Free Molecular Heat Transfer Programs for Setup and Dynamic Updating the Conductors in Thermal Desktop

Eric T. Malroy: eric.t.malroy@nasa.gov
Johnson Space Center
2101 NASA Road 1
Houston, TX 77058

Abstract

Thermal Desktop has the capability of modeling free molecular heat transfer (FMHT), but limitations are observed when working with large models during transient operation. To overcome this limitation, a MatLab program was developed that processes the Thermal Desktop free molecular conductors. It sets up the logic and arrays used to automate the updating of the conductors while SINDA/FLUINT is running using Fortran subroutines. The theory of the free molecular heating is presented along with the process required to setup the conductors, arrays and logic in Thermal Desktop.

Background

Thermal Desktop [1] has the capability of modeling free molecular heat transfer (FMHT) by using the RadCad radiation modules. The software underneath Thermal Desktop that actually solves the equations is SINDA/Fluint [2]. The user specified surface properties can account for the accommodation coefficient of FMHT and the user has to supply a constant term to account for the rest of the conductor. Although this method is possibly useful to model a small number of FMHT conductors, it quickly becomes laborious and impractical for large transient models where the conductors change over time. This method of modeling is nonstandard given that the FMHT usually modeled is external heating [1]. These observations about the capabilities of Thermal Desktop were drawn while developing a thermal model of the test facility and telescope for the James Webb Space Telescope Project. Two requirements were identified to enable a practical FMHT modeling system useful for large FMHT models.

The first requirement is to have a semi-automated process to setup the FMHT radiation conductors. Sometimes with large models it is useful to have several radiation tasks used to setup the FMHT conductors since each task is limited to 10,000 conductors. The programs developed use arrays to store the inputs, which are limited to 10,000 entries. This can require the user to sort out duplicate conductors based on the submodel names of the surface nodes. Many radiation tasks prevent the arrays used in the logic structure from being too long. Also, the user needs to eliminate the small value conductors that have little impact on the heat transfer. A user will quickly realize that the number of conductors can quickly escalate to an extreme number when large numbers of surfaces or subdivisions are used. For example, three flat surfaces which have a 10 by 10 nodal subdivision will result in 30,000 conductors if all conductors are used. A MatLab model was developed that enables the sorting and elimination of FMHT conductors that are generated from Thermal Desktop.

The second need identified to enable a practical system useful for large FMHT models was to have the conductors updated as the gas pressure, gas temperature, and surface temperatures change in transient models. A logic structure and the modules to calculate the FMHT conductors were developed to update the conductor values during transient operation. This paper describes the FMHT programs developed for modeling large FMHT systems. Also, the setup and operation of the programs is presented.

Relation of Free Molecular Heating to the Different Modes of Fluid Heat Transfer

In modeling fluids, there are three regimes of heat transfer that are typically taught in engineering heat transfer classes: forced convection, natural convection and mixed forced and natural convection (see Table 1). Forced convection is the heat transfer resulting from fluid flowing over a surface. The velocity of the fluid over the surface results in energy transfer from either the fluid to the surface or vice versa depending on the difference in temperature of the fluid and surface. In this mode of heat transfer there are different correlations for both laminar and turbulent flow cases.

Natural convection is the transfer of energy, where the velocity of the fluid remote from the surface is essentially zero. The force due to gravity drives the fluid velocity near the surface due to density differences resulting in heat transfer between the surface and fluid. Notice that natural convection can only occur when gravity is appreciable. Microgravity conditions, which are common in space environments away from planet surfaces, prevent natural convection.

Table 1. Modes of Heat Transfer for Fluids

General Area	**Heat Transfer Mode**	**Relevant Environment**	**Description**
Continuum ($Kn < 0.01$)	Forced Convection	Fluid flows over surface	Fluid flowing over surface results in heat transfer
	Natural Convection	Gravity environments where fluid away from surface has zero velocity	Body forces cause fluid flow near surface resulting in heat transfer
	Mixed Forced & Natural	Gravity environments where there is a flowing fluid with large temperature gradients	Both fluid flow and body forces result in an accumulated larger flow resulting in heat transfer
	Conduction	Microgravity conditions	With low gravity and cases where there is no forced flow, the fluid does not move so the heat transfers through the fluid by conduction
Mixed	Mixed Free Molecular & Continuum	Space environments with low pressure ($0.01 < Kn < 0.30$)	Between the continuum and free molecular mode where both modes are active
Free Molecular	Free Molecular	Space environments with low pressure ($Kn > 0.3$)	High temperature surface imparts energy into fluid molecules which travel to other low temperature surface, thus transferring energy. Intermolecular collisions are few while traversing the distance between the surfaces.

Mixed forced and natural convection is the case where both modes of heat transfer are significant. The Grashof and Reynolds number have a relation that identifies the mixed regime:

$$\frac{Gr}{\mathrm{Re}^2} << \quad 1: \text{forced convection only} \tag{1a}$$

$$\frac{Gr}{\mathrm{Re}^2} \approx \quad 1: \text{mixed convection} \tag{1b}$$

$$\frac{Gr}{\mathrm{Re}^2} >> \quad 1: \text{natural convection only} \tag{1c}$$

$$\frac{Gr}{\mathrm{Re}^2} \approx \quad \frac{buoyancy\ \ force}{inertial\ \ force} \tag{1d}$$

Conduction heat transfer of fluids is another regime of heat transfer that is often overlooked in engineering heat transfer classes. This mode of heat transfer often occurs in microgravity environments. It can also occur in low pressure conditions near the free molecular regime. The conductor is modeled by treating the gas as a solid surface and the conductor is calculated appropriately based on the cross sectional area and length between surfaces, k*A/L.

Free Molecular heating occurs in very low density gases where the mean free path of the gas molecules is large compared to the distance between the surfaces. The Knudsen number is the ratio of the mean free path and the effective length between surfaces.

$$Kn \equiv \frac{\lambda}{L_e} \tag{2}$$

Kn	Knudsen number
λ	mean free path
L_e	effective length between surfaces

When the Knudsen number is greater than 0.30 one should use the free molecular relations to account for heat transfer. It is instructive to follow a gas molecule to see how energy is transferred between surfaces in free molecular heating. A low energy gas molecule hits the high temperature surface. The vibration of the molecules on the surface will impart greater energy into the gas molecule, typically causing it to depart the surface with a higher velocity. Given that the mean free path is large compared to the distance between the surfaces, the probability is relatively high that the gas molecule will reach the other surface without any collisions with other gas molecules. The high energy gas molecule, if the trajectory is correct, then transfers the energy to the low energy surface by hitting the surface. The impact causes the vibration of molecules to increase on the low temperature surface. Energy is therefore transferred from the high temperature surface to the low temperature surface. Large numbers of molecules will transfer energy in this fashion resulting in a significant energy transfer.

When the Knudsen number is less than 0.30 and greater than 0.01, the mode of heat transfer is mixed containing both free molecular heating and continuum heating – one of the four types, but most likely conduction or natural convection. The magnitude of heat transfer is typically greater than the free molecular, but less than the full continuum heat transfer mode.

Theory of Free Molecular Heat Transfer

The mean free path tells the mean distance that a molecule will travel in a gas before a collision with another gas molecule will occur. The relation of the mean free path to the effective length between surfaces tells if a heat transfer process is in the free molecular mode. This ratio is the Knudsen number which is shown in equation (2). The mean free path is calculated by the following relation:

$$\lambda = \left(\frac{\mu}{p}\right)\sqrt{\frac{\pi R_u T}{2\ g_c\ MW}} \tag{3}$$

λ	mean free path (MFP)
μ	gas viscosity
p	gas pressure
R_u	universal gas constant
T	temperature that causes high energy collisions (T hot surface)
g_c	units conversion
MW	molecular weight

Table 2 shows when to use the free molecular or continuum equations. Notice that when the mean free path is about a third of the effective length or greater, that the free molecular equations should be used. The continuum equations should be used when the mean free path is significantly small (1/100) compared to the effective length. The mixed mode is between the two cases and caution should be used in using the free molecular heating equations since it can give an inflated value for the heat transfer.

Table 2. Determining the Mode of Heat Transfer

Heat Transfer Mode	Determining Factor	Equations to Use
Continuum	$Kn < 0.01$	Use gas conduction, natural convection or other equations
Mixed Continuum & FM	$0.01 < Kn < 0.30$	Use FM equations with caution (you will overestimate the heat transfer)
Free Molecular	$Kn > 0.30$	Use FM equations

The basic equation for FMHT between two surfaces is the following [3]:

$$Q = G\,p\,F_a\,F_{12}\,A_1\left(T_2 - T_1\right) \tag{4}$$

G	G factor
p	gas absolute pressure
F_a	accommodation coefficient **factor**
F_{12}	view factor from surface number one to surface number two
A_1	surface area of surface number one
T_1	surface temperature of surface number one
T_2	surface temperature of surface number two

The equation is analogous to radiation heat transfer between two surfaces. This is apparent when the accommodation coefficient factor is examined. Specifically, the accommodation coefficient factor, F_a, is analogous to the F_e emissivity factor in radiation so the form of the equation can be taken from radiation handbooks. The radiation equation is the following:

$$Q = \sigma\,F_e\,F_{12}\,A_1\left(T_2^4 - T_1^4\right) \tag{5}$$

σ	Stefan-Boltzmann constant
F_e	emissivity factor
F_{12}	view factor
A_1	area of surface number one
T_2	absolute temperature of surface number two
T_1	absolute temperature of surface number one

The equations are similar except for the fourth power term. In radiation the F_e term depends on the geometry and the emissivity of the surfaces (see table 3).

Table 3. Different Geometries and the Resulting Fe Equation for Radiation

Type of radiation exchange	$\frac{1}{F_e}$ **Equation**	**Comments**
Two Surfaces that Form an Enclosure	$\frac{F_{12}\cdot(1-\varepsilon_1)}{\varepsilon_1}+1+\frac{A_1\cdot F_{12}\cdot(1-\varepsilon_2)}{\varepsilon_2\cdot A_2}$	More general case
Large (Infinite) Parallel Planes	$\frac{1}{\varepsilon_1}+\frac{1}{\varepsilon_2}-1$	$F_{12} = 1$ $A_1 = A_2$
Long (Infinite) Concentric Cylinders	$\frac{1}{\varepsilon_1}+\frac{1-\varepsilon_2}{\varepsilon_2}\cdot\left(\frac{r_1}{r_2}\right)$	$F_{12} = 1$ $\frac{A_1}{A_2}=\frac{r_1}{r_2}$
Concentric Spheres	$\frac{1}{\varepsilon_1}+\frac{1-\varepsilon_2}{\varepsilon_2}\cdot\left(\frac{r_1}{r_2}\right)^2$	$F_{12} = 1$ $\frac{A_1}{A_2}=\left(\frac{r_1}{r_2}\right)^2$

If we use the equation in Table 3 with two surfaces enclosing one another with a view factor of one, then the equation results in the following:

$$\frac{1}{F_e} = \frac{1}{\varepsilon_1}+\frac{A_1}{A_2}\cdot\left[\frac{1}{\varepsilon_2}-1\right] \qquad (6)$$

F_e	emissivity factor
ε_1	emissivity of surface number one
ε_2	emissivity of surface number two
A_1	area of surface number one
A_2	area of surface number two

The accommodation coefficient factor is analogous to the emissivity factor. The form of the emissivity factor equation chosen will depend on the geometry. For equation (6) the accommodation coefficient factor equation is the following:

$$\frac{1}{F_a} = \frac{1}{a_1}+\frac{A_1}{A_2}\cdot\left[\frac{1}{a_2}-1\right] \qquad (7)$$

F_a	accommodation coefficient **factor**
a_1	accommodation coefficient based on surface number one temperature (analogous to emissivity)
a_2	accommodation coefficient based on surface number two temperature (analogous to emissivity)

A_1	area of surface number one
A_2	area of surface number two

The accommodation coefficient, **a,** is analogous to emissivity. It represents the ratio of the actual energy transfer and is a function of the surface temperature. A curve fit of the equation for helium for the temperature ranging from 20 to 300 K is the following [4]:

$$a_i = 1.30168 \cdot T_i^{-0.262249} \qquad (8)$$

T_i	absolute temperature of surface i
a_i	accommodation coefficient for surface i

The last term needed to calculate the FMHT for a surface is the G term which is the following:

$$G = \frac{\gamma + 1}{\gamma - 1} \cdot \sqrt{\frac{g_c \; R_u}{8\,\pi \cdot MW \cdot T}} \qquad (9)$$

γ	ratio of specific heats (based on hot surface)
g_c	units conversion constant
R_u	universal gas constant
MW	molecular weight
T	gas absolute temperature (hot surface)

Once all terms of the free molecular equation are calculated, equation (4) can be used to find the heat transfer. The value of the conductor is the following:

$$G_{cond} = G\, p\, F_a\, F_{12}\, A_1 \qquad (10)$$

Operation of Thermal Desktop

Thermal Desktop generates the free molecular conductors by using RadCad. A new radiation analysis group can be generated by selecting the main menu item “Thermal” and selecting the “Radiation Analysis Groups …” menu (see figure 1).

