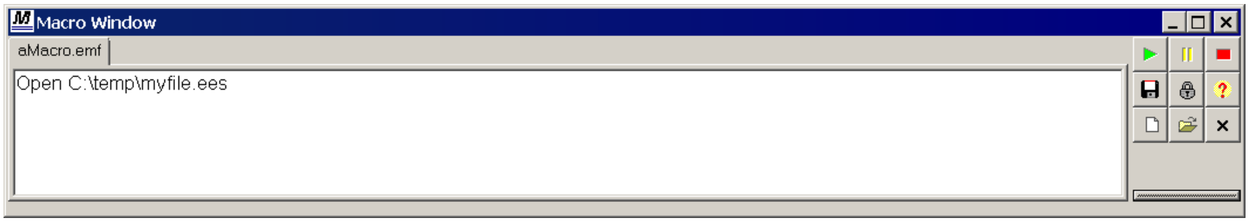
```
C:\EES32\EES.exe C:\EES32\HELLO.ees /NOSPLASH
```

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 Open or Create 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:\temp\myfile.ees
```

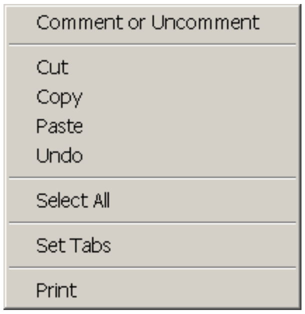
Note the control buttons on the palette at the right of the EES Macro window. They have the following functions.

▶ Play the macro	⏸ Pause execution	🟥 Stop the macro
💾 Save the macro file	🔒 Lock the Macro Window	? Show macro help info
🗋 Open a new macro	📂 Open an existing macro	✕ Close the current macro

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. Each set of macro commands is contained in a separate tabbed window. The active macro is determined by which tab set is selected. The active macro will be played when the Play button is clicke.d

One 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 Export variables and/or save the contents of the Solution and Parametric Table windows so that the solution can be returned to the program that called EES. See the EXCEL_EES.xls file for an example on how Microsoft EXCEL can call EES. See Chapter 18 in the Mastering EES book for detailed information on the use of macros.

Right-clicking the mouse in the Macro Window will display the following pop-up menu.



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;
```

EES Variable Assignment in a Macro

EESVariable = value or expression //Assign an EES variable to a value or to an expression involving previously defined variables. The variable remains assigned to this value until it is reassigned. A variable assigned with a macro command cannot be reassigned in the Equations window. Note that // signifies the start of a comment. Comments can be placed anywhere in the macro file.

Macro Command List

Macro commands can be entered in the Macro Command List window accessed from the Open or Create Macro item in the File Menu. Multiple commands can be entered on a single line if separated by a line break character (;)

AddPlotText 'PlotTabName' Xpos Ypos 'PlotText' TextSize Plain Red

{Create a text item on the plot window that is identified with 'PlotTabName' as the name on the tab. PlotTabName can be a string constant (surrounded with single quotes) or a string variable that has been defined. Xpos and Ypos determine the starting point of the text. Values of Xpos, Ypos of 0, 0 correspond to the lower left corner of the plot rectangle. Values of Xpos,Ypos of 1, 1 correspond to the upper right corner of the plot reactangle. PlotText can be a string constant or a string varaible. TextSize is the font size of the text; it can be supplied as a numerical constant or as an EES variable. The font style must be plain, bold, or italic. The color can be supplied as a string constant or string variable. The colors are identified in the NewPlot macro below.}

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. Row numbers can be specified with pre-defined variables.}

Beep 3 {Execution of this command will result in an audible sound. An integer between 1 and 5 can optionally follow the beep keyword. If the integer is present, EES will play the specified sound defined in the Sound tab of the Preferences dialog. Otherwise it will just sound the default beep.}

Check {Check the equations. A summary statement is not issued. This command is useful only in rare situations in which it is necessary to have EES review the existing equations before processing following Macro commands.}

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 Diagram 'Child Window Name' {Copy the specified Diagram window to the clipboard. A window name, e.g., 'Child Window Name' can optionally be provided as a string constant (within quotes) or as a string variable. The Diagram window corresponding to the provided name will be copied to the clipboard. If the name is not found, or if the name is 'Main' or if no name is provided, the main Diagram window will be copied to the clipboard.

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}

Copy SolutionWindow {Copies the entire contents of the Solution window in ASCII format}

Delay 100 {Pause execution for a specified number of microseconds. This command is useful when the EES macro file is being cycled and EES is periodically reading a file (with its Import command) that has been written to a disk by another application, such as a data acquisition program. This command differs from the Pause command only in that the argument is in microseconds.}

Delete 'FileName' {Delete the specified file. FileName can be a string variable (ending with character $) or a filename enclosed in single quotes}

DeleteLookup 'TableName' or DeleteLookup 1 {Delete the specified Lookup table. The Lookup table can be indicated by: 1) providing its name (optionally surrounded in single quotes), 2) providing a string variable that contains the name of the table or 3) by providing a number indicating the position of the table. The first table is number 1.}

DeleteParametric 'TableName' or DeleteParametric1 {Delete the specified Parametric table. The Parametric table can be indicated by: 1) providing its name (optionally surrounded in single quotes), 2) providing a string variable that contains the name of the table or 3) by providing a number indicating the position of the table. The first table is number 1.}

DeletePlot 'myPlot' {delete the plot window having tab name “myPlot”}

DuplicatePlot 'my Plot' 'your Plot' {If a plot window having a tab name of 'my Plot' exists, create a copy of the plot and set its tab name to 'your Plot'. Note that the names of the plot window tabs must be either string constants within single quotes or predefined EES string variables. The test for the plot tab name is case insensitive.}

GetFistFile$(*.XXX,RECURSE) will return the computer file name (including the path) of the first file found in the current directory having a file name extension of *.XXX. The file name is assigned to a string variable. The directory can be selected with the SetDirectory macro command. The GetFirstFile$ command is normally used with the GetNextFile$ command do obtain a complete list of all files in a directory with the specified file name extension. The RECURSE parameter is optional. If present, the function will look in all subdirectories within the directory set at the start.

GetNextFile$ can only be used after the GetFirstFile$ macro command. It returns the next file in the directory having the file name extension supplied in the GetFirstFile$ command. Each use of the GetNextFile$ command will increment an internal counter and return the following file name in the directory. After the last file is found, a following call to GetNextFile$ will return a null string ('').

