**A Computer Model of Fire Spread from Engine to Passenger Compartments in Post-Collision Vehicles**

by

James A. Ierardi

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Fire Protection Engineering

by

May 1999

APPROVED:

Prof. Jonathan R. Barnett, Ph.D., Major Advisor

Prof. Walter A. Kistler, Reader

Prof. David A. Lucht, P.E., Head of Department

**Abstract**

The interaction between the heat flux and fluid flow of an engine compartment fire and the windshield of a post-collision passenger vehicle has been studied using analytical methods. A computational fluid dynamics model of this scenario has been developed with TASCflow using a turbulent, reacting, multi-component fluid flow in a multi-grid domain with conjugate heat transfer objects. A group of computer programs have also been created to automate the grid generation and model construction processes. Calculation tools have also been developed using aspects of fire dynamics for the purpose of making comparisons to the results of CFD modeling as well as experimental measurements. A framework has been established for the modeling and validation of the windshield problem using the tools developed in this study.

*This research has been sponsored by General Motors Research Corporation pursuant to the GM/DOT settlement agreement.*

**Acknowledgements**

Thanks to Professor Barnett for taking an interest in my work as an undergraduate and for introducing me to the world of research. His high standards of academic excellence have challenged me to become a better student and push myself to limit.

Thanks to Professor Kistler for his contributions to this work as a whole. In taking his Stress Analysis and Fluid Mechanics classes as an undergraduate, I discovered the true meaning and benefit of working hard. His classes challenged me to extend and apply myself academically to points where I had never been before. Again I was challenged to explain the finer points of the fluid and heat transfer processes involved in my thesis. My research has benefited greatly from this and I am most thankful.

To Dr. Kenneth Strom and the General Motors Research Corporation for involving us with this interesting work.

To my parents and brothers for their undying support. Education has been their top priority and I appreciate the sacrifices they have made to invest in my education. Integrity and hard work are the center of the values that they have instilled in me, and have given me the strength to pursue my education at this level.

Most importantly, to Gina for her amazing support and patience with my academic endeavors. She has always been there to pick me up during the low points as well as to celebrate the good days. She understands what I have gone through and realizes how important this really is to me. My sanity is indebted to her.

*"There's no shame in getting knocked down,…the only shame is in not getting back up."* – Joseph J. Ierardi, my dad

## Executive Summary

The spread of fire from engine to passenger compartments of post-collision passenger vehicles has been studied. Methods for quantifying the heat flux and fluid flow from an engine compartment fire to the windshield of a post-collision vehicle have been examined for the purpose of assessing passenger compartment tenability conditions. The intent of this project was to:

(1) continue the investigation into the development of an engine compartment design fire scenario with emphasis on the identification of an appropriate design heat release rate,

(2) begin the calibration and validation process of the fire modeling results using the experimental data of GM/DOT Project B.3, “Fire Initiation and Propagation Tests,”

(3) using the TASCflow Computational Fluid Dynamics software, continue the development of an appropriately parameterized combustion model using turbulent, reacting, multi-component fluid flow with heat transfer by convection, conduction, and radiation,

(4) apply TASCflow to the case of a windshield exposed to convective and radiant heat flux from an engine fire to evaluate fire spread from an engine compartment fire into a post-crash vehicle’s passenger compartment through the windshield opening, and

(5) evaluate the quality of the windshield model developed in task 4 using the data from the GM/DOT Project B.3, “Fire Initiation and Propagation Tests”.

A literature review of the post-collision vehicle fire statistics collected by the National Fire Protection Association (NFPA) revealed that a likely scenario is the engine compartment fire. A review of passenger vehicle fire research for the purpose of locating an appropriate design heat release rate for the engine compartment fire scenario was also conducted. This resulted in the selection of a heat release rate developed by the ProfilARBED research group based on full-scale fire testing of automobiles manufactured from the late 1980's and early 1990's.

In order to calibrate the fire modeling results, standard techniques based on fire dynamics were utilized. This resulted in the development of calculation methods for the purpose of checking the output of the Computational Fluid Dynamics (CFD) model. The calculation methods include assessments of the adiabatic flame temperature, the mean flame height, as well as the fire plume temperature and velocity profiles.

Finally, an appropriately parameterized combustion model using turbulent, reacting, multi-component fluid flow with heat transfer by convection, conduction, and radiation was developed for use in the commerical CFD code TASCflow. The resultant model was used to predict windshield behavior in the post-crash vehicle engine compartment fire environment.

# Table of Contents

## List of Figures

## List of Tables

## Nomenclature

### *Symbols*

$b_{\Delta T}$ = radial distance where temperature is half the centerline temperature rise [m]
$c_p$ = specific heat of ambient air at constant pressure [kJ/kgK]
$c_{pi}$ = specific heat at constant pressure for ith product of combustion [J/molK]
D = diameter of fire [m]
E = 13.1 MJ/kg of $O_2$, the heat release per unit oxygen mass [MJ/kg]
g = acceleration due to gravity [$m/s^2$]
H = ceiling height [m]
m' = mass flowrate of gas [kg/s]
$M_{air}$ = 29 g/mol air, the molecular weight of air [g/mol]
$m_{ent}$ = mass flow rate of air entrained into the fire plume [kg/s]
$M_{O2}$ = 32 g/mol $O_2$, the molecular weight of oxygen [g/mol]
$n_i$ = number of moles of ith products of combustion [--]
Q = total heat release rate of fire [kW]
Q* = non-dimensional heat release rate [--]
$Q_c$ = convective portion of heat release rate [kW]
$T_a$ = ambient air temperature [K]
$T_{f,ad}$ = adiabatic flame temperature [K]
$T_o$ = centerline temperature of the fire plume [K]
$u_o$ = centerline velocity in the fire plume [m/s]
$X_i(t)$ = mole fraction of species *i* at time *t* [--]
$X^o_{02}$ = 0.2095 , the mole fraction of oxygen at ambient conditions [--]
$X^o_{CO2}$ = 0.0003, the mole fraction of carbon dioxide at ambient conditions [--]
$X^o_{H20}$ = mole fraction of water vapor at ambient conditions [--]
z = vertical height [m]
$z_o$ = virtual origin of point source of bouyancy [m]
$\Delta H_c$ = heat of combustion for the fuel [kJ/g]
$\Delta T_o$ = centerline temperature rise in the fire plume [K]
$\alpha$ = 1.105, empirical value for expansion factor due to incomplete combustion [--]
$\rho_\infty$ = ambient air density [$kg/m^3$]

### *Abbreviations*

ASTM – American Standard for Testing and Materials
CFD – Computational Fluid Dynamics
DOT – United States Department of Transportation
FMRC – Factory Mutual Research Corporation
FRS – Fire Research Station
GM – General Motors
GMRC – General Motors Research Corporation
HPR – High Penetration Resistance
HRR – Heat Release Rate [kW or MW]
MVSS – Motor Vehicle Safety Standard
NFIRS – National Fire Incident Reporting Service

NFPA – National Fire Protection Association
PVB – Polyvinyl Butyral
SFPE – Society of Fire Protection Engineers
WPI – Worcester Polytechnic Institute

# 1.0 Background

## *1.1 Introduction*

This report describes the results of work completed for the second deliverable to General Motors and is a continuation of work presented in the first deliverable entitled "Computational Fluid Dynamics Modeling of Post-Collision Vehicle Fires". This work describes the investigation of the windshield of a post-collision passenger vehicle subjected to the heat flux and fluid flow of an engine compartment fire. This scenario is of interest as it is one of the primary means of flame spread from the engine compartment to the passenger compartment in such fires. The purpose of this background section is to describe the statement of work that has been agreed upon and to provide a guide to the rest of this report.

## *1.2 Discussion of the Deliverable*

The work presented in this report is a continuation of a sub-project started in the second year of the GM/DOT settlement agreement (work conducted 8/1/96 to 3/31/97) and described in the May 1997 report "Computational Fluid Dynamics Modeling of Post-Collision Vehicle Fires" by Nathan B. Wittasek, Richard D. Pehrson, and Dr. Jonathan R. Barnett. The following research has been conducted by Worcester Polytechnic Institute (WPI) for the second deliverable to General Motors (GM);

1. Continue the investigation into the development of an engine compartment design fire scenario with an emphasis on an appropriate design heat release rate.

2. Begin the calibration (validation) of fire modeling work using the results of the Project B.3, “Fire Initiation and Propagation Tests”.

3. Using TASCflow Computational Fluid Dynamics software, continue the development of an appropriately parameterized combustion model using turbulent, reacting, multi-component fluid flow, with heat transfer by convection, conduction, and radiation.

4. Apply TASCflow to the case of a windshield exposed to convective and radiant heat flux from an engine fire. This case evaluates fire spread from an engine compartment fire into a post-crash vehicle’s passenger compartment through the windshield opening.

5. Evaluate the quality of the windshield model developed in task 4 using the data from the Project B.3, “Fire Initiation and Propagation Tests”.

## *1.3 Guide to the Report*

The report that follows addresses each item of the statement of work described in the previous section. It begins with the investigation and development of an engine compartment design fire scenario that is appropriate for the windshield problem. The investigation into the design scenario included an examination of post-collision vehicle fire statistics as well as a review of available vehicle fire research. Using the information

from these sources, a design heat release rate was selected for use in the modeling of the windshield problem.

Following the design fire development is a discussion of validating the selected design fire in Chapter 3. A general discussion of model calibration and validation is presented in an effort to illustrate its importance as well as its complexity. The theory of the relevant flame height and fire plume correlations are presented as a means of comparing the results of a computational model and experimental measurements.

The third item of the statement of work is addressed in Chapter 4 where the construction of the parametric TASCflow combustion model is presented. The selection of phenomena to model and the appropriate sub-models to use in modeling the windshield problem are discussed. The application of relevant boundary conditions and the determination of an initial guess are then described for the parametric combustion model. A series of computer programs have been developed to automate the creation of the computational domain and the assignment of boundary conditions for the parametric model.

Applications of the parametric model are detailed in Chapter 5. The scenarios considered are presented as well as a discussion of the results of this effort.

The evaluation of the windshield model is described in Chapter 6 beginning with a general discussion of calibration and validation. This is followed by analytic comparisons to evaluate the quality of the fire and heat transfer that is modeled. A framework is then presented for comparing the parametric model results to experimental measurements.

Chapter 7 provides the conclusions of the work and recommendations for future efforts.

## 2.0 Engine Compartment Design Fire Scenario

An important consideration in the development of a fully parametric model of a post-crash passenger vehicle windshield subjected to the heat flux and fluid flow of an engine fire is the development of an appropriate design fire scenario. The selected design fire scenario becomes the basis for the computational fluid dynamics modeling effort. A review of post-collision vehicle fire statistics was conducted to investigate the frequency of post-collision vehicle fires. An emphasis was placed on fires originating in the engine compartment as the heat flux and fluid flow of engine compartment fires pose a risk to the integrity of a post-collision passenger vehicle windshield. Cited ignition sources were listed for understanding the origin of post-collision fires in relation to the information gained from the review of car fire statistics.

A review of the available vehicle fire research was conducted in an effort to quantify the burning behavior of vehicles in lieu of experimental data pertaining to the test burns of post-crash vehicles conducted by GM at Factory Mutual as part of GM/DOT Project B.3, “Fire Initiation and Propagation Tests”. A design fire heat release rate has been selected for the case of a windshield subjected to the heat flux and fluid flow of an engine fire after considering the information gathered from post-collision vehicle fire statistics, vehicle fire ignition sources, and the available vehicle fire research. The appropriateness of the selected design fire heat release rate is discussed as a comparison is made between the types of vehicles, nature of ignition sources, and fire growth characteristics of the selected design fire heat release rate and the test burns conducted by General Motors. A methodology for defining a heat release rate from calorimetry data

collected during the GM test burns is also presented so a design heat release rate can be developed and implemented in the model once such data becomes available.

## *2.1 Post-Crash Vehicle Fire Statistics*

Post-collision vehicle fire statistics provided an indication of the frequency of such incidents and also indicated potential ignition sources. Using this information as a basis, typical design fires for post-collision vehicle fire scenarios were developed. In order to construct a reasonable fire growth model for use as the design fire of a post-collision vehicle the appropriate material property data relevant to fire growth was collected from various sources and cited in the report "Computational Fluid Dynamics Modeling of Post-Crash Vehicle Fires". A summary of these findings is included in the sections below.

The research conducted in the area of collecting post-collision vehicle fire statistics demonstrated the frequency of these types of fires. According to statistics collected by the National Fire Protection Association's (NFPA) Annual Fire Experience Survey in conjunction with the National Fire Incident Reporting Service (NFIRS) [5], of the 405,300 vehicle fires reported in 1992, the 7,990 vehicle fires resulting from collisions, overturns, and knockdowns (1.8% of all vehicle fires) accounted for 469 (65.4%) of all vehicle fire deaths and 677 (22%) of all vehicle fire injuries. Over two-thirds of all post-collision vehicle fires are attributed to fires starting in the engine compartment, running gear, or wheel area and resulted in over one-third of all post-collision vehicle fire deaths and over one-half of the injuries. One-fifth of all post-collision vehicle fires are attributed to fires originating in the passenger compartment and

account for 22.3% of all post-collision vehicle fire deaths. The remaining 12% of all post-collision vehicle fires are attributed to the fuel line or fuel tank and are responsible for 22% of all post-collision vehicle deaths. It should be noted, however, that these statistics indicate situations in which injuries and deaths occurred in vehicles that were involved in a collision that resulted in a fire. Some of the reported deaths and injuries could be attributed to the collision and have nothing to do with the resulting fire [5]. The above statistics show that post-collision vehicle fires, particularly ones originating in the engine compartment, are an important fire safety issue for all vehicle manufacturers. The number of fire related deaths and injuries could in fact be lower, indicating that there is a less severe firesafety issue than the statistics reveal due to the inability of the vehicle to protect the occupants from effects of the crash impact.

## *2.2 Ignition Sources*

The statistics on post-collision vehicle fires indicate two likely scenarios; fires originating in the engine compartment and those originating from the fuel delivery system. In order to create appropriate design fires for each scenario, the typical ignition sources and materials involved in the fire growth were investigated.

The ignition of liquid and solid fuels is briefly reviewed to put the discussion of ignition sources in proper perspective. Combustion is an exothermic reaction that is dependent upon the nature of fuel, oxidant, and ignition energy. The fuel must be in a gaseous form. For liquid fuels, the conversion to vapor form occurs due to evaporation. Evaporation can occur under a wide range of temperatures depending upon the equilibrium vapor pressure of the liquid. In the case of solid fuels, the vaporization

process is known as pyrolysis and involves the thermal decomposition of the solid into a gaseous form. The gaseous fuel must be in the presence of sufficient oxygen in order to produce a combustible mixture of fuel and oxidant. For a particular fuel, there is a lower flammable limit (LFL) where a minimum amount of oxygen is necessary for the combustion reaction. An oxygen concentration below this lower flammable limit will not result in a combustion reaction, even in the presence of an ignition source. The ignition of a flammable mixture, *ie*. a mixture of fuel and oxidant that is above the lower flammable limit, requires sufficient energy to initiate the combustion reaction depending upon the mixture. This energy can be in the form of a spark or sufficiently high temperature. The spark energy can be electrical in nature or an ember from a fire. A surface or local air volume of sufficiently high temperature can provide enough energy to ignite the mixture of fuel and oxidant. Obviously energy sufficient to cause an ignition can be present in the absence of fuel. The discussion that follows presents some mechanisms of ignition conditions and is not to be considered a complete identification of ignition sources in post-crash vehicles. It should also be noted that no attempt has been made to quantify the likelihood of the occurrence of such ignition sources, as this topic is beyond the scope of this project. The material below is provided for background information purposes only.

### 2.2.1 Engine Compartment Ignition Sources

The engine compartment contains a number of mechanical and electrical components in a vehicle and is subjected to elevated temperatures during the normal operation of a vehicle. This supports the statistical results that two-thirds of all post-

collision vehicle fires originate in the engine compartment [5]. The insulation of electrical wires can degrade when subjected to the hot temperatures of an operating engine and therefore cause an electrical fault when bare wire becomes exposed [14]. Electrical faults, or arcing, can occur in the ignition wires, at the distributor contact points, and at the battery [14] as well as any other component of the electrical system. Another possible source of ignition in the engine compartment is overheated engine components. Although most engine parts have a normal operating temperature below gasoline's auto-ignition temperature, mechanical failure can cause overheating of engine components to temperature above the auto-ignition temperature of gasoline [14].

Electrical system ignition sources are a common origin of post-collision vehicle engine fires [8]. Ignition sources in the engine compartment typically consist of

- battery-ignition coil circuitry
- battery-alternator circuit
- battery-starting motor circuit

The wires that connect these particular components are long and it is possible that they could be cut and/or spark in a collision. The energy from such a spark, as a point source of energy, has to be only 0.15 milli-joules to ignite a stoichiometric mixture of hydrocarbon and air, to initiate a fire in an engine compartment [8]. Overheated wires, even when red hot, will not ignite flammable gas/air mixtures, but an ignition can occur if the wire sparks or the insulation melts. The battery, even with lost electrolyte, is able to create enough electricity to become a spark hazard in the presence of a flammable mixture [8].

The battery and it's cables are a possible ignition source because they are located within the crash zone for a frontal impact accident and have sufficient charge to create a spark to ignite a flammable fuel/air mixture, even when on low charge [8]. The metal contacts of the battery are often uninsulated and exposed. Likewise, although battery cables are heavily insulated, the ends where connections are made are often uninsulated [8]. Starter solenoids are located close to the battery or attached to the starter. Despite being ruggedly designed, the terminals that connect the starter solenoid to the battery and the starter are exposed and can create a spark if the sheet metal is sufficiently displaced in an accident [8]. Voltage regulators are usually located in the front bulkhead and its terminals are also exposed and capable of sparking in a collision [8]. Lights and the lighting system wiring can be an ignition source because headlight filaments can stay hot enough to cause an ignition up to 0.5 seconds after the bulb breaks, which can be enough time to come into contact with a flammable mixture [8]. The wires are long and run through the bulkhead to connect to the instrument panel, therefore there is enough wire to be cut or spark in an accident [8]. The wires for the horn also pose a similar problem as headlight wiring in a vehicle collision [8]. Alternators and generators produce 12 to 14 amperes of electrical current and have exposed terminals capable of generating sparks in the event of an accident [8].

### 2.2.2 Fuel Tank/Line

The integrity of the fuel system (fuel tank and fuel line) can become compromised in a car crash and result in a fuel leak. A gasoline pool can form underneath a car due to the gasoline issuing from a leak and the diameter of this pool will increase as fuel empties

from the tank. If this fuel comes into contact with an ignition source, the subsequent fire has the potential of coming into contact with the underside of the vehicle. The ignition source in this case might be the catalytic converter, which normally operates at a temperature above the ignition temperature for gasoline [14]. A failure within the fuel line can involve a variety of situations. One extreme is the complete severing of the fuel line, which would cause gasoline to rapidly discharge under pressure. The amount of fuel discharged is limited by the safety interlock features associated with the fuel delivery system. Another situation is a small leak in the fuel line, which under pressure can cause the gasoline to atomize. This makes gasoline easier to ignite because it is in a vapor form [10]. Electronic fuel injection systems and pollution control devices in automobiles require higher operating pressures in their fuel lines than vehicles without these features [10]. High pressure becomes important when considering even small leaks in a fuel line. When fuel or any other flammable liquid comes out of an opening and contacts any surface that is at a temperature above the liquid's flashpoint, ignition of the liquid will occur.

## *2.3 Review of Vehicle Fire Research*

In order to develop an appropriate model of a windshield subjected to convective and radiant heat flux from an engine compartment fire a proper design heat release rate must be identified. Therefore, a literature review of vehicle fire experiments was performed to develop a design heat release rate by applying information contained in the literature.

In comparison to building fire research, there has been very little work performed in the area of vehicle fires. The first contribution of experimental data in this area was by the Fire Research Station in 1968 when Butcher, Langdon-Thomas, and Bedford made temperature measurements of vehicles burning in enclosed car parks [11]. Bennetts, in investigating car park fires, measured temperatures of burning cars in work done in Australia in 1985 and 1988 [11].

In a similar manner, Haksever measured gas temperatures from burning cars in car park structures in "Fire Engineering Design of Steel Framed Car-Park Buildings" in 1990 [11]. However, the objective of these studies was to measure gas temperatures above a burning vehicle for determining the response of structural steel members used in parking garage designs.

### 2.3.1 Mangs and Keski-Rahkonen

The first significant contribution of experimental data that could be used in the computer modeling of vehicle fires came from work done by Mangs and Keski-Rahkonen in 1994 entitled "Characterization of the Fire Behavior of a Burning Passenger Car. Part 1: Car Fire Experiments." The importance of this particular work is that it included measurements of the heat release rate, which is a critical parameter to consider in the design of fire protection systems or in the computer modeling of particular fire hazards.

### 2.3.2 Shipp and Spearpoint (FRS)

In assessing the fire safety issues associated with vehicles traveling via the Channel Tunnel, the Channel Tunnel Safety Unit's Department of Transport

commissioned the Fire Research Station to investigate the issue of vehicle fires. The work done by Shipp and Spearpoint [16] included the measurement of heat release rate, gas temperatures, emission of toxic products, and thermal radiation for tests of two full-scale burning automobiles.

The experimental apparatus consisted of a an enclosed canopy 2.2 m high by 3.5 m wide by 8.7 m long with a 3.0 m by 3.0 m large scale calorimeter placed at either end of the canopy. Each calorimeter was setup to measure the gas temperature and velocity as well as the concentrations of oxygen, carbon dioxide, and carbon monoxide.

Two vehicles were tested in this experiment, a 1982 Austin Maestro and a 1986 Citroen BX. The Maestro was used in a seat ignition test and the Citroen was used in an engine ignition test. Both vehicles were in working order with gas tanks at least three-quarters full, some luggage in the trunk, and papers on the seats and dashboard. The windows were open and doors closed.

In the seat ignition test of the Maestro, a No.7 wood crib was used as the ignition source with an estimated peak heat release rate of 10 kW. This particular ignition source was chosen because it was felt, after two separate ignition tests of the seat alone, that the wood crib would only affect the initial fire growth in the vehicle.

During the experiment it was noted that within a minute of the wood crib being ignited that flames were visible in the vehicle. The interior of the car was fully involved in the fire 6 minutes after the test began with a growing pool fire below the car that consisted of burning melted plastic and car fluids. The rest of the car was involved in the fire after 11 minutes and after 13 minutes fuel began to leak out of the fuel tank and the fire grew larger. At this point, smoke began to escape from the test rig and flames were

present in the ducts, resulting in the loss of some instrumentation. At 17 minutes the test was concluded as the fire brigade extinguished the fire due to concerns of the test rig collapsing. After the test, it was found that the rubber hose connecting the filler pipe to the fuel tank had been compromised in the fire. Therefore, the fuel would have spilled directly out of the tank and was the mostly likely cause of the increase in fire intensity.

The canopy thermocouple measurements for the Maestro seat ignition test were as high as 1250 C at 13 minutes, when flames completely engulfed the test rig. The thermocouple placed on the roof of the vehicle showed a peak temperature of 1100 C. The thermocouple measurements recorded inside of the vehicle showed peak temperatures of 1000 C. It was noted that there was no evidence of layering (a distinct smoke layer) in the vehicle. According to heat flux measurements taken in this test, the severity of the fire increased dramatically after 13 minutes with readings of 60 to 80 $kW/m^2$. The peak heat release rate value recorded in the test was 7.5 MW. However, a peak value of 8.5 MW was estimated due to a failure of oxygen sensor in the primary exhaust hood at 13 minutes.

For the engine fire, the hood of the Citroen was open and a foil tray filled with 400 ml of gasoline. The gas in the tray was ignited and the hood was closed. After two minutes visible flames were observed around the hood of the vehicle. Inside the car, smoke was visible within four minutes of ignition. Five minutes after ignition, flames were seen coming in from underneath the dashboard. This is consistent with a failure of the bulkhead to prevent flames from spreading from the engine compartment into the passenger compartment. The fire in the Citroen was not as severe, in terms of the peak heat release rate, as that in the Maestro test and burned down to a tire fire and was

extinguished after 57 minutes. The peak temperatures measured by the thermocouples in the canopy were about 1125 C. The roof thermocouple of the car had a maximum temperature of 1100 C and the interior thermocouples measured temperatures around 1250 C. The peak heat flux measurements were 50 to 60 kW/$m^2$ after 18 minutes into the test. The peak heat release rate for this test was 4.5 MW.

The peak heat release rates of 8.5 and 4.5 MW measured by Shipp and Spearpoint were much greater than the values of 1.5 to 2 MW recorded by Mangs and Keski-Rahkonen. The differences between these measurements could be the result of two factors. The first of which is the fact that the Shipp and Spearpoint experiments were conducted in an enclosure as opposed to the work by Mangs and Keski-Rahkonen that was conducted without an enclosure. The presence of the enclosure would create a greater amount of re-radiation to the burning vehicle and produce a fire of greater intensity than that of one in the open. The second factor could be that the vehicles tested by Mangs and Keski-Rahkonen were from the 1970s and contained less plastic components than the vehicles from the early to mid 1980s used by Shipp and Spearpoint.

### 2.3.3 ProfilARBED Research Center

A comprehensive study of car fires is documented in "Development of Design Rules for Steel Structures Subjected to Natural Fires" by the ProfilARBED Research Center in Luxembourg [1]. A series of nine full-scale experiments were carried out in a simulated car park to measured heat release rate, gas temperatures, species concentrations, and heat transfer to steel structural members of the parking structure. Each test involved either one or two vehicles and includes vehicles of different makes,

models, and sizes in order to account for the various vehicles that would be found in a car park structure. The tests consisted of the following vehicles manufactured in the 1980s (tests 1 through 6) and 1990s (tests 7,8, and 9).

- *Test 1* -- Mazda 323 (829 kg) and Talbot Solara (weight not measured)
- *Test 2* -- Renault 18 (951 kg)
- *Test 3* -- Renault 5 (757 kg)
- *Test 4* -- Renault 18 (955 kg)
- *Test 5* -- BMW (1150 kg) and Renault 18 (736 kg)
- *Test 6* -- Citroen BX (870 kg) and Peugeot 305 (1073 kg)
- *Test 7* -- large car (make and model not specified) (1303 kg)
- *Test 8* -- small car (make and model not specified) (830 kg)
- *Test 9* -- small (789 kg) and large car (1306 kg) (make and model not specified)

Each test vehicle contained all relevant fluids, a spare tire, and the fuel tanks were 2/3 full of gasoline. The test rig consisted of a 5 m by 5 m calorimeter hood and a simulated ceiling at a height of 2.3 m for the first 2 tests and at a height of 2.6 m for the remaining tests. The sides of the test rig were completely open to ambient air, unlike the experiments conducted by Shipp and Spearpoint. The ignition source used in the first seven tests was 1.5 liters of gasoline in a tray placed under the left front seat of the vehicle. The final two tests had used 1 liter of gasoline in a tray placed under the gear box of the vehicle. For the vehicle being ignited, the driver's side window was left completely open while the passenger side window was left half-open. For the instances in which two cars were tested, both windows were completely closed in the secondary vehicle.

The results of this work provided some general insights into the burning behavior of motor vehicles. The mass loss from each burning vehicle was between 15% to 19% of the vehicle's initial mass. Using the mass loss and heat release rate information gathered from the experiments the ProfileARBED researchers were able to deduce an effective heat of combustion for a motor vehicle in the range of 16.6 MJ/kg to 29 MJ/kg. Using the data collected in the experiments, a reference curve for the heat release rate of a burning motor vehicle was developed for use in fire safety engineering calculations. The design curve is plotted with the heat release rates recorded during the car fire experiments and is shown below;

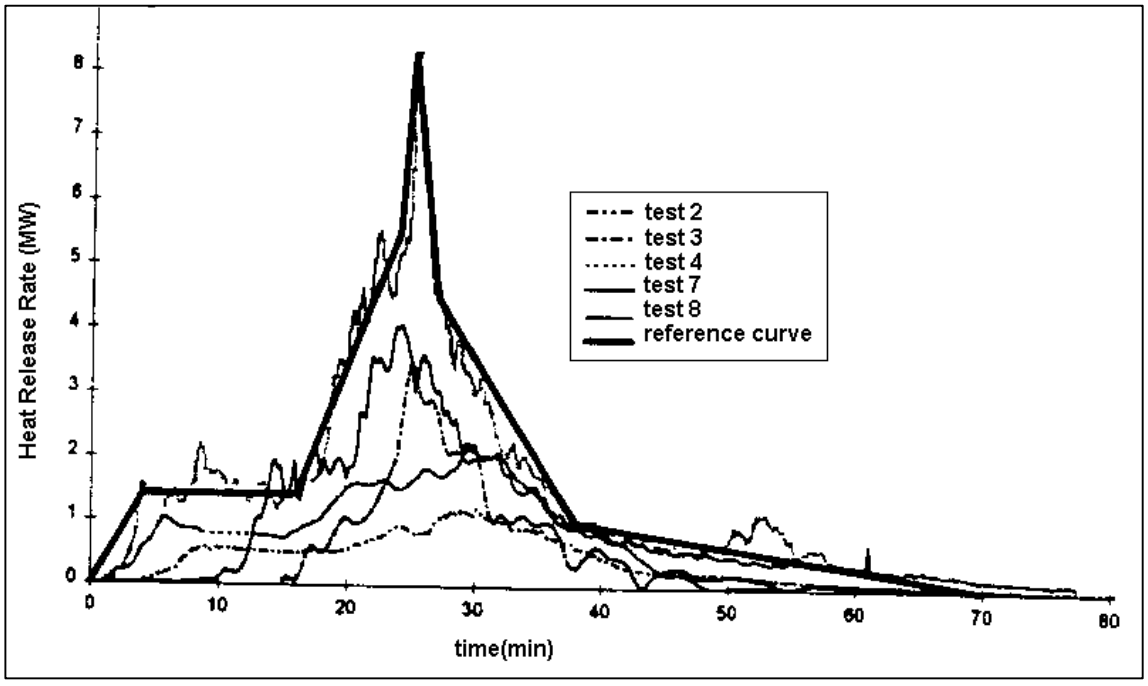


**Figure 1-- ProfilARBED design heat release rate for a burning motor vehicle plotted with experimental heat release rates.**

The reference curve contains the following data points;

| time (min) | HRR (MW) |
|---|---|
| 0 | 0 |
| 4 | 1.4 |
| 16 | 1.4 |
| 24 | 5.5 |
| 25 | 8.3 |
| 27 | 4.5 |
| 38 | 1 |
| 70 | 0 |

**Table 1 -- Data points for ProfilARBED reference heat release rate curve of a burning motor vehicle.**

A method has been presented for calculating the heat release rate of a burning car based on the theory of oxygen calorimetry [1]. The theory of oxygen calorimetry is that for every kilogram of oxygen consumed by a fire, 13.1 Megajoules of energy is released.

$$Q(t) = E\frac{\Phi(t)}{1+\Phi(t)(a-1)}m(t)\frac{M_{O_2}}{M_{air}}\left(1-X^{o}_{H_2O}-X^{o}_{CO_2}\right)X^{o}_{O_2} \qquad (1)$$

This assumption has been proved through various experiments to be accurate within 5% [1]. This method allows for the determination of the rate of heat release for a given object. The method for determining the rate of heat release of a burning vehicle is shown below.

$$\Phi(t) = \frac{X^{o}_{O_2}\left(1-X_{CO_2}(t)-X_{CO}(t)\right)-X_{O_2}(t)\left(1-X^{o}_{CO_2}\right)}{\left(1-X_{O_2}(t)-X_{CO_2}(t)-X_{CO}(t)\right)X^{o}_{O_2}} \qquad (2)$$

where;

$E$ = 13.1 MJ/kg of $O_2$, the heat release per oxygen mass.

$M_{O2}$ = 32 g/mol $O_2$, the molecular weight of oxygen.

$M_{air}$ = 29 g/mol air, the molecular weight of air.

$a$ = 1.105, empirical value for expansion factor due to incomplete combustion.

$m(t)$ is the mass flowrate of the gases at time $t$ and is calculated from the volume flowrate and temperature of the gases entering the exhaust hood at a given density of 1.19 kg/m$^3$ and a temperature of 300 K.

$X^o_{H20}$ = , the mole fraction of water vapor at ambient conditions.

$X^o_{CO2}$ = 0.0003, the mole fraction of carbon dioxide at ambient conditions.

$X^o_{02}$ = 0.2095 , the mole fraction of oxygen at ambient conditions.

$X_i(t)$ is the mole fraction of species $i$ at time $t$.

If the mole fractions of oxygen, carbon dioxide, carbon monoxide, and water vapor as well as the mass flowrate of gases into the exhaust hood are known for different times, then the heat release rate for a motor vehicle as a function of time can be calculated with this method. However, it must be noted that this equation is based upon the theory of oxygen calorimetry, and therefore, the products of combustion must be captured by the calorimeter's collection hood in order to ensure an accurate estimation of the heat release rate. If some products of combustion escape, as the size of the fire increases for example, then the calculated heat release rate would underestimate the severity of the fire.

This heat release rate calculation can be put into a spreadsheet format that accepts the mole fraction of each species at specified time intervals and computes the heat release rate as a function of time. This can be used as a means of generating design fires based on experimental data for input into the Computational Fluid Dynamics model. This would provide a method of calibrating the CFD model with experimental data.

With such a limited knowledge base to serve as a foundation for further work, there are many unresolved issues in the area of vehicle fires. Such issues include flame spread and the effect of compartment ventilation. Despite the fact that a great amount of research has been conducted in the area of building fire safety, a parallel cannot necessarily be drawn between compartment fires and vehicle fires. One of the major obstacles in the computer modeling of vehicle fires is that since the passenger compartment of a vehicle is much smaller in comparison to a building compartment, we cannot make a two-zone assumption for the ambient air and products of combustion.

## *2.4 Selection of Design Fire Heat Release Rate*

Pending the identification of a more appropriate design heat release rate or the acquisition of experimental data, the design heat release rate that will be used for modeling the windshield of a post-collision vehicle subjected to an engine fire will be the ProfilARBED reference curve. The model also accepts input from the ProfilARBED method for calculating the heat release rate using time-dependent gas concentrations from full-scale experiments in the event that such data becomes available. The heat release rate increases linearly from 0 to 1.4 MW over 4 minutes and it remains at this level until 16 minutes where it increases to 8.3 MW at 25 minutes. This increasing portion in the curve most likely represents the spread of the fire from the engine compartment into the passenger compartment. Therefore, the first 16 minutes of the curve will be considered for the modeling of the windshield problem. At the same time, the conditions of the experiments conducted by GM at Factory Mutual need to be considered. The experiments were concluded at the point when the headliner of the vehicle's interior was

ignited. Therefore, the portion of the design curve selected for the modeling effort is consistent with the criteria of the vehicle fire tests conducted by GM. However, it should be noted that the design heat release rate curve has not been evaluated in comparison to the heat release rates of the vehicles tested by General Motors.

## *2.5 Conclusions*

An engine compartment design fire scenario has been developed from the heat release rates in ProfilARBED tests. The engine compartment design fire scenario was constructed after considering post-collision vehicle fire statistics that revealed the most common fire scenario originates in the engine compartment. A review of vehicle fire research was conducted to locate an appropriate heat release rate for a vehicle fire originating in the engine compartment. Two methods have been identified for specifying the heat release rate in the model; the ProfilARBED reference curve and an oxygen calorimetry procedure.

# 3.0 Validation of Fire Modeling

Assessing the validity of fire modeling is an important, yet complicated process that is a necessary part of a computer simulation. This section of the report focuses on the appropriateness of the selected design fire simulated in the TASCflow model. A discussion of some of the issues involved in the calibration and validation of fire modeling is presented first. In lieu of experimental results, a series of comparisons of the model output to analytic expressions have been made in an attempt to evaluate the modeling of the design fire. Finally, a methodology has been developed for calibrating the simulation of a design fire derived from experimental data, if and when such experimental results become available.