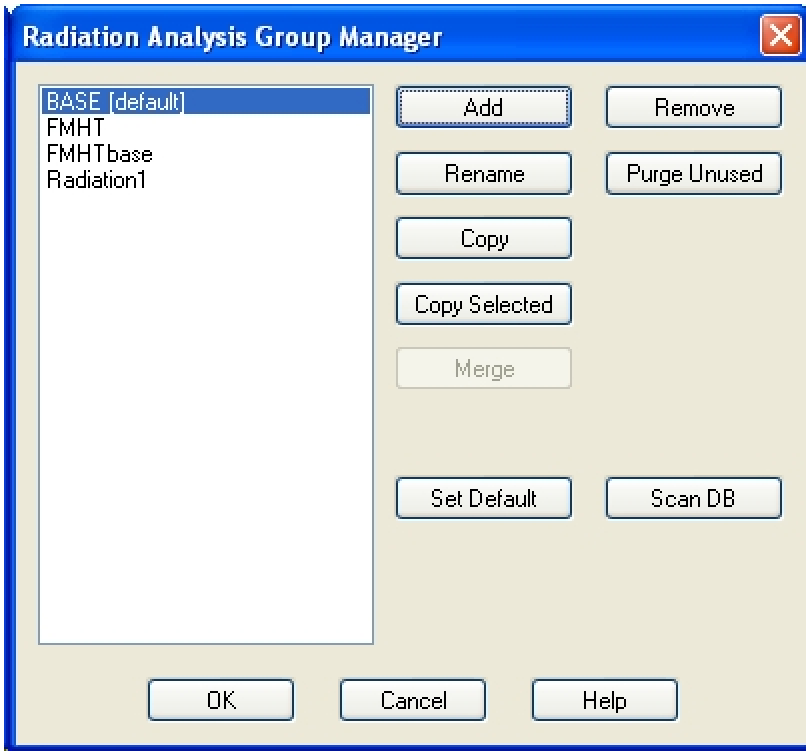
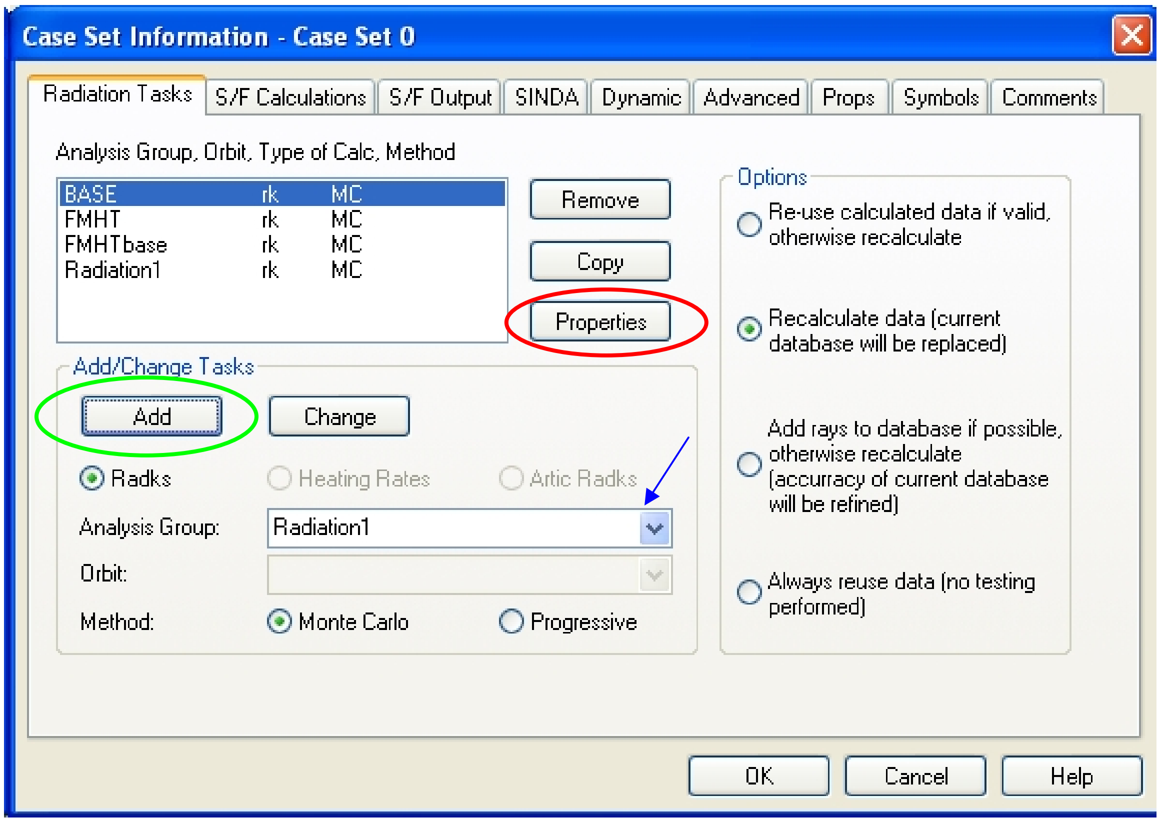


Figure 1. Radiation Analysis Group Manager Page.

Notice that three analysis groups have been added in Figure 1: FMHT, FMHTbase, Radiation1. The "BASE" group will be shown when the "Radiation Analysis Group Manager" window has been opened for the first time. The user will add additional groups as desired. **It should be noted here that the properties cannot be changed here under the "Radiation Analysis Group Manager". It is used to add a new group, change the group name, remove groups, copy groups, purge unused groups and set the default group.** Typically, this is where the user defines the name of the RadCad radiation groups. The mode of heat transfer for the group can be defined as free molecular heating under the properties menu in the "Case Set Manager" section.

Under the "Case Set Manager" menu the user can find the "Radiation Tasks" tab which brings up the page allowing the user to generate radiation tasks (see figure 2). This needs to be completed after the name has been added to the "Radiation Analysis Group Manager" as discussed above. Figure 2 shows that four Radiation tasks have been added: BASE, FMHT, FMHTbase, and Radiation1. These were added to the tasks by selecting them individually from the "Analysis Group" pull down menu (see the blue arrow in Figure 2). Next, the "Add" button is pushed to put them in the analysis group task list (see the green ellipse around the button in Figure 2). The properties of the analysis group are set by selecting the analysis group from the list and then pushing the "properties" button (see the red ellipse in Figure 2). The "Radiation Analysis Data" page pops up when the button is pushed.


Case Set Information - Case Set 0

Radiation Tasks | S/F Calculations | S/F Output | SINDA | Dynamic | Advanced | Props | Symbols | Comments

Analysis Group, Orbit, Type of Calc, Method

BASE	rk	MC
FMHT	rk	MC
FMHTbase	rk	MC
Radiation1	rk	MC

Remove | Copy | Properties

Add/Change Tasks

Add | Change

[✓ Radks] Heating Rates | Artic Radks

Analysis Group: Radiation1

Orbit:

Method: [✓ Monte Carlo] Progressive

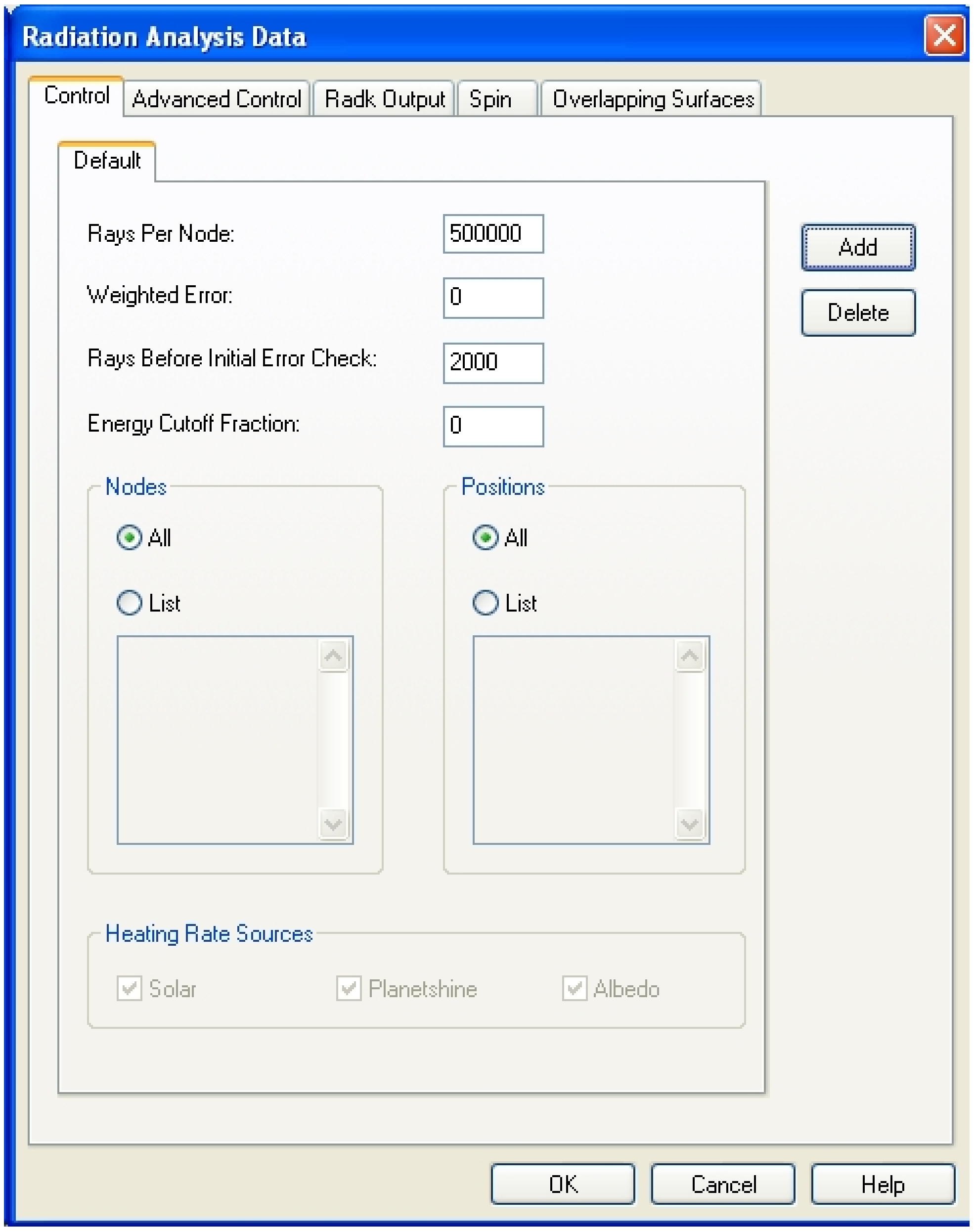
Options

- Re-use calculated data if valid, otherwise recalculate
- [✓ Recalculate data (current database will be replaced)]
- Add rays to database if possible, otherwise recalculate (accurracy of current database will be refined)
- Always reuse data (no testing performed)

OK | Cancel | Help

Figure 2. “Radiation Tasks” page can generate a number of radiation tasks.

The “Radiation Analysis Data” page has a number of menu tabs that allows the properties of the tasks to be set (see Figure 3). The “Control” tab page allows the number of rays shot from the nodes to be specified which defines the view factor or Bij value. Either the view factor will be calculated or the Bij value will be calculated based on the method chosen under the “Radiation Calculation” menu, when this method of calculation is chosen to generate the conductors. This menu item is under the main “Thermal” menu. The user can also specify a select number of nodes to shoot additional rays from on the “Control” tab page. The energy cutoff fraction can also be specified, which will prevent the low energy conductors from being generated. The help menu on the page will explain all the options more clearly.



Rays Per Node: 500000

Weighted Error: 0

Rays Before Initial Error Check: 2000

Energy Cutoff Fraction: 0

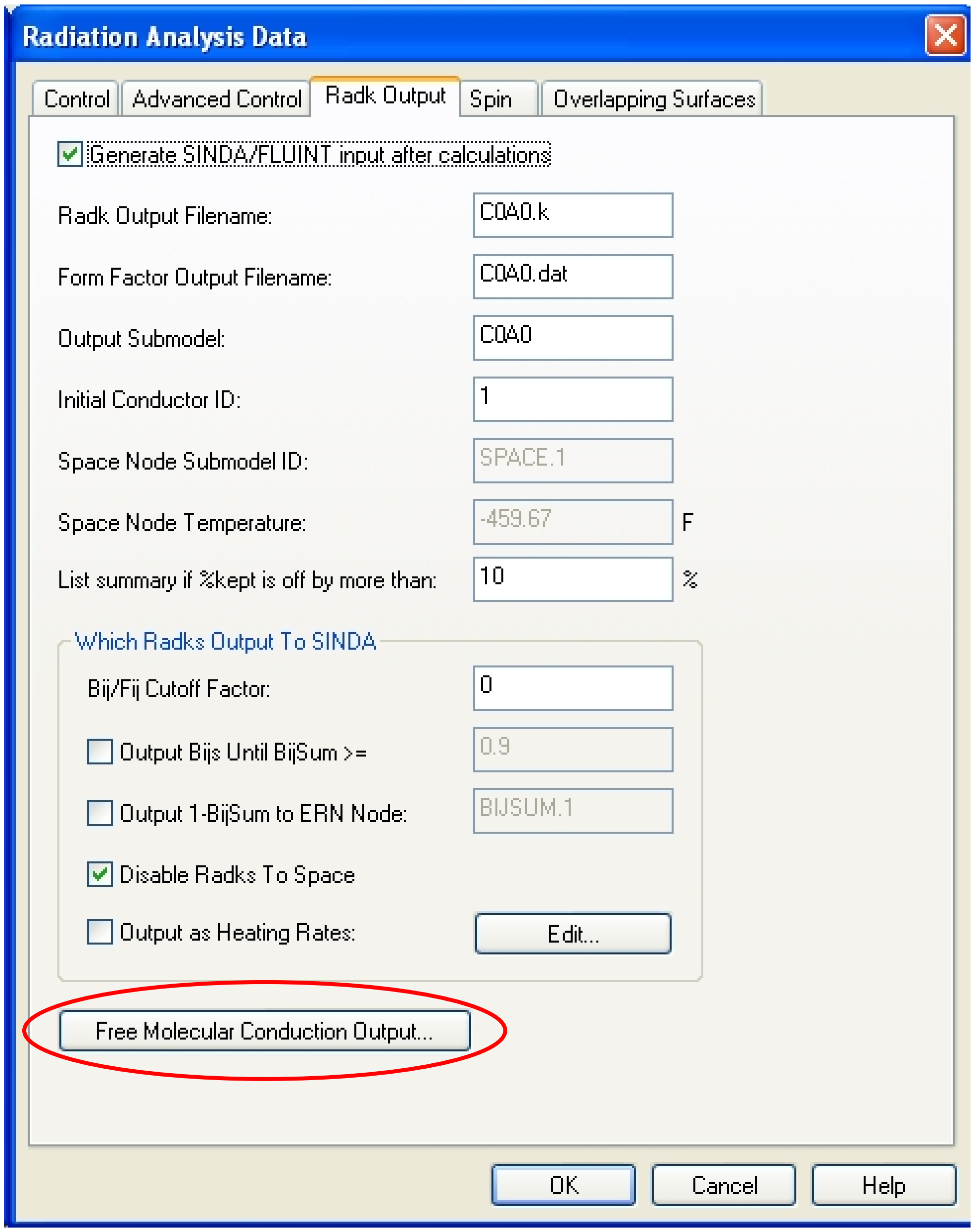
Nodes: [✓ All] List

Positions: [✓ All] List

Heating Rate Sources: [✓ Solar] [✓ Planetshine] [✓ Albedo]

Figure 3. "Radiation Analysis Data" page sets the properties of radiation groups.

The "Radk Output" tab opens a page that can specify the radiation group to be free molecular (see Figure 4). The red ellipse in Figure 4 shows the "Free molecular Output" button. This button will bring up the popup window shown in Figure 5. The conductors can be specified to be linear which is required of free molecular conductors. The free molecular multiplier can also be specified on this window. It is unnecessary to define this multiplier when the programs defined in this paper are used. The programs will update the conductors during operation.



