Improvements in Thermal Protection Systems Sizing Capabilities for TCAT

Stephen James Izon

AE 8900 Special Project
December 14, 2001

School of Aerospace Engineering
Georgia Institute of Technology

Advisor: Dr. John R. Olds

Acknowledgements

First and foremost, I would like to thank my mother and father for their immeasurable support, guidance, and love throughout the years. The wisdom and advice you have provided me has become such a part of my life and you have helped me realize that everything always works out in the end.

I would also like thank my friends who have always been there for me all these years. A very special thanks to Gerald Brenner, my best of friends, who has taught me many valuable lessons about school, life, and friendship. Thanks also to Camella Challender who has always helped me see things in a different light.

The deepest thanks to Emily Mayer, who has been with me from the beginning. You have always been a continuous source of motivation and inspiration to me.

I also want to thank my research sponsor, D.R. Komar at NASA Marshall Space Flight Center. You have been an invaluable source of input and support.

To the members of the Space Systems Design Lab, I send you great thanks. You are such an incredible group of individuals, always been willing to help in times of need.

Finally, I would like to thank Dr. John Olds for giving me the opportunity to become a member of the Space Systems Design Lab. Without your advice and generosity, this research would not have been possible.

Table of Contents

Abstract

TCAT, originally developed for the Space Systems Design Lab at the Georgia Institute of Technology is a conceptual design tool capable of integrating aeroheating analysis into conceptual reusable launch vehicle design. It provides TPS unit thicknesses and acreage percentages based on the geometry of the vehicle and a reference trajectory to be used in calculation of the total cost and weight of the vehicle design.

TCAT has proven to be reasonably accurate at calculating the TPS unit weights for in-flight trajectories; however, it does not have the capability of sizing TPS materials above cryogenic fuel tanks for ground hold operations. During ground hold operations, the vehicle is held for a brief period (generally about 2 hours) during which heat transfer from the TPS materials to the cryogenic fuel occurs. If too much heat is extracted from the TPS material, the surface temperature may fall below the freezing point of water, thereby freezing any condensation that may be present at the surface of the TPS. Condensation or ice on the surface of the vehicle is potentially hazardous to the mission and can also damage the TPS. It is questionable whether or not the TPS thicknesses provided by the aeroheating analysis would be sufficiently thick to insulate the surface of the TPS from the heat transfer to the fuel. Therefore, a design tool has been developed that is capable of sizing TPS materials at these cryogenic fuel tank locations to augment TCAT's TPS sizing capabilities.

In addition, the Crank-Nicolson method was substituted for the simple implicit (Laasonen) Method used in the numerical solution for the interior nodal points of TCAT. The simple implicit method has first-order accuracy with a truncation error of $O\left[(\Delta t),(\Delta x)^2\right]$. The Crank-Nicolson Method makes use of trapezoidal differencing to achieve second-order accuracy with a truncation error of $O\left[(\Delta t)^2,(\Delta x)^2\right]$. This method was selected because it is the same method used in the commercial heating code SINDA. The Crank-Nicolson method is proven to require fewer nodes and less execution time than the simple impicit method. Therefore, the solution process may take less time to complete, and the numerical accuracy should be better than that of the simple implicit method.

List of Tables

List of Figures

Nomenclature

ADS	Automated Design Synthesis
CFBI	Composite Flexible Blanket Insulation
CGI	Common Gateway Interface
DSM	Design Structure Matrix
GrEx	Graphite Epoxy
HTML	Hypertext Markup Language
LI-2200	22 lbm/ft^3 Rigid Ceramic Tile Insulation
NASA	National Aeronautics and Space Administration
PC	Personal Computer
RCC	Reinforced Carbon-Carbon
RLV	Reusable Launch Vehicle
SiC	Silicon Carbide
STS	Space Transportation System
TCAT	Thermal Calculation Analysis Tool
TPS	Thermal Protection System
TPS*X*	Thermal Protection System Expert
TUFI	Toughened Uni-Piece Fibrous Insulation

List of Symbols

α Thermal diffusivity, $\left[\frac{m^2}{s}\right]$ or $\left[\frac{ft^2}{s}\right]$

Δx Spatial Step, $[m]$ or $[ft]$

Δt Temporal Step, $[s]$

$\frac{\partial T}{\partial t}$ Partial derivative of temperature with respect to t

$\frac{\partial^2 T}{\partial x^2}$ Second partial derivative of temperature with respect to x

$\frac{dT}{dx}$ Derivative of temperature with respect to x

ε Material emissivity

σ Boltzmann constant, $5.67051\times10^{-8}\left[\frac{W}{m^2\cdot K^4}\right]$ or $4.75644\times10^{-13}\left[\frac{Btu}{s\cdot m^2\cdot K^4}\right]$

f_i System of equations formed after discretization of the heat equation

i Spatial Position Index

$J_{i,j}$ Jacobian matrix

k Material thermal conductivity, $\left[\frac{W}{m\cdot K}\right]$ or $\left[\frac{Btu}{lbm\cdot s\cdot R}\right]$

L Material stack thickness, $[m]$ or $[ft]$

N Maximum number of nodes

n Time index

q Heat transfer rate, $\left[\frac{W}{m^2}\right]$ or $\left[\frac{Btu}{ft^2\cdot s}\right]$

T Temperature, $[K]$ or $[^\circ F]$

t Time, $[s]$

x Spatial position in material normal to exposed surface, $[m]$ or $[ft]$

1.0 Project Motivation

Ground Hold Analysis Tool

Conceptual design of launch vehicles is generally an iterative process in which disciplinary analyses are performed individually. At the Space Systems Design Lab at the Georgia Institute of Technology, research in the conceptual design of third generation launch vehicles flows through a sequence of disciplinary analyses found in a Design Structure Matrix. The results of each disciplinary analysis are sent to other disciplines whose analyses are dependent upon those results. A typical design structure matrix is illustrated in Figure 1.

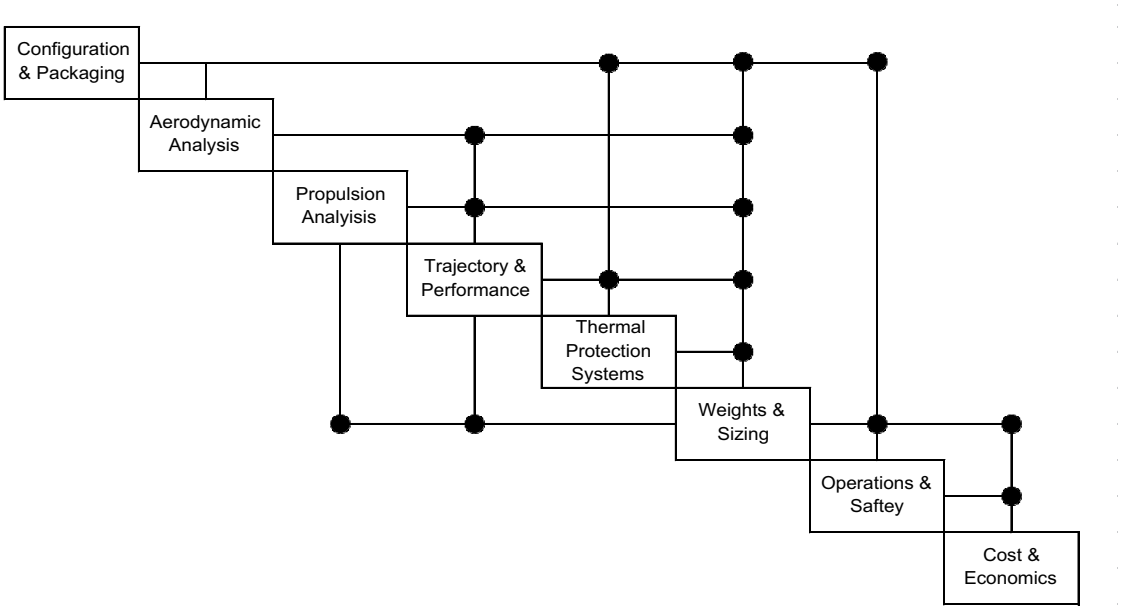

Figure 1: Design Structure Matrix

From Figure 1 we can see that the TPS sizing is dependent on the trajectory and configuration disciplines. More specifically, trajectory data such as altitude, relative velocity, and angle of attack versus time as well as geometry data are required in order to perform the aeroheating analysis and generate the TPS thicknesses and unit weights.

Once the TPS material thicknesses and unit weights are determined for the iteration, the data is sent to the weights and sizing discipline.

TCAT, the Thermal Calculation Analysis Tool is a conceptual design tool originally developed for the Space Systems Design Lab to integrate aeroheating analysis into conceptual reusable launch vehicle design. TCAT is a FORTRAN code that uses finite difference methods to perform transient in-depth one-dimensional conduction analysis over the center mold line of the vehicle. TCAT couples with MINIVER, TPS*X*, and ADS to correctly size the vehicle's thermal protection system. MINIVER is an Aeroheating code that produces centerline radiation equilibrium temperatures, convective heating rates, and head loads over simplified vehicle geometries. TPS*X* is a NASA Ames material properties database that is available on the World Wide Web. ADS (Automated Design Synthesis) is a numerical optimizer that uses algorithms to solve constrained and unconstrained design problems. A schematic of the TPS sizing process using TCAT is shown in Figure 1.

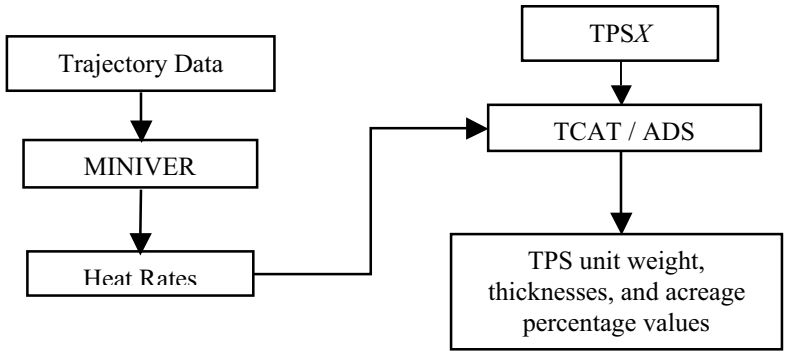

Figure 2: TPS Sizing Process

When TCAT was originally created, its goal was to size TPS materials for in-flight aeroheating. However, sizing the TPS solely for in-flight aeroheating may not guarantee that the vehicle will meet all of its performance requirements. During in-flight operations, the surface of the vehicle experiences very high temperatures as well as high heat rates induced by convective heating. Therefore, TPS materials are selected so that the surface temperature of the materials does not exceed their operating temperature

limits. The materials are sized so that the temperatures experienced by the rest of the vehicle, namely the vehicle structure or tank structure, do not exceed some critical temperature, which would result in thermal failure.

What TCAT did not account for is that during ground hold operations, the vehicle is held at the launch facility for a brief period while it is prepared for launch. The vehicle is exposed to the ambient temperature of the air, as opposed to the high temperatures caused by aeroheating during flight. Heat is transferred from the atmosphere to the surface of the TPS at a much lower rate and heat transfer occurs within the depth of the TPS material stack. TPS materials applied over a cryogenic tank must be sized to prevent the formation of ice on the outer surface of the TPS material. Since the fuel inside the cryogenic fuel tank is very low (-423.67 F for LH2, -297.67 for LOx), heat is extracted from the tank structure as well as the TPS materials directly in contact with the tank structure, causing temperatures to decrease throughout. If the temperature of the outer surface of the TPS falls below the freezing point of water (32F) then frosting will occur. Condensation or frosting on the surface of the TPS material is considered hazardous since it will negatively affect the performance of the vehicle or damage the TPS. Therefore, a design tool capable of sizing TPS materials during ground hold operations must be developed to ensure that the overall TPS thicknesses selected for the vehicle will be adequate to satisfy not only the in-flight aeroheating constraints, but also the ground hold constraints.

Crank-Nicolson Method and TCAT

At the conceptual design level, since the level of fidelity is not as high as it would be for detailed design, conceptual design tools that provide a high level of accuracy with low computational times are desired. TCAT is solves the one-dimensional heat equation numerically using a simple implicit (Laasonen) method for the interior nodes of a TPS material stackup. Although this method is unconditionally stable it is only first order accurate with a truncation error of $O\left[(\Delta t),(\Delta x)^2\right]$. The Crank-Nicolson method is another implicit numerical scheme that is also unconditionally stable; however, it makes use of trapezoidal differencing to achieve second-order accuracy with a truncation error of

$O\left[(\Delta t)^2, (\Delta x)^2\right]$. This method is the same as that used in the commercial heating code SINDA. The solution method of the resulting system of equations is the same as for the simple implicit method, but generally the Crank-Nicolson method requires fewer nodes and less execution time. Therefore, substitution of the Crank-Nicolson method for the simple implicit method may make TCAT a more valuable and practical conceptual design tool.

2.0 Ground Hold Analysis Tool Development

The original version of TCAT operates by taking aeroheating data from MINIVER (a commercial aeroheating code) and sizing the TPS material for each body point specified by MINIVER. The data provided by MINIVER is the radiation equilibrium temperature and the convective heat rate versus time. During groundhold it is assumed that the heat transfer is constant and determined by the ambient temperature of the air. Since the ambient temperature of the surrounding atmosphere is constant, and the heat transfer rate is constant, it is not necessary to create a file like those provided by MINIVER. Instead, these constants can be entered explicitly into the numerical solution process.

The ground hold analysis only needs to be applied to one representative body point above a cryogenic fuel tank location. This is because it is assumed that the conditions at one body point location would be the same as any other location above the same cryogenic fuel tank. The unit weight calculated from the single body point analysis can then be applied to the total area of the vehicle surface above cryogenic fuel tank locations for which the same TPS material will be used under the same ambient conditions.

Numerical Solution Method

TCAT uses a fully implicit method to solve the parabolic, one-dimensional unsteady heat conduction equation by marching in time.

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{1}$$

The boundary condition given by equation (2) is applied to the top surface of the TPS material. This equation is the energy balance relationship for the top surface of the TPS material. The convective, radiative, and conductive heat transfers are summed to equal zero in order to preserve the conservation of energy.

$$q_{conv} - \varepsilon\sigma T_s^4 + k\frac{dT}{dx} = 0 \text{ at } x = 0 \tag{2}$$

However, for groundhold, a constant heat transfer mechanism is assumed, where the heat transfer coefficient is 1 $BTU/\left(ft^2 \cdot hour \cdot {}^\circ F\right)$. Here, the atmosphere is treated as an infinite atmosphere in which heat transfer to the surface TPS material can occur, but the temperature of the atmosphere does not change. Therefore, the boundary condition given in equation (3) is used for the groundhold analysis instead of equation (2).

$$k\frac{dT}{dx} = q_{in} \text{ at } x = 0 \tag{3}$$

At the backface of the TPS material the original TCAT uses the boundary condition in equation (4), which assumes that there is an adiabatic wall at the backface of the material.

$$\frac{dT}{dx} = 0 \text{ at } x = L \tag{4}$$

However, for groundhold an isothermal boundary condition is applied.

$$T = T_{fuel} = const \text{ at } x = L \tag{5}$$

This isothermal assumption is applied because during groundhold it is assumed that the temperature of the fuel remains constant, which is in direct contact with the backface of the cryogenic fuel tank structure. Heat can be transferred from the material to the fuel at the interface of the tank and fuel; however, it is assumed that the heat transfer that does occur does not affect the overall temperature of the fuel. This assumption is valid because the fuel can be treated as an infinite reservoir, due to the large volume of fuel that is contained inside the cryogenic fuel tanks.

At the surface, using the boundary condition given by equation (3) the finite difference equation that results is given in equation (6).

$$T_1^n + \frac{2\alpha_1 \Delta t}{\Delta x}\frac{q}{k_1} = -\frac{2\alpha_1 \Delta t}{\Delta x^2}\left(T_2^{n+1} - T_1^{n+1}\right) + T_1^{n+1} \tag{6}$$

In equation (6), the constant q is the heat transfer from the atmosphere, and is equal to the ambient temperature times the heat transfer coefficient.

At the interior nodes, using a simple implicit central finite difference scheme, the resulting discretization is given by equation (7).

$$T_i^n = -\frac{\alpha_i \Delta t}{\Delta x^2}\left(T_{i-1}^{n+1} - 2T_i^{n+1} + T_{i+1}^{n+1}\right) + T_i^{n+1} \tag{7}$$

On the back surface of the material, the isothermal boundary condition of equation (5) is applied and results in equation (8).

$$T_N^n = T_N^{n+1} = T_{fuel} = const \tag{8}$$

Examining these equations, we see that we now have a system of linear equations. In the numerical method used by TCAT for the aeroheating analysis, a system of non-linear equations resulted, because of the fourth-order temperature dependence of the radiative heat transfer term. Since our system is now linear, the solution process can be made much simpler because we no longer have to reiterate to converge to the solution. The system of equations can be rearranged into the form $Ax = b$ as shown in equation (9) to simplify the solution process, where A is a matrix of coefficients, x is an array of temperatures at the next time step, and b is an array of temperatures at the current time step.

$$\begin{bmatrix} 1+\frac{\alpha_1\Delta t}{\Delta x^2} & -\frac{2\alpha_1\Delta t}{\Delta x^2} & 0 & 0 & 0 & 0 \\ -\frac{\alpha_i\Delta t}{\Delta x^2} & \frac{2\alpha_i\Delta t}{\Delta x^2} & -\frac{\alpha_i\Delta t}{\Delta x^2} & 0 & 0 & 0 \\ 0 & -\frac{\alpha_i\Delta t}{\Delta x^2} & \frac{2\alpha_i\Delta t}{\Delta x^2} & -\frac{\alpha_i\Delta t}{\Delta x^2} & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & -\frac{\alpha_i\Delta t}{\Delta x^2} & \frac{2\alpha_i\Delta t}{\Delta x^2} & -\frac{\alpha_i\Delta t}{\Delta x^2} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_{N-1} \\ T_N \end{bmatrix}^{n+1} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_{N-1} \\ T_N \end{bmatrix}^{n} \tag{9}$$

This solution of this system is easily obtained using the Thomas Algorithm, which is a numerical algorithm capable of solving tridiagonal banded matrices.

Optimization Method

For the ground hold analysis problem, the only constraint that needs to be enforced is the surface frosting condition. As long as the temperature of the surface of the TPS stays above the designated frosting condition, then the constraint is satisfied. If there were no heat transfer from the atmosphere, then we would expect the temperature profile of the surface of the TPS to decrease with time. In other words, the maximum temperature would be at time $t = 0$ and the minimum temperature would be at the end of the analysis. If there is heat transfer from the atmosphere, then we might expect the temperature of the surface to rise slightly, and then decrease to a minimum temperature at the end of the analysis. In both cases, we see that the minimum temperature is always at the end of the ground hold analysis time. Therefore, our only constraint would be to ensure that the temperature of the surface at the end of the analysis remains equal to or greater than our frosting condition temperature. Sizing the TPS to meet this constraint would guarantee that the design would be optimized, because it would be the minimum thickness required to properly insulate the surface of the TPS from the cryogenic fuel tank.

The ADS optimizer used in TCAT's aeroheating analysis is a "general problem optimizer,, meaning that it was created to handle many different types of engineering design problems. ADS has many controlling parameters that have to be fine-tuned in

order to achieve an optimal answer. Therefore, a problem specific optimizer was written into the main FORTRAN code to find the optimum design for the analysis. The Newton-Raphson method was implemented to find the root of the function F, where F is given by equation (10).

$$F = T_1^N - templmt(1) \tag{10}$$

In this equation, T_1^N is the surface temperature of the TPS at the final time, and $templmt(1)$ is the frosting condition temperature. Since the temperature T_1^N is a function of the thickness of the material being sized, F is also a function of the same thickness. Therefore, to find the optimal design, we want to drive the function F to zero.

The Newton-Raphson method is an iterative method used to find the root of a function. The Newton-Raphson formula is given in equation (11).

$$x_{i+1} = x_i - \frac{F(x_i)}{F'(x_i)} \tag{11}$$

In this notation, F is the same as in equation (10), and x is the root. In this case, we do not know the value of $F'(x_i)$, so we can approximate it using a backward finite divided difference. This gives us equation (12).

$$F'(x_i) = \frac{F(x_{i-1}) - F(x_i)}{x_{i-1} - x_i} \tag{12}$$

Finally, substitution into equation (11) results in equation (13), which was integrated into the main FORTRAN code.

$$x_{i+1} = x_i - \frac{F(x_i)(x_{i-1} - x_i)}{F(x_{i-1}) - F(x_i)} \tag{13}$$

We see that two initial guesses for the root must first be made before any subsequent approximations for the root can be made. Therefore, in the main code, two guesses for the root are made before the main optimization loop begins. Once the difference in consecutive approximations falls below 0.0001, convergence is declared and the optimized thickness is obtained.

