8-12 July 2018, Albuquerque, New Mexico

Considerations when Building Thermal Models that Require Conversion between Formats

Hume L. Peabody[1]
NASA-GSFC, Greenbelt, MD 20771

and

Lisa M. Grob[2]
a. i. Solutions, Inc, Lanham, MD 20706

At times, it is inevitable to require conversion of thermal models from one software format to another. This most often occurs for missions with international partners where not all parties utilize the same software packages for thermal analysis. Mandating a single tool for all parties is one possible solution, but this approach can introduce problems if significant effort is required to overcome inexperience with the designated tool and may result in difficulty meeting analysis schedule requirements. Alternatively, allowing all parties to use their own familiar tools minimizes the impact to analysis schedules but does introduce the need to convert the models later to a common format for analysis at a higher level of assembly. External conversion tools and formats have been developed through the years to aid in this process, but have had limited success in fully converting models seamlessly. Having a basic familiarity with tool capabilities on both sides of the conversion process allows for models to be built in a manner to better facilitate conversion by avoiding features and capabilities which are unsupported by the destination tool or for which no workarounds exist. Also, the effort to convert a model is often neglected when developing the schedules for analysis at the higher assembly levels; delivery of models preconditioned for convertibility minimizes the schedule risk. This paper seeks to provide some guidance on modeling techniques to avoid when developing Geometrical Math Models (GMM) and Thermal Math Models (TMM) when conversion is required. The recommendations are based on GMM conversion experiences between TSS/Thermal Desktop/ESARAD and TMM conversions between SINDA-FLUINT/ESATAN.

Nomenclature

ACG	*= Automatic Conductance Generation*
ESA	*= European Space Agency*
ESATAN	*= European Space Agency Thermal Analyzer Network*
ESATAN-TMS	*= ESATAN - Thermal Modeling Suite*
FE	*= Finite Element*
FORTRAN	*= Formula Translator Language*
GMM	*= Geometrical Math Model*
NASA	*= National Aeronautics and Space Administration*
SINDA/FLUINT	*= Systems Integrated Numerical Difference Analyzer with Fluid Integrator*
TMM	*= Thermal Math Model*
TSS	*= Thermal Synthesizer System*

[1] Staff Thermal Engineer, Mail Stop 545, Goddard Space Flight Center, Greenbelt MD 20771.
[2] Thermal Systems Engineer, a. i. Solutions, Inc, 4500 Forbes Boulevard, Suite 300, Lanham, MD 20706.

I. Introduction

THERMAL models used by the aerospace industry are often based on a two part modeling effort. First, a Geometrical Math Model (GMM) simulates a design with a collection of surfaces to compute the geometric radiation interchange of a design with its surroundings, be it other components and instruments or celestial sources. These surfaces are assigned node numbers, coating properties, and radiatively active sides to produce mathematical relations between surfaces in the form of radiative conductors (Radks or GRs) between discrete nodes. Similarly, heat loads may also be applied on these nodes from sources of known intensity, such as solar or planetary heating. Analogous to an electrical network, these GMM generated conductors and sources are combined with other nodal sources/capacitances and conductors (both radiative and linear) that describe the full design in the Thermal Math Model (TMM). The full network is solved to generate temperature predictions for each node as well as heat loads where a control system may be adding heat to maintain temperatures above a design limit. In essence, regardless of what simulation code is being used, the underlying inputs and outputs are based on this. Differences between simulation codes most often come down to how much of the TMM is generated by information in the GMM and not necessarily differences in how they approach simulating the design.

Current thermal analysis codes have made great strides in the last 10 years or so to generate nearly all of the TMM for information contained in the GMM. This has led to the addition of features in the GMM that are not necessarily geometric in nature (such as nodal heat loads, thermostatically controlled heaters, user-defined conductances, bulk material properties, and thicknesses). These additional modeling parameters are used to generate more of the capacitances, conductive network (with possible temperature dependence), insulation resistances typically represented by an ε^*, time or temperature-dependent heat loads, and model solution control parameters (such as convergence or maximum allowable temperature changes over a timestep). However, in this area, the commercial codes have diverged with different subsets of features and capabilities supported. While the generated TMM lines of code may be similar, the methodologies used to generate them can be quite different.