[✓] Generate SINDA/FLUINT input after calculations

Radk Output Filename: C0A0.k

Form Factor Output Filename: C0A0.dat

Output Submodel: C0A0

Initial Conductor ID: 1

Space Node Submodel ID: SPACE.1

Space Node Temperature: -459.67 F

List summary if %kept is off by more than: 10 %

Which Radks Output To SINDA

Bij/Fij Cutoff Factor: 0

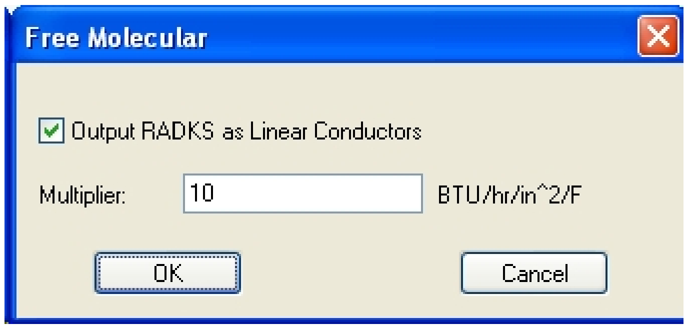
[ ] Output Bijs Until BijSum >= 0.9

[ ] Output 1-BijSum to ERN Node: BIJSUM.1

[✓] Disable Radks To Space

[ ] Output as Heating Rates:

Figure 4. The "Radiation Analysis Data" page also has the tab menu page "Radk Output" which has the "Free Molecular Conduction Output" button (red ellipse highlights button).


Free Molecular

☑ Output RADKS as Linear Conductors

Multiplier: 10 BTU/hr/in^2/F

Figure 5. The "Free Molecular" popup window enables the conductors to be linear.

There are essentially two ways to generate the radiation or free molecular conductors once the radiation tasks (including free molecular tasks) are defined. The first is to use the "Radiation Calculation" menu found under the main "Thermal" menu (see Figure 6).

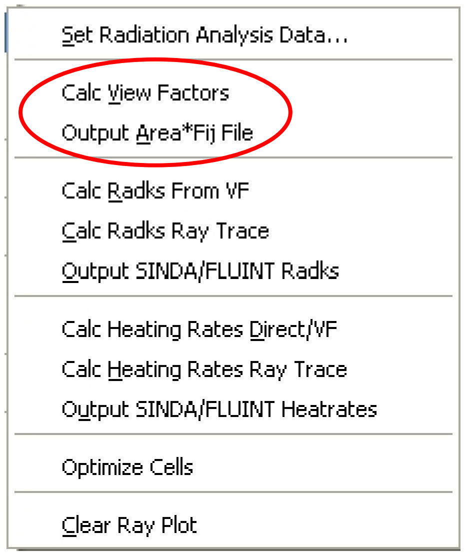
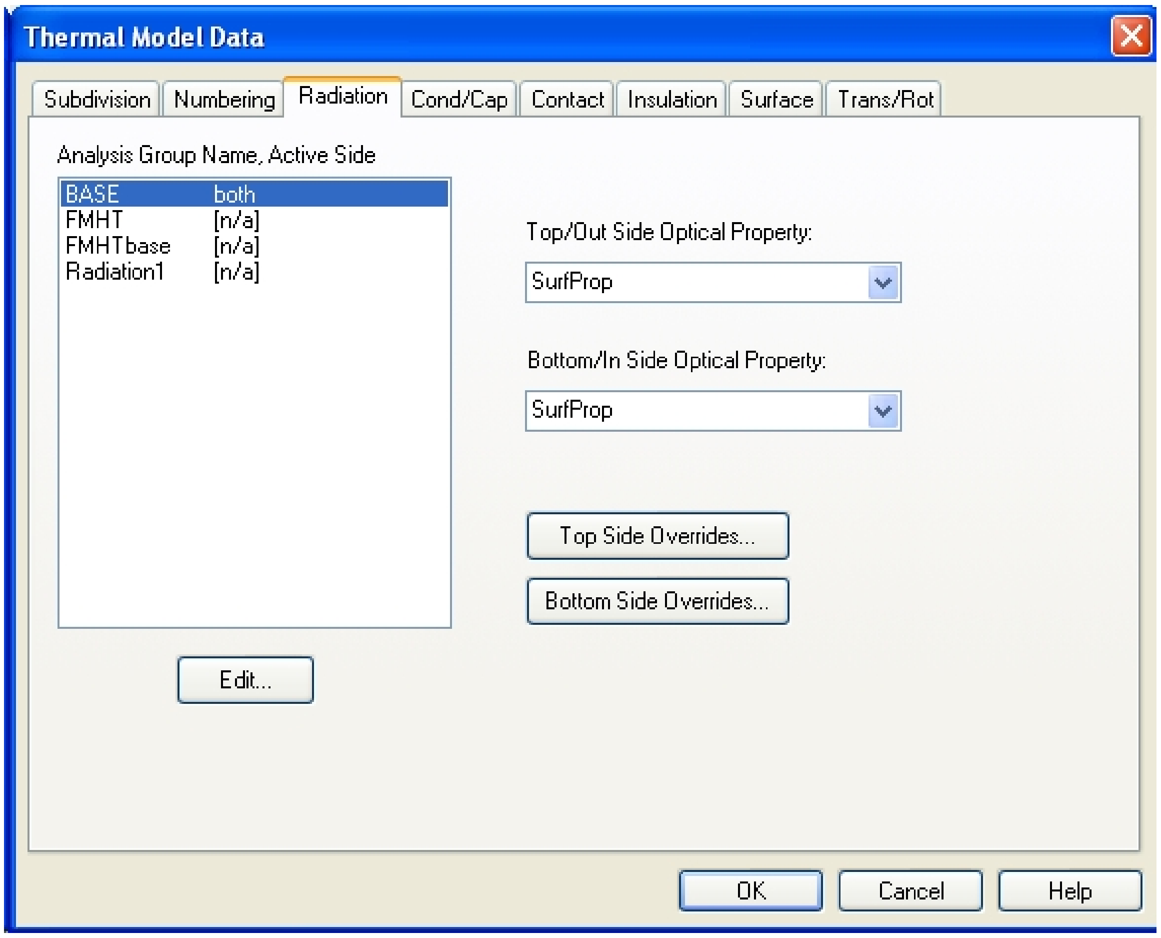

Set Radiation Analysis Data...
Calc View Factors
Output Area*Fij File
Calc Radks From VF
Calc Radks Ray Trace
Output SINDA/FLUINT Radks
Calc Heating Rates Direct/VF
Calc Heating Rates Ray Trace
Output SINDA/FLUINT Heatrates
Optimize Cells
Clear Ray Plot

Figure 6. The menu items are found under the "Radiation Calculations" menu. The red ellipse shows the two options used to generate the area file AREAFIJ.ar and free molecular conductors file AREAFIJ.DAT for the programs in this paper.

Prior to using these options found in Figure 6, the user should apply the radiation group to the appropriate surfaces. This is accomplished by first selecting the surface and then selecting the "Edit" menu under the main "Thermal" menu. Also the "Edit" button can be selected from the toolbar button that shows a star and a pencil. When this toolbar button is highlighted the description "Edit Any Thermal Desktop Object" should be highlighted. The "Thermal Model Data" page should appear once the edit button is pushed. Under the "Radiation" tab the active surfaces are defined for each radiation or free molecular group. Figure 7 shows the "Radiation" tab of the "Thermal Model Data" page. Figure 7 shows both surfaces active for the BASE group. The groups FMHT, FMHTbase and Radiation1 are not applied to the surface of either side.



Analysis Group Name, Active Side

BASE	both
FMHT	[n/a]
FMHTbase	[n/a]
Radiation1	[n/a]

Top/Out Side Optical Property: SurfProp

Bottom/In Side Optical Property: SurfProp

Figure 7. "Thermal Model Data" page that associates a surface with the radiation analysis group.

After the radiation groups are applied to the surfaces it is then possible to generate the output files found in Figure 6. To output the radiation or free molecular group, it must first be set as the default group under the "Radiation Analysis Group Manager" page found in Figure 1. The desired radiation or free molecular group needs to be selected and the "Set Default" needs to be pushed. Once this is complete, the menu items of Figure 6 can be selected to output files.

The second way to generate the radiation or free molecular conductors is to use the "Case Set Manager" page. This option automates the output of the conductors, just prior to the Sinda/Fluint run. This method is not the choice when the free molecular programs presented in this paper are used. Further details about Thermal Desktop or Sinda/Fluint are found in the user manuals [1,2].

Operation of the MatLab Program

Prior to running the MatLab program FMHTPRE.m, the area file and the conductors need to be output from Thermal Desktop. These files are found in Appendix A and B. The Radiation Calculation menu under the main menu "Thermal" shows the RadCad modules that will output the area file and the conductors (see red ellipse in Figure 6) . First, the user needs to set the default free molecular group from the "Radiation Analysis Group Manager" window (see Figure 1). All conductors are generated based on the free molecular group selected. The second task is to calculate the view factors, which is the first menu item circled in red in Figure 6. The name on the menu item is "Calc View Factors" and the file AREAFIJ.ar is output when this menu item is selected. The directory "<radiation group name>.rcf" is also generated by this task which is needed for outputting the conductors. Appendix A shows a shortened version of this file. The submodel name is given along with the node number and the associated area of the surface.

The third task is to generate the conductors by selecting the "Output Area*Fij File" menu item (see second menu item in red ellipse of Figure 6). The file AREAFIJ.DAT is generated by this action. Appendix B shows a shorten version of this file. Once this is complete, the user is ready to run the MatLab program FMHTPRE.m. This file is found in Appendix D along with some of the other modules required by the program. From the command line of the MatLab window, the user should run the program FMHTPRE.m by typing in the directory and MatLab program name. For example, the user could type in "C:\Workarea\FMHT\FMHTPRE" and hit "return" to run the MatLab program from the command line if the MatLab file FMHTPRE.m is in the directory. This causes the MatLab program to execute.

The MatLab program first prompts for the base value for the array numbers. Table 4 shows a case where the base value is 700. The MatLab program next prompts for the increment value of the arrays. For the array number shown in Table 4 the increment value was 10. The arrays that are generated by the program are used with the logic in SINDA FLUINT. Once the data is generated, the arrays can be pasted into the different locations as shown in Table 4 in the "Location" column. For example, arrays 710 through 780 (or whatever the base value and increment value is used) would all go in the ARRAY DATA HEADER for the specified submodel. Figure 8 shows the "SINDA" tab page in the "Case Set Information" menu where the information can be pasted manually in the ARRAY DATA of the appropriate submodel. The string array should be pasted in the CARRAY DATA of the appropriate submodel (array 800 in Table 4).

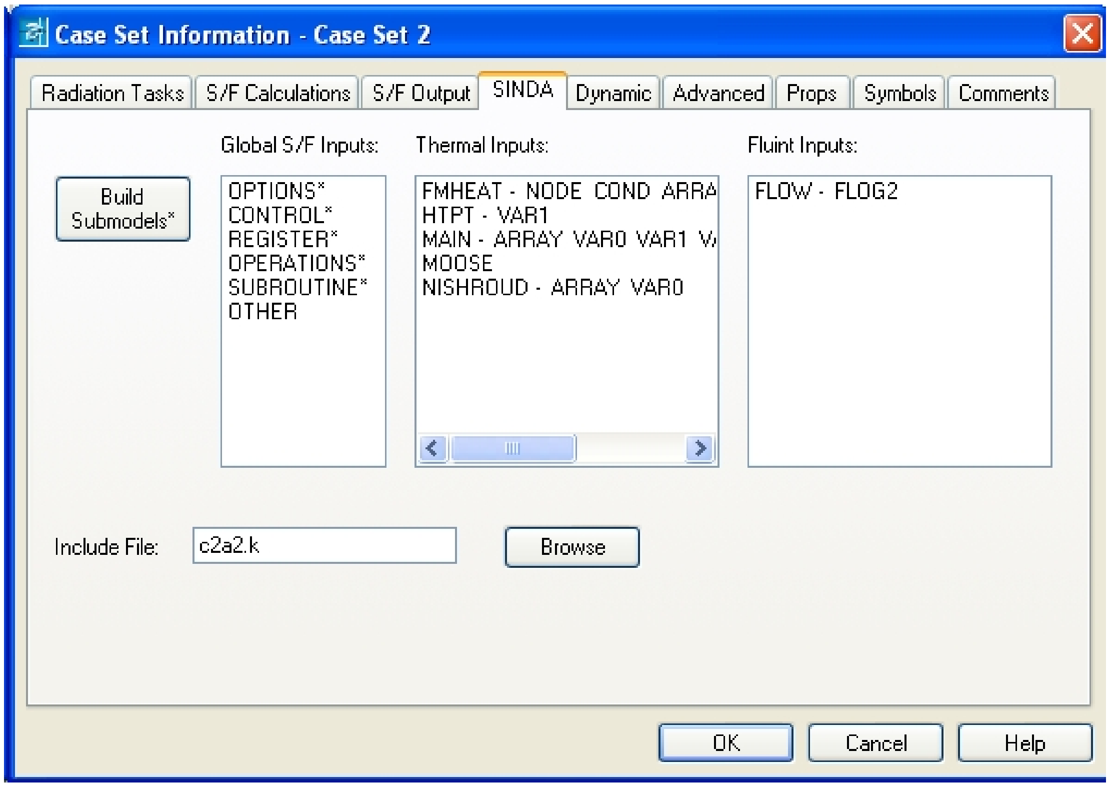
The MatLab program then prompts for the name of the submodel that contains the viscosity array. It needs to be in a different submodel than arrays 710 through 790. Table 4 shows the value of 333. This number can be selected arbitrarily, but it must not conflict with the other arrays. The program next prompts for the input and output file names. The input files must exist and contain the information that was previously generated (see Figure 6). Also, MatLab must recognize the directories and the files. The output file name can be whatever the user desires, but it must not overwrite another file name. The next prompt asks the user if the header data should be processed. "No" should always be input for this prompt since the function is not needed.

TABLE 4. Arrays constructed by the MatLab program

Array Number	Array Description	Array Type	Location
User Specified (e.g. 333)	Viscosity	Real	Different submodel than 700 ARRAY DATA
710	Conductor number a	Integer	ARRAY DATA of user specified submodel
720	First node number of conductor	Integer	“
730	Second node number of conductor	Integer	“
740	Area associated with first node number	Real	“
750	Area associated with the second node number	Real	“
760	View factor	Real	“
770	Effective length array	Real	“
780	Array that has the index of the submodel name of node 1 found in the CARRAY	Integer	“
790	Array that has the index of submodel name of node 2 found in the CARRAY	Integer	“
800	CARRAY data that has strings listing the submodels	String	CARRAY DATA of user specified submodel

The next prompt is for the cutoff value for the conductors. This input enables the conductors to be eliminated from the model which are smaller than the cutoff value. The very small conductors will tend to have less impact on the model and can typically be deleted without much effect. A warning should be given here; however, since a small conductor with a very high temperature difference between the nodes can have a significant impact on the results. The user needs to use this option with care and good judgment. Also, the nodal subdivision can determine when the free molecular conductors are retained or not. This can cause important conductors to be eliminated from a model when actually the conductors have a significant impact on the results.