## *3.1 About Validation and Calibration of Fire Modeling*

The validity of computer model simulations depends upon the predictive power of such models [15]. In the specific case of computer fire modeling, the validity of simulations can be assessed by making comparisons to experimental data or to analytic solutions. However, this process can only result in the calibration of the computer fire model to the specific set of experimental data. This is a necessary step towards model validation but it does not result in complete model validation. In fact one may therefore conclude that it is not possible to completely validate a computer fire model, rather one can only calibrate the model to a set of data. Nonetheless, for simplicity, throughout the remainder of this report the word validation will be used to represent model calibration.

One of the main issues that needs to be considered in calibrating a model is the differences that exist between the conditions under which the experiments were

performed and the assumptions that were made in developing the computer model. If only analytic methods are available for validation, the differences between the conditions under which the analytic expressions are valid and the assumptions of the computer model must be carefully considered. The computer model may be limited by its underlying assumptions, resulting in simulation results that are only valid under certain specific conditions. The validation effort of modeling the engine compartment fire is discussed in this section.

In this study the Eddy Dissipation model was used. The Eddy Dissipation model assumes that the chemical reaction rate for combustion is small when compared to the rate of the small-scale mixing rate for the reactants [6]. The small-scale turbulent mixing rate is expressed in terms of the turbulent kinetic energy, *k*, and the dissipation rate, ε. In the case of turbulent diffusion flames, the assumptions of the Eddy Dissipation model are valid because the time scale of the combustion reaction is shorter in comparison to the time scale for the oxygen from the surroundings to diffuse into the gaseous volatiles by turbulent mixing.

## *3.2 Fire Modeling Analytic Comparisons*

In lieu of experimental data about the fire, a series of analytic expressions for flame temperature, flame height and the fire plume characteristics were evaluated for the purpose of assessing the appropriateness of the selected design heat release rate. The adiabatic flame temperature provides an upper bound for temperatures calculated in a CFD model. The evaluation of flame height is performed using a flame height correlation to check the extent of the combustion zone in the TASCflow model. The

characteristics of the fire plume, such as mass flow rate of entrained air, are examined using correlations in order to make a comparison to the thermal plume predicted by the TASCflow model.

### 3.2.1 Flame Temperature

The flame temperatures calculated in the CFD model can be compared to the adiabatic flame temperature for the fuel specified in the model. It should be noted that the quality of this evaluation depends upon the appropriateness of the fuel selected for modeling the engine fire. The adiabatic flame temperature calculation for hydrocarbon fuels begins with a balanced chemical equation of the stoichiometric combustion reaction in order to determine the number of moles of each product. The calculation of the adiabatic flame temperature takes the following form;

$$T_{f,ad} = T_a + \frac{\Delta H_c}{\sum_{i=1}^{n} n_i c_{pi}} \qquad (3)$$

where,

$T_{f,ad}$ = adiabatic flame temperature [K]

$T_a$ = ambient air temperature [K]

$\Delta H_c$ = heat of combustion for the fuel [J/mol]

$n_I$ = number of moles of ith products of combustion [--]

$c_{pi}$ = specific heat at constant pressure for ith products of combustion [J/molK]

The balanced chemical equation for methane combustion is [15];

$$CH_4 + 2O_2 + 7.52N_2 \rightarrow CO_2 + 2H_2O + 7.52N_2 \qquad (4)$$

The heat of combustion of methane is given as [15];

$\Delta H_c$ = 49.6 kJ/g

Which can be expressed on a molar basis as;

$$\Delta H_c = (49.6kJ/g)\left(\frac{1000J}{1kJ}\right)\left(\frac{16gCH_4}{1molCH_4}\right) = 793{,}600J/molCH_4 \qquad (5)$$

The specific heat of the combustion products are evaluated at 1000K because the value of specific heat varies with temperature, therefore it is appropriate to consider this property at an intermediate temperature [7].

| Combustion Product | Cp (J/molK) @ 1000K |
|---|---|
| $CO_2$ | 54.3 |
| $H_20$ | 41.2 |
| $N_2$ | 32.7 |

**Table 2 -- Specific heat of combustion products at 1000K.**

The sum of the number of moles times the specific heat for each combustion product is;

$$\sum n_i c_{pi} = (1)(54.3) + (2)(41.2) + (7.52)(32.7) = 382.6J/molK \qquad (6)$$

Therefore, the adiabatic flame temperature for stoichiometric combustion of methane is;

$$T_{f,ad} = T_a + \frac{\Delta H_c}{\sum_{i=1}^{n} n_i c_{pi}} = 298 + \frac{793{,}600J/mol}{382.6J/molK} = 2074K \qquad (7)$$

The resulting adiabatic flame temperature of 2074K was used as an upper bound in assessing the fluid temperature predicted in the TASCflow model. The actual flame temperature will experience cooling due to heat losses from the turbulent mixing with ambient air and entrainment into the plume as well as from convection and radiation. The adiabatic flame temperature is useful for determining if a solution in the TASCflow model with the combustion sub-model enabled has converged. Temperatures predicted by the model that are above this adiabatic temperature will indicate problems associated with the combustion modeling. The methodology above can be applied for calculating the adiabatic flame temperature of other hydrocarbon fuels as long as the balanced chemical equation for stoichiometric combustion reaction and the heat of combustion on a molar basis are known.

### 3.2.2 Flame Height

Flames are the physical manifestation of the combustion reaction. The luminosity of the flame is governed by the emission of soot particles when subjected to the energy release of the combustion reaction. The flame represents the extent of the combustion reaction in that the boundary between the flame and surrounding air is the boundary at which the combustion reaction is complete. The vertical extent of the combustion reaction is the flame height. However, the definition of flame height can be defined in many ways. Diffusion flames experience a certain degree of turbulence due to density and velocity gradients, which influences the mixing of fuel and air. These phenomena result in a combustion region that is constantly fluctuating, which causes the flame to pulsate or flicker. A turbulent diffusion flame has three distinct combustion regions that

relate to three possible definitions of flame height. The figure below shows the combustion regions in a turbulent diffusion flame.

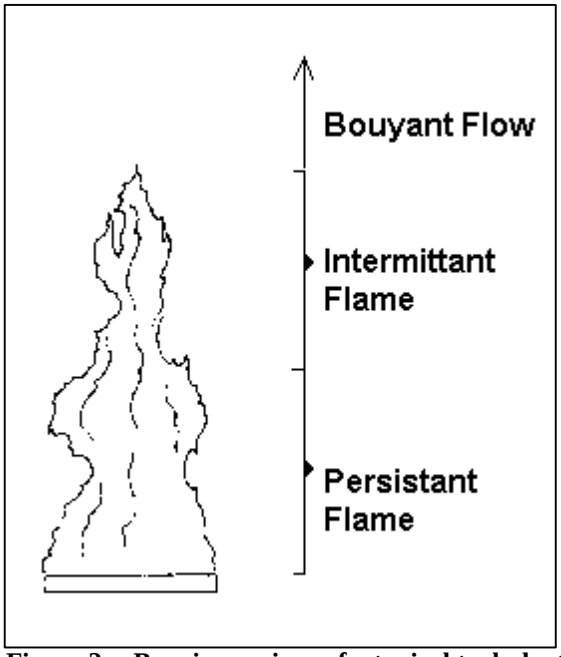


**Figure 2 -- Burning regions of a typical turbulent diffusion flame.**

The persistent flame region in the figure represents the 100% intermittency zone where the flame always exists. The intermittent flame region in the figure represents a combustion zone where the fire exists a percentage of a given time that is less than 100%. The boundary between the intermittent flame and the buoyant plume is the 0% intermittency zone. The mean flame height can be defined as the 50% intermittency zone where the flame exists 50% of the time. Researchers using experimental results for various fuel types have developed flame height correlations. These correlations share a theoretical basis in using the non-dimensional square root of the Froude number expressed in terms of the heat release rate, or Q* [15].

$$\dot{Q}^* = \frac{\dot{Q}}{r_\infty C_p T_\infty \sqrt{gD}D^2} \qquad (8)$$

The Froude number is the ratio of inertial forces to gravitational forces and, therefore, is useful in assessing the buoyant flows associated with fires.

An investigation was conducted into the use of several flame height correlations for the validation of the fire modeled in TASCflow. The flame height correlations considered are shown below [15];

Zukoski

Zukoski's flame height correlation falls into three regimes based on experiments with a natural gas burner with diameters of 10 to 50 cm. The first regime is for Q* values that are less than 0.15, where;

$$\frac{H}{D} = 40\dot{Q}^{*2} \qquad (9)$$

The second regime includes values of Q* from 0.15 to 1, where;

$$\frac{H}{D} = 3.3\dot{Q}^{*2/3} \qquad (10)$$

The third regime involves Q* values between 1 and 40, where;

$$\frac{H}{D} = 3.3\dot{Q}^{*2/5} \qquad (11)$$

Cox and Chitty

The flame height correlation developed by Cox and Chitty is based upon experiments conducted with square natural gas burners with 45 and 60cm lengths and is valid in two regimes. The first is for Q* in the range of 0.13 to 0.28, where;

$$\frac{H}{D} = 15.1\dot{Q}^{*2} \qquad (12)$$

The second regime is for Q* values between 0.28 and 0.55, where;

$$\frac{H}{D} = 3.2\dot{Q}^{*} \qquad (13)$$

Thomas

Thomas's flame height correlation was developed using wood cribs varying from 10 to 200 cm each side. This correlation applies for Q* in the range of 0.75 to 8.8.

$$\frac{H}{D} = 3.4\dot{Q}^{*0.61} \qquad (14)$$

Heskstad

Heskstad's flame height correlation for is described as being for routine use [15] because it was developed using information derived from gaseous, liquid, and solid fuels as well as the literature. This correlation is valid for $0.12 \leq Q^* \leq 1.2E+4$, where;

$$\frac{H}{D} = 3.7\dot{Q}^{*2/5} - 1.02 \qquad (15)$$

These correlations were compared against one another to investigate their behavior for various values of Q*. As indicated in the brief discussion above, Heskestad's flame height correlation applies for the largest range of Q* and although the other flame height correlations showed very good agreement relative to Heskestad's correlation, however, the others were limited in terms of the valid values of Q*. Therefore, Heskestad's flame height correlation will be used because it applies for a wide

range of Q* values and is based on experiments using gaseous, liquid, and solid fuels. It should be noted that these flame height correlations do not directly take the fuel type into account, if we look at Q* and the expression for the flame height. However, in deriving the flame height correlation, fuel types are taken into account.

### 3.2.3 Fire Plume

The adiabatic flame temperature calculation provides a limited amount of insight into the quality of the fire modeling conducted with the CFD model. This is because it only describes the maximum expected amount of energy released, additional information can be used to look at the nature of the energy distribution by the fire. This information can be obtained by examining the characteristics of the fire plume. The fire plume is created by density differences that arise due to the local combustion reaction within an ambient environment and consists of the fire and the products of combustion. The characteristics of the fire plume depend upon vertical and radial locations with respect to the origin of the bouyancy force and centerline of the fire, respectively. Below the mean flame height, where the combustion reaction occurs, the plume is characterized by large density gradients due to the flame temperature and the entrainment of ambient air to provide oxygen to the fuel as well as replace hot gases moving upward due to bouyancy. Above the mean flame height, where the combustion reaction is complete, the plume is characterized by an upward buoyant flow of combustion products and, in order to conserve mass, ambient air is entrained into the fire plume. The air entrained into the plume has a cooling effect on the fire gases and increases the amount of mass in the plume. The velocity and temperature of the fire plume above the mean flame height

attain their maximum values at the centerline of the plume. As the radial distance is increased from the centerline of the plume, the temperature and velocity decreases due to the cooling effects of ambient air that has been entrained. These decreases are often expressed mathematically in terms of gaussian profiles, although there is no theoretical basis to support this assumption. At the same time, the centerline temperature and velocity values decrease with vertical height due to the entrainment of ambient air.

Heskestad has developed a set of plume correlations to describe the plume half-width, the centerline plume temperature rise, and the centerline velocity based on a study of axisymmetric fire plumes. These correlations are referred to as “strong”, or “near field”, plume correlations because they apply to large density differences between the fire gas and ambient air exist and, therefore, are appropriate when examining locations within or near the flame. Different sets of correlations are available for “weak”, or “far field” plumes where density differences between the plume and ambient air are smaller. In the TASCflow model developed for the windshield problem, the fluid domain extending above the plane of the top of the vehicle windshield has been included for the purpose of properly capture the fluid flow and combustion reaction as well as to resolve the constant pressure boundary condition applied. The fluid region extending above the plane at the top of the windshield in the computational domain involves locations that would be considered to be “near field”. As a result, the “strong” plume correlations have been evaluated for the purpose of evaluating the centerline temperature and velocity of the fire plume predicted in the CFD modeling effort. The extension of the fluid domain for the purpose of evaluating the “far field” behavior of the plume alone would introduce additional nodes that would increase the CPU time for the simulation without affecting

the phenomena local to windshield. To this end, Heskestad's "near field" fire plume correlations are presented for the purpose of making a general comparison to the results of the TASCflow model of the windshield problem. It should be noted that these correlations are general in nature and do not provide exact solutions of the plume correlations. However, they are useful for as a guideline for evaluating the general quality of the simulations.

The radial distance to edge of plume where the temperature is half the centerline plume temperature rise is given as [15];

$$b_{\Delta T} = 0.12\left(T_0 / T_\infty\right)^{1/2}\left(z - z_0\right) \qquad (16)$$

where;

$b_{\Delta T}$ = radial distance where temperature is half the centerline temperature rise [m]

$T_o$ = centerline temperature of the fire plume [K]

$T_\infty$ = ambient air temperature [K]

z = vertical height [m]

$z_o$ = virtual origin [m]

The centerline temperature rise in the fire plume is [15];

$$\Delta T_0 = 9.1\left[T_\infty / \left(g c_p{}^2 r_\infty{}^2\right)\right]^{1/3} \dot{Q}_c{}^{2/3}\left(z - z_0\right)^{-5/3} \qquad (17)$$

where;

$\Delta T_o$ = centerline temperature rise in the fire plume [K]

$T_\infty$ = ambient air temperature [K]

g = acceleration due to gravity [m/s$^2$]

$c_p$ = specific heat of ambient air at constant pressure [kJ/kgK]

$\rho_\infty$ = ambient air density [kg/m$^3$]

$Q_c$ = convective portion of heat release rate [kW]

z = vertical height [m]

$z_o$ = virtual origin [m]

The centerline velocity in the fire plume can be calculated from [15];

$$u_0 = 3.4\left[g/\left(c_p r_\infty T_\infty\right)\right]^{1/3} \dot{Q}_c^{\,1/3} \left(z - z_0\right)^{-1/3} \qquad (18)$$

where;

$u_o$ = centerline velocity in the fire plume [m/s]

g = acceleration due to gravity [m/s$^2$]

$c_p$ = specific heat of ambient air at constant pressure [kJ/kgK]

$\rho_\infty$ = ambient air density [kg/m$^3$]

$T_\infty$ = ambient air temperature [K]

$Q_c$ = convective portion of heat release rate [kW]

z = vertical height [m]

$z_o$ = virtual origin [m]

The virtual origin is a mathematical artifact of the plume correlations and is a direct consequence of the assumption that the plume results from a point source of bouyancy. The location of this point source dictates the width of the resulting plume and

often is not coincident with the base of the fire. In most instances, the virtual origin is situated below the base of the fuel which causes the plume width at the base of the fire to match the diameter of the fire.

Heskestad's virtual origin equation is expressed as [15];

$$\frac{z_0}{D} = -1.02 + 0.083\frac{\dot{Q}^{2/5}}{D} \qquad (19)$$

where;

$z_o$ = virtual origin of plume point source of bouyancy [m]

D = diameter of fire [m]

Q = total heat release rate of fire [kW]

The entrainment rate of ambient air into the fire plume can be calculated from expressions developed by Heskestad. These calculations account for two regimes;

1. Entrainment rate at or below the mean flame height [15]

$$\dot{m}_{ent}(kg/s) = 0.0056\dot{Q}_c(kW)\cdot z/L \qquad (20)$$

2. Entrainment rate above the mean flame height [15]

$$\dot{m}_{ent}(kg/s) = 0.071\dot{Q}_c^{1/3}(z-z_0)^{5/3}\cdot\left[1+0.027\dot{Q}_c^{2/3}(z-z_0)^{-5/3}\right] \qquad (21)$$

The expressions above can be used to calculate the centerline temperature and velocity in the plume as well as the entrainment of ambient air into the plume. However,

the plume correlations above are valid at heights above the mean flame height. The plume correlation developed by McCaffrey is based on a 0.3 m square porous burner for a methane diffusion flame and is valid within the flaming, intermittant, and plume regions. This correlation provides centerline temperature and velocity values. The general form of the correlations are shown below [7];

The centerline velocity is expressed as;

$$\frac{u_o}{Q^{1/5}} = k\left(\frac{z}{Q^{2/5}}\right)^h \qquad (22)$$

The centerline temperature rise is written in the following manner;

$$\frac{2g\Delta T_o}{T_\infty} = \left(\frac{k}{C}\right)^2\left(\frac{z}{\dot{Q}^{2/5}}\right)^{2h-1} \qquad (23)$$

The ratio of $z/Q'^{2/5}$ determines whether the vertical location is within the flaming, intermittent, or plume region. The constants necessary for each region is shown below.

| **Region** | **$z/Q'^{2/5}$** | **k** | **η** | **C** |
|---|---|---|---|---|
| **Flame** | <0.08 | 6.8 | 1/2 | 0.9 |
| **Intermittent** | 0.08-0.2 | 1.9 | 0 | 0.9 |
| **Plume** | >0.2 | 1.1 | -1/3 | 0.9 |

**Table 3 -- Constants for McCaffrey's plume correlation.**

This information can be used to make comparisons with the results of the modeling effort.

## *3.3 Conclusions*

A framework has been established for using concepts from fire dynamics to make some general comparisons to the fire being modeled. These comparisons involve making considerations for the adiabatic flame temperature and flame height as well as the centerline temperature and velocity in the resulting fire plume.

# 4.0 TASCflow Parametric Model

The construction of a parametric combustion model using the TASCflow computational fluid dynamics code is described in this section of the report. The proper design of the parametric model required an appropriate understanding of the phenomena involved in the windshield problem. To this end, information pertaining to the nature of the windshield problem is presented. This presentation begins with observations made during vehicle fire testing involving the windshield subjected to the heat flux and fluid flow of an engine compartment fire and is followed by a discussion of the various heat transfer processes involved. These phenomena are then placed in the context of the capabilities of TASCflow's sub-models for the modeling effort. The issue of parameterization is discussed in terms of its importance to the project and how it has been implemented in the TASCflow model. The process of grid generation is described to demonstrate the need for a multi-grid approach in examining the windshield problem as well as assembling the model from the required grid objects. The application of boundary conditions is provided as the final component of constructing the multi-grid parametric combustion model.

## *4.1 About the Windshield Problem*

### 4.1.1 Observation of Windshield Subject to Engine Fire

The information presented in this section is based on observations made during a full-scale test of a vehicle subjected to an engine fire after a front impact collision at the Factory Mutual Systems Test Center in Rhode Island. Details of the test will be published separately as a report for the GM/DOT Settlement Project B.3 Test. The vehicle was a

1997 Chevrolet Camaro involved in a front impact with a 14" diameter pole at 35 mph in a testing facility. The inclination of an undamaged Camaro windshield is 17 degrees. The windshield was shattered due to stresses induced by the displacement of the windshield frame and the deployment of the passenger side airbag. The characteristic glass fragments in the windshield were approximately 0.25"x0.25" on the passenger side of the windshield with larger fragments in the area of 2"x2" towards the bottom to 4"x"6" near the top. The windshield sagged inward at two locations on the passenger side. Each depression was approximately six inches wide and extended from the bottom of the windshield up to about four inches from the top.

A mixture of typical liquids found in the engine compartment consisting of brake fluid, motor oil, anti-freeze, and gasoline was poured on the ground underneath the engine compartment and sprayed into the engine compartment. The ignition source for the engine fire was a ring shaped propane burner that was fired for 90 seconds. After about five minutes of white-gray smoke coming out of the engine compartment, flames of a height of about one foot were visible from the top of the hood. These flames were centered at the passenger side of the hood and remained straight. Within a minute the flames increased to a height of approximately 18" at a diameter of about two feet. After about two minutes of visible flames at the hood, the passenger side of the windshield began to sag inward. After three minutes of visible flames the windshield began to sag further at the passenger side with flames about four feet in height and four feet in diameter. Shortly after this change in flame height, a two-foot diameter piece of windshield material on the passenger side fell inward and was resting on the dashboard. The falling windshield piece exhibited a stretching behavior at the edges as the viscosity

of the fluid was overcome by the weight of the windshield fragment. It was observed that the plastic inter-layer was vaporizing out of the cracks in the glass and was being entrained by the flames of the engine fire. More of the windshield was falling onto the dashboard in pieces about 4 inches wide and spanning the entire height of the windshield. The windshield plastic vaporized due to the radiant energy transmitted through the windshield opening. The vapors were initially entrained by the engine fire plume as conditions necessary for the ignition of these vapors were not present. The fire spread through the bulkhead at an opening for the air conditioning system. The dashboard caught fire when the passenger-side half of the windshield was completely collapsed into the car and the radiant energy at the dashboard surface was sufficient to support combustion. The fire on the dashboard windshield fragments ignited and spread towards the passenger side. At this point a distinct change in flame height was observed as the flames grew to approximately six feet in height. The hole in the windshield grew as more fragments on the passenger side fell inward and the hole behaved as a chimney, venting out the products of combustion from the growing dashboard fire. Within a minute of the dash catching on fire the headliner of the vehicle caught on fire.

From a personal communication with infrared technicians and viewing the infrared video of the windshield after the test, some temperature data associated with the initial melting of the windshield was obtained. As the first piece of windshield fell inside the vehicle, the temperature of the outside of the windshield was between 123 and 125 C and inside windshield temperature was about 105-108 C as it hit the dashboard. The flame temperature at this point was approximately 800 C. Also from the time display of

the infrared video, the time for complete melting of the windshield was about three minutes.

### 4.1.2 Heat Transfer in the Windshield Problem

All three modes of heat transfer, radiation, convection, and radiation, are present in the windshield problem. These heat transfer processes involve specific entities in the context of a passenger vehicle windshield problem; the engine compartment fire, the three-layer windshield, and the occupant of the passenger compartment.

The engine compartment fire provides the source of convective and radiative energy that drives the heat transfer processes in the windshield problem. This turbulent diffusion flame produces both convective and radiant energy during the combustion process. The convective portion of the energy released by the fire is primarily directed upward due to bouyancy forces that arise due to the density differences in the combustion region. The air heated by the combustion reaction rises upward and is replaced by ambient air that is entrained in order to conserve mass in the combustion region. The air near the fire is heated in the horizontal direction as well, just to a lesser extent because the bouyancy forces govern the fluid motion. The horizontal heating of air near the fire can be attributed to gas phase conduction in the air as well as the absorption of radiant energy by the ambient air. Radiant energy is released radially from the fire and the intensity of this radiant energy depends upon the heat release rate of the fire and the radiation properties within the combustion zone. The heat release rate of the fire depends upon the amount of fuel combusted and the heat of combustion of the fuel. The radiation properties within the combustion zone involve the wavelength dependence of gaseous

combustion byproducts such as water vapor, carbon dioxide, and carbon monoxide, as well as the continuous contribution of soot. A portion of the radiated energy is directed to the fuel and contributes to the net surface heat flux of the fuel. This net surface heat flux converts the solid fuel into a combustible vapor form. Once sufficient oxygen has diffused into the vapor, the combustion reaction can take place.

The energy released by the fire results in a heat flux at the surface of the windshield that consists of both radiative and convective contributions. The radiative portion depends upon the intensity of radiation produced by the fire, the amount of radiant energy that is absorbed or scattered by the air space between the fire and the windshield, and the amount of radiation reflected by the windshield's surface. The convective portion depends upon the horizontal separation between the windshield and the fire. The two contributions, the convective flux and the net radiative flux, at the surface of the windshield result in a net heat flux that is drives the process of heat conduction into the solid. A portion of the incident radiation will also be absorbed by the windshield material as heat is being conducted. The heat will be conducted to the interior surface of the windshield and will provide a source of radiant and convective energy based on it's surface temperature. This results in a radiative and convective heat flux in conjunction will radiant energy that has been transmitted through the windshield layers. The aggregate heat flux travels through the air space of the vehicle's interior and interacts with surfaces such as the dashboard and the occupants of the vehicle.

## *4.2 About Passenger Vehicle Windshields*

Research was conducted in the area of passenger vehicle windshields and was used in conjunction with observations of a full-scale test in order to identify windshield sub-models for the examination of a windshield subjected to the convective and radiative heat flux of an engine fire. The process of developing passenger vehicle sub-models suitable for use in TASCflow included the investigation of the passenger vehicle windshield composition, material properties, and behavior when subjected to an engine fire. Motor vehicle regulations with respect to the flammability of glazing materials are presented at the end of the section.

### 4.2.1 Windshield Composition and Material Properties

Windshields must be resistant to abrasions from the outside and inside as well so glass is preferred because it provides good visibility and is resistant to scratches. Today's windshields are High Penetration Resistant (HPR) and were introduced in the 1968 model year. Tempered glass is used in side and rear windows because of its strength characteristics. Automotive glazing must meet MVSS 205 - Glazing Materials, and the changes made in 1983 allow glass-plastic glazing of a prescribed performance level for use in windshield, side and rear windows. Early windshield designs featured annealed glass and a cellulose nitrate inter-layer, but this inter-layer degraded and discolored with time. Polyvinyl butyral (PVB) was used for the inter-layer in the 1930s and is still used today [13].

In the 1960s Rodloff discovered a way of reducing facial lacerations by decreasing the adhesion between the glass and the PVB inter layer. This worked because

it was able to increase the head velocity necessary for penetrating the windshield [13]. This was accomplished by increasing the moisture content of the PVB by DuPont and Monsanto manufacturers and increasing the thickness of the inter layer from 0.015" to 0.030". The thickness of each glass layer was 1/8" of annealed plate or sheet glass. Further changes to the windshield design have included the use of annealed float glass, decreasing the thickness of the glass to reduce weight and improve safety performance [13].

For side and rear windows, tempered glass is used because when it breaks the glass shatters into pieces the are roughly of the height and width to the thickness of the glass. This shatter mode leaves less jagged edges. Tempered glass is thinner than and weighs less than laminated glass. Tempering is accomplished by rapidly cooling the outside surfaces of the glass with high-speed jets of air. The surface of the glass cools while the center remains hot. As the center cools it pulls the outer surfaces into compression. Automotive glass is tempered so the surface compression is at least 20,000 psi. The glass breaks when the surface goes into tension and reaches its ultimate tensile stress that is typically in excess of 30,000 psi [13].

*About Glass*

Glass is in the group of ceramics that are non-crystalline silicates that contain oxides such as CaO, $Na_2O$, $K_2O$, and $Al_2O_3$ [4]. Glass solidifies in a way that is much different than crystalline solids in that glass becomes increasingly viscous with decreasing temperature. Additionally, glass does not have a definite temperature at which it becomes a solid like crystalline solids do. This is manifested in the difference

between specific volume for crystallines and non-crystallines. Crystalline materials experience a discontinuous decrease in volume when they reach their melting point. On the other hand, volume decreases continuously with reduction and a small decrease in slope is observed at the glass transition temperature. At temperatures below glass transition temperature the material is classified as being glass. When the temperature is above this glass transition temperature, the material is considered to be a super-cooled liquid and at even higher temperatures, it becomes a liquid. The relationship between viscosity and temperature is important in glass-forming processes and this corresponds to five specific points used in glass forming [4].

- Melting point -- the temperature at which the viscosity is 10 Pa-s and is considered a liquid.
- Working point -- the temperature at which the viscosity is $10^3$ Pa-s and the glass is easily deformed.
- Softening point -- the temperature at which the viscosity is $4 \times 10^6$ Pa-s and the glass can be handled without significant alterations to the geometry.
- Annealing point -- the temperature at which the viscosity is $10^{12}$ Pa-s and the atomic diffusion is rapid enough that residual stresses are dissipated in about 15 minutes.
- Strain point -- the temperature at which the viscosity is $3 \times 10^{13}$ Pa-s and at temperatures below this, fracture will occur before plastic deformation. The glass transition temperature is above the strain point.

Automotive windshield glass is manufactured commercially by Libbey-Owens Ford, Pilkington, and Glaverbel and is classified in the soda-lime category [2]. These glasses have the following weight compositions,

| Manufacturer | $SiO_2$ | $Al_2O_3$ | $Na_2O$ | $K_2O$ | MgO | CaO | $Fe_2O_3$ | $SO_3$ |
|---|---|---|---|---|---|---|---|---|
| Libbey-Owens Ford | 72.9 | 0.1 | 13.9 | 0.0 | 4.0 | 8.6 | 0.1 | 0.3 |
| Pilkington | 72.7 | 1.0 | 13.0 | 0.6 | 3.9 | 8.3 | 0.095 | 0.22 |
| Glaverbel | 72. | 1.0 | 13.6 | 0.3 | 4.2 | 8.5 | 0.1 | 0.3 |

**Table 4 -- Weight Composition of Windshield Glass by Manufacturer**

Soda-lime glass has the following properties [**2**].

| Softening Point (C) | Annealing Point (C) | Strain Point (C) | Working Point (C) | Coefficient of Thermal Expansion ($10^{-7}$/°C) |
|---|---|---|---|---|
| 734 | 551 | 522 | 1047 | 80 |

**Table 5 -- Glass properties of tempered windshield glass.**

The thermal properties of windshield glass vary with temperature and are shown in the tables below [**2**],

| Thermal Conductivity (kcal/m hr K) | Temperature (C) |
|---|---|
| 0.70 | 450 |
| 0.9 | 850 |
| 1.2 | 1050 |
| 1.55 | 1250 |
| 2.25 | 1500 |

**Table 6 -- Thermal conductivity of tempered glass at different temperatures.**

| Density ($g/cm^3$) | Temperature (C) |
|---|---|
| 2.334 | 20 |
| 2.270 | 987 |
| 2.240 | 1249 |
| 2.220 | 1388 |

**Table 7 -- Density of tempered glass at different temperatures.**

The specific heat of tempered glass remains relatively constant from room temperature to the glass transition and has a range of values of 628 to 1256 J/kg K for soda-lime glass in general [2].

The spectral transmittance of soda-lime glass has distinct wavelength cutoffs as shown in the figure below.

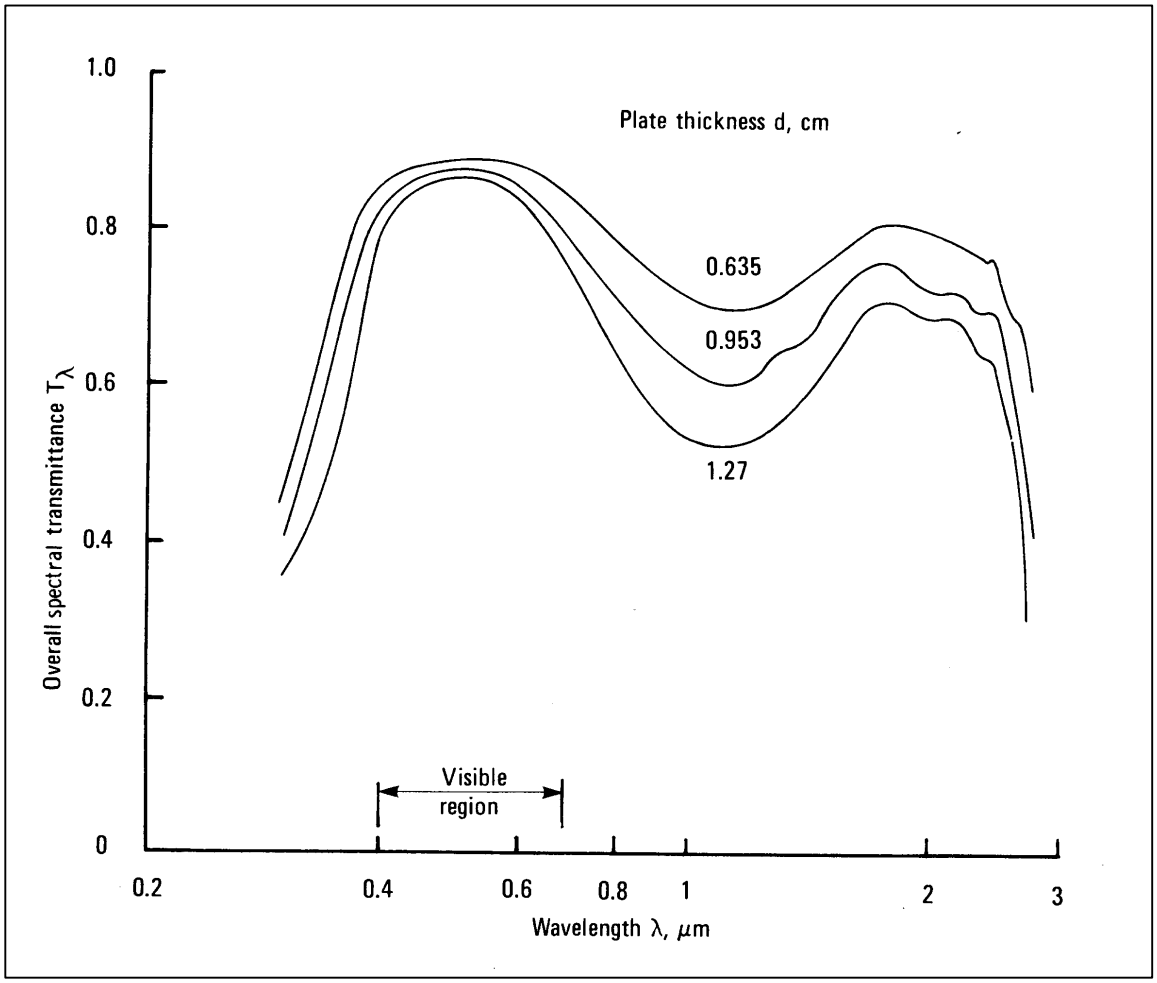


**Figure 3 -- Spectral transmittance, including surface reflection, for soda-lime glass.**

Soda-lime glass exhibits strong cutoffs in the far ultraviolet range with $\lambda = 0.17\mu m$ as well as in the near infrared with $\lambda = 2.5\mu m$. Therefore, soda-lime glass will behave as a strong emitter/absorber of radiant energy at wavelengths below $0.17\mu m$ and above $2.5\mu m$. It should be noted that the overall thickness of typical windshield glass is 0.635cm, as shown in the upper curve of Figure 3.

The spectral emission of soda-lime glass at 1000 C is shown below for thickness of 0.1, 0.3, 1, 3, and 10 cm.

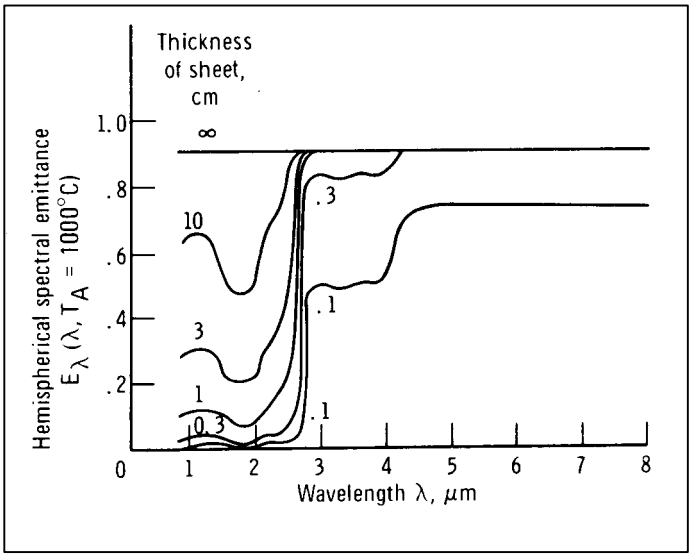


**Figure 4 -- Spectral emission of soda-lime glass at 1000 C.**

It can be seen that at the cutoff wavelength of 2.5μm that for glass thickness greater than 1cm, the emissivity approaches the limit of 0.9 at 1000 C.