GOTO 10 //Transfer execution of the next macro command to the line having label 10. Labels must be numerical values ranging between 1 and 30000 followed by a color (:). Labels will usually ge used with the IF THEN ELSE command, e.g., IF (X<Y) then GOTO 10

DeletePlot 1 {Delete Plot Window 1.}

EXCEL.Cell(A1,'text') {The two parameters separated by a list separator character can be predefined EES string variables. The first parameter indicates the cell location. The letter provides the column and the number provides the row. The second parameter provides the text that will be placed in the specified cell in a previously opened EXCEL file.}

EXCEL.FileNew {Start EXCEL if it is not already open and create a new empty worksheet.}

EXCEL.FileOpen('FileName') {Start EXCEL and open the specified EXCEL file. It is necessary to provide the complete file name, including the filename extension, e.g., 'C:\EES32\USERLIB\EXAMPLES\EXCEL_WS1.xlsx'.}

EXCEL.FileSaveAs('FileName') {Save the current EXCEL file with the specified filename. It is necessary to provide the complete file name, including the filename extension, e.g., 'C:\EES32\USERLIB\EXAMPLES\EXCEL_WS1.xlsx'.}

EXCEL.Hide {Hide the open EXCEL application}

EXCEL.Paste {Paste the current contents of the Clipboard into EXCEL at the current selection point. (See EXCEL.RANGE)}

EXCEL.RANGE('A2:C4') {Set the selection for the active EXCEL worksheet. In this example, A2 is the upper left corner of the selection and C4 is the lower right corner of the selection. To set the insertion point for a paste operation, set the upper left and lower right corners to the upper left of the insertion point, e.g., 'A1:A1'.}

EXCEL.Quit {Close EXCEL and shut down communication.}

EXCEL.Sheet('Sheet2') {Activate the EXCEL sheet with the given name in the current EXCEL application. If this sheet does not exist, a new sheet having this name will be created and activated.}

EXCEL.Show {Make the open EXCEL worksheet visible.}

Export /A 'FileName' X,Y,Z {Export opens the specified file and writes the values of the specified EES variables to that file. Filename can be Clipboard. This command has the same format and options as the $EXPORT directive. For example, the optional /A option appends the specified data to the existing file.}

ExportText /A 'FileName' S$[1..4] opens the specified file and writes the values of the specified EES variables to that file with each string on a separate line. Filename can be 'Clipboard'. This command has the same format and options as the $ExportText directive. Array range notation can be used to specify the string variables. The optional /A appends the specified data to the end of an existing file.

GetDirectory {F$=GetDirectory will assign the specified string variable (F$ in this case) to the current Windows directory.}

GOTO 10 {Transfer execution of the next macro command to the line having label 10. Labels must be numerical values ranging between 1 and 30000 followed by a colon (:). Labels will usually be used with the IF THEN ELSE command, e.g., IF (X<Y) then GOTO 10}

HideWindow WindowName {This command will hide the specified window. WindowName can be any of the following: Equations, Formatted, Solution, Residuals, Arrays, Parametric, Lookup, Integral, Plot, Diagram, EES, ErrorMessages}

If (X<=Y) then Z=3 else Z=4 {The If THEN ELSE command in a macro file can test variables that have been defined in the macro or set in the previous execution of the Solve command. The tests can use any of the following logic operators: <, <=, =, <>, >, >= The entire command must be all on one line. The ELSE clause is optional. The IF command will usually be used with the GOTO command and line labels. The IF statement in a macro file can use one or more AND and OR logical operators. The AND/OR operators must be surrounded by spaces. The order in which the logical tests are processed can be controlled by using parentheses. The entire IF statement must be placed on one lines. As an example, the following IF statement is valid in a macro.

```
IF ((x>=y) OR (y=2)) AND (Z=3) then SOLVE}
```

Import 'FileName' X,Y,Z {Import opens the specified file and reads the values of the specified EES variables from that file. Filename can be Clipboard. This command has the same format and options as the $IMPORT directive.}

ImportText 'FileName' S$[j..k] ImportText opens the specified file and reads the text information in the specified EES string variables. Filename can be Clipboard. Note that array range notation can be used with a string array, provided that the indices (j and k) are predefined. This command has the same capabilities and options as the $ImportText directive.

InsertLookupColumn 'TableName' After 'ColumnName\Units' ColumnFormat
{Inserts one column into the existing Lookup table that has name (showing on the tab in the window) TableName. TableName may be a string variable containing the name of the table. The column is inserted after column number After, where After is an integer. If After is 0, the column will be inserted as the first column in the table. If After is greater than the number of columns in the table, the column will be inserted as the last column in the table. 'ColumnName\Units' is optional. If 'ColumnName\Units' is provided, the column name will be set to "ColumnName' and its units will be set to Units. 'ColumnName\Units' may be a string variable. Note that a \ separates the column name and the units. If a \ is not provided, the units will be left blank. ColumnFormat is optional. If this field is provided, the 'ColumnName\Units' field must also be provided. ColumnFormat is a 2 digit field. The first digit can be A (auto), F (fixed decimal), E (float decimal) or S (string). The second digit is a number between 0 and 9.}

InsertLookupRows 'TableName' After NRows
{Inserts NRows into the existing Lookup table that has name (showing on the tab in the window) TableName. TableName may be a string variable containing the name of the table. If NRows is not provided, it is assumed to be 1. The row(s) are inserted after row number After, where After is an integer. If After is 0, the row will be inserted as the first row in the table. If After is greater than the number of rows in the table, the row will be inserted as the last row in the table. After and NRows may be variables defined in the macro or in the EES program.}

Load #n {This command is useful only for macro files (.emf filename extension) that control the execution of a distributable file. The distributable file can contain up to 100 separate programs. This command will clear the memory space and then load program n where n is a number between 1 and the number of programs provided with the distributable program.}

LoadLib C:\EES32\myLib.lib {Load a library file called C:\EES32\myLib.lib}

Log 'Message' or ON/OFF {If the word OFF appears after the Log keyword, logging of information to macro log file will cease. If the word ON appears after the Log keyword, logging of information to the macro log file will be resumed. If a message other than ON or OFF follows the Log keyword, the message will be written to the log file that is created when the macro is played. 'Message' can be a string constant or an EES string variable. Macro commands are processed sequentially. The log file may be helpful for debugging a macro command file.}

LogFileName FileName {By default, EES writes a log file showing each of the macro commands that it executed. The log file name is named, by default, EESMacro.log, and it is placed in the same directory as the macro file. However, you can change the filename for the log file with the LogFileName macro command. The Filename that follows the LogFileName keyword normally has a .log filename extension, but any extension representing a text file is appropriate. The directory can be indicated in the filename. If a directory is not specified, the log file will be placed in the same directory as the macro file. Note that Filename cannot be represented with an EES string variable. If a filename is not provided after the LogFileName keyword (i.e., the filename is blank) a log file will not be written.}

Lookup['Lookup 1', 2,3]=10 {Set the value in row 2 column 3 of the Lookup table having name 'Lookup 1' to 10}

LookupColInfo 'LookupTableName' Column# 'ColumnName' 'Units' {LookupColInfo sets the name and units of a column in an existing Lookup table. The name of the table (which appears on the window tab) is 'TableName'. 'TableName' can be a string constant or EES string variable. Column# is an integer between 1 and the number of the columns in the table. Column# can be provided in a variable defined in the macro. ColumnName and Units are the name of the column and its (optional) units. Both can be string constants or string variables.}

Maximize G a,b,c {Maximize variable G with respect to variables a, b, and c. Note that the all variables must be defined in the Equations Windowand variables a, b, and c must have non-infinite lower and upper limits. Error checking is not provided.}

MATLAB.Execute('MATLAB Command') {Execute the MATLAB command. The command can be a string constant or a string variable. The command is executed in the MATLAB command window. The MATLAB command is case-sensitive. It must be entered exactly as it would be entered in the MATLAB Command Window. The result of the command is displayed in the macro log file. Note that MATLAB.Open command must be used before this command can be issued. Example: MATLAB.Execute('version').