The user also has the option of eliminating conductors based on the names of the submodels associated with the nodes of the conductors. There are cases where the temperature of two submodels will be nearly the same. Rather than including these conductors and increasing the solve time, the user may decide to eliminate them. Another case may be when the temperature of a submodel is nearly the same. The user can specify that the free molecular conductors attached between the nodes of the same submodel be eliminated. Again, this will speed up the solve time. A final example is the case where the view factor between the surfaces of two submodels is small. The free molecular conductors can be eliminated for this case also. Engineering judgment is needed when using this option.



Global S/F Inputs:

```
OPTIONS*
CONTROL*
REGISTER*
OPERATIONS*
SUBROUTINE*
OTHER
```

Thermal Inputs:

```
FMHEAT - NODE COND ARRA
HTPT - VAR1
MAIN - ARRAY VAR0 VAR1 V
MOOSE
NISHROUD - ARRAY VAR0
```

Fluint Inputs:

```
FLOW - FLOG2
```

Include File: c2a2.k

Figure 8. The SINDA tab under the "Case Set Information" page

The user also has the option of selecting the conductor numbers which will not be eliminated no matter the names of the submodels of the two nodes. This option allows for critical conductors to be retained. The last prompt is the option specifying the effective length based on the submodel names of the two nodes of the conductor. This is only an approximation and is used when the free-molecular Fortran subroutines check the mode of heat transfer. Ideally the user will find the distances between the surfaces of the conductors, but most the time this is not practical when the free molecular conductors are many. The user then can input the worst case conditions to see when the mode of heat transfer changes from free molecular to continuum. If this should happen the user will need to modify the free molecular Fortran subroutines to switch the modeling so the appropriate continuum equations are used. Possibly, logic will need to be incorporated to turn off the free molecular submodel and to turn on another submodel of conductors which model the appropriate continuum equations. If it is known that the heat transfer mode is always in the free molecular region, it is important that the user select effective lengths appropriately so the free molecular equations are used without any added factor applied to the free molecular conductor. Appendix C shows the inputs and prompts for the MatLab m-file FMHTPRE.m. Appendixes A and B shows, respectively, the file AREAFIJ.ar and AREAIJ.DAT (partial file). The output from FMHTPRE.m is found in Appendix E.

Setup of Thermal Desktop for Modeling Dynamic Free Molecular Heat Transfer

Once the output files are generated in MatLab the user has to manually insert portions of the data in the correct "Global Sinda Inputs" and "Thermal Inputs" sections of Thermal Desktop (see Figure 8). The submodel name where the conductors are placed is the third input specified in the MatLab program FMHTPRE.m. If the input file InputFMHT.dat (see Appendix C) was input for FMHTPRE.m, then these conductors should be placed in submodel "MAIN" as well as arrays 510, 520, 530, 540, 550, 560, 570, 580, 590, 333 and 600 (see Appendix E). Figure 9 shows the different header names used in a SINDA submodel. The conductors should be copied to the "Conductor" header inputs page while the arrays should be copied to the "Array" header inputs page (see figure 9). Array 600 should be copied to the "Carray" header inputs page of submodel Main, while the viscosity array should be copied to the appropriate submodel that the user supplied in the fourth input to the MatLab program FMHTPRE.m. The array should be copied to the "Array" header page of that submodel. All the calls to the subroutine "ARYTRN" should be copied to the global SINDA/FLUINT input "OPERATIONS" page (see Figure 8). The calls to "ARYTRN" put the memory locations of the pointers of the arrays into array 333 to enable the rapid updating of the conductors. The call to FMHTCOND should be located where the user wants the conductors updated. For a steady state run this should be in VARIABLES 0. For a transient run, the location is in VARIABLES 0, but the user may opt to speed up the solve time by putting the call in the OUTPUT header (see Figure 9). The conductors will only update as often as the output calls are called. The "SINDA Control Information" page controls the length of time between output calls.

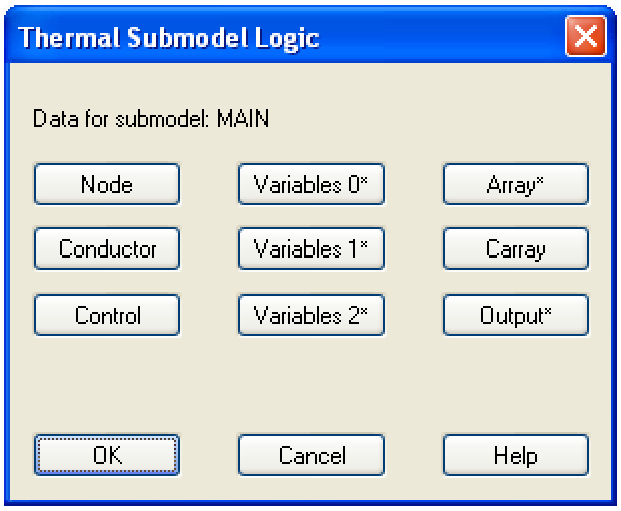


Figure 9. The Headers used for submodel MAIN. This menu is activated by double clicking on the submodel name under the "Thermal Inputs" section observed in the previous figure.

Appendix F shows the Fortran modules which need to be copied to the "Subroutine" Global SINDA/FLUINT input page (see figure 8). The first subroutine is used to calculate the Knudsen number. This determines the mode of heat transfer. The subroutine FMHT calculates the conductor value and assumes that helium is the fluid. If helium is not the gas then the equations for the "ATERM1" and "ATERM2" variables need to be modified for the appropriate gas [5]. The module is also written assuming English units, which is easy to modify. The subroutine FMHTCOND is the main module. One weakness of the module is when the heating mode is the mixed or continuum. Currently, the subroutine uses the free molecular heating rate for the mixed mode and uses a factor times the free molecular heating rate for the continuum mode. The user may want to have two submodels: one to model the free molecular heating and the other the continuum . The user would have to setup the logic to shut off and on the appropriate submodels, depending on the type of heating. The FMHTCOND subroutine could be modified to accomplish this. The current module works if the mode of heat transfer is in the free molecular mode, or near this mode in the mixed region. Outside of this region, the user may need to modify the module by changing the factor applied to the free molecular heating rate.

Once the Thermal Desktop model is setup, the analysis can begin by activating Thermal Desktop. SINDA/FLUINT runs underneath the Thermal Desktop GUI solving the model. The user has to be careful in the setup of the model to avoid excessive solve times when the free molecular conductors are used.

Conclusion

The programs, arrays and logic structure were developed to enable the dynamic update of conductors in thermal desktop. The MatLab program FMHTPRE.m processes the Thermal Desktop conductors and sets up the arrays. The user needs to manually copy portions of the output to different input regions in Thermal Desktop. Also, Fortran subroutines are provided that perform the actual updates to the conductors. The subroutines are setup for helium gas, but the equations can be modified for other gases. The maximum number of free molecular conductors allowed is 10,000 for a given radiation task. Additional radiation tasks for FMHT can be generated to account for more conductors. Modifications to the Fortran subroutines may be warranted, when the mode of heat transfer is in the mixed or continuum mode. The FMHT Thermal Desktop model should be activated by using the "Case Set Manager" once the model is setup. Careful setup of the model is needed to avoid excessive solve times.

References

1. Panczak, T., Ring, S., Welch, M., Johnson, D., “Thermal Desktop User’s Manual, CAD Based Thermal Analysis and Design”, Version 5.0, C&R Technologies, Oct., 2006.
2. Cullimore, B., Ring, S., Johnson, D., “User’s Manual SINDA/FLUINT General Purpose Thermal/Fluid Network Analyzer”, Version 5.0, C&R Technologies, Oct., 2006.
3. Sutherlin, Steven, “The X-Ray Calibration Facility (XRCF) Thermal Characterization Test Cycle 3 Data Correlation”, Report No.: MG-02-878, George C. Marshall Space Flight Center Engineering Directorate, Contract NAS8-00187, Sverdrup, A Jacobs Company, Dec 20, 2002.
4. Cleveland, Paul E., E-mail correspondence with attached work, Energy Solutions Inter., Apr 14, 2005. The curve fit equation was provided in the work.
5. Barron, Randall, “Cryogenic Heat Transfer”, Taylor & Francis, 1 edition, May 1 1999, pp 250.

Index to Appendixes

A …….....	AREAFIJ.ar: Thermal Desktop generates this file showing the surface area associated with the node number and submodel name. It is needed as an input file for the MatLabA program FMHTPRE.m. Only part of the file is shown here since the full file is very large. This partial file shows the general format of the file.
B ………	AREAFIJ.DAT: This example file shows the free molecular conductors generated by Thermal Desktop. This input file is a needed for the MatLab program FMHTPRE.m.
C ………	InputFMHT.dat: This file shows the inputs for the MatLab program FMHTPRE.m which are shown in bold red-color font.
D ………	FMHTPRE.m: The MatLab program used to setup the free molecular conductors and the arrays used by the SINDA logic. Files AREAFIJ.ar and AREAFIJ.DAT are needed for the execution of this file.
E ………	OutputFMHT.dat: This file shows the results generated by the MatLab program FMHTPRE.m.
F ………	Fortran Free Molecular Subroutines

APPENDIX A

AREAFIJ.ar

Here is a condensed version of file AREAFIJ.ar. It is the file that shows the area associated with each node. This file is only an example file.

```
     MAIN.1    0.455347
     MAIN.2    0.910694
     MAIN.3    0.910694
     MAIN.4    0.910694
     MAIN.5    0.910694
     MAIN.6    0.910694
     MAIN.7    0.910694
     MAIN.8    0.910694
 NISHROUD.1     12.8991
 NISHROUD.2     12.8991
 NISHROUD.3     12.8991
 NISHROUD.4     12.8991
 NISHROUD.5     12.8991
 NISHROUD.6     12.8991
 NISHROUD.7     12.8991
 NISHROUD.8     12.8991
 NISHROUD.9     12.8991
NISHROUD.10     12.8991
NISHROUD.11     12.8991
NISHROUD.12     12.8991
```