*Polyvinyl Butyral (PVB)*

Material property information for polyvinyl butyral (PVB) from SFPE Handbook Table C-3 [15]

PVB: $C_8H_{14}O_2$

- MW=142.10
- $H_c$ gross = 32.90 MJ/kg
- $H_c$ net= 30.70 MJ/kg

- $H_{c'/r,o}$ = 13.00 MJ/kg
- $r_o$ = 2.365

Material Properties [**9**]

| | |
|---|---|
| Thermal Conductivity | $k = 1.731\left(0.127 - 0.000165\cdot\left(\frac{9}{5}\cdot T_c + 32\right)\right)\frac{W}{mK}$ |
| Specific Heat | $c = 4187\left(0.49 + 0.00025\cdot\left(\frac{9}{5}\cdot T_c + 32\right)\right)\frac{joule}{kgK}$ |
| Heat Transfer Coefficient | $htc = 0.5275\left(45.5 + 0.024\left(\frac{9}{5}T_c + 32\right)\right)$ |
| Thermal Diffusivity | $a = 0.000316\frac{m^2}{hr}$ at 50 C |

**Table 8 -- Material properties for polyvinyl butyral from Mark Gold at Saflex [9].**

Polyvinyl butyral has an auto-ignition temperature of 395 C, a flashpoint of 180 C, and begins to decompose around 200 C [9]. The material has a burning rate, according to ASTM D568-56T, of 3.62 seconds/in$^2$. The emissivity of polyvinyl butyral is 0.95 at 73 C. It has a melting temperature of 250 C where the viscosity decreases steadily and oxidizes at 325 C in air.

4.2.2 Burning Behavior of Plastics

Polymers undergo a thermal decomposition when subjected to fire. This decomposition process consists of the molecules in the polymer chain breaking down into smaller molecules [15]. The smaller molecules will vaporize, while the larger molecules remain in the condensed phase until sufficient energy is available for vaporization or to break it into smaller molecules. Once vaporized, oxygen diffuses into the fuel and a diffusion flame is created once the proper mixture of fuel and air is at or above it's auto-

ignition temperature [15]. Many materials will produce carbonaceous char when they undergo thermal decomposition. This will affect the further decomposition of the material because the char layer makes it more difficult for the incident heat flux to penetrate through to the unburned material [15]. At the same time, the volatiles produced by the unburned material must travel through the char layer so it is more difficult for oxygen to react with the volatiles [15]. Therefore, the charring process helps to reduce the rate of fuel supplied to a fire and is a factor in considering the firesafety aspects of plastics [15].

For thermoplastics the main physical change is from the glass state to a fluid state. If the fluid state occurs at a temperature well below the decomposition temperature, then the material will tend to flow or drip [15]. In the case of the windshield problem, the melted plastic from the windshield can fall onto the dash or onto the floor board of the passenger compartment. This now introduces a ventilation condition that increases as more windshield fragments fall into the passenger compartment.

### 4.2.3 Regulations for Windshield Glazing Materials

Motor Vehicle Safety Standard (MVSS) 205 [12] prescribes testing for glazing materials used in automotive applications. The specific areas of concern for this examination of the windshield problem are contained in sections 5.23 and 5.24 that describe the testing procedures for determining flammability of glazing materials.

Section 5.23

Section 5.23 of MVSS 205 is used to assess the rate of burning for glazing materials less than 0.050 in (1.27mm) thick. Six 12 inch by 1 inch flat samples are

submitted and marked into 1/2 inch squares. The test apparatus consists of a shield that is constructed out a non-combustible material such as sheet metal that is 12 inches wide, 12 inches deep and 30 inches high with an open top. The shield assembly is elevated so as to allow for 1 inch of ventilation height around the assembly. A viewing window is required within the shield assembly so as to allow the entire length of the test specimen to be seen by an observer. A clamp attached to the shield assembly allows the specimen to be held in a vertical fashion at the center of the shield in front of the viewing window. A drop of benzene is placed about 6mm above the bottom end of the sample so it can run down and a large drop at the bottom. Within seven seconds of application, the drop is ignited by a high-potential, low-energy spark that is consistent with one that can be generated by an automobile ignition coil or by a safety match. The time for the specimen to self extinguish or become entirely burned is recorded and the burned or charred area is estimated to the nearest 1/4 $in^2$ using the demarcations on the sample as a guide. The test results should not exceed the following guidelines;

| Thickness (in) | Thickness (mm) | Vertical Burning Rate ($in^2/s$) | Vertical Burning Rate ($mm^2/s$) |
|---|---|---|---|
| 0.005-0.010 | 0.13-0.25 | 1.0 | 645 |
| 0.011-0.015 | 0.28-0.38 | 0.50 | 323 |
| 0.015-0.050 | 0.41-1.27 | 0.25 | 161 |

**Table 9 -- MVSS 205 maximum burning rates.**

Glazing materials that are greater than 0.050 in (1.27 mm) thick are tested for flammability according to section 5.24 of MVSS 205. The testing procedure requires three specimens that measure 6 in by 0.5 in (152 mm by 13 mm). Two lines are drawn into the surface of the specimen one at 1 in (25 mm) and the other at 4 in (102 mm) from

one end. The specimen is then clamped with the longitudinal axis horizontal and the transverse axis inclined at an angle of 45 degrees with respect to the horizontal. A Bunsen burner with a 20 mesh per 25.4 mm gauze that measures about 5 $in^2$ (127 $mm^2$) is placed 1.14 in (29 mm) below the edge of the specimen. About 0.5 in (13 mm) of the sample will extend beyond the edge of the gauge. The flame used in the test is that of a Bunsen burner or alcohol lamp and is between 1/2 and 3/4 in (13 to 19 mm) in height. The flame source is adjusted so that the flame tip is just in contact with the specimen. At the end of thirty seconds the burner is removed. Time is recorded beginning with the flame front reaching the 1 inch mark and continues until the 4 inch mark is reached. In the event that the specimen self extinguishes, the burner shall be placed under the free end for another 30 seconds. If the flame still does not reach the 4 inch mark then the sample is considered self-extinguishing. The sample must not have a horizontal burning rate greater than 3.5 in/min (1.48 mm/s).

## *4.3 Selection of Phenomena to Model*

The goal of this work was to examine the heat transfer and fluid flow to a windshield subjected to an engine fire with the aid of Computational Fluid Dynamics modeling. Using the available research as a starting point, the appropriate phenomena were chosen that would be modeled using the TASCflow CFD code. The three-layer high impact resistant windshield was the focal point of the study. The hood of the vehicle is also of particular importance, as it serves as the source of the engine compartment fire and influences the nature of the local air flows. The fluid region above the engine compartment and around the windshield is also considered in the construction of the

model. The Eddy Dissipation Combustion Model is used to represent the engine fire and the design heat release rate is translated into an appropriate inlet boundary condition. The standard k-ε turbulence is employed in modeling the turbulent nature of the flow fields in this situation. The heat transfer occurring through the vehicle's windshield is accounted for in the use of Conjugate Heat Transfer Objects which consider the effect of incident radiant and convective heat fluxes on the conduction heat transfer. The Finite Volume Radiation model is used for radiation calculations in conjunction with radiation properties calculated using the narrow-band model RADCAL.

In order to model the fluid flow and heat transfer to a windshield subjected to an engine compartment fire, the appropriate components of the post-collision vehicle were identified. The geometric and nodal specifications of all regions are defined in terms of parameters that are assigned numeric values in one file. Therefore, the problem geometry or grid size can be modified in an efficient manner. Because the examination of the windshield problem involves a larger physical area, and therefore a larger computational domain, the use of symmetry is imperative in order to make predictions of sufficient accuracy at a reasonable computational time. Symmetry is used along the middle of the grid essentially "cutting" the car in half lengthwise. This technique requires nearly half of the nodes necessary to examine this problem, thus the same level of accuracy as a whole model can be obtained in half the computation time. The only limitation to the use of symmetry is that the conditions of the problem must be symmetric about the width and height of the model. However, since this work focuses on the effect of an engine fire, which can be assumed to be symmetric, this assumption was not perceived to be a severe limitation of the model.

## *4.4 Importance of Parameterization*

In order to construct a CFD model that will account for the heat flux and fluid flow from engine to passenger compartments in any arbitrary post-collision passenger vehicle fire, the issue of parameterization was considered. The CFD model developed in this study could be used as an engineering design tool for the purpose of evaluating different windshield materials and configurations. The capability of being able to model an arbitrary engine fire and an arbitrary windshield in an arbitrary post-collision vehicle becomes a significant issue when considering the use of such a design tool. Therefore, the geometric and nodal information necessary to construct a CFD model of an automotive windshield exposed to the heat flux and fluid flow of an engine compartment fire was expressed in parametric form. An input file was created containing the essential geometric and nodal information as well as all the parametric relationships needed to construct the model. The details of the grid generation process and the role of the input file are discussed in the sections that follow.

### 4.4.1 Input File

The input file used in formulating the windshield model is *input.txt* and the full text of the file appears in Appendix A.

After the header is the section titled “VARIABLE DEFINITIONS” which is read by the TASCgrid utilities and allows for variable names to be applied to various components of the geometric and nodal specification of the grid objects.

```
!************************************************
!*  This file contains the variable definitions   *
!*  for all parts of the vehicle assembly.        *
!*                                                *
!*  USED FOR GRID REFINEMENT (NODES REDEFINED)    *
!*                                                *
!*  Last modified: 28 Oct 1998                    *
!*  By: James Ierardi for General Motors Project. *
!************************************************
```

VARIABLE DEFINITIONS

The first section of the file can be edited by the user to specify geometric and nodal information for the problem of interest. A symmetry boundary condition is imposed on this problem, therefore the half-width of the engine compartment is required. The windshield object is specified by indicating the thickness of the three layers of the windshield material in addition to the thickness of the fluid regions attached to the inner and outer surfaces.

```
!*************************
!*  BEGIN USER VARIABLES  *
!*************************
!-----------------------------------------------------------------------
! Dimensions in meters
!-----------------------------------------------------------------------
! Engine
x_eng = 1.000           ! length of engine compartment
y_eng = 0.844           ! half-width of engine compartment (windshield,
etc.)

! Windshield
x_airout = 0.2          ! Thickness of air extension outside car
x_glass1 = 0.00635      ! Thickness of outer glass layer
x_pvb = 0.000762        ! Thickness of pvb inter-layer
x_glass2 = 0.00635      ! Thickness of inner glass layer
x_airin = 0.2           ! Thickness of air extension inside car
x_wind = 0.8382         ! Horizontal projection of windshield
z_wind = 0.8966         ! Vertical projection of windshield

! Fluid Region Extensions
z_aira = 0.5   !Air Above Vehicle height
y_airs = 0.5   !Air Beside Vehicle width
```

Each geometric entity has a corresponding nodal specification. The location of the fire source can also be specified. The comments beside each component of the fire

specification provide guidance as to the appropriate limits on the nodal values. This ensures that the fire location is properly specified within the physical bounds of the associated grid object.

```
!-----------------------------------------------------------------------
! Number of nodes (adjusted per readme.txt in Grid_Refine directory)
!-----------------------------------------------------------------------
nodex_front = 15  ! Eng_Front, Air_Above, and Top_Front
nodex_airout = 8  ! nodes for thickness of air extension outside car
nodex_glass1 = 4  ! nodes for thickness of outer glass layer
nodex_pvb = 4     ! nodes for thickness of pvb inter-layer
nodex_glass2 = 4  ! nodes for thickness of inner glass layer
nodex_airin = 8   ! nodes for thickness of air extension inside car
nodey_car = 15    ! Engf, Engb, Pass_BF, Pass_BB, Aireng, Wind, Pass_TB
nodey_air = 8
nodez_wind = 15   ! Aireng, Wind, Pass_TB
nodez_top = 8     ! Top_Front, Top_Mid, Top_Back
! Fire Specification
nodex_fire1_s = 8 ! Start node of FIRE1 length, should be greater than
1
nodey_fire1_e = 8 ! End node of FIRE1&FIRE2 width, should be <nodey_car
nodez_fire1_e = 9 ! End node of FIRE1 height, should be < nodez_wind
nodex_fire2_e = 4 ! End node of FIRE2 length, should be < nodex_airout
nodez_fire2_e = 9 ! End node of FIRE2 height, should be < nodez_wind
!-----------------------------------------------------------------------
```

Relationships between various nodal quantities are then established as part of the parametrization of the model.

```
! Windshield
nodey_wind = nodey_car  ! nodes for width of windshield
!Fluid Extension for Windshield
nodey_ext_sm = nodey_air

! Air Above Engine
nodex_aireng = nodex_front
nodey_aireng = nodey_car
nodez_aireng = nodez_wind
! Fluid Extension for Air Above Engine
nodey_ext_sf = nodey_air
```

Expansion factors are then assigned for the application of nodes along specific coordinate directions. The expansion factors are used to bias nodes in a certain direction, which is often done to resolve numerical instabilities associated with large gradients. A value of unity for an expansion factor results in equal spacing of all nodes along the

coordinate direction. A value of less than unity for the expansion factor results in a nodal distribution density that increases with positive linear coordinate direction. An expansion factor greater than unity results in a nodal distribution that decreases in density along a positive linear coordinate direction.

```
!-----------------------------------------------------------------------
! Expansion factors
!-----------------------------------------------------------------------

! Windshield
r_x_airout = .25  ! expansion factor for outside air nodes in x dir.
r_x_glass1 = 1.   ! expansion factor for outside glass nodes in x dir.
r_x_pvb     = 1.  ! expansion factor for pvb nodes in x dir.
r_x_glass2 = 1.   ! expansion factor for inside glass nodes in x dir.
r_x_airin = 4.    ! expansion factor for inside air nodes in x dir.
r_y_wind = 1.0
r_z_wind = 1.0    ! expansion factor for windshield nodes in z dir.
! Fluid extension
r_y_ext_sm = 1.

! Air above windshield
r_x_aireng = 1.
r_y_aireng = 1.
r_z_aireng = 1.
! Fluid Extension of Air Above Windshield
r_x_ext_sf = 1.
r_y_ext_sf = 1.
r_z_ext_sf = 1.

! Fluid Extenion Top_Front
r_x_air_tf = 1
r_y_air_tf = 1
r_z_air_tf = 1

! Fluid Extenion Top_Mid
r_x_air_tm = 1
r_y_air_tm = 1
r_z_air_tm = 1
```

The starting points defined below establish the location of the origin for the construction of each grid object. These points are defined with respect to the west south bottom (wsb) point for each object because the coordinate system is defined as west to east for the x-direction, south to north for the y-direction, and bottom to top for the z-direction.

```
!-----------------------------------------------------------------------
! Starting points
!-----------------------------------------------------------------------

! Windshield
wind_wsb_x = 0.
wind_wsb_y = 0.
wind_wsb_z = 0.

! Air Above Engine
aireng_wsb_x = 0.
aireng_wsb_y = 0.
aireng_wsb_z = 0.

! Fluid Extension Top_Front
air_tf_wsb_x = 0.
air_tf_wsb_y = 0.
air_tf_wsb_z = 0.

! Fluid Extension Top_Mid
air_tm_wsb_x = 0.
air_tm_wsb_y = 0.
air_tm_wsb_z = 0.
```

The variables that can be defined by the user have been presented in the sections above. Following this section are variable names that are based upon geometric and nodal relationships to construct a windshield model based on the input specified by the user in the sections above. The values below are influenced by the information specified in the above sections and are parametric in nature.

```
!************************
!*  END USER VARIABLES  *
!************************
!-----------------------------------------------------------------------
! Geometry
!-----------------------------------------------------------------------

! Fluid Extension Top_Front
x_air_tf = x_eng - x_airout + x_wind
y_air_tf = y_eng + y_airs

! Fluid Extension Top_Mid
x_air_tm = x_airout + x_glass1 + x_pvb + x_glass2 + x_airin
y_air_tm = y_eng + y_airs
```

Based on the geometry of the specific grid object, the start and end points are defined. Therefore, a more elegant .gdf file can be constructed and all nodal relationships can be changed from the input file only.

```
!-----------------------------------------------------------------------
! Start and End Point Definitions
!-----------------------------------------------------------------------

! Windshield
x_airout_sb = wind_wsb_x      ! starting and end points along bottom of
x_glass1_sb = wind_wsb_x + x_airout        ! the windshield
x_pvb_sb = wind_wsb_x + x_airout + x_glass1
x_pvb_eb = wind_wsb_x + x_airout + x_glass1 + x_pvb
x_glass2_eb = wind_wsb_x + x_airout + x_glass1 + x_pvb + x_glass2
x_airin_eb = wind_wsb_x + x_airout + x_glass1 + x_pvb + x_glass2 +
x_airin

x_airout_st = wind_wsb_x + x_wind   ! starting and end points along top
x_glass1_st = wind_wsb_x + x_airout + x_wind       ! of the windshield
x_pvb_st = wind_wsb_x + x_airout + x_glass1 + x_wind
x_pvb_et = wind_wsb_x + x_airout + x_glass1 + x_pvb + x_wind
x_glass2_et = wind_wsb_x + x_airout + x_glass1 + x_pvb + x_glass2 +
x_wind
x_airin_et = wind_wsb_x + x_airout + x_glass1 + x_pvb + x_glass2 +
x_airin + x_wind

y_wind_s = wind_wsb_y
y_wind_e = wind_wsb_y + y_eng
z_wind_s = wind_wsb_z
z_wind_e = wind_wsb_z + z_wind

! Air above engine compartment
x_aireng_sb = aireng_wsb_x
x_aireng_eb = aireng_wsb_x + x_eng - x_airout
x_aireng_st = aireng_wsb_x
x_aireng_et = aireng_wsb_x + x_eng - x_airout + x_wind
y_aireng_s = aireng_wsb_y
y_aireng_e = aireng_wsb_y + y_eng
z_aireng_s = aireng_wsb_z
z_aireng_e = aireng_wsb_z + z_wind
! Fluid Extension

! Fluid Extension Top_Front
x_air_tf_s = air_tf_wsb_x
x_air_tf_e = air_tf_wsb_x + x_air_tf
y_air_tf_s = air_tf_wsb_y
y_air_tf_e = air_tf_wsb_y + y_air_tf
z_air_tf_s = air_tf_wsb_z
z_air_tf_e = air_tf_wsb_z + z_aira

! Fluid Extension Top_Mid
x_air_tm_s = air_tm_wsb_x
```

```
x_air_tm_e = air_tm_wsb_x + x_air_tm
y_air_tm_s = air_tm_wsb_y
y_air_tm_e = air_tm_wsb_y + y_air_tm
z_air_tm_s = air_tm_wsb_z
z_air_tm_e = air_tm_wsb_z + z_aira
```

The total node definitions provide a reference point for defining nodal locations in the definition files for each grid object for the purpose of specifying nodal distributions and blocking off objects. These are used in the .cdf files to define the maximum I, J, and K values for the specific grid object.

```
!-----------------------------------------------------------------------
! Total node definitions for length, width, and height
!-----------------------------------------------------------------------

! Windshield nodes
nlw1 = nodex_airout
nlw2 = nodex_airout + nodex_glass1
nlw3 = nodex_airout + nodex_glass1 + nodex_pvb
nlw4 = nodex_airout + nodex_glass1 + nodex_pvb + nodex_glass2
nlw5 = nodex_airout + nodex_glass1 + nodex_pvb + nodex_glass2 +
nodex_airin
nww1 = nodey_car
nww2 = nodey_car + nodey_air
nhw1 = nodez_wind

! Air Above Engine nodes
nlae1 = nodex_front
nwae1 = nodey_car
nwae2 = nodey_car + nodey_air
nhae1 = nodez_wind

! Fluid Extension Top_Front
nlatf1 = nodex_front
nwatf1 = nodey_car + nodey_air
nhatf1 = nodez_top

! Fluid Extension Top_Mid
nlatm1 = nodex_airout + nodex_glass1 + nodex_pvb + nodex_glass2 +
nodex_airin
nwatm1 = nodey_car + nodey_air
nhatm1 = nodez_top

! Fire Specification
nlf1 = nodex_fire1_s
nwf1 = nodey_fire1_e
nhf1 = nodez_fire1_e
nlf2 = nodex_fire2_e
nhf2 = nodez_fire2_e

!End
```

## *4.5 Grid Generation*

The process of grid generation allows the user to specify the geometry of the physical domain as well as the number and distribution of nodes in the computational domain. Grid generation can be accomplished with various commercially available software packages such as PowerMesh, however, the grids generated for the windshield problem were done using the command line-based TASCgrid grid generation utility included with TASCflow. This utility consists of four parts; the geometry phase (tascgridg), the curve phase (tascgridc), the surface phase (tascgrids), and the interior phase (tascgridi). Each phase requires user specified definition files that are explained in greater detail in the following sections. An input file, *input.txt*, is used to govern the relationships that exist between geometric and nodal specifications in the construction of this parametric model. The details of the input file are shown in the following section. The construction of each grid object is then described using the object for above the engine compartment as an example with appropriate references to the input file.

The computational domain for the windshield problem consists of the windshield, the fluid region above the engine compartment, and two fluid regions that attach above the windshield and above the engine compartment to better capture the fluid flow. The grid creation process is described in detail below using the creation of the Above_Eng (fluid region above the engine compartment) grid object and the role of the various definition files and TASCgrid utilities used in constructing it. All grid objects were created within their own subdirectories, where the geometry definition file (.gdf), curve definition file (.cdf), surface definition file (.sdf) and interior definition file (.idf) were located. The problem name must first be established so the TASCgrid utilities can

properly label the files created during the grid generation process. The *lun* command is executed in batch mode for naming files in the grid generation process.

```
>lun above_eng -b
```

The lun command is issued using the prefix “above_eng” in conjunction with file extensions and the –b flag executes the command in batch mode. This creates a file *name.lun* within the problem directory, and the text of this file is shown below.

```
problem above_eng.
include #/batch.lun
include #/systemu.lun
```

4.5.1 Geometry Phase (Tascgridg)

The geometry phase requires a geometry definition file (*gdf*) that is used as input for *tascgridg*. The *gdf* file used for the “Above_Eng” grid object is called *above_eng.gdf* and is shown in detail below.

Following the header is the *include* statement which reads the input.txt file and stores the defined variables in memory.

```
!*************************************************
!* Air above engine compartment template.        *
!*                                               *
!* Last modified: 2 Feb 1998                     *
!* By: James Ierardi for the General Motors Project. *
!*************************************************

include ../input.txt
```

The geometric location of the points used in constructing the “Above_Eng” grid object are defined in three dimensional space according to the variables defined in the *input.txt* file. The *aireng* objects refer to the points associated with the fluid region directly above the hood of the engine compartment and the *ext_sf* objects refer to the

points associated with the fluid extension beside the engine compartment. This fluid extension region is established to capture the flow field beside the vehicle and to resolve numerical instabilities that occur due to temperature, pressure, and velocity gradients directly over the engine due to the fire and subsequent thermal plume.

```
POINT
aireng_wsb (x_aireng_sb,y_aireng_s,z_aireng_s)
aireng_esb (x_aireng_eb,y_aireng_s,z_aireng_s)
aireng_wst (x_aireng_st,y_aireng_s,z_aireng_e)
aireng_est (x_aireng_et,y_aireng_s,z_aireng_e)
aireng_wnb (x_aireng_sb,y_aireng_e,z_aireng_s)
aireng_enb (x_aireng_eb,y_aireng_e,z_aireng_s)
aireng_wnt (x_aireng_st,y_aireng_e,z_aireng_e)
aireng_ent (x_aireng_et,y_aireng_e,z_aireng_e)
ext_sf_wnb (x_aireng_sb,y_aireng_e+y_airs,z_aireng_s)
ext_sf_enb (x_aireng_eb,y_aireng_e+y_airs,z_aireng_s)
ext_sf_wnt (x_aireng_st,y_aireng_e+y_airs,z_aireng_e)
ext_sf_ent (x_aireng_et,y_aireng_e+y_airs,z_aireng_e)
```

A series of linear curves are then constructed between the points established above. This facilitates the distribution of nodes used in the TASCgrid utilities. The first entry constructs a linear curve called *aireng_sb*, which designates the curve as being on the south bottom of the object. The curve begins at *aireng_wsb* and moves along the x-direction to *aireng_esb* along the south bottom of the object. The remaining curves are constructed in a similar fashion.

```
Curve aireng_sb Linear
aireng_wsb; aireng_esb

Curve aireng_nb Linear
aireng_wnb; aireng_enb

Curve ext_sf_nb Linear
ext_sf_wnb; ext_sf_enb

Curve aireng_st Linear
aireng_wst; aireng_est

Curve aireng_nt Linear
aireng_wnt; aireng_ent

Curve ext_sf_nt Linear
```

```
ext_sf_wnt; ext_sf_ent

Curve aireng_wb Linear
aireng_wsb; aireng_wnb; ext_sf_wnb

Curve aireng_eb Linear
aireng_esb; aireng_enb; ext_sf_enb

Curve aireng_wt Linear
aireng_wst; aireng_wnt; ext_sf_wnt

Curve aireng_et Linear
aireng_est; aireng_ent; ext_sf_ent

Curve aireng_ws Linear
aireng_wsb; aireng_wst

Curve aireng_es Linear
aireng_esb; aireng_est

Curve aireng_wn Linear
aireng_wnb; aireng_wnt

Curve ext_sf_wn Linear
ext_sf_wnb; ext_sf_wnt

Curve aireng_en Linear
aireng_enb; aireng_ent

Curve ext_sf_en Linear
ext_sf_enb; ext_sf_ent
```

The curves defined above are then used to construct surfaces along the object. The convention used utilizes a positive clockwise convention, therefore, negative signs are used on certain curves to establish the surface. The first entry below constructs the south surface of the “Above_Eng” grid object called “Aireng_S”. It is constructed by connecting the south bottom curve, to the east south curve, to the south top (but in reverse direction), to the west south (again, in reverse direction) in order to form a closed curve.

```
! South Surface of whole thing
Surface Aireng_S By Curves Bilinear
aireng_sb; aireng_es; -aireng_st; -aireng_ws

! North Surface of whole thing
Surface Aireng_N By Curves Bilinear
ext_sf_nb; ext_sf_en; -ext_sf_nt; -ext_sf_wn
```

```
! Bottom Surface of whole thing
Surface Aireng_B By Curves Bilinear
aireng_sb; aireng_eb; -ext_sf_nb; -aireng_wb

! Top Surface of whole thing
Surface Aireng_T By Curves Bilinear
aireng_st; aireng_et; -ext_sf_nt; -aireng_wt

! West Surface of whole thing
Surface Aireng_W By Curves Bilinear
aireng_wb; ext_sf_wn; -aireng_wt; -aireng_ws

! East Surface of whole thing
Surface Aireng_E By Curves Bilinear
aireng_eb; ext_sf_en; -aireng_et; -aireng_es

! End
```

Based on the above geometry, points in three-dimensional space are defined in order to describe the region of fluid above the engine compartment. This geometry follows a convention of describing three-dimensional space in terms of west to east for movements in the positive x-direction, south to north for movements in the positive y-direction, and bottom to top for movements in the positive z-direction. The geometric points associated with the corners of the fluid region can be found under the heading "POINT" in the *.gdf* file. The *.gdf* file also contains instructions to construct linear curves between defined points and surfaces from the linear curves. This information is used in the curve and surface phases of the TASCgrid utility and will be explained in greater detail in the appropriate sections. Once the necessary information about the problem geometry has been specified, then *tascgridg* can be executed. The geometry for the problem is processed and the curve phase can begin.

The above file is read into the TASCgridg utility for defining the geometry of the object. A sample session is shown below with comments.

```
 +--------------------------------------------------------------------
 |                 CFX - T A S C g r i d    Version 2.7.4-51
 |
 | Copyright 1997 Advanced Scientific Computing Ltd. All rights
reserved.
 +--------------------------------------------------------------------


                   GEOMETRY PHASE  --  Geometry Definition
                   =======================================


 Reading GDF (Geometry Definition) file: above_eng.gdf

 Processing geometry.

 Processing connectivity.

 Processing vertices.
```

The *above_eng.gdf* file is read in by TASCgridg that references the *input.txt* file as shown in the section above. The *input.txt* file establishes the variables used in the definition files of the "Above_Eng" object as well as the other grid objects. The geometry is processed by establishing points, constructing curves between the points, and defining vertices along the curves.

```
   TASCgrid Geometry Phase : TOP-LEVEL MENU

    EDIT - Edit input file
       H - Help, print this menu
       P - Enter graphics environment and Plot immediately
       Q - Quit program
       R - Reprocess input files
       V - Enter graphics environment (Viewing)
       W - Write output file(s)
     G/X - Write output file(s) and quit program
```

The "X" option is chosen from the menu to write the internal geometry template (.igt) and curve distribution template (.cdt) files.

```
TASCgrid{Geometry Phase}: x

Write IGT (Internal Geometry) file ({Y}/N) ? y
 Writing unformatted IGT file: above_eng.igt

Write CDT (Curve Distribution Template) file ({Y}/N) ? y
 Writing CDT file: above_eng.cdt
```

4.5.2 Curve Phase (Tascgridc)

The curve phase requires a curve definition file (*cdf*) that is used as input for *tascgridc*. The *cdf* file used for the "Above_Eng" grid object in the windshield problem is called *air_eng.cdf* and is shown in detail below. This file defines the number of nodes required in the computational domain under the heading "Grid Dimensions" in the *cdf* file which correspond to the maximum node values in each coordinate direction. For example, the "Above_Eng" object has two values in the y-direction; the width of the engine compartment and the width of the fluid extension beside the engine compartment. Therefore, the maximum node value in the J direction would be the sum of the nodes in the y-direction of the engine compartment and the nodes in the y-direction of the fluid extension beside the engine compartment. This sum was defined in the constructing the variable *nwae2* in the *input.txt* file.

```
Grid Dimensions
ID=nlae1
JD=nwae2
KD=nhae1
-------------------------------------------------
Variable Definitions
-------------------------------------------------
```

The geometric points defined in the *.gdf* file are then given a nodal coordinate location in three-dimensional space using variables defined in the *input.txt* file.

```
Vertex Attachments
aireng_wsb  (1,1,1)
aireng_esb  (nlae1,1,1)
aireng_wnb  (1,nwae1,1)
aireng_enb  (nlae1,nwae1,1)
aireng_wst  (1,1,nhae1)
aireng_est  (nlae1,1,nhae1)
aireng_wnt  (1,nwae1,nhae1)
aireng_ent  (nlae1,nwae1,nhae1)
ext_sf_wnb  (1,nwae2,1)
ext_sf_enb  (nlae1,nwae2,1)
ext_sf_wnt  (1,nwae2,nhae1)
ext_sf_ent  (nlae1,nwae2,nhae1)
-------------------------------------------------
```

After assigning nodal coordinates to the vertices of the grid object, the number and density of nodes distributed along each curve can be defined. This is accomplished in the "Node Distributions" section of the *.cdf* file where start and end vertices are defined and the associated expansion factor, from the *input.txt* file, is given.

```
Node Distributions
Default=1
!start-vertex     end-vertex  expansion-factor
aireng_wsb  aireng_esb  100 @ r r_x_aireng
aireng_wnb  aireng_enb  100 @ r r_x_aireng
aireng_wst  aireng_est  100 @ r r_x_aireng
aireng_wnt  aireng_ent  100 @ r r_x_aireng
aireng_wsb  aireng_wnb  100 @ r r_y_aireng
aireng_esb  aireng_enb  100 @ r r_y_aireng
aireng_wst  aireng_wnt  100 @ r r_y_aireng
aireng_est  aireng_ent  100 @ r r_y_aireng
aireng_wsb  aireng_wst  100 @ r r_z_aireng
aireng_esb  aireng_est  100 @ r r_z_aireng
aireng_wnb  aireng_wnt  100 @ r r_z_aireng
aireng_enb  aireng_ent  100 @ r r_z_aireng
ext_sf_wnb  ext_sf_enb  100 @ r r_x_ext_sf
ext_sf_wnt  ext_sf_ent  100 @ r r_x_ext_sf
aireng_wnb  ext_sf_wnb  100 @ r r_y_ext_sf
aireng_enb  ext_sf_enb  100 @ r r_y_ext_sf
aireng_wnt  ext_sf_wnt  100 @ r r_y_ext_sf
aireng_ent  ext_sf_ent  100 @ r r_y_ext_sf
ext_sf_wnb  ext_sf_wnt  100 @ r r_z_ext_sf
ext_sf_enb  ext_sf_ent  100 @ r r_z_ext_sf
! End
```

The tascgridc utility is then executed using the *above_eng.cdf* file as input. A sample session is shown below.

```
 +--------------------------------------------------------------------
 |                 CFX - T A S C g r i d    Version 2.7.4-51
 |
 | Copyright 1997 Advanced Scientific Computing Ltd. All rights
reserved.
 +--------------------------------------------------------------------


              CURVE PHASE  --  Distribution of Nodes on Curves
              ================================================


 Reading unformatted IGT (Internal Geometry) file: above_eng.igt

 Reading CDF (Curve Distribution) file: above_eng.cdf
```

```
 Assigning nodes at vertices.

 Distributing nodes on curves.
```

The *above_eng*.igt and *above_eng.cdf* files are read into the TASCgridc utility and nodes are assigned to vertices and distributed along curves according to the number of nodes and expansion factors specified.

```
   TASCgrid Curve Phase : TOP-LEVEL MENU

    EDIT - Edit input file
       H - Help, print this menu
       P - Enter graphics environment and Plot immediately
       Q - Quit program
       R - Reprocess input files
       V - Enter graphics environment (Viewing)
       W - Write output file(s)
     G/X - Write output file(s) and quit program
```

The “X” option is selected to create the curve node distribution (.cnd) and surface distribution template (.sdt) for the problem.

```
TASCgrid{Curve Phase}: x

Write CND (Curve Node Distribution) file ({Y}/N) ? y
 Writing unformatted CND file: above_eng.cnd

Write SDT (Surface Distribution Template) file ({Y}/N) ? y
 Writing SDT file: above_eng.sdt
```

### 4.5.3 Surface Phase (Tascgrids)

The surface phase requires a surface definition file (sdf) that is used as input for tascgrids. The sdf file used for the “Above_Eng” grid object is called air_eng.sdf and is shown below. This file describes the surfaces that comprise the computational domain, distributes nodes along each surface, and defines the control volume for the problem. The south, north, bottom, top, west, and east regions are defined for the object.

```
Region Aireng_S
------------------------------------------------------------------
Region Aireng_N
------------------------------------------------------------------
Region Aireng_B
------------------------------------------------------------------
Region Aireng_T
------------------------------------------------------------------
Region Aireng_W
------------------------------------------------------------------
Region Aireng_E
------------------------------------------------------------------
```

The tascgrids utility is then executed using the internal geometry template (.igt), curve node distribution (.cnd), and surface definition (sdf) files. The overall grid dimensions are established and the regions specified in the *.sdf* file are then processed via interpolation by parametrically straight lines. In some instances of the interpolation, nodes along the edges of certain surfaces may have already been assigned in a previous surface with a common edge to the current surface. In these instances the program notes the nodes in question have already been assigned and subsequently will not be moved.

```
 +--------------------------------------------------------------------
 |                 CFX - T A S C g r i d    Version 2.7.4-51
 |
 | Copyright 1997 Advanced Scientific Computing Ltd. All rights
reserved.
 +--------------------------------------------------------------------


              SURFACE PHASE  --  Surface Node Interpolation
              ============================================


 Reading unformatted IGT (Internal Geometry) file: above_eng.igt

 Reading unformatted CND (Curve Node Distribution) file: above_eng.cnd

 Grid dimensions:  (ID,JD,KD)=(  15,  23,  15).

 Reading SDF (Surface Distribution) file: above_eng.sdf

 Processing region #   1: AIRENG_S
   Interpolation by parametrically straight lines

 Processing region #   2: AIRENG_N
```

```
   Interpolation by parametrically straight lines

 Processing region #   3: AIRENG_B

 NOTICE: Nodes that are already assigned have been found in the
interior
   of this region.  These nodes will not be moved.
   Nodes in the range  [2:14,15,1]  already assigned.
   Interpolation by parametrically straight lines

 Processing region #   4: AIRENG_T

 NOTICE: Nodes that are already assigned have been found in the
interior
   of this region.  These nodes will not be moved.
   Nodes in the range  [2:14,15,15]  already assigned.
   Interpolation by parametrically straight lines

 Processing region #   5: AIRENG_W

 NOTICE: Nodes that are already assigned have been found in the
interior
   of this region.  These nodes will not be moved.
   Node  [1,15,2]  already assigned.
   Node  [1,15,3]  already assigned.
   Node  [1,15,4]  already assigned.
   Node  [1,15,5]  already assigned.
   Node  [1,15,6]  already assigned.
   Node  [1,15,7]  already assigned.
   Node  [1,15,8]  already assigned.
   Node  [1,15,9]  already assigned.
   Node  [1,15,10]  already assigned.
   Node  [1,15,11]  already assigned.
   Node  [1,15,12]  already assigned.
   Node  [1,15,13]  already assigned.
   Node  [1,15,14]  already assigned.
   Interpolation by parametrically straight lines

 Processing region #   6: AIRENG_E

 NOTICE: Nodes that are already assigned have been found in the
interior
   of this region.  These nodes will not be moved.
   Node  [15,15,2]  already assigned.
   Node  [15,15,3]  already assigned.
   Node  [15,15,4]  already assigned.
   Node  [15,15,5]  already assigned.
   Node  [15,15,6]  already assigned.
   Node  [15,15,7]  already assigned.
   Node  [15,15,8]  already assigned.
   Node  [15,15,9]  already assigned.
   Node  [15,15,10]  already assigned.
   Node  [15,15,11]  already assigned.
   Node  [15,15,12]  already assigned.
   Node  [15,15,13]  already assigned.
   Node  [15,15,14]  already assigned.
   Interpolation by parametrically straight lines
```

```
   TASCgrid Surface Phase : TOP-LEVEL MENU

    EDIT - Edit input file
       H - Help, print this menu
       P - Enter graphics environment and Plot immediately
       Q - Quit program
       R - Reprocess input files
       V - Enter graphics environment (Viewing)
       W - Write output file(s)
     G/X - Write output file(s) and quit program
```

The “X” option is selected and the surface node distribution (.snd) file is written.

```
TASCgrid{Surface Phase}: x

Write SND (Surface Node Distribution) file ({Y}/N) ? y
 Writing unformatted SND file: above_eng.snd
```

4.5.4 Interior Phase (Tascgridi)

The interior phase requires an interior definition file (*idf*) that is used as input for *tascgridi*. The *idf* file for the “Above_Eng” grid object is called *air_eng.idf* and shown below. This file defines the west south bottom and east north top corners of the computational domain and nodes are distributed to the interior regions. A maximum number of iterations of 320 is required to generate the object in this example via elliptic interpolation and a convergence criteria of is established for the amount of error that can be allowed in the x,y, and z directions. The final grid file is generated with the proper geometry and nodal distributions.

```
!  Engine Compartment Template
!  Jay Ierardi
Region aireng elliptic
errxyz=4e-7
itmax=320
aireng_wsb,ext_sf_ent
```

The TASCgridi utility is the used which reads in the internal geometry (.igt), surface node distribution (.snd), and the interior definition (.idf) files. The region is processed with successive iterations of elliptic interpolation using an initial guess from

algebraic interpolation. The iterations continue until the region attains the specified convergence criteria or the interpolation is unsuccessful at the end of the designated maximum number of iterations.

```
 +-------------------------------------------------------------------
 |                 CFX - T A S C g r i d    Version 2.7.4-51
 |
 | Copyright 1997 Advanced Scientific Computing Ltd. All rights
reserved.
 +-------------------------------------------------------------------


            INTERIOR PHASE  --  Interpolation of Interior Regions
            ====================================================


 Reading unformatted IGT (Internal Geometry) file: above_eng.igt

 Reading unformatted SND (Surface Node Distribution) file:
above_eng.snd

 Grid dimensions:  (ID,JD,KD)=(  15,  23,  15).

 Reading IDF (Interior Distribution) file: above_eng.idf

 Processing region #   1: AIRENG
   Elliptic interpolation
   Using algebraic interpolation as initial guess.
   Semi-isogeometric interpolation
   Control functions by Semi-isogeometric interpolation
 Iter   Max residuals -- (I,J,K) location         RMS residuals
cpu time
  1  6.64E-07 6.58E-07 7.06E-07 12  2 14  1.76E-09 1.89E-09 2.16E-09
0.14
  2  2.37E-07 2.51E-07 2.25E-07 12 21  6  6.80E-10 8.94E-10 8.73E-10
0.21
 Converged to a tolerance of  4.00E-07 in this region after    2
iterations

   TASCgrid Interior Phase : TOP-LEVEL MENU

    EDIT - Edit input file
       H - Help, print this menu
       P - Enter graphics environment and Plot immediately
       Q - Quit program
       R - Reprocess input files
       V - Enter graphics environment (Viewing)
       W - Write output file(s)
     G/X - Write output file(s) and quit program
```

The “X” option is selected and the grid (.grd) file is written for the object.

```
TASCgrid{Interior Phase}: x

Write GRD (Grid) file ({Y}/N) ? y
 Writing unformatted GRD file: above_eng.grd
```

This process was used in constructing the “Fire”, “Above_Fire”, “Top_Front”, “Top_Mid”, and “Wind” grid objects from their respective definition files. In order to facilitate a more efficient formulation of the grid objects, a program was written in the C programming language to automate this process. The program is called Quickgrid and the development is described below.

