Advanced Post Processing Techniques of Thermal Models using Microsoft Excel®

Hume Peabody
Thermal Modeling Solutions, LLC

ABSTRACT

As thermal models have become increasingly large and detailed, the process of data reduction to evaluate results has become more challenging. Analysts often write out specific results at run time that are expected to be important; however, not all important data is identifiable ahead of time. Once the output data is available, it is often imported into Microsoft Excel® or other spreadsheet tools for further evaluation. While most analysts are familiar with Excel, this paper seeks to demonstrate some of the more advanced functions available to further simplify data reduction. By outputting all data from the thermal model and using combinations of text manipulation and lookup functions, dynamic workbooks can be created that respond to user changes. Combinations of relatively few Excel functions allow a user to quickly look up data (heat loads, temperature, capacitances, etc.) for nodes or groups of nodes. A simple task using basic commands is introduced and further functionality is added through subsequent examples in the paper. Additional "tips and tricks" are also discussed, including Visual Basic macros and adding text information to images.

INTRODUCTION

While thermal analysis time has been reduced with newer and more efficient model generation tools and solution times have decreased with faster solvers, the effort to post process data has generally remained the same. However, if the output data has been imported into a spreadsheet, like Microsoft Excel®, then the task of finding, processing and displaying relevant data can be easily accomplished using the extensive library of functions like those found in Excel. The use of text manipulation, lookup, and calculation functions can quickly locate, compute, and display data. The INDIRECT command allows a user to build complicated formulas based on the inputs of other cells (Row Location, Column Location, Worksheet location, etc.). Lastly, additional functionality can be added by generating functions and subroutines using Visual Basic for Applications.

GENERAL EXCEL USAGE

Excel allows a user to enter formulas or text/numbers into various cells on worksheets. Formulas may reference the Excel library of functions and/or other cell locations. The formulas are then evaluated to find the value of the entered formula. Up to seven nested layers of functions may be included in a single formula.

References to cell addresses in formulas may contain the $ character. This anchors the row or column so that when the formula is copied or filled across/down, the column/row does not change. Therefore, placing a $ in front of the column letter will make the column stay fixed, while placing a $ in front of the row number will make the row stay fixed when the formula is copied. Use of the F4 function key with a range selected in the formula bar toggles between all combinations of anchoring (Row and Column, Row, Column, None).

TEXT MANIPULATION

The MID function returns a substring from the input text and has the following syntax: *MID(Text, StartIndex, Length)*. This function could be used to return the node number from a {*Submodel.NodeNumber*} syntax. However, often the starting location of the substring to be returned is not known explicitly. For this, the FIND and LEN functions are used. FIND returns the location in the string where a particular search string is found and has the following syntax: *FIND(FindText, WithinText, StartIndex)*. A *StartIndex* of 1 indicates to begin searching from the beginning of the text. The LEN function simply returns the length of the argument text and has the following syntax: *LEN(Text)*. Combining the LEN and FIND results allows a user to determine the number of characters to return from the MID function. The TEXT function can be used to output text into the desired format and has the syntax: *TEXT(text, format)*. Finally, the & character can be used to combine substrings into a single output string. The following examples assume that cell B3 contains the string "SPACE.1234".

=MID(B3, 7, 4) yields 1234
=MID(B3, 1, FIND(".",B3,1)-1) yields SPACE
=MID(B3, FIND(".",B3,1)+1, LEN(B3)-FIND(".", B3, 1)) yields 1234
=TEXT(MID(B3, 7, 4),"0.00E+00") yields 1.23E+03
="SPACE" & ".1234" yields SPACE.1234

CONDITIONAL and BOOLEAN OPERATIONS

IF is a simple condition check, with the following syntax: *IF(LogicalTest, Value If True, Value If False)*. ISERROR, ISTEXT, ISNUMBER, and ISBLANK are all used to determine if a particular cell or value returns an error or is text, numeric or blank respectively.