APPENDIX B

AREAFIJTEST.DAT

This file is generated by Thermal Desktop.

```
HEADER CONDUCTOR DATA, MAIN
C      SINDA/FLUINT data created with Thermal Desktop 4.8 Patch 4s
C      Generated on Sun Apr 22 18:10:55 2007
C
C         Generated from database FMHT.rcf
C         Bij Cutoff factor:      0.0010000
C         Free Molecular Output Enabled
C         Conductor Units are: BTU/hr/F
C         Free Molecular Multiplier 0.17611 BTU/hr/ft^2/F
C         (more information at end of file)
C
C  PSTOP causes the radks not to be echoed to the preprocessor pp.out file
PSTOP
C    Symbol names, values, comments
C    AddHT, 12.0, additional height to include for Ni Pane...
C    AddMylar, 2.0, distance between Ni panel and Mylar shee...
C    AngHt, 10*pi/180, Angle for the heating xy plane
C    AngHtMx, Angle - (LHTBOX/(pi*RadHe_in*2))*360.0, Maximum angle to move the surface
heat l...
C    Angle, 22.5, Angle of Helium shrouds [degrees]
C    ArcLeng, pi*Radius_He*2*Angle/360
C    ArcL_in, ArcLeng*12.0, Arc Length in Inches
C    AREA, pi*(Diam)^2/4.0, Flow Area of tubes [in^2]
C    AREAHDR, pi*DiamHdr^2/4, Cross sectional area of header
C    Diam, .625, Tube diameter [in]
C    DiamHdr, 1.5, Diameter of header [in]
C    EndHt, Ht_in + AddHT*2, This is used to increase the Ni panel, m...
C    Gthck, 0.125
C    Height, 41, Height of Helium Shrouds
C    HeTemp, 15, Temperature Helium Shrouds [K]
C    Ht_in, Height*12, Height of shroud in inches
C    LengthHT, 100, Length used to position the heating surf...
C    LHTBOX, 12.0, Edge of Heating Box used to model the `p...
C    LngHTMX, Ht_in - LHTBOX, Maximum height to apply point heating
C    LTube, Ht_in/NTube, Length of individual tube [in]
C    MAXAVDT, 0.0, Maximum Average Temperature of the GHe p...
C    MAXDT, 0.0, Maximum temperature difference over the ...
C    mdot, 14.5, Mass flow rate [gram/sec]
C    NiTemp, 90, Nitrogen shroud temperature [K]
C    Npipes, 5, Number of tubes in shroud
C    NTube, 20, Number of TD Tubes per height
C    Pout, 14.0, Outlet Pressure
C    QQheat, .25, Heat Load on surface [Watt/ft^2]
C    QQPOINT, 0.5, Heat Load on Point Surface [Watt/ft^2]
C    RadHe_in, Radius_He*12.0, Radius of Helium Shroud [in]
C    RadHT, 273, Radius for Heating Surface to position i...
C    Radius_He, 22.625, Radius of Helium shrouds [ft]
C    RadMylar, RadNi+AddMylar, Radius of mylar sheets
```

```
C    RadNi, 330
C    RADSET, 80/pi, Radius to setup piping
C    RadWall, 390, Radius of wall [in] (65 ft diam)
C    spacing, ArcL_in/Npipes, Spacing of tubes [in]
C    StartHt, -AddHT, Starting height for Nitrogen Panel, Myla...
C    TAVER, 0.0, Average Temperature over the GHe Panel f...
C    TAVMN, 0.0, The minimum average temperature up to th...
C    TAVMX, 0.0, The maximum average temperature up to th...
C    Thick, .125, Thickness of shroud - Inches
C    ThkTube, .1875
C    TZMAX, 0.0, Maximum time step over the GHe Panel at ...
C    TZMIN, 0.0, Minimum Temperature over the GHe Panel a...
C    Vol, AREA*LTube
C    VolHDR, AREAHDR*spacing/2.0, Volume of Header tubes
C    Walltemp, 80
C    XHEAT, RadHT*cos(AngHt*pi/180), Location of X variable for hotspot on GH...
C    YHEAT, RadHT*sin(AngHt*pi/180), Y coordinate for hotspot location
C    ZHEAT, LengthHT, Z coordinate that marks the Heating surf...
C
C   view factor format:
C   cond_id    node_1     node_2     Area*Fij   $  Fij   Fji
C
C MAIN.1 to Space disabled, Bij = 0.35500
C MAIN.2 to Space disabled, Bij = 0.35520
C MAIN.3 to Space disabled, Bij = 0.35860
C MAIN.4 to Space disabled, Bij = 0.34960
C MAIN.5 to Space disabled, Bij = 0.35380
C MAIN.6 to Space disabled, Bij = 0.33960
C MAIN.7 to Space disabled, Bij = 0.35460
C MAIN.8 to Space disabled, Bij = 0.35140
C MAIN.9 to Space disabled, Bij = 0.34680
C MAIN.10 to Space disabled, Bij = 0.35540
C MAIN.11 to Space disabled, Bij = 0.35480
C MAIN.12 to Space disabled, Bij = 0.24280
C MAIN.13 to Space disabled, Bij = 0.23620
C MAIN.14 to Space disabled, Bij = 0.23160
C MAIN.15 to Space disabled, Bij = 0.22540
C MAIN.16 to Space disabled, Bij = 0.22440
C MAIN.17 to Space disabled, Bij = 0.23300
C MAIN.18 to Space disabled, Bij = 0.22680
C MAIN.19 to Space disabled, Bij = 0.23860
C MAIN.20 to Space disabled, Bij = 0.23000
C MAIN.21 to Space disabled, Bij = 0.24000
C MAIN.22 to Space disabled, Bij = 0.23800
C MAIN.23 to Space disabled, Bij = 0.14260
C MAIN.24 to Space disabled, Bij = 0.13240
C MAIN.25 to Space disabled, Bij = 0.12940
C MAIN.26 to Space disabled, Bij = 0.13120
C NISHROUD.95 to Space disabled, Bij = 0.60760
C NISHROUD.96 to Space disabled, Bij = 0.60020
C NISHROUD.97 to Space disabled, Bij = 0.69620
C NISHROUD.98 to Space disabled, Bij = 0.82020
C NISHROUD.99 to Space disabled, Bij = 0.91120
C NISHROUD.100 to Space disabled, Bij = 0.93760
            1,         MAIN.1,     NISHROUD.1,     0.00097614  $    0.012173,0.00042971
            2,         MAIN.1,     NISHROUD.2,      0.0032378  $    0.040376, 0.0014253
            3,         MAIN.1,     NISHROUD.3,      0.0089540  $     0.11166, 0.0039416
```

```
           4,         MAIN.1,     NISHROUD.4,       0.012005  $     0.14970, 0.0052846
           5,         MAIN.1,     NISHROUD.5,      0.0062183  $    0.077543, 0.0027373
           6,         MAIN.1,     NISHROUD.6,      0.0018606  $    0.023202,0.00081904
           7,         MAIN.1,     NISHROUD.7,     0.00062431  $   0.0077853,0.00027483
           8,         MAIN.1,     NISHROUD.8,     0.00014434  $  0.0018000,6.3541e-005
           9,         MAIN.1,    NISHROUD.11,     0.00057407  $   0.0071588,0.00025271
          10,         MAIN.1,    NISHROUD.12,      0.0015188  $    0.018940,0.00066859
          11,         MAIN.1,    NISHROUD.13,      0.0028819  $    0.035937, 0.0012686
          12,         MAIN.1,    NISHROUD.14,      0.0036195  $    0.045136, 0.0015933
          13,         MAIN.1,    NISHROUD.15,      0.0027296  $    0.034039, 0.0012016
          14,         MAIN.1,    NISHROUD.16,      0.0010112  $    0.012610,0.00044513
          15,         MAIN.1,    NISHROUD.17,     0.00024057  $   0.0030000,0.00010590
          16,         MAIN.1,    NISHROUD.18,     0.00020850  $  0.0026000,9.1782e-005
          17,         MAIN.1,    NISHROUD.21,     0.00036888  $   0.0046000,0.00016238
          18,         MAIN.1,    NISHROUD.22,     0.00047956  $   0.0059802,0.00021111
          19,         MAIN.1,    NISHROUD.23,     0.00070568  $   0.0088000,0.00031065
          20,         MAIN.1,    NISHROUD.24,     0.00076345  $   0.0095204,0.00033608
          21,         MAIN.1,    NISHROUD.25,     0.00065416  $   0.0081575,0.00028797
          22,         MAIN.1,    NISHROUD.26,     0.00022454  $  0.0028000,9.8842e-005
          23,         MAIN.1,    NISHROUD.27,     0.00022454  $  0.0028000,9.8842e-005
          24,         MAIN.1,    NISHROUD.28,     0.00014434  $  0.0018000,6.3541e-005
          25,         MAIN.1,    NISHROUD.32,     0.00016038  $  0.0020000,7.0601e-005
          26,         MAIN.1,    NISHROUD.33,     0.00014434  $  0.0018000,6.3541e-005
          27,         MAIN.1,    NISHROUD.34,     0.00016038  $  0.0020000,7.0601e-005
          28,         MAIN.1,    NISHROUD.35,     0.00017642  $  0.0022000,7.7662e-005
          29,         MAIN.1,    NISHROUD.46,     0.00047037  $   0.0058656,0.00020706
          30,         MAIN.2,     NISHROUD.1,      0.0019482  $    0.012147,0.00085760
          31,         MAIN.2,     NISHROUD.2,      0.0047106  $    0.029371, 0.0020736
          32,         MAIN.2,     NISHROUD.3,       0.013273  $    0.082761, 0.0058430
          33,         MAIN.2,     NISHROUD.4,       0.021955  $     0.13689, 0.0096649
          34,         MAIN.2,     NISHROUD.5,       0.015291  $    0.095343, 0.0067314
          35,         MAIN.2,     NISHROUD.6,      0.0056966  $    0.035519, 0.0025077
          36,         MAIN.2,     NISHROUD.7,      0.0018638  $    0.011621,0.00082044
          37,         MAIN.2,     NISHROUD.8,     0.00064710  $   0.0040348,0.00028486
          38,         MAIN.2,    NISHROUD.11,      0.0010725  $   0.0066874,0.00047214
       11953,     NISHROUD.1,     NISHROUD.3,      0.0028018  $   0.0012334, 0.0012334
       11954,     NISHROUD.1,     NISHROUD.4,      0.0027260  $   0.0012000, 0.0012000
       11955,     NISHROUD.1,     NISHROUD.5,      0.0034226  $   0.0015067, 0.0015067
       11956,     NISHROUD.1,     NISHROUD.6,      0.0025195  $   0.0011091, 0.0011091
       11957,     NISHROUD.1,     NISHROUD.7,      0.0026849  $   0.0011819, 0.0011819
       11958,     NISHROUD.1,     NISHROUD.8,      0.0038622  $   0.0017002, 0.0017002
       11959,     NISHROUD.1,     NISHROUD.9,      0.0026849  $   0.0011819, 0.001181
C      NISHROUD.93      12.899      5000    1.0000   0.16235   0.001             5.6
C      NISHROUD.94      12.899      5000    1.0000   0.30297   0.000             6.5
C      NISHROUD.95      12.899      5000    1.0000   0.39068   0.000             7.1
C      NISHROUD.96      12.899      5000    1.0000   0.39197   0.000             6.9
C      NISHROUD.97      12.899      5000    1.0000   0.30350   0.000             6.5
C      NISHROUD.98      12.899      5000    1.0000   0.16974   0.001             5.8
C      NISHROUD.99      12.899      5000    1.0000  0.082379   0.000             4.6
C     NISHROUD.100      12.899      5000    1.0000  0.042768   0.001             4.1
C
PSTART
C    331 calculation nodes
C    12870 radks were output, 2216 radks were filtered out
C    166 nodes(50%) have bij sum + bij inactive < .9
C    5000 average number of rays per node shot
```

APPENDIX C

InputFMHT.dat

This file shows the input used for the MatLab program FMHTPRE.m. The prompts by the program are in black while the inputs are in bold red.

```
>> FMHTPRE
***********************************************************
Enter the base value for the array numbers.
A description of the array numbers is the following:
%      .....Pntr to numeric value of viscosity array  REAL Array
%   710.....Conductor Number Array                    INTEGER ARRAY
%   720.....Node# 1 array  (first Node# on conductor) INTEGER ARRAY
%   730.....Node# 2 array (second Node# on conductor) INTEGER ARRAY
%   740.....Area array of Node 1                      REAL ARRAY
%   750.....Area array of Node 2                      REAL ARRAY
%   760.....View Factor array                         REAL ARRAY
%   770.....Effective Length Array                    REAL ARRAY
%   780.....Nodal index submodel array                INTEGER ARRAY
%   790.....Nodal index submodel array                INTEGER ARRAY
%   800.....Index to string array of submodels        STRING ARRAY

 Notice that the "base value" used for these arrays was 700
 The increment value was 10

Enter the "base value" for the arrays:
200
Enter the "increment value" for the arrays:
10
***********************************************************
Enter the submodel name that the arrays above will be in.
Enter the submodel name:
MAIN
***********************************************************
Enter the submodel name that contains the viscosity array.
If you don't know this just enter something now and you
can change it later in the output manually.

Enter the viscosity submodel name:
VISC
Enter the number of the viscosity array:
555
***********************************************************
Enter the directory and filename to get the data:
Example: "C:\Mydirect\happyface\clownface\base\AREAFIJ.DAT"
 AREAFIJTEST.DAT
***********************************************************
Enter the directory and filename to write out the preprocessed data.
Example: "fmht.dat"  or "C:\Squid\swim\prefmht.dat"
 OutputFMHT.out
***********************************************************
```

```
Enter the directory and filename to get Area information.
Example: "AREAFIJ.ar" or "C:\EM_WORK\James_Webb\ERICS_WORK\AREAFIJ.ar"
Enter director and filename:
AREAFIJ.ar

Do you want to process the header data?
 Enter "y" or "yes" if you do.
no
************************************************************
Enter the cutoff value for the conductors
You enter a small value to eliminate some of the conductors
that are not really needed. Engineering judgement is needed
for this. Enter 0 if you don"t want to eliminate conductors
Enter the cutoff value for the conductors
0.00018
************************************************************
Next you will be deciding which pairs of submodel names to
eliminate conductors between. For example, the free molecular
conduction between like surfaces that are nearly the same
temperature is small. You can eliminate those conductors,
saving solve time, without significant impact on the model
Do you want to eliminate conductors based upon the names of
the node submodels?
Do you want to eliminate conductors by node submodel name?
 Enter "y" for yes
y
Each group of components in your thermal model have submodels associated with them.
The cases where you may want to eliminate FMHTing conductors are the following:
  1) You may want to eliminate conductors to the same submodel. The reason for
     this is that the temperature may be so close together that the FMHT is
     insignificant. Why have the added conductors if they are not important?
  2) The two different submodels may have temperatures that are expected to
     be the same. If they are close in temperature eliminate the FMHT conductors
  3) The two different submodels may not be visible to each other. If the
     visibility is poor the conduction by free molecular heat transfer will be small.
  4) After having run your model, you may observe that the heat transfer is small
     (or relatively small) between the two submodels. To prevent large solve times
     with a large model you can eliminate these conductors.

Now you need to enter the submodel names which you want to eliminate FMHT conductors
from. First I will ask you the first submodel name. Next, I will ask you the other.
submodel name.

Enter pairs of submodel names to eliminate conductors
Enter the first submodel name
MAIN
Enter the second submodel name
MAIN
Continue? <y> n
y

Enter pairs of submodel names to eliminate conductors
Enter the first submodel name
NISHROUD
Enter the second submodel name
NISHROUD
Continue? <y> n
```

```
n
You may want to keep some FMHT conductors in the model even if you eliminated
the conductors by using the previous option (using submodel names). You can
now specify the conductor numbers that you want to keep in the model no matter
the name of the submodel associated with the nodes. You may know that there are
a few conductors that are important which you wish to not eliminate. This option
allows you to keep those conductors in the model.
Do you want to keep some FMHT conductors in the model? <y>
y
Enter the FMHT conductor to keep in the model
100
Continue? <y> n
y
Enter the FMHT conductor to keep in the model
101
Continue? <y> n
y
Enter the FMHT conductor to keep in the model
102
Continue? <y> n
y
Enter the FMHT conductor to keep in the model
103
Continue? <y> n
n
***********************************
EFFECTIVE LENGTHS BETWEEN SUBMODELS
***********************************
The effective length is used to find the Regime of fluid heat transfer
This module is used so the user is able to setup the array of effective
lengths -- APPROXIMATE VALUES based on the submodel names
It is recommended that the user modify this and put in the actual
effective lengths -- But this is useful to get the model running.
Also, if you already know the regime that you are in then this is not
really needed. Just make sure that the logic calls the correct solve
method to calculate the FMHT conductors. Use feet as the units
Do you want to define the effective length between submodels? <y> n
y

First you will enter the pair of submodel names.
Next you will enter the effective length associated with the two submodels
Enter the first submodel Name (for effective length)
NISHROUD
Enter the second submodel Name (for effective length)
MAIN
Enter the effective length for the two submodels [ft]
55
Do you want to add more effective lengths? <y> n
n
Enter the default effective length if the submodels are not found
1.0

============================================================
```