The CFD model consists of six separate grid objects connected by grid attaching. The central grid object is the windshield. The windshield object consists of three solid objects that are specified within a fluid region. Fluid regions exist at the inner and outer surfaces of the windshield. These fluid regions which are included in the windshield grid object allow for nodes to be biased towards the windshield surfaces, where the largest temperature and velocity gradients are expected for the windshield object. The biasing of these nodes towards the windshield surface helps to capture the large gradients that exist at these surfaces. A grid object representing the hood of the vehicle and the fluid region above it was created with the intention of specifying the fire at the surface of hood and being able to capture the combustion reaction

Representative samples of the grid objects are shown in the figures that follow, to provide a visual representation of the components required for the multi-grid approach to the windshield problem.

First is the “Wind” object, representing the 3-layer passenger vehicle windshield with fluid extensions at the outer and inner surfaces. Although it cannot be demonstrated with this particular illustration of the windshield grid object, the width of the object

contains the half-width of a passenger vehicle windshield and a fluid extension that has been included for the purpose of capturing the fluid flow around the side of the vehicle. This feature will become apparent when the boundary conditions are illustrated in the next section. The "Wind" object is able to assume arbitrary windshield geometries, and since it is the focal point of the study, other fluid regions must accommodate the "Wind" object.

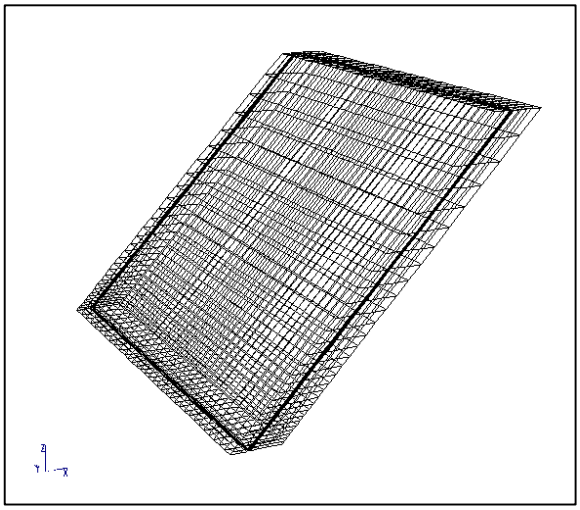

**Figure 5 -- Isometric view of the "Wind" grid object. The grid dimensions are 15x36x32.**

The outer fluid region of the "Wind" object (the yz-plane in the foreground) requires a wedge-shaped object that will bridge the angular nature of the windshield grid object to a rectangular grid object more suitable for the vertical flow fields anticipated by the combustion source. This wedge-shaped grid object is called "Above_Eng" and represents the fluid region above the hood of the vehicle between the "Wind" object and the fluid region where the engine fire will be specified.

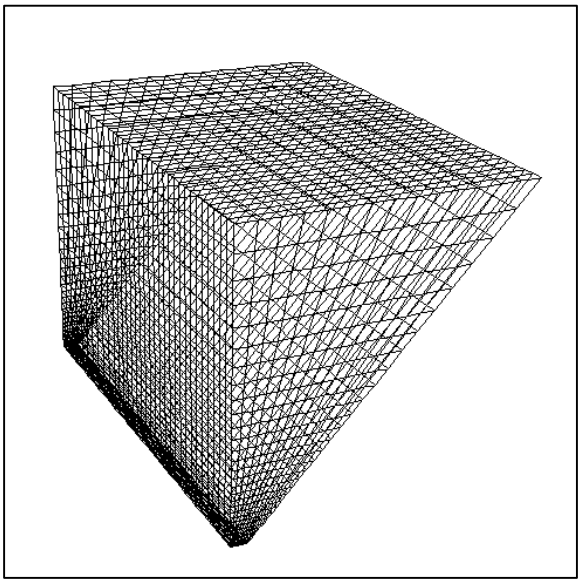

**Figure 6 – "Above_Eng" grid object that represents the airspace above the hood between the windshield and fire objects. The grid dimensions are 16x36x32.**

The yz-plane of the "Above_Eng" grid object attaches to the "Fire" grid object. The "Fire" object represents the fluid region above the hood of the vehicle where the combustion reaction takes place within a portion of the object. The illustration of the inlet boundary condition in the section that follows will provide a clearer representation.

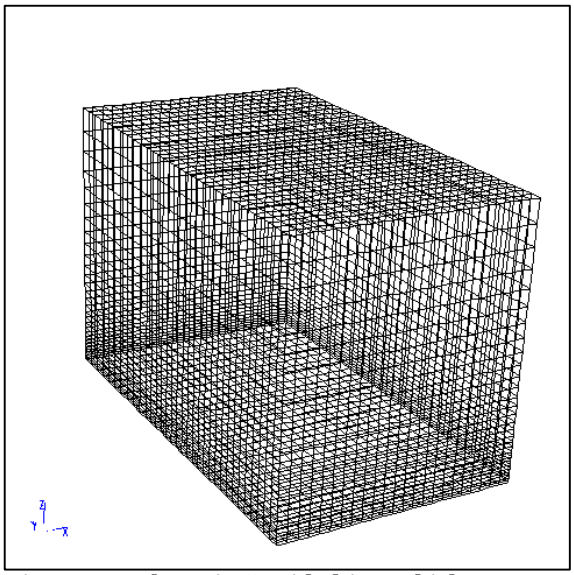

**Figure 7 -- The "Fire" grid object which represents the airspace above the hood and contains the combustion region. The grid dimensions are 24x36x32.**

In order to properly capture the fluid flow occurring at locations above the plane of the top of the windshield, the computational domain was extended by stacking an additional grid object on top of the three primary grid objects; “Wind”, “Above_Eng”, and “Fire”. Each object used for this vertical extension of the fluid region has the same xy-plane dimensions and node numbers as the primary grid object. The secondary object is “Above_Fire” which has the same xy-plane dimensions and number of nodes as the “Fire” primary object. However, this provides a vertical extension of the fluid region in order to capture the extent of the combustion reaction of the engine fire and the resulting fire plume.

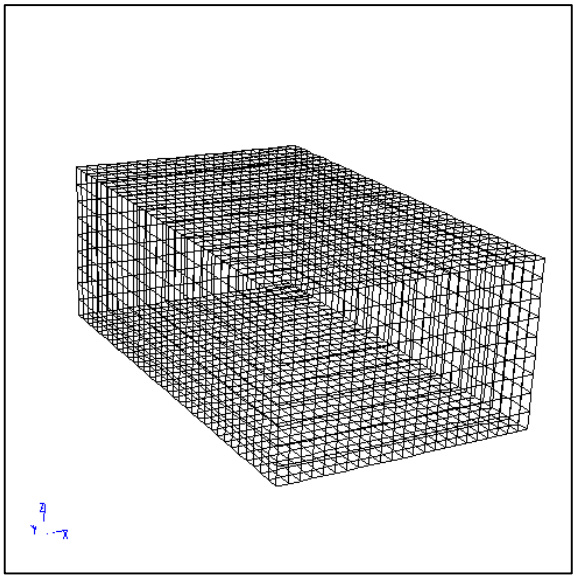

**Figure 8 – “Above_Fire” grid object representing the fluid region above the “Fire” object. The grid dimensions are 24x36x12.**

The next secondary object is “Top_Front”, which is set on top of the “Above_Eng” primary grid object. It has the same xy-plane dimensions and number of nodes as the “Above_Eng” object and extends the fluid region vertically to capture the effects of the fire plume as it rises upward and expands. The yz-plane faces of the

"Above_Fire" and "Top_Front" objects are attached to one another as the multi-grid assembly was conducted.

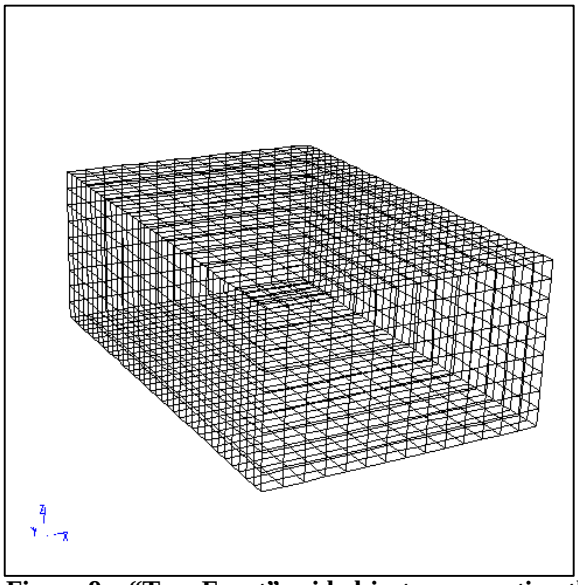

**Figure 9 – "Top_Front" grid object representing the fluid region above the "Above_Eng" object. The grid dimensions are 16x36x12.**

The vertical fluid extension for the windshield grid object is called "Top_Mid" and, like the other secondary grid objects, has the same xy-plane dimensions and number of nodes as the primary grid object "Wind". This allows for the fluid flow occurring along the outer face of the windshield to be properly captured during the modeling effort.

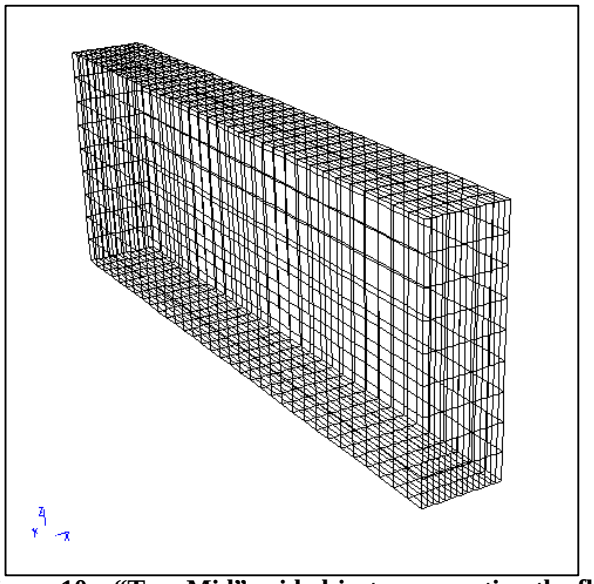

**Figure 10—"Top_Mid" grid object representing the fluid region above the windshield grid object. The grid dimensions are 15x36x12.**

The TASCbob3d utility was used to conduct the grid attachment operations between the six grid objects required for the windshield problem. The multi-grid assembly is shown in the figure below and illustrates the fluid extension that has been included beside the vehicle as well as the conjugate heat transfer object that has been blocked off in the domain for the windshield.

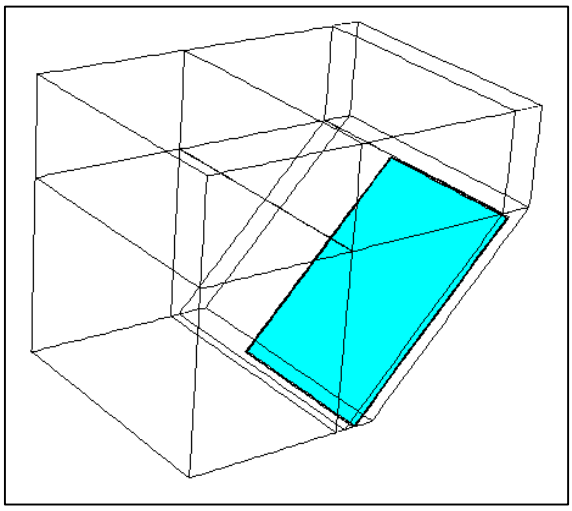

**Figure 11 -- Isometric view of assembled TASCflow model for the windshield problem. The attachments between the six grid objects presented in this section can be seen.**

The grid objects shown in the multi-grid assembly are as follows, using a clockwise convention beginning with the object in the lower left-hand corner of the figure; “Fire”, “Above_Fire”, “Top_Front”, “Top_Mid”, “Wind”, and “Above_Eng”. With the multi-grid assembly for the windshield problem completed, boundary conditions were then assigned.

## *4.6 Boundary Conditions*

The boundary conditions applied to the multi-grid assembly of the windshield problem were conducted using the TASCbob3d utility available in TASCflow. A symmetry boundary condition was applied along the length of the windshield model for all objects in order to substantially reduce the amount of nodes required to model the problem as well as to reduce the CPU time associated with the simulations. Implicit in the application of this boundary condition is the assumption of axi-symmetric conditions along the length of the vehicle. Therefore, an axi-symmetric fire was assumed in the modeling effort.

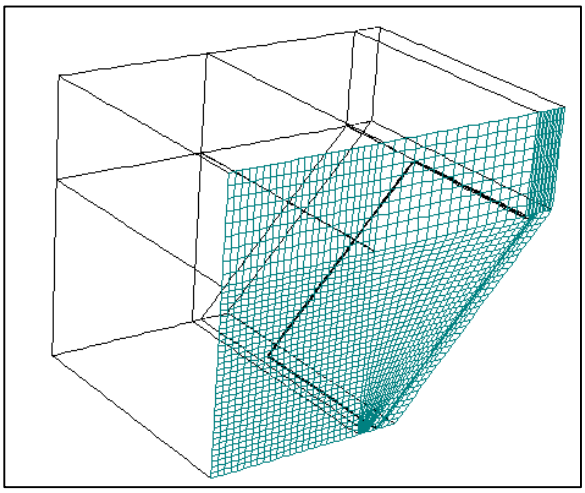

**Figure 12 -- Isometric view of the symmetry boundary condition applied to the TASCflow windshield model.**

A no-slip wall boundary condition has been applied along what would constitute the hood of a passenger vehicle. An emmissivity value consistent with mild steel of 0.9 has been specified. For convenience this boundary condition has also been applied directly underneath the windshield and into the beginning of the passenger compartment. This has been done to satisfy the requirement that conjugate heat transfer objects, like the windshield, at the exterior of the fluid domain have a wall-type boundary condition applied to them.

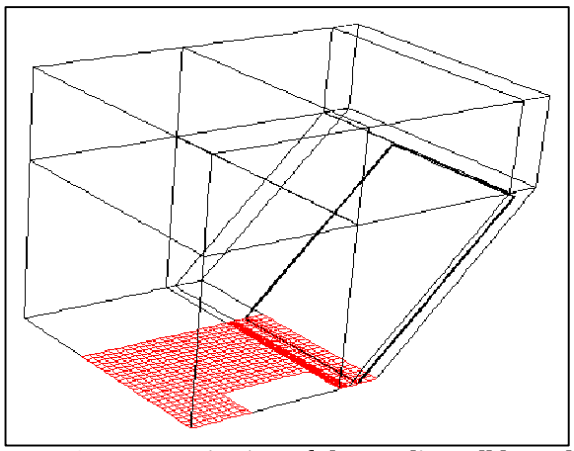

**Figure 13 -- Isometric view of the no-slip wall boundary condition applied to the hood of the vehicle. This reveals the fluid extension beside the hood which is used to capture the flow field of air near the vehicle.**

A no-slip wall boundary condition has been applied to the external surfaces of the windshield conjugate heat transfer object. An emmissivity of 0.05 has been specified due to the reflective and transparent nature of the windshield. The application of this boundary condition is shown in the figure that follows.

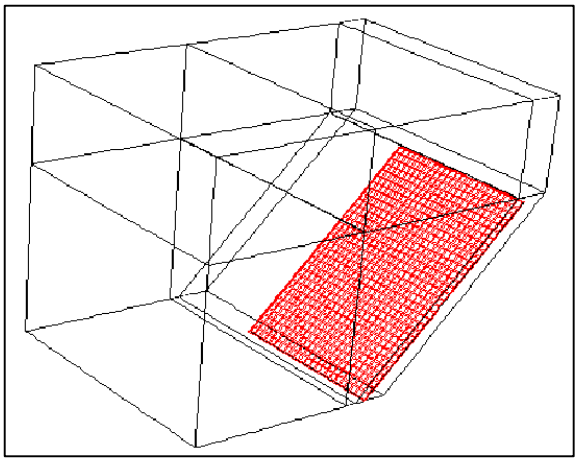

**Figure 14 -- Isometric view of the no-slip wall boundary condition applied to the exterior surface of the 3-layer windshield that is exposed to fluid.**

An inlet boundary condition has been applied to a rectangular region in the "Fire" object for the purpose of supplying fuel for the combustion reaction. A diameter of 0.5m was assumed for the base of the fire, which translated into an area equivalent diameter in the following manner;

$$x^2 = \frac{pD^2}{4} \Rightarrow x = \frac{D}{2}\sqrt{p} \qquad (24)$$

For D = 0.5, the area equivalent diameter was found to be 0.44m. The appropriate nodal region was then selected based on this diameter for the purpose of specifying the inlet boundary condition.

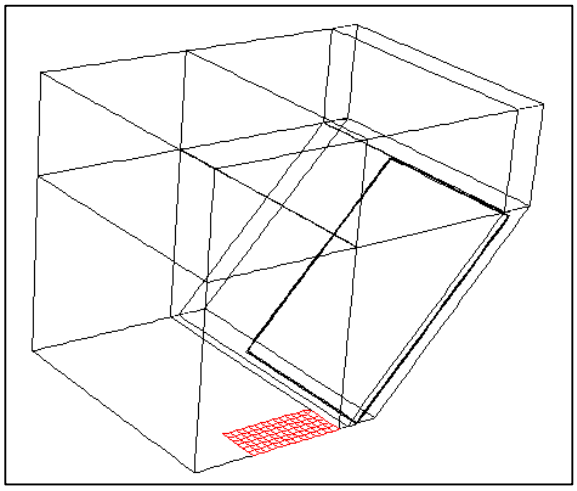

**Figure 15 -- Isometric view of inlet boundary condition.**

The mass flow rate from this inlet condition was specified in terms of the ProfilARBED design heat release curve specified as an equivalent source of methane and in accordance to the mass fraction of the fluid components. The fluid components defined for use in the combustion model are Fuel, Oxidant, Products and two non-reacting components; Soot and Nitrogen. The heat release rate during the first four minutes of the curve can be expressed as a function of time in the following manner;

$$Q'(t) = 1{,}400*(t/240)\ \text{kW} \quad \text{for } 0 \leq t \leq 240\ \text{s} \qquad (25)$$

where;

Q' -- the total heat release rate of the fire [kW]

t -- time [s]

The heat release rate of a fire is related to the mass flow rate and heat of combustion of the fuel as follows;

$$Q'(t) = m'(t)\Delta H_c \qquad (26)$$

where;

m' -- the mass flow rate of fuel [kg/s]

$\Delta H_c$ -- the heat of combustion for the fuel [kJ/kg]

Using the heat release rate function from above with a heat of combustion of 49,600kJ/kg for methane, the mass flow rate can be expressed as;

$$m'(t) = [1{,}400*(t/240)]/[\Delta H_c] \qquad (27a)$$

$$m'(t) = [5.833*t]/[49{,}600] \qquad (27b)$$

$$m'(t) = 0.0001176*t \text{ kg/s} \qquad (27c)$$

The fuel mass fractions specified at the inlet condition were based on an assumed equivalence ratio (ratio of fuel mass to air mass). Five components of the combustion reaction were specified; Fuel, Oxygen, Products, Soot, and Nitrogen. Soot and Nitrogen were specified as non-reacting components of the combustion reaction. The order of specification is significant as the first four quantities are explicit and the last quantity, Nitrogen in this case, is deduced by subtracting the sum of the mass fractions from unity. Therefore, the mass fractions at the inlet boundary summed to unity as shown below.

$$X_{Fuel} + X_{Oxidant} + X_{\Pr oducts} + X_{Soot} + X_{Nitrogen} = 1 \qquad (28)$$

The equivalence ratio, Φ, describes the ratio of fuel mass to air mass, and is useful for specifying the excess fuel at the inlet. Therefore, the fuel mass fraction can be described in the following manner, assuming that the ambient air is made up of Nitrogen, Oxygen, and Products (carbon dioxide, carbon monoxide, and water vapor);

$$X_{Fuel} = \Phi X_{Air} = \Phi\left(X_{Nitrogen} + X_{Oxygen} + X_{\Pr oducts}\right) \qquad (29)$$

The soot yield, $y_s$, for a given fuel is typically reported as a fraction of the fuel mass loss rate, therefore, the soot mass fraction is expressed as;

$$X_{soot} = y_s X_{Fuel} \qquad (30)$$

Equation 28 can be simplified into the following form;

$$X_{Fuel} + X_{Air} + X_{Soot} = 1 \qquad (31)$$

Substituting Equations 29 and 30 into 31, we have;

$$\Phi X_{Air} + y_s \Phi X_{Air} + X_{Air} = 1 \qquad (32)$$

Therefore, Equation 32 can be solved for the mass fraction of air using the following;

$$X_{Air} = \frac{1}{\left(\Phi + y_s \Phi + 1\right)} \qquad (33)$$

The mass fractions of Oxygen, Nitrogen, and Products entering the domain at the inlet boundary are determined by multiplying the air mass fraction by the ambient mass fractions of Oxygen, Nitrogen, and Products. Using the known mass fraction of air, Equation 31 can be rewritten as;

$$X_{Fuel} + X_{Air} + y_s X_{Fuel} = 1 \qquad (34)$$

The fuel mass fraction can be expressed as;

$$X_{Fuel} = \frac{1 - X_{Air}}{1 + y_s} \qquad (35)$$

The mass fraction of soot is calculated using Equation 30 with the know value for the fuel mass fraction.

For an equivalence ratio of 5, the following mass fractions were determined using the method above for the inlet boundary condition.

Fuel = 0.7519

Oxidant = 0.0316

Products = 0.0045

Soot = 0.0977

The remaining 0.1143 is deduced as Nitrogen. In light of the fuel mass fraction the inflow boundary condition must be redefined by dividing by the fuel mass fraction to ensure that the required amount of fuel enters the domain for combustion. The relatively smaller mass fractions for the other fluid components are used to seed the combustion reaction as suggested in the TASCflow user documentation to facilitate better convergence in the combustion sub-model. The function for the inlet boundary mass flow rate is then;

m'(t) = (0.0001176*t)/(0.7519) (36a)

m'(t) = 0.0001564*t kg/s (36b)

Therefore, the mass flow rates into the domain via the inlet boundary condition at 30 second intervals were;

| time [s] | m' inlet [kg/s] |
|---|---|
| 30 | 0.00469 |
| 60 | 0.00938 |
| 90 | 0.01408 |
| 120 | 0.01877 |
| 150 | 0.02345 |
| 180 | 0.02816 |
| 210 | 0.03285 |
| 240 | 0.03754 |

**Table 10 -- Inlet boundary condition mass flow rates at 30 second intervals.**

The remaining external fluid boundaries were included in an opening boundary condition. The opening boundary condition allows fluid to flow in and out of the domain depending upon the flow field conditions. Fluid flows out of the domain at the local calculated values (pressure, temperature, species, etc.) and fluid flows into the domain under ambient conditions. This is preferable in the interest of properly capturing the fluid flow, in comparison to imposing the fluid flow in the domain by specifying separate inflow and outflow boundary conditions at specific faces of the computational domain.

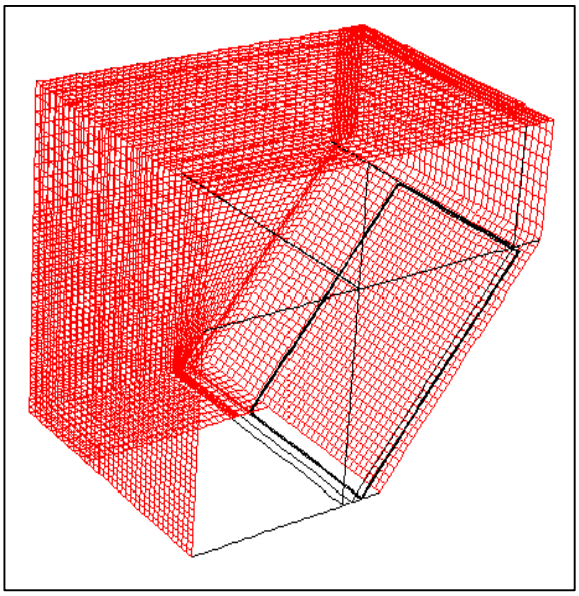

**Figure 16 -- Isometric view of opening boundary condition applied to all exterior boundaries of the computational domain that are exposed to fluid.**

A series of programs that interact with the command line of the various TASCflow utilities were developed in the C programming language to automate model construction and assure consistency geometric and nodal specifications as well as in the application of boundary conditions.

## *4.7 Conclusions*

An appropriately parameterized combustion model using turbulent, reacting, multi-component fluid flow, with heat transfer by convection, conduction, and radiation has been developed for the windshield problem using TASCflow Computational Fluid Dynamics software. The parametric nature of the windshield problem has been considered by establishing an input file that governs the creation of all associated grid objects in terms of geometry and nodal distributions. A multi-grid approach has been

utilized in formulating the TASCflow model in order to account for the slope of the windshield as well as the rectangular nature of the combustion region above the hood.

# 5.0 Applications of TASCflow

Approximately 250 runs have been completed in the course of this project. While earlier runs were performed to help establish proper grid sizing, appropriate boundary conditions, and model sensitivities of to various input parameters, later runs were performed in a systematic fashion to assist in studying the windshield problem.

## *5.1 TASCflow Governing Equations*

The conservation equations for mass, momentum, and energy for a Newtonian fluid can be expressed using index notation as [17],

$$\frac{\partial r}{\partial t} + \frac{\partial}{\partial x_j}\left(r u_j\right) = 0 \qquad (37)$$

$$\frac{\partial}{\partial t}\left(r u_i\right) + \frac{\partial}{\partial x_j}\left(r u_j u_i\right) = -\frac{\partial P}{\partial x_i} + \frac{\partial t_{ij}}{\partial x_j} + S_{ui} \qquad (38)$$

$$\frac{\partial}{\partial t}\left(rH\right) - \frac{\partial P}{\partial t} + \frac{\partial}{\partial x_j}\left(r u_j H\right) = -\frac{\partial q_j}{\partial x_j}\left(u_j t_{ij}\right) + S_E \qquad (39)$$

where $u_i$ is the velocity components in the $x_i$-coordinate directions, $P$ is the static pressure, $H$ is total enthalpy, $r$ is density, $t_{ij}$ is the viscous stress tensor, $q_j$ is the conduction, and the $S$ terms are additional source terms [17].

Total enthalpy is,

$$H = h + \frac{u_i u_i}{2} \qquad (40)$$

where $h$ is the fluid's static enthalpy.

The viscous stress tensor and the conduction expressions can be written in terms of velocity, temperature and concentration gradients in the following manner [17],

$$t_{ij} = +m\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \frac{2}{3}m\frac{\partial u_l}{\partial x_l}d_{ij} \qquad (41)$$

$$q_j = -l\frac{\partial T}{\partial x_j} - \sum_k^n \Gamma_k h_k \frac{\partial Y_k}{\partial x_j} \qquad (42)$$

where $m$ is the dynamic fluid viscosity, $l$ is the conductivity of the fluid, $G_k$ is the molecular diffusion coefficient, $h_k$ is the static enthalpy and $Y_k$ is the mass fraction of species [17].