MATLAB.Open {Start MATLAB and open the MATLAB Command Window.}

MATLAB.Quit {Close the instance of MATLAB opened with the MATLAB.Open macro command.}

MaximizeTable G a b c /Table='TableName' /Rows=1..10 /Method=Direct

{Maximize variable G with respect to variables a, b, and c for rows 1 through 10 in the Parametric table having name 'TableName'. Variables G, a, b, and c must exist in the specified Parametric table. TableName can be a string variable. The numerical values following the /Rows keyword can be EES variables. Note that variables a, b, and c must have non-infinite lower and upper limits. The VarInfo macro command can be used to provide limits. The optimization method specification is optional. If it is provided, it can appear can be appear anywhere after the Maximize keyword. The optimization method keyword following the equal (=) sign can be Golden or Quadratic for single degree of freedom optimization problems. For two or more degrees of freedom, the keyword can be Direct, Variable, Genetic, or Nelder. If no optimization method is provided, EES will use the method that was last used. If the /Rows keyword is not provided, the first and last rows last used in a Parametric table calculation will be assumed.

MessageDialog([OK, Cancel], 'message text') {MessageDialog will put up a user-provided message dialog with 1, 2 or 3 buttons. The button captions (which each should be less that 16 characters) are provided in brackets as the first parameter. For example, [OK, Cancel] will result in a message dialog with two buttons captioned OK and Cancel. The message text is the second parameter. The message text can reference a URL. For example, the message text could be 'Please visit http:\\fchart.com. Press the OK button when you return.' The URL name is hot so that clicking on it will start your browser and point it at the specified URL. Note that backslash \\ characters must be used because // is interpreted to be the start of a comment. The dialog will optionally return the button number (1, 2, or 3) if a variable name is provided in the following manner. R=MessageDialog([OK, Cancel], 'message text'). In this case R will be set to 1 if the OK button is selected and it will be set to 2 if the Cancel button is selected.}

Minimize G a,b,c {Minimize variable G with respect to variables a, b, and c. Note that the all variables must be defined in the Equations Window and variables a, b, and c must have non-infinite lower and upper limits. Error checking is not provided.}

MinimizeTable G a b c /Table='TableName' /Rows=1..10 /Method=Genetic {Minimize variable G with respect to variables a, b, and c for rows 1 through 10 in the Parametric table having name 'TableName'. Variables G, a, b, and c must exist in the specified Parametric table. TableName can be a string variable. The numerical values following the /Rows keyword can be EES variables. Note that variables a, b, and c must have non-infinite lower and upper limits. The VarInfo macro command can be used to provide limits. The optimization method specification is optional. If it is provided, it can appear can be appear anywhere after the Maximize keyword. The optimization method keyword following the equal (=) sign can be Golden or Quadratic for single degree of freedom optimization problems. For two or more degrees of freedom, the keyword can be Direct, Variable, Genetic, or Nelder. If no optimization method is provided, EES will use the method that was last used. If the /Rows keyword is not provided, the first and last rows last used in a Parametric table calculation will be assumed.}

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. The Lookup table name can be a previously defined string variable, e.g., L$}

NewContourPlot Table=PAR1 X=x Y=y Z=z TYPE=Isometric LEGEND=YES RESOLUTION=25 XMIN=0 XMAX=1 XINT=0.1 ZINT=0.2
{creates a new contour plot in which contours of variable identified by Z are displayed in one of three ways as a function of variables identified by X and Y. The data used to construct the plot are taken from the specified table. The table can either be identified by its name within quotes or by specifying the identifier for the table (PAR, LOOK, ARR, INT, followed by the position of the table. For example, PAR1 designates the first parametric table. ARR designates the arrays table. The three types of plots are identified by keywords: Isometric, Colorbands, and Gradient. The Legend and Resolution are optional. If Legend is provided, it must be followed by =Yes or =No. Resolution must be followed (without spaces) by a equal sign and a number that is between 0 and 100. A default value is used if no resolution is provided. The limits and interval of the X, Y and Z plotting variables can optionally be specified, as shown in the above example. Note that only plots based on 3 columns of data are currently supported.}

NewPlot Name='My Plot' Table='myTable' X=P2 Y=T2 Rows=1..10 Line=1 Symbol=1 SymbolSize=6 Color=Red Legend ErrorBars
{Create a new plot window. Set the name of the tab for this plot window to ' My Plot'. Note the the plot tab name must be either a string constant within single quotes or a predefined EES string variable. Plot variables T2 vs P2 using the first 10 rows of data from the Parametric table or Lookup table that has the name 'myTable'. Note that the table name

can be provided as a string variable, e.g., TN$. (If 'myTable' is the name of both a Parametric table and a Lookup table, the data will be plotted from the Parametric table.) Use plot symbol 1 with a size of 6 pixels. Allowable sizes range from 4 to 18. Note that the table could also have been identified by entering PARN or LOOKN (without the single quotes) where N is the tab number of the Parametric or Lookup table. To plot other tables, use ARR for the Arrays table or INT for the Integral table (without quotes). (When plotting variables from the Arrays table (ARR), specify the array name as XXX[i], where XXX is the parent name of the array variable.) The Legend keyword causes a legend item to be created. The Errorbars keyword will draw the plot lines iwth error bars if the data being plotted contain uncertainty information. All parameters except Table, X, and Y are optional. A default value will be used if a parameter is not specified. Note that the Line, Symbol, SymbolSize and Color can be specified using integers or variables. The coding for these parameters is as follows:

Line=0 No line	Symbol=1 Open circle	Color=0 Black
Line=1 Solid line	Symbol=2 Open rectangle	Color=1 Blue
Line=2 Dotted line	Symbol=3 Open triangle (up)	Color=2 Red
Line=3 Thick line	Symbol=4 Open butterfly (horiz)	Color=3 Green
Line=4 Dashed line	Symbol=5 Open triangle (down)	Color=4 Purple
Line=5 Thick dotted Line	Symbol=6 Open butterfly (vert)	Color=5 Maroon
Line=6 Thick dashed line	Symbol=7 Open diamond	Color=6 Yellow
Line=7 Dash-dot line	Symbol=8 Closed circle	Color=7 Lime
Line=8 Thick dash-dot line	Symbol=9 Closed rectangle	Color=8 Aqua
	Symbol=10 Closed triangle (up)	Color=9 Gray
	Symbol=11 Closed butterfly (horiz)	Color=10 Fuchsia
	Symbol=12 Closed triangle (down)	Color=11 Navy
	Symbol=13 Closed butterly (vert)	Color=12 Teal
	Symbol=14 Closed diamond	Color=13 Silver
	Symbol=15 Cross	Color=14 Orange
		Color=15 White

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. The Parametric table name can be a previously defined string variable, e.g., T$}

OnError GOTO XX command provides a way to modify the default behavior after an error occurs while executing commands in a Macro file. After this command has been executed, an error will result in a jump to the macro command having label XX. ONERROR GOTO 0 returns the default behavior.