Two codes that are commonly used include Thermal Desktop® (from Cullimore and Ring Technologies) and ESATAN-TMS (from ITP Aero). Thermal Desktop is an add-on to the AutoCAD program, which includes modules for the radiation calculations and for generating the corresponding TMM. The TMM itself is executed in SINDA/FLUINT, which is also maintained by the same vendor. Conductors generated internal to a subdivided surface are based on known shapes and closed- form solutions (i.e. simple kA/L or $2\pi Lk/\ln(r_o/r_i)$ relationships). Thermal Desktop users often model with either edge nodes or finite elements to handle conduction between surfaces through a shared edge with identical nodes. Furthermore, contactors may be used to model conduction between two faces or along two edges that do not share a common nodal boundary. The ESATAN-TMS suite supports advanced cutting capabilities and Boolean geometry and also includes modules for radiation calculations and generating the lines of code to be executed by the ESATAN solver. It does have some finite element support, but renumbers and merges nodes automatically to prevent free edges. Furthermore, ESATAN-TMS uses its Automatic Conductor Generation (ACG) capability using the Far Field method to create the conductors between surfaces at the identified interfaces.

Occasionally, it is necessary to deliver models in one of these formats that is not familiar to the thermal analyst. This occurs most often on projects with international partnerships and can involve deliveries of instrument models to a spacecraft contractor or deliveries of an observatory model to a launch contractor. Conversion of these models can take considerable effort and both parties should make efforts to be as accommodating as possible to get the best product available without having to expend more resources than necessary. This paper seeks to provide some recommendations on features and capabilities to avoid, or whose use should be minimized, to facilitate a smoother and timely conversion and model delivery.

II. Conversion Process

For thermal model conversion, both the GMM and TMM models need to be converted separately while preserving the minimum definitions to replicate results. It is nearly impossible to convert the full GMM and retain all the functionality to the destination code since features and capabilities differ. At a minimum for the GMM, the following aspects must be converted: surface size/orientation/location, optical properties, radiative activity, and nodal assignments for use in the radiation calculations. For the TMM, the node, conductor, array, and constant data (i.e. variable definitions) blocks need conversion as well as the user logic blocks that affect the solution. User logic for specialized output (beyond the vendor supplied output routines) generally does not need conversion as it does not impact the solution results.

Some commercial codes do include import and export capabilities, but maintaining these capabilities is often challenging due to the competitive nature of commercial vendors and an unwillingness to share competition-

sensitive capabilities and details. Furthermore, the developers often consider supporting competitors' codes to be less important than developing new capabilities and features of their own code. In essence, software developers would rather their code be used than their competitors code, and so development is focused on staying ahead of the competition rather than supporting portability between their code and the competition.

However, organizations that are primarily end users of the codes have spent resources to develop utility programs to aid in the conversion of models, but these may also be difficult to maintain and update to accommodate new features as they are added to the commercial codes. Two prominent examples of converters include ESA's TASverter[1] and a NASA-GSFC developed GMM/TMM converter[2]. The TASVerter code utilized STEP-TAS as an intermediate format and also supports export to TRASYS. The NASA GMM converter cannot directly read the Thermal Desktop dwg format (which is a proprietary format for AutoCAD) and therefore reads the Thermal Synthesizer System (TSS) format. The TSS format is an ASCII file that includes object properties that more closely matches those of Thermal Desktop or STEP-TAS than the TRAYS format is readily exported from Thermal Desktop and preserves the above mentioned aspects of the GMM. Consequently, this adds another layer in the conversion process from Thermal Desktop to ESATAN-TMS.

III. General Considerations

As a user with a need to convert models, it is beneficial to understand and be aware of the differences between codes to have some knowledge of what features are readily convertible, what features have workarounds, and what features are impossible to convert to a given destination tool. Models requiring conversion should include only the portions needed. While this may seem like an obvious statement, the complexity of model having more capability to simulate many configurations (Op vs. Non-Op, Stowed vs. Deployed, Hot Biased vs. Cold Biased, etc.) results in logic or different configurations that need to be converted depending on the capabilities of the destination code. For example, delivering a full observatory model with complex logic for battery charging/ discharging, dissipations as a function of reaction wheel speed, and various instrument on/off states for a launch analysis results in considerable effort to convert logic that will never need to be executed in the applicable analyses.