For turbulent flows, the scalar variables fluctuate and the instantaneous value of the scalar can be expressed by summing the mean and fluctuating components. This is done by a time-averaging process and the results are shown below as Favre averaged equations [17],

$$\frac{\partial \overline{r}}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{r}\tilde{u}_j\right) = 0 \qquad (43)$$

$$\frac{\partial}{\partial t}\left(\overline{r}\tilde{u}_i\right) + \frac{\partial}{\partial x_j}\left(\overline{r}\tilde{u}_j\tilde{u}_i\right) = -\frac{\partial \overline{P}}{\partial x_i} - \frac{\partial}{\partial x_j}\left(t_{ij} - \overline{r u_i'' u_j''}\right) + \overline{S_{ui}} \qquad (44)$$

$$\frac{\partial}{\partial t}\left(\overline{r}\tilde{H}\right) - \frac{\partial \overline{P}}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{r}\tilde{u}_j\tilde{H}\right) = -\frac{\partial}{\partial x_j}\left(\overline{q_j} + \overline{r u_j'' H''}\right) + \frac{\partial}{\partial x_j}\left\{\tilde{u}_i\left(\overline{t_{ij}}\right) - \overline{u_i'' t_{ij}''}\right\} + \overline{S_E} \qquad (45)$$

The mean value of the total enthalpy is,

$$\tilde{H} = \tilde{h} + \frac{\tilde{u}_i \tilde{u}_i}{2} + \tilde{k} \qquad (46)$$

where the turbulent kinetic energy is,

$$\tilde{k} = \frac{1}{2} \overline{\rho u_i'' u_i''} / \bar{\rho} \qquad (47)$$

The fluctuating component of the total enthalpy is given as,

$$H'' = h'' + \tilde{u}_i u_i'' + \overline{\frac{1}{2} \rho' u_i'' u_i''} \qquad (48)$$

***Turbulence Model***

Using an eddy viscosity assumption, the Reynolds stress and turbulent flux terms can be related to the mean flow variables in the following manner,

$$\overline{\rho u_i'' u_j''} = -\mu_t \left( \frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} \right) + \frac{2}{3} \left( \mu_t \frac{\partial \tilde{u}_l}{\partial x_l} + \bar{\rho} \tilde{k} \right) \delta_{ij} \qquad (49)$$

For incompressible fluid flows, we have,

Using the eddy diffusivity assumption, the turbulent energy fluxes can be expressed as,

$$-\overline{\rho u_j'' h''} = \frac{\mu_t}{\mathrm{Pr}_t} \frac{\partial \tilde{h}}{\partial x_j} \qquad (50)$$

$$-\overline{\rho u_j'' h''} = -\overline{\rho u_j' h'} = \frac{\mu_t}{\mathrm{Pr}_t} \frac{\partial \bar{h}}{\partial x_j} \qquad (51)$$

$$\frac{\partial\left(\overline{u_i^{"}t_{ij}^{"}}\right)}{\partial x_j} \approx \overline{\frac{\partial}{\partial x_j}\left(m\frac{\partial\tilde{k}}{\partial x_j}\right)} \qquad (52)$$

$$m_t = rc_m\frac{k^2}{e} \qquad (53)$$

$$\frac{\partial(rk)}{\partial t}+\frac{\partial\left(r\overline{u}_j k\right)}{\partial x_j}=\frac{\partial}{\partial x_j}\left(\Gamma_k\frac{\partial k}{\partial x_j}\right)+P_k-re \qquad (54)$$

$$\frac{\partial(re)}{\partial t}+\frac{\partial\left(r\overline{u}_j e\right)}{\partial x_j}=\frac{\partial}{\partial x_j}\left(\Gamma_e\frac{\partial e}{\partial x_j}\right)+\frac{e}{k}\left(c_{e1}P_k-rc_{e2}e\right) \qquad (55)$$

$$\Gamma_k = m+\frac{m_t}{s_k} \qquad (56)$$

$$\Gamma_e = m+\frac{m_t}{s_e} \qquad (57)$$

$$P_k = -\overline{ru_i^{'}u_j^{'}}\frac{\partial\overline{u}_i}{\partial x_j} \qquad (58)$$

$$P_k = m_t\left(\frac{\partial\overline{u}_i}{\partial x_j}+\frac{\partial\overline{u}_j}{\partial x_i}\right)\frac{\partial\overline{u}_i}{\partial x_j}-\frac{2}{3}\left(rk+m_t\frac{\partial\overline{u}_l}{\partial x_l}\right)\frac{\partial\overline{u}_k}{\partial x_k} \qquad (59)$$

$$P_k = m_t\left(\frac{\partial\overline{u}_i}{\partial x_j}+\frac{\partial\overline{u}_j}{\partial x_i}\right)\frac{\partial\overline{u}_i}{\partial x_j} \qquad (60)$$

Mass

$$\frac{\partial r}{\partial t}+\frac{\partial}{\partial x_j}\left(ru_j\right)=0 \qquad (61)$$

Momentum

$$\frac{\partial}{\partial t}(\rho u_i)+\frac{\partial}{\partial x_j}(\rho u_j u_i)=-\frac{\partial P^*}{\partial x_i}+S_{ui}+\frac{\partial}{\partial x_j}\left\{\mu_{eff}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)-\frac{2}{3}\mu_{eff}\frac{\partial u_l}{\partial x_l}\delta_{ij}\right\} \qquad (62)$$

$$\mu_{eff}=\mu+\mu_t \qquad (63)$$

$$\frac{}{t}(\rho u) \quad \frac{}{x}(\rho u u) \quad \frac{P}{x} \quad \frac{}{x} \quad \mu \quad \frac{u}{x} \quad \frac{u}{x} \quad +S_{ui} \qquad (64)$$

Energy

$$\frac{\partial}{\partial t}(\rho H)-\frac{\partial P}{\partial t}+\frac{\partial}{\partial x_j}(\rho u_j H)=\frac{\partial}{\partial x_j}\left(\lambda\frac{\partial T}{\partial x_j}+\frac{\mu_t}{\mathrm{Pr}_t}\frac{\partial h}{\partial x_j}\right)+S_E+\frac{\partial}{\partial x_j}\left\{u_i\left[\mu_{eff}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)-\frac{2}{3}\mu_{eff}\frac{\partial u_l}{\partial x_l}\delta_{ij}\right]+\mu\frac{\partial k}{\partial x_j}\right\} \qquad (65)$$

Where,

$$H=h+\frac{1}{2}u_i u_i+k \qquad (66)$$

$$\frac{\partial}{\partial t}(\rho H)-\frac{\partial P}{\partial t}+\frac{\partial}{\partial x_j}(\rho u_j H)=\frac{\partial}{\partial x_j}\left(\lambda\frac{\partial T}{\partial x_j}+\frac{\mu_t}{\mathrm{Pr}_t}\frac{\partial h}{\partial x_j}\right)+S_E \qquad (67)$$

$$\frac{\partial}{\partial t}(\rho H)-\frac{\partial P}{\partial t}+\frac{\partial}{\partial x_j}(\rho u_j H)=\frac{\partial}{\partial x_j}\left(\lambda\frac{\partial T}{\partial x_j}+\frac{\mu_t}{\mathrm{Pr}_t}\frac{\partial h}{\partial x_j}\right)+S_E+\frac{\partial}{\partial x_j}\left\{u_i\left[\mu_t\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)-\frac{2}{3}\mu_t\frac{\partial u_l}{\partial x_l}\delta_{ij}\right]\right\} \qquad (68)$$

$$\frac{\partial}{\partial t}(\rho h)-\frac{\partial P}{\partial t}+\frac{\partial}{\partial x_j}(\rho u_j h)=\frac{\partial}{\partial x_j}\left(\lambda\frac{\partial T}{\partial x_j}+\frac{\mu_t}{\mathrm{Pr}_t}\frac{\partial h}{\partial x_j}\right)+S_E \qquad (69)$$

$$\frac{\partial(rcT)}{\partial t} = \frac{\partial}{\partial x_j}\left( l \frac{\partial T}{\partial x_j} \right) \qquad (70)$$

$$\overline{ru_i''u_j''} = r\overline{u_i'u_j'} = -m_t\left( \frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} \right) + \frac{2}{3} r d_{ij} k \qquad (71)$$

**Finite Volume Radiation Model as Implemented in TASCflow**

Without a participating medium, photons travel in every possible direction without interference through the medium [18]. In the presence of a participating medium, some photons are absorbed and some are redirected, or scattered, by collisions. Due to emission of the medium, "new" photons appear. The conservation of radiant energy for a volume $V$ and a discrete solid angle $w^l$ is expressed as [18];

$$\int_{w^l}\int_S I\vec{s}\cdot\vec{n}\,dS\,dw = \int_{w^l}\int_V K_m(I_b - I)\,dV\,dw + \int_{w^l}\int_V s_m(I_{Sc} - I)\,dV\,dw \qquad (72)$$

where

$$I_{Sc} = \frac{1}{4p}\int_{4p} I(\vec{s}')\Phi(\vec{s}', \vec{s})\,dw' \qquad (73)$$

where

I is the radiant intensity

$I_b = \sigma T_m^4/\pi$ is the intensity of emission of the medium

$I_{Sc}$ is the intensity of radiant energy scattered into the direction $\vec{s}$

$K_m$ is the medium absorption coefficient

$\sigma_m$ is the medium scattering coefficient

$\Phi$ is the scattering phase function

The first term of the radiation conservation equation is the net flow of radiant energy within $\omega^l$ leaving through the control volume surfaces. The next term is the net augmentation of radiant energy within the volume V and within $\omega^l$ due to emission and absorption. This term represents transfer between thermal and radiant energy, and the negative of this term summed over all $\omega^l$ is a source term in the thermal energy equation. The last term is the net augmentation of radiant energy within the volume V due to inscattering into $\omega^l$ and outscattering from $\omega^l$.

The discrete form of the radiation conservation equation for node *N* and solid angle $\omega^l$ is expressed as;

$$\sum_{ip} I_{ip}^l \vec{N}^l \cdot \vec{n}_{ip} S_{ip} = \left(K_{m,N}\left(I_{b,N} - I_N^l\right)V_{N^{w^l}}\right) + \left(s_{m,N}\left(\bar{I}_{Sc,N}^l - I_N^l\right)V_{N^{w^l}}\right) \qquad (74)$$

where

$$\vec{N}^l = \int_{w^l} \vec{s}\, dw \qquad (75)$$

$$\bar{I}_{Sc,N}^l = \sum_m I_N^m \overline{\Phi}(m,l) \qquad (76)$$

$$\overline{\Phi}(m,l) = \frac{1}{4p}\int_{w^l}\int_{w^m} \Phi(\vec{s}', \vec{s})\, dw' dw \qquad (77)$$

$\vec{N}^l$ is the mass flux vector for photons contained in the solid angle $w^l$ and $I_{ip}^l$ is the advected quantity.

**Particle Participation in the Finite Volume Radiation Model**

Particle emission and absorption of radiation within the solid angle $w^l$ can be accounted for by adding the source term below to the right hand side of conservative form of the radiation equation [18].

$$\int \left(K_p \bar{I}_{bp}\left(T_p\right) - K_p \bar{I}\right) V dw \qquad (78)$$

The total number of particles on all paths, *G*, within the discrete volume, *V*, are

$$N_p = \sum_{g=1}^{G} \int_{S_g} \left( \frac{\dot{N}_p}{\left|\vec{V}_p\right|} \right)_g ds \qquad (79)$$

The total radiant energy emitted from all particles along all paths is

$$E_{rad,tot} = \sum_{g=1}^{G} \int_{S_g} e_P 4p^2 r^2 I_{bP} \left( \frac{\dot{N}_p}{\left|\vec{V}_p\right|} \right)_g ds = 4p^2 \overline{e_p r_p^2 I_{bp}} N_p \qquad (80)$$

The radiant energy is assumed to be emitted isotropically and the portion emitted in $dw$ is expressed as $dw/(4p)$. Therefore, the emission term above can be expressed as

$$K_p \bar{I}_{bp} = p \overline{e_p r_p^2 I_{bp}} N_p / V \qquad (81)$$

In a similar manner, the total amount of energy of intensity *I* traveling in *dw* that is absorbed within the discrete volume *V* can be expressed as

$$\int_{w^l} K_p \bar{I} V dw = \sum_{g=1}^{G} \int_{S_g} pe_p r_p^2 \bar{I} \frac{\dot{N}_p}{\left|\vec{V}_p\right|} ds dw = \bar{I} p \overline{er_p^2} N_p \qquad (82)$$

Particle absorptivity is expressed as

$$K_p = p \overline{er_p^2} \frac{N_p}{V} \qquad (83)$$

**Basic Solution of Finite Volume Radiation Model Equations**

The nodal intensities are updated for each direction using the given intensities leaving the boundaries. The nodal intensities are used in the energy balances on the element boundary faces to update the intensity values leaving the boundaries. The process is repeated in an iterative manner until the convergence criteria are achieved.

## *5.2 Limitations of TASCflow Parametric Model*

As with any model, a mathematical representation of a physical situation is made. This representation includes assumptions and approximations that are required in the interest of simplifying certain mathematical complexities, to reduce computational time, or to deal with poorly understood phenomena. Therefore, it is imperative that the mathematical modeler understands the assumptions and approximations that have been made to ensure that the model is being applied only to situations that it is intended for. These items are presented below;

- The most significant limitation of the model developed for the windshield problem is that it has only been validated with correlations based on general principles of fire dynamics. The quality of this model must be ensured by simulating controlled fire experiments of a passenger vehicle windshield subjected to the heat flux and fluid flow of an engine compartment fire and making comparisons of the model output to the results of such controlled fire experiments. The fact that the computational fluid dynamics model is able to provide a converged solution for a turbulent, reacting, multi-component fluid flow with radiation and conjugate heat transfer objects does not imply that the solution is correct. A converged solution could be based on poorly selected boundary conditions that produce a flow field solution with environmental variables that are physically unrealistic with respect to the situation the model is designed to simulate. Comparisons to experimental measures can evaluate the overall quality of the CFD model.

- The model that has been developed for the windshield problem assumes axi-symmetric behavior along the entire length of the car. The intention of this assumption is to reduce the total number of nodes required to model the phenomena associated with the windshield problem, which has a direct influence on the amount of computer memory, disk storage space, and CPU time required for the simulations. For example, if the entire fluid domain consists is 10 by 11 by 8 nodes, applying a symmetry boundary condition along the xz-plane yields a domain that is 10 by 6 by 8 nodes. The total number of nodes in using the symmetry condition is 480 as opposed to the 880 needed for the full fluid domain. Therefore, the model simulates a fire that is symmetric with respect to the xz-plane.

- The model for the windshield problem simulates a fire occurring at the surface of the vehicle's hood. This is based on observations made during the fire testing conducted at Factory Mutual by GM. Since flame spread and fire development within the engine compartment is poorly understood, the fire is assumed to occur at the top of the hood surface. The advantage to this approach is that the total number of nodes to model windshield problem is reduced by not having to account for regions below the hood-level of the vehicle.

- The fire modeled for the windshield problem is an inlet supply of methane that is sufficient to supply an energy output equivalent to that specified in the ProfilARBED design heat release rate. This curve is intended for use in parking garage design, but was applied to the windshield problem. The approximation with respect to the fuel is also based on a poor understanding of flame spread and fire growth within the engine compartment of a passenger vehicle. This approximation is appropriate with respect to the amount of energy released by the fire, however, the energy distribution, extent of the combustion reaction, and species production are fuel-specific quantities.

- The geometry of the inlet condition is assumed to have an equivalent diameter of 0.5m and influences the resulting flame height of the engine fire. In reality the diameter of the fire will change with the heat release rate of the fire. A 0.5m diameter has been assumed based on an assessment of what a reasonable fire diameter would be for the windshield problem and considering that the diameter of the fire would be limited by the width of the engine compartment that is in the vicinity of 1.5m. By following the methodology that has been established for this calculation procedure, the sensitivity of

flame height on the selection of a fire diameter can be assessed for other heat release rates, such as those recorded during controlled fire experiments.

- The windshield constructed in the model is straight, not curved, and because of limitations imposed by the grid generation process using TASCgrid. This represents the worst-case scenario in terms of a windshield subjected to the heat flux and fluid flow of an engine compartment fire because the distance from the fire to the windshield is shorter in the model than for a windshield with curvature. This is particularly true at the extreme ends of the windshield.

- The crack-patterns in the windshield due to the collision are not accounted for in the model. The primary reason for this is that the nature of post-collision windshield crack patterns is poorly understood because it is related to the complex collision dynamics. From a practical standpoint, the number of nodes required to account for crack patterns in the vehicle windshield would place a tremendous burden with respect to computational time as well as memory and disk spaces storage requirements.

- The hood specified in the model is flat not curved. Again, this has been done in the interest of keeping a rectangular computational domain for the fluid region above the hood where the combustion reaction takes place. The slant and curvature of the hood was not considered to be a significant factor in influencing the heat flux and fluid flow from an engine compartment fire to a passenger vehicle windshield.

- The model is only valid until fire spreads from the engine to the passenger compartment, therefore, only the initial portion of the ProfilARBED design heat release rate curve has been used. Ignition and fire development in the passenger compartment is not accounted for in the model because occupant tenability has been compromised.

Computational fluid dynamics modeling presents a wide variety of challenges that may be encountered while setting up complex scenario such as the windshield problem, therefore, an iterative approach is needed in initial development stages.

One major concern in setting up the windshield problem was the appropriate number and spacing of nodes in the computational domain. The larger the number of nodes used to model a problem results in increased CPU time with the benefit of increased solution accuracy. Proper consideration had to be given to the location of large gradients expected for environmental variables such as velocity and temperature in the computational domain. For example, a large velocity gradient would appear in the fluid region close to and including a no-slip wall boundary condition where the fluid velocity local to the wall surface changes much more rapidly than the bulk fluid velocity throughout the rest of the computational domain. Convergence problems could arise in computing the fluid velocity near no-slip walls because of this relatively larger gradient. Therefore, three methods can be employed to resolve this issue;

1. Increasing the number of nodes throughout the entire fluid domain
2. Embedding additional nodes in the vicinity of the gradient by refining the existing grid
3. Biasing the nodes in the direction of the gradient so that the nodes are spaced closer to the gradient.

The following gradient locations were anticipated in the windshield model;

- *The airspace near the inner and outer surfaces of the windshield* – a no-slip wall boundary condition has been applied to these surfaces where large velocity and temperature gradients were expected. The nodes in the airspace perpendicular to the

windshield's inner and outer surfaces have been biased towards the windshield so that the node spacing is closer to the anticipated gradients.

- *The airspace near the surface of the hood* – a no-slip boundary condition has been applied at the surface of the hood where a large velocity gradient is expected with respect to the bulk fluid velocity in the computational domain. This has been dealt with by biasing the nodes in the airspace above the hood in the direction of the hood so that the nodes are biased in the vertical direction and tightly packed near the surface of the hood.

- *The airspace near the fuel inlet* – the inlet boundary condition represents a large gradient for velocity, species distribution, and temperature in comparison to the rest of the domain. In order to resolve the gradients associated with velocity and species distribution, the grid in the vicinity of the inlet condition has been refined by embedding a region of finer node spacing parallel to the inlet condition in the x and y coordinate directions. This region extends beyond the length and width of the inlet plane to capture the associated edge effects and extends in the vertical direction above the inlet. Refinement in the vertical direction, however, was not necessary because the global node spacing above the hood had already been biased towards the hood. The temperature and vertical velocity gradients above the inlet will be resolved by this node biasing associated with the airspace above the hood of the vehicle.

An iterative approach was applied in order to determine an appropriate number and spacing of nodes in the computational domain, including the application of nodal biasing and grid refinement. In following an incremental development of the solution of the windshield problem, deficiencies in the grid became apparent in later stages of the

solution as more complex sub-models were employed. The solution technique, described in detail later in this chapter, consisted of obtaining the fluid solution at first and then using the solution as an initial guess for the addition of another sub-model. The incremental inclusion of sub-models was in the following order; fluid, temperature and conjugate heat transfer, turbulence, multi-component fluid mixture, combustion, and radiation. After an appropriate node distribution was developed for the fluid region, the solution was developed incrementally. The inclusion of the combustion sub-model however, revealed deficiencies in the nodal distribution as poor convergence was observed in the combustion zone. Physically unrealistic temperatures in the vicinity of the inlet boundary condition characterized this poor convergence. Often, the gas temperature was below ambient and became negative causing the fluid solver to shut down. The refinement in the are of the inlet condition required special consideration in order to capture the proper temperature distribution resulting from the combustion reaction. A new restart file had to be interpolated for the computational domain when the number or distribution of nodes was altered with respect to the previous grid. The new restart file was generated using the `genfin` command in the tasctool utility, which requires the location of the previous grid, boundary condition, and restart file used for the problem.

## *5.3 Scenarios*

The parametric combustion model of the windshield problem was applied to a series of engine compartment fire scenarios using the selected design heat release rate curve. The location of the fire was moved in the x-direction with respect to the base of

the windshield. The closest distance, or worst-case scenario, was selected as being 10cm from the base of the windshield to the leading edge of the inlet boundary condition, which would be 32cm from the base of the windshield to the center of the fire. The environmental conditions were examined at 30 second intervals during the first 4 minutes of fire growth for the design heat release rate curve.

## *5.4 Initial Guess*

The initial guess for each problem is made using the TASCtool utility to specify the velocity, temperature, species, and turbulence fields for the problem. These fields are initialized on a global basis using the `initial` command in TASCtool, which is suitable for starting simple flow problems. After initializing the required fields, TASCtool writes a restart (.rso) file that is used to start the simulation. However, fatal errors in the flow solver were experienced in using the global initialization. It became apparent that a more detailed initial guess would be required for simulations as complex as those considered in the windshield problem. After initializing the fields on a global basis and creating the restart file in TASCtool, the restart file is then read into memory using the `read rso` command. Now local values could be assigned for the various field quantities using the `calc` command in TASCtool.

For example, convergence problems were experienced in the vicinity of the inlet boundary condition because the incoming fuel introduced large gradients on a local basis relative to the global values of species concentration, velocity, and temperature. The global value for the fuel mass fraction was set to zero using the `initial` command. However, the inlet boundary condition introduces a fuel mass fraction of 0.994 into the

domain. In order to help achieve better convergence in the vicinity of the inlet condition, the fuel mass fraction was overwritten in a region that extended 3 nodes in all three coordinate directions with respect to the edge of the inlet boundary condition.

To illustrate this process, consider a "Fire" grid object of nodal dimensions 24x36x32 with an inlet boundary condition specified at [11:24,1:8,1]. The region where the fuel mass fraction will be overwritten would be [8:24,1:11,1:3] for the "Fire" object and [1:3,1:11,1:3] for the "Above_Eng" object attached to the "Fire object". A fuel mass fraction of 0.2 will be assigned in these regions in order to provide a transition from the 0.994 at the inlet condition to the 0.0 mass fraction in the rest of the initialized fluid domain. This value of 0.2 is consistent with a recommendation in the TASCtool user documentation that the initial guess err on the low side in the interest of improved solution convergence. These local values would be assigned in TASCtool using the following commands;

```
>calc ch4[8:24,1:11,1:3]:fire = 0.2
>calc ch4[1:3,1:11,1:3]:above_eng = 0.2
```

Other local specifications for the other fluid components, temperature, and u,v, and w velocities were made using similar commands in TASCtool. Once these adjustments were made to the initial guess, the information was written to the restart file by using the `write rso` command in TASCtool.

## *5.5 Solution Development*

The use of relaxation parameters help to establish a stable solution for flow fields where large gradients exist. After a stable solution has been established, the amount of

relaxation must be decreased in the interest of developing a converged solution. Therefore, a solution methodology has been developed and used in order to achieve a stable and converged solution of this complex flow problem. The procedure, in general, consists of beginning with the fluid solution only and including other sub-models such as turbulence, combustion, and radiation, in an incremental fashion using the result of the previous solution as the initial guess for the current solution. For each sub-model included, the relaxation parameters are set to 0.2, which provides a large degree of relaxation for the particular solution. Once a stable solution has been achieved after the specified number of iterations, the amount of relaxation must be reduced in order to formulate a converged solution of the flow field using the sub-models that have been enabled.

The following methodology was used in developing the flow field solution of the TASCflow model of the windshield problem.

1. After establishing the initial conditions in the computational domain, the fluid solution was the only model selected in the parameter file and all relaxation parameters were set to 0.2.

2. The relaxation parameters in the parameter file were increased to 0.5 and the resulting output of step 1 was used as the initial guess.

3. The relaxation parameters were increased to 0.75, the output file of step 2 was used as the initial guess for the fluid only solution.

4. The relaxation parameters were increased to 1.0 so there was no relaxation of parameters, and the output file of step 3 was used as input to this step. At the end of the iterations, the fluid solution was obtained and the output file was saved for the purpose of

having an appropriate restart file in the event of the solution became corrupted by the inclusion of subsequent models.

5. The temperature and conjugate heat transfer sub-models were enabled and all relaxation parameters were set to 0.2 to provide sufficient relaxation in the solution. The initial conditions for the temperature field were established by using the `read` rso command in TASCtool and performing series of calc statements to assign temperature values in the domain. The role of parameter relaxation is important as the fluid solution obtained at the end of step 4 would be affected by the change in energy occurring in the computational domain with the inclusion of the temperature and CHT models.

6. The solution obtained in step 5 is then used as the initial guess for a solution with relaxation parameters set to 0.5.

7. The amount of relaxation is reduced further as the parameters are set to 0.75 with the output file of step 6 becoming the initial guess for the flow field solution at this step.

8. The final fluid solution including the energy equation and conjugate heat transfer is obtained by using the result of step 7 as an initial guess with all relaxation removed in setting all relaxation parameters to 1.0. The final solution including fluids, temperature, and CHT properties is saved in order to be a restart file in the event that the use of subsequent submodels corrupts the solution.

9. The turbulence model is enabled and acts on the solution obtained in step 8 with all relaxation parameters set to 0.2. The values of TKE and EPSILON are frozen as the parameter TKE_AND_EPSILON is set to false. This allows for a smoother transition from the laminar to turbulent solution.

10. The turbulent solution is processed again with all relaxation parameters set to 0.5 and by using the solution of step 9 as the initial guess.

11. The relaxation parameters are all set to 0.75 and the turbulent solution continues using the results of step 10 as the initial guess.

12. The final solution for the fluid, temperature, CHT, and turbulence sub-model solution is obtained by using the result of step 11 as the initial guess with all relaxation parameters set to 1.0.

13. The TKE_AND_EPSILON parameter is set to true which allows the turbulent intensity and eddy length scale values to fluctuate as the solution proceeds, using the result of step 12 as the initial guess with all relaxation parameters set to 0.2.

14. The amount of relaxation is reduced by setting the parameters to 0.5 and resuming the solution with the output of step 13 as the initial guess.

15. The turbulent solution continues with all relaxation parameters set to 0.75 and the initial guess for this portion of the solution is the result of step 14.

16. The turbulent solution with fluid, temperature, and CHT sub-models is acquired by using the result of step 15 as the initial guess with all relaxation parameters set to 1.0. This solution is saved in the interest of having an appropriate restart file in the event that complications arise in achieving the next phase of the solution which includes the combustion model. The enabling of the combustion model introduces a dramatic change in the energy distribution throughout the computational domain.

17. The combustion model sub-model is enabled and all relaxation parameters are set to 0.2. The initial guess for this segment of the solution is the turbulent solution obtained at the conclusion of step 16.

18. The amount of relaxation is reduced as the relaxation parameters are set to 0.5 and the solution is resumed using the output file from step 17 as the initial guess.

19. The reacting portion of the solution continues with an initial guess that is based on the result of step 18 with all relaxation parameters set to 0.75.

20. The final solution of the reacting flow with fluid, temperature, CHT, and turbulence sub-models is obtained by using the result of step 19 as an initial guess with all relaxation parameters set to 1.0. This final result is saved in the interest of having an appropriate restart file in the event of the solution becoming corrupted in formulating the next phase of the solution that enables the radiation sub-model.

21. The radiation model is enabled and all relaxation parameters are set to 0.2. The solution obtained in step 20 becomes the initial guess for this phase.

22. The radiation solution continues by using the result of step 21 as the initial guess and decreasing the amount of relaxation by setting all relaxation parameters to 0.5.

23. All relaxation parameters are then set to 0.75 and the solution is resumed using the result of step 22 as the initial guess.

The final solution of the problem that includes the radiation, fluids, temperature, CHT, turbulence, and combustion sub-models is obtained by taking the solution to step 23 and using it as the initial guess for all relaxation parameters set to 1.0. The converged form of this solution is the final result of the windshield as the time of interest.

## *5.6 Results*

Using the adiabatic flame temperature as an upper bound, the temperatures within the combustion zone have been assessed. The maximum gas temperatures predicted in

the simulations have been in the range of 1800 to 1850K which is below the adiabatic flame temperature of 2075K calculated in Chapter 2. The maximum temperatures were observed within the center of the combustion zone, which agrees with the characteristics of the fire plume in that the maximum gas temperatures are along the centerline of the plume.

A temperature profile has been constructed along the inner and outer surfaces of the windshield. The particular results shown below are 30 seconds into the simulation with the fire located 0.32m from the base of the windshield and the heat release rate is 175kW. Twenty-five nodal locations have been selected. The outer surface points correspond to the yz-plane with x = 3, where the exterior windshield surface is blocked-off in the fluid domain. The inner surface points correspond to the yz-plane with x = 12, where the interior windshield surface is blocked-off in the fluid domain.

Outer Windshield Surface
Temperature Distribution (K)

| | y node | | | | |
|---|---|---|---|---|---|
| z node | 1 | 6 | 11 | 16 | 21 |
| 4 | 419 | 411 | 406 | 395 | 394 |
| 10 | 370 | 343 | 344 | 326 | 318 |
| 16 | 347 | 326 | 337 | 330 | 316 |
| 22 | 320 | 308 | 319 | 325 | 320 |
| 28 | 300 | 298 | 301 | 307 | 310 |

**Table 11 -- Windshield exterior surface temperature distribution at 30 seconds with a 175kW fire located 0.32m from the base of the windshield.**

Inner Windshield Surface
Temperature Distribution (K)

| | y node | | | | |
|---|---|---|---|---|---|
| z node | 1 | 6 | 11 | 16 | 21 |
| 4 | 348 | 302 | 298 | 298 | 298 |
| 10 | 338 | 301 | 299 | 298 | 298 |
| 16 | 317 | 300 | 298 | 298 | 298 |
| 22 | 309 | 298 | 298 | 298 | 298 |
| 28 | 298 | 298 | 298 | 298 | 298 |

**Table 12 -- Windshield interior surface temperature distribution at 30 seconds with a 175kW fire located 0.32m from the base of the windshield.**

## *5.7 Conclusions*

The application of the parametric TASCflow model of the windshield problem has provided a methodology for modeling additional scenarios. Therefore, an experimental heat release rate, or a design heat release rate more suitable for engine compartment fires, can be used within the framework that has been established for the windshield problem.

## 6.0 Evaluation of Windshield Model

At this time a proper evaluation of the windshield model cannot be made because the information necessary to make comparisons between model output and the results of controlled fire experiments are unavailable. Once such data becomes available, a new model solution must be developed using geometry and boundary conditions that are appropriate for and consistent with the conditions under which such controlled fire experiments were conducted. Once a converged solution for the specific scenario was achieved, comparisons could then be made to the available experimental measurements. However, a framework has been established for developing additional fire simulations and comparing the output to fire dynamics concepts as well as experimental results, if available.

# 7.0 Conclusions and Recommendations

The windshield model has been examined using both theoretical and analytical means. The results of the research are summarized below. Recommendations for future work based on this research are also presented.

## *7.1 Conclusions*

1. An investigation has been conducted into the development of an engine compartment design fire scenario with an emphasis on an appropriate design heat release rate. An inspection of post-collision vehicle fire statistics was supplemented with common ignition source information to determine that the engine compartment was the most common origin of post-collision vehicle fires and the ignition source is presumably electrical in nature. The design fire scenario was determined to be an engine compartment fire producing a heat flux and fluid flow exposed to the windshield of the post-collision vehicle. This scenario is appropriate as it represents a common mode of fire spread from engine to passenger compartments for fires originating in the engine compartment, as observed in test burns of post-collision vehicles conducted by General Motors at Factory Mutual. A design heat release rate for the engine compartment design fire has been selected after a reviewing motor vehicle fire testing literature. It was found that most motor vehicle fire testing is conducted for the purpose of only measuring gas temperatures for parking garage design. The heat release rate, in these instances is not significant for designing parking garages, however, the heat release rate is an essential quantity in fire modeling. However, a recent series of motor vehicle fire tests by the ProfilARBED research group examined the burning behavior of several cars manufactured in the 90's and measured the heat release rates. A design heat release rate

curve was constructed from a compilation of heat release rate curves for the various test vehicles. The curve developed by the ProfilARBED research group has been selected as a design heat release rate for an engine compartment design fire scenario because it is based on vehicle testing involving cars manufactured in the 90's.

2. In lieu of the results of Project B.3, a framework has been established for using concepts from fire dynamics to make some general comparisons to the fire being modeled. These comparisons involve making considerations for the adiabatic flame temperature and flame height as well as the centerline temperature and velocity in the resulting fire plume.

3. An appropriately parameterized combustion model using turbulent, reacting, multi-component fluid flow, with heat transfer by convection, conduction, and radiation has been developed for the windshield problem using TASCflow Computational Fluid Dynamics software. The parametric nature of the windshield problem has been considered by establishing an input file that governs the creation of all associated grid objects in terms of geometry and nodal distributions. A multi-grid approach has been utilized in formulating the TASCflow model in order to account for the slope of the windshield as well as the rectangular nature of the combustion region above the hood.

4. The parametric combustion model has been applied to engine compartment fire scenarios consistent with the General Motors vehicle fire testing using the selected design heat release rate curve. The results available at this time show reasonable agreement to

comparisons made to general correlations and calculation methods from fire dynamics. In particular, the gas temperatures predicted by the model are less than the adiabatic flame temperature calculated for the fuel. The maximum gas temperatures predicted by the model occur along the centerline of the resulting thermal plume which is consistent with plume theory from fire dynamics.

## *7.2 Recommendations for Future Work*

The validation of the framework for modeling the interaction between an engine compartment fire and the windshield of a post-collision vehicle is necessary in order to determine if the approach is appropriate. The validation process would involve making comparisons between experimental data sets of controlled fire experiments and simulations of such experiments using the methodology presented in this work. The experimental data sets should provide sufficient detail for making meaningful comparisons to the CFD simulations. The following environmental variables could provide information for validating simulations;

- Heat release rate
- Mean flame height
- Gas temperatures
- Windshield surface temperatures
- Windshield surface heat fluxes
- Gas velocities

The location of the instrumentation recording the environmental variables described above will be essential to the validation processes. Monitor points can be

placed within the CFD model to monitor the environmental variables in order to make a comparison to the experimental results. Criteria must be established for the quality of the comparisons of the model and experimental results, eg within 20% of the experimental measurements.

The modeling procedure could be applied to windshield design in the interest of evaluating windshield designs. Different materials used in candidate designs to assess the resulting impact on fire safety.

## References


[1] Schleich, J., L. Cajot, M. Pierre, and M. Brasseur. "Development of Design Rules for Steel Structures Subject to Natural Fires in Closed Car Parks: Final Report.", ProfilARBED Research Center, Luxembourg, CEC Agreement 7210-SA/211/318/518/620/933, 1996.

[2] Bauccio, M. ASM Engineered Materials Reference Book, 2nd Ed. ASM International, 1994.

[3] Browne, A.L. "Windshield Impact Response: An Empirical Study of the Standard Three-Ply Construction", General Motors Research Labs, 1989.

[4] Callister, W.D. "Materials Science and Engineering: An Introduction", John Wiley & Sons, pp. 420-423, 1994.

[5] Conley, C. “Deadlier Than You Think”. *NFPA Journal*, 90(5):79,1996.

[6] Cox, G. *Combustion Fundamentals of Fire*, Harcourt Brace & Co., London, 1995.

[7] Drysdale, D. “An Introduction to Fire Dynamics”, John Wiley and Sons, New York, 1985.

[8] Gatlin, C.I. and N.B. Johnson. "Prevention of Electrical System Ignition of Automotive Crash Fire" Department of Transportation Report AVSER 2310/20-70-7, Washington, DC, 1970.

[9] Gold, Mark. Information provided in personal communication on Saflex brand automotive windshields, Solutia, Inc., 1997.

[10] Gustin, B. "New Fire Tactics for New-Car Fires". *Fire Engineering*, 149(4):43, 1996.

[11] Mangs, J. and Keski-Rahkonen. "Characterization of the Fire Behavior of a Burning Passenger Car. Part 1: Car Fire Experiements.", *Fire Safety Journal*, Vol. No. 23, 1994.

[12] *Motor Vehicle Safety Standard 205*, 1985.

[13] Patrick, L.M. "Glazing for Motor Vehicles -1995", Wayne State University, 1995.

[14] Roberts, C. "Motor Vehicle Fire Analysis". *Automotive Engineering and Litigation*, 4, 1991.

[15] SFPE Handbook of Fire Protection Engineering, 2nd Edition, 1995.

[16] Shipp, M. and M. Spearpoint. "Measurements of the Severity of Fires Involving Private Motor Vehicles". *Fire and Materials*, Vol. 19, 143-151, 1995.

[17] *TASCflow3D v2.4 Theory Documentation*, Advanced Scientific Computing, Waterloo, Ontario, Canada, 1995.

[18] *Prototype Capability FV Radiation Model*, AEA/CFX-TASCflow, Advanced Scientific Computing, Waterloo, Ontario, Canada, 1998.