To combine *LogicalTests*, values may be ANDed, ORed, or NOTed. The syntax is as follows: *AND|OR|NOT(Condition1, Condition2, Condition3 …)* For AND, all conditions must be TRUE for a TRUE value to be returned; for OR, if any of the arguments are TRUE, then a TRUE value is returned. NOT simply inverts a FALSE to a TRUE and vice versa. Nesting these values allows a user to create complicated conditions to be tested including all three Boolean operations. Assuming Cell B3 contains the string "SPACE.12", the following formulas would yield the results listed.

=IF(MID(B3, 7, 2)="12", "Is12", "IsNot12") yields Is12
=IF(NOT(MID(B3, 1, 5)="SPACE")),"Not Space", "Is Space") yields Is Space
=IF(OR(MID(B3, 7 ,2)="12", MID(B3, 1, 5)="SPACE"), "Is12orSpace","IsNot12orSpace") yields Is12orSpace
=IF(AND(MID(B3, 7, 2)="13", MID(B3, 1, 5)="SPACE"), "IsSpace13","IsNotSpace13") yields IsNotSpace13

SUMIF

The SUMIF function searches for a specified value over a range of data and returns the sum of all values found from either the search range or a specified return range. The syntax is as follows: *SUMIF(SearchRange, SearchValue, ReturnRange)*. If a user enters a column of data for the *SearchRange* and a different column for the *ReturnRange*, then any values matching the *SearchValue* would return the value in the *ReturnRange* at the same row/column. If no *ReturnRange* is specified, then the summed values will be taken from the *SearchRange*. For unique data (i.e. thermal nodes), SUMIF will return only one value.

	A	B	C	D
1	Submodel	Node	T(1)	T(2)
2	SUB1	SUB1.1	25.2	26.0
3	SUB1	SUB1.2	26.7	27.0
4	SUB1	SUB1.3	31.8	31.8
5	SUB2	SUB2.1	29.6	29.7
6	SUB2	SUB2.2	50.2	52.0

Sample Data 1

=SUMIF($B$2:$B$6,"SUB1.2",$C$2:$C$6) yields 26.7
=SUMIF($B$2:$B$6,"SUB2.2",$D$2:$D$6) yields 52.0
=SUMIF($B$2:$B$6,"SUB3.2",$D$2:$D$6) yields 0.0

SUMIF can also be used to find the total heatload applied to a group of nodes.

	A	B	C	D
1	Submodel	Node	Q(1)	Q(2)
2	SUB1	SUB1.1	2.5	2.5
3	SUB1	SUB1.2	2.5	2.5
4	SUB1	SUB1.3	3.5	3.5
5	SUB2	SUB2.1	1.71	1.75
6	SUB2	SUB2.2	1.63	1.67

Sample Data 2

=SUMIF($A$2:$A$6,"SUB1",$C$2:$C$6) yields 8.5
=SUMIF($A$2:$A$6,"SUB2",$D$2:$D$6) yields 3.42
=SUMIF($A$2:$A$6,"SUB3",$D$2:$D$6) yields 0.0 (not found)

COUNTIF and COUNTBLANK

Similar to SUMIF is the COUNTIF function. However, instead of returning a sum, it returns a count of the number of SearchValues found. The syntax is *COUNTIF(SearchRange, SearchValue)*. This could be used to determine the number of nodes in a particular submodel. It is also very useful in conjunction with the SUMIF command for finding the average of a group of nodes. If the search values are numeric, then a user may also determine the number of entries that are greater or less than a specified value. This could be used to find the number of nodes whose margin or deviation was greater than a specified value. COUNTBLANK determines the number of blank cells in the specified range and has the following syntax: *COUNTBLANK(SearchRange)*.