Thermal Desktop includes the capability to model each configuration in a separate Case Set with symbol and property overrides, which may affect geometry (e.g. Stowed vs. Deployed configurations) as well as network features (e.g. Heat Loads, Enable/Disable of Heaters, etc.). It is important when exporting from Thermal Desktop to set all the current values of symbols to the values as specified in a relevant Case Set; Thermal Desktop does provide a command for performing this as well as resetting back to the default values. A minimum number of GMM configurations should be converted; if the GMM is the same between two Case Sets, it should only be converted once. However, delivery of each configuration of the TMM as a separate file (e.g. On Deployed, Off Deployed, Off Stowed, Calibration Mode, etc) can be a challenge to integrate at the next higher level of assembly. These configurations are best controlled through the use of state flags to allow for variable changes during the execution of the thermal model rather than variable changes during the generation of the model as shown in Figure 1.

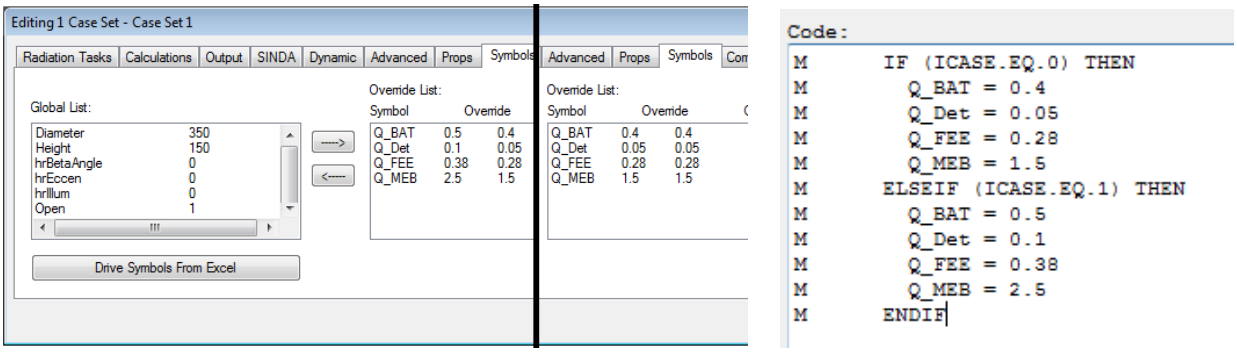

Figure 1. Two Different Approaches for Variables: *The left view shows two Case Set Symbol overrides to change the power variables. The right shows a preferred method using user logic that updates the variables based on a configuration flag. This approach allows a single model conversion to be performed that can simulate both configurations within the same TMM file simply with a single flag change.*

IV. Geometric Math Model Conversion

By far, the most difficult thing to accommodate in GMM conversion is the use of cutting operations in ESATAN-TMS. Nearly no other thermal code includes this feature to the level of ESATAN-TMS. While it is a powerful feature for ESATAN-TMS users when coupled with the ACG, it is considerable effort to replicate the effects

manually. Therefore, its use should be considered carefully before implementing that feature for models requiring conversion. Cuts that can be accomplished with native surface capabilities (such as an axial cut on a disc, Figure 2) should never be done using cutting operations. Cuts that leave only the inside planar surfaces of a triangular prism are better to represent with quadrilaterals and triangles rather than cut resultants.

Cutting away entire surfaces of an assembly should be avoided (i.e. only perform cuts on surfaces that are actually cut; surfaces that are entirely removed by a cutting operations should have never been included). Handling of cutting operations by the NASA-GSFC converter simply includes the cutter surface as an additive Boolean but disabled for radiation. Therefore, any surfaces that are fully cut away are also preserved. In one instance (Figure 3), a spacecraft panel with 22 nodes was duplicated 10 times, one for each cut operation with a triangular prism, resulting in the conversion of the entire panel 10 times resulting in 10 duplicated surfaces. An example where cutting operations may be replicated by a native Thermal Desktop surface is an ellipse, which does not exist as a base surface type in ESATAN-TMS but does in other codes. In general, cutting operations should be used as sparingly as possible if model conversion is required.