Open C:\EES32\Examples\Regen.ees {open the specified file}

OpenLookup C:\EES32\myTable.lkt {open the specified Lookup file} If ? is provided in place of a file name, a select file dialog will appear in which the file name can be chosen. The filename extension can follow the ?, e.g., ?.csv and then the select file dialog will only display files having this file name extension. The table having this file name will then be opened. A string variable can be provided after the ?. If provided, the string variable is set to be the name of the Lookup file that has been read in. This name is valid for the remainder of the macro file execution.
Example: OpenLookup ?.csv TableName$ /FORMAT myFormat.fmt

OverlayPlot Name='myPlot' Table='myTable' X=P[4] Y=Eff Rows=1..10 Line=1 Symbol=4 SymbolSize=10 Color=Blue Right Legend //overlay a plot on existing plot window with tab name 'myPlot' of Eff vs. P[4] from data in rows 1..10 of the Parametric table named myTable. Use the right Y-scale. Use plot symbol 4 with a size of 10 pixels. Allowable sizes range from 4 to 18. To plot other tables, use LookN for Tab N of the Lookup table, Arr for Arrays table or Int for Integral table. The Legend keyword causes a legend item to be created. The Errorbars keyword will draw the plot lines iwth error bars if the data being plotted contain uncertainty information.

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' /A R2 C3 {Paste the contents of the clipboard into the Lookup table named 'Lookup 1' starting at row 2 column 3. The /A is optional. If present, the paste operation will add more rows to the table if needed. }

Paste Parametric 'Table 1' /A R2 C3 {Paste the contents of the clipboard into the Parametric table named 'Table 1' starting at row 2 column 3. The /A is optional. If present, the paste operation will add more rows to the table if needed. }

Pause 2.5 {Pause execution for a specified number of seconds. This command is useful when the EES macro file is being cycled and EES is to periodically read a file that has been written to a disk by another application, such as a data acquisition program.}

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}

PropPlot AirH2O PSY 6 10 15 20 25 30 35 4 0.8 0.85 0.9 095 P=101.3
{Create a Psychrometric chart with 6 wetbulb lines at values of 10, 15, 20, 35, 30, 35 C and 4 specific volume lines of 0.8, 0.85, 0.90, and 0.95 m3/kg at a total pressure of 101 kPa. Note that variable defined in the EES program or in the macro can be used in place of the numerical constants to specify the values of the constant properties.}

Quit {Close the EES application}

Rename 'FileName1' 'FileName2' Rename the specified file. FileName1 and FileName2 can be string variables (ending with character $) or filenames enclosed in single quotes.

RenamePlot 'my Plot' 'your Plot' {rename the plot having tab name ‘my Plot’ to ‘your plot’}

Repeat ...... Until (X-Y>5) {The commands in the macro file between the Repeat and Until keywords (which must be on separate lines) are repeated in sequence until the expression with the parentheses is true. Each command appears on a separate line. The parentheses are optional. Note that any legal expression involving existing variables that are defined in the main section of the Equations window can be used in the expression. If the expression is never true, the macro instructions will be repeated indefinitely or until the Esc key or Macro Window stop button is pressed. Macro commands can be nested.

Reset a, b {Reset will reset the status of the list of variables that follow the Reset keyword (e.g., a and b) to unassigned. This command may be useful if a variable has previously been used in a macro command list and is now to be used in another manner in the same macro command list.}

ResetGuesses {Reset the guess values of all variables to their default values}

Run Program name and parameters {The Run command will start the indicated program with the given parameters, just as if this string were entered with the Run command in the Windows Start menu. For example, to start EXCEL enter:

```
Run EXCEL
```

You can start EXCEL and open the file C:\temp\TEST.xls with the following macro command:

```
Run c:\temp\text.xls
```

EES knows that the .xls filename extension is an EXCEL file. Using the Run command, it is possible to start external programs that read data exported by EES and then have EES import the data output by the program.}

Save {Save the current file with its existing file name}

SaveArrays myArrays.txt {Save the contents of the Arrays table to a file called myArrays.txt.

SaveLookup 'Lookup 1' myTable.txt /A /T /N /E /Q {Save the contents of the Lookup table having the tab name 'Lookup 1' to a file called mytable.txt. The /A /E /F /N /T and /Q are optional parameters as explained below under SaveTable.}

SavePlot 'my Plot' 'PlotFileName.jpg' {If a plot having a tab name of 'myPlot' exists, save the plot in a file having filename 'PlotFileName.jpg'. Both the plot tab name and the file name may be supplied as string constants (surrounded by single quotes) or as predefined string variables (that end with a $ character). Alternatively, the plot tab name may be indicated with an integer. Use 0 to indicate the foremost plot. The filename extension may be .jpg, .bmp, .tif or .emf. EES will save the plot in the designated format.}

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 /A /T /N /E /Q {save the contents of the table named 'Table 1' in the Parametric Table Window to an existing file called myTable.txt. The /A /E /F /N /T and /Q are optional parameters. These parameters are not applicable for .LKT files.

/A append to existing file.

/E save in EES format so that the file can be read directly by the EES Open Lookup Table command (applicable only for.TXT files)

/F Normally, results are exported for each run in the Parametric table. If the /F is present, only the results for the last run in the Parametric table will be exported. This option is ignored if the calculations do not involve the Parametric table.

/N include column name and unit information

/T transpose rows and columns (can not be used with /E option)

/Q do not surround strings with single quotes}

Serial.Close COM1 {Close serial port COM1. The port name, COM1 in this example, can be a string variable, e.g., C$, that has been previously set to the port name. If the specified serial port is not open, this command will do nothing.}

Serial.Open Port=COM1 Baud=9600 Parity=n Stop=1 BufferSize=2048 TimeOut=200 {Opens serial port COM1 for reading and/or writing. The port name, COM1 in this example, can be a string variable, e.g., C$, that has been previously set to the port name. The baud rate is the number of bits per second that data is transferred in serial communication. Parity and Stop specify framing bits for the asynchronous communication. The baud rate, parity, and stop bits must match the parameters of the device that EES is attempting to communicate with. The defaults for Baud, Parity, and Stop are 9600, n, and 1. If these parameters are not provided, the defaults are used. Buffersize is the number of bytes used for the internal buffers for reading and writing. The maximum buffer size is 8192. If butfersize is set to zero or if this parameter is not provided, the default buffersize of 1024 is used. TimeOut is the number of milleseconds that EES will wait before giving up while trying to read data from the serial port with the Serial.ReadChar, Serial.ReadString, Serial. ReadNumber or Serial.ReadtoLookup serial macro commands. Timeout can be specified with a number or with an EES variable that has previously been set to value. The choice for Timeout depends on the rate at which the serial device is sending data. The Pause macro can also be used within a Repeat-Until loop to control the timing. (See Using a Macro for Serial Port Communications)}

Serial.ReadChar COM1 H$ {Reads a single ASCII character from the specified serial port. The first parameter is the port name, COM1 in this example, and it can be a string variable, e.g., C$, that has been previously set to the port name. The specified port must have been previously opened with the Serial.Open command. An EES string variable, e.g., H$, must be provided. The string variable will be set to the character read from the serial port.}

