EESy Solutions

Engineering Equation Solver Newsletter ***No. 15, Spring 2005***

Welcome

EESy Solutions is a newsletter developed to provide news, tips, and tricks relating to Engineering Equation Solver. **EESy Solutions** is provided at no cost to all users of EES. Did you miss any of the previous issues? These newsletters and other useful information can be downloaded from our web site: www.fchart.com.

We Moved!

F-Chart Software has moved to a new location with more space. Our mailing and website addresses are as they were, but the telephone and fax numbers have changed. The new numbers are provided on the bottom of this page. *Make a note of it*.

Instant Update Service

All new licenses of EES are provided with one year of Instant Update Service. Instant Update Service is a valuable addition to EES, since it is updated frequently. (There have been 170 new versions released since the last EESy Solutions was composed in Spring, 2004.) The latest version is placed on our server so that subscribers can download a new version whenever they wish to do so. If your version of EES was purchased within the last twelve months, you can access the Instant Update server conveniently with the EES Instant Update menu item in the Help menu. The fee to continue Instant Update Service after the first year is 20% of current cost of the program per year. Contact F-Chart Software or use the web (fchart.com) if you wish to subscribe to Instant Update Service and take advantage of some of the new features described below.

Academic Site Licenses

Both Academic and Academic/Professional site licenses are available to educational institutions for a one-time fee of $1,000 and $3,750, respectively. See www.fchart.com or contact use for additional details.

What's New?

As in years past, literally hundreds of changes have made to EES during the past year. The capabilities of the program continue to grow. What follows is a short description of some of the more important new capabilities.

Unit Checking

Other than solving equations, unit checking is the most important function that EES provides. A number of improvements have been made to the internal algorithms. In addition, changes have been made to make the entering of units more convenient. For example:

1. Units for variables that on the left side of an equal sign and have not been previously specified will be automatically set to match the units of the right side of the equation. Variables with units that have been automatically set are shown in purple in the Solution window and the Variable Information dialog. A message is displayed to inform the user that the units shown in purple have been set by EES and should be checked.
2. Changing the units of any array variable changes the units of all units in the array to the new setting.
3. A control has been added to the Preferences dialog to turn the automatic unit setting off or on. The $AutoSetUnits On/Off directive provides the same capability.
4. In previous versions, EES would often suggest that an equation such as Q=UA*(T2-T1) has unit errors even if the units were correctly specified. The problem occurred because UA has units of [W/K] and T1 and T2 have units of [C]. One way to eliminate the error message was to specify the units of UA to be [W/C]. However, this is no longer necessary.

Plotting

Plotting is also a very important capability provided by EES. Our goal is to have EES provide publication-quality plots with little effort. Some of the improvements that have been made to the plotting capability are:

F-Chart Software **Phone: (608) 255-0842**
http://www.fChart.com **FAX: (608) 255-0841**

1. EES plots can be copied and pasted into other applications such as Word or PowerPoint. However, the size of the pasted plot differed from its size in EES. In fact, the pasted plot was often huge. Although the size can usually be changed in the other application, this was inconvenient. EES plots are now pasted at their specified size. The plots will display and print in the application with a resolution specified in the Preferences/Plot dialog. Note that the plots can be copied in black/white or color by changing the setting in this dialog.
2. A control has been added to the Modify Plot dialog to allow the plot size to be specified in inches or centimeters. The specified size is now the size of the entire plot, not just the size of the rectangular plot region.
3. 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 tool bar is visible, all text and graphic objects within this rectangle will be selected when the mouse button is released. If the plot window tool bar is not visible, 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.
4. When there is more than one plot line within a plot window, the plots are drawn in the order in which they were created. This order is displayed in the plot line list in the Modify Plot dialog. In some circumstances, the appearance of a plot may be improved by changing the order in which the plots are drawn. The order can be changed by selecting a plot line and dragging it to the desired location while holding the right mouse button down.
5. Five additional line styles have been added to the line tool in the Plot windows.

Diagram Window

The Diagram window was so named because it initially was developed to provide a place to display a diagram of the system that was being modeled. The capabilities of the Diagram Window have grown substantially. It now provides a complete graphical user interface (GUI) thtat interacts with the user for data input and output in one or more windows. Buttons can be placed on the Diagram window to initiate calculations, show plots, link to other programs, save data, or display a help file. The Diagram window also provides powerful animation capabilities, as described in EESy Solutions No. 14. You may wish to try some of the animation examples accessible from the Example menu. Additional capabilities that have been added to the Diagram windows are summarized below.