Another challenging aspect of conversion from Thermal Desktop to ESATAN-TMS is the difference in ability to number subdivided surfaces. Thermal Desktop (and TSS) both support full user control over the numbering without imposing a *starting node*, *increment* schema to the numbering. The NASA-GSFC converter has an option to export these as individual surfaces in order to preserve the nodal numbering within the node numbering limitation imposed by ESATAN-TMS. Thermal Desktop also supports the assignment of different submodels, MLI, and optical properties to sub-areas of a surface (i.e. a part of the surface but not all) as shown in Figure 4. Upon export to TSS, these sub-area assignments are not preserved since the TSS format cannot support this. Therefore, this capability should be avoided if model conversions are necessary. A simple workaround is to create multiple surfaces in Thermal Desktop with the appropriate parameters to facilitate conversion.

The next most challenging aspect to convert are finite elements, which are handled differently between Thermal Desktop and ESATAN-TMS. Similar to the ability to assign numbers in any pattern to surfaces, finite elements can have any numbering assigned to the corner nodes in Thermal Desktop. It is incumbent on the user to ensure that these nodes are either identically numbered or merged to ensure conduction between elements in Thermal Desktop. However, ESATAN-TMS finds the largest assigned node number and then renumbers and merges all FE nodes within a geometrical tolerance. Unfortunately, this approach will only work if the TMM itself is generated by ESATAN-TMS and would fail in a converted model since the existing

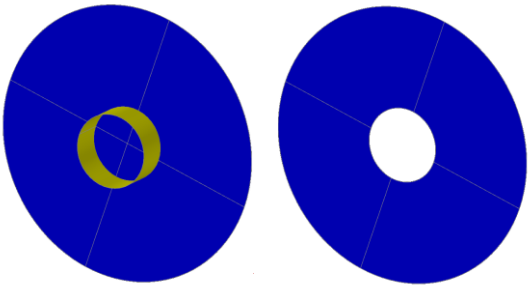

Figure 2. Unnecessary Cutting Operation: *The existing capabilities allow for an inner radius to be specified. No need to cut an on-axis hole in a disc as was delivered.*

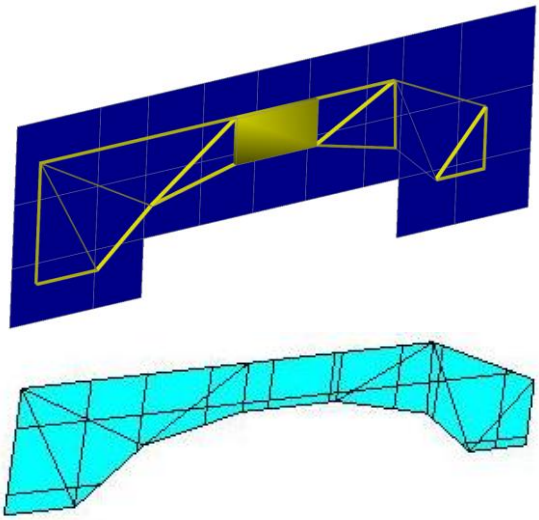

Figure 3. Avoidable Cutting Operation: *The resultant shape was accomplished with 10 copies of the blue structure and 9 triangular prism and 1 box cutting operations. This would have been better accomplished using quadrilaterals and triangles if needing conversion*

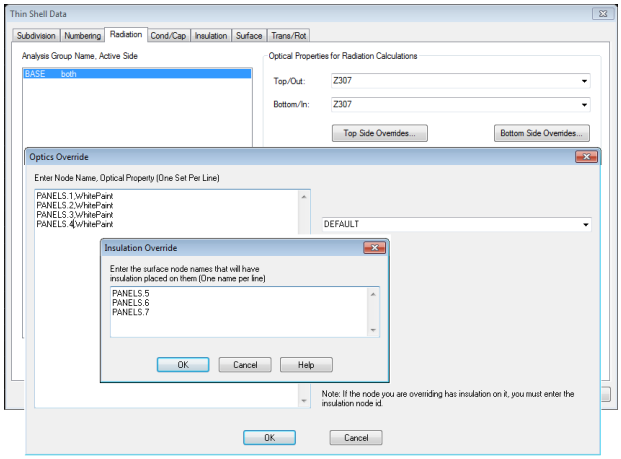

Figure 4. Sub Area Assignments: *Thermal Desktop allows for overrides of optical properties or insulation for a sub-area of a surface*