Serial.ReadNumber COM1 V {Reads a numerical value (integer or real) from the specified serial port. The first parameter is the port name, COM1 in this example, and it can be a string variable, e.g., C$, that has been previously set to the port name. The specified port must have been previously opened with the Serial.Open command. An EES variable, e.g., V, must be provided. The value of the variable will be set to the numerical value read from the serial port. Note that the numerical value is provided to the serial port by the external program as a string of numerical characters terminated by CRLF, i.e., carriage return (ASCII13) and line feed (ASCII 10) characters. The ReadNumber macro command will continue to look for the number on the serial port until it reads a CRLF or the Timeout interval specified in the Serial.Open command is exceeded.}

Serial.ReadString COM1 S$ {Reads a string consisting a one or more ASCII characters from the specified serial port. The first parameter is the port name, COM1 in this example, and it can be a string variable, e.g., C$, that has been previously set to the port name. The specified port must have been previously opened with the Serial.Open command. An EES string variable, e.g., S$, must be provided. The string variable will be set to the character string read from the serial port. Note that string provided to the serial port by the external program must be terminated by CRLF, i.e., carriage return (ASCII13) and line feed (ASCII 10) characters. The ReadString macro command will continue to look for the string on the serial port until it reads a CRLF or the Timeout interval specified in the Serial.Open command is exceeded.}

Serial.ReadtoLookup COM1 'TableName' Row Col {Reads a numerical value from the specified serial port directly into a specified cell of a specified Lookup table. The first parameter is the port name, COM1 in this example, and it can be a string variable, e.g., C$, that has been previously set to the port name. The specified port must have been previously opened with the Serial.Open command. The second parameter, TableName, is the name of the Lookup table that appears on the tab of the Lookup table. TableName can be either a string constant or string variable. The Lookup table must exist. It can be created using the New Lookup Table menu or the NewLookup macro command. Row and Col are respectively, the row and column in which the numerical value will be placed after the read operation is completed. Row and Col can be provided as numerical values or as EES variables that have been previously set to numerical values. The column can also be indicated by providing the column name as a string constant. If Row is greater than the number of rows in the table, the number of rows is increased. However, this operation takes time and may result in lost data. It is better to use the InserLookupRows macro command to ensure that there are sufficient rows in the designated lookup table. If Col is greater than the number of columns in the table, the value will be ignored. Note that the numerical value is provided to the serial port by the external program as a string of numerical characters forming a single numerical value terminated by CRLF, i.e., carriage return (ASCII13) and line feed (ASCII 10) characters. The ReadtoLookup macro command will continue to look for the number on the serial port until it reads a CRLF or the Timeout interval specified in the Serial.Open command is exceeded.}

Serial.WriteChar COM1 'H' {Writes one or more ASCII characters to the specified serial port. The first parameter is the port name, COM1 in this example, and it can be a string variable, e.g., C$, that has been previously set to the port name. The specified port must have been previously opened with the Serial.Open command. The second parameter is a string constant (enclosed within single quotes) or an EES string variable containing the character(s) that are to be written. Note that this command differs from the Serial.WriteString command in that CRLF (carriage return line feed characters) are not written to the serial port.}

Serial.WriteNumber COM1 V {Writes the ASCII characters representing a number to the specified serial port. The first parameter is the port name, COM1 in this example, and it can be a string variable, e.g., C$, that has been previously set to the port name. The specified port must have been previously opened with the Serial.Open command. The second parameter is either the numerical value or an EES variable, e.g., V, containing the numerical value that is to be written. CRLF (carriage return line feed characters) are appended to the number that is written to the serial port.}

Serial.WriteString COM1 S$ {Writes one or more ASCII characters to the specified serial port. The first parameter is the port name, COM1 in this example, and it can be a string variable, e.g., C$, that has been previously set to the port name. The specified port must have been previously opened with the Serial.Open command. The second parameter is a numerical value or an EES string containing the character(s) that are to be written. Note that this command differs from the Serial.WriteChar command in that CRLF (carriage return line feed characters) are appended to the characters that are written to the serial port.}

SetDirectory(F$) {SetDirectory(F$) will set the current Windows directory to the string constant or string variable within the parentheses.}

ShowWindow Arrays //Bring the Arrays table the front

ShowWindow Diagram //Bring the Diagram window to the front

ShowWindow EES //If EES was minimized, show it in normal mode.

ShowWindow Equations //Bring the Equations window to the front

ShowWindow Format //Bring the Formatted Equations window to the front

ShowWindow Integrals //Bring the Integrals table the front

ShowWindow Lookup 'Table Name' //Bring the Lookuptable to the front. The name of the Lookup table (surrounded with single quotes) can be optionally provided in which case the table having this name on its tab will be selected.

ShowWindow Parametric 'Table Name' //Bring the Parametric table to the front. The name of the Parametric table (surrounded with single quotes) can be optionally provided in which case the table having this name on its tab will be selected.

ShowWindow Plot 'Plot Name' //Bring the plot to the front. The name of the plot (surrounded with single quotes) can be optionally provided in which case the plot window having this name on its tab will be selected.

ShowWindow Progress {Normally, a series of EES commands initiated from a macro command list will not show the Progress window. However, if the macro has been started from the Calculate button in the Diagram window, it may be desirable to show the Progress window to reassure the user that the calculations are proceeding in a long calculation. Including this macro command will ensure that the Progress window is visible.}

ShowWindow Residuals {Bring the Residuals window to the front}

ShowWindow Solution {ring the Solutions window to the front}

ShowWindow Solution Key {ring the Solutions window to the front and show the Key Variables window if it exists.}

Solve {Solve as if the Solve command has been issued}

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. The start and stop row numbers may be specified with variables that have been previously defined, e.g., SolveTable 'Table 1' Rows=Start..Stop}

Stop (Execution of the Stop macro command will immediately stop the macro.}

StopCrit It=100 Time=3600 Res= 1.0E-0006 Var= 1.0E-0006
{Set the Stop Criteria properties as indicated}

Uncertainty h, Q, T_s=A1, Vel=R0.01 {Initiate the Uncertainty Propagation calculations to determine the uncertainty in variables h and Q as a result of uncertainties in variables T_s and Vel. The uncertainty in variable T_s is 1 on an absolute (A) basis. The uncertainty in variable Vel is 0.01 of its current value (relative basis).}

UncertaintyTable /S /U /R 'Table 1' Rows=1..10 h, Q, T_s=A1, Vel=R0.01 T_inf=Au_Tinf {Initiate the UncertaintyTable calculations to determine the uncertainty in variables h and Q as a result of uncertainties in variables T_s, Vel, and T_inf. Measured variables (here T_s, Vel and T_inf) are identified with =A or =R following their names. A indicates absolute uncertainty and R indicates relative uncertainty. The absolute or relative uncertainty is indicated with a numerical value or with a variable name following the A or R. Note that if a variable is name is indicated, the variable must be set to a value in the Equations window or in the Parametric table. Variables without an = following their name are assumed to be calculated values. All of the variables should be in the specified Parametric Table. If /U is provided, the guess values will be updated after each row of the table sucessfully completes calculations. If /R is provided the calculations will be start on the last row and proceed to the first row of the specified range. If /S is provided the calculations will stop if an error is encountered and remaining rows of the table will not be processed.}