	A	B	C	D
1	Submodel	Node	T(1)	T(2)
2	SUB1	SUB1.1	25.2	26.0
3	SUB1	SUB1.2	26.7	27.0
4	SUB1	SUB1.3	31.8	31.8
5	SUB2	SUB2.1	29.6	29.7
6	SUB2	SUB2.2	50.2	52.0

Sample Data 3

=SUMIF($A$2:$A$6,"SUB1",$C$2:$C$6) yields 83.7
=COUNTIF($A$2:$A$6,"SUB1") yields 3
=COUNTIF($C$2:$C$6,">" & C5) yields 2
=SUMIF($A$2:$A$6,"SUB1",$C$2:$C$6)/COUNTIF($A$2:$A$6,"SUB1") yields 27.9

MATCH and INDEX

While the SUMIF function is useful for returning numeric values, it cannot return a non-numeric value. For this, the combination of MATCH and INDEX is best. MATCH searches for a specified value in a specified range and returns the *index* location of the first value found that matches the SearchValue. MATCH has the following syntax: *MATCH(SearchValue, SearchRange, 0)* with the 0 indicating to match SearchValue exactly. INDEX is used to return a value at a specified location in the specified range. Its syntax is as follows: *INDEX(ReturnRange, IndexLocation)*

While the VLOOKUP (or HLOOKUP) function may also be used, they impose a limitation in that the SearchRange must be before the return range. Therefore, they could not be used with the data in the order presented above. These commands are useful for returning text information associated with a particular node, such as telemetry mnemonic, description, submodel, etc.)

	A	B	C	D
1	Description	Node	T(1)	T(2)
2	Motor	SUB1.1	25.2	26.0
3	Shaft	SUB1.2	26.7	27.0
4	Mirror	SUB1.3	31.8	31.8
5	Housing	SUB2.1	29.6	29.7
6	Baseplate	SUB2.2	50.2	52.0

Sample Data 4

=MATCH("SUB1.2",$B$2:$B$6,0) yields 2
=INDEX($A$2:$A$6, MATCH("SUB1.2",$B$2:$B$6,0)) yields "Shaft"

LINEST and TREND

The LINEST and TREND function perform a least squares fit to a set of data. LINEST has the following syntax: *LINEST(known_ys, known_xs, const, stats)*. Typically, a user should enter TRUE for *const* and FALSE for *stats* to indicate that the y-intercept value is non-zero and that additional statistics are not needed. The returned result is an array with one more value than the number of independent variables. Therefore, to fit for a fourth order polynomial, the user should have columns of X, X^2, X^3, and X^4 and enter these four columns as the *known_xs*. The returned array will have five entries, each of which can be accessed using the INDEX function (described earlier). The first index is the coefficient for the curve fit for the first X variable, the second corresponds to the second X variable, and so on. The last index corresponds to the y-intercept.

TREND takes the LINEST capability one step further and returns the calculated value for the input *new_xs* specified by the user. The syntax is as follows: *TREND(known_ys, known_xs, new_xs, const)*. One caution with the TREND function is the need to enter a value for *each* independent *new_x* value. Therefore, to predict the Y value for a fourth order polynomial at an X value of 2, *new_x* values of 2, 4, 8, and 16 would need to be entered.

INDIRECT

Perhaps the most powerful EXCEL function is the INDIRECT function. This function allows a user to specify a range as a text string and then converts that string to a range to be used further. The syntax is as follows: *INDIRECT(CellAddress, TRUE)* for A1 reference style or *INDIRECT(CellAddress, FALSE)* for the Row-

Column (R1C1) reference style. Combining INDIRECT with the above commands allows a user to change Source Worksheets, Rows or Columns referenced in other formulas simply by changing cells on other sheets, as shown in the examples below.

Temp	A	B	C	D
1	Description	Node	T(1)	T(2)
2	Motor	SUB1.1	25.2	26.0
3	Shaft	SUB1.2	26.7	27.0
4	Mirror	SUB1.3	31.8	31.8
5	Housing	SUB2.1	29.6	29.7
6	Baseplate	SUB2.2	50.2	52.0

Heat	A	B	C	D
1	Description	Node	Q(1)	Q(2)
2	Motor	SUB1.1	2.5	2.5
3	Shaft	SUB1.2	2.5	2.5
4	Mirror	SUB1.3	3.5	3.5
5	Housing	SUB2.1	1.71	1.75
6	Baseplate	SUB2.2	1.63	1.67

Calc	A	B	C	D
1	Sheet	Heat		
2	Row	3		
3	Column	4		
4	Node	SUB1.3		
5	Heading	T(1)		