Program Execution

The ground hold analysis begins with the input of the ground hold time, time step for the analysis, ambient temperature, fuel temperature, frosting temperature, and TPS stackup configuration data from a web browser. Once these are input, a CGI script ('single.cgi' for a single analysis or 'compare.cgi' for a stackup comparison) is called. The CGI script takes this data and creates an input file called 'inputs.in' which contains the constraints entered by the user, and also TPS material properties based on the materials selected by the user and data taken from the TPS*X* materials database. The FORTRAN main code ('ground.f' if the first layer is optimized or 'ground2.f' if the second layer is optimized) is called by the CGI script where the optimization is performed. The main FORTRAN code calls the subroutines 'one.f', 'two.f', 'three.f', 'four.f', or 'five.f', depending on the number of materials selected in the TPS material configuration. These subroutines solve the one-dimensional heat equation using the numerical method described in the previous section. Results are sent back to the main FORTRAN code, where the Newton-Raphson method is implemented to converge upon the solution. Once the solution is obtained, the results are sent back to the CGI script. These results are then written to the files 'results.txt' and 'thickness.txt', which are posted back to the user via the web interface. A schematic of the program flow logic is shown in Figure 3.

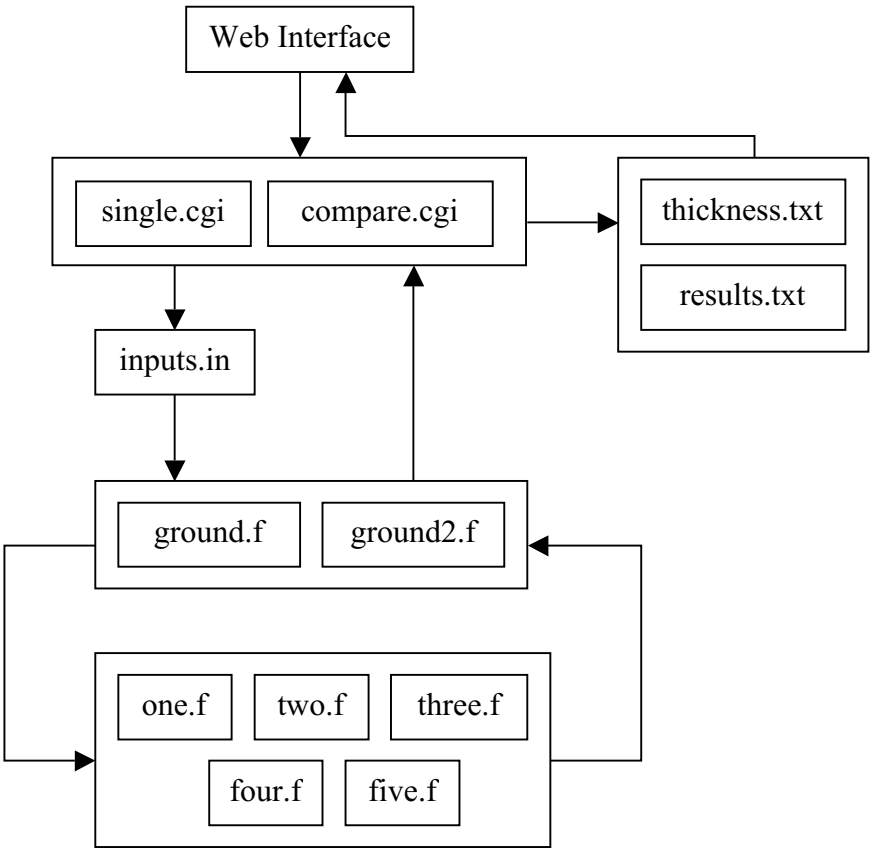

Figure 3: Schematic of Program Flow Logic

3.0 Ground Hold Analysis Test Results

To illustrate the necessity for sizing TPS during both ground hold and in-flight operations, TPS materials were sized for a 10° half-angle cone. This 10° half-angle cone could be representative of the nosecap of a reusable launch vehicle. Figure 4 illustrates the geometry of the cone used to build the MINIVER input file.

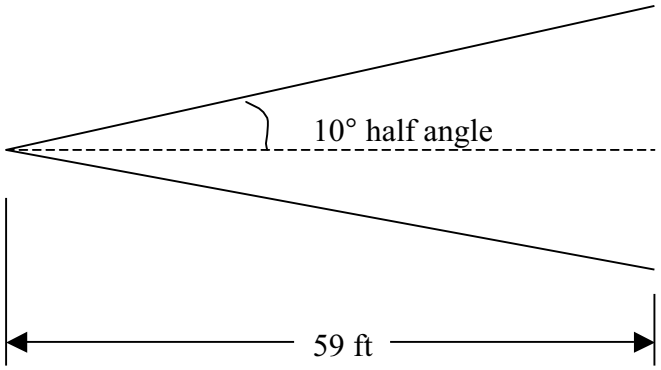

Figure 4: Schematic of 10° Half-Angle Cone

The STS-1 reentry trajectory was used as the flight profile experienced by the cone in the MINIVER input file. The STS-1 reentry trajectory in tabular form can be found in Appendix A. Body points were place two feet apart on the surface of the cone, resulting in 30 body points for both the leeward and windward sides. The TPS materials used for the analysis were CFBI, Composite Flexible Blanket Insulation, for the leeward surface and LI-2200, 22 lbm/ft^3 Rigid Ceramic Tile Insulation, for the windward surface. Schematics of the TPS material configurations are shown in Figures 5 and 6.

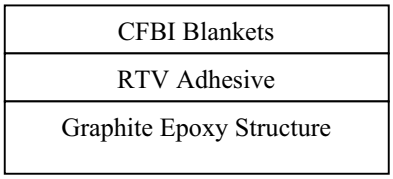

Figure 5: Schematic of CFBI TPS Configuration

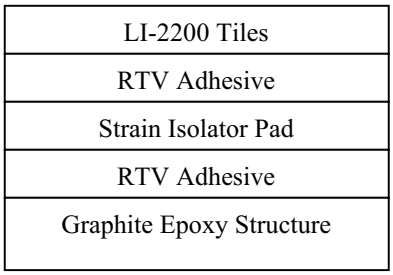

Figure 6: Schematic of LI-2200 TPS Configuration

The results for the aeroheating analysis are given in Tables 1 – 3.

Body Point	Thickness [in.]
pointw1	1.89
pointw2	1.876
pointw3	1.817
pointw4	1.778
pointw5	1.75
pointw6	1.732
pointw7	1.721
pointw8	1.713
pointw9	1.706
pointw10	1.7
pointw11	1.694
pointw12	1.688
pointw13	1.683
pointw14	1.677
pointw15	1.512
pointw16	1.512
pointw17	1.512
pointw18	1.512
pointw19	1.512
pointw20	1.512
pointw21	1.512
pointw22	1.512
pointw23	1.646
pointw24	1.644
pointw25	1.642
pointw26	1.639
pointw27	1.637
pointw28	1.635
pointw29	1.633
pointw30	1.631

Table 1: TPS Thicknesses for Windward Surface of 10° Half-Angle Cone for In-Flight Aeroheating with LI-2200 Tiles

Body Point	Thickness [in.]
pointl1	1.427
pointl2	1.007
pointl3	0.462
pointl4	0.25
pointl5	0.25
pointl6	0.25
pointl7	0.25
pointl8	0.25
pointl9	0.25
pointl10	0.25
pointl11	0.25
pointl12	0.25
pointl13	0.25
pointl14	0.25
pointl15	0.25
pointl16	0.25
pointl17	0.25
pointl18	0.25
pointl19	0.25
pointl20	0.25
pointl21	0.25
pointl22	0.25
pointl23	0.25
pointl24	0.25
pointl25	0.25
pointl26	0.25
pointl27	0.25
pointl28	0.25
pointl29	0.25
pointl30	0.25

Table 2: TPS Thicknesses for Leeward Surface of 10° Half-Angle Cone for In-Flight Aeroheating with CFBI Blankets

	Windward	*Leeward*
TPS Material	LI-2200 Tiles	CFBI Blankets
TPS Unit Weight [lb/ft^2]	3.031	0.16

Table 3: TPS Design Results for 10° Half-Angle Cone for In-Flight Aeroheating

The analysis was then performed using the ground hold analysis tool. For each surface, the TPS was sized for the presence of both a liquid hydrogen (LH2) fuel tank and a liquid oxygen (LOx) fuel tank located inside the nosecap. The parameters of the analysis are given in Table 4 and the results of this analysis are given in Tables 5 and 6.

Ground Hold Time	7800 s (2 hours)
Time Step for Analysis	10 s
Ambient Temperature	70 °F
LOx Fuel Temperature	-297.67 °F
LH2 Fuel Temperature	-423.67 °F
Frosting Condition Temperature	40 °F

Table 4: Ground Hold Analysis Parameters

	Windward	*Leeward*
TPS Material	LI-2200 Tiles	CFBI Blankets
TPS Thickness [in.]	1.438	1.228
TPS Unit Weight [lb/ft^2]	2.6359	0.6138

Table 5: TPS Sizing Results for Ground Hold Analysis Above a LOx Fuel Tank

	Windward	*Leeward*
TPS Material	LI-2200 Tiles	CFBI Blankets
TPS Thickness [in.]	1.796	1.679
TPS Unit Weight [lb/ft^2]	3.2929	0.8393

Table 6: TPS Sizing Results for Ground Hold Analysis Above a LH2 Fuel Tank

As a check to see if the results make physical sense, the temperature profiles for each TPS material configuration were graphed versus time. This will also give an indication whether or not the design constraints have been satisfied. Figures 7 and 8 show the nodal temperature profiles for the LH2 sized TPS materials, and Figures 9 and 10 show the nodal temperature profiles for the LOx sized TPS materials.

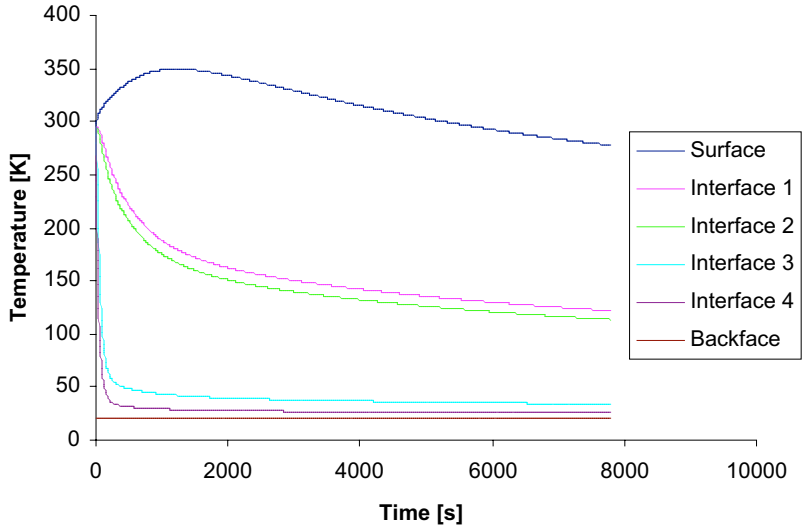

Figure 7: Nodal Temperature Histories for LI-2200 Sized Over LH2 Tank

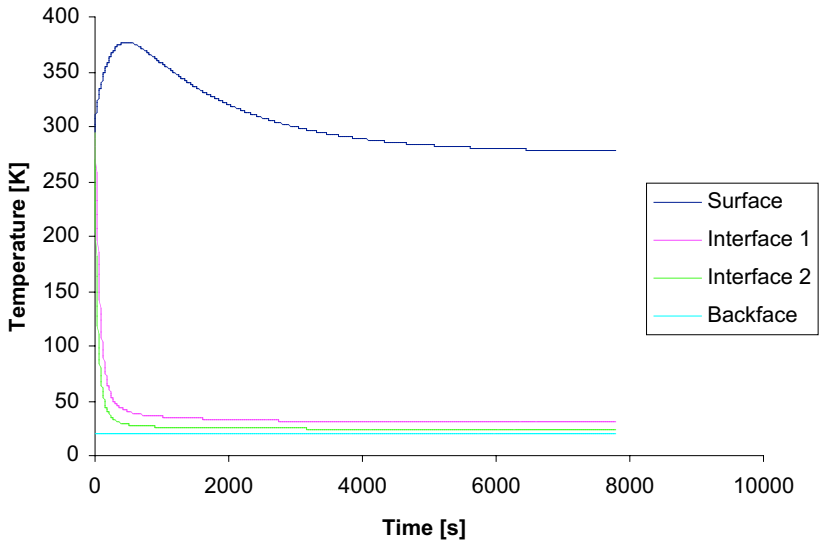

Figure 8: Nodal Temperature Histories for CFBI Sized Over LH2 Tank

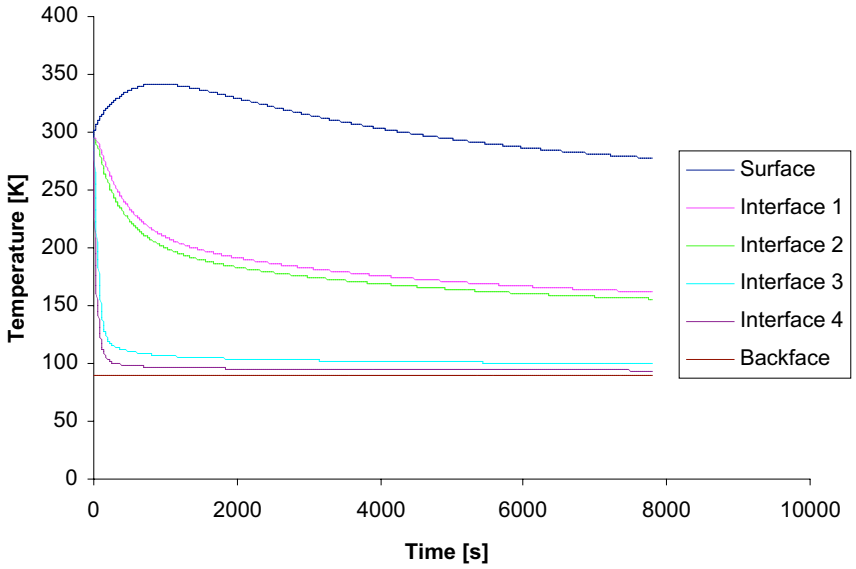

Figure 9: Nodal Temperature Histories for LI-2200 Sized Over LOx Tank

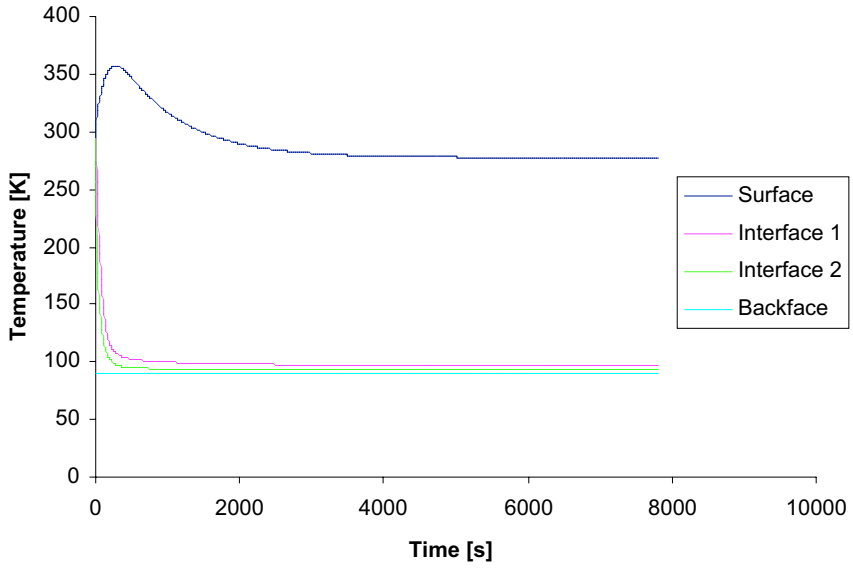

Figure 10: Nodal Temperature Histories for CFBI Sized Over LOx Tank

In each of these nodal temperature histories, we see that the temperature of the surface never drops below 277.6 K (40 °F), which is the surface frosting constraint. Also, the temperature of the backface remains constant and equal to the fuel temperature specified for each case. Therefore, we can be confident that problems constraints have been met. Also, the temperature profiles are all approaching a steady state value at the end of the analysis. The profiles all increase in temperature through the depth of the stack, starting with the lowest temperature at the backface and the highest temperature at the surface. There is also no overlap or crossing of temperature profiles during the analysis. This provides assurance that the numerical method is accurately modeling the physical heat transfer mechanism.

We see that for the windward surface, the unit weight of the LOx sized TPS is lower than the unit weight obtained from the aeroheating analysis. However, the unit weight of the LH2 sized TPS is higher than that of the aeroheating analysis. This means that the TPS sized for aeroheating is sufficiently thick to prevent surface frosting if a LOx tank is located in the nosecap, but it is not thick enough if a LH2 is located in the nosecap.

Now, examining the results of the leeward surface, we see that the unit weights of both the LOx sized and LH2 sized TPS are higher than that of the aeroheating results. If we consider the thicknesses of the TPS alone, we can see that the TPS sized for aeroheating at the very first body point of the leeward side is thick enough to insulate the TPS from the effects of a LOx tank, but still is not thick enough for the LH2 tank. This means that although the TPS may be able to sustain the heat loads experienced by the in-flight aeroheating, the TPS still is not adequately sized for a complete mission profile. If the TPS results from the aeroheating analysis alone were used to construct the vehicle, then the majority of the surface of the vehicle would experience surface frosting. This would result in adverse operating conditions, which could result in TPS damage or possible mission failure.

The increase in unit weights calculated from the ground hold analysis will of course affect the overall dry weight of the vehicle. This could possibly result in a significant increase in cost. Although cost is an important design driver, performance must not be neglected. Therefore, in order to provide the most accurate results and meet

all performance expectations, the TPS should be sized for both ground hold and in-flight aeroheating, with the TPS thicknesses and unit weights satisfying both sets of constraints selected for the final design.

4.0 Crank-Nicolson Method and TCAT

The simple implicit method used in TCAT's aeroheating analysis has been replaced with the Crank-Nicolson finite difference method in order to increase numerical accuracy and to decrease computational time.

Finite Difference Equations

The one-dimensional heat equation discretized at the interior nodes, using the nodal map shown in Figure 11, with a simple implict central finite difference scheme results in equation (14).

$$T_i^n = -\frac{\alpha_i \Delta t}{\Delta x^2}\left(T_{i-1}^{n+1} - 2T_i^{n+1} + T_{i+1}^{n+1}\right) + T_i^{n+1} \tag{14}$$

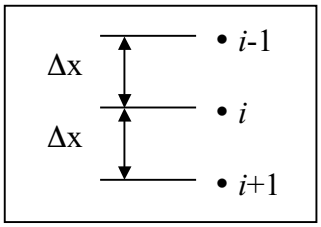

Figure 11: Nodal Mesh Map for Interior Nodes

Using the same nodal mesh map, the discretization of the one-dimensional heat equation with the Crank-Nicolson method results in equation (15).

$$\left(1 - \frac{\alpha_i \Delta t}{\Delta x^2}\right)T_i^n + \frac{\alpha_i \Delta t}{2\Delta x^2}\left(T_{i+1}^n + T_{i-1}^n\right) = -\frac{\alpha_i \Delta t}{2\Delta x^2}\left(T_{i-1}^{n+1} - 2T_i^{n+1} + T_{i+1}^{n+1}\right) + T_i^{n+1} \tag{15}$$

Rearranging this equation such that f is equal to the information at the next time step minus the information at the current time step, we arrive at equation (16).

$$f_i = -\frac{\alpha_i \Delta t}{2\Delta x^2}\left(T_{i-1}^{n+1} - 2T_i^{n+1} + T_{i+1}^{n+1}\right) + T_i^{n+1} - \left(1 - \frac{\alpha_i \Delta t}{\Delta x^2}\right)T_i^n - \frac{\alpha_i \Delta t}{2\Delta x^2}\left(T_{i+1}^n + T_{i-1}^n\right) \quad (16)$$

This equation is now in the same form as the equations used in TCAT's solution process. The resulting system of equations is iteratively solved using the Newton-Raphson method. First, the Jacobian Matrix is formed.

$$J_{i,j} = \frac{\partial f_i}{\partial T_j^{n+1}}_{i,j=1,\ldots,N} \quad (17)$$

The system of equations is now in the form $Ax = b$,given in equation (10), where A is the Jacobian matrix, x is the temperature change at time $n+1$ and $b = -f$ at time n.

$$\begin{bmatrix} \frac{\partial f_1}{\partial T_1^{n+1}} & \frac{\partial f_1}{\partial T_2^{n+1}} & 0 & 0 & 0 & 0 \\ \frac{\partial f_2}{\partial T_1^{n+1}} & \frac{\partial f_2}{\partial T_2^{n+1}} & \frac{\partial f_2}{\partial T_3^{n+1}} & 0 & 0 & 0 \\ 0 & \frac{\partial f_3}{\partial T_2^{n+1}} & \frac{\partial f_3}{\partial T_3^{n+1}} & \frac{\partial f_3}{\partial T_4^{n+1}} & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \frac{\partial f_{N-1}}{\partial T_{N-2}^{n+1}} & \frac{\partial f_{N-1}}{\partial T_{N-1}^{n+1}} & \frac{\partial f_{N-1}}{\partial T_N^{n+1}} \\ 0 & 0 & 0 & 0 & \frac{\partial f_N}{\partial T_{N-1}^{n+1}} & \frac{\partial f_N}{\partial T_N^{n+1}} \end{bmatrix} \begin{bmatrix} \Delta T_1 \\ \Delta T_2 \\ \Delta T_3 \\ \vdots \\ \Delta T_{N-1} \\ \Delta T_N \end{bmatrix}^{n+1} = \begin{bmatrix} -f_1 \\ -f_2 \\ -f_3 \\ \vdots \\ -f_{N-1} \\ -f_N \end{bmatrix}^n \quad (18)$$

Solution of this system is performed by making an initial guess for the temperature at the next time level $n+1$ and iteratively solving for the residual ΔT^{n+1} using the Thomas Algorithm until convergence is achieved.