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.}

Note that any of the commands can be sent directly from an external program to EES with the application's DDEExecute command. For example, the following command issued from EXCEL Visual Basic will cause EES to execute its SOLVETABLE command for Table 1. Note that when macro commands must be enclosed in square brackets.

```
Application.DDEExecute ChannelNumber, "[SOLVETABLE 'TABLE 1', Rows=1..10]"
```

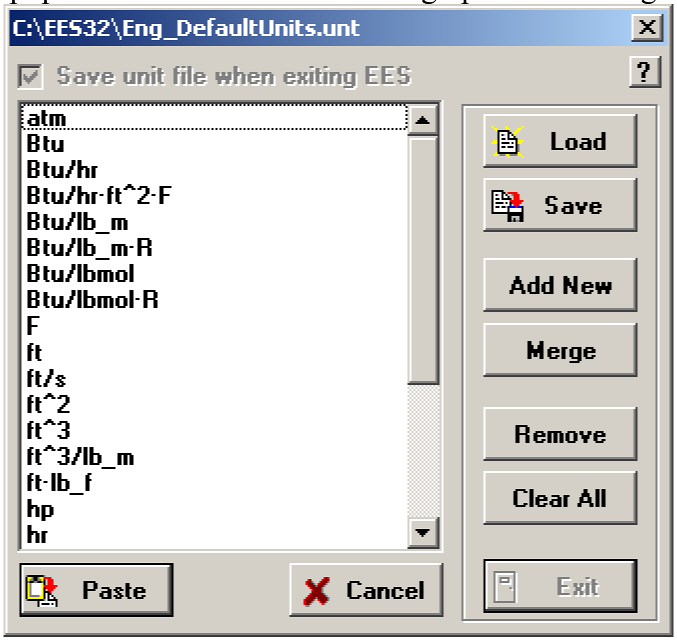
Unit Lists

EES can check the unit consistency of each equation in the Equations Window. This is a very important capability since experience has proven that unit errors are a very common problem both for students and practicing engineers. However, in order for EES to check unit consistency, the units of each variable used in the equations must be specified. There are several ways to enter unit information. Units can be entered for constants in the Equations window, as in the following example:

L = 15 [m]

Units for each variable can also be entered in the Variable Information dialog and by right clicking on a variable in the Solutions window or Parametric table. In each case, it is necessary to enter the unit designation from the keyboard, and this chore can be cumbersome. The Units List eliminates the need to key in the units.

The Units List can be accessed from the Equations window, the Variable Information dialog, the Parametric table and the Solutions window by clicking the right-mouse button and selecting Units List from the popup menu. This action will bring up the following dialog window.



C:\EES32\Eng_DefaultUnits.unt

☑ Save unit file when exiting EES

atm
Btu
Btu/hr
Btu/hr-ft^2-F
Btu/lb_m
Btu/lb_m-R
Btu/lbmol
Btu/lbmol-R
F
ft
ft/s
ft^2
ft^3
ft^3/lb_m
ft-lb_f
hp
hr

The title bar shows the name of the units list file. EES provides two default lists named SI_DefaultUnits.unt and ENG_DefaultUnits.unt that hold common units for the SI and English system, respectively. The units are shown in the list. Clicking on the desired unit and then clicking the Paste button will paste that unit string into the window that the Units List was accessed from, eliminating the need to enter the units. If you do not wish to have the pop-up list appear, move the SI_DefaultUnits.unt and ENG_DefaultUnits.unt out of the EES folder. Note that double-clicking on a units string in the list has the same effect as clicking the Paste button. If the units list was accessed from the Equations window, the unit string will be enclosed within quotes and square braces before it is pasted into the Equations window.

The name of the current units list file is shown in the title bar of the units list dialog. Note that the filename extension is .UNT, but the units file is a simple text (.txt) file that contains the desired unit strings. It is possible to develop customized units lists in separate .UNT files. The buttons located on the panel at the right of the dialog facilitate this process.

Load brings up a file open dialog from which a previously stored units list file (.UNT or .TXT) can be selected. The unit strings in this file will then be displayed in the units list.

Save brings up a file save dialog that saves the current units list in a specified .UNT file. Note that EES will automatically read the SI_DefaultUnits.UNT or the ENG_DefaultUnits.UNT file, depending on the current unit system setting, if this file is found in the EES directory or in the default file folder specified in the directories tab of the Preferences dialog.

Add New allows a new unit string to be added to the current list. Note that the information in this list is case insensitive. EES will not add a units string if it already exists.

Merge will add all of the units strings in the currently open EES file to the existing units list file.

Remove will remove the selected unit string from the units list.

Clear All will remove all units strings from the units list.

If any changes are made to the units list, the Exit button will be enabled. Clicking the Exit button will close the dialog, keeping any changes that have been made, but without pasting any information into a window. This behavior differs from the Cancel button in which the dialog is closed without saving any changes, and the Paste button in which the dialog is closed with the specified changes, but the selected units string is pasted into the window that was open when the units dialog was opened.

Note that you can edit the .UNT file with a text editor if you wish. If the .UNT file is located in a write-protected directory, it will not be possible for EES to save any specified changes. An error condition will be displayed. This situation occurs when EES is run from a write-protected server. In this case, it will be necessary to Save the units file into a directory that provides write permission to the user. An ideal location is the default file folder specified in the directories tab of the Preferences dialog. A units file, other than the default file, can be automatically loaded with the $INCLUDE directive.

Creating an Application Library

EES provides application libraries for Heat Transfer and Fluid Flow (designed to accompany the book Heat Transfer by Nellis and Klein, Cambridge University Press,. 2009), Mechanical Design, and the Component Library. Information relating to these libraries is accessed with labelled buttons in the Function Information dialog window (Options menu). This dialog also provides access to information for a user-defined application library. The available application library functions appear in the Function Information dialog in a drop-down list to the right of the radio button. Each entry in this list is associated with a folder in the UserLib directory. The contents of this folder include:

- A table of contents (.TOC) file that provides information about the organization and calling protocol of the library code units.
- One or more library (.LIB) files that provide the Functions, Procedures, or Subprograms.
- One or more .HTM, .PDF, or .CHM files that provide help for the code in the library file.
- One or more bitmap (.bmp) files that provide the figures that are shown in the Function dialog.