radiative conductors would reference nodes that have been renumbered. To accommodate this in the NASA-GSFC converter, any finite elements in a Thermal Desktop/TSS model are subdivided into their associated nodal area and made centroid surfaces for the purposes of radiation calculations. The corresponding TMM is based on the original FE nodalization and numbering. Similarly, in ESATAN-TMS, the use of finite elements should be approached carefully unless the nodal assignments made by ESATAN-TMS to the surfaces can be identified reliably. Otherwise, a mismatch between the GMM and TMM for those node numbers will exist (i.e. the assigned node number is not explicitly contained in the GMM text).

Another type of object found in Thermal Desktop/TSS that does not exist in ESATAN-TMS is a polygon (or more specifically a triangle fan, where all triangles share one point in common). For four sided polygons, this can often be converted into a quadrilateral, although the enforcement of planarity is not imposed in Thermal Desktop as it is in ESATAN-TMS. An option exists in the NASA-GSFC converted to split these surfaces into constituent triangles.

Lastly, the ESATAN-TMS approach of defining all the surfaces and then assembling them upwards until the final model is defined is unique in that any number of surfaces can be defined but never make it into the model used for calculations. These surfaces should be removed prior to delivery to eliminate the necessity to convert them and remove the possibility that they are accidentally included.

GMM Things to Avoid:

1. ***ESATAN-TMS: Cutting operations unless truly necessary***
2. ***ESATAN-TMS: Finite Elements unless surface node assignment numbering can be conclusively identified***
3. ***ESATAN-TMS: Surfaces that do not get included in final roll up of top level model***
4. ***Thermal Desktop: Sub-area optical property, submodel, or MLI assignments***
5. ***Thermal Desktop: Use of Case Sets to override variable values; include this as logic that is flag dependent.***

V. Thermal Math Model Considerations

GENERAL

The GMM tends to be slightly easier to convert because of the stricter object structure associated with surface definitions. However, the TMM is generally based on a combination of object structure (Nodes, Conductors, Arrays) and logic to control the program flow. Most current aerospace codes use FORTRAN as the underlying language to allow the user to extend basic FORTRAN capabilities (sometimes referred to as MORTRAN) in the context of the thermal model. Converting the object structure for associated Nodes, Conductors, and Arrays is relatively straightforward, but converting the *references* to these objects in the logic sections of the model is considerably more difficult, particularly when the same object type has different properties between the two codes.

ESATAN includes a hierarchal structure to submodels which limits the scope of variables, nodes, conductors, and arrays to only the same hierarchy level or below. SINDA does not have a hierarchy below the first level and all submodel objects are globally available across all submodels. To replicate SINDA's scope in ESATAN, it is advised to place only nodes in their appropriate submodel level and also to place arrays, conductors, and logic at the top ESATAN hierarchal level to ensure proper scope visibility. However, this practice does place a constraint on duplicating variable names in ESATAN in different submodels. Similarly, ESATAN's nesting of submodels can only be represented by a single level of submodels in ESATAN, which precludes two submodels having the same name, even if their parent submodel is different.

NODES

Nodes in thermal models have basic properties that are essential for the solution. These include number (or name or some other unique identifier), temperature, and capacitance. Furthermore there may be additional properties that may be useful to the thermal analyst, such as: label or description, location, associated area, mass, or volume, optical properties, etc. ESATAN node definitions often include A (area), ALP (absorptivity) and EPS (emissivity) properties for which no existing counterparts exist in SINDA. It is best to ensure that these properties are defined in the $NODES section and not changed in logic blocks since no corresponding variables exist to be referenced by SINDA. The nodal A property (area) is sometimes referenced by MLI conductors as an expression with an ε* (e.g. GR(1,10001) = 0.03 * A1;) The NASA-GSFC converter will replace any instances encountered in the logic blocks or the $CONDUCTORS section with the values identified from the $NODES block. However, if a nodal property, such as A is changed later, this update will not be propagated correctly.