APPENDIX D

FMHTPRE.m

The MatLab program that processes the free molecular conductors and sets up the arrays used in the logic of SINDA/FLUINT for the automated updating of the conductors.

```
function FMHTPRE(action)
% ****************************************************
% FMHTPRE.m program that processes data and sets up the
%  Free Molecular Heat Transfer (FMHT) logic that allows
%  for transient operation.
% *****************************************************
% FILE NAME: FMHTPRE
%  (Free Molecular Heat Transfer Preprocessor)
% PROGRAMMER: Eric Malroy (Eric.T.Malroy@nasa.gov)
% DATE: 3/03/07
% LAST MODIFICATION DATE: 8/22/07
% VERSION: 1.1
% *****************************************************
% DESCRIPTION
%  Specifically, this module sets up the the following arrays
%  that allow for transient free molecular heat transfer:
%   viscnum.....Pntr to numeric value of viscosity array  REAL Array
%   710.....Conductor Number Array                    INTEGER ARRAY
%   720.....Node# 1 array  (first Node# on conductor) INTEGER ARRAY
%   730.....Node# 2 array (second Node# on conductor) INTEGER ARRAY
%   740.....Area array of Node 1                      REAL ARRAY
%   750.....Area array of Node 2                      REAL ARRAY
%   760.....View Factor array                         REAL ARRAY
%   770.....Effective Length Array                    REAL ARRAY
%   780.....Nodal index submodel array                INTEGER ARRAY
%   790.....Nodal index submodel array                INTEGER ARRAY
%   800.....Index to UCA array CRYTRAN                STRING ARRAY
%******************************************************
% First Get Required Inputs
% ************************
% *** find base number and increment value to write out array numbers
JWSTpath2add
flg9 = 1;
if(flg9 == 1)
   disp('*********************************************************');
   disp('Enter the base value for the array numbers.');
   disp('A description of the array numbers is the following:');
   disp('%      .....Pntr to numeric value of viscosity array  REAL Array');
   disp('%   710.....Conductor Number Array                    INTEGER ARRAY');
   disp('%   720.....Node# 1 array  (first Node# on conductor) INTEGER ARRAY');
   disp('%   730.....Node# 2 array (second Node# on conductor) INTEGER ARRAY');
   disp('%   740.....Area array of Node 1                      REAL ARRAY');
   disp('%   750.....Area array of Node 2                      REAL ARRAY');
   disp('%   760.....View Factor array                         REAL ARRAY');
   disp('%   770.....Effective Length Array                    REAL ARRAY');
   disp('%   780.....Nodal index submodel array                INTEGER ARRAY');
   disp('%   790.....Nodal index submodel array                INTEGER ARRAY');
   disp('%   800.....Index to string array of submodels        STRING ARRAY');
   disp('                                       ');
   disp(' Notice that the "base value" used for these arrays was 700');
   disp(' The increment value was 10');
   disp('                                       ');
   baseN = input('Enter the "base value" for the arrays: \n');
   IncrN = input('Enter the "increment value" for the arrays: \n');
   disp('**********************************************************');
   disp('Enter the submodel name that the arrays above will be in.');
   rysub = input('Enter the submodel name:\n','s');
   disp('**********************************************************');
   disp('Enter the submodel name that contains the viscosity array.');
   disp('If you don''t know this just enter something now and you');
   disp('can change it later in the output manually.');
   viscsub = input('\nEnter the viscosity submodel name:\n','s');
   viscnum = input('Enter the number of the viscosity array:\n');
   disp('**********************************************************');
   disp('Enter the directory and filename to get the data:');
   disp('Example: "C:\Mydirect\happyface\clownface\base\AREAFIJ.DAT"');
   rfnam = input(' ','s');
   disp('**********************************************************');
   disp('Enter the directory and filename to write out the preprocessed data.');
   disp('Example: "fmht.dat"  or "C:\Squid\swim\prefmht.dat" ');
   wfnam = input(' ','s');
   disp('**********************************************************');
   disp('Enter the directory and filename to get Area information.');
   disp('Example: "AREAFIJ.ar" or "C:\EM_WORK\James_Webb\ERICS_WORK\AREAFIJ.ar"');
   afnam = input('Enter director and filename:\n','s');
   afpntr = fopen(afnam,'r');
   iter = 0;
   while (1)
      iter = iter+1;
      aline1 = fgetl(afpntr);
      jjj = rem(500,iter);
      if (jjj == 0)
         iter;
         aline1;
      end
      sz_ln = size(aline1);
      if ~ischar(aline1), break, end
      if (sz_ln(2)>10)
         [areasub{iter},nodena(iter,1),nodena(iter,2)] = areaget(aline1);
      end
   end
   fclose(afpntr);
   rfpntr = fopen(rfnam,'r');
   wfpntr = fopen(wfnam,'a');
   phdflg = input('Do you want to process the header data?\n Enter "y" or "yes" if you
do.\n','s');
   if(strcmp(phdflg,'y')|strcmp(phdflg,'yes'))
      flgs1 = 1;
      disp('Do you want to search for the free molecular multiplier?');
      sfmm = input('Enter "y" or "yes" if you do:\n','s');
      if(strcmp(sfmm,'y')|strcmp(sfmm,'yes'))
         flgs2 = 1;
      else
         flgs2 = 0;
      end
   else
      flgs1 = 0;
      flgs2 = 0;
   end

   lnflg = 1;
   while(lnflg)
      tline1 = fgetl(rfpntr);
      [newln,oflag] = strlibr(1,'add cr/lf',tline1,120,32);
      qtest1 = findstr(newln,'HEADER CONDUCTOR DATA,');
      qtest2 = findstr(newln,'PSTOP');
      qtest3 = findstr(newln,'Area*Fij');
      qtest4 = findstr(newln,'Free Molecular Multiplier');
      if (isempty(qtest1)&isempty(qtest2))
         if(exist('prline','var'))
            prline = [prline,newln];
         else
            prline = newln;
         end
      end
      if (~isempty(qtest3))
         lnflg = 0;
      end
      if (~isempty(qtest4))
         fmmnum = str2num(newln((qtest4(1)+26):(qtest4(1)+35)));
      end
   end
   if (flgs1)
      fprintf(wfpntr,'%s',prline);
   end
   if (flgs2 == 1)
      disp('Enter the Free Molecular Multiplier from the AREAFIJ.DAT');
      fmmnum = input('Enter the real value for the multiplier:  ');
   end
   disp('*************************************************************');
   disp('Enter the cutoff value for the conductors.');
   disp('You enter a small value to eliminate some of the conductors');
   disp('that are not really needed. Engineering judgment is needed');
   disp('here. Enter 0 if you don''t want to eliminate conductors.');
   cutoff = input('Enter the cutoff value for the conductors:\n');
   disp('*************************************************************');
   disp('Next you will be deciding which pairs of submodel names to ');
   disp('eliminate conductors between. For example, the free molecular');
   disp('conduction between like surfaces that are nearly the same ');
   disp('temperature is small. You can eliminate these conductors,');
   disp('saving solve time, without significant impact on the model.');
   disp('Do you want to eliminate conductors based upon the names of');
   disp('the node submodels?');
   strelim = input('Do you want to eliminate conductors by node submodel name?\n Enter "y" for
yes\n','s');
   if(isempty(strelim)|strcmp(strelim,'y')|strcmp(strelim,'yes'))
      flgs3 = 1;
      disp('Each group of components in your thermal model have submodels associated with
them.');
      disp('The cases where you may want to eliminate FMHTing conductors are the following:');
      disp('  1) You may want to eliminate conductors attached to the same submodel. The
reason');
      disp('     for this is that the temperature may be so close together that the FMHT is ');
      disp('     insignificant. Why have the added conductors if they are not important?');
      disp('  2) The two different submodels may have temperatures that are expected to');
      disp('     be the same. If they are close in temperature eliminate the FMHT conductors.');
      disp('  3) The two different submodels may not be visible to each other. If the');
      disp('     visibility is poor the conduction by free molecular heat transfer will be
small.');
      disp('  4) After running your model, you may observe that the heat transfer is small');
      disp('     (or relatively small) between the two submodels. To prevent large solve times');
      disp('     with a large model you can eliminate these conductors.');
      disp('  5) The maximum number of conductors allowed in a free molecular group is 10,000.');
      disp('     This is due to the limitation imposed by Sinda Fluint on the number of array');
      disp('     elements. If your conductors are over this number, then you may have to
eliminate');
      disp('     some conductors.');
      disp(' ');
      pause(5);
      disp('Enter the submodel names used to eliminate FMHT conductors');
      disp('First, I will ask you the first submodel name. Next, I will ask you the other');
      disp('submodel name. ');
      flg7 = 1;
      itersub = 0;
      while (flg7)
         disp(' ')
         itersub = itersub + 1;
         disp('Enter pairs of submodel names to eliminate conductors.');
         strsub{itersub,1} = input('Enter the first submodel name:\n','s');
         strsub{itersub,2} = input('Enter the second submodel name:\n','s');
         ttt = input('Continue? <y> n\n','s');
         if (isempty(ttt)|strcmp(ttt,'y')|strcmp(ttt,'yes'))
            flg7 = 1;
         else
            flg7 = 0;
            subcnt = itersub;
         end
      end
   else
      subcnt = 0;
   end
   %********************************************************************************
   % Keep some FMHT Conductors in the Model
   % ***************************************
   disp('You may want to keep some FMHT conductors in the model even if you eliminated');
   disp('the conductors by using the previous option -- using submodel names. You can');
   disp('now specify the conductor numbers that you want to keep in the model no matter');
   disp('the name of the submodel associated with the nodes. You may know that there are')
   disp('a few conductors that are inportant which you wish to not eliminate. This option');
   disp('allows you to keep those conductors in the model.')
   fkeep = input('Do you want to keep some FMHT conductors in the model? <y> \n','s');
   if(isempty(fkeep)|strcmp(fkeep,'y')|strcmp(fkeep,'yes'))
      kflag = 1;
      i = 0;
      while(kflag)
         i = i + 1;
         condn(i) = input('Enter the FMHT conductor to keep in the model:\n');
         sflg1 = input('Continue? <y> n\n','s');
         if(isempty(sflg1)|strcmp(sflg1,'y')|strcmp(sflg1,'Y')|strcmp(sflg1,'yes'))
            kflag = 1;
         else
            kflag = 0;
            keepcnt = i;
         end
         clear sflg1;
      end
   else
      keepcnt = 0;
   end
   %****************************************************************************
   % Enter Effective Lengths between Submodels
   disp('***********************************');
   disp('EFFECTIVE LENGTHS BETWEEN SUBMODELS');
   disp('***********************************');
   disp('The effective length is used to find the Regime of the free molecular heat transfer');
   disp('This module is used so the user is able to setup the array of effective');
   disp('lengths -- APPROXIMATE VALUES based on the submodel names.');
   disp('It is recommended that the user modify this and put in the actual');
   disp('effective lengths -- But this is useful to get the model running.');
   disp('Also, if you already know the regime of heat tranfer then this is not');
   disp('really needed. Just make sure that the logic calls the correct solve ');
   disp('method to calculate the FMHT conductors. Use feet as the units.');
   zflg = input('Do you want to define the effective length between submodels? <y> n\n','s')
   if(isempty(zflg)|strcmp(zflg,'y')|strcmp(zflg,'yes'))
      disp('First you will enter the pair of submodel names.');
      disp('Next you will enter the effective length associated with the two submodels.');
      itr = 0;
      efflag = 1;
      jflg = 1;
      while (jflg)
         itr = itr + 1;
         effcell{itr,1} = input('Enter the first submodel Name (for effective length):\n','s');
         effcell{itr,2} = input('Enter the second submodel Name (for effective length):\n','s');
         effry(itr) = input('Enter the effective length for the two submodels [ft]:\n');
         zflg = input('Do you want to add more effective lengths? <y> n\n','s');
         if(isempty(zflg)|strcmp(zflg,'y')|strcmp(zflg,'yes')|strcmp(zflg,'Y'))
            jflg = 1;
         else
            jflg = 0;
         end
      end
      effdef = input('Enter the default effective length if the submodels are not found:\n');
   end
   % ********************************
   % Start processing conductor lines
   % ********************************
   lnflg = 1;
   cntr = 0;
   icnt =0;
   indxk = 0;
   while(lnflg)
      icnt = icnt + 1;
      tline1 = fgets(rfpntr);
      sz_tline1 = size(tline1)
      if ~ischar(tline1), break, end
      qtest1 = findstr(tline1,'C ')
      jjj = rem(500,icnt);
      if (jjj == 0)
         tline1;
         qtest1;
         icnt;
      end
      if ((isempty(qtest1))&(sz_tline1(2)>20))
         [cnd,n1sub,nod1,n2sub,nod2,cval,vwfr] = cdatget(tline1);
         % ********************************************************
         % First go through logic to decide if conductor will be used
         kflag = 0;  % Means the conductor will not be used
         if keepcnt > 0
            for(j=1:keepcnt)
               if (cnd == condn(j))
                  kflag = 1;
                  break;
               end
            end
         end
         if (cval > cutoff)
            % next check to see that submodels are not excluded
            if subcnt == 0
               kflag = 1;
            else
               tflag = 0;
               for i=1:subcnt
                  flgz1 = strcmp(n1sub,strsub{i,1});
                  flgz2 = strcmp(n2sub,strsub{i,2});
                  flgz3 = strcmp(n1sub,strsub{i,2});
                  flgz4 = strcmp(n2sub,strsub{i,1});
                  if (((flgz1 ==1)&(flgz2 == 1))|((flgz3 == 1)&(flgz4 == 1)))
                     tflag = 1;
                     break;
                  end
               end
               if (tflag == 0)
                  kflag = 1;
               end
            end
         end
         % ************************************************************
         if (kflag == 1)   % Conductor will be used
            % **************************************************