Sample Data 5

=INDIRECT("'Temp'!R" & B2 & "C" & B3,FALSE) yields 27.0
=INDIRECT("'Heat'!R" & B2 & "C" & B3,FALSE) yields 2.5
=INDIRECT(B1 & "!R" & B2 & "C" & B3,FALSE) yields 2.5
=MATCH(B4,'Temp'!A1:A65536) yields 4
=MATCH(B5,'Temp'!A1:IV1) yields 3
=INDIRECT("'" & B1 & "'!R" & MATCH(B4, 'Temp'! A1: A65536) & "C" & MATCH(B5, 'Temp'!A1:IV1) yields 3.5

Changing cell B1 to Temp, would yield 31.8 for the above formula. Changing cell B4 to SUB2.2 and cell B5 to T(2) would then yield 52.0. With this formula, a user has complete control over the worksheet, row and column from which to retrieve data.

Further power can be realized in that the INDIRECT command may be used within a SUMIF or other functions. The following formula would allow a user to retrieve data for the SUB2.1 node from the worksheet specified in cell B1.

=SUMIF(INDIRECT("'" & B1 & "'!R1C2:R6C2",FALSE), "SUB2.1", ("'" & B1 & "'!R1C3:R6C3",FALSE))

VBA – Visual Basic for Applications

If a user needs functionality not already included in the base set of Excel functions, Visual Basic macros can be written to extend Excel's capabilities. VBA is an extension of the Visual Basic programming language which includes collections of objects specific to each application. For Excel, the primary objects available for use are Worksheets, Cells, Ranges, Charts, and Series.

To access the data in Cell B1 on Worksheet Temp, a user could use either of the following syntaxes:

Worksheets("Temp").Range("B1")
Worksheets("Temp").Cells(1,2)

To access chart information, the syntax is:

Charts(Name or Index).SeriesCollection(SeriesIndex).

To retrieve the marker style for the second point on the first series of Chart1, the syntax is:

Charts("Chart1").SeriesCollection(1).Points(2).MarkerStyle

Much of the syntax to perform Excel specific actions can be found by recording a macro and then examining the resulting VBA code in the Visual Basic Editor. This capability can be further extended by using the control/conditional structures in VBA. VBA includes the following variable types: variant, integer, long, string, single, double, boolean and uses a syntax:

```
Dim <Variable Name> As <Variable Type>
```

For example, the following lines define I as an integer type and InputStr as a string

```
Dim I As Integer
Dim InputStr As String
```

VBA also includes basic looping and conditional statements, with the most common ones listed below:

DO-WHILE

```
DO WHILE <condition>
  <code here>
LOOP
```

FOR-NEXT

```
FOR <variable> = <StartValue> to
          <EndValue> Step <Increment>
  <code here>
NEXT <variable>
```

IF-THEN-ELSEIF-ELSE

```
IF <condition> THEN
  <code here>
ELSEIF <condition> THEN
  <code here>
ELSE
  <code here>
ENDIF
```

Some examples of useful functions include a weighted average function or a function to find the minimum or maximum over a range of data where particular criteria must be satisfied (e.g. submodel), like SUMIF or COUNTIF. Some examples of useful subroutines include making all charts have the same axes properties and titles, making charts have markers every 10 data points, or deleting all blank lines on a worksheet. Using combinations of Excel objects properties and methods and Visual Basic syntax and capabilities, a user may extend the functionality of Excel nearly endlessly.

MISCELLANEOUS CAPABILITIES

Additional capabilities are also available but are not implemented as functions in Excel to be used in formulas. Examples include conditional formatting, image and text objects that may be added to worksheets, and iterative solvers. Conditional formatting allows the format of the cell to be changed based on the *value* of the cell. The general approach is to apply formatting if the cell value is less than (greater than, equal to, between, etc) the value of one or more values. This capability can also be extended by using formulas for the values instead of simple cell references (e.g. "=IF(ValueFound,10, -100)")