The required information and format for an Application Library can be explained by viewing the contents of the EES\Userlib\Component Library folder and the Pumps.TOC file that is contained in that folder and is shown below. Line numbers have been added to simplify the following discussion. Note that each entry (i.e., each line number) in the table of contents file exists on a separate line. However, some word-wrapping and indentation has been used to display lines that are too long to fit on this page.

```
1:  Pumps
2:  -3
3:  Component Library Model
4:  by G.F. Nellis and S.A. Klein
5:  2015
6:
7:  >Gear Pumps
8:  Gear Pump 1|Pumps.lib|Components.chm@1510|BMPS\GearPump1.bmp|Call
     GearPump1_CL(F$, C, T_in, P_in, P_out, N, V_disp, K_leak, eta_o: m_dot, T_out,
     W_dot, eta_p, eta_v)
9:
10:  >Centrifugal Pumps
11:  Centrifugal Pump 1|Pumps.lib|Components.chm@1610|BMPS\CentrifugalPump1.bmp|
     Call CentrifugalPump1_CL( F$, C, T_in, P_in, m_dot, N, D, D_hub: P_out, T_out,
     W_dot, eta)
```

Line 1 of the .TOC file provides the name of the library file; this name will appear in the drop down list next to the radio button in the bottom right of the top panel. Line 2 provides reserved information. Currently this line should display -4 for a User-Defined library. (-1 is used for the Heat Transfer library, -2 is used for the Mechanical Design library and -3 is used for the Component library.) Lines 3-6 provided user-supplied information that is displayed to the right of the picture of the component. Line 6 can contain a URL web link and if it does, clicking on the link will start the default browser and open it to the specified page. Following these six lines is the information that is required to provide the visual information in the Function Information dialog. Blank lines are ignored. Lines that begin with > indicate a new subheading; the subheadings are the choices that are provided in the drop-down list to the right of the picture. For example, lines 7 and 10 are the subheading choices that appear in the drop-down list (Boiling, Condensation, and Pressure Drop).

Following each subheading line is a line that provides information for each code unit (i.e., Function, Procedure or Subprogram) in the group. The code unit is selected using the scroll bar that is located below the picture. The | character is used to separate the information items on each line. The required information on each of the separate lines corresponding to the code units is as follows

1. The name of the code unit (e.g., Centrifugal Pump 1; this is the name that is displayed above the picture).
2. The name of the library file that the code unit is located in (e.g., Pumps.lib). This library file must be contained in the same folder as the .TOC file.
3. The name of the help file that is used to document the code unit (e.g., Components.chm@1610). This file can be a .PDF, .HTM or .CHM file. If a .CHM file is used, the item context number may be included by following the filename with the @ symbol and the help file context number (e.g., @2010). The item context allows the help file to be opened to a particular page when the help file includes several pages.
4. The name of the bitmap file that holds a picture representative of the selected code unit (e.g., BMPS\CentrifugalPump1.bmp – note that the BMPS\ indicates that the bitmaps are contained in a subfolder named BMPS). The bitmap file has a maximum width of 212 pixels and a maximum height of 160 pixels. Bitmaps will be scaled with a constant aspect ratio if they are larger in either of these dimensions.
5. An example that appears in the Example text box at the bottom of the Function Information dialog for the selected code unit (e.g., Call CentrifugalPump1_CL(F$, C, T_in, P_in, m_dot, N, D, D_hub: P_out, T_out, W_dot, eta). This is the text that will be pasted into the Equations Window if the Paste button is selected.

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, the order in which the equations are solved, and a summary of the unit checking results. 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 helpful if you use 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 at startup.

7. 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. 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.

10. 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. If you are using the AutoSave option (setting in the Preferences dialog), you may also find the AutoSave temporary file which is a tilda followed by the file name.

11. 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.

12. 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.

13. 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.

14. If you are working in Complex Mode, use the $COMPLEX ON directive at the top of the Equations window. It is more convenient than changing the Complex Mode setting in the Preferences dialog.

15. 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.

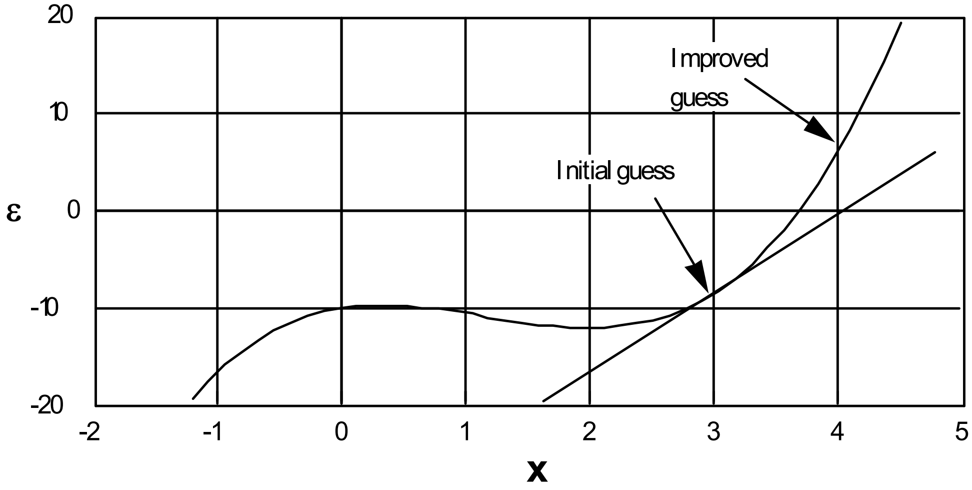


Figure 1: Residual of $x^3 - 3.5\ x^2 + 2\ x = 10$ as a function of x

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.

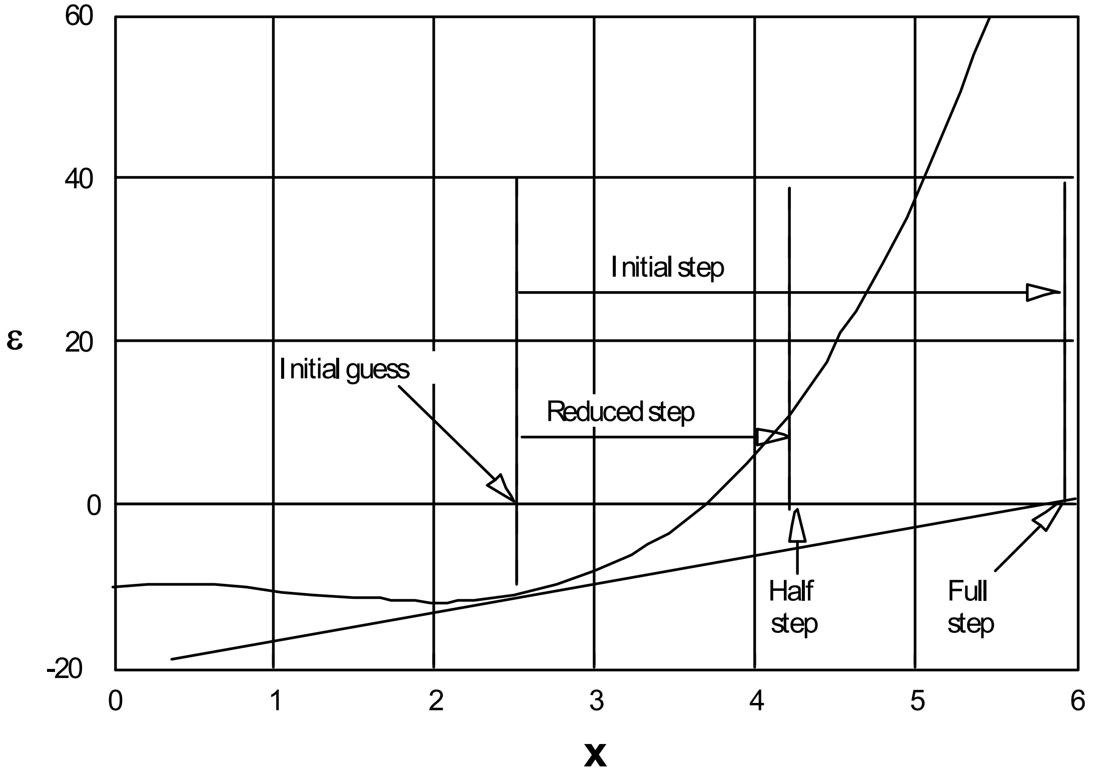


Figure 2: Use step-halving for improving the convergence

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:

$$x_1 + 2\ x_2 + 3\ x_3 = 11$$
$$5\ x_3 = 10$$
$$3\ x_2 + 2\ x_3 = 7$$

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:

$$\begin{array}{lllllllll}
 & & x_3 & & & & & + x_8 & = 11 \\
 & & & & & & x_7 & & = 7 \\
 & & & & x_5 & - x_6 & - x_7 & & = -8 \\
x_1 & & & + x_4 & & - x_6 & & & = -1 \\
 & x_2 & & & & & & + x_8 & = 10 \\
 & & x_3 & & - x_5 & & & + x_7 & = 5 \\
 & & & x_4 & & & & & = 4 \\
x_1 & & & & & + x_6 & + x_7 & & = 14
\end{array}$$

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: $x_3 = 3$

$x_3 - x_5 + x_7 = 5$ 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.

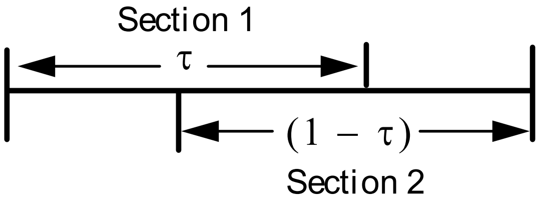


Figure 3: Region Elimination using the Golden Section Method

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.

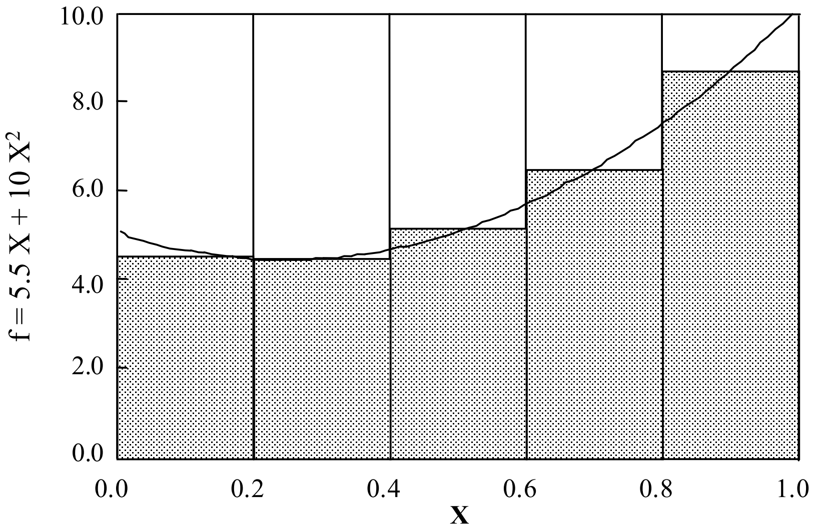


Figure 4: Numerical approximation of an integral

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)

$$P = \frac{R\,T}{v-b} + \frac{A_2 + B_2 T + C_2 e^{-\beta T/T_c}}{\left(v-b\right)^2} + \frac{A_3 + B_3 T + C_3 e^{-\beta T/T_c}}{\left(v-b\right)^3}$$
$$+ \frac{A_4 + B_4 T + C_4 e^{-\beta T/T_c}}{\left(v-b\right)^4} + \frac{A_5 + B_5 T + C_5 e^{-\beta T/T_c}}{\left(v-b\right)^5} + \frac{A_6 + B_6 T + C_6 e^{-\beta T/T_c}}{e^{\alpha v}(1 + C' e^{\alpha v})}$$

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).

Appendix D: The 64-Bit Professional Version

The Commercial and Professional versions of EES are 32-bit Windows executable programs that are designed to operate on 32-bit or 64-bit Windows operating systems, such as Windows XP, Windows Vista, and Windows 7, 8, and 10. The 64-bit Professional version is a different program that operates in exactly the same manner as the 32-bit version except that it can do calculations faster than the 32-bit version and it allows a maximum of 24,000 variables, compared to 12,000 variables in the 32-bit Professional version. Calculations speeds are about 2 times faster than the 32-bit version for computationally-intensive problems. This appendix provides information about the 64-bit version and summarizes some differences between the 32 and 64-bit versions.

- The 64-bit version (EES64.exe) requires a 64-bit Windows operating system, e.g, Windows 7/8/10 64 bit. It will not work on any 32-bit operating system.
- There is only a 64-bit Professional version. A Commercial license will not be able to run the 64-bit version.
- An extra-cost license (EES.DFT64) file is required to run the 64-bit version. This file must be purchased from http://fchart.com/ees/order.php.
- The 32-bit versions of EES do all floating point calculations using extended floating point numbers having 80 bits of numerical precision. The 64-bit version uses doubles, which are floating point numbers having 64 bits of precision. As a result, the 32-bit version provides greater precision and allows larger and smaller numerical values than the 64-bit version. The largest number that the 64-bit version can handle is approximately 1.7E308. The use of 64-bit doubles is the major reason that the 64-bit version calculates faster than the 32-bit version.
- Because numerical values are stored with different precision, the files saved by the 64-bit version necessarily differ from the 32-bit versions. The 64-bit version saves EES files with the .EES64 file name extension. EES64 files are identified by the file name extension and the yellow EES 64 icon. Double-clicking on an EES64 file will start the 64-bit version of EES. **The 32-bit version of EES cannot read or write .EES64 files.**
- Library files in the 64-bit version are saved with the .LIB64 file name extension. Similarly, EES Lookup files, external functions and procedures are identified with a 64 in the file name extension, e.g., .LKT64, .PRF64, .FDL64, .DLF64, .DLP64, and .DLL64. **None of these files can be read by the 32-bit version** due to the different format of the floating point numbers.
- The 64-bit version of EES can read and write any 32-bit or 64-bit EES or library file. However, it is necessary to convert the information in the 32-bit file to equivalent 64-bit information and this conversion process is slow. After reading a 32-bit file, it is best to save it as 64-bit file so that is will be read quickly the next time it is opened.
- It is possible for the 64-bit version to save EES and library files in a format that can be read by the 32-bit version. EES is programmed to save files in the alternative format if the file name extension is .EES or .LIB instead of .EES64 or .LIB64. The Open and Save As dialogs provide

a dropdown control to select various file types, including the older 32-bit format.

- The 64-bit version can not read external library files (.FDL, .DLP, .DLF, .DLL) that were created for the 32-bit version. It is necessary to use 64-bit versions of these files, which are identified as .FDL64, .DLP64, .DLF64, and .DLL64.