1. Check box and radio group objects can be placed in the Diagram window in the Professional version. These controls allow a set of EES equations to be associated with the check box or radio button selection. Changing the selection changes the equations that EES solves. In effect, changing the check box or radio button setting deletes specified equations from the Equations window and inserts others in its place. The display in the Diagram window can be dynamically changed by changing the values of EES variables that represent the characteristics of graphical or text items. In this way, graphical items, inputs, outputs or buttons can be made visible or hidden by changing the setting of a check box or radio button.
2. All items in the Diagram Window, e.g., text items (e.g., input or output fields), graphical items (e.g., lines and rectangles) and all button types can be given a name in the Professional version. EES uses the name to create variables, Name.left, Name.top, Name.hide, Name.width, Name.height, Name.color, and Name.angle. These variables can be set in the Equations window or in a Checkbox, Radio Group or Drop-down list on the Diagram window to dynamically control the attributes any item in the Diagram Window.

Macro Commands

A macro is a set of instructions to the EES processor. Macros are stored in text files with a .emf filename extension. Nearly all of the commands that can be entered from EES menus can also be entered as macro commands. For example, it is possible to set up a macro to open a specified EES file, solve the equations, construct a plot, open a Word document, copy the plot to the Word document, and then close both EES and Word. The macro can be initiated from the play button within the Macro window in EES or by starting EES from the command line with the macro file name provided as a parameter. This latter method of running a macro is convenient when EES is to be called from another program. Several major new capabilities for macros are:

1. EES variables can be assigned a value in a macro file. For example, the Macro command, X$='GUESS' sets the string variable X$ just as it would if this statement appeared in the Equations window.
2. Macro files now supports a Repeat ... Until(test) construct. The macro commands between the Repeat and Until keywords (on separate lines) are repeated until the test is true. The test can involve EES variables. It can be used to automatically find the periodic steady state solution by reading and writing EES variables with the IMPORT and EXPORT commands.
3. IMPORT and EXPORT commands have been added to the list of macro commands. These commands allow the values of specified variables to be written from/to the Clipboard or a file. Using these commands, it is possible to interface EES to a data acquisition program that outputs data to a file or to the Clipboard at specified time intervals. EES can repeat its calculations with each new set of data.
4. Additional controls have been added to he Macro window. In addition to the Play and Help buttons that were previously available, buttons have been added to Pause and Stop a macro, to Save the macro file and to lock/unlock the macro window.

Property Data

A distinguishing feature of the EES program is its extensive library of fluid property data that is coupled with its equation solving capability. Property data for a number of fluids (e.g, ammonia, helium, psychrometrics) have been improved. New property data have been added for Krypton and for the following high-temperature heat transfer fluids: Dowtherm Q, Dowtherm RP, Hitec XL, Salt (60% NaNO3, 40% KNO3), Syltherm 800, Therminol 59, Therminol 66, Therminol VP1, Therminol XP, and Xceltherm 600.

Debugging with the $Trace directive

The $TRACE directive allows the values of selected variables to be recorded during the process in which the equations are iteratively solved. Examination of the results in the Trace table may help identify an error in the equations.

Residuals Window

The Residuals window has been revised to allow improved display speed. Text in the Residuals window can be selected and copied to the clipboard. A status bar provides the values of variables that in the equation under the cursor.

Curve Fitting

A built-in procedure called CurveFit1D has been implemented to provide linear regression. CurveFit1D will return the curve fit parameters, RMS, Bias, R2, and standard error of each parameter fitted to linear, polynomial, power, exponential or, logarithmic forms. It is not necessary to first plot the data.

Saving Tabular Information

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 data.

Conveniences

The Variable Information dialog now provides a control so that either the guess values or the current values of the variables can be displayed.

Supercript + and - formatting is available for variables names ending with |plus and |minus.

Tips and Tricks

The commercial version of EES allows 6,000 variables and the Professional version allows 12,000. Any practical problem using this many variables will likely use arrays. Array variables are defined with one or more array indices within square brackets at the end of the variable name. For example, X[2] and Y[1,2] are array variables. Array variables are, for the most part, just like any other variable. Each array variable has a guess value, lower and upper bounds, display format and units. However array variables differ from other variables in that they can be displayed in the Arrays table and they can be passed to Functions, Procedures, Modules, and Subprograms in a compact form. Passing arrays conveniently is the focus of this issue.