5.0 World Wide Web Interface

The web version of TCAT has been completely restructured in order to provide a more intuitive user interface. The greeting page of TCAT welcomes the user with an animation in Macromedia Flash format. Following the animation, the user can select the 'TCAT' link that appears in the center of the screen to go to the main content page. It is here that the user can select which version of TCAT to use for the analysis.

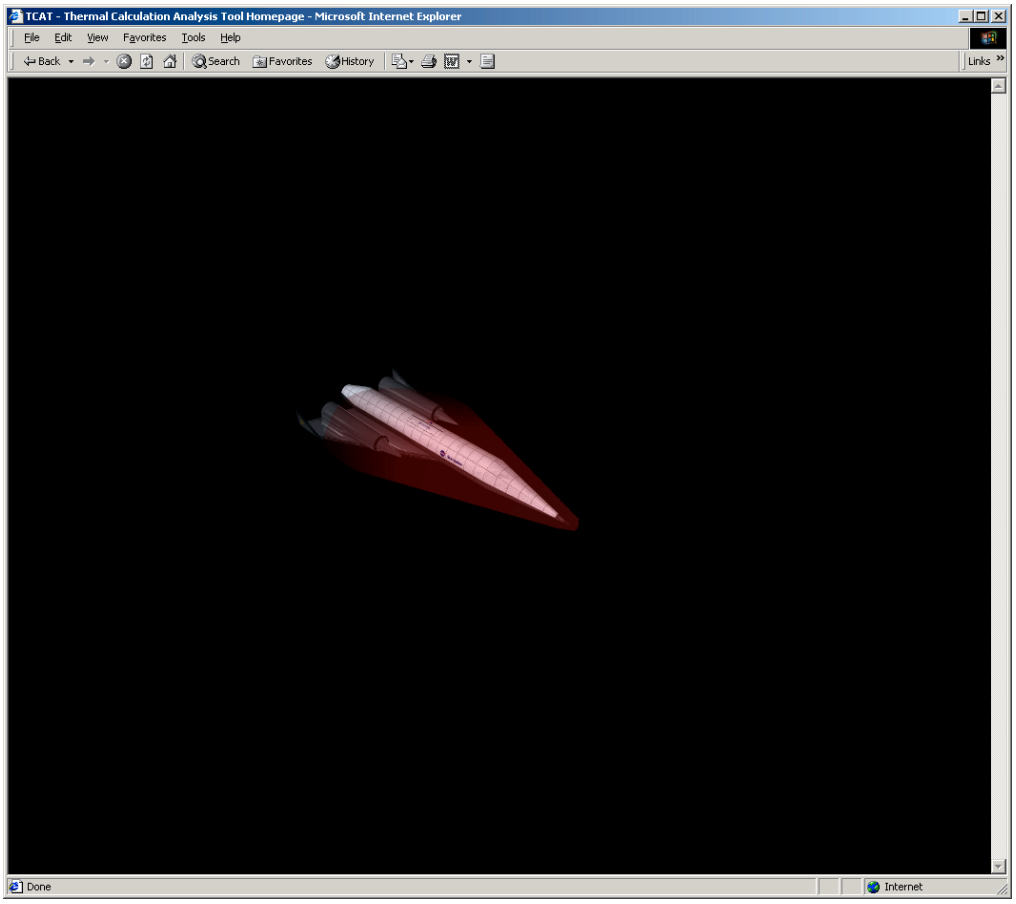

Figure 12: Screen Shot of TCAT Welcome Page

Ground Hold Analysis

If the groundhold analysis is selected, a menu appears in the content frame of the page where the user can choose to analyze a single TPS configuration, or compare three TPS configurations. If the user selects the single configuration analysis, the single analysis page appears in the content window.

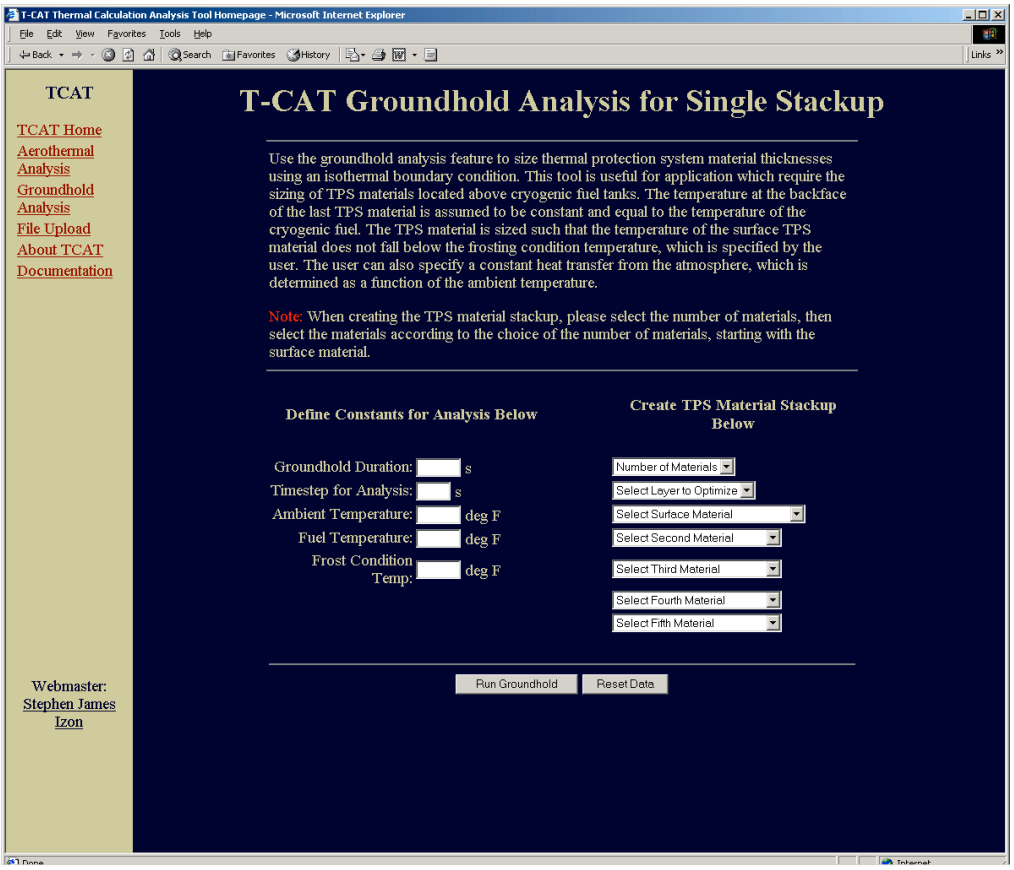

Figure 13: Screen Shot of TCAT Ground Hold Single Analysis Page

The user must input the duration of the groundhold, the time step for the analysis, the ambient temperature of the air during groundhold, the fuel temperature for the isothermal boundary condition, and the surface frosting condition temperature. The user must also build the TPS configuration using the pull-down menus on the right side of the content window. While building the TPS configuration, the user can select the number of materials to use in the configuration. There is also the option to optimize either the first

or the second layer of the TPS configuration. This option is useful because some TPS configurations, such as RCC, SiC, or TUFI are comprised of a thin composite surface layer and a diffusion layer. The surface layer remains the same thickness; however, the diffusion layer is sized for the analysis.

If the user selects the multiple configuration comparison, the multiple analysis page appears in the content window.

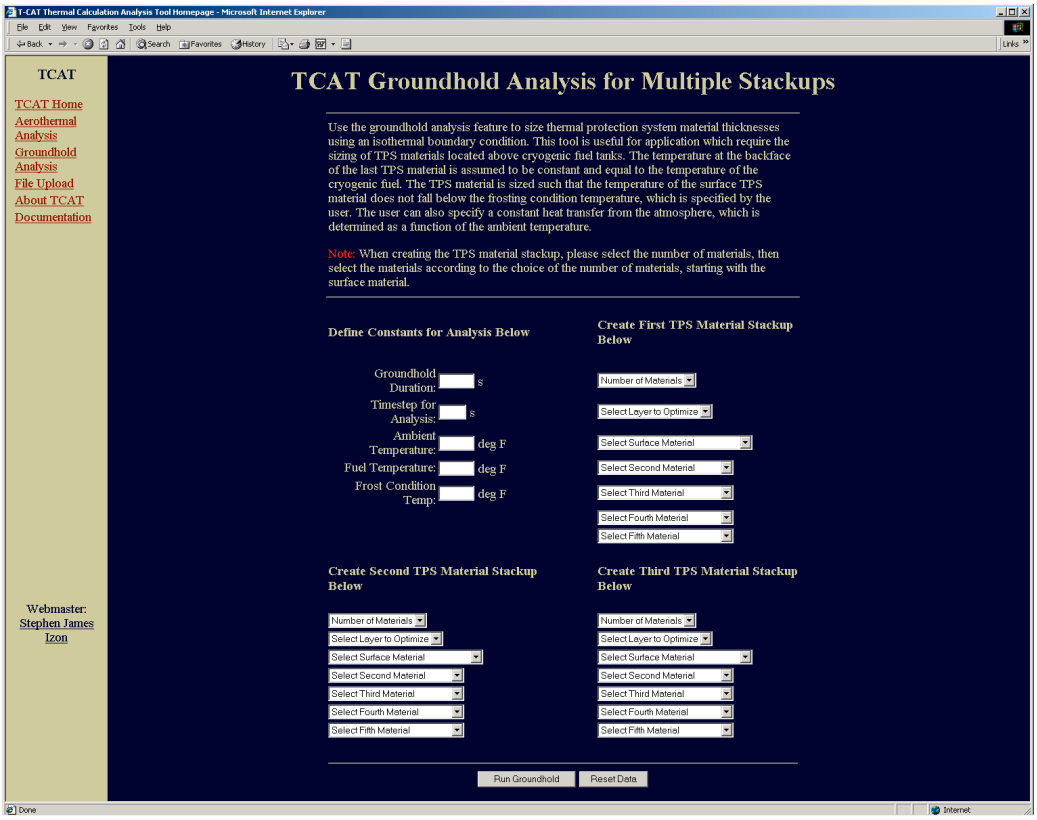

Figure 14: Screen Shot of TCAT Ground Hold Multiple Analysis Page

Here, the user can build up to three different TPS material configurations. This option works in the same way as the single stackup option, the only difference being that there are now three sets of pull down menus in which the user can build their own material stackups. The results for each analysis are stored in different files, which are compared to determine which material stackup option provides the minimum material unit weight for the analysis. The results are then returned to the user in a new window.

Aeroheating Analysis

The web pages for the original TCAT aeroheating program have been integrated into the new TCAT website. This means that both the aeroheating analysis and ground hold analysis can be performed at the same location. The original functionality of TCAT's aeroheating analysis has remained unchanged. A screen shot of TCAT's aeroheating analysis configuration is shown in Figure 15.

Figure 15: Screen Shot of TCAT Aeroheating Analysis Page

MINIVER File Upload

There is now an option available for the user to upload a MINIVER file from their PC to the Space Systems Design Lab's web server. This option was made available because in order to perform aeroheating analyses using TCAT, a MINIVER file is required. To access the file upload interface, the user can simply click on the "File

Upload,, link in the main menu, or go to the aeroheating analysis page where a link to the file upload page is also available.. When the file upload screen appears, the user can then click on the browse button to locate a MINIVER file from their PC's hard drive. The user can specify up to three files to upload at once, with a maximum file size of 1 MB. Once the upload button is pressed, the file is automatically uploaded to the SSDL web server where the aeroheating analysis files are located. The CGI script to perform the file upload was taken from the "Perl Services,, website, which is located at http://www.perlservices.net/en/programs/psupload/index.shtml. The html files associated with the 'upload.cgi' script have been modified to conform to TCAT's site layout.

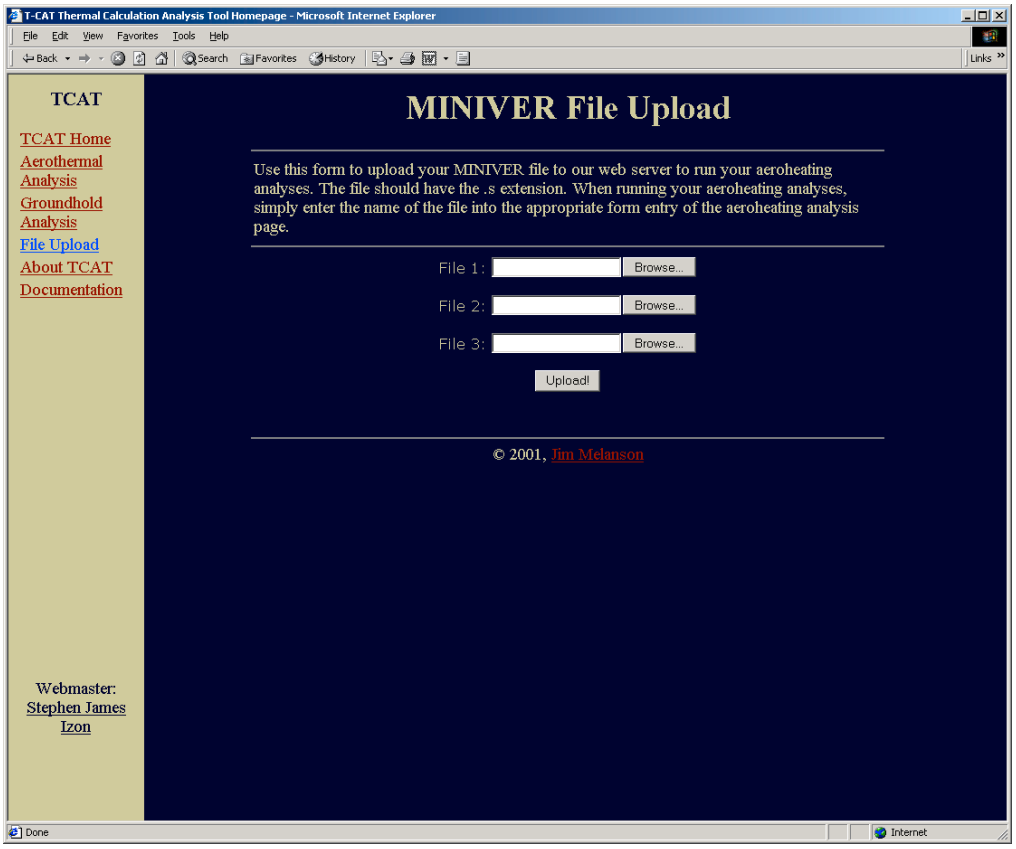

Figure 16: Screen Shot of TCAT MINIVER File Upload Page

6.0 Conclusions

The capabilities of TCAT at its inception, although accurate at calculating TPS material thicknesses when compared to commercial heating codes such as SINDA, did not provide complete results that are required for thermal protection system sizing analyses. TCAT was capable of sizing TPS materials for in-flight aeroheating; however, the capability to size TPS for ground hold operations was still missing. In order to meet this goal, TCAT's capabilities were augmented with the addition of a ground hold analysis tool. This ground hold analysis tool can size the TPS materials for body points located above cryogenic fuel tanks using an isothermal boundary condition at the backface of the fuel tank structure and a constant heat transfer rate from the atmosphere to the surface of the TPS. The results of this analysis tool can be compared to the results given by the aeroheating analysis to determine the TPS thicknesses required at each body point location to meet both the aeroheating and ground hold temperature constraints.

In our test case, we discovered that in some instances the TPS sized for aeroheating may be sufficiently thick to prevent surface frosting. This was evidenced in the case of the TPS sized for the windward side of the 10° half-angle cone with a LOx tank located inside the cone. However, for the rest of the analyses it was determined that the TPS sized for aeroheating were not sufficient to meet the constraints of the ground hold. Therefore, the design tool created specifically for the sizing of TPS during ground hold should be integrated into the TPS design process.

Even with the ground hold analysis tool developed, TCAT's capabilities are not yet entirely developed. A method of integrating the ground hold and aeroheating analysis should be developed so that the TPS thicknesses of each individual analysis do not need to be compared on a point-by-point basis. This could be achieved by integrating the ground hold temperature and heat transfer profiles for each body point into the MINIVER file that includes the radiation equilibrium temperature and heat transfer profiles during in-flight trajectory. The entire analysis could be performed using this single input file, with the method of analysis switching from ground hold constraints to in-flight aeroheating constraints, returning the optimal TPS thickness that satisfies both sets of constraints to the user.