            % The next two logic structures are used to fill the KRAY strings
            % if the name of the submodel has not been used before.
            % Check to see if node 1 submodel name is in the list of nodes to keep
            cntr = cntr + 1;
            if (indxk > 0)
               for itx=1:indxk
                  flg98 = strcmp(n1sub,kray{itx});
                  if (flg98 == 1)
                     aryindx1(cntr) = itx;
                     break;
                  end
               end
               if (flg98 == 0)
                  indxk = indxk + 1;
                  kray{indxk}=n1sub;
                  aryindx1(cntr) = indxk;
               end
            else
               kray{1} = n1sub;
               indxk = 1;
               aryindx1(cntr) = indxk;
            end
            % ***************************************************
            % Check to see if node 2 submodel name is in the list of nodes to keep
            for itx=1:indxk
               flg99 = strcmp(n2sub,kray{itx});
               if (flg99 == 1)
                  aryindx2(cntr) = itx;
                  break;
               end
            end
            if (flg99 == 0)
               indxk = indxk + 1;
               kray{indxk}=n2sub;
               aryindx2(cntr) = indxk;
            end
            % *****************************************************
            % Enter other Array data: conductor values, node #'s, view factor
            arycond(cntr) = cnd;
            aryn1(cntr) = nod1;
            aryn2(cntr) = nod2;
            aryvf(cntr) = vwfr;
            % *****************************************************
            % find the area for node #1
```

```
sz_nodena = size(nodena);
itrx = 0;
flg = 1;
while(flg)
   itrx = itrx + 1;
   if(nodena(itrx,1)==nod1)
      flg77 = strcmp(areasub{itrx},n1sub);
      if (flg77 == 1)
         flg = 0;
         arya1(cntr) = nodena(itrx,2);
      end
      if (itrx==sz_nodena(1))
         flg = 0;
      end
   end
end
if (flg77 == 0)
   strhd = ['Error: area not found in list - ',n1sub,'.',num2str(nod1)];
   disp(strhd);
   arya1(cntr) = -1.0;
end
% ***********************************************************
% find the area for node #2
itrx = 0;
flg = 1;
flg77 = 0;
while(flg)
   itrx = itrx + 1;
   if(nodena(itrx,1)==nod2)
      flg77 = strcmp(areasub{itrx},n2sub);
      if (flg77 == 1)
         flg = 0;
         arya2(cntr) = nodena(itrx,2);
      end
      if (itrx==sz_nodena(1))
         flg = 0;
      end
   end
end
if (flg77 == 0)
   strhd = ['Error: area not found in list - ',n2sub,'.',num2str(nod2)];
   disp(strhd);
   arya2(cntr) = -1.0;
end
% *************************************************************
% EFFECTIVE LENGTHS
% ****************
% Decide which effective length to use
if (efflag == 1)
   sz_effry = size(effry);
   lflag = 0;
   for i=1:sz_effry(2)
      tflg1 = strcmp(effcell{i,1},n1sub);
      tflg2 = strcmp(effcell{i,2},n2sub);
      tflg3 = strcmp(effcell{i,2},n1sub);
      tflg4 = strcmp(effcell{i,1},n2sub);
      if (((tflg1 ==1)&(tflg2 == 1))|((tflg3 == 1)&(tflg4 == 1)))
         aryeff(cntr) = effry(i);
         lflag = 1;
         break;
      end
   end
   if (lflag == 0)
```

```
                aryeff(cntr) = effdef;
             end
          end
          %
          % ************************************************************
          if(exist('prline2','var'))
             prline2 = [prline2,tline1];
          else
             prline2 = tline1;
          end
       else
          if(exist('prline2','var'))
             prline2 = [prline2,'C ',tline1];
          else
             prline2 = ['C ',tline1];
          end
       end
    end
    %if statement to find if line has "C "  qtest1 = findstr(tline,'C ');
 end
 % End of while statement for processing lines
 % **********************************************************************
 % write out conductors
 if (exist('prline2','var')&(ischar(prline2)))
    fprintf(wfpntr,'%s',prline2);
    fprintf(wfpntr,'%s\n\n','C ');
 end
 % **********************************************************************
 % Write out arrays for Thermal Desktop logic
 fprintf(wfpntr,'%s\n','C ****************************************************************');
 fprintf(wfpntr,'%s\n','C **** ARRAYS FOR THERMAL DESKTOP LOGIC ***');
 cel{1} = ['       ',int2str((baseN+1*IncrN)),'= '];
 cel{2} = '       ';
 cel{3} = ' ';
 cel{4} = ' ';
 cel{5} = ', ';
 cel{6} = '%d';
 nrw = 8;
 flgg = 1;
% cel
% arycond
% nrw
% flgg
 fprintf(wfpntr,'%s\n\n',ray2str(cel,arycond,nrw,flgg));
 cel{1} = ['       ',int2str((baseN+2*IncrN)),'= '];
 fprintf(wfpntr,'%s\n\n',ray2str(cel,aryn1,nrw,flgg));
 cel{1} = ['       ',int2str((baseN+3*IncrN)),'= '];
 fprintf(wfpntr,'%s\n\n',ray2str(cel,aryn2,nrw,flgg));
 cel{1} = ['       ',int2str((baseN+4*IncrN)),'= '];
 cel{6} = '%7.3f';
 fprintf(wfpntr,'%s\n\n',ray2str(cel,arya1,nrw,flgg));
 cel{1} = ['       ',int2str((baseN+5*IncrN)),'= '];
 cel{6} = '%7.3f';
 fprintf(wfpntr,'%s\n\n',ray2str(cel,arya2,nrw,flgg));
 cel{1} = ['       ',int2str((baseN+6*IncrN)),'= '];
 cel{6} = '%8.3f';
 fprintf(wfpntr,'%s\n\n',ray2str(cel,aryvf,nrw,flgg));
 cel{1} = ['       ',int2str((baseN+7*IncrN)),'= '];
 cel{6} = '%8.3f';
 fprintf(wfpntr,'%s\n\n',ray2str(cel,aryeff,nrw,flgg));
 cel{1} = ['       ',int2str((baseN+8*IncrN)),'= '];
 cel{6} = '%d';
 fprintf(wfpntr,'%s\n\n',ray2str(cel,aryindx1,nrw,flgg));
```

```
   cel{1} = ['       ',int2str((baseN+9*IncrN)),'= '];
   cel{6} = '%d';
   fprintf(wfpntr,'%s\n\n',ray2str(cel,aryindx2,nrw,flgg));
   kray
   kstr = cell2str(kray,':');
   fprintf(wfpntr,'%s',['       ',int2str((baseN+10*IncrN)),' =',kstr,char(10),char(13)]);
   fprintf(wfpntr,'%s',['       333=0,0,0,0,0,0,0,0,0,0,0',char(10),char(13)]);
   fprintf(wfpntr,'%s',['C
************************************************************',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL ARYTRN(''',rysub,''',333,IPNTR)',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL
ARYTRN(''',viscsub,''',',int2str(viscnum),',NA(IPNTR+1))',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL
ARYTRN(''',rysub,''',',int2str((baseN+IncrN)),',NA(IPNTR+2))',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL
ARYTRN(''',rysub,''',',int2str((baseN+2*IncrN)),',NA(IPNTR+3))',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL
ARYTRN(''',rysub,''',',int2str((baseN+3*IncrN)),',NA(IPNTR+4))',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL
ARYTRN(''',rysub,''',',int2str((baseN+4*IncrN)),',NA(IPNTR+5))',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL
ARYTRN(''',rysub,''',',int2str((baseN+5*IncrN)),',NA(IPNTR+6))',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL
ARYTRN(''',rysub,''',',int2str((baseN+6*IncrN)),',NA(IPNTR+7))',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL
ARYTRN(''',rysub,''',',int2str((baseN+7*IncrN)),',NA(IPNTR+8))',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL
ARYTRN(''',rysub,''',',int2str((baseN+8*IncrN)),',NA(IPNTR+9))',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL
ARYTRN(''',rysub,''',',int2str((baseN+9*IncrN)),',NA(IPNTR+10))',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL
CRYTRN(''',rysub,''',',int2str((baseN+10*IncrN)),',NA(IPNTR+11))',char(10),char(13)]);
   fprintf(wfpntr,'%s',['C      1e-6 torr = 1.9337E-8psi (Note: you need to input PRESS with the
correct pressure)',char(10),char(13)]);
   fprintf(wfpntr,'%s',['F      CALL
FMHTCOND(''',rysub,''',IPNTR,',int2str((baseN+10*IncrN)),',4.0032,PRESS,1.667)',char(10),char(13)
]);
   status = fclose('all');
end
%==========================================================
function [stro] = ray2str(celray,nvect,nrow,flg)
% RAY2STR.m Constructs a Thermal Desktop array out of the
% vector
% *****************************************************
% INPUT ARGUMENTS:
% ***************
% celray{1} .....first string (left margin)
% celray{2} .....middle/end string (left margin)
% celray{3} .....end row string (right margin)
% celray{4} .....end string end line (right margin, last line)
% celray{5} .....dividing string (between each number)
% celray{6} .....string print format (example '%d')
% nvect .........vector array to print as string
%                 (example: nvect = [1.4,1.5,3.6])
% nrow ..........the number of numbers per row to print to string (max 10)
% flg ...........use this to specify the type of conversion
%                It is really for future use. (set to 1)
% ***************
% OUTPUT ARGUMENTS:
% stro ..........output string
% ******************************************************
if (flg == 1)
   sz_nvect = size(nvect);
```

```
loopcnt = floor(sz_nvect(2)/nrow);
lstrow = rem(sz_nvect(2),nrow);
for (i=1:loopcnt)
   idx0 = (i-1)*nrow + 1;
   idx1 = i*nrow - 1;
   idx2 = i*nrow
   if (i == 1)
      lineb = [celray{1},sprintf([celray{6},celray{5}],nvect(1:idx1)), ...
      sprintf(celray{6},nvect(idx2)),celray{3},char(10),char(13)];
   elseif(i~=loopcnt)
      lineb = [lineb,celray{2},sprintf([celray{6},celray{5}], ...
        nvect(idx0:idx1)),sprintf(celray{6},nvect(idx2)), ...
        celray{3},char(10),char(13)];
   else
      switch lstrow
         case 0
            lineb = [lineb,celray{2},sprintf([celray{6},celray{5}], ...
            nvect(idx0:idx1)),sprintf(celray{6},nvect(idx2)), ...
            celray{4},char(10),char(13)];
         case 1
            lineb = [lineb,celray{2},sprintf([celray{6},celray{5}], ...
            nvect(idx0:idx1)),sprintf(celray{6},nvect(idx2)), ...
            celray{3},char(10),char(13),celray{2},sprintf(celray{6}, ...
            nvect(idx2+1)),celray{4},char(10),char(13)];
         case 2
            lineb = [lineb,celray{2},sprintf([celray{6},celray{5}], ...
            nvect(idx0:idx1)),sprintf(celray{6},nvect(idx2)), ...
            celray{3},char(10),char(13),celray{2},sprintf(celray{6}, ...
            nvect(idx2+1)),celray{5},sprintf(celray{6}, ...
            nvect((idx2+2))),celray{4},char(10),char(13)];
         case 3
            lineb = [lineb,celray{2},sprintf([celray{6},celray{5}], ...
            nvect(idx0:idx1)),sprintf(celray{6},nvect(idx2)), ...
            celray{3},char(10),char(13), ...
            celray{2},sprintf([celray{6},celray{5}],nvect(idx2:idx2+2)), ...
            sprintf([celray{6},celray{4}], ...
            nvect((idx2+3))),char(10),char(13)];
         case 4
            lineb = [lineb,celray{2},sprintf([celray{6},celray{5}], ...
            nvect(idx0:idx1)),sprintf(celray{6},nvect(idx2)), ...
            celray{3},char(10),char(13), ...
            celray{2},sprintf([celray{6},celray{5}],nvect(idx2:idx2+3)), ...
            sprintf([celray{6},celray{4}], ...
            nvect((idx2+4))),char(10),char(13)];
         case 5
            lineb = [lineb,celray{2},sprintf([celray{6},celray{5}], ...
            nvect(idx0:idx1)),sprintf(celray{6},nvect(idx2)), ...
            celray{3},char(10),char(13), ...
            celray{2},sprintf([celray{6},celray{5}],nvect(idx2:idx2+4)), ...
            sprintf([celray{6},celray{4}], ...
            nvect((idx2+5))),char(10),char(13)];
         case 6
            lineb = [lineb,celray{2},sprintf([celray{6},celray{5}], ...
            nvect(idx0:idx1)),sprintf(celray{6},nvect(idx2)), ...
            celray{3},char(10),char(13), ...
            celray{2},sprintf([celray{6},celray{5}],nvect(idx2:idx2+5)), ...
            sprintf([celray{6},celray{4}], ...
            nvect((idx2+6))),char(10),char(13)];
         case 7
            lineb = [lineb,celray{2},sprintf([celray{6},celray{5}], ...
            nvect(idx0:idx1)),sprintf(celray{6},nvect(idx2)), ...
            celray{3},char(10),char(13), ...
            celray{2},sprintf([celray{6},celray{5}],nvect(idx2:idx2+6)), ...
```

```
               sprintf([celray{6},celray{4}], ...
               nvect((idx2+7))),char(10),char(13)];
            case 8
               lineb = [lineb,celray{2},sprintf([celray{6},celray{5}], ...
               nvect(idx0:idx1)),sprintf(celray{6},nvect(idx2)), ...
               celray{3},char(10),char(13), ...
               celray{2},sprintf([celray{6},celray{5}],nvect(idx2:idx2+7)), ...
               sprintf([celray{6},celray{4}], ...
               nvect((idx2+8))),char(10),char(13)];
            case 9
               lineb = [lineb,celray{2},sprintf([celray{6},celray{5}], ...
               nvect(idx0:idx1)),sprintf(celray{6},nvect(idx2)), ...
               celray{3},char(10),char(13), ...
               celray{2},sprintf([celray{6},celray{5}],nvect(idx2:idx2+8)), ...
               sprintf([celray{6},celray{4}], ...
               nvect((idx2+9))),char(10),char(13)];
            otherwise
               disp('Error in module ray2str.m. Wrong case input!');
         end
      end
   end
end  % Add here for more cases

stro = lineb;

%==========================================================
function [cnd,n1sub,nod1,n2sub,nod2,cval,vwf] = cdatget(strg)
%CDATGET.m finds the relevant conductor data from the string
% *******************************************************
% INPUT ARGUMENTS:
% ****************
% strg ........character string
%  Some examples are the following lines:
%     1,    HEPAN1.1000,       NIPAN1.1,       0.028711  $     0.19040,5.1941e-006
%     2,    HEPAN1.1001,       NIPAN1.1,       0.057754  $     0.19150,1.0448e-005
%     3,    HEPAN1.1002,       NIPAN1.1,       0.057331  $     0.19010,1.0372e-005
%     4,    HEPAN1.1003,       NIPAN1.1,       0.057120  $     0.18940,1.0334e-005
% *********************************************************
% OUTPUT ARGUMENTS:
% *****************
% cnd .........conductor number integer
% n1sub .......submodel name for node 1 (char string)
% nod1 ........node 1 number (int)
% n2sub .......submodel name for node 2 (char string)
% nod2 ........node 2 number (int)
% vwf .........view factor Fij (real)
% **********************************************************
JWSTpath2add
ry = findstr(strg,',');

cnd = str2num(strg(1:(ry(1)-1)));
str1 = strg(1:(ry(1)-1));
str2 = strg((ry(1)+1):(ry(2)-1));
str3 = strg((ry(2)+1):(ry(3)-1));
str4 = strg((ry(3)+1):(ry(4)-1));
ry2 = findstr(str2,'.');
[n1sub] = strlibr(4,str2(1:(ry2(1)-1)));
sz_str2 = size(str2);
nod1 = str2num(str2((ry2(1)+1):sz_str2(2)));

ry3 = findstr(str3,'.');
[n2sub] = strlibr(4,str3(1:(ry3(1)-1)));
sz_str3 = size(str3);
```

```
nod2 = str2num(str3((ry3(1)+1):sz_str3(2)));

ry4 = findstr(str4,'$');
sz_str4 = size(str4);
cval = str2num(str4(1:(ry4(1)-1)));
str5 = str4((ry4(1)+1):sz_str4(2));
vwf = str2num(str5);

% ========================================================
% JWSTpath2add.m
% Used for the James Webb Space Telescope with the Free Molecular Heating
% **********************************************************************
addpath C:\FMHT\MatLabFMHT
addpath C:\matlablibr\strings
% addpath  other paths as needed
```