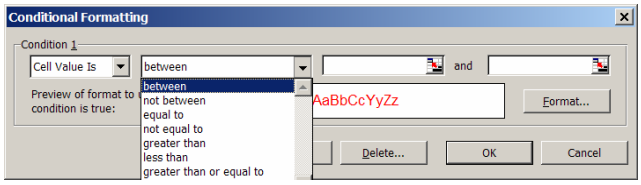

Figure 1 – Conditional Formatting

Excel also allows a user to add text boxes to images. While most users are aware of this, they may not be aware that the text box may reference a cell location and not necessarily a fixed text value. To enable this, the user must enter the Cell location preceeded by an equals sign in the formula bar (e.g. =A1), **NOT THE TEXT BOX ITSELF**. Since the values in the referenced cells may be updated based on other cells, this allows a user to change the values displayed on the image, by changing the controlling cell. For example, a user may add numerous textboxes representing the temperatures of various locations over an image of a spacecraft component. These text boxes may reference cells A2, A3, A4, etc. Furthermore, Cells A2, A3, A4, etc may include conditional IF statements in their formulas to retrieve one set of data for Cold and a different set of data for Hot depending on the value of Cell A1. Therefore, by changing the value in Cell A1, the text displayed over the image may also be changed.

Another useful capability in Excel is the ability to perform iterations to solve a linear or non-linear equation or set of equations using: Goal Seek and Solver. The Goal Seek is available in a default installation; however the Solver must be installed as add-in to access its functionality. Goal Seek is best used for relatively simple, one independent variable equations. The user enters in the Cell containing the dependent variable (Set Cell), the goal value (To), and the cell containing the independent variable (By Changing Cell). Goal Seek then attempts to find the value of the independent variable that most closely produces the specified value for the dependent variable. Clearly, the formulas to determine the dependent variable value must reference the independent variable value.

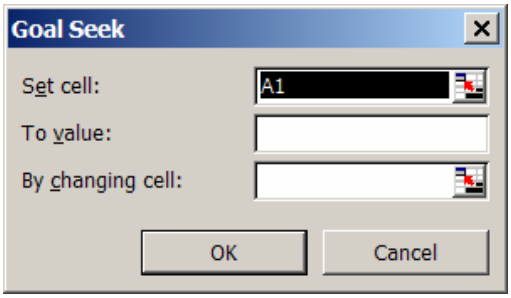

Figure 2 – Goal Seek

The Solver extends this capability, allowing the user to specify multiple independent variables as well as constraints for each if needed.

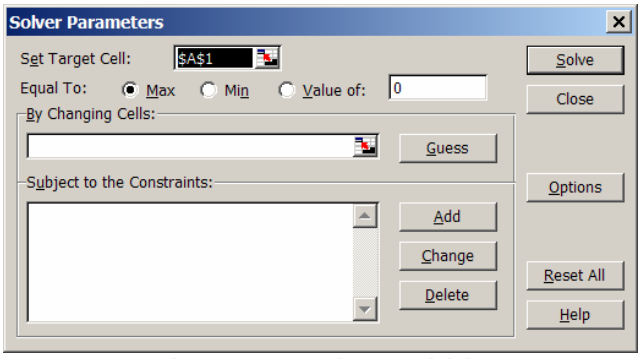

Figure 3 – Solver Add-in

CONCLUSIONS

A number of Excel functions can allow a user to quickly parse and retrieve information from data on Worksheets. The functions discussed include:

- MID, LEN, FIND, TEXT, &
- IF, ISERROR, ISBLANK, ISTEXT, ISNUMBER
- AND, OR, NOT
- SUMIF, COUNTIF, COUNTBLANK
- MATCH, INDEX
- LINEST, TREND
- INDIRECT

Use of the INDIRECT function can allow a user to assemble formulas from input values for Worksheet name, row and column. Functionality can be further extended in Excel using macros and the VBA programming language. Lastly, additional capabilities such as conditional formatting, text box linking, and Goal Seek and Solver capabilities provide even more methods for a user to accomplish their goals. If further information is desired on any of the functions presented herein, it is suggested to consult the online help for Excel.

CONTACT

Hume Peabody
Thermal Modeling Solutions, LLC
tarpthermal@comcast.net