Appendix A: STS-1 Reentry Trajectory

Time (s)	Altitude (ft)	Velocity (ft/s)	Angle of Attack	Bank Angle
45.3	373800	24590	41.26	0
90.3	351500	24620	41.21	0
135.3	329500	24640	40.46	0
180.3	308000	24660	40.65	0
225.3	288000	24660	41.9	0
270.3	269700	24610	39.39	0
315.3	256300	24470	40.71	0
360.3	250200	24220	41.74	0
405.3	247000	23920	39.95	0
450.3	244600	23610	39.28	0
495.3	242500	23280	39.66	0
545.3	240000	22880	39.07	0
569.3	238800	22680	39.32	0
593.3	237700	22470	39.02	0
617.3	236600	22250	39.25	0
641.3	235300	22020	39.48	0
665.3	233700	21780	39.97	0
689.3	232000	21530	40.01	0
713.3	232300	21260	40.25	0
737.3	230300	20980	40.46	0
761.3	227900	20680	40.14	0
785.3	225200	20360	40.16	0
809.3	223000	20020	40.16	0
833.3	219000	19650	40.27	0
857.3	215300	19240	40.4	0
881.3	211400	18790	40.12	0
905.3	207600	18300	40.3	0
929.3	205600	17760	40.01	0
953.3	202500	17180	42	0
977.3	197200	16540	40.9	0
1001	192200	15870	40.75	0
1025	187200	15130	39.87	0
1049	182600	14350	39.43	0
1073	179600	13570	39.53	0
1097	176600	12790	39.75	0
1121	172500	12010	38.96	0
1145	167500	11220	38.12	0
1169	161900	10440	36.97	0
1240	150000	8336	34.07	0
1302	133900	6757	28.2	0
1364	117400	5342	23.05	0
1426	106200	4064	20.27	0
1488	89100	2917	16.47	0
1550	76350	1915	10.86	0
1612	57440	1151	7.79	0
1674	41670	800	7.72	0
1736	27600	681	7.09	0
1860	3337	512	3.78	0
1925	-3	202	-12.2	0

Appendix B: Windward Aeroheating Analysis Input File

```
10          !/time step
48          !/number of nodes of material 1
12          !/number of nodes of material 2
12          !/number of nodes of material 3
12          !/number of nodes of material 4
12          !/number of nodes of material 5
0.899       !/emissivity of material 1
0.074420    !/thermal conductivity of material 1
0.396700    !/thermal conductivity of material 2
0.042000    !/thermal conductivity of material 3
0.396700    !/thermal conductivity of material 4
0.601300    !/thermal conductivity of material 5
352.400000  !/density of material 1
1410.000000 !/density of material 2
86.500600   !/density of material 3
1410.000000 !/density of material 4
1576.232900 !/density of material 5
0.628000    !/specific heat of material 1
1.168000    !/specific heat of material 2
1.318000    !/specific heat of material 3
1.168000    !/specific heat of material 4
0.833300    !/specific heat of material 5
294.2611111 !/initial temp (deg K)
0.075000    !/TPS thickness of material 1
0.003000    !/TPS thickness of material 2
0.004000    !/TPS thickness of material 3
0.003000    !/TPS thickness of material 4
0.003000    !/TPS thickness of material 5
277.5944444 !/Temperature limit of material 1
1644        !/Temperature limit of material 1
561.100000  !/Temperature limit of material 2
561.100000  !/Temperature limit of material 2
505.222000  !/Temperature limit of material 3
505.222000  !/Temperature limit of material 3
561.100000  !/Temperature limit of material 4
561.100000  !/Temperature limit of material 4
588.889     !/Temperature limit of material 5
19.99999999 !/backface temperature limit
6           !/istrat parameter
5           !/iopt parameter
5           !/ioned parameter
1340        !/iprint parameter
```

Appendix C: Leeward Aeroheating Analysis Input File

```
3           !/number of materials
10.         !/time step
40          !/number of nodes in material 1
10          !/number of nodes in material 2
10          !/number of nodes in material 3
0.8856      !/emissivity of material 1
0.03821     !/thermal conductivity of material 1
0.3967      !/thermal conductivity of material 2
0.6013      !/thermal conductivity of material 3
96.118      !/density of material 1
1410        !/density of material 2
1576.2329   !/density of material 3
0.7821      !/specific heat of material 1
1.168       !/specific heat of material 2
0.8333      !/specific heat of material 3
300.        !/initial temp (deg K)
0.05        !/TPS thickness of material 1 (meters)
0.003       !/TPS thickness of material 2 (meters)
0.003       !/TPS thickness of material 3 (meters)
1477.44     !/Temperature limit of material 1
1477.44     !/Temperature limit of material 1
561.1       !/Temperature limit of material 2
561.1       !/Temperature limit of material 2
588.889     !/Temperature limit of material 3
421.889     !/backface temperature limit
6           !/istrat parameter
5           !/iopt parameter
5           !/ioned parameter
0000        !/iprint parameter
```

Appendix D: Windward Ground Hold Analysis for LOx Tank Input File

```
5           !/number of materials
10          !/time step
48          !/number of nodes of material 1
12          !/number of nodes of material 2
12          !/number of nodes of material 3
12          !/number of nodes of material 4
12          !/number of nodes of material 5
0.899       !/emissivity of material 1
0.074420    !/thermal conductivity of material 1
0.396700    !/thermal conductivity of material 2
0.042000    !/thermal conductivity of material 3
0.396700    !/thermal conductivity of material 4
0.601300    !/thermal conductivity of material 5
352.400000  !/density of material 1
1410.000000 !/density of material 2
86.500600   !/density of material 3
1410.000000 !/density of material 4
1576.232900 !/density of material 5
0.628000    !/specific heat of material 1
1.168000    !/specific heat of material 2
1.318000    !/specific heat of material 3
1.168000    !/specific heat of material 4
0.833300    !/specific heat of material 5
294.2611111 !/initial temp (deg K)
0.075000    !/TPS thickness of material 1
0.003000    !/TPS thickness of material 2
0.004000    !/TPS thickness of material 3
0.003000    !/TPS thickness of material 4
0.003000    !/TPS thickness of material 5
277.5944444 !/Temperature limit of material 1
1644        !/Temperature limit of material 1
561.100000  !/Temperature limit of material 2
561.100000  !/Temperature limit of material 2
505.222000  !/Temperature limit of material 3
505.222000  !/Temperature limit of material 3
561.100000  !/Temperature limit of material 4
561.100000  !/Temperature limit of material 4
588.889     !/Temperature limit of material 5
90          !/backface temperature limit
6           !/istrat parameter
5           !/iopt parameter
5           !/ioned parameter
1340        !/iprint parameter
```

Appendix E: Leeward Ground Hold Analysis for LOx Tank Input File

```
3           !/number of materials
10          !/time step
40          !/number of nodes of material 1
10          !/number of nodes of material 2
10          !/number of nodes of material 3
0.8856      !/emissivity of material 1
0.038210    !/thermal conductivity of material 1
0.396700    !/thermal conductivity of material 2
0.601300    !/thermal conductivity of material 3
96.118000   !/density of material 1
1410.000000 !/density of material 2
1576.232900 !/density of material 3
0.782100    !/specific heat of material 1
1.168000    !/specific heat of material 2
0.833300    !/specific heat of material 3
294.2611111 !/initial temp (deg K)
0.050000    !/TPS thickness of material 1
0.003000    !/TPS thickness of material 2
0.003000    !/TPS thickness of material 3
277.5944444 !/Temperature limit of material 1
1477.44     !/Temperature limit of material 1
561.100000  !/Temperature limit of material 2
561.100000  !/Temperature limit of material 2
588.889     !/Temperature limit of material 3
90          !/backface temperature limit
6           !/istrat parameter
5           !/iopt parameter
5           !/ioned parameter
1340        !/iprint parameter
```

Appendix F: Windward Ground Hold Analysis for LH2 Tank Input File

```
5           !/number of materials
10          !/time step
48          !/number of nodes of material 1
12          !/number of nodes of material 2
12          !/number of nodes of material 3
12          !/number of nodes of material 4
12          !/number of nodes of material 5
0.899       !/emissivity of material 1
0.074420    !/thermal conductivity of material 1
0.396700    !/thermal conductivity of material 2
0.042000    !/thermal conductivity of material 3
0.396700    !/thermal conductivity of material 4
0.601300    !/thermal conductivity of material 5
352.400000  !/density of material 1
1410.000000 !/density of material 2
86.500600   !/density of material 3
1410.000000 !/density of material 4
1576.232900 !/density of material 5
0.628000    !/specific heat of material 1
1.168000    !/specific heat of material 2
1.318000    !/specific heat of material 3
1.168000    !/specific heat of material 4
0.833300    !/specific heat of material 5
294.2611111 !/initial temp (deg K)
0.075000    !/TPS thickness of material 1
0.003000    !/TPS thickness of material 2
0.004000    !/TPS thickness of material 3
0.003000    !/TPS thickness of material 4
0.003000    !/TPS thickness of material 5
277.5944444 !/Temperature limit of material 1
1644        !/Temperature limit of material 1
561.100000  !/Temperature limit of material 2
561.100000  !/Temperature limit of material 2
505.222000  !/Temperature limit of material 3
505.222000  !/Temperature limit of material 3
561.100000  !/Temperature limit of material 4
561.100000  !/Temperature limit of material 4
588.889     !/Temperature limit of material 5
19.99999999 !/backface temperature limit
6           !/istrat parameter
5           !/iopt parameter
5           !/ioned parameter
1340        !/iprint parameter
```

Appendix G: Leeward Ground Hold Analysis for LH2 Tank Input File

```
3           !/number of materials
10          !/time step
40          !/number of nodes of material 1
10          !/number of nodes of material 2
10          !/number of nodes of material 3
0.8856      !/emissivity of material 1
0.038210    !/thermal conductivity of material 1
0.396700    !/thermal conductivity of material 2
0.601300    !/thermal conductivity of material 3
96.118000   !/density of material 1
1410.000000 !/density of material 2
1576.232900 !/density of material 3
0.782100    !/specific heat of material 1
1.168000    !/specific heat of material 2
0.833300    !/specific heat of material 3
294.2611111 !/initial temp (deg K)
0.050000    !/TPS thickness of material 1
0.003000    !/TPS thickness of material 2
0.003000    !/TPS thickness of material 3
277.5944444 !/Temperature limit of material 1
1477.44     !/Temperature limit of material 1
561.100000  !/Temperature limit of material 2
561.100000  !/Temperature limit of material 2
588.889     !/Temperature limit of material 3
19.99999999 !/backface temperature limit
6           !/istrat parameter
5           !/iopt parameter
5           !/ioned parameter
1340        !/iprint parameter
```