# Appendix A – TASCflow Model Files

**Parametric Input File**

input.txt

**Above_Eng Object**

name.lun
above_eng.gdf
above_eng.cdf
above_eng.sdf
above_eng.idf

**Above_Fire Object**

name.lun
above_fire.gdf
above_fire.cdf
above_fire.sdf
above_fire.idf

**Fire Object**

name.lun
fire.gdf
fire.cdf
fire.sdf
fire.idf

**Top_Front Object**

name.lun
top_front.gdf
top_front.cdf
top_front.sdf
top_front.idf

**Top_Mid Object**

name.lun
top_mid.gdf
top_mid.cdf
top_mid.sdf
top_mid.idf

**Wind Object**

name.lun
wind.gdf
wind.cdf
wind.sdf
wind.idf

**Property Files**

propf.f
propq.f
propt.f
bcdtrn.f

**Parameter File**

top_front.prm

*Input.txt*

```
!************************************************
!*  This file contains the variable definitions   *
!*  for all parts of the vehicle assembly.        *
!*                                    *
!*  USED FOR GRID REFINEMENT (NODES REDEFINED)    *
!*                                    *
!*  Last modified: 28 Oct 1998                    *
!*  By: James Ierardi for General Motors Project. *
!************************************************

VARIABLE DEFINITIONS

!*************************
!*  BEGIN USER VARIABLES  *
!*************************
!-------------------------------------------------------------------------
--------
! Dimensions in meters
!-------------------------------------------------------------------------
--------
! Engine
x_eng = 0.1       ! length of engine compartment
y_eng = 0.844           ! half-width of engine compartment (windshield,
etc.)
z_eng = 1.0       ! height of engine compartment
z_airb = 0.25           ! height of air below engine compartment

! Fire
x_fire = 0.8            ! distance to fire facing windshield

! Windshield
x_airout = 0.05         ! Thickness of air extension outside car
x_glass1 = 0.00635      ! Thickness of outer glass layer
x_pvb = 0.000762  ! Thickness of pvb inter-layer
x_glass2 = 0.00635      ! Thickness of inner glass layer
x_airin = 0.1           ! Thickness of air extension inside car
x_wind = 0.8382         ! Horizontal projection of windshield
z_wind = 0.8966         ! Vertical projection of windshield

! Fluid Region Extensions
z_aira = 0.5   !Air Above Vehicle height
y_airs = 0.5   !Air Beside Vehicle width

!-------------------------------------------------------------------------
--------
! Number of nodes (adjusted per readme.txt in Grid_Refine directory)
!-------------------------------------------------------------------------
--------
nodex_fire = 24
nodex_front = 16  ! Eng_Front, Air_Above, and Top_Front
nodex_airout = 3  ! nodes for thickness of air extension outside car
nodex_glass1 = 3  ! nodes for thickness of outer glass layer
nodex_pvb = 3           ! nodes for thickness of pvb inter-layer
nodex_glass2 = 3  ! nodes for thickness of inner glass layer
```

```
nodex_airin = 3         ! nodes for thickness of air extension inside
car
nodey_car = 24          ! Engf, Engb, Pass_BF, Pass_BB, Aireng, Wind,
Pass_TB
nodey_air = 10
nodez_wind = 34         ! Aireng, Wind, Pass_TB
nodez_top = 10          ! Top_Front, Top_Mid, Top_Back
! Fire Specification
nodex_fire1_s = 8 ! Start node of FIRE1 length, should be greater than
1
nodey_fire1_e = 8 ! End node of FIRE1&FIRE2 width, should be <
nodey_car
nodez_fire1_e = 9 ! End node of FIRE1 height, should be < nodez_wind
nodex_fire2_e = 4 ! End node of FIRE2 length, should be < nodex_airout
nodez_fire2_e = 9 ! End node of FIRE2 height, should be < nodez_wind

!-------------------------------------------------------------------
--------

! Windshield
nodey_wind = nodey_car  ! nodes for width of windshield
!Fluid Extension for Windshield
nodey_ext_sm = nodey_air

! Air Above Engine
nodex_aireng = nodex_front
nodey_aireng = nodey_car
nodez_aireng = nodez_wind

! Fluid Extension for Air Above Engine
nodey_ext_sf = nodey_air

!-------------------------------------------------------------------
--------
! Expansion factors
!-------------------------------------------------------------------
--------

! Windshield
r_x_airout = .25  ! expansion factor for outside air nodes in x dir.
r_x_glass1 = 1.         ! expansion factor for outside glass nodes in x
dir.
r_x_pvb     = 1.        ! expansion factor for pvb nodes in x dir.
r_x_glass2 = 1.         ! expansion factor for inside glass nodes in x
dir.
r_x_airin = 4.          ! expansion factor for inside air nodes in x
dir.
r_y_wind = 1.0
r_z_wind = 1.0          ! expansion factor for windshield nodes in z
dir.
! Fluid extension
r_y_ext_sm = 1.

! Fire
r_x_fire = 1.
r_y_fire = 1.
r_z_fire = 4.
```

```
! Fluid Extension Above Fire
r_x_above_fire = 1.
r_y_above_fire = 1.
r_z_above_fire = 1.

! Air above windshield
r_x_aireng = 1.
r_y_aireng = 1.
r_z_aireng = 1.

! Fluid Extension of Air Above Windshield
r_x_ext_sf = 1.
r_y_ext_sf = 1.
r_z_ext_sf = 1.

! Fluid Extenion Top_Front
r_x_air_tf = 1
r_y_air_tf = 1
r_z_air_tf = 1

! Fluid Extenion Top_Mid
r_x_air_tm = 1
r_y_air_tm = 1
r_z_air_tm = 1

!-------------------------------------------------------------------------
--------
! Starting points
!-------------------------------------------------------------------------
--------

! Windshield
wind_wsb_x = 0.
wind_wsb_y = 0.
wind_wsb_z = 0.

! Fire
fire_wsb_x = 0.
fire_wsb_y = 0.
fire_wsb_z = 0.

! Air Above Engine
aireng_wsb_x = 0.
aireng_wsb_y = 0.
aireng_wsb_z = 0.

! Air Above Fire
above_fire_wsb_x = 0.
above_fire_wsb_y = 0.
above_fire_wsb_z = 0.

! Fluid Extension Top_Front
air_tf_wsb_x = 0.
air_tf_wsb_y = 0.
air_tf_wsb_z = 0.
```

```
! Fluid Extension Top_Mid
air_tm_wsb_x = 0.
air_tm_wsb_y = 0.
air_tm_wsb_z = 0.

!************************
!*  END USER VARIABLES  *
!************************
!-------------------------------------------------------------------------
--------
! Geometry
!-------------------------------------------------------------------------
--------

! Fluid Extension Top_Front
x_air_tf = x_eng - x_airout + x_wind
y_air_tf = y_eng + y_airs

! Fluid Extension Top_Mid
x_air_tm = x_airout + x_glass1 + x_pvb + x_glass2 + x_airin
y_air_tm = y_eng + y_airs

! Fluid Extension Above_Fire
x_above_fire = x_fire
y_above_fire = y_eng + y_airs

!-------------------------------------------------------------------------
--------
! Start and End Point Definitions
!-------------------------------------------------------------------------
--------

! Windshield
x_airout_sb = wind_wsb_x      ! starting and end points along bottom of
x_glass1_sb = wind_wsb_x + x_airout        ! the windshield
x_pvb_sb = wind_wsb_x + x_airout + x_glass1
x_pvb_eb = wind_wsb_x + x_airout + x_glass1 + x_pvb
x_glass2_eb = wind_wsb_x + x_airout + x_glass1 + x_pvb + x_glass2
x_airin_eb = wind_wsb_x + x_airout + x_glass1 + x_pvb + x_glass2 +
x_airin

x_airout_st = wind_wsb_x + x_wind   ! starting and end points along top
of
x_glass1_st = wind_wsb_x + x_airout + x_wind       ! the windshield
x_pvb_st = wind_wsb_x + x_airout + x_glass1 + x_wind
x_pvb_et = wind_wsb_x + x_airout + x_glass1 + x_pvb + x_wind
x_glass2_et = wind_wsb_x + x_airout + x_glass1 + x_pvb + x_glass2 +
x_wind
x_airin_et = wind_wsb_x + x_airout + x_glass1 + x_pvb + x_glass2 +
x_airin + x_wind

y_wind_s = wind_wsb_y
y_wind_e = wind_wsb_y + y_eng
z_wind_s = wind_wsb_z
z_wind_e = wind_wsb_z + z_wind

! Fire
```

```
x_fire_s = fire_wsb_x
x_fire_e = fire_wsb_x + x_fire
y_fire_s = fire_wsb_y
y_fire_e = fire_wsb_y + y_eng
z_fire_s = fire_wsb_z
z_fire_e = fire_wsb_z + z_wind

!Fluid extension Above_Fire
x_above_fire_s = above_fire_wsb_x
x_above_fire_e = above_fire_wsb_x + x_above_fire
y_above_fire_s = above_fire_wsb_y
y_above_fire_e = above_fire_wsb_y + y_above_fire
z_above_fire_s = above_fire_wsb_z
z_above_fire_e = above_fire_wsb_z + z_aira

! Air above engine compartment
x_aireng_sb = aireng_wsb_x
x_aireng_eb = aireng_wsb_x + x_eng - x_airout
x_aireng_st = aireng_wsb_x
x_aireng_et = aireng_wsb_x + x_eng - x_airout + x_wind
y_aireng_s = aireng_wsb_y
y_aireng_e = aireng_wsb_y + y_eng
z_aireng_s = aireng_wsb_z
z_aireng_e = aireng_wsb_z + z_wind
! Fluid Extension

! Fluid Extension Top_Front
x_air_tf_s = air_tf_wsb_x
x_air_tf_e = air_tf_wsb_x + x_air_tf
y_air_tf_s = air_tf_wsb_y
y_air_tf_e = air_tf_wsb_y + y_air_tf
z_air_tf_s = air_tf_wsb_z
z_air_tf_e = air_tf_wsb_z + z_aira

! Fluid Extension Top_Mid
x_air_tm_s = air_tm_wsb_x
x_air_tm_e = air_tm_wsb_x + x_air_tm
y_air_tm_s = air_tm_wsb_y
y_air_tm_e = air_tm_wsb_y + y_air_tm
z_air_tm_s = air_tm_wsb_z
z_air_tm_e = air_tm_wsb_z + z_aira

!-------------------------------------------------------------------------
--------
! Total node definitions for length, width, and height
!-------------------------------------------------------------------------
--------

! Windshield nodes
nlw1 = nodex_airout
nlw2 = nodex_airout + nodex_glass1
nlw3 = nodex_airout + nodex_glass1 + nodex_pvb
nlw4 = nodex_airout + nodex_glass1 + nodex_pvb + nodex_glass2
nlw5 = nodex_airout + nodex_glass1 + nodex_pvb + nodex_glass2 +
nodex_airin
nww1 = nodey_car
nww2 = nodey_car + nodey_air
```

```
nhw1 = nodez_wind

! Fire
nlf1 = nodex_fire
nwf1 = nodey_car
nwf2 = nodey_car + nodey_air
nhf1 = nodez_wind

! Above_Fire
nlaf1 = nodex_fire
nwaf1 = nodey_car
nwaf2 = nodey_car + nodey_air
nhaf1 = nodez_top

! Air Above Engine nodes
nlae1 = nodex_front
nwae1 = nodey_car
nwae2 = nodey_car + nodey_air
nhae1 = nodez_wind

! Fluid Extension Top_Front
nlatf1 = nodex_front
nwatf1 = nodey_car + nodey_air
nhatf1 = nodez_top

! Fluid Extension Top_Mid
nlatm1 = nodex_airout + nodex_glass1 + nodex_pvb + nodex_glass2 +
nodex_airin
nwatm1 = nodey_car + nodey_air
nhatm1 = nodez_top

!End
```

*Above_Eng Object*

Name.lun

```
problem above_eng.
include #/batch.lun
include #/systemu.lun
```

Above_eng.gdf

```
!*****************************************************
!* Air above engine compartment template.            *
!*                                                   *
!* Last modified: 2 Feb 1998                         *
!* By: James Ierardi for the General Motors Project. *
!*****************************************************

include ../input.txt

POINT
aireng_wsb (x_aireng_sb,y_aireng_s,z_aireng_s)
aireng_esb (x_aireng_eb,y_aireng_s,z_aireng_s)
aireng_wst (x_aireng_st,y_aireng_s,z_aireng_e)
aireng_est (x_aireng_et,y_aireng_s,z_aireng_e)
aireng_wnb (x_aireng_sb,y_aireng_e,z_aireng_s)
aireng_enb (x_aireng_eb,y_aireng_e,z_aireng_s)
aireng_wnt (x_aireng_st,y_aireng_e,z_aireng_e)
aireng_ent (x_aireng_et,y_aireng_e,z_aireng_e)
ext_sf_wnb (x_aireng_sb,y_aireng_e+y_airs,z_aireng_s)
ext_sf_enb (x_aireng_eb,y_aireng_e+y_airs,z_aireng_s)
ext_sf_wnt (x_aireng_st,y_aireng_e+y_airs,z_aireng_e)
ext_sf_ent (x_aireng_et,y_aireng_e+y_airs,z_aireng_e)

Curve aireng_sb Linear
aireng_wsb; aireng_esb

Curve aireng_nb Linear
aireng_wnb; aireng_enb

Curve ext_sf_nb Linear
ext_sf_wnb; ext_sf_enb

Curve aireng_st Linear
aireng_wst; aireng_est

Curve aireng_nt Linear
aireng_wnt; aireng_ent

Curve ext_sf_nt Linear
ext_sf_wnt; ext_sf_ent

Curve aireng_wb Linear
aireng_wsb; aireng_wnb; ext_sf_wnb

Curve aireng_eb Linear
aireng_esb; aireng_enb; ext_sf_enb

Curve aireng_wt Linear
aireng_wst; aireng_wnt; ext_sf_wnt

Curve aireng_et Linear
aireng_est; aireng_ent; ext_sf_ent

Curve aireng_ws Linear
```

```
aireng_wsb; aireng_wst

Curve aireng_es Linear
aireng_esb; aireng_est

Curve aireng_wn Linear
aireng_wnb; aireng_wnt

Curve ext_sf_wn Linear
ext_sf_wnb; ext_sf_wnt

Curve aireng_en Linear
aireng_enb; aireng_ent

Curve ext_sf_en Linear
ext_sf_enb; ext_sf_ent

! South Surface of whole thing
Surface Aireng_S By Curves Bilinear
aireng_sb; aireng_es; -aireng_st; -aireng_ws

! North Surface of whole thing
Surface Aireng_N By Curves Bilinear
ext_sf_nb; ext_sf_en; -ext_sf_nt; -ext_sf_wn

! Bottom Surface of whole thing
Surface Aireng_B By Curves Bilinear
aireng_sb; aireng_eb; -ext_sf_nb; -aireng_wb

! Top Surface of whole thing
Surface Aireng_T By Curves Bilinear
aireng_st; aireng_et; -ext_sf_nt; -aireng_wt

! West Surface of whole thing
Surface Aireng_W By Curves Bilinear
aireng_wb; ext_sf_wn; -aireng_wt; -aireng_ws

! East Surface of whole thing
Surface Aireng_E By Curves Bilinear
aireng_eb; ext_sf_en; -aireng_et; -aireng_es

! End
```

Above_eng.cdf

```
Grid Dimensions
ID=nlae1
JD=nwae2
KD=nhae1
---------------------------------------------------
Variable Definitions
---------------------------------------------------
Vertex Attachments
aireng_wsb  (1,1,1)
aireng_esb  (nlae1,1,1)
aireng_wnb  (1,nwae1,1)
aireng_enb  (nlae1,nwae1,1)
aireng_wst  (1,1,nhae1)
aireng_est  (nlae1,1,nhae1)
aireng_wnt  (1,nwae1,nhae1)
aireng_ent  (nlae1,nwae1,nhae1)
ext_sf_wnb  (1,nwae2,1)
ext_sf_enb  (nlae1,nwae2,1)
ext_sf_wnt  (1,nwae2,nhae1)
ext_sf_ent  (nlae1,nwae2,nhae1)
---------------------------------------------------
Node Distributions
Default=1
!start-vertex     end-vertex  expansion-factor
aireng_wsb  aireng_esb  100 @ r r_x_aireng
aireng_wnb  aireng_enb  100 @ r r_x_aireng
aireng_wst  aireng_est  100 @ r r_x_aireng
aireng_wnt  aireng_ent  100 @ r r_x_aireng
aireng_wsb  aireng_wnb  100 @ r r_y_aireng
aireng_esb  aireng_enb  100 @ r r_y_aireng
aireng_wst  aireng_wnt  100 @ r r_y_aireng
aireng_est  aireng_ent  100 @ r r_y_aireng
aireng_wsb  aireng_wst  100 @ r r_z_aireng
aireng_esb  aireng_est  100 @ r r_z_aireng
aireng_wnb  aireng_wnt  100 @ r r_z_aireng
aireng_enb  aireng_ent  100 @ r r_z_aireng
ext_sf_wnb  ext_sf_enb  100 @ r r_x_ext_sf
ext_sf_wnt  ext_sf_ent  100 @ r r_x_ext_sf
aireng_wnb  ext_sf_wnb  100 @ r r_y_ext_sf
aireng_enb  ext_sf_enb  100 @ r r_y_ext_sf
aireng_wnt  ext_sf_wnt  100 @ r r_y_ext_sf
aireng_ent  ext_sf_ent  100 @ r r_y_ext_sf
ext_sf_wnb  ext_sf_wnt  100 @ r r_z_ext_sf
ext_sf_enb  ext_sf_ent  100 @ r r_z_ext_sf
! End
```

Above_eng.sdf

```
Region Aireng_S
-----------------------------------------------------------------
Region Aireng_N
-----------------------------------------------------------------
Region Aireng_B
-----------------------------------------------------------------
Region Aireng_T
-----------------------------------------------------------------
Region Aireng_W
-----------------------------------------------------------------
Region Aireng_E
-----------------------------------------------------------------
```

Above_eng.idf

```
!  Engine Compartment Template
!  Jay Ierardi
Region aireng elliptic
errxyz=4e-7
itmax=320
aireng_wsb,ext_sf_ent
```

*Above_Fire Object*

Name.lun

```
problem above_fire.
include #/batch.lun
include #/systemu.lun
```

Above_fire.gdf

```
!*****************************************************
!* Passenger compartment template (bottom back).     *
!*                                                   *
!* Last modified: 2 Feb 1998                         *
!* By: James Ierardi for the General Motors Project. *
!*****************************************************

include ../input.txt

POINT
above_fire_wsb (x_above_fire_s,y_above_fire_s,z_above_fire_s)
above_fire_esb (x_above_fire_e,y_above_fire_s,z_above_fire_s)
above_fire_wst (x_above_fire_s,y_above_fire_s,z_above_fire_e)
above_fire_est (x_above_fire_e,y_above_fire_s,z_above_fire_e)
above_fire_wnb (x_above_fire_s,y_above_fire_e,z_above_fire_s)
above_fire_enb (x_above_fire_e,y_above_fire_e,z_above_fire_s)
above_fire_wnt (x_above_fire_s,y_above_fire_e,z_above_fire_e)
above_fire_ent (x_above_fire_e,y_above_fire_e,z_above_fire_e)

Curve above_fire_sb Linear
above_fire_wsb; above_fire_esb

Curve above_fire_nb Linear
above_fire_wnb; above_fire_enb

Curve above_fire_st Linear
above_fire_wst; above_fire_est

Curve above_fire_nt Linear
above_fire_wnt; above_fire_ent

Curve above_fire_wb Linear
above_fire_wsb; above_fire_wnb

Curve above_fire_eb Linear
above_fire_esb; above_fire_enb

Curve above_fire_wt Linear
above_fire_wst; above_fire_wnt

Curve above_fire_et Linear
above_fire_est; above_fire_ent

Curve above_fire_ws Linear
above_fire_wsb; above_fire_wst

Curve above_fire_es Linear
above_fire_esb; above_fire_est

Curve above_fire_wn Linear
above_fire_wnb; above_fire_wnt

Curve above_fire_en Linear
above_fire_enb; above_fire_ent
```

```
! South Surface of whole thing
Surface above_fire_S By Curves Bilinear
above_fire_sb; above_fire_es; -above_fire_st; -above_fire_ws

! North Surface of whole thing
Surface above_fire_N By Curves Bilinear
above_fire_nb; above_fire_en; -above_fire_nt; -above_fire_wn

! Bottom Surface of whole thing
Surface above_fire_B By Curves Bilinear
above_fire_sb; above_fire_eb; -above_fire_nb; -above_fire_wb

! Top Surface of whole thing
Surface above_fire_T By Curves Bilinear
above_fire_st; above_fire_et; -above_fire_nt; -above_fire_wt

! West Surface of whole thing
Surface above_fire_W By Curves Bilinear
above_fire_wb; above_fire_wn; -above_fire_wt; -above_fire_ws

! East Surface of whole thing
Surface above_fire_E By Curves Bilinear
above_fire_eb; above_fire_en; -above_fire_et; -above_fire_es

! End
```

Above_fire.cdf

```
Grid Dimensions
ID=nlaf1
JD=nwaf2
KD=nhaf1
---------------------------------------------------
Variable Definitions
---------------------------------------------------
Vertex Attachments
above_fire_wsb    (1,1,1)
above_fire_esb    (nlaf1,1,1)
above_fire_wnb    (1,nwaf2,1)
above_fire_enb    (nlaf1,nwaf2,1)
above_fire_wst    (1,1,nhaf1)
above_fire_est    (nlaf1,1,nhaf1)
above_fire_wnt    (1,nwaf2,nhaf1)
above_fire_ent    (nlaf1,nwaf2,nhaf1)
---------------------------------------------------
Node Distributions
Default=1
!start-vertex     end-vertex  expansion-factor
above_fire_wsb    above_fire_esb    100 @ r r_x_above_fire
above_fire_wnb    above_fire_enb    100 @ r r_x_above_fire
above_fire_wst    above_fire_est    100 @ r r_x_above_fire
above_fire_wnt    above_fire_ent    100 @ r r_x_above_fire
above_fire_wsb    above_fire_wnb    100 @ r r_y_above_fire
above_fire_esb    above_fire_enb    100 @ r r_y_above_fire
above_fire_wst    above_fire_wnt    100 @ r r_y_above_fire
above_fire_est    above_fire_ent    100 @ r r_y_above_fire
above_fire_wsb    above_fire_wst    100 @ r r_z_above_fire
above_fire_esb    above_fire_est    100 @ r r_z_above_fire
above_fire_wnb    above_fire_wnt    100 @ r r_z_above_fire
above_fire_enb    above_fire_ent    100 @ r r_z_above_fire
! End
```

Above_fire.sdf

```
Region above_fire_S
-------------------------------------------------------------------
Region above_fire_N
-------------------------------------------------------------------
Region above_fire_B
-------------------------------------------------------------------
Region above_fire_T
-------------------------------------------------------------------
Region above_fire_W
-------------------------------------------------------------------
Region above_fire_E
-------------------------------------------------------------------
```

Above_fire.idf

```
!  Engine Compartment Template
!  Jay Ierardi
Region above_fire elliptic
errxyz=4e-7
itmax=320
above_fire_wsb,above_fire_ent
```

*Fire Object*

Name.lun

```
problem fire.
include #/batch.lun
include #/systemu.lun
```

Fire.gdf

```
!*****************************************************
!* Air above engine compartment template.            *
!*                                                   *
!* Last modified: 2 Feb 1998                         *
!* By: James Ierardi for the General Motors Project. *
!*****************************************************

include ../input.txt

POINT
fire_wsb (x_fire_s,y_fire_s,z_fire_s)
fire_esb (x_fire_e,y_fire_s,z_fire_s)
fire_wst (x_fire_s,y_fire_s,z_fire_e)
fire_est (x_fire_e,y_fire_s,z_fire_e)
fire_wnb (x_fire_s,y_fire_e,z_fire_s)
fire_enb (x_fire_e,y_fire_e,z_fire_s)
fire_wnt (x_fire_s,y_fire_e,z_fire_e)
fire_ent (x_fire_e,y_fire_e,z_fire_e)
ext_sf_wnb (x_fire_s,y_fire_e+y_airs,z_fire_s)
ext_sf_enb (x_fire_e,y_fire_e+y_airs,z_fire_s)
ext_sf_wnt (x_fire_s,y_fire_e+y_airs,z_fire_e)
ext_sf_ent (x_fire_e,y_fire_e+y_airs,z_fire_e)

Curve fire_sb Linear
fire_wsb; fire_esb

Curve fire_nb Linear
fire_wnb; fire_enb

Curve ext_sf_nb Linear
ext_sf_wnb; ext_sf_enb

Curve fire_st Linear
fire_wst; fire_est

Curve fire_nt Linear
fire_wnt; fire_ent

Curve ext_sf_nt Linear
ext_sf_wnt; ext_sf_ent

Curve fire_wb Linear
fire_wsb; fire_wnb; ext_sf_wnb

Curve fire_eb Linear
fire_esb; fire_enb; ext_sf_enb

Curve fire_wt Linear
fire_wst; fire_wnt; ext_sf_wnt

Curve fire_et Linear
fire_est; fire_ent; ext_sf_ent

Curve fire_ws Linear
```

```
fire_wsb; fire_wst

Curve fire_es Linear
fire_esb; fire_est

Curve fire_wn Linear
fire_wnb; fire_wnt

Curve ext_sf_wn Linear
ext_sf_wnb; ext_sf_wnt

Curve fire_en Linear
fire_enb; fire_ent

Curve ext_sf_en Linear
ext_sf_enb; ext_sf_ent

! South Surface of whole thing
Surface Fire_S By Curves Bilinear
fire_sb; fire_es; -fire_st; -fire_ws

! North Surface of whole thing
Surface Fire_N By Curves Bilinear
ext_sf_nb; ext_sf_en; -ext_sf_nt; -ext_sf_wn

! Bottom Surface of whole thing
Surface Fire_B By Curves Bilinear
fire_sb; fire_eb; -ext_sf_nb; -fire_wb

! Top Surface of whole thing
Surface Fire_T By Curves Bilinear
fire_st; fire_et; -ext_sf_nt; -fire_wt

! West Surface of whole thing
Surface Fire_W By Curves Bilinear
fire_wb; ext_sf_wn; -fire_wt; -fire_ws

! East Surface of whole thing
Surface Fire_E By Curves Bilinear
fire_eb; ext_sf_en; -fire_et; -fire_es

! End
```

Fire.cdf

```
Grid Dimensions
ID=nlf1
JD=nwf2
KD=nhf1
---------------------------------------------------
Variable Definitions
---------------------------------------------------
Vertex Attachments
fire_wsb    (1,1,1)
fire_esb    (nlf1,1,1)
fire_wnb    (1,nwf1,1)
fire_enb    (nlf1,nwf1,1)
fire_wst    (1,1,nhf1)
fire_est    (nlf1,1,nhf1)
fire_wnt    (1,nwf1,nhf1)
fire_ent    (nlf1,nwf1,nhf1)
ext_sf_wnb  (1,nwf2,1)
ext_sf_enb  (nlf1,nwf2,1)
ext_sf_wnt  (1,nwf2,nhf1)
ext_sf_ent  (nlf1,nwf2,nhf1)
---------------------------------------------------
Node Distributions
Default=1
!start-vertex     end-vertex  expansion-factor
fire_wsb    fire_esb    100 @ r r_x_fire
fire_wnb    fire_enb    100 @ r r_x_fire
fire_wst    fire_est    100 @ r r_x_fire
fire_wnt    fire_ent    100 @ r r_x_fire
fire_wsb    fire_wnb    100 @ r r_y_fire
fire_esb    fire_enb    100 @ r r_y_fire
fire_wst    fire_wnt    100 @ r r_y_fire
fire_est    fire_ent    100 @ r r_y_fire
fire_wsb    fire_wst    100 @ r r_z_fire
fire_esb    fire_est    100 @ r r_z_fire
fire_wnb    fire_wnt    100 @ r r_z_fire
fire_enb    fire_ent    100 @ r r_z_fire
ext_sf_wnb  ext_sf_enb  100 @ r r_x_ext_sf
ext_sf_wnt  ext_sf_ent  100 @ r r_x_ext_sf
fire_wnb    ext_sf_wnb  100 @ r r_y_ext_sf
fire_enb    ext_sf_enb  100 @ r r_y_ext_sf
fire_wnt    ext_sf_wnt  100 @ r r_y_ext_sf
fire_ent    ext_sf_ent  100 @ r r_y_ext_sf
ext_sf_wnb  ext_sf_wnt  100 @ r r_z_ext_sf
ext_sf_enb  ext_sf_ent  100 @ r r_z_ext_sf
! End
```

Fire.sdf

```
Region Fire_S
-----------------------------------------------------------------
Region Fire_N
-----------------------------------------------------------------
Region Fire_B
-----------------------------------------------------------------
Region Fire_T
-----------------------------------------------------------------
Region Fire_W
-----------------------------------------------------------------
Region Fire_E
-----------------------------------------------------------------
```

Fire.idf

```
!  Fire Object Template
!  Jay Ierardi
Region fire elliptic
errxyz=4e-7
itmax=320
fire_wsb,ext_sf_ent
```

*Top_Front Object*

Name.lun

```
problem top_front.
include #/batch.lun
include #/systemu.lun
```

Top_front.gdf

```
!*****************************************************
!* Passenger compartment template (bottom back).     *
!*                                                   *
!* Last modified: 2 Feb 1998                         *
!* By: James Ierardi for the General Motors Project. *
!*****************************************************

include ../input.txt

POINT
air_tf_wsb (x_air_tf_s,y_air_tf_s,z_air_tf_s)
air_tf_esb (x_air_tf_e,y_air_tf_s,z_air_tf_s)
air_tf_wst (x_air_tf_s,y_air_tf_s,z_air_tf_e)
air_tf_est (x_air_tf_e,y_air_tf_s,z_air_tf_e)
air_tf_wnb (x_air_tf_s,y_air_tf_e,z_air_tf_s)
air_tf_enb (x_air_tf_e,y_air_tf_e,z_air_tf_s)
air_tf_wnt (x_air_tf_s,y_air_tf_e,z_air_tf_e)
air_tf_ent (x_air_tf_e,y_air_tf_e,z_air_tf_e)

Curve air_tf_sb Linear
air_tf_wsb; air_tf_esb

Curve air_tf_nb Linear
air_tf_wnb; air_tf_enb

Curve air_tf_st Linear
air_tf_wst; air_tf_est

Curve air_tf_nt Linear
air_tf_wnt; air_tf_ent

Curve air_tf_wb Linear
air_tf_wsb; air_tf_wnb

Curve air_tf_eb Linear
air_tf_esb; air_tf_enb

Curve air_tf_wt Linear
air_tf_wst; air_tf_wnt

Curve air_tf_et Linear
air_tf_est; air_tf_ent

Curve air_tf_ws Linear
air_tf_wsb; air_tf_wst

Curve air_tf_es Linear
air_tf_esb; air_tf_est

Curve air_tf_wn Linear
air_tf_wnb; air_tf_wnt

Curve air_tf_en Linear
air_tf_enb; air_tf_ent
```

```
! South Surface of whole thing
Surface air_tf_S By Curves Bilinear
air_tf_sb; air_tf_es; -air_tf_st; -air_tf_ws

! North Surface of whole thing
Surface air_tf_N By Curves Bilinear
air_tf_nb; air_tf_en; -air_tf_nt; -air_tf_wn

! Bottom Surface of whole thing
Surface air_tf_B By Curves Bilinear
air_tf_sb; air_tf_eb; -air_tf_nb; -air_tf_wb

! Top Surface of whole thing
Surface air_tf_T By Curves Bilinear
air_tf_st; air_tf_et; -air_tf_nt; -air_tf_wt

! West Surface of whole thing
Surface air_tf_W By Curves Bilinear
air_tf_wb; air_tf_wn; -air_tf_wt; -air_tf_ws

! East Surface of whole thing
Surface air_tf_E By Curves Bilinear
air_tf_eb; air_tf_en; -air_tf_et; -air_tf_es

! End
```

Top_front.cdf

```
Grid Dimensions
ID=nlatf1
JD=nwatf1
KD=nhatf1
---------------------------------------------------
Variable Definitions
---------------------------------------------------
Vertex Attachments
air_tf_wsb  (1,1,1)
air_tf_esb  (nlatf1,1,1)
air_tf_wnb  (1,nwatf1,1)
air_tf_enb  (nlatf1,nwatf1,1)
air_tf_wst  (1,1,nhatf1)
air_tf_est  (nlatf1,1,nhatf1)
air_tf_wnt  (1,nwatf1,nhatf1)
air_tf_ent  (nlatf1,nwatf1,nhatf1)
---------------------------------------------------
Node Distributions
Default=1
!start-vertex     end-vertex  expansion-factor
air_tf_wsb  air_tf_esb  100 @ r r_x_air_tf
air_tf_wnb  air_tf_enb  100 @ r r_x_air_tf
air_tf_wst  air_tf_est  100 @ r r_x_air_tf
air_tf_wnt  air_tf_ent  100 @ r r_x_air_tf
air_tf_wsb  air_tf_wnb  100 @ r r_y_air_tf
air_tf_esb  air_tf_enb  100 @ r r_y_air_tf
air_tf_wst  air_tf_wnt  100 @ r r_y_air_tf
air_tf_est  air_tf_ent  100 @ r r_y_air_tf
air_tf_wsb  air_tf_wst  100 @ r r_z_air_tf
air_tf_esb  air_tf_est  100 @ r r_z_air_tf
air_tf_wnb  air_tf_wnt  100 @ r r_z_air_tf
air_tf_enb  air_tf_ent  100 @ r r_z_air_tf
! End
```

Top_front.sdf

```
Region air_tf_S
-----------------------------------------------------------------
Region air_tf_N
-----------------------------------------------------------------
Region air_tf_B
-----------------------------------------------------------------
Region air_tf_T
-----------------------------------------------------------------
Region air_tf_W
-----------------------------------------------------------------
Region air_tf_E
-----------------------------------------------------------------
```

Top_front.idf

```
!  Engine Compartment Template
!  Jay Ierardi
Region air_tf elliptic
errxyz=4e-7
itmax=320
air_tf_wsb,air_tf_ent
```

*Top_Mid Object*

Name.lun

```
problem top_mid.
include #/batch.lun
include #/systemu.lun
```

Top_mid.gdf

```
!*****************************************************
!* Passenger compartment template (bottom back).     *
!*                                                   *
!* Last modified: 2 Feb 1998                         *
!* By: James Ierardi for the General Motors Project. *
!*****************************************************

include ../input.txt

POINT
air_tm_wsb (x_air_tm_s,y_air_tm_s,z_air_tm_s)
air_tm_esb (x_air_tm_e,y_air_tm_s,z_air_tm_s)
air_tm_wst (x_air_tm_s,y_air_tm_s,z_air_tm_e)
air_tm_est (x_air_tm_e,y_air_tm_s,z_air_tm_e)
air_tm_wnb (x_air_tm_s,y_air_tm_e,z_air_tm_s)
air_tm_enb (x_air_tm_e,y_air_tm_e,z_air_tm_s)
air_tm_wnt (x_air_tm_s,y_air_tm_e,z_air_tm_e)
air_tm_ent (x_air_tm_e,y_air_tm_e,z_air_tm_e)

Curve air_tm_sb Linear
air_tm_wsb; air_tm_esb

Curve air_tm_nb Linear
air_tm_wnb; air_tm_enb

Curve air_tm_st Linear
air_tm_wst; air_tm_est

Curve air_tm_nt Linear
air_tm_wnt; air_tm_ent

Curve air_tm_wb Linear
air_tm_wsb; air_tm_wnb

Curve air_tm_eb Linear
air_tm_esb; air_tm_enb

Curve air_tm_wt Linear
air_tm_wst; air_tm_wnt

Curve air_tm_et Linear
air_tm_est; air_tm_ent

Curve air_tm_ws Linear
air_tm_wsb; air_tm_wst

Curve air_tm_es Linear
air_tm_esb; air_tm_est

Curve air_tm_wn Linear
air_tm_wnb; air_tm_wnt

Curve air_tm_en Linear
air_tm_enb; air_tm_ent
```

```
! South Surface of whole thing
Surface air_tm_S By Curves Bilinear
air_tm_sb; air_tm_es; -air_tm_st; -air_tm_ws

! North Surface of whole thing
Surface air_tm_N By Curves Bilinear
air_tm_nb; air_tm_en; -air_tm_nt; -air_tm_wn

! Bottom Surface of whole thing
Surface air_tm_B By Curves Bilinear
air_tm_sb; air_tm_eb; -air_tm_nb; -air_tm_wb

! Top Surface of whole thing
Surface air_tm_T By Curves Bilinear
air_tm_st; air_tm_et; -air_tm_nt; -air_tm_wt

! West Surface of whole thing
Surface air_tm_W By Curves Bilinear
air_tm_wb; air_tm_wn; -air_tm_wt; -air_tm_ws

! East Surface of whole thing
Surface air_tm_E By Curves Bilinear
air_tm_eb; air_tm_en; -air_tm_et; -air_tm_es

! End
```

Top_mid.cdf

```
Grid Dimensions
ID=nlatm1
JD=nwatm1
KD=nhatm1
---------------------------------------------------
Variable Definitions
---------------------------------------------------
Vertex Attachments
air_tm_wsb  (1,1,1)
air_tm_esb  (nlatm1,1,1)
air_tm_wnb  (1,nwatm1,1)
air_tm_enb  (nlatm1,nwatm1,1)
air_tm_wst  (1,1,nhatm1)
air_tm_est  (nlatm1,1,nhatm1)
air_tm_wnt  (1,nwatm1,nhatm1)
air_tm_ent  (nlatm1,nwatm1,nhatm1)
---------------------------------------------------
Node Distributions
Default=1
!start-vertex     end-vertex  expansion-factor
air_tm_wsb  air_tm_esb  100 @ r r_x_air_tm
air_tm_wnb  air_tm_enb  100 @ r r_x_air_tm
air_tm_wst  air_tm_est  100 @ r r_x_air_tm
air_tm_wnt  air_tm_ent  100 @ r r_x_air_tm
air_tm_wsb  air_tm_wnb  100 @ r r_y_air_tm
air_tm_esb  air_tm_enb  100 @ r r_y_air_tm
air_tm_wst  air_tm_wnt  100 @ r r_y_air_tm
air_tm_est  air_tm_ent  100 @ r r_y_air_tm
air_tm_wsb  air_tm_wst  100 @ r r_z_air_tm
air_tm_esb  air_tm_est  100 @ r r_z_air_tm
air_tm_wnb  air_tm_wnt  100 @ r r_z_air_tm
air_tm_enb  air_tm_ent  100 @ r r_z_air_tm
! End
```