Furthermore, some of the nodes properties may be dependent on temperature itself, such as the capacitance. Table 1 shows the syntax in SINDA and ESATAN for equivalent node definitions. Nodes with a capacitance of zero are

handled differently between SINDA and ESATAN. SINDA does not allow the capacitance itself to be set to zero, but rather requires an identifier flag of a negative capacitance value to specify that a node is massless. ESATAN does not have such an identifier and excludes nodes of zero capacitance when using automatic timestep calculation. Since the automatic timestep calculation is often related to the nodes capacitance divided by the sum of conductors tied to it, a zero capacitance would result in a timestep of zero as well. However, one practice that should be avoided is the implicit assignment of zero capacitance by omission. This ambiguity makes it difficult to identify the type of node based on information contained in the $NODES section.

Node Type	SINDA	ESATAN
Diffusion	1, 25.0, 432.4	D1 = 'Node 1', T = 25.0, C = 432.4;
Arithmetic	2, 25.0, -1.0	D2 = 'Node 2', T = 25.0, C = 0.0;
Implied Arithmetic	Not allowed	D2 = 'Node 2', T = 25.0; # AVOID THIS!!!
Temp Dependent	SIV 3, 25.0, A5, 2.1	D3 = 'Node 3', T = 25.0, C = INTRP1(T3, Array5, 2.1);
Temp Dependent…	4, 25.0, 1.0	D4 = 'Node 3', T = 25.0, C = 1.0; # ASSIGN LATER!!!
…But assigned later	C4 = interpolation call	C4 = Interpolation call

Table 1. Listing of Node Type Definitions in SINDA and ESATAN

SINDA and ESATAN also approach temperature dependent capacitances slightly differently. SINDA includes explicit card identifiers (e.g. SIV) that identify a particular node as having a temperature dependence and includes a reference Capacitance vs. Temperature array reference and a multiplier, but SINDA does force the dependence to be exclusively on the node temperature itself and not some other parameter. ESATAN does not use a card identifier but rather allows the capacitance itself to be an expression which may include interpolation and multiple variables. It is advised that if capacitances in ESATAN are not constant, they should be assigned a value of 1.0 in the definition and the capacitance assignment be performed in the appropriate logic block. To summarize, all nodes in ESATAN definitions should explicitly include a capacitance value, even if it is zero with temperature dependences assigned in later logic blocks

One other caveat is the use of loop constructs (e.g. DO, REPEAT, etc) to access node properties such as below:

```
DO I =0,3
   A(101+I) = 2.1 # Assign Area property for node i
   C(101+I) = 432.1 # Assign Capacitance for node i
END DO
```

While this direct access approach is strongly discouraged by the software vendors and explicit functions provided for the access to internal variables based on their assigned numbering, it is not expressly forbidden and even some current models still utilize this approach. SINDA internally groups the diffusion nodes together followed by the arithmetic nodes, boundary nodes, and heater nodes within a submodel. Therefore the internal ordering of the nodes is not consistent between ESATAN and SINDA. Based on table 1 above, SINDA would store these nodes internally at (1,3,4,2) whereas ESATAN would store them in the input order (1,2,3,4). This makes any loop constructs in the logic that relies on this ordering very difficult to convert and therefore should be avoided. It is better to use functions that explicitly de-reference the node number to its internal index, which avoids pitfalls based on assumptions of internal numbering schema.

Lastly, ESATAN includes an option to make a node inactive (X300 = 'INACTIVE';). This feature ignores any conductors connected to the specified node during the solution. SINDA does not have an equivalent capability, although it is possible to use logic routines to determine node pairing for all conductors and set the value to zero if either node is the specified node. If the node is then cast as a boundary node, the SINDA solution could proceed, as SINDA will not solve if a node is found to be unconnected. However, for a delivered model, there are few reasons why inactive nodes should be included, especially if the node is never referenced by any configuration of the model.