Appendix H: 'single.cgi' Ground Hold Single Analysis CGI Script

```
#!/usr/local/bin/perl

read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)
{
        ($name, $value) = split(/=/, $pair);
        $value =~ tr/+/ /;
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
        $value =~ s/~!/ ~!/g;
        $FORM{$name} = $value;
}

# ------------------------------------------------------------------------------
# assign information from input website to appropriate variable
# ------------------------------------------------------------------------------

 $matnum=$FORM{matnum};
 $method=$FORM{method};
 $material_1=$FORM{material_1};
 $material_2=$FORM{material_2};
 $material_3=$FORM{material_3};
 $material_4=$FORM{material_4};
 $material_5=$FORM{material_5};
 $holdtime=$FORM{holdtime};
 $timestep=$FORM{timestep};
 $ambient=$FORM{ambient};
 $fueltemp=$FORM{fueltemp};
 $frost=$FORM{frost};

# ------------------------------------------------------------------------------
# obtain inputs from command line
#
# List of variables
# $bodyarea - surface area of the body in square feet
# $bodytile - fraction of bodyarea covered with tiles
# $bodyblanket - fraction of bodyarea covered with blankets
# $filename - name of MINIVER l_<>,s file
# $sharp - indicates the use of SHARP materials (yes or no)
# $bodyside - vehicle side analysis is conducted on (leeward or windward)
# $material - type of material to be used along with backface material
# ------------------------------------------------------------------------------

################################################################################
#####
# Prompt the user for inputs.
#
################################################################################
#####

#$holdtime=7800;
#$timestep=10;
#$heatrate=1;
#$ambient=70;
#$fueltemp=-423.67;
#$frost=40;

#print "Enter the amount of time, in seconds, the vehicle will be in groundhold.\n";
#$holdtime = readline(*STDIN);
#chop($holdtime);

#print "Enter the timestep, in seconds, for the analysis.\n";
#$timestep = readline(*STDIN);
#chop($timestep);

#print "Enter the heat transfer coefficient [BTU/(ft^2*hour*F)].\n";
#$heatrate = readline(*STDIN);
#chop($heatrate);
```

```
#print "Enter the ambient temperature [F] of the groundhold location.\n";
#$ambient = readline(*STDIN);
#chop($ambient);

#print "Enter the temperature [F] of the cryogenic fuel (ie. the isothermal boundary
conidtion).\n";
#$fueltemp = readline(*STDIN);
#chop($fueltemp);

#print "Enter the frosting condition Temperature [F] of the surface TPS material.\n";
#$frost = readline(*STDIN);
#chop($frost);

open(TIME,">time");
print TIME "$holdtime         !/Time of Groundhold";
close(TIME);

$heatrate = 1;

$heattransfer = $heatrate * $ambient *3.15459074506;

open(HEATXFER,">heat");
print HEATXFER "$heattransfer         !/Heat Flux";
close(HEATXFER);

$ambient = ($ambient-32)*(5/9)+273.15;

$frost = ($frost-32)*(5/9)+273.15;

$fueltemp = ($fueltemp-32)*(5/9)+273.15;

open(TANK,">fueltemp");
print TANK "$fueltemp               !/Temperature of fuel";
close(TANK);

#print "Please enter\n";
#print "       1. if you would like to create a new TPS material stackup for the
analysis, or\n";
#print "       2. if you would like to use an existing TPS material stackup\n";
#$material_option = readline(*STDIN);
#chop($material_option);

#print "Please enter the number of materials in the TPS stackup (up to 5).\n";
#$matnum = readline(*STDIN);
#chop($matnum);

#if ($material_option eq 1) {

#   for ($loopnum=1; $loopnum <= $matnum; $loopnum++) {

#      if ($loopnum eq 1) {

#         print "Please select the surface material of the TPS stackup.\n";
#         print "\n";
#         print "-----Laminate Surfaces-----\n";
#         print "RCC_Laminate_Surface\n";
#         print "SiC_Laminate_Surface\n";
#         print "TUFI_Composite_Surface\n";
#         print "---------- Tiles ---------.\n";
#         print "TUFI_AETB8_Diffusion_Layer\n";
#         print "TUFI_AETB12_Diffusion_Layer\n";
#         print "FRCI12\n";
#         print "FRCI20\n";
#         print "LI900\n";
#         print "LI2200\n";
#         print "---------- Blankets ----------\n";
#         print "AFRSI\n";
#         print "CFBI\n";
#         print "DURAFRSI\n";
#         print "PBI\n";
```

```
#         print "TABI\n";
#         print "\n";
#         print "Enter the material name as it appears above.\n";
#         $material_1 = readline(*STDIN);
#         chop($material_1);

#      }  elsif ($loopnum eq 2) {

#         print "Please select second material of the TPS stackup.\n";
#         print "\n";
#         print "---------- Tiles ---------.\n";
#         print "TUFI_AETB8_Diffusion_Layer\n";
#         print "TUFI_AETB12_Diffusion_Layer\n";
#         print "FRCI12\n";
#         print "FRCI20\n";
#         print "LI900\n";
#         print "LI2200\n";
#         print "RCC\n";
#         print "SiC\n";
#         print "---------- Blankets ----------\n";
#         print "AFRSI\n";
#         print "CFBI\n";
#         print "DURAFRSI\n";
#         print "PBI\n";
#         print "TABI\n";
#         print "-------- Intermediates -------\n";
#         print "Strain_Isolator_Pad\n";
#         print "RTV_Adhesive\n";
#         print "Rohacell_Foam\n";
#         print "---------- Structures --------\n";
#         print "GrEx_Structure\n";
#         print "Al_Structure\n";
#         print "TiAl_Structure\n";
#         print "\n";
#         print "Enter the material name as it appears above.\n";
#         $material_2 = readline(*STDIN);
#         chop($material_2);

#      }  elsif ($loopnum eq 3) {

#         print "Please select third material of the TPS stackup.\n";
#         print "\n";
#         print "---------- Tiles ---------.\n";
#         print "TUFI_AETB8_Diffusion_Layer\n";
#         print "TUFI_AETB12_Diffusion_Layer\n";
#         print "FRCI12\n";
#         print "FRCI20\n";
#         print "LI900\n";
#         print "LI2200\n";
#         print "RCC\n";
#         print "SiC\n";
#         print "---------- Blankets ----------\n";
#         print "AFRSI\n";
#         print "CFBI\n";
#         print "DURAFRSI\n";
#         print "PBI\n";
#         print "TABI\n";
#         print "-------- Intermediates -------\n";
#         print "Strain_Isolator_Pad\n";
#         print "RTV_Adhesive\n";
#         print "Rohacell_Foam\n";
#         print "---------- Structures --------\n";
#         print "GrEx_Structure\n";
#         print "Al_Structure\n";
#         print "TiAl_Structure\n";
#         print "\n";
#         print "Enter the material name as it appears above.\n";
#         $material_3 = readline(*STDIN);
#         chop($material_3);

#      }  elsif ($loopnum eq 4) {
```

```
#         print "Please select fourth material of the TPS stackup.\n";
#         print "\n";
#         print "---------- Tiles ---------.\n";
#         print "TUFI_AETB8_Diffusion_Layer\n";
#         print "TUFI_AETB12_Diffusion_Layer\n";
#         print "FRCI12\n";
#         print "FRCI20\n";
#         print "LI900\n";
#         print "LI2200\n";
#         print "RCC\n";
#         print "SiC\n";
#         print "---------- Blankets ----------\n";
#         print "AFRSI\n";
#         print "CFBI\n";
#         print "DURAFRSI\n";
#         print "PBI\n";
#         print "TABI\n";
#         print "-------- Intermediates -------\n";
#         print "Strain_Isolator_Pad\n";
#         print "RTV_Adhesive\n";
#         print "Rohacell_Foam\n";
#         print "---------- Structures --------\n";
#         print "GrEx_Structure\n";
#         print "Al_Structure\n";
#         print "TiAl_Structure\n";
#         print "\n";
#         print "Enter the material name as it appears above.\n";
#         $material_4 = readline(*STDIN);
#         chop($material_4);

#      }  elsif ($loopnum eq 5) {

#         print "Please select fifth material of the TPS stackup.\n";
#         print "\n";
#         print "---------- Tiles ---------.\n";
#         print "TUFI_AETB8_Diffusion_Layer\n";
#         print "TUFI_AETB12_Diffusion_Layer\n";
#         print "FRCI12\n";
#         print "FRCI20\n";
#         print "LI900\n";
#         print "LI2200\n";
#         print "RCC\n";
#         print "SiC\n";
#         print "---------- Blankets ----------\n";
#         print "AFRSI\n";
#         print "CFBI\n";
#         print "DURAFRSI\n";
#         print "PBI\n";
#         print "TABI\n";
#         print "-------- Intermediates -------\n";
#         print "Strain_Isolator_Pad\n";
#         print "RTV_Adhesive\n";
#         print "Rohacell_Foam\n";
#         print "---------- Structures --------\n";
#         print "GrEx_Structure\n";
#         print "Al_Structure\n";
#         print "TiAl_Structure\n";
#         print "\n";
#         print "Enter the material name as it appears above.\n";
#         $material_5 = readline(*STDIN);
#         chop($material_5);

#      }

#   }

#}

#print "\n";
#print "Please enter the optimization method.\n";
```

```
#print " 1. If you would like to optimize the surface material thickness, or\n";
#print " 2. If you would like to optimize the second TPS material thickness.\n";
#print "\n";
#print "NOTE: If either of the laminate materials were selected as the\n";
#print "      surface TPS material, the second optimization method will\n";
#print "      be automatically selected.\n";
#$method = readline(*STDIN);
#chop($method);
#print "\n";

$material_option=1;
#$matnum=5;
#$material_1=AFRSI2500;
#$material_1=TUFI_AETB12_Diffusion_Layer;
#$material_2=RTV_Adhesive;
#$material_3=GrEx_Structure;

#$material_3=Strain_Isolator_Pad;
#$material_4=Rohacell_Foam;
#$material_5=GrEx_Structure;

if ($material_option eq 1) {

   for ($loopnum=1; $loopnum <= $matnum; $loopnum++) {

      if ($loopnum eq 1) {

         open(DAT,"<$material_1");

         $i = 0;

         while(<DAT>) {
            $i = $i + 1;
            my($line) = $_;
            chomp($line);

            if ($i eq 1) {
               $nodes[1] = $line;
            }  elsif ($i eq 2) {
               $cond[1] = $line;
            }  elsif ($i eq 3) {
               $density[1] = $line;
            }  elsif ($i eq 4) {
               $specheat[1] = $line;
            }  elsif ($i eq 5) {
               $thick[1] = $line;
            }  elsif ($i eq 6) {
               $templimit[1]=$line;
            }  elsif ($i eq 7) {
               $emissivity=$line;
            }
         }

         close(DAT);

      }  elsif ($loopnum eq 2) {

         open(DAT,"<$material_2");

         $i = 0;

         while(<DAT>) {
            $i = $i + 1;
            my($line) = $_;
            chomp($line);

            if ($i eq 1) {
               $nodes[2] = $line;
            }  elsif ($i eq 2) {
               $cond[2] = $line;
            }  elsif ($i eq 3) {
```

```
            $density[2] = $line;
         }  elsif ($i eq 4) {
            $specheat[2] = $line;
         }  elsif ($i eq 5) {
            $thick[2] = $line;
         }  elsif ($i eq 6) {
            $templimit[2]=$line;
         }

      }

      close(DAT);

   }  elsif ($loopnum eq 3) {

      open(DAT,"<$material_3");

      $i = 0;

      while(<DAT>) {
         $i = $i + 1;
         my($line) = $_;
         chomp($line);

         if ($i eq 1) {
            $nodes[3] = $line;
         }  elsif ($i eq 2) {
            $cond[3] = $line;
         }  elsif ($i eq 3) {
            $density[3] = $line;
         }  elsif ($i eq 4) {
            $specheat[3] = $line;
         }  elsif ($i eq 5) {
            $thick[3] = $line;
         }  elsif ($i eq 6) {
            $templimit[3]=$line;
         }

      }

      close(DAT);

   }  elsif ($loopnum eq 4) {

      open(DAT,"<$material_4");

      $i = 0;

      while(<DAT>) {
         $i = $i + 1;
         my($line) = $_;
         chomp($line);

         if ($i eq 1) {
            $nodes[4] = $line;
         }  elsif ($i eq 2) {
            $cond[4] = $line;
         }  elsif ($i eq 3) {
            $density[4] = $line;
         }  elsif ($i eq 4) {
            $specheat[4] = $line;
         }  elsif ($i eq 5) {
            $thick[4] = $line;
         }  elsif ($i eq 6) {
            $templimit[4]=$line;
         }

      }

      close(DAT);
```

```
      }  elsif ($loopnum eq 5) {

         open(DAT,"<$material_5");

         $i = 0;

         while(<DAT>) {
            $i = $i + 1;
            my($line) = $_;
            chomp($line);

            if ($i eq 1) {
               $nodes[5] = $line;
            }  elsif ($i eq 2) {
               $cond[5] = $line;
            }  elsif ($i eq 3) {
               $density[5] = $line;
            }  elsif ($i eq 4) {
               $specheat[5] = $line;
            }  elsif ($i eq 5) {
               $thick[5] = $line;
            }  elsif ($i eq 6) {
               $templimit[5]=$line;
            }

         }

         close(DAT);

      }

   }

##############################################################################################
##########
# Create the TPS stackup file by reading the properties of each material selected and
entering    #
# their values into a input file recognizable by T-CAT.
#
##############################################################################################
##########

#   system("touch stackup.in");          #create TPS stackup file
#   system("chmod 777 stackup.in");      #change permissions of TPS stackup file
   open(STACKUP,">stackup.in");
   print STACKUP "$matnum              !/number of materials\n";
   print STACKUP "$timestep            !/time step\n";


   for ($loopnum=1; $loopnum <= $matnum; $loopnum++) {
       printf STACKUP "%i              !/number of nodes of material
$loopnum\n",$nodes[$loopnum];
   }

   print STACKUP "$emissivity          !/emissivity of material 1\n";

   for ($loopnum=1; $loopnum <= $matnum; $loopnum++) {
       printf STACKUP "%f      !/thermal conductivity of material
$loopnum\n",$cond[$loopnum];
   }

   for ($loopnum=1; $loopnum <= $matnum; $loopnum++) {
       printf STACKUP "%f      !/density of material $loopnum\n",$density[$loopnum];
   }

   for ($loopnum=1; $loopnum <= $matnum; $loopnum++) {
       printf STACKUP "%f      !/specific heat of material
$loopnum\n",$specheat[$loopnum];
   }

   print STACKUP "$ambient             !/initial temp (deg K)\n";
```

```
   for ($loopnum=1; $loopnum <= $matnum; $loopnum++) {
       printf STACKUP "%f     !/TPS thickness of material $loopnum\n",$thick[$loopnum];
   }

   print STACKUP "$frost              !/Temperature limit of material 1\n";
   print STACKUP "$templimit[1]              !/Temperature limit of material 1\n";

   for ($loopnum=2; $loopnum <= $matnum-1; $loopnum++) {
       printf STACKUP "%f     !/Temperature limit of material
$loopnum\n",$templimit[$loopnum];
       printf STACKUP "%f     !/Temperature limit of material
$loopnum\n",$templimit[$loopnum];
   }

   print STACKUP "$templimit[$matnum]         !/Temperature limit of material $matnum\n";
   print STACKUP "$fueltemp           !/backface temperature limit\n";
   print STACKUP "6           !/istrat parameter\n";
   print STACKUP "5           !/iopt parameter\n";
   print STACKUP "5           !/ioned parameter\n";
   print STACKUP "1340        !/iprint parameter\n";

   close(STACKUP);
#   system("chmod 777 stackup.in");

}
#}  else {


#   print "Enter the material for TPS.\n";
#   print "---------- Tiles ---------.\n";
#   print "AETB8_Al\n";
#   print "AETB8_GrEx\n";
#   print "AETB12_Al\n";
#   print "AETB12_GrEx\n";
#   print "FRCI12_Al\n";
#   print "FRCI12_GrEx\n";
#   print "FRCI20_Al\n";
#   print "FRCI20_GrEx\n";
#   print "LI900_Al\n";
#   print "LI900_GrEx\n";
#   print "LI2200_Al\n";
#   print "LI2200_GrEx\n";
#   print "RCC_Al\n";
#   print "RCC_GrEx\n";
#   print "SiC_Al\n";
#   print "SiC_GrEx\n";
#   print "RCC_Al\n";
#   print "RCC_GrEx\n";
#   print "TUFI_Al\n";
#   print "TUFI_GrEx\n";
#   print "---------- Blankets ----------\n";
#   print "AFRSI_Al\n";
#   print "AFRSI_GrEx\n";
#   print "CFBI_Al\n";
#   print "CFBI_GrEx\n";
#   print "DURAFRSI_Al\n";
#   print "DURAFRSI_GrEx\n";
#   print "PBI_Al\n";
#   print "PBI_GrEx\n";
#   print "TABI_Al\n";
#   print "TABI_GrEx\n";
#   print "\n";
#   print "Enter the material name as it appears above.\n";
#   $material = readline(*STDIN);
#   chop($material);

#}

################################################################################################
############
```

```
# This section of the program creates a mock MINIVER file.  The three columns of data
created are   #
# the time, ambient temperature, and rate of heat transfer.
#
##############################################################################################
############

$numtimes = $holdtime/$timestep + 1;

$time[1] = 0;

for ($i=2; $i <= $numtimes; $i++) {

   $time[$i] = $time[$i-1] + $timestep;

}

#system("touch l_stackup.s");        #create mock MINIVER file
#system("chmod 777 l_stackup.s");     #change permissions of file
open(MINI,">l_stackup.s");
print MINI " Mock MINIVER file for input into ground hold analysis                    10
1.0\n";

for ($i=1; $i <= $numtimes; $i++) {

   print MINI "$time[$i] $ambient $heattransfer\n";

}

print MINI "-100.0";
close(MINI);

# ------------------------------------------------------------------------------------
# MINIVER file is partitioned into body point files
# partition the MINIVER file into separate body point files that include time,
# heat rate, and radiation equilibrium temperature
# ------------------------------------------------------------------------------------

$i=0;                                #initialization of counter index "i"
$filecount=0;
open(FILE,"<l_stackup.s");           #open statement with filehandle FILE

until(eof FILE) {                    #goes through the MINVER file until the end is
reached
   START: $_=<FILE>;                 #skips the first line of the data file
   if (/^.{73}(\S*)/) {                      #searches for the bodypoint label
   $title=$1;

#   system("touch groundhold");              #create body point file
#   system("chmod 777 groundhold");   #change permissions inorder to write to body point
file

   $filecount=$filecount+1;
   } # end of if statement

   while(<FILE>) {                   #while loop that reads thru each line of the file
      $i=$i+1;

/^\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*
)\s*(\S*)\s*(\S*)/;
      $time[$i]=$1;                  #array declaration for time
      $tradeq[$i]=$2;                #array declaration for rad eq temp
      $qconv[$i]=$3;                 #array declaration for conv heat rate
      $n=$i;
      if ($1==-100) {
         open(NEWFILE,">groundhold"); #open statement for new file "newsample"
         $timesteps=$n-1;
         printf NEWFILE "$timesteps \n";     #formatted print statement"
         for($i=1; $i<$n; $i++) {
            printf NEWFILE "%f \n",$time[$i];
            printf NEWFILE "%f \n",$qconv[$i];
```

```
            printf NEWFILE "%f \n",$tradeq[$i];
         }  #end of for loop

         close(NEWFILE);
         $i=0;
         goto START
      }  #end of if statement

   }  #end of while loop
}  #end of until loop
close(FILE);

# ------------------------------------------------------------------------------------
# incrementally move each body point file into "miniver.in" and execute tcat
# perl script that creates the filenames needed inorder to batch execute tcat
# filename are created via input from the user
# inputs (file_start_number, file_end_number, body_side)
# ------------------------------------------------------------------------------------

$start=1;              #start of the file index
#$end=($filecount-1)/2;       #end of the file index
$end=1;

open(FILE,">inputs_for_outputs");
printf FILE "1\n";
close(FILE);

#system("touch thickness.txt");
#system("chmod 777 thickness.txt");
open(FINALOUTPUT,">thickness.txt");
print FINALOUTPUT "TPS Ground hold design results\n\n";

# ------------------------------------------------------------------------------------

#   system("touch miniver.in");
#   system("chmod 777 miniver.in");
   system("cp groundhold miniver.in");

# ------------------------------------------------------------------------------------
# assign material properties
# ------------------------------------------------------------------------------------

if ($material_option eq 1) {

   $inputfile = "stackup.in";

}  else {

   if ($material eq "AETB8_Al") {
      $inputfile="AETB_8_AL_5inputs.in";
   }
   if ($material eq "AETB8_GrEx") {
      $inputfile="AETB_8_GrEx_5inputs.in";
   }
   if ($material eq "AETB12_Al") {
      $inputfile="AETB_12_AL_5inputs.in";
   }
   if ($material eq "AETB12_GrEx") {
      $inputfile="AETB_12_GrEX_5inputs.in";
   }
   if ($material eq "FRCI12_Al") {
      $inputfile="FRCI_12_AL_5inputs.i";
   }
   if ($material eq "FRCI12_GrEx") {
      $inputfile="FRCI_12_GrEx_5inputs.in";
   }
   if ($material eq "FRCI20_Al") {
      $inputfile="FRCI_20_AL_5inputs.in";
   }
   if ($material eq "FRCI20_GrEx") {
      $inputfile="FRCI_20_GrEx_5inputs.in";
```

```
   }
   if ($material eq "LI900_Al") {
      $inputfile="LI_900_AL_5inputs.in";
   }
   if ($material eq "LI900_GrEx") {
      $inputfile="LI_900_GrEx_5inputs.in";
   }
   if ($material eq "LI2200_Al") {
      $inputfile="LI_2200_AL_5inputs.in";
   }
   if ($material eq "LI2200_GrEx") {
      $inputfile="LI_2200_GrEx_5inputs.in";
   }
   if ($material eq "RCC_GrEx") {
      $inputfile="RCC_GrEx_5inputs.in";
   }
   if ($material eq "RCC_Al") {
      $inputfile="RCC_AL_5inputs.in";
   }
   if ($material eq "SiC_GrEx") {
      $inputfile="SIC_GrEx_5inputs.in";
   }
   if ($material eq "SiC_Al") {
      $inputfile="SIC_AL_5inputs.in";
   }
   if ($material eq "TUFI_GrEx") {
      $inputfile="TUFI_GrEx_5inputs.in";
   }
   if ($material eq "AFRSI_Al") {
      $inputfile="AFRSI_Al_3inputs.in";
   }
   if ($material eq "AFRSI_GrEx") {
      $inputfile="AFRSI_GrEx_3inputs.in";
   }
      if ($material eq "CFBI_Al") {
      $inputfile="CFBI_Al_3inputs.in";
   }
      if ($material eq "CFBI_GrEx") {
      $inputfile="CFBI_GrEx_3inputs.in";
   }
      if ($material eq "DURAFRSI_Al") {
      $inputfile="DURAFRSI_Al_3inputs.in";
   }
      if ($material eq "DURAFRSI_GrEx") {
      $inputfile="DURAFRSI_GrEx_3inputs.in";
   }
      if ($material eq "PBI_Al") {
      $inputfile="PBI_Al_3inputs.in";
   }
      if ($material eq "PBI_GrEx") {
      $inputfile="PBI_GrEx_3inputs.in";
   }
      if ($material eq "TABI_Al") {
      $inputfile="TABI_Al_3inputs.in";
   }
      if ($material eq "TABI_GrEx") {
      $inputfile="TABI_GrEx_3inputs.in";
   }
}

#print "input file = $inputfile\n";
# ------------------------------------------------------------------------------
# Execute TCAT and send ADS output into file named "junkoutput"
# ------------------------------------------------------------------------------

#   system("touch inputs.in");
#   system("chmod 777 inputs.in");
   system("cp $inputfile inputs.in");

if ($material_option eq 1) {
```

```
   if ($method eq 1) {

      system("./ground.exe");

   }  else {

      system("./ground2.exe");

   }

}  else {

   if ($material eq "RCC_GrEx") {
      system("./ground2.exe");        # run TCAT
   }
   elsif ($material eq "RCC_Al") {
      system("./ground2.exe");        # run TCAT
   }
   elsif ($material eq "SiC_GrEx") {
      system("./ground2.exe");        # run TCAT
   }
   elsif ($material eq "SiC_Al") {
      system("./ground2.exe");        # run TCAT
   }
   elsif ($material eq "TUFI_GrEx") {
      system("./ground2.exe");        # run TCAT
   }
   else {
      system("./ground.exe"); # run TCAT
   }
}

#   system("touch $name[$loop_index]_temphist");
#   system("chmod 777 $name[$loop_index]_temphist");
   system("mv fort.90 groundhold_temphist");


#   system("touch $name[$loop_index]_heatratehist");
#   system("chmod 777 $name[$loop_index]_heatratehist");
   system("mv fort.91 groundhold_heatratehist");

   open(TCATOUTPUT,"<fort.12"); # obtain the current thickness of TPS file
   $_=<TCATOUTPUT>;
   /^\s*(\S*)/;
   $thickness=$1;
   close(TCATOUTPUT);

   $thkns=$thickness;

   printf FINALOUTPUT "Groundhold - TPS thickness = %6.3f in.\n",$thickness*100/2.54;


#system("rm point*");

# ------------------------------------------------------------------------------
# End of heating analysis
# ------------------------------------------------------------------------------

$avg_thickness=$thkns;
#for ($loop_index=$start; $loop_index <= $end; $loop_index++) {
#   $avg_thickness=$avg_thickness+$thkns[$loop_index];
#}
#$avg_thickness=$avg_thickness/$end;

open(DEN,"<material_density"); # obtain the density of TPS material
$_=<DEN>;
/^\s*(\S*)/;
$tps_density=$1;
close(DEN);

if (($sharp eq "yes") && ($bodyside eq "windward")) {
```

```
   $sharp_weight=4.6e-3*$bodyarea;    #SHARP weight in lbm
   if ($material eq "RCC_GrEx") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*15
77.8347)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;      #obtain density of tps
   }
   elsif ($material eq "RCC_Al") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*15
77.8347)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;      #obtain density of tps
   }
   elsif ($material eq "SiC_GrEx") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*24
00.00)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
   }
   elsif ($material eq "SiC_Al") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*24
00.00)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
   }
   elsif ($material eq "TUFI_GrEx") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*13
13.53)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
   }
   else {
      $tile_unit_weight=$avg_thickness*3.2808*$tps_density/0.4536/3.2808/3.2808/3.2808;
        #obtain density of tps
   }
   $tile_area_to_body_area=$bodytile;
   printf FINALOUTPUT "\n";
   printf FINALOUTPUT "Chosen TPS material was $material\n\n";
   printf FINALOUTPUT "Nose SHARP TPS: %6.2f lbm\n", $sharp_weight;
   printf FINALOUTPUT "$material unit weight: %8.4f lbm/ft2 \n ",$tile_unit_weight;
   printf FINALOUTPUT "$material TPS area to body area ratio: %4.2f \n
",$tile_area_to_body_area;

}
if (($sharp eq "yes") && ($bodyside eq "leeward")) {
   $sharp_weight=4.6e-3*$bodyarea;    #SHARP weight in lbm
   $blanket_unit_weight=$avg_thickness*3.2808*$tps_density/0.4536/3.2808/3.2808/3.2808;;
        #obtain density of tps
   $blanket_area_to_body_area=$bodyblanket;
   printf FINALOUTPUT "\n";
   printf FINALOUTPUT "Chosen TPS material was $material\n\n";
   printf FINALOUTPUT "Nose SHARP TPS: %6.2f lbm\n", $sharp_weight;
   printf FINALOUTPUT "$material unit weight: %6.2f lbm/ft2 \n ",$blanket_unit_weight;
   printf FINALOUTPUT "$material TPS area to body area ratio: %6.2f \n
",$blanket_area_to_body_area;
}
if (($sharp eq "no") && ($bodyside eq "windward")) {
   if ($material eq "RCC_GrEx") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*15
77.8347)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;      #obtain density of tps
   }
   elsif ($material eq "RCC_Al") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*15
77.8347)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;      #obtain density of tps
   }
   elsif ($material eq "SiC_GrEx") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*24
00.00)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
   }
   elsif ($material eq "SiC_Al") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*24
00.00)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
```

```
   }
   elsif ($material eq "TUFI_GrEx") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*13
13.53)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
   }
   else {
      $tile_unit_weight=$avg_thickness*3.2808*$tps_density/0.4536/3.2808/3.2808/3.2808;
        #obtain density of tps
   }
   $tile_area_to_body_area=$bodytile;
   printf FINALOUTPUT "Chosen TPS material was $material\n\n";
   printf FINALOUTPUT "$material unit weight: %6.2f lbm/ft2 \n ",$tile_unit_weight;
   printf FINALOUTPUT "$material TPS area to body area ratio: %6.2f \n
",$tile_area_to_body_area;
}
if (($sharp eq "no") && ($bodyside eq "leeward")) {
   $blanket_unit_weight=$avg_thickness*3.2808*$tps_density/0.4536/3.2808/3.2808/3.2808;;
        #obtain density of tps
   $blanket_area_to_body_area=$bodyblanket;
   printf FINALOUTPUT "Chosen TPS material was $material\n\n";
   printf FINALOUTPUT "$material unit weight: %6.2f lbm/ft2 \n ",$blanket_unit_weight;
   printf FINALOUTPUT "$material TPS area to body area ratio: %6.2f \n
",$blanket_area_to_body_area;
}

print "average thickness: $avg_thickness\n";
print "TPS density: $tps_density\n\n";

if ($material_1 eq "RCC_Laminate_Surface") {


$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*15
77.8347)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;      #obtain density of tps

}  elsif ($material_1 eq "SiC_Laminate_Surface") {


$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*24
00.00)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps

}  elsif ($material_1 eq "TUFI_Composite_Layer") {


$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*13
13.53)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps

}  else {

   $tile_unit_weight=$avg_thickness*3.2808*$tps_density/0.4536/3.2808/3.2808/3.2808;
#obtain density of tps

}

printf FINALOUTPUT "\n";

if ($method eq 1) {

   printf FINALOUTPUT "Chosen TPS material was $material_1\n\n";

} else {

   printf FINALOUTPUT "Chosen TPS material was $material_2\n\n";

}

printf FINALOUTPUT "TPS unit weight: %8.4f lbm/ft2 \n ",$tile_unit_weight;


close(FINALOUTPUT);
```

```
# ------------------------------------------------------------------------------
# print needed information to screen and file named "thickness.txt"
# ------------------------------------------------------------------------------

open(OUT,"<thickness.txt");           #opens the input file for TCAT script
open(WEB,"<tcat_output.html");               #open tcat html file
#print "Content-type:text/html\n\n";
while(<WEB>){
   if(/Insert stuff here/){
      print "<BR>\n";
      while(<OUT>){
         print "<BR>\n";
         print $_;
      }
    }
    else{
       print $_;
    }
 }
 close(WEB);
close(OUT);

# }
```