APPENDIX E

OutputFMHT.dat

This file shows the output generated by the MatLab program FMHTPRE.m. The program eliminates conductors by commenting them out by putting a “C” in front of the line. SINDA/FLUINT does not construct these conductors with the comments when the program is activated. The arrays generated below the conductors are used in the logic so the conductors can be updated periodically in a transient run of SINDA/FLUINT. Normally, the number of conductors generated is large so the length of the arrays will be large – the same number as the conductors. The arrays are limited to 10,000 entries, so at most 10,000 active free molecular conductors are allowed for this submodel. If more conductors are needed the user should try to break up the radiation task into several to prevent the conductors from being too large. The file shown here in the appendix is a condensed version of the file.

```
C          335,        MAIN.10,     NISHROUD.6,       0.013995  $    0.087263, 0.0061609
C          336,        MAIN.10,     NISHROUD.7,       0.022670  $     0.14135, 0.0099793
C          337,        MAIN.10,     NISHROUD.8,       0.014225  $    0.088697, 0.0062622
C          338,        MAIN.10,     NISHROUD.9,      0.0045859  $    0.028594, 0.0020188
C          339,        MAIN.10,    NISHROUD.10,      0.0021356  $    0.013316,0.00094011
C          340,        MAIN.10,    NISHROUD.13,     0.00050977  $   0.0031785,0.00022440
         341,        MAIN.10,    NISHROUD.14,      0.0010849  $   0.0067643,0.00047757
         342,        MAIN.10,    NISHROUD.15,      0.0030589  $    0.019073, 0.0013466
       11953,     NISHROUD.1,     NISHROUD.3,      0.0028018  $   0.0012334, 0.0012334
       11954,     NISHROUD.1,     NISHROUD.4,      0.0027260  $   0.0012000, 0.0012000
       11955,     NISHROUD.1,     NISHROUD.5,      0.0034226  $   0.0015067, 0.0015067
       11956,     NISHROUD.1,     NISHROUD.6,      0.0025195  $   0.0011091, 0.0011091
       11957,     NISHROUD.1,     NISHROUD.7,      0.0026849  $   0.0011819, 0.0011819
       11958,     NISHROUD.1,     NISHROUD.8,      0.0038622  $   0.0017002, 0.0017002
       11959,     NISHROUD.1,     NISHROUD.9,      0.0026849  $   0.0011819, 0.001181
C

C *****************************************************************
C **** ARRAYS FOR THERMAL DESKTOP LOGIC ***
      510= 341, 342, 11953, 11954, 11955, 11956, 11957, 11958


      520= 10, 10, 1, 1, 1, 1, 1, 1


      530= 14, 15, 3, 4, 5, 6, 7, 8


      540=   0.911,   0.911,  12.899,  12.899,  12.899,  12.899,  12.899,  12.899


      550=  12.899,  12.899,  12.899,  12.899,  12.899,  12.899,  12.899,  12.899


      560=    0.007,    0.019,    0.001,    0.001,    0.002,    0.001,    0.001,    0.002
```

```
       570=   23.000,   23.000,    1.000,    1.000,    1.000,    1.000,    1.000,    1.000

       580= 1, 1, 2, 2, 2, 2, 2, 2

       590= 2, 2, 2, 2, 2, 2, 2, 2

       600 =MAIN:NISHROUD
       333=0,0,0,0,0,0,0,0,0,0,0
C *************************************************************
F      CALL ARYTRN('MAIN',333,IPNTR)
F      CALL ARYTRN('VISC',555,NA(IPNTR+1))
F      CALL ARYTRN('MAIN',510,NA(IPNTR+2))
F      CALL ARYTRN('MAIN',520,NA(IPNTR+3))
F      CALL ARYTRN('MAIN',530,NA(IPNTR+4))
F      CALL ARYTRN('MAIN',540,NA(IPNTR+5))
F      CALL ARYTRN('MAIN',550,NA(IPNTR+6))
F      CALL ARYTRN('MAIN',560,NA(IPNTR+7))
F      CALL ARYTRN('MAIN',570,NA(IPNTR+8))
F      CALL ARYTRN('MAIN',580,NA(IPNTR+9))
F      CALL ARYTRN('MAIN',590,NA(IPNTR+10))
F      CALL CRYTRN('MAIN',600,NA(IPNTR+11))
C      1e-6 torr = 1.9337E-8psi (Note: you need to input PRESS with the correct pressure)
F      CALL FMHTCOND('MAIN',IPNTR,600,4.0032,PRESS,1.667)
```

APPENDIX F

Fortran Free Molecular Heat Transfer Subroutines

```
C***************************************************************
F       SUBROUTINE KNNUM(THOTSURF,PSUBM,MOLWT,PPGAS,LEFF,KNUDSEN,FLGX)
C**************************************************************
C PROGRAMMER: Eric Malroy
C DATE: 2/25/2007
C LAST MOD: 2/25/2007
C DISCRIPTION: Finds the Knudsen number. This tells the type
C   of heat transfer
C      Continuum    Kn < 0.01           FLGX = 1
C      Mixed        0.01 < Kn < 0.30    FLGX = 2
C      Free Molecular      Kn > 0.3            FLGX = 3
C**************************************************************
C INPUT ARGUMENTS:
C ****************
C    THOTSURF .......Temperature of the hot surface (not gas
C                    temperature) [R or F]
C    PSUBM ..........Pointer to array telling the dynamic
C                    viscosity [R] vs [lbm/ft-hr]
C    MOLWT ..........Molecular weight of gas [lbm/lbmole] He = 4.003
C    PPGAS ..........Pressure absolute [psi]
C    LEFF ...........Effective length [ft] or is it [in]
C OUTPUT ARGUMENTS:
C    KNUDSEN ........Knudsen number
C    FLGX ...........Integer flag telling the regime of heat transfer
C                 FLGX = 1  Continuum
C                 FLGX = 2  Mixed
C                 FLGX = 3  Free Molecular
C*****************
C***************************************************************
       CALL COMMON
       DEBOFF
FSTART
       INTEGER ARNUM,FLGX,PSUBM
FSTOP
F       REAL PPGAS,LAMBDA,FACTR1,FACTR2,MUU,GCONST1,PIIZ,RUNIV,
F    *       LEFF,THOTSURF,MOLWT,KNUDSEN,GCONST2
M       CALL D1DEG1(
F    *  (THOTSURF-ABSZRO),A(PSUBM),MUU)
FSTART
       GCONST1 = 5.9955627E-8
       GCONST2 = 32.174
       PIIZ = 3.14159
       RUNIV = 1545.0
       FACTR1 = (MUU)*(GCONST1/PPGAS)
       FACTR2 = sqrt(PIIZ*RUNIV*GCONST2*(THOTSURF-ABSZRO)/(2.0*MOLWT))
       LAMBDA = FACTR1*FACTR2
       KNUDSEN = LAMBDA/LEFF
       KN = ',F15.5,' MU = ',F15.6)
       IF (KNUDSEN .LT. 0.01) THEN
          FLGX = 1
       ELSEIF (KNUDSEN .LT. 0.30) THEN
          FLGX = 2
       ELSE
          FLGX = 3
       ENDIF
```

```
FSTOP
       RETURN
       DEBON
       END
C**************************************************************
C**************************************************************
F       SUBROUTINE FMHT(TS1,TS2,AREA1,AREA2,GAMMA,MW,PPGAS,FVF,QFMH,QPAREA,CCND)
C*************************************************************
C PROGRAMMER: Eric Malroy
C DATE: 2/26/2007
C LAST MOD: 2/26/2007
C DISCRIPTION: Finds the Free Molecular Heat Transfer between two
C  surfaces at different temperatures
C*************************************************************
C INPUT ARGUMENTS:
C ****************
C    TS1 .......Temperature of surface number one [R or F]
C    TS2 .......Temperature of surface number two [R or F]
C    AREA1 .....Area of surface number one [ft^2]
C    AREA2 .....Area of surface number two [ft^2]
C    MW ........Molecular weight of gas [lbm/lbmole] He = 4.003
C    PPGAS .....Pressure absolute [psi]
C    FVF .......View Factor F12 [none]
C OUTPUT ARGUMENTS:
C    QFMH ......Heat rate of the surfaces [BTU/hr]
C    QPAREA ....BTU/hr-ft^2 (based on surface #1)
C    CCND ......Conductor value [Btu/hr-R]
C*****************
C Important Information: If TS1 < TS2 then the heat rate will
C  be negative.
C**************************************************************
       CALL COMMON
       DEBOFF
FSTART
       INTEGER PNTAR,ARNUM,FLGX
FSTOP
F       REAL TS1,TS2,AREA1,AREA2,GAMMA,MW,PPGAS,QFMH,QPAREA,
F    *       NFACT,THOT,TCOLD,AA1,AA2,ATERM1,ATERM2,HOLD1,
F    *       FACC,GCONST,PIIZ,RUNIV,FACTR1,FACTR2,GVAL,CCND
FSTART
       IF (TS2 .GT. TS1) THEN
         THOT = (TS2-ABSZRO)
       ELSE
         THOT = (TS1-ABSZRO)
       ENDIF

C      **** First find the accommodation coefficient factor (based on helium)
C      **** Note this equation needs to be updated if another gas is used!
C      **** Curve fit equation based on Barron’s Pg 250 Cryogenic Heat Transfer
       ATERM1 = 1.30168*(TS1-ABSZRO)**(-.262249)
       ATERM2 = 1.30168*(TS2-ABSZRO)**(-.262249)
       HOLD1 = ((1 - ATERM1)/ATERM1) + (1/FVF) + (AREA1*(1 - ATERM2)/(AREA2*ATERM2))
       FACC = 1.0/HOLD1
       GCONST = 32.1740603
       PIIZ = 3.14159
       RUNIV = 1545.0
       FACTR1 = (GAMMA+1.0)/(GAMMA-1.0)
       FACTR2 = sqrt(GCONST*RUNIV/(8.0*PIIZ*MW*THOT))
       GVAL = FACTR1*FACTR2
       CCND = GVAL*PPGAS*FACC*FVF*AREA1*0.185051917
       QFMH = CCND*(TS2 - TS1)
       QPAREA = QFMH/AREA1
FSTOP
```

```
       RETURN
       DEBON
       END
C**************************************************************
C**************************************************************
F       SUBROUTINE FMHTCOND(SUBMOD,RYPNTR,MOLWT,PGAS,GAM)
C*************************************************************
C PROGRAMMER: Eric Malroy
C DATE: 2/27/2007
C LAST MOD: 4/23/2007
C DISCRIPTION: Calculates the conductor values for FMHT. Each
C iteration this module is called to update the conductors for
C transient runs where the gas pressure and surface temperatures
C are changing.
C*************************************************************
C INPUT ARGUMENTS:
C ****************
C    SUBMOD ............Submodel that contains the ARRAYS
C    NA(RYPNTR + 1) ....Pntr to numeric value of viscosity array [R] vs [lbm/ft-hr]
C    NA(RYPNTR + 2) ....Pntr to Array of conductors
C    NA(RYPNTR + 3) ....Pntr to Array of Node 1
C    NA(RYPNTR + 4) ....Pntr to Array of Node 2
C    NA(RYPNTR + 5) ....Pntr to Array of areas for Node 1
C    NA(RYPNTR + 6) ....Pntr to Array of areas for Node 2
C    NA(RYPNTR + 7) ....Pntr to Array of view factors
C    NA(RYPNTR + 8) ....Pntr to Array of effective lengths
C    NA(RYPNTR + 9) ....Pntr to Array with index to the string
C                       to use which specifies the submodel for
C                       NODE 1
C    NA(RYPNTR + 10)....Pntr to Array with index to the string
C                       to use which specifies the submodel for
C                       NODE 2
C    NA(RYPNTR + 11)....Pntr to String array which tells the submodel
C    MOLWT .........Molecular weight of gas in chamber
C    PGAS ..........Pressure of gas in chamber
C    GAM ...........GAMMA the ratio of specific heats
C    FLGT ..........Tells which method to use for finding the
C                   submodel
C OUTPUT ARGUMENTS:
C*****************
C    None
C*****************
C Important Information: this module has a conflict????
C**************************************************************
       CALL COMMON
       DEBOFF
FSTART
       CHARACTER SUBMOD*(*),SUBM1*(8),SUBM2*(8)
       REAL THOT,MOLWT,PGAS,KNUD,GAM,FMCND,TTT1,TTT2
       INTEGER STRTCND,INCCND,ENDCND, ICNTS,KCAT,RYPNTR,ICK,JCK,FLGO
       ICNTS = NA(NA(RYPNTR + 2))
       DO KCAT = 1,ICNTS
          CALL GETSTR(NA(RYPNTR+11),NA(NA(RYPNTR + 9)+KCAT),SUBM1)
           CALL GETSTR(NA(RYPNTR+11),NA(NA(RYPNTR + 10)+KCAT),SUBM2)
          TTT1 = T(INTNOD(SUBM1,NA(NA(RYPNTR+3) + KCAT)))
          TTT2 = T(INTNOD(SUBM2,NA(NA(RYPNTR+4) + KCAT)))
           IF (TTT1 .GT. TTT2) THEN
              THOT = TTT1
          ELSE
              THOT = TTT2
          ENDIF
           CALL KNNUM(THOT,NA(RYPNTR + 1),MOLWT,PGAS,A(NA(RYPNTR + 8)+KCAT),KNUD,FLGO)
           CALL FMHT(TTT1,TTT2,A(NA(RYPNTR+5)+KCAT),A(NA(RYPNTR+6)+KCAT),
     *    GAM,MOLWT,PGAS,A(NA(RYPNTR+7)+KCAT),ATEST,BTEST,FMCND)


          IF (FLGO .EQ. 3) THEN
             G(INTCON(SUBMOD,NA(NA(RYPNTR+2)+KCAT))) = FMCND
          ELSEIF (FLGO .EQ. 2) THEN
            G(INTCON(SUBMOD,NA(NA(RYPNTR+2)+KCAT))) = FMCND
          ELSE
            G(INTCON(SUBMOD,NA(NA(RYPNTR+2)+KCAT))) = FMCND*100.0
          ENDIF

       ENDDO
FSTOP
       RETURN
       DEBON
       END
C**************************************************************
C**************************************************************
F       SUBROUTINE GETSTR(KRAY,INUM,STROUT)
C*************************************************************
C PROGRAMMER: Eric Malroy
C DATE: 3/2/2007
C LAST MOD: 3/2/207
C DISCRIPTION: returns the INUM substring within a character
C   string that is separated by a colon (:).
C   Example
C     KRAY = "bigboy:hours:girl:boy:story:silly:happy"
C           note: neglect the "
C     if INUM = 3  STROUT = "boy"
C     if INUM = 1  STROUT = "bigboy"
C     if INUM = 7  STROUT = "happy"
C
C*************************************************************
C INPUT ARGUMENTS:
C ****************
C    KRAY .......Pointer to Character Array
C    INUM .......Number of the substring to select
C OUTPUT ARGUMENTS:
C    STROUT
C*****************
C Important Information: The array cannot contain ":" or " "
C**************************************************************
       CALL COMMON
       DEBOFF
FSTART
       CHARACTER(len=128) STRHLD
       CHARACTER(len=8) STROUT
       INTEGER KRAY,ITER,FLG,INDX1,INDX2,INUM,SZ_STR,IST
       FLG = 1
       INDX1 = 0
       STRHLD = UCA((KRAY))
       SZ_STR = LEN(STRHLD)
       DO ITER = 1,INUM
           INDX2 = 0
          FLG = 1
          DO WHILE (FLG==1)
             INDX2 = INDX2 + 1
C            WRITE(NUSER1,780)ICHAR(STRHLD((INDX1+INDX2):(INDX1+INDX2))),ICHAR(':')
C 780           FORMAT(I4,I4)
              IST = ICHAR(STRHLD((INDX1+INDX2):(INDX1+INDX2)))
             IF ((IST .EQ. (ICHAR(':'))).OR.(IST .EQ. 32)) THEN
                IF (ITER == INUM) THEN
                  STROUT(1:(INDX2-1)) = STRHLD(INDX1+1:(INDX1+INDX2))
```

```
                 ENDIF
              FLG = -1
              INDX1 = INDX1 + INDX2
C               WRITE(NUSER1,781)STRHLD((INDX1+INDX2):(INDX1+INDX2)),STROUT
C 781               FORMAT('HEHE',A,'  ',A)
              ENDIF
           ENDDO
       ENDDO
C       WRITE(NUSER1,777)INUM,SZ_STR,STROUT
C       TTT1 = T(INTNOD(STROUT,50))
C       WRITE(NUSER1,778)TTT1
C 778    FORMAT('Temperature is ',F15.5)
FSTOP
       RETURN
       DEBON
       END
C**************************************************************
```