CONDUCTORS

Conductors in thermal models similarly have basic properties necessary for the solution similar to nodes. These include the conductor type (Linear or Radiative), unique identifier, first node, second node, and conductance value. Similar to nodes, it is possible to have a temperature-dependent conductance and this is handled similarly to nodes described in the previous section. One practice that should be avoided is the coupling of identical nodes to themselves (e.g. node 100 to node 100). This causes convergence problems in SINDA and has no physical basis for being done.

ARRAYS

A difference between ESATAN and SINDA is in the nomenclature for arrays; SINDA uses numbers and an A# scheme to reference them in logic, while ESATAN allows for character names which are then used as variable names when referenced in logic. ESATAN also requires that the dimensions of the array be explicitly declared (e.g. MyArray(2,6) = …), whereas SINDA makes no such distinction and the type of the array (singlet or doublet) is inferred based on the calling routine in the logic block. SINDA also allows for the mixing of types (Integer or Real) within an array and uses the FORTRAN EQUIVALENCE feature to access data by its type. Although this is seldom used, this capability is used at times to pass in an array of arguments to a user developed function which has mixed types.

LOGIC

User logic is by far the most time consuming aspect of TMM model conversion. The approach employed by the GSFC converter utilizes wrapper functions that have identical names to the base code to preserve the similarity as much as possible to function and subroutine names. Over time and with more and more conversions being done, a library of wrapper functions will eventually be built up to promote the reuse of verified code. Before delivering a model, the logic should be scrubbed for lines of code that would never be executed by the model recipient. For example, a typical launch model does not need to have logic included for reaction wheel speed or solar cell efficiency as a function of load. Logic that is not needed should be removed prior to delivery to reduce the effort associated with conversion. There is no value in spending resources to convert lines of code that will never be executed.

Another caveat to converting between SINDA and ESATAN is a difference in the nodal heat value behavior at the start of each timestep. SINDA resets all the nodal Q variables to zero at the start of each timestep. However, ESATAN includes five different Q values (QS, QA, QE, QI, and QR) but these values are not reset to zero at the start of each timestep. Therefore, additive statements such as Q1 = Q1 + 1.0 would behave differently if converted to ESATAN as QR1 = QR1 + 1.0. Additive statements in SINDA are common since a node may have environment load, dissipation and heater power all applied to the total Q in SINDA. It is advised to reset all Q[x] values to zero at the start of each iteration in the $VARIABLES block to emulate the SINDA behavior when converting to ESATAN.

Additionally, a feature that exists in ESATAN that has no counterpart in SINDA is the inclusion of an $INITIAL block where initial temperatures may be set and potentially where capacitance assignments are made. This logic should be placed at the start of the OPERATIONS block in SINDA prior to any lines that would appear there from the EXECUTION block from ESATAN.

Lastly, any references to variables in the logic sections should have explicit declarations of the variables. ESATAN and SINDA also behave differently with submodels in ESATAN allowing for named variables that are only within scope for that particular submodel. SINDA includes submodel specific user variables but these follow a numbered identifier (e.g. XK100, K10) rather than a more intuitive name. However, SINDA does include named variables in the REGISTER data block, but these variables are in scope across all submodels (e.g. they act as global variables). The submodel user data (K and XK) usage has generally been replaced by the more capable REGISTER data features, but is still a valid possibility in a SINDA model. Furthermore, ESATAN will allow for undeclared variables to exist and follows the typical FORTRAN implicit naming convention (I-N for integer, all others are real). ESATAN also allows for character type variables (e.g. ‘ON’, ‘OFF’, etc.) which is not supported by SINDA REGISTERS. If model conversion is a necessity, then avoiding the use of character type variables in favor of integer types is preferable. It is simple enough to represent ‘ON’ with 1, and ‘OFF’ with 0 or similar.