Appendix I: 'compare.cgi' Ground Hold Multiple Analysis CGI Script

```
#!/usr/local/bin/perl

# --------------------------------------------------------------------------------
# Obtain Information From Web Browse
# --------------------------------------------------------------------------------
# This section of code allows a cgi script to run from a web browser.
# The information is passed using the POST and the input information is
# parceled out in a mesh type variable and stores in $FORM{$name}.
#

$PATH=`pwd`;
chop($PATH);

 read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
 @pairs = split(/&/, $buffer);
 foreach $pair (@pairs)
 {
       ($name, $value) = split(/=/, $pair);
       $value =~ tr/+/ /;
       $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
       $value =~ s/~!/ ~!/g;
       $FORM{$name} = $value;
 }

# --------------------------------------------------------------------------------
# assign information from input website to appropriate variable
# --------------------------------------------------------------------------------

 $material_option=$FORM{material_option};

 $matnum[1]=$FORM{matnum_1};
 $matnum[2]=$FORM{matnum_2};
 $matnum[3]=$FORM{matnum_3};

 $material_1[1]=$FORM{material_1_1};
 $material_2[1]=$FORM{material_2_1};
 $material_3[1]=$FORM{material_3_1};
 $material_4[1]=$FORM{material_4_1};
 $material_5[1]=$FORM{material_5_1};

 $material_1[2]=$FORM{material_1_2};
 $material_2[2]=$FORM{material_2_2};
 $material_3[2]=$FORM{material_3_2};
 $material_4[2]=$FORM{material_4_2};
 $material_5[2]=$FORM{material_5_2};

 $material_1[3]=$FORM{material_1_3};
 $material_2[3]=$FORM{material_2_3};
 $material_3[3]=$FORM{material_3_3};
 $material_4[3]=$FORM{material_4_3};
 $material_5[3]=$FORM{material_5_3};

 $method[1]=$FORM{method_1};
 $method[2]=$FORM{method_2};
 $method[3]=$FORM{method_3};

 $holdtime=$FORM{holdtime};
 $timestep=$FORM{timestep};
 $ambient=$FORM{ambient};
 $fueltemp=$FORM{fueltemp};
 $frost=$FORM{frost};

open(TIME,">time");
print TIME "$holdtime         !/Time of Groundhold";
close(TIME);

$heatrate = 1;
```

```
$heattransfer = $heatrate * $ambient *3.15459074506;

open(HEATXFER,">heat");
print HEATXFER "$heattransfer         !/Heat Flux";
close(HEATXFER);

$ambient = ($ambient-32)*(5/9)+273.15;

$frost = ($frost-32)*(5/9)+273.15;

$fueltemp = ($fueltemp-32)*(5/9)+273.15;

open(TANK,">fueltemp");
print TANK "$fueltemp               !/Temperature of fuel";
close(TANK);


for ($case=1; $case <= 3; $case++) {

if ($material_option eq 1) {

   for ($loopnum=1; $loopnum <= $matnum[$case]; $loopnum++) {

      if ($loopnum eq 1) {

         open(DAT,"<$material_1[$case]");

         $i = 0;

         while(<DAT>) {
            $i = $i + 1;
            my($line) = $_;
            chomp($line);

            if ($i eq 1) {
               $nodes[1] = $line;
            }  elsif ($i eq 2) {
               $cond[1] = $line;
            }  elsif ($i eq 3) {
               $density[1] = $line;
            }  elsif ($i eq 4) {
               $specheat[1] = $line;
            }  elsif ($i eq 5) {
               $thick[1] = $line;
            }  elsif ($i eq 6) {
               $templimit[1]=$line;
            }  elsif ($i eq 7) {
               $emissivity=$line;
            }
         }

         close(DAT);

      }  elsif ($loopnum eq 2) {

         open(DAT,"<$material_2[$case]");

         $i = 0;

         while(<DAT>) {
            $i = $i + 1;
            my($line) = $_;
            chomp($line);

            if ($i eq 1) {
               $nodes[2] = $line;
            }  elsif ($i eq 2) {
               $cond[2] = $line;
            }  elsif ($i eq 3) {
               $density[2] = $line;
            }  elsif ($i eq 4) {
```

```
            $specheat[2] = $line;
         }  elsif ($i eq 5) {
            $thick[2] = $line;
         }  elsif ($i eq 6) {
            $templimit[2]=$line;
         }

      }

      close(DAT);

   }  elsif ($loopnum eq 3) {

      open(DAT,"<$material_3[$case]");

      $i = 0;

      while(<DAT>) {
         $i = $i + 1;
         my($line) = $_;
         chomp($line);

         if ($i eq 1) {
            $nodes[3] = $line;
         }  elsif ($i eq 2) {
            $cond[3] = $line;
         }  elsif ($i eq 3) {
            $density[3] = $line;
         }  elsif ($i eq 4) {
            $specheat[3] = $line;
         }  elsif ($i eq 5) {
            $thick[3] = $line;
         }  elsif ($i eq 6) {
            $templimit[3]=$line;
         }

      }

      close(DAT);

   }  elsif ($loopnum eq 4) {

      open(DAT,"<$material_4[$case]");

      $i = 0;

      while(<DAT>) {
         $i = $i + 1;
         my($line) = $_;
         chomp($line);

         if ($i eq 1) {
            $nodes[4] = $line;
         }  elsif ($i eq 2) {
            $cond[4] = $line;
         }  elsif ($i eq 3) {
            $density[4] = $line;
         }  elsif ($i eq 4) {
            $specheat[4] = $line;
         }  elsif ($i eq 5) {
            $thick[4] = $line;
         }  elsif ($i eq 6) {
            $templimit[4]=$line;
         }

      }

      close(DAT);

   }  elsif ($loopnum eq 5) {
```

```
        open(DAT,"<$material_5[$case]");

        $i = 0;

        while(<DAT>) {
           $i = $i + 1;
           my($line) = $_;
           chomp($line);

           if ($i eq 1) {
              $nodes[5] = $line;
           }  elsif ($i eq 2) {
              $cond[5] = $line;
           }  elsif ($i eq 3) {
              $density[5] = $line;
           }  elsif ($i eq 4) {
              $specheat[5] = $line;
           }  elsif ($i eq 5) {
              $thick[5] = $line;
           }  elsif ($i eq 6) {
              $templimit[5]=$line;
           }

        }

        close(DAT);

      }

   }

##############################################################################################
##########
# Create the TPS stackup file by reading the properties of each material selected and
entering    #
# their values into a input file recognizable by T-CAT.
#
##############################################################################################
##########

#   system("touch stackup.in");          #create TPS stackup file
#   system("chmod 777 stackup.in");      #change permissions of TPS stackup file
   open(STACKUP,">stackup.in");
   print STACKUP "$matnum[$case]              !/number of materials\n";
   print STACKUP "$timestep           !/time step\n";


   for ($loopnum=1; $loopnum <= $matnum[$case]; $loopnum++) {
       printf STACKUP "%i             !/number of nodes of material
$loopnum\n",$nodes[$loopnum];
   }

   print STACKUP "$emissivity         !/emissivity of material 1\n";

   for ($loopnum=1; $loopnum <= $matnum[$case]; $loopnum++) {
       printf STACKUP "%f     !/thermal conductivity of material
$loopnum\n",$cond[$loopnum];
   }

   for ($loopnum=1; $loopnum <= $matnum[$case]; $loopnum++) {
       printf STACKUP "%f     !/density of material $loopnum\n",$density[$loopnum];
   }

   for ($loopnum=1; $loopnum <= $matnum[$case]; $loopnum++) {
       printf STACKUP "%f     !/specific heat of material
$loopnum\n",$specheat[$loopnum];
   }

   print STACKUP "$ambient            !/initial temp (deg K)\n";

   for ($loopnum=1; $loopnum <= $matnum[$case]; $loopnum++) {
```

```
       printf STACKUP "%f     !/TPS thickness of material $loopnum\n",$thick[$loopnum];
   }

   print STACKUP "$frost              !/Temperature limit of material 1\n";
   print STACKUP "$templimit[1]              !/Temperature limit of material 1\n";

   for ($loopnum=2; $loopnum <= $matnum[$case]-1; $loopnum++) {
       printf STACKUP "%f     !/Temperature limit of material
$loopnum\n",$templimit[$loopnum];
       printf STACKUP "%f     !/Temperature limit of material
$loopnum\n",$templimit[$loopnum];
   }

   print STACKUP "$templimit[$matnum[$case]]         !/Temperature limit of material
$matnum\n";
   print STACKUP "$fueltemp           !/backface temperature limit\n";
   print STACKUP "6           !/istrat parameter\n";
   print STACKUP "5           !/iopt parameter\n";
   print STACKUP "5           !/ioned parameter\n";
   print STACKUP "1340        !/iprint parameter\n";

   close(STACKUP);
}


##############################################################################################
############
# This section of the program creates a mock MINIVER file.  The three columns of data
created are   #
# the time, ambient temperature, and rate of heat transfer.
#
##############################################################################################
############

$numtimes = $holdtime/$timestep + 1;

$time[1] = 0;

for ($i=2; $i <= $numtimes; $i++) {

   $time[$i] = $time[$i-1] + $timestep;

}

#system("touch l_stackup.s");        #create mock MINIVER file
#system("chmod 777 l_stackup.s");     #change permissions of file
open(MINI,">l_stackup.s");
print MINI " Mock MINIVER file for input into ground hold analysis                    l0
1.0\n";

for ($i=1; $i <= $numtimes; $i++) {

   print MINI "$time[$i] $ambient $heattransfer\n";

}

print MINI "-100.0";
close(MINI);

# ------------------------------------------------------------------------------------
# MINIVER file is partitioned into body point files
# partition the MINIVER file into separate body point files that include time,
# heat rate, and radiation equilibrium temperature
# ------------------------------------------------------------------------------------

$i=0;                                #initialization of counter index "i"
$filecount=0;
open(FILE,"<l_stackup.s");           #open statement with filehandle FILE

until(eof FILE) {                    #goes through the MINVER file until the end is
reached
```

```
   START: $_=<FILE>;                  #skips the first line of the data file
   if (/^.{73}(\S*)/) {                       #searches for the bodypoint label
   $title=$1;

#   system("touch groundhold");                #create body point file
#   system("chmod 777 groundhold");   #change permissions inorder to write to body point
file

   $filecount=$filecount+1;
   } # end of if statement

   while(<FILE>) {                    #while loop that reads thru each line of the file
      $i=$i+1;

/^\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*)\s*(\S*
)\s*(\S*)\s*(\S*)/;
      $time[$i]=$1;                   #array declaration for time
      $tradeq[$i]=$2;                 #array declaration for rad eq temp
      $qconv[$i]=$3;                  #array declaration for conv heat rate
      $n=$i;
      if ($1==-100) {
         open(NEWFILE,">groundhold"); #open statement for new file "newsample"
         $timesteps=$n-1;
         printf NEWFILE "$timesteps \n";      #formatted print statement"
         for($i=1; $i<$n; $i++) {
            printf NEWFILE "%f \n",$time[$i];
            printf NEWFILE "%f \n",$qconv[$i];
            printf NEWFILE "%f \n",$tradeq[$i];
         }  #end of for loop

         close(NEWFILE);
         $i=0;
         goto START
      }  #end of if statement

   }  #end of while loop
}  #end of until loop
close(FILE);

# ------------------------------------------------------------------------------------
# incrementally move each body point file into "miniver.in" and execute tcat
# perl script that creates the filenames needed inorder to batch execute tcat
# filename are created via input from the user
# inputs (file_start_number, file_end_number, body_side)
# ------------------------------------------------------------------------------------

$start=1;              #start of the file index
#$end=($filecount-1)/2;       #end of the file index
$end=1;

open(FILE,">inputs_for_outputs");
printf FILE "3\n";
close(FILE);

#system("touch thickness.txt");
#system("chmod 777 thickness.txt");
open(FINALOUTPUT,">thickness.txt");
print FINALOUTPUT "TPS Ground hold design results\n\n";

# ------------------------------------------------------------------------------------

#   system("touch miniver.in");
#   system("chmod 777 miniver.in");
   system("cp groundhold miniver.in");

# ------------------------------------------------------------------------------------
# assign material properties
# ------------------------------------------------------------------------------------

if ($material_option eq 1) {
```

```
   $inputfile = "stackup.in";

}  else {

   if ($material eq "AETB8_Al") {
      $inputfile="AETB_8_AL_5inputs.in";
   }
   if ($material eq "AETB8_GrEx") {
      $inputfile="AETB_8_GrEx_5inputs.in";
   }
   if ($material eq "AETB12_Al") {
      $inputfile="AETB_12_AL_5inputs.in";
   }
   if ($material eq "AETB12_GrEx") {
      $inputfile="AETB_12_GrEX_5inputs.in";
   }
   if ($material eq "FRCI12_Al") {
      $inputfile="FRCI_12_AL_5inputs.i";
   }
   if ($material eq "FRCI12_GrEx") {
      $inputfile="FRCI_12_GrEx_5inputs.in";
   }
   if ($material eq "FRCI20_Al") {
      $inputfile="FRCI_20_AL_5inputs.in";
   }
   if ($material eq "FRCI20_GrEx") {
      $inputfile="FRCI_20_GrEx_5inputs.in";
   }
   if ($material eq "LI900_Al") {
      $inputfile="LI_900_AL_5inputs.in";
   }
   if ($material eq "LI900_GrEx") {
      $inputfile="LI_900_GrEx_5inputs.in";
   }
   if ($material eq "LI2200_Al") {
      $inputfile="LI_2200_AL_5inputs.in";
   }
   if ($material eq "LI2200_GrEx") {
      $inputfile="LI_2200_GrEx_5inputs.in";
   }
   if ($material eq "RCC_GrEx") {
      $inputfile="RCC_GrEx_5inputs.in";
   }
   if ($material eq "RCC_Al") {
      $inputfile="RCC_AL_5inputs.in";
   }
   if ($material eq "SiC_GrEx") {
      $inputfile="SIC_GrEx_5inputs.in";
   }
   if ($material eq "SiC_Al") {
      $inputfile="SIC_AL_5inputs.in";
   }
   if ($material eq "TUFI_GrEx") {
      $inputfile="TUFI_GrEx_5inputs.in";
   }
   if ($material eq "AFRSI_Al") {
      $inputfile="AFRSI_Al_3inputs.in";
   }
   if ($material eq "AFRSI_GrEx") {
      $inputfile="AFRSI_GrEx_3inputs.in";
   }
      if ($material eq "CFBI_Al") {
      $inputfile="CFBI_Al_3inputs.in";
   }
      if ($material eq "CFBI_GrEx") {
      $inputfile="CFBI_GrEx_3inputs.in";
   }
      if ($material eq "DURAFRSI_Al") {
      $inputfile="DURAFRSI_Al_3inputs.in";
   }
      if ($material eq "DURAFRSI_GrEx") {
```

```
      $inputfile="DURAFRSI_GrEx_3inputs.in";
   }
      if ($material eq "PBI_Al") {
      $inputfile="PBI_Al_3inputs.in";
   }
      if ($material eq "PBI_GrEx") {
      $inputfile="PBI_GrEx_3inputs.in";
   }
      if ($material eq "TABI_Al") {
      $inputfile="TABI_Al_3inputs.in";
   }
      if ($material eq "TABI_GrEx") {
      $inputfile="TABI_GrEx_3inputs.in";
   }
}

# ------------------------------------------------------------------------------
# Execute TCAT and send ADS output into file named "junkoutput"
# ------------------------------------------------------------------------------

#   system("touch inputs.in");
#   system("chmod 777 inputs.in");
   system("cp $inputfile inputs.in >> error 2>&1");

#   system("touch inputs[$case]");
#   system("chmod 777 inputs[$case]");
   system("cp inputs.in inputs[$case]");

if ($material_option eq 1) {

   if ($method[$case] eq 1) {

      system("./ground.exe");

   }  else {

      system("./ground2.exe");

   }

}  else {

   if ($material eq "RCC_GrEx") {
      system("go4.exe>junkoutput");   # run TCAT
   }
   elsif ($material eq "RCC_Al") {
      system("go4.exe>junkoutput");   # run TCAT
   }
   elsif ($material eq "SiC_GrEx") {
      system("go4.exe>junkoutput");   # run TCAT
   }
   elsif ($material eq "SiC_Al") {
      system("go4.exe>junkoutput");   # run TCAT
   }
   elsif ($material eq "TUFI_GrEx") {
      system("go4.exe>junkoutput");   # run TCAT
   }
   else {
      system("go3.exe>junkoutput");   # run TCAT
   }
}

#   system("touch $name[$loop_index]_temphist");
#   system("chmod 777 $name[$loop_index]_temphist");
   system("mv fort.90 groundhold_temphist[$case] >> error3");


#   system("touch $name[$loop_index]_heatratehist");
#   system("chmod 777 $name[$loop_index]_heatratehist");
   system("mv fort.91 groundhold_heatratehist[$case] >> error4");
```

```
   open(TCATOUTPUT,"<fort.12"); # obtain the current thickness of TPS file
   $_=<TCATOUTPUT>;
   /^\s*(\S*)/;
   $thickness=$1;
   close(TCATOUTPUT);

   $thkns=$thickness;

   printf FINALOUTPUT "Groundhold - TPS thickness = %6.3f in.\n",$thickness*100/2.54;


#system("rm point*");

# ------------------------------------------------------------------------------
# End of heating analysis
# ------------------------------------------------------------------------------

$avg_thickness=$thkns;
#for ($loop_index=$start; $loop_index <= $end; $loop_index++) {
#   $avg_thickness=$avg_thickness+$thkns[$loop_index];
#}
#$avg_thickness=$avg_thickness/$end;

open(DEN,"<material_density"); # obtain the density of TPS material
$_=<DEN>;
/^\s*(\S*)/;
$tps_density=$1;
close(DEN);

if (($sharp eq "yes") && ($bodyside eq "windward")) {
   $sharp_weight=4.6e-3*$bodyarea;    #SHARP weight in lbm
   if ($material eq "RCC_GrEx") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*15
77.8347)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;      #obtain density of tps
   }
   elsif ($material eq "RCC_Al") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*15
77.8347)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;      #obtain density of tps
   }
   elsif ($material eq "SiC_GrEx") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*24
00.00)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
   }
   elsif ($material eq "SiC_Al") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*24
00.00)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
   }
   elsif ($material eq "TUFI_GrEx") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*13
13.53)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
   }
   else {
      $tile_unit_weight=$avg_thickness*3.2808*$tps_density/0.4536/3.2808/3.2808/3.2808;
        #obtain density of tps
   }
   $tile_area_to_body_area=$bodytile;
   printf FINALOUTPUT "\n";
   printf FINALOUTPUT "Chosen TPS material was $material\n\n";
   printf FINALOUTPUT "Nose SHARP TPS: %6.2f lbm\n", $sharp_weight;
   printf FINALOUTPUT "$material unit weight: %8.4f lbm/ft2 \n ",$tile_unit_weight;
   printf FINALOUTPUT "$material TPS area to body area ratio: %4.2f \n
",$tile_area_to_body_area;

}
if (($sharp eq "yes") && ($bodyside eq "leeward")) {
   $sharp_weight=4.6e-3*$bodyarea;    #SHARP weight in lbm
```

```
   $blanket_unit_weight=$avg_thickness*3.2808*$tps_density/0.4536/3.2808/3.2808/3.2808;;
        #obtain density of tps
   $blanket_area_to_body_area=$bodyblanket;
   printf FINALOUTPUT "\n";
   printf FINALOUTPUT "Chosen TPS material was $material\n\n";
   printf FINALOUTPUT "Nose SHARP TPS: %6.2f lbm\n", $sharp_weight;
   printf FINALOUTPUT "$material unit weight: %6.2f lbm/ft2 \n ",$blanket_unit_weight;
   printf FINALOUTPUT "$material TPS area to body area ratio: %6.2f \n
",$blanket_area_to_body_area;
}
if (($sharp eq "no") && ($bodyside eq "windward")) {
   if ($material eq "RCC_GrEx") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*15
77.8347)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;      #obtain density of tps
   }
   elsif ($material eq "RCC_Al") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*15
77.8347)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;      #obtain density of tps
   }
   elsif ($material eq "SiC_GrEx") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*24
00.00)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
   }
   elsif ($material eq "SiC_Al") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*24
00.00)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
   }
   elsif ($material eq "TUFI_GrEx") {

$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*13
13.53)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
   }
   else {
      $tile_unit_weight=$avg_thickness*3.2808*$tps_density/0.4536/3.2808/3.2808/3.2808;
        #obtain density of tps
   }
   $tile_area_to_body_area=$bodytile;
   printf FINALOUTPUT "Chosen TPS material was $material\n\n";
   printf FINALOUTPUT "$material unit weight: %6.2f lbm/ft2 \n ",$tile_unit_weight;
   printf FINALOUTPUT "$material TPS area to body area ratio: %6.2f \n
",$tile_area_to_body_area;
}
if (($sharp eq "no") && ($bodyside eq "leeward")) {
   $blanket_unit_weight=$avg_thickness*3.2808*$tps_density/0.4536/3.2808/3.2808/3.2808;;
        #obtain density of tps
   $blanket_area_to_body_area=$bodyblanket;
   printf FINALOUTPUT "Chosen TPS material was $material\n\n";
   printf FINALOUTPUT "$material unit weight: %6.2f lbm/ft2 \n ",$blanket_unit_weight;
   printf FINALOUTPUT "$material TPS area to body area ratio: %6.2f \n
",$blanket_area_to_body_area;
}

#print "average thickness: $avg_thickness\n";
#print "TPS density: $tps_density\n\n";


if ($material_1[$case] eq "RCC_Laminate_Surface") {


$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*15
77.8347)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;      #obtain density of tps

}  elsif ($material_1[$case] eq "SiC_Laminate_Surface") {


$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*24
00.00)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps
```

```
}  elsif ($material_1[$case] eq "TUFI_Composite_Layer") {


$tile_unit_weight=($avg_thickness+0.00254)*3.2808*($avg_thickness*$tps_density+0.00254*13
13.53)/(0.00254+$avg_thickness)/0.4536/3.2808/3.2808/3.2808;        #obtain density of tps

}  else {

   $tile_unit_weight=$avg_thickness*3.2808*$tps_density/0.4536/3.2808/3.2808/3.2808;
#obtain density of tps

}

printf FINALOUTPUT "\n";

if ($method[$case] eq 1) {

   printf FINALOUTPUT "Chosen TPS material was $material_1[$case]\n\n";

}  else {

   printf FINALOUTPUT "Chosen TPS material was $material_2[$case]\n\n";

}

printf FINALOUTPUT "TPS unit weight: %8.4f lbm/ft2 \n ",$tile_unit_weight;


close(FINALOUTPUT);

$casethick[$case]=$avg_thickness;
$casedensity[$case]=$tps_density;
$unitweight[$case]=$tile_unit_weight;
system("mv thickness.txt  thickness[$case].txt >> error4");


}

if (($unitweight[1] gt $unitweight[2]) && ($unitweight[2] gt $unitweight[3])) {
   $min = 3;
}
if (($unitweight[1] gt $unitweight[3]) && ($unitweight[3] gt $unitweight[2])) {
   $min = 2;
}
if (($unitweight[2] gt $unitweight[1]) && ($unitweight[1] gt $unitweight[3])) {
   $min = 3;
}
if (($unitweight[2] gt $unitweight[3]) && ($unitweight[3] gt $unitweight[1])) {
   $min = 1;
}
if (($unitweight[3] gt $unitweight[2]) && ($unitweight[2] gt $unitweight[1])) {
   $min = 1;
}
if (($unitweight[3] gt $unitweight[1]) && ($unitweight[1] gt $unitweight[2])) {
   $min = 2;
}

open(RESULTS,">results.txt");

printf RESULTS "The optimum design for this analysis is configuration $min\n";
printf RESULTS "The configuration consists of the following materials:\n\n";

printf RESULTS "  Material 1: $material_1[$min]\n";

if ($matnum[$min] gt 1) {
   printf RESULTS "  Material 2: $material_2[$min]\n";
}

if ($matnum[$min] gt 2) {
   printf RESULTS "  Material 3: $material_3[$min]\n";
```

```
}

if ($matnum[$min] gt 3) {
   printf RESULTS "  Material 4: $material_4[$min]\n";
}

if ($matnum[$min] gt 4) {
   printf RESULTS "  Material 5: $material_5[$min]\n";
}

$finalweight=$casethick[$min]*3.2808*$casedensity[$min]/0.4536/3.2808/3.2808/3.2808;

printf RESULTS "\n";
printf RESULTS "------TPS Ground hold design results------\n\n";
printf RESULTS "You chose to optimize layer $method[$min] of the TPS stack.\n\n";

if ($method[$min] eq 1) {

   printf RESULTS "The TPS material for layer $method[$min] is $material_1[$min].\n\n";
   printf RESULTS "$material_1[$min] thickness = %6.3f
in.\n\n",$casethick[$min]*100/2.54;
   printf RESULTS "TPS unit weight: %8.4f lbm/ft2 \n ",$finalweight;

}  else {

   printf RESULTS "The TPS material for layer $method[$min] is $material_2[$min].\n\n";
   printf RESULTS "$material_2[$min] thickness = %6.3f
in.\n\n",$casethick[$min]*100/2.54;
   printf RESULTS "TPS unit weight: %8.4f lbm/ft2 \n ",$finalweight;

}

close(RESULTS);

# ------------------------------------------------------------------------------
# print needed information to screen and file named "thickness.txt"
# ------------------------------------------------------------------------------

open(OUT,"<results.txt");             #opens the input file for TCAT script
open(WEB,"<tcat_output.html");               #open tcat html file
print "Content-type:text/html\n\n";
while(<WEB>){
   if(/Insert stuff here/){
      print "<BR>\n";
      while(<OUT>){
         print "<BR>\n";
         print $_;
      }
    }
    else{
       print $_;
    }
 }
close(WEB);
close(OUT);

# }
```