Top_mid.sdf

```
Region air_tm_S
-----------------------------------------------------------------
Region air_tm_N
-----------------------------------------------------------------
Region air_tm_B
-----------------------------------------------------------------
Region air_tm_T
-----------------------------------------------------------------
Region air_tm_W
-----------------------------------------------------------------
Region air_tm_E
-----------------------------------------------------------------
```

Top_mid.idf

```
!  Engine Compartment Template
!  Jay Ierardi
Region air_tm elliptic
errxyz=4e-7
itmax=320
air_tm_wsb,air_tm_ent
```

*Wind Object*

Name.lun

```
problem wind.
include #/batch.lun
include #/systemu.lun
```

Wind.gdf

```
!*****************************************************
!*  A 3 Layer HPR Windshield Template                *
!*                                                   *
!*  Last Modified: 28 Jan 1998                       *
!*  By James Ierardi for the General Motors Project. *
!*****************************************************

include ../input.txt

POINT
! Points around outside air
airout_wsb (x_airout_sb,y_wind_s,z_wind_s)
airout_wnb (x_airout_sb,y_wind_e,z_wind_s)
airout_wst (x_airout_st,y_wind_s,z_wind_e)
airout_wnt (x_airout_st,y_wind_e,z_wind_e)
!Fluid extension
ext_airout_wnb (x_airout_sb,y_wind_e+y_airs,z_wind_s)
ext_airout_wnt (x_airout_st,y_wind_e+y_airs,z_wind_e)

! Points around outside glass layer
glass1_wsb (x_glass1_sb,y_wind_s,z_wind_s)
glass1_wnb (x_glass1_sb,y_wind_e,z_wind_s)
glass1_wst (x_glass1_st,y_wind_s,z_wind_e)
glass1_wnt (x_glass1_st,y_wind_e,z_wind_e)
!Fluid extension
ext_glass1_wnb (x_glass1_sb,y_wind_e+y_airs,z_wind_s)
ext_glass1_wnt (x_glass1_st,y_wind_e+y_airs,z_wind_e)

! Points around pvb inter-layer
pvb_wsb (x_pvb_sb,y_wind_s,z_wind_s)
pvb_wnb (x_pvb_sb,y_wind_e,z_wind_s)
pvb_wst (x_pvb_st,y_wind_s,z_wind_e)
pvb_wnt (x_pvb_st,y_wind_e,z_wind_e)
pvb_esb (x_pvb_eb,y_wind_s,z_wind_s)
pvb_enb (x_pvb_eb,y_wind_e,z_wind_s)
pvb_est (x_pvb_et,y_wind_s,z_wind_e)
pvb_ent (x_pvb_et,y_wind_e,z_wind_e)
!Fluid Extension
ext_pvb_wnb (x_pvb_sb,y_wind_e+y_airs,z_wind_s)
ext_pvb_wnt (x_pvb_st,y_wind_e+y_airs,z_wind_e)
ext_pvb_enb (x_pvb_eb,y_wind_e+y_airs,z_wind_s)
ext_pvb_ent (x_pvb_et,y_wind_e+y_airs,z_wind_e)

! Points around inside glass layer
glass2_esb (x_glass2_eb,y_wind_s,z_wind_s)
glass2_enb (x_glass2_eb,y_wind_e,z_wind_s)
glass2_est (x_glass2_et,y_wind_s,z_wind_e)
glass2_ent (x_glass2_et,y_wind_e,z_wind_e)
!Fluid Extension
ext_glass2_enb (x_glass2_eb,y_wind_e+y_airs,z_wind_s)
ext_glass2_ent (x_glass2_et,y_wind_e+y_airs,z_wind_e)

! Points around inside air
airin_esb (x_airin_eb,y_wind_s,z_wind_s)
```

```
airin_enb (x_airin_eb,y_wind_e,z_wind_s)
airin_est (x_airin_et,y_wind_s,z_wind_e)
airin_ent (x_airin_et,y_wind_e,z_wind_e)
!Fluid Extension
ext_airin_enb (x_airin_eb,y_wind_e+y_airs,z_wind_s)
ext_airin_ent (x_airin_et,y_wind_e+y_airs,z_wind_e)

Curve airout_sb Linear
airout_wsb; glass1_wsb

Curve glass1_sb Linear
glass1_wsb; pvb_wsb

Curve pvb_sb Linear
pvb_wsb; pvb_esb

Curve glass2_sb Linear
pvb_esb; glass2_esb

Curve airin_sb Linear
glass2_esb; airin_esb

Curve wind_sb Linear
airout_wsb; glass1_wsb; pvb_wsb; pvb_esb; glass2_esb; airin_esb

Curve airout_nb Linear
airout_wnb; glass1_wnb

Curve glass1_nb Linear
glass1_wnb; pvb_wnb

Curve pvb_nb Linear
pvb_wnb; pvb_enb

Curve glass2_nb Linear
pvb_enb; glass2_enb

Curve airin_nb Linear
glass2_enb; airin_enb

Curve wind_nb Linear
airout_wnb; glass1_wnb; pvb_wnb; pvb_enb; glass2_enb; airin_enb

Curve ext_wind_nb Linear
ext_airout_wnb; ext_glass1_wnb; ext_pvb_wnb; ext_pvb_enb;
ext_glass2_enb; ext_airin_enb

Curve airout_st Linear
airout_wst; glass1_wst

Curve glass1_st Linear
glass1_wst; pvb_wst

Curve pvb_st Linear
pvb_wst; pvb_est

Curve glass2_st Linear
```

```
pvb_est; glass2_est

Curve airin_st Linear
glass2_est; airin_est

Curve wind_st Linear
airout_wst; glass1_wst; pvb_wst; pvb_est; glass2_est; airin_est

Curve airout_nt Linear
airout_wnt; glass1_wnt

Curve glass1_nt Linear
glass1_wnt; pvb_wnt

Curve pvb_nt Linear
pvb_wnt; pvb_ent

Curve glass2_nt Linear
pvb_ent; glass2_ent

Curve airin_nt Linear
glass2_ent; airin_ent

Curve wind_nt Linear
airout_wnt; glass1_wnt; pvb_wnt; pvb_ent; glass2_ent; airin_ent

Curve ext_wind_nt Linear
ext_airout_wnt; ext_glass1_wnt; ext_pvb_wnt; ext_pvb_ent;
ext_glass2_ent; ext_airin_ent

Curve airout_ws Linear
airout_wsb; airout_wst

Curve glass1_ws Linear
glass1_wsb; glass1_wst

Curve pvb_ws Linear
pvb_wsb; pvb_wst

Curve pvb_es Linear
pvb_esb; pvb_est

Curve glass2_es Linear
glass2_esb; glass2_est

Curve airin_es Linear
airin_esb; airin_est

Curve airout_wn Linear
airout_wnb; airout_wnt

Curve ext_airout_wn Linear
ext_airout_wnb; ext_airout_wnt

Curve glass1_wn Linear
glass1_wnb; glass1_wnt
```

```
Curve ext_glass1_wn Linear
ext_glass1_wnb; ext_glass1_wnt

Curve pvb_wn Linear
pvb_wnb; pvb_wnt

Curve ext_pvb_wn Linear
ext_pvb_wnb; ext_pvb_wnt

Curve pvb_en Linear
pvb_enb; pvb_ent

Curve ext_pvb_en Linear
ext_pvb_enb; ext_pvb_ent

Curve glass2_en Linear
glass2_enb; glass2_ent

Curve ext_glass2_en Linear
ext_glass2_enb; ext_glass2_ent

Curve airin_en Linear
airin_enb; airin_ent

Curve ext_airin_en Linear
ext_airin_enb; ext_airin_ent

Curve wind_wb Linear
airout_wsb; airout_wnb

Curve ext_wind_wb Linear
airout_wsb; airout_wnb; ext_airout_wnb

Curve airout_wb Linear
airout_wsb; airout_wnb

Curve wind_wt Linear
airout_wst; airout_wnt

Curve ext_wind_wt Linear
airout_wst; airout_wnt; ext_airout_wnt

Curve airout_wt Linear
airout_wst; airout_wnt

Curve ext_airout_wt Linear
airout_wnt; ext_airout_wnt

Curve glass1_wb Linear
glass1_wsb; glass1_wnb

Curve ext_glass1_wb Linear
glass1_wnb; ext_glass1_wnb

Curve glass1_wt   Linear
glass1_wst; glass1_wnt
```

```
Curve ext_glass1_wt Linear
glass1_wnt; ext_glass1_wnt

Curve pvb_wb Linear
pvb_wsb; pvb_wnb

Curve ext_pvb_wb Linear
pvb_wnb; ext_pvb_wnb

Curve pvb_wt Linear
pvb_wst; pvb_wnt

Curve ext_pvb_wt Linear
pvb_wnt; ext_pvb_wnt

Curve wind_eb Linear
airin_esb; airin_enb

Curve ext_wind_eb Linear
airin_esb; airin_enb; ext_airin_enb

Curve pvb_eb Linear
pvb_esb; pvb_enb

Curve ext_pvb_eb Linear
pvb_enb; ext_pvb_enb

Curve pvb_et Linear
pvb_est; pvb_ent

Curve ext_pvb_et Linear
pvb_ent; ext_pvb_ent

Curve glass2_eb Linear
glass2_esb; glass2_enb

Curve ext_glass2_eb Linear
glass2_enb; ext_glass2_enb

Curve wind_et Linear
airin_est; airin_ent

Curve ext_wind_et Linear
airin_est; airin_ent; ext_airin_ent

Curve glass2_et Linear
glass2_est; glass2_ent

Curve ext_glass2_et Linear
glass2_ent; ext_glass2_ent

Curve airin_eb Linear
airin_esb; airin_enb

Curve ext_airin_eb Linear
airin_enb; ext_airin_enb
```

```
Curve airin_et Linear
airin_est; airin_ent

Curve ext_airin_et Linear
airin_ent; ext_airin_ent

Surface Wind_S By Curves Bilinear
wind_sb; airin_es; -wind_st; -airout_ws

Surface Wind_N By Curves Bilinear
ext_wind_nb; ext_airin_en; -ext_wind_nt; -ext_airout_wn

Surface Wind_B By Curves Bilinear
wind_sb; ext_wind_eb; -ext_wind_nb; -ext_wind_wb

Surface Wind_T By Curves Bilinear
wind_st; ext_wind_et; -ext_wind_nt; -ext_wind_wt

Surface Wind_W By Curves Bilinear
ext_wind_wb; ext_airout_wn; -ext_wind_wt; -airout_ws

Surface Wind_E By Curves Bilinear
ext_wind_eb; ext_airin_en; -ext_wind_et; -airin_es

! End
```

Wind.cdf

```
Grid Dimensions
ID=nlw5
JD=nww2
KD=nhw1
------------------------------------------------
Variable Definitions
------------------------------------------------
Vertex Attachments
airout_wsb  (1,1,1)
glass1_wsb  (nlw1,1,1)
pvb_wsb           (nlw2,1,1)
pvb_esb           (nlw3,1,1)
glass2_esb  (nlw4,1,1)
airin_esb   (nlw5,1,1)
airout_wnb  (1,nww1,1)
glass1_wnb  (nlw1,nww1,1)
pvb_wnb           (nlw2,nww1,1)
pvb_enb           (nlw3,nww1,1)
glass2_enb  (nlw4,nww1,1)
airin_enb   (nlw5,nww1,1)
airout_wst  (1,1,nhw1)
glass1_wst  (nlw1,1,nhw1)
pvb_wst           (nlw2,1,nhw1)
pvb_est           (nlw3,1,nhw1)
glass2_est  (nlw4,1,nhw1)
airin_est   (nlw5,1,nhw1)
airout_wnt  (1,nww1,nhw1)
glass1_wnt  (nlw1,nww1,nhw1)
pvb_wnt           (nlw2,nww1,nhw1)
pvb_ent           (nlw3,nww1,nhw1)
glass2_ent  (nlw4,nww1,nhw1)
airin_ent   (nlw5,nww1,nhw1)
ext_airout_wnb    (1,nww2,1)
ext_glass1_wnb    (nlw1,nww2,1)
ext_pvb_wnb (nlw2,nww2,1)
ext_pvb_enb (nlw3,nww2,1)
ext_glass2_enb    (nlw4,nww2,1)
ext_airin_enb     (nlw5,nww2,1)
ext_airout_wnt    (1,nww2,nhw1)
ext_glass1_wnt    (nlw1,nww2,nhw1)
ext_pvb_wnt (nlw2,nww2,nhw1)
ext_pvb_ent (nlw3,nww2,nhw1)
ext_glass2_ent    (nlw4,nww2,nhw1)
ext_airin_ent     (nlw5,nww2,nhw1)
------------------------------------------------
Node Distributions
Default=1
!start-vertex     end-vertex  expansion-factor
airout_wsb  glass1_wsb  100 @ r r_x_airout
airout_wnb  glass1_wnb  100 @ r r_x_airout
airout_wst  glass1_wst  100 @ r r_x_airout
airout_wnt  glass1_wnt  100 @ r r_x_airout
glass1_wsb  pvb_wsb           100 @ r r_x_glass1
glass1_wnb  pvb_wnb           100 @ r r_x_glass1
```

```
glass1_wst  pvb_wst           100 @ r r_x_glass1
glass1_wnt  pvb_wnt           100 @ r r_x_glass1
pvb_wsb           pvb_esb           100 @ r r_x_pvb
pvb_wnb           pvb_enb           100 @ r r_x_pvb
pvb_wst           pvb_est           100 @ r r_x_pvb
pvb_wnt           pvb_ent           100 @ r r_x_pvb
pvb_esb           glass2_esb  100 @ r r_x_glass2
pvb_enb           glass2_enb  100 @ r r_x_glass2
pvb_est           glass2_est  100 @ r r_x_glass2
pvb_ent           glass2_ent  100 @ r r_x_glass2
glass2_esb  airin_esb   100 @ r r_x_airin
glass2_enb  airin_enb   100 @ r r_x_airin
glass2_est  airin_est   100 @ r r_x_airin
glass2_ent  airin_ent   100 @ r r_x_airin
airout_wsb  airout_wnb  100 @ r r_y_wind
airout_wst  airout_wnt  100 @ r r_y_wind
glass1_wsb  glass1_wnb  100 @ r r_y_wind
glass1_wst  glass1_wnt  100 @ r r_y_wind
pvb_wsb           pvb_wnb           100 @ r r_y_wind
pvb_wst           pvb_wnt           100 @ r r_y_wind
pvb_esb           pvb_enb           100 @ r r_y_wind
pvb_est           pvb_ent           100 @ r r_y_wind
glass2_esb  glass2_enb  100 @ r r_y_wind
glass2_est  glass2_ent  100 @ r r_y_wind
airin_esb   airin_enb   100 @ r r_y_wind
airin_est   airin_ent   100 @ r r_y_wind
airout_wsb  airout_wst  100 @ r r_z_wind
airout_wnb  airout_wnt  100 @ r r_z_wind
glass1_wsb  glass1_wst  100 @ r r_z_wind
glass1_wnb  glass1_wnt  100 @ r r_z_wind
pvb_wsb           pvb_wst           100 @ r r_z_wind
pvb_wnb           pvb_wnt           100 @ r r_z_wind
pvb_esb           pvb_est           100 @ r r_z_wind
pvb_enb           pvb_ent           100 @ r r_z_wind
glass2_esb  glass2_est  100 @ r r_z_wind
glass2_enb  glass2_ent  100 @ r r_z_wind
airin_esb   airin_est   100 @ r r_z_wind
airin_enb   airin_ent   100 @ r r_z_wind
ext_airout_wnb    ext_glass1_wnb    100 @ r r_x_airout
ext_airout_wnt    ext_glass1_wnt    100 @ r r_x_airout
ext_glass1_wnb    ext_pvb_wnb 100 @ r r_x_glass1
ext_glass1_wnt    ext_pvb_wnt 100 @ r r_x_glass1
ext_pvb_wnb ext_pvb_enb 100 @ r r_x_pvb
ext_pvb_wnt ext_pvb_ent 100 @ r r_x_pvb
ext_pvb_enb ext_glass2_enb    100 @ r r_x_glass2
ext_pvb_ent ext_glass2_ent    100 @ r r_x_glass2
ext_glass2_enb    ext_airin_enb     100 @ r r_x_airin
ext_glass2_ent    ext_airin_ent     100 @ r r_x_airin
airout_wnb  ext_airout_wnb    100 @ r r_y_ext_sm
airout_wnt  ext_airout_wnt    100 @ r r_y_ext_sm
glass1_wnb  ext_glass1_wnb    100 @ r r_y_ext_sm
glass1_wnt  ext_glass1_wnt    100 @ r r_y_ext_sm
pvb_wnb           ext_pvb_wnb 100 @ r r_y_ext_sm
pvb_wnt           ext_pvb_wnt 100 @ r r_y_ext_sm
pvb_enb           ext_pvb_enb 100 @ r r_y_ext_sm
pvb_ent           ext_pvb_ent 100 @ r r_y_ext_sm
glass2_enb  ext_glass2_enb    100 @ r r_y_ext_sm
```

```
glass2_ent  ext_glass2_ent    100 @ r r_y_ext_sm
airin_enb   ext_airin_enb     100 @ r r_y_ext_sm
airin_ent   ext_airin_ent     100 @ r r_y_ext_sm
ext_airout_wnb    ext_airout_wnt    100 @ r r_z_wind
ext_glass1_wnb    ext_glass1_wnt    100 @ r r_z_wind
ext_pvb_wnb ext_pvb_wnt 100 @ r r_z_wind
ext_pvb_enb ext_pvb_ent 100 @ r r_z_wind
ext_glass2_enb    ext_glass2_ent    100 @ r r_z_wind
ext_airin_enb     ext_airin_ent     100 @ r r_z_wind
! End
```

Wind.sdf

```
Region Wind_S
-----------------------------------------------------------------
Region Wind_N
-----------------------------------------------------------------
Region Wind_B
-----------------------------------------------------------------
Region Wind_T
-----------------------------------------------------------------
Region Wind_W
-----------------------------------------------------------------
Region Wind_E
-----------------------------------------------------------------
```

Wind.idf

```
!  3 Ply windshield Template
!  Jay Ierardi
Region wind elliptic
errxyz=4e-7
itmax=320
airout_wsb,ext_airin_ent
```

Propf.f

```
C
C
C
C=======================================================================
=
C
      SUBROUTINE PROPF(VISCL,
     &                 U,V,W,P,T,PHI,NPHI,XYZ,
     &                 ILABEL,LABEL,I,J,K,ID,JD,KD)
C
CSCDT Calculates the fluid viscosity.
C
CSCDB
C  PROPF is a user specified routine that calculates the fluid
C  viscosity. Sutherland's law is used for the molecular
C  viscosity of a single-component fluid.
C
C  Input:
C
C     ILABEL =  0  fluid
C             >  0  MCF component index
C             <  0  solid CHT object with label given by
C                   `LABEL' (invalid)
C      LABEL : character string identifying equation solved
C      U,V,W : Cartesian velocity components
C          P : static pressure
C          T : static temperature
C        PHI : additional scalars
C       NPHI : number of additional scalars
C        XYZ : Cartesian coordinates of grid nodes
C      I,J,K : topological grid coordinates
C   ID,JD,KD : topological grid dimensions
C
C  Output:
C
C      VISCL : fluid viscosity
C
C  Local:
C
C     VISCFL : fluid viscosity
C       POFF : pressure offset (level shift)
C       TOFF : temperature offset (level shift)
C     SUTHER : logical variable indicates whether or not to use
C              Sutherland's law
C
C  Common Blocks:
C
C  /CONTRL/ is declared so that information can be added as needed
C           for desired equation of state.
C
C     IONUM  : integer, unit numbers in CONTRL common block.
C     IARR   : integer, integer switches in CONTRL common block.
C     RARR   : real, real constants in CONTRL common block.
```

```
C     CONST  : real, property constants in CONTRL common block.
C     LARR   : logical, logical switches in CONTRL common block.
C
C  /PHINUM/ contains scalar transport equation information.
C
C     IARRQ  : integer, integer switches in PHINUM common block.
C     RARRQ  : real, real constants in PHINUM common block.
C     LARRQ  : logical, logical switches in PHINUM common block.
C
C  NOTE: The true levels of pressure (P) and temperature (T) are
C        assumed to be
C           P_true = P + POFF
C           T_true = T + TOFF
C        where POFF and TOFF are level shifts in the actual dependent
C        values of P and T that are solved for.
CSCDE
C
C=======================================================================
=
C
C---------------------------
C Subroutine Arguments
C---------------------------
C
      REAL U(ID,JD,KD),V(ID,JD,KD),W(ID,JD,KD),P(ID,JD,KD),T(ID,JD,KD),
     &     PHI(ID,JD,KD,*),XYZ(ID,JD,KD,3),VISCL
C
      INTEGER ILABEL,I,J,K,ID,JD,KD,NPHI
C
      CHARACTER*(*) LABEL
C
C---------------------------
C Local Variables
C---------------------------
C
      REAL VISCFL,POFF,TOFF
C
      LOGICAL SUTHER
C
C---------------------------
C Common Blocks
C---------------------------
C
      COMMON/CONTRL/
IONUM(100),IARR(200),RARR(100),CONST(100),LARR(200)
      REAL RARR,CONST
      INTEGER IONUM,IARR
      LOGICAL LARR
C
      EQUIVALENCE (CONST(5),VISCFL),(CONST(13),POFF),(CONST(14),TOFF),
     &            (LARR(83),SUTHER)
C
C
      COMMON/PHINUM/IARRQ(100,10),RARRQ(100,10),LARRQ(100,10)
      INTEGER IARRQ
      REAL RARRQ
      LOGICAL LARRQ
```

```
C
C---------------------------
C Executable Statements
C---------------------------
C
C------ SINGLE-COMPONENT FLUID
C
      IF (ILABEL.EQ.0) THEN
C
C    -- USE SUTHERLAND's LAW FOR MOLECULAR VISCOSITY --
C
        IF (SUTHER) THEN
          VISCL = 1.458E-6*(T(I,J,K)+TOFF)**1.5/(T(I,J,K)+TOFF+110.4)
        ELSE
          VISCL = VISCFL
        ENDIF
C
C------ MULTI-COMPONENT FLUID COMPONENT
C
      ELSE IF (ILABEL.GT.0) THEN
C
        VISCL = 1.458E-6*(T(I,J,K)+TOFF)**1.5/
     &                      (T(I,J,K)+TOFF+110.4)
C
C       VISCL = 7.6181e-6 + 3.2623e-8 * T(I,J,K)
C       VISCL = RARRQ(ILABEL,9)
C
      ELSE
        STOP 'PROPF: Invalid label'
      ENDIF
C
      RETURN
      END
```

Propq.f

```
C
C
C
C=====================================================================
=
C
      SUBROUTINE PROPQ(GAMAS,
     &                 U,V,W,P,T,PHI,NPHI,XYZ,
     &                 ILABEL,LABEL,I,J,K,ID,JD,KD)
C
C     CALCULATE SCALAR DIFFUSION COEFFICIENTS
C
C  ILABEL =  0   fluid
C           >0  MCF component index
C           <0  solid CHT object with label given by `LABEL'
C
C  The common block /CONTRL/ is declared here so that information
C  can be added as needed for the desired equation of state.
C
C  NOTE: The true levels of pressure (P) and temperature (T) are
C  assumed to be
C           P_true = P + POFF
C           T_true = T + TOFF
C  where POFF and TOFF are level shifts in the actual dependent
C  values of P and T that are solved for.
C
C=====================================================================
=
C
      REAL U(ID,JD,KD),V(ID,JD,KD),W(ID,JD,KD),P(ID,JD,KD),T(ID,JD,KD),
     &     PHI(ID,JD,KD,NPHI),XYZ(ID,JD,KD,3)
      CHARACTER*(*) LABEL
      LOGICAL LARR,LARRQ
      COMMON/CONTRL/
IONUM(100),IARR(200),RARR(100),CONST(100),LARR(200)
      COMMON/PHINUM/IARRQ(100,10),RARRQ(100,10),LARRQ(100,10)
      EQUIVALENCE (CONST(13),POFF),(CONST(14),TOFF)
C
      GAMAS = 1.458E-6*(T(I,J,K)+TOFF)**1.5/(T(I,J,K)+TOFF+110.4)
C
      RETURN
      END
```

Propt.f

```
C
C
C
C=====================================================================
=
C
      SUBROUTINE PROPT(CONDL,CPHEAT,CVHEAT,
     &                 U,V,W,P,T,PHI,NPHI,XYZ,
     &                 ILABEL,LABEL,I,J,K,ID,JD,KD)
C
CSCDT Calculates thermal properties of the fluid/solid.
C
CSCDB
C  PROPT is a user specified routine that calculates thermal
C  properties of the fluid/solid.
C
C  Input:
C
C     ILABEL = 0   fluid
C            > 0  MCF component index
C            < 0  solid CHT object with label given by `LABEL'
C      LABEL : character string identifying equation solved
C      U,V,W : Cartesian velocity components
C          P : static pressure
C          T : static temperature
C        PHI : additional scalars
C       NPHI : number of additional scalars
C        XYZ : Cartesian coordinates of grid nodes
C      I,J,K : topological grid coordinates
C   ID,JD,KD : topological grid dimensions
C
C  Output:
C
C      CONDL : local thermal conductivity
C     CPHEAT : specific heat at constant pressure
C     CVHEAT : specific heat at constant volume
C
C  Local:
C
C     MAXMCF : work space parameter for multi-component fluids
C       POFF : pressure offset (level shift)
C       TOFF : temperature offset (level shift)
C     CONDFL : fluid conductivity
C     CONDSL : solid conductivity
C      CPFLD : fluid specific heat at constant pressure
C      CVFLD : fluid specific heat at constant volume
C      CCSOL : solid specific heat
C     SUTHER : logical variable indicating whether or not to use
C              Sutherland's law
C
C  Common Blocks:
C
C  /CONTRL/ is declared so that information can be added as needed
C           for desired equation of state.
```

```
C
C     IONUM  : integer, unit numbers in CONTRL common block.
C     IARR   : integer, integer switches in CONTRL common block.
C     RARR   : real, real constants in CONTRL common block.
C     CONST  : real, property constants in CONTRL common block.
C     LARR   : logical, logical switches in CONTRL common block.
C
C  /MULTIF/ contains information for multicomponent fluids.
C
C     DENSQ  : real, component density.
C     ZMOLQ  : real, component molecular weight.
C     CONDQ  : real, component conductivity.
C     CPHTQ  : real, component specific heat at constant pressure.
C     CVHTQ  : real, component specific heat at constant volume.
C     LMCFQ  : logical, multi-component fluid.
C     EQSTQ  : logical, equation of state.
C     NMCF   : integer, number of components.
C     NMCFNA : integer, maximum scalar which is a multi-component
fluid.
C
C  NOTE: The true levels of pressure (P) and temperature (T) are
C        assumed to be
C           P_true = P + POFF
C           T_true = T + TOFF
C        where POFF and TOFF are level shifts in the actual dependent
C        values of P and T that are solved for.
CSCDE
C
C=======================================================================
=
C
C-----------------------------
C Subroutine Arguments
C-----------------------------
C
      INTEGER NPHI,I,J,K,ID,JD,KD,ILABEL
C
      REAL U(ID,JD,KD),V(ID,JD,KD),W(ID,JD,KD),P(ID,JD,KD),T(ID,JD,KD),
     &     PHI(ID,JD,KD,*),XYZ(ID,JD,KD,3),CONDL,CPHEAT,CVHEAT
C
      CHARACTER*(*) LABEL
C
C-----------------------------
C Local Variables
C-----------------------------
C
      INTEGER MAXMCF
      PARAMETER (MAXMCF = 100)
C
      REAL CONDFL,CPFLD,CVFLD,CONDSL,CCSOL,POFF,TOFF
C
      LOGICAL SUTHER
C
C-----------------------------
C Common Blocks
C-----------------------------
C
```

```
      COMMON/CONTRL/
IONUM(100),IARR(200),RARR(100),CONST(100),LARR(200)
      INTEGER IONUM,IARR
      REAL RARR,CONST
      LOGICAL LARR
C
      EQUIVALENCE
(CONST(3),CONDFL),(CONST(11),CPFLD),(CONST(12),CVFLD),
     &            (CONST(29),CONDSL),(CONST(30),CCSOL),
     &            (CONST(13),POFF),(CONST(14),TOFF),(LARR(83),SUTHER)
C
      COMMON /MULTIF/ DENSQ(MAXMCF),ZMOLQ(MAXMCF),
     &                CONDQ(MAXMCF),CPHTQ(MAXMCF),CVHTQ(MAXMCF),
     &                LMCFQ(MAXMCF),EQSTQ(MAXMCF),NMCF,NMCFNA
      INTEGER NMCF,NMCFNA
      REAL DENSQ,ZMOLQ,CONDQ,CPHTQ,CVHTQ
      LOGICAL LMCFQ,EQSTQ
C
C --- Property data block
C
      PARAMETER (RGAS=8.3143)
C
      REAL CPCH4(5,2), CPO2(5,2), CPH2O(5,2), CPCO2(5,2), CPCH2O(5,2),
     &     CPN2(5,2), tmpcp1, tmpcp2, tmpcv1, tmpcv2,
     &     tmpk1, tmpk2
C
C-------------------------------
C Data Statements
C-------------------------------
      DATA CPCH4/
     &         1.63552643E+00,  1.00842795E-02, -3.36916254E-06,
     &         5.34958667E-10, -3.15518833E-14,
     &         5.14987613E+00, -1.36709788E-02,  4.91800599E-05,
     &        -4.84743026E-08,  1.66693956E-11/
      DATA CPO2/
     &         3.66096083E+00,  6.56365523E-04, -1.41149485E-07,
     &         2.05797658E-11, -1.29913248E-15,
     &         3.78245636E+00, -2.99673415E-03,  9.84730200E-06,
     &         -9.68129508E-09, 3.24372836E-12/
      DATA CPH2O/
     &         2.67703787E+00, 2.97318329E-03,-7.73769690E-07,
     &         9.44336689E-11,-4.26900959E-15,
     &         4.19864056E+00,-2.03643410E-03, 6.52040211E-06,
     &        -5.48797062E-09, 1.77197817E-12/
      DATA CPCO2/
     &         4.63659493E+00, 2.74131991E-03,-9.95828531E-07,
     &         1.60373011E-10,-9.16103468E-15,
     &         2.35677352E+00, 8.98459677E-03,-7.12356269E-06,
     &         2.45919022E-09,-1.43699548E-13/
      DATA CPCH2O/
     &         3.16952654E+00, 6.19320583E-03,-2.25056377E-06,
     &         3.65975680E-10,-2.20149470E-14,
     &         4.79372315E+00,-9.90833369E-03, 3.73220008E-05,
     &        -3.79285261E-08, 1.31772652E-11/
      DATA CPN2/
     &         2.95257626E+00, 1.39690057E-03,-4.92631691E-07,
     &         7.86010367E-11,-4.60755321E-15,
```

```
     &         3.53100528E+00,-1.23660987E-04,-5.02999437E-07,
     &         2.43530612E-09,-1.40881235E-12/

C
C------------------------------
C Executable Statements
C------------------------------
      TT = T(I,J,K) + TOFF

C------ SINGLE-COMPONENT FLUID

      IF (ILABEL.EQ.0) THEN

C------ USE SUTHERLAND'S LAW FOR THERMAL CONDUCTIVITY -----

        IF (SUTHER) THEN
            CONDL  = 2.502E-3*(TT)**1.5/(TT+194.4)
        ELSE
          CONDL = CONDFL
        ENDIF
        CPHEAT = CPFLD
        CVHEAT = CVFLD

C------ MULTI-COMPONENT FLUID COMPONENT

      ELSE IF (ILABEL.GT.0) THEN
c        CPHEAT = RGAS / WM(ILABEL)
C
        IF (TT .GT. 1000.) THEN
C
          IF(ILABEL.EQ.1) THEN
            CPHEAT = (RGAS / .016)*
     &              (CPCH4(1,1) + TT * ( CPCH4(2,1) +
     &                           TT * ( CPCH4(3,1) +
     &                           TT * ( CPCH4(4,1) +
     &                           TT * ( CPCH4(5,1) )))))
            CVHEAT = CPHEAT - RGAS / (.016)
          ELSEIF(ILABEL.EQ.2) THEN
            CPHEAT = (RGAS / .032)*
     &              (CPO2(1,1) + TT * ( CPO2(2,1) +
     &                          TT * ( CPO2(3,1) +
     &                          TT * ( CPO2(4,1) +
     &                          TT * ( CPO2(5,1) )))))
            CVHEAT = CPHEAT - RGAS / (.032)
          ELSEIF(ILABEL.EQ.3) THEN      !product
            tmpcp1 = (RGAS / .044)*
     &              (CPCO2(1,1) + TT * ( CPCO2(2,1) +
     &                           TT * ( CPCO2(3,1) +
     &                           TT * ( CPCO2(4,1) +
     &                           TT * ( CPCO2(5,1) )))))

            tmpcp2 = (RGAS / .018)*
     &              (CPH2O(1,1) + TT * ( CPH2O(2,1) +
     &                           TT * ( CPH2O(3,1) +
     &                           TT * ( CPH2O(4,1) +
     &                           TT * ( CPH2O(5,1) )))))
            cpheat = .55*tmpcp1+.45*tmpcp2
```

```
            CVHEAT = .55*(tmpcp1 - RGAS / (.044))+
     &               .45*(tmpcp2 - RGAS / (.018))
          ELSEIF(ILABEL.EQ.4) THEN
            CPHEAT = (RGAS / .028)*
     &             (CPN2(1,1) + TT * ( CPN2(2,1) +
     &                          TT * ( CPN2(3,1) +
     &                          TT * ( CPN2(4,1) +
     &                          TT * ( CPN2(5,1) )))))
            CVHEAT = CPHEAT - RGAS / (.028)
          ENDIF
C                      TT lower than 1000. K
        ELSE
          IF(ILABEL.EQ.1) THEN
            CPHEAT = (RGAS / .016)*
     &             (CPCH4(1,2) + TT * ( CPCH4(2,2) +
     &                           TT * ( CPCH4(3,2) +
     &                           TT * ( CPCH4(4,2) +
     &                           TT * ( CPCH4(5,2) )))))
            CVHEAT = CPHEAT - RGAS / (.016)
          ELSEIF(ILABEL.EQ.2) THEN
            CPHEAT = (RGAS / .032)*
     &             (CPO2(1,2) + TT * ( CPO2(2,2) +
     &                          TT * ( CPO2(3,2) +
     &                          TT * ( CPO2(4,2) +
     &                          TT * ( CPO2(5,2) )))))
            CVHEAT = CPHEAT - RGAS / (.032)
          ELSEIF(ILABEL.EQ.3) THEN
            tmpcp1 = (RGAS / .044)*
     &             (CPCO2(1,2) + TT * ( CPCO2(2,2) +
     &                           TT * ( CPCO2(3,2) +
     &                           TT * ( CPCO2(4,2) +
     &                           TT * ( CPCO2(5,2) )))))
            tmpcp2 = (RGAS / .018)*
     &             (CPH2O(1,2) + TT * ( CPH2O(2,2) +
     &                           TT * ( CPH2O(3,2) +
     &                           TT * ( CPH2O(4,2) +
     &                           TT * ( CPH2O(5,2) )))))
            cpheat = .55*tmpcp1+.45*tmpcp2
            CVHEAT = .55*(tmpcp1 - RGAS / (.044))+
     &               .45*(tmpcp2 - RGAS / (.018))
          ELSEIF(ILABEL.EQ.4) THEN
            CPHEAT = RGAS / (.028) *
     &             (CPN2(1,2) + TT * ( CPN2(2,2) +
     &                          TT * ( CPN2(3,2) +
     &                          TT * ( CPN2(4,2) +
     &                          TT * ( CPN2(5,2) )))))
            CVHEAT = CPHEAT - RGAS / (.028)
          ENDIF
        ENDIF
C
c       may want to change the condl calculation to be gas specific
c       by using the values from the h2 file...
        CONDL  = CPHEAT * 1.458E-6*(TT)**1.5/(TT+110.4)
C
C---- USE SOLID DEFAULT THERMAL PROPERTIES
C
      ELSE
```

```
         IF (ILABEL .LT. 0) THEN
          IF ((LABEL.EQ.'GLASS_OUT').OR.(LABEL.EQ.'GLASS_IN')) THEN
C---- Thermal conductivity data from CRC handbook at various
temperatures,
C---- a 2nd degree polynomial curve fit was applied with R^2=0.9992
C
              CONDSL = 0.000002*((TT-273)**2)-0.0017*(TT-273)+1.2185
C
C---- Windshield glass specific heat data from Chris Barry at
Pilkington-
C---- Libbey-Owens-Ford citing 879.27 j/kgK at 25 C and 1214.23 j/kgK
at
C---- onset of plastic behavior ~522C.  A linear curve fit was applied.
C
               CCSOL = 879.27+0.67396*(TT-298)
C
               CONDL  = CONDSL
               H2HEAT = CCSOL
               CVHEAT = CCSOL
            ENDIF
            IF(LABEL .EQ. 'PVB') THEN
C---- PVB thermal conductivity and specific heat data from Mark Gold at
C---- Solutia/Saflex.
C
               CONSL = 1.731*(0.127-0.000165*((9./5.)*(TT-273)+32))
               CCSOL = 4187*(0.49+0.00025*((9./5.)*(TT-273)+32))
               CONDL  = CONDSL
               H2HEAT = CCSOL
               CVHEAT = CCSOL
            ENDIF
         ENDIF
      ENDIF
C
      RETURN
      END
```

```
C
C
C
C=======================================================================
=
C
      SUBROUTINE BCDTRN(LABEL,VAL,STIME,KEY)
C
CSCDT User specified transient boundary condition value.
C
CSCDB
C     BCDTRN is a user supplied routine for spatially uniform
C     temporally varying boundary conditions. This subroutine is ONLY
C     called for each boundary condition set that was designated as a
C     specified function of time in the boundary condition
C     pre-processor.
C
C     Based on the variable name, LABEL, (valid names give below) an
C     the user entered KEY value (entered in TASCbob3D), the variable
C     value, VAL, must be assigned a value based on the simulation
time,
C     STIME.
C
C
C      Input:
C        LABEL  : character, the name of variable that needs a value.
C
C                 The ONLY valid strings that LABEL can assume are:
C
C                    i) 'U', 'V', 'W' : Cartesian coordinate velocity
C                        components, at an inlet boundary condition.
C                        In a rotating frame of reference, these values
C                        represent the relative frame velocity
C                        components.
C
C                   ii) 'P': static pressure at an outlet or inlet.
C
C                  iii) 'P_TOTAL': total pressure at an inlet.  In
C                        rotating frames of reference this value
C                        represents the absolute frame total pressure.
C
C                   iv) 'TOTAL_MASS_FLOW': the total mass flow through
C                        ALL faces assigned to a transient mass flow
C                        boundary condition (inlet or outlet).
C
C                    v) 'T': static temperature at an inlet.
C
C        KEY    : integer, the integer that was entered in the boundary
C                 condition pre-processor that can be used to identify
C                 transient boundary condition specifications (when
more
C                 than one transient b.c. has been specified).
C        STIME  : real, the current simulation time (units of time)
C
```

```
C      Output:
C        VAL    : real, the assigned value of the variable named
C                 LABEL, at simulation time STIME, for the boundary
C                 condition set assigned the number KEY in TASCbob3D.
C
CSCDE
C
C=======================================================================
=
C
C---------------------------
C Subroutine Arguments
C---------------------------
C
      REAL VAL,STIME
C
      INTEGER KEY
C
      CHARACTER*(*) LABEL
C
C---------------------------
C Executable Statements
C---------------------------
C
      IF (LABEL.EQ.'TOTAL_MASS_FLOW' .AND. KEY.EQ.427) THEN
C-------- Total mass flow though ALL faces assigned a transient mass
b.c
C-----------------------------------------------------------------------
-------
C-------- The mass flow rates [kg/s] correspond to the ProfilARBED CFD
design
C-------- heat release rate curve [pg 48, Fig. 3.37 and Table 3.22]
that
C-------- has been represented as an equivalent source of Methane using
the
C-------- the following;
C-------- Heat of combustion (Methane), H_c = 49.6 KJ/g [SFPE HB Table
3-4.11]
C-------- Q = m'*H_c [SFPE HB Section 3-1 eq.9] Therefore, m'=Q/H_c
[kg/s]
C-------- To account for the symmetry boundary condition applied in
this
C-------- problem, the HRR was divided by two in order to obtain the
proper
C-------- mass loss rate for the inlet boundary condition.
C-----------------------------------------------------------------------
-------
         IF (STIME.GE.0 .AND .STIME.LE.240) THEN
            VAL = 0.0000588*STIME
         ENDIF
         IF (STIME.GT.240 .AND .STIME.LE.960) THEN
            VAL = 0.014113
         ENDIF
         IF (STIME.GT.960 .AND .STIME.LE.1380) THEN
            VAL = 0.014113+0.000099125*(STIME-960)
         ENDIF
         IF (STIME.GT.1380 .AND .STIME.LE.1560) THEN
```

```
            VAL = 0.055745
         ENDIF
         IF (STIME.GT.1560 .AND .STIME.LE.2280) THEN
            VAL = 0.055745-0.000063425*(STIME-1560)
         ENDIF
         PRINT *, "Total mass flow @ ",STIME," seconds = ",VAL
      ENDIF
C
      RETURN
      END
```