It is perhaps first necessary to review the differences between Functions, Procedures, Modules, and Subprograms. A Function is an independent set of equations in EES that is provided with one or more inputs and returns a single result. A Procedure is similar to a Function, but it can return more than one result. Functions and Procedures differ from all other EES code in that they are composed of assignment statements, rather than equalities. For example, the equation: X=X+1 is legal in a Function or Procedure, but not in the EES main program or in a Module or Subprogram. In a Function or Procedure, X=X+1 instructs EES to assign the value of variable X to its current value plus one. But X can never equal X+1. An advantage of the assignment statements used in Functions and Procedures is that they can be used with logic statements such as Repeat-Until and If-Then-Else.

Modules and Subprograms are similar to Procedures in the respect that they are provided with one or more inputs and they can return one or more results. However, they differ from Procedures in that they are composed of equalities rather than assignment statements. In effect, Modules and Subprograms are like small EES programs embedded within the main EES program. The difference between Modules and Subprograms is subtle. The Subprogram is easier to understand. It truly is an independent program that is solved when it is called with the inputs that are supplied to it. A Module is not called directly. Instead, the equations in a module are integrated with the equations in the main program and the entire set of equations is solved simultaneously. A Subprogram can be called from a Function or Procedure; a Module cannot.

EES allows array range notation to be used when passing variables to Functions, Procedures, Modules, and Subprograms. For example, consider the following function which finds the sum of the squares of 5 variables.

```
FUNCTION SumSqrs(X[1], X[3],X[3], X[4],X[5])
  N=5
  Sum=0
  i=0
  Repeat
     i=i+1
     Sum=Sum+X[i]^2
  Until (i>=N)
  SumSqrs=Sum
End
```

The function could be called by a statement in the main EES program of the form:

```
SS=SumSqrs(X[1], X[3],X[3], X[4],X[5])
```

This same statement can be written more simply using array range notation as:

```
SS=SumSqrs(X[1..5])
```

The corresponding Function statement could be written as:

```
FUNCTION SumSqrs(X[1..5])
```

We can generalize the program to operate with N variables instead of 5 in the following way.

```
FUNCTION SumSqrs(N,X[1..N])
  Sum=0
  i=0
  Repeat
    i=i+1
    Sum=Sum+X[i]^2
  Until (i>=N)
  SumSqrs=Sum
End
```

We can test the Function as follows:

```
N=10
duplicate i=1,N
    X[i]=i
end
g=sumsqrs(N,X[1..N])
```

It should work fine. However, the same method of passing a variable number of variables to a Module or Subprogram would not work. For example, one might expect the following Subprogram to calculate the sum of the square of its inputs, just as the Function did.

```
SUBPROGRAM SS(N,X[1..N]: SumSqrs)
   Duplicate i=1,N
      XS[i]=X[i]^2
   end
   SumSqrs=Sum(XS[1..N])
End
```

Here is a test for the Subprogram.

```
N=5
duplicate i=1,N
   X[i]=i
end
call SS(X[1..N]: SumSqrs)
```

Note the (optional) use of the colon to separate the inputs and outputs in the Call and Subprogram statements. But, if you try this program you will find that it does not work. The reason is clear once you understand how EES operates. EES attempts to compile all of the equations (in the Subprogram as well as the main program) before any calculations are initiated. However, when EES compiles the equations in the Subprogram, it does not know what the value of N is. As a result, the following equations between the Duplicate and End statements can not be compiled and the Call to the Subprogram fails. The Subprogram would work if it were written so that N is known during compilation, e.g.,

```
SUBPROGRAM SS(X[1..5]: SumSqrs)
   N=5
   duplicate i=1,N
      XS[i]=X[i]^2
   end
   SumSqrs=Sum(XS[1..N])
End
```

However, if the Subprogram were to be used with a different number of variables, it would be necessary to change the Subprogram and Call statements and also change the N=5 equation.

There is a simple way around this limitation and that is to use the $Constant directive. A $Constant directive allows a global constant to be defined. Constants are identified with a # character as the last character in the variable name. The $Constant directive can be placed at the top of the program before the Subprogram statement. The constant value is defined in this manner and is then accessible to all routines. Using the $Constant directive, the Subprogram and its test program would be written as follows:

```
$Constant N#=10
SUBPROGRAM SS(X[1..N#]: SumSqrs)
   Duplicate i=1,N#
      XS[i]=X[i]^2
   end
   SumSqrs=Sum(XS[1..N#])
End
duplicate i=1,N#
   x[i]=i
end
call SS(x[1..N#]:SumSqrs)
```

Now, only the value in the $Constant directive needs to be changed when it is necessary to change the number of variables.