Appendix J: 'ground.f' FORTRAN Source Code

```
       program ground

************************************************************************
* Program to solve the one dimensional heat equation through a stackup      *
* of disparate TPS materials.  This program finds the optimum thickness *
* of the surface layer of the TPS material to reduce the TPS unit   *
* weight, while eliminating frosting at the surface of the TPS stackup. *
************************************************************************

        implicit none
        integer i,j,n,nodes(6),sum1,sum2,sum3,totnodes,matnum,count
       integer iter,maxiter
        real*8 dt,k(6),den(6),cp(6),initT,L(6),dx(6),sigma,epsi,eps
     1,alpha(6),lam(6),a(400),b(400),c(400),T(10000,400),normf
     1,normdelT,f(400),delT(400),twonorm,time(10000)
     1,qcond(400),qrad(400),qcv(400),g(24),obj,x(12),dTemp
     1,kair,Tb(1000),error,diff(1000),templmt(10),backT,heat
     1,root(1000),toler



************************************************************************
* Obtains information from the file 'inputs.in' which contains TPS  *
* material properties, used in the solution of the heat equation.   *
************************************************************************

        open(10,file='inputs.in')
        read(10,*) matnum               !/number of materials
        read(10,*) dt                   !/time step
        do i=1,matnum
        read(10,*) nodes(i)             !/number of nodes per material
        enddo
        read(10,*) epsi                 !/emmissivity of surface
        do i=1,matnum
        read(10,*) k(i)                 !/thermal conductivities
        enddo
        do i=1,matnum
        read(10,*) den(i)               !/densities
        enddo
        do i=1,matnum
        read(10,*) cp(i)                !/specific heats
        enddo
        read(10,*) initT                !/initial temperatures
        do i=1,matnum
        read(10,*) L(i)                 !/thicknesses
        enddo
        do i=1,2*matnum
        read(10,*) templmt(i)           !/temp limits for each material
        enddo

************************************************************************
* Writes the density of the surface TPS material for use in calculating    *
* the TPS material unit weight in the perl script.                  *
* Also writes the heat transfer rate and the backface fuel temperature  *
* to files for use in the one-dimensional heat equation solver.            *
************************************************************************

        open(40,file='heat')
        read(40,*) heat
        close(40)

        open(41,file='fueltemp')
        read(41,*) backT
        close(41)

        open(9,file='material_density')
        write(9,*) den(1)
        close(9)
```

```
************************************************************************
* Initializes optimization parameters.                                 *
************************************************************************

        toler=0.0001

        error=1.

        maxiter=100

************************************************************************
* Solves the heat equation for using two initial guesses for the    *
* thickness of the TPS.  Values obtained for the objective function *
* will be sent to the next while loop which implements the          *
* Newton-Raphson method to converge to the solution.                *
************************************************************************

        if (matnum.eq.1) then
           call one(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.2) then
           call two(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.3) then
           call three(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.4) then
           call four(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.5) then
           call five(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        endif

        diff(1)=T(n,1)-templmt(1)
        root(1)=L(1)

        L(1)=0.00001

        if (matnum.eq.1) then
           call one(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.2) then
           call two(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.3) then
           call three(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.4) then
           call four(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.5) then
           call five(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        endif

        diff(2)=T(n,1)-templmt(1)
        root(2)=L(1)

************************************************************************
* While loop to call the heat equation solver.  The while loop         *
* reiterates until the temperature of the surface lies above the    *
* frosting condition (40F), to within the specified convergence        *
* criteria (error = 0.009).                                         *
************************************************************************

        iter=2

        dowhile((error.gt.toler).and.(iter.lt.maxiter))
```

```
          iter = iter + 1

          root(iter)=root(iter-1)-((diff(iter-1)*(root(iter-2)
     +                -root(iter-1)))/(diff(iter-2)-diff(iter-1)))

          L(1)=root(iter)

          if (matnum.eq.1) then
             call one(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +            time,qcv,qcond,qrad,backT,heat)
          elseif (matnum.eq.2) then
             call two(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +            time,qcv,qcond,qrad,backT,heat)
          elseif (matnum.eq.3) then
             call three(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +            time,qcv,qcond,qrad,backT,heat)
          elseif (matnum.eq.4) then
             call four(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +            time,qcv,qcond,qrad,backT,heat)
          elseif (matnum.eq.5) then
             call five(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +            time,qcv,qcond,qrad,backT,heat)
          endif

          diff(iter)=T(n,1)-templmt(1)

          error=abs((root(iter)-root(iter-1))/root(iter))

        enddo

*************************************************************************
* Formatted file output statements to write temperature profiles    *
* at key nodes of the TPS material stackup.                         *
*************************************************************************

        time(1)=0.
        do j=2,n
           time(j)=time(j-1)+dt
        enddo
        write(91,2)
        do j=1,n
           write(91,4)time(j),qcv(j),qrad(j),qcond(j)
        enddo
        if (matnum.eq.1) then
           write(90,1)
           do j=1,n
              write(90,3)time(j),T(j,1),T(j,nodes(1))
           enddo
        elseif (matnum.eq.2) then
           write(90,5)
           do j=1,n
              write(90,6)time(j),T(j,1),T(j,nodes(1)),T(j,nodes(1)
     1+nodes(2))
           enddo
        elseif (matnum.eq.3) then
           write(90,7)
           do j=1,n
              write(90,8)time(j),T(j,1),T(j,nodes(1)),T(j,nodes(1)
     1+nodes(2)),T(j,nodes(1)+nodes(2)+nodes(3))
           enddo
        elseif (matnum.eq.4) then
           write(90,9)
           do j=1,n
              write(90,10)time(j),T(j,1),T(j,nodes(1)),T(j,nodes(1)
     1+nodes(2)),T(j,nodes(1)+nodes(2)+nodes(3)),T(j,nodes(1)+nodes(2)
     1+nodes(3)+nodes(4))
           enddo
        elseif (matnum.eq.5) then
           write(90,11)
           do j=1,n
              write(90,12)time(j),T(j,1),T(j,nodes(1)),T(j,nodes(1)
```

```
     1+nodes(2)),T(j,nodes(1)+nodes(2)+nodes(3)),T(j,nodes(1)+nodes(2)
     1+nodes(3)+nodes(4)),T(j,nodes(1)+nodes(2)+nodes(3)+nodes(4)+
     1nodes(5))
           enddo
        endif

        write(12,*)L(1)

*************************************************************************
* Formatting parameters for file output.                                 *
*************************************************************************

 1      format(2x,'time(j)',7x,'T(j,1)',5x,'T(j,nodes(1))')
 2      format(2x,'time',7x,'qconv',5x,'qrad',5x,'qcond')
 3      format(f12.2,f12.2,f12.2)
 4      format(f12.2,3x,f11.2,3x,f11.2,3x,f11.2)
 5      format(5x,'time',5x,'T(j,1)',5x,'T(j,nodes1)',5x
     1,'T(j,totnodes)')
 6      format(f12.2,f12.2,f12.2,f12.2)
 7      format(2x,'time(j)',7x,'T(j,1)',5x,'T(j,nodes(1))',2x
     1,'T(j,sum1)',5x,'T(j,sum2)')
 8      format(f8.2,7x,f8.2,5x,f8.2,4x,f8.2,5x,f8.2)
 9      format(2x,'time(j)',7x,'T(j,1)',5x,'T(j,nodes(1))',2x
     1,'T(j,sum1)',5x,'T(j,sum2)',5x,'T(j,totnodes)')
 10     format(f8.2,7x,f8.2,5x,f8.2,4x,f8.2,5x,f8.2,5x,f8.2)
 11     format(2x,'time(j)',7x,'T(j,1)',5x,'T(j,nodes(1))',2x
     1,'T(j,sum1)',5x,'T(j,sum2)',5x,'T(j,sum3)',5x,'T(j,totnodes)')
 12     format(f8.2,7x,f8.2,5x,f8.2,4x,f8.2,5x,f8.2,5x,f8.2,5x,f8.2)

        end
```

Appendix K: 'ground2.f' FORTRAN Source Code

```
       program ground

************************************************************************
* Program to solve the one dimensional heat equation through a stackup      *
* of disparate TPS materials.  This program finds the optimum thickness *
* of the surface layer of the TPS material to reduce the TPS unit   *
* weight, while eliminating frosting at the surface of the TPS stackup. *
************************************************************************

        implicit none
        integer i,j,n,nodes(6),sum1,sum2,sum3,totnodes,matnum,count
       integer iter,maxiter
        real*8 dt,k(6),den(6),cp(6),initT,L(6),dx(6),sigma,epsi,eps
     1,alpha(6),lam(6),a(400),b(400),c(400),T(10000,400),normf
     1,normdelT,f(400),delT(400),twonorm,time(10000)
     1,qcond(400),qrad(400),qcv(400),g(24),obj,x(12),dTemp
     1,kair,Tb(1000),error,diff(1000),templmt(10),backT,heat
     1,root(1000),toler



************************************************************************
* Obtains information from the file 'inputs.in' which contains TPS  *
* material properties, used in the solution of the heat equation.   *
************************************************************************

        open(10,file='inputs.in')
        read(10,*) matnum               !/number of materials
        read(10,*) dt                   !/time step
        do i=1,matnum
        read(10,*) nodes(i)             !/number of nodes per material
        enddo
        read(10,*) epsi                 !/emmissivity of surface
        do i=1,matnum
        read(10,*) k(i)                 !/thermal conductivities
        enddo
        do i=1,matnum
        read(10,*) den(i)               !/densities
        enddo
        do i=1,matnum
        read(10,*) cp(i)                !/specific heats
        enddo
        read(10,*) initT                !/initial temperatures
        do i=1,matnum
        read(10,*) L(i)                 !/thicknesses
        enddo
        do i=1,2*matnum
        read(10,*) templmt(i)           !/temp limits for each material
        enddo

************************************************************************
* Writes the density of the surface TPS material for use in calculating    *
* the TPS material unit weight in the perl script.                  *
* Also writes the heat transfer rate and the backface fuel temperature  *
* to files for use in the one-dimensional heat equation solver.            *
************************************************************************

        open(40,file='heat')
        read(40,*) heat
        close(40)

        open(41,file='fueltemp')
        read(41,*) backT
        close(41)

        open(9,file='material_density')
        write(9,*) den(2)
        close(9)
```

```
************************************************************************
* Initializes optimization parameters.                                 *
************************************************************************

        toler=0.0001

        error=1.

        maxiter=100

************************************************************************
* Solves the heat equation for using two initial guesses for the    *
* thickness of the TPS.  Values obtained for the objective function *
* will be sent to the next while loop which implements the          *
* Newton-Raphson method to converge to the solution.                *
************************************************************************

        open(44,file='opt')


        if (matnum.eq.1) then
           call one(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.2) then
           call two(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.3) then
           call three(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.4) then
           call four(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.5) then
           call five(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        endif

        diff(1)=T(n,1)-templmt(1)
        root(1)=L(2)

        write(44,*) diff(1)

        L(2)=0.006

        if (matnum.eq.1) then
           call one(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.2) then
           call two(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.3) then
           call three(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.4) then
           call four(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        elseif (matnum.eq.5) then
           call five(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +          time,qcv,qcond,qrad,backT,heat)
        endif

        diff(2)=T(n,1)-templmt(1)
        root(2)=L(2)

        write(44,*) diff(2)

************************************************************************
* While loop to call the heat equation solver.  The while loop         *
* reiterates until the temperature of the surface lies above the    *
* frosting condition (40F), to within the specified convergence        *
```

```
* criteria (error = 0.009).                                            *
************************************************************************

        iter=2

        dowhile((error.gt.toler).and.(iter.lt.maxiter))

           iter = iter + 1

           root(iter)=root(iter-1)-((diff(iter-1)*(root(iter-2)
     +                -root(iter-1)))/(diff(iter-2)-diff(iter-1)))

           L(2)=root(iter)

           if (matnum.eq.1) then
              call one(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +             time,qcv,qcond,qrad,backT,heat)
           elseif (matnum.eq.2) then
              call two(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +             time,qcv,qcond,qrad,backT,heat)
           elseif (matnum.eq.3) then
              call three(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +             time,qcv,qcond,qrad,backT,heat)
           elseif (matnum.eq.4) then
              call four(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +             time,qcv,qcond,qrad,backT,heat)
           elseif (matnum.eq.5) then
              call five(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +             time,qcv,qcond,qrad,backT,heat)
           endif

           diff(iter)=T(n,1)-templmt(1)

           error=abs((root(iter)-root(iter-1))/root(iter))

           write(44,*) diff(iter)

        enddo


        close(44)
************************************************************************
* Formatted file output statements to write temperature profiles       *
* at key nodes of the TPS material stackup.                            *
************************************************************************

        time(1)=0.
        do j=2,n
           time(j)=time(j-1)+dt
        enddo
        write(91,2)
        do j=1,n
           write(91,4)time(j),qcv(j),qrad(j),qcond(j)
        enddo
        if (matnum.eq.1) then
           write(90,1)
           do j=1,n
              write(90,3)time(j),T(j,1),T(j,nodes(1))
           enddo
        elseif (matnum.eq.2) then
           write(90,5)
           do j=1,n
              write(90,6)time(j),T(j,1),T(j,nodes(1)),T(j,nodes(1)
     1+nodes(2))
           enddo
        elseif (matnum.eq.3) then
           write(90,7)
           do j=1,n
              write(90,8)time(j),T(j,1),T(j,nodes(1)),T(j,nodes(1)
     1+nodes(2)),T(j,nodes(1)+nodes(2)+nodes(3))
           enddo
```

```
        elseif (matnum.eq.4) then
           write(90,9)
           do j=1,n
              write(90,10)time(j),T(j,1),T(j,nodes(1)),T(j,nodes(1)
     1+nodes(2)),T(j,nodes(1)+nodes(2)+nodes(3)),T(j,nodes(1)+nodes(2)
     1+nodes(3)+nodes(4))
           enddo
        elseif (matnum.eq.5) then
           write(90,11)
           do j=1,n
              write(90,12)time(j),T(j,1),T(j,nodes(1)),T(j,nodes(1)
     1+nodes(2)),T(j,nodes(1)+nodes(2)+nodes(3)),T(j,nodes(1)+nodes(2)
     1+nodes(3)+nodes(4)),T(j,nodes(1)+nodes(2)+nodes(3)+nodes(4)+
     1nodes(5))
           enddo
        endif

        write(12,*)L(2)

*************************************************************************
* Formatting parameters for file output.                                 *
*************************************************************************

 1      format(2x,'time(j)',7x,'T(j,1)',5x,'T(j,nodes(1))')
 2      format(2x,'time',7x,'qconv',5x,'qrad',5x,'qcond')
 3      format(f12.2,f12.2,f12.2)
 4      format(f12.2,3x,f11.2,3x,f11.2,3x,f11.2)
 5      format(5x,'time',5x,'T(j,1)',5x,'T(j,nodes1)',5x
     1,'T(j,totnodes)')
 6      format(f12.2,f12.2,f12.2,f12.2)
 7      format(2x,'time(j)',7x,'T(j,1)',5x,'T(j,nodes(1))',2x
     1,'T(j,sum1)',5x,'T(j,sum2)')
 8      format(f8.2,7x,f8.2,5x,f8.2,4x,f8.2,5x,f8.2)
 9      format(2x,'time(j)',7x,'T(j,1)',5x,'T(j,nodes(1))',2x
     1,'T(j,sum1)',5x,'T(j,sum2)',5x,'T(j,totnodes)')
 10     format(f8.2,7x,f8.2,5x,f8.2,4x,f8.2,5x,f8.2,5x,f8.2)
 11     format(2x,'time(j)',7x,'T(j,1)',5x,'T(j,nodes(1))',2x
     1,'T(j,sum1)',5x,'T(j,sum2)',5x,'T(j,sum3)',5x,'T(j,totnodes)')
 12     format(f8.2,7x,f8.2,5x,f8.2,4x,f8.2,5x,f8.2,5x,f8.2,5x,f8.2)

        end
```

Appendix L: 'one.f' FORTRAN Subroutine

```
      subroutine one(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +            time,qcv,qcond,qrad,backT,heat)

************************************************************************
* Subroutine to solve the one dimensional heat equation for a TPS   *
* material stackup consisting of one materials.                         *
* This subroutine incorporates an isothermal boundary condtion at the   *
* backface of the TPS material stackup (generally the cryogenic fuel    *
* temperature) and also a constant heat flux boundary condition at the  *
* surface of the TPS material stackup.  The constant heat flux is   *
* determined solely by the temperature of the ambient air.          *
************************************************************************

      implicit none
      integer i,j,n,nodes(6),sum1,sum2,sum3,totnodes,matnum,count
      real*8 dt,k(6),den(6),cp(6),initT,L(6),dx(6),sigma,epsi,eps
     1,alpha(6),lam(6),a(400),b(400),c(400),T(10000,400),normf
     1,normdelT,f(400),delT(400),twonorm,Tradeq(10000),time(10000)
     1,qcond(10000),qrad(10000),qcv(10000),g(24),obj,x(12),Tb(400)
     1,backT,heat

************************************************************************
* Calls linint subroutine to linearly interpolate convective heat flux  *
* and Radiation Equilibrium temperature values using user specified *
* time and time step for analysis.                                  *
************************************************************************

      call linint(n,dt,time,qcv,Tradeq)

************************************************************************
* Define the mathematical constants to be used in the calculations. *
************************************************************************

      eps=1.e-6
      sigma=5.67061e-8
      do i=1,matnum
         dx(i)=L(i)/(nodes(i)-1.)
         alpha(i)=k(i)/(den(i)*cp(i)*1000.)
         lam(i)=alpha(i)*dt/(dx(i)**2)
      enddo
      sum1=nodes(1)+nodes(2)
      sum2=sum1+nodes(3)
      sum3=sum2+nodes(4)
      totnodes=nodes(1)+nodes(2)+nodes(3)+nodes(4)+nodes(5)

************************************************************************
* Specify the initial conditions of the problem.                    *
************************************************************************

      do i=1,totnodes
         T(1,i)=initT
      enddo

      do i=1,totnodes
         Tb(i)=T(1,i)
      enddo

      Tb(totnodes)=backT

      do i=1,totnodes
         T(1,i)=Tb(i)
      enddo

************************************************************************
* Initialize time array.                                            *
************************************************************************

      time(1)=0.
```

```
**************************************************************************
* Do loop to solve for the temperature profiles for each time step j       *
**************************************************************************

       do j=2,n

**************************************************************************
* Setup the tridiagonal matrix to be used in the Thomas Algorithm.      *
**************************************************************************


          a(1)=-2.*lam(1)
          b(1)=1.+2.*lam(1)
          c(1)=0.
          do i=2,nodes(1)-1
             a(i)=-1.*lam(1)
             b(i)=1.+2.*lam(1)
             c(i)=-1.*lam(1)
          enddo
          a(totnodes)=0.
          b(totnodes)=1.
          c(totnodes)=0.

           time(j)=time(j-1)+dt

           Tb(1)=Tb(1)+2.*lam(1)*heat*dx(1)/k(1)

           call tridag(a,b,c,Tb,delT,totnodes)

           do i=1,totnodes
              T(j,i)=delT(i)
              Tb(i)=delT(i)
           enddo

        enddo

**************************************************************************
* Calculation of the radiative and conductive heat fluxes.          *
**************************************************************************

       do j=1,n
          qrad(j)=epsi*sigma*T(j,1)**4
          qcond(j)=qcv(j)-qrad(j)
       enddo

**************************************************************************
* End of subroutine one.                                                 *
**************************************************************************

       return
       end
```