*Parameter File*

Top_front.prm

```
absorption = .1
bcinfo = t
cvfld = 735
cpfld = 1000.16
condfl = .026
cmu = .18
cht_solid_properties = t
combustion = f
ertime = 1.0e-3
fluids = t
gravx = 0.
gravy = 0.
gravz = -9.81
iskew = 2
kntime = 2
kntlin = 12
kntrst = 5
kntuvp = 30
lpac = t
laplacian = t
num_real_blocks = 30
poff = 101325.
pref = 101325
prandt = .85
pmass = t
rhofld = 1.125
rlxpac = .5
rlxbnd = 0.5
relax_ptotal = .5
relax_mass = .5
relax_density = .5
reset_time = f
rlxuvp = .3
stoich = 2.16
trn_flow_summary = t
trntto = t
tke_and_epsilon = f
turbmd = f
user_bc_access = t
user_mass_source = t
viscfl = 1.7e-5
znuf = 1.
znuo = 1.295
dtime = .25
tmptre = t
fv_radiation_model = f
difrad = f
mixture_model = t
equation_of_state = f
solid_angle_option = 1
number_polar_angles = 8
number_azimuthal_angles = 4
medium_participate_radiation = f
```

```
absorption_coeff_medium = 0.10
min_solver_its_int = 1
max_solver_its_int = 8
max_loops_inner_rad = 1
max_loops_outer_rad = 1
target_res_red_inner_rad = 0.1
target_res_red_outer_rad = 0.001
target_solver_red_intensity = 0.1
write_properties = t
tref = 298.
pref@[2,2,2]
store_cv_spent_fuel = t
density_glass_out = 2.3
density_glass_in = 2.3
density_pvb = 2.
cond_glass_out = 0.7
cond_glass_in = 0.7
cond_pvb = 0.0564
c_glass_out = 628.
c_glass_in = 628.
c_pvb = 2650.
enthform1 = -5.47e6
cp1 = 2253.7
cv1 = 1735.4
viscosity2 = 1.59e-5
enthform2 = 0.
cp2 = 953.
cv2 = 693.
enthform3 = -1.014165e7
cp3 = 1304.95
cv3 = 1134.54
enthform4 = 0.0
cp4 = 710.
cv4 = 710.
enthform5 = 0.
cp5 = 953.
cv5 = 656.
mp1@[2,2,2]:above_eng
mp2@[2,2,2]:main
mp3@[2,2,2]:top_mid
mp4@[2,2,2]:wind
mp5@[7,2,2]:wind
mp6@[10,2,2]:wind
mp7@[13,2,2]:wind
mp8@[15,2,2]:wind
mp9@[4,4,4]:above_eng
mp10@[2,4,2]:above_eng
mp11@[4,2,4]:above_eng
beta = 1/tref
gam1 = viscfl
viscosity1 = viscfl
gam2 = viscfl
gam3 = viscfl
viscosity3 = viscfl
gam4 = viscfl
viscosity4 = viscfl
gam5 = viscfl
```

```
viscosity5 = viscfl
!%tasctool_memory = -nr22m -ni7m -nc1m
!%tascbob3d_memory = -s4
!%tascflow3d_memory = -nr40m -ni6m
```

# Appendix B – TASCflow Utility Scripts

The following programs were written in the C programming language and designed to work with the various TASCflow utilities on the Digital UNIX platform.

## *Newcar.c*

```
/* assembles a new car by using quickgrid on the six grid objects and
then
   uses Tascbob.c to assemble the multi-grid model and apply boundary
conditions
*/

#include<stdio.h>
#include<string.h>

void main(){

  char quickgrid[150] = "/station/zklzz2/local/jbarnett/Bin/quickgrid";
  char step1[150] = "Top_Front";
  char step2[150] = "../Top_Mid";
  char step3[150] = "../Above_Eng";
  char step4[150] = "../Above_Fire";
  char step5[150] = "../Wind";
  char step6[150] = "../Fire";

  chdir(step1);
  system(quickgrid);
  chdir(step2);
  system(quickgrid);
  chdir(step3);
  system(quickgrid);
  chdir(step4);
  system(quickgrid);
  chdir(step5);
  system(quickgrid);
  chdir(step6);
  system(quickgrid);
}
```

Quickgrid.c

```
/* Quickgrid -- executes the tascgrid utilities */

#include<stdio.h>

void main(){

char execute_tascgridg[200] =
"/station/zklzz2/local/jbarnett/Bin/Quickgrid_Files/g_cdf.txt |
/station/zklzz2/local/cfxtascflow/TASCflow/Utility/TASCscript tascgridg
-nline 1500 -nprm 750 > g_session.txt";

char execute_tascgridc[200] =
"/station/zklzz2/local/jbarnett/Bin/Quickgrid_Files/g_cdf.txt |
```

```
/station/zklzz2/local/cfxtascflow/TASCflow/Utility/TASCscript tascgridc
-nline 1500 -nprm 750 > c_session.txt";

char execute_tascgrids[200] =
"/station/zklzz2/local/jbarnett/Bin/Quickgrid_Files/s_idf.txt |
/station/zklzz2/local/cfxtascflow/TASCflow/Utility/TASCscript tascgrids
-nline 1500 -nprm 750 > s_session.txt";

char execute_tascgridi[200] =
"/station/zklzz2/local/jbarnett/Bin/Quickgrid_Files/s_idf.txt |
/station/zklzz2/local/cfxtascflow/TASCflow/Utility/TASCscript tascgridi
-nline 1500 -nprm 750 > i_session.txt";

char error_g[200] = "grep -w ERROR g_session.txt > g_check.txt";
char error_c[200] = "grep -w ERROR c_session.txt > c_check.txt";
char error_s[200] = "grep -w ERROR s_session.txt > s_check.txt";
char error_i[200] = "grep -w ERROR i_session.txt > i_check.txt";

char clean_g[200] = "rm -f g_session.txt g_check.txt";
char clean_c[200] = "rm -f c_session.txt c_check.txt";
char clean_s[200] = "rm -f s_session.txt s_check.txt";
char clean_i[200] = "rm -f i_session.txt i_check.txt";

system(execute_tascgridg); /* Process utility */
system(error_g); /* Check for errors */
system(clean_g); /* delete associated files */
system(execute_tascgridc);
system(error_c);
system(clean_c);
system(execute_tascgrids);
system(error_s);
system(clean_s);
system(execute_tascgridi);
system(error_i);
system(clean_i);

}
```

g_cdf.txt

```
x
y
y
```

s_idf.txt

```
x
y
```

## *Num_trans.c*

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int num_trans(char *field_name)
{
  FILE *test_nf;
```

```
  char get_nf[200] = "grep -w ";
  char file_path[200] = " ../input.txt";
  /*  char field_name[200];*/
  char get_varname[200];
  char dummy[50];
  int new_val;
  int old_val;
  char new_operand[50];
  char old_operand[50];
  int num_fields;
  int count;
  int sum;
  char counter[100];
  char kill_test_nf[20] = "rm -f test_nf";

  /*  printf("Enter field: ");
      scanf("%s",&field_name);*/

  /* Get the number of fields */
  /*printf("Field name is %s\n",field_name);*/
  strcat(get_nf, field_name);
  strcat(get_nf, file_path);
  strcat(get_nf, " | awk -f ../awk.prog > test_nf");
  /*printf("%s", get_nf);*/
  system(get_nf);

if ((test_nf = fopen("test_nf", "r")) == NULL)
    printf("test_nf could not be opened\n");

  else {
    /* Initialize variables before loop begins */
    fscanf(test_nf,"%d",&num_fields);
    count = 1;
    sum = 0;
    new_val = 0;
    new_operand[0] = '\0';

    while(count <= num_fields){

      /* Swap previous values and operands over in order to perform
       previous operand on current value */
      old_val = new_val;
      strcpy(old_operand,new_operand);

      /* Increment count variable for this loop */
      count = count + 1;

      /* Grab current values for value and operand */
      fscanf(test_nf,"%s%d%s",&dummy,&new_val,&new_operand);
      /*      printf("The new operand is *%s*\n",new_operand);
            printf("The old operand is *%s*\n",old_operand);*/

      /* Sum is initially the first value */
      if(count == 2){
      sum = new_val;
      }
```

```
      /* Sum is previous operand acting on previous sum and current
value */
      else{
      if(strcmp(old_operand,"+")==0){
        sum = sum + new_val;
          }
      else{
        sum = sum - new_val;
      }
      }
    }

  fclose(test_nf);
  /*  printf("The number of fields is %d\n",num_fields);*/
      /* printf("The sum is %d\n",sum);*/
  }
  system(kill_test_nf);
  return sum;
}
```

Num_trans.h

```
int num_space(char *field_name);
```

## *Tascbob.c*

```
/* This function sets up and executes the TASCbob3d utility
 * for the windshield problem.
 * By Jay Ierardi for General Motors Project. */

#include<string.h>
#include<stdio.h>
#include"num_trans.h"

void main(){

int input_choice;
char input_path[200] = "";
char bob_start[200] = "cat ";
char field_name[50] = "";
char dir_name[200] = "";
char tascript_dir[200] =
"/station/zklzz2/local/cfxtascflow/TASCflow/Bin/";
int nlatf1_val, nwatf1_val, nhatf1_val;
int nlatm1_val, nwatm1_val, nhatm1_val;
int nlatb1_val, nwatb1_val, nhatb1_val;
int nlw1_val, nlw2_val, nlw3_val, nlw4_val, nlw5_val, nww1_val,
nww2_val, nhw1_val;
int nlae1_val, nwae1_val, nwae2_val, nhae1_val;
int nlptb1_val, nwptb1_val, nwptb2_val, nhptb1_val;
int nlf1_val, nwf1_val, nhf1_val, nlf2_val, nhf2_val;
int fire1ex_val, fire1ey_val, fire1ez_val, fire2ex_val, fire2ey_val,
fire2ez_val;
int fire1ox_val, fire1oy_val, fire2ox_val, fire2oy_val;
```

```
FILE *bob;

  /* Create standard input file for tascbob3d */
  /* Grid Embedding/Attaching Input file begins */
  if ((bob = fopen("bob.txt", "w")) == NULL)
    printf("bob.txt could not be opened\n");

  else {
    /* Assign numbers to node locations for attachment */
    /* Top_Front Fluid Extension*/
    strcpy(field_name,"nlatf1");
    nlatf1_val=num_trans(field_name);
    strcpy(field_name,"nwatf1");
    nwatf1_val=num_trans(field_name);
    strcpy(field_name,"nhatf1");
    nhatf1_val=num_trans(field_name);

    /* Top_Mid Fluid Extension*/
    strcpy(field_name,"nlatm1");
    nlatm1_val=num_trans(field_name);
    strcpy(field_name,"nwatm1");
    nwatm1_val=num_trans(field_name);
    strcpy(field_name,"nhatm1");
    nhatm1_val=num_trans(field_name);

    /* Windshield object */
    strcpy(field_name,"nlw1");
    nlw1_val=num_trans(field_name);
    strcpy(field_name,"nlw2");
    nlw2_val=num_trans(field_name);
    strcpy(field_name,"nlw3");
    nlw3_val=num_trans(field_name);
    strcpy(field_name,"nlw4");
    nlw4_val=num_trans(field_name);
    strcpy(field_name,"nlw5");
    nlw5_val=num_trans(field_name);
    strcpy(field_name,"nww1");
    nww1_val=num_trans(field_name);
    strcpy(field_name,"nww2");
    nww2_val=num_trans(field_name);
    strcpy(field_name,"nhw1");
    nhw1_val=num_trans(field_name);

    /* Air Above Engine object */
    strcpy(field_name,"nlae1");
    nlae1_val=num_trans(field_name);
    strcpy(field_name,"nwae1");
    nwae1_val=num_trans(field_name);
    strcpy(field_name,"nwae2");
    nwae2_val=num_trans(field_name);
    strcpy(field_name,"nhae1");
    nhae1_val=num_trans(field_name);

    strcpy(field_name,"nlf1");
    nlf1_val=num_trans(field_name);
    strcpy(field_name,"nwf1");
    nwf1_val=num_trans(field_name);
```

```
    strcpy(field_name,"nhf1");
    nhf1_val=num_trans(field_name);
    strcpy(field_name,"nlf2");
    nlf2_val=num_trans(field_name);
    strcpy(field_name,"nhf2");
    nhf2_val=num_trans(field_name);

    /************************************
     *  BEGIN OBJECT EMBEDDING/ATTACHING  *
     ************************************/

    /* Attachment info for Top_Front to Top_Mid objects */
    fprintf(bob, "e\n"); /* Grid Embedding/Attaching option */
    fprintf(bob, "2\n"); /* Grid Attaching option */
    fprintf(bob, "1\n"); /* Attach new grid to existing */
    /* Region on existing grid MAIN*/
    fprintf(bob, "[%d,,]:MAIN\n",nlatf1_val);
    fprintf(bob, "Top_Mid\n");  /* Label for new grid */
    fprintf(bob, "../Top_Mid/top_mid.grd\n");  /* Name of file
containing new grid */
    fprintf(bob, "[1,1,1]:top_mid\n");
                                /* connects to [nlatf1,1,1]:MAIN */
    fprintf(bob, "[1,%d,%d]:top_mid\n",nwatm1_val,nhatm1_val);
                                /* connects to
[nlatf1,nwatf1,nhatf2]:MAIN */
    fprintf(bob, "[1,1,%d]:top_mid\n",nhatm1_val);
                                /* connects to [nlatf1,1,nhatf1]:MAIN
*/
    fprintf(bob, "1\n"); /* in case nodes don't match */

    /* Attachment info for Top_Front to Above_Eng objects */
    fprintf(bob, "e\n"); /* Grid Embedding/Attaching option */
    fprintf(bob, "2\n"); /* Grid Attaching option */
    fprintf(bob, "1\n"); /* Attach new grid to existing */
    /* Region on existing grid */
    fprintf(bob, "[,,1]:main\n");
    fprintf(bob, "Above_Eng\n");  /* Label for new grid */
    fprintf(bob, "../Above_Eng/above_eng.grd\n");
    fprintf(bob, "[1,1,%d]:above_eng\n",nhae1_val);
                                /* connects to [1,1,1]:MAIN */
    fprintf(bob,
"[%d,%d,%d]:above_eng\n",nlae1_val,nwae2_val,nhae1_val);
                                /* connects to [nlatf1,nwatf1,1]:MAIN
*/
    fprintf(bob, "[1,%d,%d]:above_eng\n",nwae2_val,nhae1_val);
                                /* connects to [1,nwatf1,1]:MAIN */
    fprintf(bob, "1\n"); /* in case nodes don't match */

    /* Attachment info for Above_Eng to Wind */
    fprintf(bob, "e\n"); /* Grid Embedding/Attaching option */
    fprintf(bob, "2\n"); /* Grid Attaching option */
    fprintf(bob, "1\n"); /* Attach grid to existing */
    /* Region on existing grid */
    fprintf(bob, "[%d,,]:above_eng\n",nlae1_val);
    fprintf(bob, "wind\n");
    fprintf(bob, "../Wind/wind.grd\n");
    fprintf(bob, "[1,1,1]:WIND\n");
```

```
                                    /* connects to [nlae1,1,1]:above_eng
*/
    fprintf(bob, "[1,%d,%d]:WIND\n",nww2_val,nhw1_val);
                                    /*connects to
[nlae1,nwae2,nhae1]:above_eng*/
    fprintf(bob, "[1,1,%d]:WIND\n",nhw1_val);
                                    /*connects to
[nlae1,1,nhae1]:above_eng*/
    fprintf(bob, "1\n"); /* in case nodes don't match */

    /* Attachment info for Top_Mid to Wind objects */
    fprintf(bob, "e\n"); /* Grid Embedding/Attaching option */
    fprintf(bob, "2\n"); /* Grid Attaching option */
    fprintf(bob, "2\n"); /* Attach two existing grids*/
    /* Region on existing grid */
    fprintf(bob, "[,,%d]:wind\n",nhw1_val);
    fprintf(bob, "[1,1,1]:top_mid\n");
    fprintf(bob, "[%d,%d,1]:top_mid\n",nlatm1_val,nwatm1_val);
    fprintf(bob, "[1,%d,1]:top_mid\n",nwatm1_val);
    fprintf(bob, "1\n"); /* in case nodes don't match */

    fprintf(bob, "e\n");
    fprintf(bob, "1\n"); /* Grid Embedding */
    fprintf(bob,
"[%d:%d,1:%d,1:%d]:above_eng\n",nlf1_val,nlae1_val,nwf1_val,nhf1_val);
    fprintf(bob, "fire1\n");
    fprintf(bob, "1\n"); /* refine current grid */
    fprintf(bob, "3\n");
    fprintf(bob, "3\n");
    fprintf(bob, "3\n");

    fire1ex_val = ((nlae1_val-nlf1_val)*3)+1;
    fire1ey_val = ((nwf1_val-1)*3)+1;
    fire1ez_val = ((nhf1_val-1)*3)+1;

    fprintf(bob, "e\n");
    fprintf(bob, "1\n"); /* Grid Embedding */
    fprintf(bob, "[1:%d,1:%d,1:%d]:wind\n",nlf2_val,nwf1_val,nhf2_val);
    fprintf(bob, "fire2\n");
    fprintf(bob, "1\n"); /* refine current grid */
    fprintf(bob, "3\n");
    fprintf(bob, "3\n");
    fprintf(bob, "3\n");

    fire2ex_val = ((nlf2_val-1)*3)+1;
    fire2ey_val = ((nwf1_val-1)*3)+1;
    fire2ez_val = ((nhf2_val-1)*3)+1;

    /* Attach FIRE1 and FIRE2 to each other */
    fprintf(bob, "e\n");
    fprintf(bob, "2\n");
    fprintf(bob, "2\n");
    fprintf(bob, "[%d,,]:FIRE1\n", fire1ex_val);
    fprintf(bob, "[1,1,1]:FIRE2\n");
    fprintf(bob, "[1,%d,%d]:FIRE2\n", fire2ey_val, fire2ez_val);
    fprintf(bob, "[1,1,%d]:FIRE2\n", fire2ez_val);
    fprintf(bob, "1\n"); /* in case nodes don't match */
```

```
/* Attach FIRE1 to Above_Eng */
fprintf(bob, "e\n");
fprintf(bob, "2\n");
fprintf(bob, "2\n");
fprintf(bob, "[,,1]:FIRE1\n");
fprintf(bob, "[%d,1,1]:above_eng\n",nlf1_val);
fprintf(bob, "[%d,%d,1]:above_eng\n",nlae1_val,nwf1_val);
fprintf(bob, "[%d,%d,1]:above_eng\n",nlf1_val,nwf1_val);
fprintf(bob, "1\n"); /* in case nodes don't match */
fprintf(bob, "e\n");
fprintf(bob, "2\n");
fprintf(bob, "2\n");
fprintf(bob, "[,1,]:FIRE1\n");
fprintf(bob, "[%d,1,1]:above_eng\n",nlf1_val);
fprintf(bob, "[%d,1,%d]:above_eng\n",nlae1_val,nhf1_val);
fprintf(bob, "[%d,1,%d]:above_eng\n",nlf1_val,nhf1_val);
fprintf(bob, "1\n"); /* in case nodes don't match */

/* Attach FIRE2 to Wind */
fprintf(bob, "e\n");
fprintf(bob, "2\n");
fprintf(bob, "2\n");
fprintf(bob, "[,,1]:FIRE2\n");
fprintf(bob, "[1,1,1]:wind\n");
fprintf(bob, "[%d,%d,1]:wind\n",nlf2_val,nwf1_val);
fprintf(bob, "[1,%d,1]:wind\n",nwf1_val);
fprintf(bob, "1\n"); /* in case nodes don't match */
fprintf(bob, "e\n");
fprintf(bob, "2\n");
fprintf(bob, "2\n");
fprintf(bob, "[,1,]:FIRE2\n");
fprintf(bob, "[1,1,1]:wind\n");
fprintf(bob, "[%d,1,%d]:wind\n",nlf2_val,nhf2_val);
fprintf(bob, "[1,1,%d]:wind\n",nhf2_val);
fprintf(bob, "1\n"); /* in case nodes don't match */
fprintf(bob, "g\n"); /* Advance to object specification */

/*******************************
 *  BEGIN OBJECT SPECIFICATION  *
 *******************************/

/* Define Windshield layer objects */

fprintf(bob, "e\n");  /* Define Object */
fprintf(bob, "GLASS_OUT\n");  /* Object Name */
fprintf(bob, "[nlw1:nlw2,1:nww1,1:(nhw1-1)]:WIND\n");
fprintf(bob, "\n");
fprintf(bob, "e\n");  /* Define Object */
fprintf(bob, "PVB\n");  /* Object Name */
fprintf(bob, "[nlw2:nlw3,1:nww1,1:(nhw1-1)]:WIND\n");
fprintf(bob, "\n");
fprintf(bob, "e\n");  /* Define Object */
fprintf(bob, "GLASS_IN\n");  /* Object Name */
fprintf(bob, "[nlw3:nlw4,1:nww1,1:(nhw1-1)]:WIND\n");
fprintf(bob, "\n");

/* Assign solid properties */
```

```
    fprintf(bob, "p\n"); /* Modify Solid Properties */
    fprintf(bob, "1\n"); /* Select Object # */
    fprintf(bob, "2\n"); /* CHT Solid */
    fprintf(bob, "2\n"); /* Select Object # */
    fprintf(bob, "2\n"); /* CHT Solid */
    fprintf(bob, "3\n"); /* Select Object # */
    fprintf(bob, "2\n"); /* CHT Solid */
    fprintf(bob, "0\n"); /* Finished Modifying Solid Objects */

    fprintf(bob, "g\n"); /* Advance to Boundary Condition Specification
*/

    /************************************
     *  BOUNDARY CONDITION SPECIFICATION  *
     ************************************/
    /* set boundary condition attributes */

    fprintf(bob, "a\n"); /* attributes */
    fprintf(bob, "y\n"); /* keep flow field solution required */
    fprintf(bob, "n\n"); /* toggle to incompressible flow */
    fprintf(bob, "y\n"); /* keep incompressible flow */
    fprintf(bob, "y\n"); /* keep same wall treatment */
    fprintf(bob, "y\n"); /* keep log wall */
    fprintf(bob, "y\n"); /* toggle to reacting flow */
    fprintf(bob, "1\n"); /* selects EDM combustion model */
    fprintf(bob, "n\n"); /* keep 1 step reaction model */
    fprintf(bob, "CH4\n"); /* fuel component name */
    fprintf(bob, "O2\n"); /* oxidant component name */
    fprintf(bob, "PROD\n"); /* products component name */
    fprintf(bob, "y\n"); /* define additional scalars */
    fprintf(bob, "N2\n");
    fprintf(bob, "n\n"); /* no additional scalars */
    fprintf(bob, "n\n"); /* no other transport eqs */
    fprintf(bob, "n\n"); /* no lagrangian particle tracking */
    fprintf(bob, "y\n"); /* use Finite Volume Radiation model */
    fprintf(bob, "n\n"); /* toggle to inertial coordinate system */
    fprintf(bob, "n\n"); /* toggle to no moving walls */
    fprintf(bob, "n\n"); /* no overlap boundary conditions */
    fprintf(bob, "n\n"); /* don't read PRO */
    fprintf(bob, "y\n"); /* transient BC's required */
    fprintf(bob, "n\n"); /* don't respecify */

    /* Symmetry Boundary Condition */
    fprintf(bob, "e\n");
    fprintf(bob, "5\n"); /* Symmetry bc */
    fprintf(bob, "3\n"); /* attach to specific regions */
    fprintf(bob, "[,1,]:main\n");
    fprintf(bob, "[,1,]:top_mid\n");
    fprintf(bob, "[,1,]:above_eng\n");
    fprintf(bob, "[,1,]:wind\n");
    fprintf(bob, "[,1,]:fire1\n");
    fprintf(bob, "[,1,]:fire2\n");
    fprintf(bob, "\n");

    /* Opening Boundary Condition */
    fprintf(bob, "e\n");
```

```
fprintf(bob, "6\n"); /* opening bc */
fprintf(bob, "3\n"); /* pressure over boundary region */
fprintf(bob, "n\n"); /* constant pressure value */
fprintf(bob, "101325\n"); /* value of pressure */
fprintf(bob, "1\n"); /* flow direction is normal to face */
fprintf(bob, ".05\n"); /* Turbulent intensity for inflow */
fprintf(bob, ".25\n"); /* Eddy length scale for inflow */
fprintf(bob, "1\n"); /* Specify temperature */
fprintf(bob, "298\n"); /* Inflow temperature value */
fprintf(bob, "2\n"); /* use local temperature values */
fprintf(bob, "0.05\n"); /* emissivity of incoming air */
fprintf(bob, "1\n"); /* CH4 scalar */
fprintf(bob, "0\n"); /* CH4 scalar value */
fprintf(bob, "1\n"); /* O2 scalar */
fprintf(bob, "0.23\n"); /* O2 scalar value */
fprintf(bob, "1\n"); /* PROD scalar */
fprintf(bob, "0\n"); /* PROD scalar value */
fprintf(bob, "3\n"); /* attach bc to specific faces */
fprintf(bob, "[1,,]:main\n");
fprintf(bob, "[1,,]:above_eng\n");
fprintf(bob, "[,,nhatf1]:main\n");
fprintf(bob, "[,nwatf1,]:main\n");
fprintf(bob, "[,,nhatm1]:top_mid\n");
fprintf(bob, "[,nwatm1,]:top_mid\n");
fprintf(bob, "[nlatm1,,]:top_mid\n");
fprintf(bob, "[,nwae2,]:above_eng\n");
fprintf(bob, "[,nww2,]:wind\n");
fprintf(bob, "[nlw5,,]:wind\n");
fprintf(bob, "[,nwae1:nwae2,1]:above_eng\n");
fprintf(bob, "[,nww1:nww2,1]:wind\n");
fprintf(bob, "\n");

/* CHT Exterior Boundary Condition */
fprintf(bob, "e\n");
fprintf(bob, "7\n"); /* CHT Exterior Boundary Conditions */
fprintf(bob, "7\n"); /* adiabatic wall */
fprintf(bob, "3\n");
fprintf(bob, "[nlw1:nlw4,1:nww1,1]:wind\n");
fprintf(bob, "\n");

fire1ox_val = fire1ex_val-5;
fire1oy_val = fire1ey_val-5;
fire2ox_val = fire2ex_val-5;
fire2oy_val = fire2ey_val-5;

/* Wall Boundary Condition */
fprintf(bob, "e\n");
fprintf(bob, "1\n"); /* Wall Boundary Conditions */
fprintf(bob, "1\n"); /* Stationary Wall */
fprintf(bob, "1\n"); /* Smooth Wall */
fprintf(bob, "7\n"); /* Adiabatic wall (ignored for CHT objects) */
fprintf(bob, "0.9\n"); /* Emissivity */
fprintf(bob, "3\n"); /* Attach BC to regions */
fprintf(bob, "[nlw1,1:nww1,1:(nhw1-1)]:wind\n");
fprintf(bob, "[nlw4,1:nww1,1:(nhw1-1)]:wind\n");
fprintf(bob, "[nlw1:nlw4,nww1,1:(nhw1-1)]:wind\n");
fprintf(bob, "[nlw1:nlw4,1:nww1,(nhw1-1)]:wind\n");
```

```
    fprintf(bob, "[1:%d,1:nwae1,1]:above_eng\n",nlf1_val);
    fprintf(bob, "[%d:,nwf1:nwae1,1]:above_eng\n",nlf1_val);
    fprintf(bob, "[1:%d,%d:nww1,1]:wind\n",nlf2_val,nwf1_val);
    fprintf(bob, "[%d:nlw1,1:nww1,1]:wind\n",nlf2_val);
    fprintf(bob, "[nlw4:nlw5,1:nww1,1]:wind\n");
    fprintf(bob, "[1:5,,1]:fire1\n");
    fprintf(bob, "[5:,%d:,1]:fire1\n",fire1oy_val);
    fprintf(bob, "[1:%d,%d:,1]:fire2\n",fire2ox_val,fire2oy_val);
    fprintf(bob, "[%d:,,1]:fire2\n",fire2ox_val);
    fprintf(bob, "\n");

    /* Inflow Boundary Condition */
    fprintf(bob, "e\n");
    fprintf(bob, "2\n"); /* Inflow Boundary Condition */
    fprintf(bob, "11\n"); /* mass flow specified */
    fprintf(bob, "2\n"); /* f(time) */
    fprintf(bob, "427\n"); /* KEY number for transient BC */
    fprintf(bob, "1\n"); /* flow is normal to face */
    fprintf(bob, "0.35\n"); /* Turbulent Intensity */
    fprintf(bob, "0.05\n"); /* Eddy Length Scale */
    fprintf(bob, "1\n"); /* Specify inlet temperature */
    fprintf(bob, "825\n"); /* Temp (K) */
    fprintf(bob, "2\n"); /* use local temp */
    fprintf(bob, "0.05\n"); /* surface emissivity */
    fprintf(bob, "1\n"); /* CH4 scalar */
    fprintf(bob, "1\n"); /* scalar value */
    fprintf(bob, "1\n"); /* O2 scalar */
    fprintf(bob, "0\n"); /* scalar value */
    fprintf(bob, "1\n"); /* PROD scalar */
    fprintf(bob, "0\n"); /* scalar value */
    fprintf(bob, "3\n"); /* attach to regions */
    /*    fprintf(bob, "[(%d+5):%d,1:(%d-
5),1]:above_eng\n",nlf1_val,nlae1_val,nwf1_val);
        fprintf(bob, "[1:(%d-5),1:(%d-
5),1]:wind\n",nlf2_val,nwf1_val);*/
    fprintf(bob, "[5:,1:%d,1]:fire1\n",fire1oy_val);
    fprintf(bob, "[1:%d,1:%d,1]:fire2\n",fire2ox_val,fire2oy_val);
    fprintf(bob, "\n");
    fprintf(bob, "g\n");

  /* Open tascbob3d and send bob.txt into tascbob3d */
  /*   strcat(bob_start, dir_name);
   strcat(bob_start, "bob.txt | ");
   strcat(bob_start, tascript_dir);
   strcat(bob_start, "tascbob3d");
   printf("%s\n",bob_start);
   system(bob_start);*/
}
}
```