TMM Things to Avoid:

1. ***ESATAN: Always explicitly define the capacitance value, but never reassign a non-zero capacitance to one that was originally defined as zero***
2. ***ESATAN: Define all arithmetic nodes with an explicit zero capacitance value***
3. ***ESATAN: Define temperature dependent capacitances with a value of 1.0 and ensure that an interpolation routine is called in the VARIABLES section***
4. ***BOTH: Do not use looping constructs that rely on internal sorting schema***
5. ***BOTH: Avoid coupling a node to itself***
6. ***ESATAN: Avoid the use of inactive nodes in delivered models***
7. ***SINDA: Avoid using the EQUIVALENCE features of arrays if possible***
8. ***BOTH: Do not include logic that will never be executed at the next higher level of analysis***
9. ***ESATAN: Ensure that additive nodal assignments are properly handled***
10. ***BOTH: Do not use any variables that are not explicitly declared and use numeric values rather than character types when possible.***

VI. Conclusions and Path Forward

If it is known early in a project that model conversion will be needed, care should be taken when constructing models to avoid using capabilities and features that are not supported or that cannot be emulated in the destination software. The effort to convert models to a different format for delivery should not be underestimated and may take considerable time, but keeping conversion considerations in mind can simplify the process. Both the GMM and TMM models need to be converted and each may pose difficulties. Structuring the GMM and TMM in a manner favorable to convertibility can save considerable time. Minimizing the use of cutting operations and finite elements in ESATAN-TMS and avoiding sub-area assignments and Case Set overrides in Thermal Desktop makes for models that are easier to convert to the other format. Similarly, care should be taken in the TMM to ensure node types are explicitly defined, additive nodal Q assignments are carefully handled, variables are all explicitly defined, and logic that is never executed is removed.

While tool developers could augment their codes to better support features of their competitors, this is unlikely to happen. Tool developers generally focus on developing their own capabilities and often do not want to appear to be copying the competition. Furthermore, fundamental differences in the code architecture may not be worthwhile to rework to accommodate the infrequent needs of a handful of users requiring model conversions. In short, it is generally assumed that the developers will focus their resources on further developing their own architecture than trying to emulate capabilities of a competitors' code.

Mission success is based on the efforts of many organizations and cooperation between organizations leads to less conflict. With limited resources available from all organizations, the effort and cost to convert models should be shared with both sides making concessions, tempering expectations, and being understanding of the capabilities and limitations of each side's analysis tools. The efforts and responsibilities for each party should be defined and agreed upon early in a program, as soon as usage of differing toolsets is identified. It should not be one side delivering a model of poor quality for conversion and the other side shouldering the entire responsibility for converting the model. Making a few concessions on the use or avoidance of features during the development of the models can streamline conversion and sharing of models across organizations and leads to a greater chance of mission success.

Recommendations

GMM Things to Avoid:

1. ***ESATAN-TMS: Cutting operations unless truly necessary***
2. ***ESATAN-TMS: Finite Elements unless surface node assignment numbering can be conclusively identified***
3. ***ESATAN-TMS: Surfaces that do not get included in final roll up of top level model***
4. ***Thermal Desktop: Sub-area optical property, submodel, or MLI assignments***
5. ***Thermal Desktop: Use of Case Sets to override variable values; include this as logic that is flag dependent.***

TMM Things to Avoid:

1. ***ESATAN: Always explicitly define the capacitance value, but never reassign a non-zero capacitance to one that was originally defined as zero***
2. ***ESATAN: Define all arithmetic nodes with an explicit zero capacitance value***
3. ***ESATAN: Define temperature dependent capacitances with a value of 1.0 and ensure that an interpolation routine is called in the VARIABLES section***
4. ***BOTH: Do not use looping constructs that rely on internal sorting schema***
5. ***BOTH: Avoid coupling a node to itself***
6. ***ESATAN: Avoid the use of inactive nodes in delivered models***
7. ***SINDA: Avoid using the EQUIVALENCE features of arrays if possible***
8. ***BOTH: Do not include logic that will never be executed at the next higher level of analysis***
9. ***ESATAN: Ensure that additive nodal assignments are properly handled***
10. ***BOTH: Do not use any variables that are not explicitly declared and use numeric values rather than character types when possible.***

References

[1] Etchells, J, "A Standard for the Exchange of Thermal Analysis Data (STEP-TAS)" 46th International Conference on Environmental Systems, 2016.

[2] Peabody, H. and Yang, K.., "Preliminary Development of a TSS and SINDA/FLUINT to ESARAD/ESATAN Thermal Model Converter", Thermal and Fluids Analysis Workshop 2014