Appendix M: 'two.f' FORTRAN Subroutine

```
        subroutine two(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +             time,qcv,qcond,qrad,backT,heat)

************************************************************************
* Subroutine to solve the one dimensional heat equation for a TPS   *
* material stackup consisting of two dissparate materials.          *
* This subroutine incorporates an isothermal boundary condtion at the       *
* backface of the TPS material stackup (generally the cryogenic fuel        *
* temperature) and also a constant heat flux boundary condition at the      *
* surface of the TPS material stackup.  The constant heat flux is   *
* determined solely by the temperature of the ambient air.          *
************************************************************************

        implicit none
        integer i,j,n,nodes(6),sum1,sum2,sum3,totnodes,matnum,count
        real*8 dt,k(6),den(6),cp(6),initT,L(6),dx(6),sigma,epsi,eps
     1,alpha(6),lam(6),a(400),b(400),c(400),T(10000,400),normf
     1,normdelT,f(400),delT(400),twonorm,Tradeq(10000),time(10000)
     1,qcond(10000),qrad(10000),qcv(10000),g(24),obj,x(12),Tb(400)
     1,backT,heat

************************************************************************
* Calls linint subroutine to linearly interpolate convective heat flux      *
* and Radiation Equilibrium temperature values using user specified *
* time and time step for analysis.                                  *
************************************************************************

        call linint(n,dt,time,qcv,Tradeq)

************************************************************************
* Define the mathematical constants to be used in the calculations. *
************************************************************************

        eps=1.e-6
        sigma=5.67061e-8
        do i=1,matnum
           dx(i)=L(i)/(nodes(i)-1.)
           alpha(i)=k(i)/(den(i)*cp(i)*1000.)
           lam(i)=alpha(i)*dt/(dx(i)**2)
        enddo
        sum1=nodes(1)+nodes(2)
        sum2=sum1+nodes(3)
        sum3=sum2+nodes(4)
        totnodes=nodes(1)+nodes(2)+nodes(3)+nodes(4)+nodes(5)

************************************************************************
* Specify the initial conditions of the problem.                    *
************************************************************************

        do i=1,totnodes
           T(1,i)=initT
        enddo

        do i=1,totnodes
           Tb(i)=T(1,i)
        enddo

        Tb(totnodes)=backT

        do i=1,totnodes
           T(1,i)=Tb(i)
        enddo

************************************************************************
* Initialize time array.                                            *
************************************************************************

        time(1)=0.
```

```
************************************************************************
* Do loop to solve for the temperature profiles for each time step j       *
************************************************************************

        do j=2,n

************************************************************************
* Setup the tridiagonal matrix to be used in the Thomas Algorithm.      *
************************************************************************


           a(1)=-2.*lam(1)
           b(1)=1.+2.*lam(1)
           c(1)=0.
           do i=2,nodes(1)-1
              a(i)=-1.*lam(1)
              b(i)=1.+2.*lam(1)
              c(i)=-1.*lam(1)
           enddo
           a(nodes(1))=-2.*lam(2)
           b(nodes(1))=1.+2.*lam(2)+2.*lam(2)*k(1)/k(2)
           c(nodes(1))=-2.*lam(2)*k(1)/k(2)
           do i=nodes(1)+1,sum1-1
              a(i)=-1.*lam(2)
              b(i)=1.+2.*lam(2)
              c(i)=-1.*lam(2)
           enddo
           a(totnodes)=0.
           b(totnodes)=1.
           c(totnodes)=0.

           time(j)=time(j-1)+dt

           Tb(1)=Tb(1)+2.*lam(1)*heat*dx(1)/k(1)

           call tridag(a,b,c,Tb,delT,totnodes)

           do i=1,totnodes
              T(j,i)=delT(i)
              Tb(i)=delT(i)
           enddo

        enddo

************************************************************************
* Calculation of the radiative and conductive heat fluxes.          *
************************************************************************

        do j=1,n
           qrad(j)=epsi*sigma*T(j,1)**4
           qcond(j)=qcv(j)-qrad(j)
        enddo

************************************************************************
* End of subroutine two.                                               *
************************************************************************

        return
        end
```

Appendix N: 'three.f' FORTRAN Subroutine

```
        subroutine three(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +             time,qcv,qcond,qrad,backT,heat)

************************************************************************
* Subroutine to solve the one dimensional heat equation for a TPS   *
* material stackup consisting of three dissparate materials.               *
* This subroutine incorporates an isothermal boundary condtion at the      *
* backface of the TPS material stackup (generally the cryogenic fuel       *
* temperature) and also a constant heat flux boundary condition at the     *
* surface of the TPS material stackup.  The constant heat flux is   *
* determined solely by the temperature of the ambient air.          *
************************************************************************

        implicit none
        integer i,j,n,nodes(6),sum1,sum2,sum3,totnodes,matnum,count
        real*8 dt,k(6),den(6),cp(6),initT,L(6),dx(6),sigma,epsi,eps
     1,alpha(6),lam(6),a(400),b(400),c(400),T(10000,400),normf
     1,normdelT,f(400),delT(400),twonorm,Tradeq(10000),time(10000)
     1,qcond(10000),qrad(10000),qcv(10000),g(24),obj,x(12),Tb(400)
     1,backT,heat

************************************************************************
* Calls linint subroutine to linearly interpolate convective heat flux      *
* and Radiation Equilibrium temperature values using user specified *
* time and time step for analysis.                                  *
************************************************************************

        call linint(n,dt,time,qcv,Tradeq)

************************************************************************
* Define the mathematical constants to be used in the calculations. *
************************************************************************

        eps=1.e-6
        sigma=5.67061e-8
        do i=1,matnum
           dx(i)=L(i)/(nodes(i)-1.)
           alpha(i)=k(i)/(den(i)*cp(i)*1000.)
           lam(i)=alpha(i)*dt/(dx(i)**2)
        enddo
        sum1=nodes(1)+nodes(2)
        sum2=sum1+nodes(3)
        sum3=sum2+nodes(4)
        totnodes=nodes(1)+nodes(2)+nodes(3)+nodes(4)+nodes(5)

************************************************************************
* Specify the initial conditions of the problem.                    *
************************************************************************

        do i=1,totnodes
           T(1,i)=initT
        enddo

        do i=1,totnodes
           Tb(i)=T(1,i)
        enddo

        Tb(totnodes)=backT

        do i=1,totnodes
           T(1,i)=Tb(i)
        enddo

************************************************************************
* Initialize time array.                                            *
************************************************************************

        time(1)=0.
```

```
************************************************************************
* Do loop to solve for the temperature profiles for each time step j       *
************************************************************************

        do j=2,n

************************************************************************
* Setup the tridiagonal matrix to be used in the Thomas Algorithm.      *
************************************************************************


           a(1)=-2.*lam(1)
           b(1)=1.+2.*lam(1)
           c(1)=0.
           do i=2,nodes(1)-1
              a(i)=-1.*lam(1)
              b(i)=1.+2.*lam(1)
              c(i)=-1.*lam(1)
           enddo
           a(nodes(1))=-2.*lam(2)
           b(nodes(1))=1.+2.*lam(2)+2.*lam(2)*k(1)/k(2)
           c(nodes(1))=-2.*lam(2)*k(1)/k(2)
           do i=nodes(1)+1,sum1-1
              a(i)=-1.*lam(2)
              b(i)=1.+2.*lam(2)
              c(i)=-1.*lam(2)
           enddo
           a(sum1)=-2.*lam(3)
           b(sum1)=1.+2.*lam(3)+2.*lam(3)*k(2)/k(3)
           c(sum1)=-2.*lam(3)*k(2)/k(3)
           do i=sum1+1,sum2-1
              a(i)=-1.*lam(3)
              b(i)=1.+2.*lam(3)
              c(i)=-1.*lam(3)
           enddo
           a(totnodes)=0.
           b(totnodes)=1.
           c(totnodes)=0.

           time(j)=time(j-1)+dt

           Tb(1)=Tb(1)+2.*lam(1)*heat*dx(1)/k(1)

           call tridag(a,b,c,Tb,delT,totnodes)

           do i=1,totnodes
              T(j,i)=delT(i)
              Tb(i)=delT(i)
           enddo

        enddo

************************************************************************
* Calculation of the radiative and conductive heat fluxes.          *
************************************************************************

        do j=1,n
           qrad(j)=epsi*sigma*T(j,1)**4
           qcond(j)=qcv(j)-qrad(j)
        enddo

************************************************************************
* End of subroutine three.                                             *
************************************************************************

        return
        end
```

Appendix O: 'four.f' FORTRAN Subroutine

```
      subroutine four(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +            time,qcv,qcond,qrad,backT,heat)

************************************************************************
* Subroutine to solve the one dimensional heat equation for a TPS   *
* material stackup consisting of four dissparate materials.         *
* This subroutine incorporates an isothermal boundary condtion at the       *
* backface of the TPS material stackup (generally the cryogenic fuel        *
* temperature) and also a constant heat flux boundary condition at the      *
* surface of the TPS material stackup.  The constant heat flux is   *
* determined solely by the temperature of the ambient air.          *
************************************************************************

      implicit none
      integer i,j,n,nodes(6),sum1,sum2,sum3,totnodes,matnum,count
      real*8 dt,k(6),den(6),cp(6),initT,L(6),dx(6),sigma,epsi,eps
     1,alpha(6),lam(6),a(400),b(400),c(400),T(10000,400),normf
     1,normdelT,f(400),delT(400),twonorm,Tradeq(10000),time(10000)
     1,qcond(10000),qrad(10000),qcv(10000),g(24),obj,x(12),Tb(400)
     1,backT,heat

************************************************************************
* Calls linint subroutine to linearly interpolate convective heat flux      *
* and Radiation Equilibrium temperature values using user specified *
* time and time step for analysis.                                  *
************************************************************************

      call linint(n,dt,time,qcv,Tradeq)

************************************************************************
* Define the mathematical constants to be used in the calculations. *
************************************************************************

      eps=1.e-6
      sigma=5.67061e-8
      do i=1,matnum
         dx(i)=L(i)/(nodes(i)-1.)
         alpha(i)=k(i)/(den(i)*cp(i)*1000.)
         lam(i)=alpha(i)*dt/(dx(i)**2)
      enddo
      sum1=nodes(1)+nodes(2)
      sum2=sum1+nodes(3)
      sum3=sum2+nodes(4)
      totnodes=nodes(1)+nodes(2)+nodes(3)+nodes(4)+nodes(5)

************************************************************************
* Specify the initial conditions of the problem.                    *
************************************************************************

      do i=1,totnodes
         T(1,i)=initT
      enddo

      do i=1,totnodes
         Tb(i)=T(1,i)
      enddo

      Tb(totnodes)=backT

      do i=1,totnodes
         T(1,i)=Tb(i)
      enddo

************************************************************************
* Initialize time array.                                            *
************************************************************************
```

```
        time(1)=0.

***************************************************************************
* Do loop to solve for the temperature profiles for each time step j       *
***************************************************************************

        do j=2,n

***************************************************************************
* Setup the tridiagonal matrix to be used in the Thomas Algorithm.      *
***************************************************************************


           a(1)=-2.*lam(1)
           b(1)=1.+2.*lam(1)
           c(1)=0.
           do i=2,nodes(1)-1
              a(i)=-1.*lam(1)
              b(i)=1.+2.*lam(1)
              c(i)=-1.*lam(1)
           enddo
           a(nodes(1))=-2.*lam(2)
           b(nodes(1))=1.+2.*lam(2)+2.*lam(2)*k(1)/k(2)
           c(nodes(1))=-2.*lam(2)*k(1)/k(2)
           do i=nodes(1)+1,sum1-1
              a(i)=-1.*lam(2)
              b(i)=1.+2.*lam(2)
              c(i)=-1.*lam(2)
           enddo
           a(sum1)=-2.*lam(3)
           b(sum1)=1.+2.*lam(3)+2.*lam(3)*k(2)/k(3)
           c(sum1)=-2.*lam(3)*k(2)/k(3)
           do i=sum1+1,sum2-1
              a(i)=-1.*lam(3)
              b(i)=1.+2.*lam(3)
              c(i)=-1.*lam(3)
           enddo
           a(sum2)=-2.*lam(4)
           b(sum2)=1.+2.*lam(4)+2.*lam(4)*k(3)/k(4)
           c(sum2)=-2.*lam(4)*k(3)/k(4)
           do i=sum2+1,sum3-1
              a(i)=-1.*lam(4)
              b(i)=1.+2.*lam(4)
              c(i)=-1.*lam(4)
           enddo
           a(sum3)=0.
           b(sum3)=1.
           c(sum3)=0.

           time(j)=time(j-1)+dt

           Tb(1)=Tb(1)+2.*lam(1)*heat*dx(1)/k(1)

           call tridag(a,b,c,Tb,delT,totnodes)

           do i=1,totnodes
              T(j,i)=delT(i)
              Tb(i)=delT(i)
           enddo

        enddo

***************************************************************************
* Calculation of the radiative and conductive heat fluxes.          *
***************************************************************************

        do j=1,n
           qrad(j)=epsi*sigma*T(j,1)**4
           qcond(j)=qcv(j)-qrad(j)
        enddo
```

```
************************************************************************
* End of subroutine four.                                              *
************************************************************************

        return
        end
```

Appendix P: 'five.f' FORTRAN Subroutine

```
      subroutine five(matnum,dt,nodes,epsi,k,den,cp,initT,L,T,n,
     +            time,qcv,qcond,qrad,backT,heat)

************************************************************************
* Subroutine to solve the one dimensional heat equation for a TPS   *
* material stackup consisting of five dissparate materials.         *
* This subroutine incorporates an isothermal boundary condtion at the       *
* backface of the TPS material stackup (generally the cryogenic fuel        *
* temperature) and also a constant heat flux boundary condition at the      *
* surface of the TPS material stackup.  The constant heat flux is   *
* determined solely by the temperature of the ambient air.          *
************************************************************************

      implicit none
      integer i,j,n,nodes(6),sum1,sum2,sum3,totnodes,matnum,count
      real*8 dt,k(6),den(6),cp(6),initT,L(6),dx(6),sigma,epsi,eps
     1,alpha(6),lam(6),a(400),b(400),c(400),T(10000,400),normf
     1,normdelT,f(400),delT(400),twonorm,Tradeq(10000),time(10000)
     1,qcond(10000),qrad(10000),qcv(10000),g(24),obj,x(12),Tb(400)
     1,backT,heat

************************************************************************
* Calls linint subroutine to linearly interpolate convective heat flux      *
* and Radiation Equilibrium temperature values using user specified *
* time and time step for analysis.                                  *
************************************************************************

      call linint(n,dt,time,qcv,Tradeq)

************************************************************************
* Define the mathematical constants to be used in the calculations. *
************************************************************************

      eps=1.e-6
      sigma=5.67061e-8
      do i=1,matnum
         dx(i)=L(i)/(nodes(i)-1.)
         alpha(i)=k(i)/(den(i)*cp(i)*1000.)
         lam(i)=alpha(i)*dt/(dx(i)**2)
      enddo
      sum1=nodes(1)+nodes(2)
      sum2=sum1+nodes(3)
      sum3=sum2+nodes(4)
      totnodes=nodes(1)+nodes(2)+nodes(3)+nodes(4)+nodes(5)

************************************************************************
* Specify the initial conditions of the problem.                    *
************************************************************************

      do i=1,totnodes
         T(1,i)=initT
      enddo

      do i=1,totnodes
         Tb(i)=T(1,i)
      enddo

      Tb(totnodes)=backT

      do i=1,totnodes
         T(1,i)=Tb(i)
      enddo

************************************************************************
* Initialize time array.                                            *
************************************************************************

      time(1)=0.
```

```
************************************************************************
* Do loop to solve for the temperature profiles for each time step j   *
************************************************************************

       do j=2,n

************************************************************************
* Setup the tridiagonal matrix to be used in the Thomas Algorithm.     *
************************************************************************


          a(1)=-2.*lam(1)
          b(1)=1.+2.*lam(1)
          c(1)=0.
          do i=2,nodes(1)-1
             a(i)=-1.*lam(1)
             b(i)=1.+2.*lam(1)
             c(i)=-1.*lam(1)
          enddo
          a(nodes(1))=-2.*lam(2)
          b(nodes(1))=1.+2.*lam(2)+2.*lam(2)*k(1)/k(2)
          c(nodes(1))=-2.*lam(2)*k(1)/k(2)
          do i=nodes(1)+1,sum1-1
             a(i)=-1.*lam(2)
             b(i)=1.+2.*lam(2)
             c(i)=-1.*lam(2)
          enddo
          a(sum1)=-2.*lam(3)
          b(sum1)=1.+2.*lam(3)+2.*lam(3)*k(2)/k(3)
          c(sum1)=-2.*lam(3)*k(2)/k(3)
          do i=sum1+1,sum2-1
             a(i)=-1.*lam(3)
             b(i)=1.+2.*lam(3)
             c(i)=-1.*lam(3)
          enddo
          a(sum2)=-2.*lam(4)
          b(sum2)=1.+2.*lam(4)+2.*lam(4)*k(3)/k(4)
          c(sum2)=-2.*lam(4)*k(3)/k(4)
          do i=sum2+1,sum3-1
             a(i)=-1.*lam(4)
             b(i)=1.+2.*lam(4)
             c(i)=-1.*lam(4)
          enddo
          a(sum3)=-2.*lam(5)
          b(sum3)=1.+2.*lam(5)+2.*lam(5)*k(4)/k(5)
          c(sum3)=-2.*lam(5)*k(4)/k(5)
          do i=sum3+1,totnodes-1
             a(i)=-1.*lam(5)
             b(i)=1.+2.*lam(5)
             c(i)=-1.*lam(5)
          enddo
          a(totnodes)=0.
          b(totnodes)=1.
          c(totnodes)=0.

          time(j)=time(j-1)+dt

          Tb(1)=Tb(1)+2.*lam(1)*heat*dx(1)/k(1)

          call tridag(a,b,c,Tb,delT,totnodes)

          do i=1,totnodes
             T(j,i)=delT(i)
             Tb(i)=delT(i)
          enddo

       enddo

************************************************************************
* Calculation of the radiative and conductive heat fluxes.             *
```

```
************************************************************************

        do j=1,n
           qrad(j)=epsi*sigma*T(j,1)**4
           qcond(j)=qcv(j)-qrad(j)
        enddo

************************************************************************
* End of subroutine four.                                              *
************************************************************************

        return
        end
```

References

1. Anderson, Dale A., Petcher, Richard H., Tannehill, John C., *Computational Fluid Mechanics and Heat Transfer*, Taylor and Francis, 1997.
2. Bejan, Adrian, *Heat Transfer*, John Wiley & Sons, Inc., 1993.
3. Chapra, S., Canale, R., *Numerical Methods for Engineers with Programming and Software Applications*, McGraw-Hill, 1998.
4. Cowart, Karl K., "A Method for Integrating Aeroheating into Conceptual Reusable Launch Vehicle Design,,, Georgia Institute of Technology, Atlanta, GA.
5. Humble, R., Henry, G., Wiley, J., *Space Propulsion Analysis and Design*, McGraw-Hill, 1995.
6. Melanson, J., "PSUpload Perl Script,,, http://www.perlservices.net/en/programs/psupload/index.shtml.
7. Nyhoff, L., Leestma, S., *FORTRAN 77 for Engineers and Scientists with an Introduction to FORTRAN 90*, Prentice-Hall, Inc., 1996.
8. Petley, D., Yarrington, P., "Design and Analysis of the Thermal Protection System for a Mach 10 Cruise Vehicle,,, 1996 JANNAF Propulsion & Joint Subcommittee Meetings, December 9-13, 1996, Albuquerque, NM.
9. Rasky, D., "Thermal Protection Systems Expert and Material Properties Database,,, http://tpsx.arc.nasa.gov/tpsxhome.shtml.